

```
#include <lib.h>
#define fchown _fchown
#include <string.h>
#include <unistd.h>

PUBLIC int fchown(fd, owner, grp)
int fd;
_mnx_Uid_t owner;
_mnx_Gid_t grp;
{
    message m;

    m.ml_i1 = fd;
    m.ml_i2 = owner;
    m.ml_i3 = grp;
    return(_syscall(FS, FCHOWN, &m));
}
```

Copyright 1992, 1993, 1994 Henry Spencer. All rights reserved.
This software is not subject to any license of the American Telephone
and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on
any computer system, and to alter it and redistribute it, subject
to the following restrictions:

1. The author is not responsible for the consequences of use of this
software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by
explicit claim or by omission. Since few users ever read sources,
credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must not be
misrepresented as being the original software. Since few users
ever read sources, credits must appear in the documentation.
4. This notice may not be removed or altered.

```
-----
/*-
 * Copyright (c) 1994
 *   The Regents of the University of California. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *   must display the following acknowledgement:
 *   This product includes software developed by the University of
 *   California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *   may be used to endorse or promote products derived from this software
 *   without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *   @(#)COPYRIGHT 8.1 (Berkeley) 3/16/94
 */
```

```
# Makefile for lib/regex.
```

```
CFLAGS="-O -D_MINIX -D_POSIX_SOURCE"
```

```
LIBRARIES=libc
```

```
libc_FILES=" \  
    regcomp.c \  
    regerror.c \  
    regexec.c \  
    regfree.c"
```

```
TYPE=both
```

@(#)WHATSNEW 8.3 (Berkeley) 3/18/94

New in alpha3.4: The complex bug alluded to below has been fixed (in a slightly kludgy temporary way that may hurt efficiency a bit; this is another "get it out the door for 4.4" release). The tests at the end of the tests file have accordingly been uncommented. The primary sign of the bug was that something like `a?b` matching `ab` matched `b` rather than `ab`. (The bug was essentially specific to this exact situation, else it would have shown up earlier.)

New in alpha3.3: The definition of word boundaries has been altered slightly, to more closely match the usual programming notion that `"_"` is an alphabetic. Stuff used for pre-ANSI systems is now in a subdir, and the makefile no longer alludes to it in mysterious ways. The makefile has generally been cleaned up some. Fixes have been made (again!) so that the regression test will run without `-DREDEBUG`, at the cost of weaker checking. A workaround for a bug in some folks' `<assert.h>` has been added. And some more things have been added to tests, including a couple right at the end which are commented out because the code currently flunks them (complex bug; fix coming). Plus the usual minor cleanup.

New in alpha3.2: Assorted bits of cleanup and portability improvement (the development base is now a BSDI system using GCC instead of an ancient Sun system, and the newer compiler exposed some glitches). Fix for a serious bug that affected REs using many `[]` (including `REG_ICASE` REs because of the way they are implemented), `*sometimes*`, depending on memory-allocation patterns. The header-file prototypes no longer name the parameters, avoiding possible name conflicts. The possibility that some clot has defined `CHAR_MIN` as (say) `'-128'` instead of `'(-128)'` is now handled gracefully. `"uchar"` is no longer used as an internal type name (too many people have the same idea). Still the same old lousy performance, alas.

New in alpha3.1: Basically nothing, this release is just a bookkeeping convenience. Stay tuned.

New in alpha3.0: Performance is no better, alas, but some fixes have been made and some functionality has been added. (This is basically the "get it out the door in time for 4.4" release.) One bug fix: `regfree()` didn't free the main internal structure (how embarrassing). It is now possible to put NULs in either the RE or the target string, using (resp.) a new `REG_PEND` flag and the old `REG_STARTEND` flag. The `REG_NOSPEC` flag to `regcomp()` makes all characters ordinary, so you can match a literal string easily (this will become more useful when performance improves!). There are now primitives to match beginnings and ends of words, although the syntax is disgusting and so is the implementation. The `REG_ATOI` debugging interface has changed a bit. And there has been considerable internal cleanup of various kinds.

New in alpha2.3: Split change list out of README, and moved flags notes into Makefile. Macro-ized the name of `regex(7)` in `regex(3)`, since it has to change for 4.4BSD. Cleanup work in `engine.c`, and some new regression tests to catch tricky cases thereof.

New in alpha2.2: Out-of-date manpages updated. `Regerror()` acquires two small extensions -- `REG_ITOA` and `REG_ATOI` -- which avoid debugging kludges in my own test program and might be useful to others for similar purposes. The regression test will now compile (and run) without `REDEBUG`. The BRE `\$` bug is fixed. Most uses of `"uchar"` are gone; it's all chars now. Char/uchar parameters are now written `int/unsigned`, to avoid possible portability problems with unpromoted parameters. Some unsigned casts have been introduced to minimize portability problems with shifting into sign bits.

New in alpha2.1: Lots of little stuff, cleanup and fixes. The one big thing is that `regex.h` is now generated, using `mkh`, rather than being supplied in the distribution; due to circularities in dependencies, you have to build `regex.h` explicitly by `"make h"`. The two known bugs have been fixed (and the regression test now checks for them), as has a problem with assertions not being suppressed in the absence of `REDEBUG`. No performance work yet.

New in alpha2: Backslash-anything is an ordinary character, not an

error (except, of course, for the handful of backslashed metacharacters in BREs), which should reduce script breakage. The regression test checks *where* null strings are supposed to match, and has generally been tightened up somewhat. Small bug fixes in parameter passing (not harmful, but technically errors) and some other areas. Debugging invoked by defining REDEBUG rather than not defining NDEBUG.

New in alpha+3: full prototyping for internal routines, using a little helper program, mkh, which extracts prototypes given in stylized comments. More minor cleanup. Buglet fix: it's CHAR_BIT, not CHAR_BITS. Simple pre-screening of input when a literal string is known to be part of the RE; this does wonders for performance.

New in alpha+2: minor bits of cleanup. Notably, the number "32" for the word width isn't hardwired into regexexec.c any more, the public header file prototypes the functions if __STDC__ is defined, and some small typos in the manpages have been fixed.

New in alpha+1: improvements to the manual pages, and an important extension, the REG_STARTEND option to regexexec().

```

/*-
 * Copyright (c) 1992, 1993, 1994 Henry Spencer.
 * Copyright (c) 1992, 1993, 1994
 *   The Regents of the University of California.  All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Henry Spencer.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *   must display the following acknowledgement:
 *   This product includes software developed by the University of
 *   California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *   may be used to endorse or promote products derived from this software
 *   without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)cclass.h      8.3 (Berkeley) 3/20/94
 */

/* character-class table */
static struct cclass {
    char *name;
    char *chars;
    char *multis;
} cclasses[] = {
    "alnum",      "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz\
0123456789",
    "alpha",      "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz",
    "blank",      "\t",
    "cntrl",      "\007\b\t\n\v\f\r1\2\3\4\5\6\16\17\20\21\22\23\24\
\25\26\27\30\31\32\33\34\35\36\37\177",
    "digit",      "0123456789",
    "graph",      "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz\
0123456789!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~",
    "lower",      "abcdefghijklmnopqrstuvwxyz",
    "print",      "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz\
0123456789!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~",
    "punct",      "!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~",
    "space",      "\t\n\v\f\r ",
    "upper",      "ABCDEFGHIJKLMNOPQRSTUVWXYZ",
    "xdigit",     "0123456789ABCDEFabcdef",
    NULL,         0,
};

```

```
/*-
 * Copyright (c) 1992, 1993, 1994 Henry Spencer.
 * Copyright (c) 1992, 1993, 1994
 *   The Regents of the University of California. All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Henry Spencer.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *   must display the following acknowledgement:
 *       This product includes software developed by the University of
 *       California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *   may be used to endorse or promote products derived from this software
 *   without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)cname.h      8.3 (Berkeley) 3/20/94
 */
```

```
/* character-name table */
```

```
static struct cname {
    char *name;
    char code;
} cnames[] = {
    "NUL", '\0',
    "SOH", '\001',
    "STX", '\002',
    "ETX", '\003',
    "EOT", '\004',
    "ENQ", '\005',
    "ACK", '\006',
    "BEL", '\007',
    "alert", '\007',
    "BS", '\010',
    "backspace", 'b',
    "HT", '\011',
    "tab", 't',
    "LF", '\012',
    "newline", 'n',
    "VT", '\013',
    "vertical-tab", 'v',
    "FF", '\014',
    "form-feed", 'f',
    "CR", '\015',
    "carriage-return", 'r',
    "SO", '\016',
    "SI", '\017',
    "DLE", '\020',
    "DC1", '\021',
    "DC2", '\022',
    "DC3", '\023',
    "DC4", '\024',
    "NAK", '\025',
    "SYN", '\026',
```

```

"ETB" , '\027' ,
"CAN" , '\030' ,
"EM" , '\031' ,
"SUB" , '\032' ,
"ESC" , '\033' ,
"IS4" , '\034' ,
"FS" , '\034' ,
"IS3" , '\035' ,
"GS" , '\035' ,
"IS2" , '\036' ,
"RS" , '\036' ,
"IS1" , '\037' ,
"US" , '\037' ,
"space" , ' ' ,
"exclamation-mark" , '!' ,
"quotation-mark" , '"' ,
"number-sign" , '#' ,
"dollar-sign" , '$' ,
"percent-sign" , '%' ,
"ampersand" , '&' ,
"apostrophe" , '\'' ,
"left-parenthesis" , '(' ,
"right-parenthesis" , ')' ,
"asterisk" , '*' ,
"plus-sign" , '+' ,
"comma" , ',' ,
"hyphen" , '-' ,
"hyphen-minus" , '-' ,
"period" , '.' ,
"full-stop" , '.' ,
"slash" , '/' ,
"solidus" , '/' ,
"zero" , '0' ,
"one" , '1' ,
"two" , '2' ,
"three" , '3' ,
"four" , '4' ,
"five" , '5' ,
"six" , '6' ,
"seven" , '7' ,
"eight" , '8' ,
"nine" , '9' ,
"colon" , ':' ,
"semicolon" , ';' ,
"less-than-sign" , '<' ,
"equals-sign" , '=' ,
"greater-than-sign" , '>' ,
"question-mark" , '?' ,
"commercial-at" , '@' ,
"left-square-bracket" , '[' ,
"backslash" , '\\' ,
"reverse-solidus" , '\\' ,
"right-square-bracket" , ']' ,
"circumflex" , '^' ,
"circumflex-accent" , '^' ,
"underscore" , '_' ,
"low-line" , '_' ,
"grave-accent" , '`' ,
"left-brace" , '{' ,
"left-curly-bracket" , '{' ,
"vertical-line" , '|' ,
"right-brace" , '}' ,
"right-curly-bracket" , '}' ,
"tilde" , '~' ,
"DEL" , '\177' ,
NULL , 0 ,

```

};


```
/*-
 * Copyright (c) 1992, 1993, 1994 Henry Spencer.
 * Copyright (c) 1992, 1993, 1994
 *   The Regents of the University of California. All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Henry Spencer.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *   must display the following acknowledgement:
 *       This product includes software developed by the University of
 *       California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *   may be used to endorse or promote products derived from this software
 *   without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)engine.c      8.5 (Berkeley) 3/20/94
 */

/*
 * The matching engine and friends. This file is #included by regex.c
 * after suitable #defines of a variety of macros used herein, so that
 * different state representations can be used without duplicating masses
 * of code.
 */

#ifdef SNames
#define matcher smatcher
#define fast sfast
#define slow sslow
#define dissect sdissect
#define backref sbackref
#define step sstep
#define print sprint
#define at sat
#define match smat
#endif
#ifdef LNames
#define matcher lmatcher
#define fast lfast
#define slow lslow
#define dissect ldissect
#define backref lbackref
#define step lstep
#define print lprint
#define at lat
#define match lmat
#endif

/* another structure passed up and down to avoid zillions of parameters */
struct match {
    struct re_guts *g;
    int eflags;
    regmatch_t *pmatch; /* [nsub+1] (0 element unused) */
}
```

```

    char *offp;                /* offsets work from here */
    char *beginp;              /* start of string -- virtual NUL precedes */
    char *endp;                /* end of string -- virtual NUL here */
    char *coldp;               /* can be no match starting before here */
    char **lastpos;            /* [nplus+1] */
    STATEVARS;
    states st;                 /* current states */
    states fresh;              /* states for a fresh start */
    states tmp;                /* temporary */
    states empty;              /* empty set of states */
};

/* ===== begin header generated by ./mkh ===== */
#ifdef __cplusplus
extern "C" {
#endif

/* == engine.c == */
static int matcher(struct re_guts *g, char *string, size_t nmatch, regmatch_t pmatch[], int eflags);
static char *dissect(struct match *m, char *start, char *stop, sopno startst, sopno stopst);
static char *backref(struct match *m, char *start, char *stop, sopno startst, sopno stopst, sopno lev);
static char *fast(struct match *m, char *start, char *stop, sopno startst, sopno stopst);
static char *slow(struct match *m, char *start, char *stop, sopno startst, sopno stopst);
static states step(struct re_guts *g, sopno start, sopno stop, states bef, int ch, states aft);
#define BOL (OUT+1)
#define EOL (BOL+1)
#define BOLEOL (BOL+2)
#define NOTHING (BOL+3)
#define BOW (BOL+4)
#define EOW (BOL+5)
#define CODEMAX (BOL+5) /* highest code used */
#define NONCHAR(c) ((c) > CHAR_MAX)
#define NNONCHAR (CODEMAX-CHAR_MAX)
#ifdef REDEBUG
static void print(struct match *m, char *caption, states st, int ch, FILE *d);
#endif
#ifdef REDEBUG
static void at(struct match *m, char *title, char *start, char *stop, sopno startst, sopno stopst);
#endif
#ifdef REDEBUG
static char *pchar(int ch);
#endif

#ifdef __cplusplus
}
#endif
/* ===== end header generated by ./mkh ===== */

#ifdef REDEBUG
#define SP(t, s, c) print(m, t, s, c, stdout)
#define AT(t, pl, p2, sl, s2) at(m, t, pl, p2, sl, s2)
#define NOTE(str) { if (m->eflags&REG_TRACE) printf("=%s\n", (str)); }
#else
#define SP(t, s, c) /* nothing */
#define AT(t, pl, p2, sl, s2) /* nothing */
#define NOTE(s) /* nothing */
#endif

/*
- matcher - the actual matching engine
== static int matcher(register struct re_guts *g, char *string, \
== size_t nmatch, regmatch_t pmatch[], int eflags);
*/
static int /* 0 success, REG_NOMATCH failure */
matcher(g, string, nmatch, pmatch, eflags)
register struct re_guts *g;
char *string;
size_t nmatch;
regmatch_t pmatch[];

```

```

int eflags;
{
    register char *endp;
    register int i;
    struct match mv;
    register struct match *m = &mv;
    register char *dp;
    const register sopno gf = g->firststate+1;      /* +1 for OEND */
    const register sopno gl = g->laststate;
    char *start;
    char *stop;

    /* simplify the situation where possible */
    if (g->cflags&REG_NOSUB)
        nmatch = 0;
    if (eflags&REG_STARTEND) {
        start = string + pmatch[0].rm_so;
        stop = string + pmatch[0].rm_eo;
    } else {
        start = string;
        stop = start + strlen(start);
    }
    if (stop < start)
        return(REG_INVARG);

    /* prescreening; this does wonders for this rather slow code */
    if (g->must != NULL) {
        for (dp = start; dp < stop; dp++)
            if (*dp == g->must[0] && stop - dp >= g->mlen &&
                memcmp(dp, g->must, (size_t)g->mlen) == 0)
                break;
        if (dp == stop) /* we didn't find g->must */
            return(REG_NOMATCH);
    }

    /* match struct setup */
    m->g = g;
    m->eflags = eflags;
    m->pmatch = NULL;
    m->lastpos = NULL;
    m->offp = string;
    m->beginp = start;
    m->endp = stop;
    STATESETUP(m, 4);
    SETUP(m->st);
    SETUP(m->fresh);
    SETUP(m->tmp);
    SETUP(m->empty);
    CLEAR(m->empty);

    /* this loop does only one repetition except for backrefs */
    for (;;) {
        endp = fast(m, start, stop, gf, gl);
        if (endp == NULL) { /* a miss */
            STATETEARDOWN(m);
            return(REG_NOMATCH);
        }
        if (nmatch == 0 && !g->backrefs)
            break; /* no further info needed */

        /* where? */
        assert(m->coldp != NULL);
        for (;;) {
            NOTE("finding start");
            endp = slow(m, m->coldp, stop, gf, gl);
            if (endp != NULL)
                break;
            assert(m->coldp < m->endp);
            m->coldp++;
        }
        if (nmatch == 1 && !g->backrefs)
            break; /* no further info needed */

        /* oh my, he wants the subexpressions... */
    }
}

```

```

    if (m->pmatch == NULL)
        m->pmatch = (regmatch_t *)malloc((m->g->nsub + 1) *
                                           sizeof(regmatch_t));

    if (m->pmatch == NULL) {
        STATETEARDOWN(m);
        return(REG_ESPACE);
    }
    for (i = 1; i <= m->g->nsub; i++)
        m->pmatch[i].rm_so = m->pmatch[i].rm_eo = -1;
    if (!g->backrefs && !(m->eflags&REG_BACKR)) {
        NOTE("dissecting");
        dp = dissect(m, m->coldp, endp, gf, gl);
    } else {
        if (g->nplus > 0 && m->lastpos == NULL)
            m->lastpos = (char **)malloc((g->nplus+1) *
                                           sizeof(char *));

        if (g->nplus > 0 && m->lastpos == NULL) {
            free(m->pmatch);
            STATETEARDOWN(m);
            return(REG_ESPACE);
        }
        NOTE("backref dissect");
        dp = backref(m, m->coldp, endp, gf, gl, (sopno)0);
    }
    if (dp != NULL)
        break;

    /* uh-oh... we couldn't find a subexpression-level match */
    assert(g->backrefs); /* must be back references doing it */
    assert(g->nplus == 0 || m->lastpos != NULL);
    for (;;) {
        if (dp != NULL || endp <= m->coldp)
            break; /* defeat */
        NOTE("backoff");
        endp = slow(m, m->coldp, endp-1, gf, gl);
        if (endp == NULL)
            break; /* defeat */
        /* try it on a shorter possibility */

        for (i = 1; i <= m->g->nsub; i++) {
            assert(m->pmatch[i].rm_so == -1);
            assert(m->pmatch[i].rm_eo == -1);
        }

        NOTE("backoff dissect");
        dp = backref(m, m->coldp, endp, gf, gl, (sopno)0);
    }
    assert(dp == NULL || dp == endp);
    if (dp != NULL) /* found a shorter one */
        break;

    /* despite initial appearances, there is no match here */
    NOTE("false alarm");
    start = m->coldp + 1; /* recycle starting later */
    assert(start <= stop);
}

/* fill in the details if requested */
if (nmatch > 0) {
    pmatch[0].rm_so = m->coldp - m->offp;
    pmatch[0].rm_eo = endp - m->offp;
}
if (nmatch > 1) {
    assert(m->pmatch != NULL);
    for (i = 1; i < nmatch; i++)
        if (i <= m->g->nsub)
            pmatch[i] = m->pmatch[i];
        else {
            pmatch[i].rm_so = -1;
            pmatch[i].rm_eo = -1;
        }
}

if (m->pmatch != NULL)

```

```

        free((char *)m->pmatch);
    if (m->lastpos != NULL)
        free((char *)m->lastpos);
    STATETEARDOWN(m);
    return(0);
}

/*
- dissect - figure out what matched what, no back references
== static char *dissect(register struct match *m, char *start, \
== char *stop, sopno startst, sopno stopst);
*/
static char *
dissect(m, start, stop, startst, stopst)
register struct match *m;
char *start;
char *stop;
sopno startst;
sopno stopst;
{
    register int i;
    register sopno ss;      /* start sop of current subRE */
    register sopno es;      /* end sop of current subRE */
    register char *sp;      /* start of string matched by it */
    register char *stp;     /* string matched by it cannot pass here */
    register char *rest;    /* start of rest of string */
    register char *tail;    /* string unmatched by rest of RE */
    register sopno ssub;    /* start sop of subsubRE */
    register sopno esub;    /* end sop of subsubRE */
    register char *ssp;     /* start of string matched by subsubRE */
    register char *sep;     /* end of string matched by subsubRE */
    register char *oldssp;  /* previous ssp */
    register char *dp;

    AT("diss", start, stop, startst, stopst);
    sp = start;
    for (ss = startst; ss < stopst; ss = es) {
        /* identify end of subRE */
        es = ss;
        switch (OP(m->g->strip[es])) {
            case OPLUS_:
            case OQUEST_:
                es += OPND(m->g->strip[es]);
                break;
            case OCH_:
                while (OP(m->g->strip[es]) != O_CH)
                    es += OPND(m->g->strip[es]);
                break;
        }
        es++;

        /* figure out what it matched */
        switch (OP(m->g->strip[ss])) {
            case OEND:
                assert(nope);
                break;
            case OCHAR:
                sp++;
                break;
            case OBOL:
            case OEOL:
            case OBOW:
            case OEOW:
                break;
            case OANY:
            case OANYOF:
                sp++;
                break;
            case OBACK_:
            case O_BACK:
                assert(nope);
                break;
            /* cases where length of match is hard to find */
            case OQUEST_:

```

```

    stp = stop;
    for (;;) {
        /* how long could this one be? */
        rest = slow(m, sp, stp, ss, es);
        assert(rest != NULL); /* it did match */
        /* could the rest match the rest? */
        tail = slow(m, rest, stop, es, stopst);
        if (tail == stop)
            break; /* yes! */
        /* no -- try a shorter match for this one */
        stp = rest - 1;
        assert(stp >= sp); /* it did work */
    }
    ssub = ss + 1;
    esub = es - 1;
    /* did innards match? */
    if (slow(m, sp, rest, ssub, esub) != NULL) {
        dp = dissect(m, sp, rest, ssub, esub);
        assert(dp == rest);
    } else /* no */
        assert(sp == rest);
    sp = rest;
    break;
case OPLUS_:
    stp = stop;
    for (;;) {
        /* how long could this one be? */
        rest = slow(m, sp, stp, ss, es);
        assert(rest != NULL); /* it did match */
        /* could the rest match the rest? */
        tail = slow(m, rest, stop, es, stopst);
        if (tail == stop)
            break; /* yes! */
        /* no -- try a shorter match for this one */
        stp = rest - 1;
        assert(stp >= sp); /* it did work */
    }
    ssub = ss + 1;
    esub = es - 1;
    ssp = sp;
    oldssp = ssp;
    for (;;) { /* find last match of innards */
        sep = slow(m, ssp, rest, ssub, esub);
        if (sep == NULL || sep == ssp)
            break; /* failed or matched null */
        oldssp = ssp; /* on to next try */
        ssp = sep;
    }
    if (sep == NULL) {
        /* last successful match */
        sep = ssp;
        ssp = oldssp;
    }
    assert(sep == rest); /* must exhaust substring */
    assert(slow(m, ssp, sep, ssub, esub) == rest);
    dp = dissect(m, ssp, sep, ssub, esub);
    assert(dp == sep);
    sp = rest;
    break;
case OCH_:
    stp = stop;
    for (;;) {
        /* how long could this one be? */
        rest = slow(m, sp, stp, ss, es);
        assert(rest != NULL); /* it did match */
        /* could the rest match the rest? */
        tail = slow(m, rest, stop, es, stopst);
        if (tail == stop)
            break; /* yes! */
        /* no -- try a shorter match for this one */
        stp = rest - 1;
        assert(stp >= sp); /* it did work */
    }
    ssub = ss + 1;

```

```

        esub = ss + OPND(m->g->strip[ss]) - 1;
        assert(OP(m->g->strip[esub]) == OOR1);
        for (;;) { /* find first matching branch */
            if (slow(m, sp, rest, ssub, esub) == rest)
                break; /* it matched all of it */
            /* that one missed, try next one */
            assert(OP(m->g->strip[esub]) == OOR1);
            esub++;
            assert(OP(m->g->strip[esub]) == OOR2);
            ssub = esub + 1;
            esub += OPND(m->g->strip[esub]);
            if (OP(m->g->strip[esub]) == OOR2)
                esub--;
            else
                assert(OP(m->g->strip[esub]) == O_CH);
        }
        dp = dissect(m, sp, rest, ssub, esub);
        assert(dp == rest);
        sp = rest;
        break;
    case O_PLUS:
    case O_QUESTION:
    case OOR1:
    case OOR2:
    case O_CH:
        assert(nope);
        break;
    case OLPAREN:
        i = OPND(m->g->strip[ss]);
        assert(0 < i && i <= m->g->nsub);
        m->pmatch[i].rm_so = sp - m->offp;
        break;
    case ORPAREN:
        i = OPND(m->g->strip[ss]);
        assert(0 < i && i <= m->g->nsub);
        m->pmatch[i].rm_eo = sp - m->offp;
        break;
    default: /* uh oh */
        assert(nope);
        break;
    }
}

assert(sp == stop);
return(sp);
}

/*
- backref - figure out what matched what, figuring in back references
== static char *backref(register struct match *m, char *start, \
== char *stop, sopno startst, sopno stopst, sopno lev);
*/
static char * /* == stop (success) or NULL (failure) */
backref(m, start, stop, startst, stopst, lev)
register struct match *m;
char *start;
char *stop;
sopno startst;
sopno stopst;
sopno lev; /* PLUS nesting level */
{
    register int i;
    register sopno ss; /* start sop of current subRE */
    register char *sp; /* start of string matched by it */
    register sopno ssub; /* start sop of subsubRE */
    register sopno esub; /* end sop of subsubRE */
    register char *ssp; /* start of string matched by subsubRE */
    register char *dp;
    register size_t len;
    register int hard;
    register sop s;
    register regoff_t offsave;
    register cset *cs;

```

```

AT("back", start, stop, startst, stopst);
sp = start;

/* get as far as we can with easy stuff */
hard = 0;
for (ss = startst; !hard && ss < stopst; ss++)
    switch (OP(s = m->g->strip[ss])) {
    case OCHAR:
        if (sp == stop || *sp++ != (char)OPND(s))
            return(NULL);
        break;
    case OANY:
        if (sp == stop)
            return(NULL);
        sp++;
        break;
    case OANYOF:
        cs = &m->g->sets[OPND(s)];
        if (sp == stop || !CHIN(cs, *sp++))
            return(NULL);
        break;
    case OBOL:
        if ( (sp == m->beginp && !(m->eflags&REG_NOTBOL)) ||
              (sp < m->endp && *(sp-1) == '\n' &&
               (m->g->cflags&REG_NEWLINE)) )
            { /* yes */ }
        else
            return(NULL);
        break;
    case OEOL:
        if ( (sp == m->endp && !(m->eflags&REG_NOTEOL)) ||
              (sp < m->endp && *sp == '\n' &&
               (m->g->cflags&REG_NEWLINE)) )
            { /* yes */ }
        else
            return(NULL);
        break;
    case OBOW:
        if ( ( (sp == m->beginp && !(m->eflags&REG_NOTBOL)) ||
                (sp < m->endp && *(sp-1) == '\n' &&
                 (m->g->cflags&REG_NEWLINE)) ||
                (sp > m->beginp &&
                 !ISWORD(*(sp-1))) ) &&
              (sp < m->endp && ISWORD(*sp)) )
            { /* yes */ }
        else
            return(NULL);
        break;
    case OEOW:
        if ( ( (sp == m->endp && !(m->eflags&REG_NOTEOL)) ||
                (sp < m->endp && *sp == '\n' &&
                 (m->g->cflags&REG_NEWLINE)) ||
                (sp < m->endp && !ISWORD(*sp)) ) &&
              (sp > m->beginp && ISWORD(*(sp-1))) )
            { /* yes */ }
        else
            return(NULL);
        break;
    case O_QUEST:
        break;
    case OOR1:
        /* matches null but needs to skip */
        ss++;
        s = m->g->strip[ss];
        do {
            assert(OP(s) == OOR2);
            ss += OPND(s);
        } while (OP(s = m->g->strip[ss]) != O_CH);
        /* note that the ss++ gets us past the O_CH */
        break;
    default:
        /* have to make a choice */
        hard = 1;
        break;
    }
if (!hard) {
    /* that was it! */

```



```

        if (sp != stop)
            return(NULL);
        return(sp);
    }
    ss--; /* adjust for the for's final increment */

    /* the hard stuff */
    AT("hard", sp, stop, ss, stopst);
    s = m->g->strip[ss];
    switch (OP(s)) {
    case OBACK_: /* the vilest depths */
        i = OPND(s);
        assert(0 < i && i <= m->g->nsub);
        if (m->pmatch[i].rm_eo == -1)
            return(NULL);
        assert(m->pmatch[i].rm_so != -1);
        len = m->pmatch[i].rm_eo - m->pmatch[i].rm_so;
        assert(stop - m->beginp >= len);
        if (sp > stop - len)
            return(NULL); /* not enough left to match */
        ssp = m->offp + m->pmatch[i].rm_so;
        if (memcmp(sp, ssp, len) != 0)
            return(NULL);
        while (m->g->strip[ss] != SOP(O_BACK, i))
            ss++;
        return(backref(m, sp+len, stop, ss+1, stopst, lev));
        break;
    case OQUEST_: /* to null or not */
        dp = backref(m, sp, stop, ss+1, stopst, lev);
        if (dp != NULL)
            return(dp); /* not */
        return(backref(m, sp, stop, ss+OPND(s)+1, stopst, lev));
        break;
    case OPLUS_:
        assert(m->lastpos != NULL);
        assert(lev+1 <= m->g->nplus);
        m->lastpos[lev+1] = sp;
        return(backref(m, sp, stop, ss+1, stopst, lev+1));
        break;
    case O_PLUS:
        if (sp == m->lastpos[lev]) /* last pass matched null */
            return(backref(m, sp, stop, ss+1, stopst, lev-1));
        /* try another pass */
        m->lastpos[lev] = sp;
        dp = backref(m, sp, stop, ss-OPND(s)+1, stopst, lev);
        if (dp == NULL)
            return(backref(m, sp, stop, ss+1, stopst, lev-1));
        else
            return(dp);
        break;
    case OCH_: /* find the right one, if any */
        ssub = ss + 1;
        esub = ss + OPND(s) - 1;
        assert(OP(m->g->strip[esub]) == OOR1);
        for (;;) { /* find first matching branch */
            dp = backref(m, sp, stop, ssub, esub, lev);
            if (dp != NULL)
                return(dp);
            /* that one missed, try next one */
            if (OP(m->g->strip[esub]) == O_CH)
                return(NULL); /* there is none */
            esub++;
            assert(OP(m->g->strip[esub]) == OOR2);
            ssub = esub + 1;
            esub += OPND(m->g->strip[esub]);
            if (OP(m->g->strip[esub]) == OOR2)
                esub--;
            else
                assert(OP(m->g->strip[esub]) == O_CH);
        }
        break;
    case OLPAREN: /* must undo assignment if rest fails */
        i = OPND(s);
        assert(0 < i && i <= m->g->nsub);

```

```

        offsave = m->pmatch[i].rm_so;
        m->pmatch[i].rm_so = sp - m->offp;
        dp = backref(m, sp, stop, ss+1, stopst, lev);
        if (dp != NULL)
            return(dp);
        m->pmatch[i].rm_so = offsave;
        return(NULL);
        break;
    case ORPAREN:
        /* must undo assignment if rest fails */
        i = OPND(s);
        assert(0 < i && i <= m->g->nsb);
        offsave = m->pmatch[i].rm_eo;
        m->pmatch[i].rm_eo = sp - m->offp;
        dp = backref(m, sp, stop, ss+1, stopst, lev);
        if (dp != NULL)
            return(dp);
        m->pmatch[i].rm_eo = offsave;
        return(NULL);
        break;
    default:
        /* uh oh */
        assert(nope);
        break;
}

/* "can't happen" */
assert(nope);
/* NOTREACHED */
}

/*
- fast - step through the string at top speed
== static char *fast(register struct match *m, char *start, \
== char *stop, sopno startst, sopno stopst);
*/
static char *
fast(m, start, stop, startst, stopst)
register struct match *m;
char *start;
char *stop;
sopno startst;
sopno stopst;
{
    register states st = m->st;
    register states fresh = m->fresh;
    register states tmp = m->tmp;
    register char *p = start;
    register int c = (start == m->beginp) ? OUT : *(start-1);
    register int lastc;
    register int flagch;
    register int i;
    register char *coldp;
    /* last p after which no match was underway */

    CLEAR(st);
    SETl(st, startst);
    st = step(m->g, startst, stopst, st, NOTHING, st);
    ASSIGN(fresh, st);
    SP("start", st, *p);
    coldp = NULL;
    for (;;) {
        /* next character */
        lastc = c;
        c = (p == m->endp) ? OUT : *p;
        if (EQ(st, fresh))
            coldp = p;

        /* is there an EOL and/or BOL between lastc and c? */
        flagch = '\0';
        i = 0;
        if ( (lastc == '\n' && m->g->cflags&REG_NEWLINE) ||
            (lastc == OUT && !(m->eflags&REG_NOTBOL)) ) {
            flagch = BOL;
            i = m->g->nbol;
        }
        if ( (c == '\n' && m->g->cflags&REG_NEWLINE) ||

```

```

        (c == OUT && !(m->eflags&REG_NOTEOL)) ) {
            flagch = (flagch == BOL) ? BOLEOL : EOL;
            i += m->g->neol;
        }
    if (i != 0) {
        for (; i > 0; i--)
            st = step(m->g, startst, stopst, st, flagch, st);
        SP("boleol", st, c);
    }

    /* how about a word boundary? */
    if ( (flagch == BOL || (lastc != OUT && !ISWORD(lastc))) &&
        (c != OUT && ISWORD(c)) ) {
        flagch = BOW;
    }
    if ( (lastc != OUT && ISWORD(lastc)) &&
        (flagch == EOL || (c != OUT && !ISWORD(c))) ) {
        flagch = EOW;
    }
    if (flagch == BOW || flagch == EOW) {
        st = step(m->g, startst, stopst, st, flagch, st);
        SP("boweow", st, c);
    }

    /* are we done? */
    if (ISSET(st, stopst) || p == stop)
        break; /* NOTE BREAK OUT */

    /* no, we must deal with this character */
    ASSIGN(tmp, st);
    ASSIGN(st, fresh);
    assert(c != OUT);
    st = step(m->g, startst, stopst, tmp, c, st);
    SP("aft", st, c);
    assert(EQ(step(m->g, startst, stopst, st, NOTHING, st), st));
    p++;
}

assert(coldp != NULL);
m->coldp = coldp;
if (ISSET(st, stopst))
    return(p+1);
else
    return(NULL);
}

/*
- slow - step through the string more deliberately
== static char *slow(register struct match *m, char *start, \
== char *stop, sopno startst, sopno stopst);
*/
static char * /* where it ended */
slow(m, start, stop, startst, stopst)
register struct match *m;
char *start;
char *stop;
sopno startst;
sopno stopst;
{
    register states st = m->st;
    register states empty = m->empty;
    register states tmp = m->tmp;
    register char *p = start;
    register int c = (start == m->beginp) ? OUT : *(start-1);
    register int lastc; /* previous c */
    register int flagch;
    register int i;
    register char *matchp; /* last p at which a match ended */

    AT("slow", start, stop, startst, stopst);
    CLEAR(st);
    SETl(st, startst);
    SP("sstart", st, *p);
    st = step(m->g, startst, stopst, st, NOTHING, st);

```

```

matchp = NULL;
for (;;) {
    /* next character */
    lastc = c;
    c = (p == m->endp) ? OUT : *p;

    /* is there an EOL and/or BOL between lastc and c? */
    flagch = '\0';
    i = 0;
    if ( (lastc == '\n' && m->g->cflags&REG_NEWLINE) ||
          (lastc == OUT && !(m->eflags&REG_NOTBOL)) ) {
        flagch = BOL;
        i = m->g->nbol;
    }
    if ( (c == '\n' && m->g->cflags&REG_NEWLINE) ||
          (c == OUT && !(m->eflags&REG_NOTEOL)) ) {
        flagch = (flagch == BOL) ? BOLEOL : EOL;
        i += m->g->neol;
    }
    if (i != 0) {
        for (; i > 0; i--)
            st = step(m->g, startst, stopst, st, flagch, st);
        SP("sboleol", st, c);
    }

    /* how about a word boundary? */
    if ( (flagch == BOL || (lastc != OUT && !ISWORD(lastc))) &&
          (c != OUT && ISWORD(c)) ) {
        flagch = BOW;
    }
    if ( (lastc != OUT && ISWORD(lastc)) &&
          (flagch == EOL || (c != OUT && !ISWORD(c))) ) {
        flagch = EOW;
    }
    if (flagch == BOW || flagch == EOW) {
        st = step(m->g, startst, stopst, st, flagch, st);
        SP("sboweow", st, c);
    }

    /* are we done? */
    if (ISSET(st, stopst))
        matchp = p;
    if (EQ(st, empty) || p == stop)
        break; /* NOTE BREAK OUT */

    /* no, we must deal with this character */
    ASSIGN(tmp, st);
    ASSIGN(st, empty);
    assert(c != OUT);
    st = step(m->g, startst, stopst, tmp, c, st);
    SP("saft", st, c);
    assert(EQ(step(m->g, startst, stopst, st, NOTHING, st), st));
    p++;
}

return(matchp);
}

/*
- step - map set of states reachable before char to set reachable after
== static states step(register struct re_guts *g, sopno start, sopno stop, \
== register states bef, int ch, register states aft);
== #define BOL (OUT+1)
== #define EOL (BOL+1)
== #define BOLEOL (BOL+2)
== #define NOTHING (BOL+3)
== #define BOW (BOL+4)
== #define EOW (BOL+5)
== #define CODEMAX (BOL+5) // highest code used
== #define NONCHAR(c) ((c) > CHAR_MAX)
== #define NNONCHAR (CODEMAX-CHAR_MAX)
*/
static states

```

```

step(g, start, stop, bef, ch, aft)
register struct re_guts *g;
sopno start;                                /* start state within strip */
sopno stop;                                 /* state after stop state within strip */
register states bef;                         /* states reachable before */
int ch;                                     /* character or NONCHAR code */
register states aft;                         /* states already known reachable after */
{
    register cset *cs;
    register sopno s;
    register sopno pc;
    register onestate here;                  /* note, macros know this name */
    register sopno look;
    register int i;

    for (pc = start, INIT(here, pc); pc != stop; pc++, INC(here)) {
        s = g->strip[pc];
        switch (OP(s)) {
            case OEND:
                assert(pc == stop-1);
                break;
            case OCHAR:
                /* only characters can match */
                assert(!NONCHAR(ch) || ch != (char)OPND(s));
                if (ch == (char)OPND(s))
                    FWD(aft, bef, 1);
                break;
            case OBOL:
                if (ch == BOL || ch == BOLEOL)
                    FWD(aft, bef, 1);
                break;
            case OEOL:
                if (ch == EOL || ch == BOLEOL)
                    FWD(aft, bef, 1);
                break;
            case OBOW:
                if (ch == BOW)
                    FWD(aft, bef, 1);
                break;
            case OEOW:
                if (ch == EOW)
                    FWD(aft, bef, 1);
                break;
            case OANY:
                if (!NONCHAR(ch))
                    FWD(aft, bef, 1);
                break;
            case OANYOF:
                cs = &g->sets[OPND(s)];
                if (!NONCHAR(ch) && CHIN(cs, ch))
                    FWD(aft, bef, 1);
                break;
            case OBACK_:
                /* ignored here */
            case O_BACK:
                FWD(aft, aft, 1);
                break;
            case OPLUS_:
                /* forward, this is just an empty */
                FWD(aft, aft, 1);
                break;
            case O_PLUS:
                /* both forward and back */
                FWD(aft, aft, 1);
                i = ISSETBACK(aft, OPND(s));
                BACK(aft, aft, OPND(s));
                if (!i && ISSETBACK(aft, OPND(s))) {
                    /* oho, must reconsider loop body */
                    pc -= OPND(s) + 1;
                    INIT(here, pc);
                }
                break;
            case OQUEST_:
                /* two branches, both forward */
                FWD(aft, aft, 1);
                FWD(aft, aft, OPND(s));
                break;
            case O_QUEST:
                /* just an empty */

```

```

        FWD(aft, aft, 1);
        break;
    case OLPAREN:          /* not significant here */
    case ORPAREN:
        FWD(aft, aft, 1);
        break;
    case OCH_:             /* mark the first two branches */
        FWD(aft, aft, 1);
        assert(OP(g->strip[pc+OPND(s)]) == OOR2);
        FWD(aft, aft, OPND(s));
        break;
    case OOR1:             /* done a branch, find the O_CH */
        if (ISSTATEIN(aft, here)) {
            for (look = 1;
                 OP(s = g->strip[pc+look]) != O_CH;
                 look += OPND(s))
                assert(OP(s) == OOR2);
            FWD(aft, aft, look);
        }
        break;
    case OOR2:             /* propagate OCH_'s marking */
        FWD(aft, aft, 1);
        if (OP(g->strip[pc+OPND(s)]) != O_CH) {
            assert(OP(g->strip[pc+OPND(s)]) == OOR2);
            FWD(aft, aft, OPND(s));
        }
        break;
    case O_CH:             /* just empty */
        FWD(aft, aft, 1);
        break;
    default:               /* oooooops... */
        assert(nope);
        break;
    }
}

return(aft);
}

#ifdef REDEBUG
/*
- print - print a set of states
== #ifdef REDEBUG
== static void print(struct match *m, char *caption, states st, \
== int ch, FILE *d);
== #endif
*/
static void
print(m, caption, st, ch, d)
struct match *m;
char *caption;
states st;
int ch;
FILE *d;
{
    register struct re_guts *g = m->g;
    register int i;
    register int first = 1;

    if (!(m->eflags&REG_TRACE))
        return;

    fprintf(d, "%s", caption);
    if (ch != '\0')
        fprintf(d, " %s", pchar(ch));
    for (i = 0; i < g->nstates; i++)
        if (ISSET(st, i)) {
            fprintf(d, "%s%d", (first) ? "\t" : ", ", i);
            first = 0;
        }
    fprintf(d, "\n");
}
/*

```

```
- at - print current situation
== #ifdef REDEBUG
== static void at(struct match *m, char *title, char *start, char *stop, \
==                                     sopno startst, sopno stopst);
== #endif
*/
static void
at(m, title, start, stop, startst, stopst)
struct match *m;
char *title;
char *start;
char *stop;
sopno startst;
sopno stopst;
{
    if (!(m->eflags&REG_TRACE))
        return;

    printf("%s%s-", title, pchar(*start));
    printf("%s ", pchar(*stop));
    printf("%ld-%ld\n", (long)startst, (long)stopst);
}

#ifndef PCHARDONE
#define PCHARDONE          /* never again */
/*
- pchar - make a character printable
== #ifdef REDEBUG
== static char *pchar(int ch);
== #endif
*
* Is this identical to regchar() over in debug.c? Well, yes. But a
* duplicate here avoids having a debugging-capable regexec.o tied to
* a matching debug.o, and this is convenient. It all disappears in
* the non-debug compilation anyway, so it doesn't matter much.
*/
static char *                /* -> representation */
pchar(ch)
int ch;
{
    static char pbuf[10];

    if (isprint(ch) || ch == ' ')
        sprintf(pbuf, "%c", ch);
    else
        sprintf(pbuf, "\\%o", ch);
    return(pbuf);
}
#endif
#endif

#undef matcher
#undef fast
#undef slow
#undef dissect
#undef backref
#undef step
#undef print
#undef at
#undef match
```

NAME

re_format – POSIX 1003.2 regular expressions

DESCRIPTION

Regular expressions (“RE”s), as defined in POSIX 1003.2, come in two forms: modern REs (roughly those of **egrep**; 1003.2 calls these “extended” REs) and obsolete REs (roughly those of **ed**; 1003.2 “basic” REs). Obsolete REs mostly exist for backward compatibility in some old programs; they will be discussed at the end. 1003.2 leaves some aspects of RE syntax and semantics open; ‘†’ marks decisions on these aspects that may not be fully portable to other 1003.2 implementations.

A (modern) RE is one† or more non-empty† *branches*, separated by ‘|’. It matches anything that matches one of the branches.

A branch is one† or more *pieces*, concatenated. It matches a match for the first, followed by a match for the second, etc.

A piece is an *atom* possibly followed by a single† ‘*’, ‘+’, ‘?’, or *bound*. An atom followed by ‘*’ matches a sequence of 0 or more matches of the atom. An atom followed by ‘+’ matches a sequence of 1 or more matches of the atom. An atom followed by ‘?’ matches a sequence of 0 or 1 matches of the atom.

A *bound* is ‘{’ followed by an unsigned decimal integer, possibly followed by ‘,’ possibly followed by another unsigned decimal integer, always followed by ‘}’. The integers must lie between 0 and RE_DUP_MAX (255†) inclusive, and if there are two of them, the first may not exceed the second. An atom followed by a bound containing one integer *i* and no comma matches a sequence of exactly *i* matches of the atom. An atom followed by a bound containing one integer *i* and a comma matches a sequence of *i* or more matches of the atom. An atom followed by a bound containing two integers *i* and *j* matches a sequence of *i* through *j* (inclusive) matches of the atom.

An atom is a regular expression enclosed in ‘()’ (matching a match for the regular expression), an empty set of ‘()’ (matching the null string)†, a *bracket expression* (see below), ‘.’ (matching any single character), ‘^’ (matching the null string at the beginning of a line), ‘\$’ (matching the null string at the end of a line), a ‘\’ followed by one of the characters ‘^.\$()|*+?{\’ (matching that character taken as an ordinary character), a ‘\’ followed by any other character† (matching that character taken as an ordinary character, as if the ‘\’ had not been present†), or a single character with no other significance (matching that character). A ‘{’ followed by a character other than a digit is an ordinary character, not the beginning of a bound†. It is illegal to end an RE with ‘\’.

A *bracket expression* is a list of characters enclosed in ‘[]’. It normally matches any single character from the list (but see below). If the list begins with ‘^’, it matches any single character (but see below) *not* from the rest of the list. If two characters in the list are separated by ‘–’, this is shorthand for the full *range* of characters between those two (inclusive) in the collating sequence, e.g. ‘[0-9]’ in ASCII matches any decimal digit. It is illegal† for two ranges to share an endpoint, e.g. ‘a-c-e’. Ranges are very collating-sequence-dependent, and portable programs should avoid relying on them.

To include a literal ‘]’ in the list, make it the first character (following a possible ‘^’). To include a literal ‘–’, make it the first or last character, or the second endpoint of a range. To use a literal ‘^’ as the first endpoint of a range, enclose it in ‘[.’ and ‘.]’ to make it a collating element (see below). With the exception of these and some combinations using ‘[’ (see next paragraphs), all other special characters, including ‘\’, lose their special significance within a bracket expression.

Within a bracket expression, a collating element (a character, a multi-character sequence that collates as if it were a single character, or a collating-sequence name for either) enclosed in ‘[.’ and ‘.]’ stands for the sequence of characters of that collating element. The sequence is a single element of the bracket expression’s list. A bracket expression containing a multi-character collating element can thus match more than one character, e.g. if the collating sequence includes a ‘ch’ collating element, then the RE ‘[[.ch.]]*c’ matches the first five characters of ‘chchcc’.

Within a bracket expression, a collating element enclosed in ‘[=’ and ‘=]’ is an equivalence class, standing for the sequences of characters of all collating elements equivalent to that one, including itself. (If there are no other equivalent collating elements, the treatment is as if the enclosing delimiters were ‘[.’ and ‘.]’.) For

example, if `o` and `ô` are the members of an equivalence class, then `'[[=o=]]'`, `'[[=ô=]]'`, and `'[oô]'` are all synonymous. An equivalence class may not† be an endpoint of a range.

Within a bracket expression, the name of a *character class* enclosed in `'[:'` and `:]'` stands for the list of all characters belonging to that class. Standard character class names are:

alnum	digit	punct
alpha	graph	space
blank	lower	upper
cntrl	print	xdigit

These stand for the character classes defined in `ctype(3)`. A locale may provide others. A character class may not be used as an endpoint of a range.

There are two special cases† of bracket expressions: the bracket expressions `'[[:<:]]'` and `'[[:>:]]'` match the null string at the beginning and end of a word respectively. A word is defined as a sequence of word characters which is neither preceded nor followed by word characters. A word character is an **alnum** character (as defined by `ctype(3)`) or an underscore. This is an extension, compatible with but not specified by POSIX 1003.2, and should be used with caution in software intended to be portable to other systems.

In the event that an RE could match more than one substring of a given string, the RE matches the one starting earliest in the string. If the RE could match more than one substring starting at that point, it matches the longest. Subexpressions also match the longest possible substrings, subject to the constraint that the whole match be as long as possible, with subexpressions starting earlier in the RE taking priority over ones starting later. Note that higher-level subexpressions thus take priority over their lower-level component subexpressions.

Match lengths are measured in characters, not collating elements. A null string is considered longer than no match at all. For example, `'bb*'` matches the three middle characters of `'abbbc'`, `'(weelweek)(knightslnights)'` matches all ten characters of `'weeknights'`, when `'(.*)*'` is matched against `'abc'` the parenthesized subexpression matches all three characters, and when `'(a*)*'` is matched against `'bc'` both the whole RE and the parenthesized subexpression match the null string.

If case-independent matching is specified, the effect is much as if all case distinctions had vanished from the alphabet. When an alphabetic that exists in multiple cases appears as an ordinary character outside a bracket expression, it is effectively transformed into a bracket expression containing both cases, e.g. `'x'` becomes `'[xX]'`. When it appears inside a bracket expression, all case counterparts of it are added to the bracket expression, so that (e.g.) `'[x]'` becomes `'[xX]'` and `'[^x]'` becomes `'[^xX]'`.

No particular limit is imposed on the length of REs†. Programs intended to be portable should not employ REs longer than 256 bytes, as an implementation can refuse to accept such REs and remain POSIX-compliant.

Obsolete (“basic”) regular expressions differ in several respects. `'|'`, `'+'`, and `'?'` are ordinary characters and there is no equivalent for their functionality. The delimiters for bounds are `'\{'` and `'\}'`, with `'{'` and `'}'` by themselves ordinary characters. The parentheses for nested subexpressions are `'\('` and `'\)'`, with `'('` and `')'` by themselves ordinary characters. `'^'` is an ordinary character except at the beginning of the RE or† the beginning of a parenthesized subexpression, `'$'` is an ordinary character except at the end of the RE or† the end of a parenthesized subexpression, and `'*'` is an ordinary character if it appears at the beginning of the RE or the beginning of a parenthesized subexpression (after a possible leading `'^'`). Finally, there is one new type of atom, a *back reference*: `'\'` followed by a non-zero decimal digit *d* matches the same sequence of characters matched by the *d*th parenthesized subexpression (numbering subexpressions by the positions of their opening parentheses, left to right), so that (e.g.) `'\[bc\]\1'` matches `'bb'` or `'cc'` but not `'bc'`.

SEE ALSO

`regex(3)`

POSIX 1003.2, section 2.8 (Regular Expression Notation).

BUGS

Having two kinds of REs is a botch.

The current 1003.2 spec says that ‘)’ is an ordinary character in the absence of an unmatched ‘(’; this was an unintentional result of a wording error, and change is likely. Avoid relying on it.

Back references are a dreadful botch, posing major problems for efficient implementations. They are also somewhat vaguely defined (does ‘a\\(b\\)*\\2\\)*d’ match ‘abbbd’?). Avoid using them.

1003.2’s specification of case-independent matching is vague. The “one case implies all cases” definition given above is current consensus among implementors as to the right interpretation.

The syntax for word boundaries is incredibly ugly.

```
/*-
 * Copyright (c) 1992, 1993, 1994 Henry Spencer.
 * Copyright (c) 1992, 1993, 1994
 *   The Regents of the University of California.  All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Henry Spencer.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *   must display the following acknowledgement:
 *       This product includes software developed by the University of
 *       California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *   may be used to endorse or promote products derived from this software
 *   without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)regcomp.c      8.5 (Berkeley) 3/20/94
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)regcomp.c      8.5 (Berkeley) 3/20/94";
#endif /* LIBC_SCCS and not lint */

#include <sys/types.h>
#include <stdio.h>
#include <string.h>
#ifdef __minix_vmd
#include <bsd/asciictype.h>
#else
#include <ctype.h>
#endif
#include <limits.h>
#include <stdlib.h>
#include <regex.h>

#include "utils.h"
#include "regex2.h"

#include "cclass.h"
#include "cname.h"

/*
 * parse structure, passed up and down to avoid global variables and
 * other clumsinesses
 */
struct parse {
    char *next;           /* next character in RE */
    char *end;            /* end of string (-> NUL normally) */
    int error;            /* has an error been seen? */
    sop *strip;           /* malloced strip */
    sopno ssize;          /* malloced strip size (allocated) */
    sopno slen;           /* malloced strip length (used) */
    int ncsalloc;         /* number of csets allocated */
    struct re_guts *g;
```

```

#       define NPAREN 10      /* we need to remember ( ) 1-9 for back refs */
       sopno pbegin[NPAREN];  /* -> ( ([0] unused) */
       sopno pend[NPAREN];    /* -> ) ([0] unused) */
};

/* ===== begin header generated by ./mkh ===== */
#ifdef __cplusplus
extern "C" {
#endif

/* === regcomp.c === */
static void p_ere(struct parse *p, int stop);
static void p_ere_exp(struct parse *p);
static void p_str(struct parse *p);
static void p_bre(struct parse *p, int endl, int end2);
static int p_simp_re(struct parse *p, int starordinary);
static int p_count(struct parse *p);
static void p_bracket(struct parse *p);
static void p_b_term(struct parse *p, cset *cs);
static void p_b_cclass(struct parse *p, cset *cs);
static void p_b_ecclass(struct parse *p, cset *cs);
static char p_b_symbol(struct parse *p);
static char p_b_coll_elem(struct parse *p, int endc);
static char othercase(int ch);
static void bothcases(struct parse *p, int ch);
static void ordinary(struct parse *p, int ch);
static void nonnewline(struct parse *p);
static void repeat(struct parse *p, sopno start, int from, int to);
static int seterr(struct parse *p, int e);
static cset *allocset(struct parse *p);
static void freeset(struct parse *p, cset *cs);
static int freezeset(struct parse *p, cset *cs);
static int firstch(struct parse *p, cset *cs);
static int nch(struct parse *p, cset *cs);
static void mcadd(struct parse *p, cset *cs, char *cp);
static void mcsb(struct parse *p, cset *cs, char *cp);
static int mcin(struct parse *p, cset *cs, char *cp);
static char *mcfind(struct parse *p, cset *cs, char *cp);
static void mcinvert(struct parse *p, cset *cs);
static void mcase(struct parse *p, cset *cs);
static int isinsets(struct re_guts *g, int c);
static int same_sets(struct re_guts *g, int c1, int c2);
static void categorize(struct parse *p, struct re_guts *g);
static sopno dupl(struct parse *p, sopno start, sopno finish);
static void doemit(struct parse *p, sop op, size_t opnd);
static void doinsert(struct parse *p, sop op, size_t opnd, sopno pos);
static void dofwd(struct parse *p, sopno pos, sop value);
static void enlarge(struct parse *p, sopno size);
static void stripsnug(struct parse *p, struct re_guts *g);
static void findmust(struct parse *p, struct re_guts *g);
static sopno pluscount(struct parse *p, struct re_guts *g);

#ifdef __cplusplus
}
#endif

/* ===== end header generated by ./mkh ===== */

static char nuls[10];          /* place to point scanner in event of error */

/*
 * macros for use with parse structure
 * BEWARE: these know that the parse structure is named 'p' !!!
 */
#define PEEK() (*p->next)
#define PEEK2() (*p->next+1)
#define MORE() (p->next < p->end)
#define MORE2() (p->next+1 < p->end)
#define SEE(c) (MORE() && PEEK() == (c))
#define SEETWO(a, b) (MORE() && MORE2() && PEEK() == (a) && PEEK2() == (b))
#define EAT(c) ((SEE(c)) ? (NEXT(), 1) : 0)
#define EATTWO(a, b) ((SEETWO(a, b)) ? (NEXT2(), 1) : 0)
#define NEXT() (p->next++)
#define NEXT2() (p->next += 2)
#define NEXTn(n) (p->next += (n))

```

```

#define GETNEXT()      (*p->next++)
#define SETERROR(e)    seterr(p, (e))
#define REQUIRE(co, e) ((co) || SETERROR(e))
#define MUSTSEE(c, e)  (REQUIRE(MORE() && PEEK() == (c), e))
#define MUSTEAT(c, e)  (REQUIRE(MORE() && GETNEXT() == (c), e))
#define MUSTNOTSEE(c, e) (REQUIRE(!MORE() || PEEK() != (c), e))
#define EMIT(op, sopnd) doemit(p, (sop)(op), (size_t)(sopnd))
#define INSERT(op, pos) doinsert(p, (sop)(op), HERE()-(pos)+1, pos)
#define AHEAD(pos)      dofwd(p, pos, HERE()-(pos))
#define ASTERN(sop, pos) EMIT(sop, HERE()-(pos))
#define HERE()          (p->slen)
#define THERE()         (p->slen - 1)
#define THERETHERE()    (p->slen - 2)
#define DROP(n) (p->slen -= (n))

#ifndef NDEBUG
static int never = 0;          /* for use in asserts; shuts lint up */
#else
#define never 0                /* some <assert.h>s have bugs too */
#endif

/*
- regcomp - interface for parser and compilation
= extern int regcomp(regex_t *, const char *, int);
= #define      REG_BASIC      0000
= #define      REG_EXTENDED   0001
= #define      REG_ICASE      0002
= #define      REG_NOSUB      0004
= #define      REG_NEWLINE    0010
= #define      REG_NOSPEC     0020
= #define      REG_PEND       0040
= #define      REG_DUMP      0200
*/
int
regcomp(preg, pattern, cflags) /* 0 success, otherwise REG_something */
regex_t *preg;
const char *pattern;
int cflags;
{
    struct parse pa;
    register struct re_guts *g;
    register struct parse *p = &pa;
    register int i;
    register size_t len;
#ifdef REDEBUG
#   define GOODFLAGS(f)      (f)
#else
#   define GOODFLAGS(f)      ((f)&~REG_DUMP)
#endif

    cflags = GOODFLAGS(cflags);
    if ((cflags&REG_EXTENDED) && (cflags&REG_NOSPEC))
        return(REG_INVARG);

    if (cflags&REG_PEND) {
        if (preg->re_endp < pattern)
            return(REG_INVARG);
        len = preg->re_endp - pattern;
    } else
        len = strlen((char *)pattern);

    /* do the mallocs early so failure handling is easy */
    g = (struct re_guts *)malloc(sizeof(struct re_guts) +
                                (NC-1)*sizeof(cat_t));

    if (g == NULL)
        return(REG_ESPACE);
    p->ssize = len/(size_t)2*(size_t)3 + (size_t)1; /* ugh */
    p->strip = (sop *)malloc(p->ssize * sizeof(sop));
    p->slen = 0;
    if (p->strip == NULL) {
        free((char *)g);
        return(REG_ESPACE);
    }
}

```

```

/* set things up */
p->g = g;
p->next = (char *)pattern;      /* convenience; we do not modify it */
p->end = p->next + len;
p->error = 0;
p->ncsalloc = 0;
for (i = 0; i < NPAREN; i++) {
    p->pbegin[i] = 0;
    p->pend[i] = 0;
}
g->csetsize = NC;
g->sets = NULL;
g->setbits = NULL;
g->ncsets = 0;
g->cflags = cflags;
g->iflags = 0;
g->nbol = 0;
g->neol = 0;
g->must = NULL;
g->mten = 0;
g->nsup = 0;
g->ncategories = 1;      /* category 0 is "everything else" */
g->categories = &g->catspace[-(CHAR_MIN)];
(void) memset((char *)g->catspace, 0, NC*sizeof(cat_t));
g->backrefs = 0;

/* do it */
EMIT(OEND, 0);
g->firststate = THERE();
if (cflags&REG_EXTENDED)
    p_ere(p, OUT);
else if (cflags&REG_NOSPEC)
    p_str(p);
else
    p_bre(p, OUT, OUT);
EMIT(OEND, 0);
g->laststate = THERE();

/* tidy up loose ends and fill things in */
categorize(p, g);
stripsnug(p, g);
findmust(p, g);
g->nplus = pluscount(p, g);
g->magic = MAGIC2;
preg->re_nsub = g->nsup;
preg->re_g = g;
preg->re_magic = MAGIC1;
#ifdef REDEBUG
/* not debugging, so can't rely on the assert() in regexec() */
if (g->iflags&BAD)
    SETERROR(REG_ASSERT);
#endif

/* win or lose, we're done */
if (p->error != 0)      /* lose */
    regfree(preg);
return(p->error);
}

/*
- p_ere - ERE parser top level, concatenation and alternation
== static void p_ere(register struct parse *p, int stop);
*/
static void
p_ere(p, stop)
register struct parse *p;
int stop;                /* character this ERE should end at */
{
    register char c;
    register sopno prevback;
    register sopno prevfwd;
    register sopno conc;
    register int first = 1;      /* is this the first alternative? */

```

```

    for (;;) {
        /* do a bunch of concatenated expressions */
        conc = HERE();
        while (MORE() && (c = PEEK()) != '|' && c != stop)
            p_ere_exp(p);
        REQUIRE(HERE() != conc, REG_EMPTY);      /* require nonempty */

        if (!EAT('|'))
            break;                                /* NOTE BREAK OUT */

        if (first) {
            INSERT(OCH_, conc);      /* offset is wrong */
            prevfwd = conc;
            prevback = conc;
            first = 0;
        }
        ASTERN(OOR1, prevback);
        prevback = THERE();
        AHEAD(prevfwd);                /* fix previous offset */
        prevfwd = HERE();
        EMIT(OOR2, 0);                /* offset is very wrong */
    }

    if (!first) {                          /* tail-end fixups */
        AHEAD(prevfwd);
        ASTERN(O_CH, prevback);
    }

    assert(!MORE() || SEE(stop));
}

/*
 - p_ere_exp - parse one subERE, an atom possibly followed by a repetition op
 == static void p_ere_exp(register struct parse *p);
 */
static void
p_ere_exp(p)
register struct parse *p;
{
    register char c;
    register sopno pos;
    register int count;
    register int count2;
    register sopno subno;
    int wascaret = 0;

    assert(MORE());                      /* caller should have ensured this */
    c = GETNEXT();

    pos = HERE();
    switch (c) {
        case '(':
            REQUIRE(MORE(), REG_EPAREN);
            p->g->nsub++;
            subno = p->g->nsub;
            if (subno < NPAREN)
                p->pbegin[subno] = HERE();
            EMIT(OLPAREN, subno);
            if (!SEE(')'))
                p_ere(p, ')');
            if (subno < NPAREN) {
                p->pend[subno] = HERE();
                assert(p->pend[subno] != 0);
            }
            EMIT(ORPAREN, subno);
            MUSTEAT(')', REG_EPAREN);
            break;
#ifdef POSIX_MISTAKE
        case ')':
            /* happens only if no current unmatched ( */
            /*
             * You may ask, why the ifdef? Because I didn't notice
             * this until slightly too late for 1003.2, and none of the
             * other 1003.2 regular-expression reviewers noticed it at
             * all. So an unmatched ) is legal POSIX, at least until

```

```

        * we can get it fixed.
        */
        SETERROR(REG_EPAREN);
        break;
#endif
    case '^':
        EMIT(OBOL, 0);
        p->g->iflags |= USEBOL;
        p->g->nbol++;
        wascaret = 1;
        break;
    case '$':
        EMIT(OEOL, 0);
        p->g->iflags |= USEEOL;
        p->g->neol++;
        break;
    case '|':
        SETERROR(REG_EMPTY);
        break;
    case '*':
    case '+':
    case '?':
        SETERROR(REG_BADRPT);
        break;
    case '.':
        if (p->g->cflags & REG_NEWLINE)
            nonnewline(p);
        else
            EMIT(OANY, 0);
        break;
    case '[':
        p_bracket(p);
        break;
    case '\\':
        REQUIRE(MORE(), REG_EESCAPE);
        c = GETNEXT();
        ordinary(p, c);
        break;
    case '{':
        /* okay as ordinary except if digit follows */
        REQUIRE(!MORE() || !isdigit(PEEK()), REG_BADRPT);
        /* FALLTHROUGH */
    default:
        ordinary(p, c);
        break;
}

if (!MORE())
    return;
c = PEEK();
/* we call { a repetition if followed by a digit */
if (!(c == '*' || c == '+' || c == '?' ||
      (c == '{' && MORE2() && isdigit(PEEK2()))))
    return;
/* no repetition, we're done */
NEXT();

REQUIRE(!wascaret, REG_BADRPT);
switch (c) {
    case '*':
        /* implemented as +? */
        /* this case does not require the (y/) trick, noKLUDGE */
        INSERT(OPLUS_, pos);
        ASTERN(O_PLUS, pos);
        INSERT(OQUEST_, pos);
        ASTERN(O_QUEST, pos);
        break;
    case '+':
        INSERT(OPLUS_, pos);
        ASTERN(O_PLUS, pos);
        break;
    case '?':
        /* KLUDGE: emit y? as (y/) until subtle bug gets fixed */
        INSERT(OCH_, pos);
        ASTERN(OOR1, pos);
        AHEAD(pos);
        EMIT(OOR2, 0);
        /* offset slightly wrong */
        /* this one's right */
        /* fix the OCH_ */
        /* offset very wrong... */

```



```

        AHEAD(THERE()); /* ...so fix it */
        ASTERN(O_CH, THERETHERE());
        break;
    case ' ':
        count = p_count(p);
        if (EAT(',')) {
            if (isdigit(PEEK())) {
                count2 = p_count(p);
                REQUIRE(count <= count2, REG_BADBR);
            } else /* single number with comma */
                count2 = INFINITY;
        } else /* just a single number */
            count2 = count;
        repeat(p, pos, count, count2);
        if (!EAT('}')) { /* error heuristics */
            while (MORE() && PEEK() != '}')
                NEXT();
            REQUIRE(MORE(), REG_EBRACE);
            SETERROR(REG_BADBR);
        }
        break;
    }

    if (!MORE())
        return;
    c = PEEK();
    if (!(c == '*' || c == '+' || c == '?' ||
        (c == '{' && MORE2() && isdigit(PEEK2()))))
        return;
    SETERROR(REG_BADRPT);
}

/*
- p_str - string (no metacharacters) "parser"
== static void p_str(register struct parse *p);
*/
static void
p_str(p)
register struct parse *p;
{
    REQUIRE(MORE(), REG_EMPTY);
    while (MORE())
        ordinary(p, GETNEXT());
}

/*
- p_bre - BRE parser top level, anchoring and concatenation
== static void p_bre(register struct parse *p, register int end1, \
== register int end2);
* Giving end1 as OUT essentially eliminates the end1/end2 check.
*
* This implementation is a bit of a kludge, in that a trailing $ is first
* taken as an ordinary character and then revised to be an anchor. The
* only undesirable side effect is that '$' gets included as a character
* category in such cases. This is fairly harmless; not worth fixing.
* The amount of lookahead needed to avoid this kludge is excessive.
*/
static void
p_bre(p, end1, end2)
register struct parse *p;
register int end1; /* first terminating character */
register int end2; /* second terminating character */
{
    register sopno start = HERE();
    register int first = 1; /* first subexpression? */
    register int wasdollar = 0;

    if (EAT('^')) {
        EMIT(OBOL, 0);
        p->g->iflags |= USEBOL;
        p->g->nbol++;
    }
    while (MORE() && !SEETWO(end1, end2)) {
        wasdollar = p_simp_re(p, first);

```

```

        first = 0;
    }
    if (wasdollar) {          /* oops, that was a trailing anchor */
        DROP(1);
        EMIT(OEOL, 0);
        p->g->iflags |= USEEOL;
        p->g->neol++;
    }

    REQUIRE(HERE() != start, REG_EMPTY);    /* require nonempty */
}

/*
 - p_simp_re - parse a simple RE, an atom possibly followed by a repetition
 == static int p_simp_re(register struct parse *p, int starordinary);
 */
static int                                /* was the simple RE an unbackslashed $? */
p_simp_re(p, starordinary)
register struct parse *p;
int starordinary;                        /* is a leading * an ordinary character? */
{
    register int c;
    register int count;
    register int count2;
    register sopno pos;
    register int i;
    register sopno subno;
#    define BACKSL (1<<CHAR_BIT)

    pos = HERE();                        /* repition op, if any, covers from here */

    assert(MORE());                      /* caller should have ensured this */
    c = GETNEXT();
    if (c == '\\') {
        REQUIRE(MORE(), REG_EESCAPE);
        c = BACKSL | (unsigned char)GETNEXT();
    }
    switch (c) {
    case '.':
        if (p->g->cflags&REG_NEWLINE)
            nonnewline(p);
        else
            EMIT(OANY, 0);
        break;
    case '[':
        p_bracket(p);
        break;
    case BACKSL | '{':
        SETERROR(REG_BADRPT);
        break;
    case BACKSL | '(':
        p->g->nsub++;
        subno = p->g->nsub;
        if (subno < NPAREN)
            p->pbegin[subno] = HERE();
        EMIT(OLPAREN, subno);
        /* the MORE here is an error heuristic */
        if (MORE() && !SEETWO('\\', '('))
            p_bre(p, '\\', '(');
        if (subno < NPAREN) {
            p->pend[subno] = HERE();
            assert(p->pend[subno] != 0);
        }
        EMIT(ORPAREN, subno);
        REQUIRE(EATTWO('\\', '('), REG_EPAREN);
        break;
    case BACKSL | ')':
        /* should not get here -- must be user */
    case BACKSL | '}':
        SETERROR(REG_EPAREN);
        break;
    case BACKSL | '1':
    case BACKSL | '2':
    case BACKSL | '3':
    case BACKSL | '4':

```

```

case BACKSL '5':
case BACKSL '6':
case BACKSL '7':
case BACKSL '8':
case BACKSL '9':
    i = (c & ~BACKSL) - '0';
    assert(i < NPAREN);
    if (p->pend[i] != 0) {
        assert(i <= p->g->nsub);
        EMIT(OBACK_, i);
        assert(p->pbegin[i] != 0);
        assert(OP(p->strip[p->pbegin[i]]) == OLPAREN);
        assert(OP(p->strip[p->pend[i]]) == ORPAREN);
        (void) dupl(p, p->pbegin[i]+1, p->pend[i]);
        EMIT(O_BACK, i);
    } else
        SETERROR(REG_ESUBREG);
    p->g->backrefs = 1;
    break;
case '*':
    REQUIRE(starordinary, REG_BADRPT);
    /* FALLTHROUGH */
default:
    ordinary(p, c & ~ BACKSL);
    break;
}

if (EAT('*')) {
    /* implemented as +? */
    /* this case does not require the (y|) trick, noKLUDGE */
    INSERT(OPLUS_, pos);
    ASTERN(O_PLUS, pos);
    INSERT(OQUEST_, pos);
    ASTERN(O_QUEST, pos);
} else if (EATTWO('\\', '{')) {
    count = p_count(p);
    if (EAT(',')) {
        if (MORE() && isdigit(PEEK())) {
            count2 = p_count(p);
            REQUIRE(count <= count2, REG_BADBR);
        } else
            /* single number with comma */
            count2 = INFINITY;
        } else
            /* just a single number */
            count2 = count;
    repeat(p, pos, count, count2);
    if (!EATTWO('\\', '}')) {
        /* error heuristics */
        while (MORE() && !SEETWO('\\', '}'))
            NEXT();
        REQUIRE(MORE(), REG_EBRACE);
        SETERROR(REG_BADBR);
    }
} else if (c == (unsigned char) '$')
    /* $ (but not \$) ends it */
    return(1);

return(0);
}

/*
- p_count - parse a repetition count
== static int p_count(register struct parse *p);
*/
static int
p_count(p)
register struct parse *p;
{
    register int count = 0;
    register int ndigits = 0;

    while (MORE() && isdigit(PEEK()) && count <= DUPMAX) {
        count = count*10 + (GETNEXT() - '0');
        ndigits++;
    }

    REQUIRE(ndigits > 0 && count <= DUPMAX, REG_BADBR);
    return(count);
}

```

```

}

/*
 * - p_bracket - parse a bracketed character list
 * == static void p_bracket(register struct parse *p);
 *
 * * Note a significant property of this code: if the allocset() did SETERROR,
 * * no set operations are done.
 */
static void
p_bracket(p)
register struct parse *p;
{
    register char c;
    register cset *cs = allocset(p);
    register int invert = 0;

    /* Dept of Truly Sickening Special-Case Kludges */
    if (p->next + 5 < p->end && strncmp(p->next, "[<:]", 6) == 0) {
        EMIT(OBOW, 0);
        NEXTn(6);
        return;
    }
    if (p->next + 5 < p->end && strncmp(p->next, "[>:]", 6) == 0) {
        EMIT(OEOW, 0);
        NEXTn(6);
        return;
    }

    if (EAT('^'))
        invert++;          /* make note to invert set at end */
    if (EAT(']'))
        CHadd(cs, ']');
    else if (EAT('-'))
        CHadd(cs, '-');
    while (MORE() && PEEK() != ']' && !SEETWO('-', ']'))
        p_b_term(p, cs);
    if (EAT('-'))
        CHadd(cs, '-');
    MUSTEAT(']', REG_EBRACK);

    if (p->error != 0)      /* don't mess things up further */
        return;

    if (p->g->cflags&REG_ICASE) {
        register int i;
        register int ci;

        for (i = p->g->csetsize - 1; i >= 0; i--)
            if (CHIN(cs, i) && isalpha(i)) {
                ci = othercase(i);
                if (ci != i)
                    CHadd(cs, ci);
            }
        if (cs->multis != NULL)
            mccase(p, cs);
    }
    if (invert) {
        register int i;

        for (i = p->g->csetsize - 1; i >= 0; i--)
            if (CHIN(cs, i))
                CHsub(cs, i);
            else
                CHadd(cs, i);
        if (p->g->cflags&REG_NEWLINE)
            CHsub(cs, '\n');
        if (cs->multis != NULL)
            mcinvert(p, cs);
    }

    assert(cs->multis == NULL);          /* xxx */

    if (nch(p, cs) == 1) {              /* optimize singleton sets */

```

```

        ordinary(p, firstch(p, cs));
        freeset(p, cs);
    } else
        EMIT(OANYOF, freezeset(p, cs));
}

/*
 - p_b_term - parse one term of a bracketed character list
 == static void p_b_term(register struct parse *p, register cset *cs);
 */
static void
p_b_term(p, cs)
register struct parse *p;
register cset *cs;
{
    register char c;
    register char start, finish;
    register int i;

    /* classify what we've got */
    switch ((MORE()) ? PEEK() : '\0') {
    case '[':
        c = (MORE2()) ? PEEK2() : '\0';
        break;
    case '-':
        SETERROR(REG_ERANGE);
        return; /* NOTE RETURN */
        break;
    default:
        c = '\0';
        break;
    }

    switch (c) {
    case ':': /* character class */
        NEXT2();
        REQUIRE(MORE(), REG_EBRACK);
        c = PEEK();
        REQUIRE(c != '-' && c != ']', REG_ECTYPE);
        p_b_cclass(p, cs);
        REQUIRE(MORE(), REG_EBRACK);
        REQUIRE(EATTWO(':', ']'), REG_ECTYPE);
        break;
    case '=': /* equivalence class */
        NEXT2();
        REQUIRE(MORE(), REG_EBRACK);
        c = PEEK();
        REQUIRE(c != '-' && c != ']', REG_ECOLLATE);
        p_b_ecclass(p, cs);
        REQUIRE(MORE(), REG_EBRACK);
        REQUIRE(EATTWO('=', ']'), REG_ECOLLATE);
        break;
    default: /* symbol, ordinary character, or range */
        /* xxx revision needed for multichar stuff */
        start = p_b_symbol(p);
        if (SEE('-',) && MORE2() && PEEK2() != ']') {
            /* range */
            NEXT();
            if (EAT('-'))
                finish = '-';
            else
                finish = p_b_symbol(p);
        } else
            finish = start;
        /* xxx what about signed chars here... */
        REQUIRE(start <= finish, REG_ERANGE);
        for (i = start; i <= finish; i++)
            CHadd(cs, i);
        break;
    }
}

/*
 - p_b_cclass - parse a character-class name and deal with it

```

```

== static void p_b_cclass(register struct parse *p, register cset *cs);
*/
static void
p_b_cclass(p, cs)
register struct parse *p;
register cset *cs;
{
    register char *sp = p->next;
    register struct cclass *cp;
    register size_t len;
    register char *u;
    register char c;

    while (MORE() && isalpha(PEEK()))
        NEXT();
    len = p->next - sp;
    for (cp = cclasses; cp->name != NULL; cp++)
        if (strncmp(cp->name, sp, len) == 0 && cp->name[len] == '\0')
            break;
    if (cp->name == NULL) {
        /* oops, didn't find it */
        SETERROR(REG_ECTYPE);
        return;
    }

    u = cp->chars;
    while ((c = *u++) != '\0')
        CHadd(cs, c);
    for (u = cp->multis; *u != '\0'; u += strlen(u) + 1)
        MCadd(p, cs, u);
}

/*
- p_b_ecclass - parse an equivalence-class name and deal with it
== static void p_b_ecclass(register struct parse *p, register cset *cs);
*
* This implementation is incomplete. xxx
*/
static void
p_b_ecclass(p, cs)
register struct parse *p;
register cset *cs;
{
    register char c;

    c = p_b_coll_elem(p, '=');
    CHadd(cs, c);
}

/*
- p_b_symbol - parse a character or [...]ed multicharacter collating symbol
== static char p_b_symbol(register struct parse *p);
*/
static char                                /* value of symbol */
p_b_symbol(p)
register struct parse *p;
{
    register char value;

    REQUIRE(MORE(), REG_EBRACK);
    if (!EATTWO('[', '.'))
        return(GETNEXT());

    /* collating symbol */
    value = p_b_coll_elem(p, '.');
    REQUIRE(EATTWO('.', ']'), REG_ECOLLATE);
    return(value);
}

/*
- p_b_coll_elem - parse a collating-element name and look it up
== static char p_b_coll_elem(register struct parse *p, int endc);
*/
static char                                /* value of collating element */

```

```

p_b_coll_elem(p, endc)
register struct parse *p;
int endc;                                /* name ended by endc, ']' */
{
    register char *sp = p->next;
    register struct cname *cp;
    register int len;
    register char c;

    while (MORE() && !SEETWO(endc, ' '))
        NEXT();
    if (!MORE()) {
        SETERROR(REG_EBRACK);
        return(0);
    }
    len = p->next - sp;
    for (cp = cnames; cp->name != NULL; cp++)
        if (strncmp(cp->name, sp, len) == 0 && cp->name[len] == '\0')
            return(cp->code);          /* known name */

    if (len == 1)
        return(*sp);                /* single character */
    SETERROR(REG_ECOLLATE);          /* neither */
    return(0);
}

/*
- othercase - return the case counterpart of an alphabetic
== static char othercase(int ch);
*/
static char                                /* if no counterpart, return ch */
othercase(ch)
int ch;
{
    assert(isalpha(ch));
    if (isupper(ch))
        return(tolower(ch));
    else if (islower(ch))
        return(toupper(ch));
    else
        return(ch);                /* peculiar, but could happen */
}

/*
- bothcases - emit a dualcase version of a two-case character
== static void bothcases(register struct parse *p, int ch);
*
* Boy, is this implementation ever a kludge...
*/
static void
bothcases(p, ch)
register struct parse *p;
int ch;
{
    register char *oldnext = p->next;
    register char *oldend = p->end;
    char bracket[3];

    assert(othercase(ch) != ch);      /* p_bracket() would recurse */
    p->next = bracket;
    p->end = bracket+2;
    bracket[0] = ch;
    bracket[1] = ' ';
    bracket[2] = '\0';
    p_bracket(p);
    assert(p->next == bracket+2);
    p->next = oldnext;
    p->end = oldend;
}

/*
- ordinary - emit an ordinary character
== static void ordinary(register struct parse *p, register int ch);
*/
static void

```

```

ordinary(p, ch)
register struct parse *p;
register int ch;
{
    register cat_t *cap = p->g->categories;

    if ((p->g->cflags&REG_ICASE) && isalpha(ch) && othercase(ch) != ch)
        bothcases(p, ch);
    else {
        EMIT(OCHAR, (unsigned char)ch);
        if (cap[ch] == 0)
            cap[ch] = p->g->ncategories++;
    }
}

/*
- nonnewline - emit REG_NEWLINE version of OANY
== static void nonnewline(register struct parse *p);
*/
/* Boy, is this implementation ever a kludge... */
static void
nonnewline(p)
register struct parse *p;
{
    register char *oldnext = p->next;
    register char *oldend = p->end;
    char bracket[4];

    p->next = bracket;
    p->end = bracket+3;
    bracket[0] = '^';
    bracket[1] = '\n';
    bracket[2] = ']';
    bracket[3] = '\0';
    p_bracket(p);
    assert(p->next == bracket+3);
    p->next = oldnext;
    p->end = oldend;
}

/*
- repeat - generate code for a bounded repetition, recursively if needed
== static void repeat(register struct parse *p, sopno start, int from, int to);
*/
static void
repeat(p, start, from, to)
register struct parse *p;
sopno start;          /* operand from here to end of strip */
int from;             /* repeated from this number */
int to;               /* to this number of times (maybe INFINITY) */
{
    register sopno finish = HERE();
#    define N      2
#    define INF    3
#    define REP(f, t)      ((f)*8 + (t))
#    define MAP(n)  (((n) <= 1) ? (n) : ((n) == INFINITY) ? INF : N)
    register sopno copy;

    if (p->error != 0)      /* head off possible runaway recursion */
        return;

    assert(from <= to);

    switch (REP(MAP(from), MAP(to))) {
case REP(0, 0):          /* must be user doing this */
        DROP(finish-start); /* drop the operand */
        break;
case REP(0, 1):          /* as x{1,1}? */
case REP(0, N):          /* as x{1,n}? */
case REP(0, INF):        /* as x{1,}? */
        /* KLUDGE: emit y? as (y|) until subtle bug gets fixed */
        INSERT(OCH_, start); /* offset is wrong... */
        repeat(p, start+1, 1, to);
    }
}

```



```

        ASTERN(OOR1, start);
        AHEAD(start); /* ... fix it */
        EMIT(OOR2, 0);
        AHEAD(THERE());
        ASTERN(O_CH, THERETHERE());
        break;
    case REP(1, 1): /* trivial case */
        /* done */
        break;
    case REP(1, N): /* as x?x{1,n-1} */
        /* KLUDGE: emit y? as (y|) until subtle bug gets fixed */
        INSERT(OCH_, start);
        ASTERN(OOR1, start);
        AHEAD(start);
        EMIT(OOR2, 0); /* offset very wrong... */
        AHEAD(THERE()); /* ...so fix it */
        ASTERN(O_CH, THERETHERE());
        copy = dupl(p, start+1, finish+1);
        assert(copy == finish+4);
        repeat(p, copy, 1, to-1);
        break;
    case REP(1, INF): /* as x+ */
        INSERT(OPLUS_, start);
        ASTERN(O_PLUS, start);
        break;
    case REP(N, N): /* as xx{m-1,n-1} */
        copy = dupl(p, start, finish);
        repeat(p, copy, from-1, to-1);
        break;
    case REP(N, INF): /* as xx{n-1,INF} */
        copy = dupl(p, start, finish);
        repeat(p, copy, from-1, to);
        break;
    default: /* "can't happen" */
        SETERROR(REG_ASSERT); /* just in case */
        break;
}
}

/*
- seterr - set an error condition
== static int seterr(register struct parse *p, int e);
*/
static int /* useless but makes type checking happy */
seterr(p, e)
register struct parse *p;
int e;
{
    if (p->error == 0) /* keep earliest error condition */
        p->error = e;
    p->next = nuls; /* try to bring things to a halt */
    p->end = nuls;
    return(0); /* make the return value well-defined */
}

/*
- allocset - allocate a set of characters for []
== static cset *allocset(register struct parse *p);
*/
static cset *
allocset(p)
register struct parse *p;
{
    register int no = p->g->ncsets++;
    register size_t nc;
    register size_t nbytes;
    register cset *cs;
    register size_t css = (size_t)p->g->csetsize;
    register int i;

    if (no >= p->ncsalloc) { /* need another column of space */
        p->ncsalloc += CHAR_BIT;
        nc = p->ncsalloc;
        assert(nc % CHAR_BIT == 0);
    }

```

```

    nbytes = nc / CHAR_BIT * css;
    if (p->g->sets == NULL)
        p->g->sets = (cset *)malloc(nc * sizeof(cset));
    else
        p->g->sets = (cset *)realloc((char *)p->g->sets,
                                     nc * sizeof(cset));
    if (p->g->setbits == NULL)
        p->g->setbits = (uch *)malloc(nbytes);
    else {
        p->g->setbits = (uch *)realloc((char *)p->g->setbits,
                                       nbytes);
        /* xxx this isn't right if setbits is now NULL */
        for (i = 0; i < no; i++)
            p->g->sets[i].ptr = p->g->setbits + css*(i/CHAR_BIT);
    }
    if (p->g->sets != NULL && p->g->setbits != NULL)
        (void) memset((char *)p->g->setbits + (nbytes - css),
                      0, css);
    else {
        no = 0;
        SETERROR(REG_ESPACE);
        /* caller's responsibility not to do set ops */
    }
}

assert(p->g->sets != NULL); /* xxx */
cs = &p->g->sets[no];
cs->ptr = p->g->setbits + css*((no)/CHAR_BIT);
cs->mask = 1 << ((no) % CHAR_BIT);
cs->hash = 0;
cs->smultis = 0;
cs->multis = NULL;

return(cs);
}

/*
- freeset - free a now-unused set
== static void freeset(register struct parse *p, register cset *cs);
*/
static void
freeset(p, cs)
register struct parse *p;
register cset *cs;
{
    register int i;
    register cset *top = &p->g->sets[p->g->ncsets];
    register size_t css = (size_t)p->g->csetsize;

    for (i = 0; i < css; i++)
        CHsub(cs, i);
    if (cs == top-1) /* recover only the easy case */
        p->g->ncsets--;
}

/*
- freezeset - final processing on a set of characters
== static int freezeset(register struct parse *p, register cset *cs);
*
* The main task here is merging identical sets. This is usually a waste
* of time (although the hash code minimizes the overhead), but can win
* big if REG_ICASE is being used. REG_ICASE, by the way, is why the hash
* is done using addition rather than xor -- all ASCII [aA] sets xor to
* the same value!
*/
static int /* set number */
freezeset(p, cs)
register struct parse *p;
register cset *cs;
{
    register uch h = cs->hash;
    register int i;
    register cset *top = &p->g->sets[p->g->ncsets];
    register cset *cs2;

```

```

    register size_t css = (size_t)p->g->csetsize;

    /* look for an earlier one which is the same */
    for (cs2 = &p->g->sets[0]; cs2 < top; cs2++)
        if (cs2->hash == h && cs2 != cs) {
            /* maybe */
            for (i = 0; i < css; i++)
                if (!!CHIN(cs2, i) != !!CHIN(cs, i))
                    break; /* no */
            if (i == css)
                break; /* yes */
        }

    if (cs2 < top) { /* found one */
        freeset(p, cs);
        cs = cs2;
    }

    return((int)(cs - p->g->sets));
}

/*
- firstch - return first character in a set (which must have at least one)
== static int firstch(register struct parse *p, register cset *cs);
*/
static int firstch(p, cs) /* character; there is no "none" value */
register struct parse *p;
register cset *cs;
{
    register int i;
    register size_t css = (size_t)p->g->csetsize;

    for (i = 0; i < css; i++)
        if (CHIN(cs, i))
            return((char)i);
    assert(never);
    return(0); /* arbitrary */
}

/*
- nch - number of characters in a set
== static int nch(register struct parse *p, register cset *cs);
*/
static int nch(p, cs)
register struct parse *p;
register cset *cs;
{
    register int i;
    register size_t css = (size_t)p->g->csetsize;
    register int n = 0;

    for (i = 0; i < css; i++)
        if (CHIN(cs, i))
            n++;
    return(n);
}

/*
- mcadd - add a collating element to a cset
== static void mcadd(register struct parse *p, register cset *cs, \
    register char *cp);
*/
static void mcadd(p, cs, cp)
register struct parse *p;
register cset *cs;
register char *cp;
{
    register size_t oldend = cs->smultis;

    cs->smultis += strlen(cp) + 1;
    if (cs->multis == NULL)

```

```

        cs->multis = malloc(cs->smultis);
    else
        cs->multis = realloc(cs->multis, cs->smultis);
    if (cs->multis == NULL) {
        SETERROR(REG_ESPACE);
        return;
    }

    (void) strcpy(cs->multis + oldend - 1, cp);
    cs->multis[cs->smultis - 1] = '\0';
}

/*
 - mcsb - subtract a collating element from a cset
 == static void mcsb(register cset *cs, register char *cp);
 */
static void
mcsb(cs, cp)
register cset *cs;
register char *cp;
{
    register char *fp = mcfind(cs, cp);
    register size_t len = strlen(fp);

    assert(fp != NULL);
    (void) memmove(fp, fp + len + 1,
                   cs->smultis - (fp + len + 1 - cs->multis));
    cs->smultis -= len;

    if (cs->smultis == 0) {
        free(cs->multis);
        cs->multis = NULL;
        return;
    }

    cs->multis = realloc(cs->multis, cs->smultis);
    assert(cs->multis != NULL);
}

/*
 - mcin - is a collating element in a cset?
 == static int mcin(register cset *cs, register char *cp);
 */
static int
mcin(cs, cp)
register cset *cs;
register char *cp;
{
    return(mcfind(cs, cp) != NULL);
}

/*
 - mcfind - find a collating element in a cset
 == static char *mcfind(register cset *cs, register char *cp);
 */
static char *
mcfind(cs, cp)
register cset *cs;
register char *cp;
{
    register char *p;

    if (cs->multis == NULL)
        return(NULL);
    for (p = cs->multis; *p != '\0'; p += strlen(p) + 1)
        if (strcmp(cp, p) == 0)
            return(p);
    return(NULL);
}

/*
 - mcinvert - invert the list of collating elements in a cset
 == static void mcinvert(register struct parse *p, register cset *cs);
 */

```

```

* This would have to know the set of possibilities. Implementation
* is deferred.
*/
static void
mcinvert(p, cs)
register struct parse *p;
register cset *cs;
{
    assert(cs->multis == NULL);    /* xxx */
}

/*
- mccase - add case counterparts of the list of collating elements in a cset
== static void mccase(register struct parse *p, register cset *cs);
*
* This would have to know the set of possibilities. Implementation
* is deferred.
*/
static void
mccase(p, cs)
register struct parse *p;
register cset *cs;
{
    assert(cs->multis == NULL);    /* xxx */
}

/*
- isinsets - is this character in any sets?
== static int isinsets(register struct re_guts *g, int c);
*/
static int                /* predicate */
isinsets(g, c)
register struct re_guts *g;
int c;
{
    register uch *col;
    register int i;
    register int ncols = (g->ncsets+(CHAR_BIT-1)) / CHAR_BIT;
    register unsigned uc = (unsigned char)c;

    for (i = 0, col = g->setbits; i < ncols; i++, col += g->csetsize)
        if (col[uc] != 0)
            return(1);

    return(0);
}

/*
- same sets - are these two characters in exactly the same sets?
== static int same sets(register struct re_guts *g, int c1, int c2);
*/
static int                /* predicate */
same sets(g, c1, c2)
register struct re_guts *g;
int c1;
int c2;
{
    register uch *col;
    register int i;
    register int ncols = (g->ncsets+(CHAR_BIT-1)) / CHAR_BIT;
    register unsigned uc1 = (unsigned char)c1;
    register unsigned uc2 = (unsigned char)c2;

    for (i = 0, col = g->setbits; i < ncols; i++, col += g->csetsize)
        if (col[uc1] != col[uc2])
            return(0);

    return(1);
}

/*
- categorize - sort out character categories
== static void categorize(struct parse *p, register struct re_guts *g);
*/
static void
categorize(p, g)

```

```

struct parse *p;
register struct re_guts *g;
{
    register cat_t *cats = g->categories;
    register int c;
    register int c2;
    register cat_t cat;

    /* avoid making error situations worse */
    if (p->error != 0)
        return;

    for (c = CHAR_MIN; c <= CHAR_MAX; c++)
        if (cats[c] == 0 && isinsets(g, c)) {
            cat = g->ncategories++;
            cats[c] = cat;
            for (c2 = c+1; c2 <= CHAR_MAX; c2++)
                if (cats[c2] == 0 && samechars(g, c, c2))
                    cats[c2] = cat;
        }
}

/*
- dupl - emit a duplicate of a bunch of sops
== static sopno dupl(register struct parse *p, sopno start, sopno finish);
*/
static sopno                                /* start of duplicate */
dupl(p, start, finish)
register struct parse *p;
sopno start;                                /* from here */
sopno finish;                               /* to this less one */
{
    register sopno ret = HERE();
    register sopno len = finish - start;

    assert(finish >= start);
    if (len == 0)
        return(ret);
    enlarge(p, p->ssoff + len);              /* this many unexpected additions */
    assert(p->ssoff >= p->slen + len);
    (void) memcpy((char *) (p->strip + p->slen),
        (char *) (p->strip + start), (size_t) len * sizeof(sop));
    p->slen += len;
    return(ret);
}

/*
- doemit - emit a strip operator
== static void doemit(register struct parse *p, sop op, size_t opnd);
*
* It might seem better to implement this as a macro with a function as
* hard-case backup, but it's just too big and messy unless there are
* some changes to the data structures. Maybe later.
*/
static void
doemit(p, op, opnd)
register struct parse *p;
sop op;
size_t opnd;
{
    /* avoid making error situations worse */
    if (p->error != 0)
        return;

    /* deal with oversize operands ("can't happen", more or less) */
    assert(opnd < 1<<OPSHIFT);

    /* deal with undersized strip */
    if (p->slen >= p->ssoff)
        enlarge(p, (p->ssoff+1) / 2 * 3);    /* +50% */
    assert(p->slen < p->ssoff);

    /* finally, it's all reduced to the easy case */
    p->strip[p->slen++] = SOP(op, opnd);
}

```

```

}

/*
- doinsert - insert a sop into the strip
== static void doinsert(register struct parse *p, sop op, size_t opnd, sopno pos);
*/
static void
doinsert(p, op, opnd, pos)
register struct parse *p;
sop op;
size_t opnd;
sopno pos;
{
    register sopno sn;
    register sop s;
    register int i;

    /* avoid making error situations worse */
    if (p->error != 0)
        return;

    sn = HERE();
    EMIT(op, opnd);          /* do checks, ensure space */
    assert(HERE() == sn+1);
    s = p->strip[sn];

    /* adjust paren pointers */
    assert(pos > 0);
    for (i = 1; i < NPAREN; i++) {
        if (p->pbegin[i] >= pos) {
            p->pbegin[i]++;
        }
        if (p->pend[i] >= pos) {
            p->pend[i]++;
        }
    }

    memmove((char *)&p->strip[pos+1], (char *)&p->strip[pos],
            (HERE()-pos-1)*sizeof(sop));
    p->strip[pos] = s;
}

/*
- dofwd - complete a forward reference
== static void dofwd(register struct parse *p, sopno pos, sop value);
*/
static void
dofwd(p, pos, value)
register struct parse *p;
register sopno pos;
sop value;
{
    /* avoid making error situations worse */
    if (p->error != 0)
        return;

    assert(value < 1<<OPSHIFT);
    p->strip[pos] = OP(p->strip[pos]) | value;
}

/*
- enlarge - enlarge the strip
== static void enlarge(register struct parse *p, sopno size);
*/
static void
enlarge(p, size)
register struct parse *p;
register sopno size;
{
    register sop *sp;

    if (p->ssize >= size)
        return;

```

```

    sp = (sop *)realloc(p->strip, size*sizeof(sop));
    if (sp == NULL) {
        SETERROR(REG_ESPACE);
        return;
    }
    p->strip = sp;
    p->ssize = size;
}

/*
 * - stripsnug - compact the strip
 * == static void stripsnug(register struct parse *p, register struct re_guts *g);
 */
static void
stripsnug(p, g)
register struct parse *p;
register struct re_guts *g;
{
    g->nstates = p->slen;
    g->strip = (sop *)realloc((char *)p->strip, p->slen * sizeof(sop));
    if (g->strip == NULL) {
        SETERROR(REG_ESPACE);
        g->strip = p->strip;
    }
}

/*
 * - findmust - fill in must and mlen with longest mandatory literal string
 * == static void findmust(register struct parse *p, register struct re_guts *g);
 *
 * This algorithm could do fancy things like analyzing the operands of |
 * for common subsequences. Someday. This code is simple and finds most
 * of the interesting cases.
 *
 * Note that must and mlen got initialized during setup.
 */
static void
findmust(p, g)
struct parse *p;
register struct re_guts *g;
{
    register sop *scan;
    sop *start;
    register sop *newstart;
    register sopno newlen;
    register sop s;
    register char *cp;
    register sopno i;

    /* avoid making error situations worse */
    if (p->error != 0)
        return;

    /* find the longest OCHAR sequence in strip */
    newlen = 0;
    scan = g->strip + 1;
    do {
        s = *scan++;
        switch (OP(s)) {
            case OCHAR: /* sequence member */
                if (newlen == 0) /* new sequence */
                    newstart = scan - 1;
                newlen++;
                break;
            case OPLUS_: /* things that don't break one */
            case OLPAREN:
            case ORPAREN:
                break;
            case OQUEST_: /* things that must be skipped */
            case OCH_:
                scan--;
                do {
                    scan += OPND(s);
                    s = *scan;
                } while (s != OCH_);
        }
    } while (scan < g->strip + p->ssize);
    newlen = scan - newstart;
    g->must = newstart;
    g->mlen = newlen;
}

```



```

        /* assert() interferes w debug printouts */
        if (OP(s) != O_QUEST && OP(s) != O_CH &&
            OP(s) != OOR2) {
            g->iflags |= BAD;
            return;
        }
    } while (OP(s) != O_QUEST && OP(s) != O_CH);
    /* fallthrough */
default: /* things that break a sequence */
    if (newlen > g->mlen) { /* ends one */
        start = newstart;
        g->mlen = newlen;
    }
    newlen = 0;
    break;
}
} while (OP(s) != OEND);

if (g->mlen == 0) /* there isn't one */
    return;

/* turn it into a character string */
g->must = malloc((size_t)g->mlen + 1);
if (g->must == NULL) { /* argh; just forget it */
    g->mlen = 0;
    return;
}
cp = g->must;
scan = start;
for (i = g->mlen; i > 0; i--) {
    while (OP(s = *scan++) != OCHAR)
        continue;
    assert(cp < g->must + g->mlen);
    *cp++ = (char)OPND(s);
}
assert(cp == g->must + g->mlen);
*cp++ = '\0'; /* just on general principles */
}

/*
- pluscount - count + nesting
== static sopno pluscount(register struct parse *p, register struct re_guts *g);
*/
static sopno /* nesting depth */
pluscount(p, g)
struct parse *p;
register struct re_guts *g;
{
    register sop *scan;
    register sop s;
    register sopno plusnest = 0;
    register sopno maxnest = 0;

    if (p->error != 0)
        return(0); /* there may not be an OEND */

    scan = g->strip + 1;
    do {
        s = *scan++;
        switch (OP(s)) {
            case OPLUS_:
                plusnest++;
                break;
            case O_PLUS:
                if (plusnest > maxnest)
                    maxnest = plusnest;
                plusnest--;
                break;
        }
    } while (OP(s) != OEND);
    if (plusnest != 0)
        g->iflags |= BAD;
    return(maxnest);
}

```

```
/*  
 * $PchId: regcomp.c,v 1.2 1996/03/12 19:10:15 philip Exp $  
 */
```

```

/*-
 * Copyright (c) 1992, 1993, 1994 Henry Spencer.
 * Copyright (c) 1992, 1993, 1994
 *     The Regents of the University of California.  All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Henry Spencer.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *        This product includes software developed by the University of
 *        California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)regerror.c  8.4 (Berkeley) 3/20/94
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)regerror.c 8.4 (Berkeley) 3/20/94";
#endif /* LIBC_SCCS and not lint */

#include <sys/types.h>
#include <stdio.h>
#include <string.h>
#ifdef __minix_vmd
#include <bsd/asciictype.h>
#else
#include <ctype.h>
#endif
#include <limits.h>
#include <stdlib.h>
#include <regex.h>

#include "utils.h"

/* ===== begin header generated by ./mkh ===== */
#ifdef __cplusplus
extern "C" {
#endif

/* == regerror.c == */
static char *regatoi(const regex_t *preg, char *localbuf);

#ifdef __cplusplus
}
#endif
/* ===== end header generated by ./mkh ===== */
/*
 = #define      REG_NOMATCH      1
 = #define      REG_BADPAT      2
 = #define      REG_ECOLLATE    3
 = #define      REG_ECTYPE     4

```

```

= #define      REG_EESCAPE      5
= #define      REG_ESUBREG      6
= #define      REG_EBRACK      7
= #define      REG_EPAREN      8
= #define      REG_EBRACE      9
= #define      REG_BADBR      10
= #define      REG_ERANGE      11
= #define      REG_ESPACE      12
= #define      REG_BADRPT      13
= #define      REG_EMPTY      14
= #define      REG_ASSERT      15
= #define      REG_INVARG      16
= #define      REG_ATOI      255      // convert name to number (!)
= #define      REG_ITOA      0400      // convert number to name (!)
*/
static struct rerr {
    int code;
    char *name;
    char *explain;
} rerrs[] = {
    REG_NOMATCH,      "REG_NOMATCH", "regexec() failed to match",
    REG_BADPAT,      "REG_BADPAT", "invalid regular expression",
    REG_ECOLLATE,      "REG_ECOLLATE", "invalid collating element",
    REG_ECTYPE,      "REG_ECTYPE", "invalid character class",
    REG_EESCAPE,      "REG_EESCAPE", "trailing backslash (\\)",
    REG_ESUBREG,      "REG_ESUBREG", "invalid backreference number",
    REG_EBRACK,      "REG_EBRACK", "brackets ( [ ] ) not balanced",
    REG_EPAREN,      "REG_EPAREN", "parentheses not balanced",
    REG_EBRACE,      "REG_EBRACE", "braces not balanced",
    REG_BADBR,      "REG_BADBR", "invalid repetition count(s)",
    REG_ERANGE,      "REG_ERANGE", "invalid character range",
    REG_ESPACE,      "REG_ESPACE", "out of memory",
    REG_BADRPT,      "REG_BADRPT", "repetition-operator operand invalid",
    REG_EMPTY,      "REG_EMPTY", "empty (sub)expression",
    REG_ASSERT,      "REG_ASSERT", "\\\"can't happen\\\" -- you found a bug",
    REG_INVARG,      "REG_INVARG", "invalid argument to regex routine",
    0,      "",      "*** unknown regexp error code ***",
};

/*
- regerror - the interface to error numbers
= extern size_t regerror(int, const regex_t *, char *, size_t);
*/
/* ARGSUSED */
size_t
regerror(errcode, preg, errbuf, errbuf_size)
int errcode;
const regex_t *preg;
char *errbuf;
size_t errbuf_size;
{
    register struct rerr *r;
    register size_t len;
    register int target = errcode &~ REG_ITOA;
    register char *s;
    char convbuf[50];

    if (errcode == REG_ATOI)
        s = regatoi(preg, convbuf);
    else {
        for (r = rerrs; r->code != 0; r++)
            if (r->code == target)
                break;

        if (errcode & REG_ITOA) {
            if (r->code != 0)
                (void) strcpy(convbuf, r->name);
            else
                sprintf(convbuf, "REG_0x%x", target);
            assert(strlen(convbuf) < sizeof(convbuf));
            s = convbuf;
        } else
            s = r->explain;
    }
}

```

```
    len = strlen(s) + 1;
    if (errbuf_size > 0) {
        if (errbuf_size > len)
            (void) strcpy(errbuf, s);
        else {
            (void) strncpy(errbuf, s, errbuf_size-1);
            errbuf[errbuf_size-1] = '\0';
        }
    }

    return(len);
}

/*
 * - regatoi - internal routine to implement REG_ATOI
 * == static char *regatoi(const regex_t *preg, char *localbuf);
 */
static char *
regatoi(preg, localbuf)
const regex_t *preg;
char *localbuf;
{
    register struct rerr *r;
    register size_t siz;
    register char *p;

    for (r = rerrs; r->code != 0; r++)
        if (strcmp(r->name, preg->re_endp) == 0)
            break;
    if (r->code == 0)
        return("0");

    sprintf(localbuf, "%d", r->code);
    return(localbuf);
}

/*
 * $PchId: regerror.c,v 1.2 1996/03/12 19:10:15 philip Exp $
 */
```

NAME

regex, regcomp, regexec, regerror, regfree – regular-expression library

SYNOPSIS

```
#include <sys/types.h>
#include <regex.h>
```

```
int regcomp(regex_t *preg, const char *pattern, int cflags);
int regexec(const regex_t *preg, const char *string, size_t nmatch, regmatch_t pmatch[], int eflags);
size_t regerror(int errcode, const regex_t *preg, char *errbuf, size_t errbuf_size);
void regfree(regex_t *preg);
```

DESCRIPTION

These routines implement POSIX 1003.2 regular expressions (“RE”s); see **re_format(7)**. **Regcomp** compiles an RE written as a string into an internal form, **regexec** matches that internal form against a string and reports results, **regerror** transforms error codes from either into human-readable messages, and **regfree** frees any dynamically-allocated storage used by the internal form of an RE.

The header `<regex.h>` declares two structure types, **regex_t** and **regmatch_t**, the former for compiled internal forms and the latter for match reporting. It also declares the four functions, a type **regoff_t**, and a number of constants with names starting with “REG_”.

Regcomp compiles the regular expression contained in the *pattern* string, subject to the flags in *cflags*, and places the results in the **regex_t** structure pointed to by *preg*. *Cflags* is the bitwise OR of zero or more of the following flags:

REG_EXTENDED	Compile modern (“extended”) REs, rather than the obsolete (“basic”) REs that are the default.
REG_BASIC	This is a synonym for 0, provided as a counterpart to REG_EXTENDED to improve readability.
REG_NOSPEC	Compile with recognition of all special characters turned off. All characters are thus considered ordinary, so the “RE” is a literal string. This is an extension, compatible with but not specified by POSIX 1003.2, and should be used with caution in software intended to be portable to other systems. REG_EXTENDED and REG_NOSPEC may not be used in the same call to <i>regcomp</i> .
REG_ICASE	Compile for matching that ignores upper/lower case distinctions. See re_format(7) .
REG_NOSUB	Compile for matching that need only report success or failure, not what was matched.
REG_NEWLINE	Compile for newline-sensitive matching. By default, newline is a completely ordinary character with no special meaning in either REs or strings. With this flag, “[^” bracket expressions and ‘.’ never match newline, a ‘^’ anchor matches the null string after any newline in the string in addition to its normal function, and the ‘\$’ anchor matches the null string before any newline in the string in addition to its normal function.
REG PEND	The regular expression ends, not at the first NUL, but just before the character pointed to by the re_endp member of the structure pointed to by <i>preg</i> . The re_endp member is of type const char * . This flag permits inclusion of NULs in the RE; they are considered ordinary characters. This is an extension, compatible with but not specified by POSIX 1003.2, and should be used with caution in software intended to be portable to other systems.

When successful, **regcomp** returns 0 and fills in the structure pointed to by *preg*. One member of that structure (other than **re_endp**) is publicized: **re_nsub**, of type **size_t**, contains the number of parenthesized subexpressions within the RE (except that the value of this member is undefined if the REG_NOSUB flag was used). If **regcomp** fails, it returns a non-zero error code; see **DIAGNOSTICS**.

Regexec matches the compiled RE pointed to by *preg* against the *string*, subject to the flags in *eflags*, and

reports results using *nmatch*, *pmatch*, and the returned value. The RE must have been compiled by a previous invocation of **regcomp**. The compiled form is not altered during execution of **regexexec**, so a single compiled RE can be used simultaneously by multiple threads.

By default, the NUL-terminated string pointed to by *string* is considered to be the text of an entire line, minus any terminating newline. The *eflags* argument is the bitwise OR of zero or more of the following flags:

REG_NOTBOL	The first character of the string is not the beginning of a line, so the ‘^’ anchor should not match before it. This does not affect the behavior of newlines under REG_NEWLINE.
REG_NOTEOL	The NUL terminating the string does not end a line, so the ‘\$’ anchor should not match before it. This does not affect the behavior of newlines under REG_NEWLINE.
REG_STARTEND	The string is considered to start at <i>string</i> + <i>pmatch</i> [0]. rm_so and to have a terminating NUL located at <i>string</i> + <i>pmatch</i> [0]. rm_eo (there need not actually be a NUL at that location), regardless of the value of <i>nmatch</i> . See below for the definition of <i>pmatch</i> and <i>nmatch</i> . This is an extension, compatible with but not specified by POSIX 1003.2, and should be used with caution in software intended to be portable to other systems. Note that a non-zero rm_so does not imply REG_NOTBOL; REG_STARTEND affects only the location of the string, not how it is matched.

See **re_format**(7) for a discussion of what is matched in situations where an RE or a portion thereof could match any of several substrings of *string*.

Normally, **regexexec** returns 0 for success and the non-zero code REG_NOMATCH for failure. Other non-zero error codes may be returned in exceptional situations; see DIAGNOSTICS.

If REG_NOSUB was specified in the compilation of the RE, or if *nmatch* is 0, **regexexec** ignores the *pmatch* argument (but see below for the case where REG_STARTEND is specified). Otherwise, *pmatch* points to an array of *nmatch* structures of type **regmatch_t**. Such a structure has at least the members **rm_so** and **rm_eo**, both of type **regoff_t** (a signed arithmetic type at least as large as an **off_t** and a **ssize_t**), containing respectively the offset of the first character of a substring and the offset of the first character after the end of the substring. Offsets are measured from the beginning of the *string* argument given to **regexexec**. An empty substring is denoted by equal offsets, both indicating the character following the empty substring.

The 0th member of the *pmatch* array is filled in to indicate what substring of *string* was matched by the entire RE. Remaining members report what substring was matched by parenthesized subexpressions within the RE; member *i* reports subexpression *i*, with subexpressions counted (starting at 1) by the order of their opening parentheses in the RE, left to right. Unused entries in the array—corresponding either to subexpressions that did not participate in the match at all, or to subexpressions that do not exist in the RE (that is, *i* > *preg*→**re_nsub**)—have both **rm_so** and **rm_eo** set to -1. If a subexpression participated in the match several times, the reported substring is the last one it matched. (Note, as an example in particular, that when the RE ‘(b*)+’ matches ‘bbb’, the parenthesized subexpression matches each of the three ‘b’s and then an infinite number of empty strings following the last ‘b’, so the reported substring is one of the empties.)

If REG_STARTEND is specified, *pmatch* must point to at least one **regmatch_t** (even if *nmatch* is 0 or REG_NOSUB was specified), to hold the input offsets for REG_STARTEND. Use for output is still entirely controlled by *nmatch*; if *nmatch* is 0 or REG_NOSUB was specified, the value of *pmatch*[0] will not be changed by a successful **regexexec**.

Regerror maps a non-zero *errcode* from either **regcomp** or **regexexec** to a human-readable, printable message. If *preg* is non-NULL, the error code should have arisen from use of the **regex_t** pointed to by *preg*, and if the error code came from **regcomp**, it should have been the result from the most recent **regcomp** using that **regex_t**. (*Regerror* may be able to supply a more detailed message using information from the **regex_t**.) **Regerror** places the NUL-terminated message into the buffer pointed to by *errbuf*, limiting the length (including the NUL) to at most *errbuf_size* bytes. If the whole message won’t fit, as much of it as

will fit before the terminating NUL is supplied. In any case, the returned value is the size of buffer needed to hold the whole message (including terminating NUL). If *errbuf_size* is 0, *errbuf* is ignored but the return value is still correct.

If the *errcode* given to **regerror** is first ORed with REG_ITOA, the “message” that results is the printable name of the error code, e.g. “REG_NOMATCH”, rather than an explanation thereof. If *errcode* is REG_ATOI, then *preg* shall be non-NULL and the **re_endp** member of the structure it points to must point to the printable name of an error code; in this case, the result in *errbuf* is the decimal digits of the numeric value of the error code (0 if the name is not recognized). REG_ITOA and REG_ATOI are intended primarily as debugging facilities; they are extensions, compatible with but not specified by POSIX 1003.2, and should be used with caution in software intended to be portable to other systems. Be warned also that they are considered experimental and changes are possible.

Regfree frees any dynamically-allocated storage associated with the compiled RE pointed to by *preg*. The remaining **regex_t** is no longer a valid compiled RE and the effect of supplying it to **regex** or **regerror** is undefined.

None of these functions references global variables except for tables of constants; all are safe for use from multiple threads if the arguments are safe.

IMPLEMENTATION CHOICES

There are a number of decisions that 1003.2 leaves up to the implementor, either by explicitly saying “undefined” or by virtue of them being forbidden by the RE grammar. This implementation treats them as follows.

See **re_format(7)** for a discussion of the definition of case-independent matching.

There is no particular limit on the length of REs, except insofar as memory is limited. Memory usage is approximately linear in RE size, and largely insensitive to RE complexity, except for bounded repetitions. See BUGS for one short RE using them that will run almost any system out of memory.

A backslashed character other than one specifically given a magic meaning by 1003.2 (such magic meanings occur only in obsolete [“basic”] REs) is taken as an ordinary character.

Any unmatched [is a REG_EBRACK error.

Equivalence classes cannot begin or end bracket-expression ranges. The endpoint of one range cannot begin another.

RE_DUP_MAX, the limit on repetition counts in bounded repetitions, is 255.

A repetition operator (?, *, +, or bounds) cannot follow another repetition operator. A repetition operator cannot begin an expression or subexpression or follow “^” or “|”.

“|” cannot appear first or last in a (sub)expression or after another “|”, i.e. an operand of “|” cannot be an empty subexpression. An empty parenthesized subexpression, “()”, is legal and matches an empty (sub)string. An empty string is not a legal RE.

A “{” followed by a digit is considered the beginning of bounds for a bounded repetition, which must then follow the syntax for bounds. A “{” *not* followed by a digit is considered an ordinary character.

“^” and “\$” beginning and ending subexpressions in obsolete (“basic”) REs are anchors, not ordinary characters.

SEE ALSO

grep(1), **re_format(7)**.

POSIX 1003.2, sections 2.8 (Regular Expression Notation) and B.5 (C Binding for Regular Expression Matching).

DIAGNOSTICS

Non-zero error codes from **regcomp** and **regex** include the following:

REG_NOMATCH	regex() failed to match
REG_BADPAT	invalid regular expression

REG_ECOLLATE	invalid collating element
REG_ECTYPE	invalid character class
REG_EESCAPE	\ applied to unescapable character
REG_ESUBREG	invalid backreference number
REG_EBRACK	brackets [] not balanced
REG_EPAREN	parentheses () not balanced
REG_EBRACE	braces { } not balanced
REG_BADBR	invalid repetition count(s) in { }
REG_ERANGE	invalid character range in []
REG_ESPACE	ran out of memory
REG_BADRPT	?, *, or + operand invalid
REG_EMPTY	empty (sub)expression
REG_ASSERT	“can’t happen”—you found a bug
REG_INVARG	invalid argument, e.g. negative-length string

HISTORY

Originally written by Henry Spencer. Altered for inclusion in the 4.4BSD distribution.

BUGS

This is an alpha release with known defects. Please report problems.

There is one known functionality bug. The implementation of internationalization is incomplete: the locale is always assumed to be the default one of 1003.2, and only the collating elements etc. of that locale are available.

The back-reference code is subtle and doubts linger about its correctness in complex cases.

Regex performance is poor. This will improve with later releases. *Nmatch* exceeding 0 is expensive; *nmatch* exceeding 1 is worse. **Regex** is largely insensitive to RE complexity *except* that back references are massively expensive. RE length does matter; in particular, there is a strong speed bonus for keeping RE length under about 30 characters, with most special characters counting roughly double.

Regcomp implements bounded repetitions by macro expansion, which is costly in time and space if counts are large or bounded repetitions are nested. An RE like, say, ‘((((a{1,100}){1,100}){1,100}){1,100}){1,100}’ will (eventually) run almost any existing machine out of swap space.

There are suspected problems with response to obscure error conditions. Notably, certain kinds of internal overflow, produced only by truly enormous REs or by multiply nested bounded repetitions, are probably not handled well.

Due to a mistake in 1003.2, things like ‘a)b’ are legal REs because ‘)’ is a special character only in the presence of a previous unmatched ‘(’. This can’t be fixed until the spec is fixed.

The standard’s definition of back references is vague. For example, does ‘a\\(b\\)*\\2\\)*d’ match ‘abbbd’? Until the standard is clarified, behavior in such cases should not be relied on.

The implementation of word-boundary matching is a bit of a kludge, and bugs may lurk in combinations of word-boundary matching and anchoring.

```

/*-
 * Copyright (c) 1992, 1993, 1994 Henry Spencer.
 * Copyright (c) 1992, 1993, 1994
 *     The Regents of the University of California.  All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Henry Spencer.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *        This product includes software developed by the University of
 *        California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)regex2.h      8.4 (Berkeley) 3/20/94
 */

/*
 * First, the stuff that ends up in the outside-world include file
 */
typedef off_t regoff_t;
typedef struct {
    int re_magic;
    size_t re_nsub;           // number of parenthesized subexpressions
    const char *re_endp;      // end pointer for REG_PEND
    struct re_guts *re_g;     // none of your business :-)
} regex_t;
typedef struct {
    regoff_t rm_so;           // start of match
    regoff_t rm_eo;           // end of match
} regmatch_t;
/*
 * internals of regex_t
 */
#define MAGIC1 (((('r'^0200)<<8) | 'e'))

/*
 * The internal representation is a *strip*, a sequence of
 * operators ending with an endmarker. (Some terminology etc. is a
 * historical relic of earlier versions which used multiple strips.)
 * Certain oddities in the representation are there to permit running
 * the machinery backwards; in particular, any deviation from sequential
 * flow must be marked at both its source and its destination. Some
 * fine points:
 *
 * - OPLUS_ and O_PLUS are *inside* the loop they create.
 * - OQUEST_ and O_QUESTION are *outside* the bypass they create.
 * - OCH_ and O_CH are *outside* the multi-way branch they create, while
 *   OOR1 and OOR2 are respectively the end and the beginning of one of
 *   the branches. Note that there is an implicit OOR2 following OCH_
 *   and an implicit OOR1 preceding O_CH.
 */

```

```

* In state representations, an operator's bit is on to signify a state
* immediately *preceding* "execution" of that operator.
*/
typedef unsigned long sop;      /* strip operator */
typedef long sopno;
#define OPRMASK 0xf8000000
#define OPDMASK 0x07ffffff
#define OPSHIFT ((unsigned)27)
#define OP(n) ((n)&OPRMASK)
#define OPND(n) ((n)&OPDMASK)
#define SOP(op, opnd) ((op)|(opnd))
/* operators meaning operand */
/* (back, fwd are offsets) */
#define OEND (1<<OPSHIFT) /* endmarker - */
#define OCHAR (2<<OPSHIFT) /* character unsigned char */
#define OBOL (3<<OPSHIFT) /* left anchor - */
#define OEOL (4<<OPSHIFT) /* right anchor - */
#define OANY (5<<OPSHIFT) /* . - */
#define OANYOF (6<<OPSHIFT) /* [...] set number */
#define OBACK_ (7<<OPSHIFT) /* begin \d paren number */
#define O_BACK (8<<OPSHIFT) /* end \d paren number */
#define OPLUS_ (9<<OPSHIFT) /* + prefix fwd to suffix */
#define O_PLUS (10<<OPSHIFT) /* + suffix back to prefix */
#define OQUEST_ (11<<OPSHIFT) /* ? prefix fwd to suffix */
#define O_QUESTION (12<<OPSHIFT) /* ? suffix back to prefix */
#define OLPAREN (13<<OPSHIFT) /* ( fwd to ) */
#define ORPAREN (14<<OPSHIFT) /* ) back to ( */
#define OCH_ (15<<OPSHIFT) /* begin choice fwd to OOR2 */
#define OOR1 (16<<OPSHIFT) /* | pt. 1 back to OOR1 or OCH_ */
#define OOR2 (17<<OPSHIFT) /* | pt. 2 fwd to OOR2 or O_CH */
#define O_CH (18<<OPSHIFT) /* end choice back to OOR1 */
#define OBOW (19<<OPSHIFT) /* begin word - */
#define OEOW (20<<OPSHIFT) /* end word - */

/*
* Structure for [] character-set representation. Character sets are
* done as bit vectors, grouped 8 to a byte vector for compactness.
* The individual set therefore has both a pointer to the byte vector
* and a mask to pick out the relevant bit of each byte. A hash code
* simplifies testing whether two sets could be identical.
*
* This will get trickier for multicharacter collating elements. As
* preliminary hooks for dealing with such things, we also carry along
* a string of multi-character elements, and decide the size of the
* vectors at run time.
*/
typedef struct {
    uch *ptr; /* -> uch [csetsize] */
    uch mask; /* bit within array */
    uch hash; /* hash code */
    size_t smultis;
    char *multis; /* -> char[smulti] ab\0cd\0ef\0\0 */
} cset;
/* note that CHadd and CHsub are unsafe, and CHIN doesn't yield 0/1 */
#define CHadd(cs, c) ((cs)->ptr[(uch)(c)] |= (cs)->mask, (cs)->hash += (c))
#define CHsub(cs, c) ((cs)->ptr[(uch)(c)] &= ~(cs)->mask, (cs)->hash -= (c))
#define CHIN(cs, c) ((cs)->ptr[(uch)(c)] & (cs)->mask)
#define MCadd(p, cs, cp) mcadd(p, cs, cp) /* regcomp() internal fns */
#define MCSub(p, cs, cp) mcsb(p, cs, cp)
#define MCin(p, cs, cp) mcin(p, cs, cp)

/* stuff for character categories */
typedef unsigned char cat_t;

/*
* main compiled-expression structure
*/
struct re_guts {
    int magic;
#define MAGIC2 (((('R'^0200)<<8)|'E'))
    sop *strip; /* malloced area for strip */
    int csetsize; /* number of bits in a cset vector */
    int ncsets; /* number of csets in use */
    cset *sets; /* -> cset [ncsets] */

```

```
    uch *setbits;          /* -> uch[csetsize][ncsets/CHAR_BIT] */
    int cflags;             /* copy of regcomp() cflags argument */
    sopno nstates;          /* = number of sops */
    sopno firststate;        /* the initial OEND (normally 0) */
    sopno laststate;        /* the final OEND */
    int iflags;             /* internal flags */
#       define USEBOL 01    /* used ^ */
#       define USEEOL 02    /* used $ */
#       define BAD 04       /* something wrong */
    int nbol;               /* number of ^ used */
    int neol;               /* number of $ used */
    int ncategories;        /* how many character categories */
    cat_t *categories;      /* -> catspace[-CHAR_MIN] */
    char *must;             /* match must contain this string */
    int mlen;               /* length of must */
    size_t nsub;            /* copy of re_nsub */
    int backrefs;           /* does it use back references? */
    sopno nplus;            /* how deep does it nest +s? */
    /* catspace must be last */
    cat_t catspace[1];      /* actually [NC] */
};

/* misc utilities */
#define OUT (CHAR_MAX+1)    /* a non-character value */
#define ISWORD(c) (isalnum(c) || (c) == '_')
```

```

/*-
 * Copyright (c) 1992, 1993, 1994 Henry Spencer.
 * Copyright (c) 1992, 1993, 1994
 *   The Regents of the University of California.  All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Henry Spencer.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *   must display the following acknowledgement:
 *       This product includes software developed by the University of
 *       California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *   may be used to endorse or promote products derived from this software
 *   without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *   @(#)regexec.c    8.3 (Berkeley) 3/20/94
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)regexec.c    8.3 (Berkeley) 3/20/94";
#endif /* LIBC_SCCS and not lint */

/*
 * the outer shell of regexec()
 */
/* This file includes engine.c *twice*, after muchos fiddling with the
 * macros that code uses.  This lets the same code operate on two different
 * representations for state sets.
 */
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#ifdef __minix_vmd
#include <bsd/asciictype.h>
#else
#include <ctype.h>
#endif
#include <regex.h>

#include "utils.h"
#include "regex2.h"

static int nope = 0;          /* for use in asserts; shuts lint up */

/* macros for manipulating states, small version */
#define states long
#define states1 states        /* for later use in regexec() decision */
#define CLEAR(v)              ((v) = 0)
#define SET0(v, n)             ((v) &= ~(1 << (n)))
#define SET1(v, n)             ((v) |= 1 << (n))
#define ISSET(v, n)            ((v) & (1 << (n)))

```

```

#define ASSIGN(d, s)      ((d) = (s))
#define EQ(a, b)          ((a) == (b))
#define STATEVARS        int dummy /* dummy version */
#define STATESETUP(m, n) /* nothing */
#define STATETEARDOWN(m) /* nothing */
#define SETUP(v)          ((v) = 0)
#define onestate          int
#define INIT(o, n)        ((o) = (unsigned)1 << (n))
#define INC(o)             ((o) <= 1)
#define ISSTATEIN(v, o)   ((v) & (o))
/* some abbreviations; note that some of these know variable names! */
/* do "if I'm here, I can also be there" etc without branches */
#define FWD(dst, src, n)  ((dst) |= ((unsigned)(src)&(here)) << (n))
#define BACK(dst, src, n) ((dst) |= ((unsigned)(src)&(here)) >> (n))
#define ISSETBACK(v, n)   ((v) & ((unsigned)here >> (n)))
/* function names */
#define SNAMES             /* engine.c looks after details */

#include "engine.c"

/* now undo things */
#undef states
#undef CLEAR
#undef SET0
#undef SET1
#undef ISSET
#undef ASSIGN
#undef EQ
#undef STATEVARS
#undef STATESETUP
#undef STATETEARDOWN
#undef SETUP
#undef onestate
#undef INIT
#undef INC
#undef ISSTATEIN
#undef FWD
#undef BACK
#undef ISSETBACK
#undef SNAMES

/* macros for manipulating states, large version */
#define states char *
#define CLEAR(v)          memset(v, 0, m->g->nstates)
#define SET0(v, n)         ((v)[n] = 0)
#define SET1(v, n)         ((v)[n] = 1)
#define ISSET(v, n)        ((v)[n])
#define ASSIGN(d, s)       memcpy(d, s, m->g->nstates)
#define EQ(a, b)           (memcmp(a, b, m->g->nstates) == 0)
#define STATEVARS          int vn; char *space
#define STATESETUP(m, nv)  { (m)->space = malloc((nv)*(m)->g->nstates); \
                           if ((m)->space == NULL) return(REG_ESPACE); \
                           (m)->vn = 0; }
#define STATETEARDOWN(m)   { free((m)->space); }
#define SETUP(v)           ((v) = &m->space[m->vn++ * m->g->nstates])
#define onestate           int
#define INIT(o, n)         ((o) = (n))
#define INC(o)              ((o)++)
#define ISSTATEIN(v, o)    ((v)[o])
/* some abbreviations; note that some of these know variable names! */
/* do "if I'm here, I can also be there" etc without branches */
#define FWD(dst, src, n)    ((dst)[here+(n)] |= (src)[here])
#define BACK(dst, src, n)  ((dst)[here-(n)] |= (src)[here])
#define ISSETBACK(v, n)    ((v)[here - (n)])
/* function names */
#define LNAMES             /* flag */

#include "engine.c"

/*
- regexec - interface for matching
= extern int regexec(const regex_t *, const char *, size_t, \
=                               regmatch_t [], int);
= #define      REG_NOTBOL      00001

```

```
= #define      REG_NOTEOL      00002
= #define      REG_STARTEND    00004
= #define      REG_TRACE      00400    // tracing of execution
= #define      REG_LARGE      01000    // force large representation
= #define      REG_BACKR      02000    // force use of backref code
*
* We put this here so we can exploit knowledge of the state representation
* when choosing which matcher to call. Also, by this point the matchers
* have been prototyped.
*/
int                                /* 0 success, REG_NOMATCH failure */
regexec(preg, string, nmatch, pmatch, eflags)
const regex_t *preg;
const char *string;
size_t nmatch;
regmatch_t pmatch[];
int eflags;
{
    register struct re_guts *g = preg->re_g;
#ifdef REDEBUG
#   define GOODFLAGS(f)      (f)
#else
#   define GOODFLAGS(f)      ((f)&(REG_NOTBOL|REG_NOTEOL|REG_STARTEND))
#endif
    if (preg->re_magic != MAGIC1 || g->magic != MAGIC2)
        return(REG_BADPAT);
    assert(!(g->iflags&BAD));
    if (g->iflags&BAD)                /* backstop for no-debug case */
        return(REG_BADPAT);
    eflags = GOODFLAGS(eflags);

    if (g->nstates <= CHAR_BIT*sizeof(states1) && !(eflags&REG_LARGE))
        return(smatcher(g, (char *)string, nmatch, pmatch, eflags));
    else
        return(lmatcher(g, (char *)string, nmatch, pmatch, eflags));
}

/*
 * $PchId: regexec.c,v 1.2 1996/03/12 19:10:15 philip Exp $
 */
```

```

/*-
 * Copyright (c) 1992, 1993, 1994 Henry Spencer.
 * Copyright (c) 1992, 1993, 1994
 *   The Regents of the University of California.  All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Henry Spencer.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *   must display the following acknowledgement:
 *       This product includes software developed by the University of
 *       California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *   may be used to endorse or promote products derived from this software
 *   without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *   @(#)regfree.c    8.3 (Berkeley) 3/20/94
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)regfree.c    8.3 (Berkeley) 3/20/94";
#endif /* LIBC_SCCS and not lint */

#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <regex.h>

#include "utils.h"
#include "regex2.h"

/*
 * - regfree - free everything
 * = extern void regfree(regex_t *);
 */
void
regfree(preg)
    regex_t *preg;
{
    register struct re_guts *g;

    if (preg->re_magic != MAGIC1) /* oops */
        return; /* nice to complain, but hard */

    g = preg->re_g;
    if (g == NULL || g->magic != MAGIC2) /* oops again */
        return;
    preg->re_magic = 0; /* mark it invalid */
    g->magic = 0; /* mark it invalid */

    if (g->strip != NULL)
        free((char *)g->strip);
    if (g->sets != NULL)
        free((char *)g->sets);
}

```



```
    if (g->setbits != NULL)
        free((char *)g->setbits);
    if (g->must != NULL)
        free(g->must);
    free((char *)g);
}

/*
 * $PchId: regfree.c,v 1.2 1996/03/12 19:10:15 philip Exp $
 */
```

```
/*-
 * Copyright (c) 1992, 1993, 1994 Henry Spencer.
 * Copyright (c) 1992, 1993, 1994
 *   The Regents of the University of California. All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Henry Spencer.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *   must display the following acknowledgement:
 *       This product includes software developed by the University of
 *       California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *   may be used to endorse or promote products derived from this software
 *   without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)utils.h      8.3 (Berkeley) 3/20/94
 */

/* utility definitions */
#ifdef _POSIX2_RE_DUP_MAX
#define DUPMAX      _POSIX2_RE_DUP_MAX      /* xxx is this right? */
#else
#define DUPMAX      255
#endif
#define INFINITY      (DUPMAX + 1)
#define NC            (CHAR_MAX - CHAR_MIN + 1)
typedef unsigned char uch;

/* switch off assertions (if not already off) if no REDEBUG */
#ifndef REDEBUG
#ifndef NDEBUG
#define NDEBUG      /* no assertions please */
#endif
#endif
#include <assert.h>

/* for old systems with bcopy() but no memmove() */
#ifdef USEBCOPY
#define memmove(d, s, c)      bcopy(s, d, c)
#endif
```

```
# Makefile for lib/stdio.
```

```
CFLAGS="-O -D_MINIX -D_POSIX_SOURCE"
```

```
LIBRARIES=libc
```

```
libc_FILES=" \  
    clearerr.c \  
    data.c \  
    doprnt.c \  
    doscan.c \  
    ecvt.c \  
    fclose.c \  
    feof.c \  
    ferror.c \  
    fflush.c \  
    fgetc.c \  
    fgetpos.c \  
    fgets.c \  
    fileno.c \  
    fillbuf.c \  
    flushbuf.c \  
    fopen.c \  
    fprintf.c \  
    fputc.c \  
    fputs.c \  
    fread.c \  
    freopen.c \  
    fscanf.c \  
    fseek.c \  
    fsetpos.c \  
    ftell.c \  
    fwrite.c \  
    getc.c \  
    getchar.c \  
    gets.c \  
    icompute.c \  
    mktemp.c \  
    perror.c \  
    printf.c \  
    putc.c \  
    putchar.c \  
    puts.c \  
    remove.c \  
    rewind.c \  
    scanf.c \  
    setbuf.c \  
    setvbuf.c \  
    sprintf.c \  
    sscanf.c \  
    tmpfile.c \  
    tmpnam.c \  
    ungetc.c \  
    vfprintf.c \  
    vprintf.c \  
    vscanf.c \  
    vsprintf.c \  
    vsscanf.c"
```

```
TYPE=both
```

```
/*
 * clearerr.c - clear error and end-of-file indicators of a stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/clearerr.c,v 1.1.1.1 2005/04/21 14:56:3
4 beng Exp $ */

#include <stdio.h>

void
(clearerr)(FILE *stream)
{
    clearerr(stream);
}
```

```
/*
 * data.c - this is the initialization for the standard streams
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/data.c,v 1.1.1.1 2005/04/21 14:56:34 be
ng Exp $ */

#include <stdio.h>

struct __iobuf __stdin = {
    0, 0, _IOREAD, 0,
    (unsigned char *)NULL, (unsigned char *)NULL,
};

struct __iobuf __stdout = {
    0, 1, _IOWRITE, 0,
    (unsigned char *)NULL, (unsigned char *)NULL,
};

struct __iobuf __stderr = {
    0, 2, _IOWRITE | _IOLBF, 0,
    (unsigned char *)NULL, (unsigned char *)NULL,
};

FILE *__iotab[FOPEN_MAX] = {
    &__stdin,
    &__stdout,
    &__stderr,
    0
};
```

```
/*
 * doprnt.c - print formatted output
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/doprnt.c,v 1.1.1.1 2005/04/21 14:56:34
beng Exp $ */

#include <ctype.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include "loc_incl.h"

/* gnum() is used to get the width and precision fields of a format. */
static const char *
gnum(register const char *f, int *ip, va_list *app)
{
    register int i, c;

    if (*f == '*') {
        *ip = va_arg((*app), int);
        f++;
    } else {
        i = 0;
        while ((c = *f - '0') >= 0 && c <= 9) {
            i = i*10 + c;
            f++;
        }
        *ip = i;
    }
    return f;
}

#if _EM_WSIZE == _EM_PSIZE
#define set_pointer(flags) /* nothing */
#elif _EM_LSIZE == _EM_PSIZE
#define set_pointer(flags) (flags |= FL_LONG)
#else
#error garbage pointer size
#define set_pointer(flags) /* compilation might continue */
#endif

/* print an ordinal number */
static char *
o_print(va_list *ap, int flags, char *s, char c, int precision, int is_signed)
{
    long signed_val;
    unsigned long unsigned_val;
    char *old_s = s;
    int base;

    switch (flags & (FL_SHORT | FL_LONG)) {
    case FL_SHORT:
        if (is_signed) {
            signed_val = (short) va_arg(*ap, int);
        } else {
            unsigned_val = (unsigned short) va_arg(*ap, unsigned);
        }
        break;
    case FL_LONG:
        if (is_signed) {
            signed_val = va_arg(*ap, long);
        } else {
            unsigned_val = va_arg(*ap, unsigned long);
        }
        break;
    default:
        if (is_signed) {
            signed_val = va_arg(*ap, int);
        } else {
            unsigned_val = va_arg(*ap, unsigned int);
        }
        break;
    }
}
```

```

    if (is_signed) {
        if (signed_val < 0) {
            *s++ = '-';
            signed_val = -signed_val;
        } else if (flags & FL_SIGN) *s++ = '+';
        else if (flags & FL_SPACE) *s++ = ' ';
        unsigned_val = signed_val;
    }
    if ((flags & FL_ALT) && (c == 'o')) *s++ = '0';
    if (!unsigned_val) {
        if (!precision)
            return s;
    } else if (((flags & FL_ALT) && (c == 'x' || c == 'X'))
        || c == 'p') {
        *s++ = '0';
        *s++ = (c == 'X' ? 'X' : 'x');
    }

    switch (c) {
    case 'b':      base = 2;      break;
    case 'o':      base = 8;      break;
    case 'd':
    case 'i':
    case 'u':      base = 10;     break;
    case 'x':
    case 'X':
    case 'p':      base = 16;     break;
    }

    s = _i_compute(unsigned_val, base, s, precision);

    if (c == 'X')
        while (old_s != s) {
            *old_s = toupper(*old_s);
            old_s++;
        }

    return s;
}

int
_doprnt(register const char *fmt, va_list ap, FILE *stream)
{
    register char    *s;
    register int     j;
    int              i, c, width, precision, zfill, flags, between_fill;
    int              nrchars=0;
    const char       *oldfmt;
    char             *s1, buf[1025];

    while (c = *fmt++) {
        if (c != '%') {
#ifdef CPM
            if (c == '\n') {
                if (putc('\r', stream) == EOF)
                    return nrchars ? -nrchars : -1;
                nrchars++;
            }
#endif
            if (putc(c, stream) == EOF)
                return nrchars ? -nrchars : -1;
            nrchars++;
            continue;
        }
        flags = 0;
        do {
            switch(*fmt) {
            case '-':      flags = FL_LJUST;      break;
            case '+':      flags = FL_SIGN;       break;
            case ' ':      flags = FL_SPACE;      break;
            case '#':      flags = FL_ALT;        break;
            case '0':      flags = FL_ZEROFILL;   break;
            default:       flags = FL_NOMORE;     continue;
            }
        } while (*fmt++);
    }

```

```

        fmt++;
    } while(!(flags & FL_NOMORE));

    oldfmt = fmt;
    fmt = gnum(fmt, &width, &ap);
    if (fmt != oldfmt) flags |= FL_WIDTHSPEC;

    if (*fmt == '.') {
        fmt++; oldfmt = fmt;
        fmt = gnum(fmt, &precision, &ap);
        if (precision >= 0) flags |= FL_PRECSPEC;
    }

    if ((flags & FL_WIDTHSPEC) && width < 0) {
        width = -width;
        flags |= FL_LJUST;
    }
    if (!(flags & FL_WIDTHSPEC)) width = 0;

    if (flags & FL_SIGN) flags &= ~FL_SPACE;

    if (flags & FL_LJUST) flags &= ~FL_ZEROFILL;

    s = s1 = buf;

    switch (*fmt) {
    case 'h':      flags |= FL_SHORT; fmt++; break;
    case 'l':      flags |= FL_LONG;  fmt++; break;
    case 'L':      flags |= FL_LONGDOUBLE; fmt++; break;
    }

    switch (c = *fmt++) {
    default:
        if (c == '\n') {
            if (putc('\r', stream) == EOF)
                return nrchars ? -nrchars : -1;
            nrchars++;
        }

        if (putc(c, stream) == EOF)
            return nrchars ? -nrchars : -1;
        nrchars++;
        continue;
    case 'n':
        if (flags & FL_SHORT)
            *va_arg(ap, short *) = (short) nrchars;
        else if (flags & FL_LONG)
            *va_arg(ap, long *) = (long) nrchars;
        else
            *va_arg(ap, int *) = (int) nrchars;
        continue;
    case 's':
        s1 = va_arg(ap, char *);
        if (s1 == NULL)
            s1 = "(null)";
        s = s1;
        while (precision || !(flags & FL_PRECSPEC)) {
            if (*s == '\0')
                break;
            s++;
            precision--;
        }
        break;
    case 'p':
        set_pointer(flags);
        /* fallthrough */
    case 'b':
    case 'o':
    case 'u':
    case 'x':
    case 'X':
        if (!(flags & FL_PRECSPEC)) precision = 1;

```

#ifdef CPM

#endif


```

        else if (c != 'p') flags &= ~FL_ZEROFILL;
        s = o_print(&ap, flags, s, c, precision, 0);
        break;
    case 'd':
    case 'i':
        flags |= FL_SIGNEDCONV;
        if (!(flags & FL_PRECSPEC)) precision = 1;
        else flags &= ~FL_ZEROFILL;
        s = o_print(&ap, flags, s, c, precision, 1);
        break;
    case 'c':
        *s++ = va_arg(ap, int);
        break;
#endif NOFLOAT
    case 'G':
    case 'g':
        if ((flags & FL_PRECSPEC) && (precision == 0))
            precision = 1;

    case 'f':
    case 'E':
    case 'e':
        if (!(flags & FL_PRECSPEC))
            precision = 6;

        if (precision >= sizeof(buf))
            precision = sizeof(buf) - 1;

        flags |= FL_SIGNEDCONV;
        s = _f_print(&ap, flags, s, c, precision);
        break;
#endif /* NOFLOAT */
    case 'r':
        ap = va_arg(ap, va_list);
        fmt = va_arg(ap, char *);
        continue;
}
zfill = ' ';
if (flags & FL_ZEROFILL) zfill = '0';
j = s - s1;

/* between_fill is true under the following conditions:
 * 1- the fill character is '0'
 * and
 * 2a- the number is of the form 0x... or 0X...
 * or
 * 2b- the number contains a sign or space
 */
between_fill = 0;
if ((flags & FL_ZEROFILL)
    && (((c == 'x' || c == 'X') && (flags & FL_ALT))
        || (c == 'p')
        || ((flags & FL_SIGNEDCONV)
            && (*s1 == '+' || *s1 == '-' || *s1 == ' '))))
    between_fill++;

if ((i = width - j) > 0)
    if (!(flags & FL_LJUST)) { /* right justify */
        nrchars += i;
        if (between_fill) {
            if (flags & FL_SIGNEDCONV) {
                j--; nrchars++;
                if (putc(*s1++, stream) == EOF)
                    return nrchars ? -nrchars : -1;
            } else {
                j -= 2; nrchars += 2;
                if ((putc(*s1++, stream) == EOF)
                    || (putc(*s1++, stream) == EOF))
                    return nrchars ? -nrchars : -1;
            }
        }
        do {
            if (putc(zfill, stream) == EOF)
                return nrchars ? -nrchars : -1;
        } while (--i);
    }

```

```
        }

        nrchars += j;
        while (--j >= 0) {
            if (putc(*s1++, stream) == EOF)
                return nrchars ? -nrchars : -1;
        }

        if (i > 0) nrchars += i;
        while (--i >= 0)
            if (putc(zfill, stream) == EOF)
                return nrchars ? -nrchars : -1;
    }
    return nrchars;
}
```

```

/*
 * doscan.c - scan formatted input
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/doscan.c,v 1.1.1.1 2005/04/21 14:56:35
beng Exp $ */

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <stdarg.h>
#include "loc_incl.h"

#if _EM_WSIZE == _EM_PSIZE
#define set_pointer(flags) /* nothing */
#elif _EM_LSIZE == _EM_PSIZE
#define set_pointer(flags) (flags |= FL_LONG)
#else
#error garbage pointer size
#define set_pointer(flags) /* compilation might continue */
#endif

#define NUMLEN 512
#define NR_CHARS 256

static char Xtable[NR_CHARS];
static char inp_buf[NUMLEN];

/* Collect a number of characters which constitute an ordinal number.
 * When the type is 'i', the base can be 8, 10, or 16, depending on the
 * first 1 or 2 characters. This means that the base must be adjusted
 * according to the format of the number. At the end of the function, base
 * is then set to 0, so strtol() will get the right argument.
 */
static char *
o_collect(register int c, register FILE *stream, char type,
          unsigned int width, int *basep)
{
    register char *bufp = inp_buf;
    register int base;

    switch (type) {
    case 'i': /* i means octal, decimal or hexadecimal */
    case 'p':
    case 'x':
    case 'X': base = 16; break;
    case 'd':
    case 'u': base = 10; break;
    case 'o': base = 8; break;
    case 'b': base = 2; break;
    }

    if (c == '-' || c == '+') {
        *bufp++ = c;
        if (--width)
            c = getc(stream);
    }

    if (width && c == '0' && base == 16) {
        *bufp++ = c;
        if (--width)
            c = getc(stream);
        if (c != 'x' && c != 'X') {
            if (type == 'i') base = 8;
        }
        else if (width) {
            *bufp++ = c;
            if (--width)
                c = getc(stream);
        }
    }
    else if (type == 'i') base = 10;

    while (width) {
        if ((base == 10) && isdigit(c))

```

```

        ((base == 16) && isxdigit(c))
        ((base == 8) && isdigit(c) && (c < '8'))
        ((base == 2) && isdigit(c) && (c < '2')) {
            *bufp++ = c;
            if (--width)
                c = getc(stream);
        }
    }
    else break;
}

if (width && c != EOF) ungetc(c, stream);
if (type == 'i') base = 0;
*basep = base;
*bufp = '\0';
return bufp - 1;
}

#ifdef NOFLOAT
/* The function f_collect() reads a string that has the format of a
 * floating-point number. The function returns as soon as a format-error
 * is encountered, leaving the offending character in the input. This means
 * that 1.e1 leaves the '1' in the input queue. Since all detection of
 * format errors is done here, _doscan() doesn't call strtod() when it's
 * not necessary, although the use of the width field can cause incomplete
 * numbers to be passed to strtod(). (e.g. 1.3e+)
 */
static char *
f_collect(register int c, register FILE *stream, register unsigned int width)
{
    register char *bufp = inp_buf;
    int digit_seen = 0;

    if (c == '-' || c == '+') {
        *bufp++ = c;
        if (--width)
            c = getc(stream);
    }

    while (width && isdigit(c)) {
        digit_seen++;
        *bufp++ = c;
        if (--width)
            c = getc(stream);
    }
    if (width && c == '.') {
        *bufp++ = c;
        if (--width)
            c = getc(stream);
        while (width && isdigit(c)) {
            digit_seen++;
            *bufp++ = c;
            if (--width)
                c = getc(stream);
        }
    }

    if (!digit_seen) {
        if (width && c != EOF) ungetc(c, stream);
        return inp_buf - 1;
    }
    else digit_seen = 0;

    if (width && (c == 'e' || c == 'E')) {
        *bufp++ = c;
        if (--width)
            c = getc(stream);
        if (width && (c == '+' || c == '-')) {
            *bufp++ = c;
            if (--width)
                c = getc(stream);
        }
        while (width && isdigit(c)) {
            digit_seen++;
            *bufp++ = c;

```

```

        if (--width)
            c = getc(stream);
    }
    if (!digit_seen) {
        if (width && c != EOF) ungetc(c, stream);
        return inp_buf - 1;
    }
}

if (width && c != EOF) ungetc(c, stream);
*bufp = '\0';
return bufp - 1;
}
#endif /* NOFLOAT */

/*
 * the routine that does the scanning
 */

int
_doscan(register FILE *stream, const char *format, va_list ap)
{
    int         done = 0;           /* number of items done */
    int         nrchars = 0;        /* number of characters read */
    int         conv = 0;           /* # of conversions */
    int         base;               /* conversion base */
    unsigned long val;              /* an integer value */
    register char *str;             /* temporary pointer */
    char        *tmp_string;        /* ditto */
    unsigned    width = 0;          /* width of field */
    int         flags;              /* some flags */
    int         reverse;            /* reverse the checking in [...] */
    int         kind;
    register int ic = EOF;          /* the input character */
#ifdef NOFLOAT
    long double ld_val;
#endif

    if (!*format) return 0;

    while (1) {
        if (isspace(*format)) {
            while (isspace(*format))
                format++;           /* skip whitespace */
            ic = getc(stream);
            nrchars++;
            while (isspace(ic)) {
                ic = getc(stream);
                nrchars++;
            }
            if (ic != EOF) ungetc(ic, stream);
            nrchars--;
        }
        if (!*format) break;        /* end of format */

        if (*format != '%') {
            ic = getc(stream);
            nrchars++;
            if (ic != *format++) break; /* error */
            continue;
        }
        format++;
        if (*format == '%') {
            ic = getc(stream);
            nrchars++;
            if (ic == '%') {
                format++;
                continue;
            }
            else break;
        }
        flags = 0;
        if (*format == '*') {

```

```

        format++;
        flags |= FL_NOASSIGN;
    }
    if (isdigit (*format)) {
        flags |= FL_WIDTHSPEC;
        for (width = 0; isdigit (*format);)
            width = width * 10 + *format++ - '0';
    }

    switch (*format) {
    case 'h': flags |= FL_SHORT; format++; break;
    case 'l': flags |= FL_LONG; format++; break;
    case 'L': flags |= FL_LONGDOUBLE; format++; break;
    }
    kind = *format;
    if ((kind != 'c') && (kind != '[') && (kind != 'n')) {
        do {
            ic = getc(stream);
            nrchars++;
        } while (isspace(ic));
        if (ic == EOF) break; /* outer while */
    } else if (kind != 'n') { /* %c or %[ */
        ic = getc(stream);
        if (ic == EOF) break; /* outer while */
        nrchars++;
    }
    switch (kind) {
    default:
        /* not recognized, like %q */
        return conv || (ic != EOF) ? done : EOF;
        break;
    case 'n':
        if (!(flags & FL_NOASSIGN)) { /* silly, though */
            if (flags & FL_SHORT)
                *va_arg(ap, short *) = (short) nrchars;
            else if (flags & FL_LONG)
                *va_arg(ap, long *) = (long) nrchars;
            else
                *va_arg(ap, int *) = (int) nrchars;
        }
        break;
    case 'p': /* pointer */
        set_pointer(flags);
        /* fallthrough */
    case 'b': /* binary */
    case 'd': /* decimal */
    case 'i': /* general integer */
    case 'o': /* octal */
    case 'u': /* unsigned */
    case 'x': /* hexadecimal */
    case 'X': /* ditto */
        if (!(flags & FL_WIDTHSPEC) || width > NUMLEN)
            width = NUMLEN;
        if (!width) return done;

        str = o_collect(ic, stream, kind, width, &base);
        if (str < inp_buf
            || (str == inp_buf
                && (*str == '-'
                    || *str == '+')))) return done;

        /*
         * Although the length of the number is str-inp_buf+1
         * we don't add the 1 since we counted it already
         */
        nrchars += str - inp_buf;

        if (!(flags & FL_NOASSIGN)) {
            if (kind == 'd' || kind == 'i')
                val = strtol(inp_buf, &tmp_string, base);
            else
                val = strtoul(inp_buf, &tmp_string, base);
            if (flags & FL_LONG)
                *va_arg(ap, unsigned long *) = (unsigned long) va

```

```

1;
                                else if (flags & FL_SHORT)
                                    *va_arg(ap, unsigned short *) = (unsigned short)
val;
                                else
                                    *va_arg(ap, unsigned *) = (unsigned) val;
                                }
                                break;
case 'c':
    if (!(flags & FL_WIDTHSPEC))
        width = 1;
    if (!(flags & FL_NOASSIGN))
        str = va_arg(ap, char *);
    if (!width) return done;

    while (width && ic != EOF) {
        if (!(flags & FL_NOASSIGN))
            *str++ = (char) ic;
        if (--width) {
            ic = getc(stream);
            nrchars++;
        }
    }

    if (width) {
        if (ic != EOF) ungetc(ic, stream);
        nrchars--;
    }
    break;
case 's':
    if (!(flags & FL_WIDTHSPEC))
        width = 0xffff;
    if (!(flags & FL_NOASSIGN))
        str = va_arg(ap, char *);
    if (!width) return done;

    while (width && ic != EOF && !isspace(ic)) {
        if (!(flags & FL_NOASSIGN))
            *str++ = (char) ic;
        if (--width) {
            ic = getc(stream);
            nrchars++;
        }
    }
    /* terminate the string */
    if (!(flags & FL_NOASSIGN))
        *str = '\0';
    if (width) {
        if (ic != EOF) ungetc(ic, stream);
        nrchars--;
    }
    break;
case '[':
    if (!(flags & FL_WIDTHSPEC))
        width = 0xffff;
    if (!width) return done;

    if ( *++format == '^' ) {
        reverse = 1;
        format++;
    } else
        reverse = 0;

    for (str = Xtable; str < &Xtable[NR_CHARS]
        ; str++)
        *str = 0;

    if (*format == ']') Xtable[*format++] = 1;

    while (*format && *format != ']') {
        Xtable[*format++] = 1;
        if (*format == '-') {
            format++;
            if (*format

```

```

        && *format != ']'
        && *(format) >= *(format -2)) {
            int c;

            for( c = *(format -2) + 1
                ; c <= *format ; c++)
                Xtable[c] = 1;
            format++;
        }
        else Xtable['-'] = 1;
    }
}
if (!*format) return done;

if (!(Xtable[ic] ^ reverse)) {
    /* MAT 8/9/96 no match must return character */
    ungetc(ic, stream);
    return done;
}

if (!(flags & FL_NOASSIGN))
    str = va_arg(ap, char *);

do {
    if (!(flags & FL_NOASSIGN))
        *str++ = (char) ic;
    if (--width) {
        ic = getc(stream);
        nrchars++;
    }
} while (width && ic != EOF && (Xtable[ic] ^ reverse));

if (width) {
    if (ic != EOF) ungetc(ic, stream);
    nrchars--;
}

if (!(flags & FL_NOASSIGN)) { /* terminate string */
    *str = '\0';
}
break;
#endif NOFLOAT

case 'e':
case 'E':
case 'f':
case 'g':
case 'G':
    if (!(flags & FL_WIDTHSPEC) || width > NUMLEN)
        width = NUMLEN;

    if (!width) return done;
    str = f_collect(ic, stream, width);

    if (str < inp_buf
        || (str == inp_buf
            && (*str == '-'
                || *str == '+'))) return done;

    /*
     * Although the length of the number is str-inp_buf+1
     * we don't add the 1 since we counted it already
     */
    nrchars += str - inp_buf;

    if (!(flags & FL_NOASSIGN)) {
        ld_val = strtod(inp_buf, &tmp_string);
        if (flags & FL_LONGDOUBLE)
            *va_arg(ap, long double *) = (long double) ld_val;

        else
            if (flags & FL_LONG)
                *va_arg(ap, double *) = (double) ld_val;
            else
                *va_arg(ap, float *) = (float) ld_val;
    }
}
;

```



```
                break;
#endif
        }                /* end switch */
        conv++;
        if (!(flags & FL_NOASSIGN) && kind != 'n') done++;
        format++;
    }
    return conv || (ic != EOF) ? done : EOF;
}
```

```
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/ecvt.c,v 1.1.1.1 2005/04/21 14:56:35 be
ng Exp $ */
```

```
#ifndef NOFLOAT
```

```
#include        "../ansi/ext_fmt.h"
```

```
void _dbl_ext_cvt(double value, struct EXTEND *e);
```

```
char *_ext_str_cvt(struct EXTEND *e, int ndigit, int *decpt, int *sign, int ecvtflag);
```

```
static char *
```

```
cvt(long double value, int ndigit, int *decpt, int *sign, int ecvtflag)
```

```
{
    struct EXTEND e;

    _dbl_ext_cvt(value, &e);
    return _ext_str_cvt(&e, ndigit, decpt, sign, ecvtflag);
}
```

```
char *
```

```
_ecvt(long double value, int ndigit, int *decpt, int *sign)
```

```
{
    return cvt(value, ndigit, decpt, sign, 1);
}
```

```
char *
```

```
_fcvt(long double value, int ndigit, int *decpt, int *sign)
```

```
{
    return cvt(value, ndigit, decpt, sign, 0);
}
```

```
#endif /* NOFLOAT */
```

```
/*
 * fclose.c - flush a stream and close the file
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fclose.c,v 1.1.1.1 2005/04/21 14:56:35
beng Exp $ */

#include <stdio.h>
#include <stdlib.h>
#include "loc_incl.h"

int _close(int d);

int
fclose(FILE *fp)
{
    register int i, retval = 0;

    for (i=0; i<FOPEN_MAX; i++)
        if (fp == __iotab[i]) {
            __iotab[i] = 0;
            break;
        }
    if (i >= FOPEN_MAX)
        return EOF;
    if (fflush(fp)) retval = EOF;
    if (_close(fileno(fp))) retval = EOF;
    if (io_testflag(fp, _IOMYBUF) && fp->_buf )
        free((void *)fp->_buf);
    if (fp != stdin && fp != stdout && fp != stderr)
        free((void *)fp);
    return retval;
}
```

```
/*
 * feof.c - test if eof on a stream occurred
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/feof.c,v 1.1.1.1 2005/04/21 14:56:35 be
ng Exp $ */

#include <stdio.h>

int
(feof)(FILE *stream)
{
    return feof(stream);
}
```

```
/*
 * ferror .c - test if an error on a stream occurred
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/ferror.c,v 1.1.1.1 2005/04/21 14:56:35
beng Exp $ */

#include <stdio.h>

int
(ferror)(FILE *stream)
{
    return ferror(stream);
}
```

```
/*
 * fflush.c - flush stream(s)
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fflush.c,v 1.3 2006/02/02 16:59:07 beng
Exp $ */

#include <sys/types.h>
#include <stdio.h>
#include <errno.h>
#include "loc_incl.h"

ssize_t _write(int d, const char *buf, size_t nbytes);
off_t _lseek(int fildes, off_t offset, int whence);

int
fflush(FILE *stream)
{
    int count, cl, i, retval = 0;

    if (!stream) {
        for(i= 0; i < FOPEN_MAX; i++)
            if (__iotab[i] && fflush(__iotab[i]))
                retval = EOF;
        return retval;
    }

    if (!stream->_buf
        || (!io_testflag(stream, _IOREADING)
            && !io_testflag(stream, _IOWRITING)))
        return 0;
    if (io_testflag(stream, _IOREADING)) {
        /* (void) fseek(stream, 0L, SEEK_CUR); */
        int adjust = 0;
        if (io_testflag(stream, _IOFIFO)) {
            /* Can't seek in a pipe. */
            return 0;
        }
        if (stream->_buf && !io_testflag(stream, _IONBF))
            adjust = -stream->_count;
        stream->_count = 0;
        if (_lseek(fileno(stream), (off_t) adjust, SEEK_CUR) == -1 &&
            errno != ESPIPE) {
            stream->_flags |= _IOERR;
            return EOF;
        }
        errno = 0;
        if (io_testflag(stream, _IOWRITE))
            stream->_flags &= ~(_IOREADING | _IOWRITING);
        stream->_ptr = stream->_buf;
        return 0;
    } else if (io_testflag(stream, _IONBF)) return 0;

    if (io_testflag(stream, _IOREAD)) /* "a" or "+" mode */
        stream->_flags &= ~_IOWRITING;

    count = stream->_ptr - stream->_buf;
    stream->_ptr = stream->_buf;

    if (count <= 0)
        return 0;

    if (io_testflag(stream, _IOAPPEND)) {
        if (_lseek(fileno(stream), 0L, SEEK_END) == -1) {
            stream->_flags |= _IOERR;
            return EOF;
        }
    }
    cl = _write(stream->_fd, (char *)stream->_buf, count);

    stream->_count = 0;

    if (count == cl)
        return 0;
}
```

```
    stream->_flags |= _IOERR;
    return EOF;
}

void
__cleanup(void)
{
    register int i;

    for(i= 0; i < FOPEN_MAX; i++)
        if (__iotab[i] && io_testflag(__iotab[i], _IOWRITING))
            (void) fflush(__iotab[i]);
}
```

```
/*  
 * fgetc - get an unsigned character and return it as an int  
 */  
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fgetc.c,v 1.1.1.1 2005/04/21 14:56:35 b  
eng Exp $ */  
  
#include          <stdio.h>  
  
int  
fgetc(FILE *stream)  
{  
    return getc(stream);  
}
```



```
/*
 * fgetpos.c - get the position in the file
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fgetpos.c,v 1.1.1.1 2005/04/21 14:56:35
   beng Exp $ */

#include <stdio.h>

int
fgetpos(FILE *stream, fpos_t *pos)
{
    *pos = ftell(stream);
    if (*pos == -1) return -1;
    return 0;
}
```

```
/*
 * fgets.c - get a string from a file
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fgets.c,v 1.1.1.1 2005/04/21 14:56:35 b
eng Exp $ */

#include <stdio.h>

char *
fgets(char *s, register int n, register FILE *stream)
{
    register int ch;
    register char *ptr;

    ptr = s;
    while (--n > 0 && (ch = getc(stream)) != EOF) {
        *ptr++ = ch;
        if (ch == '\n')
            break;
    }
    if (ch == EOF) {
        if (feof(stream)) {
            if (ptr == s) return NULL;
        } else return NULL;
    }
    *ptr = '\0';
    return s;
}
```

```
/*
 * fileno .c - map a stream to a file descriptor
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fileno.c,v 1.1.1.1 2005/04/21 14:56:35
beng Exp $ */

#include <stdio.h>

int
(fileno)(FILE *stream)
{
    return stream->_fd;
}
```

```

/*
 * fillbuf.c - fill a buffer
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fillbuf.c,v 1.1.1.1 2005/04/21 14:56:35
   beng Exp $ */

#if defined(_POSIX_SOURCE)
#include <sys/types.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include "loc_incl.h"

ssize_t _read(ssize_t d, char *buf, size_t nbytes);

int
__fillbuf(register FILE *stream)
{
    static unsigned char ch[FOPEN_MAX];
    register int i;

    stream->_count = 0;
    if (fileno(stream) < 0) return EOF;
    if (io_testflag(stream, (_IOEOF | _IOERR))) return EOF;
    if (!io_testflag(stream, _IOREAD))
        { stream->_flags |= _IOERR; return EOF; }
    if (io_testflag(stream, _IOWRITING))
        { stream->_flags |= _IOERR; return EOF; }

    if (!io_testflag(stream, _IOREADING))
        stream->_flags |= _IOREADING;

    if (!io_testflag(stream, _IONBF) && !stream->_buf) {
        stream->_buf = (unsigned char *) malloc(BUFSIZ);
        if (!stream->_buf) {
            stream->_flags |= _IONBF;
        }
        else {
            stream->_flags |= _IOMYBUF;
            stream->_bufsiz = BUFSIZ;
        }
    }

    /* flush line-buffered output when filling an input buffer */
    for (i = 0; i < FOPEN_MAX; i++) {
        if (__iotab[i] && io_testflag(__iotab[i], _IOLBF))
            if (io_testflag(__iotab[i], _IOWRITING))
                (void) fflush(__iotab[i]);
    }

    if (!stream->_buf) {
        stream->_buf = &ch[fileno(stream)];
        stream->_bufsiz = 1;
    }
    stream->_ptr = stream->_buf;
    stream->_count = _read(stream->_fd, (char *)stream->_buf, stream->_bufsiz);

    if (stream->_count <= 0) {
        if (stream->_count == 0) {
            stream->_flags |= _IOEOF;
        }
        else
            stream->_flags |= _IOERR;

        return EOF;
    }
    stream->_count--;

    return *stream->_ptr++;
}

```

```

/*
 * flushbuf.c - flush a buffer
 */
/* $Id: flushbuf.c,v 1.1.1.1 2005/04/21 14:56:35 beng Exp $ */

#include <stdio.h>
#include <stdlib.h>
#include "loc_incl.h"

#include <sys/types.h>

off_t _lseek(int fildes, off_t offset, int whence);
ssize_t _write(int d, const char *buf, int nbytes);
int _isatty(int d);
extern void (*_clean)(void);

static int
do_write(int d, char *buf, int nbytes)
{
    int c;

    /* POSIX actually allows write() to return a positive value less
       than nbytes, so loop ...
    */
    while ((c = _write(d, buf, nbytes)) > 0 && c < nbytes) {
        nbytes -= c;
        buf += c;
    }
    return c > 0;
}

int
__flushbuf(int c, FILE * stream)
{
    __clean = __cleanup;
    if (fileno(stream) < 0) return (unsigned char) c;
    if (!io_testflag(stream, _IOWRITE)) return EOF;
    if (io_testflag(stream, _IOREADING) && !feof(stream)) return EOF;

    stream->_flags &= ~_IOREADING;
    stream->_flags |= _IOWRITING;
    if (!io_testflag(stream, _IONBF)) {
        if (!stream->_buf) {
            if (stream == stdout && _isatty(fileno(stdout))) {
                if (!(stream->_buf =
                    (unsigned char *) malloc(BUFSIZ))) {
                    stream->_flags |= _IONBF;
                } else {
                    stream->_flags |= _IOLBF|_IOMYBUF;
                    stream->_bufsiz = BUFSIZ;
                    stream->_count = -1;
                }
            } else {
                if (!(stream->_buf =
                    (unsigned char *) malloc(BUFSIZ))) {
                    stream->_flags |= _IONBF;
                } else {
                    stream->_flags |= _IOMYBUF;
                    stream->_bufsiz = BUFSIZ;
                    if (!io_testflag(stream, _IOLBF))
                        stream->_count = BUFSIZ - 1;
                    else
                        stream->_count = -1;
                }
            }
            stream->_ptr = stream->_buf;
        }
    }

    if (io_testflag(stream, _IONBF)) {
        char c1 = c;

        stream->_count = 0;
        if (io_testflag(stream, _IOAPPEND)) {
            if (_lseek(fileno(stream), 0L, SEEK_END) == -1) {

```

```
        stream->_flags |= _IOERR;
        return EOF;
    }
    if (_write(fileno(stream), &c1, 1) != 1) {
        stream->_flags |= _IOERR;
        return EOF;
    }
    return (unsigned char) c;
} else if (io_testflag(stream, _IOLBF)) {
    *stream->_ptr++ = c;
    /* stream->_count has been updated in putc macro. */
    if (c == '\n' || stream->_count == -stream->_bufsiz) {
        int count = -stream->_count;

        stream->_ptr = stream->_buf;
        stream->_count = 0;

        if (io_testflag(stream, _IOAPPEND)) {
            if (_lseek(fileno(stream), 0L, SEEK_END) == -1) {
                stream->_flags |= _IOERR;
                return EOF;
            }
        }
        if (! do_write(fileno(stream), (char *)stream->_buf,
            count)) {
            stream->_flags |= _IOERR;
            return EOF;
        }
    }
} else {
    int count = stream->_ptr - stream->_buf;

    stream->_count = stream->_bufsiz - 1;
    stream->_ptr = stream->_buf + 1;

    if (count > 0) {
        if (io_testflag(stream, _IOAPPEND)) {
            if (_lseek(fileno(stream), 0L, SEEK_END) == -1) {
                stream->_flags |= _IOERR;
                return EOF;
            }
        }
        if (! do_write(fileno(stream), (char *)stream->_buf, count)) {
            *(stream->_buf) = c;
            stream->_flags |= _IOERR;
            return EOF;
        }
    }
    *(stream->_buf) = c;
}
return (unsigned char) c;
}
```

```

/*
 * fopen.c - open a stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fopen.c,v 1.4 2006/02/02 16:59:07 beng
Exp $ */

#if defined(_POSIX_SOURCE)
#include <sys/types.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include "loc_incl.h"
#include <sys/stat.h>

#define PMODE 0666

/* The next 3 defines are true in all UNIX systems known to me.
 */
#define O_RDONLY 0
#define O_WRONLY 1
#define O_RDWR 2

/* Since the O_CREAT flag is not available on all systems, we can't get it
 * from the standard library. Furthermore, even if we know that <fcntl.h>
 * contains such a flag, it's not sure whether it can be used, since we
 * might be cross-compiling for another system, which may use an entirely
 * different value for O_CREAT (or not support such a mode). The safest
 * thing is to just use the Version 7 semantics for open, and use creat()
 * whenever necessary.
 *
 * Another problem is O_APPEND, for which the same holds. When "a"
 * open-mode is used, an lseek() to the end is done before every write()
 * system-call.
 *
 * The O_CREAT, O_TRUNC and O_APPEND given here, are only for convenience.
 * They are not passed to open(), so the values don't have to match a value
 * from the real world. It is enough when they are unique.
 */
#define O_CREAT 0x010
#define O_TRUNC 0x020
#define O_APPEND 0x040

int _open(const char *path, int flags);
int _creat(const char *path, _mnx_Mode_t mode);
int _close(int d);

FILE *
fopen(const char *name, const char *mode)
{
    register int i;
    int rwmode = 0, rwflags = 0;
    FILE *stream;
    struct stat st;
    int fd, flags = 0;

    for (i = 0; __iotab[i] != 0 ; i++)
        if ( i >= FOPEN_MAX-1 )
            return (FILE *)NULL;

    switch(*mode++) {
    case 'r':
        flags |= _IOREAD | _IOREADING;
        rwmode = O_RDONLY;
        break;
    case 'w':
        flags |= _IOWRITE | _IOWRITING;
        rwmode = O_WRONLY;
        rwflags = O_CREAT | O_TRUNC;
        break;
    case 'a':
        flags |= _IOWRITE | _IOWRITING | _IOAPPEND;
        rwmode = O_WRONLY;
        rwflags |= O_APPEND | O_CREAT;
        break;
    }

```

```
default:
    return (FILE *)NULL;
}

while (*mode) {
    switch(*mode++) {
        case 'b':
            continue;
        case '+':
            rwmode = O_RDWR;
            flags |= _IOREAD | _IOWRITE;
            continue;
        /* The sequence may be followed by additional characters */
        default:
            break;
    }
    break;
}

/* Perform a creat() when the file should be truncated or when
 * the file is opened for writing and the open() failed.
 */
if ((rwflags & O_TRUNC)
    || (((fd = _open(name, rwmode)) < 0)
        && (rwflags & O_CREAT))) {
    if (((fd = _creat(name, PMODE)) > 0) && flags | _IOREAD) {
        (void) _close(fd);
        fd = _open(name, rwmode);
    }
}

if (fd < 0) return (FILE *)NULL;

if ( fstat( fd, &st ) < 0 ) {
    _close(fd);
    return (FILE *)NULL;
}

if ( S_ISFIFO(st.st_mode) ) flags |= _IOFIFO;

if ( ( stream = (FILE *) malloc(sizeof(FILE))) == NULL ) {
    _close(fd);
    return (FILE *)NULL;
}

if ((flags & (_IOREAD | _IOWRITE)) == (_IOREAD | _IOWRITE))
    flags &= ~(_IOREADING | _IOWRITING);

stream->_count = 0;
stream->_fd = fd;
stream->_flags = flags;
stream->_buf = NULL;
__iotab[i] = stream;
return stream;
}
```



```
/*
 * fprintf - write output on a stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fprintf.c,v 1.1.1.1 2005/04/21 14:56:35
   beng Exp $ */

#include <stdio.h>
#include <stdarg.h>
#include "loc_incl.h"

int
fprintf(FILE *stream, const char *format, ...)
{
    va_list ap;
    int retval;

    va_start(ap, format);

    retval = _doprnt (format, ap, stream);

    va_end(ap);

    return retval;
}
```

```
/*  
 * fputc.c - print an unsigned character  
 */  
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fputc.c,v 1.1.1.1 2005/04/21 14:56:35 b  
eng Exp $ */  
  
#include          <stdio.h>  
  
int  
fputc(int c, FILE *stream)  
{  
    return putc(c, stream);  
}
```

```
/*
 * fputs - print a string
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fputs.c,v 1.1.1.1 2005/04/21 14:56:35 b
eng Exp $ */

#include <stdio.h>

int
fputs(register const char *s, register FILE *stream)
{
    register int i = 0;

    while (*s)
        if (putc(*s++, stream) == EOF) return EOF;
        else i++;

    return i;
}
```

```
/*
 * fread.c - read a number of members into an array
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fread.c,v 1.1.1.1 2005/04/21 14:56:35 b
eng Exp $ */

#include <stdio.h>

size_t
fread(void *ptr, size_t size, size_t nmemb, register FILE *stream)
{
    register char *cp = ptr;
    register int c;
    size_t ndone = 0;
    register size_t s;

    if (size)
        while ( ndone < nmemb ) {
            s = size;
            do {
                if ((c = getc(stream)) != EOF)
                    *cp++ = c;
                else
                    return ndone;
            } while (--s);
            ndone++;
        }

    return ndone;
}
```

```

/*
 * freopen.c - open a file and associate a stream with it
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/freopen.c,v 1.4 2006/02/02 16:59:07 ben
g Exp $ */

#if      defined(_POSIX_SOURCE)
#include    <sys/types.h>
#endif
#include    <stdio.h>
#include    <stdlib.h>
#include    "loc_incl.h"
#include    <sys/stat.h>

#define PMODE      0666

/* Do not "optimize" this file to use the open with O_CREAT if the file
 * does not exist. The reason is given in fopen.c.
 */
#define O_RDONLY      0
#define O_WRONLY      1
#define O_RDWR        2

#define O_CREAT        0x010
#define O_TRUNC        0x020
#define O_APPEND        0x040

int _open(const char *path, int flags);
int _creat(const char *path, _mnx_Mode_t mode);
int _close(int d);

FILE *
freopen(const char *name, const char *mode, FILE *stream)
{
    register int i;
    struct stat st;
    int rwmode = 0, rwflags = 0;
    int fd, flags = stream->_flags & (_IONBF | _IOFBF | _IOLBF | _IOMYBUF);

    (void) fflush(stream);
    (void) _close(fileno(stream));

    switch(*mode++) {
    case 'r':
        flags |= _IOREAD;
        rwmode = O_RDONLY;
        break;
    case 'w':
        flags |= _IOWRITE;
        rwmode = O_WRONLY;
        rwflags = O_CREAT | O_TRUNC;
        break;
    case 'a':
        flags |= _IOWRITE | _IOAPPEND;
        rwmode = O_WRONLY;
        rwflags |= O_APPEND | O_CREAT;
        break;
    default:
        goto loser;
    }

    while (*mode) {
        switch(*mode++) {
        case 'b':
            continue;
        case '+':
            rwmode = O_RDWR;
            flags |= _IOREAD | _IOWRITE;
            continue;
        /* The sequence may be followed by additional characters */
        default:
            break;
        }
        break;
    }
}

```

```
    }

    if ((rwflags & O_TRUNC)
        || (((fd = _open(name, rwmode)) < 0)
            && (rwflags & O_CREAT))) {
        if (((fd = _creat(name, PMODE)) < 0) && flags | _IOREAD) {
            (void) _close(fd);
            fd = _open(name, rwmode);
        }
    }

    if (fd < 0) {
        goto loser;
    }

    if ( fstat( fd, &st ) == 0 ) {
        if ( S_ISFIFO(st.st_mode) ) flags |= _IOFIFO;
    } else {
        goto loser;
    }

    stream->_count = 0;
    stream->_fd = fd;
    stream->_flags = flags;
    return stream;

loser:
    for( i = 0; i < FOPEN_MAX; i++) {
        if (stream == __iotab[i]) {
            __iotab[i] = 0;
            break;
        }
    }
    if (stream != stdin && stream != stdout && stream != stderr)
        free((void *)stream);
    return (FILE *)NULL;
}
```

```
/*
 * fscanf.c - read formatted input from stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fscanf.c,v 1.1.1.1 2005/04/21 14:56:35
beng Exp $ */

#include <stdio.h>
#include <stdarg.h>
#include "loc_incl.h"

int
fscanf(FILE *stream, const char *format, ...)
{
    va_list ap;
    int retval;

    va_start(ap, format);

    retval = _doscan(stream, format, ap);

    va_end(ap);

    return retval;
}
```

```
/*
 * fseek.c - perform an fseek
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fseek.c,v 1.1.1.1 2005/04/21 14:56:35 b
eng Exp $ */

#include <stdio.h>

#if (SEEK_CUR != 1) || (SEEK_END != 2) || (SEEK_SET != 0)
#error SEEK_* values are wrong
#endif

#include "loc_incl.h"

#include <sys/types.h>

off_t _lseek(int fildes, off_t offset, int whence);

int
fseek(FILE *stream, long int offset, int whence)
{
    int adjust = 0;
    long pos;

    stream->_flags &= ~(_IOEOF | _IOERR);
    /* Clear both the end of file and error flags */

    if (io_testflag(stream, _IOREADING)) {
        if (whence == SEEK_CUR
            && stream->_buf
            && !io_testflag(stream, _IONBF))
            adjust = stream->_count;
        stream->_count = 0;
    } else if (io_testflag(stream, _IOWRITING)) {
        fflush(stream);
    } else /* neither reading nor writing. The buffer must be empty */
        /* EMPTY */ ;

    pos = _lseek(fileno(stream), offset - adjust, whence);
    if (io_testflag(stream, _IOREAD) && io_testflag(stream, _IOWRITE))
        stream->_flags &= ~(_IOREADING | _IOWRITING);

    stream->_ptr = stream->_buf;
    return ((pos == -1) ? -1 : 0);
}
```



```
/*
 * fsetpos.c - set the position in the file
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fsetpos.c,v 1.1.1.1 2005/04/21 14:56:35
   beng Exp $ */

#include <stdio.h>

int
fsetpos(FILE *stream, fpos_t *pos)
{
    return fseek(stream, *pos, SEEK_SET);
}
```

```
/*
 * ftell.c - obtain the value of the file-position indicator of a stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/ftell.c,v 1.1.1.1 2005/04/21 14:56:35 b
eng Exp $ */

#include <stdio.h>

#if (SEEK_CUR != 1) || (SEEK_SET != 0) || (SEEK_END != 2)
#error SEEK_* values are wrong
#endif

#include "loc_incl.h"

#include <sys/types.h>

off_t _lseek(int fildes, off_t offset, int whence);

long ftell(FILE *stream)
{
    long result;
    int adjust = 0;

    if (io_testflag(stream, _IOREADING))
        adjust = -stream->_count;
    else if (io_testflag(stream, _IOWRITING)
        && stream->_buf
        && !io_testflag(stream, _IONBF))
        adjust = stream->_ptr - stream->_buf;
    else adjust = 0;

    result = _lseek(fileno(stream), (off_t)0, SEEK_CUR);

    if ( result == -1 )
        return result;

    result += (long) adjust;
    return result;
}
```

```
/*
 * fwrite.c - write a number of array elements on a file
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/fwrite.c,v 1.1.1.1 2005/04/21 14:56:35
beng Exp $ */

#include <stdio.h>

size_t
fwrite(const void *ptr, size_t size, size_t nmemb,
        register FILE *stream)
{
    register const unsigned char *cp = ptr;
    register size_t s;
    size_t ndone = 0;

    if (size)
        while ( ndone < nmemb ) {
            s = size;
            do {
                if (putc((int)*cp, stream)
                    == EOF)
                    return ndone;

                cp++;
            }
            while (--s);
            ndone++;
        }
    return ndone;
}
```

```
/*
 * getc.c - read an unsigned character
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/getc.c,v 1.1.1.1 2005/04/21 14:56:35 be
ng Exp $ */

#include <stdio.h>

int
getc(FILE *stream)
{
    return getc(stream);
}
```

```
/*
 * getchar.c - read a character from the standard input stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/getchar.c,v 1.1.1.1 2005/04/21 14:56:35
   beng Exp $ */

#include <stdio.h>

int
(getchar)(void)
{
    return getchar();
}
```

```
/*
 * gets.c - read a line from a stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/gets.c,v 1.1.1.1 2005/04/21 14:56:35 be
ng Exp $ */

#include <stdio.h>

char *
gets(char *s)
{
    register FILE *stream = stdin;
    register int ch;
    register char *ptr;

    ptr = s;
    while ((ch = getc(stream)) != EOF && ch != '\n')
        *ptr++ = ch;

    if (ch == EOF) {
        if (feof(stream)) {
            if (ptr == s) return NULL;
        } else return NULL;
    }

    *ptr = '\0';
    return s;
}
```

```
/*
 * icompute.c - compute an integer
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/icompute.c,v 1.1.1.1 2005/04/21 14:56:3
5 beng Exp $ */

#include "loc_incl.h"

/* This routine is used in doprnt.c as well as in tmpfile.c and tmpnam.c. */

char *
_i_compute(unsigned long val, int base, char *s, int nrdigits)
{
    int c;

    c= val % base ;
    val /= base ;
    if (val || nrdigits > 1)
        s = _i_compute(val, base, s, nrdigits - 1);
    *s++ = (c>9 ? c-10+'a' : c+'0');
    return s;
}
```

```
/*
 * loc_incl.h - local include file for stdio library
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/loc_incl.h,v 1.1.1.1 2005/04/21 14:56:3
6 beng Exp $ */

#include <stdio.h>

#define io_testflag(p,x) ((p)->_flags & (x))

#include <stdarg.h>

#ifdef _ANSI
int _doprnt(const char *format, va_list ap, FILE *stream);
int _doscan(FILE * stream, const char *format, va_list ap);
char *_i_compute(unsigned long val, int base, char *s, int nrdigits);
char *_f_print(va_list *ap, int flags, char *s, char c, int precision);
void __cleanup(void);

FILE *popen(const char *command, const char *type);
FILE *fdopen(int fd, const char *mode);

#ifdef NOFLOAT
char *_ecvt(long double value, int ndigit, int *decpt, int *sign);
char *_fcvt(long double value, int ndigit, int *decpt, int *sign);
#endif /* NOFLOAT */
#endif

#define FL_LJUST      0x0001 /* left-justify field */
#define FL_SIGN       0x0002 /* sign in signed conversions */
#define FL_SPACE      0x0004 /* space in signed conversions */
#define FL_ALT        0x0008 /* alternate form */
#define FL_ZEROFILL    0x0010 /* fill with zero's */
#define FL_SHORT      0x0020 /* optional h */
#define FL_LONG        0x0040 /* optional l */
#define FL_LONGDOUBLE  0x0080 /* optional L */
#define FL_WIDTHSPEC   0x0100 /* field width is specified */
#define FL_PRECSPEC    0x0200 /* precision is specified */
#define FL_SIGNEDCONV  0x0400 /* may contain a sign */
#define FL_NOASSIGN    0x0800 /* do not assign (in scanf) */
#define FL_NOMORE      0x1000 /* all flags collected */
```



```

/*
 * Copyright (c) 1987, 1993
 * The Regents of the University of California. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in the
 * documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 * must display the following acknowledgement:
 * This product includes software developed by the University of
 * California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 * may be used to endorse or promote products derived from this software
 * without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)mktemp.c 8.1 (Berkeley) 6/4/93";
#endif /* LIBC_SCCS and not lint */

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <stdio.h>
#include <ctype.h>
#include <unistd.h>
#include <stdlib.h>

static int _gettemp(char*, int*);

mkstemp(path)
    char *path;
{
    int fd;

    return (_gettemp(path, &fd) ? fd : -1);
}

char *
mktemp(path)
    char *path;
{
    return(_gettemp(path, (int *)NULL) ? path : (char *)NULL);
}

static
_gettemp(path, doopen)
    char *path;
    register int *doopen;
{
    extern int errno;
    register char *start, *trv;
    struct stat sbuf;
    u_int pid;

    pid = getpid();

```

```
for (trv = path; *trv; ++trv);          /* extra X's get set to 0's */
while (*--trv == 'X') {
    *trv = (pid % 10) + '0';
    pid /= 10;
}

/*
 * check the target directory; if you have six X's and it
 * doesn't exist this runs for a very long time.
 */
for (start = trv + 1;; --trv) {
    if (trv <= path)
        break;
    if (*trv == '/') {
        *trv = '\0';
        if (stat(path, &sbuf))
            return(0);
        if (!S_ISDIR(sbuf.st_mode)) {
            errno = ENOTDIR;
            return(0);
        }
        *trv = '/';
        break;
    }
}

for (;;) {
    if (doopen) {
        if ((*doopen =
            open(path, O_CREAT|O_EXCL|O_RDWR, 0600)) >= 0)
            return(1);
        if (errno != EEXIST)
            return(0);
    }
    else if (stat(path, &sbuf))
        return(errno == ENOENT ? 1 : 0);

    /* tricky little algorithm for backward compatibility */
    for (trv = start;;) {
        if (!*trv)
            return(0);
        if (*trv == 'z')
            *trv++ = 'a';
        else {
            if (isdigit(*trv))
                *trv = 'a';
            else
                ++*trv;
            break;
        }
    }
}

/*NOTREACHED*/
}

/*
 * $PchId: mktemp.c,v 1.3 1995/11/20 19:10:39 philip Exp $
 */
```

```
/*
 * perror.c - print an error message on the standard error output
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/perror.c,v 1.1.1.1 2005/04/21 14:56:36
beng Exp $ */

#if defined(_POSIX_SOURCE)
#include <sys/types.h>
#endif
#include <stdio.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>
#include "loc_incl.h"

ssize_t _write(int d, const char *buf, size_t nbytes);

void
perror(const char *s)
{
    char *p;
    int fd;

    p = strerror(errno);
    fd = fileno(stderr);
    fflush(stdout);
    fflush(stderr);
    if (s && *s) {
        _write(fd, s, strlen(s));
        _write(fd, ":", 2);
    }
    _write(fd, p, strlen(p));
    _write(fd, "\n", 1);
}
```

```
/*
 * printf - write on the standard output stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/printf.c,v 1.1.1.1 2005/04/21 14:56:36
beng Exp $ */

#include <stdio.h>
#include <stdarg.h>
#include "loc_incl.h"

int
printf(const char *format, ...)
{
    va_list ap;
    int retval;

    va_start(ap, format);

    retval = _doprnt(format, ap, stdout);

    va_end(ap);

    return retval;
}
```

```
/*  
 * putc.c - print (or buffer) one character  
 */  
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/putc.c,v 1.1.1.1 2005/04/21 14:56:36 be  
ng Exp $ */  
  
#include          <stdio.h>  
  
int  
(putc)(int c, FILE *stream)  
{  
    return putc(c, stream);  
}
```

```
/*
 * putchar.c - print (or buffer) a character on the standard output stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/putchar.c,v 1.1.1.1 2005/04/21 14:56:36
   beng Exp $ */

#include <stdio.h>

int
_putchar(int c)
{
    return putchar(c);
}
```

```
/*
 * puts.c - print a string onto the standard output stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/puts.c,v 1.1.1.1 2005/04/21 14:56:36 be
ng Exp $ */

#include <stdio.h>

int
puts(register const char *s)
{
    register FILE *file = stdout;
    register int i = 0;

    while (*s) {
        if (putc(*s++, file) == EOF) return EOF;
        else i++;
    }
    if (putc('\n', file) == EOF) return EOF;
    return i + 1;
}
```

```
/*
 * remove.c - remove a file
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/remove.c,v 1.1.1.1 2005/04/21 14:56:36
beng Exp $ */

#include <stdio.h>

int _unlink(const char *path);

int
remove(const char *filename) {
    return _unlink(filename);
}
```



```
/*
 * rewind.c - set the file position indicator of a stream to the start
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/rewind.c,v 1.1.1.1 2005/04/21 14:56:36
beng Exp $ */

#include <stdio.h>
#include "loc_incl.h"

void
rewind(FILE *stream)
{
    (void) fseek(stream, 0L, SEEK_SET);
    clearerr(stream);
}
```

```
/*
 * scanf.c - read formatted input from the standard input stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/scanf.c,v 1.1.1.1 2005/04/21 14:56:36 b
eng Exp $ */

#include <stdio.h>
#include <stdarg.h>
#include "loc_incl.h"

int
scanf(const char *format, ...)
{
    va_list ap;
    int retval;

    va_start(ap, format);

    retval = _doscan(stdin, format, ap);

    va_end(ap);

    return retval;
}
```

```
/*
 * setbuf.c - control buffering of a stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/setbuf.c,v 1.1.1.1 2005/04/21 14:56:36
beng Exp $ */

#include <stdio.h>
#include "loc_incl.h"

void
setbuf(register FILE *stream, char *buf)
{
    (void) setvbuf(stream, buf, (buf ? _IOFBF : _IONBF), (size_t) BUFSIZ);
}
```

```
/*
 * setvbuf.c - control buffering of a stream
 */
/* $Id: setvbuf.c,v 1.1.1.1 2005/04/21 14:56:36 beng Exp $ */

#include <stdio.h>
#include <stdlib.h>
#include "loc_incl.h"

extern void (*_clean)(void);

int
setvbuf(register FILE *stream, char *buf, int mode, size_t size)
{
    int retval = 0;

    _clean = __cleanup;
    if (mode != _IOFBF && mode != _IOLBF && mode != _IONBF)
        return EOF;

    if (stream->_buf && io_testflag(stream, _IOMYBUF) )
        free((void *)stream->_buf);

    stream->_flags &= ~(_IOMYBUF | _IONBF | _IOLBF);

    if (buf && size <= 0) retval = EOF;
    if (!buf && (mode != _IONBF)) {
        if (size <= 0 || (buf = (char *) malloc(size)) == NULL) {
            retval = EOF;
        } else {
            stream->_flags |= _IOMYBUF;
        }
    }

    stream->_buf = (unsigned char *) buf;

    stream->_count = 0;
    stream->_flags |= mode;
    stream->_ptr = stream->_buf;

    if (!buf) {
        stream->_bufsiz = 1;
    } else {
        stream->_bufsiz = size;
    }

    return retval;
}
```

```
/*
 * sprintf - print formatted output on an array
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/sprintf.c,v 1.1.1.1 2005/04/21 14:56:36
   beng Exp $ */

#include <stdio.h>
#include <stdarg.h>
#include <limits.h>
#include "loc_incl.h"

int
sprintf(char *s, const char *format, ...)
{
    va_list ap;
    int retval;

    va_start(ap, format);

    retval = vsnprintf(s, INT_MAX, format, ap);

    va_end(ap);

    return retval;
}

int
snprintf(char *s, size_t n, const char *format, ...)
{
    va_list ap;
    int retval;

    va_start(ap, format);

    retval = vsnprintf(s, n, format, ap);

    va_end(ap);

    return retval;
}
```

```
/*
 * sscanf - read formatted output from a string
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/sscanf.c,v 1.1.1.1 2005/04/21 14:56:36
beng Exp $ */

#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include "loc_incl.h"

int sscanf(const char *s, const char *format, ...)
{
    va_list ap;
    int retval;
    FILE tmp_stream;

    va_start(ap, format);

    tmp_stream._fd      = -1;
    tmp_stream._flags   = _IOREAD + _IONBF + _IOREADING;
    tmp_stream._buf     = (unsigned char *) s;
    tmp_stream._ptr     = (unsigned char *) s;
    tmp_stream._count   = strlen(s);

    retval = _doscan(&tmp_stream, format, ap);

    va_end(ap);

    return retval;
}
```

```
/*
 * tmpfile.c - create and open a temporary file
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/tmpfile.c,v 1.1.1.1 2005/04/21 14:56:36
beng Exp $ */

#if defined(_POSIX_SOURCE)
#include <sys/types.h>
#endif
#include <stdio.h>
#include <string.h>
#include "loc_incl.h"

pid_t _getpid(void);

FILE *
tmpfile(void) {
    static char name_buffer[L_tmpnam] = "/tmp/tmp." ;
    static char *name = NULL;
    FILE *file;

    if (!name) {
        name = name_buffer + strlen(name_buffer);
        name = _i_compute(_getpid(), 10, name, 5);
        *name = '\0';
    }

    file = fopen(name_buffer, "wb+");
    if (!file) return (FILE *)NULL;
    (void) remove(name_buffer);
    return file;
}
```

```
/*
 * tmpnam.c - create a unique filename
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/tmpnam.c,v 1.1.1.1 2005/04/21 14:56:36
beng Exp $ */

#if      defined(_POSIX_SOURCE)
#include      <sys/types.h>
#endif
#include      <stdio.h>
#include      <string.h>
#include      "loc_incl.h"

pid_t _getpid(void);

char *
tmpnam(char *s) {
    static char name_buffer[L_tmpnam] = "/tmp/tmp.";
    static unsigned long count = 0;
    static char *name = NULL;

    if (!name) {
        name = name_buffer + strlen(name_buffer);
        name = _i_compute((unsigned long)_getpid(), 10, name, 5);
        *name++ = '.';
        *name = '\0';
    }
    if (++count > TMP_MAX) count = 1;      /* wrap-around */
    *_i_compute(count, 10, name, 3) = '\0';
    if (s) return strcpy(s, name_buffer);
    else return name_buffer;
}
```



```
/*
 * ungetc.c - push a character back onto an input stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/ungetc.c,v 1.1.1.1 2005/04/21 14:56:36
beng Exp $ */

#include <stdio.h>
#include "loc_incl.h"

int
ungetc(int ch, FILE *stream)
{
    unsigned char *p;

    if (ch == EOF || !io_testflag(stream, _IOREADING))
        return EOF;
    if (stream->_ptr == stream->_buf) {
        if (stream->_count != 0) return EOF;
        stream->_ptr++;
    }
    stream->_count++;
    p = --(stream->_ptr);
    /* ??? Bloody vax assembler !!! */
    /* ungetc() in sscanf() shouldn't write in rom */
    if (*p != (unsigned char) ch)
        *p = (unsigned char) ch;
    return ch;
}
```

```
/*
 * vfprintf - formatted output without ellipsis
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/vfprintf.c,v 1.1.1.1 2005/04/21 14:56:3
6 beng Exp $ */

#include <stdio.h>
#include <stdarg.h>
#include "loc_incl.h"

int
vfprintf(FILE *stream, const char *format, va_list arg)
{
    return _doprnt (format, arg, stream);
}
```

```
/*
 * vprintf - formatted output without ellipsis to the standard output stream
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/vprintf.c,v 1.1.1.1 2005/04/21 14:56:36
   beng Exp $ */

#include <stdio.h>
#include <stdarg.h>
#include "loc_incl.h"

int
vprintf(const char *format, va_list arg)
{
    return _doprnt(format, arg, stdout);
}
```

```
/*  
 * vscanf.c - read formatted input from the standard input stream  
 */  
  
#include <stdio.h>  
#include <stdarg.h>  
#include "loc_incl.h"  
  
int  
vscanf(const char *format, va_list ap)  
{  
    return _doscan(stdin, format, ap);  
}
```

```
/*
 * vsprintf - print formatted output without ellipsis on an array
 */
/* $Header: /opt/proj/minix/cvsroot/src/lib/stdio/vsprintf.c,v 1.1.1.1 2005/04/21 14:56:3
6 beng Exp $ */

#include <stdio.h>
#include <stdarg.h>
#include <limits.h>
#include "loc_incl.h"

int
vsnprintf(char *s, size_t n, const char *format, va_list arg)
{
    int retval;
    FILE tmp_stream;

    tmp_stream._fd = -1;
    tmp_stream._flags = _IOWRITE + _IONBF + _IOWRITING;
    tmp_stream._buf = (unsigned char *) s;
    tmp_stream._ptr = (unsigned char *) s;
    tmp_stream._count = n-1;

    retval = _doprnt(format, arg, &tmp_stream);
    tmp_stream._count = 1;
    putc('\0', &tmp_stream);

    return retval;
}

int
vsprintf(char *s, const char *format, va_list arg)
{
    return vsnprintf(s, INT_MAX, format, arg);
}
```

```
/*
 * vsscanf - read formatted output from a string
 */

#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include "loc_incl.h"

int vsscanf(const char *s, const char *format, va_list ap)
{
    FILE tmp_stream;

    tmp_stream._fd      = -1;
    tmp_stream._flags   = _IOREAD + _IONBF + _IOREADING;
    tmp_stream._buf     = (unsigned char *) s;
    tmp_stream._ptr     = (unsigned char *) s;
    tmp_stream._count   = strlen(s);

    return _doscan(&tmp_stream, format, ap);
}
```

```
# Makefile for lib/syscall.
```

```
LIBRARIES=libc
```

```
libc_FILES=" \  
  _exit.s \  
  _pm_findproc.s \  
  access.s \  
  alarm.s \  
  brk.s \  
  cfgetispeed.s \  
  cfgetospeed.s \  
  cfsetispeed.s \  
  cfsetospeed.s \  
  chdir.s \  
  chmod.s \  
  chown.s \  
  chroot.s \  
  close.s \  
  closedir.s \  
  creat.s \  
  devctl.s \  
  dup.s \  
  dup2.s \  
  execl.s \  
  execle.s \  
  execlp.s \  
  execv.s \  
  execve.s \  
  execvp.s \  
  fchown.s \  
  fchmod.s \  
  fcntl.s \  
  fork.s \  
  fpathconf.s \  
  fstat.s \  
  fstatfs.s \  
  getcwd.s \  
  getegid.s \  
  geteuid.s \  
  getgid.s \  
  getgroups.s \  
  getnpid.s \  
  getnprocnr.s \  
  getpgrp.s \  
  getpid.s \  
  getppid.s \  
  getpprocnr.s \  
  getprocnr.s \  
  getsigset.s \  
  getsysinfo.s \  
  getuid.s \  
  ioctl.s \  
  isatty.s \  
  kill.s \  
  link.s \  
  lseek.s \  
  lstat.s \  
  mkdir.s \  
  mkfifo.s \  
  mknod.s \  
  mount.s \  
  open.s \  
  opendir.s \  
  pathconf.s \  
  pause.s \  
  pipe.s \  
  ptrace.s \  
  read.s \  
  readdir.s \  
  readlink.s \  
  reboot.s \  
  rename.s \  
  rewinddir.s \  
"
```

rmkdir.s \
sbrk.s \
seekdir.s \
setgid.s \
setsid.s \
setuid.s \
sigaction.s \
sigaddset.s \
sigdelset.s \
sigemptyset.s \
sigfillset.s \
sigismember.s \
sigpending.s \
sigprocmask.s \
sigreturn.s \
sigsuspend.s \
sleep.s \
stat.s \
stime.s \
svrctl.s \
symlink.s \
sync.s \
sysuname.s \
tcdrain.s \
tcflow.s \
tcflush.s \
tcgetattr.s \
tcsendbreak.s \
tcsetattr.s \
time.s \
times.s \
truncate.s \
umask.s \
umount.s \
uname.s \
unlink.s \
utime.s \
wait.s \
waitpid.s \
write.s"

TYPE=both


```
.sect .text
.extern __exit
.define __exit
.align 2

__exit:
    jmp     __exit
```

```
.sect .text
.extern __pm_findproc
.define __pm_findproc
.align 2

__pm_findproc:
    jmp     __pm_findproc
```

```
.sect .text
.extern __access
.define _access
.align 2

_access:
    jmp     __access
```

```
.sect .text
.extern __alarm
.define _alarm
.align 2

_alarm:
    jmp     __alarm
```

```
.sect .text
.extern __sysuname
.define _sysuname
.align 2

_sysuname:
    jmp     __sysuname
```

```
.sect .text  
.extern __brk  
.define _brk  
.align 2
```

```
_brk:  
    jmp    __brk
```

```
.sect .text
.extern __cfgetispeed
.define _cfgetispeed
.align 2

_cfgetispeed:
    jmp     __cfgetispeed
```

```
.sect .text
.extern __cfgetospeed
.define _cfgetospeed
.align 2

_cfgetospeed:
    jmp     __cfgetospeed
```



```
.sect .text
.extern __cfsetispeed
.define _cfsetispeed
.align 2

_cfsetispeed:
    jmp     __cfsetispeed
```

```
.sect .text
.extern __cfsetospeed
.define _cfsetospeed
.align 2

_cfsetospeed:
    jmp     __cfsetospeed
```

```
.sect .text
.extern __chdir
.define _chdir
.extern __fchdir
.define _fchdir
.align 2

_chdir:
    jmp     __chdir
_fchdir:
    jmp     __fchdir
```

```
.sect .text
.extern __chmod
.define _chmod
.align 2

_chmod:
    jmp     __chmod
```

```
.sect .text
.extern __chown
.define _chown
.align 2

_chown:
    jmp     __chown
```

```
.sect .text
.extern __chroot
.define _chroot
.align 2

_chroot:
    jmp     __chroot
```

```
.sect .text
.extern __close
.define _close
.align 2

_close:
    jmp     __close
```

```
.sect .text
.extern __closedir
.define _closedir
.align 2

_closedir:
    jmp     __closedir
```



```
.sect .text
.extern __creat
.define _creat
.align 2

__creat:
    jmp     __creat
```

```
.sect .text
.extern __devctl
.define _devctl
.align 2

_devctl:
    jmp     __devctl
```

```
.sect .text
.extern __dup
.define _dup
.align 2
```

```
_dup:
    jmp     __dup
```

```
.sect .text
.extern __dup2
.define _dup2
.align 2
```

```
_dup2:
    jmp     __dup2
```

```
.sect .text
.extern __execl
.define _execl
.align 2

_execl:
    jmp     __execl
```

```
.sect .text
.extern __execle
.define _execle
.align 2

_execle:
    jmp     __execle
```

```
.sect .text
.extern __execlp
.define _execlp
.align 2

_execlp:
    jmp     __execlp
```

```
.sect .text
.extern __execv
.define _execv
.align 2

_execv:
    jmp     __execv
```



```
.sect .text
.extern __execve
.define _execve
.align 2

_execve:
    jmp     __execve
```

```
.sect .text
.extern __execvp
.define _execvp
.align 2
```

```
_execvp:
    jmp     __execvp
```

```
.sect .text
.extern __fcntl
.define _fcntl
.align 2

_fcntl:
    jmp     __fcntl
```

```
.sect .text
.extern __fork
.define _fork
.align 2

_fork:
    jmp     __fork
```

```
.sect .text
.extern __fpathconf
.define _fpathconf
.align 2

_fpathconf:
    jmp     __fpathconf
```

```
.sect .text
.extern __fstat
.define _fstat
.align 2

_fstat:
    jmp     __fstat
```

```
.sect .text
.extern __fstatfs
.define _fstatfs
.align 2

_fstatfs:
    jmp     __fstatfs
```

```
.sect .text
.extern __getcwd
.define _getcwd
.align 2

_getcwd:
    jmp     __getcwd
```



```
.sect .text
.extern __getegid
.define _getegid
.align 2

_getegid:
    jmp     __getegid
```

```
.sect .text
.extern __geteuid
.define _geteuid
.align 2

_geteuid:
    jmp     __geteuid
```

```
.sect .text
.extern __getgid
.define _getgid
.align 2

_getgid:
    jmp     __getgid
```

```
.sect .text
.extern __getgroups
.define _getgroups
.align 2

_getgroups:
    jmp     __getgroups
```

```
.sect .text
.extern __getnpid
.define _getnpid
.align 2

_getnpid:
    jmp     __getnpid
```

```
.sect .text
.extern __getnprocnr
.define _getnprocnr
.align 2

_getnprocnr:
    jmp     __getnprocnr
```

```
.sect .text
.extern __getpgrp
.define _getpgrp
.align 2

_getpgrp:
    jmp     __getpgrp
```

```
.sect .text
.extern __getpid
.define _getpid
.align 2

_getpid:
    jmp     __getpid
```



```
.sect .text
.extern __getppid
.define _getppid
.align 2

_getppid:
    jmp     __getppid
```

```
.sect .text
.extern __getpprocnr
.define _getpprocnr
.align 2

_getpprocnr:
    jmp     __getpprocnr
```

```
.sect .text
.extern __getprocnr
.define _getprocnr
.align 2

_getprocnr:
    jmp     __getprocnr
```

```
.sect .text
.extern __getsigset
.define _getsigset
.align 2

_getsigset:
    jmp     __getsigset
```

```
.sect .text
.extern __getsysinfo
.define _getsysinfo
.align 2

_getsysinfo:
    jmp     __getsysinfo
```

```
.sect .text
.extern __getuid
.define _getuid
.align 2

_getuid:
    jmp     __getuid
```

```
.sect .text
.extern __ioctl
.define _ioctl
.align 2

__ioctl:
    jmp     __ioctl
```

```
.sect .text
.extern __isatty
.define _isatty
.align 2

_isatty:
    jmp     __isatty
```



```
.sect .text
.extern __kill
.define _kill
.align 2

_kill:
    jmp     __kill
```

```
.sect .text
.extern __link
.define _link
.align 2
```

```
_link:
    jmp     __link
```

```
.sect .text
.extern __lseek
.define _lseek
.align 2

_lseek:
    jmp     __lseek
```

```
.sect .text
.extern __lstat
.define _lstat
.align 2

_lstat:
    jmp     __lstat
```

```
.sect .text
.extern __mkdir
.define _mkdir
.align 2

_mkdir:
    jmp     __mkdir
```

```
.sect .text
.extern __mkfifo
.define _mkfifo
.align 2

_mkfifo:
    jmp     __mkfifo
```

```
.sect .text
.extern __mknod
.define _mknod
.align 2

_mknod:
    jmp     __mknod
```

```
.sect .text
.extern __mount
.define _mount
.align 2

_mount:
    jmp     __mount
```



```
.sect .text
.extern __open
.define _open
.align 2
```

```
_open:
    jmp     __open
```

```
.sect .text
.extern __opendir
.define _opendir
.align 2

_opendir:
    jmp     __opendir
```

```
.sect .text
.extern __pathconf
.define _pathconf
.align 2

_pathconf:
    jmp     __pathconf
```

```
.sect .text
.extern __pause
.define _pause
.align 2

_pause:
    jmp     __pause
```

```
.sect .text
.extern __pipe
.define _pipe
.align 2

_pipe:
    jmp     __pipe
```

```
.sect .text
.extern __ptrace
.define _ptrace
.align 2

_ptrace:
    jmp     __ptrace
```

```
.sect .text
.extern __read
.define _read
.align 2

_read:
    jmp     __read
```

```
.sect .text
.extern __readdir
.define _readdir
.align 2

_readdir:
    jmp     __readdir
```



```
.sect .text
.extern __readlink
.define _readlink
.align 2

_readlink:
    jmp     __readlink
```

```
.sect .text
.extern __reboot
.define _reboot
.align 2

_reboot:
    jmp     __reboot
```

```
.sect .text
.extern __rename
.define _rename
.align 2

_rename:
    jmp     __rename
```

```
.sect .text
.extern __rewinddir
.define _rewinddir
.align 2

_rewinddir:
    jmp     __rewinddir
```

```
.sect .text
.extern __rmdir
.define _rmdir
.align 2

_rmdir:
    jmp     __rmdir
```

```
.sect .text
.extern __sbrk
.define _sbrk
.align 2
```

```
_sbrk:
    jmp     __sbrk
```

```
.sect .text
.extern __seekdir
.define _seekdir
.align 2

_seekdir:
    jmp     __seekdir
```

```
.sect .text
.extern __setgid
.define _setgid
.define _setegid
.align 2

_setgid:
    jmp     __setgid

_setegid:
    jmp     __setegid
```



```
.sect .text
.extern __setuid
.define _setuid
.align 2

_setuid:
    jmp     __setuid
```

```
.sect .text
.extern __setuid
.define _setuid
.define _seteuid
.align 2

_setuid:
    jmp     __setuid

_seteuid:
    jmp     __seteuid
```

```
.sect .text
.extern __sigaction
.define _sigaction
.align 2

_sigaction:
    jmp     __sigaction
```

```
.sect .text
.extern __sigaddset
.define _sigaddset
.align 2

_sigaddset:
    jmp     __sigaddset
```

```
.sect .text
.extern __sigdelset
.define _sigdelset
.align 2

_sigdelset:
    jmp     __sigdelset
```

```
.sect .text
.extern __sigemptyset
.define _sigemptyset
.align 2

_sigemptyset:
    jmp     __sigemptyset
```

```
.sect .text
.extern __sigfillset
.define _sigfillset
.align 2

_sigfillset:
    jmp     __sigfillset
```

```
.sect .text
.extern __sigismember
.define _sigismember
.align 2

_sigismember:
    jmp     __sigismember
```



```
.sect .text
.extern __sigpending
.define _sigpending
.align 2

_sigpending:
    jmp     __sigpending
```

```
.sect .text
.extern __sigprocmask
.define _sigprocmask
.align 2

_sigprocmask:
    jmp     __sigprocmask
```

```
.sect .text
.extern __sigreturn
.define _sigreturn
.align 2

_sigreturn:
    jmp     __sigreturn
```

```
.sect .text
.extern __sigsuspend
.define _sigsuspend
.align 2

_sigsuspend:
    jmp     __sigsuspend
```

```
.sect .text
.extern __sleep
.define _sleep
.align 2

_sleep:
    jmp     __sleep
```

```
.sect .text
.extern __stat
.define _stat
.align 2
```

```
_stat:
    jmp     __stat
```

```
.sect .text
.extern __stime
.define _stime
.align 2

_stime:
    jmp     __stime
```

```
.sect .text
.extern __svrctl
.define _svrctl
.align 2

_svrctl:
    jmp     __svrctl
```



```
.sect .text
.extern __symlink
.define _symlink
.align 2

_symlink:
    jmp     __symlink
```

```
.sect .text
.extern __sync
.define _sync
.align 2
```

```
_sync:
    jmp     __sync
```

```
.sect .text
.extern __tcdrain
.define _tcdrain
.align 2

_tcdrain:
    jmp     __tcdrain
```

```
.sect .text
.extern __tcflow
.define _tcflow
.align 2

_tcflow:
    jmp     __tcflow
```

```
.sect .text
.extern __tcflush
.define _tcflush
.align 2

_tcflush:
    jmp     __tcflush
```

```
.sect .text
.extern __tcgetattr
.define _tcgetattr
.align 2

_tcgetattr:
    jmp     __tcgetattr
```

```
.sect .text
.extern __tcsendbreak
.define _tcsendbreak
.align 2

_tcsendbreak:
    jmp     __tcsendbreak
```

```
.sect .text
.extern __tcsetattr
.define _tcsetattr
.align 2

_tcsetattr:
    jmp     __tcsetattr
```



```
.sect .text
.extern __time
.define _time
.align 2
```

```
_time:
    jmp     __time
```

```
.sect .text
.extern __times
.define _times
.align 2

_times:
    jmp     __times
```

```
.sect .text
.extern __truncate
.extern __ftruncate
.define _truncate
.define _ftruncate
.align 2

_truncate:
    jmp     __truncate

.align 2
_ftruncate:
    jmp     __ftruncate
```

```
.sect .text
.extern __umask
.define _umask
.align 2

_umask:
    jmp     __umask
```

```
.sect .text
.extern __umount
.define _umount
.align 2

_umount:
    jmp     __umount
```

```
.sect .text
.extern __uname
.define _uname
.align 2

_uname:
    jmp     __uname
```

```
.sect .text
.extern __utime
.define _utime
.align 2

_utime:
    jmp     __utime
```

```
.sect .text
.extern __unlink
.define _unlink
.align 2

_unlink:
    jmp     __unlink
```



```
.sect .text
.extern __wait
.define _wait
.align 2
```

```
_wait:
    jmp     __wait
```

```
.sect .text
.extern __waitpid
.define _waitpid
.align 2

_waitpid:
    jmp     __waitpid
```

```
.sect .text
.extern __write
.define _write
.align 2

_write:
    jmp     __write
```

```
.sect .text
.extern __fchmod
.define _fchmod
.align 2

_fchmod:
    jmp     __fchmod
```

```
.sect .text
.extern __fchown
.define _fchown
.align 2

_fchown:
    jmp     __fchown
```

```
# Makefile for lib/syslib.
```

```
CFLAGS="-O -D_MINIX -D_POSIX_SOURCE"
```

```
LIBRARIES=libsys
```

```
libsys_FILES=" \
    assert.c \
    panic.c \
    pci_attr_r16.c \
    pci_attr_r32.c \
    pci_attr_r8.c \
    pci_attr_w16.c \
    pci_attr_w32.c \
    pci_attr_w8.c \
    pci_dev_name.c \
    pci_find_dev.c \
    pci_first_dev.c \
    pci_ids.c \
    pci_init.c \
    pci_init1.c \
    pci_next_dev.c \
    pci_rescan_bus.c \
    pci_reserve.c \
    pci_slot_name.c \
    sys_abort.c \
    sys_endsig.c \
    sys_eniop.c \
    sys_exec.c \
    sys_exit.c \
    sys_fork.c \
    sys_getinfo.c \
    sys_getsig.c \
    sys_in.c \
    sys_int86.c \
    sys_irqctl.c \
    sys_kill.c \
    sys_memset.c \
    sys_newmap.c \
    sys_nice.c \
    sys_out.c \
    sys_physcopy.c \
    sys_sdevio.c \
    sys_segctl.c \
    sys_setalarm.c \
    sys_sigreturn.c \
    sys_sigsend.c \
    sys_privctl.c \
    sys_times.c \
    sys_trace.c \
    sys_umap.c \
    sys_vinb.c \
    sys_vinl.c \
    sys_vinw.c \
    sys_vircopy.c \
    sys_vm_map.c \
    sys_vm_setbuf.c \
    sys_voutb.c \
    sys_voutl.c \
    sys_voutw.c \
    taskcall.c"
```

```
TYPE=both
```

```
/*
 * assert.c - diagnostics
 */

#include <assert.h>
#include <stdio.h>
#include <minix/config.h>
#include <minix/const.h>
#include <minix/sysutil.h>

void __bad_assertion(const char *mess) {
    printf("%s", mess);
    panic(NULL, NULL, NO_NUM);
}
```

```
#include <stdlib.h>
#include <signal.h>
#include <minix/sysutil.h>

#include "syslib.h"

int panicing= 0;

/*=====
 *
 * panic
 *=====*/
PUBLIC void panic(who, mess, num)
char *who;          /* server identification */
char *mess;         /* message format string */
int num;            /* number to go with format string */
{
    /* Something awful has happened. Panics are caused when an internal
     * inconsistency is detected, e.g., a programming error or illegal
     * value of a defined constant.
     */
    message m;
    void (*suicide)(void);

    panicing= 1;
    if (NULL != who && NULL != mess) {
        if (num != NO_NUM) {
            printf("Panic in %s: %s: %d\n", who, mess, num);
        } else {
            printf("Panic in %s: %s\n", who, mess);
        }
    }

    /* Try to signal ourself */
    sys_kill(SELF, SIGKILL);

    /* If exiting nicely through PM fails for some reason, try to
     * commit suicide. E.g., message to PM might fail due to deadlock.
     */
    suicide = (void (*)(void)) -1;
    suicide();

    /* If committing suicide fails for some reason, hang. */
    for(;;) { }
}
```



```
extern int pci_procnr;
```

```
/*
pci_attr_r16.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
*                               pci_attr_r16
*
*=====*/
PUBLIC u16_t pci_attr_r16(devind, port)
int devind;
int port;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_ATTR_R16;
    m.m2_i1= devind;
    m.m2_i2= port;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_attr_r16: can't talk to PCI", r);

    if (m.m_type != 0)
        panic("pci", "pci_attr_r16: got bad reply from PCI", m.m_type);

    return m.m2_l1;
}
```

```
/*
pci_attr_r32.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
* pci_attr_r32
*=====*/
PUBLIC u32_t pci_attr_r32(devind, port)
int devind;
int port;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_ATTR_R32;
    m.m2_i1= devind;
    m.m2_i2= port;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_attr_r32: can't talk to PCI", r);

    if (m.m_type != 0)
        panic("pci", "pci_attr_r32: got bad reply from PCI", m.m_type);

    return m.m2_l1;
}
```

```
/*
pci_attr_r8.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*                               pci_attr_r8                               *
*=====*/
PUBLIC u8_t pci_attr_r8(devind, port)
int devind;
int port;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_ATTR_R8;
    m.m2_i1= devind;
    m.m2_i2= port;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_attr_r8: can't talk to PCI", r);

    if (m.m_type != 0)
        panic("pci", "pci_attr_r8: got bad reply from PCI", m.m_type);

    return m.m2_l1;
}
```

```
/*
pci_attr_w16.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
*                               pci_attr_w16
*=====*/
PUBLIC void pci_attr_w16(devind, port, value)
int devind;
int port;
ul6_t value;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_ATTR_W16;
    m.m2_i1= devind;
    m.m2_i2= port;
    m.m2_l1= value;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_attr_w16: can't talk to PCI", r);

    if (m.m_type != 0)
        panic("pci", "pci_attr_w16: got bad reply from PCI", m.m_type);
}
```

```
/*
pci_attr_w32.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
*                               pci_attr_w32
*=====*/
PUBLIC void pci_attr_w32(devind, port, value)
int devind;
int port;
u32_t value;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_ATTR_W32;
    m.m2_i1= devind;
    m.m2_i2= port;
    m.m2_l1= value;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_attr_w32: can't talk to PCI", r);

    if (m.m_type != 0)
        panic("pci", "pci_attr_w32: got bad reply from PCI", m.m_type);
}
```

```
/*
pci_attr_w8.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
*                               pci_attr_w8
*=====*/
PUBLIC void pci_attr_w8(devind, port, value)
int devind;
int port;
u8_t value;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_ATTR_W8;
    m.m2_i1= devind;
    m.m2_i2= port;
    m.m2_l1= value;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_attr_w8: can't talk to PCI", r);

    if (m.m_type != 0)
        panic("pci", "pci_attr_w8: got bad reply from PCI", m.m_type);
}
```

```
/*
pci_dev_name.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
* pci_dev_name
*=====*/
PUBLIC char *pci_dev_name(vid, did)
ul6_t vid;
ul6_t did;
{
    static char name[80];    /* We need a better interface for this */

    int r;
    message m;

    m.m_type= BUSC_PCI_DEV_NAME;
    m.ml_i1= vid;
    m.ml_i2= did;
    m.ml_i3= sizeof(name);
    m.ml_p1= name;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_dev_name: can't talk to PCI", r);

    if (m.m_type == ENOENT)
    {
        printf("pci_dev_name: got no name\n");
        return NULL;    /* No name for this device */
    }
    if (m.m_type != 0)
        panic("pci", "pci_dev_name: got bad reply from PCI", m.m_type);

    name[sizeof(name)-1]= '\0';    /* Make sure that the string is NUL
                                   * terminated.
                                   */

    #if DEBUG
        printf("pci_dev_name: got name %s\n", name);
    #endif
    return name;
}
```



```
/*
pci_find_dev.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
* pci_find_dev
*=====*/
PUBLIC int pci_find_dev(bus, dev, func, devindp)
u8_t bus;
u8_t dev;
u8_t func;
int *devindp;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_FIND_DEV;
    m.ml_i1= bus;
    m.ml_i2= dev;
    m.ml_i3= func;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_find_dev: can't talk to PCI", r);

    if (m.m_type == 1)
    {
        *devindp= m.ml_i1;
        printf("pci_find_dev: got device %d for %d.%d.%d\n",
            *devindp, bus, dev, func);
        return 1;
    }
    if (m.m_type != 0)
        panic("pci", "pci_find_dev: got bad reply from PCI", m.m_type);

    printf("pci_find_dev: got nothing\n");
    return 0;
}
```

```
/*
pci_first_dev.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
* pci_first_dev
*=====*/
PUBLIC int pci_first_dev(devindp, vidp, didp)
int *devindp;
ul6_t *vidp;
ul6_t *didp;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_FIRST_DEV;
    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_first_dev: can't talk to PCI", r);
    if (m.m_type == 1)
    {
        *devindp= m.ml_i1;
        *vidp= m.ml_i2;
        *didp= m.ml_i3;
    }
    if DEBUG
        printf("pci_first_dev: got device %d, %04x/%04x\n",
            *devindp, *vidp, *didp);
    #endif
    return 1;
}
if (m.m_type != 0)
    panic("pci", "pci_first_dev: got bad reply from PCI", m.m_type);

if DEBUG
    printf("pci_first_dev: got nothing\n");
#endif
return 0;
}
```

```
/*
pci_ids.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
*                               pci_ids
*=====*/
PUBLIC void pci_ids(devind, vidp, didp)
int devind;
ul6_t *vidp;
ul6_t *didp;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_IDS;
    m.ml_i1= devind;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_ids: can't talk to PCI", r);

    if (m.m_type != 0)
        panic("pci", "pci_ids: got bad reply from PCI", m.m_type);
    *vidp= m.ml_i1;
    *didp= m.ml_i2;
    printf("pci_ids: %04x/%04x\n", *vidp, *didp);
}
```

```
/*
pci_init.c
*/

#include "syslib.h"
#include <minix/sysutil.h>

/*=====*
*                               pci_init                               *
*=====*/
PUBLIC void pci_init()
{
    pci_init1("");
}
```

```
/*
pci_init1.c
*/

#include "pci.h"
#include "syslib.h"
#include <string.h>
#include <unistd.h>
#include <minix/sysutil.h>

int pci_procnr= ANY;

/*=====
*
*                               pci_init1
*=====*/
PUBLIC void pci_init1(name)
char *name;
{
    int r;
    size_t len;
    message m;

    r= _pm_findproc("pci", &pci_procnr);
    if (r != 0)
        panic("pci", "pci_init1: _pm_findproc failed for 'pci'", r);

    m.m_type= BUSC_PCI_INIT;
    len= strlen(name);
    if (len+1 <= sizeof(m.m3_cal))
        strcpy(m.m3_cal, name);
    else
    {
        len= sizeof(m.m3_cal)-1;
        memcpy(m.m3_cal, name, len);
        m.m3_cal[len]= '\0';
    }
    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_init1: can't talk to PCI", r);
    if (m.m_type != 0)
        panic("pci", "pci_init1: got bad reply from PCI", m.m_type);
}
```

```
/*
pci_next_dev.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
* pci_next_dev
*=====*/
PUBLIC int pci_next_dev(devindp, vidp, didp)
int *devindp;
ul6_t *vidp;
ul6_t *didp;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_NEXT_DEV;
    m.ml_i1= *devindp;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_next_dev: can't talk to PCI", r);

    if (m.m_type == 1)
    {
        *devindp= m.ml_i1;
        *vidp= m.ml_i2;
        *didp= m.ml_i3;
    }
    #if 0
        printf("pci_next_dev: got device %d, %04x/%04x\n",
            *devindp, *vidp, *didp);
    #endif
    return 1;
}
if (m.m_type != 0)
    panic("pci", "pci_next_dev: got bad reply from PCI", m.m_type);

return 0;
}
```

```
/*
pci_rescan_bus.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
* pci_rescan_bus
*=====*/
PUBLIC void pci_rescan_bus(busnr)
u8_t busnr;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_RESCAN;
    m.ml_il= busnr;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_rescan_bus: can't talk to PCI", r);

    if (m.m_type != 0)
    {
        panic("pci", "pci_rescan_bus: got bad reply from PCI",
            m.m_type);
    }
}
```

```
/*
pci_reserve.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
*                               pci_reserve
*=====*/
PUBLIC void pci_reserve(devind)
int devind;
{
    int r;
    message m;

    m.m_type= BUSC_PCI_RESERVE;
    m.ml_il= devind;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_reserve: can't talk to PCI", r);

    if (m.m_type != 0)
        panic("pci", "pci_reserve: got bad reply from PCI", m.m_type);
}
```



```
/*
pci_slot_name.c
*/

#include "pci.h"
#include "syslib.h"
#include <minix/sysutil.h>

/*=====
*
*                               pci_slot_name
*=====*/
PUBLIC char *pci_slot_name(devind)
int devind;
{
    static char name[80];    /* We need a better interface for this */

    int r;
    message m;

    m.m_type= BUSC_PCI_SLOT_NAME;
    m.ml_i1= devind;
    m.ml_i2= sizeof(name);
    m.ml_p1= name;

    r= sendrec(pci_procnr, &m);
    if (r != 0)
        panic("pci", "pci_slot_name: can't talk to PCI", r);

    if (m.m_type != 0)
        panic("pci", "pci_slot_name: got bad reply from PCI", m.m_type);

    name[sizeof(name)-1]= '\0';    /* Make sure that the string is NUL
                                   * terminated.
                                   */

    printf("pci_slot_name: got name %s\n", name);
    return name;
}
```

```
#include "syslib.h"
#include <stdarg.h>
#include <unistd.h>

PUBLIC int sys_abort(int how, ...)
{
    /* Something awful has happened.  Abandon ship. */

    message m;
    va_list ap;

    va_start(ap, how);
    if ((m.ABRT_HOW = how) == RBT_MONITOR) {
        m.ABRT_MON_ENDPT = va_arg(ap, int);
        m.ABRT_MON_ADDR = va_arg(ap, char *);
        m.ABRT_MON_LEN = va_arg(ap, size_t);
    }
    va_end(ap);

    return(_taskcall(SYSTASK, SYS_ABORT, &m));
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_endsig                               *
 *=====*/
PUBLIC int sys_endsig(proc_nr)
int proc_nr;                               /* process number */
{
    message m;
    int result;

    m.SIG_ENDPT = proc_nr;
    result = _taskcall(SYSTASK, SYS_ENDKSIG, &m);
    return(result);
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_enable_iop                               *
 *=====*/
PUBLIC int sys_enable_iop(proc_nr_e)
int proc_nr_e;                /* number of process to allow I/O */
{
    message m_iop;
    m_iop.IO_ENDPT = proc_nr_e;
    return _taskcall(SYSTASK, SYS_IOPENABLE, &m_iop);
}
```

```
#include "syslib.h"

PUBLIC int sys_exec(proc, ptr, prog_name, initpc)
int proc;                /* process that did exec */
char *ptr;               /* new stack pointer */
char *prog_name;         /* name of the new program */
vir_bytes initpc;
{
    /* A process has exec'd. Tell the kernel. */

    message m;

    m.PR_ENDPT = proc;
    m.PR_STACK_PTR = ptr;
    m.PR_NAME_PTR = prog_name;
    m.PR_IP_PTR = (char *)initpc;
    return(_taskcall(SYSTASK, SYS_EXEC, &m));
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_exit                               *
 *=====*/
PUBLIC int sys_exit(proc)
int proc;                               /* which process has exited */
{
/* A process has exited. PM tells the kernel. In addition this call can be
 * used by system processes to directly exit without passing through the
 * PM. This should be used with care to prevent inconsistent PM tables.
 */
    message m;

    m.PR_ENDPT = proc;
    return(_taskcall(SYSTASK, SYS_EXIT, &m));
}
```

```
#include "syslib.h"

PUBLIC int sys_fork(parent, child, child_endpoint, map_ptr)
int parent;           /* process doing the fork */
int child;            /* which proc has been created by the fork */
int *child_endpoint;
struct mem_map *map_ptr;
{
    /* A process has forked. Tell the kernel. */

    message m;
    int r;

    m.PR_ENDPT = parent;
    m.PR_SLOT = child;
    m.PR_MEM_PTR = map_ptr;
    r = _taskcall(SYSTASK, SYS_FORK, &m);
    *child_endpoint = m.PR_ENDPT;
    return r;
}
```

```
#include "syslib.h"

/*=====
 *
 * sys_getinfo
 *=====*/
PUBLIC int sys_getinfo(request, ptr, len, ptr2, len2)
int request;          /* system info requested */
void *ptr;            /* pointer where to store it */
int len;              /* max length of value to get */
void *ptr2;           /* second pointer */
int len2;             /* length or process nr */
{
    message m;

    m.I_REQUEST = request;
    m.I_ENDPT = SELF;          /* always store values at caller */
    m.I_VAL_PTR = ptr;
    m.I_VAL_LEN = len;
    m.I_VAL_PTR2 = ptr2;
    m.I_VAL_LEN2_E = len2;

    return(_taskcall(SYSTASK, SYS_GETINFO, &m));
}
```



```
#include "syslib.h"

/*=====*
 *                               sys_getksig                               *
 *=====*/
PUBLIC int sys_getksig(k_proc_nr, k_sig_map)
int *k_proc_nr;                /* return process number here */
sigset_t *k_sig_map;          /* return signal map here */
{
    message m;
    int result;

    result = _taskcall(SYSTASK, SYS_GETKSIG, &m);
    *k_proc_nr = m.SIG_ENDPT;
    *k_sig_map = (sigset_t) m.SIG_MAP;
    return(result);
}
```

```
#include "syslib.h"

/*=====*
 *                      sys_in                      *
 *=====*/
PUBLIC int sys_in(port, value, type)
int port;                      /* port address to read from */
unsigned long *value;          /* pointer where to store value */
int type;                      /* byte, word, long */
{
    message m_io;
    int result;

    m_io.DIO_TYPE = type;
    m_io.DIO_REQUEST = DIO_INPUT;
    m_io.DIO_PORT = port;

    result = _taskcall(SYSTASK, SYS_DEVIO, &m_io);
    *value = m_io.DIO_VALUE;
    return(result);
}
```

```
#include "syslib.h"
```

```
/*=====*  
*                               sys_int86                               *  
*=====*/  
PUBLIC int sys_int86(reg86p)  
struct reg86u *reg86p;  
{  
    message m;  
    int result;  
  
    m.m1_p1= (char *)reg86p;  
  
    result = _taskcall(SYSTASK, SYS_INT86, &m);  
    return(result);  
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_irqctl                               *
 *=====*/
PUBLIC int sys_irqctl(req, irq_vec, policy, hook_id)
int req;                               /* IRQ control request */
int irq_vec;                           /* IRQ vector to control */
int policy;                            /* bit mask for policy flags */
int *hook_id;                          /* ID of IRQ hook at kernel */
{
    message m_irq;
    int s;

    m_irq.m_type = SYS_IRQCTL;
    m_irq.IRQ_REQUEST = req;
    m_irq.IRQ_VECTOR = irq_vec;
    m_irq.IRQ_POLICY = policy;
    m_irq.IRQ_HOOK_ID = *hook_id;

    s = _taskcall(SYSTASK, SYS_IRQCTL, &m_irq);
    if (req == IRQ_SETPOLICY) *hook_id = m_irq.IRQ_HOOK_ID;
    return(s);
}
```

```
#include "syslib.h"

PUBLIC int sys_kill(proc, signr)
int proc;          /* which proc has exited */
int signr;         /* signal number: 1 - 16 */
{
    /* A proc has to be signaled via MM. Tell the kernel. */
    message m;

    m.SIG_ENDPT = proc;
    m.SIG_NUMBER = signr;
    return(_taskcall(SYSTASK, SYS_KILL, &m));
}
```

```
#include "syslib.h"

PUBLIC int sys_memset(unsigned long pattern, phys_bytes base, phys_bytes bytes)
{
    /* Zero a block of data. */
    message mess;

    if (bytes == 0L) return(OK);

    mess.MEM_PTR = (char *) base;
    mess.MEM_COUNT = bytes;
    mess.MEM_PATTERN = pattern;

    return(_taskcall(SYSTASK, SYS_MEMSET, &mess));
}
```

```
#include "syslib.h"

PUBLIC int sys_newmap(proc, ptr)
int proc;                /* process whose map is to be changed */
struct mem_map *ptr;     /* pointer to new map */
{
    /* A process has been assigned a new memory map. Tell the kernel. */

    message m;

    m.PR_ENDPT = proc;
    m.PR_MEM_PTR = (char *) ptr;
    return(_taskcall(SYSTASK, SYS_NEWMAP, &m));
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_nice                               *
 *=====*/
PUBLIC int sys_nice(int proc, int prio)
{
    message m;

    m.ml_i1 = proc;
    m.ml_i2 = prio;
    return(_taskcall(SYSTASK, SYS_NICE, &m));
}
```



```
#include "syslib.h"

/*=====*
 *                      sys_out                      *
 *=====*/
PUBLIC int sys_out(port, value, type)
int port;                      /* port address to write to */
unsigned long value;           /* value to write */
int type;                     /* byte, word, long */
{
    message m_io;

    m_io.DIO_TYPE = type;
    m_io.DIO_REQUEST = DIO_OUTPUT;
    m_io.DIO_PORT = port;
    m_io.DIO_VALUE = value;

    return _taskcall(SYSTASK, SYS_DEVIO, &m_io);
}
```

```
#include "syslib.h"

PUBLIC int sys_physcopy(src_proc, src_seg, src_vir,
                        dst_proc, dst_seg, dst_vir, bytes)
int src_proc;           /* source process */
int src_seg;            /* source memory segment */
vir_bytes src_vir;      /* source virtual address */
int dst_proc;           /* destination process */
int dst_seg;            /* destination memory segment */
vir_bytes dst_vir;      /* destination virtual address */
phys_bytes bytes;       /* how many bytes */
{
    /* Transfer a block of data. The source and destination can each either be a
    * process number or SELF (to indicate own process number). Virtual addresses
    * are offsets within LOCAL_SEG (text, stack, data), REMOTE_SEG, or BIOS_SEG.
    * Physicall addressing is also possible with PHYS_SEG.
    */

    message copy_mess;

    if (bytes == 0L) return(OK);
    copy_mess.CP_SRC_ENDPT = src_proc;
    copy_mess.CP_SRC_SPACE = src_seg;
    copy_mess.CP_SRC_ADDR = (long) src_vir;
    copy_mess.CP_DST_ENDPT = dst_proc;
    copy_mess.CP_DST_SPACE = dst_seg;
    copy_mess.CP_DST_ADDR = (long) dst_vir;
    copy_mess.CP_NR_BYTES = (long) bytes;
    return(_taskcall(SYSTASK, SYS_PHYSCOPY, &copy_mess));
}
```

```
#include "syslib.h"

int sys_privctl(int proc, int request, int i, void *p)
{
    message m;

    m.CTL_ENDPT = proc;
    m.CTL_REQUEST = request;
    m.CTL_MM_PRIV = i;
    m.CTL_ARG_PTR = p;

    return _taskcall(SYSTASK, SYS_PRIVCTL, &m);
}
```

```
#include "syslib.h"

/*=====
 *
 * sys_sdevio
 *=====*/
PUBLIC int sys_sdevio(req, port, type, proc_nr, buffer, count)
int req; /* request: DIO_INPUT/ DIO_OUTPUT */
long port; /* port address to read from */
int type; /* byte, word, long */
int proc_nr; /* process where buffer is */
void *buffer; /* pointer to buffer */
int count; /* number of elements */
{
    message m_io;
    int result;

    m_io.DIO_REQUEST = req;
    m_io.DIO_TYPE = type;
    m_io.DIO_PORT = port;
    m_io.DIO_VEC_ENDPT = proc_nr;
    m_io.DIO_VEC_ADDR = buffer;
    m_io.DIO_VEC_SIZE = count;

    return(_taskcall(SYSTASK, SYS_SDEVIO, &m_io));
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_segctl                               *
 *=====*/
PUBLIC int sys_segctl(index, seg, off, phys, size)
int *index;                               /* return index of remote segment */
ul6_t *seg;                               /* return segment selector here */
vir_bytes *off;                          /* return offset in segment here */
phys_bytes phys;                         /* physical address to convert */
vir_bytes size;                          /* size of segment */
{
    message m;
    int s;
    m.SEG_PHYS = phys;
    m.SEG_SIZE = size;
    s = _taskcall(SYSTASK, SYS_SEGCTL, &m);
    *index = (int) m.SEG_INDEX;
    *seg = (ul6_t) m.SEG_SELECT;
    *off = (vir_bytes) m.SEG_OFFSET;
    return s;
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_setalarm                               *
 *=====*/
PUBLIC int sys_setalarm(exp_time, abs_time)
clock_t exp_time;      /* expiration time for the alarm */
int abs_time;          /* use absolute or relative expiration time */
{
    /* Ask the SYSTEM schedule a synchronous alarm for the caller. The process
     * number can be SELF if the caller doesn't know its process number.
     */
    message m;
    m.ALARM_EXP_TIME = exp_time;      /* the expiration time */
    m.ALARM_ABS_TIME = abs_time;      /* time is absolute? */
    return _taskcall(SYSTASK, SYS_SETALARM, &m);
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_sigreturn                               *
 *=====*/
PUBLIC int sys_sigreturn(proc_nr, sig_ctxt)
int proc_nr;                               /* for which process */
struct sigmsg *sig_ctxt;                  /* POSIX style handling */
{
    message m;
    int result;

    m.SIG_ENDPT = proc_nr;
    m.SIG_CTXT_PTR = (char *) sig_ctxt;
    result = _taskcall(SYSTASK, SYS_SIGRETURN, &m);
    return(result);
}
```

```
#include "syslib.h"

/*=====*
 *                      sys_sigsend                      *
 *=====*/
PUBLIC int sys_sigsend(proc_nr, sig_ctxt)
int proc_nr;                      /* for which process */
struct sigmsg *sig_ctxt;         /* POSIX style handling */
{
    message m;
    int result;

    m.SIG_ENDPT = proc_nr;
    m.SIG_CTXT_PTR = (char *) sig_ctxt;
    result = _taskcall(SYSTASK, SYS_SIGSEND, &m);
    return(result);
}
```



```
#include "syslib.h"

PUBLIC int sys_times(proc, ptr)
int proc;          /* proc whose times are needed */
clock_t ptr[5];    /* pointer to time buffer */
{
    /* Fetch the accounting info for a proc. */
    message m;
    int r;

    m.T_ENDPT = proc;
    r = _taskcall(SYSTASK, SYS_TIMES, &m);
    ptr[0] = m.T_USER_TIME;
    ptr[1] = m.T_SYSTEM_TIME;
    ptr[2] = m.T_CHILD_UTIME;
    ptr[3] = m.T_CHILD_STIME;
    ptr[4] = m.T_BOOT_TICKS;
    return(r);
}
```

```
#include "syslib.h"

int sys_svrctl(int proc, int request, int priv, vir_bytes argp)
{
    message m;

    m.CTL_ENDPT = proc;
    m.CTL_REQUEST = request;
    m.CTL_MM_PRIV = priv;
    m.CTL_ARG_PTR = (char *) argp;

    return _taskcall(SYSTASK, SYS_PRIVCTL, &m);
}
```

```
#include "syslib.h"

PUBLIC int sys_trace(req, proc_nr, addr, data_p)
int req, proc_nr;
long addr, *data_p;
{
    message m;
    int r;

    m.CTL_ENDPT = proc_nr;
    m.CTL_REQUEST = req;
    m.CTL_ADDRESS = addr;
    if (data_p) m.CTL_DATA = *data_p;
    r = _taskcall(SYSTASK, SYS_TRACE, &m);
    if (data_p) *data_p = m.CTL_DATA;
    return(r);
}
```

```
#include "syslib.h"

/*=====
 *
 * sys_umap
 *=====*/
PUBLIC int sys_umap(proc_nr, seg, vir_addr, bytes, phys_addr)
int proc_nr; /* process number to do umap for */
int seg; /* T, D, or S segment */
vir_bytes vir_addr; /* address in bytes with segment*/
vir_bytes bytes; /* number of bytes to be copied */
phys_bytes *phys_addr; /* placeholder for result */
{
    message m;
    int result;

    m.CP_SRC_ENDPT = proc_nr;
    m.CP_SRC_SPACE = seg;
    m.CP_SRC_ADDR = vir_addr;
    m.CP_NR_BYTES = bytes;

    result = _taskcall(SYSTASK, SYS_UMAP, &m);
    *phys_addr = m.CP_DST_ADDR;
    return(result);
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_vinb                               *
 *=====*/
PUBLIC int sys_vinb(pvb_pairs, nr_ports)
pvb_pair_t *pvb_pairs;          /* (port,byte-value)-pairs */
int nr_ports;                   /* nr of pairs to be processed */
{
    message m_io;

    m_io.DIO_TYPE = DIO_BYTE;
    m_io.DIO_REQUEST = DIO_INPUT;
    m_io.DIO_VEC_ADDR = (char *) pvb_pairs;
    m_io.DIO_VEC_SIZE = nr_ports;
    return _taskcall(SYSTASK, SYS_VDEVIO, &m_io);
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_vinl                               *
 *=====*/
PUBLIC int sys_vinl(pvl_pairs, nr_ports)
pvl_pair_t *pvl_pairs;          /* (port,long-value)-pairs */
int nr_ports;                   /* nr of pairs to be processed */
{
    message m_io;

    m_io.DIO_TYPE = DIO_LONG;
    m_io.DIO_REQUEST = DIO_INPUT;
    m_io.DIO_VEC_ADDR = (char *) pvl_pairs;
    m_io.DIO_VEC_SIZE = nr_ports;
    return _taskcall(SYSTASK, SYS_VDEVIO, &m_io);
}
```

```
#include "syslib.h"

/*=====*
 *                      sys_vinw                      *
 *=====*/
PUBLIC int sys_vinw(pvw_pairs, nr_ports)
pvw_pair_t *pvw_pairs;          /* (port,word-value)-pairs */
int nr_ports;                  /* nr of pairs to be processed */
{
    message m_io;

    m_io.DIO_TYPE = DIO_WORD;
    m_io.DIO_REQUEST = DIO_INPUT;
    m_io.DIO_VEC_ADDR = (char *) pvw_pairs;
    m_io.DIO_VEC_SIZE = nr_ports;
    return _taskcall(SYSTASK, SYS_VDEVIO, &m_io);
}
```

```
#include "syslib.h"

PUBLIC int sys_vircopy(src_proc, src_seg, src_vir,
                      dst_proc, dst_seg, dst_vir, bytes)
int src_proc;          /* source process */
int src_seg;           /* source memory segment */
vir_bytes src_vir;     /* source virtual address */
int dst_proc;          /* destination process */
int dst_seg;           /* destination memory segment */
vir_bytes dst_vir;     /* destination virtual address */
phys_bytes bytes;     /* how many bytes */
{
    /* Transfer a block of data. The source and destination can each either be a
    * process number or SELF (to indicate own process number). Virtual addresses
    * are offsets within LOCAL_SEG (text, stack, data), REMOTE_SEG, or BIOS_SEG.
    */

    message copy_mess;

    if (bytes == 0L) return(OK);
    copy_mess.CP_SRC_ENDPT = src_proc;
    copy_mess.CP_SRC_SPACE = src_seg;
    copy_mess.CP_SRC_ADDR = (long) src_vir;
    copy_mess.CP_DST_ENDPT = dst_proc;
    copy_mess.CP_DST_SPACE = dst_seg;
    copy_mess.CP_DST_ADDR = (long) dst_vir;
    copy_mess.CP_NR_BYTES = (long) bytes;
    return(_taskcall(SYSTASK, SYS_VIRCOPY, &copy_mess));
}
```



```
#include "syslib.h"

/*=====*
 *                               sys_vm_map                               *
 *=====*/
PUBLIC int sys_vm_map(proc_nr, do_map, base, size, offset)
int proc_nr;
int do_map;
phys_bytes base;
phys_bytes size;
phys_bytes offset;
{
    message m;
    int result;

    m.m4_l1= proc_nr;
    m.m4_l2= do_map;
    m.m4_l3= base;
    m.m4_l4= size;
    m.m4_l5= offset;

    result = _taskcall(SYSTASK, SYS_VM_MAP, &m);
    return(result);
}
```

```
#include "syslib.h"

/*=====*
 *                      sys_vm_setbuf                      *
 *=====*/
PUBLIC int sys_vm_setbuf(base, size, high)
phys_bytes base;
phys_bytes size;
phys_bytes high;
{
    message m;
    int result;

    m.m4_l1= base;
    m.m4_l2= size;
    m.m4_l3= high;

    result = _taskcall(SYSTASK, SYS_VM_SETBUF, &m);
    return(result);
}
```

```
#include "syslib.h"

/*=====*
 *                      sys_voutb                      *
 *=====*/
PUBLIC int sys_voutb(pvb_pairs, nr_ports)
pvb_pair_t *pvb_pairs;          /* (port,byte-value)-pairs */
int nr_ports;                  /* nr of pairs to be processed */
{
    message m_io;
    m_io.DIO_TYPE = DIO_BYTE;
    m_io.DIO_REQUEST = DIO_OUTPUT;
    m_io.DIO_VEC_ADDR = (char *) pvb_pairs;
    m_io.DIO_VEC_SIZE = nr_ports;
    return _taskcall(SYSTASK, SYS_VDEVIO, &m_io);
}
```

```
#include "syslib.h"

/*=====*
 *                               sys_voutl                               *
 *=====*/
PUBLIC int sys_voutl(pvl_pairs, nr_ports)
pvl_pair_t *pvl_pairs;          /* (port,long-value)-pairs */
int nr_ports;                   /* nr of pairs to be processed */
{
    message m_io;

    m_io.DIO_TYPE = DIO_LONG;
    m_io.DIO_REQUEST = DIO_OUTPUT;
    m_io.DIO_VEC_ADDR = (char *) pvl_pairs;
    m_io.DIO_VEC_SIZE = nr_ports;
    return _taskcall(SYSTASK, SYS_VDEVIO, &m_io);
}
```

```
#include "syslib.h"

/*=====*
 *                      sys_voutw                      *
 *=====*/
PUBLIC int sys_voutw(pvw_pairs, nr_ports)
pvw_pair_t *pvw_pairs;          /* (port,word-value)-pairs */
int nr_ports;                  /* nr of pairs to be processed */
{
    message m_io;

    m_io.DIO_TYPE = DIO_WORD;
    m_io.DIO_REQUEST = DIO_OUTPUT;
    m_io.DIO_VEC_ADDR = (char *) pvw_pairs;
    m_io.DIO_VEC_SIZE = nr_ports;
    return _taskcall(SYSTASK, SYS_VDEVIO, &m_io);
}
```

```
/*      syslib.h - System library common definitions.      */
```

```
#define _SYSTEM
```

```
#include <lib.h>
```

```
#include <minix/com.h>
```

```
#include <minix/syslib.h>
```

```
/* _taskcall() is the same as _syscall() except it returns negative error  
 * codes directly and not in errno. This is a better interface for MM and  
 * FS.  
 */
```

```
#include <lib.h>
```

```
#include <minix/syslib.h>
```

```
PUBLIC int _taskcall(who, syscallnr, msgptr)
```

```
int who;
```

```
int syscallnr;
```

```
register message *msgptr;
```

```
{
```

```
    int status;
```

```
    msgptr->m_type = syscallnr;
```

```
    status = _sendrec(who, msgptr);
```

```
    if (status != 0) return(status);
```

```
    return(msgptr->m_type);
```

```
}
```

```
# Makefile for lib/Utils.
```

```
CFLAGS="-O -D_MINIX -D_POSIX_SOURCE"
```

```
LIBRARIES=libsysutil
```

```
libsysutil_FILES=" \  
    kmalloc.c \  
    kprintf.c \  
    kputc.c \  
    tickdelay.c \  
    getuptime.c \  
    env_get_prm.c \  
    env_parse.c \  
    env_panic.c \  
    env_prefix.c \  
    fkey_ctl.c \  
    report.c \  
    taskcall.c"
```

```
TYPE=both
```



```

#include "sysutil.h"
#include <minix/config.h>
#include <string.h>

PRIVATE int argc = 0;
PRIVATE char **argv = NULL;

FORWARD _PROTOTYPE( char *find_key, (const char *params, const char *key));

/*=====
 *                               env_setargs                               *
 *=====*/
PUBLIC void env_setargs(arg_c, arg_v)
int arg_c;
char *arg_v[];
{
    argc= arg_c;
    argv= arg_v;
}

/*=====
 *                               env_get_param                             *
 *=====*/
PUBLIC int env_get_param(key, value, max_len)
char *key;                               /* which key to look up */
char *value;                             /* where to store value */
int max_len;                             /* maximum length of value */
{
    message m;
    static char mon_params[128*sizeof(char)]; /* copy parameters here */
    char *key_value;
    int i, s, keylen;

    if (key == NULL)
        return EINVAL;

    keylen= strlen(key);
    for (i= 1; i<argc; i++)
    {
        if (strncmp(argv[i], key, keylen) != 0)
            continue;
        if (strlen(argv[i]) <= keylen)
            continue;
        if (argv[i][keylen] != '=')
            continue;
        key_value= argv[i]+keylen+1;
        if (strlen(key_value)+1 > EP_BUF_SIZE)
            return(E2BIG);
        strcpy(value, key_value);
        return OK;
    }

    /* Get copy of boot monitor parameters. */
    m.m_type = SYS_GETINFO;
    m.I_REQUEST = GET_MONPARAMS;
    m.I_ENDPT = SELF;
    m.I_VAL_LEN = sizeof(mon_params);
    m.I_VAL_PTR = mon_params;
    if ((s=_taskcall(SYSTASK, SYS_GETINFO, &m)) != OK) {
        printf("SYS_GETINFO: %d (size %u)\n", s, sizeof(mon_params));
        return(s);
    }

    /* We got a copy, now search requested key. */
    if ((key_value = find_key(mon_params, key)) == NULL)
        return(ESRCH);

    /* Value found, make the actual copy (as far as possible). */
    strncpy(value, key_value, max_len);

    /* See if it fits in the client's buffer. */
    if ((strlen(key_value)+1) > max_len) return(E2BIG);
    return(OK);
}

```

```
/*=====*
*                                     find_key                                     *
*=====*/
PRIVATE char *find_key(params,name)
const char *params;
const char *name;
{
    register const char *namep;
    register char *envp;

    for (envp = (char *) params; *envp != 0;) {
        for (namep = name; *namep != 0 && *namep == *envp; namep++, envp++)
            ;
        if (*namep == '\\0' && *envp == '=')
            return(envp + 1);
        while (*envp++ != 0)
            ;
    }
    return(NULL);
}
```

```
#include "sysutil.h"
#include <string.h>

/*=====
 *                               env_panic                               *
 *=====*/
PUBLIC void env_panic(key)
char *key;          /* environment variable whose value is bogus */
{
    static char value[EP_BUF_SIZE] = "<unknown>";
    int s;
    if ((s=env_get_param(key, value, sizeof(value))) == 0) {
        if (s != ESRCH)          /* only error allowed */
            printf("WARNING: get_mon_param() failed in env_panic(): %d\n", s);
    }
    printf("Bad environment setting: '%s = %s'\n", key, value);
    panic("", "", NO_NUM);
}
```

```

#include "sysutil.h"
#include <stdlib.h>
#include <string.h>

/*=====
 *
 *                               env_parse
 *=====*/
PUBLIC int env_parse(env, fmt, field, param, min, max)
char *env;           /* environment variable to inspect */
char *fmt;           /* template to parse it with */
int field;           /* field number of value to return */
long *param;         /* address of parameter to get */
long min, max;       /* minimum and maximum values for the parameter */
{
    /* Parse an environment variable setting, something like "DPETH0=300:3".
     * Panic if the parsing fails. Return EP_UNSET if the environment variable
     * is not set, EP_OFF if it is set to "off", EP_ON if set to "on" or a
     * field is left blank, or EP_SET if a field is given (return value through
     * *param). Punctuation may be used in the environment and format string,
     * fields in the environment string may be empty, and punctuation may be
     * missing to skip fields. The format string contains characters 'd', 'o',
     * 'x' and 'c' to indicate that 10, 8, 16, or 0 is used as the last argument
     * to strtol(). A '*' means that a field should be skipped. If the format
     * string contains something like "\4" then the string is repeated 4 characters
     * to the left.
     */
    char *val, *end;
    char value[EP_BUF_SIZE];
    char PUNCT[] = ".,;:";
    long newpar;
    int s, i, radix, r, keylen;

    if ((s=env_get_param(env, value, sizeof(value))) != 0) {
        if (s == ESRCH) return(EP_UNSET); /* only error allowed */
        printf("WARNING: get_mon_param() failed in env_parse(): %d\n", s);
        return(EP_EGETKENV);
    }
    val = value;
    if (strcmp(val, "off") == 0) return(EP_OFF);
    if (strcmp(val, "on") == 0) return(EP_ON);

    i = 0;
    r = EP_ON;
    for (;;) {
        while (*val == ' ') val++; /* skip spaces */
        if (*val == 0) return(r); /* the proper exit point */
        if (*fmt == 0) break; /* too many values */

        if (strchr(PUNCT, *val) != NULL) {
            /* Time to go to the next field. */
            if (strchr(PUNCT, *fmt) != NULL) i++;
            if (*fmt++ == *val) val++;
            if (*fmt < 32) fmt -= *fmt; /* step back? */
        } else {
            /* Environment contains a value, get it. */
            switch (*fmt) {
                case '*': radix = -1; break;
                case 'd': radix = 10; break;
                case 'o': radix = 010; break;
                case 'x': radix = 0x10; break;
                case 'c': radix = 0; break;
                default: goto badenv;
            }

            if (radix < 0) {
                /* Skip. */
                while (strchr(PUNCT, *val) == NULL) val++;
                continue;
            } else {
                /* A number. */
                newpar = strtol(val, &end, radix);

                if (end == val) break; /* not a number */
            }
        }
    }
}

```

```
        val = end;
    }
    if (i == field) {
        /* The field requested. */
        if (newpar < min || newpar > max) break;
        *param = newpar;
        r = EP_SET;
    }
}
badenv:
    env_panic(env);
}
```

```
#include "sysutil.h"
#include <stdlib.h>
#include <string.h>

/*=====
 *                               env_prefix                               *
 *=====*/
PUBLIC int env_prefix(env, prefix)
char *env;           /* environment variable to inspect */
char *prefix;        /* prefix to test for */
{
    /* An environment setting may be prefixed by a word, usually "pci".
     * Return TRUE if a given prefix is used.
     */
    char value[EP_BUF_SIZE];
    char punct[] = ".,;:";
    int i, s, keylen;
    char *val;
    size_t n;

    if ((s = env_get_param(env, value, sizeof(value))) != 0) {
        if (s != ESRCH) /* only error allowed */
            printf("WARNING: get_mon_param() failed in env_prefix(): %d\n", s);
    }
    n = strlen(prefix);
    return(value != NULL
           && strncmp(value, prefix, n) == 0
           && strchr(punct, value[n]) != NULL);
}
```

```
#include "sysutil.h"

/*=====*
 *                               fkey_ctl                               *
 *=====*/
PUBLIC int fkey_ctl(request, fkeys, sfkeys)
int request;                /* request to perform */
int *fkeys;                 /* bit masks for F1-F12 keys */
int *sfkeys;                /* bit masks for Shift F1-F12 keys */
{
/* Send a message to the TTY server to request notifications for function
 * key presses or to disable notifications. Enabling succeeds unless the key
 * is already bound to another process. Disabling only succeeds if the key is
 * bound to the current process.
 */
    message m;
    int s;
    m.FKEY_REQUEST = request;
    m.FKEY_FKEYS = (fkeys) ? *fkeys : 0;
    m.FKEY_SFKEYS = (sfkeys) ? *sfkeys : 0;
    s = _taskcall(TTY_PROC_NR, FKEY_CONTROL, &m);
    if (fkeys) *fkeys = m.FKEY_FKEYS;
    if (sfkeys) *sfkeys = m.FKEY_SFKEYS;
    return(s);
}
```

```
#include "sysutil.h"

/*=====*
 *                               getuptime                               *
 *=====*/
PUBLIC int getuptime(ticks)
clock_t *ticks;                      /* uptime in ticks */
{
    message m;
    int s;

    m.m_type = SYS_TIMES;             /* request time information */
    m.T_ENDPT = NONE;                 /* ignore process times */
    s = _taskcall(SYSTASK, SYS_TIMES, &m);
    *ticks = m.T_BOOT_TICKS;
    return(s);
}
```



```

/*      malloc(), realloc(), free() - simple memory allocation routines
 *
 * This is a very small and simple minded malloc      Author: Kees J. Bot
 * implementation. Ideal for things like a          29 Jan 1994
 * bootstrap program, or for debugging. Six times
 * slower than any good malloc.
 */
#define nil 0

#define sbrk _sbrk
#include <stddef.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <limits.h>
#if !DEBUG
#define NDEBUG 1
#define debug(expr) ((void) 0)
#else
#define debug(expr) expr
#endif
#include <assert.h>

typedef struct cell {
    size_t      size;          /* Size of a malloc()'ed object. */
#if DEBUG
    unsigned    magic;        /* To recognize a cell. */
#endif
    struct cell *next;        /* Next cell on the free list. */
#if DEBUG
    unsigned    sacred;       /* Don't touch while unallocated. */
#endif
} cell_t;

#if UINT_MAX <= 0xFFFF
#define MAGIC    0x537B
#else
#define MAGIC    0x537BC0D8
#endif

/* Size of the header of an object. */
#define HDR_SIZE    offsetof(cell_t, next)

/* An offset from a cell pointer to a next cell. */
#define offset(cp, size)    ((cell_t *) ((char *) (cp) + (size)))

/* Address of the object in a cell and back. */
#define cell2obj(cp)    ((void *) ((char *) (cp) + HDR_SIZE))
#define obj2cell(op)    ((cell_t *) ((char *) (op) - HDR_SIZE))

/* The free list. */
static cell_t *freelist;

void *malloc(size_t size)
/* Allocate an object of at least the given size. */
{
    cell_t **pcp, *cp;

    size += HDR_SIZE;
    if (size < sizeof(cell_t)) size = sizeof(cell_t);

    /* Align to a word. Use a real malloc if you need better alignment. */
    size = (size + sizeof(int) - 1) & ~(sizeof(int) - 1);

    /* Space for a magic number at the end of the chunk. */
    debug(size += sizeof(unsigned));

    for (;;) {
        /* Do a first fit search. */
        pcp = &freelist;
        while ((cp = *pcp) != nil) {
            cell_t *next = cp->next;

            assert(cp->magic == MAGIC);

```

```

        assert(cp->sacred == MAGIC);

        if (offset(cp, cp->size) == next) {
            /* Join adjacent free cells. */
            assert(next->magic == MAGIC);
            assert(next->sacred == MAGIC);

            cp->size+= next->size;
            cp->next= next->next;

            continue;                /* Try again. */
        }
        if (size <= cp->size) break;    /* Big enough. */

        /* Next cell. */
        pcp= &cp->next;
    }

    if (cp != nil) break;    /* Found a big enough chunk. */

    /* Allocate a new chunk at the break. */
    if ((cp= (cell_t *) sbrk(size)) == (cell_t *) -1) {
        return nil;
    }

    cp->size= size;
    cp->next= nil;
    debug(cp->magic= MAGIC);
    debug(cp->sacred= MAGIC);
    *pcp= cp;
}

/* We've got a cell that is big enough. Can we break it up? */
if (cp->size >= size + sizeof(cell_t)) {
    cell_t *next= offset(cp, size);

    next->size= cp->size - size;
    next->next= cp->next;
    debug(next->magic= MAGIC);
    debug(next->sacred= MAGIC);
    cp->size= size;
    cp->next= next;
}

/* Unchain the cell we've found and return an address in it. */
*pcp= cp->next;
debug(memset(cell2obj(cp), 0xAA, cp->size - HDR_SIZE));
debug(((unsigned *) offset(cp, cp->size))[-1]= MAGIC);

return cell2obj(cp);
}

void free(void *op)
/* Deallocate an object. */
{
    cell_t **prev, *next, *cp;

    if (op == nil) return;        /* Aaargh. */

    cp= obj2cell(op);
    assert(cp->magic == MAGIC);
    assert(((unsigned *) offset(cp, cp->size))[-1] == MAGIC);
    debug(cp->sacred= MAGIC);

    /* Find the spot where the object belongs. */
    prev= &freelist;
    while ((next= *prev) != nil && next < cp) {
        assert(next->magic == MAGIC);
        assert(next->sacred == MAGIC);
        prev= &next->next;
    }

    /* Put the new free cell in the list. */
    *prev= cp;

```

```
    cp->next= next;

#if DEBUG
    /* Check the rest of the list. */
    while (next != nil) {
        assert(next->magic == MAGIC);
        assert(next->sacred == MAGIC);
        next= next->next;
    }
#endif
}

void *realloc(void *op, size_t size)
/* Change the size of an object. Don't bother being smart, just copy it. */
{
    size_t oldsize;
    void *new;

    oldsize= op == nil ? 0 : obj2cell(op)->size - HDR_SIZE;

    new= malloc(size);
    memcpy(new, op, oldsize > size ? size : oldsize);
    free(op);
    return new;
}

/*
* $PchId: malloc.c,v 1.4 1996/02/22 09:15:56 philip Exp $
*/
```

```

/*      printf() - system services printf()
 *
 *
 */
Author: Kees J. Bot
15 Jan 1994

#define nil 0
#include <stdarg.h>
#include <stddef.h>
#include <limits.h>

#define isdigit(c)      ((unsigned) ((c) - '0') < (unsigned) 10)

#if !__STDC__
/* Classic C stuff, ignore. */
void kputc();
int printf(fmt) char *fmt;
#else

/* Printf() uses kputc() to print characters. */
void kputc(int c);

#define count_kputc(c) do { charcount++; kputc(c); } while(0)

int printf(const char *fmt, ...)
#endif
{
    int c, charcount = 0;
    enum { LEFT, RIGHT } adjust;
    enum { LONG, INT } intsize;
    int fill;
    int width, max, len, base;
    static char X2C_tab[] = "0123456789ABCDEF";
    static char x2c_tab[] = "0123456789abcdef";
    char *x2c;
    char *p;
    long i;
    unsigned long u;
    char temp[8 * sizeof(long) / 3 + 2];

    va_list argp;

    va_start(argp, fmt);

    while ((c = *fmt++) != 0) {
        if (c != '%') {
            /* Ordinary character. */
            count_kputc(c);
            continue;
        }

        /* Format specifier of the form:
         *      %[adjust][fill][width][.max]keys
         */
        c = *fmt++;

        adjust = RIGHT;
        if (c == '-') {
            adjust = LEFT;
            c = *fmt++;
        }

        fill = ' ';
        if (c == '0') {
            fill = '0';
            c = *fmt++;
        }

        width = 0;
        if (c == '*') {
            /* Width is specified as an argument, e.g. %*d. */
            width = va_arg(argp, int);
            c = *fmt++;
        } else
        if (isdigit(c)) {
            /* A number tells the width, e.g. %10d. */
            do {

```

```

        width= width * 10 + (c - '0');
    } while (isdigit(c= *fmt++));
}

max= INT_MAX;
if (c == '.') {
    /* Max field length coming up. */
    if ((c= *fmt++) == '*') {
        max= va_arg(argp, int);
        c= *fmt++;
    } else
    if (isdigit(c)) {
        max= 0;
        do {
            max= max * 10 + (c - '0');
        } while (isdigit(c= *fmt++));
    }
}

/* Set a few flags to the default. */
x2c= x2c_tab;
i= 0;
base= 10;
intsize= INT;
if (c == 'l' || c == 'L') {
    /* "Long" key, e.g. %ld. */
    intsize= LONG;
    c= *fmt++;
}
if (c == 0) break;

switch (c) {
    /* Decimal. */
case 'd':
    i= intsize == LONG ? va_arg(argp, long)
                      : va_arg(argp, int);
    u= i < 0 ? -i : i;
    goto int2ascii;

    /* Octal. */
case 'o':
    base= 010;
    goto getint;

    /* Pointer, interpret as %X or %lX. */
case 'p':
    if (sizeof(char *) > sizeof(int)) intsize= LONG;

    /* Hexadecimal. %X prints upper case A-F, not %lx. */
case 'X':
    x2c= X2C_tab;
case 'x':
    base= 0x10;
    goto getint;

    /* Unsigned decimal. */
case 'u':
getint:
    u= intsize == LONG ? va_arg(argp, unsigned long)
                      : va_arg(argp, unsigned int);

int2ascii:
    p= temp + sizeof(temp)-1;
    *p= 0;
    do {
        *--p= x2c[(ptrdiff_t) (u % base)];
    } while ((u /= base) > 0);
    goto string_length;

    /* A character. */
case 'c':
    p= temp;
    *p= va_arg(argp, int);
    len= 1;
    goto string_print;

```

```
        /* Simply a percent. */
    case '%':
        p= temp;
        *p= '%';
        len= 1;
        goto string_print;

        /* A string. The other cases will join in here. */
    case 's':
        p= va_arg(argp, char *);

string_length:
        for (len= 0; p[len] != 0 && len < max; len++) {}

string_print:
        width -= len;
        if (i < 0) width--;
        if (fill == '0' && i < 0) count_kputc('-');
        if (adjust == RIGHT) {
            while (width > 0) { count_kputc(fill); width--; }
        }
        if (fill == ' ' && i < 0) count_kputc('-');
        while (len > 0) { count_kputc((unsigned char) *p++); len--; }
        while (width > 0) { count_kputc(fill); width--; }
        break;

        /* Unrecognized format key, echo it back. */
    default:
        count_kputc('%');
        count_kputc(c);
    }
}

/* Mark the end with a null (should be something else, like -1). */
kputc(0);
va_end(argp);
return charcount;
}

/*
 * $PchId: kprintf.c,v 1.5 1996/04/11 06:59:05 philip Exp $
 */
```

```
/* A server must occasionally print some message. It uses a simple version of
 * printf() found in the system lib that calls kputc() to output characters.
 * Printing is done with a call to the kernel, and not by going through FS.
 *
 * This routine can only be used by servers and device drivers. The kernel
 * must define its own kputc(). Note that the log driver also defines its own
 * kputc() to directly call the TTY instead of going through this library.
 */

#include "sysutil.h"

/*=====
 *                               kputc                               *
 *=====*/
void kputc(c)
int c;
{
    /* Accumulate another character. If 0 or buffer full, print it. */
    static int buf_count;      /* # characters in the buffer */
    static char print_buf[80]; /* output is buffered here */
    message m;

    if ((c == 0 && buf_count > 0) || buf_count == sizeof(print_buf)) {
        int procs[] = OUTPUT_PROCS_ARRAY;
        int p;

        for(p = 0; procs[p] != NONE; p++) {
            /* Send the buffer to this output driver. */
            m.DIAG_BUF_COUNT = buf_count;
            m.DIAG_PRINT_BUF = print_buf;
            m.DIAG_ENDPT = SELF;
            m.m_type = DIAGNOSTICS;
            (void) _sendrec(procs[p], &m);
        }
        buf_count = 0;

        /* If the output fails, e.g., due to an ELOCKED, do not retry output
         * at the FS as if this were a normal user-land printf(). This may
         * result in even worse problems.
         */
    }
    if (c != 0) {
        /* Append a single character to the output buffer. */
        print_buf[buf_count++] = c;
    }
}
```

```
#include "sysutil.h"

/*=====*
 *                               report                               *
 *=====*/
PUBLIC void report(who, mess, num)
char *who;                /* server identification */
char *mess;               /* message format to print */
int num;                  /* number to go with the message */
{
    /* Display a message for a server. */

    if (num != NO_NUM) {
        printf("%s: %s %d\n", who, mess, num);
    } else {
        printf("%s: %s\n", who, mess);
    }
}
```



```
/*      sysutil.h - System library utilities.      */

#define _SYSTEM

#include <lib.h>          /* common to all libraries */
#include <minix/com.h>     /* need task numbers + message types */
#include <minix/syslib.h>  /* need sendrec, _taskcall, etc */
#include <minix/sysutil.h> /* prototypes in this library */
```

```
/* _taskcall() is the same as _syscall() except it returns negative error  
 * codes directly and not in errno. This is a better interface for MM and  
 * FS.  
 */
```

```
#include <lib.h>
```

```
#include <minix/syslib.h>
```

```
PUBLIC int _taskcall(who, syscallnr, msgptr)
```

```
int who;
```

```
int syscallnr;
```

```
register message *msgptr;
```

```
{
```

```
    int status;
```

```
    msgptr->m_type = syscallnr;
```

```
    status = _sendrec(who, msgptr);
```

```
    if (status != 0) return(status);
```

```
    return(msgptr->m_type);
```

```
}
```

```
#include "sysutil.h"
#include <timers.h>

/*=====
 *
 *                                tickdelay
 *=====*/
PUBLIC int tickdelay(ticks)
long ticks;                                /* number of ticks to wait */
{
    /* This function uses the synchronous alarm to delay for a while. This works
    * even if a previous synchronous alarm was scheduled, because the remaining
    * tick of the previous alarm are returned so that it can be rescheduled.
    * Note however that a long tick_delay (longer than the remaining time of the
    * previous) alarm will also delay the previous alarm.
    */
    message m, m_alarm;
    clock_t time_left;
    int s;

    if (ticks <= 0) return;                /* check for robustness */

    m.ALRM_ENDPT = SELF;                   /* SELF means this process nr */
    m.ALRM_EXP_TIME = ticks;               /* request message after ticks */
    m.ALRM_ABS_TIME = 0;                   /* ticks are relative to now */
    s = _taskcall(SYSTASK, SYS_SETALARM, &m);
    if (s != OK) return(s);

    receive(CLOCK, &m_alarm);              /* await synchronous alarm */

    /* Check if we must reschedule the current alarm. */
    if (m.ALRM_TIME_LEFT > 0 && m.ALRM_TIME_LEFT != TMR_NEVER) {
        m.ALRM_EXP_TIME = m.ALRM_TIME_LEFT - ticks;
        if (m.ALRM_EXP_TIME <= 0)
            m.ALRM_EXP_TIME = 1;
        s = _taskcall(SYSTASK, SYS_SETALARM, &m);
    }

    return(s);
}
```

```
# Makefile for lib/tmrslib.
```

```
CFLAGS="-O -D_MINIX -D_POSIX_SOURCE"
```

```
LIBRARIES=libtimers
```

```
libtimers_FILES= " \  
    tmrs_set.c \  
    tmrs_clr.c \  
    tmrs_exp.c "
```

```
TYPE=both
```

```
/* This library provides generic watchdog timer management functionality.  
 * See the comments in <timers.h> for details.  
 */
```

```
#include <timers.h>           /* definitions and function prototypes */  
#define NULL      (void *) 0   /* null-pointer definition */
```

```
#include "timers.h"

/*=====
 *                               tmrs_clrtimer                               *
 *=====*/
clock_t tmrs_clrtimer(tmrs, tp, next_time)
timer_t **tmrs;                /* pointer to timers queue */
timer_t *tp;                   /* timer to be removed */
clock_t *next_time;
{
    /* Deactivate a timer and remove it from the timers queue.
    */
    timer_t **atp;
    struct proc *p;
    clock_t prev_time;

    if(*tmrs)
        prev_time = (*tmrs)->tmr_exp_time;
    else
        prev_time = 0;

    tp->tmr_exp_time = TMR_NEVER;

    for (atp = tmrs; *atp != NULL; atp = &(*atp)->tmr_next) {
        if (*atp == tp) {
            *atp = tp->tmr_next;
            break;
        }
    }

    if(next_time) {
        if(*tmrs)
            *next_time = (*tmrs)->tmr_exp_time;
        else
            *next_time = 0;
    }

    return prev_time;
}
```

```
#include "timers.h"

/*=====*
 *                               tmrs_exptimers                               *
 *=====*/
void tmrs_exptimers(tmrs, now, new_head)
timer_t **tmrs;                /* pointer to timers queue */
clock_t now;                   /* current time */
clock_t *new_head;
{
    /* Use the current time to check the timers queue list for expired timers.
     * Run the watchdog functions for all expired timers and deactivate them.
     * The caller is responsible for scheduling a new alarm if needed.
     */
    timer_t *tp;

    while ((tp = *tmrs) != NULL && tp->tmr_exp_time <= now) {
        *tmrs = tp->tmr_next;
        tp->tmr_exp_time = TMR_NEVER;
        (*tp->tmr_func)(tp);
    }

    if(new_head) {
        if(*tmrs)
            *new_head = (*tmrs)->tmr_exp_time;
        else
            *new_head = 0;
    }
}
```

```
#include "timers.h"

/*=====
 *
 *          tmrs_settimer
 *=====*/
clock_t tmrs_settimer(tmrs, tp, exp_time, watchdog, new_head)
timer_t **tmrs;           /* pointer to timers queue */
timer_t *tp;              /* the timer to be added */
clock_t exp_time;         /* its expiration time */
tmr_func_t watchdog;      /* watchdog function to be run */
clock_t *new_head;        /* new earliest timer, if non NULL */
{
    /* Activate a timer to run function 'fp' at time 'exp_time'. If the timer is
     * already in use it is first removed from the timers queue. Then, it is put
     * in the list of active timers with the first to expire in front.
     * The caller responsible for scheduling a new alarm for the timer if needed.
     */
    timer_t **atp;
    clock_t old_head = 0;

    if(*tmrs)
        old_head = (*tmrs)->tmr_exp_time;

    /* Set the timer's variables. */
    (void) tmrs_clrtimer(tmrs, tp, NULL);
    tp->tmr_exp_time = exp_time;
    tp->tmr_func = watchdog;

    /* Add the timer to the active timers. The next timer due is in front. */
    for (atp = tmrs; *atp != NULL; atp = &(*atp)->tmr_next) {
        if (exp_time < (*atp)->tmr_exp_time) break;
    }
    tp->tmr_next = *atp;
    *atp = tp;
    if(new_head)
        (*new_head) = (*tmrs)->tmr_exp_time;
    return old_head;
}
```



```
# Makefile for lib/util.
```

```
CFLAGS="-O -D_MINIX -D_POSIX_SOURCE"
```

```
LIBRARIES=libutil
```

```
libutil_FILES=openpty.c
```

```
TYPE=both
```

```
/*
 * openpty() tries to open a pty; applications won't have to
 * duplicate this code all the time (or change it if the system
 * pty interface changes).
 *
 * First version by Ben Gras <beng@few.vu.nl>,
 * Initially heavily based on telnetd/pty.c
 * by Fred N. van Kempen, <waltje@uwalnt.nl.mugnet.org>.
 */
#include <libutil.h>
#include <termios.h>
#include <fcntl.h>
#include <grp.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioc_tty.h>

#define DEV_DIR          "/dev"

/*
 * Allocate a PTY, by trying to open one repeatedly.
 */
int openpty(int *amaster, int *aslave, char *name,
            struct termios *termp, struct winsize *winp)
{
    char buff[128], temp[128];
    register int i, j;
    int pty_fd = -1, gr;
    static char tty_name[128];
    struct group *ttygroup;
    gid_t tty_gid = 0;

    if(!amaster || !aslave) {
        errno = EINVAL;
        return -1;
    }

    for(i = 'p'; i < 'w'; i++) {
        j = 0;
        do {
            sprintf(buff, "%s/pty%c%c",
                    DEV_DIR, i, (j < 10) ? j + '0' : j + 'a' - 10);

            if((*amaster = open(buff, O_RDWR)) >= 0) {
                sprintf(tty_name, "%s/tty%c%c", DEV_DIR,
                        i, (j < 10) ? j + '0' : j + 'a' - 10);
                if((*aslave = open(tty_name, O_RDWR)) >= 0)
                    break;
            }
            close(*amaster);

            j++;
            if (j == 16) break;
        } while(1);

        /* Did we find one? */
        if (j < 16) break;
    }
    if (*amaster < 0) { errno = ENOENT; return(-1); }

    setgrent();
    ttygroup = getgrnam("tty");
    endgrent();
    if(ttygroup) tty_gid = ttygroup->gr_gid;

    if(name) strcpy(name, tty_name);

    /* Ignore errors on these. */
}
```

```
chown(tty_name, getuid(), tty_gid);
chmod(tty_name, 0620);          /* -rw--w---- */
if(term) tcsetattr(*aslave, TCSAFLUSH, term);
if(winp) ioctl(*aslave, TIOCSWINSZ, winp);

return(0);
}
```

ChangeLog file for zlib

Changes in 1.2.3 (18 July 2005)

- Apply security vulnerability fixes to contrib/inffast9 as well
- Clean up some text files (carriage returns, trailing space)
- Update testzlib, vstudio, masmx64, and masmx86 in contrib [Vollant]

Changes in 1.2.2.4 (11 July 2005)

- **Add** inflatePrime() function for starting inflation at bit boundary
- Avoid some Visual C warnings in deflate.c
- Avoid more silly Visual C warnings in inflate.c and inftrees.c for 64-bit compile
- Fix some spelling errors in comments [Betts]
- Correct inflateInit2() error return documentation in zlib.h
- **Added** zran.c example of compressed data random access to examples directory, shows use of inflatePrime()
- Fix cast for assignments to strm->state in inflate.c and inffast.c
- Fix zlibCompileFlags() in zutil.c to use 1L for long shifts [Oberhumer]
- Move declarations of gf2 functions to right place in crc32.c [Oberhumer]
- **Add** cast in trees.c to avoid a warning [Oberhumer]
- Avoid some warnings in fitblk.c, gun.c, gzjoin.c in examples [Oberhumer]
- Update make_vms.com [Zinser]
- Initialize state->write in inflateReset() since copied in inflate_fast()
- Be more strict on incomplete code sets in inflate_table() and increase ENOUGH and MAXD -- this repairs a possible security vulnerability for invalid inflate input. Thanks to Tavis Ormandy and Markus Oberhumer for discovering the vulnerability and providing test cases.
- **Add** ia64 support to configure for HP-UX [Smith]
- **Add** error return to gzread() for format or i/o error [Levin]
- Use malloc.h for OS/2 [Necasek]

Changes in 1.2.2.3 (27 May 2005)

- Replace 1U constants in inflate.c and inftrees.c for 64-bit compile
- Typecast fread() return values in gzio.c [Vollant]
- **Remove** trailing space in minigzip.c outmode (VC++ can't deal with it)
- Fix crc check bug in gzread() after gzungetc() [Heiner]
- **Add** the deflateTune() function to adjust internal compression parameters
- **Add** a fast gzip decompressor, gun.c, to examples (use of inflateBack)
- **Remove** an incorrect assertion in examples/zpipe.c
- **Add** C++ wrapper in inffast9.h [Donais]
- Fix bug in inflateCopy() when decoding fixed codes
- Note in zlib.h how much deflateSetDictionary() actually uses
- **Remove** USE_DICT_HEAD in deflate.c (would mess up inflate if used)
- **Add** _WIN32_WCE to define WIN32 in zconf.in.h [Spencer]
- Don't include stderr.h or errno.h for _WIN32_WCE in zutil.h [Spencer]
- **Add** gzdirect() function to indicate transparent reads
- Update contrib/minizip [Vollant]
- Fix compilation of deflate.c when both ASMV and FASTEST [Oberhumer]
- **Add** casts in crc32.c to avoid warnings [Oberhumer]
- **Add** contrib/masmx64 [Vollant]
- Update contrib/asm586, asm686, masmx86, testzlib, vstudio [Vollant]

Changes in 1.2.2.2 (30 December 2004)

- Replace structure assignments in deflate.c and inflate.c with memcpy to avoid implicit memcpy calls (portability for no-library compilation)
- Increase sprintf() buffer size in gzopen() to allow for large numbers
- **Add** INFLATE_STRICT to check distances against zlib header
- Improve WinCE errno handling and comments [Chang]
- **Remove** comment about no gzip header processing in FAQ
- **Add** Z_FIXED strategy option to deflateInit2() to force fixed trees
- **Add** updated make_vms.com [Coghlan], update README
- Create a new "examples" directory, move gzappend.c there, **add** zpipe.c, fitblk.c, gzlog.[ch], gzjoin.c, and zlib_how.html.
- **Add** FAQ entry and comments in deflate.c on uninitialized memory access
- **Add** Solaris 9 make options in configure [Gilbert]
- Allow strerror() usage in gzio.c for STDC
- Fix DecompressBuf in contrib/delphi/ZLib.pas [ManChesTer]
- Update contrib/masmx86/inffast32.asm and gvmat32.asm [Vollant]
- Use z_off_t for adler32_combine() and crc32_combine() lengths
- Make adler32() much faster for small len
- Use OS_CODE in deflate() default gzip header

Changes in 1.2.2.1 (31 October 2004)

- Allow inflateSetDictionary() call for raw inflate
- Fix inflate header crc check bug for file names and comments
- **Add** deflateSetHeader() and gz_header structure for custom gzip headers
- **Add** inflateGetHeader() to retrieve gzip headers
- **Add** crc32_combine() and Adler32_combine() functions
- **Add** alloc_func, free_func, in_func, out_func to Z_PREFIX list
- Use zstreamp consistently in zlib.h (inflate_back functions)
- **Remove** GUNZIP condition from definition of inflate_mode in inflate.h and in contrib/inflate86/inffast.S [Truta, Anderson]
- **Add** support for AMD64 in contrib/inflate86/inffas86.c [Anderson]
- Update projects/README.projects and projects/visualc6 [Truta]
- Update win32/DLL_FAQ.txt [Truta]
- Avoid warning under NO_GZCOMPRESS in gzio.c; fix typo [Truta]
- Deprecate Z_ASCII; use Z_TEXT instead [Truta]
- Use a new algorithm for setting strm->data_type in trees.c [Truta]
- Do not define an exit() prototype in zutil.c unless DEBUG defined
- **Remove** prototype of exit() from zutil.c, example.c, minigzip.c [Truta]
- **Add** comment in zlib.h for Z_NO_FLUSH parameter to deflate()
- Fix Darwin build version identification [Peterson]

Changes in 1.2.2 (3 October 2004)

- Update zlib.h comments on gzip in-memory processing
- Set adler to 1 in inflateReset() to support Java test suite [Walles]
- **Add** contrib/dotzlib [Ravn]
- Update win32/DLL_FAQ.txt [Truta]
- Update contrib/minizip [Vollant]
- Move contrib/visual-basic.txt to old/ [Truta]
- Fix assembler builds in projects/visualc6/ [Truta]

Changes in 1.2.1.2 (9 September 2004)

- Update INDEX file
- Fix trees.c to update strm->data_type (no one ever noticed!)
- Fix bug in error case in inflate.c, inffast.c, and inffast9.c [Brown]
- **Add** "volatile" to crc table flag declaration (for DYNAMIC_CRC_TABLE)
- **Add** limited multitasking protection to DYNAMIC_CRC_TABLE
- **Add** NO_vsnprintf for VMS in zutil.h [Mozilla]
- Don't declare strerror() under VMS [Mozilla]
- **Add** comment to DYNAMIC_CRC_TABLE to use get_crc_table() to initialize
- Update contrib/ada [Anisimkov]
- Update contrib/minizip [Vollant]
- Fix configure to not hardcode directories for Darwin [Peterson]
- Fix gzio.c to not return error on empty files [Brown]
- Fix indentation; update version in contrib/delphi/ZLib.pas and contrib/pascal/zlibpas.pas [Truta]
- Update mkasm.bat in contrib/masmx86 [Truta]
- Update contrib/untgz [Truta]
- **Add** projects/README.projects [Truta]
- **Add** project for MS Visual C++ 6.0 in projects/visualc6 [Cadieux, Truta]
- Update win32/DLL_FAQ.txt [Truta]
- Update list of Z_PREFIX symbols in zconf.h [Randers-Pehrson, Truta]
- **Remove** an unnecessary assignment to curr in inftrees.c [Truta]
- **Add** OS/2 to exe builds in configure [Poltorak]
- **Remove** err dummy parameter in zlib.h [Kientzle]

Changes in 1.2.1.1 (9 January 2004)

- Update email address in README
- Several FAQ updates
- Fix a big fat bug in inftrees.c that prevented decoding valid dynamic blocks with only literals and no distance codes -- Thanks to "Hot Emu" for the bug report and sample file
- **Add** a note to puff.c on no distance codes case.

Changes in 1.2.1 (17 November 2003)

- **Remove** a tab in contrib/gzappend/gzappend.c
- Update some interfaces in contrib for new zlib functions
- Update zlib version number in some contrib entries
- **Add** Windows CE definition for ptrdiff_t in zutil.h [Mai, Truta]
- Support shared libraries on Hurd and KFreeBSD [Brown]
- Fix error in NO_DIVIDE option of Adler32.c

Changes in 1.2.0.8 (4 November 2003)

- Update version in contrib/delphi/ZLib.pas and contrib/pascal/zlibpas.pas
- **Add** experimental NO_DIVIDE #define in Adler32.c
 - Possibly faster on some processors (let me know if it is)

- Correct Z_BLOCK to not return on first inflate call if no wrap
- Fix strm->data_type on inflate() return to correctly indicate EOB
- **Add** deflatePrime() function for appending in the middle of a byte
- **Add** contrib/gzappend for an example of appending to a stream
- Update win32/DLL_FAQ.txt [Truta]
- Delete Turbo C comment in README [Truta]
- Improve some indentation in zconf.h [Truta]
- Fix infinite loop on bad input in configure script [Church]
- Fix gzeof() for concatenated gzip files [Johnson]
- **Add** example to contrib/visual-basic.txt [Michael B.]
- **Add** -p to mkdir's in Makefile.in [vda]
- Fix configure to properly detect presence or lack of printf functions
- **Add** AS400 support [Monnerat]
- **Add** a little Cygwin support [Wilson]

Changes in 1.2.0.7 (21 September 2003)

- Correct some debug formats in contrib/inffback9
- Cast a type in a debug statement in trees.c
- Change search and replace delimiter in configure from % to # [Beebe]
- Update contrib/untgz to 0.2 with various fixes [Truta]
- **Add** build support for Amiga [Nikl]
- **Remove** some directories in old that have been updated to 1.2
- **Add** dylib building for Mac OS X in configure and Makefile.in
- **Remove** old distribution stuff from Makefile
- Update README to point to DLL_FAQ.txt, and **add** comment on Mac OS X
- Update links in README

Changes in 1.2.0.6 (13 September 2003)

- Minor FAQ updates
- Update contrib/minizip to 1.00 [Vollant]
- **Remove** test of gz functions in example.c when GZ_COMPRESS defined [Truta]
- Update POSTINC comment for 68060 [Nikl]
- **Add** contrib/inffback9 with deflate64 decoding (unsupported)
- For MVS define NO_vsnprintf and undefine FAR [van Burik]
- **Add** pragma for fdopen on MVS [van Burik]

Changes in 1.2.0.5 (8 September 2003)

- **Add** OF to inflateBackEnd() declaration in zlib.h
- Remember start when using gzopen in the middle of a file
- Use internal off_t counters in gz* **functions** to properly handle seeks
- Perform more rigorous check for distance-too-far in inffast.c
- **Add** Z_BLOCK flush option to return from inflate at block boundary
- Set strm->data_type on return from inflate
 - Indicate bits unused, if at block boundary, and if in last block
- Replace size_t with ptrdiff_t in crc32.c, and check for correct size
- **Add** condition so old NO_DEFLATE define still works for compatibility
- FAQ update regarding the Windows DLL [Truta]
- INDEX update: **add** qnx entry, **remove** aix entry [Truta]
- Install zlib.3 into mandir [Wilson]
- Move contrib/zlib_dll_FAQ.txt to win32/DLL_FAQ.txt; update [Truta]
- Adapt the zlib interface to the new DLL convention guidelines [Truta]
- Introduce ZLIB_WINAPI macro to allow the export of functions using the WINAPI calling convention, for Visual Basic [Vollant, Truta]
- Update msdos and win32 scripts and makefiles [Truta]
- Export symbols by name, not by ordinal, in win32/zlib.def [Truta]
- **Add** contrib/ada [Anisimkov]
- Move asm files from contrib/vstudio/vc70_32 to contrib/asm386 [Truta]
- Rename contrib/asm386 to contrib/masmx86 [Truta, Vollant]
- **Add** contrib/masm686 [Truta]
- Fix offsets in contrib/inflate86 and contrib/masmx86/inffas32.asm [Truta, Vollant]
- Update contrib/delphi; rename to contrib/pascal; **add** example [Truta]
- **Remove** contrib/delphi2; **add** a new contrib/delphi [Truta]
- Avoid inclusion of the nonstandard <memory.h> in contrib/iostream, and fix some method prototypes [Truta]
- Fix the ZCR_SEED2 constant to avoid warnings in contrib/minizip [Truta]
- Avoid the use of backslash (\) in contrib/minizip [Vollant]
- Fix file time handling in contrib/untgz; update makefiles [Truta]
- Update contrib/vstudio/vc70_32 to comply with the new DLL guidelines [Vollant]
- **Remove** contrib/vstudio/vc15_16 [Vollant]
- Rename contrib/vstudio/vc70_32 to contrib/vstudio/vc7 [Truta]
- Update README.contrib [Truta]

- Invert the assignment order of match_head and s->prev[...] in INSERT_STRING [Truta]
- Compare TOO_FAR with 32767 instead of 32768, to avoid 16-bit warnings [Truta]
- Compare function pointers with 0, not with NULL or Z_NULL [Truta]
- Fix prototype of syncsearch in inflate.c [Truta]
- Introduce ASMINF macro to be enabled when using an ASM implementation of inflate_fast [Truta]
- Change NO_DEFLATE to NO_GZCOMPRESS [Truta]
- Modify test_gzio in example.c to take a single file name as a parameter [Truta]
- Exit the example.c program if gzopen fails [Truta]
- **Add** type casts around strlen in example.c [Truta]
- **Remove** casting to sizeof in minigzip.c; give a proper type to the variable compared with SUFFIX_LEN [Truta]
- Update definitions of STDC and STDC99 in zconf.h [Truta]
- Synchronize zconf.h with the new Windows DLL interface [Truta]
- Use SYS16BIT instead of __32BIT__ to distinguish between 16- and 32-bit platforms [Truta]
- Use far memory allocators in small 16-bit memory models for Turbo C [Truta]
- **Add** info about the use of ASMV, ASMINF and ZLIB_WINAPI in zlibCompileFlags [Truta]
- Cygwin has vsnprintf [Wilson]
- In Windows16, OS_CODE is 0, as in MSDOS [Truta]
- In Cygwin, OS_CODE is 3 (Unix), not 11 (Windows32) [Wilson]

Changes in 1.2.0.4 (10 August 2003)

- Minor FAQ updates
- Be more strict when checking inflateInit2's windowBits parameter
- Change NO_GUNZIP compile option to NO_GZIP to cover deflate as well
- **Add** gzip wrapper option to deflateInit2 using windowBits
- **Add** updated QNX rule in configure and qnx directory [Bonnefoy]
- Make inflate distance-too-far checks more rigorous
- Clean up FAR usage in inflate
- **Add** casting to sizeof() in gzio.c and minigzip.c

Changes in 1.2.0.3 (19 July 2003)

- Fix silly error in gzungetc() implementation [Vollant]
- Update contrib/minizip and contrib/vstudio [Vollant]
- Fix printf format in example.c
- Correct cdecl support in zconf.in.h [Anisimkov]
- Minor FAQ updates

Changes in 1.2.0.2 (13 July 2003)

- **Add** ZLIB_VERNUM in zlib.h for numerical preprocessor comparisons
- Attempt to avoid warnings in crc32.c for pointer-int conversion
- **Add** AIX to configure, **remove** aix directory [Bakker]
- **Add** some casts to minigzip.c
- Improve checking after insecure sprintf() or vsprintf() calls
- **Remove** #elif's from crc32.c
- Change leave label to inf_leave in inflate.c and infback.c to avoid library conflicts
- **Remove** inflate gzip decoding by default--only enable gzip decoding by special request for stricter backward compatibility
- **Add** zlibCompileFlags() function to return compilation information
- More typecasting in deflate.c to avoid warnings
- **Remove** leading underscore from _Capital #defines [Truta]
- Fix configure to link shared library when testing
- **Add** some Windows CE target adjustments [Mail]
- **Remove** #define ZLIB_DLL in zconf.h [Vollant]
- **Add** zlib.3 [Rodgers]
- Update RFC URL in deflate.c and algorithm.txt [Mail]
- **Add** zlib_dll_FAQ.txt to contrib [Truta]
- **Add** UL to some constants [Truta]
- Update minizip and vstudio [Vollant]
- **Remove** vestigial NEED_DUMMY_RETURN from zconf.in.h
- Expand use of NO_DUMMY_DECL to avoid all dummy structures
- **Added** iostream3 to contrib [Schwardt]
- Replace rewind() with fseek() for WinCE [Truta]
- Improve setting of zlib format compression level flags
 - Report 0 for huffman and rle strategies and for level == 0 or 1
 - Report 2 only for level == 6
- Only deal with 64K limit when necessary at compile time [Truta]

- Allow TOO_FAR check to be turned off at compile time [Truta]
- **Add** gzclearerr() function [Souza]
- **Add** gzungetc() function

Changes in 1.2.0.1 (17 March 2003)

- **Add** Z_RLE strategy for run-length encoding [Truta]
 - When Z_RLE requested, restrict matches to distance one
 - Update zlib.h, minigzip.c, gzopen(), gzdopen() for Z_RLE
- Correct FASTEST compilation to allow level == 0
- Clean up what gets compiled for FASTEST
- Incorporate changes to zconf.in.h [Vollant]
 - Refine detection of Turbo C need for dummy returns
 - Refine ZLIB_DLL compilation
 - Include additional header file on VMS for off_t typedef
- Try to use _vsnprintf where it supplants vsprintf [Vollant]
- **Add** some casts in inffast.c
- Enhance comments in zlib.h on what happens if gzprintf() tries to write more than 4095 bytes before compression
- **Remove** unused state from inflateBackEnd()
- **Remove** exit(0) from minigzip.c, example.c
- Get rid of all those darn tabs
- **Add** "check" target to Makefile.in that does the same thing as "test"
- **Add** "mostlyclean" and "maintainer-clean" targets to Makefile.in
- Update contrib/inflate86 [Anderson]
- Update contrib/testzlib, contrib/vstudio, contrib/minizip [Vollant]
- **Add** msdos and win32 directories with makefiles [Truta]
- More additions and improvements to the FAQ

Changes in 1.2.0 (9 March 2003)

- New and improved inflate code
 - About 20% faster
 - Does not allocate 32K window unless and until needed
 - Automatically detects and decompresses gzip streams
 - Raw inflate no longer needs an extra dummy byte at end
 - **Added** inflateBack functions using a callback interface--even faster than inflate, useful for file utilities (gzip, zip)
 - **Added** inflateCopy() function to record state for random access on externally generated deflate streams (e.g. in gzip files)
 - More readable code (I hope)
- New and improved crc32()
 - About 50% faster, thanks to suggestions from Rodney Brown
- **Add** deflateBound() and compressBound() functions
- Fix memory leak in deflateInit2()
- Permit setting dictionary for raw deflate (for parallel deflate)
- Fix const declaration for gzwrite()
- Check for some malloc() failures in gzio.c
- Fix bug in gzopen() on single-byte file 0x1f
- Fix bug in gzread() on concatenated file with 0x1f at end of buffer and next buffer doesn't start with 0x8b
- Fix uncompress() to return Z_DATA_ERROR on truncated input
- Free memory at end of example.c
- **Remove** MAX #define in trees.c (conflicted with some libraries)
- Fix static const's in deflate.c, gzio.c, and zutil.[ch]
- Declare malloc() and free() in gzio.c if STDC not defined
- Use malloc() instead of calloc() in zutil.c if int big enough
- Define STDC for AIX
- **Add** aix/ with approach for compiling shared library on AIX
- **Add** HP-UX support for shared libraries in configure
- **Add** OpenUNIX support for shared libraries in configure
- Use \$cc instead of gcc to build shared library
- Make prefix directory if needed when installing
- Correct Macintosh avoidance of typedef Byte in zconf.h
- Correct Turbo C memory allocation when under Linux
- Use libz.a instead of -lz in Makefile (assure use of compiled library)
- Update configure to check for snprintf or vsnprintf functions and their return value, warn during make if using an insecure function
- Fix configure problem with compile-time knowledge of HAVE_UNISTD_H that is lost when library is used--resolution is to build new zconf.h
- Documentation improvements (in zlib.h):
 - Document raw deflate and inflate
 - Update RFCs URL
 - Point out that zlib and gzip formats are different
 - Note that Z_BUF_ERROR is not fatal
 - Document string limit for gzprintf() and possible buffer overflow

- Note requirement on avail_out when flushing
- Note permitted values of flush parameter of inflate()
- **Add** some FAQs (and even answers) to the FAQ
- **Add** contrib/inflate86/ for x86 faster inflate
- **Add** contrib/blast/ for PKWare Data Compression Library decompression
- **Add** contrib/puff/ simple inflate for deflate format description

Changes in 1.1.4 (11 March 2002)

- ZFREE was repeated on same allocation on some error conditions.
This creates a security problem described in
<http://www.zlib.org/advisory-2002-03-11.txt>
- Returned incorrect error (Z_MEM_ERROR) on some invalid data
- Avoid accesses before window for invalid distances with inflate window less than 32K.
- force windowBits > 8 to avoid a bug in the encoder for a window size of 256 bytes. (A complete fix will be available in 1.1.5).

Changes in 1.1.3 (9 July 1998)

- fix "an inflate input buffer bug that shows up on rare but persistent occasions" (Mark)
- fix gzread and gztell for concatenated .gz files (Didier Le Botlan)
- fix gzseek(..., SEEK_SET) in write mode
- fix crc check after a gzseek (Frank Faubert)
- fix miniunzip when the last entry in a zip file is itself a zip file (J Lillge)
- **add** contrib/asm586 and contrib/asm686 (Brian Raiter)
See <http://www.muppetlabs.com/~breadbox/software/assembly.html>
- **add** support for Delphi 3 in contrib/delphi (Bob Dellaca)
- **add** support for C++Builder 3 and Delphi 3 in contrib/delphi2 (Davide Moretti)
- do not exit prematurely in untgz if 0 at start of block (Magnus Holmgren)
- use macro EXTERN instead of extern to support DLL for BeOS (Sander Stoks)
- **added** a FAQ file
- Support gzopen on Mac with Metrowerks (Jason Linhart)
- Do not redefine Byte on Mac (Brad Pettit & Jason Linhart)
- define SEEK_END too if SEEK_SET is not defined (Albert Chin-A-Young)
- avoid some warnings with Borland C (Tom Tanner)
- fix a problem in contrib/minizip/zip.c for 16-bit MSDOS (Gilles Vollant)
- emulate utime() for WIN32 in contrib/untgz (Gilles Vollant)
- allow several arguments to configure (Tim Mooney, Frodo Looijaard)
- use libdir and includedir in Makefile.in (Tim Mooney)
- support shared libraries on OSF1 V4 (Tim Mooney)
- **remove** so_locations in "make clean" (Tim Mooney)
- fix maketree.c compilation error (Glenn, Mark)
- Python interface to zlib now in Python 1.5 (Jeremy Hylton)
- new Makefile.riscos (Rich Walker)
- initialize static descriptors in trees.c for embedded targets (Nick Smith)
- use "foo-gz" in example.c for RISCOS and VMS (Nick Smith)
- **add** the OS/2 files in Makefile.in too (Andrew Zabolotny)
- fix fdopen and halloc macros for Microsoft C 6.0 (Tom Lane)
- fix maketree.c to allow clean compilation of inffixed.h (Mark)
- fix parameter check in deflateCopy (Gunther Nikl)
- cleanup trees.c, use compressed_len only in debug mode (Christian Spieler)
- Many portability patches by Christian Spieler:
 - . zutil.c, zutil.h: **added** "const" for zmem*
 - . Make_vms.com: fixed some typos
 - . Make_vms.com: msdos/Makefile.*: **removed** zutil.h from some dependency lists
 - . msdos/Makefile.msc: **remove** "default rtl link library" info from obj files
 - . msdos/Makefile.*: use model-dependent name for the built zlib library
 - . msdos/Makefile.emx, nt/Makefile.emx, nt/Makefile.gcc:
 - new makefiles, for emx (DOS/OS2), emx&rsxnt and mingw32 (Windows 9x/NT)
- use define instead of typedef for Bytef also for MSC small/medium (Tom Lane)
- replace __far with _far for better portability (Christian Spieler, Tom Lane)
- fix test for errno.h in configure (Tim Newsham)

Changes in 1.1.2 (19 March 98)

- **added** contrib/minzip, mini zip and unzip based on zlib (Gilles Vollant)
See <http://www.winimage.com/zLibDll/unzip.html>
- preinitialize the inflate tables for fixed codes, to make the code completely thread safe (Mark)
- some simplifications and slight speed-up to the inflate code (Mark)
- fix gzeof on non-compressed files (Allan Schrum)
- **add** -stdl option in configure for OSF1 to fix gzprintf (Martin Mokrejs)
- use default value of 4K for Z_BUFSIZE for 16-bit MSDOS (Tim Wegner + Glenn)

- **added** os2/Makefile.def and os2/zlib.def (Andrew Zabolotny)
- **add** shared lib support for UNIX_SV4.2MP (MATSUURA Takanori)
- do not wrap extern "C" around system includes (Tom Lane)
- mention zlib binding for TCL in README (Andreas Kupries)
- **added** amiga/Makefile.pup for Amiga powerUP SAS/C PPC (Andreas Kleinert)
- allow "make install prefix=..." even after configure (Glenn Randers-Pehrson)
- allow "configure --prefix \$HOME" (Tim Mooney)
- **remove** warnings in example.c and gzio.c (Glenn Randers-Pehrson)
- move Makefile.sas to amiga/Makefile.sas

Changes in 1.1.1 (27 Feb 98)

- fix macros `_tr_tally_*` in `deflate.h` for debug mode (Glenn Randers-Pehrson)
- **remove** block truncation heuristic which had very marginal effect for zlib (smaller `lit_bufsize` than in `gzip 1.2.4`) and degraded a little the compression ratio on some files. This also allows inlining `_tr_tally` for matches in `deflate_slow`.
- **added** `msdos/Makefile.w32` for WIN32 Microsoft Visual C++ (Bob Frazier)

Changes in 1.1.0 (24 Feb 98)

- do not return `STREAM_END` prematurely in `inflate` (John Bowler)
- revert to the `zlib 1.0.8` `inflate` to avoid the `gcc 2.8.0` bug (Jeremy Buhler)
- compile with `-DFASTEST` to get compression code optimized for speed only
- in `minigzip`, try `mmap`'ing the input file first (Miguel Albrecht)
- increase size of I/O buffers in `minigzip.c` and `gzio.c` (not a big gain on Sun but significant on HP)
- **add** a pointer to experimental `unzip` library in `README` (Gilles Vollant)
- initialize variable `gcc` in `configure` (Chris Herborth)

Changes in 1.0.9 (17 Feb 1998)

- **added** `gzputs` and `gzgets` functions
- do not clear eof flag in `gzseek` (Mark Diekhans)
- fix `gzseek` for files in transparent mode (Mark Diekhans)
- do not assume that `vsprintf` returns the number of bytes written (Jens Krinke)
- replace `EXPORT` with `ZEXPORT` to avoid conflict with other programs
- **added** `compress2` in `zconf.h`, `zlib.def`, `zlib.dnt`
- new asm code from Gilles Vollant in `contrib/asm386`
- simplify the `inflate` code (Mark):
 - . Replace `ZALLOC`'s in `huft_build()` with single `ZALLOC` in `inflate_blocks_new()`
 - . `ZALLOC` the length list in `inflate_trees_fixed()` instead of using stack
 - . `ZALLOC` the value area for `huft_build()` instead of using stack
 - . Simplify `Z_FINISH` check in `inflate()`
- Avoid `gcc 2.8.0` comparison bug a little differently than `zlib 1.0.8`
- in `inftrees.c`, avoid `cc -O` bug on HP (Farshid Elahi)
- in `zconf.h` move the `ZLIB_DLL` stuff earlier to avoid problems with the declaration of `FAR` (Gilles Vollant)
- install `libz.so*` **with** mode 755 (executable) instead of 644 (Marc Lehmann)
- `read_buf` buf parameter of type `Bytef*` **instead** of `charf*`
- `zmemcpy` parameters are of type `Bytef*`, not `charf*` (Joseph Strout)
- do not redeclare `unlink` in `minigzip.c` for WIN32 (John Bowler)
- fix check for presence of directories in "make install" (Ian Willis)

Changes in 1.0.8 (27 Jan 1998)

- fixed offsets in `contrib/asm386/gvmat32.asm` (Gilles Vollant)
- fix `gzgetc` and `gzputc` for big endian systems (Markus Oberhumer)
- **added** `compress2()` to allow setting the compression level
- include `sys/types.h` to get `off_t` on some systems (Marc Lehmann & QingLong)
- use constant arrays for the static trees in `trees.c` instead of computing them at run time (thanks to Ken Raeburn for this suggestion). To create `trees.h`, compile with `GEN_TREES_H` and run "make test".
- check return code of `example` in "make test" and display result
- pass `minigzip` command line options to `file_compress`
- simplifying code of `inflateSync` to avoid `gcc 2.8` bug
- support `CC="gcc -Wall"` in `configure -s` (QingLong)
- avoid a flush caused by `ftell` in `gzopen` for write mode (Ken Raeburn)
- fix test for shared library support to avoid compiler warnings
- `zlib.lib` -> `zlib.dll` in `msdos/zlib.rc` (Gilles Vollant)
- check for `TARGET_OS_MAC` in addition to `MACOS` (Brad Pettit)
- do not use `fdopen` for Metrowerks on Mac (Brad Pettit)
- **add** checks for `gzputc` and `gzputc` in `example.c`
- avoid warnings in `gzio.c` and `deflate.c` (Andreas Kleinert)
- use `const` for the CRC table (Ken Raeburn)

- fixed "make uninstall" for shared libraries
- use Tracev instead of Trace in infblock.c
- in example.c use correct compressed length for test_sync
- suppress +vnocompatwarnings in configure for HPUX (not always supported)

Changes in 1.0.7 (20 Jan 1998)

- fix gzseek which was broken in write mode
- return error for gzseek to negative absolute position
- fix configure for Linux (Chun-Chung Chen)
- increase stack space for MSC (Tim Wegner)
- get_crc_table and inflateSyncPoint are EXPORTed (Gilles Vollant)
- define EXPORTVA for gzprintf (Gilles Vollant)
- **added** man page zlib.3 (Rick Rodgers)
- for contrib/untgz, fix mkdir() and improve Makefile
- check gzseek in write mode in example.c
- allocate extra buffer for seeks only if gzseek is actually called
- avoid signed/unsigned comparisons (Tim Wegner, Gilles Vollant)
- **add** inflateSyncPoint in zconf.h
- fix list of exported functions in nt/zlib.dnt and mdsos/zlib.def

Changes in 1.0.6 (19 Jan 1998)

- **add** functions gzprintf, gzputc, gzgetc, gztell, gzeof, gzseek, gzrewind and gzsetparams (thanks to Roland Giersig and Kevin Ruland for some of this code)
- Fix a deflate bug occurring only with compression level 0 (thanks to Andy Buckler for finding this one).
- In minigzip, pass transparently also the first byte for .Z files.
- return Z_BUF_ERROR instead of Z_OK if output buffer full in uncompress()
- check Z_FINISH in inflate (thanks to Marc Schluper)
- Implement deflateCopy (thanks to Adam Costello)
- make static libraries by default in configure, **add** --shared option.
- move MSDOS or Windows specific files to directory msdos
- suppress the notion of partial flush to simplify the interface (but the symbol Z_PARTIAL_FLUSH is kept for compatibility with 1.0.4)
- suppress history buffer provided by application to simplify the interface (this feature was not implemented anyway in 1.0.4)
- next_in and avail_in must be initialized before calling inflateInit or inflateInit2
- **add** EXPORT in all exported functions (for Windows DLL)
- **added** Makefile.nt (thanks to Stephen Williams)
- **added** the unsupported "contrib" directory:
 - contrib/asm386/ by Gilles Vollant <info@winimage.com>
386 asm code replacing longest_match().
 - contrib/iostream/ by Kevin Ruland <kevin@rodin.wustl.edu>
A C++ I/O streams interface to the zlib gz* **functions**
 - contrib/iostream2/ by Tyge Løvset <Tyge.Lovset@cmr.no>
Another C++ I/O streams interface
 - contrib/untgz/ by "Pedro A. Aranda Guti\irrez" <paag@tid.es>
A very simple tar.gz file extractor using zlib
 - contrib/visual-basic.txt by Carlos Rios <c_rios@sonda.cl>
How to use compress(), uncompress() and the gz* **functions** from VB.
- pass params -f (filtered data), -h (huffman only), -1 to -9 (compression level) in minigzip (thanks to Tom Lane)
- use const for rommable constants in deflate
- **added** test for gzseek and gztell in example.c
- **add** undocumented function inflateSyncPoint() (hack for Paul Mackerras)
- **add** undocumented function zError to convert error code to string (for Tim Smithers)
- Allow compilation of gzio with -DNO_DEFLATE to avoid the compression code.
- Use default memcpy for Symantec MSDOS compiler.
- **Add** EXPORT keyword for check_func (needed for Windows DLL)
- **add** current directory to LD_LIBRARY_PATH for "make test"
- create also a link for libz.so.1
- **added** support for FUJITSU UXP/DS (thanks to Toshiaki Nomura)
- use \$(SHAREDLIB) instead of libz.so in Makefile.in (for HPUX)
- **added** -soname for Linux in configure (Chun-Chung Chen,
- assign numbers to the exported functions in zlib.def (for Windows DLL)
- **add** advice in zlib.h for best usage of deflateSetDictionary
- work around compiler bug on Atari (cast Z_NULL in call of s->checkfn)
- allow compilation with ANSI keywords only enabled for TurboC in large model
- avoid "versionString"[0] (Borland bug)
- **add** NEED_DUMMY_RETURN for Borland
- use variable z_verbose for tracing in debug mode (L. Peter Deutsch).

- allow compilation with CC
- defined STDC for OS/2 (David Charlap)
- limit external names to 8 chars for MVS (Thomas Lund)
- in minigzip.c, use static buffers only for 16-bit systems
- fix suffix check for "minigzip -d foo.gz"
- do not return an error for the 2nd of two consecutive gzflush() (Felix Lee)
- use _fdopen instead of fdopen for MSC >= 6.0 (Thomas Fanslau)
- **added** makelcc.bat for lcc-win32 (Tom St Denis)
- in Makefile.dj2, use copy and del instead of install and rm (Frank Donahoe)
- Avoid expanded \$Id: ChangeLog,v 1.1 2005/09/23 22:39:00 beng Exp \$. Use "rcs -kb" or "cvs admin -kb" to avoid Id expansion.
- check for unistd.h in configure (for off_t)
- **remove** useless check parameter in inflate_blocks_free
- avoid useless assignment of s->check to itself in inflate_blocks_new
- do not flush twice in gzclose (thanks to Ken Raeburn)
- rename FOPEN as F_OPEN to avoid clash with /usr/include/sys/file.h
- use NO_ERRNO_H instead of enumeration of operating systems with errno.h
- work around buggy fclose on pipes for HP/UX
- support zlib DLL with BORLAND C++ 5.0 (thanks to Glenn Randers-Pehrson)
- fix configure if CC is already equal to gcc

Changes in 1.0.5 (3 Jan 98)

- Fix inflate to terminate gracefully when fed corrupted or invalid data
- Use const for rommable constants in inflate
- Eliminate memory leaks on error conditions in inflate
- **Removed** some vestigial code in inflate
- Update web address in README

Changes in 1.0.4 (24 Jul 96)

- In very rare conditions, deflate(s, Z_FINISH) could fail to produce an EOF bit, so the decompressor could decompress all the correct data but went on to attempt decompressing extra garbage data. This affected minigzip too.
- zlibVersion and gzerror return const char* (**needed** for DLL)
- port to RISCOS (no fdopen, no multiple dots, no unlink, no fileno)
- use z_error only for DEBUG (avoid problem with DLLs)

Changes in 1.0.3 (2 Jul 96)

- use z_stream instead of z_stream *, which is now a far pointer in MSDOS small and medium models; this makes the library incompatible with previous versions for these models. (No effect in large model or on other systems.)
- return OK instead of BUF_ERROR if previous deflate call returned with avail_out as zero but there is nothing to do
- **added** memcmp for non STDC compilers
- define NO_DUMMY_DECL for more Mac compilers (.h files merged incorrectly)
- define __32BIT__ if __386__ or i386 is defined (pb. with Watcom and SCO)
- better check for 16-bit mode MSC (avoids problem with Symantec)

Changes in 1.0.2 (23 May 96)

- **added** Windows DLL support
- **added** a function zlibVersion (for the DLL support)
- fixed declarations using Bytef in infutil.c (pb with MSDOS medium model)
- Bytef is define's instead of typedef'd only for Borland C
- avoid reading uninitialized memory in example.c
- mention in README that the zlib format is now RFC1950
- updated Makefile.dj2
- **added** algorithm.doc

Changes in 1.0.1 (20 May 96) [1.0 skipped to avoid confusion]

- fix array overlay in deflate.c which sometimes caused bad compressed data
- fix inflate bug with empty stored block
- fix MSDOS medium model which was broken in 0.99
- fix deflateParams() which could generated bad compressed data.
- Bytef is define'd instead of typedef'ed (work around Borland bug)
- **added** an INDEX file
- new makefiles for DJGPP (Makefile.dj2), 32-bit Borland (Makefile.b32), Watcom (Makefile.wat), Amiga SAS/C (Makefile.sas)
- speed up adler32 for modern machines without auto-increment
- **added** -ansi for IRIX in configure
- static_init_done in trees.c is an int
- define unlink as delete for VMS
- fix configure for QNX
- **add** configure branch for SCO and HP/UX
- avoid many warnings (unused variables, dead assignments, etc...)
- no fdopen for BeOS

- fix the Watcom fix for 32 bit mode (define FAR as empty)
- **removed** redefinition of Byte for MKWERKS
- work around an MKWERKS bug (incorrect merge of all .h files)

Changes in 0.99 (27 Jan 96)

- allow preset dictionary shared between compressor and decompressor
- allow compression level 0 (no compression)
- **add** deflateParams in zlib.h: allow dynamic change of compression level and compression strategy.
- test large buffers and deflateParams in example.c
- **add** optional "configure" to build zlib as a shared library
- suppress Makefile.gnx, use configure instead
- fixed deflate for 64-bit systems (detected on Cray)
- fixed inflate_blocks for 64-bit systems (detected on Alpha)
- declare Z_DEFLATED in zlib.h (possible parameter for deflateInit2)
- always return Z_BUF_ERROR when deflate() has nothing to do
- deflateInit and inflateInit are now macros to allow version checking
- prefix all global functions and types with z_ with -DZ_PREFIX
- make fallocc completely reentrant (infrees.c)
- fixed very unlikely race condition in ct_static_init
- free in reverse order of allocation to help memory manager
- use zlib-1.0/* **instead** of zlib/* **inside** the tar.gz
- make zlib warning-free with "gcc -O3 -Wall -Wwrite-strings -Wpointer-arith -Wconversion -Wstrict-prototypes -Wmissing-prototypes"
- allow gzread on concatenated .gz files
- deflateEnd now returns Z_DATA_ERROR if it was premature
- deflate is finally (?) fully deterministic (no matches beyond end of input)
- Document Z_SYNC_FLUSH
- **add** uninstall in Makefile
- Check for __cplusplus in zlib.h
- Better test in ct_align for partial flush
- avoid harmless warnings for Borland C++
- initialize hash_head in deflate.c
- avoid warning on fdopen (gzio.c) for HP cc -Aa
- include stdlib.h for STDC compilers
- include errno.h for Cray
- ignore error if ranlib doesn't exist
- call ranlib twice for NeXTSTEP
- use exec_prefix instead of prefix for libz.a
- renamed ct_* **as** _tr_* **to** avoid conflict with applications
- clear z->msg in inflateInit2 before any error return
- initialize opaque in example.c, gzio.c, deflate.c and inflate.c
- fixed typo in zconf.h (__GNUC__ => __GNUG__)
- check for WIN32 in zconf.h and zutil.c (avoid farmalloc in 32-bit mode)
- fix typo in Make_vms.com (f\$trnlrm -> f\$getsyi)
- in fcalloc, normalize pointer if size > 65520 bytes
- don't use special fcalloc for 32 bit Borland C++
- use STDC instead of __GO32__ to avoid redeclaring exit, calloc, etc...
- use Z_BINARY instead of BINARY
- document that gzclose after gzopen will close the file
- allow "a" as mode in gzopen.
- fix error checking in gzread
- allow skipping .gz extra-field on pipes
- **added** reference to Perl interface in README
- put the crc table in FAR data (I dislike more and more the medium model :)
- **added** get_crc_table
- **added** a dimension to all arrays (Borland C can't count).
- workaround Borland C bug in declaration of inflate_codes_new & inflate_fast
- guard against multiple inclusion of *.h (for precompiled header on Mac)
- Watcom C pretends to be Microsoft C small model even in 32 bit mode.
- don't use unsized arrays to avoid silly warnings by Visual C++:
warning C4746: 'inflate_mask' : unsized array treated as '__far'
(what's wrong with far data in far model?).
- define enum out of inflate_blocks_state to allow compilation with C++

Changes in 0.95 (16 Aug 95)

- fix MSDOS small and medium model (now easier to adapt to any compiler)
- inlined send_bits
- fix the final (-) bug for deflate with flush (output was correct but not completely flushed in rare occasions).
- default window size is same for compression and decompression (it's now sufficient to set MAX_WBITS in zconf.h).
- voidp -> voidpf and voidnp -> voidp (for consistency with other typedefs and because voidnp was not near in large model).

Changes in 0.94 (13 Aug 95)

- support MSDOS medium model
- fix deflate with flush (could sometimes generate bad output)
- fix deflateReset (zlib header was incorrectly suppressed)
- **added** support for VMS
- allow a compression level in gzopen()
- gzflush now calls fflush
- For deflate with flush, flush even if no more input is provided.
- rename libgz.a as libz.a
- avoid complex expression in infcodes.c triggering Turbo C bug
- work around a problem with gcc on Alpha (in INSERT_STRING)
- don't use inline functions (problem with some gcc versions)
- allow renaming of Byte, uInt, etc... with #define.
- avoid warning about (unused) pointer before start of array in deflate.c
- avoid various warnings in gzio.c, example.c, infblock.c, adler32.c, zutil.c
- avoid reserved word 'new' in trees.c

Changes in 0.93 (25 June 95)

- temporarily disable inline functions
- make deflate deterministic
- give enough lookahead for PARTIAL_FLUSH
- Set binary mode for stdin/stdout in minigzip.c for OS/2
- don't even use signed char in inflate (not portable enough)
- fix inflate memory leak for segmented architectures

Changes in 0.92 (3 May 95)

- don't assume that char is signed (problem on SGI)
- Clear bit buffer when starting a stored block
- no memcpy on Pyramid
- suppressed inftest.c
- optimized fill_window, put longest_match inline for gcc
- optimized inflate on stored blocks.
- untabify all sources to simplify patches

Changes in 0.91 (2 May 95)

- Default MEM_LEVEL is 8 (not 9 for Unix) as documented in zlib.h
- Document the memory requirements in zconf.h
- **added** "make install"
- fix sync search logic in inflateSync
- deflate(Z_FULL_FLUSH) now works even if output buffer too short
- after inflateSync, don't scare people with just "lo world"
- **added** support for DJGPP

Changes in 0.9 (1 May 95)

- don't assume that zalloc clears the allocated memory (the TurboC bug was Mark's bug after all:)
- let again gzread copy uncompressed data unchanged (was working in 0.71)
- deflate(Z_FULL_FLUSH), inflateReset and inflateSync are now fully implemented
- **added** a test of inflateSync in example.c
- moved MAX_WBITS to zconf.h because users might want to change that.
- document explicitly that zalloc(64K) on MSDOS must return a normalized pointer (zero offset)
- **added** Makefiles for Microsoft C, Turbo C, Borland C++
- faster crc32()

Changes in 0.8 (29 April 95)

- **added** fast inflate (inffast.c)
- deflate(Z_FINISH) now returns Z_STREAM_END when done. Warning: this is incompatible with previous versions of zlib which returned Z_OK.
- work around a TurboC compiler bug (bad code for b < 0, see infutil.h) (actually that was not a compiler bug, see 0.81 above)
- gzread no longer reads one extra byte in certain cases
- In gzio destroy(), don't reference a freed structure
- avoid many warnings for MSDOS
- avoid the ERROR symbol which is used by MS Windows

Changes in 0.71 (14 April 95)

- Fixed more MSDOS compilation problems : (There is still a bug with TurboC large model.

Changes in 0.7 (14 April 95)

- **Added** full inflate support.
- Simplified the crc32() interface. The pre- and post-conditioning

(one's complement) is now done inside `crc32()`. WARNING: this is incompatible with previous versions; see `zlib.h` for the new usage.

Changes in 0.61 (12 April 95)

- workaround for a bug in TurboC. `example` and `minigzip` now work on MSDOS.

Changes in 0.6 (11 April 95)

- **added** `minigzip.c`
- **added** `gzdopen` to reopen a file descriptor as `gzFile`
- **added** transparent reading of non-gzipped files in `gzread`.
- fixed bug in `gzread` (don't read crc as data)
- fixed bug in `destroy` (`gzio.c`) (don't return `Z_STREAM_END` for `gzclose`).
- don't allocate big arrays in the stack (for MSDOS)
- fix some MSDOS compilation problems

Changes in 0.5:

- do real compression in `deflate.c`. `Z_PARTIAL_FLUSH` is supported but not yet `Z_FULL_FLUSH`.
- support decompression but only in a single step (forced `Z_FINISH`)
- **added** opaque object for `zalloc` and `zfree`.
- **added** `deflateReset` and `inflateReset`
- **added** a variable `zlib_version` for consistency checking.
- renamed the 'filter' parameter of `deflateInit2` as 'strategy'.
- **Added** `Z_FILTERED` and `Z_HUFFMAN_ONLY` constants.

Changes in 0.4:

- avoid "zip" everywhere, use `zlib` instead of `ziplib`.
- suppress `Z_BLOCK_FLUSH`, interpret `Z_PARTIAL_FLUSH` as block flush if compression method == 8.
- **added** `adler32` and `crc32`
- renamed `deflateOptions` as `deflateInit2`, call one or the other but not both
- **added** the method parameter for `deflateInit2`.
- **added** `inflateInit2`
- simplified considerably `deflateInit` and `inflateInit` by not supporting user-provided history buffer. This is supported only in `deflateInit2` and `inflateInit2`.

Changes in 0.3:

- prefix all macro names with `Z_`
- use `Z_FINISH` instead of `deflateEnd` to finish compression.
- **added** `Z_HUFFMAN_ONLY`
- **added** `gzerror()`

Frequently Asked Questions about zlib

If your question is not there, please check the zlib home page <http://www.zlib.org> which may have more recent information.
The latest zlib FAQ is at http://www.gzip.org/zlib/zlib_faq.html

1. Is zlib Y2K-compliant?

Yes. zlib doesn't handle dates.

2. Where can I get a Windows DLL version?

The zlib sources can be compiled without change to produce a DLL.
See the file win32/DLL_FAQ.txt in the zlib distribution.
Pointers to the precompiled DLL are found in the zlib web site at <http://www.zlib.org>.

3. Where can I get a Visual Basic interface to zlib?

See

- * <http://www.dogma.net/markn/articles/zlibtool/zlibtool.htm>
- * contrib/visual-basic.txt in the zlib distribution
- * win32/DLL_FAQ.txt in the zlib distribution

4. compress() returns Z_BUF_ERROR.

Make sure that before the call of compress, the length of the compressed buffer is equal to the total size of the compressed buffer and not zero. For Visual Basic, check that this parameter is passed by reference ("as any"), not by value ("as long").

5. deflate() or inflate() returns Z_BUF_ERROR.

Before making the call, make sure that avail_in and avail_out are not zero. When setting the parameter flush equal to Z_FINISH, also make sure that avail_out is big enough to allow processing all pending input. Note that a Z_BUF_ERROR is not fatal--another call to deflate() or inflate() can be made with more input or output space. A Z_BUF_ERROR may in fact be unavoidable depending on how the functions are used, since it is not possible to tell whether or not there is more output pending when strm.avail_out returns with zero.

6. Where's the zlib documentation (man pages, etc.)?

It's in zlib.h for the moment, and Francis S. Lin has converted it to a web page [zlib.html](http://www.zlib.org/zlib.html). Volunteers to transform this to Unix-style man pages, please contact us (zlib@gzip.org). Examples of zlib usage are in the files example.c and minigzip.c.

7. Why don't you use GNU autoconf or libtool or ...?

Because we would like to keep zlib as a very small and simple package. zlib is rather portable and doesn't need much configuration.

8. I found a bug in zlib.

Most of the time, such problems are due to an incorrect usage of zlib. Please try to reproduce the problem with a small program and send the corresponding source to us at zlib@gzip.org. Do not send multi-megabyte data files without prior agreement.

9. Why do I get "undefined reference to gzputc"?

If "make test" produces something like

```
example.o(.text+0x154): undefined reference to 'gzputc'
```

check that you don't have old files libz.* in /usr/lib, /usr/local/lib or /usr/X11R6/lib. Remove any old versions, then do "make install".

10. I need a Delphi interface to zlib.

See the contrib/delphi directory in the zlib distribution.

11. Can zlib handle .zip archives?

Not by itself, no. See the directory contrib/minizip in the zlib distribution.

12. Can zlib handle .Z files?

No, sorry. You have to spawn an uncompress or gunzip subprocess, or adapt the code of uncompress on your own.

13. How can I make a Unix shared library?

```
make clean
./configure -s
make
```

14. How do I install a shared zlib library on Unix?

After the above, then:

```
make install
```

However, many flavors of Unix come with a shared zlib already installed. Before going to the trouble of compiling a shared version of zlib and trying to install it, you may want to check if it's already there! If you can `#include <zlib.h>`, it's there. The `-lz` option will probably link to it.

15. I have a question about OttoPDF.

We are not the authors of OttoPDF. The real author is on the OttoPDF web site: Joel Hainley, jhainley@myndkryme.com.

16. Can zlib decode Flate data in an Adobe PDF file?

Yes. See <http://www.fastio.com/> (ClibPDF), or <http://www.pdflib.com/>. To modify PDF forms, see <http://sourceforge.net/projects/acroformtool/>.

17. Why am I getting this "register_frame_info not found" error on Solaris?

After installing zlib 1.1.4 on Solaris 2.6, running applications using zlib generates an error such as:

```
ld.so.1: rpm: fatal: relocation error: file /usr/local/lib/libz.so:
symbol __register_frame_info: referenced symbol not found
```

The symbol `__register_frame_info` is not part of zlib, it is generated by the C compiler (cc or gcc). You must recompile applications using zlib which have this problem. This problem is specific to Solaris. See <http://www.sunfreeware.com> for Solaris versions of zlib and applications using zlib.

18. Why does gzip give an error on a file I make with compress/deflate?

The compress and deflate functions produce data in the zlib format, which is different and incompatible with the gzip format. The `gz*` functions in zlib on the other hand use the gzip format. Both the zlib and gzip formats use the same compressed data format internally, but have different headers and trailers around the compressed data.

19. Ok, so why are there two different formats?

The gzip format was designed to retain the directory information about a single file, such as the name and last modification date. The zlib format on the other hand was designed for in-memory and communication channel applications, and has a much more compact header and trailer and uses a faster integrity check than gzip.

20. Well that's nice, but how do I make a gzip file in memory?

You can request that deflate write the gzip format instead of the zlib format using `deflateInit2()`. You can also request that inflate decode

the gzip format using `inflateInit2()`. Read `zlib.h` for more details.

21. Is zlib thread-safe?

Yes. However any library routines that zlib uses and any application-provided memory allocation routines must also be thread-safe. zlib's `gz*` functions use `stdio` library routines, and most of zlib's functions use the library memory allocation routines by default. zlib's `Init` functions allow for the application to provide custom memory allocation routines.

Of course, you should only operate on any given zlib or gzip stream from a single thread at a time.

22. Can I use zlib in my commercial application?

Yes. Please read the license in `zlib.h`.

23. Is zlib under the GNU license?

No. Please read the license in `zlib.h`.

24. The license says that altered source versions must be "plainly marked". So what exactly do I need to do to meet that requirement?

You need to change the `ZLIB_VERSION` and `ZLIB_VERNUM` `#defines` in `zlib.h`. In particular, the final version number needs to be changed to "f", and an identification string should be appended to `ZLIB_VERSION`. Version numbers `x.x.x.f` are reserved for modifications to zlib by others than the zlib maintainers. For example, if the version of the base zlib you are altering is "1.2.3.4", then in `zlib.h` you should change `ZLIB_VERNUM` to `0x123f`, and `ZLIB_VERSION` to something like "1.2.3.f-zachary-mods-v3". You can also update the version strings in `deflate.c` and `inftrees.c`.

For altered source distributions, you should also note the origin and nature of the changes in `zlib.h`, as well as in `ChangeLog` and `README`, along with the dates of the alterations. The origin should include at least your name (or your company's name), and an email address to contact for help or issues with the library.

Note that distributing a compiled zlib library along with `zlib.h` and `zconf.h` is also a source distribution, and so you should change `ZLIB_VERSION` and `ZLIB_VERNUM` and note the origin and nature of the changes in `zlib.h` as you would for a full source distribution.

25. Will zlib work on a big-endian or little-endian architecture, and can I exchange compressed data between them?

Yes and yes.

26. Will zlib work on a 64-bit machine?

It should. It has been tested on 64-bit machines, and has no dependence on any data types being limited to 32-bits in length. If you have any difficulties, please provide a complete problem report to zlib@gzip.org

27. Will zlib decompress data from the PKWare Data Compression Library?

No. The PKWare DCL uses a completely different compressed data format than does PKZIP and zlib. However, you can look in zlib's `contrib/blast` directory for a possible solution to your problem.

28. Can I access data randomly in a compressed stream?

No, not without some preparation. If when compressing you periodically use `Z_FULL_FLUSH`, carefully write all the pending data at those points, and keep an index of those locations, then you can start decompression at those points. You have to be careful to not use `Z_FULL_FLUSH` too often, since it can significantly degrade compression.

29. Does zlib work on MVS, OS/390, CICS, etc.?

We don't know for sure. We have heard occasional reports of success on these systems. If you do use it on one of these, please provide us with a report, instructions, and patches that we can reference when we get

these questions. Thanks.

30. Is there some simpler, easier to read version of inflate I can look at to understand the deflate format?

First off, you should read RFC 1951. Second, yes. Look in zlib's contrib/puff directory.

31. Does zlib infringe on any patents?

As far as we know, no. In fact, that was originally the whole point behind zlib. Look here for some more information:

<http://www.gzip.org/#faq11>

32. Can zlib work with greater than 4 GB of data?

Yes. inflate() and deflate() will process any amount of data correctly. Each call of inflate() or deflate() is limited to input and output chunks of the maximum value that can be stored in the compiler's "unsigned int" type, but there is no limit to the number of chunks. Note however that the strm.total_in and strm.total_out counters may be limited to 4 GB. These counters are provided as a convenience and are not used internally by inflate() or deflate(). The application can easily set up its own counters updated after each call of inflate() or deflate() to count beyond 4 GB. compress() and uncompress() may be limited to 4 GB, since they operate in a single call. gzseek() and gztell() may be limited to 4 GB depending on how zlib is compiled. See the zlibCompileFlags() function in zlib.h.

The word "may" appears several times above since there is a 4 GB limit only if the compiler's "long" type is 32 bits. If the compiler's "long" type is 64 bits, then the limit is 16 exabytes.

33. Does zlib have any security vulnerabilities?

The only one that we are aware of is potentially in gzprintf(). If zlib is compiled to use sprintf() or vsprintf(), then there is no protection against a buffer overflow of a 4K string space, other than the caller of gzprintf() assuring that the output will not exceed 4K. On the other hand, if zlib is compiled to use snprintf() or vsnprintf(), which should normally be the case, then there is no vulnerability. The ./configure script will display warnings if an insecure variation of sprintf() will be used by gzprintf(). Also the zlibCompileFlags() function will return information on what variant of sprintf() is used by gzprintf().

If you don't have snprintf() or vsnprintf() and would like one, you can find a portable implementation here:

<http://www.ijs.si/software/snprintf/>

Note that you should be using the most recent version of zlib. Versions 1.1.3 and before were subject to a double-free vulnerability.

34. Is there a Java version of zlib?

Probably what you want is to use zlib in Java. zlib is already included as part of the Java SDK in the java.util.zip package. If you really want a version of zlib written in the Java language, look on the zlib home page for links: <http://www.zlib.org/>

35. I get this or that compiler or source-code scanner warning when I crank it up to maximally-pedantic. Can't you guys write proper code?

Many years ago, we gave up attempting to avoid warnings on every compiler in the universe. It just got to be a waste of time, and some compilers were downright silly. So now, we simply make sure that the code always works.

36. Valgrind (or some similar memory access checker) says that deflate is performing a conditional jump that depends on an uninitialized value. Isn't that a bug?

No. That is intentional for performance reasons, and the output of deflate is not affected. This only started showing up recently since

zlib 1.2.x uses malloc() by default for allocations, whereas earlier versions used calloc(), which zeros out the allocated memory.

37. Will zlib read the (insert any ancient or arcane format here) compressed data format?

Probably not. Look in the comp.compression FAQ for pointers to various formats and associated software.

38. How can I encrypt/decrypt zip files with zlib?

zlib doesn't support encryption. The original PKZIP encryption is very weak and can be broken with freely available programs. To get strong encryption, use GnuPG, <http://www.gnupg.org/>, which already includes zlib compression. For PKZIP compatible "encryption", look at <http://www.info-zip.org/>

39. What's the difference between the "gzip" and "deflate" HTTP 1.1 encodings?

"gzip" is the gzip format, and "deflate" is the zlib format. They should probably have called the second one "zlib" instead to avoid confusion with the raw deflate compressed data format. While the HTTP 1.1 RFC 2616 correctly points to the zlib specification in RFC 1950 for the "deflate" transfer encoding, there have been reports of servers and browsers that incorrectly produce or expect raw deflate data per the deflate specification in RFC 1951, most notably Microsoft. So even though the "deflate" transfer encoding using the zlib format would be the more efficient approach (and in fact exactly what the zlib format was designed for), using the "gzip" transfer encoding is probably more reliable due to an unfortunate choice of name on the part of the HTTP 1.1 authors.

Bottom line: use the gzip format for HTTP 1.1 encoding.

40. Does zlib support the new "Deflate64" format introduced by PKWare?

No. PKWare has apparently decided to keep that format proprietary, since they have not documented it as they have previous compression formats. In any case, the compression improvements are so modest compared to other more modern approaches, that it's not worth the effort to implement.

41. Can you please sign these lengthy legal documents and fax them back to us so that we can use your software in our product?

No. Go away. Shoo.

ChangeLog	history of changes
FAQ	Frequently Asked Questions about zlib
INDEX	this file
Makefile	makefile for Unix (generated by configure)
Makefile.in	makefile for Unix (template for configure)
README	guess what
algorithm.txt	description of the (de)compression algorithm
configure	configure script for Unix
zconf.in.h	template for zconf.h (used by configure)
amiga/	makefiles for Amiga SAS C
as400/	makefiles for IBM AS/400
msdos/	makefiles for MSDOS
old/	makefiles for various architectures and zlib documentation files that have not yet been updated for zlib 1.2.x
projects/	projects for various Integrated Development Environments
qnx/	makefiles for QNX
win32/	makefiles for Windows
	 zlib public header files (must be kept):
zconf.h	
zlib.h	
	 private source files used to build the zlib library:
adler32.c	
compress.c	
crc32.c	
crc32.h	
deflate.c	
deflate.h	
gzio.c	
inffback.c	
inffast.c	
inffast.h	
inffixed.h	
inflate.c	
inflate.h	
inftrees.c	
inftrees.h	
trees.c	
trees.h	
uncompr.c	
zutil.c	
zutil.h	
	 source files for sample programs:
example.c	
minigzip.c	
	 unsupported contribution by third parties
See contrib/README.contrib	

```
# Makefile for zlib
```

```
CC=cc
```

```
CFLAGS=-O
```

```
LIBRARIES=libz
```

```
libz_FILES="adler32.c compress.c crc32.c gzio.c uncompr.c deflate.c trees.c \  
            zutil.c inflate.c infback.c inftrees.c inffast.c"
```

```
TYPE=both
```

ZLIB DATA COMPRESSION LIBRARY

zlib 1.2.3 is a general purpose data compression library. All the code is thread safe. The data format used by the zlib library is described by RFCs (Request for Comments) 1950 to 1952 in the files <http://www.ietf.org/rfc/rfc1950.txt> (zlib format), [rfc1951.txt](http://www.ietf.org/rfc/rfc1951.txt) (deflate format) and [rfc1952.txt](http://www.ietf.org/rfc/rfc1952.txt) (gzip format). These documents are also available in other formats from <ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html>

All functions of the compression library are documented in the file `zlib.h` (volunteer to write man pages welcome, contact zlib@gzip.org). A usage example of the library is given in the file `example.c` which also tests that the library is working correctly. Another example is given in the file `minigzip.c`. The compression library itself is composed of all source files except `example.c` and `minigzip.c`.

To compile all files and run the test program, follow the instructions given at the top of `Makefile`. In short "make test; make install" should work for most machines. For Unix: `./configure; make test; make install`. For MSDOS, use one of the special makefiles such as `Makefile.msc`. For VMS, use `make_vms.com`.

Questions about zlib should be sent to [<zlib@gzip.org>](mailto:zlib@gzip.org), or to Gilles Vollant [<info@winimage.com>](mailto:info@winimage.com) for the Windows DLL version. The zlib home page is <http://www.zlib.org> or <http://www.gzip.org/zlib/> Before reporting a problem, please check this site to verify that you have the latest version of zlib; otherwise get the latest version and check whether the problem still exists or not.

PLEASE read the zlib FAQ http://www.gzip.org/zlib/zlib_faq.html before asking for help.

Mark Nelson [<markn@ieee.org>](mailto:markn@ieee.org) wrote an article about zlib for the Jan. 1997 issue of Dr. Dobbs's Journal; a copy of the article is available in <http://dogma.net/markn/articles/zlibtool/zlibtool.htm>

The changes made in version 1.2.3 are documented in the file `ChangeLog`.

Unsupported third party contributions are provided in directory "contrib".

A Java implementation of zlib is available in the Java Development Kit <http://java.sun.com/j2se/1.4.2/docs/api/java/util/zip/package-summary.html> See the zlib home page <http://www.zlib.org> for details.

A Perl interface to zlib written by Paul Marquess [<pmqs@cpan.org>](mailto:pmqs@cpan.org) is in the CPAN (Comprehensive Perl Archive Network) sites <http://www.cpan.org/modules/by-module/Compress/>

A Python interface to zlib written by A.M. Kuchling [<amk@amk.ca>](mailto:amk@amk.ca) is available in Python 1.5 and later versions, see <http://www.python.org/doc/lib/module-zlib.html>

A zlib binding for TCL written by Andreas Kupries [<a.kupries@westend.com>](mailto:a.kupries@westend.com) is available at http://www.oche.de/~akupries/soft/trf/trf_zip.html

An experimental package to read and write files in .zip format, written on top of zlib by Gilles Vollant [<info@winimage.com>](mailto:info@winimage.com), is available in the contrib/minizip directory of zlib.

Notes for some targets:

- For Windows DLL versions, please see `win32/DLL_FAQ.txt`
- For 64-bit Irix, `deflate.c` must be compiled without any optimization. With `-O`, one `libpng` test fails. The test works in 32 bit mode (with the `-n32` compiler flag). The compiler bug has been reported to SGI.
- zlib doesn't work with gcc 2.6.3 on a DEC 3000/300LX under OSF/1 2.1 it works when compiled with cc.
- On Digital Unix 4.0D (formerly OSF/1) on AlphaServer, the cc option `-std1` is necessary to get `gzprintf` working correctly. This is done by `configure`.
- zlib doesn't work on HP-UX 9.05 with some versions of `/bin/cc`. It works with

other compilers. Use "make test" to check your compiler.

- gzopen is not supported on RISCOS, BEOS and by some Mac compilers.
- For PalmOs, see <http://palmzlib.sourceforge.net/>
- When building a shared, i.e. dynamic library on Mac OS X, the library must be installed before testing (do "make install" before "make test"), since the library location is specified in the library.

Acknowledgments:

The deflate format used by zlib was defined by Phil Katz. The deflate and zlib specifications were written by L. Peter Deutsch. Thanks to all the people who reported problems and suggested various improvements in zlib; they are too numerous to cite here.

Copyright notice:

(C) 1995-2004 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly
jloup@gzip.org

Mark Adler
madler@alumni.caltech.edu

If you use the zlib library in a product, we would appreciate *not* receiving lengthy legal documents to sign. The sources are provided for free but without warranty of any kind. The library has been entirely written by Jean-loup Gailly and Mark Adler; it does not include third-party code.

If you redistribute modified sources, we would appreciate that you include in the file ChangeLog history information documenting your changes. Please read the FAQ for more information on the distribution of modified source versions.


```

/* Adler-32.c -- compute the Adler-32 checksum of a data stream
 * Copyright (C) 1995-2004 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id: adler32.c,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#define ZLIB_INTERNAL
#include "zlib.h"

#define BASE 65521UL    /* largest prime smaller than 65536 */
#define NMAX 5552
/* NMAX is the largest n such that 255n(n+1)/2 + (n+1)(BASE-1) <= 2^32-1 */

#define DO1(buf,i)  {adler += (buf)[i]; sum2 += adler;}
#define DO2(buf,i)  DO1(buf,i); DO1(buf,i+1);
#define DO4(buf,i)  DO2(buf,i); DO2(buf,i+2);
#define DO8(buf,i)  DO4(buf,i); DO4(buf,i+4);
#define DO16(buf)   DO8(buf,0); DO8(buf,8);

/* use NO_DIVIDE if your processor does not do division in hardware */
#ifndef NO_DIVIDE
#  define MOD(a) \
    do { \
        if (a >= (BASE << 16)) a -= (BASE << 16); \
        if (a >= (BASE << 15)) a -= (BASE << 15); \
        if (a >= (BASE << 14)) a -= (BASE << 14); \
        if (a >= (BASE << 13)) a -= (BASE << 13); \
        if (a >= (BASE << 12)) a -= (BASE << 12); \
        if (a >= (BASE << 11)) a -= (BASE << 11); \
        if (a >= (BASE << 10)) a -= (BASE << 10); \
        if (a >= (BASE << 9)) a -= (BASE << 9); \
        if (a >= (BASE << 8)) a -= (BASE << 8); \
        if (a >= (BASE << 7)) a -= (BASE << 7); \
        if (a >= (BASE << 6)) a -= (BASE << 6); \
        if (a >= (BASE << 5)) a -= (BASE << 5); \
        if (a >= (BASE << 4)) a -= (BASE << 4); \
        if (a >= (BASE << 3)) a -= (BASE << 3); \
        if (a >= (BASE << 2)) a -= (BASE << 2); \
        if (a >= (BASE << 1)) a -= (BASE << 1); \
        if (a >= BASE) a -= BASE; \
    } while (0)
#  define MOD4(a) \
    do { \
        if (a >= (BASE << 4)) a -= (BASE << 4); \
        if (a >= (BASE << 3)) a -= (BASE << 3); \
        if (a >= (BASE << 2)) a -= (BASE << 2); \
        if (a >= (BASE << 1)) a -= (BASE << 1); \
        if (a >= BASE) a -= BASE; \
    } while (0)
#else
#  define MOD(a) a %= BASE
#  define MOD4(a) a %= BASE
#endif

/* ===== */
uLong ZEXPORT adler32(adler, buf, len)
uLong adler;
const Bytef *buf;
uInt len;
{
    unsigned long sum2;
    unsigned n;

    /* split Adler-32 into component sums */
    sum2 = (adler >> 16) & 0xffff;
    adler &= 0xffff;

    /* in case user likes doing a byte at a time, keep it fast */
    if (len == 1) {
        adler += buf[0];
        if (adler >= BASE)
            adler -= BASE;
        sum2 += adler;
    }

```

```

    if (sum2 >= BASE)
        sum2 -= BASE;
    return Adler | (sum2 << 16);
}

/* initial Adler-32 value (deferred check for len == 1 speed) */
if (buf == Z_NULL)
    return 1L;

/* in case short lengths are provided, keep it somewhat fast */
if (len < 16) {
    while (len-- > 0) {
        Adler += *buf++;
        sum2 += Adler;
    }
    if (Adler >= BASE)
        Adler -= BASE;
    MOD4(sum2);
    return Adler | (sum2 << 16);
}

/* do length NMAX blocks -- requires just one modulo operation */
while (len >= NMAX) {
    len -= NMAX;
    n = NMAX / 16;
    do {
        DO16(buf);
        buf += 16;
    } while (--n);
    MOD(Adler);
    MOD(sum2);
}

/* do remaining bytes (less than NMAX, still just one modulo) */
if (len) {
    while (len >= 16) {
        len -= 16;
        DO16(buf);
        buf += 16;
    }
    while (len-- > 0) {
        Adler += *buf++;
        sum2 += Adler;
    }
    MOD(Adler);
    MOD(sum2);
}

/* return recombined sums */
return Adler | (sum2 << 16);
}

/* ===== */
uLong ZEXPORT Adler32_combine(adler1, adler2, len2)
uLong adler1;
uLong adler2;
z_off_t len2;
{
    unsigned long sum1;
    unsigned long sum2;
    unsigned rem;

    /* the derivation of this formula is left as an exercise for the reader */
    rem = (unsigned)(len2 % BASE);
    sum1 = adler1 & 0xffff;
    sum2 = rem * sum1;
    MOD(sum2);
    sum1 += (adler2 & 0xffff) + BASE - 1;
    sum2 += ((adler1 >> 16) & 0xffff) + ((adler2 >> 16) & 0xffff) + BASE - rem;
    if (sum1 > BASE) sum1 -= BASE;
    if (sum1 > BASE) sum1 -= BASE;
    if (sum2 > (BASE << 1)) sum2 -= (BASE << 1);
    if (sum2 > BASE) sum2 -= BASE;
    return sum1 | (sum2 << 16);
}

```

```
}
```

1. Compression algorithm (deflate)

The deflation algorithm used by gzip (also zip and zlib) is a variation of LZ77 (Lempel-Ziv 1977, see reference below). It finds duplicated strings in the input data. The second occurrence of a string is replaced by a pointer to the previous string, in the form of a pair (distance, length). Distances are limited to 32K bytes, and lengths are limited to 258 bytes. When a string does not occur anywhere in the previous 32K bytes, it is emitted as a sequence of literal bytes. (In this description, 'string' must be taken as an arbitrary sequence of bytes, and is not restricted to printable characters.)

Literals or match lengths are compressed with one Huffman tree, and match distances are compressed with another tree. The trees are stored in a compact form at the start of each block. The blocks can have any size (except that the compressed data for one block must fit in available memory). A block is terminated when deflate() determines that it would be useful to start another block with fresh trees. (This is somewhat similar to the behavior of LZW-based `_compress_`.)

Duplicated strings are found using a hash table. All input strings of length 3 are inserted in the hash table. A hash index is computed for the next 3 bytes. If the hash chain for this index is not empty, all strings in the chain are compared with the current input string, and the longest match is selected.

The hash chains are searched starting with the most recent strings, to favor small distances and thus take advantage of the Huffman encoding. The hash chains are singly linked. There are no deletions from the hash chains, the algorithm simply discards matches that are too old.

To avoid a worst-case situation, very long hash chains are arbitrarily truncated at a certain length, determined by a runtime option (level parameter of `deflateInit`). So `deflate()` does not always find the longest possible match but generally finds a match which is long enough.

`deflate()` also defers the selection of matches with a lazy evaluation mechanism. After a match of length N has been found, `deflate()` searches for a longer match at the next input byte. If a longer match is found, the previous match is truncated to a length of one (thus producing a single literal byte) and the process of lazy evaluation begins again. Otherwise, the original match is kept, and the next match search is attempted only N steps later.

The lazy match evaluation is also subject to a runtime parameter. If the current match is long enough, `deflate()` reduces the search for a longer match, thus speeding up the whole process. If compression ratio is more important than speed, `deflate()` attempts a complete second search even if the first match is already long enough.

The lazy match evaluation is not performed for the fastest compression modes (level parameter 1 to 3). For these fast modes, new strings are inserted in the hash table only when no match was found, or when the match is not too long. This degrades the compression ratio but saves time since there are both fewer insertions and fewer searches.

2. Decompression algorithm (inflate)

2.1 Introduction

The key question is how to represent a Huffman code (or any prefix code) so that you can decode fast. The most important characteristic is that shorter codes are much more common than longer codes, so pay attention to decoding the short codes fast, and let the long codes take longer to decode.

`inflate()` sets up a first level table that covers some number of bits of input less than the length of longest code. It gets that many bits from the stream, and looks it up in the table. The table will tell if the next code is that many bits or less and how many, and if it is, it will tell the value, else it will point to the next level table for which `inflate()` grabs more bits and tries to decode a longer code.

How many bits to make the first lookup is a tradeoff between the time it

takes to decode and the time it takes to build the table. If building the table took no time (and if you had infinite memory), then there would only be a first level table to cover all the way to the longest code. However, building the table ends up taking a lot longer for more bits since short codes are replicated many times in such a table. What inflate() does is simply to make the number of bits in the first table a variable, and then to set that variable for the maximum speed.

For inflate, which has 286 possible codes for the literal/length tree, the size of the first table is nine bits. Also the distance trees have 30 possible values, and the size of the first table is six bits. Note that for each of those cases, the table ended up one bit longer than the 'average' code length, i.e. the code length of an approximately flat code which would be a little more than eight bits for 286 symbols and a little less than five bits for 30 symbols.

2.2 More details on the inflate table lookup

Ok, you want to know what this cleverly obfuscated inflate tree actually looks like. You are correct that it's not a Huffman tree. It is simply a lookup table for the first, let's say, nine bits of a Huffman symbol. The symbol could be as short as one bit or as long as 15 bits. If a particular symbol is shorter than nine bits, then that symbol's translation is duplicated in all those entries that start with that symbol's bits. For example, if the symbol is four bits, then it's duplicated 32 times in a nine-bit table. If a symbol is nine bits long, it appears in the table once.

If the symbol is longer than nine bits, then that entry in the table points to another similar table for the remaining bits. Again, there are duplicated entries as needed. The idea is that most of the time the symbol will be short and there will only be one table look up. (That's whole idea behind data compression in the first place.) For the less frequent long symbols, there will be two lookups. If you had a compression method with really long symbols, you could have as many levels of lookups as is efficient. For inflate, two is enough.

So a table entry either points to another table (in which case nine bits in the above example are gobbled), or it contains the translation for the symbol and the number of bits to gobble. Then you start again with the next ungobbled bit.

You may wonder: why not just have one lookup table for how ever many bits the longest symbol is? The reason is that if you do that, you end up spending more time filling in duplicate symbol entries than you do actually decoding. At least for deflate's output that generates new trees every several 10's of kbytes. You can imagine that filling in a 2^{15} entry table for a 15-bit code would take too long if you're only decoding several thousand symbols. At the other extreme, you could make a new table for every bit in the code. In fact, that's essentially a Huffman tree. But then you spend too much time traversing the tree while decoding, even for short symbols.

So the number of bits for the first lookup table is a trade of the time to fill out the table vs. the time spent looking at the second level and above of the table.

Here is an example, scaled down:

The code being decoded, with 10 symbols, from 1 to 6 bits long:

A: 0
B: 10
C: 1100
D: 11010
E: 11011
F: 11100
G: 11101
H: 11110
I: 111110
J: 111111

Let's make the first table three bits long (eight entries):

000: A,1

```
001: A,1
010: A,1
011: A,1
100: B,2
101: B,2
110: -> table X (gobble 3 bits)
111: -> table Y (gobble 3 bits)
```

Each entry is what the bits decode as and how many bits that is, i.e. how many bits to gobble. Or the entry points to another table, with the number of bits to gobble implicit in the size of the table.

Table X is two bits long since the longest code starting with 110 is five bits long:

```
00: C,1
01: C,1
10: D,2
11: E,2
```

Table Y is three bits long since the longest code starting with 111 is six bits long:

```
000: F,2
001: F,2
010: G,2
011: G,2
100: H,2
101: H,2
110: I,3
111: J,3
```

So what we have here are three tables with a total of 20 entries that had to be constructed. That's compared to 64 entries for a single table. Or compared to 16 entries for a Huffman tree (six two entry tables and one four entry table). Assuming that the code ideally represents the probability of the symbols, it takes on the average 1.25 lookups per symbol. That's compared to one lookup for the single table, or 1.66 lookups per symbol for the Huffman tree.

There, I think that gives you a picture of what's going on. For inflate, the meaning of a particular symbol is often more than just a letter. It can be a byte (a "literal"), or it can be either a length or a distance which indicates a base value and a number of bits to fetch after the code that is added to the base value. Or it might be the special end-of-block code. The data structures created in inftrees.c try to encode all that information compactly in the tables.

Jean-loup Gailly
jloup@gzip.org

Mark Adler
madler@alumni.caltech.edu

References:

[LZ77] Ziv J., Lempel A., ``A Universal Algorithm for Sequential Data Compression,`` IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.

``DEFLATE Compressed Data Format Specification`` available in
<http://www.ietf.org/rfc/rfc1951.txt>

```

/* compress.c -- compress a memory buffer
 * Copyright (C) 1995-2003 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id: compress.c,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#define ZLIB_INTERNAL
#include "zlib.h"

/* =====
   Compresses the source buffer into the destination buffer. The level
   parameter has the same meaning as in deflateInit. sourceLen is the byte
   length of the source buffer. Upon entry, destLen is the total size of the
   destination buffer, which must be at least 0.1% larger than sourceLen plus
   12 bytes. Upon exit, destLen is the actual size of the compressed buffer.

   compress2 returns Z_OK if success, Z_MEM_ERROR if there was not enough
   memory, Z_BUF_ERROR if there was not enough room in the output buffer,
   Z_STREAM_ERROR if the level parameter is invalid.
 */
int ZEXPORT compress2 (dest, destLen, source, sourceLen, level)
    Bytef *dest;
    uLongf *destLen;
    const Bytef *source;
    uLong sourceLen;
    int level;
{
    z_stream stream;
    int err;

    stream.next_in = (Bytef*)source;
    stream.avail_in = (uInt)sourceLen;
#ifdef MAXSEG_64K
    /* Check for source > 64K on 16-bit machine: */
    if ((uLong)stream.avail_in != sourceLen) return Z_BUF_ERROR;
#endif
    stream.next_out = dest;
    stream.avail_out = (uInt)*destLen;
    if ((uLong)stream.avail_out != *destLen) return Z_BUF_ERROR;

    stream.zalloc = (alloc_func)0;
    stream.zfree = (free_func)0;
    stream.opaque = (voidpf)0;

    err = deflateInit(&stream, level);
    if (err != Z_OK) return err;

    err = deflate(&stream, Z_FINISH);
    if (err != Z_STREAM_END) {
        deflateEnd(&stream);
        return err == Z_OK ? Z_BUF_ERROR : err;
    }
    *destLen = stream.total_out;

    err = deflateEnd(&stream);
    return err;
}

/* =====
 */
int ZEXPORT compress (dest, destLen, source, sourceLen)
    Bytef *dest;
    uLongf *destLen;
    const Bytef *source;
    uLong sourceLen;
{
    return compress2(dest, destLen, source, sourceLen, Z_DEFAULT_COMPRESSION);
}

/* =====
   If the default memLevel or windowBits for deflateInit() is changed, then
   this function needs to be updated.
 */

```

```
uLong ZEXPORT compressBound (sourceLen)
    uLong sourceLen;
{
    return sourceLen + (sourceLen >> 12) + (sourceLen >> 14) + 11;
}
```



```

#!/bin/sh
# configure script for zlib. This script is needed only if
# you wish to build a shared library and your system supports them,
# of if you need special compiler, flags or install directory.
# Otherwise, you can just use directly "make test; make install"
#
# To create a shared library, use "configure --shared"; by default a static
# library is created. If the primitive shared library support provided here
# does not work, use ftp://prep.ai.mit.edu/pub/gnu/libtool-*.tar.gz
#
# To impose specific compiler or flags or install directory, use for example:
#   prefix=$HOME CC=cc CFLAGS="-O4" ./configure
# or for csh/tcsh users:
#   (setenv prefix $HOME; setenv CC cc; setenv CFLAGS "-O4"; ./configure)
# LDSHARED is the command to be used to create a shared library

# Incorrect settings of CC or CFLAGS may prevent creating a shared library.
# If you have problems, try without defining CC and CFLAGS before reporting
# an error.

LIBS=libz.a
LDFLAGS="-L. ${LIBS}"
VER=`sed -n -e '/VERSION "/s.*"(.*)".*/1p' < zlib.h`
VER2=`sed -n -e '/VERSION "/s.*"([0-9]*\.[0-9]*)\.[0-9]*"/1p' < zlib.h`
VER1=`sed -n -e '/VERSION "/s.*"([0-9]*)\.[0-9]*"/1p' < zlib.h`
AR=${AR-"ar rc"}
RANLIB=${RANLIB-"ranlib"}
prefix=${prefix-/usr/local}
exec_prefix=${exec_prefix-${prefix}}
libdir=${libdir-${exec_prefix}/lib}
includedir=${includedir-${prefix}/include}
mandir=${mandir-${prefix}/share/man}
shared_ext='.so'
shared=0
gcc=0
old_cc="$CC"
old_cflags="$CFLAGS"

while test $# -ge 1
do
case "$1" in
-h* | --h*)
echo 'usage:'
echo ' configure [--shared] [--prefix=PREFIX] [--exec_prefix=EXPREFIX]'
echo ' [--libdir=LIBDIR] [--includedir=INCLUDEDIR]'
exit 0;;
-p*=* | --p*=*) prefix=`echo $1 | sed 's/[-a-z_]*=//'; shift;;
-e*=* | --e*=*) exec_prefix=`echo $1 | sed 's/[-a-z_]*=//'; shift;;
-l*=* | --l*=*) libdir=`echo $1 | sed 's/[-a-z_]*=//'; shift;;
-i*=* | --i*=*) includedir=`echo $1 | sed 's/[-a-z_]*=//'; shift;;
-p* | --p*) prefix="$2"; shift; shift;;
-e* | --e*) exec_prefix="$2"; shift; shift;;
-l* | --l*) libdir="$2"; shift; shift;;
-i* | --i*) includedir="$2"; shift; shift;;
-s* | --s*) shared=1; shift;;
*) echo "unknown option: $1"; echo "$0 --help for help"; exit 1;;
esac
done

test=ztest$$
cat > $test.c <<EOF
extern int getchar();
int hello() {return getchar();}
EOF

test -z "$CC" && echo Checking for gcc...
cc=${CC-gcc}
cflags=${CFLAGS-"-O3"}
# to force the asm version use: CFLAGS="-O3 -DASMV" ./configure
case "$cc" in
*gcc*) gcc=1;;
esac

if test "$gcc" -eq 1 && ($cc -c $cflags $test.c) 2>/dev/null; then

```

```

CC="$cc"
SFLAGS=${CFLAGS-"-fPIC -O3"}
CFLAGS="$cflags"
case `(uname -s || echo unknown) 2>/dev/null` in
Linux | linux | GNU | GNU/*) LDSHARED=${LDSHARED-"$cc -shared -Wl,-soname,libz.so.1"} ;;
CYGWIN* | Cygwin* | cygwin* | OS/2*)
    EXE='.exe' ;;
QNX*) # This is for QNX6. I suppose that the QNX rule below is for QNX2,QNX4
      # (alain.bonnefooy@icbt.com)
      LDSHARED=${LDSHARED-"$cc -shared -Wl,-hlibz.so.1"} ;;
HP-UX*)
    LDSHARED=${LDSHARED-"$cc -shared $SFLAGS"}
    case `(uname -m || echo unknown) 2>/dev/null` in
        ia64)
            shared_ext='.so'
            SHAREDLIB='libz.so' ;;
        *)
            shared_ext='.sl'
            SHAREDLIB='libz.sl' ;;
    esac ;;
Darwin*) shared_ext='.dylib'
          SHAREDLIB=libz$shared_ext
          SHAREDLIBV=libz.$VER$shared_ext
          SHAREDLIBM=libz.$VER1$shared_ext
          LDSHARED=${LDSHARED-"$cc -dynamiclib -install_name $libdir/$SHAREDLIBM -compatibility_version
$VER1 -current_version $VER"} ;;
*) LDSHARED=${LDSHARED-"$cc -shared"} ;;
esac
else
# find system name and corresponding cc options
CC=${CC-cc}
case `(uname -sr || echo unknown) 2>/dev/null` in
HP-UX*) SFLAGS=${CFLAGS-"-O +z"}
        CFLAGS=${CFLAGS-"-O"}
        LDSHARED=${LDSHARED-"ld -b +vnocompatwarnings"}
        LDSHARED=${LDSHARED-"ld -b"}
        case `(uname -m || echo unknown) 2>/dev/null` in
            ia64)
                shared_ext='.so'
                SHAREDLIB='libz.so' ;;
            *)
                shared_ext='.sl'
                SHAREDLIB='libz.sl' ;;
        esac ;;
IRIX*) SFLAGS=${CFLAGS-"-ansi -O2 -rpath ." }
        CFLAGS=${CFLAGS-"-ansi -O2"}
        LDSHARED=${LDSHARED-"cc -shared"} ;;
OSF1\ V4*) SFLAGS=${CFLAGS-"-O -std1"}
           CFLAGS=${CFLAGS-"-O -std1"}
           LDSHARED=${LDSHARED-"cc -shared -Wl,-soname,libz.so -Wl,-msym -Wl,-rpath,$(libdir) -Wl,-set_ver
sion,$(VER):1.0"} ;;
OSF1*) SFLAGS=${CFLAGS-"-O -std1"}
        CFLAGS=${CFLAGS-"-O -std1"}
        LDSHARED=${LDSHARED-"cc -shared"} ;;
QNX*) SFLAGS=${CFLAGS-"-4 -O"}
       CFLAGS=${CFLAGS-"-4 -O"}
       LDSHARED=${LDSHARED-"cc"}
       RANLIB=${RANLIB-"true"}
       AR="cc -A" ;;
SCO_SV\ 3.2*) SFLAGS=${CFLAGS-"-O3 -dy -KPIC"}
              CFLAGS=${CFLAGS-"-O3"}
              LDSHARED=${LDSHARED-"cc -dy -KPIC -G"} ;;
SunOS\ 5*) SFLAGS=${CFLAGS-"-fast -xpg89 -KPIC -R."}
           CFLAGS=${CFLAGS-"-fast -xpg89"}
           LDSHARED=${LDSHARED-"cc -G"} ;;
SunOS\ 4*) SFLAGS=${CFLAGS-"-O2 -PIC"}
           CFLAGS=${CFLAGS-"-O2"}
           LDSHARED=${LDSHARED-"ld"} ;;
SunStudio\ 9*) SFLAGS=${CFLAGS-"-DUSE_MMAP -fast -xcode=pic32 -xtarget=ultra3 -xarch=v9b"}
               CFLAGS=${CFLAGS-"-DUSE_MMAP -fast -xtarget=ultra3 -xarch=v9b"}
               LDSHARED=${LDSHARED-"cc -xarch=v9b"} ;;
UNIX_System_V\ 4.2.0)
    SFLAGS=${CFLAGS-"-KPIC -O"}
    CFLAGS=${CFLAGS-"-O"}

```

```

        LDSHARED=${LDSHARED-"cc -G"};;
UNIX_SV\ 4.2MP)
        SFLAGS=${CFLAGS-"-Kconform_pic -O"}
        CFLAGS=${CFLAGS-"-O"}
        LDSHARED=${LDSHARED-"cc -G"};;
OpenUNIX\ 5)
        SFLAGS=${CFLAGS-"-KPIC -O"}
        CFLAGS=${CFLAGS-"-O"}
        LDSHARED=${LDSHARED-"cc -G"};;
AIX*) # Courtesy of dbakker@arrayasolutions.com
        SFLAGS=${CFLAGS-"-O -qmaxmem=8192"}
        CFLAGS=${CFLAGS-"-O -qmaxmem=8192"}
        LDSHARED=${LDSHARED-"xlc -G"};;
# send working options for other systems to support@gzip.org
*)
        SFLAGS=${CFLAGS-"-O"}
        CFLAGS=${CFLAGS-"-O"}
        LDSHARED=${LDSHARED-"cc -shared"};;
esac
fi

SHAREDLIB=${SHAREDLIB-"libz$shared_ext"}
SHAREDLIBV=${SHAREDLIBV-"libz$shared_ext.$VER"}
SHAREDLIBM=${SHAREDLIBM-"libz$shared_ext.$VER1"}

if test $shared -eq 1; then
    echo Checking for shared library support...
    # we must test in two steps (cc then ld), required at least on SunOS 4.x
    if test "$(CC -c $SFLAGS $test.c) 2>&1" = "" &&
        test "$LDSHARED -o $test$shared_ext $test.o) 2>&1" = ""; then
        CFLAGS="$SFLAGS"
        LIBS="$SHAREDLIBV"
        echo Building shared library $SHAREDLIBV with $CC.
    elif test -z "$old_cc" -a -z "$old_cflags"; then
        echo No shared library support.
        shared=0;
    else
        echo 'No shared library support; try without defining CC and CFLAGS'
        shared=0;
    fi
fi

if test $shared -eq 0; then
    LDSHARED="$CC"
    echo Building static library $LIBS version $VER with $CC.
else
    LDFLAGS="-L. ${SHAREDLIBV}"
fi

cat > $test.c <<EOF
#include <unistd.h>
int main() { return 0; }
EOF
if test "$(CC -c $CFLAGS $test.c) 2>&1" = ""; then
    sed < zconf.in.h "/HAVE_UNISTD_H/s/%0%1%" > zconf.h
    echo "Checking for unistd.h... Yes."
else
    cp -p zconf.in.h zconf.h
    echo "Checking for unistd.h... No."
fi

cat > $test.c <<EOF
#include <stdio.h>
#include <stdarg.h>
#include "zconf.h"

int main()
{
#ifdef STDC
    choke me
#endif

    return 0;
}
EOF

```

```
if test "(`$CC -c $CFLAGS $test.c) 2>&1`" = ""; then
  echo "Checking whether to use vs[n]printf() or s[n]printf()... using vs[n]printf()"
```

```
    cat > $test.c <<EOF
#include <stdio.h>
#include <stdarg.h>

int mytest(char *fmt, ...)
{
    char buf[20];
    va_list ap;

    va_start(ap, fmt);
    vsnprintf(buf, sizeof(buf), fmt, ap);
    va_end(ap);
    return 0;
}

int main()
{
    return (mytest("Hello%d\n", 1));
}
EOF
```

```
    if test "(`$CC $CFLAGS -o $test $test.c) 2>&1`" = ""; then
      echo "Checking for vsnprintf() in stdio.h... Yes."
```

```
        cat >$test.c <<EOF
#include <stdio.h>
#include <stdarg.h>

int mytest(char *fmt, ...)
{
    int n;
    char buf[20];
    va_list ap;

    va_start(ap, fmt);
    n = vsnprintf(buf, sizeof(buf), fmt, ap);
    va_end(ap);
    return n;
}

int main()
{
    return (mytest("Hello%d\n", 1));
}
EOF
```

```
    if test "(`$CC -c $CFLAGS $test.c) 2>&1`" = ""; then
      echo "Checking for return value of vsnprintf()... Yes."
    else
      CFLAGS="$CFLAGS -DHAS_vsnprintf_void"
      echo "Checking for return value of vsnprintf()... No."
      echo "  WARNING: apparently vsnprintf() does not return a value. zlib"
      echo "  can build but will be open to possible string-format security"
      echo "  vulnerabilities."
```

```
    fi
  else
    CFLAGS="$CFLAGS -DNO_vsnprintf"
    echo "Checking for vsnprintf() in stdio.h... No."
    echo "  WARNING: vsnprintf() not found, falling back to vsprintf(). zlib"
    echo "  can build but will be open to possible buffer-overflow security"
    echo "  vulnerabilities."
```

```
    cat >$test.c <<EOF
#include <stdio.h>
#include <stdarg.h>

int mytest(char *fmt, ...)
{
    int n;
    char buf[20];
    va_list ap;
```

```

va_start(ap, fmt);
n = vsprintf(buf, fmt, ap);
va_end(ap);
return n;
}

int main()
{
    return (mytest("Hello%d\n", 1));
}
EOF

if test "`($CC -c $CFLAGS $test.c) 2>&1`" = ""; then
    echo "Checking for return value of vsprintf()... Yes."
else
    CFLAGS="$CFLAGS -DHAS_vsprintf_void"
    echo "Checking for return value of vsprintf()... No."
    echo " WARNING: apparently vsprintf() does not return a value. zlib"
    echo " can build but will be open to possible string-format security"
    echo " vulnerabilities."
fi
else
    echo "Checking whether to use vs[n]printf() or s[n]printf()... using s[n]printf()"

    cat >$test.c <<EOF
#include <stdio.h>

int mytest()
{
    char buf[20];

    snprintf(buf, sizeof(buf), "%s", "foo");
    return 0;
}

int main()
{
    return (mytest());
}
EOF

if test "`($CC $CFLAGS -o $test $test.c) 2>&1`" = ""; then
    echo "Checking for snprintf() in stdio.h... Yes."

    cat >$test.c <<EOF
#include <stdio.h>

int mytest()
{
    char buf[20];

    return snprintf(buf, sizeof(buf), "%s", "foo");
}

int main()
{
    return (mytest());
}
EOF

if test "`($CC -c $CFLAGS $test.c) 2>&1`" = ""; then
    echo "Checking for return value of snprintf()... Yes."
else
    CFLAGS="$CFLAGS -DHAS_snprintf_void"
    echo "Checking for return value of snprintf()... No."
    echo " WARNING: apparently snprintf() does not return a value. zlib"
    echo " can build but will be open to possible string-format security"
    echo " vulnerabilities."
fi
else
    CFLAGS="$CFLAGS -DNO_snprintf"
    echo "Checking for snprintf() in stdio.h... No."

```

```

echo " WARNING: snprintf() not found, falling back to sprintf(). zlib "
echo " can build but will be open to possible buffer-overflow security "
echo " vulnerabilities."

cat >$test.c <<EOF
#include <stdio.h>

int mytest()
{
    char buf[20];

    return sprintf(buf, "%s", "foo");
}

int main()
{
    return (mytest());
}
EOF

if test "`($CC -c $CFLAGS $test.c) 2>&1`" = ""; then
    echo "Checking for return value of sprintf()... Yes."
else
    CFLAGS="$CFLAGS -DHAS_sprintf_void"
    echo "Checking for return value of sprintf()... No."
    echo " WARNING: apparently sprintf() does not return a value. zlib "
    echo " can build but will be open to possible string-format security "
    echo " vulnerabilities."
fi
fi

cat >$test.c <<EOF
#include <errno.h>
int main() { return 0; }
EOF
if test "`($CC -c $CFLAGS $test.c) 2>&1`" = ""; then
    echo "Checking for errno.h... Yes."
else
    echo "Checking for errno.h... No."
    CFLAGS="$CFLAGS -DNO_ERRNO_H"
fi

cat > $test.c <<EOF
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
caddr_t hello() {
    return mmap((caddr_t)0, (off_t)0, PROT_READ, MAP_SHARED, 0, (off_t)0);
}
EOF
if test "`($CC -c $CFLAGS $test.c) 2>&1`" = ""; then
    CFLAGS="$CFLAGS -DUSE_MMAP"
    echo Checking for mmap support... Yes.
else
    echo Checking for mmap support... No.
fi

CPP=${CPP-"$CC -E"}
case $CFLAGS in
    *ASMV*)
        if test "`nm $test.o | grep _hello`" = ""; then
            CPP="$CPP -DNO_UNDERLINE"
            echo Checking for underline in external names... No.
        else
            echo Checking for underline in external names... Yes.
        fi;
    esac

rm -f $test.[co] $test $test$shared_ext

# update Makefile
sed < Makefile.in "
/^CC *=/s#=#=$CC#

```

```
/^CFLAGS *=/s#=. *#=$CFLAGS#
/^CPP *=/s#=. *#=$CPP#
/^LDSHARED *=/s#=. *#=$LDSHARED#
/^LIBS *=/s#=. *#=$LIBS#
/^SHAREDLIB *=/s#=. *#=$SHAREDLIB#
/^SHAREDLIBV *=/s#=. *#=$SHAREDLIBV#
/^SHAREDLIBM *=/s#=. *#=$SHAREDLIBM#
/^AR *=/s#=. *#=$AR#
/^RANLIB *=/s#=. *#=$RANLIB#
/^EXE *=/s#=. *#=$EXE#
/^prefix *=/s#=. *#=$prefix#
/^exec_prefix *=/s#=. *#=$exec_prefix#
/^libdir *=/s#=. *#=$libdir#
/^includedir *=/s#=. *#=$includedir#
/^mandir *=/s#=. *#=$mandir#
/^LDFLAGS *=/s#=. *#=$LDFLAGS#
" > Makefile
```

```

/* crc32.c -- compute the CRC-32 of a data stream
 * Copyright (C) 1995-2005 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 *
 * Thanks to Rodney Brown <rbrown64@csc.com.au> for his contribution of faster
 * CRC methods: exclusive-oring 32 bits of data at a time, and pre-computing
 * tables for updating the shift register in one step with three exclusive-ors
 * instead of four steps with four exclusive-ors. This results in about a
 * factor of two increase in speed on a Power PC G4 (PPC7455) using gcc -O3.
 */

/* @(#) $Id: crc32.c,v 1.1 2005/09/23 22:39:00 beng Exp $ */

/*
 * Note on the use of DYNAMIC_CRC_TABLE: there is no mutex or semaphore
 * protection on the static variables used to control the first-use generation
 * of the crc tables. Therefore, if you #define DYNAMIC_CRC_TABLE, you should
 * first call get_crc_table() to initialize the tables before allowing more than
 * one thread to use crc32().
 */

#ifdef MAKECRCH
# include <stdio.h>
# ifndef DYNAMIC_CRC_TABLE
#   define DYNAMIC_CRC_TABLE
# endif /* !DYNAMIC_CRC_TABLE */
#endif /* MAKECRCH */

#include "zutil.h" /* for STDC and FAR definitions */

#define local static

/* Find a four-byte integer type for crc32_little() and crc32_big(). */
#ifndef NOBYFOUR
# ifdef STDC /* need ANSI C limits.h to determine sizes */
#   include <limits.h>
#   define BYFOUR
#   if (UINT_MAX == 0xffffffffUL)
#     typedef unsigned int u4;
#   else
#     if (ULONG_MAX == 0xffffffffUL)
#       typedef unsigned long u4;
#     else
#       if (USHRT_MAX == 0xffffffffUL)
#         typedef unsigned short u4;
#       else
#         undef BYFOUR /* can't find a four-byte integer type! */
#       endif
#     endif
#   endif
# endif /* STDC */
#endif /* NOBYFOUR */

/* Definitions for doing the crc four data bytes at a time. */
#ifdef BYFOUR
# define REV(w) (((w)>>24)+(((w)>>8)&0xff00)+ \
                (((w)&0xff00)<<8)+(((w)&0xff)<<24))
  local unsigned long crc32_little OF((unsigned long,
                                       const unsigned char FAR *, unsigned));
  local unsigned long crc32_big OF((unsigned long,
                                    const unsigned char FAR *, unsigned));
# define TBLS 8
#else
# define TBLS 1
#endif

/* Local functions for crc concatenation */
local unsigned long gf2_matrix_times OF((unsigned long *mat,
                                         unsigned long vec));
local void gf2_matrix_square OF((unsigned long *square, unsigned long *mat));

#ifdef DYNAMIC_CRC_TABLE

local volatile int crc_table_empty = 1;

```



```

local unsigned long FAR crc_table[TBLS][256];
local void make_crc_table OF((void));
#ifdef MAKECRCH
    local void write_table OF((FILE *, const unsigned long FAR *));
#endif /* MAKECRCH */
/*
    Generate tables for a byte-wise 32-bit CRC calculation on the polynomial:
    x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x+1.

    Polynomials over GF(2) are represented in binary, one bit per coefficient,
    with the lowest powers in the most significant bit. Then adding polynomials
    is just exclusive-or, and multiplying a polynomial by x is a right shift by
    one. If we call the above polynomial p, and represent a byte as the
    polynomial q, also with the lowest power in the most significant bit (so the
    byte 0xb1 is the polynomial x^7+x^3+x+1), then the CRC is (q*x^32) mod p,
    where a mod b means the remainder after dividing a by b.

    This calculation is done using the shift-register method of multiplying and
    taking the remainder. The register is initialized to zero, and for each
    incoming bit, x^32 is added mod p to the register if the bit is a one (where
    x^32 mod p is p+x^32 = x^26+...+1), and the register is multiplied mod p by
    x (which is shifting right by one and adding x^32 mod p if the bit shifted
    out is a one). We start with the highest power (least significant bit) of
    q and repeat for all eight bits of q.

    The first table is simply the CRC of all possible eight bit values. This is
    all the information needed to generate CRCs on data a byte at a time for all
    combinations of CRC register values and incoming bytes. The remaining tables
    allow for word-at-a-time CRC calculation for both big-endian and little-
    endian machines, where a word is four bytes.
*/
local void make_crc_table()
{
    unsigned long c;
    int n, k;
    unsigned long poly;          /* polynomial exclusive-or pattern */
    /* terms of polynomial defining this crc (except x^32): */
    static volatile int first = 1; /* flag to limit concurrent making */
    static const unsigned char p[] = {0,1,2,4,5,7,8,10,11,12,16,22,23,26};

    /* See if another task is already doing this (not thread-safe, but better
       than nothing -- significantly reduces duration of vulnerability in
       case the advice about DYNAMIC_CRC_TABLE is ignored) */
    if (first) {
        first = 0;

        /* make exclusive-or pattern from polynomial (0xedb88320UL) */
        poly = 0UL;
        for (n = 0; n < sizeof(p)/sizeof(unsigned char); n++)
            poly |= 1UL << (31 - p[n]);

        /* generate a crc for every 8-bit value */
        for (n = 0; n < 256; n++) {
            c = (unsigned long)n;
            for (k = 0; k < 8; k++)
                c = c & 1 ? poly ^ (c >> 1) : c >> 1;
            crc_table[0][n] = c;
        }

#ifdef BYFOUR
        /* generate crc for each value followed by one, two, and three zeros,
           and then the byte reversal of those as well as the first table */
        for (n = 0; n < 256; n++) {
            c = crc_table[0][n];
            crc_table[4][n] = REV(c);
            for (k = 1; k < 4; k++) {
                c = crc_table[0][c & 0xff] ^ (c >> 8);
                crc_table[k][n] = c;
                crc_table[k + 4][n] = REV(c);
            }
        }
#endif
    }
}
#endif /* BYFOUR */
/* BYFOUR */

crc_table_empty = 0;

```

```

    }
    else {          /* not first */
        /* wait for the other guy to finish (not efficient, but rare) */
        while (crc_table_empty)
            ;
    }
}

#ifdef MAKECRCH
/* write out CRC tables to crc32.h */
{
    FILE *out;

    out = fopen("crc32.h", "w");
    if (out == NULL) return;
    fprintf(out, "/* crc32.h — tables for rapid CRC calculation\n" );
    fprintf(out, " * Generated automatically by crc32.c\n *\n\n" );
    fprintf(out, "local const unsigned long FAR " );
    fprintf(out, "crc_table[TBLS][256]=\n{\n {\n" );
    write_table(out, crc_table[0]);
#   ifdef BYFOUR
    fprintf(out, "#ifdef BYFOUR\n");
    for (k = 1; k < 8; k++) {
        fprintf(out, " },\n {\n");
        write_table(out, crc_table[k]);
    }
    fprintf(out, "#endif\n");
#   endif /* BYFOUR */
    fprintf(out, " }\n};\n");
    fclose(out);
}

#endif /* MAKECRCH */

#ifdef MAKECRCH
local void write_table(out, table)
    FILE *out;
    const unsigned long FAR *table;
{
    int n;

    for (n = 0; n < 256; n++)
        fprintf(out, "%s0x%08lxUL%s", n % 5 ? " " : " ", table[n],
            n == 255 ? "\n" : (n % 5 == 4 ? ",\n" : ", "));
}

#endif /* MAKECRCH */

#else /* !DYNAMIC_CRC_TABLE */
/* =====
 * Tables of CRC-32s of all single-byte values, made by make_crc_table().
 */
#include "crc32.h"
#endif /* DYNAMIC_CRC_TABLE */

/* =====
 * This function can be used by asm versions of crc32()
 */
const unsigned long FAR * ZEXPORT get_crc_table()
{
#ifdef DYNAMIC_CRC_TABLE
    if (crc_table_empty)
        make_crc_table();
#endif /* DYNAMIC_CRC_TABLE */
    return (const unsigned long FAR *)crc_table;
}

/* ===== */
#define DO1 crc = crc_table[0][((int)crc ^ (*buf++)) & 0xff] ^ (crc >> 8)
#define DO8 DO1; DO1; DO1; DO1; DO1; DO1; DO1; DO1

/* ===== */
unsigned long ZEXPORT crc32(crc, buf, len)
    unsigned long crc;
    const unsigned char FAR *buf;
    unsigned len;

```

```

{
    if (buf == Z_NULL) return 0UL;
#ifdef DYNAMIC_CRC_TABLE
    if (crc_table_empty)
        make_crc_table();
#endif /* DYNAMIC_CRC_TABLE */
#ifdef BYFOUR
    if (sizeof(void *) == sizeof(ptrdiff_t)) {
        u4 endian;

        endian = 1;
        if (*(unsigned char *)&endian)
            return crc32_little(crc, buf, len);
        else
            return crc32_big(crc, buf, len);
    }
#endif /* BYFOUR */
    crc = crc ^ 0xffffffffUL;
    while (len >= 8) {
        DO8;
        len -= 8;
    }
    if (len) do {
        DO1;
    } while (--len);
    return crc ^ 0xffffffffUL;
}

#ifdef BYFOUR

/* ===== */
#define DOLIT4 c ^= *buf4++; \
    c = crc_table[3][c & 0xff] ^ crc_table[2][(c >> 8) & 0xff] ^ \
        crc_table[1][(c >> 16) & 0xff] ^ crc_table[0][c >> 24]
#define DOLIT32 DOLIT4; DOLIT4; DOLIT4; DOLIT4; DOLIT4; DOLIT4; DOLIT4; DOLIT4
/* ===== */
local unsigned long crc32_little(crc, buf, len)
    unsigned long crc;
    const unsigned char FAR *buf;
    unsigned len;
{
    register u4 c;
    register const u4 FAR *buf4;

    c = (u4)crc;
    c = ~c;
    while (len && ((ptrdiff_t)buf & 3)) {
        c = crc_table[0][(c ^ *buf++) & 0xff] ^ (c >> 8);
        len--;
    }

    buf4 = (const u4 FAR *) (const void FAR *) buf;
    while (len >= 32) {
        DOLIT32;
        len -= 32;
    }
    while (len >= 4) {
        DOLIT4;
        len -= 4;
    }
    buf = (const unsigned char FAR *) buf4;

    if (len) do {
        c = crc_table[0][(c ^ *buf++) & 0xff] ^ (c >> 8);
    } while (--len);
    c = ~c;
    return (unsigned long)c;
}

/* ===== */
#define DOBIG4 c ^= *++buf4; \

```

```

        c = crc_table[4][c & 0xff] ^ crc_table[5][(c >> 8) & 0xff] ^ \
        crc_table[6][(c >> 16) & 0xff] ^ crc_table[7][c >> 24]
#define DOBIG32 DOBIG4; DOBIG4; DOBIG4; DOBIG4; DOBIG4; DOBIG4; DOBIG4; DOBIG4
/* ===== */
local unsigned long crc32_big(crc, buf, len)
    unsigned long crc;
    const unsigned char FAR *buf;
    unsigned len;
{
    register u4 c;
    register const u4 FAR *buf4;

    c = REV((u4)crc);
    c = ~c;
    while (len && ((ptrdiff_t)buf & 3)) {
        c = crc_table[4][(c >> 24) ^ *buf++] ^ (c << 8);
        len--;
    }

    buf4 = (const u4 FAR *) (const void FAR *) buf;
    buf4--;
    while (len >= 32) {
        DOBIG32;
        len -= 32;
    }
    while (len >= 4) {
        DOBIG4;
        len -= 4;
    }
    buf4++;
    buf = (const unsigned char FAR *) buf4;

    if (len) do {
        c = crc_table[4][(c >> 24) ^ *buf++] ^ (c << 8);
    } while (--len);
    c = ~c;
    return (unsigned long)(REV(c));
}

#endif /* BYFOUR */

#define GF2_DIM 32      /* dimension of GF(2) vectors (length of CRC) */

/* ===== */
local unsigned long gf2_matrix_times(mat, vec)
    unsigned long *mat;
    unsigned long vec;
{
    unsigned long sum;

    sum = 0;
    while (vec) {
        if (vec & 1)
            sum ^= *mat;
        vec >>= 1;
        mat++;
    }
    return sum;
}

/* ===== */
local void gf2_matrix_square(square, mat)
    unsigned long *square;
    unsigned long *mat;
{
    int n;

    for (n = 0; n < GF2_DIM; n++)
        square[n] = gf2_matrix_times(mat, mat[n]);
}

/* ===== */
uLong ZEXPORT crc32_combine(crc1, crc2, len2)

```

```
uLong crc1;
uLong crc2;
z_off_t len2;
{
    int n;
    unsigned long row;
    unsigned long even[GF2_DIM];    /* even-power-of-two zeros operator */
    unsigned long odd[GF2_DIM];     /* odd-power-of-two zeros operator */

    /* degenerate case */
    if (len2 == 0)
        return crc1;

    /* put operator for one zero bit in odd */
    odd[0] = 0xedb88320L;            /* CRC-32 polynomial */
    row = 1;
    for (n = 1; n < GF2_DIM; n++) {
        odd[n] = row;
        row <<= 1;
    }

    /* put operator for two zero bits in even */
    gf2_matrix_square(even, odd);

    /* put operator for four zero bits in odd */
    gf2_matrix_square(odd, even);

    /* apply len2 zeros to crc1 (first square will put the operator for one
       zero byte, eight zero bits, in even) */
    do {
        /* apply zeros operator for this bit of len2 */
        gf2_matrix_square(even, odd);
        if (len2 & 1)
            crc1 = gf2_matrix_times(even, crc1);
        len2 >>= 1;

        /* if no more bits set, then done */
        if (len2 == 0)
            break;

        /* another iteration of the loop with odd and even swapped */
        gf2_matrix_square(odd, even);
        if (len2 & 1)
            crc1 = gf2_matrix_times(odd, crc1);
        len2 >>= 1;

        /* if no more bits set, then done */
    } while (len2 != 0);

    /* return combined crc */
    crc1 ^= crc2;
    return crc1;
}
```

```
/* crc32.h -- tables for rapid CRC calculation
 * Generated automatically by crc32.c
 */
```

```
local const unsigned long FAR crc_table[TBLS][256] =
{
    {
        0x00000000UL, 0x77073096UL, 0xee0e612cUL, 0x990951baUL, 0x076dc419UL,
        0x706af48fUL, 0xe963a535UL, 0x9e6495a3UL, 0x0edb8832UL, 0x79dcb8a4UL,
        0xe0d5e91eUL, 0x97d2d988UL, 0x09b64c2bUL, 0x7eb17cbdUL, 0xe7b82d07UL,
        0x90bfb1d9UL, 0x1db71064UL, 0x6ab020f2UL, 0xf3b97148UL, 0x84be41deUL,
        0x1dad47dUL, 0x6ddde4ebUL, 0xf4d4b551UL, 0x83d385c7UL, 0x136c9856UL,
        0x646ba8c0UL, 0xfd62f97aUL, 0x8a65c9ecUL, 0x14015c4fUL, 0x63066cd9UL,
        0xfa0f3d63UL, 0x8d080df5UL, 0x3b6e20c8UL, 0x4c69105eUL, 0xd56041e4UL,
        0xa2677172UL, 0x3c03e4d1UL, 0x4b04d447UL, 0xd20d85fdUL, 0xa50ab56bUL,
        0x35b5a8faUL, 0x42b2986cUL, 0xdbbbc9d6UL, 0xacbcf940UL, 0x32d86ce3UL,
        0x45df5c75UL, 0xdcd60dcfUL, 0xabd13d59UL, 0x26d930acUL, 0x51de003aUL,
        0xc8d75180UL, 0xbfd06116UL, 0x21b4f4b5UL, 0x56b3c423UL, 0xcfba9599UL,
        0xb8bda50fUL, 0x2802b89eUL, 0x5f058808UL, 0xc60cd9b2UL, 0xb10be924UL,
        0x2f6f7c87UL, 0x58684c11UL, 0xc1611dabUL, 0xb6662d3dUL, 0x76dc4190UL,
        0x01db7106UL, 0x98d220bcUL, 0xefd5102aUL, 0x71b18589UL, 0x06b6b51fUL,
        0x9fbfe4a5UL, 0xe8b8d433UL, 0x7807c9a2UL, 0xf00f934UL, 0x9609a88eUL,
        0xe10e9818UL, 0xf7f6a0dbbUL, 0x086d3d2dUL, 0x91646c97UL, 0xe6635c01UL,
        0x6b6b51f4UL, 0x1c6c6162UL, 0x856530d8UL, 0xf262004eUL, 0x6c0695edUL,
        0x1b01a57bUL, 0x8208f4c1UL, 0xf50fc457UL, 0x65b0d9c6UL, 0x12b7e950UL,
        0x8bbeb8eaUL, 0xfcb9887cUL, 0x62dd1dddfUL, 0x15da2d49UL, 0x8cd37cf3UL,
        0xfbd44c65UL, 0x4db26158UL, 0x3ab551ceUL, 0xa3bc0074UL, 0xd4bb30e2UL,
        0xa4dfa541UL, 0x3dd895d7UL, 0xa4d1c46dUL, 0xd3d6f4fbUL, 0x4369e96aUL,
        0x346ed9fcUL, 0xad678846UL, 0xda60b8d0UL, 0x44042d73UL, 0x33031de5UL,
        0xaa0a4c5fUL, 0xdd0d7cc9UL, 0x5005713cUL, 0x270241aaUL, 0xbe0b1010UL,
        0xc90c2086UL, 0x5768b525UL, 0x206f85b3UL, 0xb966d409UL, 0xce61e49fUL,
        0x5edef90eUL, 0x29d9c998UL, 0xb0d09822UL, 0xc7d7a8b4UL, 0x59b33d17UL,
        0x2eb40d81UL, 0xb7bd5c3bUL, 0xc0ba6cadUL, 0xedb88320UL, 0x9abfb3b6UL,
        0x03b6e20cUL, 0x74b1d29aUL, 0xeada5473UL, 0x9dd277afUL, 0x04db2615UL,
        0x73dc1683UL, 0xe3630b12UL, 0x94643b84UL, 0x0d6d6a3eUL, 0x7a6a5aa8UL,
        0xe40ecf0bUL, 0x9309ff9dUL, 0x0a00ae27UL, 0x7d079eb1UL, 0xf00f9344UL,
        0x8708a3d2UL, 0x1e01f268UL, 0x6906c2feUL, 0xf762575dUL, 0x806567cbUL,
        0x196c3671UL, 0x6e6b06e7UL, 0xfed41b76UL, 0x89d32be0UL, 0x10da7a5aUL,
        0x67dd4accUL, 0xf9b9df6fUL, 0x8ebeefff9UL, 0x17b7be43UL, 0x60b08ed5UL,
        0xd6d6a3e8UL, 0xa1d1937eUL, 0x38d8c2c4UL, 0x4fdff252UL, 0xd1bb67f1UL,
        0xa6bc5767UL, 0x3fb506ddUL, 0x48b2364bUL, 0xd80d2bdaUL, 0xaf0a1b4cUL,
        0x36034af6UL, 0x41047a60UL, 0xdf60efc3UL, 0xa867df55UL, 0x316e8eefUL,
        0x4669be79UL, 0xcb61b38cUL, 0xbc66831aUL, 0x256fd2a0UL, 0x5268e236UL,
        0xcc0c7795UL, 0xbb0b4703UL, 0x220216b9UL, 0x5505262fUL, 0xc5ba3bbeUL,
        0xb2bd0b28UL, 0x2bb45a92UL, 0x5cb36a04UL, 0xc2d7ffa7UL, 0xb5d0cf31UL,
        0x2cd99e8bUL, 0x5bdeae1dUL, 0x9b64c2b0UL, 0xec63f226UL, 0x756aa39cUL,
        0x026d930aUL, 0x9c0906a9UL, 0xeb0e363fUL, 0x72076785UL, 0x05005713UL,
        0x95bf4a82UL, 0xe2b87a14UL, 0x7bb12baeUL, 0x0cb61b38UL, 0x92d28e9bUL,
        0xe5d5be0dUL, 0x7cdcefb7UL, 0x0bdbdf21UL, 0x86d3d2d4UL, 0xf1d4e242UL,
        0x68ddb3f8UL, 0x1fda836eUL, 0x81be16cdUL, 0xf6b9265bUL, 0x6fb077e1UL,
        0x18b74777UL, 0x88085ae6UL, 0xff0f6a70UL, 0x66063bcaUL, 0x11010b5cUL,
        0x8f659effUL, 0xf862ae69UL, 0x616bffd3UL, 0x166ccf45UL, 0xa0ae278UL,
        0xd70dd2eeUL, 0x4e048354UL, 0x3903b3c2UL, 0xa7672661UL, 0xd06016f7UL,
        0x4969474dUL, 0x3e6e77dbUL, 0xaed16a4aUL, 0xd9d65adcUL, 0x40df0b66UL,
        0x37d83bf0UL, 0xa9bcae53UL, 0xdeb9ec5UL, 0x47b2cf7fUL, 0x30b5ffe9UL,
        0xbdbdf21cUL, 0xcabac28aUL, 0x53b39330UL, 0x24b4a3a6UL, 0xbad03605UL,
        0xcdd70693UL, 0x54de5729UL, 0x23d967bfUL, 0xb3667a2eUL, 0xc4614ab8UL,
        0x5d681b02UL, 0x2a6f2b94UL, 0xb40bbe37UL, 0xc30c8ea1UL, 0x5a05df1bUL,
        0x2d02ef8dUL,
#ifdef BYFOUR
    },
    {
        0x00000000UL, 0x191b3141UL, 0x32366282UL, 0x2b2d53c3UL, 0x646cc504UL,
        0x7d77f445UL, 0x565aa786UL, 0x4f4196c7UL, 0xc8d98a08UL, 0xd1c2bb49UL,
        0xfaefe88aUL, 0xe3f4d9cbUL, 0xacb54f0cUL, 0xb5ae7e4dUL, 0x9e832d8eUL,
        0x87981ccfUL, 0x4ac21251UL, 0x53d92310UL, 0x78f470d3UL, 0x61ef4192UL,
        0x2eae755UL, 0x37b5e614UL, 0x1c98b5d7UL, 0x05838496UL, 0x821b9859UL,
        0x9b00a918UL, 0xb02dfadbUL, 0xa936cb9aUL, 0xe6775d5dUL, 0xff6c6c1cUL,
        0xd4413fdfUL, 0xcd5a0e9eUL, 0x958424a2UL, 0x8c9f15e3UL, 0xa7b24620UL,
        0xbea97761UL, 0xf1e8e1a6UL, 0xe8f3d0e7UL, 0xc3de8324UL, 0xdac5b265UL,
        0x5d5daaaaUL, 0x44469febUL, 0x6f6bcc28UL, 0x7670fd69UL, 0x39316baeUL,
        0x202a5aefUL, 0x0b07092cUL, 0x121c386dUL, 0xdf4636f3UL, 0xc65d07b2UL,
        0xed705471UL, 0xf46b6530UL, 0xbb2af3f7UL, 0xa231c2b6UL, 0x891c9175UL,
        0x9007a034UL, 0x179fbcfbUL, 0x0e848dbaUL, 0x25a9de79UL, 0x3cb2ef38UL,

```

```
0x73f379ffUL, 0x6ae848beUL, 0x41c51b7dUL, 0x58de2a3cUL, 0xf0794f05UL,
0xe9627e44UL, 0xc24f2d87UL, 0xdb541cc6UL, 0x94158a01UL, 0x8d0ebb40UL,
0xa623e883UL, 0xbf38d9c2UL, 0x38a0c50dUL, 0x21bbf44cUL, 0x0a96a78fUL,
0x138d96ceUL, 0x5ccc0009UL, 0x45d73148UL, 0x6efa628bUL, 0x77e153caUL,
0xbabb5d54UL, 0xa3a06c15UL, 0x888d3fd6UL, 0x91960e97UL, 0xded79850UL,
0xc7cca911UL, 0xecelfad2UL, 0xf5facb93UL, 0x7262d75cUL, 0x6b79e61dUL,
0x4054b5deUL, 0x594f849fUL, 0x160e1258UL, 0x0f152319UL, 0x243870daUL,
0x3d23419bUL, 0x65fd6ba7UL, 0x7ce65ae6UL, 0x57cb0925UL, 0x4ed03864UL,
0x0191aea3UL, 0x188a9fe2UL, 0x33a7cc21UL, 0x2abcf60UL, 0xad24e1afUL,
0xb43fd0eeUL, 0x9f12832dUL, 0x8609b26cUL, 0xc94824abUL, 0xd05315eaUL,
0xfb7e4629UL, 0xe2657768UL, 0x2f3f79f6UL, 0x362448b7UL, 0x1d091b74UL,
0x04122a35UL, 0x4b53bcf2UL, 0x52488db3UL, 0x7965de70UL, 0x607eef31UL,
0xe7e6f3feUL, 0xfefdc2bfUL, 0xd5d0917cUL, 0xcccb03dUL, 0x838a36faUL,
0x9a9107bbUL, 0xb1bc5478UL, 0xa8a76539UL, 0x3b83984bUL, 0x2298a90aUL,
0x09b5fac9UL, 0x10aecb88UL, 0x5fef5d4fUL, 0x46f46c0eUL, 0x6dd93fcdUL,
0x74c20e8cUL, 0xf35a1243UL, 0xea412302UL, 0xc16c70c1UL, 0xd8774180UL,
0x9736d747UL, 0x8e2de606UL, 0xa500b5c5UL, 0xbc1b8484UL, 0x71418a1aUL,
0x685abb5bUL, 0x4377e898UL, 0x5a6cd9d9UL, 0x152d4f1eUL, 0x0c367e5fUL,
0x271b2d9cUL, 0x3e001cddUL, 0xb9980012UL, 0xa0833153UL, 0x8bae6290UL,
0x92b553d1UL, 0xddf4c516UL, 0xc4eff457UL, 0xefc2a794UL, 0xf6d996d5UL,
0xae07bce9UL, 0xb71c8da8UL, 0x9c31de6bUL, 0x852aef2aUL, 0xca6b79edUL,
0xd37048acUL, 0xf85d1b6fUL, 0xe1462a2eUL, 0x66de36e1UL, 0x7fc507a0UL,
0x54e85463UL, 0x4df36522UL, 0x02b2f3e5UL, 0x1ba9c2a4UL, 0x30849167UL,
0x299fa026UL, 0xe4c5aeb8UL, 0xfdde9ff9UL, 0xd6f3cc3aUL, 0xcfe8fd7bUL,
0x80a96bbcUL, 0x99b25afdUL, 0xb29f093eUL, 0xab84387fUL, 0x2c1c24b0UL,
0x350715f1UL, 0x1e2a4632UL, 0x07317773UL, 0x4870e1b4UL, 0x516bd0f5UL,
0x7a468336UL, 0x635db277UL, 0xcbfad74eUL, 0xd2e1e60fUL, 0xf9ccb5ccUL,
0xe0d7848dUL, 0xaf96124aUL, 0xb68d230bUL, 0x9da070c8UL, 0x84bb4189UL,
0x03235d46UL, 0x1a386c07UL, 0x31153fc4UL, 0x280e0e85UL, 0x674f9842UL,
0x7e54a903UL, 0x5579fac0UL, 0x4c62cb81UL, 0x8138c51fUL, 0x9823f45eUL,
0xb30ea79dUL, 0xaa1596dcUL, 0xe554001bUL, 0xfc4f315aUL, 0xd7626299UL,
0xce7953d8UL, 0x49e14f17UL, 0x50fa7e56UL, 0x7bd72d95UL, 0x62cc1cd4UL,
0x2d8d8a13UL, 0x3496bb52UL, 0x1fbbe891UL, 0x06a0d9d0UL, 0x5e7ef3ecUL,
0x4765c2adUL, 0x6c48916eUL, 0x7553a02fUL, 0x3a1236e8UL, 0x23097a9UL,
0x082456aUL, 0x113f652bUL, 0x96a779e4UL, 0x8fbc48a5UL, 0xa4911b66UL,
0xbd8a2a27UL, 0xf2cbbce0UL, 0xebd08da1UL, 0xc0fdde62UL, 0xd9e6ef23UL,
0x14bcelbdUL, 0x0da7d0fcUL, 0x268a833fUL, 0x3f91b27eUL, 0x70d024b9UL,
0x69cb15f8UL, 0x42e6463bUL, 0x5bfd777aUL, 0xdc656bb5UL, 0xc57e5af4UL,
0xee530937UL, 0xf7483876UL, 0xb809aeb1UL, 0xa1129ff0UL, 0x8a3fcc33UL,
0x9324fd72UL
```

```
0x00000000UL, 0x01c26a37UL, 0x0384d46eUL, 0x0246be59UL, 0x0709a8dcUL,
0x06cbc2ebUL, 0x048d7cb2UL, 0x054f1685UL, 0x0e1351b8UL, 0x0fd13b8fUL,
0x0d9785d6UL, 0x0c55efe1UL, 0x091af964UL, 0x08d89353UL, 0x0a9e2d0aUL,
0x0b5c473dUL, 0x1c26a370UL, 0x1de4bc94UL, 0x1fa2771eUL, 0x1e601d29UL,
0x1b2f0bacUL, 0x1aed619bUL, 0x18abdcafUL, 0x1969b5f5UL, 0x1235f2c8UL,
0x13f798ffUL, 0x11b126a6UL, 0x10734c91UL, 0x153c5a14UL, 0x14fe3023UL,
0x16b88e7aUL, 0x177ae44dUL, 0x384d46e0UL, 0x398f2cd7UL, 0x3bc9928eUL,
0x3a0bf8b9UL, 0x3f44ee3cUL, 0x3e86840bUL, 0x3cc03a52UL, 0x3d025065UL,
0x365e1758UL, 0x379c7d6fUL, 0x35dac336UL, 0x3418a901UL, 0x3157bf84UL,
0x3095d5b3UL, 0x32d36beaUL, 0x331101ddUL, 0x246be590UL, 0x25a98fa7UL,
0x27ef31feUL, 0x262d5bc9UL, 0x23624d4cUL, 0x22a0277bUL, 0x20e69922UL,
0x2124f315UL, 0x2a78b428UL, 0x2bbade1fUL, 0x29fc6046UL, 0x283e0a71UL,
0x2d711cf4UL, 0x2cb376c3UL, 0x2ef5c89aUL, 0x2f37a2adUL, 0x709a8dc0UL,
0x7158e7f7UL, 0x731e59aeUL, 0x72dc3399UL, 0x7793251cUL, 0x76514f2bUL,
0x7417f172UL, 0x75d59b45UL, 0x7e89dc78UL, 0x7f4bb64fUL, 0x7d0d0816UL,
0x7ccf6221UL, 0x798074a4UL, 0x78421e93UL, 0x7a04a0caUL, 0x7bc6cafdUL,
0x6cbc2eb0UL, 0x6d7e4487UL, 0x6f38fadeUL, 0x6efa90e9UL, 0x6bb5866cUL,
0x6a77ec5bUL, 0x68315202UL, 0x69f33835UL, 0x62af7f08UL, 0x636d153fUL,
0x612bab66UL, 0x60e9c151UL, 0x65a6d7d4UL, 0x6464bde3UL, 0x662203baUL,
0x67e0698dUL, 0x48d7cb20UL, 0x4915a117UL, 0x4b531f4eUL, 0x4a917579UL,
0x4fde63fcUL, 0x4e1c09cbUL, 0x4c5ab792UL, 0x4d9dda5UL, 0x46c49a98UL,
0x4706f0afUL, 0x45404ef6UL, 0x448224c1UL, 0x41cd3244UL, 0x400f5873UL,
0x4249e62aUL, 0x438b8c1dUL, 0x54f16850UL, 0x55330267UL, 0x5775bc3eUL,
0x56b7d609UL, 0x53f8c08cUL, 0x523aaabbUL, 0x507c14e2UL, 0x51be7ed5UL,
0x5ae239e8UL, 0x5b2053dfUL, 0x596ed86UL, 0x58a487b1UL, 0x5deb9134UL,
0x5c29fb03UL, 0x5e6f455aUL, 0x5fad2f6dUL, 0xe1351b80UL, 0xe0f771b7UL,
0xe2b1cfeeUL, 0xe373a5d9UL, 0xe63cb35cUL, 0xe7fed96bUL, 0xe5b86732UL,
0xe47a0d05UL, 0xef264a38UL, 0xee4200fUL, 0xca29e56UL, 0xed60f461UL,
0xe82fe2e4UL, 0xe9ed88d3UL, 0xebab368aUL, 0xea695cbdUL, 0xfd13b8f0UL,
0xfcd1d2c7UL, 0xfe976c9eUL, 0xff5506a9UL, 0xfa1a102cUL, 0xfbd87a1bUL,
0xf99ec442UL, 0xf85cae75UL, 0xf300e948UL, 0xf2c2837fUL, 0xf0843d26UL,
0xf1465711UL, 0xf4094194UL, 0xf5cb2ba3UL, 0xf78d95faUL, 0xf64fffcUL,
```

```
0xd9785d60UL, 0xd8ba3757UL, 0xdaafc890eUL, 0xdb3ee339UL, 0xde71f5bcUL,
0xdfb39f8bUL, 0xddf521d2UL, 0xdc374be5UL, 0xd76b0cd8UL, 0xd6a966efUL,
0xd4efd8b6UL, 0xd52db281UL, 0xd062a404UL, 0xd1a0ce33UL, 0xd3e6706aUL,
0xd2241a5dUL, 0xc55efe10UL, 0xc49c9427UL, 0xc6da2a7eUL, 0xc7184049UL,
0xc25756ccUL, 0xc3953cfbUL, 0xc1d382a2UL, 0xc011e895UL, 0xcb4dafa8UL,
0xca8fc59fUL, 0xc8c97bc6UL, 0xc90b11f1UL, 0xcc440774UL, 0xcd866d43UL,
0xcfc0d31aUL, 0xce02b92dUL, 0x91af9640UL, 0x906dfc77UL, 0x922b422eUL,
0x93e92819UL, 0x96a63e9cUL, 0x976454abUL, 0x9522eaf2UL, 0x94e080c5UL,
0x9fbcc7f8UL, 0x9e7eadcfUL, 0x9c381396UL, 0x9dfa79a1UL, 0x98b56f24UL,
0x99770513UL, 0x9b31bb4aUL, 0x9af3d17dUL, 0x8d893530UL, 0x8c4b5f07UL,
0x8e0de15eUL, 0x8fcf8b69UL, 0x8a809decUL, 0x8b42f7dbUL, 0x89044982UL,
0x88c623b5UL, 0x839a6488UL, 0x82580ebfUL, 0x801eb0e6UL, 0x81dcdad1UL,
0x8493cc54UL, 0x8551a663UL, 0x8717183aUL, 0x86d5720dUL, 0xa9e2d0a0UL,
0xa820ba97UL, 0xaa6604ceUL, 0xaba46ef9UL, 0xaeeb787cUL, 0xaf29124bUL,
0xad6fac12UL, 0xacadc625UL, 0xa7f18118UL, 0xa633eb2fUL, 0xa4755576UL,
0xa5b73f41UL, 0xa0f829c4UL, 0xa13a43f3UL, 0xa37cfdaaUL, 0xa2be979dUL,
0xb5c473d0UL, 0xb40619e7UL, 0xb640a7beUL, 0xb782cd89UL, 0xb2cddb0cUL,
0xb30fb13bUL, 0xb1490f62UL, 0xb08b6555UL, 0xbbd72268UL, 0xba15485fUL,
0xb853f606UL, 0xb9919c31UL, 0xbcde8ab4UL, 0xbd1ce083UL, 0xbf5a5edaUL,
0xbe9834edUL
```

```
0x00000000UL, 0xb8bc6765UL, 0xaa09c88bUL, 0x12b5afeeUL, 0x8f629757UL,
0x37def032UL, 0x256b5fdcUL, 0x9dd738b9UL, 0xc5b428efUL, 0x7d084f8aUL,
0x6fbde064UL, 0xd7018701UL, 0x4ad6bfb8UL, 0xf26ad8ddUL, 0xe0df7733UL,
0x58631056UL, 0x5019579fUL, 0xe8a530faUL, 0xfa109f14UL, 0x42acf871UL,
0xdf7bc0c8UL, 0x67c7a7adUL, 0x75720843UL, 0xcdce6f26UL, 0x95ad7f70UL,
0x2d111815UL, 0x3fa4b7fbUL, 0x8718d09eUL, 0x1acfe827UL, 0xa2738f42UL,
0xb0c620acUL, 0x087a47c9UL, 0xa032af3eUL, 0x188ec85bUL, 0x0a3b67b5UL,
0xb28700d0UL, 0x2f503869UL, 0x97ec5f0cUL, 0x8559f0e2UL, 0x3de59787UL,
0x658687d1UL, 0xdd3ae0b4UL, 0xcf8f4f5aUL, 0x7733283fUL, 0xae41086UL,
0x525877e3UL, 0x40edd80dUL, 0xf851bf68UL, 0xf02bf8a1UL, 0x48979fc4UL,
0x5a22302aUL, 0xe29e574fUL, 0x7f496ff6UL, 0xc7f50893UL, 0xd540a77dUL,
0x6dfcc018UL, 0x359fd04eUL, 0x8d23b72bUL, 0x9f9618c5UL, 0x272a7fa0UL,
0xbafcd471UL, 0x0241207cUL, 0x10f48f92UL, 0xa848e8f7UL, 0x9b14583dUL,
0x23a83f58UL, 0x311d90b6UL, 0x89a1f7d3UL, 0x1476cf6aUL, 0xaccaa80fUL,
0xbe7f07e1UL, 0x06c36084UL, 0x5ea070d2UL, 0xe61c17b7UL, 0xf4a9b859UL,
0x4c15df3cUL, 0xd1c2e785UL, 0x697e80e0UL, 0x7bcb2f0eUL, 0xc377486bUL,
0xcb0d0fa2UL, 0x73b168c7UL, 0x6104c729UL, 0xd9b8a04cUL, 0x446f98f5UL,
0xfcd3ff90UL, 0xee66507eUL, 0x56da371bUL, 0x0eb9274dUL, 0xb6054028UL,
0xa4b0efc6UL, 0x1c0c88a3UL, 0x81dbb01aUL, 0x3967d077UL, 0x2bd27891UL,
0x936e1ff4UL, 0x3b26f703UL, 0x839a9066UL, 0x912f3f88UL, 0x299358edUL,
0xb4446054UL, 0x0cf80731UL, 0x1e4da8dfUL, 0xa6f1cfbaUL, 0xfe92dfecUL,
0x462eb889UL, 0x549b1767UL, 0xec277002UL, 0x71f048bbUL, 0xc94c2fdeUL,
0xdbf98030UL, 0x6345e755UL, 0x6b3fa09cUL, 0xd383c7f9UL, 0xc1366817UL,
0x798a0f72UL, 0xe45d37cbUL, 0x5ce150aeUL, 0x4e54ff40UL, 0xf6e89825UL,
0xae8b8873UL, 0x1637ef16UL, 0x048240f8UL, 0xbc3e279dUL, 0x21e91f24UL,
0x99557841UL, 0x8be0d7afUL, 0x335cb0caUL, 0xed59b63bUL, 0x55e5d15eUL,
0x47507eb0UL, 0xffec19d5UL, 0x623b216cUL, 0xda874609UL, 0xc832e9e7UL,
0x708e8e82UL, 0x28ed9ed4UL, 0x9051f9b1UL, 0x82e4565fUL, 0x3a58313aUL,
0xa78f0983UL, 0x1f336ee6UL, 0x0d86c108UL, 0xb53aa66dUL, 0xbd40ela4UL,
0x05fc86c1UL, 0x1749292fUL, 0xaf54e4aUL, 0x322276f3UL, 0x8a9e1196UL,
0x982bbe78UL, 0x2097d91dUL, 0x78f4c94bUL, 0xc048ae2eUL, 0xd2fd01c0UL,
0x6a4166a5UL, 0xf7965e1cUL, 0x4f2a3979UL, 0x5d9f9697UL, 0xe523f1f2UL,
0x4d6b1905UL, 0xf5d77e60UL, 0xe762d18eUL, 0xfdeb6ebUL, 0xc2098e52UL,
0x7ab5e937UL, 0x680046d9UL, 0xd0bc21bcUL, 0x88df31eaUL, 0x3063568fUL,
0x22d6f961UL, 0x9a6a9e04UL, 0x07bda6bdUL, 0xbf01c1d8UL, 0xadba4e36UL,
0x15080953UL, 0x1d724e9aUL, 0xa5ce29ffUL, 0xb77b8611UL, 0xfc7e174UL,
0x9210d9cdUL, 0x2aacbea8UL, 0x38191146UL, 0x80a57623UL, 0xd8c66675UL,
0x607a0110UL, 0x72cfaefeUL, 0xca73c99bUL, 0x57a4f122UL, 0xef189647UL,
0xfdad39a9UL, 0x45115eccUL, 0x764dee06UL, 0xcef18963UL, 0xdc44268dUL,
0x64f841e8UL, 0xf92f7951UL, 0x41931e34UL, 0x5326b1daUL, 0xeb9ad6bfUL,
0xb3f9c6e9UL, 0x0b45a18cUL, 0x19f00e62UL, 0xa14c6907UL, 0x3c9b51beUL,
0x842736dbUL, 0x96929935UL, 0x2e2efe50UL, 0x2654b999UL, 0x9e8defcUL,
0x8c5d7112UL, 0x34e11677UL, 0xa9362eceUL, 0x118a49abUL, 0x033fe645UL,
0xbb838120UL, 0xe3e09176UL, 0x5b5cf613UL, 0x49e959fdUL, 0xf1553e98UL,
0x6c820621UL, 0xd43e6144UL, 0xc68bceaaUL, 0x7e37a9cfUL, 0xd67f4138UL,
0x6ec3265dUL, 0x7c7689b3UL, 0xc4cae6dUL, 0x591dd66fUL, 0xela1b10aUL,
0xf3141ee4UL, 0x4ba87981UL, 0x13cb69d7UL, 0xab770eb2UL, 0xb9c2a15cUL,
0x017ec639UL, 0x9ca9fe80UL, 0x241599e5UL, 0x36a0360bUL, 0x8e1c516eUL,
0x866616a7UL, 0x3eda71c2UL, 0x2c6fde2cUL, 0x94d3b949UL, 0x090481f0UL,
0xb1b8e695UL, 0xa30d497bUL, 0x1bb12e1eUL, 0x43d23e48UL, 0xfb6e592dUL,
0x9dbf6c3UL, 0x516791a6UL, 0xccb0a91fUL, 0x740cce7aUL, 0x66b96194UL,
0xde0506f1UL
```


},

```
0x00000000UL, 0x96300777UL, 0x2c610eeeUL, 0xba510999UL, 0x19c46d07UL,
0x8ff46a70UL, 0x35a563e9UL, 0xa395649eUL, 0x3288db0eUL, 0xa4b8dc79UL,
0x1ee9d5e0UL, 0x88d9d297UL, 0x2b4cb609UL, 0xbd7cb17eUL, 0x072db8e7UL,
0x911dbf90UL, 0x6410b71dUL, 0xf220b06aUL, 0x4871b9f3UL, 0xde41be84UL,
0x7dd4da1aUL, 0xebe4dd6dUL, 0x51b5d4f4UL, 0xc785d383UL, 0x56986c13UL,
0xc0a86b64UL, 0x7af962fdUL, 0xecc9658aUL, 0x4f5c0114UL, 0xd96c0663UL,
0x633d0ffaUL, 0xf50d088dUL, 0xc8206e3bUL, 0x5e10694cUL, 0xe44160d5UL,
0x727167a2UL, 0xd1e4033cUL, 0x47d4044bUL, 0xfd850dd2UL, 0x6bb50aa5UL,
0xfaa8b535UL, 0x6c98b242UL, 0xd6c9bbdbUL, 0x40f9bcacUL, 0xe36cd832UL,
0x755cdf45UL, 0xcf0dd6dcUL, 0x593dd1abUL, 0xac30d926UL, 0x3a00de51UL,
0x8051d7c8UL, 0x1661d0bfUL, 0xb5f4b421UL, 0x23c4b356UL, 0x9995bacfUL,
0x0fa5bdb8UL, 0x9eb80228UL, 0x0888055fUL, 0xb2d90cc6UL, 0x24e90bb1UL,
0x877c6f2fUL, 0x114c6858UL, 0xab1d61c1UL, 0x3d2d66b6UL, 0x9041dc76UL,
0x0671db01UL, 0xbc20d298UL, 0x2a10d5efUL, 0x8985b171UL, 0x1fb5b606UL,
0xa5e4bf9fUL, 0x33d4b8e8UL, 0xa2c90778UL, 0x34f9000fUL, 0x8ea80996UL,
0x18980ee1UL, 0xbb0d6a7fUL, 0x2d3d6d08UL, 0x976c6491UL, 0x015c63e6UL,
0xf4516b6bUL, 0x62616c1cUL, 0xd8306585UL, 0x4e0062f2UL, 0xed95066cUL,
0x7ba5011bUL, 0xc1f40882UL, 0x57c40ff5UL, 0xc6d9b065UL, 0x50e9b712UL,
0xeab8be8bUL, 0x7c88b9fcUL, 0xdf1ddd62UL, 0x492dda15UL, 0xf37cd38cUL,
0x654cd4fbUL, 0x5861b24dUL, 0xce51b53aUL, 0x7400bca3UL, 0xe230bbd4UL,
0x41a5df4aUL, 0xd795d83dUL, 0x6dc4d1a4UL, 0xfbf4d6d3UL, 0x6ae96943UL,
0xfcd96e34UL, 0x468867adUL, 0xd0b860daUL, 0x732d0444UL, 0xe51d0333UL,
0x5f4c0aaaUL, 0xc97c0dddUL, 0x3c710550UL, 0xaa410227UL, 0x10100bbeUL,
0x86200cc9UL, 0x25b56857UL, 0xb3856f20UL, 0x09d466b9UL, 0x9fe461ceUL,
0x0ef9de5eUL, 0x98c9d929UL, 0x2298d0b0UL, 0xb4a8d7c7UL, 0x173db359UL,
0x810db42eUL, 0x3b5cbdb7UL, 0xad6cbac0UL, 0x2083b8edUL, 0xb6b3bf9aUL,
0x0ce2b603UL, 0x9ad2b174UL, 0x3947d5eaUL, 0xaf77d29dUL, 0x1526db04UL,
0x8316dc73UL, 0x120b63e3UL, 0x843b6494UL, 0x3e6a6d0dUL, 0xa85a6a7aUL,
0x0bcf0ee4UL, 0x9dff0993UL, 0x27ae000aUL, 0xb19e077dUL, 0x44930ff0UL,
0xd2a30887UL, 0x68f2011eUL, 0xfec20669UL, 0x5d5762f7UL, 0xcb676580UL,
0x71366c19UL, 0xe7066b6eUL, 0x761bd4feUL, 0xe02bd389UL, 0x5a7ada10UL,
0xcc4add67UL, 0x6fdfb9f9UL, 0xf9efbe8eUL, 0x43beb717UL, 0xd58eb060UL,
0xe8a3d6d6UL, 0x7e93d1a1UL, 0xc4c2d838UL, 0x52f2df4fUL, 0xf167b5d1UL,
0x6757bca6UL, 0xdd06b53fUL, 0x4b36b248UL, 0xda2b0dd8UL, 0x4c1b0aafUL,
0xf64a0336UL, 0x607a0441UL, 0xc3ef60dfUL, 0x55df67a8UL, 0xef8e6e31UL,
0x79be6946UL, 0x8cb361cbUL, 0x1a8366bcUL, 0xa0d26f25UL, 0x36e26852UL,
0x95770cccUL, 0x03470bbbUL, 0xb9160222UL, 0x2f260555UL, 0xbe3bbac5UL,
0x280bbdb2UL, 0x925ab42bUL, 0x046ab35cUL, 0xa7ffd7c2UL, 0x31cfd0b5UL,
0x8b9ed92cUL, 0x1daede5bUL, 0xb0c2649bUL, 0x26f263ecUL, 0x9ca36a75UL,
0x0a936d02UL, 0xa906099cUL, 0x3f360eebUL, 0x85670772UL, 0x13570005UL,
0x824abf95UL, 0x147ab8e2UL, 0xae2bb17bUL, 0x381bb60cUL, 0x9b8ed292UL,
0x0dbed5e5UL, 0xb7efdc7cUL, 0x21dfdb0bUL, 0xd4d2d386UL, 0x42e2d4f1UL,
0xf84b3dd6UL, 0x6e83da1fUL, 0xcd16be81UL, 0x5b26b9f6UL, 0xe177b06fUL,
0x7747b718UL, 0xe65a0888UL, 0x706a0fffUL, 0xca3b0666UL, 0x5c0b0111UL,
0xff9e658fUL, 0x69ae62f8UL, 0xd3ff6b61UL, 0x45cf6c16UL, 0x78e20aa0UL,
0xfeed20ddUL, 0x5483044eUL, 0xc2b30339UL, 0x612667a7UL, 0xf71660d0UL,
0x4d476949UL, 0xdb776e3eUL, 0x4a6ad1aeUL, 0xdc5ad6d9UL, 0x60bdf40UL,
0xf03bd837UL, 0x53aebca9UL, 0xc59ebbdeUL, 0x7fcfb247UL, 0xe9ffb530UL,
0x1cf2bdbdUL, 0x8ac2bacaUL, 0x3093b353UL, 0xa6a3b424UL, 0x0536d0baUL,
0x9306d7cdUL, 0x2957de54UL, 0xbf67d923UL, 0x2e7a66b3UL, 0xb84a61c4UL,
0x021b685dUL, 0x942d6f2aUL, 0x37be0bb4UL, 0xa18e0cc3UL, 0x1bdf055aUL,
0x8def022dUL
```

},

```
0x00000000UL, 0x41311b19UL, 0x82623632UL, 0xc3532d2bUL, 0x04c56c64UL,
0x45f4777dUL, 0x86a75a56UL, 0xc796414fUL, 0x088ad9c8UL, 0x49bbc2d1UL,
0x8aeefffaUL, 0xcdbd9f4e3UL, 0x0c4fb5acUL, 0x4d7eaeb5UL, 0x8e2d839eUL,
0xcf1c9887UL, 0x5112c24aUL, 0x1023d953UL, 0xd370f478UL, 0x9241ef61UL,
0x55d7ae2eUL, 0x14e6b537UL, 0xd7b5981cUL, 0x96848305UL, 0x59981b82UL,
0x18a9009bUL, 0xdbfa2db0UL, 0x9acb36a9UL, 0x5d5d77e6UL, 0x1c6c6cffUL,
0xdf3f41d4UL, 0x9e0e5acdUL, 0xa2248495UL, 0xe3159f8cUL, 0x2046b2a7UL,
0x6177a9beUL, 0xa6e1e8f1UL, 0xe7d0f3e8UL, 0x2483dec3UL, 0x65b2c5daUL,
0xaaae5d5dUL, 0xeb9f4644UL, 0x28c6b6fUL, 0x69fd7076UL, 0xae6b3139UL,
0xef5a2a20UL, 0x2c09070bUL, 0x6d381c12UL, 0xf33646dfUL, 0xb2075dc6UL,
0x715470edUL, 0x30656bf4UL, 0xf7f32abbUL, 0xb6c231a2UL, 0x75911c89UL,
0x34a00790UL, 0xfbbc9f17UL, 0xba8d840eUL, 0x79dea925UL, 0x38efb23cUL,
0xff79f373UL, 0xbe48e86aUL, 0x7d1bc541UL, 0x3c2ade58UL, 0x054f79f0UL,
0x447e62e9UL, 0x872d4fc2UL, 0xc61c54dbUL, 0x018a1594UL, 0x40bb0e8dUL,
0x83e823a6UL, 0xc2d938bfUL, 0x0dc5a038UL, 0x4cf4bb21UL, 0x8fa7960aUL,
0xce968d13UL, 0x0900cc5cUL, 0x4831d745UL, 0x8b62fa6eUL, 0xca53e177UL,
0x545dbbbaUL, 0x156ca0a3UL, 0xd63f8d88UL, 0x970e9691UL, 0x5098d7deUL,
0x11a9ccc7UL, 0xd2faelecUL, 0x93cbfaf5UL, 0x5cd76272UL, 0x1de6796bUL,
```

```

0xdeb55440UL, 0x9f844f59UL, 0x58120e16UL, 0x1923150fUL, 0xda703824UL,
0x9b41233dUL, 0xa76bfd65UL, 0xe65ae67cUL, 0x2509cb57UL, 0x6438d04eUL,
0xa3ae9101UL, 0xe29f8a18UL, 0x21cca733UL, 0x60fdbcb2aUL, 0xafc124adUL,
0xced03fb4UL, 0x2d83129fUL, 0x6cb20986UL, 0xab2448c9UL, 0xea1553d0UL,
0x29467efbUL, 0x687765e2UL, 0xf6793f2fUL, 0xb7482436UL, 0x741b091dUL,
0x352a1204UL, 0xf2bc534bUL, 0xb38d4852UL, 0x70de6579UL, 0x31ef7e60UL,
0xfef3e6e7UL, 0xbfc2fdfeUL, 0x7c91d0d5UL, 0x3da0cbccUL, 0xfa368a83UL,
0xbb07919aUL, 0x7854bcb1UL, 0x3965a7a8UL, 0x4b98833bUL, 0x0aa99822UL,
0xc9fab509UL, 0x88cbae10UL, 0x4f5def5fUL, 0x0e6cf446UL, 0xcd3fd96dUL,
0x8c0ec274UL, 0x43125af3UL, 0x022341eaUL, 0xc1706cc1UL, 0x804177d8UL,
0x47d73697UL, 0x06e62d8eUL, 0xc5b500a5UL, 0x84841bbcUL, 0x1a8a4171UL,
0x5bbb5a68UL, 0x98e87743UL, 0xd9d96c5aUL, 0x1e4f2d15UL, 0x5f7e360cUL,
0x9c2d1b27UL, 0xdd1c003eUL, 0x120098b9UL, 0x533183a0UL, 0x9062ae8bUL,
0xd153b592UL, 0x16c5f4ddUL, 0x57f4efc4UL, 0x94a7c2efUL, 0xd596d9f6UL,
0xe9bc07aeUL, 0xa88d1cb7UL, 0x6bde319cUL, 0x2aef2a85UL, 0xed796bcaUL,
0xac4870d3UL, 0x6f1b5df8UL, 0x2e2a46e1UL, 0xe136de66UL, 0xa007c57fUL,
0x6354e854UL, 0x2265f34dUL, 0xe5f3b202UL, 0xa4c2a91bUL, 0x67918430UL,
0x26a09f29UL, 0xb8aec5e4UL, 0xf99fdefdUL, 0x3accf3d6UL, 0x7bfde8cfUL,
0xbc6ba980UL, 0xfd5ab299UL, 0x3e099fb2UL, 0x7f3884abUL, 0xb0241c2cUL,
0xf1150735UL, 0x32462a1eUL, 0x73773107UL, 0xb4e17048UL, 0xf5d06b51UL,
0x3683467aUL, 0x77b25d63UL, 0x4ed7facbUL, 0x0fe6e1d2UL, 0xccb5ccf9UL,
0x8d84d7e0UL, 0x4a1296afUL, 0x0b238db6UL, 0xc870a09dUL, 0x8941bb84UL,
0x465d2303UL, 0x076c381aUL, 0xc43f1531UL, 0x850e0e28UL, 0x42984f67UL,
0x03a9547eUL, 0xc0fa7955UL, 0x81cb624cUL, 0x1fc53881UL, 0x5ef42398UL,
0x9da70eb3UL, 0xdc9615aaUL, 0x1b0054e5UL, 0x5a314ffcUL, 0x996262d7UL,
0xd85379ceUL, 0x174fe149UL, 0x567efa50UL, 0x952dd77bUL, 0xd41ccc62UL,
0x138a8d2dUL, 0x52bb9634UL, 0x91e8bb1fUL, 0xd0d9a006UL, 0xecf37e5eUL,
0xadc26547UL, 0x6e91486cUL, 0x2fa05375UL, 0xe836123aUL, 0xa9070923UL,
0x6a542408UL, 0x2b653f11UL, 0xe479a796UL, 0xa548bc8fUL, 0x661b91a4UL,
0x272a8abdUL, 0xe0bccbf2UL, 0xa18dd0ebUL, 0x62defdc0UL, 0x23efe6d9UL,
0xbde1bc14UL, 0xfcd0a70dUL, 0x3f838a26UL, 0x7eb2913fUL, 0xb924d070UL,
0xf815cb69UL, 0x3b46e642UL, 0x7a77fd5bUL, 0xb56b65dcUL, 0xf45a7ec5UL,
0x370953eeUL, 0x763848f7UL, 0xb1ae09b8UL, 0xf09f12a1UL, 0x33cc3f8aUL,
0x72fd2493UL

```

{,

```

0x00000000UL, 0x376ac201UL, 0x6ed48403UL, 0x59be4602UL, 0xdca80907UL,
0xebc2cb06UL, 0xb27c8d04UL, 0x85164f05UL, 0xb851130eUL, 0x8f3bd10fUL,
0xd685970dUL, 0xe1ef550cUL, 0x64f91a09UL, 0x5393d808UL, 0x0a2d9e0aUL,
0x3d475c0bUL, 0x70a3261cUL, 0x47c9e41dUL, 0x1e77a21fUL, 0x291d601eUL,
0xac0b2f1bUL, 0x9b61ed1aUL, 0xc2dfab18UL, 0xf5b56919UL, 0xc8f23512UL,
0xff98f713UL, 0xa626b111UL, 0x914c7310UL, 0x145a3c15UL, 0x2330fe14UL,
0x7a8eb816UL, 0x4de47a17UL, 0xe0464d38UL, 0xd72c8f39UL, 0x8e92c93bUL,
0xb9f80b3aUL, 0x3cee443fUL, 0x0b84863eUL, 0x523ac03cUL, 0x6550023dUL,
0x58175e36UL, 0x6f7d9c37UL, 0x36c3da35UL, 0x01a91834UL, 0x84bf5731UL,
0xb3d59530UL, 0xea6bd332UL, 0xdd011133UL, 0x90e56b2dUL, 0xa78fa925UL,
0xfe31ef27UL, 0xc95b2d26UL, 0x4c4d6223UL, 0x7b27a022UL, 0x2299e620UL,
0x15f32421UL, 0x28b4782aUL, 0x1fdeba2bUL, 0x4660fc29UL, 0x710a3e28UL,
0xf41c712dUL, 0xc376b32cUL, 0x9ac8f52eUL, 0xada2372fUL, 0xc08d9a70UL,
0xf7e75871UL, 0xae591e73UL, 0x9933dc72UL, 0x1c259377UL, 0x2b4f5176UL,
0x72f11774UL, 0x459bd575UL, 0x78dc897eUL, 0x4fb64b7fUL, 0x16080d7dUL,
0x2162cf7cUL, 0xa7448079UL, 0x931e4278UL, 0xcaa0047aUL, 0xfdcac67bUL,
0xb02ebc6cUL, 0x87447e6dUL, 0xdfa386fUL, 0xe990fa6eUL, 0x6c86b56bUL,
0x5bec776aUL, 0x02523168UL, 0x3538f369UL, 0x087faf62UL, 0x3f156d63UL,
0x66ab2b61UL, 0x51c1e960UL, 0xd4d7a665UL, 0xe3bd6464UL, 0xba032266UL,
0x8d69e067UL, 0x20cbd748UL, 0x17a11549UL, 0x4e1f534bUL, 0x7975914aUL,
0xf63de4fUL, 0xcb091c4eUL, 0x92b75a4cUL, 0xa5dd984dUL, 0x989ac446UL,
0xaf00647UL, 0xf64e4045UL, 0xc1248244UL, 0x4432cd41UL, 0x73580f40UL,
0x2ae64942UL, 0x1d8c8b43UL, 0x5068f154UL, 0x67023355UL, 0x3ebc7557UL,
0x09d6b756UL, 0x8cc0f853UL, 0xbbaa3a52UL, 0xe2147c50UL, 0xd57ebe51UL,
0xe839e25aUL, 0xdf53205bUL, 0x86ed6659UL, 0xb187a458UL, 0x3491eb5dUL,
0x03fb295cUL, 0x5a456f5eUL, 0x6d2fad5fUL, 0x801b35e1UL, 0xb771f7e0UL,
0xeecfb1e2UL, 0xd9a573e3UL, 0x5cb33ce6UL, 0x6bd9fee7UL, 0x3267b8e5UL,
0x050d7ae4UL, 0x384a26efUL, 0x0f20e4eeUL, 0x569ea2ecUL, 0x61f460edUL,
0xe4e22fe8UL, 0xd388ede9UL, 0xa36abebUL, 0xbd5c69eaUL, 0xf0b813fdUL,
0xc7d2d1fcUL, 0x9e6c97feUL, 0xa90655ffUL, 0x2c101afaUL, 0x1b7ad8fbUL,
0x42c49ef9UL, 0x75ae5cf8UL, 0x48e900f3UL, 0x7f83c2f2UL, 0x263d84f0UL,
0x115746f1UL, 0x944109f4UL, 0xa32bcbf5UL, 0xfa958df7UL, 0xcdff4ff6UL,
0x605d78d9UL, 0x5737bad8UL, 0x0e89fcdadUL, 0x39e33edbUL, 0xbcf571deUL,
0x8b9fb3dfUL, 0xd221f5ddUL, 0xe54b37dcUL, 0xd80c6bd7UL, 0xef66a9d6UL,
0xb6d8efd4UL, 0x81b22dd5UL, 0x04a462d0UL, 0x33cea0d1UL, 0x6a70e6d3UL,
0x5d1a24d2UL, 0x10fe5ec5UL, 0x27949cc4UL, 0x7e2adac6UL, 0x494018c7UL,
0xcc5657c2UL, 0xfb3c95c3UL, 0xa282d3c1UL, 0x95e811c0UL, 0xa8af4dcbUL,
0x9fc58fcaUL, 0xc67bc9c8UL, 0xf1110bc9UL, 0x740744ccUL, 0x436d86cdUL,

```

```
0x1ad3c0cfUL, 0x2db902ceUL, 0x4096af91UL, 0x77fc6d90UL, 0x2e422b92UL,
0x1928e993UL, 0x9c3ea696UL, 0xab546497UL, 0xf2ea2295UL, 0xc580e094UL,
0xf8c7bc9fUL, 0xcfad7e9eUL, 0x9613389cUL, 0xa179fa9dUL, 0x246fb598UL,
0x13057799UL, 0x4abb319bUL, 0x7dd1f39aUL, 0x3035898dUL, 0x075f4b8cUL,
0x5ee10d8eUL, 0x698bcf8fUL, 0xec9d808aUL, 0xdbf7428bUL, 0x82490489UL,
0xb523c688UL, 0x88649a83UL, 0xbf0e5882UL, 0xe6b01e80UL, 0xd1dad8c1UL,
0x54cc9384UL, 0x63a65185UL, 0x3a181787UL, 0x0d72d586UL, 0xa0d0e2a9UL,
0x97ba20a8UL, 0xce0466aaUL, 0xf96ea4abUL, 0x7c78ebaeUL, 0x4b1229afUL,
0x12ac6fadUL, 0x25c6adacUL, 0x1881f1a7UL, 0x2feb33a6UL, 0x765575a4UL,
0x413fb7a5UL, 0xc429f8a0UL, 0xf3433aa1UL, 0xaafd7ca3UL, 0x9d97bea2UL,
0xd073c4b5UL, 0xe71906b4UL, 0xbea740b6UL, 0x89cd82b7UL, 0x0cdcbdb2UL,
0x3bb10fb3UL, 0x620f49b1UL, 0x55658bb0UL, 0x6822d7bbUL, 0x5f4815baUL,
0x06f653b8UL, 0x319c91b9UL, 0xb48adebcUL, 0x83e01cbdUL, 0xda5e5abfUL,
0xed3498beUL
```

```
0x00000000UL, 0x6567bcb8UL, 0x8bc809aaUL, 0xeeafb512UL, 0x5797628fUL,
0x32f0de37UL, 0xdc5f6b25UL, 0xb938d79dUL, 0xef28b4c5UL, 0x8a4f087dUL,
0x64e0bd6fUL, 0x018701d7UL, 0xb8bfd64aUL, 0xdd86af2UL, 0x3377dfe0UL,
0x56106358UL, 0x9f571950UL, 0xfa30a5e8UL, 0x149f10faUL, 0x71f8ac42UL,
0xc8c07bdfUL, 0xada7c767UL, 0x43087275UL, 0x266fcecUL, 0x707fad95UL,
0x1518112dUL, 0xfbb7a43fUL, 0x9ed01887UL, 0x27e8cf1aUL, 0x428f73a2UL,
0xac20c6b0UL, 0xc9477a08UL, 0x3ead32a0UL, 0x5bc88e18UL, 0xb5673b0aUL,
0xd00087b2UL, 0x6938502fUL, 0x0c5fec97UL, 0xe2f05985UL, 0x8797e53dUL,
0xd1878665UL, 0xb4e03addUL, 0x5a4f8fcfUL, 0x3f283377UL, 0x8610e4eaUL,
0xe3775852UL, 0x0dd8ed40UL, 0x68bf51f8UL, 0xa1f82bf0UL, 0xc49f9748UL,
0x2a30225aUL, 0x4f579ee2UL, 0xf66f497fUL, 0x9308f5c7UL, 0x7da740d5UL,
0x18c0fc6dUL, 0x4ed09f35UL, 0x2bb7238dUL, 0xc518969fUL, 0xa07f2a27UL,
0x1947fdbbaUL, 0x7c204102UL, 0x928ff410UL, 0xf7e848a8UL, 0x3d58149bUL,
0x583fa823UL, 0xb6901d31UL, 0xd3f7a189UL, 0x6acf7614UL, 0x0fa8caacUL,
0xe1077fbeUL, 0x8460c306UL, 0xd270a05eUL, 0xb7171ce6UL, 0x59b8a9f4UL,
0x3cdf154cUL, 0x85e7c2d1UL, 0xe0807e69UL, 0x0e2fcb7bUL, 0x6b4877c3UL,
0xa20f0dcbUL, 0xc768b173UL, 0x29c70461UL, 0x4ca0b8d9UL, 0xf5986f44UL,
0x90ffd3fcUL, 0x7e5066eeUL, 0x1b37da56UL, 0x4d27b90eUL, 0x284005b6UL,
0xc6efb0a4UL, 0xa3880c1cUL, 0x1ab0db81UL, 0x7fd76739UL, 0x9178d22bUL,
0xf41f6e93UL, 0x03f7263bUL, 0x66909a83UL, 0x883f2f91UL, 0xed589329UL,
0x546044b4UL, 0x3107f80cUL, 0xdfa84d1eUL, 0xbacffa6UL, 0xecdf92feUL,
0x89b82e46UL, 0x67179b54UL, 0x027027ecUL, 0xbb48f071UL, 0xde2f4cc9UL,
0x3080f9dbUL, 0x55e74563UL, 0x9ca03f6bUL, 0xf9c783d3UL, 0x176836c1UL,
0x720f8a79UL, 0xcb375de4UL, 0xae50e15cUL, 0x40ff544eUL, 0x2598e8f6UL,
0x73888baeUL, 0x16ef3716UL, 0xf8408204UL, 0x9d273ebcUL, 0x241fe921UL,
0x41785599UL, 0xafd7e08bUL, 0xcab05c33UL, 0x3bb659edUL, 0x5ed1e555UL,
0xb07e5047UL, 0xd519ecffUL, 0x6c213b62UL, 0x094687daUL, 0xe7e932c8UL,
0x828e8e70UL, 0xd49eed28UL, 0xb1f95190UL, 0x5f56e482UL, 0x3a31583aUL,
0x83098fa7UL, 0xe66e331fUL, 0x08c1860dUL, 0x6da63ab5UL, 0xa4e140bdUL,
0xc186fc05UL, 0x2f294917UL, 0x4a4ef5afUL, 0xf3762232UL, 0x96119e8aUL,
0x78be2b98UL, 0x1dd99720UL, 0x4bc9f478UL, 0x2eae48c0UL, 0xc001fdd2UL,
0xa566416aUL, 0x1c5e96f7UL, 0x79392a4fUL, 0x97969f5dUL, 0xf2f123e5UL,
0x05196b4dUL, 0x607ed7f5UL, 0x8ed162e7UL, 0xebb6de5fUL, 0x528e09c2UL,
0x37e9b57aUL, 0xd9460068UL, 0xbc21bcd0UL, 0xea31df88UL, 0x8f566330UL,
0x61f9d622UL, 0x049e6a9aUL, 0xbda6bd07UL, 0xd8c101bfUL, 0x366eb4adUL,
0x53090815UL, 0x9a4e721dUL, 0xff29cea5UL, 0x11867bb7UL, 0x74e1c70fUL,
0xcdd91092UL, 0xa8beac2aUL, 0x46111938UL, 0x2376a580UL, 0x7566c6d8UL,
0x10017a60UL, 0xfeaecf72UL, 0x9bc973caUL, 0x22f1a457UL, 0x479618efUL,
0xa939adfdUL, 0xcc5e1145UL, 0x06ee4d76UL, 0x6389f1ceUL, 0x8d2644dcUL,
0xe841f864UL, 0x51792ff9UL, 0x341e9341UL, 0xdab12653UL, 0xbfd69aebUL,
0xe9c6f9b3UL, 0x8ca1450bUL, 0x620ef019UL, 0x07694ca1UL, 0xbe519b3cUL,
0xdb362784UL, 0x35999296UL, 0x50fe2e2eUL, 0x99b95426UL, 0xfcdee89eUL,
0x12715d8cUL, 0x7716e134UL, 0xce2e36a9UL, 0xab498a11UL, 0x45e63f03UL,
0x208183bbUL, 0x7691e0e3UL, 0x13f65c5bUL, 0xfd59e949UL, 0x983e55f1UL,
0x2106826cUL, 0x44613ed4UL, 0xaace8bc6UL, 0xcfa9377eUL, 0x38417fd6UL,
0x5d26c36eUL, 0xb389767cUL, 0xd6eeca4UL, 0x6fd61d59UL, 0x0ab1a1e1UL,
0xe41e14f3UL, 0x8179a84bUL, 0xd769cb13UL, 0xb20e77abUL, 0x5ca1c2b9UL,
0x39c67e01UL, 0x80fea99cUL, 0xe5991524UL, 0xb36a036UL, 0x6e511c80UL,
0xa7166686UL, 0xc271da3eUL, 0x2cde6f2cUL, 0x49b9d394UL, 0xf0810409UL,
0x95e6b8b1UL, 0x7b490da3UL, 0x1e2eb11bUL, 0x483ed243UL, 0x2d596efbUL,
0xc3f6dbe9UL, 0xa6916751UL, 0x1fa9b0ccUL, 0x7ace0c74UL, 0x9461b966UL,
0xf10605deUL
```

#endif

};

```

/* deflate.c -- compress data using the deflation algorithm
 * Copyright (C) 1995-2005 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 * ALGORITHM
 *
 * The "deflation" process depends on being able to identify portions
 * of the input text which are identical to earlier input (within a
 * sliding window trailing behind the input currently being processed).
 *
 * The most straightforward technique turns out to be the fastest for
 * most input files: try all possible matches and select the longest.
 * The key feature of this algorithm is that insertions into the string
 * dictionary are very simple and thus fast, and deletions are avoided
 * completely. Insertions are performed at each input character, whereas
 * string matches are performed only when the previous match ends. So it
 * is preferable to spend more time in matches to allow very fast string
 * insertions and avoid deletions. The matching algorithm for small
 * strings is inspired from that of Rabin & Karp. A brute force approach
 * is used to find longer strings when a small match has been found.
 * A similar algorithm is used in comic (by Jan-Mark Wams) and freeze
 * (by Leonid Broukhis).
 *
 * A previous version of this file used a more sophisticated algorithm
 * (by Fiala and Greene) which is guaranteed to run in linear amortized
 * time, but has a larger average cost, uses more memory and is patented.
 * However the F&G algorithm may be faster for some highly redundant
 * files if the parameter max_chain_length (described below) is too large.
 *
 * ACKNOWLEDGEMENTS
 *
 * The idea of lazy evaluation of matches is due to Jan-Mark Wams, and
 * I found it in 'freeze' written by Leonid Broukhis.
 * Thanks to many people for bug reports and testing.
 *
 * REFERENCES
 *
 * Deutsch, L.P., "DEFLATE Compressed Data Format Specification".
 * Available in http://www.ietf.org/rfc/rfc1951.txt
 *
 * A description of the Rabin and Karp algorithm is given in the book
 * "Algorithms" by R. Sedgewick, Addison-Wesley, p252.
 *
 * Fiala, E.R., and Greene, D.H.
 * Data Compression with Finite Windows, Comm.ACM, 32,4 (1989) 490-595
 */

/* @(#) $Id: deflate.c,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#include "deflate.h"

const char deflate_copyright[] =
    " deflate 1.2.3 Copyright 1995-2005 Jean-loup Gailly " ;

/*
 * If you use the zlib library in a product, an acknowledgment is welcome
 * in the documentation of your product. If for some reason you cannot
 * include such an acknowledgment, I would appreciate that you keep this
 * copyright string in the executable of your product.
 */

/*
 * =====
 * Function prototypes.
 */

typedef enum {
    need_more,      /* block not completed, need more input or more output */
    block_done,     /* block flush performed */
    finish_started, /* finish started, need only more output at next deflate */
    finish_done,    /* finish done, accept no more input or output */
} block_state;

typedef block_state (*compress_func) OF((deflate_state *s, int flush));
/* Compression function. Returns the block state after the call. */

```

```

local void fill_window      OF((deflate_state *s));
local block_state deflate_stored OF((deflate_state *s, int flush));
local block_state deflate_fast  OF((deflate_state *s, int flush));
#ifdef FASTEST
local block_state deflate_slow  OF((deflate_state *s, int flush));
#endif
local void lm_init          OF((deflate_state *s));
local void putShortMSB      OF((deflate_state *s, uInt b));
local void flush_pending    OF((z_streamp strm));
local int read_buf          OF((z_streamp strm, Bytef *buf, unsigned size));
#ifdef FASTEST
#ifdef ASMV
    void match_init OF((void)); /* asm code initialization */
    uInt longest_match OF((deflate_state *s, IPos cur_match));
#else
local uInt longest_match OF((deflate_state *s, IPos cur_match));
#endif
#endif
local uInt longest_match_fast OF((deflate_state *s, IPos cur_match));

#ifdef DEBUG
local void check_match OF((deflate_state *s, IPos start, IPos match,
                          int length));
#endif

/* =====
 * Local data
 */

#define NIL 0
/* Tail of hash chains */

#ifdef TOO_FAR
# define TOO_FAR 4096
#endif
/* Matches of length 3 are discarded if their distance exceeds TOO_FAR */

#define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)
/* Minimum amount of lookahead, except at the end of the input file.
 * See deflate.c for comments about the MIN_MATCH+1.
 */

/* Values for max_lazy_match, good_match and max_chain_length, depending on
 * the desired pack level (0..9). The values given below have been tuned to
 * exclude worst case performance for pathological files. Better values may be
 * found for specific files.
 */
typedef struct config_s {
    ush good_length; /* reduce lazy search above this match length */
    ush max_lazy;    /* do not perform lazy search above this match length */
    ush nice_length; /* quit search above this match length */
    ush max_chain;
    compress_func func;
} config;

#ifdef FASTEST
local const config configuration_table[2] = {
/*    good lazy nice chain */
/* 0 */ {0, 0, 0, 0, deflate_stored}, /* store only */
/* 1 */ {4, 4, 8, 4, deflate_fast}; /* max speed, no lazy matches */
#else
local const config configuration_table[10] = {
/*    good lazy nice chain */
/* 0 */ {0, 0, 0, 0, deflate_stored}, /* store only */
/* 1 */ {4, 4, 8, 4, deflate_fast}, /* max speed, no lazy matches */
/* 2 */ {4, 5, 16, 8, deflate_fast},
/* 3 */ {4, 6, 32, 32, deflate_fast},

/* 4 */ {4, 4, 16, 16, deflate_slow}, /* lazy matches */
/* 5 */ {8, 16, 32, 32, deflate_slow},
/* 6 */ {8, 16, 128, 128, deflate_slow},
/* 7 */ {8, 32, 128, 256, deflate_slow},
/* 8 */ {32, 128, 258, 1024, deflate_slow},

```

```

/* 9 */ {32, 258, 258, 4096, deflate_slow}}; /* max compression */
#endif

/* Note: the deflate() code requires max_lazy >= MIN_MATCH and max_chain >= 4
 * For deflate_fast() (levels <= 3) good is ignored and lazy has a different
 * meaning.
 */

#define EQUAL 0
/* result of memcmp for equal strings */

#ifndef NO_DUMMY_DECL
struct static_tree_desc_s {int dummy;}; /* for buggy compilers */
#endif

/* =====
 * Update a hash value with the given input byte
 * IN assertion: all calls to to UPDATE_HASH are made with consecutive
 * input characters, so that a running hash key can be computed from the
 * previous key instead of complete recalculation each time.
 */
#define UPDATE_HASH(s,h,c) (h = (((h)<<s->hash_shift) ^ (c)) & s->hash_mask)

/* =====
 * Insert string str in the dictionary and set match_head to the previous head
 * of the hash chain (the most recent string with same hash key). Return
 * the previous length of the hash chain.
 * If this file is compiled with -DFASTEST, the compression level is forced
 * to 1, and no hash chains are maintained.
 * IN assertion: all calls to to INSERT_STRING are made with consecutive
 * input characters and the first MIN_MATCH bytes of str are valid
 * (except for the last MIN_MATCH-1 bytes of the input file).
 */
#ifdef FASTEST
#define INSERT_STRING(s, str, match_head) \
    (UPDATE_HASH(s, s->ins_h, s->window[(str) + (MIN_MATCH-1)]), \
     match_head = s->head[s->ins_h], \
     s->head[s->ins_h] = (Pos)(str))
#else
#define INSERT_STRING(s, str, match_head) \
    (UPDATE_HASH(s, s->ins_h, s->window[(str) + (MIN_MATCH-1)]), \
     match_head = s->prev[(str) & s->w_mask] = s->head[s->ins_h], \
     s->head[s->ins_h] = (Pos)(str))
#endif

/* =====
 * Initialize the hash table (avoiding 64K overflow for 16 bit systems).
 * prev[] will be initialized on the fly.
 */
#define CLEAR_HASH(s) \
    s->head[s->hash_size-1] = NIL; \
    zmemzero((Bytef *)s->head, (unsigned)(s->hash_size-1)*sizeof(*s->head));

/* ===== */
int ZEXPORT deflateInit_(strm, level, version, stream_size)
    z_streamp strm;
    int level;
    const char *version;
    int stream_size;
{
    return deflateInit2_(strm, level, Z_DEFLATED, MAX_WBITS, DEF_MEM_LEVEL,
        Z_DEFAULT_STRATEGY, version, stream_size);
    /* To do: ignore strm->next_in if we use it as window */
}

/* ===== */
int ZEXPORT deflateInit2_(strm, level, method, windowBits, memLevel, strategy,
    version, stream_size)
    z_streamp strm;
    int level;
    int method;
    int windowBits;
    int memLevel;

```

```

    int strategy;
    const char *version;
    int stream_size;
}

deflate_state *s;
int wrap = 1;
static const char my_version[] = ZLIB_VERSION;

ushf *overlay;
/* We overlay pending_buf and d_buf+l_buf. This works since the average
 * output size for (length,distance) codes is <= 24 bits.
 */

if (version == Z_NULL || version[0] != my_version[0] ||
    stream_size != sizeof(z_stream)) {
    return Z_VERSION_ERROR;
}
if (strm == Z_NULL) return Z_STREAM_ERROR;

strm->msg = Z_NULL;
if (strm->zalloc == (alloc_func)0) {
    strm->zalloc = zcalloc;
    strm->opaque = (voidpf)0;
}
if (strm->zfree == (free_func)0) strm->zfree = zcfree;

#ifdef FASTEST
    if (level != 0) level = 1;
#else
    if (level == Z_DEFAULT_COMPRESSION) level = 6;
#endif

    if (windowBits < 0) { /* suppress zlib wrapper */
        wrap = 0;
        windowBits = -windowBits;
    }
#ifdef GZIP
    else if (windowBits > 15) {
        wrap = 2; /* write gzip wrapper instead */
        windowBits -= 16;
    }
#endif
    if (memLevel < 1 || memLevel > MAX_MEM_LEVEL || method != Z_DEFLATED ||
        windowBits < 8 || windowBits > 15 || level < 0 || level > 9 ||
        strategy < 0 || strategy > Z_FIXED) {
        return Z_STREAM_ERROR;
    }
    if (windowBits == 8) windowBits = 9; /* until 256-byte window bug fixed */
    s = (deflate_state *) ZALLOC(strm, 1, sizeof(deflate_state));
    if (s == Z_NULL) return Z_MEM_ERROR;
    strm->state = (struct internal_state FAR *)s;
    s->strm = strm;

    s->wrap = wrap;
    s->gzhead = Z_NULL;
    s->w_bits = windowBits;
    s->w_size = 1 << s->w_bits;
    s->w_mask = s->w_size - 1;

    s->hash_bits = memLevel + 7;
    s->hash_size = 1 << s->hash_bits;
    s->hash_mask = s->hash_size - 1;
    s->hash_shift = ((s->hash_bits+MIN_MATCH-1)/MIN_MATCH);

    s->window = (Bytef *) ZALLOC(strm, s->w_size, 2*sizeof(Byte));
    s->prev = (Posf *) ZALLOC(strm, s->w_size, sizeof(Pos));
    s->head = (Posf *) ZALLOC(strm, s->hash_size, sizeof(Pos));

    s->lit_bufsize = 1 << (memLevel + 6); /* 16K elements by default */

    overlay = (ushf *) ZALLOC(strm, s->lit_bufsize, sizeof(ush)+2);
    s->pending_buf = (uchf *) overlay;
    s->pending_buf_size = (ulg)s->lit_bufsize * (sizeof(ush)+2L);

```

```

    if (s->window == Z_NULL || s->prev == Z_NULL || s->head == Z_NULL ||
        s->pending_buf == Z_NULL) {
        s->status = FINISH_STATE;
        strm->msg = (char*)ERR_MSG(Z_MEM_ERROR);
        deflateEnd (strm);
        return Z_MEM_ERROR;
    }
    s->d_buf = overlay + s->lit_bufsize/sizeof(ush);
    s->l_buf = s->pending_buf + (1+sizeof(ush))*s->lit_bufsize;

    s->level = level;
    s->strategy = strategy;
    s->method = (Byte)method;

    return deflateReset(strm);
}

/* ===== */
int ZEXPORT deflateSetDictionary (strm, dictionary, dictLength)
    z_streamp strm;
    const Bytef *dictionary;
    uInt dictLength;
{
    deflate_state *s;
    uInt length = dictLength;
    uInt n;
    IPos hash_head = 0;

    if (strm == Z_NULL || strm->state == Z_NULL || dictionary == Z_NULL ||
        strm->state->wrap == 2 ||
        (strm->state->wrap == 1 && strm->state->status != INIT_STATE))
        return Z_STREAM_ERROR;

    s = strm->state;
    if (s->wrap)
        strm->adler = Adler32(strm->adler, dictionary, dictLength);

    if (length < MIN_MATCH) return Z_OK;
    if (length > MAX_DIST(s)) {
        length = MAX_DIST(s);
        dictionary += dictLength - length; /* use the tail of the dictionary */
    }
    zmemcpy(s->window, dictionary, length);
    s->strstart = length;
    s->block_start = (long)length;

    /* Insert all strings in the hash table (except for the last two bytes).
     * s->lookahead stays null, so s->ins_h will be recomputed at the next
     * call of fill_window.
     */
    s->ins_h = s->window[0];
    UPDATE_HASH(s, s->ins_h, s->window[1]);
    for (n = 0; n <= length - MIN_MATCH; n++) {
        INSERT_STRING(s, n, hash_head);
    }
    if (hash_head) hash_head = 0; /* to make compiler happy */
    return Z_OK;
}

/* ===== */
int ZEXPORT deflateReset (strm)
    z_streamp strm;
{
    deflate_state *s;

    if (strm == Z_NULL || strm->state == Z_NULL ||
        strm->zalloc == (alloc_func)0 || strm->zfree == (free_func)0) {
        return Z_STREAM_ERROR;
    }

    strm->total_in = strm->total_out = 0;
    strm->msg = Z_NULL; /* use zfree if we ever allocate msg dynamically */
    strm->data_type = Z_UNKNOWN;

```



```

s = (deflate_state *)strm->state;
s->pending = 0;
s->pending_out = s->pending_buf;

if (s->wrap < 0) {
    s->wrap = -s->wrap; /* was made negative by deflate(..., Z_FINISH); */
}
s->status = s->wrap ? INIT_STATE : BUSY_STATE;
strm->adler =
#ifdef GZIP
    s->wrap == 2 ? crc32(0L, Z_NULL, 0) :
#endif
    Adler32(0L, Z_NULL, 0);
s->last_flush = Z_NO_FLUSH;

_tr_init(s);
lm_init(s);

return Z_OK;
}

/* ===== */
int ZEXPORT deflateSetHeader (strm, head)
    z_streamp strm;
    gz_headerp head;
{
    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    if (strm->state->wrap != 2) return Z_STREAM_ERROR;
    strm->state->gzhead = head;
    return Z_OK;
}

/* ===== */
int ZEXPORT deflatePrime (strm, bits, value)
    z_streamp strm;
    int bits;
    int value;
{
    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    strm->state->bi_valid = bits;
    strm->state->bi_buf = (ush)(value & ((1 << bits) - 1));
    return Z_OK;
}

/* ===== */
int ZEXPORT deflateParams(strm, level, strategy)
    z_streamp strm;
    int level;
    int strategy;
{
    deflate_state *s;
    compress_func func;
    int err = Z_OK;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    s = strm->state;

#ifdef FASTEST
    if (level != 0) level = 1;
#else
    if (level == Z_DEFAULT_COMPRESSION) level = 6;
#endif
    if (level < 0 || level > 9 || strategy < 0 || strategy > Z_FIXED) {
        return Z_STREAM_ERROR;
    }
    func = configuration_table[s->level].func;

    if (func != configuration_table[level].func && strm->total_in != 0) {
        /* Flush the last buffer: */
        err = deflate(strm, Z_PARTIAL_FLUSH);
    }
    if (s->level != level) {
        s->level = level;
        s->max_lazy_match = configuration_table[level].max_lazy;
    }
}

```

```

        s->good_match      = configuration_table[level].good_length;
        s->nice_match       = configuration_table[level].nice_length;
        s->max_chain_length = configuration_table[level].max_chain;
    }
    s->strategy = strategy;
    return err;
}

/* ===== */
int ZEXPORT deflateTune(strm, good_length, max_lazy, nice_length, max_chain)
    z_streamp strm;
    int good_length;
    int max_lazy;
    int nice_length;
    int max_chain;
{
    deflate_state *s;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    s = strm->state;
    s->good_match = good_length;
    s->max_lazy_match = max_lazy;
    s->nice_match = nice_length;
    s->max_chain_length = max_chain;
    return Z_OK;
}

/* =====
 * For the default windowBits of 15 and memLevel of 8, this function returns
 * a close to exact, as well as small, upper bound on the compressed size.
 * They are coded as constants here for a reason--if the #define's are
 * changed, then this function needs to be changed as well. The return
 * value for 15 and 8 only works for those exact settings.
 *
 * For any setting other than those defaults for windowBits and memLevel,
 * the value returned is a conservative worst case for the maximum expansion
 * resulting from using fixed blocks instead of stored blocks, which deflate
 * can emit on compressed data for some combinations of the parameters.
 *
 * This function could be more sophisticated to provide closer upper bounds
 * for every combination of windowBits and memLevel, as well as wrap.
 * But even the conservative upper bound of about 14% expansion does not
 * seem onerous for output buffer allocation.
 */
uLong ZEXPORT deflateBound(strm, sourceLen)
    z_streamp strm;
    uLong sourceLen;
{
    deflate_state *s;
    uLong destLen;

    /* conservative upper bound */
    destLen = sourceLen +
        ((sourceLen + 7) >> 3) + ((sourceLen + 63) >> 6) + 11;

    /* if can't get parameters, return conservative bound */
    if (strm == Z_NULL || strm->state == Z_NULL)
        return destLen;

    /* if not default parameters, return conservative bound */
    s = strm->state;
    if (s->w_bits != 15 || s->hash_bits != 8 + 7)
        return destLen;

    /* default settings: return tight bound for that case */
    return compressBound(sourceLen);
}

/* =====
 * Put a short in the pending buffer. The 16-bit value is put in MSB order.
 * IN assertion: the stream state is correct and there is enough room in
 * pending_buf.
 */
local void putShortMSB (s, b)

```

```

deflate_state *s;
uInt b;
{
    put_byte(s, (Byte)(b >> 8));
    put_byte(s, (Byte)(b & 0xff));
}

/* =====
 * Flush as much pending output as possible. All deflate() output goes
 * through this function so some applications may wish to modify it
 * to avoid allocating a large strm->next_out buffer and copying into it.
 * (See also read_buf()).
 */
local void flush_pending(strm)
    z_streamp strm;
{
    unsigned len = strm->state->pending;

    if (len > strm->avail_out) len = strm->avail_out;
    if (len == 0) return;

    zmemcpy(strm->next_out, strm->state->pending_out, len);
    strm->next_out += len;
    strm->state->pending_out += len;
    strm->total_out += len;
    strm->avail_out -= len;
    strm->state->pending -= len;
    if (strm->state->pending == 0) {
        strm->state->pending_out = strm->state->pending_buf;
    }
}

/* ===== */
int ZEXPORT deflate (strm, flush)
    z_streamp strm;
    int flush;
{
    int old_flush; /* value of flush param for previous deflate call */
    deflate_state *s;

    if (strm == Z_NULL || strm->state == Z_NULL ||
        flush > Z_FINISH || flush < 0) {
        return Z_STREAM_ERROR;
    }
    s = strm->state;

    if (strm->next_out == Z_NULL ||
        (strm->next_in == Z_NULL && strm->avail_in != 0) ||
        (s->status == FINISH_STATE && flush != Z_FINISH)) {
        ERR_RETURN(strm, Z_STREAM_ERROR);
    }
    if (strm->avail_out == 0) ERR_RETURN(strm, Z_BUF_ERROR);

    s->strm = strm; /* just in case */
    old_flush = s->last_flush;
    s->last_flush = flush;

    /* Write the header */
    if (s->status == INIT_STATE) {
#ifdef GZIP
        if (s->wrap == 2) {
            strm->adler = crc32(0L, Z_NULL, 0);
            put_byte(s, 31);
            put_byte(s, 139);
            put_byte(s, 8);
            if (s->gzhead == NULL) {
                put_byte(s, 0);
                put_byte(s, 0);
                put_byte(s, 0);
                put_byte(s, 0);
                put_byte(s, 0);
                put_byte(s, s->level == 9 ? 2 :
                    (s->strategy >= Z_HUFFMAN_ONLY || s->level < 2 ?
                     4 : 0));

```

```

        put_byte(s, OS_CODE);
        s->status = BUSY_STATE;
    }
    else {
        put_byte(s, (s->gzhead->text ? 1 : 0) +
            (s->gzhead->hcrc ? 2 : 0) +
            (s->gzhead->extra == Z_NULL ? 0 : 4) +
            (s->gzhead->name == Z_NULL ? 0 : 8) +
            (s->gzhead->comment == Z_NULL ? 0 : 16)
        );
        put_byte(s, (Byte)(s->gzhead->time & 0xff));
        put_byte(s, (Byte)((s->gzhead->time >> 8) & 0xff));
        put_byte(s, (Byte)((s->gzhead->time >> 16) & 0xff));
        put_byte(s, (Byte)((s->gzhead->time >> 24) & 0xff));
        put_byte(s, s->level == 9 ? 2 :
            (s->strategy >= Z_HUFFMAN_ONLY || s->level < 2 ?
            4 : 0));
        put_byte(s, s->gzhead->os & 0xff);
        if (s->gzhead->extra != NULL) {
            put_byte(s, s->gzhead->extra_len & 0xff);
            put_byte(s, (s->gzhead->extra_len >> 8) & 0xff);
        }
        if (s->gzhead->hcrc)
            strm->adler = crc32(strm->adler, s->pending_buf,
                s->pending);

        s->gzindex = 0;
        s->status = EXTRA_STATE;
    }
}
else
#endif
{
    uInt header = (Z_DEFLATED + ((s->w_bits-8)<<4)) << 8;
    uInt level_flags;

    if (s->strategy >= Z_HUFFMAN_ONLY || s->level < 2)
        level_flags = 0;
    else if (s->level < 6)
        level_flags = 1;
    else if (s->level == 6)
        level_flags = 2;
    else
        level_flags = 3;
    header |= (level_flags << 6);
    if (s->strstart != 0) header |= PRESET_DICT;
    header += 31 - (header % 31);

    s->status = BUSY_STATE;
    putShortMSB(s, header);

    /* Save the Adler32 of the preset dictionary: */
    if (s->strstart != 0) {
        putShortMSB(s, (uInt)(strm->adler >> 16));
        putShortMSB(s, (uInt)(strm->adler & 0xffff));
    }
    strm->adler = adler32(0L, Z_NULL, 0);
}
}
#endif
GZIP
if (s->status == EXTRA_STATE) {
    if (s->gzhead->extra != NULL) {
        uInt beg = s->pending; /* start of bytes to update crc */

        while (s->gzindex < (s->gzhead->extra_len & 0xffff)) {
            if (s->pending == s->pending_buf_size) {
                if (s->gzhead->hcrc && s->pending > beg)
                    strm->adler = crc32(strm->adler, s->pending_buf + beg,
                        s->pending - beg);
                flush_pending(strm);
                beg = s->pending;
                if (s->pending == s->pending_buf_size)
                    break;
            }
            put_byte(s, s->gzhead->extra[s->gzindex]);
        }
    }
}

```

```

        s->gzindex++;
    }
    if (s->gzhead->hcrc && s->pending > beg)
        strm->adler = crc32(strm->adler, s->pending_buf + beg,
                             s->pending - beg);
    if (s->gzindex == s->gzhead->extra_len) {
        s->gzindex = 0;
        s->status = NAME_STATE;
    }
}
else
    s->status = NAME_STATE;
}
if (s->status == NAME_STATE) {
    if (s->gzhead->name != NULL) {
        uInt beg = s->pending; /* start of bytes to update crc */
        int val;

        do {
            if (s->pending == s->pending_buf_size) {
                if (s->gzhead->hcrc && s->pending > beg)
                    strm->adler = crc32(strm->adler, s->pending_buf + beg,
                                           s->pending - beg);

                flush_pending(strm);
                beg = s->pending;
                if (s->pending == s->pending_buf_size) {
                    val = 1;
                    break;
                }
            }
            val = s->gzhead->name[s->gzindex++];
            put_byte(s, val);
        } while (val != 0);
        if (s->gzhead->hcrc && s->pending > beg)
            strm->adler = crc32(strm->adler, s->pending_buf + beg,
                                s->pending - beg);

        if (val == 0) {
            s->gzindex = 0;
            s->status = COMMENT_STATE;
        }
    }
    else
        s->status = COMMENT_STATE;
}
if (s->status == COMMENT_STATE) {
    if (s->gzhead->comment != NULL) {
        uInt beg = s->pending; /* start of bytes to update crc */
        int val;

        do {
            if (s->pending == s->pending_buf_size) {
                if (s->gzhead->hcrc && s->pending > beg)
                    strm->adler = crc32(strm->adler, s->pending_buf + beg,
                                           s->pending - beg);

                flush_pending(strm);
                beg = s->pending;
                if (s->pending == s->pending_buf_size) {
                    val = 1;
                    break;
                }
            }
            val = s->gzhead->comment[s->gzindex++];
            put_byte(s, val);
        } while (val != 0);
        if (s->gzhead->hcrc && s->pending > beg)
            strm->adler = crc32(strm->adler, s->pending_buf + beg,
                                s->pending - beg);

        if (val == 0)
            s->status = HCRC_STATE;
    }
    else
        s->status = HCRC_STATE;
}
if (s->status == HCRC_STATE) {

```

```

    if (s->gzhead->hcrc) {
        if (s->pending + 2 > s->pending_buf_size)
            flush_pending(strm);
        if (s->pending + 2 <= s->pending_buf_size) {
            put_byte(s, (Byte)(strm->adler & 0xff));
            put_byte(s, (Byte)((strm->adler >> 8) & 0xff));
            strm->adler = crc32(0L, Z_NULL, 0);
            s->status = BUSY_STATE;
        }
    }
    else
        s->status = BUSY_STATE;
}
#endif

/* Flush as much pending output as possible */
if (s->pending != 0) {
    flush_pending(strm);
    if (strm->avail_out == 0) {
        /* Since avail_out is 0, deflate will be called again with
         * more output space, but possibly with both pending and
         * avail_in equal to zero. There won't be anything to do,
         * but this is not an error situation so make sure we
         * return OK instead of BUF_ERROR at next call of deflate:
         */
        s->last_flush = -1;
        return Z_OK;
    }
}

/* Make sure there is something to do and avoid duplicate consecutive
 * flushes. For repeated and useless calls with Z_FINISH, we keep
 * returning Z_STREAM_END instead of Z_BUF_ERROR.
 */
} else if (strm->avail_in == 0 && flush <= old_flush &&
            flush != Z_FINISH) {
    ERR_RETURN(strm, Z_BUF_ERROR);
}

/* User must not provide more input after the first FINISH: */
if (s->status == FINISH_STATE && strm->avail_in != 0) {
    ERR_RETURN(strm, Z_BUF_ERROR);
}

/* Start a new block or continue the current one.
 */
if (strm->avail_in != 0 || s->lookahead != 0 ||
    (flush != Z_NO_FLUSH && s->status != FINISH_STATE)) {
    block_state bstate;

    bstate = (*(configuration_table[s->level].func))(s, flush);

    if (bstate == finish_started || bstate == finish_done) {
        s->status = FINISH_STATE;
    }
    if (bstate == need_more || bstate == finish_started) {
        if (strm->avail_out == 0) {
            s->last_flush = -1; /* avoid BUF_ERROR next call, see above */
        }
        return Z_OK;
        /* If flush != Z_NO_FLUSH && avail_out == 0, the next call
         * of deflate should use the same flush parameter to make sure
         * that the flush is complete. So we don't have to output an
         * empty block here, this will be done at next call. This also
         * ensures that for a very small output buffer, we emit at most
         * one empty block.
         */
    }
}
if (bstate == block_done) {
    if (flush == Z_PARTIAL_FLUSH) {
        _tr_align(s);
    } else { /* FULL_FLUSH or SYNC_FLUSH */
        _tr_stored_block(s, (char*)0, 0L, 0);
        /* For a full flush, this empty block will be recognized
         * as a special marker by inflate_sync().
         */
    }
}

```

```

        */
        if (flush == Z_FULL_FLUSH) {
            CLEAR_HASH(s); /* forget history */
        }
    }
    flush_pending(strm);
    if (strm->avail_out == 0) {
        s->last_flush = -1; /* avoid BUF_ERROR at next call, see above */
        return Z_OK;
    }
}
Assert(strm->avail_out > 0, "bug2");

if (flush != Z_FINISH) return Z_OK;
if (s->wrap <= 0) return Z_STREAM_END;

/* Write the trailer */
#ifdef GZIP
if (s->wrap == 2) {
    put_byte(s, (Byte)(strm->adler & 0xff));
    put_byte(s, (Byte)((strm->adler >> 8) & 0xff));
    put_byte(s, (Byte)((strm->adler >> 16) & 0xff));
    put_byte(s, (Byte)((strm->adler >> 24) & 0xff));
    put_byte(s, (Byte)(strm->total_in & 0xff));
    put_byte(s, (Byte)((strm->total_in >> 8) & 0xff));
    put_byte(s, (Byte)((strm->total_in >> 16) & 0xff));
    put_byte(s, (Byte)((strm->total_in >> 24) & 0xff));
}
else
#endif
{
    putShortMSB(s, (uInt)(strm->adler >> 16));
    putShortMSB(s, (uInt)(strm->adler & 0xffff));
}
flush_pending(strm);
/* If avail_out is zero, the application will call deflate again
 * to flush the rest.
 */
if (s->wrap > 0) s->wrap = -s->wrap; /* write the trailer only once! */
return s->pending != 0 ? Z_OK : Z_STREAM_END;
}

/* ===== */
int ZEXPORT deflateEnd (strm)
    z_streamp strm;
{
    int status;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;

    status = strm->state->status;
    if (status != INIT_STATE &&
        status != EXTRA_STATE &&
        status != NAME_STATE &&
        status != COMMENT_STATE &&
        status != HCRC_STATE &&
        status != BUSY_STATE &&
        status != FINISH_STATE) {
        return Z_STREAM_ERROR;
    }

    /* Deallocate in reverse order of allocations: */
    TRY_FREE(strm, strm->state->pending_buf);
    TRY_FREE(strm, strm->state->head);
    TRY_FREE(strm, strm->state->prev);
    TRY_FREE(strm, strm->state->window);

    ZFREE(strm, strm->state);
    strm->state = Z_NULL;

    return status == BUSY_STATE ? Z_DATA_ERROR : Z_OK;
}

```

```

/* =====
 * Copy the source state to the destination state.
 * To simplify the source, this is not supported for 16-bit MSDOS (which
 * doesn't have enough memory anyway to duplicate compression states).
 */
int ZEXPORT deflateCopy (dest, source)
    z_stream dest;
    z_stream source;
{
#ifdef MAXSEG_64K
    return Z_STREAM_ERROR;
#else
    deflate_state *ds;
    deflate_state *ss;
    ushf *overlay;

    if (source == Z_NULL || dest == Z_NULL || source->state == Z_NULL) {
        return Z_STREAM_ERROR;
    }

    ss = source->state;

    zmemcpy(dest, source, sizeof(z_stream));

    ds = (deflate_state *) ZALLOC(dest, 1, sizeof(deflate_state));
    if (ds == Z_NULL) return Z_MEM_ERROR;
    dest->state = (struct internal_state FAR *) ds;
    zmemcpy(ds, ss, sizeof(deflate_state));
    ds->strm = dest;

    ds->window = (Bytef *) ZALLOC(dest, ds->w_size, 2*sizeof(Byte));
    ds->prev = (Posf *) ZALLOC(dest, ds->w_size, sizeof(Pos));
    ds->head = (Posf *) ZALLOC(dest, ds->hash_size, sizeof(Pos));
    overlay = (ushf *) ZALLOC(dest, ds->lit_bufsize, sizeof(ush)+2);
    ds->pending_buf = (uchf *) overlay;

    if (ds->window == Z_NULL || ds->prev == Z_NULL || ds->head == Z_NULL ||
        ds->pending_buf == Z_NULL) {
        deflateEnd (dest);
        return Z_MEM_ERROR;
    }
    /* following zmemcpy do not work for 16-bit MSDOS */
    zmemcpy(ds->window, ss->window, ds->w_size * 2 * sizeof(Byte));
    zmemcpy(ds->prev, ss->prev, ds->w_size * sizeof(Pos));
    zmemcpy(ds->head, ss->head, ds->hash_size * sizeof(Pos));
    zmemcpy(ds->pending_buf, ss->pending_buf, (uInt)ds->pending_buf_size);

    ds->pending_out = ds->pending_buf + (ss->pending_out - ss->pending_buf);
    ds->d_buf = overlay + ds->lit_bufsize/sizeof(ush);
    ds->l_buf = ds->pending_buf + (1+sizeof(ush))*ds->lit_bufsize;

    ds->l_desc.dyn_tree = ds->dyn_ltree;
    ds->d_desc.dyn_tree = ds->dyn_dtree;
    ds->bl_desc.dyn_tree = ds->bl_tree;

    return Z_OK;
#endif /* MAXSEG_64K */
}

/* =====
 * Read a new buffer from the current input stream, update the Adler32
 * and total number of bytes read. All deflate() input goes through
 * this function so some applications may wish to modify it to avoid
 * allocating a large strm->next_in buffer and copying from it.
 * (See also flush_pending()).
 */
local int read_buf(strm, buf, size)
    z_stream strm;
    Bytef *buf;
    unsigned size;
{
    unsigned len = strm->avail_in;

```



```

    if (len > size) len = size;
    if (len == 0) return 0;

    strm->avail_in -= len;

    if (strm->state->wrap == 1) {
        strm->adler = Adler32(strm->adler, strm->next_in, len);
    }
#ifdef GZIP
    else if (strm->state->wrap == 2) {
        strm->adler = crc32(strm->adler, strm->next_in, len);
    }
#endif
    zmemcpy(buf, strm->next_in, len);
    strm->next_in += len;
    strm->total_in += len;

    return (int)len;
}

/* =====
 * Initialize the "longest match" routines for a new zlib stream
 */
local void lm_init (s)
    deflate_state *s;
{
    s->window_size = (ulg)2L*s->w_size;

    CLEAR_HASH(s);

    /* Set the default configuration parameters:
     */
    s->max_lazy_match = configuration_table[s->level].max_lazy;
    s->good_match = configuration_table[s->level].good_length;
    s->nice_match = configuration_table[s->level].nice_length;
    s->max_chain_length = configuration_table[s->level].max_chain;

    s->strstart = 0;
    s->block_start = 0L;
    s->lookahead = 0;
    s->match_length = s->prev_length = MIN_MATCH-1;
    s->match_available = 0;
    s->ins_h = 0;
#ifdef FASTEST
#endif
#ifdef ASMV
    match_init(); /* initialize the asm code */
#endif
}

#ifdef FASTEST
/* =====
 * Set match_start to the longest match starting at the given string and
 * return its length. Matches shorter or equal to prev_length are discarded,
 * in which case the result is equal to prev_length and match_start is
 * garbage.
 * IN assertions: cur_match is the head of the hash chain for the current
 * string (strstart) and its distance is <= MAX_DIST, and prev_length >= 1
 * OUT assertion: the match length is not greater than s->lookahead.
 */
#ifdef ASMV
/* For 80x86 and 680x0, an optimized version will be provided in match.asm or
 * match.S. The code will be functionally equivalent.
 */
local uInt longest_match(s, cur_match)
    deflate_state *s;
    IPos cur_match;
    /* current match */
{
    unsigned chain_length = s->max_chain_length; /* max hash chain length */
    register Bytef *scan = s->window + s->strstart; /* current string */
    register Bytef *match; /* matched string */
    register int len; /* length of current match */
    int best_len = s->prev_length; /* best match length so far */
    int nice_match = s->nice_match; /* stop if match long enough */

```

```

IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
    s->strstart - (IPos)MAX_DIST(s) : NIL;
/* Stop when cur_match becomes <= limit. To simplify the code,
 * we prevent matches with the string of window index 0.
 */
Posf *prev = s->prev;
uInt wmask = s->w_mask;

#ifdef UNALIGNED_OK
/* Compare two bytes at a time. Note: this is not always beneficial.
 * Try with and without -DUNALIGNED_OK to check.
 */
register Bytef *strend = s->window + s->strstart + MAX_MATCH - 1;
register ush scan_start = *(ushf*)scan;
register ush scan_end   = *(ushf*)(scan+best_len-1);
#else
register Bytef *strend = s->window + s->strstart + MAX_MATCH;
register Byte scan_end1 = scan[best_len-1];
register Byte scan_end  = scan[best_len];
#endif

/* The code is optimized for HASH_BITS >= 8 and MAX_MATCH-2 multiple of 16.
 * It is easy to get rid of this optimization if necessary.
 */
Assert(s->hash_bits >= 8 && MAX_MATCH == 258, "Code too clever");

/* Do not waste too much time if we already have a good match: */
if (s->prev_length >= s->good_match) {
    chain_length >>= 2;
}
/* Do not look for matches beyond the end of the input. This is necessary
 * to make deflate deterministic.
 */
if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;

Assert((ulg)s->strstart <= s->window_size-MIN_LOOKAHEAD, "need lookahead");

do {
    Assert(cur_match < s->strstart, "no future");
    match = s->window + cur_match;

    /* Skip to next match if the match length cannot increase
     * or if the match length is less than 2. Note that the checks below
     * for insufficient lookahead only occur occasionally for performance
     * reasons. Therefore uninitialized memory will be accessed, and
     * conditional jumps will be made that depend on those values.
     * However the length of the match is limited to the lookahead, so
     * the output of deflate is not affected by the uninitialized values.
     */
#ifdef (defined(UNALIGNED_OK) && MAX_MATCH == 258)
    /* This code assumes sizeof(unsigned short) == 2. Do not use
     * UNALIGNED_OK if your compiler uses a different size.
     */
    if (*(ushf*)(match+best_len-1) != scan_end ||
        *(ushf*)match != scan_start) continue;

    /* It is not necessary to compare scan[2] and match[2] since they are
     * always equal when the other bytes match, given that the hash keys
     * are equal and that HASH_BITS >= 8. Compare 2 bytes at a time at
     * strstart+3, +5, ... up to strstart+257. We check for insufficient
     * lookahead only every 4th comparison; the 128th check will be made
     * at strstart+257. If MAX_MATCH-2 is not a multiple of 8, it is
     * necessary to put more guard bytes at the end of the window, or
     * to check more often for insufficient lookahead.
     */
    Assert(scan[2] == match[2], "scan[2]?");
    scan++, match++;
    do {
        while (*(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
            *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
            *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
            *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
            scan < strend);
        /* The funny "do {}" generates better code on most compilers */

```

```

/* Here, scan <= window+strstart+257 */
Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");
if (*scan == *match) scan++;

len = (MAX_MATCH - 1) - (int)(strend-scan);
scan = strend - (MAX_MATCH-1);

#else /* UNALIGNED_OK */

    if (match[best_len] != scan_end ||
        match[best_len-1] != scan_end1 ||
        *match != *scan ||
        *++match != scan[1])        continue;

/* The check at best_len-1 can be removed because it will be made
 * again later. (This heuristic is not always a win.)
 * It is not necessary to compare scan[2] and match[2] since they
 * are always equal when the other bytes match, given that
 * the hash keys are equal and that HASH_BITS >= 8.
 */
scan += 2, match++;
Assert(*scan == *match, "match[2]?");

/* We check for insufficient lookahead only every 8th comparison;
 * the 256th check will be made at strstart+258.
 */
do {
} while ((*++scan == *++match && *++scan == *++match &&
        *++scan == *++match && *++scan == *++match &&
        *++scan == *++match && *++scan == *++match &&
        *++scan == *++match && *++scan == *++match &&
        scan < strend);

Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");

len = MAX_MATCH - (int)(strend - scan);
scan = strend - MAX_MATCH;

#endif /* UNALIGNED_OK */

    if (len > best_len) {
        s->match_start = cur_match;
        best_len = len;
        if (len >= nice_match) break;
#ifdef UNALIGNED_OK
        scan_end = *(ushf*)(scan+best_len-1);
#else
        scan_end1 = scan[best_len-1];
        scan_end = scan[best_len];
#endif
    }
    while ((cur_match = prev[cur_match & wmask]) > limit
        && --chain_length != 0);

    if ((uInt)best_len <= s->lookahead) return (uInt)best_len;
    return s->lookahead;
}
#endif /* ASMV */
#endif /* FASTEST */

/* -----
 * Optimized version for level == 1 or strategy == Z_RLE only
 */
local uInt longest_match_fast(s, cur_match)
    deflate_state *s;
    IPos cur_match;
    /* current match */
{
    register Bytef *scan = s->window + s->strstart; /* current string */
    register Bytef *match; /* matched string */
    register int len; /* length of current match */
    register Bytef *strend = s->window + s->strstart + MAX_MATCH;

    /* The code is optimized for HASH_BITS >= 8 and MAX_MATCH-2 multiple of 16.

```

```

    * It is easy to get rid of this optimization if necessary.
    */
    Assert(s->hash_bits >= 8 && MAX_MATCH == 258, "Code too clever");

    Assert((ulg)s->strstart <= s->window_size-MIN_LOOKAHEAD, "need lookahead");

    Assert(cur_match < s->strstart, "no future");

    match = s->window + cur_match;

    /* Return failure if the match length is less than 2:
    */
    if (match[0] != scan[0] || match[1] != scan[1]) return MIN_MATCH-1;

    /* The check at best_len-1 can be removed because it will be made
    * again later. (This heuristic is not always a win.)
    * It is not necessary to compare scan[2] and match[2] since they
    * are always equal when the other bytes match, given that
    * the hash keys are equal and that HASH_BITS >= 8.
    */
    scan += 2, match += 2;
    Assert(*scan == *match, "match[2]?");

    /* We check for insufficient lookahead only every 8th comparison;
    * the 256th check will be made at strstart+258.
    */
    do {
    } while (++++scan == ++match && ++scan == ++match &&
        ++scan == ++match && ++scan == ++match &&
        ++scan == ++match && ++scan == ++match &&
        scan < strend);

    Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");

    len = MAX_MATCH - (int)(strend - scan);

    if (len < MIN_MATCH) return MIN_MATCH - 1;

    s->match_start = cur_match;
    return (uInt)len <= s->lookahead ? (uInt)len : s->lookahead;
}

#ifdef DEBUG
/* =====
 * Check that the match at match_start is indeed a match.
 */
local void check_match(s, start, match, length)
    deflate_state *s;
    IPos start, match;
    int length;
{
    /* check that the match is indeed a match */
    if (zmemcmp(s->window + match,
        s->window + start, length) != EQUAL) {
        fprintf(stderr, " start %u, match %u, length %d\n",
            start, match, length);
        do {
            fprintf(stderr, "%c%c", s->window[match++], s->window[start++]);
        } while (--length != 0);
        z_error("invalid match");
    }
    if (z_verbose > 1) {
        fprintf(stderr, "\\[%d,%d]", start-match, length);
        do { putc(s->window[start++], stderr); } while (--length != 0);
    }
}
#else
#define check_match(s, start, match, length)
#endif

/* =====
 * Fill the window when the lookahead becomes insufficient.
 * Updates strstart and lookahead.
 */

```

```

*
* IN assertion: lookahead < MIN_LOOKAHEAD
* OUT assertions: strstart <= window_size-MIN_LOOKAHEAD
*   At least one byte has been read, or avail_in == 0; reads are
*   performed for at least two bytes (required for the zip translate_eol
*   option -- not supported here).
*/
local void fill_window(s)
    deflate_state *s;
{
    register unsigned n, m;
    register Posf *p;
    unsigned more; /* Amount of free space at the end of the window. */
    uInt wsize = s->w_size;

    do {
        more = (unsigned)(s->window_size - (ulg)s->lookahead - (ulg)s->strstart);

        /* Deal with !@#$$ 64K limit: */
        if (sizeof(int) <= 2) {
            if (more == 0 && s->strstart == 0 && s->lookahead == 0) {
                more = wsize;
            }
            else if (more == (unsigned)(-1)) {
                /* Very unlikely, but possible on 16 bit machine if
                 * strstart == 0 && lookahead == 1 (input done a byte at time)
                 */
                more--;
            }
        }

        /* If the window is almost full and there is insufficient lookahead,
         * move the upper half to the lower one to make room in the upper half.
         */
        if (s->strstart >= wsize+MAX_DIST(s)) {

            zmemcpy(s->window, s->window+wsize, (unsigned)wsize);
            s->match_start -= wsize;
            s->strstart -= wsize; /* we now have strstart >= MAX_DIST */
            s->block_start -= (long) wsize;

            /* Slide the hash table (could be avoided with 32 bit values
             * at the expense of memory usage). We slide even when level == 0
             * to keep the hash table consistent if we switch back to level > 0
             * later. (Using level 0 permanently is not an optimal usage of
             * zlib, so we don't care about this pathological case.)
             */
            /* %% avoid this when Z_RLE */
            n = s->hash_size;
            p = &s->head[n];
            do {
                m = *--p;
                *p = (Pos)(m >= wsize ? m-wsize : NIL);
            } while (--n);

            n = wsize;
            #ifndef FASTEST
            p = &s->prev[n];
            do {
                m = *--p;
                *p = (Pos)(m >= wsize ? m-wsize : NIL);
                /* If n is not on any hash chain, prev[n] is garbage but
                 * its value will never be used.
                 */
            } while (--n);
            #endif
            more += wsize;
        }
        if (s->strm->avail_in == 0) return;

        /* If there was no sliding:
         *   strstart <= WSIZE+MAX_DIST-1 && lookahead <= MIN_LOOKAHEAD - 1 &&
         *   more == window_size - lookahead - strstart
         * => more >= window_size - (MIN_LOOKAHEAD-1 + WSIZE + MAX_DIST-1)

```

```

    * => more >= window_size - 2*WSIZE + 2
    * In the BIG_MEM or MMAP case (not yet supported),
    *   window_size == input_size + MIN_LOOKAHEAD  &&
    *   strstart + s->lookahead <= input_size => more >= MIN_LOOKAHEAD.
    * Otherwise, window_size == 2*WSIZE so more >= 2.
    * If there was sliding, more >= WSIZE. So in all cases, more >= 2.
    */
    Assert(more >= 2, "more<2");

    n = read_buf(s->strm, s->window + s->strstart + s->lookahead, more);
    s->lookahead += n;

    /* Initialize the hash value now that we have some input: */
    if (s->lookahead >= MIN_MATCH) {
        s->ins_h = s->window[s->strstart];
        UPDATE_HASH(s, s->ins_h, s->window[s->strstart+1]);
#ifdef MIN_MATCH != 3
        Call UPDATE_HASH() MIN_MATCH-3 more times
#endif
    }
    /* If the whole input has less than MIN_MATCH bytes, ins_h is garbage,
     * but this is not important since only literal bytes will be emitted.
     */

    } while (s->lookahead < MIN_LOOKAHEAD && s->strm->avail_in != 0);
}

/* =====
 * Flush the current block, with given end-of-file flag.
 * IN assertion: strstart is set to the end of the current match.
 */
#define FLUSH_BLOCK_ONLY(s, eof) { \
    _tr_flush_block(s, (s->block_start >= 0L ? \
        (charf *)&s->window[(unsigned)s->block_start] : \
        (charf *)Z_NULL), \
        (ulg)((long)s->strstart - s->block_start), \
        (eof)); \
    s->block_start = s->strstart; \
    flush_pending(s->strm); \
    Tracev((stderr, "[FLUSH]")); \
}

/* Same but force premature exit if necessary. */
#define FLUSH_BLOCK(s, eof) { \
    FLUSH_BLOCK_ONLY(s, eof); \
    if (s->strm->avail_out == 0) return (eof) ? finish_started : need_more; \
}

/* =====
 * Copy without compression as much as possible from the input stream, return
 * the current block state.
 * This function does not insert new strings in the dictionary since
 * uncompressible data is probably not useful. This function is used
 * only for the level=0 compression option.
 * NOTE: this function should be optimized to avoid extra copying from
 * window to pending_buf.
 */
local block_state deflate_stored(s, flush)
    deflate_state *s;
    int flush;
{
    /* Stored blocks are limited to 0xffff bytes, pending_buf is limited
     * to pending_buf_size, and each stored block has a 5 byte header:
     */
    ulg max_block_size = 0xffff;
    ulg max_start;

    if (max_block_size > s->pending_buf_size - 5) {
        max_block_size = s->pending_buf_size - 5;
    }

    /* Copy as much as possible from input to output: */
    for (;;) {
        /* Fill the window as much as possible: */

```

```

    if (s->lookahead <= 1) {
        Assert(s->strstart < s->w_size+MAX_DIST(s) ||
            s->block_start >= (long)s->w_size, "slide too late");

        fill_window(s);
        if (s->lookahead == 0 && flush == Z_NO_FLUSH) return need_more;

        if (s->lookahead == 0) break; /* flush the current block */
    }
    Assert(s->block_start >= 0L, "block gone");

    s->strstart += s->lookahead;
    s->lookahead = 0;

    /* Emit a stored block if pending_buf will be full: */
    max_start = s->block_start + max_block_size;
    if (s->strstart == 0 || (ulg)s->strstart >= max_start) {
        /* strstart == 0 is possible when wraparound on 16-bit machine */
        s->lookahead = (uInt)(s->strstart - max_start);
        s->strstart = (uInt)max_start;
        FLUSH_BLOCK(s, 0);
    }
    /* Flush if we may have to slide, otherwise block_start may become
     * negative and the data will be gone:
     */
    if (s->strstart - (uInt)s->block_start >= MAX_DIST(s)) {
        FLUSH_BLOCK(s, 0);
    }
    FLUSH_BLOCK(s, flush == Z_FINISH);
    return flush == Z_FINISH ? finish_done : block_done;
}

/* =====
 * Compress as much as possible from the input stream, return the current
 * block state.
 * This function does not perform lazy evaluation of matches and inserts
 * new strings in the dictionary only for unmatched strings or for short
 * matches. It is used only for the fast compression options.
 */
local block_state deflate_fast(s, flush)
    deflate_state *s;
    int flush;
{
    IPos hash_head = NIL; /* head of the hash chain */
    int bflush;           /* set if current block must be flushed */

    for (;;) {
        /* Make sure that we always have enough lookahead, except
         * at the end of the input file. We need MAX_MATCH bytes
         * for the next match, plus MIN_MATCH bytes to insert the
         * string following the next match.
         */
        if (s->lookahead < MIN_LOOKAHEAD) {
            fill_window(s);
            if (s->lookahead < MIN_LOOKAHEAD && flush == Z_NO_FLUSH) {
                return need_more;
            }
            if (s->lookahead == 0) break; /* flush the current block */
        }

        /* Insert the string window[strstart .. strstart+2] in the
         * dictionary, and set hash_head to the head of the hash chain:
         */
        if (s->lookahead >= MIN_MATCH) {
            INSERT_STRING(s, s->strstart, hash_head);
        }

        /* Find the longest match, discarding those <= prev_length.
         * At this point we have always match_length < MIN_MATCH
         */
        if (hash_head != NIL && s->strstart - hash_head <= MAX_DIST(s)) {
            /* To simplify the code, we prevent matches with the string

```

```

        * of window index 0 (in particular we have to avoid a match
        * of the string with itself at the start of the input file).
        */
#ifdef FASTEST
    if ((s->strategy != Z_HUFFMAN_ONLY && s->strategy != Z_RLE) ||
        (s->strategy == Z_RLE && s->strstart - hash_head == 1)) {
        s->match_length = longest_match_fast (s, hash_head);
    }
#else
    if (s->strategy != Z_HUFFMAN_ONLY && s->strategy != Z_RLE) {
        s->match_length = longest_match (s, hash_head);
    } else if (s->strategy == Z_RLE && s->strstart - hash_head == 1) {
        s->match_length = longest_match_fast (s, hash_head);
    }
#endif

    /* longest_match() or longest_match_fast() sets match_start */
    if (s->match_length >= MIN_MATCH) {
        check_match(s, s->strstart, s->match_start, s->match_length);

        _tr_tally_dist(s, s->strstart - s->match_start,
                      s->match_length - MIN_MATCH, bflush);

        s->lookahead -= s->match_length;

        /* Insert new strings in the hash table only if the match length
        * is not too large. This saves time but degrades compression.
        */
#ifdef FASTEST
        if (s->match_length <= s->max_insert_length &&
            s->lookahead >= MIN_MATCH) {
            s->match_length--; /* string at strstart already in table */
            do {
                s->strstart++;
                INSERT_STRING(s, s->strstart, hash_head);
                /* strstart never exceeds WSIZE-MAX_MATCH, so there are
                * always MIN_MATCH bytes ahead.
                */
            } while (--s->match_length != 0);
            s->strstart++;
        } else
#endif
        {
            s->strstart += s->match_length;
            s->match_length = 0;
            s->ins_h = s->window[s->strstart];
            UPDATE_HASH(s, s->ins_h, s->window[s->strstart+1]);
#ifdef MIN_MATCH != 3
            Call UPDATE_HASH() MIN_MATCH-3 more times
#endif
#endif

            /* If lookahead < MIN_MATCH, ins_h is garbage, but it does not
            * matter since it will be recomputed at next deflate call.
            */
        }
    } else {
        /* No match, output a literal byte */
        Tracevv((stderr, "%c", s->window[s->strstart]));
        _tr_tally_lit (s, s->window[s->strstart], bflush);
        s->lookahead--;
        s->strstart++;
    }
    if (bflush) FLUSH_BLOCK(s, 0);
}
FLUSH_BLOCK(s, flush == Z_FINISH);
return flush == Z_FINISH ? finish_done : block_done;
}

#ifdef FASTEST
/* =====
* Same as above, but achieves better compression. We use a lazy
* evaluation for matches: a match is finally adopted only if there is
* no better match at the next window position.
*/
local block_state deflate_slow(s, flush)

```



```

deflate_state *s;
int flush;

{
    IPos hash_head = NIL;    /* head of hash chain */
    int bflush;              /* set if current block must be flushed */

    /* Process the input block. */
    for (;;) {
        /* Make sure that we always have enough lookahead, except
         * at the end of the input file. We need MAX_MATCH bytes
         * for the next match, plus MIN_MATCH bytes to insert the
         * string following the next match.
         */
        if (s->lookahead < MIN_LOOKAHEAD) {
            fill_window(s);
            if (s->lookahead < MIN_LOOKAHEAD && flush == Z_NO_FLUSH) {
                return need_more;
            }
            if (s->lookahead == 0) break; /* flush the current block */
        }

        /* Insert the string window[strstart .. strstart+2] in the
         * dictionary, and set hash_head to the head of the hash chain:
         */
        if (s->lookahead >= MIN_MATCH) {
            INSERT_STRING(s, s->strstart, hash_head);
        }

        /* Find the longest match, discarding those <= prev_length.
         */
        s->prev_length = s->match_length, s->prev_match = s->match_start;
        s->match_length = MIN_MATCH-1;

        if (hash_head != NIL && s->prev_length < s->max_lazy_match &&
            s->strstart - hash_head <= MAX_DIST(s)) {
            /* To simplify the code, we prevent matches with the string
             * of window index 0 (in particular we have to avoid a match
             * of the string with itself at the start of the input file).
             */
            if (s->strategy != Z_HUFFMAN_ONLY && s->strategy != Z_RLE) {
                s->match_length = longest_match(s, hash_head);
            } else if (s->strategy == Z_RLE && s->strstart - hash_head == 1) {
                s->match_length = longest_match_fast(s, hash_head);
            }
            /* longest_match() or longest_match_fast() sets match_start */
        }
        if (s->match_length <= 5 && (s->strategy == Z_FILTERED
#ifdef TOO_FAR
            || (s->match_length == MIN_MATCH &&
                s->strstart - s->match_start > TOO_FAR)
#endif
        )) {
            /* If prev_match is also MIN_MATCH, match_start is garbage
             * but we will ignore the current match anyway.
             */
            s->match_length = MIN_MATCH-1;
        }
    }
    /* If there was a match at the previous step and the current
     * match is not better, output the previous match:
     */
    if (s->prev_length >= MIN_MATCH && s->match_length <= s->prev_length) {
        uInt max_insert = s->strstart + s->lookahead - MIN_MATCH;
        /* Do not insert strings in hash table beyond this. */

        check_match(s, s->strstart-1, s->prev_match, s->prev_length);

        _tr_tally_dist(s, s->strstart-1 - s->prev_match,
            s->prev_length - MIN_MATCH, bflush);

        /* Insert in hash table all strings up to the end of the match.
         * strstart-1 and strstart are already inserted. If there is not
         * enough lookahead, the last two strings are not inserted in

```

```

        * the hash table.
        */
        s->lookahead -= s->prev_length-1;
        s->prev_length -= 2;
        do {
            if (++s->strstart <= max_insert) {
                INSERT_STRING(s, s->strstart, hash_head);
            }
        } while (--s->prev_length != 0);
        s->match_available = 0;
        s->match_length = MIN_MATCH-1;
        s->strstart++;

        if (bflush) FLUSH_BLOCK(s, 0);

    } else if (s->match_available) {
        /* If there was no match at the previous position, output a
         * single literal. If there was a match but the current match
         * is longer, truncate the previous match to a single literal.
         */
        Tracevv((stderr, "%c", s->window[s->strstart-1]));
        _tr_tally_lit(s, s->window[s->strstart-1], bflush);
        if (bflush) {
            FLUSH_BLOCK_ONLY(s, 0);
        }
        s->strstart++;
        s->lookahead--;
        if (s->strm->avail_out == 0) return need_more;
    } else {
        /* There is no previous match to compare with, wait for
         * the next step to decide.
         */
        s->match_available = 1;
        s->strstart++;
        s->lookahead--;
    }
}
Assert (flush != Z_NO_FLUSH, "no flush?");
if (s->match_available) {
    Tracevv((stderr, "%c", s->window[s->strstart-1]));
    _tr_tally_lit(s, s->window[s->strstart-1], bflush);
    s->match_available = 0;
}
FLUSH_BLOCK(s, flush == Z_FINISH);
return flush == Z_FINISH ? finish_done : block_done;
}
#endif /* FASTEST */

#if 0
/* =====
 * For Z_RLE, simply look for runs of bytes, generate matches only of distance
 * one. Do not maintain a hash table. (It will be regenerated if this run of
 * deflate switches away from Z_RLE.)
 */
local block_state deflate_rle(s, flush)
    deflate_state *s;
    int flush;
{
    int bflush;           /* set if current block must be flushed */
    uInt run;             /* length of run */
    uInt max;             /* maximum length of run */
    uInt prev;            /* byte at distance one to match */
    Bytef *scan;          /* scan for end of run */

    for (;;) {
        /* Make sure that we always have enough lookahead, except
         * at the end of the input file. We need MAX_MATCH bytes
         * for the longest encodable run.
         */
        if (s->lookahead < MAX_MATCH) {
            fill_window(s);
            if (s->lookahead < MAX_MATCH && flush == Z_NO_FLUSH) {
                return need_more;
            }
        }

```

```
        if (s->lookahead == 0) break; /* flush the current block */
    }

    /* See how many times the previous byte repeats */
    run = 0;
    if (s->strstart > 0) { /* if there is a previous byte, that is */
        max = s->lookahead < MAX_MATCH ? s->lookahead : MAX_MATCH;
        scan = s->window + s->strstart - 1;
        prev = *scan++;
        do {
            if (*scan++ != prev)
                break;
        } while (++run < max);
    }

    /* Emit match if have run of MIN_MATCH or longer, else emit literal */
    if (run >= MIN_MATCH) {
        check_match(s, s->strstart, s->strstart - 1, run);
        _tr_tally_dist(s, 1, run - MIN_MATCH, bflush);
        s->lookahead -= run;
        s->strstart += run;
    } else {
        /* No match, output a literal byte */
        Tracevv((stderr, "%c", s->window[s->strstart]));
        _tr_tally_lit(s, s->window[s->strstart], bflush);
        s->lookahead--;
        s->strstart++;
    }
    if (bflush) FLUSH_BLOCK(s, 0);
}
FLUSH_BLOCK(s, flush == Z_FINISH);
return flush == Z_FINISH ? finish_done : block_done;
}
#endif
```

```
/* deflate.h -- internal compression state
 * Copyright (C) 1995-2004 Jean-loup Gailly
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
 * part of the implementation of the compression library and is
 * subject to change. Applications should only use zlib.h.
 */

/* @(#) $Id: deflate.h,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#ifndef DEFLATE_H
#define DEFLATE_H

#include "zutil.h"

/* define NO_GZIP when compiling if you want to disable gzip header and
 * trailer creation by deflate(). NO_GZIP would be used to avoid linking in
 * the crc code when it is not needed. For shared libraries, gzip encoding
 * should be left enabled. */
#ifndef NO_GZIP
# define GZIP
#endif

/* =====
 * Internal compression state.
 */

#define LENGTH_CODES 29
/* number of length codes, not counting the special END_BLOCK code */

#define LITERALS 256
/* number of literal bytes 0..255 */

#define L_CODES (LITERALS+1+LENGTH_CODES)
/* number of Literal or Length codes, including the END_BLOCK code */

#define D_CODES 30
/* number of distance codes */

#define BL_CODES 19
/* number of codes used to transfer the bit lengths */

#define HEAP_SIZE (2*L_CODES+1)
/* maximum heap size */

#define MAX_BITS 15
/* All codes must not exceed MAX_BITS bits */

#define INIT_STATE 42
#define EXTRA_STATE 69
#define NAME_STATE 73
#define COMMENT_STATE 91
#define HCRC_STATE 103
#define BUSY_STATE 113
#define FINISH_STATE 666
/* Stream status */

/* Data structure describing a single value and its code string. */
typedef struct ct_data_s {
    union {
        ush freq; /* frequency count */
        ush code; /* bit string */
    } fc;
    union {
        ush dad; /* father node in Huffman tree */
        ush len; /* length of bit string */
    } dl;
} FAR ct_data;

#define Freq fc.freq
#define Code fc.code
```

```

#define Dad    dl.dad
#define Len    dl.len

typedef struct static_tree_desc_s  static_tree_desc;

typedef struct tree_desc_s {
    ct_data *dyn_tree;           /* the dynamic tree */
    int      max_code;           /* largest code with non zero frequency */
    static_tree_desc *stat_desc; /* the corresponding static tree */
} FAR tree_desc;

typedef ush Pos;
typedef Pos FAR Posf;
typedef unsigned IPos;

/* A Pos is an index in the character window. We use short instead of int to
 * save space in the various tables. IPos is used only for parameter passing.
 */

typedef struct internal_state {
    z_streamp strm;           /* pointer back to this zlib stream */
    int status;               /* as the name implies */
    Bytef *pending_buf;       /* output still pending */
    ulg pending_buf_size;     /* size of pending_buf */
    Bytef *pending_out;       /* next pending byte to output to the stream */
    uInt pending;              /* nb of bytes in the pending buffer */
    int wrap;                  /* bit 0 true for zlib, bit 1 true for gzip */
    gz_headerp gzhead;        /* gzip header information to write */
    uInt gzindex;              /* where in extra, name, or comment */
    Byte method;               /* STORED (for zip only) or DEFLATED */
    int last_flush;            /* value of flush param for previous deflate call */

    /* used by deflate.c: */

    uInt w_size;               /* LZ77 window size (32K by default) */
    uInt w_bits;               /* log2(w_size)  (8..16) */
    uInt w_mask;               /* w_size - 1 */

    Bytef *window;
    /* Sliding window. Input bytes are read into the second half of the window,
     * and move to the first half later to keep a dictionary of at least wSize
     * bytes. With this organization, matches are limited to a distance of
     * wSize-MAX_MATCH bytes, but this ensures that IO is always
     * performed with a length multiple of the block size. Also, it limits
     * the window size to 64K, which is quite useful on MSDOS.
     * To do: use the user input buffer as sliding window.
     */

    ulg window_size;
    /* Actual size of window: 2*wSize, except when the user input buffer
     * is directly used as sliding window.
     */

    Posf *prev;
    /* Link to older string with same hash index. To limit the size of this
     * array to 64K, this link is maintained only for the last 32K strings.
     * An index in this array is thus a window index modulo 32K.
     */

    Posf *head; /* Heads of the hash chains or NIL. */

    uInt ins_h;          /* hash index of string to be inserted */
    uInt hash_size;       /* number of elements in hash table */
    uInt hash_bits;        /* log2(hash_size) */
    uInt hash_mask;        /* hash_size-1 */

    uInt hash_shift;
    /* Number of bits by which ins_h must be shifted at each input
     * step. It must be such that after MIN_MATCH steps, the oldest
     * byte no longer takes part in the hash key, that is:
     * hash_shift * MIN_MATCH >= hash_bits
     */

    long block_start;

```

```

/* Window position at the beginning of the current output block. Gets
 * negative when the window is moved backwards.
 */

uInt match_length;           /* length of best match */
IPos prev_match;            /* previous match */
int match_available;         /* set if previous match exists */
uInt strstart;              /* start of string to insert */
uInt match_start;           /* start of matching string */
uInt lookahead;             /* number of valid bytes ahead in window */

uInt prev_length;
/* Length of the best match at previous step. Matches not greater than this
 * are discarded. This is used in the lazy match evaluation.
 */

uInt max_chain_length;
/* To speed up deflation, hash chains are never searched beyond this
 * length. A higher limit improves compression ratio but degrades the
 * speed.
 */

uInt max_lazy_match;
/* Attempt to find a better match only when the current match is strictly
 * smaller than this value. This mechanism is used only for compression
 * levels >= 4.
 */
# define max_insert_length max_lazy_match
/* Insert new strings in the hash table only if the match length is not
 * greater than this length. This saves time but degrades compression.
 * max_insert_length is used only for compression levels <= 3.
 */

int level; /* compression level (1..9) */
int strategy; /* favor or force Huffman coding*/

uInt good_match;
/* Use a faster search when the previous match is longer than this */

int nice_match; /* Stop searching when current match exceeds this */

/* used by trees.c: */
/* Didn't use ct_data typedef below to suppress compiler warning */
struct ct_data_s dyn_ltree[HEAP_SIZE]; /* literal and length tree */
struct ct_data_s dyn_dtree[2*D_CODES+1]; /* distance tree */
struct ct_data_s bl_tree[2*BL_CODES+1]; /* Huffman tree for bit lengths */

struct tree_desc_s l_desc; /* desc. for literal tree */
struct tree_desc_s d_desc; /* desc. for distance tree */
struct tree_desc_s bl_desc; /* desc. for bit length tree */

ush bl_count[MAX_BITS+1];
/* number of codes at each bit length for an optimal tree */

int heap[2*L_CODES+1]; /* heap used to build the Huffman trees */
int heap_len; /* number of elements in the heap */
int heap_max; /* element of largest frequency */
/* The sons of heap[n] are heap[2*n] and heap[2*n+1]. heap[0] is not used.
 * The same heap array is used to build all trees.
 */

uch depth[2*L_CODES+1];
/* Depth of each subtree used as tie breaker for trees of equal frequency
 */

uchf *l_buf; /* buffer for literals or lengths */

uInt lit_bufsize;
/* Size of match buffer for literals/lengths. There are 4 reasons for
 * limiting lit_bufsize to 64K:
 * - frequencies can be kept in 16 bit counters
 * - if compression is not successful for the first block, all input
 * data is still in the window so we can still emit a stored block even
 * when input comes from standard input. (This can also be done for

```

```

*      all blocks if lit_bufsize is not greater than 32K.)
*      - if compression is not successful for a file smaller than 64K, we can
*      even emit a stored file instead of a stored block (saving 5 bytes).
*      This is applicable only for zip (not gzip or zlib).
*      - creating new Huffman trees less frequently may not provide fast
*      adaptation to changes in the input data statistics. (Take for
*      example a binary file with poorly compressible code followed by
*      a highly compressible string table.) Smaller buffer sizes give
*      fast adaptation but have of course the overhead of transmitting
*      trees more frequently.
*      - I can't count above 4
*/

uInt last_lit;      /* running index in l_buf */

ushf *d_buf;
/* Buffer for distances. To simplify the code, d_buf and l_buf have
 * the same number of elements. To use different lengths, an extra flag
 * array would be necessary.
 */

ulg opt_len;        /* bit length of current block with optimal trees */
ulg static_len;     /* bit length of current block with static trees */
uInt matches;       /* number of string matches in current block */
int last_eob_len;   /* bit length of EOB code for last block */

#ifdef DEBUG
    ulg compressed_len; /* total bit length of compressed file mod 2^32 */
    ulg bits_sent;      /* bit length of compressed data sent mod 2^32 */
#endif

ush bi_buf;
/* Output buffer. bits are inserted starting at the bottom (least
 * significant bits).
 */
int bi_valid;
/* Number of valid bits in bi_buf. All bits above the last valid bit
 * are always zero.
 */

} FAR deflate_state;

/* Output a byte on the stream.
 * IN assertion: there is enough room in pending_buf.
 */
#define put_byte(s, c) {s->pending_buf[s->pending++] = (c);}

#define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)
/* Minimum amount of lookahead, except at the end of the input file.
 * See deflate.c for comments about the MIN_MATCH+1.
 */

#define MAX_DIST(s) ((s)->w_size-MIN_LOOKAHEAD)
/* In order to simplify the code, particularly on 16 bit machines, match
 * distances are limited to MAX_DIST instead of WSIZE.
 */

/* in trees.c */
void _tr_init      OF((deflate_state *s));
int  _tr_tally     OF((deflate_state *s, unsigned dist, unsigned lc));
void _tr_flush_block OF((deflate_state *s, charf *buf, ulg stored_len,
                        int eof));
void _tr_align     OF((deflate_state *s));
void _tr_stored_block OF((deflate_state *s, charf *buf, ulg stored_len,
                        int eof));

#define d_code(dist) \
    ((dist) < 256 ? _dist_code[dist] : _dist_code[256+((dist)>>7)])
/* Mapping from a distance to a distance code. dist is the distance - 1 and
 * must not have side effects. _dist_code[256] and _dist_code[257] are never
 * used.
 */

```

```
#ifndef DEBUG
/* Inline versions of _tr_tally for speed: */

#if defined(GEN_TREES_H) || !defined(STDC)
    extern uch _length_code[];
    extern uch _dist_code[];
#else
    extern const uch _length_code[];
    extern const uch _dist_code[];
#endif

# define _tr_tally_lit(s, c, flush) \
    { uch cc = (c); \
      s->d_buf[s->last_lit] = 0; \
      s->l_buf[s->last_lit++] = cc; \
      s->dyn_ltree[cc].Freq++; \
      flush = (s->last_lit == s->lit_bufsize-1); \
    }

# define _tr_tally_dist(s, distance, length, flush) \
    { uch len = (length); \
      ush dist = (distance); \
      s->d_buf[s->last_lit] = dist; \
      s->l_buf[s->last_lit++] = len; \
      dist--; \
      s->dyn_ltree[_length_code[len]+LITERALS+1].Freq++; \
      s->dyn_dtree[d_code(dist)].Freq++; \
      flush = (s->last_lit == s->lit_bufsize-1); \
    }

#else
# define _tr_tally_lit(s, c, flush) flush = _tr_tally(s, 0, c)
# define _tr_tally_dist(s, distance, length, flush) \
    flush = _tr_tally(s, distance, length)
#endif

#endif /* DEFLATE_H */
```



```

/* example.c -- usage example of the zlib compression library
 * Copyright (C) 1995-2004 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id: example.c,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#include <stdio.h>
#include "zlib.h"

#ifdef STDC
# include <string.h>
# include <stdlib.h>
#endif

#if defined(VMS) || defined(RISCOS)
# define TESTFILE "foo-gz"
#else
# define TESTFILE "foo.gz"
#endif

#define CHECK_ERR(err, msg) { \
    if (err != Z_OK) { \
        fprintf(stderr, "%s error: %d\n", msg, err); \
        exit(1); \
    } \
}

const char hello[] = "hello,hello!";
/* "hello world" would be more standard, but the repeated "hello"
 * stresses the compression code better, sorry...
 */

const char dictionary[] = "hello";
uLong dictId; /* Adler32 value of the dictionary */

void test_compress      OF((Byte *compr, uLong comprLen,
                           Byte *uncompr, uLong uncomprLen));
void test_gzio          OF((const char *fname,
                           Byte *uncompr, uLong uncomprLen));
void test_deflate       OF((Byte *compr, uLong comprLen));
void test_inflate       OF((Byte *compr, uLong comprLen,
                           Byte *uncompr, uLong uncomprLen));
void test_large_deflate OF((Byte *compr, uLong comprLen,
                           Byte *uncompr, uLong uncomprLen));
void test_large_inflate OF((Byte *compr, uLong comprLen,
                           Byte *uncompr, uLong uncomprLen));
void test_flush         OF((Byte *compr, uLong *comprLen));
void test_sync          OF((Byte *compr, uLong comprLen,
                           Byte *uncompr, uLong uncomprLen));
void test_dict_deflate  OF((Byte *compr, uLong comprLen));
void test_dict_inflate  OF((Byte *compr, uLong comprLen,
                           Byte *uncompr, uLong uncomprLen));
int main                OF((int argc, char *argv[]));

/* =====
 * Test compress() and uncompress()
 */
void test_compress(compr, comprLen, uncompr, uncomprLen)
    Byte *compr, *uncompr;
    uLong comprLen, uncomprLen;
{
    int err;
    uLong len = (uLong)strlen(hello)+1;

    err = compress(compr, &comprLen, (const Bytef*)hello, len);
    CHECK_ERR(err, "compress");

    strcpy((char*)uncompr, "garbage");

    err = uncompress(uncompr, &uncomprLen, compr, comprLen);
    CHECK_ERR(err, "uncompress");

    if (strcmp((char*)uncompr, hello)) {

```

```

        fprintf(stderr, "bad uncompress\n");
        exit(1);
    } else {
        printf("uncompress(): %s\n", (char *)uncompr);
    }
}

/* =====
 * Test read/write of .gz files
 */
void test_gzio(fname, uncompr, uncomprLen)
    const char *fname; /* compressed file name */
    Byte *uncompr;
    uLong uncomprLen;
{
#ifdef NO_GZCOMPRESS
    fprintf(stderr, "NO_GZCOMPRESS -- gz* functions cannot compress\n");
#else
    int err;
    int len = (int)strlen(hello)+1;
    gzFile file;
    z_off_t pos;

    file = gzopen(fname, "wb");
    if (file == NULL) {
        fprintf(stderr, "gzopen error\n");
        exit(1);
    }
    gzputc(file, 'h');
    if (gzputs(file, "ello") != 4) {
        fprintf(stderr, "gzputs err: %s\n", gzerror(file, &err));
        exit(1);
    }
    if (gzprintf(file, ",%s!", "hello") != 8) {
        fprintf(stderr, "gzprintf err: %s\n", gzerror(file, &err));
        exit(1);
    }
    gzseek(file, 1L, SEEK_CUR); /* add one zero byte */
    gzclose(file);

    file = gzopen(fname, "rb");
    if (file == NULL) {
        fprintf(stderr, "gzopen error\n");
        exit(1);
    }
    strcpy((char*)uncompr, "garbage");

    if (gzread(file, uncompr, (unsigned)uncomprLen) != len) {
        fprintf(stderr, "gzread err: %s\n", gzerror(file, &err));
        exit(1);
    }
    if (strcmp((char*)uncompr, hello)) {
        fprintf(stderr, "bad gzread: %s\n", (char*)uncompr);
        exit(1);
    } else {
        printf("gzread(): %s\n", (char*)uncompr);
    }

    pos = gzseek(file, -8L, SEEK_CUR);
    if (pos != 6 || gztell(file) != pos) {
        fprintf(stderr, "gzseek error, pos=%ld, gztell=%ld\n",
            (long)pos, (long)gztell(file));
        exit(1);
    }

    if (gzgetc(file) != ' ') {
        fprintf(stderr, "gzgetc error\n");
        exit(1);
    }

    if (gzungetc(' ', file) != ' ') {
        fprintf(stderr, "gzungetc error\n");
        exit(1);
    }
}

```

```

gzgets(file, (char*)uncompr, (int)uncomprLen);
if (strlen((char*)uncompr) != 7) { /* "hello!" */
    fprintf(stderr, "gzgets err after gzseek: %s\n", gzerror(file, &err));
    exit(1);
}
if (strcmp((char*)uncompr, hello + 6)) {
    fprintf(stderr, "bad gzgets after gzseek\n");
    exit(1);
} else {
    printf("gzgets() after gzseek: %s\n", (char*)uncompr);
}

gzclose(file);
#endif
}

/* =====
 * Test deflate() with small buffers
 */
void test_deflate(compr, comprLen)
    Byte *compr;
    uLong comprLen;
{
    z_stream c_stream; /* compression stream */
    int err;
    uLong len = (uLong)strlen(hello)+1;

    c_stream.zalloc = (alloc_func)0;
    c_stream.zfree = (free_func)0;
    c_stream.opaque = (voidpf)0;

    err = deflateInit(&c_stream, Z_DEFAULT_COMPRESSION);
    CHECK_ERR(err, "deflateInit");

    c_stream.next_in = (Bytef*)hello;
    c_stream.next_out = compr;

    while (c_stream.total_in != len && c_stream.total_out < comprLen) {
        c_stream.avail_in = c_stream.avail_out = 1; /* force small buffers */
        err = deflate(&c_stream, Z_NO_FLUSH);
        CHECK_ERR(err, "deflate");
    }
    /* Finish the stream, still forcing small buffers: */
    for (;;) {
        c_stream.avail_out = 1;
        err = deflate(&c_stream, Z_FINISH);
        if (err == Z_STREAM_END) break;
        CHECK_ERR(err, "deflate");
    }

    err = deflateEnd(&c_stream);
    CHECK_ERR(err, "deflateEnd");
}

/* =====
 * Test inflate() with small buffers
 */
void test_inflate(compr, comprLen, uncompr, uncomprLen)
    Byte *compr, *uncompr;
    uLong comprLen, uncomprLen;
{
    int err;
    z_stream d_stream; /* decompression stream */

    strcpy((char*)uncompr, "garbage");

    d_stream.zalloc = (alloc_func)0;
    d_stream.zfree = (free_func)0;
    d_stream.opaque = (voidpf)0;

    d_stream.next_in = compr;
    d_stream.avail_in = 0;
    d_stream.next_out = uncompr;

```

```

err = inflateInit(&d_stream);
CHECK_ERR(err, "inflateInit");

while (d_stream.total_out < uncomprLen && d_stream.total_in < comprLen) {
    d_stream.avail_in = d_stream.avail_out = 1; /* force small buffers */
    err = inflate(&d_stream, Z_NO_FLUSH);
    if (err == Z_STREAM_END) break;
    CHECK_ERR(err, "inflate");
}

err = inflateEnd(&d_stream);
CHECK_ERR(err, "inflateEnd");

if (strcmp((char*)uncompr, hello)) {
    fprintf(stderr, "bad inflate\n");
    exit(1);
} else {
    printf("inflate(): %s\n", (char *)uncompr);
}
}

/* =====
 * Test deflate() with large buffers and dynamic change of compression level
 */
void test_large_deflate(compr, comprLen, uncompr, uncomprLen)
    Byte *compr, *uncompr;
    uLong comprLen, uncomprLen;
{
    z_stream c_stream; /* compression stream */
    int err;

    c_stream.zalloc = (alloc_func)0;
    c_stream.zfree = (free_func)0;
    c_stream.opaque = (voidpf)0;

    err = deflateInit(&c_stream, Z_BEST_SPEED);
    CHECK_ERR(err, "deflateInit");

    c_stream.next_out = compr;
    c_stream.avail_out = (uInt)comprLen;

    /* At this point, uncompr is still mostly zeroes, so it should compress
     * very well:
     */
    c_stream.next_in = uncompr;
    c_stream.avail_in = (uInt)uncomprLen;
    err = deflate(&c_stream, Z_NO_FLUSH);
    CHECK_ERR(err, "deflate");
    if (c_stream.avail_in != 0) {
        fprintf(stderr, "deflate not greedy\n");
        exit(1);
    }

    /* Feed in already compressed data and switch to no compression: */
    deflateParams(&c_stream, Z_NO_COMPRESSION, Z_DEFAULT_STRATEGY);
    c_stream.next_in = compr;
    c_stream.avail_in = (uInt)comprLen/2;
    err = deflate(&c_stream, Z_NO_FLUSH);
    CHECK_ERR(err, "deflate");

    /* Switch back to compressing mode: */
    deflateParams(&c_stream, Z_BEST_COMPRESSION, Z_FILTERED);
    c_stream.next_in = uncompr;
    c_stream.avail_in = (uInt)uncomprLen;
    err = deflate(&c_stream, Z_NO_FLUSH);
    CHECK_ERR(err, "deflate");

    err = deflate(&c_stream, Z_FINISH);
    if (err != Z_STREAM_END) {
        fprintf(stderr, "deflate should report Z_STREAM_END\n");
        exit(1);
    }
    err = deflateEnd(&c_stream);

```

```

    CHECK_ERR(err, "deflateEnd");
}

/* =====
 * Test inflate() with large buffers
 */
void test_large_inflate(compr, comprLen, uncompr, uncomprLen)
    Byte *compr, *uncompr;
    uLong comprLen, uncomprLen;
{
    int err;
    z_stream d_stream; /* decompression stream */

    strcpy((char*)uncompr, "garbage");

    d_stream.zalloc = (alloc_func)0;
    d_stream.zfree = (free_func)0;
    d_stream.opaque = (voidpf)0;

    d_stream.next_in = compr;
    d_stream.avail_in = (uInt)comprLen;

    err = inflateInit(&d_stream);
    CHECK_ERR(err, "inflateInit");

    for (;;) {
        d_stream.next_out = uncompr; /* discard the output */
        d_stream.avail_out = (uInt)uncomprLen;
        err = inflate(&d_stream, Z_NO_FLUSH);
        if (err == Z_STREAM_END) break;
        CHECK_ERR(err, "large inflate");
    }

    err = inflateEnd(&d_stream);
    CHECK_ERR(err, "inflateEnd");

    if (d_stream.total_out != 2*uncomprLen + comprLen/2) {
        fprintf(stderr, "bad large inflate: %ld\n", d_stream.total_out);
        exit(1);
    } else {
        printf("large_inflate(): OK\n");
    }
}

/* =====
 * Test deflate() with full flush
 */
void test_flush(compr, comprLen)
    Byte *compr;
    uLong *comprLen;
{
    z_stream c_stream; /* compression stream */
    int err;
    uInt len = (uInt)strlen(hello)+1;

    c_stream.zalloc = (alloc_func)0;
    c_stream.zfree = (free_func)0;
    c_stream.opaque = (voidpf)0;

    err = deflateInit(&c_stream, Z_DEFAULT_COMPRESSION);
    CHECK_ERR(err, "deflateInit");

    c_stream.next_in = (Bytef*)hello;
    c_stream.next_out = compr;
    c_stream.avail_in = 3;
    c_stream.avail_out = (uInt)*comprLen;
    err = deflate(&c_stream, Z_FULL_FLUSH);
    CHECK_ERR(err, "deflate");

    compr[3]++; /* force an error in first compressed block */
    c_stream.avail_in = len - 3;

    err = deflate(&c_stream, Z_FINISH);
    if (err != Z_STREAM_END) {

```

```

        CHECK_ERR(err, "deflate");
    }
    err = deflateEnd(&c_stream);
    CHECK_ERR(err, "deflateEnd");

    *comprLen = c_stream.total_out;
}

/* =====
 * Test inflateSync()
 */
void test_sync(compr, comprLen, uncompr, uncomprLen)
    Byte *compr, *uncompr;
    uLong comprLen, uncomprLen;
{
    int err;
    z_stream d_stream; /* decompression stream */

    strcpy((char*)uncompr, "garbage");

    d_stream.zalloc = (alloc_func)0;
    d_stream.zfree = (free_func)0;
    d_stream.opaque = (voidpf)0;

    d_stream.next_in = compr;
    d_stream.avail_in = 2; /* just read the zlib header */

    err = inflateInit(&d_stream);
    CHECK_ERR(err, "inflateInit");

    d_stream.next_out = uncompr;
    d_stream.avail_out = (uInt)uncomprLen;

    inflate(&d_stream, Z_NO_FLUSH);
    CHECK_ERR(err, "inflate");

    d_stream.avail_in = (uInt)comprLen-2; /* read all compressed data */
    err = inflateSync(&d_stream); /* but skip the damaged part */
    CHECK_ERR(err, "inflateSync");

    err = inflate(&d_stream, Z_FINISH);
    if (err != Z_DATA_ERROR) {
        fprintf(stderr, "inflate should report DATA_ERROR\n");
        /* Because of incorrect Adler32 */
        exit(1);
    }
    err = inflateEnd(&d_stream);
    CHECK_ERR(err, "inflateEnd");

    printf("after inflateSync(): hel%s\n", (char *)uncompr);
}

/* =====
 * Test deflate() with preset dictionary
 */
void test_dict_deflate(compr, comprLen)
    Byte *compr;
    uLong comprLen;
{
    z_stream c_stream; /* compression stream */
    int err;

    c_stream.zalloc = (alloc_func)0;
    c_stream.zfree = (free_func)0;
    c_stream.opaque = (voidpf)0;

    err = deflateInit(&c_stream, Z_BEST_COMPRESSION);
    CHECK_ERR(err, "deflateInit");

    err = deflateSetDictionary(&c_stream,
                              (const Bytef*)dictionary, sizeof(dictionary));
    CHECK_ERR(err, "deflateSetDictionary");

    dictId = c_stream.adler;

```

```

    c_stream.next_out = compr;
    c_stream.avail_out = (uInt)comprLen;

    c_stream.next_in = (Bytef*)hello;
    c_stream.avail_in = (uInt)strlen(hello)+1;

    err = deflate(&c_stream, Z_FINISH);
    if (err != Z_STREAM_END) {
        fprintf(stderr, "deflate should report Z_STREAM_END\n");
        exit(1);
    }
    err = deflateEnd(&c_stream);
    CHECK_ERR(err, "deflateEnd");
}

/* =====
 * Test inflate() with a preset dictionary
 */
void test_dict_inflate(compr, comprLen, uncompr, uncomprLen)
    Byte *compr, *uncompr;
    uLong comprLen, uncomprLen;
{
    int err;
    z_stream d_stream; /* decompression stream */

    strcpy((char*)uncompr, "garbage");

    d_stream.zalloc = (alloc_func)0;
    d_stream.zfree = (free_func)0;
    d_stream.opaque = (voidpf)0;

    d_stream.next_in = compr;
    d_stream.avail_in = (uInt)comprLen;

    err = inflateInit(&d_stream);
    CHECK_ERR(err, "inflateInit");

    d_stream.next_out = uncompr;
    d_stream.avail_out = (uInt)uncomprLen;

    for (;;) {
        err = inflate(&d_stream, Z_NO_FLUSH);
        if (err == Z_STREAM_END) break;
        if (err == Z_NEED_DICT) {
            if (d_stream.adler != dictId) {
                fprintf(stderr, "unexpected dictionary");
                exit(1);
            }
            err = inflateSetDictionary(&d_stream, (const Bytef*)dictionary,
                                      sizeof(dictionary));
        }
        CHECK_ERR(err, "inflate with dict");
    }

    err = inflateEnd(&d_stream);
    CHECK_ERR(err, "inflateEnd");

    if (strcmp((char*)uncompr, hello)) {
        fprintf(stderr, "bad inflate with dict\n");
        exit(1);
    } else {
        printf("inflate with dictionary: %s\n", (char *)uncompr);
    }
}

/* =====
 * Usage:  example [output.gz [input.gz]]
 */

int main(argc, argv)
    int argc;
    char *argv[];
{
    Byte *compr, *uncompr;

```

```
uLong comprLen = 10000*sizeof(int); /* don't overflow on MSDOS */
uLong uncomprLen = comprLen;
static const char* myVersion = ZLIB_VERSION;

if (zlibVersion()[0] != myVersion[0]) {
    fprintf(stderr, "incompatible zlib version\n");
    exit(1);
} else if (strcmp(zlibVersion(), ZLIB_VERSION) != 0) {
    fprintf(stderr, "warning: different zlib version\n");
}

printf("zlib version %s = 0x%04x, compile flags = 0x%lx\n",
       ZLIB_VERSION, ZLIB_VERNUM, zlibCompileFlags());

compr    = (Byte*)calloc((uInt)comprLen, 1);
uncompr  = (Byte*)calloc((uInt)uncomprLen, 1);
/* compr and uncompr are cleared to avoid reading uninitialized
 * data and to ensure that uncompr compresses well.
 */
if (compr == Z_NULL || uncompr == Z_NULL) {
    printf("out of memory\n");
    exit(1);
}
test_compress(compr, comprLen, uncompr, uncomprLen);

test_gzio((argc > 1 ? argv[1] : TESTFILE),
          uncompr, uncomprLen);

test_deflate(compr, comprLen);
test_inflate(compr, comprLen, uncompr, uncomprLen);

test_large_deflate(compr, comprLen, uncompr, uncomprLen);
test_large_inflate(compr, comprLen, uncompr, uncomprLen);

test_flush(compr, &comprLen);
test_sync(compr, comprLen, uncompr, uncomprLen);
comprLen = uncomprLen;

test_dict_deflate(compr, comprLen);
test_dict_inflate(compr, comprLen, uncompr, uncomprLen);

free(compr);
free(uncompr);

return 0;
}
```



```

/* gzio.c -- IO on .gz files
 * Copyright (C) 1995-2005 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 *
 * Compile this file with -DNO_GZCOMPRESS to avoid the compression code.
 */

/* @(#) $Id: gzio.c,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#include <stdio.h>

#include "zutil.h"

#ifdef NO_DEFLATE /* for compatibility with old definition */
# define NO_GZCOMPRESS
#endif

#ifdef NO_DUMMY_DECL
struct internal_state {int dummy;}; /* for buggy compilers */
#endif

#ifdef Z_BUFSIZE
# ifdef MAXSEG_64K
#   define Z_BUFSIZE 4096 /* minimize memory usage for 16-bit DOS */
# else
#   define Z_BUFSIZE 16384
# endif
# endif
#ifdef Z_PRINTF_BUFSIZE
# define Z_PRINTF_BUFSIZE 4096
# endif

#ifdef __MVS__
# pragma map (fdopen , "\174\174FDOPEN")
# FILE *fdopen(int, const char *);
# endif

#ifdef STDC
extern voidp malloc OF((uInt size));
extern void free OF((voidpf ptr));
# endif

#define ALLOC(size) malloc(size)
#define TRYFREE(p) {if (p) free(p);}

static int const gz_magic[2] = {0x1f, 0x8b}; /* gzip magic header */

/* gzip flag byte */
#define ASCII_FLAG 0x01 /* bit 0 set: file probably ascii text */
#define HEAD_CRC 0x02 /* bit 1 set: header CRC present */
#define EXTRA_FIELD 0x04 /* bit 2 set: extra field present */
#define ORIG_NAME 0x08 /* bit 3 set: original file name present */
#define COMMENT 0x10 /* bit 4 set: file comment present */
#define RESERVED 0xE0 /* bits 5..7: reserved */

typedef struct gz_stream {
    z_stream stream;
    int z_err; /* error code for last stream operation */
    int z_eof; /* set if end of input file */
    FILE *file; /* .gz file */
    Byte *inbuf; /* input buffer */
    Byte *outbuf; /* output buffer */
    uLong crc; /* crc32 of uncompressed data */
    char *msg; /* error message */
    char *path; /* path name for debugging only */
    int transparent; /* 1 if input file is not a .gz file */
    char mode; /* 'w' or 'r' */
    z_off_t start; /* start of compressed data in file (header skipped) */
    z_off_t in; /* bytes into deflate or inflate */
    z_off_t out; /* bytes out of deflate or inflate */
    int back; /* one character push-back */
    int last; /* true if push-back is last character */
} gz_stream;

```

```

local gzFile gz_open      OF((const char *path, const char *mode, int fd));
local int do_flush        OF((gzFile file, int flush));
local int get_byte        OF((gz_stream *s));
local void check_header   OF((gz_stream *s));
local int destroy         OF((gz_stream *s));
local void putLong        OF((FILE *file, uLong x));
local uLong getLong       OF((gz_stream *s));

/* =====
   Opens a gzip (.gz) file for reading or writing. The mode parameter
   is as in fopen ("rb" or "wb"). The file is given either by file descriptor
   or path name (if fd == -1).
   gz_open returns NULL if the file could not be opened or if there was
   insufficient memory to allocate the (de)compression state; errno
   can be checked to distinguish the two cases (if errno is zero, the
   zlib error is Z_MEM_ERROR).
*/
local gzFile gz_open (path, mode, fd)
    const char *path;
    const char *mode;
    int fd;
{
    int err;
    int level = Z_DEFAULT_COMPRESSION; /* compression level */
    int strategy = Z_DEFAULT_STRATEGY; /* compression strategy */
    char *p = (char*)mode;
    gz_stream *s;
    char fmode[80]; /* copy of mode, without the compression level */
    char *m = fmode;

    if (!path || !mode) return Z_NULL;

    s = (gz_stream *)ALLOC(sizeof(gz_stream));
    if (!s) return Z_NULL;

    s->stream.zalloc = (alloc_func)0;
    s->stream.zfree = (free_func)0;
    s->stream.opaque = (voidpf)0;
    s->stream.next_in = s->inbuf = Z_NULL;
    s->stream.next_out = s->outbuf = Z_NULL;
    s->stream.avail_in = s->stream.avail_out = 0;
    s->file = NULL;
    s->z_err = Z_OK;
    s->z_eof = 0;
    s->in = 0;
    s->out = 0;
    s->back = EOF;
    s->crc = crc32(0L, Z_NULL, 0);
    s->msg = NULL;
    s->transparent = 0;

    s->path = (char*)ALLOC(strlen(path)+1);
    if (s->path == NULL) {
        return destroy(s), (gzFile)Z_NULL;
    }
    strcpy(s->path, path); /* do this early for debugging */

    s->mode = '\0';
    do {
        if (*p == 'r') s->mode = 'r';
        if (*p == 'w' || *p == 'a') s->mode = 'w';
        if (*p >= '0' && *p <= '9') {
            level = *p - '0';
        } else if (*p == 'f') {
            strategy = Z_FILTERED;
        } else if (*p == 'h') {
            strategy = Z_HUFFMAN_ONLY;
        } else if (*p == 'R') {
            strategy = Z_RLE;
        } else {
            *m++ = *p; /* copy the mode */
        }
    } while (*p++ && m != fmode + sizeof(fmode));
}

```

```

    if (s->mode == '\0') return destroy(s), (gzFile)Z_NULL;

    if (s->mode == 'w') {
#ifdef NO_GZCOMPRESS
        err = Z_STREAM_ERROR;
#else
        err = deflateInit2(&(s->stream), level,
                           Z_DEFLATED, -MAX_WBITS, DEF_MEM_LEVEL, strategy);
        /* windowBits is passed < 0 to suppress zlib header */

        s->stream.next_out = s->outbuf = (Byte*)ALLOC(Z_BUFSIZE);
#endif
        if (err != Z_OK || s->outbuf == Z_NULL) {
            return destroy(s), (gzFile)Z_NULL;
        }
    } else {
        s->stream.next_in = s->inbuf = (Byte*)ALLOC(Z_BUFSIZE);

        err = inflateInit2(&(s->stream), -MAX_WBITS);
        /* windowBits is passed < 0 to tell that there is no zlib header.
         * Note that in this case inflate *requires* an extra "dummy" byte
         * after the compressed stream in order to complete decompression and
         * return Z_STREAM_END. Here the gzip CRC32 ensures that 4 bytes are
         * present after the compressed stream.
         */
        if (err != Z_OK || s->inbuf == Z_NULL) {
            return destroy(s), (gzFile)Z_NULL;
        }
    }
    s->stream.avail_out = Z_BUFSIZE;

    errno = 0;
    s->file = fd < 0 ? F_OPEN(path, fmode) : (FILE*)fdopen(fd, fmode);

    if (s->file == NULL) {
        return destroy(s), (gzFile)Z_NULL;
    }
    if (s->mode == 'w') {
        /* Write a very simple .gz header:
         */
        fprintf(s->file, "%c%c%c%c%c%c%c%c%c%c", gz_magic[0], gz_magic[1],
                    Z_DEFLATED, 0 /*flags*/, 0,0,0,0 /*time*/, 0 /*xflags*/, OS_CODE);
        s->start = 10L;
        /* We use 10L instead of ftell(s->file) to because ftell causes an
         * fflush on some systems. This version of the library doesn't use
         * start anyway in write mode, so this initialization is not
         * necessary.
         */
    } else {
        check_header(s); /* skip the .gz header */
        s->start = ftell(s->file) - s->stream.avail_in;
    }

    return (gzFile)s;
}

/* =====
   Opens a gzip (.gz) file for reading or writing.
*/
gzFile ZEXPORT gzopen (path, mode)
    const char *path;
    const char *mode;
{
    return gz_open (path, mode, -1);
}

/* =====
   Associate a gzFile with the file descriptor fd. fd is not dup'ed here
   to mimic the behavior(u)r of fdopen.
*/
gzFile ZEXPORT gzdopen (fd, mode)
    int fd;
    const char *mode;
{

```

```

char name[46];          /* allow for up to 128-bit integers */

if (fd < 0) return (gzFile)Z_NULL;
sprintf(name, "<fd:%d>", fd); /* for debugging */

return gz_open (name, mode, fd);
}

/* =====
 * Update the compression level and strategy
 */
int ZEXPORT gzsetparams (file, level, strategy)
    gzFile file;
    int level;
    int strategy;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || s->mode != 'w') return Z_STREAM_ERROR;

    /* Make room to allow flushing */
    if (s->stream.avail_out == 0) {

        s->stream.next_out = s->outbuf;
        if (fwrite(s->outbuf, 1, Z_BUFSIZE, s->file) != Z_BUFSIZE) {
            s->z_err = Z_ERRNO;
        }
        s->stream.avail_out = Z_BUFSIZE;
    }

    return deflateParams (&(s->stream), level, strategy);
}

/* =====
    Read a byte from a gz_stream; update next_in and avail_in. Return EOF
    for end of file.
    IN assertion: the stream s has been sucessfully opened for reading.
 */
local int get_byte(s)
    gz_stream *s;
{
    if (s->z_eof) return EOF;
    if (s->stream.avail_in == 0) {
        errno = 0;
        s->stream.avail_in = (uInt)fread(s->inbuf, 1, Z_BUFSIZE, s->file);
        if (s->stream.avail_in == 0) {
            s->z_eof = 1;
            if (ferror(s->file)) s->z_err = Z_ERRNO;
            return EOF;
        }
        s->stream.next_in = s->inbuf;
    }
    s->stream.avail_in--;
    return *(s->stream.next_in)++;
}

/* =====
    Check the gzip header of a gz_stream opened for reading. Set the stream
    mode to transparent if the gzip magic header is not present; set s->err
    to Z_DATA_ERROR if the magic header is present but the rest of the header
    is incorrect.
    IN assertion: the stream s has already been created sucessfully;
    s->stream.avail_in is zero for the first time, but may be non-zero
    for concatenated .gz files.
 */
local void check_header(s)
    gz_stream *s;
{
    int method; /* method byte */
    int flags; /* flags byte */
    uInt len;
    int c;

    /* Assure two bytes in the buffer so we can peek ahead -- handle case

```

```

    where first byte of header is at the end of the buffer after the last
    gzip segment */
len = s->stream.avail_in;
if (len < 2) {
    if (len) s->inbuf[0] = s->stream.next_in[0];
    errno = 0;
    len = (uInt)fread(s->inbuf + len, 1, Z_BUF_SIZE >> len, s->file);
    if (len == 0 && ferror(s->file)) s->z_err = Z_ERRNO;
    s->stream.avail_in += len;
    s->stream.next_in = s->inbuf;
    if (s->stream.avail_in < 2) {
        s->transparent = s->stream.avail_in;
        return;
    }
}

/* Peek ahead to check the gzip magic header */
if (s->stream.next_in[0] != gz_magic[0] ||
    s->stream.next_in[1] != gz_magic[1]) {
    s->transparent = 1;
    return;
}
s->stream.avail_in -= 2;
s->stream.next_in += 2;

/* Check the rest of the gzip header */
method = get_byte(s);
flags = get_byte(s);
if (method != Z_DEFLATED || (flags & RESERVED) != 0) {
    s->z_err = Z_DATA_ERROR;
    return;
}

/* Discard time, xflags and OS code: */
for (len = 0; len < 6; len++) (void)get_byte(s);

if ((flags & EXTRA_FIELD) != 0) { /* skip the extra field */
    len = (uInt)get_byte(s);
    len += ((uInt)get_byte(s)) << 8;
    /* len is garbage if EOF but the loop below will quit anyway */
    while (len-- != 0 && get_byte(s) != EOF) ;
}
if ((flags & ORIG_NAME) != 0) { /* skip the original file name */
    while ((c = get_byte(s)) != 0 && c != EOF) ;
}
if ((flags & COMMENT) != 0) { /* skip the .gz file comment */
    while ((c = get_byte(s)) != 0 && c != EOF) ;
}
if ((flags & HEAD_CRC) != 0) { /* skip the header crc */
    for (len = 0; len < 2; len++) (void)get_byte(s);
}
s->z_err = s->z_eof ? Z_DATA_ERROR : Z_OK;
}

/* =====
 * Cleanup then free the given gz_stream. Return a zlib error code.
 * Try freeing in the reverse order of allocations.
 */
local int destroy (s)
    gz_stream *s;
{
    int err = Z_OK;

    if (!s) return Z_STREAM_ERROR;

    TRYFREE(s->msg);

    if (s->stream.state != NULL) {
        if (s->mode == 'w') {
#ifdef NO_GZCOMPRESS
            err = Z_STREAM_ERROR;
#else
            err = deflateEnd(&(s->stream));
#endif
        }
    }
}

```

```

    } else if (s->mode == 'r') {
        err = inflateEnd(&(s->stream));
    }
}
if (s->file != NULL && fclose(s->file)) {
#ifdef ESPIPE
    if (errno != ESPIPE) /* fclose is broken for pipes in HP/UX */
#endif
        err = Z_ERRNO;
}
if (s->z_err < 0) err = s->z_err;

TRYFREE(s->inbuf);
TRYFREE(s->outbuf);
TRYFREE(s->path);
TRYFREE(s);
return err;
}

/* =====
   Reads the given number of uncompressed bytes from the compressed file.
   gzread returns the number of bytes actually read (0 for end of file).
*/
int ZEXPORT gzread (file, buf, len)
    gzFile file;
    voidp buf;
    unsigned len;
{
    gz_stream *s = (gz_stream*)file;
    Bytef *start = (Bytef*)buf; /* starting point for crc computation */
    Byte *next_out; /* == stream.next_out but not forced far (for MSDOS) */

    if (s == NULL || s->mode != 'r') return Z_STREAM_ERROR;

    if (s->z_err == Z_DATA_ERROR || s->z_err == Z_ERRNO) return -1;
    if (s->z_err == Z_STREAM_END) return 0; /* EOF */

    next_out = (Byte*)buf;
    s->stream.next_out = (Bytef*)buf;
    s->stream.avail_out = len;

    if (s->stream.avail_out && s->back != EOF) {
        *next_out++ = s->back;
        s->stream.next_out++;
        s->stream.avail_out--;
        s->back = EOF;
        s->out++;
        start++;
        if (s->last) {
            s->z_err = Z_STREAM_END;
            return 1;
        }
    }

    while (s->stream.avail_out != 0) {
        if (s->transparent) {
            /* Copy first the lookahead bytes: */
            uInt n = s->stream.avail_in;
            if (n > s->stream.avail_out) n = s->stream.avail_out;
            if (n > 0) {
                zmemcpy(s->stream.next_out, s->stream.next_in, n);
                next_out += n;
                s->stream.next_out = next_out;
                s->stream.next_in += n;
                s->stream.avail_out -= n;
                s->stream.avail_in -= n;
            }
            if (s->stream.avail_out > 0) {
                s->stream.avail_out -=
                    (uInt)fread(next_out, 1, s->stream.avail_out, s->file);
            }
            len -= s->stream.avail_out;
            s->in += len;
        }
    }
}

```

```

        s->out += len;
        if (len == 0) s->z_eof = 1;
        return (int)len;
    }
    if (s->stream.avail_in == 0 && !s->z_eof) {

        errno = 0;
        s->stream.avail_in = (uInt)fread(s->inbuf, 1, Z_BUFSIZE, s->file);
        if (s->stream.avail_in == 0) {
            s->z_eof = 1;
            if (ferror(s->file)) {
                s->z_err = Z_ERRNO;
                break;
            }
        }
        s->stream.next_in = s->inbuf;
    }
    s->in += s->stream.avail_in;
    s->out += s->stream.avail_out;
    s->z_err = inflate(&(s->stream), Z_NO_FLUSH);
    s->in -= s->stream.avail_in;
    s->out -= s->stream.avail_out;

    if (s->z_err == Z_STREAM_END) {
        /* Check CRC and original size */
        s->crc = crc32(s->crc, start, (uInt)(s->stream.next_out - start));
        start = s->stream.next_out;

        if (getLong(s) != s->crc) {
            s->z_err = Z_DATA_ERROR;
        } else {
            (void)getLong(s);
            /* The uncompressed length returned by above getlong() may be
             * different from s->out in case of concatenated .gz files.
             * Check for such files:
             */
            check_header(s);
            if (s->z_err == Z_OK) {
                inflateReset(&(s->stream));
                s->crc = crc32(0L, Z_NULL, 0);
            }
        }
    }
    if (s->z_err != Z_OK || s->z_eof) break;
}
s->crc = crc32(s->crc, start, (uInt)(s->stream.next_out - start));

if (len == s->stream.avail_out &&
    (s->z_err == Z_DATA_ERROR || s->z_err == Z_ERRNO))
    return -1;
return (int)(len - s->stream.avail_out);
}

/* =====
   Reads one byte from the compressed file. gzgetc returns this byte
   or -1 in case of end of file or error.
*/
int ZEXPORT gzgetc(file)
    gzFile file;
{
    unsigned char c;

    return gzread(file, &c, 1) == 1 ? c : -1;
}

/* =====
   Push one byte back onto the stream.
*/
int ZEXPORT gzungetc(c, file)
    int c;
    gzFile file;
{

```

```

gz_stream *s = (gz_stream*)file;

if (s == NULL || s->mode != 'r' || c == EOF || s->back != EOF) return EOF;
s->back = c;
s->out--;
s->last = (s->z_err == Z_STREAM_END);
if (s->last) s->z_err = Z_OK;
s->z_eof = 0;
return c;
}

/* =====
   Reads bytes from the compressed file until len-1 characters are
   read, or a newline character is read and transferred to buf, or an
   end-of-file condition is encountered. The string is then terminated
   with a null character.
   gzgets returns buf, or Z_NULL in case of error.

   The current implementation is not optimized at all.
*/
char * ZEXPORT gzgets(file, buf, len)
gzFile file;
char *buf;
int len;
{
    char *b = buf;
    if (buf == Z_NULL || len <= 0) return Z_NULL;

    while (--len > 0 && gzread(file, buf, 1) == 1 && *buf++ != '\n') ;
    *buf = '\0';
    return b == buf && len > 0 ? Z_NULL : b;
}

#ifdef NO_GZCOMPRESS
/* =====
   Writes the given number of uncompressed bytes into the compressed file.
   gzwrite returns the number of bytes actually written (0 in case of error).
*/
int ZEXPORT gzwrite (file, buf, len)
gzFile file;
voidpc buf;
unsigned len;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || s->mode != 'w') return Z_STREAM_ERROR;

    s->stream.next_in = (Bytef*)buf;
    s->stream.avail_in = len;

    while (s->stream.avail_in != 0) {
        if (s->stream.avail_out == 0) {
            s->stream.next_out = s->outbuf;
            if (fwrite(s->outbuf, 1, Z_BUFSIZE, s->file) != Z_BUFSIZE) {
                s->z_err = Z_ERRNO;
                break;
            }
            s->stream.avail_out = Z_BUFSIZE;
        }
        s->in += s->stream.avail_in;
        s->out += s->stream.avail_out;
        s->z_err = deflate(&(s->stream), Z_NO_FLUSH);
        s->in -= s->stream.avail_in;
        s->out -= s->stream.avail_out;
        if (s->z_err != Z_OK) break;
    }
    s->crc = crc32(s->crc, (const Bytef *)buf, len);

    return (int)(len - s->stream.avail_in);
}

```



```

/* =====
   Converts, formats, and writes the args to the compressed file under
   control of the format string, as in fprintf. gzprintf returns the number of
   uncompressed bytes actually written (0 in case of error).
*/
#ifdef STDC
#include <stdarg.h>

int ZEXPORTVA gzprintf (gzFile file, const char *format, /* args */ ...)
{
    char buf[Z_PRINTF_BUFSIZE];
    va_list va;
    int len;

    buf[sizeof(buf) - 1] = 0;
    va_start(va, format);
#ifdef NO_vsnprintf
#ifdef HAS_vsprintf_void
    (void)vsprintf(buf, format, va);
    va_end(va);
    for (len = 0; len < sizeof(buf); len++)
        if (buf[len] == 0) break;
#else
    len = vsprintf(buf, format, va);
    va_end(va);
#endif
#else
#ifdef HAS_vsnprintf_void
    (void)vsnprintf(buf, sizeof(buf), format, va);
    va_end(va);
    len = strlen(buf);
#else
    len = vsnprintf(buf, sizeof(buf), format, va);
    va_end(va);
#endif
#endif
    if (len <= 0 || len >= (int)sizeof(buf) || buf[sizeof(buf) - 1] != 0)
        return 0;
    return gzwrite(file, buf, (unsigned)len);
}
#else /* not ANSI C */

int ZEXPORTVA gzprintf (file, format, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10,
                       a11, a12, a13, a14, a15, a16, a17, a18, a19, a20)
{
    gzFile file;
    const char *format;
    int a1, a2, a3, a4, a5, a6, a7, a8, a9, a10,
        a11, a12, a13, a14, a15, a16, a17, a18, a19, a20;
    {
        char buf[Z_PRINTF_BUFSIZE];
        int len;

        buf[sizeof(buf) - 1] = 0;
#ifdef NO_snprintf
#ifdef HAS_sprintf_void
        sprintf(buf, format, a1, a2, a3, a4, a5, a6, a7, a8,
                a9, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19, a20);
        for (len = 0; len < sizeof(buf); len++)
            if (buf[len] == 0) break;
#else
        len = sprintf(buf, format, a1, a2, a3, a4, a5, a6, a7, a8,
                      a9, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19, a20);
#endif
#else
#ifdef HAS_snprintf_void
        snprintf(buf, sizeof(buf), format, a1, a2, a3, a4, a5, a6, a7, a8,
                a9, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19, a20);
        len = strlen(buf);
#else
        len = snprintf(buf, sizeof(buf), format, a1, a2, a3, a4, a5, a6, a7, a8,
                      a9, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19, a20);
#endif
#endif
    }
}

```

```

#endif
    if (len <= 0 || len >= sizeof(buf) || buf[sizeof(buf) - 1] != 0)
        return 0;
    return gzwrite(file, buf, len);
}
#endif

/* =====
   Writes c, converted to an unsigned char, into the compressed file.
   gzputc returns the value that was written, or -1 in case of error.
*/
int ZEXPORT gzputc(file, c)
    gzFile file;
    int c;
{
    unsigned char cc = (unsigned char) c; /* required for big endian systems */

    return gzwrite(file, &cc, 1) == 1 ? (int)cc : -1;
}

/* =====
   Writes the given null-terminated string to the compressed file, excluding
   the terminating null character.
   gzputs returns the number of characters written, or -1 in case of error.
*/
int ZEXPORT gzputs(file, s)
    gzFile file;
    const char *s;
{
    return gzwrite(file, (char*)s, (unsigned)strlen(s));
}

/* =====
   Flushes all pending output into the compressed file. The parameter
   flush is as in the deflate() function.
*/
local int do_flush (file, flush)
    gzFile file;
    int flush;
{
    uInt len;
    int done = 0;
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || s->mode != 'w') return Z_STREAM_ERROR;

    s->stream.avail_in = 0; /* should be zero already anyway */

    for (;;) {
        len = Z_BUF_SIZE - s->stream.avail_out;

        if (len != 0) {
            if ((uInt)fwrite(s->outbuf, 1, len, s->file) != len) {
                s->z_err = Z_ERRNO;
                return Z_ERRNO;
            }
            s->stream.next_out = s->outbuf;
            s->stream.avail_out = Z_BUF_SIZE;
        }
        if (done) break;
        s->out += s->stream.avail_out;
        s->z_err = deflate(&(s->stream), flush);
        s->out -= s->stream.avail_out;

        /* Ignore the second of two consecutive flushes: */
        if (len == 0 && s->z_err == Z_BUF_ERROR) s->z_err = Z_OK;

        /* deflate has finished flushing only when it hasn't used up
         * all the available space in the output buffer:
         */
        done = (s->stream.avail_out != 0 || s->z_err == Z_STREAM_END);
    }
}

```

```

        if (s->z_err != Z_OK && s->z_err != Z_STREAM_END) break;
    }
    return s->z_err == Z_STREAM_END ? Z_OK : s->z_err;
}

int ZEXPORT gzflush (file, flush)
    gzFile file;
    int flush;
{
    gz_stream *s = (gz_stream*)file;
    int err = do_flush (file, flush);

    if (err) return err;
    fflush(s->file);
    return s->z_err == Z_STREAM_END ? Z_OK : s->z_err;
}
#endif /* NO_GZCOMPRESS */

/* =====
   Sets the starting position for the next gzread or gzwrite on the given
   compressed file. The offset represents a number of bytes in the
   gzseek returns the resulting offset location as measured in bytes from
   the beginning of the uncompressed stream, or -1 in case of error.
   SEEK_END is not implemented, returns error.
   In this version of the library, gzseek can be extremely slow.
*/
z_off_t ZEXPORT gzseek (file, offset, whence)
    gzFile file;
    z_off_t offset;
    int whence;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || whence == SEEK_END ||
        s->z_err == Z_ERRNO || s->z_err == Z_DATA_ERROR) {
        return -1L;
    }

    if (s->mode == 'w') {
#ifdef NO_GZCOMPRESS
        return -1L;
#else
        if (whence == SEEK_SET) {
            offset -= s->in;
        }
        if (offset < 0) return -1L;

        /* At this point, offset is the number of zero bytes to write. */
        if (s->inbuf == Z_NULL) {
            s->inbuf = (Byte*)ALLOC(Z_BUFSIZE); /* for seeking */
            if (s->inbuf == Z_NULL) return -1L;
            zmemzero(s->inbuf, Z_BUFSIZE);
        }
        while (offset > 0) {
            uInt size = Z_BUFSIZE;
            if (offset < Z_BUFSIZE) size = (uInt)offset;

            size = gzwrite(file, s->inbuf, size);
            if (size == 0) return -1L;

            offset -= size;
        }
        return s->in;
#endif
    }
}
/* Rest of function is for reading only */

/* compute absolute position */
if (whence == SEEK_CUR) {
    offset += s->out;
}
if (offset < 0) return -1L;

if (s->transparent) {

```

```

    /* map to fseek */
    s->back = EOF;
    s->stream.avail_in = 0;
    s->stream.next_in = s->inbuf;
    if (fseek(s->file, offset, SEEK_SET) < 0) return -1L;

    s->in = s->out = offset;
    return offset;
}

/* For a negative seek, rewind and use positive seek */
if (offset >= s->out) {
    offset -= s->out;
} else if (gzrewind(file) < 0) {
    return -1L;
}
/* offset is now the number of bytes to skip. */

if (offset != 0 && s->outbuf == Z_NULL) {
    s->outbuf = (Byte*)ALLOC(Z_BUFSIZE);
    if (s->outbuf == Z_NULL) return -1L;
}
if (offset && s->back != EOF) {
    s->back = EOF;
    s->out++;
    offset--;
    if (s->last) s->z_err = Z_STREAM_END;
}
while (offset > 0) {
    int size = Z_BUFSIZE;
    if (offset < Z_BUFSIZE) size = (int)offset;

    size = gzread(file, s->outbuf, (uInt)size);
    if (size <= 0) return -1L;
    offset -= size;
}
return s->out;
}

/* =====
   Rewinds input file.
*/
int ZEXPORT gzrewind (file)
    gzFile file;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || s->mode != 'r') return -1;

    s->z_err = Z_OK;
    s->z_eof = 0;
    s->back = EOF;
    s->stream.avail_in = 0;
    s->stream.next_in = s->inbuf;
    s->crc = crc32(0L, Z_NULL, 0);
    if (!s->transparent) (void)inflateReset(&s->stream);
    s->in = 0;
    s->out = 0;
    return fseek(s->file, s->start, SEEK_SET);
}

/* =====
   Returns the starting position for the next gzread or gzwrite on the
   given compressed file. This position represents a number of bytes in the
   uncompressed data stream.
*/
z_off_t ZEXPORT gztell (file)
    gzFile file;
{
    return gzseek(file, 0L, SEEK_CUR);
}

/* =====
   Returns 1 when EOF has previously been detected reading the given

```

```

    input stream, otherwise zero.
*/
int ZEXPORT gzeof (file)
    gzFile file;
{
    gz_stream *s = (gz_stream*)file;

    /* With concatenated compressed files that can have embedded
     * crc trailers, z_eof is no longer the only/best indicator of EOF
     * on a gz_stream. Handle end-of-stream error explicitly here.
     */
    if (s == NULL || s->mode != 'r') return 0;
    if (s->z_eof) return 1;
    return s->z_err == Z_STREAM_END;
}

/* =====
   Returns 1 if reading and doing so transparently, otherwise zero.
*/
int ZEXPORT gzdirect (file)
    gzFile file;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || s->mode != 'r') return 0;
    return s->transparent;
}

/* =====
   Outputs a long in LSB order to the given file
*/
local void putLong (file, x)
    FILE *file;
    uLong x;
{
    int n;
    for (n = 0; n < 4; n++) {
        fputc((int)(x & 0xff), file);
        x >>= 8;
    }
}

/* =====
   Reads a long in LSB order from the given gz_stream. Sets z_err in case
   of error.
*/
local uLong getLong (s)
    gz_stream *s;
{
    uLong x = (uLong)get_byte(s);
    int c;

    x += ((uLong)get_byte(s))<<8;
    x += ((uLong)get_byte(s))<<16;
    c = get_byte(s);
    if (c == EOF) s->z_err = Z_DATA_ERROR;
    x += ((uLong)c)<<24;
    return x;
}

/* =====
   Flushes all pending output if necessary, closes the compressed file
   and deallocates all the (de)compression state.
*/
int ZEXPORT gzclose (file)
    gzFile file;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL) return Z_STREAM_ERROR;

    if (s->mode == 'w') {
#ifdef NO_GZCOMPRESS
        return Z_STREAM_ERROR;

```

```

#else
    if (do_flush (file, Z_FINISH) != Z_OK)
        return destroy((gz_stream*)file);

    putLong (s->file, s->crc);
    putLong (s->file, (uLong)(s->in & 0xffffffff));
#endif
}
return destroy((gz_stream*)file);
}

#ifdef STDC
# define zstrerror(errnum) strerror(errnum)
#else
# define zstrerror(errnum) ""
#endif

/* =====
   Returns the error message for the last error which occurred on the
   given compressed file. errnum is set to zlib error number. If an
   error occurred in the file system and not in the compression library,
   errnum is set to Z_ERRNO and the application may consult errno
   to get the exact error code.
*/
const char * ZEXPORT gzerror (file, errnum)
    gzFile file;
    int *errnum;
{
    char *m;
    gz_stream *s = (gz_stream*)file;

    if (s == NULL) {
        *errnum = Z_STREAM_ERROR;
        return (const char*)ERR_MSG(Z_STREAM_ERROR);
    }
    *errnum = s->z_err;
    if (*errnum == Z_OK) return (const char*)"";

    m = (char*)(*errnum == Z_ERRNO ? zstrerror(errno) : s->stream.msg);

    if (m == NULL || *m == '\0') m = (char*)ERR_MSG(s->z_err);

    TRYFREE(s->msg);
    s->msg = (char*)ALLOC(strlen(s->path) + strlen(m) + 3);
    if (s->msg == Z_NULL) return (const char*)ERR_MSG(Z_MEM_ERROR);
    strcpy(s->msg, s->path);
    strcat(s->msg, ": ");
    strcat(s->msg, m);
    return (const char*)s->msg;
}

/* =====
   Clear the error and end-of-file flags, and do the same for the real file.
*/
void ZEXPORT gzcLEARerr (file)
    gzFile file;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL) return;
    if (s->z_err != Z_STREAM_END) s->z_err = Z_OK;
    s->z_eof = 0;
    clearerr(s->file);
}

```

```

/* inffback.c -- inflate using a call-back interface
 * Copyright (C) 1995-2005 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 * This code is largely copied from inflate.c. Normally either inffback.o or
 * inflate.o would be linked into an application--not both. The interface
 * with inffast.c is retained so that optimized assembler-coded versions of
 * inflate_fast() can be used with either inflate.c or inffback.c.
 */

#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"

/* function prototypes */
local void fixedtables OF((struct inflate_state FAR *state));

/*
 * strm provides memory allocation functions in zalloc and zfree, or
 * Z_NULL to use the library memory allocation functions.
 *
 * windowBits is in the range 8..15, and window is a user-supplied
 * window and output buffer that is 2**windowBits bytes.
 */
int ZEXPORT inflateBackInit_(strm, windowBits, window, version, stream_size)
z_streamp strm;
int windowBits;
unsigned char FAR *window;
const char *version;
int stream_size;
{
    struct inflate_state FAR *state;

    if (version == Z_NULL || version[0] != ZLIB_VERSION[0] ||
        stream_size != (int)(sizeof(z_stream)))
        return Z_VERSION_ERROR;
    if (strm == Z_NULL || window == Z_NULL ||
        windowBits < 8 || windowBits > 15)
        return Z_STREAM_ERROR;
    strm->msg = Z_NULL; /* in case we return an error */
    if (strm->zalloc == (alloc_func)0) {
        strm->zalloc = zcalloc;
        strm->opaque = (voidpf)0;
    }
    if (strm->zfree == (free_func)0) strm->zfree = zcfree;
    state = (struct inflate_state FAR *)ZALLOC(strm, 1,
                                                sizeof(struct inflate_state));
    if (state == Z_NULL) return Z_MEM_ERROR;
    Tracev((stderr, "inflate: allocated\n"));
    strm->state = (struct internal_state FAR *)state;
    state->dmax = 32768U;
    state->wbits = windowBits;
    state->wsize = 1U << windowBits;
    state->window = window;
    state->write = 0;
    state->whave = 0;
    return Z_OK;
}

/*
 * Return state with length and distance decoding tables and index sizes set to
 * fixed code decoding. Normally this returns fixed tables from inffixed.h.
 * If BUILDFIXED is defined, then instead this routine builds the tables the
 * first time it's called, and returns those tables the first time and
 * thereafter. This reduces the size of the code by about 2K bytes, in
 * exchange for a little execution time. However, BUILDFIXED should not be
 * used for threaded applications, since the rewriting of the tables and virgin
 * may not be thread-safe.
 */
local void fixedtables(state)
struct inflate_state FAR *state;

```

```

{
#ifdef BUILDFIXED
    static int virgin = 1;
    static code *lenfix, *distfix;
    static code fixed[544];

    /* build fixed huffman tables if first call (may not be thread safe) */
    if (virgin) {
        unsigned sym, bits;
        static code *next;

        /* literal/length table */
        sym = 0;
        while (sym < 144) state->lens[sym++] = 8;
        while (sym < 256) state->lens[sym++] = 9;
        while (sym < 280) state->lens[sym++] = 7;
        while (sym < 288) state->lens[sym++] = 8;
        next = fixed;
        lenfix = next;
        bits = 9;
        inflate_table(LENS, state->lens, 288, &(amp;next), &(amp;bits), state->work);

        /* distance table */
        sym = 0;
        while (sym < 32) state->lens[sym++] = 5;
        distfix = next;
        bits = 5;
        inflate_table(DISTS, state->lens, 32, &(amp;next), &(amp;bits), state->work);

        /* do this just once */
        virgin = 0;
    }
#else /* !BUILDFIXED */
    # include "inffixed.h"
#endif /* BUILDFIXED */
    state->lencode = lenfix;
    state->lenbits = 9;
    state->distcode = distfix;
    state->distbits = 5;
}

/* Macros for inflateBack(): */

/* Load returned state from inflate_fast() */
#define LOAD() \
do { \
    put = strm->next_out; \
    left = strm->avail_out; \
    next = strm->next_in; \
    have = strm->avail_in; \
    hold = state->hold; \
    bits = state->bits; \
} while (0)

/* Set state from registers for inflate_fast() */
#define RESTORE() \
do { \
    strm->next_out = put; \
    strm->avail_out = left; \
    strm->next_in = next; \
    strm->avail_in = have; \
    state->hold = hold; \
    state->bits = bits; \
} while (0)

/* Clear the input bit accumulator */
#define INITBITS() \
do { \
    hold = 0; \
    bits = 0; \
} while (0)

/* Assure that some input is available.  If input is requested, but denied,
then return a Z_BUF_ERROR from inflateBack(). */

```



```

#define PULL() \
do { \
    if (have == 0) { \
        have = in(in_desc, &next); \
        if (have == 0) { \
            next = Z_NULL; \
            ret = Z_BUF_ERROR; \
            goto inf_leave; \
        } \
    } \
} while (0)

/* Get a byte of input into the bit accumulator, or return from inflateBack()
with an error if there is no input available. */
#define PULLBYTE() \
do { \
    PULL(); \
    have--; \
    hold += (unsigned long)(*next++) << bits; \
    bits += 8; \
} while (0)

/* Assure that there are at least n bits in the bit accumulator.  If there is
not enough available input to do that, then return from inflateBack() with
an error. */
#define NEEDBITS(n) \
do { \
    while (bits < (unsigned)(n)) \
        PULLBYTE(); \
} while (0)

/* Return the low n bits of the bit accumulator (n < 16) */
#define BITS(n) \
((unsigned)hold & ((1U << (n)) - 1))

/* Remove n bits from the bit accumulator */
#define DROPBITS(n) \
do { \
    hold >>= (n); \
    bits -= (unsigned)(n); \
} while (0)

/* Remove zero to seven bits as needed to go to a byte boundary */
#define BYTEBITS() \
do { \
    hold >>= bits & 7; \
    bits -= bits & 7; \
} while (0)

/* Assure that some output space is available, by writing out the window
if it's full.  If the write fails, return from inflateBack() with a
Z_BUF_ERROR. */
#define ROOM() \
do { \
    if (left == 0) { \
        put = state->window; \
        left = state->wsize; \
        state->whave = left; \
        if (out(out_desc, put, left)) { \
            ret = Z_BUF_ERROR; \
            goto inf_leave; \
        } \
    } \
} while (0)

/*
strm provides the memory allocation functions and window buffer on input,
and provides information on the unused input on return.  For Z_DATA_ERROR
returns, strm will also provide an error message.

in() and out() are the call-back input and output functions.  When
inflateBack() needs more input, it calls in().  When inflateBack() has
filled the window with output, or when it completes with data in the
window, it calls out() to write out the data.  The application must not

```

change the provided input until `in()` is called again or `inflateBack()` returns. The application must not change the window/output buffer until `inflateBack()` returns.

`in()` and `out()` are called with a descriptor parameter provided in the `inflateBack()` call. This parameter can be a structure that provides the information required to do the read or write, as well as accumulated information on the input and output such as totals and check values.

`in()` should return zero on failure. `out()` should return non-zero on failure. If either `in()` or `out()` fails, then `inflateBack()` returns a `Z_BUF_ERROR`. `strm->next_in` can be checked for `Z_NULL` to see whether it was `in()` or `out()` that caused in the error. Otherwise, `inflateBack()` returns `Z_STREAM_END` on success, `Z_DATA_ERROR` for an deflate format error, or `Z_MEM_ERROR` if it could not allocate memory for the state. `inflateBack()` can also return `Z_STREAM_ERROR` if the input parameters are not correct, i.e. `strm` is `Z_NULL` or the state was not initialized.

```

*/
int ZEXPORT inflateBack(strm, in, in_desc, out, out_desc)
z_streamp strm;
in_func in;
void FAR *in_desc;
out_func out;
void FAR *out_desc;
{
    struct inflate_state FAR *state;
    unsigned char FAR *next;      /* next input */
    unsigned char FAR *put;       /* next output */
    unsigned have, left;          /* available input and output */
    unsigned long hold;           /* bit buffer */
    unsigned bits;                /* bits in bit buffer */
    unsigned copy;                /* number of stored or match bytes to copy */
    unsigned char FAR *from;      /* where to copy match bytes from */
    code this;                    /* current decoding table entry */
    code last;                    /* parent table entry */
    unsigned len;                 /* length to copy for repeats, bits to drop */
    int ret;                      /* return code */
    static const unsigned short order[19] = /* permutation of code lengths */
        {16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15};

    /* Check that the strm exists and that the state was initialized */
    if (strm == Z_NULL || strm->state == Z_NULL)
        return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;

    /* Reset the state */
    strm->msg = Z_NULL;
    state->mode = TYPE;
    state->last = 0;
    state->whave = 0;
    next = strm->next_in;
    have = next != Z_NULL ? strm->avail_in : 0;
    hold = 0;
    bits = 0;
    put = state->window;
    left = state->wsize;

    /* Inflate until end of block marked as last */
    for (;;)
        switch (state->mode) {
            case TYPE:
                /* determine and dispatch block type */
                if (state->last) {
                    BYTEBITS();
                    state->mode = DONE;
                    break;
                }
                NEEDBITS(3);
                state->last = BITS(1);
                DROPBITS(1);
                switch (BITS(2)) {
                    case 0: /* stored block */
                        Tracev((stderr, "inflate: stored block%s\n",
                                state->last ? " (last)" : ""));

```

```

        state->mode = STORED;
        break;
    case 1: /* fixed block */
        fixedtables(state);
        Tracev((stderr, "inflate: fixed codes block%s\n",
            state->last ? " (last)" : ""));
        state->mode = LEN; /* decode codes */
        break;
    case 2: /* dynamic block */
        Tracev((stderr, "inflate: dynamic codes block%s\n",
            state->last ? " (last)" : ""));
        state->mode = TABLE;
        break;
    case 3:
        strm->msg = (char *) "invalid block type";
        state->mode = BAD;
    }
    DROPBITS(2);
    break;

case STORED:
    /* get and verify stored block length */
    BYTEBITS(); /* go to byte boundary */
    NEEDBITS(32);
    if ((hold & 0xffff) != ((hold >> 16) ^ 0xffff)) {
        strm->msg = (char *) "invalid stored block lengths";
        state->mode = BAD;
        break;
    }
    state->length = (unsigned)hold & 0xffff;
    Tracev((stderr, "inflate: stored length %u\n",
        state->length));
    INITBITS();

    /* copy stored block from input to output */
    while (state->length != 0) {
        copy = state->length;
        PULL();
        ROOM();
        if (copy > have) copy = have;
        if (copy > left) copy = left;
        zmemcpy(put, next, copy);
        have -= copy;
        next += copy;
        left -= copy;
        put += copy;
        state->length -= copy;
    }
    Tracev((stderr, "inflate: stored end\n"));
    state->mode = TYPE;
    break;

case TABLE:
    /* get dynamic table entries descriptor */
    NEEDBITS(14);
    state->nlen = BITS(5) + 257;
    DROPBITS(5);
    state->ndist = BITS(5) + 1;
    DROPBITS(5);
    state->ncode = BITS(4) + 4;
    DROPBITS(4);
#ifndef PKZIP_BUG_WORKAROUND
    if (state->nlen > 286 || state->ndist > 30) {
        strm->msg = (char *) "too many length or distance symbols";
        state->mode = BAD;
        break;
    }
#endif
    Tracev((stderr, "inflate: table sizes ok\n"));

    /* get code length code lengths (not a typo) */
    state->have = 0;
    while (state->have < state->ncode) {
        NEEDBITS(3);

```

```

        state->lens[order[state->have++]] = (unsigned short)BITS(3);
        DROPBITS(3);
    }
    while (state->have < 19)
        state->lens[order[state->have++]] = 0;
    state->next = state->codes;
    state->lencode = (code const FAR *)(state->next);
    state->lenbits = 7;
    ret = inflate_table(CODES, state->lens, 19, &(amp;state->next),
        &(state->lenbits), state->work);

    if (ret) {
        strm->msg = (char *)"invalid code lengths set";
        state->mode = BAD;
        break;
    }
    Tracev((stderr, "inflate:   code lengths ok\n"));

    /* get length and distance code code lengths */
    state->have = 0;
    while (state->have < state->nlen + state->ndist) {
        for (;;) {
            this = state->lencode[BITS(state->lenbits)];
            if ((unsigned)(this.bits) <= bits) break;
            PULLBYTE();
        }
        if (this.val < 16) {
            NEEDBITS(this.bits);
            DROPBITS(this.bits);
            state->lens[state->have++] = this.val;
        }
        else {
            if (this.val == 16) {
                NEEDBITS(this.bits + 2);
                DROPBITS(this.bits);
                if (state->have == 0) {
                    strm->msg = (char *)"invalid bit length repeat";
                    state->mode = BAD;
                    break;
                }
                len = (unsigned)(state->lens[state->have - 1]);
                copy = 3 + BITS(2);
                DROPBITS(2);
            }
            else if (this.val == 17) {
                NEEDBITS(this.bits + 3);
                DROPBITS(this.bits);
                len = 0;
                copy = 3 + BITS(3);
                DROPBITS(3);
            }
            else {
                NEEDBITS(this.bits + 7);
                DROPBITS(this.bits);
                len = 0;
                copy = 11 + BITS(7);
                DROPBITS(7);
            }
            if (state->have + copy > state->nlen + state->ndist) {
                strm->msg = (char *)"invalid bit length repeat";
                state->mode = BAD;
                break;
            }
            while (copy--)
                state->lens[state->have++] = (unsigned short)len;
        }
    }

    /* handle error breaks in while */
    if (state->mode == BAD) break;

    /* build code tables */
    state->next = state->codes;
    state->lencode = (code const FAR *)(state->next);
    state->lenbits = 9;

```

```

ret = inflate_table(LENS, state->lens, state->nlen, &(state->next),
                    &(state->lenbits), state->work);

if (ret) {
    strm->msg = (char *) "invalid literal/lengths set";
    state->mode = BAD;
    break;
}
state->distcode = (code const FAR *) (state->next);
state->distbits = 6;
ret = inflate_table(DISTS, state->lens + state->nlen, state->ndist,
                    &(state->next), &(state->distbits), state->work);

if (ret) {
    strm->msg = (char *) "invalid distances set";
    state->mode = BAD;
    break;
}
Tracev((stderr, "inflate:   codes ok\n"));
state->mode = LEN;

case LEN:
    /* use inflate_fast() if we have enough input and output */
    if (have >= 6 && left >= 258) {
        RESTORE();
        if (state->whave < state->wsize)
            state->whave = state->wsize - left;
        inflate_fast(strm, state->wsize);
        LOAD();
        break;
    }

    /* get a literal, length, or end-of-block code */
    for (;;) {
        this = state->lencode[BITS(state->lenbits)];
        if ((unsigned)(this.bits) <= bits) break;
        PULLBYTE();
    }
    if (this.op && (this.op & 0xf0) == 0) {
        last = this;
        for (;;) {
            this = state->lencode[last.val +
                              (BITS(last.bits + last.op) >> last.bits)];
            if ((unsigned)(last.bits + this.bits) <= bits) break;
            PULLBYTE();
        }
        DROPBITS(last.bits);
    }
    DROPBITS(this.bits);
    state->length = (unsigned)this.val;

    /* process literal */
    if (this.op == 0) {
        Tracevv((stderr, this.val >= 0x20 && this.val < 0x7f ?
                "inflate:   literal '%c'\n" :
                "inflate:   literal 0x%02x\n", this.val));
        ROOM();
        *put++ = (unsigned char)(state->length);
        left--;
        state->mode = LEN;
        break;
    }

    /* process end of block */
    if (this.op & 32) {
        Tracev((stderr, "inflate:   end of block\n"));
        state->mode = TYPE;
        break;
    }

    /* invalid code */
    if (this.op & 64) {
        strm->msg = (char *) "invalid literal/length code";
        state->mode = BAD;
        break;
    }

```

```

/* length code -- get extra bits, if any */
state->extra = (unsigned)(this.op) & 15;
if (state->extra != 0) {
    NEEDBITS(state->extra);
    state->length += BITS(state->extra);
    DROPBITS(state->extra);
}
Tracevv((stderr, "inflate:    length %u\n", state->length));

/* get distance code */
for (;;) {
    this = state->distcode[BITS(state->distbits)];
    if ((unsigned)(this.bits) <= bits) break;
    PULLBYTE();
}
if ((this.op & 0xf0) == 0) {
    last = this;
    for (;;) {
        this = state->distcode[last.val +
            (BITS(last.bits + last.op) >> last.bits)];
        if ((unsigned)(last.bits + this.bits) <= bits) break;
        PULLBYTE();
    }
    DROPBITS(last.bits);
}
DROPBITS(this.bits);
if (this.op & 64) {
    strm->msg = (char *) "invalid distance code";
    state->mode = BAD;
    break;
}
state->offset = (unsigned)this.val;

/* get distance extra bits, if any */
state->extra = (unsigned)(this.op) & 15;
if (state->extra != 0) {
    NEEDBITS(state->extra);
    state->offset += BITS(state->extra);
    DROPBITS(state->extra);
}
if (state->offset > state->wsize - (state->whave < state->wsize ?
                                left : 0)) {
    strm->msg = (char *) "invalid distance too far back";
    state->mode = BAD;
    break;
}
Tracevv((stderr, "inflate:    distance %u\n", state->offset));

/* copy match from window to output */
do {
    ROOM();
    copy = state->wsize - state->offset;
    if (copy < left) {
        from = put + copy;
        copy = left - copy;
    }
    else {
        from = put - state->offset;
        copy = left;
    }
    if (copy > state->length) copy = state->length;
    state->length -= copy;
    left -= copy;
    do {
        *put++ = *from++;
    } while (--copy);
} while (state->length != 0);
break;

case DONE:
    /* inflate stream terminated properly -- write leftover output */
    ret = Z_STREAM_END;
    if (left < state->wsize) {

```

```
        if (out(out_desc, state->window, state->wsize - left))
            ret = Z_BUF_ERROR;
    }
    goto inf_leave;

case BAD:
    ret = Z_DATA_ERROR;
    goto inf_leave;

default:
    /* can't happen, but makes compilers happy */
    ret = Z_STREAM_ERROR;
    goto inf_leave;
}

/* Return unused input */
inf_leave:
    strm->next_in = next;
    strm->avail_in = have;
    return ret;
}

int ZEXPORT inflateBackEnd(strm)
z_stream * strm;
{
    if (strm == Z_NULL || strm->state == Z_NULL || strm->zfree == (free_func)0)
        return Z_STREAM_ERROR;
    ZFREE(strm, strm->state);
    strm->state = Z_NULL;
    Tracev((stderr, "inflate: end\n"));
    return Z_OK;
}
```

```

/* inffast.c -- fast decoding
 * Copyright (C) 1995-2004 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"

#ifdef ASMINF

/* Allow machine dependent optimization for post-increment or pre-increment.
Based on testing to date,
Pre-increment preferred for:
- PowerPC G3 (Adler)
- MIPS R5000 (Randers-Pehrson)
Post-increment preferred for:
- none
No measurable difference:
- Pentium III (Anderson)
- M68060 (Nikl)
*/
#endif

#ifdef POSTINC
# define OFF 0
# define PUP(a) *(a)++
#else
# define OFF 1
# define PUP(a) *++(a)
#endif

/*
Decode literal, length, and distance codes and write out the resulting
literal and match bytes until either not enough input or output is
available, an end-of-block is encountered, or a data error is encountered.
When large enough input and output buffers are supplied to inflate(), for
example, a 16K input buffer and a 64K output buffer, more than 95% of the
inflate execution time is spent in this routine.

Entry assumptions:
    state->mode == LEN
    strm->avail_in >= 6
    strm->avail_out >= 258
    start >= strm->avail_out
    state->bits < 8

On return, state->mode is one of:
    LEN -- ran out of enough output space or enough available input
    TYPE -- reached end of block code, inflate() to interpret next block
    BAD -- error in block data

Notes:
- The maximum input bits used by a length/distance pair is 15 bits for the
length code, 5 bits for the length extra, 15 bits for the distance code,
and 13 bits for the distance extra. This totals 48 bits, or six bytes.
Therefore if strm->avail_in >= 6, then there is enough input to avoid
checking for available input while decoding.
- The maximum bytes that a single length/distance pair can output is 258
bytes, which is the maximum length that can be coded. inflate_fast()
requires strm->avail_out >= 258 for each loop to avoid checking for
output space.
*/
void inflate_fast(strm, start)
z_streamp strm;
unsigned start;          /* inflate()'s starting value for strm->avail_out */
{
    struct inflate_state FAR *state;
    unsigned char FAR *in;          /* local strm->next_in */
    unsigned char FAR *last;        /* while in < last, enough input available */
    unsigned char FAR *out;         /* local strm->next_out */

```



```

    unsigned char FAR *beg;      /* inflate()'s initial strm->next_out */
    unsigned char FAR *end;      /* while out < end, enough space available */
#ifdef INFLATE_STRICT
    unsigned dmax;               /* maximum distance from zlib header */
#endif
    unsigned wsize;              /* window size or zero if not using window */
    unsigned whave;              /* valid bytes in the window */
    unsigned write;              /* window write index */
    unsigned char FAR *window;   /* allocated sliding window, if wsize != 0 */
    unsigned long hold;          /* local strm->hold */
    unsigned bits;               /* local strm->bits */
    code const FAR *lcode;       /* local strm->lencode */
    code const FAR *dcode;       /* local strm->distcode */
    unsigned lmask;              /* mask for first level of length codes */
    unsigned dmask;              /* mask for first level of distance codes */
    code this;                   /* retrieved table entry */
    unsigned op;                 /* code bits, operation, extra bits, or */
                                /* window position, window bytes to copy */
    unsigned len;                /* match length, unused bytes */
    unsigned dist;               /* match distance */
    unsigned char FAR *from;     /* where to copy match from */

    /* copy state to local variables */
    state = (struct inflate_state FAR *)strm->state;
    in = strm->next_in - OFF;
    last = in + (strm->avail_in - 5);
    out = strm->next_out - OFF;
    beg = out - (start - strm->avail_out);
    end = out + (strm->avail_out - 257);
#ifdef INFLATE_STRICT
    dmax = state->dmax;
#endif
    wsize = state->wsize;
    whave = state->whave;
    write = state->write;
    window = state->window;
    hold = state->hold;
    bits = state->bits;
    lcode = state->lencode;
    dcode = state->distcode;
    lmask = (1U << state->lenbits) - 1;
    dmask = (1U << state->distbits) - 1;

    /* decode literals and length/distances until end-of-block or not enough
    input data or output space */
    do {
        if (bits < 15) {
            hold += (unsigned long)(PUP(in)) << bits;
            bits += 8;
            hold += (unsigned long)(PUP(in)) << bits;
            bits += 8;
        }
        this = lcode[hold & lmask];
dolen:
        op = (unsigned)(this.bits);
        hold >>= op;
        bits -= op;
        op = (unsigned)(this.op);
        if (op == 0) { /* literal */
            Tracevv((stderr, this.val >= 0x20 && this.val < 0x7f ?
                "inflate:   literal '%c'\n" :
                "inflate:   literal 0x%02x\n", this.val));
            PUP(out) = (unsigned char)(this.val);
        }
        else if (op & 16) { /* length base */
            len = (unsigned)(this.val);
            op &= 15; /* number of extra bits */
            if (op) {
                if (bits < op) {
                    hold += (unsigned long)(PUP(in)) << bits;
                    bits += 8;
                }
                len += (unsigned)hold & ((1U << op) - 1);
                hold >>= op;
            }

```

```

        bits -= op;
    }
    Tracevv((stderr, "inflate:   length %u\n", len));
    if (bits < 15) {
        hold += (unsigned long)(PUP(in)) << bits;
        bits += 8;
        hold += (unsigned long)(PUP(in)) << bits;
        bits += 8;
    }
    this = dcode[hold & dmask];
dodist:
    op = (unsigned)(this.bits);
    hold >>= op;
    bits -= op;
    op = (unsigned)(this.op);
    if (op & 16) { /* distance base */
        dist = (unsigned)(this.val);
        op &= 15; /* number of extra bits */
        if (bits < op) {
            hold += (unsigned long)(PUP(in)) << bits;
            bits += 8;
            if (bits < op) {
                hold += (unsigned long)(PUP(in)) << bits;
                bits += 8;
            }
        }
        dist += (unsigned)hold & ((1U << op) - 1);
#ifdef INFLATE_STRICT
        if (dist > dmax) {
            strm->msg = (char *)"invalid distance too far back";
            state->mode = BAD;
            break;
        }
#endif
    }

    hold >>= op;
    bits -= op;
    Tracevv((stderr, "inflate:   distance %u\n", dist));
    op = (unsigned)(out - beg); /* max distance in output */
    if (dist > op) { /* see if copy from window */
        op = dist - op; /* distance back in window */
        if (op > whave) {
            strm->msg = (char *)"invalid distance too far back";
            state->mode = BAD;
            break;
        }
        from = window - OFF;
        if (write == 0) { /* very common case */
            from += wsize - op;
            if (op < len) { /* some from window */
                len -= op;
                do {
                    PUP(out) = PUP(from);
                } while (--op);
                from = out - dist; /* rest from output */
            }
        }
        else if (write < op) { /* wrap around window */
            from += wsize + write - op;
            op -= write;
            if (op < len) { /* some from end of window */
                len -= op;
                do {
                    PUP(out) = PUP(from);
                } while (--op);
                from = window - OFF;
                if (write < len) { /* some from start of window */
                    op = write;
                    len -= op;
                    do {
                        PUP(out) = PUP(from);
                    } while (--op);
                    from = out - dist; /* rest from output */
                }
            }
        }
    }
}

```

```

    }
    else {
        from += write - op;
        if (op < len) {
            len -= op;
            do {
                PUP(out) = PUP(from);
            } while (--op);
            from = out - dist; /* rest from output */
        }
    }
    while (len > 2) {
        PUP(out) = PUP(from);
        PUP(out) = PUP(from);
        PUP(out) = PUP(from);
        len -= 3;
    }
    if (len) {
        PUP(out) = PUP(from);
        if (len > 1)
            PUP(out) = PUP(from);
    }
}
else {
    from = out - dist;
    do {
        PUP(out) = PUP(from);
        PUP(out) = PUP(from);
        PUP(out) = PUP(from);
        len -= 3;
    } while (len > 2);
    if (len) {
        PUP(out) = PUP(from);
        if (len > 1)
            PUP(out) = PUP(from);
    }
}

}
else if ((op & 64) == 0) {
    this = dcode[this.val + (hold & ((1U << op) - 1))];
    goto dodist;
}
else {
    strm->msg = (char *) "invalid distance code";
    state->mode = BAD;
    break;
}
}
else if ((op & 64) == 0) {
    this = lcode[this.val + (hold & ((1U << op) - 1))];
    goto dolen;
}
else if (op & 32) {
    Tracevv((stderr, "inflate: end of block\n"));
    state->mode = TYPE;
    break;
}
else {
    strm->msg = (char *) "invalid literal/length code";
    state->mode = BAD;
    break;
}
}
while (in < last && out < end);

/* return unused bytes (on entry, bits < 8, so in won't go too far back) */
len = bits >> 3;
in -= len;
bits -= len << 3;
hold &= (1U << bits) - 1;

/* update state and return */
strm->next_in = in + OFF;
strm->next_out = out + OFF;
strm->avail_in = (unsigned)(in < last ? 5 + (last - in) : 5 - (in - last));

```

```
    strm->avail_out = (unsigned)(out < end ?
                                257 + (end - out) : 257 - (out - end));
    state->hold = hold;
    state->bits = bits;
    return;
}

/*
inflate_fast() speedups that turned out slower (on a PowerPC G3 750CXe):
- Using bit fields for code structure
- Different op definition to avoid & for extra bits (do & for table bits)
- Three separate decoding do-loops for direct, window, and write == 0
- Special case for distance > 1 copies to do overlapped load and store copy
- Explicit branch predictions (based on measured branch probabilities)
- Deferring match copy and interspersed it with decoding subsequent codes
- Swapping literal/length else
- Swapping window/direct else
- Larger unrolled copy loops (three is about right)
- Moving len -= 3 statement into middle of loop
*/

#endif /* !ASMINF */
```

```
/* inffast.h -- header to use inffast.c
 * Copyright (C) 1995-2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

void inflate_fast OF((z_streamp strm, unsigned start));
```

```
/* inffixed.h -- table for decoding fixed codes
 * Generated automatically by makefixed().
 */
```

```
/* WARNING: this file should *not* be used by applications. It
 * is part of the implementation of the compression library and
 * is subject to change. Applications should only use zlib.h.
 */
```

```
static const code lenfix[512] = {
    {96,7,0},{0,8,80},{0,8,16},{20,8,115},{18,7,31},{0,8,112},{0,8,48},
    {0,9,192},{16,7,10},{0,8,96},{0,8,32},{0,9,160},{0,8,0},{0,8,128},
    {0,8,64},{0,9,224},{16,7,6},{0,8,88},{0,8,24},{0,9,144},{19,7,59},
    {0,8,120},{0,8,56},{0,9,208},{17,7,17},{0,8,104},{0,8,40},{0,9,176},
    {0,8,8},{0,8,136},{0,8,72},{0,9,240},{16,7,4},{0,8,84},{0,8,20},
    {21,8,227},{19,7,43},{0,8,116},{0,8,52},{0,9,200},{17,7,13},{0,8,100},
    {0,8,36},{0,9,168},{0,8,4},{0,8,132},{0,8,68},{0,9,232},{16,7,8},
    {0,8,92},{0,8,28},{0,9,152},{20,7,83},{0,8,124},{0,8,60},{0,9,216},
    {18,7,23},{0,8,108},{0,8,44},{0,9,184},{0,8,12},{0,8,140},{0,8,76},
    {0,9,248},{16,7,3},{0,8,82},{0,8,18},{21,8,163},{19,7,35},{0,8,114},
    {0,8,50},{0,9,196},{17,7,11},{0,8,98},{0,8,34},{0,9,164},{0,8,2},
    {0,8,130},{0,8,66},{0,9,228},{16,7,7},{0,8,90},{0,8,26},{0,9,148},
    {20,7,67},{0,8,122},{0,8,58},{0,9,212},{18,7,19},{0,8,106},{0,8,42},
    {0,9,180},{0,8,10},{0,8,138},{0,8,74},{0,9,244},{16,7,5},{0,8,86},
    {0,8,22},{64,8,0},{19,7,51},{0,8,118},{0,8,54},{0,9,204},{17,7,15},
    {0,8,102},{0,8,38},{0,9,172},{0,8,6},{0,8,134},{0,8,70},{0,9,236},
    {16,7,9},{0,8,94},{0,8,30},{0,9,156},{20,7,99},{0,8,126},{0,8,62},
    {0,9,220},{18,7,27},{0,8,110},{0,8,46},{0,9,188},{0,8,14},{0,8,142},
    {0,8,78},{0,9,252},{96,7,0},{0,8,81},{0,8,17},{21,8,131},{18,7,31},
    {0,8,113},{0,8,49},{0,9,194},{16,7,10},{0,8,97},{0,8,33},{0,9,162},
    {0,8,1},{0,8,129},{0,8,65},{0,9,226},{16,7,6},{0,8,89},{0,8,25},
    {0,9,146},{19,7,59},{0,8,121},{0,8,57},{0,9,210},{17,7,17},{0,8,105},
    {0,8,41},{0,9,178},{0,8,9},{0,8,137},{0,8,73},{0,9,242},{16,7,4},
    {0,8,85},{0,8,21},{16,8,258},{19,7,43},{0,8,117},{0,8,53},{0,9,202},
    {17,7,13},{0,8,101},{0,8,37},{0,9,170},{0,8,5},{0,8,133},{0,8,69},
    {0,9,234},{16,7,8},{0,8,93},{0,8,29},{0,9,154},{20,7,83},{0,8,125},
    {0,8,61},{0,9,218},{18,7,23},{0,8,109},{0,8,45},{0,9,186},{0,8,13},
    {0,8,141},{0,8,77},{0,9,250},{16,7,3},{0,8,83},{0,8,19},{21,8,195},
    {19,7,35},{0,8,115},{0,8,51},{0,9,198},{17,7,11},{0,8,99},{0,8,35},
    {0,9,166},{0,8,3},{0,8,131},{0,8,67},{0,9,230},{16,7,7},{0,8,91},
    {0,8,27},{0,9,150},{20,7,67},{0,8,123},{0,8,59},{0,9,214},{18,7,19},
    {0,8,107},{0,8,43},{0,9,182},{0,8,11},{0,8,139},{0,8,75},{0,9,246},
    {16,7,5},{0,8,87},{0,8,23},{64,8,0},{19,7,51},{0,8,119},{0,8,55},
    {0,9,206},{17,7,15},{0,8,103},{0,8,39},{0,9,174},{0,8,7},{0,8,135},
    {0,8,71},{0,9,238},{16,7,9},{0,8,95},{0,8,31},{0,9,158},{20,7,99},
    {0,8,127},{0,8,63},{0,9,222},{18,7,27},{0,8,111},{0,8,47},{0,9,190},
    {0,8,15},{0,8,143},{0,8,79},{0,9,254},{96,7,0},{0,8,80},{0,8,16},
    {20,8,115},{18,7,31},{0,8,112},{0,8,48},{0,9,193},{16,7,10},{0,8,96},
    {0,8,32},{0,9,161},{0,8,0},{0,8,128},{0,8,64},{0,9,225},{16,7,6},
    {0,8,88},{0,8,24},{0,9,145},{19,7,59},{0,8,120},{0,8,56},{0,9,209},
    {17,7,17},{0,8,104},{0,8,40},{0,9,177},{0,8,8},{0,8,136},{0,8,72},
    {0,9,241},{16,7,4},{0,8,84},{0,8,20},{21,8,227},{19,7,43},{0,8,116},
    {0,8,52},{0,9,201},{17,7,13},{0,8,100},{0,8,36},{0,9,169},{0,8,4},
    {0,8,132},{0,8,68},{0,9,233},{16,7,8},{0,8,92},{0,8,28},{0,9,153},
    {20,7,83},{0,8,124},{0,8,60},{0,9,217},{18,7,23},{0,8,108},{0,8,44},
    {0,9,185},{0,8,12},{0,8,140},{0,8,76},{0,9,249},{16,7,3},{0,8,82},
    {0,8,18},{21,8,163},{19,7,35},{0,8,114},{0,8,50},{0,9,197},{17,7,11},
    {0,8,98},{0,8,34},{0,9,165},{0,8,2},{0,8,130},{0,8,66},{0,9,229},
    {16,7,7},{0,8,90},{0,8,26},{0,9,149},{20,7,67},{0,8,122},{0,8,58},
    {0,9,213},{18,7,19},{0,8,106},{0,8,42},{0,9,181},{0,8,10},{0,8,138},
    {0,8,74},{0,9,245},{16,7,5},{0,8,86},{0,8,22},{64,8,0},{19,7,51},
    {0,8,118},{0,8,54},{0,9,205},{17,7,15},{0,8,102},{0,8,38},{0,9,173},
    {0,8,6},{0,8,134},{0,8,70},{0,9,237},{16,7,9},{0,8,94},{0,8,30},
    {0,9,157},{20,7,99},{0,8,126},{0,8,62},{0,9,221},{18,7,27},{0,8,110},
    {0,8,46},{0,9,189},{0,8,14},{0,8,142},{0,8,78},{0,9,253},{96,7,0},
    {0,8,81},{0,8,17},{21,8,131},{18,7,31},{0,8,113},{0,8,49},{0,9,195},
    {16,7,10},{0,8,97},{0,8,33},{0,9,163},{0,8,1},{0,8,129},{0,8,65},
    {0,9,227},{16,7,6},{0,8,89},{0,8,25},{0,9,147},{19,7,59},{0,8,121},
    {0,8,57},{0,9,211},{17,7,17},{0,8,105},{0,8,41},{0,9,179},{0,8,9},
    {0,8,137},{0,8,73},{0,9,243},{16,7,4},{0,8,85},{0,8,21},{16,8,258},
    {19,7,43},{0,8,117},{0,8,53},{0,9,203},{17,7,13},{0,8,101},{0,8,37},
    {0,9,171},{0,8,5},{0,8,133},{0,8,69},{0,9,235},{16,7,8},{0,8,93},
    {0,8,29},{0,9,155},{20,7,83},{0,8,125},{0,8,61},{0,9,219},{18,7,23},
    {0,8,109},{0,8,45},{0,9,187},{0,8,13},{0,8,141},{0,8,77},{0,9,251},

```

```
{16,7,3},{0,8,83},{0,8,19},{21,8,195},{19,7,35},{0,8,115},{0,8,51},
{0,9,199},{17,7,11},{0,8,99},{0,8,35},{0,9,167},{0,8,3},{0,8,131},
{0,8,67},{0,9,231},{16,7,7},{0,8,91},{0,8,27},{0,9,151},{20,7,67},
{0,8,123},{0,8,59},{0,9,215},{18,7,19},{0,8,107},{0,8,43},{0,9,183},
{0,8,11},{0,8,139},{0,8,75},{0,9,247},{16,7,5},{0,8,87},{0,8,23},
{64,8,0},{19,7,51},{0,8,119},{0,8,55},{0,9,207},{17,7,15},{0,8,103},
{0,8,39},{0,9,175},{0,8,7},{0,8,135},{0,8,71},{0,9,239},{16,7,9},
{0,8,95},{0,8,31},{0,9,159},{20,7,99},{0,8,127},{0,8,63},{0,9,223},
{18,7,27},{0,8,111},{0,8,47},{0,9,191},{0,8,15},{0,8,143},{0,8,79},
{0,9,255}
};

static const code distfix[32] = {
{16,5,1},{23,5,257},{19,5,17},{27,5,4097},{17,5,5},{25,5,1025},
{21,5,65},{29,5,16385},{16,5,3},{24,5,513},{20,5,33},{28,5,8193},
{18,5,9},{26,5,2049},{22,5,129},{64,5,0},{16,5,2},{23,5,385},
{19,5,25},{27,5,6145},{17,5,7},{25,5,1537},{21,5,97},{29,5,24577},
{16,5,4},{24,5,769},{20,5,49},{28,5,12289},{18,5,13},{26,5,3073},
{22,5,193},{64,5,0}
};
```

```

/* inflate.c -- zlib decompression
 * Copyright (C) 1995-2005 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 * Change history:
 *
 * 1.2.beta0    24 Nov 2002
 * - First version -- complete rewrite of inflate to simplify code, avoid
 *   creation of window when not needed, minimize use of window when it is
 *   needed, make inffast.c even faster, implement gzip decoding, and to
 *   improve code readability and style over the previous zlib inflate code
 *
 * 1.2.beta1    25 Nov 2002
 * - Use pointers for available input and output checking in inffast.c
 * - Remove input and output counters in inffast.c
 * - Change inffast.c entry and loop from avail_in >= 7 to >= 6
 * - Remove unnecessary second byte pull from length extra in inffast.c
 * - Unroll direct copy to three copies per loop in inffast.c
 *
 * 1.2.beta2    4 Dec 2002
 * - Change external routine names to reduce potential conflicts
 * - Correct filename to inffixed.h for fixed tables in inflate.c
 * - Make hbuf[] unsigned char to match parameter type in inflate.c
 * - Change strm->next_out[-state->offset] to *(strm->next_out - state->offset)
 *   to avoid negation problem on Alphas (64 bit) in inflate.c
 *
 * 1.2.beta3    22 Dec 2002
 * - Add comments on state->bits assertion in inffast.c
 * - Add comments on op field in inftrees.h
 * - Fix bug in reuse of allocated window after inflateReset()
 * - Remove bit fields--back to byte structure for speed
 * - Remove distance extra == 0 check in inflate_fast()--only helps for lengths
 * - Change post-increments to pre-increments in inflate_fast(), PPC biased?
 * - Add compile time option, POSTINC, to use post-increments instead (Intel?)
 * - Make MATCH copy in inflate() much faster for when inflate_fast() not used
 * - Use local copies of stream next and avail values, as well as local bit
 *   buffer and bit count in inflate()--for speed when inflate_fast() not used
 *
 * 1.2.beta4    1 Jan 2003
 * - Split ptr - 257 statements in inflate_table() to avoid compiler warnings
 * - Move a comment on output buffer sizes from inffast.c to inflate.c
 * - Add comments in inffast.c to introduce the inflate_fast() routine
 * - Rearrange window copies in inflate_fast() for speed and simplification
 * - Unroll last copy for window match in inflate_fast()
 * - Use local copies of window variables in inflate_fast() for speed
 * - Pull out common write == 0 case for speed in inflate_fast()
 * - Make op and len in inflate_fast() unsigned for consistency
 * - Add FAR to lcode and dcode declarations in inflate_fast()
 * - Simplified bad distance check in inflate_fast()
 * - Added inflateBackInit(), inflateBack(), and inflateBackEnd() in new
 *   source file inffback.c to provide a call-back interface to inflate for
 *   programs like gzip and unzip -- uses window as output buffer to avoid
 *   window copying
 *
 * 1.2.beta5    1 Jan 2003
 * - Improved inflateBack() interface to allow the caller to provide initial
 *   input in strm.
 * - Fixed stored blocks bug in inflateBack()
 *
 * 1.2.beta6    4 Jan 2003
 * - Added comments in inffast.c on effectiveness of POSTINC
 * - Typecasting all around to reduce compiler warnings
 * - Changed loops from while (1) or do {} while (1) to for (;;) again to
 *   make compilers happy
 * - Changed type of window in inflateBackInit() to unsigned char *
 *
 * 1.2.beta7    27 Jan 2003
 * - Changed many types to unsigned or unsigned short to avoid warnings
 * - Added inflateCopy() function
 *
 * 1.2.0        9 Mar 2003
 * - Changed inflateBack() interface to provide separate opaque descriptors

```



```

*   for the in() and out() functions
* - Changed inflateBack() argument and in_func typedef to swap the length
*   and buffer address return values for the input function
* - Check next_in and next_out for Z_NULL on entry to inflate()
*
* The history for versions after 1.2.0 are in ChangeLog in zlib distribution.
*/

#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"

#ifdef MAKEFIXED
# ifndef BUILDFIXED
#   define BUILDFIXED
# endif
#endif

/* function prototypes */
local void fixedtables OF((struct inflate_state FAR *state));
local int updatewindow OF((z_stream strm, unsigned out));
#ifdef BUILDFIXED
    void makefixed OF((void));
#endif
local unsigned syncsearch OF((unsigned FAR *have, unsigned char FAR *buf,
                             unsigned len));

int ZEXPORT inflateReset(strm)
z_stream strm;
{
    struct inflate_state FAR *state;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;
    strm->total_in = strm->total_out = state->total = 0;
    strm->msg = Z_NULL;
    strm->adler = 1;          /* to support ill-conceived Java test suite */
    state->mode = HEAD;
    state->last = 0;
    state->havedict = 0;
    state->dmax = 32768U;
    state->head = Z_NULL;
    state->wsize = 0;
    state->whave = 0;
    state->write = 0;
    state->hold = 0;
    state->bits = 0;
    state->lencode = state->distcode = state->next = state->codes;
    Tracev((stderr, "inflate: reset\n"));
    return Z_OK;
}

int ZEXPORT inflatePrime(strm, bits, value)
z_stream strm;
int bits;
int value;
{
    struct inflate_state FAR *state;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;
    if (bits > 16 || state->bits + bits > 32) return Z_STREAM_ERROR;
    value &= (1L << bits) - 1;
    state->hold += value << state->bits;
    state->bits += bits;
    return Z_OK;
}

int ZEXPORT inflateInit2_(strm, windowBits, version, stream_size)
z_stream strm;
int windowBits;
const char *version;
int stream_size;

```

```

{
    struct inflate_state FAR *state;

    if (version == Z_NULL || version[0] != ZLIB_VERSION[0] ||
        stream_size != (int)(sizeof(z_stream)))
        return Z_VERSION_ERROR;
    if (strm == Z_NULL) return Z_STREAM_ERROR;
    strm->msg = Z_NULL; /* in case we return an error */
    if (strm->zalloc == (alloc_func)0) {
        strm->zalloc = zcalloc;
        strm->opaque = (voidpf)0;
    }
    if (strm->zfree == (free_func)0) strm->zfree = zcfree;
    state = (struct inflate_state FAR *)
        ZALLOC(strm, 1, sizeof(struct inflate_state));
    if (state == Z_NULL) return Z_MEM_ERROR;
    Tracev((stderr, "inflate: allocated\n"));
    strm->state = (struct internal_state FAR *)state;
    if (windowBits < 0) {
        state->wrap = 0;
        windowBits = -windowBits;
    }
    else {
        state->wrap = (windowBits >> 4) + 1;
#ifdef GUNZIP
        if (windowBits < 48) windowBits &= 15;
#endif
    }
    if (windowBits < 8 || windowBits > 15) {
        ZFREE(strm, state);
        strm->state = Z_NULL;
        return Z_STREAM_ERROR;
    }
    state->wbits = (unsigned)windowBits;
    state->window = Z_NULL;
    return inflateReset(strm);
}

int ZEXPORT inflateInit_(strm, version, stream_size)
z_streamp strm;
const char *version;
int stream_size;
{
    return inflateInit2_(strm, DEF_WBITS, version, stream_size);
}

/*
   Return state with length and distance decoding tables and index sizes set to
   fixed code decoding. Normally this returns fixed tables from inffixed.h.
   If BUILDFIXED is defined, then instead this routine builds the tables the
   first time it's called, and returns those tables the first time and
   thereafter. This reduces the size of the code by about 2K bytes, in
   exchange for a little execution time. However, BUILDFIXED should not be
   used for threaded applications, since the rewriting of the tables and virgin
   may not be thread-safe.
*/
local void fixedtables(state)
struct inflate_state FAR *state;
{
#ifdef BUILDFIXED
    static int virgin = 1;
    static code *lenfix, *distfix;
    static code fixed[544];

    /* build fixed huffman tables if first call (may not be thread safe) */
    if (virgin) {
        unsigned sym, bits;
        static code *next;

        /* literal/length table */
        sym = 0;
        while (sym < 144) state->lens[sym++] = 8;
        while (sym < 256) state->lens[sym++] = 9;
        while (sym < 280) state->lens[sym++] = 7;

```

```

    while (sym < 288) state->lens[sym++] = 8;
    next = fixed;
    lenfix = next;
    bits = 9;
    inflate_table(LENS, state->lens, 288, &(next), &(bits), state->work);

    /* distance table */
    sym = 0;
    while (sym < 32) state->lens[sym++] = 5;
    distfix = next;
    bits = 5;
    inflate_table(DISTS, state->lens, 32, &(next), &(bits), state->work);

    /* do this just once */
    virgin = 0;
}
#else /* !BUILDFIXED */
#   include "inffixed.h"
#endif /* BUILDFIXED */
    state->lencode = lenfix;
    state->lenbits = 9;
    state->distcode = distfix;
    state->distbits = 5;
}

#ifdef MAKEFIXED
#include <stdio.h>

/*
Write out the inffixed.h that is #include'd above.  Defining MAKEFIXED also
defines BUILDFIXED, so the tables are built on the fly.  makefixed() writes
those tables to stdout, which would be piped to inffixed.h.  A small program
can simply call makefixed to do this:

    void makefixed(void);

    int main(void)
    {
        makefixed();
        return 0;
    }

Then that can be linked with zlib built with MAKEFIXED defined and run:

    a.out > inffixed.h
*/
void makefixed()
{
    unsigned low, size;
    struct inflate_state state;

    fixedtables(&state);
    puts(" /* inffixed.h -- table for decoding fixed codes */");
    puts("  * Generated automatically by makefixed().");
    puts("  */");
    puts("");
    puts(" /* WARNING: this file should *not* be used by applications. */");
    puts("    It is part of the implementation of this library and is");
    puts("    subject to change. Applications should only use zlib.h.");
    puts("  */");
    puts("");
    size = 1U << 9;
    printf("    static const code lenfix[%u] = {", size);
    low = 0;
    for (;;) {
        if ((low % 7) == 0) printf("\n        ");
        printf("{%u,%u,%d}", state.lencode[low].op, state.lencode[low].bits,
            state.lencode[low].val);
        if (++low == size) break;
        putchar(',');
    }
    puts("\n    };");
    size = 1U << 5;
    printf("\n    static const code distfix[%u] = {", size);

```

```

low = 0;
for (;;) {
    if ((low % 6) == 0) printf("\n    ");
    printf("{%u,%u,%d}", state.distcode[low].op, state.distcode[low].bits,
           state.distcode[low].val);
    if (++low == size) break;
    putchar(',');
}
puts("\n    ");
}
#endif /* MAKEFIXED */

/*
Update the window with the last wsize (normally 32K) bytes written before
returning.  If window does not exist yet, create it.  This is only called
when a window is already in use, or when output has been written during this
inflate call, but the end of the deflate stream has not been reached yet.
It is also called to create a window for dictionary data when a dictionary
is loaded.

Providing output buffers larger than 32K to inflate() should provide a speed
advantage, since only the last 32K of output is copied to the sliding window
upon return from inflate(), and since all distances after the first 32K of
output will fall in the output data, making match copies simpler and faster.
The advantage may be dependent on the size of the processor's data caches.
*/
local int updatewindow(strm, out)
z_streamp strm;
unsigned out;
{
    struct inflate_state FAR *state;
    unsigned copy, dist;

    state = (struct inflate_state FAR *)strm->state;

    /* if it hasn't been done already, allocate space for the window */
    if (state->window == Z_NULL) {
        state->window = (unsigned char FAR *)
            ZALLOC(strm, 1U << state->wbits,
                   sizeof(unsigned char));
        if (state->window == Z_NULL) return 1;
    }

    /* if window not in use yet, initialize */
    if (state->wsize == 0) {
        state->wsize = 1U << state->wbits;
        state->write = 0;
        state->whave = 0;
    }

    /* copy state->wsize or less output bytes into the circular window */
    copy = out - strm->avail_out;
    if (copy >= state->wsize) {
        zmemcpy(state->window, strm->next_out - state->wsize, state->wsize);
        state->write = 0;
        state->whave = state->wsize;
    }
    else {
        dist = state->wsize - state->write;
        if (dist > copy) dist = copy;
        zmemcpy(state->window + state->write, strm->next_out - copy, dist);
        copy -= dist;
        if (copy) {
            zmemcpy(state->window, strm->next_out - copy, copy);
            state->write = copy;
            state->whave = state->wsize;
        }
        else {
            state->write += dist;
            if (state->write == state->wsize) state->write = 0;
            if (state->whave < state->wsize) state->whave += dist;
        }
    }
}
return 0;

```

```

}

/* Macros for inflate(): */

/* check function to use Adler32() for zlib or CRC32() for gzip */
#ifndef GUNZIP
# define UPDATE(check, buf, len) \
    (state->flags ? CRC32(check, buf, len) : Adler32(check, buf, len))
#else
# define UPDATE(check, buf, len) Adler32(check, buf, len)
#endif

/* check macros for header CRC */
#ifndef GUNZIP
# define CRC2(check, word) \
    do { \
        hbuf[0] = (unsigned char)(word); \
        hbuf[1] = (unsigned char)((word) >> 8); \
        check = CRC32(check, hbuf, 2); \
    } while (0)
# define CRC4(check, word) \
    do { \
        hbuf[0] = (unsigned char)(word); \
        hbuf[1] = (unsigned char)((word) >> 8); \
        hbuf[2] = (unsigned char)((word) >> 16); \
        hbuf[3] = (unsigned char)((word) >> 24); \
        check = CRC32(check, hbuf, 4); \
    } while (0)
#endif

/* Load registers with state in inflate() for speed */
#define LOAD() \
    do { \
        put = strm->next_out; \
        left = strm->avail_out; \
        next = strm->next_in; \
        have = strm->avail_in; \
        hold = state->hold; \
        bits = state->bits; \
    } while (0)

/* Restore state from registers in inflate() */
#define RESTORE() \
    do { \
        strm->next_out = put; \
        strm->avail_out = left; \
        strm->next_in = next; \
        strm->avail_in = have; \
        state->hold = hold; \
        state->bits = bits; \
    } while (0)

/* Clear the input bit accumulator */
#define INITBITS() \
    do { \
        hold = 0; \
        bits = 0; \
    } while (0)

/* Get a byte of input into the bit accumulator, or return from inflate()
   if there is no input available. */
#define PULLBYTE() \
    do { \
        if (have == 0) goto inf_leave; \
        have--; \
        hold += (unsigned long)(*next++) << bits; \
        bits += 8; \
    } while (0)

/* Assure that there are at least n bits in the bit accumulator.  If there is
   not enough available input to do that, then return from inflate(). */
#define NEEDBITS(n) \
    do { \

```

```

        while (bits < (unsigned)(n)) \
            PULLBYTE(); \
    } while (0)

/* Return the low n bits of the bit accumulator (n < 16) */
#define BITS(n) \
    ((unsigned)hold & ((1U << (n)) - 1))

/* Remove n bits from the bit accumulator */
#define DROPBITS(n) \
    do { \
        hold >>= (n); \
        bits -= (unsigned)(n); \
    } while (0)

/* Remove zero to seven bits as needed to go to a byte boundary */
#define BYTEBITS() \
    do { \
        hold >>= bits & 7; \
        bits -= bits & 7; \
    } while (0)

/* Reverse the bytes in a 32-bit value */
#define REVERSE(q) \
    (((q) >> 24) & 0xff) + (((q) >> 8) & 0xff00) + \
    (((q) & 0xff00) << 8) + (((q) & 0xff) << 24))

/*
inflate() uses a state machine to process as much input data and generate as
much output data as possible before returning. The state machine is
structured roughly as follows:

    for (;;) switch (state) {
    ...
    case STATEn:
        if (not enough input data or output space to make progress)
            return;
        ... make progress ...
        state = STATEm;
        break;
    ...
    }

so when inflate() is called again, the same case is attempted again, and
if the appropriate resources are provided, the machine proceeds to the
next state. The NEEDBITS() macro is usually the way the state evaluates
whether it can proceed or should return. NEEDBITS() does the return if
the requested bits are not available. The typical use of the BITS macros
is:

    NEEDBITS(n);
    ... do something with BITS(n) ...
    DROPBITS(n);

where NEEDBITS(n) either returns from inflate() if there isn't enough
input left to load n bits into the accumulator, or it continues. BITS(n)
gives the low n bits in the accumulator. When done, DROPBITS(n) drops
the low n bits off the accumulator. INITBITS() clears the accumulator
and sets the number of available bits to zero. BYTEBITS() discards just
enough bits to put the accumulator on a byte boundary. After BYTEBITS()
and a NEEDBITS(8), then BITS(8) would return the next byte in the stream.

NEEDBITS(n) uses PULLBYTE() to get an available byte of input, or to return
if there is no input available. The decoding of variable length codes uses
PULLBYTE() directly in order to pull just enough bytes to decode the next
code, and no more.

Some states loop until they get enough input, making sure that enough
state information is maintained to continue the loop where it left off
if NEEDBITS() returns in the loop. For example, want, need, and keep
would all have to actually be part of the saved state in case NEEDBITS()
returns:

    case STATEw:

```

```

    while (want < need) {
        NEEDBITS(n);
        keep[want++] = BITS(n);
        DROPBITS(n);
    }
    state = STATEx;
case STATEx:

```

As shown above, if the next state is also the next case, then the break is omitted.

A state may also return if there is not enough output space available to complete that state. Those states are copying stored data, writing a literal byte, and copying a matching string.

When returning, a "goto inf_leave" is used to update the total counters, update the check value, and determine whether any progress has been made during that inflate() call in order to return the proper return code. Progress is defined as a change in either strm->avail_in or strm->avail_out. When there is a window, goto inf_leave will update the window with the last output written. If a goto inf_leave occurs in the middle of decompression and there is no window currently, goto inf_leave will create one and copy output to the window for the next call of inflate().

In this implementation, the flush parameter of inflate() only affects the return code (per zlib.h). inflate() always writes as much as possible to strm->next_out, given the space available and the provided input--the effect documented in zlib.h of Z_SYNC_FLUSH. Furthermore, inflate() always defers the allocation of and copying into a sliding window until necessary, which provides the effect documented in zlib.h for Z_FINISH when the entire input stream available. So the only thing the flush parameter actually does is: when flush is set to Z_FINISH, inflate() cannot return Z_OK. Instead it will return Z_BUF_ERROR if it has not reached the end of the stream.

*/

```

int ZEXPORT inflate(strm, flush)
z_streamp strm;
int flush;
{
    struct inflate_state FAR *state;
    unsigned char FAR *next;    /* next input */
    unsigned char FAR *put;     /* next output */
    unsigned have, left;        /* available input and output */
    unsigned long hold;         /* bit buffer */
    unsigned bits;              /* bits in bit buffer */
    unsigned in, out;           /* save starting available input and output */
    unsigned copy;              /* number of stored or match bytes to copy */
    unsigned char FAR *from;    /* where to copy match bytes from */
    code this;                  /* current decoding table entry */
    code last;                  /* parent table entry */
    unsigned len;               /* length to copy for repeats, bits to drop */
    int ret;                    /* return code */
#ifdef GUNZIP
    unsigned char hbuf[4];      /* buffer for gzip header crc calculation */
#endif
    static const unsigned short order[19] = /* permutation of code lengths */
        {16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15};

    if (strm == Z_NULL || strm->state == Z_NULL || strm->next_out == Z_NULL ||
        (strm->next_in == Z_NULL && strm->avail_in != 0))
        return Z_STREAM_ERROR;

    state = (struct inflate_state FAR *)strm->state;
    if (state->mode == TYPE) state->mode = TYPEDO;    /* skip check */
    LOAD();
    in = have;
    out = left;
    ret = Z_OK;
    for (;;)
        switch (state->mode) {
            case HEAD:
                if (state->wrap == 0) {
                    state->mode = TYPEDO;
                    break;

```

```

    }
    NEEDBITS(16);
#ifdef GUNZIP
    if ((state->wrap & 2) && hold == 0x8b1f) { /* gzip header */
        state->check = crc32(0L, Z_NULL, 0);
        CRC2(state->check, hold);
        INITBITS();
        state->mode = FLAGS;
        break;
    }
    state->flags = 0; /* expect zlib header */
    if (state->head != Z_NULL)
        state->head->done = -1;
    if (!(state->wrap & 1) || /* check if zlib header allowed */
#ifdef else
    if (
#endif
        ((BITS(8) << 8) + (hold >> 8)) % 31) {
        strm->msg = (char *) "incorrect header check";
        state->mode = BAD;
        break;
    }
    if (BITS(4) != Z_DEFLATED) {
        strm->msg = (char *) "unknown compression method";
        state->mode = BAD;
        break;
    }
    DROPBITS(4);
    len = BITS(4) + 8;
    if (len > state->wbits) {
        strm->msg = (char *) "invalid window size";
        state->mode = BAD;
        break;
    }
    state->dmax = 1U << len;
    Tracev((stderr, "inflate: zlib header ok\n"));
    strm->adler = state->check = Adler32(0L, Z_NULL, 0);
    state->mode = hold & 0x200 ? DICTID : TYPE;
    INITBITS();
    break;
#ifdef GUNZIP
    case FLAGS:
        NEEDBITS(16);
        state->flags = (int)(hold);
        if ((state->flags & 0xff) != Z_DEFLATED) {
            strm->msg = (char *) "unknown compression method";
            state->mode = BAD;
            break;
        }
        if (state->flags & 0xe000) {
            strm->msg = (char *) "unknown header flags set";
            state->mode = BAD;
            break;
        }
        if (state->head != Z_NULL)
            state->head->text = (int)((hold >> 8) & 1);
        if (state->flags & 0x0200) CRC2(state->check, hold);
        INITBITS();
        state->mode = TIME;
    case TIME:
        NEEDBITS(32);
        if (state->head != Z_NULL)
            state->head->time = hold;
        if (state->flags & 0x0200) CRC4(state->check, hold);
        INITBITS();
        state->mode = OS;
    case OS:
        NEEDBITS(16);
        if (state->head != Z_NULL) {
            state->head->xflags = (int)(hold & 0xff);
            state->head->os = (int)(hold >> 8);
        }
        if (state->flags & 0x0200) CRC2(state->check, hold);
        INITBITS();

```



```

    state->mode = EXLEN;
case EXLEN:
    if (state->flags & 0x0400) {
        NEEDBITS(16);
        state->length = (unsigned)(hold);
        if (state->head != Z_NULL)
            state->head->extra_len = (unsigned)hold;
        if (state->flags & 0x0200) CRC2(state->check, hold);
        INITBITS();
    }
    else if (state->head != Z_NULL)
        state->head->extra = Z_NULL;
    state->mode = EXTRA;
case EXTRA:
    if (state->flags & 0x0400) {
        copy = state->length;
        if (copy > have) copy = have;
        if (copy) {
            if (state->head != Z_NULL &&
                state->head->extra != Z_NULL) {
                len = state->head->extra_len - state->length;
                zmemcpy(state->head->extra + len, next,
                    len + copy > state->head->extra_max ?
                        state->head->extra_max - len : copy);
            }
            if (state->flags & 0x0200)
                state->check = crc32(state->check, next, copy);
            have -= copy;
            next += copy;
            state->length -= copy;
        }
        if (state->length) goto inf_leave;
    }
    state->length = 0;
    state->mode = NAME;
case NAME:
    if (state->flags & 0x0800) {
        if (have == 0) goto inf_leave;
        copy = 0;
        do {
            len = (unsigned)(next[copy++]);
            if (state->head != Z_NULL &&
                state->head->name != Z_NULL &&
                state->length < state->head->name_max)
                state->head->name[state->length++] = len;
        } while (len && copy < have);
        if (state->flags & 0x0200)
            state->check = crc32(state->check, next, copy);
        have -= copy;
        next += copy;
        if (len) goto inf_leave;
    }
    else if (state->head != Z_NULL)
        state->head->name = Z_NULL;
    state->length = 0;
    state->mode = COMMENT;
case COMMENT:
    if (state->flags & 0x1000) {
        if (have == 0) goto inf_leave;
        copy = 0;
        do {
            len = (unsigned)(next[copy++]);
            if (state->head != Z_NULL &&
                state->head->comment != Z_NULL &&
                state->length < state->head->comm_max)
                state->head->comment[state->length++] = len;
        } while (len && copy < have);
        if (state->flags & 0x0200)
            state->check = crc32(state->check, next, copy);
        have -= copy;
        next += copy;
        if (len) goto inf_leave;
    }
    else if (state->head != Z_NULL)

```

```

        state->head->comment = Z_NULL;
        state->mode = HCRC;
    case HCRC:
        if (state->flags & 0x0200) {
            NEEDBITS(16);
            if (hold != (state->check & 0xffff)) {
                strm->msg = (char *) "header crc mismatch";
                state->mode = BAD;
                break;
            }
            INITBITS();
        }
        if (state->head != Z_NULL) {
            state->head->hcrc = (int)((state->flags >> 9) & 1);
            state->head->done = 1;
        }
        strm->adler = state->check = crc32(0L, Z_NULL, 0);
        state->mode = TYPE;
        break;
#endif

    case DICTID:
        NEEDBITS(32);
        strm->adler = state->check = REVERSE(hold);
        INITBITS();
        state->mode = DICT;
    case DICT:
        if (state->havedict == 0) {
            RESTORE();
            return Z_NEED_DICT;
        }
        strm->adler = state->check = Adler32(0L, Z_NULL, 0);
        state->mode = TYPE;
    case TYPE:
        if (flush == Z_BLOCK) goto inf_leave;
    case TYPEDO:
        if (state->last) {
            BYTEBITS();
            state->mode = CHECK;
            break;
        }
        NEEDBITS(3);
        state->last = BITS(1);
        DROPBITS(1);
        switch (BITS(2)) {
            case 0: /* stored block */
                Tracev((stderr, "inflate: stored block%s\n",
                    state->last ? " (last)" : ""));
                state->mode = STORED;
                break;
            case 1: /* fixed block */
                fixedtables(state);
                Tracev((stderr, "inflate: fixed codes block%s\n",
                    state->last ? " (last)" : ""));
                state->mode = LEN; /* decode codes */
                break;
            case 2: /* dynamic block */
                Tracev((stderr, "inflate: dynamic codes block%s\n",
                    state->last ? " (last)" : ""));
                state->mode = TABLE;
                break;
            case 3:
                strm->msg = (char *) "invalid block type";
                state->mode = BAD;
        }
        DROPBITS(2);
        break;
    case STORED:
        BYTEBITS(); /* go to byte boundary */
        NEEDBITS(32);
        if ((hold & 0xffff) != ((hold >> 16) ^ 0xffff)) {
            strm->msg = (char *) "invalid stored block lengths";
            state->mode = BAD;
            break;
        }

```

```

        state->length = (unsigned)hold & 0xffff;
        Tracev((stderr, "inflate:   stored length %u\n",
                    state->length));
        INITBITS();
        state->mode = COPY;
    case COPY:
        copy = state->length;
        if (copy) {
            if (copy > have) copy = have;
            if (copy > left) copy = left;
            if (copy == 0) goto inf_leave;
            zmemcpy(put, next, copy);
            have -= copy;
            next += copy;
            left -= copy;
            put += copy;
            state->length -= copy;
            break;
        }
        Tracev((stderr, "inflate:   stored end\n"));
        state->mode = TYPE;
        break;
    case TABLE:
        NEEDBITS(14);
        state->nlen = BITS(5) + 257;
        DROPBITS(5);
        state->ndist = BITS(5) + 1;
        DROPBITS(5);
        state->ncode = BITS(4) + 4;
        DROPBITS(4);
#ifndef PKZIP_BUG_WORKAROUND
        if (state->nlen > 286 || state->ndist > 30) {
            strm->msg = (char *) "too many length or distance symbols";
            state->mode = BAD;
            break;
        }
#endif
        Tracev((stderr, "inflate:   table sizes ok\n"));
        state->have = 0;
        state->mode = LENLENS;
    case LENLENS:
        while (state->have < state->ncode) {
            NEEDBITS(3);
            state->lens[order[state->have++]] = (unsigned short)BITS(3);
            DROPBITS(3);
        }
        while (state->have < 19)
            state->lens[order[state->have++]] = 0;
        state->next = state->codes;
        state->lencode = (code const FAR *) (state->next);
        state->lenbits = 7;
        ret = inflate_table(CODES, state->lens, 19, &(state->next),
                            &(state->lenbits), state->work);
        if (ret) {
            strm->msg = (char *) "invalid code lengths set";
            state->mode = BAD;
            break;
        }
        Tracev((stderr, "inflate:   code lengths ok\n"));
        state->have = 0;
        state->mode = CODELENS;
    case CODELENS:
        while (state->have < state->nlen + state->ndist) {
            for (;;) {
                this = state->lencode[BITS(state->lenbits)];
                if ((unsigned)(this.bits) <= bits) break;
                PULLBYTE();
            }
            if (this.val < 16) {
                NEEDBITS(this.bits);
                DROPBITS(this.bits);
                state->lens[state->have++] = this.val;
            }
            else {

```

```

        if (this.val == 16) {
            NEEDBITS(this.bits + 2);
            DROPBITS(this.bits);
            if (state->have == 0) {
                strm->msg = (char *) "invalid bit length repeat";
                state->mode = BAD;
                break;
            }
            len = state->lens[state->have - 1];
            copy = 3 + BITS(2);
            DROPBITS(2);
        }
        else if (this.val == 17) {
            NEEDBITS(this.bits + 3);
            DROPBITS(this.bits);
            len = 0;
            copy = 3 + BITS(3);
            DROPBITS(3);
        }
        else {
            NEEDBITS(this.bits + 7);
            DROPBITS(this.bits);
            len = 0;
            copy = 11 + BITS(7);
            DROPBITS(7);
        }
        if (state->have + copy > state->nlen + state->ndist) {
            strm->msg = (char *) "invalid bit length repeat";
            state->mode = BAD;
            break;
        }
        while (copy--)
            state->lens[state->have++] = (unsigned short)len;
    }

    /* handle error breaks in while */
    if (state->mode == BAD) break;

    /* build code tables */
    state->next = state->codes;
    state->lencode = (code const FAR *) (state->next);
    state->lenbits = 9;
    ret = inflate_table(LENS, state->lens, state->nlen, &(state->next),
                        &(state->lenbits), state->work);
    if (ret) {
        strm->msg = (char *) "invalid literal/lengths set";
        state->mode = BAD;
        break;
    }
    state->distcode = (code const FAR *) (state->next);
    state->distbits = 6;
    ret = inflate_table(DISTS, state->lens + state->nlen, state->ndist,
                        &(state->next), &(state->distbits), state->work);
    if (ret) {
        strm->msg = (char *) "invalid distances set";
        state->mode = BAD;
        break;
    }
    Tracev((stderr, "inflate:   codes ok\n"));
    state->mode = LEN;
case LEN:
    if (have >= 6 && left >= 258) {
        RESTORE();
        inflate_fast(strm, out);
        LOAD();
        break;
    }
    for (;;) {
        this = state->lencode[BITS(state->lenbits)];
        if ((unsigned)(this.bits) <= bits) break;
        PULLBYTE();
    }
    if (this.op && (this.op & 0xf0) == 0) {

```

```

        last = this;
        for (;;) {
            this = state->lencode[last.val +
                (BITS(last.bits + last.op) >> last.bits)];
            if ((unsigned)(last.bits + this.bits) <= bits) break;
            PULLBYTE();
        }
        DROPBITS(last.bits);
    }
    DROPBITS(this.bits);
    state->length = (unsigned)this.val;
    if ((int)(this.op) == 0) {
        Tracevv((stderr, this.val >= 0x20 && this.val < 0x7f ?
            "inflate:     literal '%c'\n" :
            "inflate:     literal 0x%02x\n", this.val));
        state->mode = LIT;
        break;
    }
    if (this.op & 32) {
        Tracevv((stderr, "inflate:     end of block\n"));
        state->mode = TYPE;
        break;
    }
    if (this.op & 64) {
        strm->msg = (char *) "invalid literal/length code";
        state->mode = BAD;
        break;
    }
    state->extra = (unsigned)(this.op) & 15;
    state->mode = LENEXT;
case LENEXT:
    if (state->extra) {
        NEEDBITS(state->extra);
        state->length += BITS(state->extra);
        DROPBITS(state->extra);
    }
    Tracevv((stderr, "inflate:     length %u\n", state->length));
    state->mode = DIST;
case DIST:
    for (;;) {
        this = state->distcode[BITS(state->distbits)];
        if ((unsigned)(this.bits) <= bits) break;
        PULLBYTE();
    }
    if ((this.op & 0xf0) == 0) {
        last = this;
        for (;;) {
            this = state->distcode[last.val +
                (BITS(last.bits + last.op) >> last.bits)];
            if ((unsigned)(last.bits + this.bits) <= bits) break;
            PULLBYTE();
        }
        DROPBITS(last.bits);
    }
    DROPBITS(this.bits);
    if (this.op & 64) {
        strm->msg = (char *) "invalid distance code";
        state->mode = BAD;
        break;
    }
    state->offset = (unsigned)this.val;
    state->extra = (unsigned)(this.op) & 15;
    state->mode = DISTEXT;
case DISTEXT:
    if (state->extra) {
        NEEDBITS(state->extra);
        state->offset += BITS(state->extra);
        DROPBITS(state->extra);
    }
#ifdef INFLATE_STRICT
    if (state->offset > state->dmax) {
        strm->msg = (char *) "invalid distance too far back";
        state->mode = BAD;
        break;
    }

```

```

    }
#endif
    if (state->offset > state->whave + out - left) {
        strm->msg = (char *) "invalid distance too far back";
        state->mode = BAD;
        break;
    }
    Tracevv((stderr, "inflate: distance %u\n", state->offset));
    state->mode = MATCH;
case MATCH:
    if (left == 0) goto inf_leave;
    copy = out - left;
    if (state->offset > copy) {          /* copy from window */
        copy = state->offset - copy;
        if (copy > state->write) {
            copy -= state->write;
            from = state->window + (state->wsize - copy);
        }
        else
            from = state->window + (state->write - copy);
        if (copy > state->length) copy = state->length;
    }
    else {                              /* copy from output */
        from = put - state->offset;
        copy = state->length;
    }
    if (copy > left) copy = left;
    left -= copy;
    state->length -= copy;
    do {
        *put++ = *from++;
    } while (--copy);
    if (state->length == 0) state->mode = LEN;
    break;
case LIT:
    if (left == 0) goto inf_leave;
    *put++ = (unsigned char)(state->length);
    left--;
    state->mode = LEN;
    break;
case CHECK:
    if (state->wrap) {
        NEEDBITS(32);
        out -= left;
        strm->total_out += out;
        state->total += out;
        if (out)
            strm->adler = state->check =
                UPDATE(state->check, put - out, out);
        out = left;
    }
    if ((
#ifdef GUNZIP
        state->flags ? hold :
#endif
        REVERSE(hold)) != state->check) {
        strm->msg = (char *) "incorrect data check";
        state->mode = BAD;
        break;
    }
    INITBITS();
    Tracev((stderr, "inflate: check matches trailer\n"));
}
#ifdef GUNZIP
state->mode = LENGTH;
case LENGTH:
    if (state->wrap && state->flags) {
        NEEDBITS(32);
        if (hold != (state->total & 0xffffffffUL)) {
            strm->msg = (char *) "incorrect length check";
            state->mode = BAD;
            break;
        }
        INITBITS();
        Tracev((stderr, "inflate: length matches trailer\n"));
    }

```

```

    }
#endif
    state->mode = DONE;
    case DONE:
        ret = Z_STREAM_END;
        goto inf_leave;
    case BAD:
        ret = Z_DATA_ERROR;
        goto inf_leave;
    case MEM:
        return Z_MEM_ERROR;
    case SYNC:
    default:
        return Z_STREAM_ERROR;
    }

    /*
     * Return from inflate(), updating the total counts and the check value.
     * If there was no progress during the inflate() call, return a buffer
     * error. Call updatewindow() to create and/or update the window state.
     * Note: a memory error from inflate() is non-recoverable.
     */
    inf_leave:
    RESTORE();
    if (state->wsize || (state->mode < CHECK && out != strm->avail_out))
        if (updatewindow(strm, out)) {
            state->mode = MEM;
            return Z_MEM_ERROR;
        }
    in -= strm->avail_in;
    out -= strm->avail_out;
    strm->total_in += in;
    strm->total_out += out;
    state->total += out;
    if (state->wrap && out)
        strm->adler = state->check =
            UPDATE(state->check, strm->next_out - out, out);
    strm->data_type = state->bits + (state->last ? 64 : 0) +
        (state->mode == TYPE ? 128 : 0);
    if (((in == 0 && out == 0) || flush == Z_FINISH) && ret == Z_OK)
        ret = Z_BUF_ERROR;
    return ret;
}

int ZEXPORT inflateEnd(strm)
z_stream *strm;
{
    struct inflate_state FAR *state;
    if (strm == Z_NULL || strm->state == Z_NULL || strm->zfree == (free_func)0)
        return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;
    if (state->window != Z_NULL) ZFREE(strm, state->window);
    ZFREE(strm, strm->state);
    strm->state = Z_NULL;
    Tracev((stderr, "inflate: end\n"));
    return Z_OK;
}

int ZEXPORT inflateSetDictionary(strm, dictionary, dictLength)
z_stream *strm;
const Bytef *dictionary;
uInt dictLength;
{
    struct inflate_state FAR *state;
    unsigned long id;

    /* check state */
    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;
    if (state->wrap != 0 && state->mode != DICT)
        return Z_STREAM_ERROR;

    /* check for correct dictionary id */
    if (state->mode == DICT) {

```

```

        id = Adler32(0L, Z_NULL, 0);
        id = Adler32(id, dictionary, dictLength);
        if (id != state->check)
            return Z_DATA_ERROR;
    }

    /* copy dictionary to window */
    if (updatewindow(strm, strm->avail_out)) {
        state->mode = MEM;
        return Z_MEM_ERROR;
    }
    if (dictLength > state->wsize) {
        zmemcpy(state->window, dictionary + dictLength - state->wsize,
            state->wsize);
        state->whave = state->wsize;
    }
    else {
        zmemcpy(state->window + state->wsize - dictLength, dictionary,
            dictLength);
        state->whave = dictLength;
    }
    state->havedict = 1;
    Tracev((stderr, "inflate: dictionary set\n"));
    return Z_OK;
}

int ZEXPORT inflateGetHeader(strm, head)
z_streamp strm;
gz_headerp head;
{
    struct inflate_state FAR *state;

    /* check state */
    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;
    if ((state->wrap & 2) == 0) return Z_STREAM_ERROR;

    /* save header structure */
    state->head = head;
    head->done = 0;
    return Z_OK;
}

/*
Search buf[0..len-1] for the pattern: 0, 0, 0xff, 0xff. Return when found
or when out of input. When called, *have is the number of pattern bytes
found in order so far, in 0..3. On return *have is updated to the new
state. If on return *have equals four, then the pattern was found and the
return value is how many bytes were read including the last byte of the
pattern. If *have is less than four, then the pattern has not been found
yet and the return value is len. In the latter case, syncsearch() can be
called again with more data and the *have state. *have is initialized to
zero for the first call.
*/
local unsigned syncsearch(have, buf, len)
unsigned FAR *have;
unsigned char FAR *buf;
unsigned len;
{
    unsigned got;
    unsigned next;

    got = *have;
    next = 0;
    while (next < len && got < 4) {
        if ((int)(buf[next]) == (got < 2 ? 0 : 0xff))
            got++;
        else if (buf[next])
            got = 0;
        else
            got = 4 - got;
        next++;
    }
    *have = got;

```



```

    return next;
}

int ZEXPORT inflateSync(strm)
z_streamp strm;
{
    unsigned len;                /* number of bytes to look at or looked at */
    unsigned long in, out;        /* temporary to save total_in and total_out */
    unsigned char buf[4];        /* to restore bit buffer to byte string */
    struct inflate_state FAR *state;

    /* check parameters */
    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;
    if (strm->avail_in == 0 && state->bits < 8) return Z_BUF_ERROR;

    /* if first time, start search in bit buffer */
    if (state->mode != SYNC) {
        state->mode = SYNC;
        state->hold <=& state->bits & 7;
        state->bits -= state->bits & 7;
        len = 0;
        while (state->bits >= 8) {
            buf[len++] = (unsigned char)(state->hold);
            state->hold >>= 8;
            state->bits -= 8;
        }
        state->have = 0;
        syncsearch(&(state->have), buf, len);
    }

    /* search available input */
    len = syncsearch(&(state->have), strm->next_in, strm->avail_in);
    strm->avail_in -= len;
    strm->next_in += len;
    strm->total_in += len;

    /* return no joy or set up to restart inflate() on a new block */
    if (state->have != 4) return Z_DATA_ERROR;
    in = strm->total_in;  out = strm->total_out;
    inflateReset(strm);
    strm->total_in = in;  strm->total_out = out;
    state->mode = TYPE;
    return Z_OK;
}

/*
Returns true if inflate is currently at the end of a block generated by
Z_SYNC_FLUSH or Z_FULL_FLUSH. This function is used by one PPP
implementation to provide an additional safety check. PPP uses
Z_SYNC_FLUSH but removes the length bytes of the resulting empty stored
block. When decompressing, PPP checks that at the end of input packet,
inflate is waiting for these length bytes.
*/
int ZEXPORT inflateSyncPoint(strm)
z_streamp strm;
{
    struct inflate_state FAR *state;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;
    return state->mode == STORED && state->bits == 0;
}

int ZEXPORT inflateCopy(dest, source)
z_streamp dest;
z_streamp source;
{
    struct inflate_state FAR *state;
    struct inflate_state FAR *copy;
    unsigned char FAR *window;
    unsigned wsize;

    /* check input */

```

```

if (dest == Z_NULL || source == Z_NULL || source->state == Z_NULL ||
    source->zalloc == (alloc_func)0 || source->zfree == (free_func)0)
    return Z_STREAM_ERROR;
state = (struct inflate_state FAR *)source->state;

/* allocate space */
copy = (struct inflate_state FAR *)
    ZALLOC(source, 1, sizeof(struct inflate_state));
if (copy == Z_NULL) return Z_MEM_ERROR;
window = Z_NULL;
if (state->window != Z_NULL) {
    window = (unsigned char FAR *)
        ZALLOC(source, 1U << state->wbits, sizeof(unsigned char));
    if (window == Z_NULL) {
        ZFREE(source, copy);
        return Z_MEM_ERROR;
    }
}

/* copy state */
zmemcpy(dest, source, sizeof(z_stream));
zmemcpy(copy, state, sizeof(struct inflate_state));
if (state->lencode >= state->codes &&
    state->lencode <= state->codes + ENOUGH - 1) {
    copy->lencode = copy->codes + (state->lencode - state->codes);
    copy->distcode = copy->codes + (state->distcode - state->codes);
}
copy->next = copy->codes + (state->next - state->codes);
if (window != Z_NULL) {
    wsize = 1U << state->wbits;
    zmemcpy(window, state->window, wsize);
}
copy->window = window;
dest->state = (struct internal_state FAR *)copy;
return Z_OK;
}

```

```

/* inflate.h -- internal inflate state definition
 * Copyright (C) 1995-2004 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
 part of the implementation of the compression library and is
 subject to change. Applications should only use zlib.h.
 */

/* define NO_GZIP when compiling if you want to disable gzip header and
 trailer decoding by inflate(). NO_GZIP would be used to avoid linking in
 the crc code when it is not needed. For shared libraries, gzip decoding
 should be left enabled. */
#ifndef NO_GZIP
# define GUNZIP
#endif

/* Possible inflate modes between inflate() calls */
typedef enum {
    HEAD,          /* i: waiting for magic header */
    FLAGS,         /* i: waiting for method and flags (gzip) */
    TIME,          /* i: waiting for modification time (gzip) */
    OS,            /* i: waiting for extra flags and operating system (gzip) */
    EXLEN,         /* i: waiting for extra length (gzip) */
    EXTRA,        /* i: waiting for extra bytes (gzip) */
    NAME,          /* i: waiting for end of file name (gzip) */
    COMMENT,       /* i: waiting for end of comment (gzip) */
    HCRC,          /* i: waiting for header crc (gzip) */
    DICTID,        /* i: waiting for dictionary check value */
    DICT,          /* i: waiting for inflateSetDictionary() call */
    TYPE,          /* i: waiting for type bits, including last-flag bit */
    TYPEDO,        /* i: same, but skip check to exit inflate on new block */
    STORED,        /* i: waiting for stored size (length and complement) */
    COPY,          /* i/o: waiting for input or output to copy stored block */
    TABLE,        /* i: waiting for dynamic block table lengths */
    LENLENS,       /* i: waiting for code length code lengths */
    CODELENS,      /* i: waiting for length/lit and distance code lengths */
    LEN,           /* i: waiting for length/lit code */
    LENEXT,        /* i: waiting for length extra bits */
    DIST,          /* i: waiting for distance code */
    DISTEXT,       /* i: waiting for distance extra bits */
    MATCH,         /* o: waiting for output space to copy string */
    LIT,           /* o: waiting for output space to write literal */
    CHECK,         /* i: waiting for 32-bit check value */
    LENGTH,        /* i: waiting for 32-bit length (gzip) */
    DONE,          /* finished check, done -- remain here until reset */
    BAD,           /* got a data error -- remain here until reset */
    MEM,           /* got an inflate() memory error -- remain here until reset */
    SYNC           /* looking for synchronization bytes to restart inflate() */
} inflate_mode;

/*
 * State transitions between above modes -
 *
 * (most modes can go to the BAD or MEM mode -- not shown for clarity)
 *
 * Process header:
 *   HEAD -> (gzip) or (zlib)
 *   (gzip) -> FLAGS -> TIME -> OS -> EXLEN -> EXTRA -> NAME
 *   NAME -> COMMENT -> HCRC -> TYPE
 *   (zlib) -> DICTID or TYPE
 *   DICTID -> DICT -> TYPE
 * Read deflate blocks:
 *   TYPE -> STORED or TABLE or LEN or CHECK
 *   STORED -> COPY -> TYPE
 *   TABLE -> LENLENS -> CODELENS -> LEN
 * Read deflate codes:
 *   LEN -> LENEXT or LIT or TYPE
 *   LENEXT -> DIST -> DISTEXT -> MATCH -> LEN
 *   LIT -> LEN
 * Process trailer:
 *   CHECK -> LENGTH -> DONE
 */

```

```
/* state maintained between inflate() calls. Approximately 7K bytes. */
struct inflate_state {
    inflate_mode mode;          /* current inflate mode */
    int last;                   /* true if processing last block */
    int wrap;                   /* bit 0 true for zlib, bit 1 true for gzip */
    int havedict;               /* true if dictionary provided */
    int flags;                  /* gzip header method and flags (0 if zlib) */
    unsigned dmax;              /* zlib header max distance (INFLATE_STRICT) */
    unsigned long check;        /* protected copy of check value */
    unsigned long total;        /* protected copy of output count */
    gz_headerp head;           /* where to save gzip header information */

    /* sliding window */
    unsigned wbits;             /* log base 2 of requested window size */
    unsigned wsize;             /* window size or zero if not using window */
    unsigned whave;             /* valid bytes in the window */
    unsigned write;             /* window write index */
    unsigned char FAR *window; /* allocated sliding window, if needed */

    /* bit accumulator */
    unsigned long hold;         /* input bit accumulator */
    unsigned bits;             /* number of bits in "in" */

    /* for string and stored block copying */
    unsigned length;            /* literal or length of data to copy */
    unsigned offset;            /* distance back to copy string from */

    /* for table and code decoding */
    unsigned extra;             /* extra bits needed */

    /* fixed and dynamic code tables */
    code const FAR *lencode;    /* starting table for length/literal codes */
    code const FAR *distcode;   /* starting table for distance codes */
    unsigned lenbits;           /* index bits for lencode */
    unsigned distbits;          /* index bits for distcode */

    /* dynamic table building */
    unsigned ncode;             /* number of code length code lengths */
    unsigned nlen;              /* number of length code lengths */
    unsigned ndist;             /* number of distance code lengths */
    unsigned have;              /* number of code lengths in lens[] */
    code FAR *next;             /* next available space in codes[] */
    unsigned short lens[320];    /* temporary storage for code lengths */
    unsigned short work[288];    /* work area for code table building */
    code codes[ENOUGH];         /* space for code tables */
};
```

```

/* inftrees.c -- generate Huffman trees for efficient decoding
 * Copyright (C) 1995-2005 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "inftrees.h"

#define MAXBITS 15

const char inflate_copyright[] =
    " inflate 1.2.3 Copyright 1995-2005 Mark Adler ";
/*
 * If you use the zlib library in a product, an acknowledgment is welcome
 * in the documentation of your product. If for some reason you cannot
 * include such an acknowledgment, I would appreciate that you keep this
 * copyright string in the executable of your product.
 */
/*
 * Build a set of tables to decode the provided canonical Huffman code.
 * The code lengths are lens[0..codes-1]. The result starts at *table,
 * whose indices are 0..2^bits-1. work is a writable array of at least
 * lens shorts, which is used as a work area. type is the type of code
 * to be generated, CODES, LENS, or DISTS. On return, zero is success,
 * -1 is an invalid code, and +1 means that ENOUGH isn't enough. table
 * on return points to the next available entry's address. bits is the
 * requested root table index bits, and on return it is the actual root
 * table index bits. It will differ if the request is greater than the
 * longest code or if it is less than the shortest code.
 */
int inflate_table(type, lens, codes, table, bits, work)
codetype type;
unsigned short FAR *lens;
unsigned codes;
code FAR * FAR *table;
unsigned FAR *bits;
unsigned short FAR *work;
{
    unsigned len;                /* a code's length in bits */
    unsigned sym;                /* index of code symbols */
    unsigned min, max;           /* minimum and maximum code lengths */
    unsigned root;               /* number of index bits for root table */
    unsigned curr;               /* number of index bits for current table */
    unsigned drop;               /* code bits to drop for sub-table */
    int left;                    /* number of prefix codes available */
    unsigned used;               /* code entries in table used */
    unsigned huff;               /* Huffman code */
    unsigned incr;               /* for incrementing code, index */
    unsigned fill;               /* index for replicating entries */
    unsigned low;                /* low bits for current root entry */
    unsigned mask;               /* mask for low root bits */
    code this;                   /* table entry for duplication */
    code FAR *next;              /* next available space in table */
    const unsigned short FAR *base; /* base value table to use */
    const unsigned short FAR *extra; /* extra bits table to use */
    int end;                     /* use base and extra for symbol > end */
    unsigned short count[MAXBITS+1]; /* number of codes of each length */
    unsigned short offs[MAXBITS+1]; /* offsets in table for each length */
    static const unsigned short lbase[31] = { /* Length codes 257..285 base */
        3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 19, 23, 27, 31,
        35, 43, 51, 59, 67, 83, 99, 115, 131, 163, 195, 227, 258, 0, 0};
    static const unsigned short lext[31] = { /* Length codes 257..285 extra */
        16, 16, 16, 16, 16, 16, 16, 16, 17, 17, 17, 17, 18, 18, 18, 18,
        19, 19, 19, 19, 20, 20, 20, 20, 21, 21, 21, 21, 16, 201, 196};
    static const unsigned short dbase[32] = { /* Distance codes 0..29 base */
        1, 2, 3, 4, 5, 7, 9, 13, 17, 25, 33, 49, 65, 97, 129, 193,
        257, 385, 513, 769, 1025, 1537, 2049, 3073, 4097, 6145,
        8193, 12289, 16385, 24577, 0, 0};
    static const unsigned short dext[32] = { /* Distance codes 0..29 extra */
        16, 16, 16, 16, 17, 17, 18, 18, 18, 19, 19, 20, 20, 21, 21, 22, 22,
        23, 23, 24, 24, 25, 25, 26, 26, 27, 27,
        28, 28, 29, 29, 64, 64};

```

```

/*
    Process a set of code lengths to create a canonical Huffman code.  The
    code lengths are lens[0..codes-1].  Each length corresponds to the
    symbols 0..codes-1.  The Huffman code is generated by first sorting the
    symbols by length from short to long, and retaining the symbol order
    for codes with equal lengths.  Then the code starts with all zero bits
    for the first code of the shortest length, and the codes are integer
    increments for the same length, and zeros are appended as the length
    increases.  For the deflate format, these bits are stored backwards
    from their more natural integer increment ordering, and so when the
    decoding tables are built in the large loop below, the integer codes
    are incremented backwards.

    This routine assumes, but does not check, that all of the entries in
    lens[] are in the range 0..MAXBITS.  The caller must assure this.
    1..MAXBITS is interpreted as that code length.  zero means that that
    symbol does not occur in this code.

    The codes are sorted by computing a count of codes for each length,
    creating from that a table of starting indices for each length in the
    sorted table, and then entering the symbols in order in the sorted
    table.  The sorted table is work[], with that space being provided by
    the caller.

    The length counts are used for other purposes as well, i.e. finding
    the minimum and maximum length codes, determining if there are any
    codes at all, checking for a valid set of lengths, and looking ahead
    at length counts to determine sub-table sizes when building the
    decoding tables.
*/

/* accumulate lengths for codes (assumes lens[] all in 0..MAXBITS) */
for (len = 0; len <= MAXBITS; len++)
    count[len] = 0;
for (sym = 0; sym < codes; sym++)
    count[lens[sym]]++;

/* bound code lengths, force root to be within code lengths */
root = *bits;
for (max = MAXBITS; max >= 1; max--)
    if (count[max] != 0) break;
if (root > max) root = max;
if (max == 0) {
    this.op = (unsigned char)64; /* no symbols to code at all */
    this.bits = (unsigned char)1; /* invalid code marker */
    this.val = (unsigned short)0;
    (*table)++ = this; /* make a table to force an error */
    (*table)++ = this;
    *bits = 1;
    return 0; /* no symbols, but wait for decoding to report error */
}
for (min = 1; min <= MAXBITS; min++)
    if (count[min] != 0) break;
if (root < min) root = min;

/* check for an over-subscribed or incomplete set of lengths */
left = 1;
for (len = 1; len <= MAXBITS; len++) {
    left <= 1;
    left -= count[len];
    if (left < 0) return -1; /* over-subscribed */
}
if (left > 0 && (type == CODES || max != 1))
    return -1; /* incomplete set */

/* generate offsets into symbol table for each length for sorting */
offs[1] = 0;
for (len = 1; len < MAXBITS; len++)
    offs[len + 1] = offs[len] + count[len];

/* sort symbols by length, by symbol order within each length */
for (sym = 0; sym < codes; sym++)
    if (lens[sym] != 0) work[offs[lens[sym]]++] = (unsigned short)sym;

```

```

/*
    Create and fill in decoding tables.  In this loop, the table being
    filled is at next and has curr index bits.  The code being used is huff
    with length len.  That code is converted to an index by dropping drop
    bits off of the bottom.  For codes where len is less than drop + curr,
    those top drop + curr - len bits are incremented through all values to
    fill the table with replicated entries.

    root is the number of index bits for the root table.  When len exceeds
    root, sub-tables are created pointed to by the root entry with an index
    of the low root bits of huff.  This is saved in low to check for when a
    new sub-table should be started.  drop is zero when the root table is
    being filled, and drop is root when sub-tables are being filled.

    When a new sub-table is needed, it is necessary to look ahead in the
    code lengths to determine what size sub-table is needed.  The length
    counts are used for this, and so count[] is decremented as codes are
    entered in the tables.

    used keeps track of how many table entries have been allocated from the
    provided *table space.  It is checked when a LENS table is being made
    against the space in *table, ENOUGH, minus the maximum space needed by
    the worst case distance code, MAXD.  This should never happen, but the
    sufficiency of ENOUGH has not been proven exhaustively, hence the check.
    This assumes that when type == LENS, bits == 9.

    sym increments through all symbols, and the loop terminates when
    all codes of length max, i.e. all codes, have been processed.  This
    routine permits incomplete codes, so another loop after this one fills
    in the rest of the decoding tables with invalid code markers.
*/

/* set up for code type */
switch (type) {
case CODES:
    base = extra = work;    /* dummy value--not used */
    end = 19;
    break;
case LENS:
    base = lbase;
    base -= 257;
    extra = lext;
    extra -= 257;
    end = 256;
    break;
default:                /* DISTS */
    base = dbase;
    extra = dext;
    end = -1;
}

/* initialize state for loop */
huff = 0;                /* starting code */
sym = 0;                  /* starting code symbol */
len = min;                /* starting code length */
next = *table;            /* current table to fill in */
curr = root;              /* current table index bits */
drop = 0;                 /* current bits to drop from code for index */
low = (unsigned)(-1);      /* trigger new sub-table when len > root */
used = 1U << root;        /* use root table entries */
mask = used - 1;          /* mask for comparing low */

/* check available table space */
if (type == LENS && used >= ENOUGH - MAXD)
    return 1;

/* process all codes and make table entries */
for (;;) {
    /* create table entry */
    this.bits = (unsigned char)(len - drop);
    if ((int)(work[sym]) < end) {
        this.op = (unsigned char)0;
        this.val = work[sym];
    }
}

```

```

else if ((int)(work[sym]) > end) {
    this.op = (unsigned char)(extra[work[sym]]);
    this.val = base[work[sym]];
}
else {
    this.op = (unsigned char)(32 + 64);      /* end of block */
    this.val = 0;
}

/* replicate for those indices with low len bits equal to huff */
incr = 1U << (len - drop);
fill = 1U << curr;
min = fill;                                /* save offset to next table */
do {
    fill -= incr;
    next[(huff >> drop) + fill] = this;
} while (fill != 0);

/* backwards increment the len-bit code huff */
incr = 1U << (len - 1);
while (huff & incr)
    incr >>= 1;
if (incr != 0) {
    huff &= incr - 1;
    huff += incr;
}
else
    huff = 0;

/* go to next symbol, update count, len */
sym++;
if (--(count[len]) == 0) {
    if (len == max) break;
    len = lens[work[sym]];
}

/* create new sub-table if needed */
if (len > root && (huff & mask) != low) {
    /* if first time, transition to sub-tables */
    if (drop == 0)
        drop = root;

    /* increment past last table */
    next += min;      /* here min is 1 << curr */

    /* determine length of next table */
    curr = len - drop;
    left = (int)(1 << curr);
    while (curr + drop < max) {
        left -= count[curr + drop];
        if (left <= 0) break;
        curr++;
        left <= 1;
    }

    /* check for enough space */
    used += 1U << curr;
    if (type == LENS && used >= ENOUGH - MAXD)
        return 1;

    /* point entry in root table to sub-table */
    low = huff & mask;
    (*table)[low].op = (unsigned char)curr;
    (*table)[low].bits = (unsigned char)root;
    (*table)[low].val = (unsigned short)(next - *table);
}
}

/*
Fill in rest of table for incomplete codes. This loop is similar to the
loop above in incrementing huff for table indices. It is assumed that
len is equal to curr + drop, so there is no loop needed to increment
through high index bits. When the current sub-table is filled, the loop
drops back to the root table to fill in any remaining entries there.

```



```
    /*
this.op = (unsigned char)64;          /* invalid code marker */
this.bits = (unsigned char)(len - drop);
this.val = (unsigned short)0;
while (huff != 0) {
    /* when done with sub-table, drop back to root table */
    if (drop != 0 && (huff & mask) != low) {
        drop = 0;
        len = root;
        next = *table;
        this.bits = (unsigned char)len;
    }

    /* put invalid code marker in table */
    next[huff >> drop] = this;

    /* backwards increment the len-bit code huff */
    incr = 1U << (len - 1);
    while (huff & incr)
        incr >>= 1;
    if (incr != 0) {
        huff &= incr - 1;
        huff += incr;
    }
    else
        huff = 0;
}

/* set return parameters */
*table += used;
*bits = root;
return 0;
}
```

```
/* inftrees.h -- header to use inftrees.c
 * Copyright (C) 1995-2005 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
 * part of the implementation of the compression library and is
 * subject to change. Applications should only use zlib.h.
 */

/* Structure for decoding tables. Each entry provides either the
 * information needed to do the operation requested by the code that
 * indexed that table entry, or it provides a pointer to another
 * table that indexes more bits of the code. op indicates whether
 * the entry is a pointer to another table, a literal, a length or
 * distance, an end-of-block, or an invalid code. For a table
 * pointer, the low four bits of op is the number of index bits of
 * that table. For a length or distance, the low four bits of op
 * is the number of extra bits to get after the code. bits is
 * the number of bits in this code or part of the code to drop off
 * of the bit buffer. val is the actual byte to output in the case
 * of a literal, the base length or distance, or the offset from
 * the current table to the next table. Each entry is four bytes. */
typedef struct {
    unsigned char op;          /* operation, extra bits, table bits */
    unsigned char bits;       /* bits in this part of the code */
    unsigned short val;        /* offset in table or code value */
} code;

/* op values as set by inflate_table():
   00000000 - literal
   0000tttt - table link, tttt != 0 is the number of table index bits
   0001eeee - length or distance, eeee is the number of extra bits
   01100000 - end of block
   01000000 - invalid code
 */

/* Maximum size of dynamic tree. The maximum found in a long but non-
 * exhaustive search was 1444 code structures (852 for length/literals
 * and 592 for distances, the latter actually the result of an
 * exhaustive search). The true maximum is not known, but the value
 * below is more than safe. */
#define ENOUGH 2048
#define MAXD 592

/* Type of code to build for inftable() */
typedef enum {
    CODES,
    LENS,
    DISTS
} codetype;

extern int inflate_table OF((codetype type, unsigned short FAR *lens,
                             unsigned codes, code FAR * FAR *table,
                             unsigned FAR *bits, unsigned short FAR *work));
```

```
$! make libz under VMS written by
$! Martin P.J. Zinser
$! <zinser@zinser.no-ip.info or zinser@sysdev.deutsche-boerse.com>
$!
$! on error then goto err_exit
$!
$!
$! Just some general constants...
$!
$ true  = 1
$ false = 0
$ tmpnam = "temp_" + f$getjpi("", "pid")
$ SAY = "WRITE SYS$OUTPUT"
$!
$! Setup variables holding "config" information
$!
$ Make      = ""
$ name      = "Zlib"
$ version   = "?.?.?"
$ v_string  = "ZLIB_VERSION"
$ v_file    = "zlib.h"
$ ccopt     = ""
$ lopts     = ""
$ linkonly  = false
$ optfile   = name + ".opt"
$ its_decc  = false
$ its_vaxc  = false
$ its_gnuc  = false
$ axp       = f$getsyi("HW_MODEL").ge.1024
$ s_case    = false
$! Check for MMK/MMS
$!
$ If F$Search ("Sys$System:MMS.EXE") .nes. "" Then Make = "MMS"
$ If F$Type (MMK) .eqs. "STRING" Then Make = "MMK"
$!
$!
$ gosub find_version
$!
$ gosub check_opts
$!
$! Look for the compiler used
$!
$ gosub check_compiler
$ if its_decc
$ then
$   ccopt = "/prefix=all" + ccopt
$   if f$trnlm("SYS") .eqs. ""
$   then
$     if axp
$     then
$       define sys sys$library:
$     else
$       ccopt = "/decc" + ccopt
$       define sys decc$library_include:
$     endif
$   endif
$ endif
$ if its_vaxc .or. its_gnuc
$ then
$   if f$trnlm("SYS").eqs."" then define sys sys$library:
$ endif
$!
$! Build the thing plain or with mms
$!
$ write sys$output "Compiling Zlib sources ..."
$ if make.eqs.""
$ then
$   dele example.obj;* ,minigzip.obj;*
$   CALL MAKE Adler32.OBJ "CC 'CCOPT' adler32" -
$     adler32.c zlib.h zconf.h
$   CALL MAKE compress.OBJ "CC 'CCOPT' compress" -
$     compress.c zlib.h zconf.h
$   CALL MAKE crc32.OBJ "CC 'CCOPT' crc32" -
$     crc32.c zlib.h zconf.h
```

```

$ CALL MAKE deflate.OBJ "CC 'CCOPT' deflate" -
    deflate.c deflate.h zutil.h zlib.h zconf.h
$ CALL MAKE gzio.OBJ "CC 'CCOPT' gzio" -
    gzio.c zutil.h zlib.h zconf.h
$ CALL MAKE infback.OBJ "CC 'CCOPT' infback" -
    infback.c zutil.h inftrees.h inflate.h inffast.h inffixed.h
$ CALL MAKE inffast.OBJ "CC 'CCOPT' inffast" -
    inffast.c zutil.h zlib.h zconf.h inffast.h
$ CALL MAKE inflate.OBJ "CC 'CCOPT' inflate" -
    inflate.c zutil.h zlib.h zconf.h inffast.h
$ CALL MAKE inftrees.OBJ "CC 'CCOPT' inftrees" -
    inftrees.c zutil.h zlib.h zconf.h inftrees.h
$ CALL MAKE trees.OBJ "CC 'CCOPT' trees" -
    trees.c deflate.h zutil.h zlib.h zconf.h
$ CALL MAKE uncompr.OBJ "CC 'CCOPT' uncompr" -
    uncompr.c zlib.h zconf.h
$ CALL MAKE zutil.OBJ "CC 'CCOPT' zutil" -
    zutil.c zutil.h zlib.h zconf.h
$ write sys$output "Building Zlib ..."
$ CALL MAKE libz.OLB "lib/crea libz.olb *.obj" *.OBJ
$ write sys$output "Building example..."
$ CALL MAKE example.OBJ "CC 'CCOPT' example" -
    example.c zlib.h zconf.h
$ call make example.exe "LINK example,libz.olb/lib" example.obj libz.olb
$ if f$search("xllvms:xvmsutils.olb") .nes. ""
$ then
$     write sys$output "Building minigzip..."
$     CALL MAKE minigzip.OBJ "CC 'CCOPT' minigzip" -
        minigzip.c zlib.h zconf.h
$     call make minigzip.exe -
        "LINK minigzip,libz.olb/lib,xllvms:xvmsutils.olb/lib" -
        minigzip.obj libz.olb
$ endif
$ else
$     gosub crea_mms
$     SAY "Make 'name' 'version' with 'Make' "
$     'make'
$ endif
$!
$! Alpha gets a shareable image
$!
$ If axp
$ Then
$     gosub crea_olist
$     write sys$output "Creating libzshr.exe"
$     call anal_obj_axp modules.opt _link.opt
$     if s_case
$     then
$         open/append optf modules.opt
$         write optf "case_sensitive=YES"
$         close optf
$     endif
$     LINK '_lopts'/SHARE=libzshr.exe modules.opt/opt,_link.opt/opt
$ endif
$ write sys$output "Zlib build completed"
$ exit
$CC_ERR:
$ write sys$output "C compiler required to build 'name'"
$ goto err_exit
$ERR_EXIT:
$ set message/facil/ident/sever/text
$ write sys$output "Exiting..."
$ exit 2
$!
$!
$MAKE: SUBROUTINE      !SUBROUTINE TO CHECK DEPENDENCIES
$ V = 'F$Verify(0)
$! P1 = What we are trying to make
$! P2 = Command to make it
$! P3 - P8 What it depends on
$
$ If F$Search(P1) .Eqs. "" Then Goto Makeit
$ Time = F$CvTime(F$File(P1,"RDT"))
$arg=3

```

```

$Loop:
$   Argument = P'arg
$   If Argument .Eqs. "" Then Goto Exit
$   El=0
$Loop2:
$   File = F$Element(El," ",Argument)
$   If File .Eqs. " " Then Goto EndL
$   AFile = ""
$Loop3:
$   OFile = AFile
$   AFile = F$Search(File)
$   If AFile .Eqs. "" .Or. AFile .Eqs. OFile Then Goto NextEl
$   If F$CvTime(F$File(AFile,"RDT")) .Ges. Time Then Goto Makeit
$   Goto Loop3
$NextEL:
$   El = El + 1
$   Goto Loop2
$EndL:
$ arg=arg+1
$ If arg .Le. 8 Then Goto Loop
$ Goto Exit
$
$Makeit:
$ VV=F$VERIFY(0)
$ write sys$output P2
$ 'P2
$ VV='F$Verify(VV)
$Exit:
$ If V Then Set Verify
$ENDSUBROUTINE
$!-----
$!
$! Check command line options and set symbols accordingly
$!
$ CHECK_OPTS:
$ i = 1
$ OPT_LOOP:
$ if i .lt. 9
$ then
$   cparm = f$edit(p'i',"upcase")
$   if cparm .eqs. "DEBUG"
$   then
$     ccopt = ccopt + "/noopt/deb"
$     lopts = lopts + "/deb"
$   endif
$   if f$locate("CCOPT=",cparm) .lt. f$length(cparm)
$   then
$     start = f$locate("=",cparm) + 1
$     len = f$length(cparm) - start
$     ccopt = ccopt + f$extract(start,len,cparm)
$     if f$locate("AS_IS",f$edit(ccopt,"UPCASE")) .lt. f$length(ccopt) -
$       then s_case = true
$   endif
$   if cparm .eqs. "LINK" then linkonly = true
$   if f$locate("LOPTS=",cparm) .lt. f$length(cparm)
$   then
$     start = f$locate("=",cparm) + 1
$     len = f$length(cparm) - start
$     lopts = lopts + f$extract(start,len,cparm)
$   endif
$   if f$locate("CC=",cparm) .lt. f$length(cparm)
$   then
$     start = f$locate("=",cparm) + 1
$     len = f$length(cparm) - start
$     cc_com = f$extract(start,len,cparm)
$     if (cc_com .nes. "DECC") .and. -
$       (cc_com .nes. "VAXC") .and. -
$       (cc_com .nes. "GNUC")
$     then
$       write sys$output "Unsupported compiler choice ''cc_com' ignored"
$       write sys$output "Use DECC, VAXC, or GNUC instead"
$     else
$       if cc_com .eqs. "DECC" then its_decc = true
$       if cc_com .eqs. "VAXC" then its_vaxc = true

```

```

$      if cc_com .eqs. "GNUC" then its_gnuc = true
$      endif
$      endif
$      if f$locate("MAKE=",cparm) .lt. f$length(cparm)
$      then
$          start = f$locate("=",cparm) + 1
$          len = f$length(cparm) - start
$          mmks = f$extract(start,len,cparm)
$          if (mmks .eqs. "MMK") .or. (mmks .eqs. "MMS")
$          then
$              make = mmks
$          else
$              write sys$output "Unsupported make choice 'mmks' ignored"
$              write sys$output "Use MMK or MMS instead"
$          endif
$      endif
$      i = i + 1
$      goto opt_loop
$      endif
$      return
$!-----
$!
$! Look for the compiler used
$!
$CHECK_COMPILER:
$ if (.not. (its_decc .or. its_vaxc .or. its_gnuc))
$ then
$     its_decc = (f$search("SYS$SYSTEM:DECC$COMPILER.EXE") .nes. "")
$     its_vaxc = .not. its_decc .and. (F$search("SYS$System:VAXC.Exe") .nes. "")
$     its_gnuc = .not. (its_decc .or. its_vaxc) .and. (f$trnlrm("gnu_cc") .nes. "")
$ endif
$!
$! Exit if no compiler available
$!
$ if (.not. (its_decc .or. its_vaxc .or. its_gnuc))
$ then goto CC_ERR
$ else
$     if its_decc then write sys$output "CC compiler check ... Compaq C"
$     if its_vaxc then write sys$output "CC compiler check ... VAX C"
$     if its_gnuc then write sys$output "CC compiler check ... GNU C"
$ endif
$ return
$!-----
$!
$! If MMS/MMK are available dump out the descrip.mms if required
$!
$CREA_MMS:
$ write sys$output "Creating descrip.mms..."
$ create descrip.mms
$ open/append out descrip.mms
$ copy sys$input: out
$ deck
# descrip.mms: MMS description file for building zlib on VMS
# written by Martin P.J. Zinser
# <zinser@zinser.no-ip.info or zinser@sysdev.deutsche-boerse.com>

OBJS = adler32.obj, compress.obj, crc32.obj, gzio.obj, uncompr.obj, infback.obj\
       deflate.obj, trees.obj, zutil.obj, inflate.obj, \
       inftrees.obj, inffast.obj

$ eod
$ write out "CFLAGS=", ccopt
$ write out "LOPTS=", lopts
$ copy sys$input: out
$ deck

all : example.exe minigzip.exe libz.olb
    @ write sys$output " Example applications available"

libz.olb : libz.olb$(OBJS)
    @ write sys$output " libz available"

example.exe : example.obj libz.olb
    link $(LOPTS) example,libz.olb/lib

```

```
minigzip.exe : minigzip.obj libz.olb
              link $(LOPTS) minigzip,libz.olb/lib,xllvms:xvmsutils.olb/lib
```

```
clean :
      delete *.obj;*,libz.olb;*,*.opt;*,*.exe;*
```

```
# Other dependencies.
```

```
adler32.obj  : adler32.c zutil.h zlib.h zconf.h
compress.obj : compress.c zlib.h zconf.h
crc32.obj    : crc32.c zutil.h zlib.h zconf.h
deflate.obj  : deflate.c deflate.h zutil.h zlib.h zconf.h
example.obj  : example.c zlib.h zconf.h
gzio.obj     : gzio.c zutil.h zlib.h zconf.h
inffast.obj  : inffast.c zutil.h zlib.h zconf.h inftrees.h inffast.h
inflate.obj   : inflate.c zutil.h zlib.h zconf.h
inftrees.obj : inftrees.c zutil.h zlib.h zconf.h inftrees.h
minigzip.obj  : minigzip.c zlib.h zconf.h
trees.obj    : trees.c deflate.h zutil.h zlib.h zconf.h
uncompr.obj  : uncompr.c zlib.h zconf.h
zutil.obj    : zutil.c zutil.h zlib.h zconf.h
inffast.obj  : inffast.c zutil.h inftrees.h inflate.h inffast.h inffixed.h
```

```
$ eod
```

```
$ close out
```

```
$ return
```

```
$!-----
$!
```

```
$! Read list of core library sources from makefile.in and create options
```

```
$! needed to build shareable image
```

```
$!
```

```
$CREA_OLIST:
```

```
$ open/read min makefile.in
```

```
$ open/write mod modules.opt
```

```
$ src_check = "OBSJ ="
```

```
$MRLOOP:
```

```
$ read/end=mrdone min rec
```

```
$ if (f$extract(0,6,rec) .nes. src_check) then goto mrloop
```

```
$ rec = rec - src_check
```

```
$ gosub extra_filnam
```

```
$ if (f$element(1,"\",rec) .eqs. "\") then goto mrdone
```

```
$MRSLOOP:
```

```
$ read/end=mrdone min rec
```

```
$ gosub extra_filnam
```

```
$ if (f$element(1,"\",rec) .nes. "\") then goto mrsloop
```

```
$MRDONE:
```

```
$ close min
```

```
$ close mod
```

```
$ return
```

```
$!-----
$!
```

```
$! Take record extracted in crea_olist and split it into single filenames
```

```
$!
```

```
$EXTRA_FILNAM:
```

```
$ myrec = f$edit(rec - "\", "trim,compress")
```

```
$ i = 0
```

```
$FELOOP:
```

```
$ srcfil = f$element(i, " ", myrec)
```

```
$ if (srcfil .nes. " ")
```

```
$ then
```

```
$   write mod f$parse(srcfil,,, "NAME"), ".obj"
```

```
$   i = i + 1
```

```
$   goto feloop
```

```
$ endif
```

```
$ return
```

```
$!-----
$!
```

```
$! Find current Zlib version number
```

```
$!
```

```
$FIND_VERSION:
```

```
$ open/read h_in 'v_file'
```

```
$hloop:
```

```
$ read/end=hdone h_in rec
```

```
$ rec = f$edit(rec,"TRIM")
```

```

$ if (f$extract(0,1,rec) .nes. "#") then goto hloop
$ rec = f$edit(rec - "#", "TRIM")
$ if f$element(0," ",rec) .nes. "define" then goto hloop
$ if f$element(1," ",rec) .eqs. v_string
$ then
$   version = 'f$element(2," ",rec)'
$   goto hdone
$ endif
$ goto hloop
$ hdone:
$ close h_in
$ return
$!-----
$!
$! Analyze Object files for OpenVMS AXP to extract Procedure and Data
$! information to build a symbol vector for a shareable image
$! All the "brains" of this logic was suggested by Hartmut Becker
$! (Hartmut.Becker@compaq.com). All the bugs were introduced by me
$! (zinser@decus.de), so if you do have problem reports please do not
$! bother Hartmut/HP, but get in touch with me
$!
$ ANAL_OBJ_AXP: Subroutine
$ V = 'F$Verify(0)
$ SAY := "WRITE_ SYS$OUTPUT"
$
$ IF F$SEARCH("''P1'") .EQS. ""
$ THEN
$   SAY "ANAL_OBJ_AXP-E-NOSUCHFILE: Error, inputfile ''p1' not available"
$   goto exit_aa
$ ENDIF
$ IF "''P2'" .EQS. ""
$ THEN
$   SAY "ANAL_OBJ_AXP: Error, no output file provided"
$   goto exit_aa
$ ENDIF
$
$ open/read in 'p1
$ create a.tmp
$ open/append atmp a.tmp
$ loop:
$ read/end=end_loop in line
$ f= f$search(line)
$ if f .eqs. ""
$ then
$   write sys$output "ANAL_OBJ_AXP-w-nosuchfile, ''line'"
$   goto loop
$ endif
$ define/user sys$output nl:
$ define/user sys$error nl:
$ anal/obj/gsd 'f /out=x.tmp
$ open/read xtmp x.tmp
$ XLOOP:
$ read/end=end_xloop xtmp xline
$ xline = f$edit(xline,"compress")
$ write atmp xline
$ goto xloop
$ END_XLOOP:
$ close xtmp
$ goto loop
$ end_loop:
$ close in
$ close atmp
$ if f$search("a.tmp") .eqs. "" -
$   then $ exit
$ ! all global definitions
$ search a.tmp "symbol:", "EGSY$V_DEF 1", "EGSY$V_NORM 1"/out=b.tmp
$ ! all procedures
$ search b.tmp "EGSY$V_NORM 1"/wind=(0,1) /out=c.tmp
$ search c.tmp "symbol:"/out=d.tmp
$ define/user sys$output nl:
$ edito/edt/command=sys$input d.tmp
sub/symbol: "/symbol_vector=(/whole
sub/"/=PROCEDURE)/whole
exit

```



```
$ ! all data
$ search b.tmp "EGSY$V_DEF 1"/wind=(0,1) /out=e.tmp
$ search e.tmp "symbol:"/out=f.tmp
$ define/user sys$output nl:
$ edito/edt/command=sys$input f.tmp
sub/symbol: "/symbol_vector=(/whole
sub/"/=DATA)/whole
exit
$ sort/nodupl d.tmp,f.tmp 'p2'
$ delete a.tmp;*,b.tmp;*,c.tmp;*,d.tmp;*,e.tmp;*,f.tmp;*
$ if f$search("x.tmp") .nes. "" -
    then $ delete x.tmp;*
$!
$ EXIT_AA:
$ if V then set verify
$ endsubroutine
$!-----
```

```

/* trees.c -- output deflated data using Huffman coding
* Copyright (C) 1995-2005 Jean-loup Gailly
* For conditions of distribution and use, see copyright notice in zlib.h
*/

/*
*   ALGORITHM
*
*       The "deflation" process uses several Huffman trees. The more
*       common source values are represented by shorter bit sequences.
*
*       Each code tree is stored in a compressed form which is itself
*       a Huffman encoding of the lengths of all the code strings (in
*       ascending order by source values). The actual code strings are
*       reconstructed from the lengths in the inflate process, as described
*       in the deflate specification.
*
*   REFERENCES
*
*       Deutsch, L.P., "'Deflate' Compressed Data Format Specification".
*       Available in ftp.uu.net:/pub/archiving/zip/doc/deflate-1.1.doc
*
*       Storer, James A.
*       Data Compression: Methods and Theory, pp. 49-50.
*       Computer Science Press, 1988. ISBN 0-7167-8156-5.
*
*       Sedgewick, R.
*       Algorithms, p290.
*       Addison-Wesley, 1983. ISBN 0-201-06672-6.
*/

/* @(#) $Id: trees.c,v 1.1 2005/09/23 22:39:00 beng Exp $ */

/* #define GEN_TREES_H */

#include "deflate.h"

#ifdef DEBUG
# include <ctype.h>
#endif

/* =====
* Constants
*/

#define MAX_BL_BITS 7
/* Bit length codes must not exceed MAX_BL_BITS bits */

#define END_BLOCK 256
/* end of block literal code */

#define REP_3_6 16
/* repeat previous bit length 3-6 times (2 bits of repeat count) */

#define REPZ_3_10 17
/* repeat a zero length 3-10 times (3 bits of repeat count) */

#define REPZ_11_138 18
/* repeat a zero length 11-138 times (7 bits of repeat count) */

local const int extra_lbits[LENGTH_CODES] /* extra bits for each length code */
    = {0,0,0,0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,0};

local const int extra_dbits[D_CODES] /* extra bits for each distance code */
    = {0,0,0,0,1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,12,12,13,13};

local const int extra_blbits[BL_CODES] /* extra bits for each bit length code */
    = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,3,7};

local const uch bl_order[BL_CODES]
    = {16,17,18,0,8,7,9,6,10,5,11,4,12,3,13,2,14,1,15};
/* The lengths of the bit length codes are sent in order of decreasing
* probability, to avoid transmitting the lengths for unused bit length codes.
*/

```

```

#define Buf_size (8 * 2*sizeof(char))
/* Number of bits used within bi_buf. (bi_buf might be implemented on
 * more than 16 bits on some systems.)
 */

/* =====
 * Local data. These are initialized only once.
 */

#define DIST_CODE_LEN 512 /* see definition of array dist_code below */

#if defined(GEN_TREES_H) || !defined(STDC)
/* non ANSI compilers may not accept trees.h */

local ct_data static_ltree[L_CODES+2];
/* The static literal tree. Since the bit lengths are imposed, there is no
 * need for the L_CODES extra codes used during heap construction. However
 * The codes 286 and 287 are needed to build a canonical tree (see _tr_init
 * below).
 */

local ct_data static_dtree[D_CODES];
/* The static distance tree. (Actually a trivial tree since all codes use
 * 5 bits.)
 */

uch _dist_code[DIST_CODE_LEN];
/* Distance codes. The first 256 values correspond to the distances
 * 3 .. 258, the last 256 values correspond to the top 8 bits of
 * the 15 bit distances.
 */

uch _length_code[MAX_MATCH-MIN_MATCH+1];
/* length code for each normalized match length (0 == MIN_MATCH) */

local int base_length[LENGTH_CODES];
/* First normalized length for each code (0 = MIN_MATCH) */

local int base_dist[D_CODES];
/* First normalized distance for each code (0 = distance of 1) */

#else
#include "trees.h"
#endif /* GEN_TREES_H */

struct static_tree_desc_s {
    const ct_data *static_tree; /* static tree or NULL */
    const intf *extra_bits; /* extra bits for each code or NULL */
    int extra_base; /* base index for extra_bits */
    int elems; /* max number of elements in the tree */
    int max_length; /* max bit length for the codes */
};

local static_tree_desc static_l_desc =
{static_ltree, extra_lbits, LITERALS+1, L_CODES, MAX_BITS};

local static_tree_desc static_d_desc =
{static_dtree, extra_dbits, 0, D_CODES, MAX_BITS};

local static_tree_desc static_bl_desc =
{((const ct_data *)0), extra_blbits, 0, BL_CODES, MAX_BL_BITS};

/* =====
 * Local (static) routines in this file.
 */

local void tr_static_init OF((void));
local void init_block OF((deflate_state *s));
local void pqdownheap OF((deflate_state *s, ct_data *tree, int k));
local void gen_bitlen OF((deflate_state *s, tree_desc *desc));
local void gen_codes OF((ct_data *tree, int max_code, ushf *bl_count));
local void build_tree OF((deflate_state *s, tree_desc *desc));
local void scan_tree OF((deflate_state *s, ct_data *tree, int max_code));

```

```

local void send_tree      OF((deflate_state *s, ct_data *tree, int max_code));
local int  build_bl_tree  OF((deflate_state *s));
local void send_all_trees OF((deflate_state *s, int lcodes, int dcodes,
                             int blcodes));
local void compress_block OF((deflate_state *s, ct_data *ltree,
                             ct_data *dtree));
local void set_data_type  OF((deflate_state *s));
local unsigned bi_reverse OF((unsigned value, int length));
local void bi_windup      OF((deflate_state *s));
local void bi_flush       OF((deflate_state *s));
local void copy_block     OF((deflate_state *s, charf *buf, unsigned len,
                             int header));

#ifdef GEN_TREES_H
local void gen_trees_header OF((void));
#endif

#ifdef DEBUG
# define send_code(s, c, tree) send_bits(s, tree[c].Code, tree[c].Len)
/* Send a code of the given tree. c and tree must not have side effects */

#else /* DEBUG */
# define send_code(s, c, tree) \
    { if (z_verbose>2) fprintf(stderr, "\ncd %3d ", (c)); \
      send_bits(s, tree[c].Code, tree[c].Len); }
#endif

/* =====
 * Output a short LSB first on the stream.
 * IN assertion: there is enough room in pendingBuf.
 */
#define put_short(s, w) { \
    put_byte(s, (uch)((w) & 0xff)); \
    put_byte(s, (uch)((ush)(w) >> 8)); \
}

/* =====
 * Send a value on a given number of bits.
 * IN assertion: length <= 16 and value fits in length bits.
 */
#ifdef DEBUG
local void send_bits      OF((deflate_state *s, int value, int length));

local void send_bits(s, value, length)
    deflate_state *s;
    int value; /* value to send */
    int length; /* number of bits */
{
    Tracevv((stderr, "l%2d v%4x ", length, value));
    Assert(length > 0 && length <= 15, "invalid length");
    s->bits_sent += (ulg)length;

    /* If not enough room in bi_buf, use (valid) bits from bi_buf and
     * (16 - bi_valid) bits from value, leaving (width - (16-bi_valid))
     * unused bits in value.
     */
    if (s->bi_valid > (int)Buf_size - length) {
        s->bi_buf |= (value << s->bi_valid);
        put_short(s, s->bi_buf);
        s->bi_buf = (ush)value >> (Buf_size - s->bi_valid);
        s->bi_valid += length - Buf_size;
    } else {
        s->bi_buf |= value << s->bi_valid;
        s->bi_valid += length;
    }
}
#else /* !DEBUG */
#define send_bits(s, value, length) \
{ int len = length; \
  if (s->bi_valid > (int)Buf_size - len) { \
    int val = value; \
    s->bi_buf |= (val << s->bi_valid); \
    put_short(s, s->bi_buf); \
  }

```

```

    s->bi_buf = (ush)val >> (Buf_size - s->bi_valid);\
    s->bi_valid += len - Buf_size;\
} else {\
    s->bi_buf |= (value) << s->bi_valid;\
    s->bi_valid += len;\
}\
}\
#endif /* DEBUG */

/* the arguments must not have side effects */

/* =====
 * Initialize the various 'constant' tables.
 */
local void tr_static_init()
{
#ifdef GEN_TREES_H || !defined(STDC)
    static int static_init_done = 0;
    int n;          /* iterates over tree elements */
    int bits;       /* bit counter */
    int length;     /* length value */
    int code;       /* code value */
    int dist;       /* distance index */
    ush bl_count[MAX_BITS+1];
    /* number of codes at each bit length for an optimal tree */

    if (static_init_done) return;

    /* For some embedded targets, global variables are not initialized: */
    static_l_desc.static_tree = static_ltree;
    static_l_desc.extra_bits = extra_lbits;
    static_d_desc.static_tree = static_dtree;
    static_d_desc.extra_bits = extra_dbits;
    static_bl_desc.extra_bits = extra_blbits;

    /* Initialize the mapping length (0..255) -> length code (0..28) */
    length = 0;
    for (code = 0; code < LENGTH_CODES-1; code++) {
        base_length[code] = length;
        for (n = 0; n < (1<<extra_lbits[code]); n++) {
            _length_code[length++] = (uch)code;
        }
    }
    Assert (length == 256, "tr_static_init: length != 256");
    /* Note that the length 255 (match length 258) can be represented
     * in two different ways: code 284 + 5 bits or code 285, so we
     * overwrite length_code[255] to use the best encoding:
     */
    _length_code[length-1] = (uch)code;

    /* Initialize the mapping dist (0..32K) -> dist code (0..29) */
    dist = 0;
    for (code = 0 ; code < 16; code++) {
        base_dist[code] = dist;
        for (n = 0; n < (1<<extra_dbits[code]); n++) {
            _dist_code[dist++] = (uch)code;
        }
    }
    Assert (dist == 256, "tr_static_init: dist != 256");
    dist >>= 7; /* from now on, all distances are divided by 128 */
    for ( ; code < D_CODES; code++) {
        base_dist[code] = dist << 7;
        for (n = 0; n < (1<<(extra_dbits[code]-7)); n++) {
            _dist_code[256 + dist++] = (uch)code;
        }
    }
    Assert (dist == 256, "tr_static_init: 256+dist != 512");

    /* Construct the codes of the static literal tree */
    for (bits = 0; bits <= MAX_BITS; bits++) bl_count[bits] = 0;
    n = 0;
    while (n <= 143) static_ltree[n++].Len = 8, bl_count[8]++;
    while (n <= 255) static_ltree[n++].Len = 9, bl_count[9]++;

```

```

while (n <= 279) static_ltree[n++].Len = 7, bl_count[7]++;
while (n <= 287) static_ltree[n++].Len = 8, bl_count[8]++;
/* Codes 286 and 287 do not exist, but we must include them in the
 * tree construction to get a canonical Huffman tree (longest code
 * all ones)
 */
gen_codes((ct_data *)static_ltree, L_CODES+1, bl_count);

/* The static distance tree is trivial: */
for (n = 0; n < D_CODES; n++) {
    static_dtree[n].Len = 5;
    static_dtree[n].Code = bi_reverse((unsigned)n, 5);
}
static_init_done = 1;

# ifdef GEN_TREES_H
    gen_trees_header();
# endif
#endif /* defined(GEN_TREES_H) || !defined(STDC) */
}

/* =====
 * Generate the file trees.h describing the static trees.
 */
#ifdef GEN_TREES_H
# ifndef DEBUG
#   include <stdio.h>
# endif

# define SEPARATOR(i, last, width) \
    ((i) == (last)? "\n;\n\n" : \
     ((i) % (width) == (width)-1 ? ",\n" : ", "))

void gen_trees_header()
{
    FILE *header = fopen("trees.h", "w");
    int i;

    Assert (header != NULL, "Can't open trees.h");
    fprintf(header,
        "/* header created automatically with -DGEN_TREES_H */\n\n");

    fprintf(header, "local const ct_data static_ltree[L_CODES+2] = {\n");
    for (i = 0; i < L_CODES+2; i++) {
        fprintf(header, "{{%3u},{%3u}}%s", static_ltree[i].Code,
            static_ltree[i].Len, SEPARATOR(i, L_CODES+1, 5));
    }

    fprintf(header, "local const ct_data static_dtree[D_CODES] = {\n");
    for (i = 0; i < D_CODES; i++) {
        fprintf(header, "{{%2u},{%2u}}%s", static_dtree[i].Code,
            static_dtree[i].Len, SEPARATOR(i, D_CODES-1, 5));
    }

    fprintf(header, "const uch _dist_code[DIST_CODE_LEN] = {\n");
    for (i = 0; i < DIST_CODE_LEN; i++) {
        fprintf(header, "%2u%s", _dist_code[i],
            SEPARATOR(i, DIST_CODE_LEN-1, 20));
    }

    fprintf(header, "const uch _length_code[MAX_MATCH-MIN_MATCH+1]= {\n");
    for (i = 0; i < MAX_MATCH-MIN_MATCH+1; i++) {
        fprintf(header, "%2u%s", _length_code[i],
            SEPARATOR(i, MAX_MATCH-MIN_MATCH, 20));
    }

    fprintf(header, "local const int base_length[LENGTH_CODES] = {\n");
    for (i = 0; i < LENGTH_CODES; i++) {
        fprintf(header, "%1u%s", base_length[i],
            SEPARATOR(i, LENGTH_CODES-1, 20));
    }

    fprintf(header, "local const int base_dist[D_CODES] = {\n");
    for (i = 0; i < D_CODES; i++) {

```

```

        fprintf(header, "%5u%s", base_dist[i],
            SEPARATOR(i, D_CODES-1, 10));
    }

    fclose(header);
}
#endif /* GEN_TREES_H */

/* =====
 * Initialize the tree data structures for a new zlib stream.
 */
void _tr_init(s)
    deflate_state *s;
{
    tr_static_init();

    s->l_desc.dyn_tree = s->dyn_ltree;
    s->l_desc.stat_desc = &static_l_desc;

    s->d_desc.dyn_tree = s->dyn_dtree;
    s->d_desc.stat_desc = &static_d_desc;

    s->bl_desc.dyn_tree = s->bl_tree;
    s->bl_desc.stat_desc = &static_bl_desc;

    s->bi_buf = 0;
    s->bi_valid = 0;
    s->last_eob_len = 8; /* enough lookahead for inflate */
#ifdef DEBUG
    s->compressed_len = 0L;
    s->bits_sent = 0L;
#endif

    /* Initialize the first block of the first file: */
    init_block(s);
}

/* =====
 * Initialize a new block.
 */
local void init_block(s)
    deflate_state *s;
{
    int n; /* iterates over tree elements */

    /* Initialize the trees. */
    for (n = 0; n < L_CODES; n++) s->dyn_ltree[n].Freq = 0;
    for (n = 0; n < D_CODES; n++) s->dyn_dtree[n].Freq = 0;
    for (n = 0; n < BL_CODES; n++) s->bl_tree[n].Freq = 0;

    s->dyn_ltree[END_BLOCK].Freq = 1;
    s->opt_len = s->static_len = 0L;
    s->last_lit = s->matches = 0;
}

#define SMALLEST 1
/* Index within the heap array of least frequent node in the Huffman tree */

/* =====
 * Remove the smallest element from the heap and recreate the heap with
 * one less element. Updates heap and heap_len.
 */
#define pqremove(s, tree, top) \
{ \
    top = s->heap[SMALLEST]; \
    s->heap[SMALLEST] = s->heap[s->heap_len--]; \
    pqdownheap(s, tree, SMALLEST); \
}

/* =====
 * Compares to subtrees, using the tree depth as tie breaker when
 * the subtrees have equal frequency. This minimizes the worst case length.
 */

```

```

#define smaller(tree, n, m, depth) \
    (tree[n].Freq < tree[m].Freq || \
    (tree[n].Freq == tree[m].Freq && depth[n] <= depth[m]))

/* =====
 * Restore the heap property by moving down the tree starting at node k,
 * exchanging a node with the smallest of its two sons if necessary, stopping
 * when the heap property is re-established (each father smaller than its
 * two sons).
 */
local void pqdownheap(s, tree, k)
    deflate_state *s;
    ct_data *tree; /* the tree to restore */
    int k;          /* node to move down */
{
    int v = s->heap[k];
    int j = k << 1; /* left son of k */
    while (j <= s->heap_len) {
        /* Set j to the smallest of the two sons: */
        if (j < s->heap_len &&
            smaller(tree, s->heap[j+1], s->heap[j], s->depth)) {
            j++;
        }
        /* Exit if v is smaller than both sons */
        if (smaller(tree, v, s->heap[j], s->depth)) break;

        /* Exchange v with the smallest son */
        s->heap[k] = s->heap[j]; k = j;

        /* And continue down the tree, setting j to the left son of k */
        j <= 1;
    }
    s->heap[k] = v;
}

/* =====
 * Compute the optimal bit lengths for a tree and update the total bit length
 * for the current block.
 * IN assertion: the fields freq and dad are set, heap[heap_max] and
 * above are the tree nodes sorted by increasing frequency.
 * OUT assertions: the field len is set to the optimal bit length, the
 * array bl_count contains the frequencies for each bit length.
 * The length opt_len is updated; static_len is also updated if stree is
 * not null.
 */
local void gen_bitlen(s, desc)
    deflate_state *s;
    tree_desc *desc; /* the tree descriptor */
{
    ct_data *tree      = desc->dyn_tree;
    int max_code       = desc->max_code;
    const ct_data *stree = desc->stat_desc->static_tree;
    const intf *extra   = desc->stat_desc->extra_bits;
    int base           = desc->stat_desc->extra_base;
    int max_length     = desc->stat_desc->max_length;
    int h;             /* heap index */
    int n, m;          /* iterate over the tree elements */
    int bits;           /* bit length */
    int xbits;          /* extra bits */
    ush f;             /* frequency */
    int overflow = 0;   /* number of elements with bit length too large */

    for (bits = 0; bits <= MAX_BITS; bits++) s->bl_count[bits] = 0;

    /* In a first pass, compute the optimal bit lengths (which may
     * overflow in the case of the bit length tree).
     */
    tree[s->heap[s->heap_max]].Len = 0; /* root of the heap */

    for (h = s->heap_max+1; h < HEAP_SIZE; h++) {
        n = s->heap[h];
        bits = tree[tree[n].Dad].Len + 1;
        if (bits > max_length) bits = max_length, overflow++;
        tree[n].Len = (ush)bits;
    }
}

```



```

    /* We overwrite tree[n].Dad which is no longer needed */

    if (n > max_code) continue; /* not a leaf node */

    s->bl_count[bits]++;
    xbits = 0;
    if (n >= base) xbits = extra[n-base];
    f = tree[n].Freq;
    s->opt_len += (ulg)f * (bits + xbits);
    if (stree) s->static_len += (ulg)f * (stree[n].Len + xbits);
}
if (overflow == 0) return;

Trace((stderr, "\nbit length overflow\n"));
/* This happens for example on obj2 and pic of the Calgary corpus */

/* Find the first bit length which could increase: */
do {
    bits = max_length-1;
    while (s->bl_count[bits] == 0) bits--;
    s->bl_count[bits]--; /* move one leaf down the tree */
    s->bl_count[bits+1] += 2; /* move one overflow item as its brother */
    s->bl_count[max_length]--;
    /* The brother of the overflow item also moves one step up,
     * but this does not affect bl_count[max_length]
     */
    overflow -= 2;
} while (overflow > 0);

/* Now recompute all bit lengths, scanning in increasing frequency.
 * h is still equal to HEAP_SIZE. (It is simpler to reconstruct all
 * lengths instead of fixing only the wrong ones. This idea is taken
 * from 'ar' written by Haruhiko Okumura.)
 */
for (bits = max_length; bits != 0; bits--) {
    n = s->bl_count[bits];
    while (n != 0) {
        m = s->heap[--h];
        if (m > max_code) continue;
        if ((unsigned) tree[m].Len != (unsigned) bits) {
            Trace((stderr, "code %d bits %d->%d\n", m, tree[m].Len, bits));
            s->opt_len += ((long)bits - (long)tree[m].Len)
                * (long)tree[m].Freq;
            tree[m].Len = (ush)bits;
        }
        n--;
    }
}

/* =====
 * Generate the codes for a given tree and bit counts (which need not be
 * optimal).
 * IN assertion: the array bl_count contains the bit length statistics for
 * the given tree and the field len is set for all tree elements.
 * OUT assertion: the field code is set for all tree elements of non
 * zero code length.
 */
local void gen_codes (tree, max_code, bl_count)
    ct_data *tree; /* the tree to decorate */
    int max_code; /* largest code with non zero frequency */
    ushf *bl_count; /* number of codes at each bit length */
{
    ush next_code[MAX_BITS+1]; /* next code value for each bit length */
    ush code = 0; /* running code value */
    int bits; /* bit index */
    int n; /* code index */

    /* The distribution counts are first used to generate the code values
     * without bit reversal.
     */
    for (bits = 1; bits <= MAX_BITS; bits++) {
        next_code[bits] = code = (code + bl_count[bits-1]) << 1;
    }
}

```

```

/* Check that the bit counts in bl_count are consistent. The last code
 * must be all ones.
 */
Assert (code + bl_count[MAX_BITS]-1 == (1<<MAX_BITS)-1,
        "inconsistent bit counts");
Tracev((stderr, "\ngen_codes: max_code %d ", max_code));

for (n = 0; n <= max_code; n++) {
    int len = tree[n].Len;
    if (len == 0) continue;
    /* Now reverse the bits */
    tree[n].Code = bi_reverse(next_code[len]++, len);

    Tracecv(tree != static_ltree, (stderr, "\nn %3d %c l %2d c %4x (%x) ",
        n, (isgraph(n) ? n : ' '), len, tree[n].Code, next_code[len]-1));
}
}

/* =====
 * Construct one Huffman tree and assigns the code bit strings and lengths.
 * Update the total bit length for the current block.
 * IN assertion: the field freq is set for all tree elements.
 * OUT assertions: the fields len and code are set to the optimal bit length
 * and corresponding code. The length opt_len is updated; static_len is
 * also updated if stree is not null. The field max_code is set.
 */
local void build_tree(s, desc)
    deflate_state *s;
    tree_desc *desc; /* the tree descriptor */
{
    ct_data *tree        = desc->dyn_tree;
    const ct_data *stree = desc->stat_desc->static_tree;
    int elems            = desc->stat_desc->elems;
    int n, m;            /* iterate over heap elements */
    int max_code = -1; /* largest code with non zero frequency */
    int node;           /* new node being created */

    /* Construct the initial heap, with least frequent element in
     * heap[SMALLEST]. The sons of heap[n] are heap[2*n] and heap[2*n+1].
     * heap[0] is not used.
     */
    s->heap_len = 0, s->heap_max = HEAP_SIZE;

    for (n = 0; n < elems; n++) {
        if (tree[n].Freq != 0) {
            s->heap[++(s->heap_len)] = max_code = n;
            s->depth[n] = 0;
        } else {
            tree[n].Len = 0;
        }
    }

    /* The pkzip format requires that at least one distance code exists,
     * and that at least one bit should be sent even if there is only one
     * possible code. So to avoid special checks later on we force at least
     * two codes of non zero frequency.
     */
    while (s->heap_len < 2) {
        node = s->heap[++(s->heap_len)] = (max_code < 2 ? ++max_code : 0);
        tree[node].Freq = 1;
        s->depth[node] = 0;
        s->opt_len--; if (stree) s->static_len -= stree[node].Len;
        /* node is 0 or 1 so it does not have extra bits */
    }
    desc->max_code = max_code;

    /* The elements heap[heap_len/2+1 .. heap_len] are leaves of the tree,
     * establish sub-heaps of increasing lengths:
     */
    for (n = s->heap_len/2; n >= 1; n--) pqdownheap(s, tree, n);

    /* Construct the Huffman tree by repeatedly combining the least two
     * frequent nodes.
     */

```

```

node = elems;                /* next internal node of the tree */
do {
    pqremove(s, tree, n); /* n = node of least frequency */
    m = s->heap[SMALLEST]; /* m = node of next least frequency */

    s->heap[--(s->heap_max)] = n; /* keep the nodes sorted by frequency */
    s->heap[--(s->heap_max)] = m;

    /* Create a new node father of n and m */
    tree[node].Freq = tree[n].Freq + tree[m].Freq;
    s->depth[node] = (uch)((s->depth[n] >= s->depth[m] ?
                           s->depth[n] : s->depth[m]) + 1);
    tree[n].Dad = tree[m].Dad = (ush)node;
#ifdef DUMP_BL_TREE
    if (tree == s->bl_tree) {
        fprintf(stderr, "\nnode %d(%d), sons %d(%d) %d(%d)",
            node, tree[node].Freq, n, tree[n].Freq, m, tree[m].Freq);
    }
#endif
    /* and insert the new node in the heap */
    s->heap[SMALLEST] = node++;
    pqdownheap(s, tree, SMALLEST);

} while (s->heap_len >= 2);

s->heap[--(s->heap_max)] = s->heap[SMALLEST];

/* At this point, the fields freq and dad are set. We can now
 * generate the bit lengths.
 */
gen_bitlen(s, (tree_desc *)desc);

/* The field len is now set, we can generate the bit codes */
gen_codes ((ct_data *)tree, max_code, s->bl_count);
}

/* =====
 * Scan a literal or distance tree to determine the frequencies of the codes
 * in the bit length tree.
 */
local void scan_tree (s, tree, max_code)
    deflate_state *s;
    ct_data *tree; /* the tree to be scanned */
    int max_code; /* and its largest code of non zero frequency */
{
    int n; /* iterates over all tree elements */
    int prevlen = -1; /* last emitted length */
    int curlen; /* length of current code */
    int nextlen = tree[0].Len; /* length of next code */
    int count = 0; /* repeat count of the current code */
    int max_count = 7; /* max repeat count */
    int min_count = 4; /* min repeat count */

    if (nextlen == 0) max_count = 138, min_count = 3;
    tree[max_code+1].Len = (ush)0xffff; /* guard */

    for (n = 0; n <= max_code; n++) {
        curlen = nextlen; nextlen = tree[n+1].Len;
        if (++count < max_count && curlen == nextlen) {
            continue;
        } else if (count < min_count) {
            s->bl_tree[curlen].Freq += count;
        } else if (curlen != 0) {
            if (curlen != prevlen) s->bl_tree[curlen].Freq++;
            s->bl_tree[REP_3_6].Freq++;
        } else if (count <= 10) {
            s->bl_tree[REPZ_3_10].Freq++;
        } else {
            s->bl_tree[REPZ_11_138].Freq++;
        }
        count = 0; prevlen = curlen;
        if (nextlen == 0) {
            max_count = 138, min_count = 3;
        } else if (curlen == nextlen) {

```

```

        max_count = 6, min_count = 3;
    } else {
        max_count = 7, min_count = 4;
    }
}

/* =====
 * Send a literal or distance tree in compressed form, using the codes in
 * bl_tree.
 */
local void send_tree (s, tree, max_code)
    deflate_state *s;
    ct_data *tree; /* the tree to be scanned */
    int max_code; /* and its largest code of non zero frequency */
{
    int n; /* iterates over all tree elements */
    int prevlen = -1; /* last emitted length */
    int curlen; /* length of current code */
    int nextlen = tree[0].Len; /* length of next code */
    int count = 0; /* repeat count of the current code */
    int max_count = 7; /* max repeat count */
    int min_count = 4; /* min repeat count */

    /* tree[max_code+1].Len = -1; */ /* guard already set */
    if (nextlen == 0) max_count = 138, min_count = 3;

    for (n = 0; n <= max_code; n++) {
        curlen = nextlen; nextlen = tree[n+1].Len;
        if (++count < max_count && curlen == nextlen) {
            continue;
        } else if (count < min_count) {
            do { send_code(s, curlen, s->bl_tree); } while (--count != 0);
        } else if (curlen != 0) {
            if (curlen != prevlen) {
                send_code(s, curlen, s->bl_tree); count--;
            }
            Assert(count >= 3 && count <= 6, "3_6?");
            send_code(s, REP_3_6, s->bl_tree); send_bits(s, count-3, 2);
        } else if (count <= 10) {
            send_code(s, REPZ_3_10, s->bl_tree); send_bits(s, count-3, 3);
        } else {
            send_code(s, REPZ_11_138, s->bl_tree); send_bits(s, count-11, 7);
        }
        count = 0; prevlen = curlen;
        if (nextlen == 0) {
            max_count = 138, min_count = 3;
        } else if (curlen == nextlen) {
            max_count = 6, min_count = 3;
        } else {
            max_count = 7, min_count = 4;
        }
    }
}

/* =====
 * Construct the Huffman tree for the bit lengths and return the index in
 * bl_order of the last bit length code to send.
 */
local int build_bl_tree(s)
    deflate_state *s;
{
    int max_blindex; /* index of last bit length code of non zero freq */

    /* Determine the bit length frequencies for literal and distance trees */
    scan_tree(s, (ct_data *)s->dyn_ltree, s->l_desc.max_code);
    scan_tree(s, (ct_data *)s->dyn_dtree, s->d_desc.max_code);

    /* Build the bit length tree: */
    build_tree(s, (tree_desc *)&(s->bl_desc));
    /* opt_len now includes the length of the tree representations, except

```

```

    * the lengths of the bit lengths codes and the 5+5+4 bits for the counts.
    */

/* Determine the number of bit length codes to send. The pkzip format
 * requires that at least 4 bit length codes be sent. (appnote.txt says
 * 3 but the actual value used is 4.)
 */
for (max_blinindex = BL_CODES-1; max_blinindex >= 3; max_blinindex--) {
    if (s->bl_tree[bl_order[max_blinindex]].Len != 0) break;
}
/* Update opt_len to include the bit length tree and counts */
s->opt_len += 3*(max_blinindex+1) + 5+5+4;
Tracev((stderr, "\ndyn trees: dyn %ld, stat %ld",
        s->opt_len, s->static_len));

return max_blinindex;
}

/* =====
 * Send the header for a block using dynamic Huffman trees: the counts, the
 * lengths of the bit length codes, the literal tree and the distance tree.
 * IN assertion: lcodes >= 257, dcodes >= 1, blcodes >= 4.
 */
local void send_all_trees(s, lcodes, dcodes, blcodes)
    deflate_state *s;
    int lcodes, dcodes, blcodes; /* number of codes for each tree */
{
    int rank; /* index in bl_order */

    Assert (lcodes >= 257 && dcodes >= 1 && blcodes >= 4, "not enough codes");
    Assert (lcodes <= L_CODES && dcodes <= D_CODES && blcodes <= BL_CODES,
            "too many codes");
    Tracev((stderr, "\nbl counts: "));
    send_bits(s, lcodes-257, 5); /* not +255 as stated in appnote.txt */
    send_bits(s, dcodes-1, 5);
    send_bits(s, blcodes-4, 4); /* not -3 as stated in appnote.txt */
    for (rank = 0; rank < blcodes; rank++) {
        Tracev((stderr, "\nbl code %2d ", bl_order[rank]));
        send_bits(s, s->bl_tree[bl_order[rank]].Len, 3);
    }
    Tracev((stderr, "\nbl tree: sent %ld", s->bits_sent));

    send_tree(s, (ct_data *)s->dyn_ltree, lcodes-1); /* literal tree */
    Tracev((stderr, "\nlit tree: sent %ld", s->bits_sent));

    send_tree(s, (ct_data *)s->dyn_dtree, dcodes-1); /* distance tree */
    Tracev((stderr, "\ndist tree: sent %ld", s->bits_sent));
}

/* =====
 * Send a stored block
 */
void _tr_stored_block(s, buf, stored_len, eof)
    deflate_state *s;
    charf *buf; /* input block */
    ulg stored_len; /* length of input block */
    int eof; /* true if this is the last block for a file */
{
    send_bits(s, (STORED_BLOCK<<1)+eof, 3); /* send block type */
#ifdef DEBUG
    s->compressed_len = (s->compressed_len + 3 + 7) & (ulg)~7L;
    s->compressed_len += (stored_len + 4) << 3;
#endif
    copy_block(s, buf, (unsigned)stored_len, 1); /* with header */
}

/* =====
 * Send one empty static block to give enough lookahead for inflate.
 * This takes 10 bits, of which 7 may remain in the bit buffer.
 * The current inflate code requires 9 bits of lookahead. If the
 * last two codes for the previous block (real code plus EOB) were coded
 * on 5 bits or less, inflate may have only 5+3 bits of lookahead to decode
 * the last real code. In this case we send two empty static blocks instead
 * of one. (There are no problems if the previous block is stored or fixed.)
 */

```

```

* To simplify the code, we assume the worst case of last real code encoded
* on one bit only.
*/
void _tr_align(s)
    deflate_state *s;
{
    send_bits(s, STATIC_TREES<<1, 3);
    send_code(s, END_BLOCK, static_ltree);
#ifdef DEBUG
    s->compressed_len += 10L; /* 3 for block type, 7 for EOB */
#endif
    bi_flush(s);
    /* Of the 10 bits for the empty block, we have already sent
     * (10 - bi_valid) bits. The lookahead for the last real code (before
     * the EOB of the previous block) was thus at least one plus the length
     * of the EOB plus what we have just sent of the empty static block.
     */
    if (1 + s->last_eob_len + 10 - s->bi_valid < 9) {
        send_bits(s, STATIC_TREES<<1, 3);
        send_code(s, END_BLOCK, static_ltree);
#ifdef DEBUG
        s->compressed_len += 10L;
#endif
    }
    bi_flush(s);
}
s->last_eob_len = 7;
}

/* =====
 * Determine the best encoding for the current block: dynamic trees, static
 * trees or store, and output the encoded block to the zip file.
 */
void _tr_flush_block(s, buf, stored_len, eof)
    deflate_state *s;
    charf *buf; /* input block, or NULL if too old */
    ulg stored_len; /* length of input block */
    int eof; /* true if this is the last block for a file */
{
    ulg opt_lenb, static_lenb; /* opt_len and static_len in bytes */
    int max_blindex = 0; /* index of last bit length code of non zero freq */

    /* Build the Huffman trees unless a stored block is forced */
    if (s->level > 0) {
        /* Check if the file is binary or text */
        if (stored_len > 0 && s->strm->data_type == Z_UNKNOWN)
            set_data_type(s);

        /* Construct the literal and distance trees */
        build_tree(s, (tree_desc *)&(s->l_desc));
        Tracev((stderr, "\nlit data: dyn %ld, stat %ld", s->opt_len,
            s->static_len));

        build_tree(s, (tree_desc *)&(s->d_desc));
        Tracev((stderr, "\ndist data: dyn %ld, stat %ld", s->opt_len,
            s->static_len));
        /* At this point, opt_len and static_len are the total bit lengths of
         * the compressed block data, excluding the tree representations.
         */

        /* Build the bit length tree for the above two trees, and get the index
         * in bl_order of the last bit length code to send.
         */
        max_blindex = build_bl_tree(s);

        /* Determine the best encoding. Compute the block lengths in bytes. */
        opt_lenb = (s->opt_len+3+7)>>3;
        static_lenb = (s->static_len+3+7)>>3;

        Tracev((stderr, "\nopt %lu(%lu) stat %lu(%lu) stored %lu lit %u ",
            opt_lenb, s->opt_len, static_lenb, s->static_len, stored_len,
            s->last_lit));

        if (static_lenb <= opt_lenb) opt_lenb = static_lenb;
    }
}

```

```

    } else {
        Assert(buf != (char*)0, "lost buf");
        opt_lenb = static_lenb = stored_len + 5; /* force a stored block */
    }

#ifdef FORCE_STORED
    if (buf != (char*)0) { /* force stored block */
#else
    if (stored_len+4 <= opt_lenb && buf != (char*)0) {
        /* 4: two words for the lengths */
#endif
    /* The test buf != NULL is only necessary if LIT_BUFSIZE > WSIZE.
     * Otherwise we can't have processed more than WSIZE input bytes since
     * the last block flush, because compression would have been
     * successful. If LIT_BUFSIZE <= WSIZE, it is never too late to
     * transform a block into a stored block.
     */
    _tr_stored_block(s, buf, stored_len, eof);

#ifdef FORCE_STATIC
    } else if (static_lenb >= 0) { /* force static trees */
#else
    } else if (s->strategy == Z_FIXED || static_lenb == opt_lenb) {
#endif
        send_bits(s, (STATIC_TREES<<1)+eof, 3);
        compress_block(s, (ct_data *)static_ltree, (ct_data *)static_dtree);
#ifdef DEBUG
        s->compressed_len += 3 + s->static_len;
#endif
    } else {
        send_bits(s, (DYN_TREES<<1)+eof, 3);
        send_all_trees(s, s->l_desc.max_code+1, s->d_desc.max_code+1,
            max_blindex+1);
        compress_block(s, (ct_data *)s->dyn_ltree, (ct_data *)s->dyn_dtree);
#ifdef DEBUG
        s->compressed_len += 3 + s->opt_len;
#endif
    }
    Assert (s->compressed_len == s->bits_sent, "bad compressed size");
    /* The above check is made mod 2^32, for files larger than 512 MB
     * and uLong implemented on 32 bits.
     */
    init_block(s);

    if (eof) {
        bi_windup(s);
#ifdef DEBUG
        s->compressed_len += 7; /* align on byte boundary */
#endif
    }
    Tracev((stderr, "\ncomprlen %lu(%lu) ", s->compressed_len>>3,
        s->compressed_len-7*eof));
}

/* =====
 * Save the match info and tally the frequency counts. Return true if
 * the current block must be flushed.
 */
int _tr_tally (s, dist, lc)
    deflate_state *s;
    unsigned dist; /* distance of matched string */
    unsigned lc; /* match length-MIN_MATCH or unmatched char (if dist==0) */
{
    s->d_buf[s->last_lit] = (ush)dist;
    s->l_buf[s->last_lit++] = (uch)lc;
    if (dist == 0) {
        /* lc is the unmatched char */
        s->dyn_ltree[lc].Freq++;
    } else {
        s->matches++;
        /* Here, lc is the match length - MIN_MATCH */
        dist--; /* dist = match distance - 1 */
        Assert((ush)dist < (ush)MAX_DIST(s) &&

```

```

        (ush)lc <= (ush)(MAX_MATCH-MIN_MATCH) &&
        (ush)d_code(dist) < (ush)D_CODES,  "_tr_tally: bad match");

    s->dyn_ltree[_length_code[lc]+LITERALS+1].Freq++;
    s->dyn_dtree[d_code(dist)].Freq++;
}

#ifdef TRUNCATE_BLOCK
/* Try to guess if it is profitable to stop the current block here */
if ((s->last_lit & 0xlfff) == 0 && s->level > 2) {
    /* Compute an upper bound for the compressed length */
    ulg out_length = (ulg)s->last_lit*8L;
    ulg in_length = (ulg)((long)s->strstart - s->block_start);
    int dcode;
    for (dcode = 0; dcode < D_CODES; dcode++) {
        out_length += (ulg)s->dyn_dtree[dcode].Freq *
            (5L+extra_dbits[dcode]);
    }
    out_length >>= 3;
    Tracev((stderr, "\nlast_lit %u, in %ld, out ~%ld(%ld%%) ",
        s->last_lit, in_length, out_length,
        100L - out_length*100L/in_length));
    if (s->matches < s->last_lit/2 && out_length < in_length/2) return 1;
}
#endif
return (s->last_lit == s->lit_bufsize-1);
/* We avoid equality with lit_bufsize because of wraparound at 64K
 * on 16 bit machines and because stored blocks are restricted to
 * 64K-1 bytes.
 */
}

/* =====
 * Send the block data compressed using the given Huffman trees
 */
local void compress_block(s, ltree, dtree)
    deflate_state *s;
    ct_data *ltree; /* literal tree */
    ct_data *dtree; /* distance tree */
{
    unsigned dist;      /* distance of matched string */
    int lc;             /* match length or unmatched char (if dist == 0) */
    unsigned lx = 0;    /* running index in l_buf */
    unsigned code;      /* the code to send */
    int extra;          /* number of extra bits to send */

    if (s->last_lit != 0) do {
        dist = s->d_buf[lx];
        lc = s->l_buf[lx++];
        if (dist == 0) {
            send_code(s, lc, ltree); /* send a literal byte */
            Tracev(isgraph(lc), (stderr, "%c", lc));
        } else {
            /* Here, lc is the match length - MIN_MATCH */
            code = _length_code[lc];
            send_code(s, code+LITERALS+1, ltree); /* send the length code */
            extra = extra_lbits[code];
            if (extra != 0) {
                lc -= base_length[code];
                send_bits(s, lc, extra); /* send the extra length bits */
            }
            dist--; /* dist is now the match distance - 1 */
            code = d_code(dist);
            Assert (code < D_CODES, "bad d_code");

            send_code(s, code, dtree); /* send the distance code */
            extra = extra_dbits[code];
            if (extra != 0) {
                dist -= base_dist[code];
                send_bits(s, dist, extra); /* send the extra distance bits */
            }
        } /* literal or match pair ? */

        /* Check that the overlay between pending_buf and d_buf+l_buf is ok: */
    }

```



```

    Assert((uInt)(s->pending) < s->lit_bufsize + 2*lx,
           "pendingBuf overflow");

    } while (lx < s->last_lit);

    send_code(s, END_BLOCK, ltree);
    s->last_eob_len = ltree[END_BLOCK].Len;
}

/* =====
 * Set the data type to BINARY or TEXT, using a crude approximation:
 * set it to Z_TEXT if all symbols are either printable characters (33 to 255)
 * or white spaces (9 to 13, or 32); or set it to Z_BINARY otherwise.
 * IN assertion: the fields Freq of dyn_ltree are set.
 */
local void set_data_type(s)
    deflate_state *s;
{
    int n;

    for (n = 0; n < 9; n++)
        if (s->dyn_ltree[n].Freq != 0)
            break;
    if (n == 9)
        for (n = 14; n < 32; n++)
            if (s->dyn_ltree[n].Freq != 0)
                break;
    s->strm->data_type = (n == 32) ? Z_TEXT : Z_BINARY;
}

/* =====
 * Reverse the first len bits of a code, using straightforward code (a faster
 * method would use a table)
 * IN assertion: 1 <= len <= 15
 */
local unsigned bi_reverse(code, len)
    unsigned code; /* the value to invert */
    int len; /* its bit length */
{
    register unsigned res = 0;
    do {
        res |= code & 1;
        code >>= 1, res <<= 1;
    } while (--len > 0);
    return res >> 1;
}

/* =====
 * Flush the bit buffer, keeping at most 7 bits in it.
 */
local void bi_flush(s)
    deflate_state *s;
{
    if (s->bi_valid == 16) {
        put_short(s, s->bi_buf);
        s->bi_buf = 0;
        s->bi_valid = 0;
    } else if (s->bi_valid >= 8) {
        put_byte(s, (Byte)s->bi_buf);
        s->bi_buf >>= 8;
        s->bi_valid -= 8;
    }
}

/* =====
 * Flush the bit buffer and align the output on a byte boundary
 */
local void bi_windup(s)
    deflate_state *s;
{
    if (s->bi_valid > 8) {
        put_short(s, s->bi_buf);
    } else if (s->bi_valid > 0) {
        put_byte(s, (Byte)s->bi_buf);
    }
}

```

```
    }
    s->bi_buf = 0;
    s->bi_valid = 0;
#ifdef DEBUG
    s->bits_sent = (s->bits_sent+7) & ~7;
#endif
}

/* =====
 * Copy a stored block, storing first the length and its
 * one's complement if requested.
 */
local void copy_block(s, buf, len, header)
    deflate_state *s;
    charf *buf; /* the input data */
    unsigned len; /* its length */
    int header; /* true if block header must be written */
{
    bi_windup(s); /* align on byte boundary */
    s->last_eob_len = 8; /* enough lookahead for inflate */

    if (header) {
        put_short(s, (ush)len);
        put_short(s, (ush)~len);
#ifdef DEBUG
        s->bits_sent += 2*16;
#endif
    }
#ifdef DEBUG
    s->bits_sent += (ulg)len<<3;
#endif
    while (len--) {
        put_byte(s, *buf++);
    }
}
```

```

/* minigzip.c -- simulate gzip using the zlib compression library
 * Copyright (C) 1995-2005 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 * minigzip is a minimal implementation of the gzip utility. This is
 * only an example of using zlib and isn't meant to replace the
 * full-featured gzip. No attempt is made to deal with file systems
 * limiting names to 14 or 8+3 characters, etc... Error checking is
 * very limited. So use minigzip only for testing; use gzip for the
 * real thing. On MSDOS, use only on file names without extension
 * or in pipe mode.
 */

/* @(#) $Id: minigzip.c,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#include <stdio.h>
#include "zlib.h"

#ifdef STDC
# include <string.h>
# include <stdlib.h>
#endif

#ifdef USE_MMAP
# include <sys/types.h>
# include <sys/mman.h>
# include <sys/stat.h>
#endif

#if defined(MSDOS) || defined(OS2) || defined(WIN32) || defined(__CYGWIN__)
# include <fcntl.h>
# include <io.h>
# define SET_BINARY_MODE(file) setmode(fileno(file), O_BINARY)
#else
# define SET_BINARY_MODE(file)
#endif

#ifdef VMS
# define unlink delete
# define GZ_SUFFIX "-gz"
#endif
#ifdef RISCOS
# define unlink remove
# define GZ_SUFFIX "-gz"
# define fileno(file) file->__file
#endif
#if defined(__MWERKS__) && __dest_os != __be_os && __dest_os != __win32_os
# include <unix.h> /* for fileno */
#endif

#ifndef WIN32 /* unlink already in stdio.h for WIN32 */
# extern int unlink OF((const char *));
#endif

#ifndef GZ_SUFFIX
# define GZ_SUFFIX ".gz"
#endif
#define SUFFIX_LEN (sizeof(GZ_SUFFIX)-1)

#define BUFLLEN 16384
#define MAX_NAME_LEN 1024

#ifdef MAXSEG_64K
# define local static
/* Needed for systems with limitation on stack size. */
#else
# define local
#endif

char *prog;

void error
    OF((const char *msg));

```

```

void gz_compress      OF((FILE      *in, gzFile out));
#ifdef USE_MMAP
int  gz_compress_mmap OF((FILE      *in, gzFile out));
#endif
void gz_uncompress     OF((gzFile in, FILE      *out));
void file_compress     OF((char      *file, char *mode));
void file_uncompress   OF((char      *file));
int  main              OF((int argc, char *argv[]));

/* =====
 * Display error message and exit
 */
void error(msg)
    const char *msg;
{
    fprintf(stderr, "%s: %s\n", prog, msg);
    exit(1);
}

/* =====
 * Compress input to output then close both files.
 */

void gz_compress(in, out)
    FILE      *in;
    gzFile out;
{
    local char buf[BUFLen];
    int len;
    int err;

#ifdef USE_MMAP
    /* Try first compressing with mmap. If mmap fails (minigzip used in a
     * pipe), use the normal fread loop.
     */
    if (gz_compress_mmap(in, out) == Z_OK) return;
#endif
    for (;;) {
        len = (int)fread(buf, 1, sizeof(buf), in);
        if (ferror(in)) {
            perror("fread");
            exit(1);
        }
        if (len == 0) break;

        if (gzwrite(out, buf, (unsigned)len) != len) error(gzerror(out, &err));
    }
    fclose(in);
    if (gzclose(out) != Z_OK) error("failed gzclose");
}

#ifdef USE_MMAP /* MMAP version, Miguel Albrecht <malbrech@eso.org> */

/* Try compressing the input file at once using mmap. Return Z_OK if
 * if success, Z_ERRNO otherwise.
 */
int gz_compress_mmap(in, out)
    FILE      *in;
    gzFile out;
{
    int len;
    int err;
    int ifd = fileno(in);
    caddr_t buf; /* mmap'ed buffer for the entire input file */
    off_t buf_len; /* length of the input file */
    struct stat sb;

    /* Determine the size of the file, needed for mmap: */
    if (fstat(ifd, &sb) < 0) return Z_ERRNO;
    buf_len = sb.st_size;
    if (buf_len <= 0) return Z_ERRNO;

    /* Now do the actual mmap: */
    buf = mmap((caddr_t) 0, buf_len, PROT_READ, MAP_SHARED, ifd, (off_t)0);

```

```
    if (buf == (caddr_t)(-1)) return Z_ERRNO;

    /* Compress the whole file at once: */
    len = gzwrite(out, (char *)buf, (unsigned)buf_len);

    if (len != (int)buf_len) error(gzerror(out, &err));

    munmap(buf, buf_len);
    fclose(in);
    if (gzclose(out) != Z_OK) error("failed gzclose");
    return Z_OK;
}
#endif /* USE_MMAP */

/* =====
 * Uncompress input to output then close both files.
 */
void gz_uncompress(in, out)
    gzFile in;
    FILE *out;
{
    local char buf[BUFLen];
    int len;
    int err;

    for (;;) {
        len = gzread(in, buf, sizeof(buf));
        if (len < 0) error(gzerror(in, &err));
        if (len == 0) break;

        if ((int)fwrite(buf, 1, (unsigned)len, out) != len) {
            error("failed fwrite");
        }
    }
    if (fclose(out)) error("failed fclose");

    if (gzclose(in) != Z_OK) error("failed gzclose");
}

/* =====
 * Compress the given file: create a corresponding .gz file and remove the
 * original.
 */
void file_compress(file, mode)
    char *file;
    char *mode;
{
    local char outfile[MAX_NAME_LEN];
    FILE *in;
    gzFile out;

    strcpy(outfile, file);
    strcat(outfile, GZ_SUFFIX);

    in = fopen(file, "rb");
    if (in == NULL) {
        perror(file);
        exit(1);
    }
    out = gzopen(outfile, mode);
    if (out == NULL) {
        fprintf(stderr, "%s: can't gzopen %s\n", prog, outfile);
        exit(1);
    }
    gz_compress(in, out);

    unlink(file);
}

/* =====
 * Uncompress the given file and remove the original.
 */
```

```

void file_uncompress(file)
    char *file;
{
    local char buf[MAX_NAME_LEN];
    char *infile, *outfile;
    FILE *out;
    gzFile in;
    uInt len = (uInt)strlen(file);

    strcpy(buf, file);

    if (len > SUFFIX_LEN && strcmp(file+len-SUFFIX_LEN, GZ_SUFFIX) == 0) {
        infile = file;
        outfile = buf;
        outfile[len-3] = '\0';
    } else {
        outfile = file;
        infile = buf;
        strcat(infile, GZ_SUFFIX);
    }
    in = gzopen(infile, "rb");
    if (in == NULL) {
        fprintf(stderr, "%s: can't gzopen %s\n", prog, infile);
        exit(1);
    }
    out = fopen(outfile, "wb");
    if (out == NULL) {
        perror(file);
        exit(1);
    }

    gz_uncompress(in, out);

    unlink(infile);
}

/* =====
 * Usage: minigzip [-d] [-f] [-h] [-r] [-1 to -9] [files...]
 *      -d : decompress
 *      -f : compress with Z_FILTERED
 *      -h : compress with Z_HUFFMAN_ONLY
 *      -r : compress with Z_RLE
 *      -1 to -9 : compression level
 */

int main(argc, argv)
    int argc;
    char *argv[];
{
    int uncompr = 0;
    gzFile file;
    char outmode[20];

    strcpy(outmode, "wb6 ");

    prog = argv[0];
    argc--, argv++;

    while (argc > 0) {
        if (strcmp(*argv, "-d") == 0)
            uncompr = 1;
        else if (strcmp(*argv, "-f") == 0)
            outmode[3] = 'f';
        else if (strcmp(*argv, "-h") == 0)
            outmode[3] = 'h';
        else if (strcmp(*argv, "-r") == 0)
            outmode[3] = 'R';
        else if ((*argv)[0] == '-' && (*argv)[1] >= '1' && (*argv)[1] <= '9' &&
            (*argv)[2] == 0)
            outmode[2] = (*argv)[1];
        else
            break;
        argc--, argv++;
    }

```

```
}
if (outmode[3] == ' ')
    outmode[3] = 0;
if (argc == 0) {
    SET_BINARY_MODE(stdin);
    SET_BINARY_MODE(stdout);
    if (uncompr) {
        file = gzdopen(fileno(stdin), "rb");
        if (file == NULL) error("can't gzdopen stdin");
        gz_uncompress(file, stdout);
    } else {
        file = gzdopen(fileno(stdout), outmode);
        if (file == NULL) error("can't gzdopen stdout");
        gz_compress(stdin, file);
    }
} else {
    do {
        if (uncompr) {
            file_uncompress(*argv);
        } else {
            file_compress(*argv, outmode);
        }
    } while (argv++, --argc);
}
return 0;
}
```

```
/* header created automatically with -DGEN_TREES_H */
```

```
local const ct_data static_ltree[L_CODES+2] = {
{ 12, 8, 140, 8, 76, 8, 204, 8, 44, 8,
 172, 8, 108, 8, 236, 8, 28, 8, 156, 8,
 92, 8, 220, 8, 60, 8, 188, 8, 124, 8,
 252, 8, 2, 8, 130, 8, 66, 8, 194, 8,
 34, 8, 162, 8, 98, 8, 226, 8, 18, 8,
 146, 8, 82, 8, 210, 8, 50, 8, 178, 8,
 114, 8, 242, 8, 10, 8, 138, 8, 74, 8,
 202, 8, 42, 8, 170, 8, 106, 8, 234, 8,
 26, 8, 154, 8, 90, 8, 218, 8, 58, 8,
 186, 8, 122, 8, 250, 8, 6, 8, 134, 8,
 70, 8, 198, 8, 38, 8, 166, 8, 102, 8,
 230, 8, 22, 8, 150, 8, 86, 8, 214, 8,
 54, 8, 182, 8, 118, 8, 246, 8, 14, 8,
 142, 8, 78, 8, 206, 8, 46, 8, 174, 8,
 110, 8, 238, 8, 30, 8, 158, 8, 94, 8,
 222, 8, 62, 8, 190, 8, 126, 8, 254, 8,
 1, 8, 129, 8, 65, 8, 193, 8, 33, 8,
 161, 8, 97, 8, 225, 8, 17, 8, 145, 8,
 81, 8, 209, 8, 49, 8, 177, 8, 113, 8,
 241, 8, 9, 8, 137, 8, 73, 8, 201, 8,
 41, 8, 169, 8, 105, 8, 233, 8, 25, 8,
 153, 8, 89, 8, 217, 8, 57, 8, 185, 8,
 121, 8, 249, 8, 5, 8, 133, 8, 69, 8,
 197, 8, 37, 8, 165, 8, 101, 8, 229, 8,
 21, 8, 149, 8, 85, 8, 213, 8, 53, 8,
 181, 8, 117, 8, 245, 8, 13, 8, 141, 8,
 77, 8, 205, 8, 45, 8, 173, 8, 109, 8,
 237, 8, 29, 8, 157, 8, 93, 8, 221, 8,
 61, 8, 189, 8, 125, 8, 253, 8, 19, 9,
 275, 9, 147, 9, 403, 9, 83, 9, 339, 9,
 211, 9, 467, 9, 51, 9, 307, 9, 179, 9,
 435, 9, 115, 9, 371, 9, 243, 9, 499, 9,
 11, 9, 267, 9, 139, 9, 395, 9, 75, 9,
 331, 9, 203, 9, 459, 9, 43, 9, 299, 9,
 171, 9, 427, 9, 107, 9, 363, 9, 235, 9,
 491, 9, 27, 9, 283, 9, 155, 9, 411, 9,
 91, 9, 347, 9, 219, 9, 475, 9, 59, 9,
 315, 9, 187, 9, 443, 9, 123, 9, 379, 9,
 251, 9, 507, 9, 7, 9, 263, 9, 135, 9,
 391, 9, 71, 9, 327, 9, 199, 9, 455, 9,
 39, 9, 295, 9, 167, 9, 423, 9, 103, 9,
 359, 9, 231, 9, 487, 9, 23, 9, 279, 9,
 151, 9, 407, 9, 87, 9, 343, 9, 215, 9,
 471, 9, 55, 9, 311, 9, 183, 9, 439, 9,
 119, 9, 375, 9, 247, 9, 503, 9, 15, 9,
 271, 9, 143, 9, 399, 9, 79, 9, 335, 9,
 207, 9, 463, 9, 47, 9, 303, 9, 175, 9,
 431, 9, 111, 9, 367, 9, 239, 9, 495, 9,
 31, 9, 287, 9, 159, 9, 415, 9, 95, 9,
 351, 9, 223, 9, 479, 9, 63, 9, 319, 9,
 191, 9, 447, 9, 127, 9, 383, 9, 255, 9,
 511, 9, 0, 7, 64, 7, 32, 7, 96, 7,
 16, 7, 80, 7, 48, 7, 112, 7, 8, 7,
 72, 7, 40, 7, 104, 7, 24, 7, 88, 7,
 56, 7, 120, 7, 4, 7, 68, 7, 36, 7,
 100, 7, 20, 7, 84, 7, 52, 7, 116, 7,
 3, 8, 131, 8, 67, 8, 195, 8, 35, 8,
 163, 8, 99, 8, 227, 8,
};
```

```
local const ct_data static_dtree[D_CODES] = {
{ { 0, 5 }, { 16, 5 }, { 8, 5 }, { 24, 5 }, { 4, 5 },
 { 20, 5 }, { 12, 5 }, { 28, 5 }, { 2, 5 }, { 18, 5 },
 { 10, 5 }, { 26, 5 }, { 6, 5 }, { 22, 5 }, { 14, 5 },
 { 30, 5 }, { 1, 5 }, { 17, 5 }, { 9, 5 }, { 25, 5 },
 { 5, 5 }, { 21, 5 }, { 13, 5 }, { 29, 5 }, { 3, 5 },
 { 19, 5 }, { 11, 5 }, { 27, 5 }, { 7, 5 }, { 23, 5 }
};
```

```
const uch _dist_code[DIST_CODE_LEN] = {
0, 1, 2, 3, 4, 4, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 8, 8, 8, 8,
```



```
8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 0, 0, 16, 17,
18, 18, 19, 19, 20, 20, 20, 20, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
};
```

```
const uch _length_code[MAX_MATCH-MIN_MATCH+1]= {
0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 9, 9, 10, 10, 11, 11, 12, 12, 12, 12,
13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 16,
17, 17, 17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19,
19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
};
```

```
local const int base_length[LENGTH_CODES] = {
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56,
64, 80, 96, 112, 128, 160, 192, 224, 0
};
```

```
local const int base_dist[D_CODES] = {
0, 1, 2, 3, 4, 6, 8, 12, 16, 24,
32, 48, 64, 96, 128, 192, 256, 384, 512, 768,
1024, 1536, 2048, 3072, 4096, 6144, 8192, 12288, 16384, 24576
};
```

```

/* uncompr.c -- decompress a memory buffer
 * Copyright (C) 1995-2003 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id: uncompr.c,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#define ZLIB_INTERNAL
#include "zlib.h"

/* =====
   Decompresses the source buffer into the destination buffer.  sourceLen is
   the byte length of the source buffer. Upon entry, destLen is the total
   size of the destination buffer, which must be large enough to hold the
   entire uncompressed data. (The size of the uncompressed data must have
   been saved previously by the compressor and transmitted to the decompressor
   by some mechanism outside the scope of this compression library.)
   Upon exit, destLen is the actual size of the compressed buffer.
   This function can be used to decompress a whole file at once if the
   input file is mmap'ed.

   uncompress returns Z_OK if success, Z_MEM_ERROR if there was not
   enough memory, Z_BUF_ERROR if there was not enough room in the output
   buffer, or Z_DATA_ERROR if the input data was corrupted.
*/
int ZEXPORT uncompress (dest, destLen, source, sourceLen)
    Bytef *dest;
    uLongf *destLen;
    const Bytef *source;
    uLong sourceLen;
{
    z_stream stream;
    int err;

    stream.next_in = (Bytef*)source;
    stream.avail_in = (uInt)sourceLen;
    /* Check for source > 64K on 16-bit machine: */
    if ((uLong)stream.avail_in != sourceLen) return Z_BUF_ERROR;

    stream.next_out = dest;
    stream.avail_out = (uInt)*destLen;
    if ((uLong)stream.avail_out != *destLen) return Z_BUF_ERROR;

    stream.zalloc = (alloc_func)0;
    stream.zfree = (free_func)0;

    err = inflateInit(&stream);
    if (err != Z_OK) return err;

    err = inflate(&stream, Z_FINISH);
    if (err != Z_STREAM_END) {
        inflateEnd(&stream);
        if (err == Z_NEED_DICT || (err == Z_BUF_ERROR && stream.avail_in == 0))
            return Z_DATA_ERROR;
        return err;
    }
    *destLen = stream.total_out;

    err = inflateEnd(&stream);
    return err;
}

```

```
/* zconf.h -- configuration of the zlib compression library
 * Copyright (C) 1995-2005 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id: zconf.h,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#ifndef ZCONF_H
#define ZCONF_H

/*
 * If you *really* need a unique prefix for all types and library functions,
 * compile with -DZ_PREFIX. The "standard" zlib should be compiled without it.
 */
#ifdef Z_PREFIX
#define deflateInit_      z_deflateInit_
#define deflate           z_deflate
#define deflateEnd        z_deflateEnd
#define inflateInit_     z_inflateInit_
#define inflate          z_inflate
#define inflateEnd        z_inflateEnd
#define deflateInit2_    z_deflateInit2_
#define deflateSetDictionary z_deflateSetDictionary
#define deflateCopy      z_deflateCopy
#define deflateReset     z_deflateReset
#define deflateParams    z_deflateParams
#define deflateBound     z_deflateBound
#define deflatePrime     z_deflatePrime
#define inflateInit2_    z_inflateInit2_
#define inflateSetDictionary z_inflateSetDictionary
#define inflateSync      z_inflateSync
#define inflateSyncPoint z_inflateSyncPoint
#define inflateCopy      z_inflateCopy
#define inflateReset     z_inflateReset
#define inflateBack      z_inflateBack
#define inflateBackEnd   z_inflateBackEnd
#define compress         z_compress
#define compress2        z_compress2
#define compressBound    z_compressBound
#define uncompress       z_uncompress
#define adler32          z_adler32
#define crc32            z_crc32
#define get_crc_table    z_get_crc_table
#define zError           z_zError

#define alloc_func        z_alloc_func
#define free_func         z_free_func
#define in_func           z_in_func
#define out_func          z_out_func
#define Byte              z_Byte
#define uInt              z_uInt
#define uLong             z_uLong
#define Bytef             z_Bytef
#define charf             z_charf
#define intf              z_intf
#define uIntf             z_uIntf
#define uLongf            z_uLongf
#define voidpf            z_voidpf
#define voidp             z_voidp
#endif

#endif

#ifdef __MSDOS__ && !defined(MSDOS)
#define MSDOS
#endif

#ifdef (defined(OS_2) || defined(__OS2__)) && !defined(OS2)
#define OS2
#endif

#ifdef __WINDOWS__ && !defined(WINDOWS)
#define WINDOWS
#endif

#ifdef __WIN32__ || defined(_WIN32_WCE) || defined(__WIN32__)
#ifndef WIN32
#define WIN32
#endif
#endif
```

```
#endif
#if (defined(MSDOS) || defined(OS2) || defined(WINDOWS)) && !defined(WIN32)
# if !defined(__GNUC__) && !defined(__FLAT__) && !defined(__386__)
#   ifndef SYS16BIT
#     define SYS16BIT
#   endif
# endif
#endif

/*
 * Compile with -D_MAXSEG_64K if the alloc function cannot allocate more
 * than 64k bytes at a time (needed on systems with 16-bit int).
 */
#ifdef SYS16BIT
# define MAXSEG_64K
#endif
#ifdef MSDOS
# define UNALIGNED_OK
#endif

#ifdef __STDC_VERSION__
# ifndef STDC
#   define STDC
# endif
# if __STDC_VERSION__ >= 199901L
#   ifndef STDC99
#     define STDC99
#   endif
# endif
#endif
#if !defined(STDC) && (defined(__STDC__) || defined(__cplusplus))
# define STDC
#endif
#if !defined(STDC) && (defined(__GNUC__) || defined(__BORLANDC__))
# define STDC
#endif
#if !defined(STDC) && (defined(MSDOS) || defined(WINDOWS) || defined(WIN32))
# define STDC
#endif
#if !defined(STDC) && (defined(OS2) || defined(__HOS_AIX__))
# define STDC
#endif

#if defined(__OS400__) && !defined(STDC) /* iSeries (formerly AS/400). */
# define STDC
#endif

#ifndef STDC
# ifndef const /* cannot use !defined(STDC) && !defined(const) on Mac */
#   define const /* note: need a more gentle solution here */
# endif
#endif

/* Some Mac compilers merge all .h files incorrectly: */
#if defined(__MWERKS__) || defined(applec) || defined(THINK_C) || defined(__SC__)
# define NO_DUMMY_DECL
#endif

/* Maximum value for memLevel in deflateInit2 */
#ifndef MAX_MEM_LEVEL
# ifdef MAXSEG_64K
#   define MAX_MEM_LEVEL 8
# else
#   define MAX_MEM_LEVEL 9
# endif
#endif

/* Maximum value for windowBits in deflateInit2 and inflateInit2.
 * WARNING: reducing MAX_WBITS makes minigzip unable to extract .gz files
 * created by gzip. (Files created by minigzip can still be extracted by
 * gzip.)
 */
#ifndef MAX_WBITS
# define MAX_WBITS 15 /* 32K LZ77 window */

```

```

#endif

/* The memory requirements for deflate are (in bytes):
    (1 << (windowBits+2)) + (1 << (memLevel+9))
that is: 128K for windowBits=15 + 128K for memLevel = 8 (default values)
plus a few kilobytes for small objects. For example, if you want to reduce
the default memory requirements from 256K to 128K, compile with
    make CFLAGS="-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7"
Of course this will generally degrade compression (there's no free lunch).

    The memory requirements for inflate are (in bytes) 1 << windowBits
that is, 32K for windowBits=15 (default value) plus a few kilobytes
for small objects.
*/

                /* Type declarations */

#ifndef OF /* function prototypes */
#  ifdef STDC
#    define OF(args)  args
#  else
#    define OF(args)  ()
#  endif
#endif

/* The following definitions for FAR are needed only for MSDOS mixed
 * model programming (small or medium model with some far allocations).
 * This was tested only with MSC; for other MSDOS compilers you may have
 * to define NO_MEMCPY in zutil.h.  If you don't need the mixed model,
 * just define FAR to be empty.
 */
#ifndef SYS16BIT
#  if defined(M_I86SM) || defined(M_I86MM)
/* MSC small or medium model */
#    define SMALL_MEDIUM
#    ifdef _MSC_VER
#      define FAR _far
#    else
#      define FAR far
#    endif
#  endif
#  if (defined(__SMALL__) || defined(__MEDIUM__))
/* Turbo C small or medium model */
#    define SMALL_MEDIUM
#    ifdef __BORLANDC__
#      define FAR _far
#    else
#      define FAR far
#    endif
#  endif
#endif

#if defined(WINDOWS) || defined(WIN32)
/* If building or using zlib as a DLL, define ZLIB_DLL.
 * This is not mandatory, but it offers a little performance increase.
 */
#  ifdef ZLIB_DLL
#    if defined(WIN32) && (!defined(__BORLANDC__) || (__BORLANDC__ >= 0x500))
#      ifdef ZLIB_INTERNAL
#        define ZEXTERN extern __declspec(dllexport)
#      else
#        define ZEXTERN extern __declspec(dllimport)
#      endif
#    endif
#  endif /* ZLIB_DLL */
/* If building or using zlib with the WINAPI/WINAPIV calling convention,
 * define ZLIB_WINAPI.
 * Caution: the standard ZLIB1.DLL is NOT compiled using ZLIB_WINAPI.
 */
#  ifdef ZLIB_WINAPI
#    ifdef FAR
#      undef FAR
#    endif
#    include <windows.h>

```

```

/* No need for _export, use ZLIB.DEF instead. */
/* For complete Windows compatibility, use WINAPI, not __stdcall. */
# define ZEXPORT WINAPI
# ifdef WIN32
#   define ZEXPORTVA WINAPIV
# else
#   define ZEXPORTVA FAR CDECL
# endif
# endif
#endif

#if defined (__BEOS__)
# ifdef ZLIB_DLL
#   ifdef ZLIB_INTERNAL
#     define ZEXPORT __declspec(dllexport)
#     define ZEXPORTVA __declspec(dllexport)
#   else
#     define ZEXPORT __declspec(dllimport)
#     define ZEXPORTVA __declspec(dllimport)
#   endif
# endif
# endif

#ifndef ZEXTERN
# define ZEXTERN extern
#endif

#ifndef ZEXPORT
# define ZEXPORT
#endif

#ifndef ZEXPORTVA
# define ZEXPORTVA
#endif

#ifndef FAR
# define FAR
#endif

#if !defined (__MACTYPES__)
typedef unsigned char Byte; /* 8 bits */
typedef unsigned int uInt; /* 16 bits or more */
typedef unsigned long uLong; /* 32 bits or more */
#endif

#ifndef SMALL_MEDIUM
/* Borland C/C++ and some old MSC versions ignore FAR inside typedef */
# define Bytef Byte FAR
#else
typedef Byte FAR Bytef;
#endif
typedef char FAR charf;
typedef int FAR intf;
typedef uInt FAR uIntf;
typedef uLong FAR uLongf;

#ifdef STDC
typedef void const *voidpc;
typedef void FAR *voidpf;
typedef void *voidp;
#else
typedef Byte const *voidpc;
typedef Byte FAR *voidpf;
typedef Byte *voidp;
#endif

#if 0
/* HAVE_UNISTD_H -- this line is updated by ./configure */
# include <sys/types.h> /* for off_t */
# include <unistd.h> /* for SEEK_* and off_t */
# ifdef VMS
#   include <unixio.h> /* for off_t */
# endif
# define z_off_t off_t
#endif
#ifndef SEEK_SET
# define SEEK_SET 0 /* Seek from beginning of file. */

```

```
# define SEEK_CUR      1      /* Seek from current position. */
# define SEEK_END      2      /* Set file pointer to EOF plus "offset" */
#endif
#ifndef z_off_t
# define z_off_t long
#endif

#if defined(__OS400__)
# define NO_vsnprintf
#endif

#if defined(__MVS__)
# define NO_vsnprintf
# ifdef FAR
# undef FAR
# endif
# endif
#endif

/* MVS linker does not support external names larger than 8 bytes */
#if defined(__MVS__)
# pragma map(deflateInit_,"DEIN")
# pragma map(deflateInit2_,"DEIN2")
# pragma map(deflateEnd,"DEEND")
# pragma map(deflateBound,"DEBND")
# pragma map(inflateInit_,"ININ")
# pragma map(inflateInit2_,"ININ2")
# pragma map(inflateEnd,"INEND")
# pragma map(inflateSync,"INSY")
# pragma map(inflateSetDictionary,"INSEDI")
# pragma map(compressBound,"CMBND")
# pragma map(inflate_table,"INTABL")
# pragma map(inflate_fast,"INFA")
# pragma map(inflate_copyright,"INCOPY")
#endif

#endif /* ZCONF_H */
```

```
/* zconf.h -- configuration of the zlib compression library
 * Copyright (C) 1995-2005 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id: zconf.in.h,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#ifndef ZCONF_H
#define ZCONF_H

/*
 * If you *really* need a unique prefix for all types and library functions,
 * compile with -DZ_PREFIX. The "standard" zlib should be compiled without it.
 */
#ifdef Z_PREFIX
#define deflateInit_      z_deflateInit_
#define deflate           z_deflate
#define deflateEnd        z_deflateEnd
#define inflateInit_     z_inflateInit_
#define inflate           z_inflate
#define inflateEnd        z_inflateEnd
#define deflateInit2_    z_deflateInit2_
#define deflateSetDictionary z_deflateSetDictionary
#define deflateCopy       z_deflateCopy
#define deflateReset      z_deflateReset
#define deflateParams     z_deflateParams
#define deflateBound      z_deflateBound
#define deflatePrime      z_deflatePrime
#define inflateInit2_    z_inflateInit2_
#define inflateSetDictionary z_inflateSetDictionary
#define inflateSync       z_inflateSync
#define inflateSyncPoint z_inflateSyncPoint
#define inflateCopy       z_inflateCopy
#define inflateReset      z_inflateReset
#define inflateBack       z_inflateBack
#define inflateBackEnd    z_inflateBackEnd
#define compress          z_compress
#define compress2          z_compress2
#define compressBound     z_compressBound
#define uncompress         z_uncompress
#define adler32            z_adler32
#define crc32              z_crc32
#define get_crc_table     z_get_crc_table
#define zError             z_zError

#define alloc_func         z_alloc_func
#define free_func          z_free_func
#define in_func            z_in_func
#define out_func           z_out_func
#define Byte              z_Byte
#define uInt               z_uInt
#define uLong              z_uLong
#define Bytef              z_Bytef
#define charf              z_charf
#define intf               z_intf
#define uIntf              z_uIntf
#define uLongf             z_uLongf
#define voidpf             z_voidpf
#define voidp              z_voidp
#endif

#ifdef __MSDOS__ && !defined(MSDOS)
#define MSDOS
#endif
#ifdef (defined(OS_2) || defined(__OS2__)) && !defined(OS2)
#define OS2
#endif
#ifdef __WINDOWS__ && !defined(WINDOWS)
#define WINDOWS
#endif
#ifdef __WIN32__ || defined(_WIN32_WCE) || defined(__WIN32__)
#ifndef WIN32
#define WIN32
#endif
#endif
```



```
#endif
#if (defined(MSDOS) || defined(OS2) || defined(WINDOWS)) && !defined(WIN32)
# if !defined(__GNUC__) && !defined(__FLAT__) && !defined(__386__)
#   ifndef SYS16BIT
#     define SYS16BIT
#   endif
# endif
#endif

/*
 * Compile with -D_MAXSEG_64K if the alloc function cannot allocate more
 * than 64k bytes at a time (needed on systems with 16-bit int).
 */
#ifdef SYS16BIT
# define MAXSEG_64K
#endif
#ifdef MSDOS
# define UNALIGNED_OK
#endif

#ifdef __STDC_VERSION__
# ifndef STDC
#   define STDC
# endif
# if __STDC_VERSION__ >= 199901L
#   ifndef STDC99
#     define STDC99
#   endif
# endif
#endif
#if !defined(STDC) && (defined(__STDC__) || defined(__cplusplus))
# define STDC
#endif
#if !defined(STDC) && (defined(__GNUC__) || defined(__BORLANDC__))
# define STDC
#endif
#if !defined(STDC) && (defined(MSDOS) || defined(WINDOWS) || defined(WIN32))
# define STDC
#endif
#if !defined(STDC) && (defined(OS2) || defined(__HOS_AIX__))
# define STDC
#endif

#if defined(__OS400__) && !defined(STDC) /* iSeries (formerly AS/400). */
# define STDC
#endif

#ifndef STDC
# ifndef const /* cannot use !defined(STDC) && !defined(const) on Mac */
#   define const /* note: need a more gentle solution here */
# endif
#endif

/* Some Mac compilers merge all .h files incorrectly: */
#if defined(__MWERKS__) || defined(applec) || defined(THINK_C) || defined(__SC__)
# define NO_DUMMY_DECL
#endif

/* Maximum value for memLevel in deflateInit2 */
#ifndef MAX_MEM_LEVEL
# ifdef MAXSEG_64K
#   define MAX_MEM_LEVEL 8
# else
#   define MAX_MEM_LEVEL 9
# endif
#endif

/* Maximum value for windowBits in deflateInit2 and inflateInit2.
 * WARNING: reducing MAX_WBITS makes minigzip unable to extract .gz files
 * created by gzip. (Files created by minigzip can still be extracted by
 * gzip.)
 */
#ifndef MAX_WBITS
# define MAX_WBITS 15 /* 32K LZ77 window */

```

```

#endif

/* The memory requirements for deflate are (in bytes):
    (1 << (windowBits+2)) + (1 << (memLevel+9))
that is: 128K for windowBits=15 + 128K for memLevel = 8 (default values)
plus a few kilobytes for small objects. For example, if you want to reduce
the default memory requirements from 256K to 128K, compile with
    make CFLAGS="-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7"
Of course this will generally degrade compression (there's no free lunch).

    The memory requirements for inflate are (in bytes) 1 << windowBits
that is, 32K for windowBits=15 (default value) plus a few kilobytes
for small objects.
*/

/* Type declarations */

#ifndef OF /* function prototypes */
#  ifdef STDC
#    define OF(args)  args
#  else
#    define OF(args)  ()
#  endif
#endif

/* The following definitions for FAR are needed only for MSDOS mixed
 * model programming (small or medium model with some far allocations).
 * This was tested only with MSC; for other MSDOS compilers you may have
 * to define NO_MEMCPY in zutil.h.  If you don't need the mixed model,
 * just define FAR to be empty.
 */
#ifdef SYS16BIT
#  if defined(M_I86SM) || defined(M_I86MM)
/* MSC small or medium model */
#    define SMALL_MEDIUM
#    ifdef _MSC_VER
#      define FAR _far
#    else
#      define FAR far
#    endif
#  endif
#  if (defined(__SMALL__) || defined(__MEDIUM__))
/* Turbo C small or medium model */
#    define SMALL_MEDIUM
#    ifdef __BORLANDC__
#      define FAR _far
#    else
#      define FAR far
#    endif
#  endif
#endif

#if defined(WINDOWS) || defined(WIN32)
/* If building or using zlib as a DLL, define ZLIB_DLL.
 * This is not mandatory, but it offers a little performance increase.
 */
#  ifdef ZLIB_DLL
#    if defined(WIN32) && (!defined(__BORLANDC__) || (__BORLANDC__ >= 0x500))
#      ifdef ZLIB_INTERNAL
#        define ZEXTERN extern __declspec(dllexport)
#      else
#        define ZEXTERN extern __declspec(dllimport)
#      endif
#    endif
#  endif /* ZLIB_DLL */
/* If building or using zlib with the WINAPI/WINAPIV calling convention,
 * define ZLIB_WINAPI.
 * Caution: the standard ZLIB1.DLL is NOT compiled using ZLIB_WINAPI.
 */
#  ifdef ZLIB_WINAPI
#    ifdef FAR
#      undef FAR
#    endif
#    include <windows.h>

```

```
/* No need for _export, use ZLIB.DEF instead. */
/* For complete Windows compatibility, use WINAPI, not __stdcall. */
# define ZEXPORT WINAPI
# ifdef WIN32
#   define ZEXPORTVA WINAPIV
# else
#   define ZEXPORTVA FAR CDECL
# endif
# endif
#endif

#if defined (__BEOS__)
# ifdef ZLIB_DLL
#   ifdef ZLIB_INTERNAL
#     define ZEXPORT __declspec(dllexport)
#     define ZEXPORTVA __declspec(dllexport)
#   else
#     define ZEXPORT __declspec(dllimport)
#     define ZEXPORTVA __declspec(dllimport)
#   endif
# endif
#endif

#ifndef ZEXTERN
# define ZEXTERN extern
#endif

#ifndef ZEXPORT
# define ZEXPORT
#endif

#ifndef ZEXPORTVA
# define ZEXPORTVA
#endif

#ifndef FAR
# define FAR
#endif

#if !defined (__MACTYPES__)
typedef unsigned char Byte; /* 8 bits */
typedef unsigned int uInt; /* 16 bits or more */
typedef unsigned long uLong; /* 32 bits or more */
#endif

#ifndef SMALL_MEDIUM
/* Borland C/C++ and some old MSC versions ignore FAR inside typedef */
# define Bytef Byte FAR
#else
typedef Byte FAR Bytef;
#endif
typedef char FAR charf;
typedef int FAR intf;
typedef uInt FAR uIntf;
typedef uLong FAR uLongf;

#ifndef STDC
typedef void const *voidpc;
typedef void FAR *voidpf;
typedef void *voidp;
#else
typedef Byte const *voidpc;
typedef Byte FAR *voidpf;
typedef Byte *voidp;
#endif

#if 0
/* HAVE_UNISTD_H -- this line is updated by ./configure */
# include <sys/types.h> /* for off_t */
# include <unistd.h> /* for SEEK_* and off_t */
# ifdef VMS
#   include <unixio.h> /* for off_t */
# endif
# define z_off_t off_t
#endif
#ifndef SEEK_SET
# define SEEK_SET 0 /* Seek from beginning of file. */

```

```
# define SEEK_CUR      1      /* Seek from current position. */
# define SEEK_END      2      /* Set file pointer to EOF plus "offset" */
#endif
#ifndef z_off_t
# define z_off_t long
#endif

#if defined(__OS400__)
# define NO_vsnprintf
#endif

#if defined(__MVS__)
# define NO_vsnprintf
# ifdef FAR
# undef FAR
# endif
#endif

/* MVS linker does not support external names larger than 8 bytes */
#if defined(__MVS__)
# pragma map(deflateInit_,"DEIN")
# pragma map(deflateInit2_,"DEIN2")
# pragma map(deflateEnd,"DEEND")
# pragma map(deflateBound,"DEBND")
# pragma map(inflateInit_,"ININ")
# pragma map(inflateInit2_,"ININ2")
# pragma map(inflateEnd,"INEND")
# pragma map(inflateSync,"INSY")
# pragma map(inflateSetDictionary,"INSEDI")
# pragma map(compressBound,"CMBND")
# pragma map(inflate_table,"INTABL")
# pragma map(inflate_fast,"INFA")
# pragma map(inflate_copyright,"INCOPY")
#endif

#endif /* ZCONF_H */
```

```
.TH ZLIB 3 "18 July 2005"
.SH NAME
zlib \- compression/decompression library
.SH SYNOPSIS
[see
.I zlib.h
for full description]
.SH DESCRIPTION
The
.I zlib
library is a general purpose data compression library.
The code is thread safe.
It provides in-memory compression and decompression functions,
including integrity checks of the uncompressed data.
This version of the library supports only one compression method (deflation)
but other algorithms will be added later
and will have the same stream interface.
.LP
Compression can be done in a single step if the buffers are large enough
(for example if an input file is mmap'ed),
or can be done by repeated calls of the compression function.
In the latter case,
the application must provide more input and/or consume the output
(providing more output space) before each call.
.LP
The library also supports reading and writing files in
.IR gzip (1)
(.gz) format
with an interface similar to that of stdio.
.LP
The library does not install any signal handler.
The decoder checks the consistency of the compressed data,
so the library should never crash even in case of corrupted input.
.LP
All functions of the compression library are documented in the file
.IR zlib.h .
The distribution source includes examples of use of the library
in the files
.I example.c
and
.IR minigzip.c .
.LP
Changes to this version are documented in the file
.I ChangeLog
that accompanies the source,
and are concerned primarily with bug fixes and portability enhancements.
.LP
A Java implementation of
.I zlib
is available in the Java Development Kit 1.1:
.IP
http://www.javasoft.com/products/JDK/1.1/docs/api/Package-java.util.zip.html
.LP
A Perl interface to
.IR zlib ,
written by Paul Marquess (pmqs@cpan.org),
is available at CPAN (Comprehensive Perl Archive Network) sites,
including:
.IP
http://www.cpan.org/modules/by-module/Compress/
.LP
A Python interface to
.IR zlib ,
written by A.M. Kuchling (amk@magnet.com),
is available in Python 1.5 and later versions:
.IP
http://www.python.org/doc/lib/module-zlib.html
.LP
A
.I zlib
binding for
.IR tcl (1),
written by Andreas Kupries (a.kupries@westend.com),
is available at:
```

.IP
<http://www.westend.com/~kupries/doc/trf/man/man.html>
.LP
An experimental package to read and write files in .zip format,
written on top of
.I zlib
by Gilles Vollant (info@winimage.com),
is available at:
.IP
<http://www.winimage.com/zLibDll/unzip.html>
and also in the
.I contrib/minizip
directory of the main
.I zlib
web site.
.SH "SEE ALSO"
The
.I zlib
web site can be found at either of these locations:
.IP
<http://www.zlib.org>
.br
<http://www.gzip.org/zlib/>
.LP
The data format used by the zlib library is described by RFC
(Request for Comments) 1950 to 1952 in the files:
.IP
<http://www.ietf.org/rfc/rfc1950.txt> (concerning zlib format)
.br
<http://www.ietf.org/rfc/rfc1951.txt> (concerning deflate format)
.br
<http://www.ietf.org/rfc/rfc1952.txt> (concerning gzip format)
.LP
These documents are also available in other formats from:
.IP
<ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html>
.LP
Mark Nelson (markn@ieee.org) wrote an article about
.I zlib
for the Jan. 1997 issue of Dr. Dobbs's Journal;
a copy of the article is available at:
.IP
<http://dogma.net/markn/articles/zlibtool/zlibtool.htm>
.SH "REPORTING PROBLEMS"
Before reporting a problem,
please check the
.I zlib
web site to verify that you have the latest version of
.IR zlib ;
otherwise,
obtain the latest version and see if the problem still exists.
Please read the
.I zlib
FAQ at:
.IP
http://www.gzip.org/zlib/zlib_faq.html
.LP
before asking for help.
Send questions and/or comments to zlib@gzip.org,
or (for the Windows DLL version) to Gilles Vollant (info@winimage.com).
.SH AUTHORS
Version 1.2.3
Copyright (C) 1995-2005 Jean-loup Gailly (jloup@gzip.org)
and Mark Adler (madler@alumni.caltech.edu).
.LP
This software is provided "as-is,"
without any express or implied warranty.
In no event will the authors be held liable for any damages
arising from the use of this software.
See the distribution directory with respect to requirements
governing redistribution.
The deflate format used by
.I zlib
was defined by Phil Katz.

The deflate and
.I zlib
specifications were written by L. Peter Deutsch.
Thanks to all the people who reported problems and suggested various
improvements in
.IR zlib ;
who are too numerous to cite here.
.LP
UNIX manual page by R. P. C. Rodgers,
U.S. National Library of Medicine (rodgers@nlm.nih.gov).
.\" end of man page

```
/* zlib.h -- interface of the 'zlib' general purpose compression library
   version 1.2.3, July 18th, 2005
```

Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly
jloup@gzip.org

Mark Adler
madler@alumni.caltech.edu

The data format used by the zlib library is described by RFCs (Request for Comments) 1950 to 1952 in the files <http://www.ietf.org/rfc/rfc1950.txt> (zlib format), [rfc1951.txt](http://www.ietf.org/rfc/rfc1951.txt) (deflate format) and [rfc1952.txt](http://www.ietf.org/rfc/rfc1952.txt) (gzip format).

```
*/
```

```
#ifndef ZLIB_H
#define ZLIB_H
```

```
#include "zconf.h"
```

```
#ifdef __cplusplus
extern "C" {
#endif
```

```
#define ZLIB_VERSION "1.2.3"
#define ZLIB_VERNUM 0x1230
```

```
/*
```

The 'zlib' compression library provides in-memory compression and decompression functions, including integrity checks of the uncompressed data. This version of the library supports only one compression method (deflation) but other algorithms will be added later and will have the same stream interface.

Compression can be done in a single step if the buffers are large enough (for example if an input file is mmap'ed), or can be done by repeated calls of the compression function. In the latter case, the application must provide more input and/or consume the output (providing more output space) before each call.

The compressed data format used by default by the in-memory functions is the zlib format, which is a zlib wrapper documented in RFC 1950, wrapped around a deflate stream, which is itself documented in RFC 1951.

The library also supports reading and writing files in gzip (.gz) format with an interface similar to that of stdio using the functions that start with "gz". The gzip format is different from the zlib format. gzip is a gzip wrapper, documented in RFC 1952, wrapped around a deflate stream.

This library can optionally read and write gzip streams in memory as well.

The zlib format was designed to be compact and fast for use in memory and on communications channels. The gzip format was designed for single-file compression on file systems, has a larger header than zlib to maintain directory information, and uses a different, slower check method than zlib.

The library does not install any signal handler. The decoder checks the consistency of the compressed data, so the library should never crash even in case of corrupted input.


```

*/

typedef voidpf (*alloc_func) OF((voidpf opaque, uInt items, uInt size));
typedef void (*free_func) OF((voidpf opaque, voidpf address));

struct internal_state;

typedef struct z_stream_s {
    Bytef *next_in; /* next input byte */
    uInt avail_in; /* number of bytes available at next_in */
    uLong total_in; /* total nb of input bytes read so far */

    Bytef *next_out; /* next output byte should be put there */
    uInt avail_out; /* remaining free space at next_out */
    uLong total_out; /* total nb of bytes output so far */

    char *msg; /* last error message, NULL if no error */
    struct internal_state FAR *state; /* not visible by applications */

    alloc_func zalloc; /* used to allocate the internal state */
    free_func zfree; /* used to free the internal state */
    voidpf opaque; /* private data object passed to zalloc and zfree */

    int data_type; /* best guess about the data type: binary or text */
    uLong Adler; /* Adler32 value of the uncompressed data */
    uLong reserved; /* reserved for future use */
} z_stream;

typedef z_stream FAR *z_stream;

/*
    gzip header information passed to and from zlib routines. See RFC 1952
    for more details on the meanings of these fields.
*/
typedef struct gz_header_s {
    int text; /* true if compressed data believed to be text */
    uLong time; /* modification time */
    int xflags; /* extra flags (not used when writing a gzip file) */
    int os; /* operating system */
    Bytef *extra; /* pointer to extra field or Z_NULL if none */
    uInt extra_len; /* extra field length (valid if extra != Z_NULL) */
    uInt extra_max; /* space at extra (only when reading header) */
    Bytef *name; /* pointer to zero-terminated file name or Z_NULL */
    uInt name_max; /* space at name (only when reading header) */
    Bytef *comment; /* pointer to zero-terminated comment or Z_NULL */
    uInt comm_max; /* space at comment (only when reading header) */
    int hcrc; /* true if there was or will be a header crc */
    int done; /* true when done reading gzip header (not used
               when writing a gzip file) */
} gz_header;

typedef gz_header FAR *gz_headerp;

/*
    The application must update next_in and avail_in when avail_in has
    dropped to zero. It must update next_out and avail_out when avail_out
    has dropped to zero. The application must initialize zalloc, zfree and
    opaque before calling the init function. All other fields are set by the
    compression library and must not be updated by the application.

    The opaque value provided by the application will be passed as the first
    parameter for calls of zalloc and zfree. This can be useful for custom
    memory management. The compression library attaches no meaning to the
    opaque value.

    zalloc must return Z_NULL if there is not enough memory for the object.
    If zlib is used in a multi-threaded application, zalloc and zfree must be
    thread safe.

    On 16-bit systems, the functions zalloc and zfree must be able to allocate
    exactly 65536 bytes, but will not be required to allocate more than this
    if the symbol MAXSEG_64K is defined (see zconf.h). WARNING: On MSDOS,
    pointers returned by zalloc for objects of exactly 65536 bytes *must*
    have their offset normalized to zero. The default allocation function

```

provided by this library ensures this (see zutil.c). To reduce memory requirements and avoid any allocation of 64K objects, at the expense of compression ratio, compile the library with `-DZLIB_WBITS=14` (see `zconf.h`).

The fields `total_in` and `total_out` can be used for statistics or progress reports. After compression, `total_in` holds the total size of the uncompressed data and may be saved for use in the decompressor (particularly if the decompressor wants to decompress everything in a single step).

```

*/

                                /* constants */

#define Z_NO_FLUSH      0
#define Z_PARTIAL_FLUSH 1 /* will be removed, use Z_SYNC_FLUSH instead */
#define Z_SYNC_FLUSH    2
#define Z_FULL_FLUSH    3
#define Z_FINISH        4
#define Z_BLOCK         5
/* Allowed flush values; see deflate() and inflate() below for details */

#define Z_OK             0
#define Z_STREAM_END     1
#define Z_NEED_DICT      2
#define Z_ERRNO          (-1)
#define Z_STREAM_ERROR   (-2)
#define Z_DATA_ERROR     (-3)
#define Z_MEM_ERROR      (-4)
#define Z_BUF_ERROR      (-5)
#define Z_VERSION_ERROR  (-6)
/* Return codes for the compression/decompression functions. Negative
 * values are errors, positive values are used for special but normal events.
 */

#define Z_NO_COMPRESSION      0
#define Z_BEST_SPEED          1
#define Z_BEST_COMPRESSION    9
#define Z_DEFAULT_COMPRESSION (-1)
/* compression levels */

#define Z_FILTERED            1
#define Z_HUFFMAN_ONLY        2
#define Z_RLE                 3
#define Z_FIXED               4
#define Z_DEFAULT_STRATEGY    0
/* compression strategy; see deflateInit2() below for details */

#define Z_BINARY              0
#define Z_TEXT                1
#define Z_ASCII               Z_TEXT /* for compatibility with 1.2.2 and earlier */
#define Z_UNKNOWN             2
/* Possible values of the data_type field (though see inflate()) */

#define Z_DEFLATED            8
/* The deflate compression method (the only one supported in this version) */

#define Z_NULL                0 /* for initializing zalloc, zfree, opaque */

#define zlib_version zlibVersion()
/* for compatibility with versions < 1.0.2 */

                                /* basic functions */

ZEXTERN const char * ZEXPORT zlibVersion OF((void));
/* The application can compare zlibVersion and ZLIB_VERSION for consistency.
 * If the first character differs, the library code actually used is
 * not compatible with the zlib.h header file used by the application.
 * This check is automatically made by deflateInit and inflateInit.
 */

/*
ZEXTERN int ZEXPORT deflateInit OF((z_streamp strm, int level));

    Initializes the internal stream state for compression. The fields

```

`zalloc`, `zfree` and `opaque` must be initialized before by the caller. If `zalloc` and `zfree` are set to `Z_NULL`, `deflateInit` updates them to use default allocation functions.

The compression level must be `Z_DEFAULT_COMPRESSION`, or between 0 and 9: 1 gives best speed, 9 gives best compression, 0 gives no compression at all (the input data is simply copied a block at a time). `Z_DEFAULT_COMPRESSION` requests a default compromise between speed and compression (currently equivalent to level 6).

`deflateInit` returns `Z_OK` if success, `Z_MEM_ERROR` if there was not enough memory, `Z_STREAM_ERROR` if level is not a valid compression level, `Z_VERSION_ERROR` if the zlib library version (`zlib_version`) is incompatible with the version assumed by the caller (`ZLIB_VERSION`). `msg` is set to null if there is no error message. `deflateInit` does not perform any compression: this will be done by `deflate()`.

*/

```
ZEXTERN int ZEXPORT deflate OF((z_stream* strm, int flush));
```

/*

`deflate` compresses as much data as possible, and stops when the input buffer becomes empty or the output buffer becomes full. It may introduce some output latency (reading input without producing any output) except when forced to flush.

The detailed semantics are as follows. `deflate` performs one or both of the following actions:

- Compress more input starting at `next_in` and update `next_in` and `avail_in` accordingly. If not all input can be processed (because there is not enough room in the output buffer), `next_in` and `avail_in` are updated and processing will resume at this point for the next call of `deflate()`.
- Provide more output starting at `next_out` and update `next_out` and `avail_out` accordingly. This action is forced if the parameter `flush` is non zero. Forcing flush frequently degrades the compression ratio, so this parameter should be set only when necessary (in interactive applications). Some output may be provided even if `flush` is not set.

Before the call of `deflate()`, the application should ensure that at least one of the actions is possible, by providing more input and/or consuming more output, and updating `avail_in` or `avail_out` accordingly; `avail_out` should never be zero before the call. The application can consume the compressed output when it wants, for example when the output buffer is full (`avail_out == 0`), or after each call of `deflate()`. If `deflate` returns `Z_OK` and with zero `avail_out`, it must be called again after making room in the output buffer because there might be more output pending.

Normally the parameter `flush` is set to `Z_NO_FLUSH`, which allows `deflate` to decide how much data to accumulate before producing output, in order to maximize compression.

If the parameter `flush` is set to `Z_SYNC_FLUSH`, all pending output is flushed to the output buffer and the output is aligned on a byte boundary, so that the decompressor can get all input data available so far. (In particular `avail_in` is zero after the call if enough output space has been provided before the call.) Flushing may degrade compression for some compression algorithms and so it should be used only when necessary.

If `flush` is set to `Z_FULL_FLUSH`, all output is flushed as with `Z_SYNC_FLUSH`, and the compression state is reset so that decompression can restart from this point if previous compressed data has been damaged or if random access is desired. Using `Z_FULL_FLUSH` too often can seriously degrade compression.

If `deflate` returns with `avail_out == 0`, this function must be called again with the same value of the `flush` parameter and more output space (updated `avail_out`), until the flush is complete (`deflate` returns with non-zero `avail_out`). In the case of a `Z_FULL_FLUSH` or `Z_SYNC_FLUSH`, make sure that `avail_out` is greater than six to avoid repeated flush markers due to `avail_out == 0` on return.

If the parameter `flush` is set to `Z_FINISH`, pending input is processed,

pending output is flushed and deflate returns with Z_STREAM_END if there was enough output space; if deflate returns with Z_OK, this function must be called again with Z_FINISH and more output space (updated avail_out) but no more input data, until it returns with Z_STREAM_END or an error. After deflate has returned Z_STREAM_END, the only possible operations on the stream are deflateReset or deflateEnd.

Z_FINISH can be used immediately after deflateInit if all the compression is to be done in a single step. In this case, avail_out must be at least the value returned by deflateBound (see below). If deflate does not return Z_STREAM_END, then it must be called again as described above.

deflate() sets strm->adler to the Adler32 checksum of all input read so far (that is, total_in bytes).

deflate() may update strm->data_type if it can make a good guess about the input data type (Z_BINARY or Z_TEXT). In doubt, the data is considered binary. This field is only for information purposes and does not affect the compression algorithm in any manner.

deflate() returns Z_OK if some progress has been made (more input processed or more output produced), Z_STREAM_END if all input has been consumed and all output has been produced (only when flush is set to Z_FINISH), Z_STREAM_ERROR if the stream state was inconsistent (for example if next_in or next_out was NULL), Z_BUF_ERROR if no progress is possible (for example avail_in or avail_out was zero). Note that Z_BUF_ERROR is not fatal, and deflate() can be called again with more input and more output space to continue compressing.

*/

```
ZEXTERN int ZEXPORT deflateEnd OF((z_stream * strm));
```

/*

All dynamically allocated data structures for this stream are freed. This function discards any unprocessed input and does not flush any pending output.

deflateEnd returns Z_OK if success, Z_STREAM_ERROR if the stream state was inconsistent, Z_DATA_ERROR if the stream was freed prematurely (some input or output was discarded). In the error case, msg may be set but then points to a static string (which must not be deallocated).

*/

/*

```
ZEXTERN int ZEXPORT inflateInit OF((z_stream * strm));
```

Initializes the internal stream state for decompression. The fields next_in, avail_in, zalloc, zfree and opaque must be initialized before by the caller. If next_in is not Z_NULL and avail_in is large enough (the exact value depends on the compression method), inflateInit determines the compression method from the zlib header and allocates all data structures accordingly; otherwise the allocation will be deferred to the first call of inflate. If zalloc and zfree are set to Z_NULL, inflateInit updates them to use default allocation functions.

inflateInit returns Z_OK if success, Z_MEM_ERROR if there was not enough memory, Z_VERSION_ERROR if the zlib library version is incompatible with the version assumed by the caller. msg is set to null if there is no error message. inflateInit does not perform any decompression apart from reading the zlib header if present: this will be done by inflate(). (So next_in and avail_in may be modified, but next_out and avail_out are unchanged.)

*/

```
ZEXTERN int ZEXPORT inflate OF((z_stream * strm, int flush));
```

/*

inflate decompresses as much data as possible, and stops when the input buffer becomes empty or the output buffer becomes full. It may introduce some output latency (reading input without producing any output) except when forced to flush.

The detailed semantics are as follows. inflate performs one or both of the

following actions:

- Decompress more input starting at `next_in` and update `next_in` and `avail_in` accordingly. If not all input can be processed (because there is not enough room in the output buffer), `next_in` is updated and processing will resume at this point for the next call of `inflate()`.
- Provide more output starting at `next_out` and update `next_out` and `avail_out` accordingly. `inflate()` provides as much output as possible, until there is no more input data or no more space in the output buffer (see below about the flush parameter).

Before the call of `inflate()`, the application should ensure that at least one of the actions is possible, by providing more input and/or consuming more output, and updating the `next_*` and `avail_*` values accordingly. The application can consume the uncompressed output when it wants, for example when the output buffer is full (`avail_out == 0`), or after each call of `inflate()`. If `inflate` returns `Z_OK` and with zero `avail_out`, it must be called again after making room in the output buffer because there might be more output pending.

The flush parameter of `inflate()` can be `Z_NO_FLUSH`, `Z_SYNC_FLUSH`, `Z_FINISH`, or `Z_BLOCK`. `Z_SYNC_FLUSH` requests that `inflate()` flush as much output as possible to the output buffer. `Z_BLOCK` requests that `inflate()` stop if and when it gets to the next deflate block boundary. When decoding the zlib or gzip format, this will cause `inflate()` to return immediately after the header and before the first block. When doing a raw inflate, `inflate()` will go ahead and process the first block, and will return when it gets to the end of that block, or when it runs out of data.

The `Z_BLOCK` option assists in appending to or combining deflate streams. Also to assist in this, on return `inflate()` will set `strm->data_type` to the number of unused bits in the last byte taken from `strm->next_in`, plus 64 if `inflate()` is currently decoding the last block in the deflate stream, plus 128 if `inflate()` returned immediately after decoding an end-of-block code or decoding the complete header up to just before the first byte of the deflate stream. The end-of-block will not be indicated until all of the uncompressed data from that block has been written to `strm->next_out`. The number of unused bits may in general be greater than seven, except when bit 7 of `data_type` is set, in which case the number of unused bits will be less than eight.

`inflate()` should normally be called until it returns `Z_STREAM_END` or an error. However if all decompression is to be performed in a single step (a single call of `inflate`), the parameter flush should be set to `Z_FINISH`. In this case all pending input is processed and all pending output is flushed; `avail_out` must be large enough to hold all the uncompressed data. (The size of the uncompressed data may have been saved by the compressor for this purpose.) The next operation on this stream must be `inflateEnd` to deallocate the decompression state. The use of `Z_FINISH` is never required, but can be used to inform `inflate` that a faster approach may be used for the single `inflate()` call.

In this implementation, `inflate()` always flushes as much output as possible to the output buffer, and always uses the faster approach on the first call. So the only effect of the flush parameter in this implementation is on the return value of `inflate()`, as noted below, or when it returns early because `Z_BLOCK` is used.

If a preset dictionary is needed after this call (see `inflateSetDictionary` below), `inflate` sets `strm->adler` to the Adler32 checksum of the dictionary chosen by the compressor and returns `Z_NEED_DICT`; otherwise it sets `strm->adler` to the Adler32 checksum of all output produced so far (that is, total_out bytes) and returns `Z_OK`, `Z_STREAM_END` or an error code as described below. At the end of the stream, `inflate()` checks that its computed Adler32 checksum is equal to that saved by the compressor and returns `Z_STREAM_END` only if the checksum is correct.

`inflate()` will decompress and check either zlib-wrapped or gzip-wrapped deflate data. The header type is detected automatically. Any information contained in the gzip header is not retained, so applications that need that information should instead use raw `inflate`, see `inflateInit2()` below, or `inflateBack()` and perform their own processing of the gzip header and trailer.

`inflate()` returns `Z_OK` if some progress has been made (more input processed or more output produced), `Z_STREAM_END` if the end of the compressed data has been reached and all uncompressed output has been produced, `Z_NEED_DICT` if a preset dictionary is needed at this point, `Z_DATA_ERROR` if the input data was corrupted (input stream not conforming to the zlib format or incorrect check value), `Z_STREAM_ERROR` if the stream structure was inconsistent (for example if `next_in` or `next_out` was `NULL`), `Z_MEM_ERROR` if there was not enough memory, `Z_BUF_ERROR` if no progress is possible or if there was not enough room in the output buffer when `Z_FINISH` is used. Note that `Z_BUF_ERROR` is not fatal, and `inflate()` can be called again with more input and more output space to continue decompressing. If `Z_DATA_ERROR` is returned, the application may then call `inflateSync()` to look for a good compression block if a partial recovery of the data is desired.

*/

```
ZEXTERN int ZEXPORT inflateEnd OF((z_streamp strm));
```

/*

All dynamically allocated data structures for this stream are freed. This function discards any unprocessed input and does not flush any pending output.

`inflateEnd` returns `Z_OK` if success, `Z_STREAM_ERROR` if the stream state was inconsistent. In the error case, `msg` may be set but then points to a static string (which must not be deallocated).

*/

/* Advanced functions */

/*

The following functions are needed only in some special applications.

*/

/*

```
ZEXTERN int ZEXPORT deflateInit2 OF((z_streamp strm,
                                     int level,
                                     int method,
                                     int windowBits,
                                     int memLevel,
                                     int strategy));
```

This is another version of `deflateInit` with more compression options. The fields `next_in`, `zalloc`, `zfree` and `opaque` must be initialized before by the caller.

The `method` parameter is the compression method. It must be `Z_DEFLATED` in this version of the library.

The `windowBits` parameter is the base two logarithm of the window size (the size of the history buffer). It should be in the range 8..15 for this version of the library. Larger values of this parameter result in better compression at the expense of memory usage. The default value is 15 if `deflateInit` is used instead.

`windowBits` can also be -8..-15 for raw deflate. In this case, `-windowBits` determines the window size. `deflate()` will then generate raw deflate data with no zlib header or trailer, and will not compute an `adler32` check value.

`windowBits` can also be greater than 15 for optional gzip encoding. Add 16 to `windowBits` to write a simple gzip header and trailer around the compressed data instead of a zlib wrapper. The gzip header will have no file name, no extra data, no comment, no modification time (set to zero), no header crc, and the operating system will be set to 255 (unknown). If a gzip stream is being written, `strm->adler` is a `crc32` instead of an `adler32`.

The `memLevel` parameter specifies how much memory should be allocated for the internal compression state. `memLevel=1` uses minimum memory but is slow and reduces compression ratio; `memLevel=9` uses maximum memory for optimal speed. The default value is 8. See `zconf.h` for total memory usage as a function of `windowBits` and `memLevel`.

The `strategy` parameter is used to tune the compression algorithm. Use the value `Z_DEFAULT_STRATEGY` for normal data, `Z_FILTERED` for data produced by a

filter (or predictor), Z_HUFFMAN_ONLY to force Huffman encoding only (no string match), or Z_RLE to limit match distances to one (run-length encoding). Filtered data consists mostly of small values with a somewhat random distribution. In this case, the compression algorithm is tuned to compress them better. The effect of Z_FILTERED is to force more Huffman coding and less string matching; it is somewhat intermediate between Z_DEFAULT and Z_HUFFMAN_ONLY. Z_RLE is designed to be almost as fast as Z_HUFFMAN_ONLY, but give better compression for PNG image data. The strategy parameter only affects the compression ratio but not the correctness of the compressed output even if it is not set appropriately. Z_FIXED prevents the use of dynamic Huffman codes, allowing for a simpler decoder for special applications.

deflateInit2 returns Z_OK if success, Z_MEM_ERROR if there was not enough memory, Z_STREAM_ERROR if a parameter is invalid (such as an invalid method). msg is set to null if there is no error message. deflateInit2 does not perform any compression: this will be done by deflate().

*/

```
ZEXTERN int ZEXPORT deflateSetDictionary OF((z_streamp strm,
                                           const Bytef *dictionary,
                                           uInt dictLength));
```

/*

Initializes the compression dictionary from the given byte sequence without producing any compressed output. This function must be called immediately after deflateInit, deflateInit2 or deflateReset, before any call of deflate. The compressor and decompressor must use exactly the same dictionary (see inflateSetDictionary).

The dictionary should consist of strings (byte sequences) that are likely to be encountered later in the data to be compressed, with the most commonly used strings preferably put towards the end of the dictionary. Using a dictionary is most useful when the data to be compressed is short and can be predicted with good accuracy; the data can then be compressed better than with the default empty dictionary.

Depending on the size of the compression data structures selected by deflateInit or deflateInit2, a part of the dictionary may in effect be discarded, for example if the dictionary is larger than the window size in deflate or deflate2. Thus the strings most likely to be useful should be put at the end of the dictionary, not at the front. In addition, the current implementation of deflate will use at most the window size minus 262 bytes of the provided dictionary.

Upon return of this function, strm->adler is set to the Adler32 value of the dictionary; the decompressor may later use this value to determine which dictionary has been used by the compressor. (The Adler32 value applies to the whole dictionary even if only a subset of the dictionary is actually used by the compressor.) If a raw deflate was requested, then the Adler32 value is not computed and strm->adler is not set.

deflateSetDictionary returns Z_OK if success, or Z_STREAM_ERROR if a parameter is invalid (such as NULL dictionary) or the stream state is inconsistent (for example if deflate has already been called for this stream or if the compression method is bsort). deflateSetDictionary does not perform any compression: this will be done by deflate().

*/

```
ZEXTERN int ZEXPORT deflateCopy OF((z_streamp dest,
                                   z_streamp source));
```

/*

Sets the destination stream as a complete copy of the source stream.

This function can be useful when several compression strategies will be tried, for example when there are several ways of pre-processing the input data with a filter. The streams that will be discarded should then be freed by calling deflateEnd. Note that deflateCopy duplicates the internal compression state which can be quite large, so this strategy is slow and can consume lots of memory.

deflateCopy returns Z_OK if success, Z_MEM_ERROR if there was not enough memory, Z_STREAM_ERROR if the source stream state was inconsistent (such as zalloc being NULL). msg is left unchanged in both source and destination.

```
*/

ZEXTERN int ZEXPORT deflateReset OF((z_streamp strm));
/*
   This function is equivalent to deflateEnd followed by deflateInit,
   but does not free and reallocate all the internal compression state.
   The stream will keep the same compression level and any other attributes
   that may have been set by deflateInit2.

   deflateReset returns Z_OK if success, or Z_STREAM_ERROR if the source
   stream state was inconsistent (such as zalloc or state being NULL).
*/

ZEXTERN int ZEXPORT deflateParams OF((z_streamp strm,
                                     int level,
                                     int strategy));
/*
   Dynamically update the compression level and compression strategy. The
   interpretation of level and strategy is as in deflateInit2. This can be
   used to switch between compression and straight copy of the input data, or
   to switch to a different kind of input data requiring a different
   strategy. If the compression level is changed, the input available so far
   is compressed with the old level (and may be flushed); the new level will
   take effect only at the next call of deflate().

   Before the call of deflateParams, the stream state must be set as for
   a call of deflate(), since the currently available input may have to
   be compressed and flushed. In particular, strm->avail_out must be non-zero.

   deflateParams returns Z_OK if success, Z_STREAM_ERROR if the source
   stream state was inconsistent or if a parameter was invalid, Z_BUF_ERROR
   if strm->avail_out was zero.
*/

ZEXTERN int ZEXPORT deflateTune OF((z_streamp strm,
                                   int good_length,
                                   int max_lazy,
                                   int nice_length,
                                   int max_chain));
/*
   Fine tune deflate's internal compression parameters. This should only be
   used by someone who understands the algorithm used by zlib's deflate for
   searching for the best matching string, and even then only by the most
   fanatic optimizer trying to squeeze out the last compressed bit for their
   specific input data. Read the deflate.c source code for the meaning of the
   max_lazy, good_length, nice_length, and max_chain parameters.

   deflateTune() can be called after deflateInit() or deflateInit2(), and
   returns Z_OK on success, or Z_STREAM_ERROR for an invalid deflate stream.
*/

ZEXTERN uLong ZEXPORT deflateBound OF((z_streamp strm,
                                      uLong sourceLen));
/*
   deflateBound() returns an upper bound on the compressed size after
   deflation of sourceLen bytes. It must be called after deflateInit()
   or deflateInit2(). This would be used to allocate an output buffer
   for deflation in a single pass, and so would be called before deflate().
*/

ZEXTERN int ZEXPORT deflatePrime OF((z_streamp strm,
                                    int bits,
                                    int value));
/*
   deflatePrime() inserts bits in the deflate output stream. The intent
   is that this function is used to start off the deflate output with the
   bits leftover from a previous deflate stream when appending to it. As such,
   this function can only be used for raw deflate, and must be used before the
   first deflate() call after a deflateInit2() or deflateReset(). bits must be
   less than or equal to 16, and that many of the least significant bits of
   value will be inserted in the output.

   deflatePrime returns Z_OK if success, or Z_STREAM_ERROR if the source
   stream state was inconsistent.
*/
```


*/

```
ZEXTERN int ZEXPORT deflateSetHeader OF((z_streamp strm,
                                         gz_headerp head));
```

/*

`deflateSetHeader()` provides gzip header information for when a gzip stream is requested by `deflateInit2()`. `deflateSetHeader()` may be called after `deflateInit2()` or `deflateReset()` and before the first call of `deflate()`. The text, time, os, extra field, name, and comment information in the provided `gz_header` structure are written to the gzip header (xflag is ignored -- the extra flags are set according to the compression level). The caller must assure that, if not `Z_NULL`, name and comment are terminated with a zero byte, and that if extra is not `Z_NULL`, that extra_len bytes are available there. If `hcrc` is true, a gzip header crc is included. Note that the current versions of the command-line version of gzip (up through version 1.3.x) do not support header crc's, and will report that it is a "multi-part gzip file" and give up.

If `deflateSetHeader` is not used, the default gzip header has text false, the time set to zero, and os set to 255, with no extra, name, or comment fields. The gzip header is returned to the default state by `deflateReset()`.

`deflateSetHeader` returns `Z_OK` if success, or `Z_STREAM_ERROR` if the source stream state was inconsistent.

*/

/*

```
ZEXTERN int ZEXPORT inflateInit2 OF((z_streamp strm,
                                     int windowBits));
```

This is another version of `inflateInit` with an extra parameter. The fields `next_in`, `avail_in`, `zalloc`, `zfree` and `opaque` must be initialized before by the caller.

The `windowBits` parameter is the base two logarithm of the maximum window size (the size of the history buffer). It should be in the range 8..15 for this version of the library. The default value is 15 if `inflateInit` is used instead. `windowBits` must be greater than or equal to the `windowBits` value provided to `deflateInit2()` while compressing, or it must be equal to 15 if `deflateInit2()` was not used. If a compressed stream with a larger window size is given as input, `inflate()` will return with the error code `Z_DATA_ERROR` instead of trying to allocate a larger window.

`windowBits` can also be -8..-15 for raw inflate. In this case, `-windowBits` determines the window size. `inflate()` will then process raw deflate data, not looking for a zlib or gzip header, not generating a check value, and not looking for any check values for comparison at the end of the stream. This is for use with other formats that use the deflate compressed data format such as zip. Those formats provide their own check values. If a custom format is developed using the raw deflate format for compressed data, it is recommended that a check value such as an `adler32` or a `crc32` be applied to the uncompressed data as is done in the zlib, gzip, and zip formats. For most applications, the zlib format should be used as is. Note that comments above on the use in `deflateInit2()` applies to the magnitude of `windowBits`.

`windowBits` can also be greater than 15 for optional gzip decoding. Add 32 to `windowBits` to enable zlib and gzip decoding with automatic header detection, or add 16 to decode only the gzip format (the zlib format will return a `Z_DATA_ERROR`). If a gzip stream is being decoded, `strm->adler` is a `crc32` instead of an `adler32`.

`inflateInit2` returns `Z_OK` if success, `Z_MEM_ERROR` if there was not enough memory, `Z_STREAM_ERROR` if a parameter is invalid (such as a null `strm`). `msg` is set to null if there is no error message. `inflateInit2` does not perform any decompression apart from reading the zlib header if present: this will be done by `inflate()`. (So `next_in` and `avail_in` may be modified, but `next_out` and `avail_out` are unchanged.)

*/

```
ZEXTERN int ZEXPORT inflateSetDictionary OF((z_streamp strm,
                                             const Bytef *dictionary,
                                             uInt dictLength));
```

/*

Initializes the decompression dictionary from the given uncompressed byte

sequence. This function must be called immediately after a call of `inflate`, if that call returned `Z_NEED_DICT`. The dictionary chosen by the compressor can be determined from the `adler32` value returned by that call of `inflate`. The compressor and decompressor must use exactly the same dictionary (see `deflateSetDictionary`). For raw `inflate`, this function can be called immediately after `inflateInit2()` or `inflateReset()` and before any call of `inflate()` to set the dictionary. The application must insure that the dictionary that was used for compression is provided.

`inflateSetDictionary` returns `Z_OK` if success, `Z_STREAM_ERROR` if a parameter is invalid (such as `NULL` dictionary) or the stream state is inconsistent, `Z_DATA_ERROR` if the given dictionary doesn't match the expected one (incorrect `adler32` value). `inflateSetDictionary` does not perform any decompression: this will be done by subsequent calls of `inflate()`.

*/

ZEXTERN int ZEXPORT inflateSync OF((z_streamp strm));

/*

Skips invalid compressed data until a full flush point (see above the description of `deflate` with `Z_FULL_FLUSH`) can be found, or until all available input is skipped. No output is provided.

`inflateSync` returns `Z_OK` if a full flush point has been found, `Z_BUF_ERROR` if no more input was provided, `Z_DATA_ERROR` if no flush point has been found, or `Z_STREAM_ERROR` if the stream structure was inconsistent. In the success case, the application may save the current value of `total_in` which indicates where valid compressed data was found. In the error case, the application may repeatedly call `inflateSync`, providing more input each time, until success or end of the input data.

*/

ZEXTERN int ZEXPORT inflateCopy OF((z_streamp dest,
z_streamp source));

/*

Sets the destination stream as a complete copy of the source stream.

This function can be useful when randomly accessing a large stream. The first pass through the stream can periodically record the `inflate` state, allowing restarting `inflate` at those points when randomly accessing the stream.

`inflateCopy` returns `Z_OK` if success, `Z_MEM_ERROR` if there was not enough memory, `Z_STREAM_ERROR` if the source stream state was inconsistent (such as `zalloc` being `NULL`). `msg` is left unchanged in both source and destination.

*/

ZEXTERN int ZEXPORT inflateReset OF((z_streamp strm));

/*

This function is equivalent to `inflateEnd` followed by `inflateInit`, but does not free and reallocate all the internal decompression state. The stream will keep attributes that may have been set by `inflateInit2`.

`inflateReset` returns `Z_OK` if success, or `Z_STREAM_ERROR` if the source stream state was inconsistent (such as `zalloc` or state being `NULL`).

*/

ZEXTERN int ZEXPORT inflatePrime OF((z_streamp strm,
int bits,
int value));

/*

This function inserts bits in the `inflate` input stream. The intent is that this function is used to start inflating at a bit position in the middle of a byte. The provided bits will be used before any bytes are used from `next_in`. This function should only be used with raw `inflate`, and should be used before the first `inflate()` call after `inflateInit2()` or `inflateReset()`. `bits` must be less than or equal to 16, and that many of the least significant bits of `value` will be inserted in the input.

`inflatePrime` returns `Z_OK` if success, or `Z_STREAM_ERROR` if the source stream state was inconsistent.

*/

```
ZEXTERN int ZEXPORT inflateGetHeader OF((z_streamp strm,
                                         gz_headerp head));
```

```
/*
```

inflateGetHeader() requests that gzip header information be stored in the provided `gz_header` structure. *inflateGetHeader()* may be called after *inflateInit2()* or *inflateReset()*, and before the first call of *inflate()*. As *inflate()* processes the gzip stream, `head->done` is zero until the header is completed, at which time `head->done` is set to one. If a zlib stream is being decoded, then `head->done` is set to -1 to indicate that there will be no gzip header information forthcoming. Note that `Z_BLOCK` can be used to force *inflate()* to return immediately after header processing is complete and before any actual data is decompressed.

The `text`, `time`, `xflags`, and `os` fields are filled in with the gzip header contents. `hcrc` is set to true if there is a header CRC. (The header CRC was valid if `done` is set to one.) If `extra` is not `Z_NULL`, then `extra_max` contains the maximum number of bytes to write to `extra`. Once `done` is true, `extra_len` contains the actual `extra` field length, and `extra` contains the `extra` field, or that field truncated if `extra_max` is less than `extra_len`. If `name` is not `Z_NULL`, then up to `name_max` characters are written there, terminated with a zero unless the length is greater than `name_max`. If `comment` is not `Z_NULL`, then up to `comm_max` characters are written there, terminated with a zero unless the length is greater than `comm_max`. When any of `extra`, `name`, or `comment` are not `Z_NULL` and the respective field is not present in the header, then that field is set to `Z_NULL` to signal its absence. This allows the use of *deflateSetHeader()* with the returned structure to duplicate the header. However if those fields are set to allocated memory, then the application will need to save those pointers elsewhere so that they can be eventually freed.

If *inflateGetHeader* is not used, then the header information is simply discarded. The header is always checked for validity, including the header CRC if present. *inflateReset()* will reset the process to discard the header information. The application would need to call *inflateGetHeader()* again to retrieve the header from the next gzip stream.

inflateGetHeader returns `Z_OK` if success, or `Z_STREAM_ERROR` if the source stream state was inconsistent.

```
*/
```

```
/*
```

```
ZEXTERN int ZEXPORT inflateBackInit OF((z_streamp strm, int windowBits,
                                         unsigned char FAR *window));
```

Initialize the internal stream state for decompression using *inflateBack()* calls. The fields `zalloc`, `zfree` and `opaque` in `strm` must be initialized before the call. If `zalloc` and `zfree` are `Z_NULL`, then the default library-derived memory allocation routines are used. `windowBits` is the base two logarithm of the window size, in the range 8..15. `window` is a caller supplied buffer of that size. Except for special applications where it is assured that *deflate* was used with small window sizes, `windowBits` must be 15 and a 32K byte window must be supplied to be able to decompress general *deflate* streams.

See *inflateBack()* for the usage of these routines.

inflateBackInit will return `Z_OK` on success, `Z_STREAM_ERROR` if any of the parameters are invalid, `Z_MEM_ERROR` if the internal state could not be allocated, or `Z_VERSION_ERROR` if the version of the library does not match the version of the header file.

```
*/
```

```
typedef unsigned (*in_func) OF((void FAR *, unsigned char FAR * FAR *));
typedef int (*out_func) OF((void FAR *, unsigned char FAR *, unsigned));
```

```
ZEXTERN int ZEXPORT inflateBack OF((z_streamp strm,
                                     in_func in, void FAR *in_desc,
                                     out_func out, void FAR *out_desc));
```

```
/*
```

inflateBack() does a raw *inflate* with a single call using a call-back interface for input and output. This is more efficient than *inflate()* for file i/o applications in that it avoids copying between the output and the sliding window by simply making the window itself the output buffer. This function trusts the application to not change the output buffer passed by

the output function, at least until inflateBack() returns.

inflateBackInit() must be called first to allocate the internal state and to initialize the state with the user-provided window buffer. inflateBack() may then be used multiple times to inflate a complete, raw deflate stream with each call. inflateBackEnd() is then called to free the allocated state.

A raw deflate stream is one with no zlib or gzip header or trailer. This routine would normally be used in a utility that reads zip or gzip files and writes out uncompressed files. The utility would decode the header and process the trailer on its own, hence this routine expects only the raw deflate stream to decompress. This is different from the normal behavior of inflate(), which expects either a zlib or gzip header and trailer around the deflate stream.

inflateBack() uses two subroutines supplied by the caller that are then called by inflateBack() for input and output. inflateBack() calls those routines until it reads a complete deflate stream and writes out all of the uncompressed data, or until it encounters an error. The function's parameters and return types are defined above in the in_func and out_func typedefs. inflateBack() will call in(in_desc, &buf) which should return the number of bytes of provided input, and a pointer to that input in buf. If there is no input available, in() must return zero--buf is ignored in that case--and inflateBack() will return a buffer error. inflateBack() will call out(out_desc, buf, len) to write the uncompressed data buf[0..len-1]. out() should return zero on success, or non-zero on failure. If out() returns non-zero, inflateBack() will return with an error. Neither in() nor out() are permitted to change the contents of the window provided to inflateBackInit(), which is also the buffer that out() uses to write from. The length written by out() will be at most the window size. Any non-zero amount of input may be provided by in().

For convenience, inflateBack() can be provided input on the first call by setting strm->next_in and strm->avail_in. If that input is exhausted, then in() will be called. Therefore strm->next_in must be initialized before calling inflateBack(). If strm->next_in is Z_NULL, then in() will be called immediately for input. If strm->next_in is not Z_NULL, then strm->avail_in must also be initialized, and then if strm->avail_in is not zero, input will initially be taken from strm->next_in[0 .. strm->avail_in - 1].

The in_desc and out_desc parameters of inflateBack() is passed as the first parameter of in() and out() respectively when they are called. These descriptors can be optionally used to pass any information that the caller-supplied in() and out() functions need to do their job.

On return, inflateBack() will set strm->next_in and strm->avail_in to pass back any unused input that was provided by the last in() call. The return values of inflateBack() can be Z_STREAM_END on success, Z_BUF_ERROR if in() or out() returned an error, Z_DATA_ERROR if there was a format error in the deflate stream (in which case strm->msg is set to indicate the nature of the error), or Z_STREAM_ERROR if the stream was not properly initialized. In the case of Z_BUF_ERROR, an input or output error can be distinguished using strm->next_in which will be Z_NULL only if in() returned an error. If strm->next is not Z_NULL, then the Z_BUF_ERROR was due to out() returning non-zero. (in() will always be called before out(), so strm->next_in is assured to be defined if out() returns non-zero.) Note that inflateBack() cannot return Z_OK.

```
*/
ZEXTERN int ZEXPORT inflateBackEnd OF((z_streamp strm));
/*
```

All memory allocated by inflateBackInit() is freed.

inflateBackEnd() returns Z_OK on success, or Z_STREAM_ERROR if the stream state was inconsistent.

```
*/
ZEXTERN uLong ZEXPORT zlibCompileFlags OF((void));
/* Return flags indicating compile-time options.
```

Type sizes, two bits each, 00 = 16 bits, 01 = 32, 10 = 64, 11 = other:
 1.0: size of uInt
 3.2: size of uLong

5.4: size of voidpf (pointer)
 7.6: size of z_off_t

Compiler, assembler, and debug options:

8: DEBUG
 9: ASMV or ASMINF -- use ASM code
 10: ZLIB_WINAPI -- exported functions use the WINAPI calling convention
 11: 0 (reserved)

One-time table building (smaller code, but not thread-safe if true):

12: BUILDFIXED -- build static block decoding tables when needed
 13: DYNAMIC_CRC_TABLE -- build CRC calculation tables when needed
 14,15: 0 (reserved)

Library content (indicates missing functionality):

16: NO_GZCOMPRESS -- gz* functions cannot compress (to avoid linking deflate code when not needed)
 17: NO_GZIP -- deflate can't write gzip streams, and inflate can't detect and decode gzip streams (to avoid linking crc code)
 18-19: 0 (reserved)

Operation variations (changes in library functionality):

20: PKZIP_BUG_WORKAROUND -- slightly more permissive inflate
 21: FASTEST -- deflate algorithm with only one, lowest compression level
 22,23: 0 (reserved)

The sprintf variant used by gzprintf (zero is best):

24: 0 = vs*, 1 = s* -- 1 means limited to 20 arguments after the format
 25: 0 = *nprintf, 1 = *printf -- 1 means gzprintf() not secure!
 26: 0 = returns value, 1 = void -- 1 means inferred string length returned

Remainder:

27-31: 0 (reserved)

*/

/* utility functions */

/*

The following utility functions are implemented on top of the basic stream-oriented functions. To simplify the interface, some default options are assumed (compression level and memory usage, standard memory allocation functions). The source code of these utility functions can easily be modified if you need special options.

*/

ZEXTERN int ZEXPORT compress OF((Bytef *dest, uLongf *destLen,
 const Bytef *source, uLong sourceLen));

/*

Compresses the source buffer into the destination buffer. sourceLen is the byte length of the source buffer. Upon entry, destLen is the total size of the destination buffer, which must be at least the value returned by compressBound(sourceLen). Upon exit, destLen is the actual size of the compressed buffer.

This function can be used to compress a whole file at once if the input file is mmap'ed.

compress returns Z_OK if success, Z_MEM_ERROR if there was not enough memory, Z_BUF_ERROR if there was not enough room in the output buffer.

*/

ZEXTERN int ZEXPORT compress2 OF((Bytef *dest, uLongf *destLen,
 const Bytef *source, uLong sourceLen,
 int level));

/*

Compresses the source buffer into the destination buffer. The level parameter has the same meaning as in deflateInit. sourceLen is the byte length of the source buffer. Upon entry, destLen is the total size of the destination buffer, which must be at least the value returned by compressBound(sourceLen). Upon exit, destLen is the actual size of the compressed buffer.

compress2 returns Z_OK if success, Z_MEM_ERROR if there was not enough memory, Z_BUF_ERROR if there was not enough room in the output buffer,

```

    Z_STREAM_ERROR if the level parameter is invalid.
*/

ZEXTERN uLong ZEXPORT compressBound OF((uLong sourceLen));
/*
    compressBound() returns an upper bound on the compressed size after
    compress() or compress2() on sourceLen bytes. It would be used before
    a compress() or compress2() call to allocate the destination buffer.
*/

ZEXTERN int ZEXPORT uncompress OF((Bytef *dest, uLongf *destLen,
                                   const Bytef *source, uLong sourceLen));
/*
    Decompresses the source buffer into the destination buffer. sourceLen is
    the byte length of the source buffer. Upon entry, destLen is the total
    size of the destination buffer, which must be large enough to hold the
    entire uncompressed data. (The size of the uncompressed data must have
    been saved previously by the compressor and transmitted to the decompressor
    by some mechanism outside the scope of this compression library.)
    Upon exit, destLen is the actual size of the compressed buffer.
    This function can be used to decompress a whole file at once if the
    input file is mmap'ed.

    uncompress returns Z_OK if success, Z_MEM_ERROR if there was not
    enough memory, Z_BUF_ERROR if there was not enough room in the output
    buffer, or Z_DATA_ERROR if the input data was corrupted or incomplete.
*/

typedef voidp gzFile;

ZEXTERN gzFile ZEXPORT gzopen OF((const char *path, const char *mode));
/*
    Opens a gzip (.gz) file for reading or writing. The mode parameter
    is as in fopen ("rb" or "wb") but can also include a compression level
    ("wb9") or a strategy: 'f' for filtered data as in "wb6f", 'h' for
    Huffman only compression as in "wb1h", or 'R' for run-length encoding
    as in "wb1R". (See the description of deflateInit2 for more information
    about the strategy parameter.)

    gzopen can be used to read a file which is not in gzip format; in this
    case gzread will directly read from the file without decompression.

    gzopen returns NULL if the file could not be opened or if there was
    insufficient memory to allocate the (de)compression state; errno
    can be checked to distinguish the two cases (if errno is zero, the
    zlib error is Z_MEM_ERROR). */

ZEXTERN gzFile ZEXPORT gzdopen OF((int fd, const char *mode));
/*
    gzdopen() associates a gzFile with the file descriptor fd. File
    descriptors are obtained from calls like open, dup, creat, pipe or
    fileno (in the file has been previously opened with fopen).
    The mode parameter is as in gzopen.
    The next call of gzclose on the returned gzFile will also close the
    file descriptor fd, just like fclose(fdopen(fd), mode) closes the file
    descriptor fd. If you want to keep fd open, use gzdopen(dup(fd), mode).
    gzdopen returns NULL if there was insufficient memory to allocate
    the (de)compression state.
*/

ZEXTERN int ZEXPORT gzsetparams OF((gzFile file, int level, int strategy));
/*
    Dynamically update the compression level or strategy. See the description
    of deflateInit2 for the meaning of these parameters.
    gzsetparams returns Z_OK if success, or Z_STREAM_ERROR if the file was not
    opened for writing.
*/

ZEXTERN int ZEXPORT gzread OF((gzFile file, voidp buf, unsigned len));
/*
    Reads the given number of uncompressed bytes from the compressed file.
    If the input file was not in gzip format, gzread copies the given number
    of bytes into the buffer.

```

*gzread returns the number of uncompressed bytes actually read (0 for end of file, -1 for error). */*

```
ZEXTERN int ZEXPORT      gzwrite OF((gzFile file,
                                     voidpc buf, unsigned len));
```

/
Writes the given number of uncompressed bytes into the compressed file.
gzwrite returns the number of uncompressed bytes actually written
(0 in case of error).
/

```
ZEXTERN int ZEXPORTVA    gzprintf OF((gzFile file, const char *format, ...));
```

/
Converts, formats, and writes the args to the compressed file under
control of the format string, as in fprintf. gzprintf returns the number of
uncompressed bytes actually written (0 in case of error). The number of
uncompressed bytes written is limited to 4095. The caller should assure that
this limit is not exceeded. If it is exceeded, then gzprintf() will return
return an error (0) with nothing written. In this case, there may also be a
buffer overflow with unpredictable consequences, which is possible only if
zlib was compiled with the insecure functions sprintf() or vsprintf()
because the secure snprintf() or vsnprintf() functions were not available.
/

```
ZEXTERN int ZEXPORT      gzputs OF((gzFile file, const char *s));
```

/
Writes the given null-terminated string to the compressed file, excluding
the terminating null character.
gzputs returns the number of characters written, or -1 in case of error.
/

```
ZEXTERN char * ZEXPORT    gzgets OF((gzFile file, char *buf, int len));
```

/
Reads bytes from the compressed file until len-1 characters are read, or
a newline character is read and transferred to buf, or an end-of-file
condition is encountered. The string is then terminated with a null
character.
gzgets returns buf, or Z_NULL in case of error.
/

```
ZEXTERN int ZEXPORT      gzputc OF((gzFile file, int c));
```

/
Writes c, converted to an unsigned char, into the compressed file.
gzputc returns the value that was written, or -1 in case of error.
/

```
ZEXTERN int ZEXPORT      gzgetc OF((gzFile file));
```

/
Reads one byte from the compressed file. gzgetc returns this byte
or -1 in case of end of file or error.
/

```
ZEXTERN int ZEXPORT      gzungetc OF((int c, gzFile file));
```

/
Push one character back onto the stream to be read again later.
Only one character of push-back is allowed. gzungetc() returns the
character pushed, or -1 on failure. gzungetc() will fail if a
character has been pushed but not read yet, or if c is -1. The pushed
character will be discarded if the stream is repositioned with gzseek()
or gzrewind().
/

```
ZEXTERN int ZEXPORT      gzflush OF((gzFile file, int flush));
```

/
Flushes all pending output into the compressed file. The parameter
flush is as in the deflate() function. The return value is the zlib
error number (see function gzerror below). gzflush returns Z_OK if
the flush parameter is Z_FINISH and all output could be flushed.
gzflush should be called only when strictly necessary because it can
degrade compression.
/

```
ZEXTERN z_off_t ZEXPORT    gzseek OF((gzFile file,
                                     z_off_t offset, int whence));
```

```
/*
    Sets the starting position for the next gzread or gzwrite on the
    given compressed file. The offset represents a number of bytes in the
    uncompressed data stream. The whence parameter is defined as in lseek(2);
    the value SEEK_END is not supported.
    If the file is opened for reading, this function is emulated but can be
    extremely slow. If the file is opened for writing, only forward seeks are
    supported; gzseek then compresses a sequence of zeroes up to the new
    starting position.

    gzseek returns the resulting offset location as measured in bytes from
    the beginning of the uncompressed stream, or -1 in case of error, in
    particular if the file is opened for writing and the new starting position
    would be before the current position.
*/

ZEXTERN int ZEXPORT      gzrewind OF((gzFile file));
/*
    Rewinds the given file. This function is supported only for reading.

    gzrewind(file) is equivalent to (int)gzseek(file, 0L, SEEK_SET)
*/

ZEXTERN z_off_t ZEXPORT  gztell OF((gzFile file));
/*
    Returns the starting position for the next gzread or gzwrite on the
    given compressed file. This position represents a number of bytes in the
    uncompressed data stream.

    gtell(file) is equivalent to gzseek(file, 0L, SEEK_CUR)
*/

ZEXTERN int ZEXPORT  gzeof OF((gzFile file));
/*
    Returns 1 when EOF has previously been detected reading the given
    input stream, otherwise zero.
*/

ZEXTERN int ZEXPORT  gzdirect OF((gzFile file));
/*
    Returns 1 if file is being read directly without decompression, otherwise
    zero.
*/

ZEXTERN int ZEXPORT  gzclose OF((gzFile file));
/*
    Flushes all pending output if necessary, closes the compressed file
    and deallocates all the (de)compression state. The return value is the zlib
    error number (see function gzerror below).
*/

ZEXTERN const char * ZEXPORT gzerror OF((gzFile file, int *errnum));
/*
    Returns the error message for the last error which occurred on the
    given compressed file. errnum is set to zlib error number. If an
    error occurred in the file system and not in the compression library,
    errnum is set to Z_ERRNO and the application may consult errno
    to get the exact error code.
*/

ZEXTERN void ZEXPORT  gzclearerr OF((gzFile file));
/*
    Clears the error and end-of-file flags for file. This is analogous to the
    clearerr() function in stdio. This is useful for continuing to read a gzip
    file that is being written concurrently.

    /* checksum functions */

/*
    These functions are not related to compression but are exported
    anyway because they might be useful in applications using the
    compression library.
*/
```



```

ZEXTERN uLong ZEXPORT Adler32 OF((uLong Adler, const Bytef *buf, uInt len));
/*
    Update a running Adler-32 checksum with the bytes buf[0..len-1] and
    return the updated checksum. If buf is NULL, this function returns
    the required initial value for the checksum.
    An Adler-32 checksum is almost as reliable as a CRC32 but can be computed
    much faster. Usage example:

        uLong Adler = Adler32(0L, Z_NULL, 0);

        while (read_buffer(buffer, length) != EOF) {
            Adler = Adler32(Adler, buffer, length);
        }
        if (Adler != original_Adler) error();
*/

ZEXTERN uLong ZEXPORT Adler32_combine OF((uLong Adler1, uLong Adler2,
                                         z_off_t len2));
/*
    Combine two Adler-32 checksums into one. For two sequences of bytes, seq1
    and seq2 with lengths len1 and len2, Adler-32 checksums were calculated for
    each, Adler1 and Adler2. Adler32_combine() returns the Adler-32 checksum of
    seq1 and seq2 concatenated, requiring only Adler1, Adler2, and len2.
*/

ZEXTERN uLong ZEXPORT Crc32 OF((uLong Crc, const Bytef *buf, uInt len));
/*
    Update a running CRC-32 with the bytes buf[0..len-1] and return the
    updated CRC-32. If buf is NULL, this function returns the required initial
    value for the for the Crc. Pre- and post-conditioning (one's complement) is
    performed within this function so it shouldn't be done by the application.
    Usage example:

        uLong Crc = Crc32(0L, Z_NULL, 0);

        while (read_buffer(buffer, length) != EOF) {
            Crc = Crc32(Crc, buffer, length);
        }
        if (Crc != original_Crc) error();
*/

ZEXTERN uLong ZEXPORT Crc32_combine OF((uLong Crc1, uLong Crc2, z_off_t len2));
/*
    Combine two CRC-32 check values into one. For two sequences of bytes,
    seq1 and seq2 with lengths len1 and len2, CRC-32 check values were
    calculated for each, Crc1 and Crc2. Crc32_combine() returns the CRC-32
    check value of seq1 and seq2 concatenated, requiring only Crc1, Crc2, and
    len2.
*/

        /* various hacks, don't look :) */

/* deflateInit and inflateInit are macros to allow checking the zlib version
 * and the compiler's view of z_stream:
 */
ZEXTERN int ZEXPORT deflateInit_ OF((z_stream *strm, int level,
                                     const char *version, int stream_size));
ZEXTERN int ZEXPORT inflateInit_ OF((z_stream *strm,
                                     const char *version, int stream_size));
ZEXTERN int ZEXPORT deflateInit2_ OF((z_stream *strm, int level, int method,
                                     int windowBits, int memLevel,
                                     int strategy, const char *version,
                                     int stream_size));
ZEXTERN int ZEXPORT inflateInit2_ OF((z_stream *strm, int windowBits,
                                     const char *version, int stream_size));
ZEXTERN int ZEXPORT inflateBackInit_ OF((z_stream *strm, int windowBits,
                                         unsigned char FAR *window,
                                         const char *version,
                                         int stream_size));

#define deflateInit(strm, level) \
    deflateInit_((strm), (level), ZLIB_VERSION, sizeof(z_stream))

```

```
#define inflateInit(strm) \
    inflateInit_((strm),          ZLIB_VERSION, sizeof(z_stream))
#define deflateInit2(strm, level, method, windowBits, memLevel, strategy) \
    deflateInit2_((strm),(level),(method),(windowBits),(memLevel),\
        (strategy),          ZLIB_VERSION, sizeof(z_stream))
#define inflateInit2(strm, windowBits) \
    inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))
#define inflateBackInit(strm, windowBits, window) \
    inflateBackInit_((strm), (windowBits), (window), \
        ZLIB_VERSION, sizeof(z_stream))

#if !defined(ZUTIL_H) && !defined(NO_DUMMY_DECL)
    struct internal_state {int dummy;}; /* hack for buggy compilers */
#endif

ZEXTERN const char * ZEXPORT zError      OF((int));
ZEXTERN int ZEXPORT inflateSyncPoint OF((z_stream * z));
ZEXTERN const uLongf * ZEXPORT get_crc_table OF((void));

#ifdef __cplusplus
}
#endif

#endif /* ZLIB_H */
```

```

/* zutil.c -- target dependent utility functions for the compression library
 * Copyright (C) 1995-2005 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

```

```

/* @(#) $Id: zutil.c,v 1.1 2005/09/23 22:39:00 beng Exp $ */

```

```

#include "zutil.h"

```

```

#ifdef NO_DUMMY_DECL
struct internal_state {int dummy;}; /* for buggy compilers */
#endif

```

```

const char * const z_errmsg[10] = {
"need dictionary",      /* Z_NEED_DICT       2  */
"stream end",           /* Z_STREAM_END      1  */
"",                    /* Z_OK               0  */
"file error",           /* Z_ERRNO           (-1) */
"stream error",         /* Z_STREAM_ERROR    (-2) */
"data error",           /* Z_DATA_ERROR      (-3) */
"insufficient memory",  /* Z_MEM_ERROR       (-4) */
"buffer error",         /* Z_BUF_ERROR       (-5) */
"incompatible version", /* Z_VERSION_ERROR   (-6) */
""};

```

```

const char * ZEXPORT zlibVersion()
{
    return ZLIB_VERSION;
}

```

```

uLong ZEXPORT zlibCompileFlags()
{
    uLong flags;

    flags = 0;
    switch (sizeof(uInt)) {
    case 2:      break;
    case 4:      flags += 1;      break;
    case 8:      flags += 2;      break;
    default:     flags += 3;
    }
    switch (sizeof(uLong)) {
    case 2:      break;
    case 4:      flags += 1 << 2;      break;
    case 8:      flags += 2 << 2;      break;
    default:     flags += 3 << 2;
    }
    switch (sizeof(voidpf)) {
    case 2:      break;
    case 4:      flags += 1 << 4;      break;
    case 8:      flags += 2 << 4;      break;
    default:     flags += 3 << 4;
    }
    switch (sizeof(z_off_t)) {
    case 2:      break;
    case 4:      flags += 1 << 6;      break;
    case 8:      flags += 2 << 6;      break;
    default:     flags += 3 << 6;
    }
}

```

```

#ifdef DEBUG
    flags += 1 << 8;
#endif
#ifdef defined(ASMV) || defined(ASMINF)
    flags += 1 << 9;
#endif
#ifdef ZLIB_WINAPI
    flags += 1 << 10;
#endif
#ifdef BUILDFIXED
    flags += 1 << 12;
#endif
#ifdef DYNAMIC_CRC_TABLE
    flags += 1 << 13;

```

```

#endif
#ifdef NO_GZCOMPRESS
    flags += 1L << 16;
#endif
#ifdef NO_GZIP
    flags += 1L << 17;
#endif
#ifdef PKZIP_BUG_WORKAROUND
    flags += 1L << 20;
#endif
#ifdef FASTEST
    flags += 1L << 21;
#endif
#ifdef STDC
#   ifdef NO_vsnprintf
        flags += 1L << 25;
#       ifdef HAS_vsprintf_void
            flags += 1L << 26;
#       endif
#   else
#       ifdef HAS_vsnprintf_void
            flags += 1L << 26;
#       endif
#   endif
#else
    flags += 1L << 24;
#   ifdef NO_snprintf
        flags += 1L << 25;
#       ifdef HAS_sprintf_void
            flags += 1L << 26;
#       endif
#   else
#       ifdef HAS_snprintf_void
            flags += 1L << 26;
#       endif
#   endif
#endif
    return flags;
}

#ifdef DEBUG

#   ifndef verbose
#       define verbose 0
#   endif
    int z_verbose = verbose;

void z_error (m)
    char *m;
{
    fprintf(stderr, "%s\n", m);
    exit(1);
}
#endif

/* exported to allow conversion of error code to string for compress() and
 * uncompress()
 */
const char * ZEXPORT zError(err)
    int err;
{
    return ERR_MSG(err);
}

#ifdef defined(_WIN32_WCE)
    /* The Microsoft C Run-Time Library for Windows CE doesn't have
     * errno. We define it as a global variable to simplify porting.
     * Its value is always 0 and should not be used.
     */
    int errno = 0;
#endif

#ifdef HAVE_MEMCPY

```

```

void zmemcpy(dest, source, len)
    Bytef* dest;
    const Bytef* source;
    uInt len;
{
    if (len == 0) return;
    do {
        *dest++ = *source++; /* ??? to be unrolled */
    } while (--len != 0);
}

int zmemcmp(s1, s2, len)
    const Bytef* s1;
    const Bytef* s2;
    uInt len;
{
    uInt j;

    for (j = 0; j < len; j++) {
        if (s1[j] != s2[j]) return 2*(s1[j] > s2[j])-1;
    }
    return 0;
}

void zmemzero(dest, len)
    Bytef* dest;
    uInt len;
{
    if (len == 0) return;
    do {
        *dest++ = 0; /* ??? to be unrolled */
    } while (--len != 0);
}
#endif

#ifdef SYS16BIT

#ifdef __TURBOC__
/* Turbo C in 16-bit mode */

# define MY_ZCALLOC

/* Turbo C malloc() does not allow dynamic allocation of 64K bytes
 * and farmalloc(64K) returns a pointer with an offset of 8, so we
 * must fix the pointer. Warning: the pointer must be put back to its
 * original form in order to free it, use zcfree().
 */

#define MAX_PTR 10
/* 10*64K = 640K */

local int next_ptr = 0;

typedef struct ptr_table_s {
    voidpf org_ptr;
    voidpf new_ptr;
} ptr_table;

local ptr_table table[MAX_PTR];
/* This table is used to remember the original form of pointers
 * to large buffers (64K). Such pointers are normalized with a zero offset.
 * Since MSDOS is not a preemptive multitasking OS, this table is not
 * protected from concurrent access. This hack doesn't work anyway on
 * a protected system like OS/2. Use Microsoft C instead.
 */

voidpf zcalloc (voidpf opaque, unsigned items, unsigned size)
{
    voidpf buf = opaque; /* just to make some compilers happy */
    ulg bsize = (ulg)items*size;

    /* If we allocate less than 65520 bytes, we assume that farmalloc
     * will return a usable pointer which doesn't have to be normalized.

```

```

    */
    if (bsize < 65520L) {
        buf = farmalloc(bsize);
        if (*(ush*)&buf != 0) return buf;
    } else {
        buf = farmalloc(bsize + 16L);
    }
    if (buf == NULL || next_ptr >= MAX_PTR) return NULL;
    table[next_ptr].org_ptr = buf;

    /* Normalize the pointer to seg:0 */
    *((ush*)&buf+1) += ((ush)((uch*)&buf-0) + 15) >> 4;
    *(ush*)&buf = 0;
    table[next_ptr++].new_ptr = buf;
    return buf;
}

void zcfree (voidpf opaque, voidpf ptr)
{
    int n;
    if (*(ush*)&ptr != 0) { /* object < 64K */
        farfree(ptr);
        return;
    }
    /* Find the original pointer */
    for (n = 0; n < next_ptr; n++) {
        if (ptr != table[n].new_ptr) continue;

        farfree(table[n].org_ptr);
        while (++n < next_ptr) {
            table[n-1] = table[n];
        }
        next_ptr--;
        return;
    }
    ptr = opaque; /* just to make some compilers happy */
    Assert(0, "zcfree: ptr not found");
}

#endif /* __TURBOC__ */

#ifdef M_I86
/* Microsoft C in 16-bit mode */

# define MY_ZCALLOC

#if (!defined(_MSC_VER) || (_MSC_VER <= 600))
# define _halloc halloc
# define _hfree hfree
#endif

voidpf zcalloc (voidpf opaque, unsigned items, unsigned size)
{
    if (opaque) opaque = 0; /* to make compiler happy */
    return _halloc((long)items, size);
}

void zcfree (voidpf opaque, voidpf ptr)
{
    if (opaque) opaque = 0; /* to make compiler happy */
    _hfree(ptr);
}

#endif /* M_I86 */

#endif /* SYS16BIT */

#ifndef MY_ZCALLOC /* Any system without a special alloc function */

#ifdef STDC
extern voidp malloc OF((uInt size));
extern voidp calloc OF((uInt items, uInt size));

```

```
extern void free OF((voidpf ptr));
#endif

voidpf zcalloc (opaque, items, size)
    voidpf opaque;
    unsigned items;
    unsigned size;
{
    if (opaque) items += size - size; /* make compiler happy */
    return sizeof(uInt) > 2 ? (voidpf)malloc(items * size) :
        (voidpf)calloc(items, size);
}

void zcfree (opaque, ptr)
    voidpf opaque;
    voidpf ptr;
{
    free(ptr);
    if (opaque) return; /* make compiler happy */
}

#endif /* MY_ZCALLOC */
```

```
/* zutil.h -- internal interface and configuration of the compression library
 * Copyright (C) 1995-2005 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
 * part of the implementation of the compression library and is
 * subject to change. Applications should only use zlib.h.
 */

/* @(#) $Id: zutil.h,v 1.1 2005/09/23 22:39:00 beng Exp $ */

#ifndef ZUTIL_H
#define ZUTIL_H

#define ZLIB_INTERNAL
#include "zlib.h"

#ifdef STDC
#  ifndef _WIN32_WCE
#    include <stddef.h>
#  endif
#  include <string.h>
#  include <stdlib.h>
#else
#ifdef NO_ERRNO_H
#  ifndef _WIN32_WCE
/* The Microsoft C Run-Time Library for Windows CE doesn't have
 * errno. We define it as a global variable to simplify porting.
 * Its value is always 0 and should not be used. We rename it to
 * avoid conflict with other libraries that use the same workaround.
 */
#    define errno z_errno
#  endif
#  extern int errno;
#else
#  ifndef _WIN32_WCE
#    include <errno.h>
#  endif
#endif
#endif

#ifndef local
#  define local static
#endif
/* compile with -Dlocal if your debugger can't find static symbols */

typedef unsigned char  uch;
typedef uch FAR uchf;
typedef unsigned short ush;
typedef ush FAR ushf;
typedef unsigned long  ulg;

extern const char * const z_errmsg[10]; /* indexed by 2-zlib_error */
/* (size given to avoid silly warnings with Visual C++) */

#define ERR_MSG(err) z_errmsg[Z_NEED_DICT-(err)]

#define ERR_RETURN(strm,err) \
  return (strm->msg = (char*)ERR_MSG(err), (err))
/* To be used only when the state is known to be valid */

/* common constants */

#ifndef DEF_WBITS
#define DEF_WBITS MAX_WBITS
#endif
/* default windowBits for decompression. MAX_WBITS is for compression only */

#if MAX_MEM_LEVEL >= 8
#define DEF_MEM_LEVEL 8
#else
#define DEF_MEM_LEVEL  MAX_MEM_LEVEL
#endif
/* default memLevel */
```



```
#define STORED_BLOCK 0
#define STATIC_TREES 1
#define DYN_TREES 2
/* The three kinds of block type */

#define MIN_MATCH 3
#define MAX_MATCH 258
/* The minimum and maximum match lengths */

#define PRESET_DICT 0x20 /* preset dictionary flag in zlib header */

/* target dependencies */

#if defined(MSDOS) || (defined(WINDOWS) && !defined(WIN32))
# define OS_CODE 0x00
# if defined(__TURBOC__) || defined(__BORLANDC__)
# if (__STDC__ == 1) && (defined(__LARGE__) || defined(__COMPACT__))
/* Allow compilation with ANSI keywords only enabled */
void _Cdecl farfree( void *block );
void *_Cdecl farmalloc( unsigned long nbytes );
# else
# include <alloc.h>
# endif
# else /* MSC or DJGPP */
# include <malloc.h>
# endif
#endif

#ifdef AMIGA
# define OS_CODE 0x01
#endif

#if defined(VAXC) || defined(VMS)
# define OS_CODE 0x02
# define F_OPEN(name, mode) \
    fopen((name), (mode), "mbc=60", "ctx=stm", "rfm=fix", "mrs=512")
#endif

#if defined(ATARI) || defined(atarist)
# define OS_CODE 0x05
#endif

#ifdef OS2
# define OS_CODE 0x06
# ifdef M_I86
# include <malloc.h>
# endif
#endif

#if defined(MACOS) || defined(TARGET_OS_MAC)
# define OS_CODE 0x07
# if defined(__MWERKS__) && __dest_os != __be_os && __dest_os != __win32_os
# include <unix.h> /* for fdopen */
# else
# ifndef fdopen
# define fdopen(fd,mode) NULL /* No fdopen() */
# endif
# endif
#endif

#ifdef TOPS20
# define OS_CODE 0x0a
#endif

#ifdef WIN32
# ifndef __CYGWIN__ /* Cygwin is Unix, not Win32 */
# define OS_CODE 0x0b
# endif
#endif

#ifdef __50SERIES /* Prime/PRIMOS */
# define OS_CODE 0x0f
#endif
```

```

#if defined(_BEOS_) || defined(RISCOS)
# define fdopen(fd,mode) NULL /* No fdopen() */
#endif

#if (defined(_MSC_VER) && (_MSC_VER > 600))
# if defined(_WIN32_WCE)
#   define fdopen(fd,mode) NULL /* No fdopen() */
#   ifndef _PTRDIFF_T_DEFINED
#       typedef int ptrdiff_t;
#       define _PTRDIFF_T_DEFINED
#   endif
# else
#   define fdopen(fd,type)  _fdopen(fd,type)
# endif
#endif

    /* common defaults */

#ifndef OS_CODE
# define OS_CODE 0x03 /* assume Unix */
#endif

#ifndef F_OPEN
# define F_OPEN(name, mode) fopen((name), (mode))
#endif

    /* functions */

#if defined(STDC99) || (defined(__TURBOC__) && __TURBOC__ >= 0x550)
# ifndef HAVE_VSNPRINTF
#   define HAVE_VSNPRINTF
# endif
#endif
#if defined(__CYGWIN__)
# ifndef HAVE_VSNPRINTF
#   define HAVE_VSNPRINTF
# endif
#endif
#ifndef HAVE_VSNPRINTF
# ifndef MSDOS
#   /* vsnprintf may exist on some MS-DOS compilers (DJGPP?),
#    but for now we just assume it doesn't. */
#   define NO_vsnprintf
# endif
# ifndef __TURBOC__
#   define NO_vsnprintf
# endif
# ifndef WIN32
#   /* In Win32, vsnprintf is available as the "non-ANSI" _vsnprintf. */
#   if !defined(vsnprintf) && !defined(NO_vsnprintf)
#       define vsnprintf _vsnprintf
#   endif
# endif
# ifndef __SASC
#   define NO_vsnprintf
# endif
#endif
#ifndef VMS
# define NO_vsnprintf
#endif

#if defined(pyr)
# define NO_MEMCPY
#endif
#if defined(SMALL_MEDIUM) && !defined(_MSC_VER) && !defined(__SC__)
/* Use our own functions for small and medium model with MSC <= 5.0.
 * You may have to use the same strategy for Borland C (untested).
 * The __SC__ check is for Symantec.
 */
# define NO_MEMCPY
#endif
#if defined(STDC) && !defined(HAVE_MEMCPY) && !defined(NO_MEMCPY)
# define HAVE_MEMCPY

```

```

#endif
#ifdef HAVE_MEMCPY
#  ifdef SMALL_MEDIUM /* MSDOS small or medium model */
#    define zmemcpy _fmemcpy
#    define zmemcmp _fmemcmp
#    define zmemzero(dest, len) _fmemset(dest, 0, len)
#  else
#    define zmemcpy memcpy
#    define zmemcmp memcmp
#    define zmemzero(dest, len) memset(dest, 0, len)
#  endif
#else
extern void zmemcpy  OF((Bytef* dest, const Bytef* source, uInt len));
extern int  zmemcmp  OF((const Bytef* s1, const Bytef* s2, uInt len));
extern void zmemzero OF((Bytef* dest, uInt len));
#endif

/* Diagnostic functions */
#ifdef DEBUG
#  include <stdio.h>
#  extern int z_verbose;
#  extern void z_error  OF((char *m));
#  define Assert(cond,msg) {if(!(cond)) z_error(msg);}
#  define Trace(x) {if (z_verbose>=0) fprintf x ;}
#  define Tracev(x) {if (z_verbose>0) fprintf x ;}
#  define Tracevv(x) {if (z_verbose>1) fprintf x ;}
#  define Tracec(c,x) {if (z_verbose>0 && (c)) fprintf x ;}
#  define Tracecv(c,x) {if (z_verbose>1 && (c)) fprintf x ;}
#else
#  define Assert(cond,msg)
#  define Trace(x)
#  define Tracev(x)
#  define Tracevv(x)
#  define Tracec(c,x)
#  define Tracecv(c,x)
#endif

voidpf zcalloc OF((voidpf opaque, unsigned items, unsigned size));
void     zcfree  OF((voidpf opaque, voidpf ptr));

#define ZALLOC(strm, items, size) \
    (*((strm)->zalloc))((strm)->opaque, (items), (size))
#define ZFREE(strm, addr)    (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))
#define TRY_FREE(s, p) {if (p) ZFREE(s, p);}

#endif /* ZUTIL_H */

```

```
# Amiga powerUP (TM) Makefile
# makefile for libpng and SAS C V6.58/7.00 PPC compiler
# Copyright (C) 1998 by Andreas R. Kleinert

LIBNAME = libzip.a

CC      = scppc
CFLAGS  = NOSTKCHK NOSINT OPTIMIZE OPTGO OPTPEEP OPTINLOCAL OPTINL \
          OPTLOOP OPTRDEP=8 OPTDEP=8 OPTCOMP=8 NOVER
AR       = ppc-amigaos-ar cr
RANLIB   = ppc-amigaos-ranlib
LD       = ppc-amigaos-ld -r
LDFLAGS  = -o
LDLIBS   = LIB:scppc.a LIB:end.o
RM       = delete quiet

OBJS = Adler32.o compress.o crc32.o gzio.o uncompr.o deflate.o trees.o \
       zutil.o inflate.o infback.o inftrees.o inffast.o

TEST_OBJS = example.o minigzip.o

all: example minigzip

check: test
test: all
      example
      echo hello world | minigzip | minigzip -d

$(LIBNAME): $(OBJS)
           $(AR) $@ $(OBJS)
           -$(RANLIB) $@

example: example.o $(LIBNAME)
           $(LD) $(LDFLAGS) $@ LIB:c_ppc.o $@.o $(LIBNAME) $(LDLIBS)

minigzip: minigzip.o $(LIBNAME)
           $(LD) $(LDFLAGS) $@ LIB:c_ppc.o $@.o $(LIBNAME) $(LDLIBS)

mostlyclean: clean
clean:
           $(RM) *.o example minigzip $(LIBNAME) foo.gz

zip:
           zip -ul9 zlib README ChangeLog Makefile Make?????.??? Makefile.?? \
           descrip.mms *.[ch]

tgz:
           cd ../; tar cfz zlib/zlib.tgz zlib/README zlib/ChangeLog zlib/Makefile \
           zlib/Make?????.??? zlib/Makefile.?? zlib/descrip.mms zlib/*.[ch]

# DO NOT DELETE THIS LINE -- make depend depends on it.

adler32.o: zlib.h zconf.h
compress.o: zlib.h zconf.h
crc32.o: crc32.h zlib.h zconf.h
deflate.o: deflate.h zutil.h zlib.h zconf.h
example.o: zlib.h zconf.h
gzio.o: zutil.h zlib.h zconf.h
inffast.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
inflate.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
infback.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
inftrees.o: zutil.h zlib.h zconf.h inftrees.h
minigzip.o: zlib.h zconf.h
trees.o: deflate.h zutil.h zlib.h zconf.h trees.h
uncompr.o: zlib.h zconf.h
zutil.o: zutil.h zlib.h zconf.h
```

```
# SMakefile for zlib
# Modified from the standard UNIX Makefile Copyright Jean-loup Gailly
# Osmo Ahvenlampi <Osmo.Ahvenlampi@hut.fi>
# Amiga, SAS/C 6.56 & Smake

CC=sc
CFLAGS=OPT
#CFLAGS=OPT CPU=68030
#CFLAGS=DEBUG=LINE
LDFLAGS=LIB z.lib

SCOPTIONS=OPTSCHED OPTINLINE OPTALIAS OPTTIME OPTINLOCAL STRMERGE \
NOICONS PARMS=BOTH NOSTACKCHECK UTILLIB NOVERSION ERRORREXX \
DEF=POSTINC

OBJS = adler32.o compress.o crc32.o gzio.o uncompr.o deflate.o trees.o \
zutil.o inflate.o infback.o inftrees.o inffast.o

TEST_OBJS = example.o minigzip.o

all: SCOPTIONS example minigzip

check: test
test: all
    example
    echo hello world | minigzip | minigzip -d

install: z.lib
    copy clone zlib.h zconf.h INCLUDE:
    copy clone z.lib LIB:

z.lib: $(OBJS)
    oml z.lib r $(OBJS)

example: example.o z.lib
    $(CC) $(CFLAGS) LINK TO $@ example.o $(LDFLAGS)

minigzip: minigzip.o z.lib
    $(CC) $(CFLAGS) LINK TO $@ minigzip.o $(LDFLAGS)

mostlyclean: clean
clean:
    -delete force quiet example minigzip *.o z.lib foo.gz *.lnk SCOPTIONS

SCOPTIONS: Makefile.sas
    copy to $@ <from <
$(SCOPTIONS)
<

# DO NOT DELETE THIS LINE -- make depend depends on it.

adler32.o: zlib.h zconf.h
compress.o: zlib.h zconf.h
crc32.o: crc32.h zlib.h zconf.h
deflate.o: deflate.h zutil.h zlib.h zconf.h
example.o: zlib.h zconf.h
gzio.o: zutil.h zlib.h zconf.h
inffast.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
inflate.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
infback.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
inftrees.o: zutil.h zlib.h zconf.h inftrees.h
minigzip.o: zlib.h zconf.h
trees.o: deflate.h zutil.h zlib.h zconf.h trees.h
uncompr.o: zlib.h zconf.h
zutil.o: zutil.h zlib.h zconf.h
```

```
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('ZLIB')

/*****
/*      Version 1.1.3 entry points.
*****/

*****
/*      *MODULE      ADLER32      ZLIB      01/02/01  00:15:09      */
*****

EXPORT SYMBOL("adler32")

*****
/*      *MODULE      COMPRESS     ZLIB      01/02/01  00:15:09      */
*****

EXPORT SYMBOL("compress")
EXPORT SYMBOL("compress2")

*****
/*      *MODULE      CRC32       ZLIB      01/02/01  00:15:09      */
*****

EXPORT SYMBOL("crc32")
EXPORT SYMBOL("get_crc_table")

*****
/*      *MODULE      DEFLATE      ZLIB      01/02/01  00:15:09      */
*****

EXPORT SYMBOL("deflate")
EXPORT SYMBOL("deflateEnd")
EXPORT SYMBOL("deflateSetDictionary")
EXPORT SYMBOL("deflateCopy")
EXPORT SYMBOL("deflateReset")
EXPORT SYMBOL("deflateParams")
EXPORT SYMBOL("deflatePrime")
EXPORT SYMBOL("deflateInit_")
EXPORT SYMBOL("deflateInit2_")

*****
/*      *MODULE      GZIO        ZLIB      01/02/01  00:15:09      */
*****

EXPORT SYMBOL("gzopen")
EXPORT SYMBOL("gzdopen")
EXPORT SYMBOL("gzsetparams")
EXPORT SYMBOL("gzread")
EXPORT SYMBOL("gzwrite")
EXPORT SYMBOL("gzprintf")
EXPORT SYMBOL("gzputs")
EXPORT SYMBOL("gzgets")
EXPORT SYMBOL("gzputc")
EXPORT SYMBOL("gzgetc")
EXPORT SYMBOL("gzflush")
EXPORT SYMBOL("gzseek")
EXPORT SYMBOL("gzrewind")
EXPORT SYMBOL("gztell")
EXPORT SYMBOL("gzeof")
EXPORT SYMBOL("gzclose")
EXPORT SYMBOL("gzerror")

*****
/*      *MODULE      INFLATE     ZLIB      01/02/01  00:15:09      */
*****

EXPORT SYMBOL("inflate")
EXPORT SYMBOL("inflateEnd")
EXPORT SYMBOL("inflateSetDictionary")
EXPORT SYMBOL("inflateSync")
EXPORT SYMBOL("inflateReset")
EXPORT SYMBOL("inflateInit_")
EXPORT SYMBOL("inflateInit2_")
EXPORT SYMBOL("inflateSyncPoint")
```

```

/*****
/*  *MODULE      UNCOMPR      ZLIB      01/02/01  00:15:09      */
/*****

EXPORT SYMBOL("uncompress")

/*****
/*  *MODULE      ZUTIL      ZLIB      01/02/01  00:15:09      */
/*****

EXPORT SYMBOL("zlibVersion")
EXPORT SYMBOL("zError")

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*
/*  Version 1.2.1 additional entry points.
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*

/*****
/*  *MODULE      COMPRESS      ZLIB      01/02/01  00:15:09      */
/*****

EXPORT SYMBOL("compressBound")

/*****
/*  *MODULE      DEFLATE      ZLIB      01/02/01  00:15:09      */
/*****

EXPORT SYMBOL("deflateBound")

/*****
/*  *MODULE      GZIO      ZLIB      01/02/01  00:15:09      */
/*****

EXPORT SYMBOL("gzungetc")
EXPORT SYMBOL("gzclearerr")

/*****
/*  *MODULE      INFLBACK      ZLIB      01/02/01  00:15:09      */
/*****

EXPORT SYMBOL("inflateBack")
EXPORT SYMBOL("inflateBackEnd")
EXPORT SYMBOL("inflateBackInit_")

/*****
/*  *MODULE      INFLATE      ZLIB      01/02/01  00:15:09      */
/*****

EXPORT SYMBOL("inflateCopy")

/*****
/*  *MODULE      ZUTIL      ZLIB      01/02/01  00:15:09      */
/*****

EXPORT SYMBOL("zlibCompileFlags")

ENDPGMEXP

```

```

/*****
/*
/*  ZLIB
/*
/*  Compile sources into modules and link them into a service program.
/*
/*
*****/

PGM

/*  Configuration adjustable parameters.

DCL      VAR(&SRCLIB) TYPE(*CHAR) LEN(10) +
          VALUE('ZLIB') /* Source library. */
DCL      VAR(&SRCFILE) TYPE(*CHAR) LEN(10) +
          VALUE('SOURCES') /* Source member file. */
DCL      VAR(&CTLFIL) TYPE(*CHAR) LEN(10) +
          VALUE('TOOLS') /* Control member file. */

/

DCL      VAR(&MODLIB) TYPE(*CHAR) LEN(10) +
          VALUE('ZLIB') /* Module library. */

DCL      VAR(&SRVLIB) TYPE(*CHAR) LEN(10) +
          VALUE('LGPL') /* Service program library. */

DCL      VAR(&CFLAGS) TYPE(*CHAR) +
          VALUE('OPTIMIZE(40)') /* Compile options. */

/*  Working storage.

DCL      VAR(&CMDLEN) TYPE(*DEC) LEN(15 5) VALUE(300) /* Command length. */
DCL      VAR(&CMD) TYPE(*CHAR) LEN(512)

/*  Compile sources into modules.

CHGVAR   VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/ADLER32) SRCFILE(' *TCAT
                        &SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL     PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR   VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/COMPRESS) SRCFILE(' *TCAT
                        &SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL     PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR   VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/CRC32) SRCFILE(' *TCAT
                        &SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL     PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR   VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/DEFLATE) SRCFILE(' *TCAT
                        &SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL     PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR   VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/GZIO) SRCFILE(' *TCAT
                        &SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL     PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR   VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/INFLATE) SRCFILE(' *TCAT
                        &SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)

```



```

CALL      PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR    VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/INFFAST)          SRCFILE(' *TCAT      +
&SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT      +
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL      PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR    VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/INFLATE)          SRCFILE(' *TCAT      +
&SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT      +
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL      PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR    VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/INFTREES)          SRCFILE(' *TCAT      +
&SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT      +
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL      PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR    VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/TREES)          SRCFILE(' *TCAT      +
&SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT      +
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL      PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR    VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/UNCOMPR)          SRCFILE(' *TCAT      +
&SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT      +
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL      PGM(QCMDEXC) PARM(&CMD &CMDLEN)

CHGVAR    VAR(&CMD) VALUE('CRTCMOD MODULE(' *TCAT &MODLIB *TCAT +
                        '/ZUTIL)          SRCFILE(' *TCAT      +
&SRCLIB *TCAT '/' *TCAT &SRCFILE *TCAT      +
                        ') SYSIFCOPT(*IFSIO)' *BCAT &CFLAGS)
CALL      PGM(QCMDEXC) PARM(&CMD &CMDLEN)

```

```

/*      Link modules into a service program.      */

```

```

CRTSRVPGM  SRVPGM(&SRVLIB/ZLIB) +
            MODULE(&MODLIB/ADLER32    &MODLIB/COMPRESS      +
                  &MODLIB/CRC32      &MODLIB/DEFLATE      +
                  &MODLIB/GZIO       &MODLIB/INFBACK      +
                  &MODLIB/INFFAST    &MODLIB/INFLATE      +
                  &MODLIB/INFTREES   &MODLIB/TREES        +
                  &MODLIB/UNCOMPR    &MODLIB/ZUTIL)      +
            SRCFILE(&SRCLIB/&CTLFIL) SRCMBR(BNDSRC) +
            TEXT('ZLIB 1.2.3') TGTRLS(V4R4M0)

```

```

ENDPGM

```

ZLIB version 1.2.3 for AS400 installation instructions

I) From an AS400 *SAVF file:

1) Unpacking archive to an AS400 save file

On the AS400:

_ Create the ZLIB AS400 library:

```
CRTLIB LIB(ZLIB) TYPE(PROD) TEXT('ZLIB compression API library')
```

_ Create a work save file, for example:

```
CRTSAVF FILE(ZLIB/ZLIBSAVF)
```

On a PC connected to the target AS400:

_ Unpack the save file image to a PC file "ZLIBSAVF"

_ Upload this file into the save file on the AS400, for example
using ftp in BINARY mode.

2) Populating the ZLIB AS400 source library

On the AS400:

_ Extract the saved objects into the ZLIB AS400 library using:

```
RSTOBJ OBJ(*ALL) SAVLIB(ZLIB) DEV(*SAVF) SAVF(ZLIB/ZLIBSAVF) RSTLIB(ZLIB)
```

3) Customize installation:

_ Edit CL member ZLIB/TOOLS(COMPILE) and change parameters if needed,
according to the comments.

_ Compile this member with:

```
CRTCLPGM PGM(ZLIB/COMPILE) SRCFILE(ZLIB/TOOLS) SRCMBR(COMPILE)
```

4) Compile and generate the service program:

_ This can now be done by executing:

```
CALL PGM(ZLIB/COMPILE)
```

II) From the original source distribution:

1) On the AS400, create the source library:

```
CRTLIB LIB(ZLIB) TYPE(PROD) TEXT('ZLIB compression API library')
```

2) Create the source files:

```
CRTSRCPF FILE(ZLIB/SOURCES) RCDLEN(112) TEXT('ZLIB library modules')
```

```
CRTSRCPF FILE(ZLIB/H) RCDLEN(112) TEXT('ZLIB library includes')
```

```
CRTSRCPF FILE(ZLIB/TOOLS) RCDLEN(112) TEXT('ZLIB library control utilities')
```

3) From the machine hosting the distribution files, upload them (with FTP in text mode, for example) according to the following table:

Original file	AS400 file	AS400 member	AS400 type	AS400 description
	SOURCES			Original ZLIB C subprogram sources
adler32.c		ADLER32	C	ZLIB - Compute the Adler-32 checksum of a data stream
compress.c		COMPRESS	C	ZLIB - Compress a memory buffer
crc32.c		CRC32	C	ZLIB - Compute the CRC-32 of a data stream
deflate.c		DEFLATE	C	ZLIB - Compress data using the deflation algorithm
gzio.c		GZIO	C	ZLIB - IO on .gz files
inffback.c		INFFBACK	C	ZLIB - Inflate using a callback interface

inffast.c	INFFAST	C	ZLIB - Fast proc. literals & length/distance pairs
inflate.c	INFLATE	C	ZLIB - Interface to inflate modules
inftrees.c	INFTREES	C	ZLIB - Generate Huffman trees for efficient decode
trees.c	TREES	C	ZLIB - Output deflated data using Huffman coding
uncompr.c	UNCOMPR	C	ZLIB - Decompress a memory buffer
zutil.c	ZUTIL	C	ZLIB - Target dependent utility functions
			Original ZLIB C and ILE/RPG include files
crc32.h	CRC32	C	ZLIB - CRC32 tables
deflate.h	DEFLATE	C	ZLIB - Internal compression state
inffast.h	INFFAST	C	ZLIB - Header to use inffast.c
inffixed.h	INFFIXED	C	ZLIB - Table for decoding fixed codes
inflate.h	INFLATE	C	ZLIB - Internal inflate state definitions
inftrees.h	INFTREES	C	ZLIB - Header to use inftrees.c
trees.h	TREES	C	ZLIB - Created automatically with -DGEN_TREES_H
zconf.h	ZCONF	C	ZLIB - Compression library configuration
zlib.h	ZLIB	C	ZLIB - Compression library C user interface
as400/zlib.inc	ZLIB.INC	RPGLE	ZLIB - Compression library ILE RPG user interface
zutil.h	ZUTIL	C	ZLIB - Internal interface and configuration
			Building source software & AS/400 README
as400/bndsrc	BNDSRC		Entry point exportation list
as400/compile.clp	COMPILE	CLP	Compile sources & generate service program
as400/readme.txt	README	TXT	Installation instructions

4) Continue as in I)3).

Notes: For AS400 ILE RPG programmers, a /copy member defining the ZLIB API prototypes for ILE RPG can be found in ZLIB/H(ZLIB.INC). Please read comments in this member for more information.

Remember that most foreign textual data are ASCII coded: this implementation does not handle conversion from/to ASCII, so text data code conversions must be done explicitly.

Always open zipped files in binary mode.

```

*   ZLIB.INC - Interface to the general purpose compression library
*
*   ILE RPG400 version by Patrick Monnerat, DATASPHERE.
*   Version 1.2.3
*
*   WARNING:
*       Procedures inflateInit(), inflateInit2(), deflateInit(),
*       deflateInit2() and inflateBackInit() need to be called with
*       two additional arguments:
*       the package version string and the stream control structure.
*       size. This is needed because RPG lacks some macro feature.
*       Call these procedures as:
*           inflateInit(...: ZLIB_VERSION: %size(z_stream))
*
/if not defined(ZLIB_H_)
/define ZLIB_H_
*
*****
*                               Constants
*****
*
*   Versioning information.
*
D ZLIB_VERSION      C              '1.2.3'
D ZLIB_VERNUM       C              X'1230'
*
*   Other equates.
*
D Z_NO_FLUSH        C              0
D Z_SYNC_FLUSH      C              2
D Z_FULL_FLUSH      C              3
D Z_FINISH          C              4
D Z_BLOCK           C              5
*
D Z_OK              C              0
D Z_STREAM_END      C              1
D Z_NEED_DICT       C              2
D Z_ERRNO           C             -1
D Z_STREAM_ERROR    C             -2
D Z_DATA_ERROR      C             -3
D Z_MEM_ERROR       C             -4
D Z_BUF_ERROR       C             -5
DZ_VERSION_ERROR    C             -6
*
D Z_NO_COMPRESSION... C              0
D Z_BEST_SPEED      C              1
D Z_BEST_COMPRESSION... C              9
D Z_DEFAULT_COMPRESSION... D             -1
*
D Z_FILTERED        C              1
D Z_HUFFMAN_ONLY    C              2
D Z_RLE             C              3
D Z_DEFAULT_STRATEGY... D              0
*
D Z_BINARY          C              0
D Z_ASCII           C              1
D Z_UNKNOWN         C              2
*
D Z_DEFLATED        C              8
*
D Z_NULL           C              0
*
*****
*                               Types
*****
*
D z_stream          S              *              Stream st
ruct ptr
D gzFile            S              *              File poin

```

```

ter
D z_off_t          S          10i 0          Stream of
fsets
*
*****
*
Structures
*****
*
*   The GZIP encode/decode stream support structure.
*
D z_stream          DS          align based(z_streamp)
D zs_next_in        *          Next input
t byte
D zs_avail_in       10U 0       Byte cnt
at next_in
D zs_total_in       10U 0       Total byte
es read
D zs_next_out        *          Output buffer
ffer ptr
D zs_avail_out      10U 0       Room left
@ next_out
D zs_total_out      10U 0       Total byte
es written
D zs_msg            *          Last error
sg or null
D zs_state          *          Internal
state
D zs_zalloc         *          procptr
e allocator
D zs_free           *          procptr
e dealloc.
D zs_opaque         *          Private data
lloc. data
D zs_data_type      10i 0       ASC/BIN byte
est guess
D zs_adler          10u 0       Uncompressed
adler32 val
D                  10U 0       Reserved
D                  10U 0       Ptr. alignment
nment
*
*****
*
Utility function prototypes
*****
*
D compress          PR          10I 0 extproc('compress')
D dest              32767       options(*varsize)       Destination
on buffer
D destLen           10U 0       Destination
on length
D source            32767       const options(*varsize)   Source buffer
ffer
D sourceLen         10u 0 value       Source length
ngth
*
D compress2         PR          10I 0 extproc('compress2')
D dest              32767       options(*varsize)       Destination
on buffer
D destLen           10U 0       Destination
on length
D source            32767       const options(*varsize)   Source buffer
ffer
D sourceLen         10U 0 value       Source length
ngth
D level             10I 0 value       Compression level
on level
*
D compressBound     PR          10U 0 extproc('compressBound')
D sourceLen         10U 0 value
*
D uncompress        PR          10I 0 extproc('uncompress')
D dest              32767       options(*varsize)       Destination
on buffer
D destLen           10U 0       Destination

```

on length					
ffer	D source		32767	const options(*varsize)	Source bu
ngth	D sourceLen		10U 0	value	Source le
	*				
	D gzopen	PR		extproc('gzopen')	
	D			like(gzFile)	
name	D path		*	value options(*string)	File path
	D mode		*	value options(*string)	Open mode
	*				
	D gzdopen	PR		extproc('gzdopen')	
	D			like(gzFile)	
riptor	D fd		10i 0	value	File desc
	D mode		*	value options(*string)	Open mode
	*				
	D gzsetparams	PR	10I 0	extproc('gzsetparams')	
ter	D file			value like(gzFile)	File poin
	D level		10I 0	value	
	D strategy		10i 0	value	
	*				
	D gzread	PR	10I 0	extproc('gzread')	
ter	D file			value like(gzFile)	File poin
	D buf		32767	options(*varsize)	Buffer
ngth	D len		10u 0	value	Buffer le
	*				
	D gzwrite	PR	10I 0	extproc('gzwrite')	
ter	D file			value like(gzFile)	File poin
	D buf		32767	const options(*varsize)	Buffer
ngth	D len		10u 0	value	Buffer le
	*				
	D gzputs	PR	10I 0	extproc('gzputs')	
ter	D file			value like(gzFile)	File poin
	D s		*	value options(*string)	String to
output	*				
	D gzgets	PR	*	extproc('gzgets')	
ter	D file			value like(gzFile)	File poin
	D buf		32767	options(*varsize)	Read buff
er	D len		10i 0	value	Buffer le
ngth	*				
	D gzflush	PR	10i 0	extproc('gzflush')	
ter	D file			value like(gzFile)	File poin
	D flush		10I 0	value	Type of f
lush	*				
	D gzseek	PR		extproc('gzseek')	
	D			like(z_off_t)	
ter	D file			value like(gzFile)	File poin
	D offset			value like(z_off_t)	Offset
	D whence		10i 0	value	Origin
	*				
	D gzrewind	PR	10i 0	extproc('gzrewind')	
ter	D file			value like(gzFile)	File poin
	*				
	D gztell	PR		extproc('gztell')	
	D			like(z_off_t)	
ter	D file			value like(gzFile)	File poin
	*				

Sep 24, 05 2:39		lib/zlib-1.2.3/as400/zlib.inc			Page 4/7
ter	D gzeof	PR	10i 0	extproc('gzeof')	
	D file			value like(gzFile)	File poin
ter	*				
	D gzclose	PR	10i 0	extproc('gzclose')	
ing	D file			value like(gzFile)	File poin
	*				
ter	D gzerror	PR	*	extproc('gzerror')	Error str
	D file			value like(gzFile)	File poin
e	D errnum		10I 0		Error cod
	*				
ter	D gzclearerr	PR		extproc('gzclearerr')	
	D file			value like(gzFile)	File poin

* Basic function prototypes *					

tring	D zlibVersion	PR	*	extproc('zlibVersion')	Version s
	*				
pression	D deflateInit	PR	10I 0	extproc('deflateInit_')	Init. com
	D strm			like(z_stream)	Compressi
on stream	D level		10I 0	value	Compressi
	D version		*	value options(*string)	Version s
tring	D stream_size		10i 0	value	Stream st
	*				
data	D deflate	PR	10I 0	extproc('deflate')	Compress
	D strm			like(z_stream)	Compressi
on stream	D flush		10I 0	value	Flush typ
	*				
ompression	D deflateEnd	PR	10I 0	extproc('deflateEnd')	Termin. c
	D strm			like(z_stream)	Compressi
on stream	*				
	D inflateInit	PR	10I 0	extproc('inflateInit_')	Init. exp
ansion	D strm			like(z_stream)	Expansion
	D version		*	value options(*string)	Version s
tring	D stream_size		10i 0	value	Stream st
	*				
ta	D inflate	PR	10I 0	extproc('inflate')	Expand da
	D strm			like(z_stream)	Expansion
stream	D flush		10I 0	value	Flush typ
	*				
e required	D inflateEnd	PR	10I 0	extproc('inflateEnd')	Termin. e
	D strm			like(z_stream)	Expansion
xpansion	*				
	*				

* Advanced function prototypes *					

*					

Sep 24, 05 2:39	lib/zlib-1.2.3/as400/zlib.inc			Page 5/7
D deflateInit2	PR	10I 0	extproc('deflateInit2_')	Init. com
pression				
D strm			like(z_stream)	Compressi
on stream				
D level		10I 0	value	Compressi
on level				
D method		10I 0	value	Compressi
on method				
D windowBits		10I 0	value	log2(wind
ow size)				
D memLevel		10I 0	value	Mem/cmpre
ss tradeoff				
D strategy		10I 0	value	Compressi
on strategy				
D version		*	value options(*string)	Version s
tring				
D stream_size		10i 0	value	Stream st
ruct. size				
*				
D deflateSetDictionary...				
D	PR	10I 0	extproc('deflateSetDictionary')	Init. dic
tionary				
D strm			like(z_stream)	Compressi
on stream				
D dictionary		32767	const options(*varsize)	Dictionar
y bytes				
D dictLength		10U 0	value	Dictionar
y length				
*				
D deflateCopy	PR	10I 0	extproc('deflateCopy')	Compress
strm 2 strm				
D dest			like(z_stream)	Destinati
on stream				
D source			like(z_stream)	Source st
ream				
*				
D deflateReset	PR	10I 0	extproc('deflateReset')	End and i
nit. stream				
D strm			like(z_stream)	Compressi
on stream				
*				
D deflateParams	PR	10I 0	extproc('deflateParams')	Change le
vel & strat				
D strm			like(z_stream)	Compressi
on stream				
D level		10I 0	value	Compressi
on level				
D strategy		10I 0	value	Compressi
on strategy				
*				
D deflateBound	PR	10U 0	extproc('deflateBound')	Change le
vel & strat				
D strm			like(z_stream)	Compressi
on stream				
D sourcelen		10U 0	value	Compressi
on level				
*				
D deflatePrime	PR	10I 0	extproc('deflatePrime')	Change le
vel & strat				
D strm			like(z_stream)	Compressi
on stream				
D bits		10I 0	value	Number of
bits to insert				
D value		10I 0	value	Bits to i
nsert				
*				
D inflateInit2	PR	10I 0	extproc('inflateInit2_')	Init. exp
ansion				
D strm			like(z_stream)	Expansion
stream				
D windowBits		10I 0	value	log2(wind
ow size)				
D version		*	value options(*string)	Version s
tring				

Sep 24, 05 2:39	lib/zlib-1.2.3/as400/zlib.inc			Page 6/7
D stream_size		10i 0 value		Stream st
ruct. size				
*				
D inflateSetDictionary...	PR	10I 0 extproc('inflateSetDictionary')		Init. dic
tionary				
D strm		like(z_stream)		Expansion
stream				
D dictionary		32767 const options(*varsize)		Dictionar
y bytes				
D dictLength		10U 0 value		Dictionar
y length				
*				
D inflateSync	PR	10I 0 extproc('inflateSync')		Sync. exp
ansion				
D strm		like(z_stream)		Expansion
stream				
*				
D inflateCopy	PR	10I 0 extproc('inflateCopy')		
D dest		like(z_stream)		Destinati
on stream				
D source		like(z_stream)		Source st
ream				
*				
D inflateReset	PR	10I 0 extproc('inflateReset')		End and i
nit. stream				
D strm		like(z_stream)		Expansion
stream				
*				
D inflateBackInit...				
D	PR	10I 0 extproc('inflateBackInit_')		
D strm		like(z_stream)		Expansion
stream				
D windowBits		10I 0 value		Log2(buff
er size)				
D window		32767 options(*varsize)		Buffer
D version		* value options(*string)		Version s
tring				
D stream_size		10i 0 value		Stream st
ruct. size				
*				
D inflateBack	PR	10I 0 extproc('inflateBack')		
D strm		like(z_stream)		Expansion
stream				
D in		* value procptr		Input fun
ction				
D in_desc		* value		Input des
criptor				
D out		* value procptr		Output fu
nction				
D out_desc		* value		Output de
scriptor				
*				
D inflateBackEnd	PR	10I 0 extproc('inflateBackEnd')		
D strm		like(z_stream)		Expansion
stream				
*				
D zlibCompileFlags...				
D	PR	10U 0 extproc('zlibCompileFlags')		
*				

* Checksum function prototypes				

*				
D Adler32	PR	10U 0 extproc('adler32')		New check
sum				
D Adler		10U 0 value		Old check
sum				
D buf		32767 const options(*varsize)		Bytes to
accumulate				
D len		10U 0 value		Buffer le
ngth				
*				
D crc32	PR	10U 0 extproc('crc32')		New check

```
sum
D  crc                      10U 0 value          Old check
sum
D  buf                      32767  const options(*varsize)  Bytes to
accumulate
D  len                      10U 0 value          Buffer le
ngth
*
*****
*
*               Miscellaneous function prototypes
*
*****
*
D  zError          PR          *   extproc('zError')          Error str
ing
D  err             10I 0 value          Error cod
e
*
D  inflateSyncPoint...
D                   PR          10I 0  extproc('inflateSyncPoint')
D  strm                                like(z_stream)          Expansion
stream
*
D  get_crc_table    PR          *   extproc('get_crc_table')    Ptr to ul
ongs
*
/endif
```

All files under this contrib directory are UNSUPPORTED. There were provided by users of zlib and were not tested by the authors of zlib. Use at your own risk. Please contact the authors of the contributions for help about these, not the zlib authors. Thanks.

ada/ by Dmitriy Anisimkov <anisimkov@yahoo.com>
Support for Ada
See <http://zlib-ada.sourceforge.net/>

asm586/
asm686/ by Brian Raiter <breadbox@muppetlabs.com>
asm code for Pentium and PPro/PII, using the AT&T (GNU as) syntax
See <http://www.muppetlabs.com/~breadbox/software/assembly.html>

blast/ by Mark Adler <madler@alumni.caltech.edu>
Decompressor for output of PKWare Data Compression Library (DCL)

delphi/ by Cosmin Truta <cosmint@cs.ubbcluj.ro>
Support for Delphi and C++ Builder

dotzlib/ by Henrik Ravn <henrik@ravn.com>
Support for Microsoft .Net and Visual C++ .Net

infback9/ by Mark Adler <madler@alumni.caltech.edu>
Unsupported diffs to infback to decode the deflate64 format

inflate86/ by Chris Anderson <christop@charm.net>
Tuned x86 gcc asm code to replace inflate_fast()

iostream/ by Kevin Ruland <kevin@rodin.wustl.edu>
A C++ I/O streams interface to the zlib gz* functions

iostream2/ by Tyge Løvset <Tyge.Lovset@cmr.no>
Another C++ I/O streams interface

iostream3/ by Ludwig Schwardt <schwardt@sun.ac.za>
 and Kevin Ruland <kevin@rodin.wustl.edu>
Yet another C++ I/O streams interface

masm686/ by Dan Higdon <hdan@kinesoft.com>
 and Chuck Walbourn <chuckw@kinesoft.com>
asm code for Pentium Pro/PII, using the MASM syntax

masmx64/ by Gilles Vollant <info@winimage.com>
x86 64-bit (AMD64 and Intel EM64t) code for x64 assembler to
replace longest_match() and inflate_fast()

masmx86/ by Gilles Vollant <info@winimage.com>
x86 asm code to replace longest_match() and inflate_fast(),
for Visual C++ and MASM

minizip/ by Gilles Vollant <info@winimage.com>
Mini zip and unzip based on zlib
See <http://www.winimage.com/zLibDll/unzip.html>

pascal/ by Bob Dellaca <bobdl@extra.co.nz> et al.
Support for Pascal

puff/ by Mark Adler <madler@alumni.caltech.edu>
Small, low memory usage inflate. Also serves to provide an
unambiguous description of the deflate format.

testzlib/ by Gilles Vollant <info@winimage.com>
Example of the use of zlib

untgz/ by Pedro A. Aranda Gutierrez <paag@tid.es>
A very simple tar.gz file extractor using zlib

vstudio/ by Gilles Vollant <info@winimage.com>
Building a minizip-enhanced zlib with Microsoft Visual Studio

```
-----
--  ZLib for Ada thick binding.                                     --
--                                                                 --
--  Copyright (C) 2002-2004 Dmitriy Anisimkov                     --
--                                                                 --
--  Open source license information is in the zlib.ads file.      --
-----
--
--  $Id: buffer_demo.adb,v 1.1 2005/09/23 22:39:01 beng Exp $
--
--  This demo program provided by Dr Steve Sangwine <sjs@essex.ac.uk>
--
--  Demonstration of a problem with Zlib-Ada (already fixed) when a buffer
--  of exactly the correct size is used for decompressed data, and the last
--  few bytes passed in to Zlib are checksum bytes.
--
--  This program compresses a string of text, and then decompresses the
--  compressed text into a buffer of the same size as the original text.

with Ada.Streams; use Ada.Streams;
with Ada.Text_IO;

with ZLib; use ZLib;

procedure Buffer_Demo is
  EOL  : Character renames ASCII.LF;
  Text : constant String
    := "Four score and seven years ago our fathers brought forth," & EOL &
       "upon this continent, a new nation, conceived in liberty," & EOL &
       "and dedicated to the proposition that 'all men are created equal'.";

  Source : Stream_Element_Array (1 .. Text'Length);
  for Source'Address use Text'Address;

begin
  Ada.Text_IO.Put (Text);
  Ada.Text_IO.New_Line;
  Ada.Text_IO.Put_Line
    ("Uncompressed size: " & Positive'Image (Text'Length) & " bytes");

  declare
    Compressed_Data : Stream_Element_Array (1 .. Text'Length);
    L                : Stream_Element_Offset;
  begin
    Compress : declare
      Compressor : Filter_Type;
      I : Stream_Element_Offset;
    begin
      Deflate_Init (Compressor);

      --  Compress the whole of T at once.

      Translate (Compressor, Source, I, Compressed_Data, L, Finish);
      pragma Assert (I = Source'Last);

      Close (Compressor);

      Ada.Text_IO.Put_Line
        ("Compressed size: "
         & Stream_Element_Offset'Image (L) & " bytes");
    end Compress;

    --  Now we decompress the data, passing short blocks of data to Zlib
    --  (because this demonstrates the problem - the last block passed will
    --  contain checksum information and there will be no output, only a
    --  check inside Zlib that the checksum is correct).

    Decompress : declare
      Decompressor : Filter_Type;

      Uncompressed_Data : Stream_Element_Array (1 .. Text'Length);

      Block_Size : constant := 4;
      --  This makes sure that the last block contains
```

```
-- only Adler checksum data.

P : Stream_Element_Offset := Compressed_Data'First - 1;
O : Stream_Element_Offset;
begin
  Inflate_Init (Decompressor);

  loop
    Translate
      (Decompressor,
       Compressed_Data
        (P + 1 .. Stream_Element_Offset'Min (P + Block_Size, L)),
       P,
       Uncompressed_Data
        (Total_Out (Decompressor) + 1 .. Uncompressed_Data'Last),
       O,
       No_Flush);

    Ada.Text_IO.Put_Line
      ("Total in:" & Count'Image (Total_In (Decompressor)) &
       ",out:" & Count'Image (Total_Out (Decompressor)));

    exit when P = L;
  end loop;

  Ada.Text_IO.New_Line;
  Ada.Text_IO.Put_Line
    ("Decompressed text matches original text:"
     & Boolean'Image (Uncompressed_Data = Source));
end Decompress;
end;
```

end Buffer_Demo;

```

-----
-- ZLib for Ada thick binding.                                     --
--                                                                 --
-- Copyright (C) 2002-2003 Dmitriy Anisimkov                     --
--                                                                 --
-- Open source license information is in the zlib.ads file.      --
-----
-- Continuous test for ZLib multithreading. If the test would fail
-- we should provide thread safe allocation routines for the Z_Stream.
--
-- $Id: mtest.adb,v 1.1 2005/09/23 22:39:01 beng Exp $

with ZLib;
with Ada.Streams;
with Ada.Numerics.Discrete_Random;
with Ada.Text_IO;
with Ada.Exceptions;
with Ada.Task_Identification;

procedure MTest is
  use Ada.Streams;
  use ZLib;

  Stop : Boolean := False;

  pragma Atomic (Stop);

  subtype Visible_Symbols is Stream_Element range 16#20# .. 16#7E#;

  package Random_Elements is
    new Ada.Numerics.Discrete_Random (Visible_Symbols);

  task type Test_Task;

  task body Test_Task is
    Buffer : Stream_Element_Array (1 .. 100_000);
    Gen : Random_Elements.Generator;

    Buffer_First : Stream_Element_Offset;
    Compare_First : Stream_Element_Offset;

    Deflate : Filter_Type;
    Inflate : Filter_Type;

    procedure Further (Item : in Stream_Element_Array);

    procedure Read_Buffer
      (Item : out Ada.Streams.Stream_Element_Array;
       Last : out Ada.Streams.Stream_Element_Offset);

    -----
    -- Further --
    -----

    procedure Further (Item : in Stream_Element_Array) is

      procedure Compare (Item : in Stream_Element_Array);

      -----
      -- Compare --
      -----

      procedure Compare (Item : in Stream_Element_Array) is
        Next_First : Stream_Element_Offset := Compare_First + Item'Length;
      begin
        if Buffer (Compare_First .. Next_First - 1) /= Item then
          raise Program_Error;
        end if;

        Compare_First := Next_First;
      end Compare;

      procedure Compare_Write is new ZLib.Write (Write => Compare);
    begin

```

```

    Compare_Write (Inflate, Item, No_Flush);
end Further;

-----
-- Read_Buffer --
-----

procedure Read_Buffer
  (Item : out Ada.Streams.Stream_Element_Array;
   Last : out Ada.Streams.Stream_Element_Offset)
is
  Buff_Diff    : Stream_Element_Offset := Buffer'Last - Buffer_First;
  Next_First   : Stream_Element_Offset;
begin
  if Item'Length <= Buff_Diff then
    Last := Item'Last;

    Next_First := Buffer_First + Item'Length;

    Item := Buffer (Buffer_First .. Next_First - 1);

    Buffer_First := Next_First;
  else
    Last := Item'First + Buff_Diff;
    Item (Item'First .. Last) := Buffer (Buffer_First .. Buffer'Last);
    Buffer_First := Buffer'Last + 1;
  end if;
end Read_Buffer;

procedure Translate is new Generic_Translate
  (Data_In  => Read_Buffer,
   Data_Out => Further);

begin
  Random_Elements.Reset (Gen);

  Buffer := (others => 20);

  Main : loop
    for J in Buffer'Range loop
      Buffer (J) := Random_Elements.Random (Gen);

      Deflate_Init (Deflate);
      Inflate_Init (Inflate);

      Buffer_First := Buffer'First;
      Compare_First := Buffer'First;

      Translate (Deflate);

      if Compare_First /= Buffer'Last + 1 then
        raise Program_Error;
      end if;

      Ada.Text_IO.Put_Line
        (Ada.Task_Identification.Image
         (Ada.Task_Identification.Current_Task)
         & Stream_Element_Offset'Image (J)
         & ZLib.Count'Image (Total_Out (Deflate)));

      Close (Deflate);
      Close (Inflate);

      exit Main when Stop;
    end loop;
  end loop Main;
exception
  when E : others =>
    Ada.Text_IO.Put_Line (Ada.Exceptions.Exception_Information (E));
    Stop := True;
end Test_Task;

Test : array (1 .. 4) of Test_Task;

```

```
pragma Unreferenced (Test);  
  
Dummy : Character;  
  
begin  
  Ada.Text_IO.Get_Immediate (Dummy);  
  Stop := True;  
end MTest;
```



```
-----
-- ZLib for Ada thick binding.                                     --
--                                                                 --
-- Copyright (C) 2002-2003 Dmitriy Anisimkov                     --
--                                                                 --
-- Open source license information is in the zlib.ads file.      --
-----

-- $Id: read.adb,v 1.1 2005/09/23 22:39:01 beng Exp $

-- Test/demo program for the generic read interface.

with Ada.Numerics.Discrete_Random;
with Ada.Streams;
with Ada.Text_IO;

with ZLib;

procedure Read is

  use Ada.Streams;

  -----
  -- Test configuration parameters --
  -----

  File_Size    : Stream_Element_Offset := 100_000;

  Continuous   : constant Boolean      := False;
  -- If this constant is True, the test would be repeated again and again,
  -- with increment File_Size for every iteration.

  Header       : constant ZLib.Header_Type := ZLib.Default;
  -- Do not use Header other than Default in ZLib versions 1.1.4 and older.

  Init_Random  : constant := 8;
  -- We are using the same random sequence, in case of we catch bug,
  -- so we would be able to reproduce it.

  -- End --

  Pack_Size : Stream_Element_Offset;
  Offset     : Stream_Element_Offset;

  Filter      : ZLib.Filter_Type;

  subtype Visible_Symbols
    is Stream_Element range 16#20# .. 16#7E#;

  package Random_Elements is new
    Ada.Numerics.Discrete_Random (Visible_Symbols);

  Gen : Random_Elements.Generator;
  Period : constant Stream_Element_Offset := 200;
  -- Period constant variable for random generator not to be very random.
  -- Bigger period, harder random.

  Read_Buffer : Stream_Element_Array (1 .. 2048);
  Read_First  : Stream_Element_Offset;
  Read_Last   : Stream_Element_Offset;

  procedure Reset;

  procedure Read
    (Item : out Stream_Element_Array;
     Last : out Stream_Element_Offset);
  -- this procedure is for generic instantiation of
  -- ZLib.Read
  -- reading data from the File_In.

  procedure Read is new ZLib.Read
    (Read,
     Read_Buffer,
     Rest_First => Read_First,
```

```
Rest_Last => Read_Last);
```

```
-----  
-- Read --  
-----
```

```
procedure Read  
(Item : out Stream_Element_Array;  
Last : out Stream_Element_Offset) is  
begin  
  Last := Stream_Element_Offset'Min  
    (Item'Last,  
     Item'First + File_Size - Offset);  
  
  for J in Item'First .. Last loop  
    if J < Item'First + Period then  
      Item (J) := Random_Elements.Random (Gen);  
    else  
      Item (J) := Item (J - Period);  
    end if;  
  
    Offset := Offset + 1;  
  end loop;  
end Read;
```

```
-----  
-- Reset --  
-----
```

```
procedure Reset is  
begin  
  Random_Elements.Reset (Gen, Init_Random);  
  Pack_Size := 0;  
  Offset := 1;  
  Read_First := Read_Buffer'Last + 1;  
  Read_Last := Read_Buffer'Last;  
end Reset;
```

```
begin  
  Ada.Text_IO.Put_Line ("ZLib " & ZLib.Version);  
  
  loop  
    for Level in ZLib.Compression_Level'Range loop  
  
      Ada.Text_IO.Put ("Level=" & ZLib.Compression_Level'Image (Level));  
  
      -- Deflate using generic instantiation.  
  
      ZLib.Deflate_Init  
        (Filter,  
         Level,  
         Header => Header);  
  
      Reset;  
  
      Ada.Text_IO.Put  
        (Stream_Element_Offset'Image (File_Size) & " ->");  
  
      loop  
        declare  
          Buffer : Stream_Element_Array (1 .. 1024);  
          Last : Stream_Element_Offset;  
        begin  
          Read (Filter, Buffer, Last);  
  
          Pack_Size := Pack_Size + Last - Buffer'First + 1;  
  
          exit when Last < Buffer'Last;  
        end;  
      end loop;  
  
      Ada.Text_IO.Put_Line (Stream_Element_Offset'Image (Pack_Size));
```

```
        ZLib.Close (Filter);  
    end loop;  
  
    exit when not Continuous;  
  
    File_Size := File_Size + 1;  
end loop;  
end Read;
```

ZLib for Ada thick binding (ZLib.Ada)
Release 1.3

ZLib.Ada is a thick binding interface to the popular ZLib data compression library, available at <http://www.gzip.org/zlib/>. It provides Ada-style access to the ZLib C library.

Here are the main changes since ZLib.Ada 1.2:

- Attention: ZLib.Read generic routine have a initialization requirement for Read_Last parameter now. It is a bit incompatibile with previous version, but extends functionality, we could use new parameters Allow_Read_Some and Flush now.
- Added Is_Open routines to ZLib and ZLib.Streams packages.
- Add pragma Assert to check Stream_Element is 8 bit.
- Fix extraction to buffer with exact known decompressed size. Error reported by Steve Sangwine.
- Fix definition of ULong (changed to unsigned_long), fix regression on 64 bits computers. Patch provided by Pascal Obry.
- Add Status_Error exception definition.
- Add pragma Assertion that Ada.Streams.Stream_Element size is 8 bit.

How to build ZLib.Ada under GNAT

You should have the ZLib library already build on your computer, before building ZLib.Ada. Make the directory of ZLib.Ada sources current and issue the command:

```
gnatmake test -largz -L<directory where libz.a is> -lz
```

Or use the GNAT project file build for GNAT 3.15 or later:

```
gnatmake -Pzlib.gpr -L<directory where libz.a is>
```

How to build ZLib.Ada under Aonix ObjectAda for Win32 7.2.2

1. Make a project with all *.ads and *.adb files from the distribution.
2. Build the libz.a library from the ZLib C sources.
3. Rename libz.a to z.lib.
4. Add the library z.lib to the project.
5. Add the libc.lib library from the ObjectAda distribution to the project.
6. Build the executable using test.adb as a main procedure.

How to use ZLib.Ada

The source files test.adb and read.adb are small demo programs that show the main functionality of ZLib.Ada.

The routines from the package specifications are commented.

Homepage: <http://zlib-ada.sourceforge.net/>
Author: Dmitriy Anisimkov <anisimkov@yahoo.com>

Contributors: Pascal Obry <pascal@obry.org>, Steve Sangwine <sjs@essex.ac.uk>

```

-----
--  ZLib for Ada thick binding.                                     --
--                                                                 --
--  Copyright (C) 2002-2003 Dmitriy Anisimkov                     --
--                                                                 --
--  Open source license information is in the zlib.ads file.      --
-----

--  $Id: test.adb,v 1.1 2005/09/23 22:39:01 beng Exp $

--  The program has a few aims.
--  1. Test ZLib.Ada95 thick binding functionality.
--  2. Show the example of use main functionality of the ZLib.Ada95 binding.
--  3. Build this program automatically compile all ZLib.Ada95 packages under
--     GNAT Ada95 compiler.

with ZLib.Streams;
with Ada.Streams.Stream_IO;
with Ada.Numerics.Discrete_Random;

with Ada.Text_IO;

with Ada.Calendar;

procedure Test is

  use Ada.Streams;
  use Stream_IO;

  -----
  --  Test configuration parameters --
  -----

  File_Size    : Count    := 100_000;
  Continuous   : constant Boolean := False;

  Header       : constant ZLib.Header_Type := ZLib.Default;
                                     --  ZLib.None;
                                     --  ZLib.Auto;
                                     --  ZLib.GZip;

  --  Do not use Header other then Default in ZLib versions 1.1.4
  --  and older.

  Strategy     : constant ZLib.Strategy_Type := ZLib.Default_Strategy;
  Init_Random  : constant := 10;

  --  End --

  In_File_Name : constant String := "testzlib.in";
  --  Name of the input file

  Z_File_Name  : constant String := "testzlib.zlb";
  --  Name of the compressed file.

  Out_File_Name : constant String := "testzlib.out";
  --  Name of the decompressed file.

  File_In      : File_Type;
  File_Out     : File_Type;
  File_Back    : File_Type;
  File_Z       : ZLib.Streams.Stream_Type;

  Filter       : ZLib.Filter_Type;

  Time_Stamp   : Ada.Calendar.Time;

  procedure Generate_File;
  --  Generate file of spetsified size with some random data.
  --  The random data is repeatable, for the good compression.

  procedure Compare_Streams
    (Left, Right : in out Root_Stream_Type'Class);
  --  The procedure compearing data in 2 streams.
  --  It is for compare data before and after compression/decompression.

```

```

procedure Compare_Files (Left, Right : String);
--   Compare files. Based on the Compare_Streams.

procedure Copy_Streams
  (Source, Target : in out Root_Stream_Type'Class;
   Buffer_Size    : in      Stream_Element_Offset := 1024);
--   Copying data from one stream to another. It is for test stream
--   interface of the library.

procedure Data_In
  (Item : out Stream_Element_Array;
   Last : out Stream_Element_Offset);
--   this procedure is for generic instantiation of
--   ZLib.Generic_Translate.
--   reading data from the File_In.

procedure Data_Out (Item : in Stream_Element_Array);
--   this procedure is for generic instantiation of
--   ZLib.Generic_Translate.
--   writing data to the File_Out.

procedure Stamp;
--   Store the timestamp to the local variable.

procedure Print_Statistic (Msg : String; Data_Size : ZLib.Count);
--   Print the time statistic with the message.

procedure Translate is new ZLib.Generic_Translate
  (Data_In  => Data_In,
   Data_Out => Data_Out);
--   This procedure is moving data from File_In to File_Out
--   with compression or decompression, depend on initialization of
--   Filter parameter.

-----
-- Compare_Files --
-----

procedure Compare_Files (Left, Right : String) is
  Left_File, Right_File : File_Type;
begin
  Open (Left_File, In_File, Left);
  Open (Right_File, In_File, Right);
  Compare_Streams (Stream (Left_File).all, Stream (Right_File).all);
  Close (Left_File);
  Close (Right_File);
end Compare_Files;

-----
-- Compare_Streams --
-----

procedure Compare_Streams
  (Left, Right : in out Ada.Streams.Root_Stream_Type'Class)
is
  Left_Buffer, Right_Buffer : Stream_Element_Array (0 .. 16#FFF#);
  Left_Last, Right_Last : Stream_Element_Offset;
begin
  loop
    Read (Left, Left_Buffer, Left_Last);
    Read (Right, Right_Buffer, Right_Last);

    if Left_Last /= Right_Last then
      Ada.Text_IO.Put_Line ("Compare error:"
        & Stream_Element_Offset'Image (Left_Last)
        & " /= "
        & Stream_Element_Offset'Image (Right_Last));

      raise Constraint_Error;

    elsif Left_Buffer (0 .. Left_Last)
      /= Right_Buffer (0 .. Right_Last)
    then

```

```

Ada.Text_IO.Put_Line ("ERROR: IN and OUT files is not equal.");
raise Constraint_Error;

end if;

exit when Left_Last < Left_Buffer'Last;
end loop;
end Compare_Streams;

-----
-- Copy_Streams --
-----

procedure Copy_Streams
(Source, Target : in out Ada.Streams.Root_Stream_Type'Class;
 Buffer_Size    : in      Stream_Element_Offset := 1024)
is
  Buffer : Stream_Element_Array (1 .. Buffer_Size);
  Last   : Stream_Element_Offset;
begin
  loop
    Read (Source, Buffer, Last);
    Write (Target, Buffer (1 .. Last));

    exit when Last < Buffer'Last;
  end loop;
end Copy_Streams;

-----
-- Data_In --
-----

procedure Data_In
(Item : out Stream_Element_Array;
 Last : out Stream_Element_Offset) is
begin
  Read (File_In, Item, Last);
end Data_In;

-----
-- Data_Out --
-----

procedure Data_Out (Item : in Stream_Element_Array) is
begin
  Write (File_Out, Item);
end Data_Out;

-----
-- Generate_File --
-----

procedure Generate_File is
  subtype Visible_Symbols is Stream_Element range 16#20# .. 16#7E#;

  package Random_Elements is
    new Ada.Numerics.Discrete_Random (Visible_Symbols);

  Gen      : Random_Elements.Generator;
  Buffer    : Stream_Element_Array := (1 .. 77 => 16#20#) & 10;

  Buffer_Count : constant Count := File_Size / Buffer'Length;
  -- Number of same buffers in the packet.

  Density : constant Count := 30; -- from 0 to Buffer'Length - 2;

  procedure Fill_Buffer (J, D : in Count);
  -- Change the part of the buffer.

  -----
  -- Fill_Buffer --
  -----

  procedure Fill_Buffer (J, D : in Count) is

```

```

begin
  for K in 0 .. D loop
    Buffer
      (Stream_Element_Offset ((J + K) mod (Buffer'Length - 1) + 1))
      := Random_Elements.Random (Gen);

    end loop;
  end Fill_Buffer;

begin
  Random_Elements.Reset (Gen, Init_Random);

  Create (File_In, Out_File, In_File_Name);

  Fill_Buffer (1, Buffer'Length - 2);

  for J in 1 .. Buffer_Count loop
    Write (File_In, Buffer);

    Fill_Buffer (J, Density);
  end loop;

  -- fill remain size.

  Write
    (File_In,
     Buffer
      (1 .. Stream_Element_Offset
        (File_Size - Buffer'Length * Buffer_Count)));

  Flush (File_In);
  Close (File_In);
end Generate_File;

-----
-- Print_Statistic --
-----

procedure Print_Statistic (Msg : String; Data_Size : ZLib.Count) is
  use Ada.Calendar;
  use Ada.Text_IO;

  package Count_IO is new Integer_IO (ZLib.Count);

  Curr_Dur : Duration := Clock - Time_Stamp;
begin
  Put (Msg);

  Set_Col (20);
  Ada.Text_IO.Put ("size=");

  Count_IO.Put
    (Data_Size,
     Width => Stream_IO.Count'Image (File_Size)'Length);

  Put_Line (" duration=" & Duration'Image (Curr_Dur));
end Print_Statistic;

-----
-- Stamp --
-----

procedure Stamp is
begin
  Time_Stamp := Ada.Calendar.Clock;
end Stamp;

begin
  Ada.Text_IO.Put_Line ("ZLib " & ZLib.Version);

  loop
    Generate_File;

    for Level in ZLib.Compression_Level'Range loop

```



```
Ada.Text_IO.Put_Line ("Level="
  & ZLib.Compression_Level'Image (Level));

-- Test generic interface.
Open  (File_In, In_File, In_File_Name);
Create (File_Out, Out_File, Z_File_Name);

Stamp;

-- Deflate using generic instantiation.

ZLib.Deflate_Init
  (Filter  => Filter,
   Level   => Level,
   Strategy => Strategy,
   Header  => Header);

Translate (Filter);
Print_Statistic ("Generic compress", ZLib.Total_Out (Filter));
ZLib.Close (Filter);

Close (File_In);
Close (File_Out);

Open  (File_In, In_File, Z_File_Name);
Create (File_Out, Out_File, Out_File_Name);

Stamp;

-- Inflate using generic instantiation.

ZLib.Inflate_Init (Filter, Header => Header);

Translate (Filter);
Print_Statistic ("Generic decompress", ZLib.Total_Out (Filter));

ZLib.Close (Filter);

Close (File_In);
Close (File_Out);

Compare_Files (In_File_Name, Out_File_Name);

-- Test stream interface.

-- Compress to the back stream.

Open  (File_In, In_File, In_File_Name);
Create (File_Back, Out_File, Z_File_Name);

Stamp;

ZLib.Streams.Create
  (Stream      => File_Z,
   Mode        => ZLib.Streams.Out_Stream,
   Back        => ZLib.Streams.Stream_Access
                 (Stream (File_Back)),
   Back_Compressed => True,
   Level       => Level,
   Strategy    => Strategy,
   Header      => Header);

Copy_Streams
  (Source => Stream (File_In).all,
   Target => File_Z);

-- Flushing internal buffers to the back stream.

ZLib.Streams.Flush (File_Z, ZLib.Finish);

Print_Statistic ("Write compress",
  ZLib.Streams.Write_Total_Out (File_Z));
```

```
ZLib.Streams.Close (File_Z);

Close (File_In);
Close (File_Back);

-- Compare reading from original file and from
-- decompression stream.

Open (File_In, In_File, In_File_Name);
Open (File_Back, In_File, Z_File_Name);

ZLib.Streams.Create
  (Stream      => File_Z,
   Mode        => ZLib.Streams.In_Stream,
   Back        => ZLib.Streams.Stream_Access
                 (Stream (File_Back)),
   Back_Compressed => True,
   Header      => Header);

Stamp;
Compare_Streams (Stream (File_In).all, File_Z);

Print_Statistic ("Read decompress",
                 ZLib.Streams.Read_Total_Out (File_Z));

ZLib.Streams.Close (File_Z);
Close (File_In);
Close (File_Back);

-- Compress by reading from compression stream.

Open (File_Back, In_File, In_File_Name);
Create (File_Out, Out_File, Z_File_Name);

ZLib.Streams.Create
  (Stream      => File_Z,
   Mode        => ZLib.Streams.In_Stream,
   Back        => ZLib.Streams.Stream_Access
                 (Stream (File_Back)),
   Back_Compressed => False,
   Level        => Level,
   Strategy     => Strategy,
   Header      => Header);

Stamp;
Copy_Streams
  (Source => File_Z,
   Target => Stream (File_Out).all);

Print_Statistic ("Read compress",
                 ZLib.Streams.Read_Total_Out (File_Z));

ZLib.Streams.Close (File_Z);

Close (File_Out);
Close (File_Back);

-- Decompress to decompression stream.

Open (File_In, In_File, Z_File_Name);
Create (File_Back, Out_File, Out_File_Name);

ZLib.Streams.Create
  (Stream      => File_Z,
   Mode        => ZLib.Streams.Out_Stream,
   Back        => ZLib.Streams.Stream_Access
                 (Stream (File_Back)),
   Back_Compressed => False,
   Header      => Header);

Stamp;

Copy_Streams
  (Source => Stream (File_In).all,
```

```
        Target => File_Z);

    Print_Statistic ("Write decompress",
                    ZLib.Streams.Write_Total_Out (File_Z));

    ZLib.Streams.Close (File_Z);
    Close (File_In);
    Close (File_Back);

    Compare_Files (In_File_Name, Out_File_Name);
end loop;

    Ada.Text_IO.Put_Line (Count'Image (File_Size) & " Ok.");

    exit when not Continuous;

    File_Size := File_Size + 1;
end loop;
end Test;
```

```

-----
-- ZLib for Ada thick binding.
--
-- Copyright (C) 2002-2003 Dmitriy Anisimkov
--
-- Open source license information is in the zlib.ads file.
-----

-- $Id: zlib-streams.adb,v 1.1 2005/09/23 22:39:01 beng Exp $

with Ada.Unchecked_Deallocation;

package body ZLib.Streams is

    -----
    -- Close --
    -----

    procedure Close (Stream : in out Stream_Type) is
        procedure Free is new Ada.Unchecked_Deallocation
            (Stream_Element_Array, Buffer_Access);
    begin
        if Stream.Mode = Out_Stream or Stream.Mode = Duplex then
            -- We should flush the data written by the writer.

            Flush (Stream, Finish);

            Close (Stream.Writer);
        end if;

        if Stream.Mode = In_Stream or Stream.Mode = Duplex then
            Close (Stream.Reader);
            Free (Stream.Buffer);
        end if;
    end Close;

    -----
    -- Create --
    -----

    procedure Create
        (Stream           : out Stream_Type;
         Mode             : in Stream_Mode;
         Back             : in Stream_Access;
         Back_Compressed  : in Boolean;
         Level            : in Compression_Level := Default_Compression;
         Strategy         : in Strategy_Type := Default_Strategy;
         Header           : in Header_Type := Default;
         Read_Buffer_Size : in Ada.Streams.Stream_Element_Offset
                               := Default_Buffer_Size;
         Write_Buffer_Size : in Ada.Streams.Stream_Element_Offset
                               := Default_Buffer_Size)
    is
        subtype Buffer_Subtype is Stream_Element_Array (1 .. Read_Buffer_Size);

        procedure Init_Filter
            (Filter : in out Filter_Type;
             Compress : in Boolean);

        -----
        -- Init_Filter --
        -----

        procedure Init_Filter
            (Filter : in out Filter_Type;
             Compress : in Boolean) is
        begin
            if Compress then
                Deflate_Init
                    (Filter, Level, Strategy, Header => Header);
            else
                Inflate_Init (Filter, Header => Header);
            end if;
        end Init_Filter;
    end Create;

```

```

    end Init_Filter;

begin
    Stream.Back := Back;
    Stream.Mode := Mode;

    if Mode = Out_Stream or Mode = Duplex then
        Init_Filter (Stream.Writer, Back_Compressed);
        Stream.Buffer_Size := Write_Buffer_Size;
    else
        Stream.Buffer_Size := 0;
    end if;

    if Mode = In_Stream or Mode = Duplex then
        Init_Filter (Stream.Reader, not Back_Compressed);

        Stream.Buffer      := new Buffer_Subtype;
        Stream.Rest_First := Stream.Buffer'Last + 1;
        Stream.Rest_Last  := Stream.Buffer'Last;
    end if;
end Create;

-----
-- Flush --
-----

procedure Flush
  (Stream : in out Stream_Type;
   Mode   : in      Flush_Mode := Sync_Flush)
is
  Buffer : Stream_Element_Array (1 .. Stream.Buffer_Size);
  Last  : Stream_Element_Offset;
begin
  loop
    Flush (Stream.Writer, Buffer, Last, Mode);

    Ada.Streams.Write (Stream.Back.all, Buffer (1 .. Last));

    exit when Last < Buffer'Last;
  end loop;
end Flush;

-----
-- Is_Open --
-----

function Is_Open (Stream : Stream_Type) return Boolean is
begin
  return Is_Open (Stream.Reader) or else Is_Open (Stream.Writer);
end Is_Open;

-----
-- Read --
-----

procedure Read
  (Stream : in out Stream_Type;
   Item   : out Stream_Element_Array;
   Last   : out Stream_Element_Offset)
is
  procedure Read
    (Item : out Stream_Element_Array;
     Last : out Stream_Element_Offset);

  -----
  -- Read --
  -----

  procedure Read
    (Item : out Stream_Element_Array;
     Last : out Stream_Element_Offset) is
  begin
    Ada.Streams.Read (Stream.Back.all, Item, Last);

```

```
end Read;

procedure Read is new ZLib.Read
  (Read      => Read,
   Buffer     => Stream.Buffer.all,
   Rest_First => Stream.Rest_First,
   Rest_Last  => Stream.Rest_Last);

begin
  Read (Stream.Reader, Item, Last);
end Read;

-----
-- Read_Total_In --
-----

function Read_Total_In (Stream : in Stream_Type) return Count is
begin
  return Total_In (Stream.Reader);
end Read_Total_In;

-----
-- Read_Total_Out --
-----

function Read_Total_Out (Stream : in Stream_Type) return Count is
begin
  return Total_Out (Stream.Reader);
end Read_Total_Out;

-----
-- Write --
-----

procedure Write
  (Stream : in out Stream_Type;
   Item   : in   Stream_Element_Array)
is
  procedure Write (Item : in Stream_Element_Array);

  -----
  -- Write --
  -----

  procedure Write (Item : in Stream_Element_Array) is
  begin
    Ada.Streams.Write (Stream.Back.all, Item);
  end Write;

  procedure Write is new ZLib.Write
    (Write      => Write,
     Buffer_Size => Stream.Buffer_Size);

begin
  Write (Stream.Writer, Item, No_Flush);
end Write;

-----
-- Write_Total_In --
-----

function Write_Total_In (Stream : in Stream_Type) return Count is
begin
  return Total_In (Stream.Writer);
end Write_Total_In;

-----
-- Write_Total_Out --
-----

function Write_Total_Out (Stream : in Stream_Type) return Count is
begin
  return Total_Out (Stream.Writer);
```

```
    end Write_Total_Out;  
end ZLib.Streams;
```

```

-----
-- ZLib for Ada thick binding.
--
-- Copyright (C) 2002-2003 Dmitriy Anisimkov
--
-- Open source license information is in the zlib.ads file.
-----

-- $Id: zlib-streams.ads,v 1.1 2005/09/23 22:39:01 beng Exp $

package ZLib.Streams is

  type Stream_Mode is (In_Stream, Out_Stream, Duplex);

  type Stream_Access is access all Ada.Streams.Root_Stream_Type'Class;

  type Stream_Type is
    new Ada.Streams.Root_Stream_Type with private;

  procedure Read
    (Stream : in out Stream_Type;
     Item   : out Ada.Streams.Stream_Element_Array;
     Last   : out Ada.Streams.Stream_Element_Offset);

  procedure Write
    (Stream : in out Stream_Type;
     Item   : in Ada.Streams.Stream_Element_Array);

  procedure Flush
    (Stream : in out Stream_Type;
     Mode   : in Flush_Mode := Sync_Flush);
  -- Flush the written data to the back stream,
  -- all data placed to the compressor is flushing to the Back stream.
  -- Should not be used untill necessary, because it is decreasing
  -- compression.

  function Read_Total_In (Stream : in Stream_Type) return Count;
  pragma Inline (Read_Total_In);
  -- Return total number of bytes read from back stream so far.

  function Read_Total_Out (Stream : in Stream_Type) return Count;
  pragma Inline (Read_Total_Out);
  -- Return total number of bytes read so far.

  function Write_Total_In (Stream : in Stream_Type) return Count;
  pragma Inline (Write_Total_In);
  -- Return total number of bytes written so far.

  function Write_Total_Out (Stream : in Stream_Type) return Count;
  pragma Inline (Write_Total_Out);
  -- Return total number of bytes written to the back stream.

  procedure Create
    (Stream           : out Stream_Type;
     Mode             : in Stream_Mode;
     Back             : in Stream_Access;
     Back_Compressed  : in Boolean;
     Level            : in Compression_Level := Default_Compression;
     Strategy         : in Strategy_Type := Default_Strategy;
     Header           : in Header_Type := Default;
     Read_Buffer_Size : in Ada.Streams.Stream_Element_Offset
                          := Default_Buffer_Size;
     Write_Buffer_Size : in Ada.Streams.Stream_Element_Offset
                          := Default_Buffer_Size);
  -- Create the Compression/Decompression stream.
  -- If mode is In_Stream then Write operation is disabled.
  -- If mode is Out_Stream then Read operation is disabled.

  -- If Back_Compressed is true then
  -- Data written to the Stream is compressing to the Back stream
  -- and data read from the Stream is decompressed data from the Back stream.

  -- If Back_Compressed is false then
  -- Data written to the Stream is decompressing to the Back stream

```



```
-- and data read from the Stream is compressed data from the Back stream.

-- !!! When the Need_Header is False ZLib-Ada is using undocumented
-- ZLib 1.1.4 functionality to do not create/wait for ZLib headers.

function Is_Open (Stream : Stream_Type) return Boolean;

procedure Close (Stream : in out Stream_Type);

private

use Ada.Streams;

type Buffer_Access is access all Stream_Element_Array;

type Stream_Type
  is new Root_Stream_Type with
  record
    Mode          : Stream_Mode;

    Buffer         : Buffer_Access;
    Rest_First    : Stream_Element_Offset;
    Rest_Last     : Stream_Element_Offset;
    -- Buffer for Read operation.
    -- We need to have this buffer in the record
    -- because not all read data from back stream
    -- could be processed during the read operation.

    Buffer_Size    : Stream_Element_Offset;
    -- Buffer size for write operation.
    -- We do not need to have this buffer
    -- in the record because all data could be
    -- processed in the write operation.

    Back          : Stream_Access;
    Reader        : Filter_Type;
    Writer        : Filter_Type;
  end record;

end ZLib.Streams;
```

```
-----
-- ZLib for Ada thick binding.                                     --
--                                                                 --
-- Copyright (C) 2002-2003 Dmitriy Anisimkov                     --
--                                                                 --
-- Open source license information is in the zlib.ads file.      --
-----

-- $Id: zlib-thin.adb,v 1.1 2005/09/23 22:39:01 beng Exp $

package body ZLib.Thin is

  ZLIB_VERSION   : constant Chars_Ptr := zlibVersion;

  Z_Stream_Size  : constant Int := Z_Stream'Size / System.Storage_Unit;

  -----
  -- Avail_In --
  -----

  function Avail_In (Strm : in Z_Stream) return UInt is
  begin
    return Strm.Avail_In;
  end Avail_In;

  -----
  -- Avail_Out --
  -----

  function Avail_Out (Strm : in Z_Stream) return UInt is
  begin
    return Strm.Avail_Out;
  end Avail_Out;

  -----
  -- Deflate_Init --
  -----

  function Deflate_Init
    (strm      : Z_Stream;
     level     : Int;
     method    : Int;
     windowBits : Int;
     memLevel  : Int;
     strategy  : Int)
    return Int is
  begin
    return deflateInit2
      (strm,
       level,
       method,
       windowBits,
       memLevel,
       strategy,
       ZLIB_VERSION,
       Z_Stream_Size);
  end Deflate_Init;

  -----
  -- Inflate_Init --
  -----

  function Inflate_Init (strm : Z_Stream; windowBits : Int) return Int is
  begin
    return inflateInit2 (strm, windowBits, ZLIB_VERSION, Z_Stream_Size);
  end Inflate_Init;

  -----
  -- Last_Error_Message --
  -----

  function Last_Error_Message (Strm : in Z_Stream) return String is
  use Interfaces.C.Strings;
  begin
```

```
    if Strm.msg = Null_Ptr then
        return "";
    else
        return Value (Strm.msg);
    end if;
end Last_Error_Message;

-----
-- Set_In --
-----

procedure Set_In
  (Strm   : in out Z_Stream;
   Buffer : in      Voidp;
   Size   : in      UInt) is
begin
  Strm.Next_In := Buffer;
  Strm.Avail_In := Size;
end Set_In;

-----
-- Set_Mem_Func --
-----

procedure Set_Mem_Func
  (Strm   : in out Z_Stream;
   Opaque : in      Voidp;
   Alloc  : in      alloc_func;
   Free   : in      free_func) is
begin
  Strm.opaque := Opaque;
  Strm.zalloc := Alloc;
  Strm.zfree  := Free;
end Set_Mem_Func;

-----
-- Set_Out --
-----

procedure Set_Out
  (Strm   : in out Z_Stream;
   Buffer : in      Voidp;
   Size   : in      UInt) is
begin
  Strm.Next_Out := Buffer;
  Strm.Avail_Out := Size;
end Set_Out;

-----
-- Total_In --
-----

function Total_In (Strm : in Z_Stream) return ULong is
begin
  return Strm.Total_In;
end Total_In;

-----
-- Total_Out --
-----

function Total_Out (Strm : in Z_Stream) return ULong is
begin
  return Strm.Total_Out;
end Total_Out;

end ZLib.Thin;
```

```

-----
-- ZLib for Ada thick binding.
--
-- Copyright (C) 2002-2003 Dmitriy Anisimkov
--
-- Open source license information is in the zlib.ads file.
-----

-- $Id: zlib-thin.ads,v 1.1 2005/09/23 22:39:01 beng Exp $

with Interfaces.C.Strings;

with System;

private package ZLib.Thin is

  -- From zconf.h

  MAX_MEM_LEVEL : constant := 9;          -- zconf.h:105
                                           -- zconf.h:105
  MAX_WBITS : constant := 15;             -- zconf.h:115
                                           -- 32K LZ77 window
                                           -- zconf.h:115
  SEEK_SET : constant := 8#0000#;         -- zconf.h:244
                                           -- Seek from beginning of file.
                                           -- zconf.h:244
  SEEK_CUR : constant := 1;               -- zconf.h:245
                                           -- Seek from current position.
                                           -- zconf.h:245
  SEEK_END : constant := 2;               -- zconf.h:246
                                           -- Set file pointer to EOF plus "offset"
                                           -- zconf.h:246

  type Byte is new Interfaces.C.unsigned_char; -- 8 bits
                                           -- zconf.h:214
  type UInt is new Interfaces.C.unsigned;      -- 16 bits or more
                                           -- zconf.h:216
  type Int is new Interfaces.C.int;

  type ULong is new Interfaces.C.unsigned_long; -- 32 bits or more
                                           -- zconf.h:217
  subtype Chars_Ptr is Interfaces.C.Strings.chars_ptr;

  type ULong_Access is access ULong;
  type Int_Access is access Int;

  subtype Voidp is System.Address;           -- zconf.h:232

  subtype Byte_Access is Voidp;

  Nul : constant Voidp := System.Null_Address;
  -- end from zconf

  Z_NO_FLUSH : constant := 8#0000#;         -- zlib.h:125
                                           -- zlib.h:125
  Z_PARTIAL_FLUSH : constant := 1;          -- zlib.h:126
                                           -- will be removed, use
                                           -- Z_SYNC_FLUSH instead
                                           -- zlib.h:126
  Z_SYNC_FLUSH : constant := 2;             -- zlib.h:127
                                           -- zlib.h:127
  Z_FULL_FLUSH : constant := 3;             -- zlib.h:128
                                           -- zlib.h:128
  Z_FINISH : constant := 4;                 -- zlib.h:129
                                           -- zlib.h:129
  Z_OK : constant := 8#0000#;               -- zlib.h:132
                                           -- zlib.h:132
  Z_STREAM_END : constant := 1;             -- zlib.h:133
                                           -- zlib.h:133
  Z_NEED_DICT : constant := 2;              -- zlib.h:134
                                           -- zlib.h:134
  Z_ERRNO : constant := -1;                 -- zlib.h:135
                                           -- zlib.h:135
  Z_STREAM_ERROR : constant := -2;          -- zlib.h:136

```

```

-- zlib.h:136
Z_DATA_ERROR : constant := -3;      -- zlib.h:137
-- zlib.h:137
Z_MEM_ERROR : constant := -4;      -- zlib.h:138
-- zlib.h:138
Z_BUF_ERROR : constant := -5;      -- zlib.h:139
-- zlib.h:139
Z_VERSION_ERROR : constant := -6;   -- zlib.h:140
-- zlib.h:140
Z_NO_COMPRESSION : constant := 8#0000#; -- zlib.h:145
-- zlib.h:145
Z_BEST_SPEED : constant := 1;       -- zlib.h:146
-- zlib.h:146
Z_BEST_COMPRESSION : constant := 9;  -- zlib.h:147
-- zlib.h:147
Z_DEFAULT_COMPRESSION : constant := -1; -- zlib.h:148
-- zlib.h:148
Z_FILTERED : constant := 1;         -- zlib.h:151
-- zlib.h:151
Z_HUFFMAN_ONLY : constant := 2;     -- zlib.h:152
-- zlib.h:152
Z_DEFAULT_STRATEGY : constant := 8#0000#; -- zlib.h:153
-- zlib.h:153
Z_BINARY : constant := 8#0000#;    -- zlib.h:156
-- zlib.h:156
Z_ASCII : constant := 1;           -- zlib.h:157
-- zlib.h:157
Z_UNKNOWN : constant := 2;         -- zlib.h:158
-- zlib.h:158
Z_DEFLATED : constant := 8;        -- zlib.h:161
-- zlib.h:161
Z_NULL : constant := 8#0000#;      -- zlib.h:164
-- for initializing zalloc, zfree, opaque
-- zlib.h:164
type gzFile is new Voidp;          -- zlib.h:646

type Z_Stream is private;

type Z_Streamp is access all Z_Stream; -- zlib.h:89

type alloc_func is access function
  (Opaque : Voidp;
   Items  : UInt;
   Size   : UInt)
  return Voidp; -- zlib.h:63

type free_func is access procedure (opaque : Voidp; address : Voidp);

function zlibVersion return Chars_Ptr;

function Deflate (strm : Z_Streamp; flush : Int) return Int;

function DeflateEnd (strm : Z_Streamp) return Int;

function Inflate (strm : Z_Streamp; flush : Int) return Int;

function InflateEnd (strm : Z_Streamp) return Int;

function deflateSetDictionary
  (strm      : Z_Streamp;
   dictionary : Byte_Access;
   dictLength : UInt)
  return Int;

function deflateCopy (dest : Z_Streamp; source : Z_Streamp) return Int;
-- zlib.h:478

function deflateReset (strm : Z_Streamp) return Int; -- zlib.h:495

function deflateParams
  (strm      : Z_Streamp;
   level     : Int;
   strategy  : Int)
  return Int; -- zlib.h:506

```

```

function inflateSetDictionary
  (strm      : Z_Stream;
   dictionary : Byte_Access;
   dictLength : UInt)
  return      Int; --  zlib.h:548

function inflateSync (strm : Z_Stream) return Int; --  zlib.h:565

function inflateReset (strm : Z_Stream) return Int; --  zlib.h:580

function compress
  (dest      : Byte_Access;
   destLen   : ULong_Access;
   source    : Byte_Access;
   sourceLen : ULong)
  return      Int; --  zlib.h:601

function compress2
  (dest      : Byte_Access;
   destLen   : ULong_Access;
   source    : Byte_Access;
   sourceLen : ULong;
   level     : Int)
  return      Int; --  zlib.h:615

function uncompress
  (dest      : Byte_Access;
   destLen   : ULong_Access;
   source    : Byte_Access;
   sourceLen : ULong)
  return      Int;

function gzopen (path : Chars_Ptr; mode : Chars_Ptr) return gzFile;

function gzdopen (fd : Int; mode : Chars_Ptr) return gzFile;

function gzsetparams
  (file      : gzFile;
   level     : Int;
   strategy  : Int)
  return      Int;

function gzread
  (file : gzFile;
   buf  : Voidp;
   len  : UInt)
  return Int;

function gzwrite
  (file : in gzFile;
   buf  : in Voidp;
   len  : in UInt)
  return Int;

function gzprintf (file : in gzFile; format : in Chars_Ptr) return Int;

function gzputs (file : in gzFile; s : in Chars_Ptr) return Int;

function gzgets
  (file : gzFile;
   buf  : Chars_Ptr;
   len  : Int)
  return Chars_Ptr;

function gzputc (file : gzFile; char : Int) return Int;

function gzgetc (file : gzFile) return Int;

function gzflush (file : gzFile; flush : Int) return Int;

function gzseek
  (file      : gzFile;
   offset    : Int;

```

```
whence : Int)
return  Int;

function gzrewind (file : gzFile) return Int;

function gtell (file : gzFile) return Int;

function gzeof (file : gzFile) return Int;

function gzclose (file : gzFile) return Int;

function gzerror (file : gzFile; errnum : Int_Access) return Chars_Ptr;

function Adler32
(adler : ULong;
 buf   : Byte_Access;
 len   : UInt)
return  ULong;

function CRC32
(crc   : ULong;
 buf   : Byte_Access;
 len   : UInt)
return  ULong;

function deflateInit
(strm      : Z_Stream;
 level     : Int;
 version   : Chars_Ptr;
 stream_size : Int)
return     Int;

function deflateInit2
(strm      : Z_Stream;
 level     : Int;
 method    : Int;
 windowBits : Int;
 memLevel  : Int;
 strategy  : Int;
 version   : Chars_Ptr;
 stream_size : Int)
return     Int;

function Deflate_Init
(strm      : Z_Stream;
 level     : Int;
 method    : Int;
 windowBits : Int;
 memLevel  : Int;
 strategy  : Int)
return     Int;
pragma Inline (Deflate_Init);

function inflateInit
(strm      : Z_Stream;
 version   : Chars_Ptr;
 stream_size : Int)
return     Int;

function inflateInit2
(strm      : in Z_Stream;
 windowBits : in Int;
 version    : in Chars_Ptr;
 stream_size : in Int)
return     Int;

function inflateBackInit
(strm      : in Z_Stream;
 windowBits : in Int;
 window    : in Byte_Access;
 version    : in Chars_Ptr;
 stream_size : in Int)
return     Int;
-- Size of window have to be 2**windowBits.
```

```

function Inflate_Init (strm : Z_Stream; windowBits : Int) return Int;
pragma Inline (Inflate_Init);

function zError (err : Int) return Chars_Ptr;

function inflateSyncPoint (z : Z_Stream) return Int;

function get_crc_table return ULong_Access;

-- Interface to the available fields of the z_stream structure.
-- The application must update next_in and avail_in when avail_in has
-- dropped to zero. It must update next_out and avail_out when avail_out
-- has dropped to zero. The application must initialize zalloc, zfree and
-- opaque before calling the init function.

procedure Set_In
  (Strm   : in out Z_Stream;
   Buffer  : in Voidp;
   Size   : in UInt);
pragma Inline (Set_In);

procedure Set_Out
  (Strm   : in out Z_Stream;
   Buffer  : in Voidp;
   Size   : in UInt);
pragma Inline (Set_Out);

procedure Set_Mem_Func
  (Strm   : in out Z_Stream;
   Opaque : in Voidp;
   Alloc  : in alloc_func;
   Free   : in free_func);
pragma Inline (Set_Mem_Func);

function Last_Error_Message (Strm : in Z_Stream) return String;
pragma Inline (Last_Error_Message);

function Avail_Out (Strm : in Z_Stream) return UInt;
pragma Inline (Avail_Out);

function Avail_In (Strm : in Z_Stream) return UInt;
pragma Inline (Avail_In);

function Total_In (Strm : in Z_Stream) return ULong;
pragma Inline (Total_In);

function Total_Out (Strm : in Z_Stream) return ULong;
pragma Inline (Total_Out);

function inflateCopy
  (dest   : in Z_Stream;
   Source : in Z_Stream)
  return Int;

function compressBound (Source_Len : in ULong) return ULong;

function deflateBound
  (Strm       : in Z_Stream;
   Source_Len : in ULong)
  return ULong;

function gzungetc (C : in Int; File : in gzFile) return Int;

function zlibCompileFlags return ULong;

private

type Z_Stream is record
  Next_In   : Voidp;      := Nul;  -- zlib.h:68
  Avail_In  : UInt;       := 0;    -- next input byte
  Total_In  : ULong;      := 0;    -- number of bytes available at next_in
  Next_Out  : Voidp;      := Nul;  -- total nb of input bytes read so far
  Avail_Out : UInt;       := 0;    -- next output byte should be put there
  remaining : ULong;      := 0;    -- remaining free space at next_out

```



```

Total_Out : ULong      := 0;      -- total nb of bytes output so far
msg       : Chars_Ptr;  -- last error message, NULL if no error
state     : Voidp;      -- not visible by applications
zalloc    : alloc_func := null;   -- used to allocate the internal state
zfree     : free_func  := null;   -- used to free the internal state
opaque    : Voidp;      -- private data object passed to
                        -- zalloc and zfree
data_type : Int;        -- best guess about the data type:
                        -- ascii or binary
adler     : ULong;      -- Adler32 value of the uncompressed
                        -- data
reserved  : ULong;      -- reserved for future use
end record;

```

```
pragma Convention (C, Z_Stream);
```

```

pragma Import (C, zlibVersion, "zlibVersion");
pragma Import (C, Deflate, "deflate");
pragma Import (C, DeflateEnd, "deflateEnd");
pragma Import (C, Inflate, "inflate");
pragma Import (C, InflateEnd, "inflateEnd");
pragma Import (C, deflateSetDictionary, "deflateSetDictionary");
pragma Import (C, deflateCopy, "deflateCopy");
pragma Import (C, deflateReset, "deflateReset");
pragma Import (C, deflateParams, "deflateParams");
pragma Import (C, inflateSetDictionary, "inflateSetDictionary");
pragma Import (C, inflateSync, "inflateSync");
pragma Import (C, inflateReset, "inflateReset");
pragma Import (C, compress, "compress");
pragma Import (C, compress2, "compress2");
pragma Import (C, uncompress, "uncompress");
pragma Import (C, gzopen, "gzopen");
pragma Import (C, gzdopen, "gzdopen");
pragma Import (C, gzsetparams, "gzsetparams");
pragma Import (C, gzread, "gzread");
pragma Import (C, gzwrite, "gzwrite");
pragma Import (C, gzprintf, "gzprintf");
pragma Import (C, gzputs, "gzputs");
pragma Import (C, gzgets, "gzgets");
pragma Import (C, gzputc, "gzputc");
pragma Import (C, gzgetc, "gzgetc");
pragma Import (C, gzflush, "gzflush");
pragma Import (C, gzseek, "gzseek");
pragma Import (C, gzrewind, "gzrewind");
pragma Import (C, gztell, "gztell");
pragma Import (C, gzeof, "gzeof");
pragma Import (C, gzclose, "gzclose");
pragma Import (C, gzerror, "gzerror");
pragma Import (C, adler32, "adler32");
pragma Import (C, crc32, "crc32");
pragma Import (C, deflateInit, "deflateInit_");
pragma Import (C, inflateInit, "inflateInit_");
pragma Import (C, deflateInit2, "deflateInit2_");
pragma Import (C, inflateInit2, "inflateInit2_");
pragma Import (C, zError, "zError");
pragma Import (C, inflateSyncPoint, "inflateSyncPoint");
pragma Import (C, get_crc_table, "get_crc_table");

```

```
-- since zlib 1.2.0:
```

```

pragma Import (C, inflateCopy, "inflateCopy");
pragma Import (C, compressBound, "compressBound");
pragma Import (C, deflateBound, "deflateBound");
pragma Import (C, gzungetc, "gzungetc");
pragma Import (C, zlibCompileFlags, "zlibCompileFlags");

```

```
pragma Import (C, inflateBackInit, "inflateBackInit_");
```

```

-- I stopped binding the inflateBack routines, because realize that
-- it does not support zlib and gzip headers for now, and have no
-- symmetric deflateBack routines.
-- ZLib-Ada is symmetric regarding deflate/inflate data transformation
-- and has a similar generic callback interface for the
-- deflate/inflate transformation based on the regular Deflate/Inflate

```

```
-- routines.  
  
-- pragma Import (C, inflateBack, "inflateBack");  
-- pragma Import (C, inflateBackEnd, "inflateBackEnd");  
  
end ZLib.Thin;
```

```

-----
-- ZLib for Ada thick binding.
--
-- Copyright (C) 2002-2004 Dmitriy Anisimkov
--
-- Open source license information is in the zlib.ads file.
-----

-- $Id: zlib.adb,v 1.1 2005/09/23 22:39:01 beng Exp $

with Ada.Exceptions;
with Ada.Unchecked_Conversion;
with Ada.Unchecked_Deallocation;

with Interfaces.C.Strings;

with ZLib.Thin;

package body ZLib is

  use type Thin.Int;

  type Z_Stream is new Thin.Z_Stream;

  type Return_Code_Enum is
    (OK,
     STREAM_END,
     NEED_DICT,
     ERRNO,
     STREAM_ERROR,
     DATA_ERROR,
     MEM_ERROR,
     BUF_ERROR,
     VERSION_ERROR);

  type Flate_Step_Function is access
    function (Strm : in Thin.Z_Stream; Flush : in Thin.Int) return Thin.Int;
  pragma Convention (C, Flate_Step_Function);

  type Flate_End_Function is access
    function (Ctrm : in Thin.Z_Stream) return Thin.Int;
  pragma Convention (C, Flate_End_Function);

  type Flate_Type is record
    Step : Flate_Step_Function;
    Done : Flate_End_Function;
  end record;

  subtype Footer_Array is Stream_Element_Array (1 .. 8);

  Simple_GZip_Header : constant Stream_Element_Array (1 .. 10)
    := (16#1f#, 16#8b#, -- Magic header
        16#08#, -- Z_DEFLATED
        16#00#, -- Flags
        16#00#, 16#00#, 16#00#, 16#00#, -- Time
        16#00#, -- XFlags
        16#03#, -- OS code
        );
  -- The simplest gzip header is not for informational, but just for
  -- gzip format compatibility.
  -- Note that some code below is using assumption
  -- Simple_GZip_Header'Last > Footer_Array'Last, so do not make
  -- Simple_GZip_Header'Last <= Footer_Array'Last.

  Return_Code : constant array (Thin.Int range <>) of Return_Code_Enum
    := (0 => OK,
        1 => STREAM_END,
        2 => NEED_DICT,
        -1 => ERRNO,
        -2 => STREAM_ERROR,
        -3 => DATA_ERROR,
        -4 => MEM_ERROR,
        -5 => BUF_ERROR,
        -6 => VERSION_ERROR);

```

```

Flate : constant array (Boolean) of Flate_Type
:= (True => (Step => Thin.Deflate'Access,
            Done => Thin.DeflateEnd'Access),
    False => (Step => Thin.Inflate'Access,
            Done => Thin.InflateEnd'Access));

Flush_Finish : constant array (Boolean) of Flush_Mode
:= (True => Finish, False => No_Flush);

procedure Raise_Error (Stream : in Z_Stream);
pragma Inline (Raise_Error);

procedure Raise_Error (Message : in String);
pragma Inline (Raise_Error);

procedure Check_Error (Stream : in Z_Stream; Code : in Thin.Int);

procedure Free is new Ada.Unchecked_Deallocation
    (Z_Stream, Z_Stream_Access);

function To_Thin_Access is new Ada.Unchecked_Conversion
    (Z_Stream_Access, Thin.Z_Stream);

procedure Translate_GZip
    (Filter      : in out Filter_Type;
     In_Data     : in      Ada.Streams.Stream_Element_Array;
     In_Last     : out     Ada.Streams.Stream_Element_Offset;
     Out_Data    : out     Ada.Streams.Stream_Element_Array;
     Out_Last    : out     Ada.Streams.Stream_Element_Offset;
     Flush       : in      Flush_Mode);
-- Separate translate routine for make gzip header.

procedure Translate_Auto
    (Filter      : in out Filter_Type;
     In_Data     : in      Ada.Streams.Stream_Element_Array;
     In_Last     : out     Ada.Streams.Stream_Element_Offset;
     Out_Data    : out     Ada.Streams.Stream_Element_Array;
     Out_Last    : out     Ada.Streams.Stream_Element_Offset;
     Flush       : in      Flush_Mode);
-- translate routine without additional headers.

-----
-- Check_Error --
-----

procedure Check_Error (Stream : in Z_Stream; Code : in Thin.Int) is
    use type Thin.Int;
begin
    if Code /= Thin.Z_OK then
        Raise_Error
            (Return_Code_Enum'Image (Return_Code (Code))
             & " : " & Last_Error_Message (Stream));
    end if;
end Check_Error;

-----
-- Close --
-----

procedure Close
    (Filter      : in out Filter_Type;
     Ignore_Error : in      Boolean := False)
is
    Code : Thin.Int;
begin
    if not Ignore_Error and then not Is_Open (Filter) then
        raise Status_Error;
    end if;

    Code := Flate (Filter.Compression).Done (To_Thin_Access (Filter.Strm));

    if Ignore_Error or else Code = Thin.Z_OK then
        Free (Filter.Strm);

```

```

else
  declare
    Error_Message : constant String
      := Last_Error_Message (Filter.Strm.all);
  begin
    Free (Filter.Strm);
    Ada.Exceptions.Raise_Exception
      (ZLib_Error'Identity,
       Return_Code_Enum'Image (Return_Code (Code))
        & ":" & Error_Message);
  end;
end if;
end Close;

-----
-- CRC32 --
-----

function CRC32
  (CRC   : in Unsigned_32;
   Data  : in Ada.Streams.Stream_Element_Array)
  return Unsigned_32
is
  use Thin;
begin
  return Unsigned_32 (crc32 (ULong (CRC),
                             Data'Address,
                             Data'Length));
end CRC32;

procedure CRC32
  (CRC   : in out Unsigned_32;
   Data  : in Ada.Streams.Stream_Element_Array) is
begin
  CRC := CRC32 (CRC, Data);
end CRC32;

-----
-- Deflate_Init --
-----

procedure Deflate_Init
  (Filter      : in out Filter_Type;
   Level       : in Compression_Level := Default_Compression;
   Strategy    : in Strategy_Type     := Default_Strategy;
   Method      : in Compression_Method := Deflated;
   Window_Bits : in Window_Bits_Type  := Default_Window_Bits;
   Memory_Level : in Memory_Level_Type := Default_Memory_Level;
   Header      : in Header_Type        := Default)
is
  use type Thin.Int;
  Win_Bits : Thin.Int := Thin.Int (Window_Bits);
begin
  if Is_Open (Filter) then
    raise Status_Error;
  end if;

  -- We allow ZLib to make header only in case of default header type.
  -- Otherwise we would either do header by ourselves, or do not do
  -- header at all.

  if Header = None or else Header = GZip then
    Win_Bits := -Win_Bits;
  end if;

  -- For the GZip CRC calculation and make headers.

  if Header = GZip then
    Filter.CRC := 0;
    Filter.Offset := Simple_GZip_Header'First;
  else
    Filter.Offset := Simple_GZip_Header'Last + 1;
  end if;

```

```

Filter.Strm      := new Z_Stream;
Filter.Compression := True;
Filter.Stream_End := False;
Filter.Header     := Header;

if Thin.Deflate_Init
  (To_Thin_Access (Filter.Strm),
   Level          => Thin.Int (Level),
   method         => Thin.Int (Method),
   windowBits     => Win_Bits,
   memLevel       => Thin.Int (Memory_Level),
   strategy       => Thin.Int (Strategy)) /= Thin.Z_OK
then
  Raise_Error (Filter.Strm.all);
end if;
end Deflate_Init;

-----
-- Flush --
-----

procedure Flush
  (Filter      : in out Filter_Type;
   Out_Data    : out Ada.Streams.Stream_Element_Array;
   Out_Last    : out Ada.Streams.Stream_Element_Offset;
   Flush       : in Flush_Mode)
is
  No_Data : Stream_Element_Array := (1 .. 0 => 0);
  Last    : Stream_Element_Offset;
begin
  Translate (Filter, No_Data, Last, Out_Data, Out_Last, Flush);
end Flush;

-----
-- Generic_Translate --
-----

procedure Generic_Translate
  (Filter      : in out ZLib.Filter_Type;
   In_Buffer_Size : in Integer := Default_Buffer_Size;
   Out_Buffer_Size : in Integer := Default_Buffer_Size)
is
  In_Buffer  : Stream_Element_Array
    (1 .. Stream_Element_Offset (In_Buffer_Size));
  Out_Buffer : Stream_Element_Array
    (1 .. Stream_Element_Offset (Out_Buffer_Size));
  Last       : Stream_Element_Offset;
  In_Last    : Stream_Element_Offset;
  In_First   : Stream_Element_Offset;
  Out_Last   : Stream_Element_Offset;
begin
  Main : loop
    Data_In (In_Buffer, Last);

    In_First := In_Buffer'First;

    loop
      Translate
        (Filter    => Filter,
         In_Data   => In_Buffer (In_First .. Last),
         In_Last   => In_Last,
         Out_Data  => Out_Buffer,
         Out_Last  => Out_Last,
         Flush     => Flush_Finish (Last < In_Buffer'First));

      if Out_Buffer'First <= Out_Last then
        Data_Out (Out_Buffer (Out_Buffer'First .. Out_Last));
      end if;

      exit Main when Stream_End (Filter);

      -- The end of in buffer.

      exit when In_Last = Last;
    end loop;
  end loop;
end Generic_Translate;

```

```

        In_First := In_Last + 1;
    end loop;
end loop Main;

end Generic_Translate;

-----
-- Inflate_Init --
-----

procedure Inflate_Init
(Filter      : in out Filter_Type;
 Window_Bits : in      Window_Bits_Type := Default_Window_Bits;
 Header      : in      Header_Type      := Default)
is
    use type Thin.Int;
    Win_Bits : Thin.Int := Thin.Int (Window_Bits);

    procedure Check_Version;
    -- Check the latest header types compatibility.

    procedure Check_Version is
    begin
        if Version <= "1.1.4" then
            Raise_Error
                ("Inflate header type " & Header_Type'Image (Header)
                 & " incompatible with ZLib version " & Version);
        end if;
    end Check_Version;

begin
    if Is_Open (Filter) then
        raise Status_Error;
    end if;

    case Header is
        when None =>
            Check_Version;

            -- Inflate data without headers determined
            -- by negative Win_Bits.

            Win_Bits := -Win_Bits;
        when GZip =>
            Check_Version;

            -- Inflate gzip data defined by flag 16.

            Win_Bits := Win_Bits + 16;
        when Auto =>
            Check_Version;

            -- Inflate with automatic detection
            -- of gzip or native header defined by flag 32.

            Win_Bits := Win_Bits + 32;
        when Default => null;
    end case;

    Filter.Strm      := new Z_Stream;
    Filter.Compression := False;
    Filter.Stream_End := False;
    Filter.Header     := Header;

    if Thin.Inflate_Init
        (To_Thin_Access (Filter.Strm), Win_Bits) /= Thin.Z_OK
    then
        Raise_Error (Filter.Strm.all);
    end if;
end Inflate_Init;

-----
-- Is_Open --
-----

```

```

-----
function Is_Open (Filter : in Filter_Type) return Boolean is
begin
    return Filter.Strm /= null;
end Is_Open;

-----
-- Raise_Error --
-----

procedure Raise_Error (Message : in String) is
begin
    Ada.Exceptions.Raise_Exception (ZLib_Error'Identity, Message);
end Raise_Error;

procedure Raise_Error (Stream : in Z_Stream) is
begin
    Raise_Error (Last_Error_Message (Stream));
end Raise_Error;

-----
-- Read --
-----

procedure Read
(Filter : in out Filter_Type;
 Item   : out Ada.Streams.Stream_Element_Array;
 Last   : out Ada.Streams.Stream_Element_Offset;
 Flush  : in Flush_Mode := No_Flush)
is
    In_Last      : Stream_Element_Offset;
    Item_First    : Ada.Streams.Stream_Element_Offset := Item'First;
    V_Flush       : Flush_Mode := Flush;

begin
    pragma Assert (Rest_First in Buffer'First .. Buffer'Last + 1);
    pragma Assert (Rest_Last in Buffer'First - 1 .. Buffer'Last);

    loop
        if Rest_Last = Buffer'First - 1 then
            V_Flush := Finish;

        elsif Rest_First > Rest_Last then
            Read (Buffer, Rest_Last);
            Rest_First := Buffer'First;

            if Rest_Last < Buffer'First then
                V_Flush := Finish;
            end if;
        end if;

        Translate
            (Filter    => Filter,
             In_Data   => Buffer (Rest_First .. Rest_Last),
             In_Last   => In_Last,
             Out_Data  => Item (Item_First .. Item'Last),
             Out_Last  => Last,
             Flush     => V_Flush);

        Rest_First := In_Last + 1;

        exit when Stream_End (Filter)
            or else Last = Item'Last
            or else (Last >= Item'First and then Allow_Read_Some);

        Item_First := Last + 1;
    end loop;
end Read;

-----
-- Stream_End --
-----

```



```

function Stream_End (Filter : in Filter_Type) return Boolean is
begin
  if Filter.Header = GZip and Filter.Compression then
    return Filter.Stream_End
    and then Filter.Offset = Footer_Array'Last + 1;
  else
    return Filter.Stream_End;
  end if;
end Stream_End;

-----
-- Total_In --
-----

function Total_In (Filter : in Filter_Type) return Count is
begin
  return Count (Thin.Total_In (To_Thin_Access (Filter.Strm).all));
end Total_In;

-----
-- Total_Out --
-----

function Total_Out (Filter : in Filter_Type) return Count is
begin
  return Count (Thin.Total_Out (To_Thin_Access (Filter.Strm).all));
end Total_Out;

-----
-- Translate --
-----

procedure Translate
(Filter      : in out Filter_Type;
 In_Data    : in      Ada.Streams.Stream_Element_Array;
 In_Last    :          out Ada.Streams.Stream_Element_Offset;
 Out_Data   :          out Ada.Streams.Stream_Element_Array;
 Out_Last   :          out Ada.Streams.Stream_Element_Offset;
 Flush      : in      Flush_Mode) is
begin
  if Filter.Header = GZip and then Filter.Compression then
    Translate_GZip
      (Filter  => Filter,
       In_Data => In_Data,
       In_Last => In_Last,
       Out_Data => Out_Data,
       Out_Last => Out_Last,
       Flush   => Flush);
  else
    Translate_Auto
      (Filter  => Filter,
       In_Data => In_Data,
       In_Last => In_Last,
       Out_Data => Out_Data,
       Out_Last => Out_Last,
       Flush   => Flush);
  end if;
end Translate;

-----
-- Translate_Auto --
-----

procedure Translate_Auto
(Filter      : in out Filter_Type;
 In_Data    : in      Ada.Streams.Stream_Element_Array;
 In_Last    :          out Ada.Streams.Stream_Element_Offset;
 Out_Data   :          out Ada.Streams.Stream_Element_Array;
 Out_Last   :          out Ada.Streams.Stream_Element_Offset;
 Flush      : in      Flush_Mode)
is
  use type Thin.Int;
  Code : Thin.Int;

```

```

begin
  if not Is_Open (Filter) then
    raise Status_Error;
  end if;

  if Out_Data'Length = 0 and then In_Data'Length = 0 then
    raise Constraint_Error;
  end if;

  Set_Out (Filter.Strm.all, Out_Data'Address, Out_Data'Length);
  Set_In  (Filter.Strm.all, In_Data'Address, In_Data'Length);

  Code := Flate (Filter.Compression).Step
    (To_Thin_Access (Filter.Strm),
     Thin.Int (Flush));

  if Code = Thin.Z_STREAM_END then
    Filter.Stream_End := True;
  else
    Check_Error (Filter.Strm.all, Code);
  end if;

  In_Last  := In_Data'Last
    - Stream_Element_Offset (Avail_In (Filter.Strm.all));
  Out_Last := Out_Data'Last
    - Stream_Element_Offset (Avail_Out (Filter.Strm.all));
end Translate_Auto;

-----
-- Translate_GZip --
-----

procedure Translate_GZip
(Filter      : in out Filter_Type;
 In_Data    : in      Ada.Streams.Stream_Element_Array;
 In_Last    :         out Ada.Streams.Stream_Element_Offset;
 Out_Data    :         out Ada.Streams.Stream_Element_Array;
 Out_Last    :         out Ada.Streams.Stream_Element_Offset;
 Flush      : in      Flush_Mode)
is
  Out_First : Stream_Element_Offset;

  procedure Add_Data (Data : in Stream_Element_Array);
  -- Add data to stream from the Filter.Offset till necessary,
  -- used for add gzip headr/footer.

  procedure Put_32
    (Item : in out Stream_Element_Array;
     Data : in      Unsigned_32);
  pragma Inline (Put_32);

  -----
  -- Add_Data --
  -----

  procedure Add_Data (Data : in Stream_Element_Array) is
    Data_First : Stream_Element_Offset renames Filter.Offset;
    Data_Last  : Stream_Element_Offset;
    Data_Len   : Stream_Element_Offset; -- -1
    Out_Len    : Stream_Element_Offset; -- -1
  begin
    Out_First := Out_Last + 1;

    if Data_First > Data'Last then
      return;
    end if;

    Data_Len := Data'Last - Data_First;
    Out_Len  := Out_Data'Last - Out_First;

    if Data_Len <= Out_Len then
      Out_Last := Out_First + Data_Len;
      Data_Last := Data'Last;
    else

```

```

        Out_Last := Out_Data'Last;
        Data_Last := Data_First + Out_Len;
    end if;

    Out_Data (Out_First .. Out_Last) := Data (Data_First .. Data_Last);

    Data_First := Data_Last + 1;
    Out_First := Out_Last + 1;
end Add_Data;

-----
-- Put_32 --
-----

procedure Put_32
  (Item : in out Stream_Element_Array;
   Data : in      Unsigned_32)
is
  D : Unsigned_32 := Data;
begin
  for J in Item'First .. Item'First + 3 loop
    Item (J) := Stream_Element (D and 16#FF#);
    D := Shift_Right (D, 8);
  end loop;
end Put_32;

begin
  Out_Last := Out_Data'First - 1;

  if not Filter.Stream_End then
    Add_Data (Simple_GZip_Header);

    Translate_Auto
      (Filter => Filter,
       In_Data => In_Data,
       In_Last => In_Last,
       Out_Data => Out_Data (Out_First .. Out_Data'Last),
       Out_Last => Out_Last,
       Flush => Flush);

    CRC32 (Filter.CRC, In_Data (In_Data'First .. In_Last));
  end if;

  if Filter.Stream_End and then Out_Last <= Out_Data'Last then
    -- This detection method would work only when
    -- Simple_GZip_Header'Last > Footer_Array'Last

    if Filter.Offset = Simple_GZip_Header'Last + 1 then
      Filter.Offset := Footer_Array'First;
    end if;

    declare
      Footer : Footer_Array;
    begin
      Put_32 (Footer, Filter.CRC);
      Put_32 (Footer (Footer'First + 4 .. Footer'Last),
              Unsigned_32 (Total_In (Filter)));
      Add_Data (Footer);
    end;
  end if;
end Translate_GZip;

-----
-- Version --
-----

function Version return String is
begin
  return Interfaces.C.Strings.Value (Thin.zlibVersion);
end Version;

-----
-- Write --
-----

```

```
procedure Write
(Filter : in out Filter_Type;
 Item   : in      Ada.Streams.Stream_Element_Array;
 Flush  : in      Flush_Mode := No_Flush)
is
  Buffer      : Stream_Element_Array (1 .. Buffer_Size);
  In_Last    : Stream_Element_Offset;
  Out_Last   : Stream_Element_Offset;
  In_First   : Stream_Element_Offset := Item'First;
begin
  if Item'Length = 0 and Flush = No_Flush then
    return;
  end if;

  loop
    Translate
      (Filter    => Filter,
       In_Data   => Item (In_First .. Item'Last),
       In_Last   => In_Last,
       Out_Data  => Buffer,
       Out_Last  => Out_Last,
       Flush     => Flush);

    if Out_Last >= Buffer'First then
      Write (Buffer (1 .. Out_Last));
    end if;

    exit when In_Last = Item'Last or Stream_End (Filter);

    In_First := In_Last + 1;
  end loop;
end Write;

end ZLib;
```

```

-----
--                               ZLib for Ada thick binding.                               --
--                                                                                       --
--                               Copyright (C) 2002-2004 Dmitriy Anisimkov              --
--                                                                                       --
--   This library is free software; you can redistribute it and/or modify              --
--   it under the terms of the GNU General Public License as published by              --
--   the Free Software Foundation; either version 2 of the License, or (at           --
--   your option) any later version.                                                    --
--                                                                                       --
--   This library is distributed in the hope that it will be useful, but               --
--   WITHOUT ANY WARRANTY; without even the implied warranty of                       --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU               --
--   General Public License for more details.                                          --
--                                                                                       --
--   You should have received a copy of the GNU General Public License                 --
--   along with this library; if not, write to the Free Software Foundation,         --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                  --
--                                                                                       --
--   As a special exception, if other files instantiate generics from this            --
--   unit, or you link this unit with other files to produce an executable,          --
--   this unit does not by itself cause the resulting executable to be               --
--   covered by the GNU General Public License. This exception does not              --
--   however invalidate any other reasons why the executable file might be           --
--   covered by the GNU Public License.                                                --
-----

-- $Id: zlib.ads,v 1.1 2005/09/23 22:39:01 beng Exp $

with Ada.Streams;

with Interfaces;

package ZLib is

  ZLib_Error    : exception;
  Status_Error  : exception;

  type Compression_Level is new Integer range -1 .. 9;

  type Flush_Mode is private;

  type Compression_Method is private;

  type Window_Bits_Type is new Integer range 8 .. 15;

  type Memory_Level_Type is new Integer range 1 .. 9;

  type Unsigned_32 is new Interfaces.Unsigned_32;

  type Strategy_Type is private;

  type Header_Type is (None, Auto, Default, GZip);
  -- Header type usage have a some limitation for inflate.
  -- See comment for Inflate_Init.

  subtype Count is Ada.Streams.Stream_Element_Count;

  Default_Memory_Level : constant Memory_Level_Type := 8;
  Default_Window_Bits  : constant Window_Bits_Type  := 15;

  -----
  -- Compression method constants --
  -----

  Deflated : constant Compression_Method;
  -- Only one method allowed in this ZLib version

  -----
  -- Compression level constants --
  -----

  No_Compression      : constant Compression_Level := 0;
  Best_Speed          : constant Compression_Level := 1;

```

```

Best_Compression      : constant Compression_Level := 9;
Default_Compression   : constant Compression_Level := -1;

-----
-- Flush mode constants --
-----

No_Flush              : constant Flush_Mode;
-- Regular way for compression, no flush

Partial_Flush         : constant Flush_Mode;
-- Will be removed, use Z_SYNC_FLUSH instead

Sync_Flush            : constant Flush_Mode;
-- All pending output is flushed to the output buffer and the output
-- is aligned on a byte boundary, so that the decompressor can get all
-- input data available so far. (In particular avail_in is zero after the
-- call if enough output space has been provided before the call.)
-- Flushing may degrade compression for some compression algorithms and so
-- it should be used only when necessary.

Block_Flush           : constant Flush_Mode;
-- Z_BLOCK requests that inflate() stop
-- if and when it get to the next deflate block boundary. When decoding the
-- zlib or gzip format, this will cause inflate() to return immediately
-- after the header and before the first block. When doing a raw inflate,
-- inflate() will go ahead and process the first block, and will return
-- when it gets to the end of that block, or when it runs out of data.

Full_Flush            : constant Flush_Mode;
-- All output is flushed as with SYNC_FLUSH, and the compression state
-- is reset so that decompression can restart from this point if previous
-- compressed data has been damaged or if random access is desired. Using
-- Full_Flush too often can seriously degrade the compression.

Finish                : constant Flush_Mode;
-- Just for tell the compressor that input data is complete.

-----
-- Compression strategy constants --
-----

-- RLE strategy could be used only in version 1.2.0 and later.

Filtered              : constant Strategy_Type;
Huffman_Only          : constant Strategy_Type;
RLE                    : constant Strategy_Type;
Default_Strategy       : constant Strategy_Type;

Default_Buffer_Size    : constant := 4096;

type Filter_Type is tagged limited private;
-- The filter is for compression and for decompression.
-- The usage of the type is depend of its initialization.

function Version return String;
pragma Inline (Version);
-- Return string representation of the ZLib version.

procedure Deflate_Init
(Filter      : in out Filter_Type;
 Level      : in      Compression_Level := Default_Compression;
 Strategy   : in      Strategy_Type     := Default_Strategy;
 Method     : in      Compression_Method := Deflated;
 Window_Bits : in      Window_Bits_Type := Default_Window_Bits;
 Memory_Level : in      Memory_Level_Type := Default_Memory_Level;
 Header     : in      Header_Type       := Default);
-- Compressor initialization.
-- When Header parameter is Auto or Default, then default zlib header
-- would be provided for compressed data.
-- When Header is GZip, then gzip header would be set instead of
-- default header.
-- When Header is None, no header would be set for compressed data.

```

```

procedure Inflate_Init
  (Filter      : in out Filter_Type;
   Window_Bits : in      Window_Bits_Type := Default_Window_Bits;
   Header      : in      Header_Type      := Default);
-- Decompressor initialization.
-- Default header type mean that ZLib default header is expecting in the
-- input compressed stream.
-- Header type None mean that no header is expecting in the input stream.
-- GZip header type mean that GZip header is expecting in the
-- input compressed stream.
-- Auto header type mean that header type (GZip or Native) would be
-- detected automatically in the input stream.
-- Note that header types parameter values None, GZip and Auto are
-- supported for inflate routine only in ZLib versions 1.2.0.2 and later.
-- Deflate_Init is supporting all header types.

```

```

function Is_Open (Filter : in Filter_Type) return Boolean;

```

```

pragma Inline (Is_Open);

```

```

-- Is the filter opened for compression or decompression.

```

```

procedure Close

```

```

  (Filter      : in out Filter_Type;
   Ignore_Error : in      Boolean := False);
-- Closing the compression or decompressor.
-- If stream is closing before the complete and Ignore_Error is False,
-- The exception would be raised.

```

```

generic

```

```

  with procedure Data_In

```

```

    (Item : out Ada.Streams.Stream_Element_Array;

```

```

     Last : out Ada.Streams.Stream_Element_Offset);

```

```

  with procedure Data_Out

```

```

    (Item : in Ada.Streams.Stream_Element_Array);

```

```

procedure Generic_Translate

```

```

  (Filter      : in out Filter_Type;
   In_Buffer_Size  : in      Integer := Default_Buffer_Size;
   Out_Buffer_Size : in      Integer := Default_Buffer_Size);
-- Compress/decompress data fetch from Data_In routine and pass the result
-- to the Data_Out routine. User should provide Data_In and Data_Out
-- for compression/decompression data flow.
-- Compression or decompression depend on Filter initialization.

```

```

function Total_In (Filter : in Filter_Type) return Count;

```

```

pragma Inline (Total_In);

```

```

-- Returns total number of input bytes read so far

```

```

function Total_Out (Filter : in Filter_Type) return Count;

```

```

pragma Inline (Total_Out);

```

```

-- Returns total number of bytes output so far

```

```

function CRC32

```

```

  (CRC : in Unsigned_32;

```

```

   Data : in Ada.Streams.Stream_Element_Array)

```

```

  return Unsigned_32;

```

```

pragma Inline (CRC32);

```

```

-- Compute CRC32, it could be necessary for make gzip format

```

```

procedure CRC32

```

```

  (CRC : in out Unsigned_32;

```

```

   Data : in      Ada.Streams.Stream_Element_Array);

```

```

pragma Inline (CRC32);

```

```

-- Compute CRC32, it could be necessary for make gzip format

```

```

-----
-- Below is more complex low level routines. --
-----

```

```

procedure Translate

```

```

  (Filter      : in out Filter_Type;

```

```

   In_Data     : in      Ada.Streams.Stream_Element_Array;

```

```

   In_Last     : out      Ada.Streams.Stream_Element_Offset;

```

```

   Out_Data    : out      Ada.Streams.Stream_Element_Array;

```

```

   Out_Last    : out      Ada.Streams.Stream_Element_Offset;

```

```

   Flush      : in      Flush_Mode);

```

```
-- Compress/decompress the In_Data buffer and place the result into
-- Out_Data. In_Last is the index of last element from In_Data accepted by
-- the Filter. Out_Last is the last element of the received data from
-- Filter. To tell the filter that incoming data are complete put the
-- Flush parameter to Finish.
```

```
function Stream_End (Filter : in Filter_Type) return Boolean;
```

```
pragma Inline (Stream_End);
```

```
-- Return the true when the stream is complete.
```

```
procedure Flush
```

```
(Filter      : in out Filter_Type;
 Out_Data    :      out Ada.Streams.Stream_Element_Array;
 Out_Last    :      out Ada.Streams.Stream_Element_Offset;
 Flush       : in      Flush_Mode);
```

```
pragma Inline (Flush);
```

```
-- Flushing the data from the compressor.
```

```
generic
```

```
  with procedure Write
```

```
    (Item : in Ada.Streams.Stream_Element_Array);
```

```
-- User should provide this routine for accept
```

```
-- compressed/decompressed data.
```

```
  Buffer_Size : in Ada.Streams.Stream_Element_Offset
```

```
    := Default_Buffer_Size;
```

```
-- Buffer size for Write user routine.
```

```
procedure Write
```

```
(Filter : in out Filter_Type;
 Item    : in      Ada.Streams.Stream_Element_Array;
 Flush   : in      Flush_Mode := No_Flush);
```

```
-- Compress/Decompress data from Item to the generic parameter procedure
```

```
-- Write. Output buffer size could be set in Buffer_Size generic parameter.
```

```
generic
```

```
  with procedure Read
```

```
    (Item : out Ada.Streams.Stream_Element_Array;
```

```
     Last : out Ada.Streams.Stream_Element_Offset);
```

```
-- User should provide data for compression/decompression
```

```
-- thru this routine.
```

```
  Buffer : in out Ada.Streams.Stream_Element_Array;
```

```
-- Buffer for keep remaining data from the previous
```

```
-- back read.
```

```
  Rest_First, Rest_Last : in out Ada.Streams.Stream_Element_Offset;
```

```
-- Rest_First have to be initialized to Buffer'Last + 1
```

```
-- Rest_Last have to be initialized to Buffer'Last
```

```
-- before usage.
```

```
  Allow_Read_Some : in Boolean := False;
```

```
-- Is it allowed to return Last < Item'Last before end of data.
```

```
procedure Read
```

```
(Filter : in out Filter_Type;
 Item    :      out Ada.Streams.Stream_Element_Array;
 Last    :      out Ada.Streams.Stream_Element_Offset;
 Flush   : in      Flush_Mode := No_Flush);
```

```
-- Compress/Decompress data from generic parameter procedure Read to the
```

```
-- Item. User should provide Buffer and initialized Rest_First, Rest_Last
```

```
-- indicators. If Allow_Read_Some is True, Read routines could return
```

```
-- Last < Item'Last only at end of stream.
```

```
private
```

```
  use Ada.Streams;
```

```
pragma Assert (Ada.Streams.Stream_Element'Size = 8);
```

```
pragma Assert (Ada.Streams.Stream_Element'Modulus = 2**8);
```

```
type Flush_Mode is new Integer range 0 .. 5;
```

```
type Compression_Method is new Integer range 8 .. 8;
```



```
type Strategy_Type is new Integer range 0 .. 3;

No_Flush      : constant Flush_Mode := 0;
Partial_Flush : constant Flush_Mode := 1;
Sync_Flush    : constant Flush_Mode := 2;
Full_Flush    : constant Flush_Mode := 3;
Finish        : constant Flush_Mode := 4;
Block_Flush   : constant Flush_Mode := 5;

Filtered      : constant Strategy_Type := 1;
Huffman_Only  : constant Strategy_Type := 2;
RLE           : constant Strategy_Type := 3;
Default_Strategy : constant Strategy_Type := 0;

Deflated : constant Compression_Method := 8;

type Z_Stream;

type Z_Stream_Access is access all Z_Stream;

type Filter_Type is tagged limited record
  Strm      : Z_Stream_Access;
  Compression : Boolean;
  Stream_End : Boolean;
  Header     : Header_Type;
  CRC        : Unsigned_32;
  Offset     : Stream_Element_Offset;
  -- Offset for gzip header/footer output.
end record;

end ZLib;
```

```
project Zlib is

  for Languages use ("Ada");
  for Source_Dirs use (".");
  for Object_Dir use ".";
  for Main use ("test.adb", "mtest.adb", "read.adb", "buffer_demo");

  package Compiler is
    for Default_Switches ("ada") use ("-gnatwcfilopru", "-gnatVcdfimorst", "-gnatyabcef",
hiklmnoprst");
  end Compiler;

  package Linker is
    for Default_Switches ("ada") use ("-lz");
  end Linker;

  package Builder is
    for Default_Switches ("ada") use ("-s", "-gnatQ");
  end Builder;

end Zlib;
```

This is a patched version of zlib modified to use Pentium-optimized assembly code in the deflation algorithm. The files changed/added by this patch are:

README.586
match.S

The effectiveness of these modifications is a bit marginal, as the the program's bottleneck seems to be mostly L1-cache contention, for which there is no real way to work around without rewriting the basic algorithm. The speedup on average is around 5-10% (which is generally less than the amount of variance between subsequent executions). However, when used at level 9 compression, the cache contention can drop enough for the assembly version to achieve 10-20% speedup (and sometimes more, depending on the amount of overall redundancy in the files). Even here, though, cache contention can still be the limiting factor, depending on the nature of the program using the zlib library. This may also mean that better improvements will be seen on a Pentium with MMX, which suffers much less from L1-cache contention, but I have not yet verified this.

Note that this code has been tailored for the Pentium in particular, and will not perform well on the Pentium Pro (due to the use of a partial register in the inner loop).

If you are using an assembler other than GNU as, you will have to translate match.S to use your assembler's syntax. (Have fun.)

Brian Raiter
breadbox@muppetlabs.com
April, 1998

Added for zlib 1.1.3:

The patches come from
<http://www.muppetlabs.com/~breadbox/software/assembly.html>

To compile zlib with this asm file, copy match.S to the zlib directory then do:

```
CFLAGS="-O3 -DASMV" ./configure  
make OBJA=match.o
```

```

/* match.s -- Pentium-optimized version of longest_match()
 * Written for zlib 1.1.2
 * Copyright (C) 1998 Brian Raiter <breadbox@muppetlabs.com>
 *
 * This is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License.
 */

#ifndef NO_UNDERLINE
#define match_init      _match_init
#define longest_match    _longest_match
#endif

#define MAX_MATCH        (258)
#define MIN_MATCH        (3)
#define MIN_LOOKAHEAD    (MAX_MATCH + MIN_MATCH + 1)
#define MAX_MATCH_8      ((MAX_MATCH + 7) & ~7)

/* stack frame offsets */

#define wmask            0          /* local copy of s->wmask      */
#define window           4          /* local copy of s->window      */
#define windowbestlen     8          /* s->window + bestlen          */
#define chainlenscanend  12         /* high word: current chain len */
/* low word: last bytes sought */
#define scanstart        16         /* first two bytes of string    */
#define scanalign        20         /* dword-misalignment of string */
#define nicematch        24         /* a good enough match size    */
#define bestlen          28         /* size of best match so far    */
#define scan             32         /* ptr to string wanting match */

#define LocalVarsSize     (36)
/* saved ebx          36 */
/* saved edi          40 */
/* saved esi          44 */
/* saved ebp          48 */
/* return address     52 */
#define deflatestate      56        /* the function arguments      */
#define curmatch          60

/* Offsets for fields in the deflate_state structure. These numbers
 * are calculated from the definition of deflate_state, with the
 * assumption that the compiler will dword-align the fields. (Thus,
 * changing the definition of deflate_state could easily cause this
 * program to crash horribly, without so much as a warning at
 * compile time. Sigh.)
 */

/* All the +zlib1222add offsets are due to the addition of fields
 * in zlib in the deflate_state structure since the asm code was first written
 * (if you compile with zlib 1.0.4 or older, use "zlib1222add equ (-4)").
 * (if you compile with zlib between 1.0.5 and 1.2.2.1, use "zlib1222add equ 0").
 * if you compile with zlib 1.2.2.2 or later , use "zlib1222add equ 8").
 */

#define zlib1222add        (8)

#define dsWSzSize         (36+zlib1222add)
#define dsWMask           (44+zlib1222add)
#define dsWindow          (48+zlib1222add)
#define dsPrev            (56+zlib1222add)
#define dsMatchLen        (88+zlib1222add)
#define dsPrevMatch       (92+zlib1222add)
#define dsStrStart        (100+zlib1222add)
#define dsMatchStart      (104+zlib1222add)
#define dsLookahead       (108+zlib1222add)
#define dsPrevLen         (112+zlib1222add)
#define dsMaxChainLen     (116+zlib1222add)
#define dsGoodMatch       (132+zlib1222add)
#define dsNiceMatch       (136+zlib1222add)

.file "match.S"

```

```
.globl match_init, longest_match

.text

/* uInt longest_match(deflate_state *deflatestate, IPos curmatch) */
longest_match:

/* Save registers that the compiler may be using, and adjust %esp to
 * make room for our stack frame. */

    pushl    %ebp
    pushl    %edi
    pushl    %esi
    pushl    %ebx
    subl     $LocalVarsSize, %esp

/* Retrieve the function arguments. %ecx will hold cur_match
 * throughout the entire function. %edx will hold the pointer to the
 * deflate_state structure during the function's setup (before
 * entering the main loop). */

    movl     deflatestate(%esp), %edx
    movl     curmatch(%esp), %ecx

/* if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead; */

    movl     dsNiceMatch(%edx), %eax
    movl     dsLookahead(%edx), %ebx
    cmpl     %eax, %ebx
    jl       LookaheadLess
    movl     %eax, %ebx
LookaheadLess: movl     %ebx, nicematch(%esp)

/* register Bytef *scan = s->window + s->strstart; */

    movl     dsWindow(%edx), %esi
    movl     %esi, window(%esp)
    movl     dsStrStart(%edx), %ebp
    lea      (%esi,%ebp), %edi
    movl     %edi, scan(%esp)

/* Determine how many bytes the scan ptr is off from being
 * dword-aligned. */

    movl     %edi, %eax
    negl     %eax
    andl     $3, %eax
    movl     %eax, scanalign(%esp)

/* IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
 * s->strstart - (IPos)MAX_DIST(s) : NIL; */

    movl     dsWSize(%edx), %eax
    subl     $MIN_LOOKAHEAD, %eax
    subl     %eax, %ebp
    jg       LimitPositive
    xorl     %ebp, %ebp
LimitPositive:

/* unsigned chain_length = s->max_chain_length;
 * if (s->prev_length >= s->good_match) {
 *     chain_length >>= 2;
 * } */

    movl     dsPrevLen(%edx), %eax
    movl     dsGoodMatch(%edx), %ebx
    cmpl     %ebx, %eax
    movl     dsMaxChainLen(%edx), %ebx
    jl       LastMatchGood
    shr     $2, %ebx
LastMatchGood:

/* chainlen is decremented once beforehand so that the function can */
```

```

/* use the sign flag instead of the zero flag for the exit test.          */
/* It is then shifted into the high word, to make room for the scanend    */
/* scanend value, which it will always accompany.                        */
                                decl    %ebx
                                shll    $16, %ebx

/* int best_len = s->prev_length;                                         */
                                movl    dsPrevLen(%edx), %eax
                                movl    %eax, bestlen(%esp)

/* Store the sum of s->window + best_len in %esi locally, and in %esi.    */
                                addl    %eax, %esi
                                movl    %esi, windowbestlen(%esp)

/* register ush scan_start = *(ushf*)scan;                               */
/* register ush scan_end   = *(ushf*)(scan+best_len-1);                  */
                                movw    (%edi), %bx
                                movw    %bx, scanstart(%esp)
                                movw    -1(%edi,%eax), %bx
                                movl    %ebx, chainlenscanend(%esp)

/* Posf *prev = s->prev;                                                 */
/* uInt wmask = s->w_mask;                                               */
                                movl    dsPrev(%edx), %edi
                                movl    dsWMask(%edx), %edx
                                mov     %edx, wmask(%esp)

/* Jump into the main loop.                                             */
                                jmp      LoopEntry

.balign 16

/* do {
 *   match = s->window + cur_match;
 *   if (*(ushf*)(match+best_len-1) != scan_end ||
 *       *(ushf*)match != scan_start) continue;
 *   [...]
 * } while ((cur_match = prev[cur_match & wmask]) > limit
 *         && --chain_length != 0);
 *
 * Here is the inner loop of the function. The function will spend the
 * majority of its time in this loop, and majority of that time will
 * be spent in the first ten instructions.
 *
 * Within this loop:
 * %ebx = chainlenscanend - i.e., ((chainlen << 16) | scanend)
 * %ecx = curmatch
 * %edx = curmatch & wmask
 * %esi = windowbestlen - i.e., (window + bestlen)
 * %edi = prev
 * %ebp = limit
 *
 * Two optimization notes on the choice of instructions:
 *
 * The first instruction uses a 16-bit address, which costs an extra,
 * unpairable cycle. This is cheaper than doing a 32-bit access and
 * zeroing the high word, due to the 3-cycle misalignment penalty which
 * would occur half the time. This also turns out to be cheaper than
 * doing two separate 8-bit accesses, as the memory is so rarely in the
 * L1 cache.
 *
 * The window buffer, however, apparently spends a lot of time in the
 * cache, and so it is faster to retrieve the word at the end of the
 * match string with two 8-bit loads. The instructions that test the
 * word at the beginning of the match string, however, are executed
 * much less frequently, and there it was cheaper to use 16-bit
 * instructions, which avoided the necessity of saving off and
 * subsequently reloading one of the other registers.

```

```

*/
LookupLoop:
                                /* 1 U & V */
                                /* 2 U pipe */
                                /* 2 V pipe */
                                /* 3 U pipe */
                                /* 3 V pipe */
                                /* 4 U pipe */
                                /* 4 V pipe */
LoopEntry:  movb    (%edi,%edx,2), %cx
                                /* 5 U pipe */
                                /* 5 V pipe */
                                /* 6 U pipe */
                                /* 6 V pipe */
                                andl    %ecx, %edx
                                cmpb    %bl, %al
                                jnz     LookupLoop
                                movb    (%esi,%ecx), %ah
                                cmpb    %bh, %ah
                                jnz     LookupLoop
                                movl    window(%esp), %eax
                                movw    (%eax,%ecx), %ax
                                cmpw    scanstart(%esp), %ax
                                jnz     LookupLoop

/* Store the current value of chainlen. */

                                movl    %ebx, chainlenscanend(%esp)

/* Point %edi to the string under scrutiny, and %esi to the string we
/* are hoping to match it up with. In actuality, %esi and %edi are
/* both pointed (MAX_MATCH_8 - scanalign) bytes ahead, and %edx is
/* initialized to -(MAX_MATCH_8 - scanalign). */

                                movl    window(%esp), %esi
                                movl    scan(%esp), %edi
                                addl    %ecx, %esi
                                movl    scanalign(%esp), %eax
                                movl    $(-MAX_MATCH_8), %edx
                                lea     MAX_MATCH_8(%edi,%eax), %edi
                                lea     MAX_MATCH_8(%esi,%eax), %esi

/* Test the strings for equality, 8 bytes at a time. At the end,
* adjust %edx so that it is offset to the exact byte that mismatched.
*
* We already know at this point that the first three bytes of the
* strings match each other, and they can be safely passed over before
* starting the compare loop. So what this code does is skip over 0-3
* bytes, as much as necessary in order to dword-align the %edi
* pointer. (%esi will still be misaligned three times out of four.)
*
* It should be confessed that this loop usually does not represent
* much of the total running time. Replacing it with a more
* straightforward "rep cmpsb" would not drastically degrade
* performance.
*/
LoopCmps:
                                movl    (%esi,%edx), %eax
                                movl    (%edi,%edx), %ebx
                                xorl    %ebx, %eax
                                jnz     LeaveLoopCmps
                                movl    4(%esi,%edx), %eax
                                movl    4(%edi,%edx), %ebx
                                xorl    %ebx, %eax
                                jnz     LeaveLoopCmps4
                                addl    $8, %edx
                                jnz     LoopCmps
                                jmp     LenMaximum
LeaveLoopCmps4: addl    $4, %edx
LeaveLoopCmps:  testl    $0x0000FFFF, %eax
                                jnz     LenLower
                                addl    $2, %edx
                                shr    $16, %eax
LenLower:     subb     $1, %al
                                adcl    $0, %edx

/* Calculate the length of the match. If it is longer than MAX_MATCH,
/* then automatically accept it as the best possible match and leave.

```

```

        lea    (%edi,%edx), %eax
        movl   scan(%esp), %edi
        subl   %edi, %eax
        cmpl   $MAX_MATCH, %eax
        jge    LenMaximum

/* If the length of the match is not longer than the best match we      */
/* have so far, then forget it and return to the lookup loop.          */
/*                                                                    */

        movl   deflatestate(%esp), %edx
        movl   bestlen(%esp), %ebx
        cmpl   %ebx, %eax
        jg     LongerMatch
        movl   chainlenscanend(%esp), %ebx
        movl   windowbestlen(%esp), %esi
        movl   dsPrev(%edx), %edi
        movl   wmask(%esp), %edx
        andl   %ecx, %edx
        jmp    LookupLoop

/*
s→match_start = cur_match;
best_len = len;
if (len ≥ nice_match) break;
scan_end = *(ushf*)(scan+best_len-1);
*/

LongerMatch:
        movl   nicematch(%esp), %ebx
        movl   %eax, bestlen(%esp)
        movl   %ecx, dsMatchStart(%edx)
        cmpl   %ebx, %eax
        jge    LeaveNow
        movl   window(%esp), %esi
        addl   %eax, %esi
        movl   %esi, windowbestlen(%esp)
        movl   chainlenscanend(%esp), %ebx
        movw   -1(%edi,%eax), %bx
        movl   dsPrev(%edx), %edi
        movl   %ebx, chainlenscanend(%esp)
        movl   wmask(%esp), %edx
        andl   %ecx, %edx
        jmp    LookupLoop

/* Accept the current string, with the maximum possible length.      */
/*                                                                    */

LenMaximum:
        movl   deflatestate(%esp), %edx
        movl   $MAX_MATCH, bestlen(%esp)
        movl   %ecx, dsMatchStart(%edx)

/* if ((uInt)best_len ≤ s→lookahead) return (uInt)best_len;          */
/* return s→lookahead;                                              */
/*                                                                    */

LeaveNow:
        movl   deflatestate(%esp), %edx
        movl   bestlen(%esp), %ebx
        movl   dsLookahead(%edx), %eax
        cmpl   %eax, %ebx
        jg     LookaheadRet
        movl   %ebx, %eax

LookaheadRet:

/* Restore the stack and return from whence we came.                */
/*                                                                    */

        addl   $LocalVarsSize, %esp
        popl   %ebx
        popl   %esi
        popl   %edi
        popl   %ebp
match_init:
        ret

```


This is a patched version of zlib, modified to use Pentium-Pro-optimized assembly code in the deflation algorithm. The files changed/added by this patch are:

README.686
match.S

The speedup that this patch provides varies, depending on whether the compiler used to build the original version of zlib falls afoul of the PPro's speed traps. My own tests show a speedup of around 10-20% at the default compression level, and 20-30% using -9, against a version compiled using gcc 2.7.2.3. Your mileage may vary.

Note that this code has been tailored for the PPro/PII in particular, and will not perform particularly well on a Pentium.

If you are using an assembler other than GNU as, you will have to translate match.S to use your assembler's syntax. (Have fun.)

Brian Raiter
breadbox@muppetlabs.com
April, 1998

Added for zlib 1.1.3:

The patches come from
<http://www.muppetlabs.com/~breadbox/software/assembly.html>

To compile zlib with this asm file, copy match.S to the zlib directory then do:

```
CFLAGS="-O3 -DASMV" ./configure  
make OBJA=match.o
```

```

/* match.s -- Pentium-Pro-optimized version of longest_match()
 * Written for zlib 1.1.2
 * Copyright (C) 1998 Brian Raiter <breadbox@muppetlabs.com>
 *
 * This is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License.
 */

#ifndef NO_UNDERLINE
#define match_init      _match_init
#define longest_match    _longest_match
#endif

#define MAX_MATCH        (258)
#define MIN_MATCH        (3)
#define MIN_LOOKAHEAD    (MAX_MATCH + MIN_MATCH + 1)
#define MAX_MATCH_8      ((MAX_MATCH + 7) & ~7)

/* stack frame offsets */

#define chainlenwmask      0      /* high word: current chain len */
/* low word: s->wmask */
#define window              4      /* local copy of s->window */
#define windowbestlen      8      /* s->window + bestlen */
#define scanstart          16     /* first two bytes of string */
#define scanend            12     /* last two bytes of string */
#define scanalign          20     /* dword-misalignment of string */
#define nicematch          24     /* a good enough match size */
#define bestlen            28     /* size of best match so far */
#define scan              32     /* ptr to string wanting match */

#define LocalVarsSize      (36)
/* saved ebx */
/* saved edi */
/* saved esi */
/* saved ebp */
/* return address */
#define deflatestate       56     /* the function arguments */
#define curmatch           60

/* All the +zlib1222add offsets are due to the addition of fields
 * in zlib in the deflate_state structure since the asm code was first written
 * (if you compile with zlib 1.0.4 or older, use "zlib1222add equ (-4)").
 * (if you compile with zlib between 1.0.5 and 1.2.2.1, use "zlib1222add equ 0").
 * if you compile with zlib 1.2.2.2 or later , use "zlib1222add equ 8").
 */

#define zlib1222add        (8)

#define dsWSize            (36+zlib1222add)
#define dsWMask           (44+zlib1222add)
#define dsWindow          (48+zlib1222add)
#define dsPrev            (56+zlib1222add)
#define dsMatchLen        (88+zlib1222add)
#define dsPrevMatch       (92+zlib1222add)
#define dsStrStart        (100+zlib1222add)
#define dsMatchStart      (104+zlib1222add)
#define dsLookahead       (108+zlib1222add)
#define dsPrevLen         (112+zlib1222add)
#define dsMaxChainLen     (116+zlib1222add)
#define dsGoodMatch       (132+zlib1222add)
#define dsNiceMatch       (136+zlib1222add)

.file "match.S"

.globl match_init, longest_match

.text

/* uInt longest_match(deflate_state *deflatestate, IPos curmatch) */

longest_match:

```

```

/* Save registers that the compiler may be using, and adjust %esp to
/* make room for our stack frame.
*/
*/

        pushl    %ebp
        pushl    %edi
        pushl    %esi
        pushl    %ebx
        subl     $LocalVarsSize, %esp

/* Retrieve the function arguments. %ecx will hold cur_match
/* throughout the entire function. %edx will hold the pointer to the
/* deflate_state structure during the function's setup (before
/* entering the main loop).
*/
*/

        movl     deflatestate(%esp), %edx
        movl     curmatch(%esp), %ecx

/* uInt wmask = s->w_mask;
/* unsigned chain_length = s->max_chain_length;
/* if (s->prev_length ≥ s->good_match) {
/*     chain_length >= 2;
/* }
*/
*/

        movl     dsPrevLen(%edx), %eax
        movl     dsGoodMatch(%edx), %ebx
        cmpl     %ebx, %eax
        movl     dsWMask(%edx), %eax
        movl     dsMaxChainLen(%edx), %ebx
        jl       LastMatchGood
        shr     $2, %ebx
LastMatchGood:

/* chainlen is decremented once beforehand so that the function can
/* use the sign flag instead of the zero flag for the exit test.
/* It is then shifted into the high word, to make room for the wmask
/* value, which it will always accompany.
*/
*/

        decl     %ebx
        shll     $16, %ebx
        orl      %eax, %ebx
        movl     %ebx, chainlenwmask(%esp)

/* if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;
*/
*/

        movl     dsNiceMatch(%edx), %eax
        movl     dsLookahead(%edx), %ebx
        cmpl     %eax, %ebx
        jl       LookaheadLess
        movl     %eax, %ebx
LookaheadLess:  movl     %ebx, nicematch(%esp)

/* register Bytef *scan = s->>window + s->strstart;
*/
*/

        movl     dsWindow(%edx), %esi
        movl     %esi, window(%esp)
        movl     dsStrStart(%edx), %ebp
        lea      (%esi,%ebp), %edi
        movl     %edi, scan(%esp)

/* Determine how many bytes the scan ptr is off from being
/* dword-aligned.
*/
*/

        movl     %edi, %eax
        negl     %eax
        andl     $3, %eax
        movl     %eax, scanalign(%esp)

/* IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
/*     s->strstart - (IPos)MAX_DIST(s) : NIL;
*/
*/

        movl     dsWSize(%edx), %eax
        subl     $MIN_LOOKAHEAD, %eax
        subl     %eax, %ebp
        jg       LimitPositive

```

```

        xorl    %ebp, %ebp
LimitPositive:

/* int best_len = s->prev_length; */

        movl    dsPrevLen(%edx), %eax
        movl    %eax, bestlen(%esp)

/* Store the sum of s->window + best_len in %esi locally, and in %esi. */

        addl    %eax, %esi
        movl    %esi, windowbestlen(%esp)

/* register ush scan_start = *(ushf*)scan; */
/* register ush scan_end   = *(ushf*)(scan+best_len-1); */
/* Posf *prev = s->prev; */

        movzwl  (%edi), %ebx
        movl    %ebx, scanstart(%esp)
        movzwl  -1(%edi,%eax), %ebx
        movl    %ebx, scanend(%esp)
        movl    dsPrev(%edx), %edi

/* Jump into the main loop. */

        movl    chainlenwmask(%esp), %edx
        jmp     LoopEntry

.balign 16

/* do {
 *   match = s->window + cur_match;
 *   if (*(ushf*)(match+best_len-1) != scan_end ||
 *      *(ushf*)match != scan_start) continue;
 *   [...]
 * } while ((cur_match = prev[cur_match & wmask]) > limit
 *          && --chain_length != 0);
 *
 * Here is the inner loop of the function. The function will spend the
 * majority of its time in this loop, and majority of that time will
 * be spent in the first ten instructions.
 *
 * Within this loop:
 * %ebx = scanend
 * %ecx = curmatch
 * %edx = chainlenwmask - i.e., ((chainlen << 16) | wmask)
 * %esi = windowbestlen - i.e., (window + bestlen)
 * %edi = prev
 * %ebp = limit
 */
LookupLoop:
        andl    %edx, %ecx
        movzwl  (%edi,%ecx,2), %ecx
        cmpl    %ebp, %ecx
        jbe     LeaveNow
        subl    $0x00010000, %edx
        js      LeaveNow
LoopEntry:
        movzwl  -1(%esi,%ecx), %eax
        cmpl    %ebx, %eax
        jnz     LookupLoop
        movl    window(%esp), %eax
        movzwl  (%eax,%ecx), %eax
        cmpl    scanstart(%esp), %eax
        jnz     LookupLoop

/* Store the current value of chainlen. */

        movl    %edx, chainlenwmask(%esp)

/* Point %edi to the string under scrutiny, and %esi to the string we
 * are hoping to match it up with. In actuality, %esi and %edi are
 * both pointed (MAX_MATCH_8 - scanalign) bytes ahead, and %edx is
 * initialized to -(MAX_MATCH_8 - scanalign).

```

```

        movl    window(%esp), %esi
        movl    scan(%esp), %edi
        addl    %ecx, %esi
        movl    scanalign(%esp), %eax
        movl    $(-MAX_MATCH_8), %edx
        lea     MAX_MATCH_8(%edi,%eax), %edi
        lea     MAX_MATCH_8(%esi,%eax), %esi

/* Test the strings for equality, 8 bytes at a time. At the end,
 * adjust %edx so that it is offset to the exact byte that mismatched.
 *
 * We already know at this point that the first three bytes of the
 * strings match each other, and they can be safely passed over before
 * starting the compare loop. So what this code does is skip over 0-3
 * bytes, as much as necessary in order to dword-align the %edi
 * pointer. (%esi will still be misaligned three times out of four.)
 *
 * It should be confessed that this loop usually does not represent
 * much of the total running time. Replacing it with a more
 * straightforward "rep cmpsb" would not drastically degrade
 * performance.
 */
LoopCmps:
        movl    (%esi,%edx), %eax
        xorl    (%edi,%edx), %eax
        jnz     LeaveLoopCmps
        movl    4(%esi,%edx), %eax
        xorl    4(%edi,%edx), %eax
        jnz     LeaveLoopCmps4
        addl    $8, %edx
        jnz     LoopCmps
        jmp     LenMaximum
LeaveLoopCmps4:
        addl    $4, %edx
LeaveLoopCmps:
        testl   $0x0000FFFF, %eax
        jnz     LenLower
        addl    $2, %edx
        shrl    $16, %eax
LenLower:
        subb    $1, %al
        adcl    $0, %edx

/* Calculate the length of the match. If it is longer than MAX_MATCH,
 * then automatically accept it as the best possible match and leave.
 */
        lea     (%edi,%edx), %eax
        movl    scan(%esp), %edi
        subl    %edi, %eax
        cmpl    $MAX_MATCH, %eax
        jge     LenMaximum

/* If the length of the match is not longer than the best match we
 * have so far, then forget it and return to the lookup loop.
 */
        movl    deflatestate(%esp), %edx
        movl    bestlen(%esp), %ebx
        cmpl    %ebx, %eax
        jg      LongerMatch
        movl    windowbestlen(%esp), %esi
        movl    dsPrev(%edx), %edi
        movl    scanend(%esp), %ebx
        movl    chainlenwmask(%esp), %edx
        jmp     LookupLoop

/*
 * s->match_start = cur_match;
 * best_len = len;
 * if (len >= nice_match) break;
 * scan_end = *(ushf*)(scan+best_len-1);
 */
LongerMatch:
        movl    nicematch(%esp), %ebx
        movl    %eax, bestlen(%esp)
        movl    %ecx, dsMatchStart(%edx)
        cmpl    %ebx, %eax
        jge     LeaveNow
        movl    window(%esp), %esi
        addl    %eax, %esi

```

```
        movl    %esi, windowbestlen(%esp)
        movzwl  -1(%edi,%eax), %ebx
        movl    dsPrev(%edx), %edi
        movl    %ebx, scanend(%esp)
        movl    chainlenwmask(%esp), %edx
        jmp     LookupLoop

/* Accept the current string, with the maximum possible length.      */
LenMaximum:    movl    deflatestate(%esp), %edx
               movl    $MAX_MATCH, bestlen(%esp)
               movl    %ecx, dsMatchStart(%edx)

/* if ((uInt)best_len ≤ s→lookahead) return (uInt)best_len;        */
/* return s→lookahead;                                             */
LeaveNow:      movl    deflatestate(%esp), %edx
               movl    bestlen(%esp), %ebx
               movl    dsLookahead(%edx), %eax
               cmpl    %eax, %ebx
               jg      LookaheadRet
               movl    %ebx, %eax
LookaheadRet:

/* Restore the stack and return from whence we came.              */
               addl    $LocalVarsSize, %esp
               popl    %ebx
               popl    %esi
               popl    %edi
               popl    %ebp
match_init:   ret
```

```
blast: blast.c blast.h  
        cc -DTEST -o blast blast.c  
  
test: blast  
        blast < test.pk | cmp - test.txt  
  
clean:  
        rm -f blast blast.o
```

Read blast.h for purpose and usage.

Mark Adler
madler@alumni.caltech.edu


```

/* blast.c
 * Copyright (C) 2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in blast.h
 * version 1.1, 16 Feb 2003
 *
 * blast.c decompresses data compressed by the PKWare Compression Library.
 * This function provides functionality similar to the explode() function of
 * the PKWare library, hence the name "blast".
 *
 * This decompressor is based on the excellent format description provided by
 * Ben Rudiak-Gould in comp.compression on August 13, 2001. Interestingly, the
 * example Ben provided in the post is incorrect. The distance 110001 should
 * instead be 111000. When corrected, the example byte stream becomes:
 *
 *      00 04 82 24 25 8f 80 7f
 *
 * which decompresses to "AIAIAIAIAIAIA" (without the quotes).
 */

/*
 * Change history:
 *
 * 1.0  12 Feb 2003      - First version
 * 1.1  16 Feb 2003      - Fixed distance check for > 4 GB uncompressed data
 */

#include <setjmp.h>          /* for setjmp(), longjmp(), and jmp_buf */
#include "blast.h"          /* prototype for blast() */

#define local static        /* for local function definitions */
#define MAXBITS 13         /* maximum code length */
#define MAXWIN 4096        /* maximum window size */

/* input and output state */
struct state {
    /* input state */
    blast_in infun;          /* input function provided by user */
    void *inhow;             /* opaque information passed to infun() */
    unsigned char *in;       /* next input location */
    unsigned left;           /* available input at in */
    int bitbuf;             /* bit buffer */
    int bitcnt;             /* number of bits in bit buffer */

    /* input limit error return state for bits() and decode() */
    jmp_buf env;

    /* output state */
    blast_out outfun;        /* output function provided by user */
    void *outhow;           /* opaque information passed to outfun() */
    unsigned next;          /* index of next write location in out[] */
    int first;              /* true to check distances (for first 4K) */
    unsigned char out[MAXWIN]; /* output buffer and sliding window */
};

/*
 * Return need bits from the input stream. This always leaves less than
 * eight bits in the buffer. bits() works properly for need == 0.
 *
 * Format notes:
 *
 * - Bits are stored in bytes from the least significant bit to the most
 *   significant bit. Therefore bits are dropped from the bottom of the bit
 *   buffer, using shift right, and new bytes are appended to the top of the
 *   bit buffer, using shift left.
 */
local int bits(struct state *s, int need)
{
    int val;                /* bit accumulator */

    /* load at least need bits into val */
    val = s->bitbuf;
    while (s->bitcnt < need) {
        if (s->left == 0) {
            s->left = s->infun(s->inhow, &(s->in));

```

```

        if (s->left == 0) longjmp(s->env, 1);          /* out of input */
    }
    val |= (int)(*(s->in)++) << s->bitcnt;             /* load eight bits */
    s->left--;
    s->bitcnt += 8;
}

/* drop need bits and update buffer, always zero to seven bits left */
s->bitbuf = val >> need;
s->bitcnt -= need;

/* return need bits, zeroing the bits above that */
return val & ((1 << need) - 1);
}

/*
 * Huffman code decoding tables.  count[1..MAXBITS] is the number of symbols of
 * each length, which for a canonical code are stepped through in order.
 * symbol[] are the symbol values in canonical order, where the number of
 * entries is the sum of the counts in count[].  The decoding process can be
 * seen in the function decode() below.
 */
struct huffman {
    short *count;          /* number of symbols of each length */
    short *symbol;         /* canonically ordered symbols */
};

/*
 * Decode a code from the stream s using huffman table h.  Return the symbol or
 * a negative value if there is an error.  If all of the lengths are zero, i.e.
 * an empty code, or if the code is incomplete and an invalid code is received,
 * then -9 is returned after reading MAXBITS bits.
 *
 * Format notes:
 *
 * - The codes as stored in the compressed data are bit-reversed relative to
 *   a simple integer ordering of codes of the same lengths.  Hence below the
 *   bits are pulled from the compressed data one at a time and used to
 *   build the code value reversed from what is in the stream in order to
 *   permit simple integer comparisons for decoding.
 *
 * - The first code for the shortest length is all ones.  Subsequent codes of
 *   the same length are simply integer decrements of the previous code.  When
 *   moving up a length, a one bit is appended to the code.  For a complete
 *   code, the last code of the longest length will be all zeros.  To support
 *   this ordering, the bits pulled during decoding are inverted to apply the
 *   more "natural" ordering starting with all zeros and incrementing.
 */
local int decode(struct state *s, struct huffman *h)
{
    int len;          /* current number of bits in code */
    int code;         /* len bits being decoded */
    int first;        /* first code of length len */
    int count;        /* number of codes of length len */
    int index;        /* index of first code of length len in symbol table */
    int bitbuf;       /* bits from stream */
    int left;         /* bits left in next or left to process */
    short *next;      /* next number of codes */

    bitbuf = s->bitbuf;
    left = s->bitcnt;
    code = first = index = 0;
    len = 1;
    next = h->count + 1;
    while (1) {
        while (left--) {
            code |= (bitbuf & 1) ^ 1;  /* invert code */
            bitbuf >>= 1;
            count = *next++;
            if (code < first + count) { /* if length len, return symbol */
                s->bitbuf = bitbuf;
                s->bitcnt = (s->bitcnt - len) & 7;
                return h->symbol[index + (code - first)];
            }
        }
    }
}

```

```

        index += count;                /* else update for next length */
        first += count;
        first <= 1;
        code <= 1;
        len++;
    }
    left = (MAXBITS+1) - len;
    if (left == 0) break;
    if (s->left == 0) {
        s->left = s->infun(s->inhow, &(s->in));
        if (s->left == 0) longjmp(s->env, 1);    /* out of input */
    }
    bitbuf = *(s->in)++;
    s->left--;
    if (left > 8) left = 8;
}
return -9;                            /* ran out of codes */
}

/*
 * Given a list of repeated code lengths rep[0..n-1], where each byte is a
 * count (high four bits + 1) and a code length (low four bits), generate the
 * list of code lengths. This compaction reduces the size of the object code.
 * Then given the list of code lengths length[0..n-1] representing a canonical
 * Huffman code for n symbols, construct the tables required to decode those
 * codes. Those tables are the number of codes of each length, and the symbols
 * sorted by length, retaining their original order within each length. The
 * return value is zero for a complete code set, negative for an over-
 * subscribed code set, and positive for an incomplete code set. The tables
 * can be used if the return value is zero or positive, but they cannot be used
 * if the return value is negative. If the return value is zero, it is not
 * possible for decode() using that table to return an error--any stream of
 * enough bits will resolve to a symbol. If the return value is positive, then
 * it is possible for decode() using that table to return an error for received
 * codes past the end of the incomplete lengths.
 */
local int construct(struct huffman *h, const unsigned char *rep, int n)
{
    int symbol;        /* current symbol when stepping through length[] */
    int len;           /* current length when stepping through h->count[] */
    int left;          /* number of possible codes left of current length */
    short offs[MAXBITS+1]; /* offsets in symbol table for each length */
    short length[256]; /* code lengths */

    /* convert compact repeat counts into symbol bit length list */
    symbol = 0;
    do {
        len = *rep++;
        left = (len >> 4) + 1;
        len &= 15;
        do {
            length[symbol++] = len;
        } while (--left);
    } while (--n);
    n = symbol;

    /* count number of codes of each length */
    for (len = 0; len <= MAXBITS; len++)
        h->count[len] = 0;
    for (symbol = 0; symbol < n; symbol++)
        (h->count[length[symbol]])++; /* assumes lengths are within bounds */
    if (h->count[0] == n) /* no codes! */
        return 0;        /* complete, but decode() will fail */

    /* check for an over-subscribed or incomplete set of lengths */
    left = 1;           /* one possible code of zero length */
    for (len = 1; len <= MAXBITS; len++) {
        left <= 1;      /* one more bit, double codes left */
        left -= h->count[len]; /* deduct count from possible codes */
        if (left < 0) return left; /* over-subscribed--return negative */
    } /* left > 0 means incomplete */

    /* generate offsets into symbol table for each length for sorting */
    offs[1] = 0;

```

```

    for (len = 1; len < MAXBITS; len++)
        offs[len + 1] = offs[len] + h->count[len];

    /*
     * put symbols in table sorted by length, by symbol order within each
     * length
     */
    for (symbol = 0; symbol < n; symbol++)
        if (length[symbol] != 0)
            h->symbol[offs[length[symbol]]++] = symbol;

    /* return zero for complete set, positive for incomplete set */
    return left;
}

/*
 * Decode PKWare Compression Library stream.
 *
 * Format notes:
 *
 * - First byte is 0 if literals are uncoded or 1 if they are coded. Second
 *   byte is 4, 5, or 6 for the number of extra bits in the distance code.
 *   This is the base-2 logarithm of the dictionary size minus six.
 *
 * - Compressed data is a combination of literals and length/distance pairs
 *   terminated by an end code. Literals are either Huffman coded or
 *   uncoded bytes. A length/distance pair is a coded length followed by a
 *   coded distance to represent a string that occurs earlier in the
 *   uncompressed data that occurs again at the current location.
 *
 * - A bit preceding a literal or length/distance pair indicates which comes
 *   next, 0 for literals, 1 for length/distance.
 *
 * - If literals are uncoded, then the next eight bits are the literal, in the
 *   normal bit order in the stream, i.e. no bit-reversal is needed. Similarly,
 *   no bit reversal is needed for either the length extra bits or the distance
 *   extra bits.
 *
 * - Literal bytes are simply written to the output. A length/distance pair is
 *   an instruction to copy previously uncompressed bytes to the output. The
 *   copy is from distance bytes back in the output stream, copying for length
 *   bytes.
 *
 * - Distances pointing before the beginning of the output data are not
 *   permitted.
 *
 * - Overlapped copies, where the length is greater than the distance, are
 *   allowed and common. For example, a distance of one and a length of 518
 *   simply copies the last byte 518 times. A distance of four and a length of
 *   twelve copies the last four bytes three times. A simple forward copy
 *   ignoring whether the length is greater than the distance or not implements
 *   this correctly.
 */
local int decomp(struct state *s)
{
    int lit;           /* true if literals are coded */
    int dict;          /* log2(dictionary size) - 6 */
    int symbol;         /* decoded symbol, extra bits for distance */
    int len;           /* length for copy */
    int dist;          /* distance for copy */
    int copy;          /* copy counter */
    unsigned char *from, *to; /* copy pointers */

    static int virgin = 1; /* build tables once */
    static short litcnt[MAXBITS+1], litsym[256]; /* litcode memory */
    static short lencnt[MAXBITS+1], lensym[16]; /* lencode memory */
    static short distcnt[MAXBITS+1], distsym[64]; /* distcode memory */
    static struct huffman litcode = {litcnt, litsym}; /* length code */
    static struct huffman lencode = {lencnt, lensym}; /* length code */
    static struct huffman distcode = {distcnt, distsym}; /* distance code */
    /* bit lengths of literal codes */
    static const unsigned char litlen[] = {
        11, 124, 8, 7, 28, 7, 188, 13, 76, 4, 10, 8, 12, 10, 12, 10, 8, 23, 8,
        9, 7, 6, 7, 8, 7, 6, 55, 8, 23, 24, 12, 11, 7, 9, 11, 12, 6, 7, 22, 5,
        7, 24, 6, 11, 9, 6, 7, 22, 7, 11, 38, 7, 9, 8, 25, 11, 8, 11, 9, 12,

```

```

    8, 12, 5, 38, 5, 38, 5, 11, 7, 5, 6, 21, 6, 10, 53, 8, 7, 24, 10, 27,
    44, 253, 253, 253, 252, 252, 252, 13, 12, 45, 12, 45, 12, 61, 12, 45,
    44, 173};
    /* bit lengths of length codes 0..15 */
    static const unsigned char lenlen[] = {2, 35, 36, 53, 38, 23};
    /* bit lengths of distance codes 0..63 */
    static const unsigned char distlen[] = {2, 20, 53, 230, 247, 151, 248};
    static const short base[16] = {          /* base for length codes */
        3, 2, 4, 5, 6, 7, 8, 9, 10, 12, 16, 24, 40, 72, 136, 264};
    static const char extra[16] = {          /* extra bits for length codes */
        0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8};

    /* set up decoding tables (once--might not be thread-safe) */
    if (virgin) {
        construct(&litcode, litlen, sizeof(litlen));
        construct(&lencode, lenlen, sizeof(lenlen));
        construct(&distcode, distlen, sizeof(distlen));
        virgin = 0;
    }

    /* read header */
    lit = bits(s, 8);
    if (lit > 1) return -1;
    dict = bits(s, 8);
    if (dict < 4 || dict > 6) return -2;

    /* decode literals and length/distance pairs */
    do {
        if (bits(s, 1)) {
            /* get length */
            symbol = decode(s, &lencode);
            len = base[symbol] + bits(s, extra[symbol]);
            if (len == 519) break;          /* end code */

            /* get distance */
            symbol = len == 2 ? 2 : dict;
            dist = decode(s, &distcode) << symbol;
            dist += bits(s, symbol);
            dist++;
            if (s->first && dist > s->next)
                return -3;          /* distance too far back */

            /* copy length bytes from distance bytes back */
            do {
                to = s->out + s->next;
                from = to - dist;
                copy = MAXWIN;
                if (s->next < dist) {
                    from += copy;
                    copy = dist;
                }
                copy -= s->next;
                if (copy > len) copy = len;
                len -= copy;
                s->next += copy;
                do {
                    *to++ = *from++;
                } while (--copy);
                if (s->next == MAXWIN) {
                    if (s->outfun(s->outhow, s->out, s->next)) return 1;
                    s->next = 0;
                    s->first = 0;
                }
            } while (len != 0);
        }
        else {
            /* get literal and write it */
            symbol = lit ? decode(s, &litcode) : bits(s, 8);
            s->out[s->next++] = symbol;
            if (s->next == MAXWIN) {
                if (s->outfun(s->outhow, s->out, s->next)) return 1;
                s->next = 0;
                s->first = 0;
            }
        }
    }

```

```

    }
    } while (1);
    return 0;
}

/* See comments in blast.h */
int blast(blast_in infun, void *inhow, blast_out outfun, void *outhow)
{
    struct state s;          /* input/output state */
    int err;                 /* return value */

    /* initialize input state */
    s.infun = infun;
    s.inhow = inhow;
    s.left = 0;
    s.bitbuf = 0;
    s.bitcnt = 0;

    /* initialize output state */
    s.outfun = outfun;
    s.outhow = outhow;
    s.next = 0;
    s.first = 1;

    /* return if bits() or decode() tries to read past available input */
    if (setjmp(s.env) != 0) /* if came back here via longjmp(), */
        err = 2;           /* then skip decomp(), return error */
    else
        err = decomp(&s);  /* decompress */

    /* write any leftover output and update the error code if needed */
    if (err != 1 && s.next && s.outfun(s.outhow, s.out, s.next) && err == 0)
        err = 1;
    return err;
}

#ifdef TEST
/* Example of how to use blast() */
#include <stdio.h>
#include <stdlib.h>

#define CHUNK 16384

local unsigned inf(void *how, unsigned char **buf)
{
    static unsigned char hold[CHUNK];

    *buf = hold;
    return fread(hold, 1, CHUNK, (FILE *)how);
}

local int outf(void *how, unsigned char *buf, unsigned len)
{
    return fwrite(buf, 1, len, (FILE *)how) != len;
}

/* Decompress a PKWare Compression Library stream from stdin to stdout */
int main(void)
{
    int ret, n;

    /* decompress to stdout */
    ret = blast(inf, stdin, outf, stdout);
    if (ret != 0) fprintf(stderr, "blast error: %d\n", ret);

    /* see if there are any leftover bytes */
    n = 0;
    while (getchar() != EOF) n++;
    if (n) fprintf(stderr, "blast warning: %d unused bytes of input\n", n);

    /* return blast() error code */
    return ret;
}
#endif

```

```
/* blast.h -- interface for blast.c
Copyright (C) 2003 Mark Adler
version 1.1, 16 Feb 2003
```

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Mark Adler madler@alumni.caltech.edu

```
*/
```

```
/*
 * blast() decompresses the PKWare Data Compression Library (DCL) compressed
 * format. It provides the same functionality as the explode() function in
 * that library. (Note: PKWare overused the "implode" verb, and the format
 * used by their library implode() function is completely different and
 * incompatible with the implode compression method supported by PKZIP.)
 */
```

```
typedef unsigned (*blast_in)(void *how, unsigned char **buf);
```

```
typedef int (*blast_out)(void *how, unsigned char *buf, unsigned len);
```

```
/* Definitions for input/output functions passed to blast(). See below for
 * what the provided functions need to do.
 */
```

```
int blast(blast_in infun, void *inhow, blast_out outfun, void *outhow);
```

```
/* Decompress input to output using the provided infun() and outfun() calls.
 * On success, the return value of blast() is zero. If there is an error in
 * the source data, i.e. it is not in the proper format, then a negative value
 * is returned. If there is not enough input available or there is not enough
 * output space, then a positive error is returned.
 */
```

```
 * The input function is invoked: len = infun(how, &buf), where buf is set by
 * infun() to point to the input buffer, and infun() returns the number of
 * available bytes there. If infun() returns zero, then blast() returns with
 * an input error. (blast() only asks for input if it needs it.) inhow is for
 * use by the application to pass an input descriptor to infun(), if desired.
 */
```

```
 * The output function is invoked: err = outfun(how, buf, len), where the bytes
 * to be written are buf[0..len-1]. If err is not zero, then blast() returns
 * with an output error. outfun() is always called with len <= 4096. outhow
 * is for use by the application to pass an output descriptor to outfun(), if
 * desired.
 */
```

```
 * The return codes are:
```

```
 *   2: ran out of input before completing decompression
 *   1: output error before completing decompression
 *   0: successful decompression
 *  -1: literal flag not zero or one
 *  -2: dictionary size not in 4..6
 *  -3: distance is too far back
 */
```

```
 * At the bottom of blast.c is an example program that uses blast() that can be
 * compiled to produce a command-line decompression filter by defining TEST.
 */
```

AIAIAIAIAIAIA


```

{*****}
{
  Borland Delphi Supplemental Components
  ZLIB Data Compression Interface Unit
}
{
  Copyright (c) 1997,99 Borland Corporation
}
{*****}

{ Updated for zlib 1.2.x by Cosmin Truta <cosmint@cs.ubbcluj.ro> }

unit ZLib;

interface

uses SysUtils, Classes;

type
  TAlloc = function (AppData: Pointer; Items, Size: Integer): Pointer; cdecl;
  TFree = procedure (AppData, Block: Pointer); cdecl;

  // Internal structure. Ignore.
  TZStreamRec = packed record
    next_in: PChar;      // next input byte
    avail_in: Integer;    // number of bytes available at next_in
    total_in: Longint;    // total nb of input bytes read so far

    next_out: PChar;      // next output byte should be put here
    avail_out: Integer;   // remaining free space at next_out
    total_out: Longint;   // total nb of bytes output so far

    msg: PChar;           // last error message, NULL if no error
    internal: Pointer;    // not visible by applications

    zalloc: TAlloc;       // used to allocate the internal state
    zfree: TFree;         // used to free the internal state
    AppData: Pointer;     // private data object passed to zalloc and zfree

    data_type: Integer;   // best guess about the data type: ascii or binary
    adler: Longint;       // Adler32 value of the uncompressed data
    reserved: Longint;    // reserved for future use
  end;

  // Abstract ancestor class
  TCustomZlibStream = class(TStream)
  private
    FStrm: TStream;
    FStrmPos: Integer;
    FOnProgress: TNotifyEvent;
    FZRec: TZStreamRec;
    FBuffer: array [Word] of Char;
  protected
    procedure Progress(Sender: TObject); dynamic;
    property OnProgress: TNotifyEvent read FOnProgress write FOnProgress;
    constructor Create(Strm: TStream);
  end;

  { TCompressionStream compresses data on the fly as data is written to it, and
  stores the compressed data to another stream.

  TCompressionStream is write-only and strictly sequential. Reading from the
  stream will raise an exception. Using Seek to move the stream pointer
  will raise an exception.

  Output data is cached internally, written to the output stream only when
  the internal output buffer is full. All pending output data is flushed
  when the stream is destroyed.

  The Position property returns the number of uncompressed bytes of
  data that have been written to the stream so far.

  CompressionRate returns the on-the-fly percentage by which the original
  data has been compressed: (1 - (CompressedBytes / UncompressedBytes)) * 100
  If raw data size = 100 and compressed data size = 25, the CompressionRate

```

is 75%

The *OnProgress* event is called each time the output buffer is filled and written to the output stream. This is useful for updating a progress indicator when you are writing a large chunk of data to the compression stream in a single call.}

```
TCompressionLevel = (clNone, clFastest, clDefault, clMax);
```

```
TCompressionStream = class(TCustomZlibStream)
```

```
private
```

```
  function GetCompressionRate: Single;
```

```
public
```

```
  constructor Create(CompressionLevel: TCompressionLevel; Dest: TStream);
```

```
  destructor Destroy; override;
```

```
  function Read(var Buffer; Count: Longint): Longint; override;
```

```
  function Write(const Buffer; Count: Longint): Longint; override;
```

```
  function Seek(Offset: Longint; Origin: Word): Longint; override;
```

```
  property CompressionRate: Single read GetCompressionRate;
```

```
  property OnProgress;
```

```
end;
```

{ *TDecompressionStream* decompresses data on the fly as data is read from it.

Compressed data comes from a separate source stream. *TDecompressionStream* is read-only and unidirectional; you can seek forward in the stream, but not backwards. The special case of setting the stream position to zero is allowed. Seeking forward decompresses data until the requested position in the uncompressed data has been reached. Seeking backwards, seeking relative to the end of the stream, requesting the size of the stream, and writing to the stream will raise an exception.

The *Position* property returns the number of bytes of uncompressed data that have been read from the stream so far.

The *OnProgress* event is called each time the internal input buffer of compressed data is exhausted and the next block is read from the input stream. This is useful for updating a progress indicator when you are reading a large chunk of data from the decompression stream in a single call.}

```
TDecompressionStream = class(TCustomZlibStream)
```

```
public
```

```
  constructor Create(Source: TStream);
```

```
  destructor Destroy; override;
```

```
  function Read(var Buffer; Count: Longint): Longint; override;
```

```
  function Write(const Buffer; Count: Longint): Longint; override;
```

```
  function Seek(Offset: Longint; Origin: Word): Longint; override;
```

```
  property OnProgress;
```

```
end;
```

{ *CompressBuf* compresses data, buffer to buffer, in one call.

In: *InBuf* = ptr to compressed data

InBytes = number of bytes in *InBuf*

Out: *OutBuf* = ptr to newly allocated buffer containing decompressed data

OutBytes = number of bytes in *OutBuf* }

```
procedure CompressBuf(const InBuf: Pointer; InBytes: Integer;
```

```
  out OutBuf: Pointer; out OutBytes: Integer);
```

{ *DecompressBuf* decompresses data, buffer to buffer, in one call.

In: *InBuf* = ptr to compressed data

InBytes = number of bytes in *InBuf*

OutEstimate = zero, or est. size of the decompressed data

Out: *OutBuf* = ptr to newly allocated buffer containing decompressed data

OutBytes = number of bytes in *OutBuf* }

```
procedure DecompressBuf(const InBuf: Pointer; InBytes: Integer;
```

```
  OutEstimate: Integer; out OutBuf: Pointer; out OutBytes: Integer);
```

{ *DecompressToUserBuf* decompresses data, buffer to buffer, in one call.

In: *InBuf* = ptr to compressed data

InBytes = number of bytes in *InBuf*

```
    Out: OutBuf = ptr to user-allocated buffer to contain decompressed data
    BufSize = number of bytes in OutBuf }
procedure DecompressToUserBuf(const InBuf: Pointer; InBytes: Integer;
    const OutBuf: Pointer; BufSize: Integer);

const
    zlib_version = '1.2.3';

type
    EZlibError = class(Exception);
    ECompressionError = class(EZlibError);
    EDecompressionError = class(EZlibError);

implementation

uses ZLibConst;

const
    Z_NO_FLUSH      = 0;
    Z_PARTIAL_FLUSH = 1;
    Z_SYNC_FLUSH    = 2;
    Z_FULL_FLUSH    = 3;
    Z_FINISH        = 4;

    Z_OK            = 0;
    Z_STREAM_END    = 1;
    Z_NEED_DICT     = 2;
    Z_ERRNO         = (-1);
    Z_STREAM_ERROR  = (-2);
    Z_DATA_ERROR    = (-3);
    Z_MEM_ERROR     = (-4);
    Z_BUF_ERROR     = (-5);
    Z_VERSION_ERROR = (-6);

    Z_NO_COMPRESSION      = 0;
    Z_BEST_SPEED          = 1;
    Z_BEST_COMPRESSION    = 9;
    Z_DEFAULT_COMPRESSION = (-1);

    Z_FILTERED            = 1;
    Z_HUFFMAN_ONLY        = 2;
    Z_RLE                 = 3;
    Z_DEFAULT_STRATEGY    = 0;

    Z_BINARY      = 0;
    Z_ASCII       = 1;
    Z_UNKNOWN     = 2;

    Z_DEFLATED = 8;

{$L adler32.obj}
{$L compress.obj}
{$L crc32.obj}
{$L deflate.obj}
{$L infback.obj}
{$L inffast.obj}
{$L inflate.obj}
{$L inftrees.obj}
{$L trees.obj}
{$L uncompr.obj}
{$L zutil.obj}

procedure adler32; external;
procedure compressBound; external;
procedure crc32; external;
procedure deflateInit2; external;
procedure deflateParams; external;

function _malloc(Size: Integer): Pointer; cdecl;
begin
    Result := AllocMem(Size);
end;
```

```
procedure _free(Block: Pointer); cdecl;
begin
  FreeMem(Block);
end;

procedure _memset(P: Pointer; B: Byte; count: Integer); cdecl;
begin
  FillChar(P^, count, B);
end;

procedure _memcpy(dest, source: Pointer; count: Integer); cdecl;
begin
  Move(source^, dest^, count);
end;

// deflate compresses data
function deflateInit_(var strm: TZStreamRec; level: Integer; version: PChar;
  reccsize: Integer): Integer; external;
function deflate(var strm: TZStreamRec; flush: Integer): Integer; external;
function deflateEnd(var strm: TZStreamRec): Integer; external;

// inflate decompresses data
function inflateInit_(var strm: TZStreamRec; version: PChar;
  reccsize: Integer): Integer; external;
function inflate(var strm: TZStreamRec; flush: Integer): Integer; external;
function inflateEnd(var strm: TZStreamRec): Integer; external;
function inflateReset(var strm: TZStreamRec): Integer; external;

function zlibAllocMem(AppData: Pointer; Items, Size: Integer): Pointer; cdecl;
begin
  // GetMem(Result, Items*Size);
  Result := AllocMem(Items * Size);
end;

procedure zlibFreeMem(AppData, Block: Pointer); cdecl;
begin
  FreeMem(Block);
end;

{function zlibCheck(code: Integer): Integer;
begin
  Result := code;
  if code < 0 then
    raise EZlibError.Create('error');    ///!
end;}

function CCheck(code: Integer): Integer;
begin
  Result := code;
  if code < 0 then
    raise ECompressionError.Create('error'); ///!
end;

function DCheck(code: Integer): Integer;
begin
  Result := code;
  if code < 0 then
    raise EDecompressionError.Create('error'); ///!
end;

procedure CompressBuf(const InBuf: Pointer; InBytes: Integer;
  out OutBuf: Pointer; out OutBytes: Integer);
var
  strm: TZStreamRec;
  P: Pointer;
begin
  FillChar(strm, sizeof(strm), 0);
  strm.zalloc := zlibAllocMem;
  strm.zfree := zlibFreeMem;
  OutBytes := ((InBytes + (InBytes div 10) + 12) + 255) and not 255;
  GetMem(OutBuf, OutBytes);
```

```
try
  strm.next_in := InBuf;
  strm.avail_in := InBytes;
  strm.next_out := OutBuf;
  strm.avail_out := OutBytes;
  CCheck(deflateInit_(strm, Z_BEST_COMPRESSION, zlib_version, sizeof(strm)));
try
  while CCheck(deflate(strm, Z_FINISH)) <> Z_STREAM_END do
  begin
    P := OutBuf;
    Inc(OutBytes, 256);
    ReallocMem(OutBuf, OutBytes);
    strm.next_out := PChar(Integer(OutBuf) + (Integer(strm.next_out) - Integer(P)));
    strm.avail_out := 256;
  end;
finally
  CCheck(deflateEnd(strm));
end;
ReallocMem(OutBuf, strm.total_out);
OutBytes := strm.total_out;
except
  FreeMem(OutBuf);
  raise
end;
end;

procedure DecompressBuf(const InBuf: Pointer; InBytes: Integer;
  OutEstimate: Integer; out OutBuf: Pointer; out OutBytes: Integer);
var
  strm: TZStreamRec;
  P: Pointer;
  BufInc: Integer;
begin
  FillChar(strm, sizeof(strm), 0);
  strm.zalloc := zlibAllocMem;
  strm.zfree := zlibFreeMem;
  BufInc := (InBytes + 255) and not 255;
  if OutEstimate = 0 then
    OutBytes := BufInc
  else
    OutBytes := OutEstimate;
  GetMem(OutBuf, OutBytes);
  try
    strm.next_in := InBuf;
    strm.avail_in := InBytes;
    strm.next_out := OutBuf;
    strm.avail_out := OutBytes;
    DCheck(inflateInit_(strm, zlib_version, sizeof(strm)));
  try
    while DCheck(inflate(strm, Z_NO_FLUSH)) <> Z_STREAM_END do
    begin
      P := OutBuf;
      Inc(OutBytes, BufInc);
      ReallocMem(OutBuf, OutBytes);
      strm.next_out := PChar(Integer(OutBuf) + (Integer(strm.next_out) - Integer(P)));
      strm.avail_out := BufInc;
    end;
  finally
    DCheck(inflateEnd(strm));
  end;
  ReallocMem(OutBuf, strm.total_out);
  OutBytes := strm.total_out;
except
  FreeMem(OutBuf);
  raise
end;
end;

procedure DecompressToUserBuf(const InBuf: Pointer; InBytes: Integer;
  const OutBuf: Pointer; BufSize: Integer);
var
  strm: TZStreamRec;
begin
```

```
FillChar(strm, sizeof(strm), 0);
strm.zalloc := zlibAllocMem;
strm.zfree := zlibFreeMem;
strm.next_in := InBuf;
strm.avail_in := InBytes;
strm.next_out := OutBuf;
strm.avail_out := BufSize;
DCheck(inflateInit_(strm, zlib_version, sizeof(strm)));
try
  if DCheck(inflate(strm, Z_FINISH)) <> Z_STREAM_END then
    raise EZlibError.CreateRes(@sTargetBufferTooSmall);
finally
  DCheck(inflateEnd(strm));
end;
end;

// TCustomZlibStream

constructor TCustomZlibStream.Create(Strm: TStream);
begin
  inherited Create;
  FStrm := Strm;
  FStrmPos := Strm.Position;
  FZRec.zalloc := zlibAllocMem;
  FZRec.zfree := zlibFreeMem;
end;

procedure TCustomZlibStream.Progress(Sender: TObject);
begin
  if Assigned(FOnProgress) then FOnProgress(Sender);
end;

// TCompressionStream

constructor TCompressionStream.Create(CompressionLevel: TCompressionLevel;
  Dest: TStream);
const
  Levels: array [TCompressionLevel] of ShortInt =
    (Z_NO_COMPRESSION, Z_BEST_SPEED, Z_DEFAULT_COMPRESSION, Z_BEST_COMPRESSION);
begin
  inherited Create(Dest);
  FZRec.next_out := FBuffer;
  FZRec.avail_out := sizeof(FBuffer);
  CCheck(deflateInit_(FZRec, Levels[CompressionLevel], zlib_version, sizeof(FZRec)));
end;

destructor TCompressionStream.Destroy;
begin
  FZRec.next_in := nil;
  FZRec.avail_in := 0;
  try
    if FStrm.Position <> FStrmPos then FStrm.Position := FStrmPos;
    while (CCheck(deflate(FZRec, Z_FINISH)) <> Z_STREAM_END)
      and (FZRec.avail_out = 0) do
      begin
        FStrm.WriteBuffer(FBuffer, sizeof(FBuffer));
        FZRec.next_out := FBuffer;
        FZRec.avail_out := sizeof(FBuffer);
      end;
    if FZRec.avail_out < sizeof(FBuffer) then
      FStrm.WriteBuffer(FBuffer, sizeof(FBuffer) - FZRec.avail_out);
  finally
    deflateEnd(FZRec);
  end;
  inherited Destroy;
end;

function TCompressionStream.Read(var Buffer; Count: Longint): Longint;
begin
  raise ECompressionError.CreateRes(@sInvalidStreamOp);
end;

function TCompressionStream.Write(const Buffer; Count: Longint): Longint;
```

```
begin
  FZRec.next_in := @Buffer;
  FZRec.avail_in := Count;
  if FStrm.Position <> FStrmPos then FStrm.Position := FStrmPos;
  while (FZRec.avail_in > 0) do
    begin
      CCheck(deflate(FZRec, 0));
      if FZRec.avail_out = 0 then
        begin
          FStrm.WriteBuffer(FBuffer, sizeof(FBuffer));
          FZRec.next_out := FBuffer;
          FZRec.avail_out := sizeof(FBuffer);
          FStrmPos := FStrm.Position;
          Progress(Self);
        end;
      end;
      Result := Count;
    end;

function TCompressionStream.Seek(Offset: Longint; Origin: Word): Longint;
begin
  if (Offset = 0) and (Origin = soFromCurrent) then
    Result := FZRec.total_in
  else
    raise ECompressionError.CreateRes(@sInvalidStreamOp);
  end;

function TCompressionStream.GetCompressionRate: Single;
begin
  if FZRec.total_in = 0 then
    Result := 0
  else
    Result := (1.0 - (FZRec.total_out / FZRec.total_in)) * 100.0;
  end;

// TDecompressionStream

constructor TDecompressionStream.Create(Source: TStream);
begin
  inherited Create(Source);
  FZRec.next_in := FBuffer;
  FZRec.avail_in := 0;
  DCheck(inflateInit_(FZRec, zlib_version, sizeof(FZRec)));
end;

destructor TDecompressionStream.Destroy;
begin
  FStrm.Seek(-FZRec.avail_in, 1);
  inflateEnd(FZRec);
  inherited Destroy;
end;

function TDecompressionStream.Read(var Buffer; Count: Longint): Longint;
begin
  FZRec.next_out := @Buffer;
  FZRec.avail_out := Count;
  if FStrm.Position <> FStrmPos then FStrm.Position := FStrmPos;
  while (FZRec.avail_out > 0) do
    begin
      if FZRec.avail_in = 0 then
        begin
          FZRec.avail_in := FStrm.Read(FBuffer, sizeof(FBuffer));
          if FZRec.avail_in = 0 then
            begin
              Result := Count - FZRec.avail_out;
              Exit;
            end;
          FZRec.next_in := FBuffer;
          FStrmPos := FStrm.Position;
          Progress(Self);
        end;
      CCheck(inflate(FZRec, 0));
    end;
  end;
```

```
    Result := Count;
end;

function TDecompressionStream.Write(const Buffer; Count: Longint): Longint;
begin
    raise EDecompressionError.CreateRes(@sInvalidStreamOp);
end;

function TDecompressionStream.Seek(Offset: Longint; Origin: Word): Longint;
var
    I: Integer;
    Buf: array [0..4095] of Char;
begin
    if (Offset = 0) and (Origin = soFromBeginning) then
    begin
        DCheck(inflateReset(FZRec));
        FZRec.next_in := FBuffer;
        FZRec.avail_in := 0;
        FStrm.Position := 0;
        FStrmPos := 0;
    end
    else if ( (Offset >= 0) and (Origin = soFromCurrent)) or
            ( ((Offset - FZRec.total_out) > 0) and (Origin = soFromBeginning)) then
    begin
        if Origin = soFromBeginning then Dec(Offset, FZRec.total_out);
        if Offset > 0 then
        begin
            for I := 1 to Offset div sizeof(Buf) do
                ReadBuffer(Buf, sizeof(Buf));
            ReadBuffer(Buf, Offset mod sizeof(Buf));
        end;
    end
    else
        raise EDecompressionError.CreateRes(@sInvalidStreamOp);
    Result := FZRec.total_out;
end;

end.
```



```
unit ZLibConst;
```

```
interface
```

```
resourcestring
```

```
  sTargetBufferTooSmall = 'ZLib error: target buffer may be too small' ;
```

```
  sInvalidStreamOp = 'Invalid stream operation' ;
```

```
implementation
```

```
end.
```

Overview

=====

This directory contains an update to the ZLib interface unit, distributed by Borland as a Delphi supplemental component.

The original ZLib unit is Copyright (c) 1997,99 Borland Corp., and is based on zlib version 1.0.4. There are a series of bugs and security problems associated with that old zlib version, and we recommend the users to update their ZLib unit.

Summary of modifications

=====

- Improved makefile, adapted to zlib version 1.2.1.
- Some field types from TZStreamRec are changed from Integer to Longint, for consistency with the zlib.h header, and for 64-bit readiness.
- The zlib_version constant is updated.
- The new Z_RLE strategy has its corresponding symbolic constant.
- The allocation and deallocation functions and function types (TAlloc, TFree, zlibAllocMem and zlibFreeMem) are now cdecl, and _malloc and _free are added as C RTL stubs. As a result, the original C sources of zlib can be compiled out of the box, and linked to the ZLib unit.

Suggestions for improvements

=====

Currently, the ZLib unit provides only a limited wrapper around the zlib library, and much of the original zlib functionality is missing. Handling compressed file formats like ZIP/GZIP or PNG cannot be implemented without having this functionality. Applications that handle these formats are either using their own, duplicated code, or not using the ZLib unit at all.

Here are a few suggestions:

- Checksum class wrappers around Adler32() and CRC32(), similar to the Java classes that implement the java.util.zip.Checksum interface.
- The ability to read and write raw deflate streams, without the zlib stream header and trailer. Raw deflate streams are used in the ZIP file format.
- The ability to read and write gzip streams, used in the GZIP file format, and normally produced by the gzip program.
- The ability to select a different compression strategy, useful to PNG and MNG image compression, and to multimedia compression in general. Besides the compression level

```
TCompressionLevel = (clNone, clFastest, clDefault, clMax);
```

which, in fact, could have used the 'z' prefix and avoided TColor-like symbols

```
TCompressionLevel = (zcNone, zcFastest, zcDefault, zcMax);
```

there could be a compression strategy

```
TCompressionStrategy = (zsDefault, zsFiltered, zsHuffmanOnly, zsRle);
```

- ZIP and GZIP stream handling via TStreams.

--
Cosmin Truta <cosmint@cs.ubbcluj.ro>

```
# Makefile for zlib
# For use with Delphi and C++ Builder under Win32
# Updated for zlib 1.2.x by Cosmin Truta

# ----- Borland C++ -----

# This project uses the Delphi (fastcall/register) calling convention:
LOC = -DZEXPORT=__fastcall -DZEXPORTVA=__cdecl

CC = bcc32
LD = bcc32
AR = tlib
# do not use "-pr" in CFLAGS
CFLAGS = -a -d -k- -O2 $(LOC)
LDFLAGS =

# variables
ZLIB_LIB = zlib.lib

OBJ1 = Adler32.obj compress.obj crc32.obj deflate.obj gzio.obj inffast.obj
OBJ2 = inffast.obj inflate.obj inftrees.obj trees.obj uncompress.obj zutil.obj
OBJP1 = +Adler32.obj+compress.obj+crc32.obj+deflate.obj+gzio.obj+inffast.obj
OBJP2 = +inffast.obj+inflate.obj+inftrees.obj+trees.obj+uncompress.obj+zutil.obj

# targets
all: $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
    $(CC) -c $(CFLAGS) *.c

adler32.obj: adler32.c zlib.h zconf.h
compress.obj: compress.c zlib.h zconf.h
crc32.obj: crc32.c zlib.h zconf.h crc32.h
deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h
gzio.obj: gzio.c zutil.h zlib.h zconf.h
inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h
inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h
inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h
trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h
uncompress.obj: uncompress.c zlib.h zconf.h
zutil.obj: zutil.c zutil.h zlib.h zconf.h
example.obj: example.c zlib.h zconf.h
minigzip.obj: minigzip.c zlib.h zconf.h

# For the sake of the old Borland make,
# the command line is cut to fit in the MS-DOS 128 byte limit:
$(ZLIB_LIB): $(OBJ1) $(OBJ2)
    -del $(ZLIB_LIB)
    $(AR) $(ZLIB_LIB) $(OBJP1)
    $(AR) $(ZLIB_LIB) $(OBJP2)

# testing
test: example.exe minigzip.exe
```

```
example
echo hello world | minigzip | minigzip -d

example.exe: example.obj $(ZLIB_LIB)
$(LD) $(LDFLAGS) example.obj $(ZLIB_LIB)

minigzip.exe: minigzip.obj $(ZLIB_LIB)
$(LD) $(LDFLAGS) minigzip.obj $(ZLIB_LIB)

# cleanup
clean:
    -del *.obj
    -del *.exe
    -del *.lib
    -del *.tds
    -del zlib.bak
    -del foo.gz
```

```
ï»¿<?xml version="1.0" encoding="utf-8" ?>
<project name="DotZLib" default="build" basedir="./DotZLib">
  <description>A .Net wrapper library around ZLib1.dll</description>

  <property name="nunit.location" value="c:/program files/NUnit V2.1/bin" />
  <property name="build.root" value="bin" />

  <property name="debug" value="true" />
  <property name="nunit" value="true" />

  <property name="build.folder" value="${build.root}/debug/" if="${debug}" />
  <property name="build.folder" value="${build.root}/release/" unless="${debug}" />

  <target name="clean" description="Remove all generated files">
    <delete dir="${build.root}" failonerror="false" />
  </target>

  <target name="build" description="compiles the source code">
    <mkdir dir="${build.folder}" />
    <csc target="library" output="${build.folder}DotZLib.dll" debug="${debug}"
">
      <references basedir="${nunit.location}">
        <includes if="${nunit}" name="nunit.framework.dll" />
      </references>
      <sources>
        <includes name="*.cs" />
        <excludes name="UnitTests.cs" unless="${nunit}" />
      </sources>
      <arg value="/d:nunit" if="${nunit}" />
    </csc>
  </target>
</project>
```

ITSF^C^@^@^@`^@^@^A^@^@LÚZç^F^D^@^@PÝ^A|ª{Đ^QM-^^

^@ É" æî ^QŸ^A | ^a { Ð^QM-^^

Tuesday June 13, 2006 649/1462

@

651/1462

^N/protfield.gif^AM-^SïtM-^G^R^O/protmethod.gif^AM-^S÷^FM-^G

653/1462

@

655/1462

[illegible]

^@U^@n^@c^@o^@m^@p^@r^@e^@s^@s^@e^@d^@^@^@

```

^@M^@S^@C^@o^@m^@p^@r^@e^@s^@s^@e^@d^@^@^@{^@7^@F^@C^@2^@8^@9^@4^@0^@-^@9^@D^@3^@1^@-^@1
^@1^@D^@0^@eûs^@F^@^@^@^@^@F^@^@^@LZXC^@B^@^@^@B^@^@^@B^@^@^@A^@^@^@^@^@^@B^@^@^@
^@^@^@^@^@^@(^@^@^@eûs^@F^@^@^@^@^@M-^@JÛ^@^@^@^@^@^@M-^@^@^@^@^@^@^@^@^@^@^@^@^@N^@^@
^@^@^@^@r^@S^@^@^@^@^@^@P_^@^@^@^@^@^@z$^@^@^@^@^@^@î1^@^@^@^@^@^@È6^@^@^@^@^@^@4C^@^@^@
^@^@^@^@ðH^@^@^@^@^@^@M-^@JV^@^@^@^@^@^@M-^@H_^@^@^@^@^@^@M-^@Tv^@^@^@^@^@^@T¿^@^@^@^@^@^@C^@
^@^@
^@^@D^@^@¶eLA          ^@^@V^@^@HHA Version 4.74.8702^@^@D^@^@$^@          ^D^@^@^@^@^@^@^@A^@^@^@^@A^@^@^@^@
^@^@^@^@^@^@°EWçM-^@M-^@]Ä^@A^@^@^@^@^@^@^@^@^@B^@^@
^@^@DotZLib.html^@^@C^@^@N^@^@Documentation^@^@F^@^@N^@^@documentation^@^@E^@          ^@MsdnHelp^@^@G^@
^@D^@^@^@Hç

```

659/1462

[illegible]

pÖËâ^EÄ^VM-^Dvc, 'w' ÍM-^TM-^F³%, 'M-^HM-^ZÎ³M-^XÑ%©M-^VÖÜðâ^Y3ÄM-^TZ¾^YgeöçdÝ^2^SÆ>Û
M-^Mð

\$°%-îTÖM-^XM-^YM-^NâfÄM-^WYqM-^C£ù5^Dî^@÷^Zï | ;MM-^B98V^TR¼"ÖÁ:¹ ;TötZxM-^PM~@{kUÄI^RM-^Iup
r[c>'WÄ&

s´^D%5aÄ^\{Vè 8^FM-^_@M-^_à´M-^NÜüØM-^EÆZμçÐ

ó ¼ÃOñA6 ^H^O¹

M-^_H\óü:»M-^LM-^X&@M-^x^\j^P^GðĪÔ!M-^Nñ▯M-^PKM-^^jaM-^_û?Í"îpj\M-^AâÕM-^WM-^SUU&M-^L^Cî

Ä¿Ä|Meİo^VM-^_û

wß6vM-^[^_URP'^CBM-^I^G8 îVàm-^]»

M-^IrM-^HÎ¶^@[b3Öñ^FM-^QoX"^QB6M-^SiðF^CÖM-^MĐnM-^[^2@ M-^] m
đ, BDì^KtM-^W-M-^M^P=:^Oα^â^f^AOQ7EM-^Eh5è]_7<@^aJM-^CM-^Nh^-^3o^Bä#á ±^AO¶}^TÈè|éÉ ^R^Gùm
M-^E¶ÊM-^R^UtÄóÁ^[@eM-^RU^D×^_^_) ^2îwÁX#Î`M-^HÛçαM-^JOM-^W¼Î^R3ÕrdRÛ-^YM-^_Xñ^8Q^YM-^]/
M-^M/M-^Tx^W^OlM-^F\$KãĚ%R^E4\$á¶^Sãm-^D!IhM-^@M-^OM-^M@æV"9%-5.K^B*2-öM^FoM-^M5 >-^A?{>
M-^\\éÛaý¿æÂM-^_f^_p^Dó`Îà^Oâm-^EĚè?ü>`;z('1M-^GûĂ÷^@¶ÎÂM-^G^]M-^MĂ^r
ÔnÓÂµñ^N\ÝM-^EÄNo^3: %ý^•^DĚ`úaôY)M-^_`Õó^R^_OM-^T`É2Ě%U•ÍÇMMo)ö^]HdKç^Q ÓM-^E
«ÅZ)àm-^J|^PIăIdĐM-^S\$M-^YÀ|Ă@^G-^Q`a^C:M-^M^C! ^GíÍM-^R^oĂ¶÷±^VÍgđV^Bç%Ö0]Æ)óM-^SÛ
-M-^XM-^Y^@M-^Z1^UZ^AÓ1Z^o^\\;jMR±^C\\û:èD^aé¿dM-^R^]¥OM-^Y»NöŮŊM-^Pxc^^B^YëcűëU.ÊM-^\\1^@©

jâÀÈXA^C^V^W¼^PM-^H^ZøÛÉM-^CÂG&|M-^BM-^FýFpM-^@;)^Cú%úóo\$^NM¾^CM-^AÈ²M-^D³M-^Z^?M-^^ òì3¥
²çÍp@«°^]M-^IÁ-â|évM-^@Aö-^C+ðÛæP9Ö6@¨°}äÐìuM-^@Õ-^NÈ2X(¶§âÝ=)Ăç³Ă-^S}Z^KR-¬èÖGmð7^^M-^P
M-^B^Zz¥^RÛôr¨ý«M-^\\øów"&Xí,ÌM-^P M-^XXÊM_´Ö¼M-^Xon^^B
M-^O^]\$M-^SM-^AM-^SM-^ZâĂM-^_ ^S'M-^PÛø°îM-^ZM-^Mæ§öxmM-^G¨J»MÇM-^ZÒ¶+ªZl6ÆªÖo[½^PöÎ¿ÝÛ^A±
M-^M•M-^W"M-^GâÎelÑ^H¥NnM-^U×X^F^U-M-^^0GJ½,zM-^E£¥û7t§U` }V^O^]M-^^M-^^µSÚ "ásÇçQ
M-^FæyèÇN/,îDM-^KXM-^G/.
M-^AFĂçrQM-^[M-^F^E 4hfD°M-^]M-^K_ ^FNÿæöø³dM-^Bâ^OĂM-^_Cjo!EÛM-^B0|Núhp^[´«V
^PáŮ,M-^QZpxàYM-^TZHÆ
M-^Qám-^[Ó/^DÀ[øÊ/^UM-^K
0bXĂ_4|M-^M^NSiÊ^Wê^PF°¥ĂM-^]HV' ĨM-^MÊM-^PMøDM-^\\4#©Pm½ÊM-^C^\\ŮIM-^IM-^R_çC0Ö)ĂM-^SâªĂPîô
^TEââ`M-^[rR•°RP.ûLbM-^C^R^Dnÿ^T^AÐ/^°ô»mM-^@çM-^Jnê<^Y,
lÔ~n¹M-^[^@^WUµëà7^PwĂ¨¨Z«M-^L;½^[^ [M-^P^WaiýÔî*ð^UM-^EM-^BM-^Jb#z`M-^EE:J P°êEM-^EĂú
M-^DFK+^S²á^F¶Ŏî,§íM-^V°M-^^×³M-^Iµn •^[âM-^Nz•]M-^_CimF:£PmM-^WÖŮª°ízM-^PÖÖ}h§jîM-^MM-^T
M-^IÆÐHM-^Tf,M-^LMM-^QV^DM-^K ^Nx^NK;d^Z^KÖÎ\$¶İăÖQĬM-^@xuĬ2§fM-^TNWòm-^WøÆà^S!xÝÇ^M-^G ā
=QÍ^D(^EM-^Ijq^[×[M-^Q^CĬM-^Bú|' yŮ{M-^QSö| Híēm-^@^SŮ2^HĂH\$!QX

s^S\$yăpM-^PĂ½; äM-^DÎkĂ¿×ă§'®]+M-^T+CZ§8)1]ªqIP^WM-^JM-^OUO n%đM-^@e§øM-^FIÂK5M-^JËđ•
^HM-^CM-^Mgöwo^BÛñ-

M-^^E@^NÖ^[i

â½ÿ´<;æc4e&Ů>M-^]¼M-^F^A-uö^Y^RðF^Y^C´b>
£[jôÑÍ\$B:D|M-^@@)gZMM-^P`ß\$øÉ^^¿¶..#°\$çú¬&M-^^|8¼Î¬¬tM-^UäpTvÄIM-^Mø æÿXM-^Híi^Cf[^[M-^^
5Wî%î^\\,çM-^L0V»ðM-^K^ßl;psPû^F´ðdäöM-^[´ûroL/^H;í^TM-^Z¬vZIGY½t/\$XHoÈL\^R^XM-^\\cæf:ðM-^K
ê´´ÀH{M-^EðXÎ^H^^\àPU¬^CpO^G1>\$n?M-^WM-^VÖléôM-^N^X÷H([fWvrM-^^ðM-^KéØL¹¬z|êM-^Wý
+D]+1ý²/×_)Ů]Á^]ð\îÍM-^WM-^N^NY^Qð,ÊM-^W^]çÅðYó\
M-^@M-^K

«ÉÉD^Wü

UÄWs.M-^_,M-^CÊ¹^Hù7Ôñ¨î&ÄQ\R^Z^ZbO^_eV×M-^L^N½¥J&ÝM-^X

Ů°Ê 6Nr"À|GJ|"^NÖE^XM-^^TIG|Kás8=6l^WÇÂ"ßçv^KÝ
 lî,;^WM-^ZµM-^UKf/D@÷~^Y¬Ñf^AôÛM-^V°Ô+äǝ¿:M-^5
 øH/Õ{ }x'±#" -

^C¼M-^Px^U^F^A>M-^]ç^^^SCµM-^^]ÈNr;7Ñ«°D¥¥"M-^ZOİ- Uò•Ũ=SM-^OM-^Pªð×Ã^KK+´q¼M-^_ŨEô
M-^HM-^L
i},<^-ÀÊPõ^?M-^ZM-^L^\\^K•^S~Fâ1^TªcêßåóS×M-^W^VvÀeáQl>M-^RAíM-^^İ{O^_µ@r@0İ÷ĚHDÎM-^Q.ä\æö;
Ç^]*,S¿^^tVÀE^@æbCø0P|ĂÏp2Æ' {M-^Z]æ>Ă)-gª8ñ.^?øi/r^KqT¿ÂM-^\\.^B^SnðŒD)'fÿĚŨŨxc_rú^A÷{¿2*
M-^Iă±¼g"á"u^A•M-^LÂ^Da`{pâM-^Yi¥'óİ-xµJM-^H»Óà³2;İHŨ

M-^I^VpáU;^?ee¼fμŪM-^AŸM-^LkT]K¿\$M-^DM-^ZM-^NÝî%M-^ZM-^YÝpEÑM-^RòM-^T^Uü(ŪÀX^]î,[zM-^Oo^P
^Yiz°°,M-^@ñĐ'Ô))ö7^B²âWÓ²gÊí-÷«^Xö<^HöMNM-^_Ÿz¼<B»lěňôBÊû°LŸl'fääM-^U¿wrëM-^Dĩzú©©-
E6{´
^CI°°^Nß hM-^@ &ËJ^Y^?M-^@\$l0¶M-^]M-^@Do4×^NéĤ))1BM-^ZlæÆ-+\${AîP.v]M-^M¼M-^PM-^MMî/Ë3+T`_
A
g-¤d%|^\\ûWx M-^^:cM-^E1p-Éa>8©M-^RWð/M-^KM-^@«
M-^TboøM-^YÓYİı^]^[~æ^VuðM-^Sı^PM-^HµM-^Sê^OŪA^K-M-^UmM-^PÊ^CÉM>Ă'ÇnYê^OëµİæÅM-^O"Å-M-^\
:WM-^WÖĤ÷uM-^W |AŸ½M-^Wµ±^Væ%M-^Oô^S ^_4üt;Wôâ^H÷÷ĂŸè>^*^?{î¹^?Ū´^O| Ă^^+ù5Ū>²^P^Tó,°
M-^D6=M-^LBPRØM-^K^^ŪM^[ËĂG-MŪ^N^DEM-^H\$ /M-^XZ+xá6a^[ě^U°M-^Qù^V

{M-^@[î^^°/M-^DË=^YM-^AîfUô^BrçgM-^D Ó| w•M-^[^^Uú¼{M-^GM-^KÉOp9fμ•bBâ, _øRãEnpwIM-^Zdjÿ]
+TlM-^AM-^A^OM-^L{M-^M•½ÍÃ^\\-XM-^A%ä3;kÓTp^F^\\ôPg6í;Pè^P4»Ó:~\\ 7Ø"êd7öûÿÃ7-^Bó-äM-^R½A|Ã
Q^QM-^WM-^K]qr>=^S¥²?^S^VÑOã)M-^UM-^\\M-^IM-^PÀ'^^]ÈM-^ZÃμ+^KTM-^GÛ4Rü(£îîã W^FÇYÈ@^YÿöÛ@G
B^-°C^E}ö m^DM-^RëOM-^F

M^^;^M-^BxXdû°ÍTn^_M-^V-^?1Zì^_W8M-^LPĚ_afÆéÓ=«^W2neÁrGÑCM-^Rw*Ê fm-^BãM-^AiÛ"¼\i(
M-^K«÷ aãWððêFM-^X=ÿM-^J2ðû^YM-^LEM-^^àm-^Y;ôM-^PNM-^JM-^Lp^Uâa+iJÊOÀM-^XfM-^I-Đđ;¶@Éª-]
^_GM-^E[^Fv{|^]ÛM-^Nï´Y^Wú|.³%u^Xüi^PM-^@¹1ĐA\o!QA³^Z^YpM-^Yð!^GhðØĚĂM-^UJ×Jç^K^WÝă]çtĂø0
fÓ°M-^VpysJBăcM-^_H^C´æM-^S-M-^Pq
å½ónp^DM-^It^KĂHM-^M3^@^ZI5M-^C3®»GĩM-^_~rM-^S´M-^Aî"M-^BîM-^KM-^WM-^IÈBC)sĩ^Hù)

"mM-^R^QĂM-^HĖi2û,Ön^@^QfF^K6M-^E:M-^Y\$M-^B5b6<áLc#rî}7^XXM-^Bîđí_,l@§^YM-^E.ŮUM-^M-^PèĂ
xăí:d^^M-^L^°^S+ó
M-^_{M-^Pđ
Ńm7ĂĂ^A [³ø1^P•M-^Fđ^_ĂTŃ^Pî^[aé^PŮM-^[Öfđ@ûýM-^HM-^Tz
Ócα^C<eéM-^FéH5tC^DQfM-^J
DM-^_|Ă<@Ăo¶,ŮM-^O^TaĐI^NM-^MjĂ3WcM-^N(^_QM-^[Ů,dç^FÝM-^]^Q«É*yŮnđ
wè•|Ă:ă}^W|Éu|m-IE¼^UM-^YM-^H^B
i|B`1f½IlĂĖelM-^UEîŃŸ•^P^FM-^ZîŸαĐjĂđQ¶X^Q¿TM-^Uì¹M-^Rèđ^FM-^\\n,JM-^W@^_@•¹^V^K^QNâ|
^U¹ĖM-^QM-^_^]æ!%ŸčÈ^SçæİŸë6r5^O^]îđ²p[M-^X¶Ń^Zi 9ÍM-^M¶q^C^FM-^Zh´^F^Z7Ōh¼Ń^Ză¿fxŎ,ĖŸ;`
M-^A|h ü^[M-^VßĂĖê^D?năWOÇcM-^BŸæáµαOM-^R:çèM-^O-³
A^YM-^Er^Gă^BŮ`~?7ă«uŎ¼=M-^AM-^Z}đ Ÿ:¶M-^K^Al^PĐăà2DHW^SM-^FM-^Eî^GTP^WÍÇĂx^GpM-^C
M-^EİĖŸ&âx^Gìo=zđ-7³;^X^Ė3m^RzfFă7îR#M-^VîGđ{jîM-^WŸ
%Q^O: 7íWso¼ŮĖM-^Vú_{^_M-^_Q_^X)RăĐ\øê+LøM-^RĂ^@M-^ZWM-^P!ÎŎ´èŃŮ^R ,^KŎ5{|M-^Iù^U^Tß
5^D½ŮĖŮM-^L 8/đ*³\$ăăŮKŎ^ă½M-^KM-^_¶ĖZ^Gđ^¼@m^F^°^VŮVM-^Gf7^SpM-^S=^^M-^@6÷¹wŮçM-^Fç\$^\\ST
QT^ZĖM-^Răw_1^°^RŃŮEHpiŮRç^VRðM-^HKİđđ^ZçαéİŸ½^a½qM-^G
"Ėgrzfo (^T
Ů^R
ŃĖLŮ^TŮM-^MŮM-^AyM-^L~pŸăC•M-^FM-^JoM-^X=rM-^REhđĂ"zçM-^O>PPFíŎd@{í±%mRMM-^A+êû|^BðPM-^CŸ
M-^LăP^M-^^0ăj#T^@Ŏ^S+»^GM-^U?e^Z^]-~?^ŮðM-^CzĖ^ZúXŮ(q[ăă^Oa.{³M-^\\óĖ^EiNíæéM-^@*ŸWb)`:&^
0Ŏă#^P6
Đ
Ă^P8M-^S±M-^C@
^PM-^IV^D>M-^HM-^LTİM-^JM-^R½ĂĖL^S^M-^U¶^M-^[u´´?^_^^ĖNQM-^@nf M-^\\!A:^^M-^ZbL^gRM-^TM-^H
J|M-^Jđ^]îçQ÷dM-^WM-^@`TĖ^XŎ|ĖĖöY|M-^YĖ[²ùJđ@qZoM-^ZM-^^^NĂùI`-ŮŮ>\$ĂM-^S²^\\q^QD<X^N^YA²6q
"Ŏα^C^W>|Ů¶@^HđăiĖ;M-^^\$ZM-^Y:«f^]ăM-^KĐM-^OŎM-^IŎf^:°[Dl^Sù¶Ŏ^H
M-^I^[Ů^PĐM-^[^A«M-^L^]
Q¿É,LŮM-^@ĖT(L^ă[Ă^[sM-^H[M-^EG¹İĖM-^N-ßİm¿ø½^]~{M-^LM-^E^D_@^BŃ¼ĂĂM-^QQ^F(ăŎÇĐĂĐŸM-^@^P
M-^ZaM-^A^FŸ¿»M-^^^A^\\M-^MŸ"^Qpă1X^P^QĂtĖM-^CM-^^GO^B_*4öI)ĖM-^XŎM-^@HpM-^QăŸM-^H,çRø>BD
ĖŸ>zŮM-^MM-^C@đ^R*Ÿ,í^Hİđ@Ax\ŃĖZ^G-PA-^VĂŎ ĖM-^U^KM-^C_AM-^C2÷¿ăM-^OM-^AM-^OR^ZQ^KM-^G¼^\$
Ė:~^ó-^Q./8^V¼M-^Q^H<^]ĂŸαİ
?*#^ZăÇĂ5P(^^^U6
DM-^ToŮNŸ^NŎFĖ¶ŎQ8<","ŮM-^UŸ" ^TM-^Iă^^^^?M-^LhM-^^M-^E#^[^KGøM-^X ^04.^X½M-^D ^_
M-^H7ĂŃM-^Me^Hă&\ŎM-^G^UM-^C^KaßLă>`M-^@D±^Rµ÷^O(a

¶YvèM-^J@M-^SM-^Z(^3^R}D&^P6ïM-^T^3^û×^RÿàÀÖÄö&æØe^\vM-^Z/ôM-^Y^]Z^2M-^QèEM-^D^Tç-<M-^_pKf^A
?^?öþ@^Z7\$äÑKM-^Z.¼»M-^LH>" ^RÛM-^V^3v@M-^@ø^Z^^D^Tò^B%M-^BfâM-^VÔM-^GÀ\$M-^I~Ã@èy^P@CÝ
Ûþ9^B`)îÐx@þ
^\\pU(p!ËRM-^AÅhî68ü^P^P^B^@^@^@3^C@3^@^@oeLM-^PÚ^_AqM-^A^D^2nØÆ,1YèM-^N~A¶M-^PM-^DÆM-^H
M-^PÛvÛ¶Û|î^3¶îM-^RfGóM-^Hâ^O^D^W×^PÖ^@^@^Y^Y,ç^C^@è^C^_H`^M-^PM-^Z^H^A+^A^B>À-Âr^UCoM-^@]
ë´ÍÂvî@-XÖv@»wM-^Kjmk÷¼--îÝ»Û-µññM-^QM-^W^aÛM-^Nñ•Ç^a»ÆM-^J/<ÔçZmxM-^IWgnM-^LM-^E^B¾ð±M-^G@
^Dâû^@!^@^@Q1^C£0M-^@¾?þûîöÝîÿÝS•ÝM-^LeY^3^Wßñ¶ÖkíÖiT@M-^]M-^TM3^P-[M-^[tfM-^P``*4^HLç
M-^T^GM-^^xM-^^ M-^BM-^@^BM-^B@^HM-^J2^B M-^@?M-^@M-^C^@M-^Oð?Â ^XM-^K^_M-^\\ ¥ÈM-^
^2^YM-^Iñs)ÛBM-^V5ÍM-^[^Y4M-^TM-^F^[£cv``7øâ'êÛM-^NM-^QM-^I×fnjó&ä^]^ZQ3^RÊ.±Y^OS«^YÃM-^Ngf
@?A»C=yM-^[M-^F-y£Zzß E-°4ejM-^JM-^CM-^YÊ

² " ĚM-^RZM-^AĚĚĚM-^V½^?M-^E~ŮóÂtMPăàM-^P^Vv@;e^G1M-^WM-^T½^KsM-^[Ě; îŮ^? šŸ@;\M-^E{M-^OM-^X
#Dù
M-^TR¹ăébμŸm°M-^KÁĵŮŇ^S½M-^JM-^îiŮCM-^ô: , ^MÀî□JđÓ«xM-^MöŸ: ^X°M-^M-^W_ě{M-^Z^KM
,
mîvZ
ZM-^[OĚŮM-^M9M-^XM-^[02{ òĚŮ. ^E-ÂÂ@M-^gT•öê"é-Ÿ^EđM-^iěěJÁĀŸ
[ùK9YvM-^_«<' ^SM-^E; (<qbĵ®Ů«W¼êéš²M-^B^CÔ°B@qzDA ^CP^RM-^QŇ}M-^_M-^@Cx`YM-^HCÔ^XKE
8y/vM-^Q^Yr^U²đM-^XŮĂM-^_ ^NM-^G^] IŇM-^[M-^WM-^IeĚ#; ^?M-^M-^AM-^J^RŸZ | p&Pw½M-^Cû[äM-^Es1¶
ßM-^[Z_űđM-^C<æG

M-^WtİáİPô¼Å!§cİİİİPİ^HâM-^VM-^BcM-^I¼^?ßM-^L^HÛ.Pİp¹´4Ú7ìà4°ØÅ±;Í¨&^ET^V^V|cÊ|^-Ô¶+δÛÛK'VD
NM-^S:KAM-^GðVM-^KÈabM-^Km/Åb+Ê; ;lH^B!M-^HkM-^QÑªÈ'M-^OðÐ^QUáfæM-^MO>xc¹ÔM-^V-cxvM-^Xİ[ÓO
IÁpÖ«:ÃxK^GââM-^V)Þ è^@^]íM-^BþM-^NÀ
¥çN^-ègt'µ^K»M-^Jã÷ZîM-^Að^^øvvòXæM-^^3tb¿,²àh¼TöVÔM-^Jëm-^Z%M-^æè@•¾
hM-^S^R^OM-^F8»h>\$µ½^_Û-Ðİf¶İâM'^ZM-^A'1±Uèð)M-^S^[+çfíM-^TM-^Bvb^RĀQâ^M-^O^RM-^Q!M-^V^ZM
Àü^QÛbc\M-^@^H^S^Xsó^V^OvM-^S¼÷øú
¾D'M-^O27àg~y5/Æ^XG3vÛ#ÆÈS×Û»k7¥¥=FfM-^MjM-^L
*M-^JM-^H~qþİ-#ç½zÂM-^]M-^IfÛV^_N^O6ûú;g^XĀĀ^DAÿòİðáZ±°h+3gé9«è1©3'ÞÛAİă]¼8F[Â;Kie×ø¼^N
-M-^XÛ^RM-^E Â¿Ç^K'M-^H's Û1^ZKÐxM-^AMM-^ÐĀİİİ?Ā^TÛÐj!¬@^?FM-^@M-^K*±,M-^Ný1Ā°M-^A

=B'E7dÈĀ#øt÷½2^@ðă:3=' \$^QL_à¿MÇÕăĀ^]ăăĕİ•M-^S^WýM-^PvM-^Yð;é£j_ úê?;M-^^Q² ò•^X^X1I.Û<~Q"
çW^_µM:ÿ

^T^QÓê508êM-^Tg j÷3d èk| ĩM-^]O^ø^Q^] Āăçe^NM-^Q^_NŮ•éC¬İM-^JUM-^MN%M-^Gòl¹¬Ú(ĩSÂM-^C^K
)©ı^N^YpùûhŮ^DM-^Gì^]ç^FŸ
ÝDŸM-^@^^M-^^}ăVM-^J^Vj/^@æqM-^NÓĐ÷4Afu6^QFÍ\^_AM-^BçÔĂ/mç| cM-^VM-^Dw
9/ù^ÍŌăĂM-^_@Æ[M-^_/Ě³
â^GñM-^J<ı2Yi;lŸM-^@^Eð&³^ZĐ"M-^MM-^F]M-^_ø»^W±¿^PĤăAç9Ůú``¶Q°đŮi1<ă^YŮM-^PM-^G@ÑIhÈ2^_ÔîĐ
sô¹p[©M-^G?lM-^R°M-^M; ,M-^DşüÔÆîĚ^Z¼^aM-^Y^KÎşîŸ^W~¶<RLĂEŮĚâ^S"``M-^R^S ^AM-^@Æ^T*ŮM-^G{Ě^P
^Y`Ă•>GĬafă;ŌhM-^BŮ^_L\$^OÊ»ăöûĐç-E³qf"^^]M-^P6eM-^O÷İ-`FM-^IûŮ^X=áM-^_±gGw^O^^M-^X°[P1Ĭ
M-^[á^RiæñRøbiĂçM-^ZG^QĂ¼úM-^FM-^K<ñQ»óM-^E%Rs8ĂcM-^NbĂM-^IKh¶-đçé

^O8H⁻ÈëÇm/ôM-^MmÇ

ÅM-^BaXZÜé" M-^VF^GM-^E¹M-^TÀÍM-^XK÷c×f\$Ôç»V!^_Eú' ; ^\ \$ç) 8M-^Vu½àóR©9Ró0¥ | M-^XM-^G1YîM-^HgÃ
 QM-^Lä^A[CM-^D^\U^UÇÃ^Cò" zM-^N^•a5M-^YbNè#ÑÔ^VWyÑÃ\ / ^TFäM-^RbV¼Æw5\$þM-^@ö* " ^Wä' cM-^SêM-^H
 j>" ? " 0^B=αŪòa3M^CP£ÃÃà^KÚM-^A~EpM-^Ãô

ç ,pM-^BTÖ " ' ĚP«â) ^N¶Fz©@¾ =^KοÇÂ^TM-^K^RP7âGFM-^TM-^IúFX} {¾^H^Bü^Ua

M-^B½^\\$M-^Z¥¼,M-^PM-^Rö_ÔM-^HÚĬ^DY=!x=hWýM-^IÀvâG|M-^^^NmiIf{°ê5¼¤^C&,æLYc-2ÇM-^WYQM-^G
M-^KlM-^J-TM-^Q4fn^@YOopM-^NL5M-^SM-^PO`^DÄ}eĚ^DPkM-^PñS!^F&êM-^X|M-^F.e%%^_\-i2(9sÇö@
M-^Fð%^^óè?`Ě-1M-^DÜM-^A\$¼M-^S^^^U1âô+â*M-^SĚ²ĤM-^WĬi\$Óø'M-^Sđ!lÇæM-^BßsøB5ø²Ě_BôjÛ£8â!iµ
ÍE½ØÎL1ÁáM-^IÆ"Phöà¼äµ^UÇÆ
g4B^@ îM-^S¤û¿Â-M-^G¼M-^W^X^FM-^L,Ó \$/^Vyá¤M-^VR^Aä®!M-^V«^C,ìü7

y6M-^I. ; x^XÜÑ^QBM-^_] ö-ô¹bK¶K^VKÇùM-^MÆ^S "M-^W-ð^QXB^R^F | ^ZbÐâøM-^MM-^^^_M-^_Ê

“Ç^QM-^X^E^E^v|M-^KM-^VÜzçm-¼e:M-^Ix9÷M-^Flİt^Ctç”~İ Ê^NÀeDM-^D
 NH:^NαhM-^LÇ-”M-^V"ÿÇôM-^Ci#

^OM-^LògãK¹^@â^HÎ^BM-^BW»M-^Q£M-^Jă?=JGS^RM-^]îÛüéÜ•M-^OýEß§M-^XM-^F;ùhM-^L©M-^Ll<áμç´
M-^@^Q-^CMM-^Së dx -^Nà%CDM-^]R5*ûKM-^T/(^P^K^@t^@^U^@^A^@^@ ^Z^@^HM-^@^\\êc*ê□°M-^A^@
@^A^@ÆúøSM-^Dü.ÀNM-^CFðnA\$u/£»E.}3n^HíM-^PM-^A/RjQ>öÜlÓcä^Gâ\$^K[Uw^E-^G0^[_¿^H^KM-^CWðR!O
^V'ÖÖ^Z"Ûp.j,Đª§5÷¹M-^\\[L³8ÅM-^_P&M-^D<mçZ'Ê£Ã
:M-^Vh<ZÊöM-^UÚ¼PM-^J±l¿â¶M-^KÕx7*M-^6ð³M-^]ªÍ°9[àxø¼^On^WrM-^MÉãÍ^V/ý^UFÉM-^YÓÀ'ì(3â
M-^I@=/&èÎ n°^DM-^_Øw âîM-^T}ú^HJC/Í•^uRM-^T^X<>¡#»M-^TL •8L¹efM-^[}M-^TÚvM-^FM-^D
^]\\M-^Wr»M-^P^U^M-^E^Sg]ßö-CàÖM-^JM-^QM-^UM-^H
'2^A-ð}c-ð^SEM-^]¡pîtýÊ°ÕA½^H^Z^RcÄ%5é¥#ÐAxpÑ
M-^IH=I#'ß^P BÜ-^P&

tM-^LM-^G/ěÅkÂâPzM-^HAd£

¬M-^Láê^FM-^A^DT¬43^[óPrbzýzBM-^UEâM-^X#M-^^âyZöÂûûfi^K^?PÚ7z•^Ttnİ=g^^8~Nñc•b&:M-^GM-^@ö
M-^KM-^F?M-^I^So¿}P¹0ð³MJ^[âoBpRÈðİ-OÛ<mM-^F»æÆEİpM-^S}8¿^_©M-^]t)^^ê¼çÿ×;ıç<j7İç/İc(»i*
İM^T^]¹F=M-^HcÔ^ZVL{gÀû^\&_KouÁ{QİûyM-^[M-^\\$@İ3kL<M-^YM-^SÖÓ,|w|ÿö(^b~¹lM-^Hâ%
GÚK8.3ĂQ1\$M-^_M-^RL-|ÓŮ!ûM^ă¿

^_Q^-;0½^K~,ö/M-^C

JM-^NÝ^V^Tn-ÖÍ§Yÿ\$BüDwM-^@^VM-^REÿ\$å«ì-G#^O?xZ :ÊÈ^Oh,ÔäM-^CN^H&ÕOèM-^MÃ3;eQ,>F?²À^N¶^@§i
Á@8)&•wM-^\\^Y^X0R^R4wM-^@M-^DM-^C%M-^RİðÕz^@{

M-^RÂM-^B1M-^Wôéâ^T~kª6Í'ï^T²M-^\\ç•TÊ^T*/^]^ZwÇ^X^@çnM-^]Å\$M-^Mª\\^ES¶W}±Đ&íØD^\\^]^SM-^R|D

^WÝM-^BC^@´M-^Nè¥^T4ø^B7^[M-^P§È"úM-^R úi^\\M-^Xr

-^^Ü"VM-^A*P±^KP1M-^F6^K; }^R}S,M-^HOÿÖÖM-^Qn^B^E^@αyâM-^Wë•

M-^Bİ@" İ^@^^96"M-^]çHùä^KM-^/*•M-^K¼î"ñK°áVWdwPÿ^W/^ÅYE;¿LIÝ
M-^Bı¶M-^[^_úM-^RM-^E|
òOoòM-^H71M-^]NQ^F,^q•xD[s+eiæÝV^^Û=é`;M-^L³V|bïaî«¬µþM-^@Í

^@|^GÖ;è°e»êN{] # (G+ýàWM-^[îýwU£M-^G^W¼"o&Uù³M-^E^N÷[ç £×æ3M-^L^Eqbb\O^BQ^T2^M-^Uμ-Á^?
M-^\\Û@M-^MXÈM-^\"«M-^C7ÊUM-^KM-^GÑ v^EM-^^ ^?V`ÕHúk`M-^PM-^Ga@`)ÕÛøÅ` [ðåM-^@P4û7Úóñ6^F¼^X
B^PäsE^A^HûM-^[Qf5æ0nμÐ'5G±M-^]^V-íGμBĭâM-^^^G^XAÖË^ZC';Á:pM-^HrŮQM-^CŮĬÀ|R[s,`ç9`Gf[Dü
M-^MS}GM-^]Ç^ECØŮ,çl#Ů8M-^JM-^PRt1 ^]^GbQkdê
°Ñ|μM-^KzM-^P+!BM-^U[Ø½%ª[RM-^I|*
*¿#î=é,ÂØ M-^G<M-^JÊ^GâÐF>qJ«ñññ-zM-^RSM-^TpυμUE@¿Ð¹çËjM-^Pÿ

A
^ZM-^WM-^J^FYâ^ÛâGαÎM-^H " ^B^G | û@PY^TU^N^R0ĤM-^Jø^T^7nM-^EmM-^J*gÛM-^JJM-^I^Zé

ªçû¿^QUM-^GQ~^H&ï¿M-^@MÎXM-^YÖ_M-^U^H;M-^CM-^X^T^BM-^U^N¾é¬á[M-^FUbP#GMM-^X'M-^YÉ'2:-HLq
WYiü^VM-^WQ ÔïlÛ¹^VM-^NíçCzM-^CkM-^KÚ«ÑK^T¶]M-^I~^HqÀ'nM-^DtA/ñWäBöók%%-á(ÛM-^Vàæ" FðÄET
M-^WM-^XH'M-^BæÍ^Dª^M-^FmïDW^Q'Té0^Hè^X5_~^Up!\$tªzÓ°BdC&
M-^S .M-^_w¾&°JE^^bxÚ,^AE^@d<^EÄxLM-^P« ò¿÷-M-^R^Sää,ÄwU-egzi ^B^_'^F&gäMM-^BM-^Iú

@KZr^^M-^X^V•ÚO|_ ,.^_@Îcÿ}Bù`+9ÎôïÛî5N^Bp^-+G8^FéøP

þ£^Us\$îM-^^*Kü´M-^L^U0-@b@M-^A^T4{´^W"gp^M-^\\l;©^ZâB%îífoM-^HM-^W(^UPÿ«æ9M-^Z3°M-^E.?ÆxÇ
M-^HŮ^?/§÷O"XX«Ç¶vu-©¹+Hó´•M-^C´usèÄ8å« Eðû>^Uİ«M-^KÖ^NM-^W^CP³M-^XomlÂTzßFøÍL•ERp{÷M-^D
c1[°QYç^Z•2l^K°M-^AÖxLæÈdhibí^GM-^B@M-^ZT¼-M-^DTáªM-^D÷8FÃ;M-^H~Ã¾HM-^E`M-^Q•-Eä^HÖM-^P&Ôô
M-^Mj}M-^Nne½^K^Qß²Ý-7Ö/ßM-^]M-^DM-^K^?:?M-^Eà×vÝ;§^SùúŮM-^[«pÉ°~lþoh^CM-^QM-^E×^@jÃã3Đ
^Uæ2l7^Ué M-^\\<å#ßn^AŮÖ|âM-^G,öĚqf^Qè^]^-M-^[Ě}²M-^U)đO©ÓE«Ñß4´(ØĚ^AàÖxªÔJ^KŮ°ç´Q8<M-^@*İ
^]°ÂĚ^Z
^XM-^PŭM-^CùM-^SŭM-^ZèyM-^KM-^G□M-^X8M-^C4îM-^ULÊ^WxMª[\$Dî^_M-^YÇĚ^Q1M-^QĂ

21Nmó#.E,þM-^S•&RN4Î•îBþ-ÊghM-^[5ÿEM-^FM-^GðÆM-^Dyα²GÝ<ðw\$ø8^?ÈM-^XÑŸ!apÄ{HªiM-^MÔL6v<8JM
HjfxæM-^Fw@9b^RM-^@=.VEq.Î^F^^Û%jM-^Cô{çM-^Só¼&ÂÛµ
î0`M-^RC^B¾Æ#7,T :^]û-^D#þéPİBû^GÝ7-^Ha\íäø-Ÿ!öM-^E.µM-^RäGACİQOR×þ^RøªM-^Q^VK^ZmM-^P EfŸ
M-^@|Q]OŸ Enq^[Û9`ÛF|ZzöjαM-^\

ò"ë0WQrÛÄM-^OöbÚJä\$^DαXêM-^], IM-^J^S|ßHR`yŷgM-^OM-^@ö\$zPÛÇ>M-^UM-^Q'ÁÉ+&nZk^[ÖÖ-û8M-^LaÄ«
akÄË^[ÖL½^2^«IÄ%`Ä;M-^\01ŷûø0}^^Eαé,^PM-^P3`
^^mM-^RM-^BiÄ+\$b8^E M-^[iÆD°ŷV¶ÛÔ^HM-^RQÇSâ¿^F~M-^FM-^J`Ç8M-^T^A^QÐ^E@UzĤ
'LñM-^Z^F-bá/^_,3ß´^TäÑdUM-^GM-^E^Z^T)M-^X^PM-^R^EC@IaM-^T^Oêb%M-^^M-^R•GL~M-^HfB2 /M-^TÚ
vÆ^^V^B/ÛPå´òB¬ÊcM-^X[çòz\$PαÇU^B*Ä^V(¬,^T^U6°M-^RdM-^JÎM-^OO\$:J^K^T^@\$Ä

P^@M-^@^Cvú^@@^P^P^B^@^@^3c@3^@^@OeùĐŸM-^C(N° Ö,ÛHG^D^K]±%)ÈM-^E%4¶M-^D4¶C¶Ÿ;ăİv°»M-^XÛ
 á¼çø^C^AÅÖM-^DŸ^@^@daëh^@^@lâ^_ ^B^BiM-^D^ZB ^GM-^A^Pðø[-^N4\$@¶UËÄzv•ù-g7PúnîM-^ZŸ»Ûo-
 xrÛ=} .Ê²Ûññ-'M-^V²^]|ß•mîÆÇ•¼Ÿk.-âM-^F-î¶ø^DM-^EM-^S¼[,^_ ^@^O;K^\\ñ^@^D^@hfà ^O

{ i1½öiÝi³Ý^?{ |[M-^QÍ^]©izÛë×µ-mŎÍ(CÚÄ]JA^HÔ^ZÛ¼'-KF I B^H^X^Rô\$M-^H^HX
E%~^AM-^@q^A,~^@^B!ó+^BM-^HM-^@^@M-^Z~³l@yǎ`M-^I^X;P^Z^SSvM-^F]í;^PiYíYM-^H*LsĚµWŎ^YÄ11sx
ÛQ1^T^Z.3@éÖx^PM-^C#ß; ;¥M-^XM-^Y]|ç^TÊ°+l"vVM-^GM-^VSî×ǎçc7XÓü

-Dj"ë)M-^H+M-^UİM-^B-^Yÿ^Z3Â2^C
d}YŪM-^[\å2-zo35M-^Gñ°`M-^_•°^NzÂYt&ñ%ÇM-^Jðæ^]M-^M¿[ê^R5|y,pÂîÆ"ohM-^@ûô¿
-bM-^^\WÄ¶AA¾,İ^Ôí^Psm-^IM-^K¿M-^[IUM-^J`Xñk³ª7^\r^Zf
^P¾E«-^[^Dë|i4iM-^D±Ū}9^Q¼Y^YM-^B¶¶!^K[úM-^RE,oxÂ\$ {M-^MJfç´ç^CÔFb!ŪÔ4'şRÉúøì¥M-^]_ ^FŪ-{æ
°#)\p ŪrĐ44è^FXU^O^Ob*0±İe^_ ^C,^@=¶M-^MçeM-^D^Y^O=M-^S4
/A<çnĤM-^]ÿÂWêÂäM-^D+µaç^TÒñàgŁđ#W2´%zM-^^\`\
Đë^C^N^Kİóò-4îr^VmÝ?t|¶:¥ú±^V"M-^Kn>ÂæİÓ#^Z¹İuÖÇîK/ÃëM-^U6P^[j^^³İ=•\\$_@ELM-^Aò
owßİuìú¤UM-^YW^V¼«İ |eÉ|èÔª^H^S,-.¬: ^Bé^Xðf-Ô=M-^JM-^ZM-^TK^Wei^Çù°Ã~ú\$Uc°^O-p}ŪĂ@½;Ū8^Qñ
M-^M-^^^WDÍiT

]M-^DÍùüÃµµp^Q^[M-^Z^Nêaça-ü-^\\ÄBz²Đ'M-^PM-^Nöý!inéM-^VHjO; 'S^H^TÚhýo¼j^\\

õ^]

îGÖïðnM-^]ÖÇ^Re^V)/øM-^S5Ø*ÅÆ^UM-^M-^KM-^KÅüîµ<^XÇ;¶^PÊ}l^OÝÐÖ,jþ)

ĐýM-^@M-^Ml.M-^Bă>¶ëM-^Gá1M-^Kjµ^^^M-^FÚmÛwF.±M-^A^@#M-^HÖ@M-^HÉDZĤ|3[\$tH²^^H qĕ^CFM-^B¼ĩ
:Ô³F}^NMc^HNAă(GH¨zM-^HµĐ^GĒLø^NMü¬^UÚĒ^[±ĤUÔ ^X÷ÖOM-^@M-^F8§3ÆĪ(%0j#r ^TM-^IÄM-^Z(,¶^Y^_
±i^TR~6~M-^Vt^DÕ´àYñ¹M-^Bé^Yvj¹í´Ē³aM-^ZŸßÖIŸLŸÛü^Y|M-^@÷Q^HM-^SϱOM-^Eó0µK¼q&0ó8¨^PözQ*Ů
M-^^M-^BM-^@{Ů^F^U|đí^[µ4N^H^N^Scù^^ñß¼pĥ M-^½M-^Yă,8_M-^V^^xM-^IÆ^^^Nú^Zĩ^[íÊk^³6
M-^HnM-^C;ă½ŮM-^Y^Xv

p9Q^Y«M-^IÑ^Y^E)'§úÊÑÚê¶^A^?ö{wM-^¿îÿSñM-^Wip^^íª#àséá&©'-;^Tî»SQ^HPM-^Xê^Zß17Êkßu*BM-^CM-^NÂM-^MðM-^_MBt0M-^Y\,ãø]lÛ«^TÝÆ|M-^QÆ2»ÝÞ^EÀ{ígw^Så^T*çM-^Yç°ÔÂ[E-0³^K-•|T/\M-^KvÔj^Wç
afdE^Eì-B1M-^Q^H'LªAìÆ6#eM-^Njâ,^F³}M-^BizM-^^=#ÚpUM-^GàM-^]M-^UªM-^KàCw^WÖÂi-ì]¼×lha
òFvÔkM-^X^S;¥@ÌªÖÛÿryqM-^KGÃ"åM-^OfN(M-^NJµM-^G--^Gé-ëm-^R^TM-^Dw^WµàLQM-^IEM-^O÷
=M-^K^?åM-^D^²T«ÞM-^QØ^RfJ,]~2pM-^NuJyf^W|'A%•BßM-^K,ÂM-^Cp¹^^ÚM-^KôïM-^ClM-^CÂîc/±÷4½:i/
d@?Ò0ò<&~ó

í»<^^M-^WM-^^»uăŮí^ZÛM-^[ÉW£ă9GTCM-^Q^B`jĂ^QŸø½ŮM-^S°6^_Í^^C-7^YM-^@àM-^J¼%JuÓQ^@ĂaCÝ`÷^G
àÔÔÌM-^FŮ×ŸöîĤÆM-^^»}^^N.2#ùÔŮÆ³7^NM-^Y^_^-°;M-^Po^HŮéq^Oμ^VǾfŸM-^]ø¿M-^]1÷M-^Oμ^-a+Ēw
M-^BÇ«@[ǾXQ ^YNIŮ^ÈwhJ^a"\$^O¼EŌz&ñ[òpŭ@M-^KŮ7ĐŮŮ; \$¹h`M-^B[M-^O^R<μM-^@^^ŌI{M-^I^C^})
M-^Iî^VM-^\\!M-^NyB_yĐĂk-æ:J^UM-^JxM-^DíM-^G'¹M-^_UCμ¹Æ÷)vX6×Ă°ĒM-^V+ĒT

Ñ³M-^_3XT°ùÿ}ŸxD /
 ÇQ`/ã?îEï-~w'Ô^OM-^WC^S°i^]µÈðW[Í^DM-^QJÞ^KM-^Að²ÜÊ•øM-^Iqi^FM-^RÔ°^EÛâ ^?¶bÑÓÓÁ8q=\$8^Z_B
 ª:ÏM-^B^OM-^B´ÀÿlM-^TM-^RKÿ^U¼Ñ
 ë;+p ÅÂîÉöM-^H

'M-^B^P^[M-^P^D9AM-^SÈu|^OÛ ðà
Äê4AM-^F;r_Be
°&

Ëù<] o§Ú´7□¶f³□L-^@ÃcÆ^FM-^DÕk8ÓZM-^FnM-^_YØ^T¹M-^LgÝu¹M-^P}Aâ^SM-^KØ¿Â&»\~oÍ~^E^?&³î-_³'
)ùæC±õ÷rSÛ}nSÛ[ÛMvM-^Ki#"Ç±6oäqÇmÃ^A"^^?P3ÿHåÐ;°`M-^Iüð@ü(£^SÐ]İkÕİkèè@úMM-^A(ûã CG5ø^PM-^B
q >9!

^N^H^Q1^@T"

çf`M-^C¬P}^QqöÉÎ^R.þ[~z^KM-^OµnM-^RX^Sj_\$_ù°'g

^TñÉeC9Ã»M-^A\$ÇÁ&

)M-^AM-^Id^O/ĖŮ0M-^Sĩî αM-^@^YĐó^F»³í^Fè0P~]&B EM-^Uĩì'>=

ç\Ä³
M-^UM-^M«)M-^M©M-^F^Am-ç^Pï|M-^JŸdô ÷^[M-^P^T+M-^JûtîgM-^Oêï¹M-^P%TF|Đ¶ÜÑÂ6M-^A1MM-^@
ÅkhM-^[M-^L[P^Q^x^_ß>ç| \$ÂÜÄp°M-^R^Z%uð`¼<M-^RßM-^Jó±| }M-^CoM-^WóµA1! ^Vă&;^BM-^Ao } \B

qM-^MîFoM-^MTÇM-^J dJ_^[^CÛM% ^VM-^Y|Ç)^ZYαhÅ'6("Î6ðM-^T^S'îEqM-^U:^O^-!M-^]M-^ADJÿµ@^RçnÃ
@Djdr&M-^X^-*M-^H^T-ä-Dû@Öÿ|i^X

•7M-^S a^a^eM-^Pf^Xö2ÎM-^B08ø^Zâê

0@0DM-^X*=¥!9^]à7BÛÉ¹.Û´ÆM-^Y´| %İiêY_f8,M-^F-ı| ^Pq2Ó{AM-^J&£(!Ä8Âç^KñM-^C^HcM-^Cı¹=7M-^RÃ
áû
5^Oäö9»ÄVfÄô°=<GM-^] ^VÍ[M-^E,nÄM-^CM-^P M-^WêM-^T½&nM-^CM-^KIW3
ìøföM-^Z,^Eà\M-^@M-^NfÖùÄ_M-^O-^câU©./ÄËÑz2XsM-^OM-^Lð^T^HÛcU+4.ñ: ^Vv WM-^SW
^O 1TC:-M-^R´Û~}M-^GM-^HäòU/Ûa"ÈM-^QÛM-^@JM-^GA³öt^^^GnEM-^H^[İ^X^QÖ^GM-^_öPùFfM-^O}1SA7ú
µËY0^]M-^PpW^^\$M-^ZáJVİp1ÊßİÄß7àM-^F M-^\\,M-^\\«D^T^U^R^B;^OM-^[^K-ÄË^?z^Wÿ^D>M-^Ya\$^W^o^^
^HM-^E´:ü5R40ÔÄûëEM-^İçÖq^KsÄô_M-^FM-^LM-^JµÛáóM-^^Ó^ZQ|û^ân´^YD9M-^CáMM-^U`mWM-^RM-^KÄ^Q
\\ÜdM-^K^\\æM-^M:M-^M^^wsö6½İM-^NHM-^T4ö½^A_ ^F/M-^A4^K\\@ä¹ÄbÛ;ëÜXÿ³ªí²kKñë|M-^PÓM-^QM-^Rl/9^W
M-^^ÇvCÉöM-^U-¾^WM-^Y<M-^S«^K
ÑBM-^G"½öÄ" M-^L^^^Z00#óp•.Mä^Z^NUäÄM-^YM-^K9é^TÄM-^]ð^Dif^DvM-^Cv1M-^_Üóikv|×7óýUë
M-^H`M-^]k^HtÉ|; >-ÜEİ^NİÖÖ= QêüæQM-^A ^\ø¹j^Yð2?Rú-ÿM-^K´^]M-^P^X^GRİÖû,^\\OM-^[M-^HM-^P^Z
?v¶İ-<¼ ÖM-^H^\\ö^YyWÿ^H\$?ñ}M-^]Æ^Bi{^YM-^BD½t^FR^OúdM-^Gi^X.İñM-^AÔ^G\$åDByEM-^Nê^R<^R5Ö´H
M-^[©Æ£Aö 1M-^M7^P^GrÛmİÖö3ã^Z_t^@v±2èMR^C^\\M-^RbM-^^ÆZ^R7ñ<Öê^FÖXeöfPM-^WWp j)¾ßIp[
(==;9Ö^A&^YëkÖöKÖª[\$×|qFö|÷0^@óM-^L/SİËM-^P\$ "M-^C^F °:}óÓ\$^Q@M-^Fİ<ç@G ^[Û^W Òãm-^H¥äeGç
^O^ZçÆqM-^M" M-^T^oQî•^VM-^Dÿ´ÉJM-^[¹JöÿİM-^D
Đ&DI9_İ2M-^J}îm´»5?ZÉC(°ø<pİ_?AM-^\\İlµm•ÊM-^Cđı±¾æa\8vöêöS/DU;nî>!Ü)+sÖ]^BÿİUÜ ^^²^Tsä\$ø
^Yd^]%^C^W9M@^K#ùs nV ,°.zØØÑg^BY5Öİ_ ^O^UÇ3äµ¾^]áL?xê^PM-^MâêİçÖt÷^YLmE¿`Ñ#ÍWhânM-^W^N,^H
Û^@Ä=T^GM-^A^BO?TM-^C^VüA^\\^U±.đ^Y.Öñ^A²îã,²â^Pİ^++t>téM-^GvUjµ6M-^JmM-^OM-^_VbTİ^YÓM-^Uø
ÊfM-^XØ»EzxfRm-^ZOft¿^U«`_²7U4

-^~N&Wbr:çM-^WăuM-^L^XĂn^KÁRê'M-^F'^X)~ÑY8°J^NÊ^H1Ñü±Ö-,~4.^ér×59M-^]3^Oð3

^HM-^F>□ãQFÓDM-^N^Fxiäµÿ×M-^Y-M-^^ zÚ^CM-^L^G½|ÿ W^OM-^E-?ÂÿeM-^V•mÄ3`^+ÃM-^D úí
M-^Y-•^Q\$¼4b¬.ù7©³\$^?-v<.M-^M@aM-^THNò,M-^OçP]Ñè1^S^J>+-cû!|ü^G
gÍ&Pÿ[M-^Z^^PM-^ZM-^Vä
+^^PW°JòrFà}<JÜM-^(^Q÷^A

È<M-^VM-^R^K-ÈM¶¼GM-^J{` ,M-^A^YâäRUâM-^FM-^RăjÚÔBÊ-M-^T~EµM-^S@H
^CM-^ElÛð^FM-^P«!i{^H@^CÑ\$´ÀÚA^éé¹j³µÆ^Eâ^M-^Y²?|Ÿ^@§ÊM-^QM-^OûîM-^JxBqë^X+^H}öM-^X^F
M-^T^OM-^[«^]X=, ßM-^BhmM-^D^Nă^-\n
i^Vx|òM-^UĐáoû»9TM-^^M-^A9°f^[e•19%ß\$µöÆ6``M-^T`³ß"ç^OBM-^U^KM-^T^]R^V»ăç-¬ăHyAÔF4M-^[SN
M-^[úsfsJTT»ÔE M-^UÔ^-^A^A^@>ð ^KM-^L^@^T^@^@M-^@M-^^^AM-^P
{^@pàÝM-^QÓM-^Nî½^CT^@°^@^@Ó^O^P×;^aM-^V^Fuú~öá2M-^A+~^K(^U|^N(XJV×ÑÝM-^E``ăıŒ e¿°ă^]X,×q)ô
è^@Ô"•AbM-^EÔ;ö^RtËx²M-^[M-^^~^X/¼yò/Ê^aM-^D^]P?àî(Ëpa^Că!KÓ6¶^^Jaé;7ÚM-^ZM-^Kî&Ů¿M-^]¶"#
^]ôË•
±LL´ñnhîÑÓ.^KÛhw^_W>g0¬¬ëîðo-éÛ]UÝz¹Y?^[-±îi,ăy i*M-^\\«+ÿØ\İ+ÖM-^R.YT^Cæ uM-^U^äüø¬ZOâ
M-^Y4`M-^UM-^U^\
M-^[^G^aêOM-^H5« Œ!V^ET73^Nv!^A;=ËuNM-^ZØMeĂăçø

ñà. ^U^OùM-^FM-^RÀM-^HóÈëck¾; !ÀÛí«¹ötë^]wÂ^KÄm2ªÜHG^]i1hà^\ãzØÁ#ã0``M-^B~XáB M-^S-ä^K{M-^_
M-^L^Z^EÝ M-^Uý°M-^WBteJÍÖ^Üp^U^[»^P^YÉM-^BM-^\8myÉ°M-^SÉM-^XM-^K ¥^?^NsU e^PÎ*M-^D°w;
M-^OÑDÆ@hDè^FÄ^\M-^MM-^Q@S4Wîù³Íjð@ë_^\îæ
½ø^?M-^Q°àN¾t©M-^B+kúzf÷*^UU÷;ÛØñ;PsÖM-^XV^_/^W*°¹M-^D³ÅM-^RQ\!ÿß0¾LêM-^@^W9«M-^F^Y^PY
M-^LM-^Z}³``M-^]s(\$\$r];ÌM-^HWW<M-^OM-^WZ{±á±MÉ×5Õµ[M-^R«;0b[^_Óú¿M-^J,Ô^KèéáWM-^@^_NM-^BÄ
A^SðâM-^HÌà^VçjHð«k°î£+Ûypñç.^UÛM-^Rô^P÷°µM-^EZ!^?M^]ôSÛø^B|½»^]#Ê¼^C@PÛ^[È-^Uiym-^W°ý¿
£yCM-^H' "ÖM-^\M-^K^?M-^ÄbèïçÖáªæð.¾^]¶zë.ød^KòýÖ×nV•c^A½M-^DM-^Föè²«hşá'AI%NM-^FM-^S•
^Q~
V+ú»á^DM-^BαfÇM-^T*şb9ËÜ6Wpë7oe[HM-^T^KÝ{çM-^I;^^û/-È>M-^TRy~xelM-^Piôm-^TËÆð©M-^SÔFÓ%ñ
M-^IçGoM-^Kø1M-^P-M-^\M-^\YøüNýDM-^[M-^G^B]M-^CÓ
;Æ/^AM-^JM-^SÛ-QJùç^N@^K3|RâLððçÜÂ_6^-@ZXÄM-^VN-M-^CPFÀw^@

7ç ° ðM-^I¿Ö>¼e£Z,5|geb3

¾Dä, ÿàm-^_ö>V

bIú4dÅøfp£<^E3EŨh*;IM-^VM-^F'~>^'_R(Ý^\î^Ub{M-^@ÖR±M-^W¾EøL(M-^FÈx^K^O^W ¥İ"ß@ç\Æİ½ÄT
M-^SM-^]l4Ó^OM-^FÊ>¾î tkçM-^Y^_c(ě^P&½M-^T^AôÄÈM-^B>!tĚé»^]¹¾¬±dCM-^]®"yM-^W^C^CM-^PudÎti
c@oYă^Y©¥Mx°ç^R□M-^XŮ^@M-^QTÆåvé=©^Xqçă-Ó&©M-^AuJü^EM-^[\±'RKÇ288^\I-M-^Dò"yç4%Tb?I)9!@rèè
öM-^O^VîÔ^Z^^ -ĚÈbpBJ^[yM-^I^BM-^GZaîÔr²Ÿ'dÓe^UM-^QĚM-^[H_M[ÎM-^MI^@M-^B
^@¬ ^K^@|@^U^@^AÄ^@^X 9°^@wM-^X^@ ^@^@M-^@ö^@^@^@^@ô^A^QL%7Æ¹}f6QÕªe1";Ã*ÔJ'M-^YÎM-^EçûIb
M-^Z@W«M-^Y?^S-E^H^_Xê^^Àì,à¬xŮÊxªÊsö*M-^Aª80^K, ,À

^@Ô*ú(^Vâµ M-^OQM-^FM-^G!70M-^CÎIp¾M-^A>dM-^L~
Ë¥ðÍKáL^R.4èA. D-M-^P^@@^P^P^B^@@^@@3^C04^@@^@OUM-^X i? M-^H
^R^2^^Z5ä^P,PÇ¶¬(^VM-^VØÖ^QM-^Z{
w•7eĩmÖnÔÊh_QpM-^A^@âpM-^Dö^@^@^d^F®h^@^@úÀM-^GM-^P íî,cM-^A^A^HÂ7ßFÖM-^@&À°m+;í³^Y□Ö
-Ûì-Ýµ-Û²»îŮsÍÛîöVËßdÇM-^L-q±vîeüñ-m³M-^UñM-^MM-^WîM-^VV^W+>ñbË^FYÁ^Aop^BM-^S^D M-^@^@q^@
^@^Q)Äª4^@½Giù¾w¶÷½Û^]Öª-M-^Yv^]¾wWµmÛ;UM-^XgYTÖ^Z;DM\$ÿ[M-^EN^R!QM-^AQ~D^V¾^B^T^Rp8ð9^N
^\\À^A^A^NM-^\\ÃM-^Gù^XM-^_

ŷ#ŷM-^S^U^YM-^R^Bi^X²\$±M-^ShÉ

M-^KG^SM-^S î^TM-^RâE^?^UWS\$æ.yTK¥0Ûê\$&PV^X^CUÄjc>Ç<c'ø^OÖ 6^[ëð2@M-^J)M-^VÔM-^[ε|S
"Ô^Y\ÇbÛXvvîL@Ä°^Y³%±M-^[ÄÍ'MýøGM-^QJóþE^SM-^],Ýð²M-^NM-^C^?}iM-^WüM-^HM-^IM-^VÔ2ä2³ªø
M-^L¹É*¼üîM-^M-^YM-^@LÖM-^FcM-^MîMmM-^X;ç^R8ÔM-^Yé#ÔM-^DbfW

M-^Q-eEv^M-^f)<
%kß±¥R(lwç^M-^Süm-^M-^YÛé-M-^YkM-^F¹3@/^A¶^Pí%ã#óî^U#^]0eYR^XJÖfM-^Zñ^HÜé^GM-^B@lèP»ñ
M-^[: ^]²^CM-^Qk^^²ªU-^".^SM-^Kß²Ý^?í2,ÊYh÷ÛÿÔ^P^Vp°-°^\SÆ3-^Y^°@M-^U^EpxÊ\$Ní•ârÄ^RóM-^EY
^UZÝªM-^KÐÛ%M-^M-^_) '[%îÖ2kM-^V±ZREKÖÖãüæ" \$M-^EáM-^]kênfM-^^\ÐìÇ^Vù{j^Gî^XM-^S&¼bÈM-^[
M-^Jî×î»Ôê>ÍV;ë^E½È~ª³.E[=M-^[ÚN@•M-^Jæ^N³^RÓ±p±XØ|¬'Èûð^X^[KÄÖÇM-^J^K^HÝ^RjM-^NM¶È•^T~û¼
ßh{^DìkM-^TíwM-^Y[^cM-^H>îgÛ^^fM-^]¬yîû°XEÇµ*Ö^Vª-M-^Gðµ;.Ä^^ØOy^Kmk^H^^^Qy^U:¬ÊJM-^Dø+X¶
ãçø&^DM-^BÔ'Ä^RM-^JãîðúµU^YäÇ¿/M-^_dðÊQ5M-^^\þGM-^F^^»¶:YvK^\çM-^Iðè@[^]Ê@¶M-^[é(«aâq0Ä92
ðÄ[æeN^^M-^YM-^SM-^VEÔüäqÛ-M-^WÄM-^[ÖÇM-^^«ÇOc^N\ðXM-^P¥ÛZÛM-^IM-^Xðäv07j^o|dù-ÈÄM-^IrM°
}Ä
rM-^X»f¼ÄGM-^P_ð<'d^X¥!l@¹ß^@ZM-^F#kXM-^ADjT\$³«vÑð'ñ"\9³3<^C [M-^EAM-^Q8û¼uM^UÄ³^@÷Ú±#d
M-^XSðî^?M-^D`M-^A(zª^F»fjóä
â^Uø/M-^XyM-^NH^NRÖÓ[*lé^\^HÇ^ZC²éÛD&^HR-îîd^YM-^@ÄM-^C¬í^UÊCÖ´¼°yy¥M-^@c'"Êr0\$YM-^LM-^Y
M-^[ÐmW^A^W½^]x^]Ë504M-^L:M-^X¿>üÓ3ø^CM-^[¼¼*M-^_Z^NM-^RkZ
îM-^Hk^AðãÓ^?Z^Vt-M-^Z^Vý^ZM-^D|ZÇ½÷jRM-^G²M-^S=î^?d!Ôç+]M-^ISM-^VîÛM-^UM-^^j?^FM-^M\$ð
M-^Vð^Waz^M-^I,úM-^Dm)½ ÝEM-^U@ð^R^Wüð)j¶hsM-^XM-^[PÛÛ1ûx^Qw9óÔAvß¿M-^Q#æ;æe^VOM-^[û\øF
'±3>îëw^UGM-^FM-^^Z^F,!^N^O
ë¼!M-^[
^/?äC¹d.èXî^?oÍM-^UôÛJM-^^\Ó°M-^YM-^I7V)M-^U7¼^U"ÄM-^@V¼.M-^BWM-^H.J^AÖ\$M-^Q@Ço; ,dfM-^R2o
²M-^Hcäíe%\;«ÆcÊ^Y½;8_bÛ?¾ã"}M-^B^]

á[-ÿ2ÃðBM-^Z\^aÜ:×QWTS^T=/N½XOËµM-^KX« ø^m^Cö}M-^Gšš^UØ=à×/á<^PfË×^GQ»^O.¼=M-^^^Ä¹Tù%³ù
M-^TÎ]>F^NM-^]üÄ,*]v>M-^]5M.RM-^W-^S,¡³^OJjv-Mu
¥M-^QÀ»bKpM-^SjÜM-^[eìæ>æ M-^Z^V¹A°6M-^F^UM-^HM-^TÚcó¶M-^Xfðm^R´^Tuî/M-^[ÛQp¾PM-^Tm
aUÆ»]pïìXÖM-^F|>p/M-^OÔM-^Yøø^Y^Dt¹æð¶}ïeI%F/s-sóñí¥íÜÄ^Y2ÄwäM-^ZM-^KM-^_g3M-^AY^N^ZGöÇ[
^T^V»µ=2^GM-^OQ^Eİß^S^SÖ]°3M-^O^RqY»ß¥^Dp°cö;M-^Y^VM-^UdM-^Eà|Àç!]g^Xpÿ^ [j^DM^[H´^BU9^Sú\$
A¹UM-^[üé{M-^Q-Ú^Ry½¿;FÜ]M-^DiÇ}M-^I\$M-^Pc^?ö^X½\$îM-^@dMiðM-^U¾Ûk%Ig^Z/^~n^@/^Ha^O!^_Ñet\$
=¹)^Hö ÝùíîM-^R0Ê
ô£ð^Qàû¬çM-^T^aŸ¨ðb½FÝM-^VÓM-^]M-^CM-^Li^a)¤«³^Xx
ÅÊ^Wİç^EM-^[W^U~*ÖçÑÜ½fM-^^M-^]2,´^X|±#M-^SÄDÖüQàm-^Xí9µÈßê^^@M-^^M-^T(]{^PÛ¾M-^O2^O].Uq
^DM-^]Zêçjp/¬M-^X^B,M-^BoM-^^í^HîM-^JzÚ°iM-^_M-^N«^Bî;^VÜmaÃð^R´4µxmM-^_[j9Óôî4M-^F^Ai´Ê.
íÔ!éTu^BÄôM-^FötôÄ bM-^G\$ð2í_ÔM-^G^Y;äg»
M-^F^@!w
/N f)hM-^DYWM¹M-^[ÔÂ©Vß^FM-^ROM-^G^U,M-^Y°M-^JîM-^JlM-^GôWxßßÂ{îM-^Pöp3Æ

ó]M-^EJú=^ZM-^QÚ£^G^R¶I7F÷^ßM-^Pg^[M-^MÐ=¼ÁI+p[M-^C-^ZjÆ¿^?Ýû£C4Ð^V±ë§Ó^S @M-^O<M-^S¿ö^Fô
 Ũ^HúÝ(jVímM-^AM-^KÇM-^HM-^PÊM-^FjÞh^Y4=1ZOJM-^S%ëM-^ZM-^A! OM-^]^@M-^@¨åøM-^UìM-^PÝð
 ¨ëM-^DM-^L^YC^TM-^BÈëP@oksM-^U4M-^V|ßÚM|6a17M-^J4α'M"M-^W^_Yëð²M-^P§½ékM-^KuÅ!

?sèzĚa: ' #81 .ÍĚ-M-^[^OĎ^[^M-^Få; □M-^OîĚăÓM-^P«ăç5•çM-^RM-^M-^C^^ M-^]; M-^D*^GÎŮZM-^[Æ, I
M-^OM-^DnEÛÿ%1 ^M-^Vkä~^SZs^Qx^CF; qYÊfM-^Nî^ZM-^W?îÑĀn^Gă^^£!M-^N^F^G^S^X/

M-^Xísájó°ß÷^ZİêaøÍAİM-^TPAà, ; ``P!^Qçû©: [^ [^WpM-^E¬v!±¤•DM-^Le^ZÓI¤Á

M-^RM-^E-•1+M-^L
M-^WgápçÍ¿ä¹îÔ ^AM-^H¹+M-^Yl^D^Nα¬qi¥^Ov^SSç±ô2@^T«PqçM-^Y¼^T_oðyrÆÐ«»\$»BÓM-^RiC]pÓw^Eð
M-^OM-^Y¼2I~}M-^LiGM-^BjÊß^Y=¬(PýaeúĬ (µâxM-^C¶ĬAM-^YĬb/4wĬ¿DÑ5^GĀ:SĀ^EÆ¼M-^N-q
xBCçh Oi:ÝQĀuM-^@ÓM-^SæwN/Z^F^E}«M-^_Æ]^*_¬¿æ^X>Lr¶M-^>^QsĚĚĚ©^En¨^@§M-^PM-^BéwMX^N
M-^[M-^FM-^Q
M-^X}ep]^UM-^OàqT¹sM-^Ua]G½;BbGdØ

ôa1^W^G?M-^Tb3;H^xg;¥\$¹©©½ă,d^D(^]ç81Êè^Y

J<M^H5Ô@ Û^]qÊM-^W;ùM-^JrbM-^Lú:cM-^Rý^_^^n^PM-^RÈÆyØ7¹³M-^P_éÛ~Èú^DyM-^Bç^@M-^P•M-^Síî^[
 ,M-^B)^SØÐ=;qÛ^A^D^KÊñª'UgM-^Z^@®EM-^C ,¬¬M-^Mà^E±\$M-^X°M-^EmCcM-^Cî½/^Of°öí^OÖ´ªú>èpnJU©
 ^Vu(M-^UÄM-^KÍ^XÕÑ^A`~%j
 Ôâíù³T½Ä2^]:é±vXM-^Ys|M-^JmWÉ~jM-^RöM-^MîM-^H,M-^C@:4Æ

iDM-^^,\]^Câ;æiM-^XM-^NM-^Oİ;ó;-î^O:<ñ^Q¼àÈÆ>öW6s^Fs^P^H^U^C±^Sö1^WÂ½Ö#^D±M-^A^M-^OnázÚ`
 ÂĒwX0^F^UM-^N4R9Ni°5?^GYíÉám-^H3{vt²6-^G^EýŸÛØM-^HÔàF#½(!tM-^CHÈM-^La»M-^Sß^EùxM-^DY&æ:
 M-^_ÔâĐ2M-^K/sđ^A?ó^C; «M-^[^?.Ĥ*Õ Ĩ"6{8|uZM-^H»ðøª)Güm-^IM-^C½«ti>Ö¹9•xM-^QĀ(^B
 -Ů0f^E3^RnĤ¼b^EM-^X,^U^FM-^@W «tām-^NK;vûjĭÍĠ^?^M-^TG4r^Q ¼z^V¼ĭ°4đM-^Ij\$M-^K2Āo1
 ~^E[«tĀæ:«%M-^S¼á1CĒî^W~Ué^\\ØŌĀ`ĭ;ðÈM-^R DWHEN^S0^TÑzM-^ZĀð@f)M-^[ö fðèŸ^[ĒX\$D^@^@»^AĪ^H
 ^H}ĀPH;ĭc^[ñŸăùÔM-^M-^O¼eBç<_q1- \xO^NM-^SĀ{CyĀM-^_P^Pă\Ÿù-^M-^AÑăĀüëfZM-^BM-^P-^DásÓNr
 ^WM-^DP^@©`^C^@^@M-^D^M-^U•M-^AD^KøĬŮ°^_M-^]p HØ°°z(?`ÈámíÑ!Ô:ë([_QM-^@PM-^@M-^I.©JkŸy
 M-^ZM/*Ůıd^@ı~öM-^Xß0Xr AW^M-^MÉ^E^@Ø½
 ^ZDx~(fyHM-^FĀæ^@:ç^OVM-^Z^N^DoM-^A^HxM-^G5!p^CM-^R3îOn^YM-^DüĀ^AtAX^M-^TĐ^[|^ANcĀM-^A±
 ^Nà[oēm-^PşM-^J
 OM-^Dü@;»êĀŸ^TxßM-^I9M-^EM-^Wü^V^AŸ^E³^]^^°KgÓÔ^Z\$LM-^M9ÔNăŸxK>ŌM-^X1O´M-^UµM-^C^U^P*Jê2
 M-^VN^DŌ^Kē'ām-^LM-^R>^P1^E8^]öDM-^HM-^A^N;ĀM-^N^Dñ`^W?G\$½M-^TĒh`M-^TLç^YD^@M-^@Ī^A4p^HvP
)¿y^HŮŌ KJ â?^XM-^OM-^B;M-^E^PöAŸCĒ^C^W^@^@P^G^@^@à;^@^@^@^@^@^@^@ĒM-^X^F4^@^A@^@^@;^@û
 M-^@dM-^D^G^\\^@^@â ^@

^@^@`^@^@^@^@ÁCM-^J^D[°M-^CÖ&âM-^_%ÑKyýÔc^FA-ê¤^^Q¼\$^XÃM-^D,WM-^XM-^X>ßIh´ñ7,^X°Ä2M-^DÃ
A:1

Áo M-^@i^]^GM-^TM-^V"m^X3ä^^FE^N¾æM-^EM-^F}fÜ@M-^B^O÷`Ü±AIBáÁM-^P^V\lM-^SM-^TM-^[Ô;«^Bk^
M-^@í^Z¼<B^Y^N|°±î?DM-^OPX`ÝÄ;
ËM-^YM-^D
Öh^Nbee¹\$ü"-âM-^Xn;´M-^NzM-^C%:gÌM-^P[ÄeM-^_M-^Khî-
^Z;Ūn6ô^T´3_TèDM-^PÓÁ@M-^H^\\^BßM-^@ø².^Z)5^@Ñ<èZ°Ö^?M-^BìM-^[§^^êÊ;dÈØ:ÄM-^@u^QuM-^J^_
M-^Qö±M-^[aō^O-F1Z^?}ÖØÑUOÏM-^BWßM-^\\M-^Oã´=£-M-^HYòsfİ^Z°Ê²o°QM-^SM-^SÜöw¿Iç.^O^P2lM-^Wó
;M-^Mg•M-^_!^E4S•¿M-^D¶v¹M-^]j[x-M-^CŪÁp
M-^WÈÖpM-^@W^\\äüM-^G,?#)M-^]´İ)nw
£M-^TăpM-^DM-^EÈ^]\$^R\^KømM-^XŸM-^B4\$(1T½áT"m-^C"ouß^TM-^D>M-^Wím-^MŸ-;^TM-^WĂM-^Uô^\
M-^UŸÓëÖøjér8M-^M\$â\$yßRM-^LMV,ÍWûM-^INM-^AÛM-^SñÔM-^M9ôM-^I`OæM-^M ŸM-^V¹ìFò¿DM-^K~íw; &©
^ZR^?`ßñÍ\$µükGÑ¿]dM-^^ İu^H5|M-^S¼Æ`^5Ø£èE`^DJ^C#^NúM-^VoĂ±M-^AJ=1^A¶ÇM-^^M-^QM-^Cù
eİİM-^P`2ÇM-^TÂM-^BM-^E^?şM-^]ö^Eİ]"o^]í¿M-^OM-^UM-^½³st±Ūă^W2^YGV-KÊM-^N@M-^NM-^AÉ^NÆ
M-^]M-^^ĐVo^@|Y£ ^OŪ¿M-^Sj0¬\$k=AM-^U ¶-Y*İøM-^Wtj^B+^O6M-^NşM-^H,<À|^A?^N?UÈ

/

^@^V`ëM-^ZkÃĐQÇfÿ^Pÿ(M-^Tú'3<èÍ1^U^_è@-M-^NG£~'ÓXăĂM-^Ný
ØM-^D M-^T-ö)^RDOĂüëmQăP^G=^@"ña²M-^C(ÇM-^Byê: +^V#TM-^EÈ|ÚdM-^MM-^C
M-^C@GQ2³ĂM-^LM-^M^HM-^SbZ_ăM-^B^@@^S^C°®^Eu^@ĚšKPyĂ\;½M-^PêØŮM-^P}g@-½%^@ăăİSó?İ+Q?£YøŮ
!©ç9M-^@3^@^@M-^TĂÿ'ÿM-^J^H^X^OD\$-ÛH^Pb>|°^XM-^Mo]FM-^MM-^AĤ^P_F^Np²u3lfİiM-^R«M-^_|Ŏ»qí;
nĐM-^Hü,^[M-^Q^^Ĥd'xM-^Q^VM-^T¶^YαÍM-^M28^XfĚM-^DM-^LĂnGbGM-^X%hM-^KaĤ^M

M-^A^G^@#ÃÃM-^Og^F^X^]^ZLÎM-^AjÕM-^Eå>ø²Øßû`kM-^E3M-^J~M-^U¬ M-^@IÖáÉ§^X«M-^CÑç^QÄÄİM-^Vb
DñÃ_İ8M-^IM-^]¨"M-^KM-^K¼M-^Xİ^OkJÓ°aÄ´^WIãGö"]M-^G°WM-^äÄØ•M-^I^P,)s¶êSM-^X
M-^X9\cÑðM-^IÝEðÊ^]dM-^Zçþ+iM-^_ÿM-^Z)İxRâxM-^îMÒ^XT^B#´M-^KV¶¼ëm-^LíM-^RfM-^GM-^Qb¥^X
M-^DsË;³M-^T=ÔÿHLM-^ZÅx>M-^Jh^TëBB□Qç°\EQþ
ÝM-^LHëŮWwŮnPuB´+c¶Uê?Jþ&M-^S»Va6ë*7~NÔM-^D}©^@QÁþ

M-^ZËëfM-^T?M-^Vöñ
 »ªM-^Má¼íÕH{@M-^³A,ö^BD?£

?à1TÀ~KÉM-^FÖÍM-^UÄ0¬M-^NmM-^TR?M-^PH^RM-^JM^U9ûç6i^YM-^Q©*²de»M-^]¼>áWÎ^CìÕîÏ" }QE,M-^W
M-^6M-^\"a wHÖ^PlÈ6Ï°ìVöt,HæÐM-^Otø^FM-^P^F*ña^]BíM-^JM-^Y^@b^X"•¿UM-^JM-^I^X¬ç×M-^@Gµß@+
M-^FJJ ýþI&^Vq^T©bM-^^ðßM-^GÛM-^IM-^^

4á0%M-^]üwV^S^KXYα-C2°´ØØ^EM-^OäĚ#=ĐùQwM-^DzëM-^NšjCúĐ%ø)Ú.ûéá'pÇjĚ!½ŸM-^OM-^M-^C-ü1hX
M-^BbM-^TM-^Zx|^D

^Vã" ÝM-^G^QĚßĐ^NQ³Ç\Ý/M-^]N^D«Êê^Zhéêu}ǾM-^B^-M-^X•@«iÀŸfç*ÉM-^Wp7ûAÕ±Zh¹+7Mye¼^SIcîÛ^ó
M-^Mě^W^?SzSòM-^Y^^^FIÛ=ðÉD/^UìÊ3aM-^Zz"M-^Râ(Mù^Yù6^UÁ^AM-^]Ä M-^UPjM-^^^h
M-^_ÄKÉM-^PM-^QQ^KÖ@;^WM-^Q,
iT]^-èYM-^M^XT^C•AU¼ç^?ñ©q^WM-^T^@'^S^@HA^Y^@M-^\
^@^@'^@Úx^F*^@^W(^@ (Ö4^@^@^P^P^B^@^@^@4#`5^@^@}p»Ûö°Ÿäü¼^V§ÕðauúØM-^I¼M-^[u»^ê×Ǿû^_lwv»mç
-°×TǾÛoûozî{ }½÷j•4°,e0M-^VØF^QM-^HfM-^JM-^E1^C^D^A;A^PZzâ±6 \$^@Ä^_jU^PM-^H ^V^?

M-^X^@^A^@^YM-^Y°ç^C^@Q^@§&M-^M²Û%Å\$^REİHXñI^R4êÄÚO^S[J^AwÝ^\£M-^PM½÷BnmÛîñV^FŮl÷;]6¶Ô¾9
^\\FmmzrJŮŮ)~y`×^V- | |´*®V¼NÄ^]EA92¹Ã^A(M-^PHŮ^\²iM-^R@8^@^@f^Dİh

^@Ç
ùM-^ ^nîÛî]»M-^Y-¶¶^P]•M-^Zî^»Ûíw×@-[^TÖ•îÝw°¶pÛM-^RJ^WQ2M-^PM-^@9M-^RH^Hb)12QM-^@^H^H^R
M-^MB^Q^A\$M-^Gù^Qá^Gø^_âAM-^Pé^GM-^@=}^?M-^VL^D^C;ÿ%^K^S^EM-^O^QrÈI^OHM-^E^AõÏ@,M-^EúM-^D
^Tdm-^N^KúÄM-^EM-^N9
M-^F0zJM-^@*Jadb©9M-^_M-^F&«KBM-^GM-^T@)9U``M-^R+CM-^Ir"l^RÐ3Fgæ

^H9QM-^Gå!´gÑ

¬^RM-^UÎ;ö^V^@^?^^^Z@)T^M-^C%©FM-^Tü^S^GM-^Ltc?gÖ½|6

M-^EØô° M-^E@d

úÊ\^RM-^UÚ^YĐÉç^C^O^M-^S<¥RiM-^]M-^MÑaM-^H\$Ê{M-^\\iëiTM-^Dİv4)9UÕ*,HY_ØM-^U,8~úü'yûBM-^[
 _&ë%M-^QM-^MM-^R^KaM-^Q)ÈĂ|ú|NDM-^DñÉ"çM-^K^D×.M-^Uå^D4^CQ'M-^N^SZHÖM-^x(M-^IeL':İ^S^HM3
 ĂÉª#;M-^M-Ó@ÆM-^R^DKyÿĂ¥J^VHĂðsªtDM-^H3Ă£¬^W0M-^Fă^\>k'JÜE°
 M-^I£ÊR°P§^^^WEİxM-^PDa+^\»M-^AM-^Ev#^HñTÉM-^IM-^HªM-^PÊ!ĂézM-^UM-^E¬\Uý\$ĂM-^RLÚsBµM-^]
 ^Pñt¶M-^TP³²÷²A^^T^KαPÉSMIxui[A¹M-^V^E+^P^TM-^K-Ê^U^PM-^Qµuù^U÷UM-^WM-^ĚQXôM-^U?
 M-^Y^Z^RaÎO^PM-^CÔG'PÑQM-^D^^-ŸM-^PM-^VvM-^Rh^PşêJ=M-^Bgḡbù-ñ"M-^W|^WgM-^[ÎM-^InM-^OM-^Y
 M-^U;^PM-^^ÖbĚíôúM-^D^->#{YC\$h
 CT"@ĂM-^ZM-^QĂ•\$´¬¬ăÎ"µM-^P½8¿´RM-^@¿M-^[³',M-^@ĐM-^EdpOM-^G2jUgV^Gu^SM-^DM-^HUĚk
 mU

z²~ÖúÄvIdM-^L

ÖV,^R'èÄÄp^R FM-^AM-^U^]°^S:úÛé¿^DM-^Q,Zi2_êÔø^CC^?M-^_ô|ú5%ð*ÇÄ=FNT°Ø*n:^Z%^H25@Lç|tu-§³
M-^EçM-^L^]^Qð.Ô3yCα°^YÏÔ4ÆÎêØM-^W:[ONM-^PKM-^ZM-^L^Q»Z¶aM-^IWM-^\\^"Ô^SÔYð^G:§^[M-^L^äêA
^Uänucc"ðM-^HµM-^ä^ÓNM-^R,M-^Jc^[M-^Q¿î\$"[W'^Aø3u|j¼Û^TM-^PtÏG^UM-^Ez"~d?nëäM-^_M-^Q°ÿ@D-¥
Q½^@/»ÖáÔuM-^V^Vj^aûÎM-^O^M-^QZùzbba^f^GrK3ú?,_vM-^Q:ü"~è^UM-^CfÄÄÿÄ^]ÝDw/^Xt";AYM-^RM-^[Û
M-^Dl~M-^VßøM-^_P^GçM-^IÈ^G^ZRM•M-^Eo⊗×,Fmâ§ðzLK^GDêÔû\ÿ^Z~\Ê¼A9M-^CÏÄ\$JM-^V%uM-^N¶c§^O-
^OÇ^-M-^_w^]N^YMçδ+M-^SfÔVt^]û&eM-^G1àm-^VÄM-^OÝ^GM-^Eqj^[ÏÇ39ð|)^aNË1½,M-^[ðÖ;í-OÄ^@a#B
M-^C^T×nÇ?M-^Ua6Ësyñö^PðÉM-^\\i4^]M-^UM-^MM-^KM-^Q'M-^U!ç²NM-^WµéÊKß^@D¥HM-^RäOð&T
[RM-^]Vαð\(\§ám-^¨±Í|s`5\ÈÏÛ^K6"éM-^O-^S^Sr^ZÍM-^JJÄM-^F^ZÔäaÖeicù>^[iÑßM-^]Ö¼M-^DM-^_½~@
^Kë^ZlûA^Klñ~0.M-^Z^E¾"~T\Æð²j£^Bè^O^^jÔu°ð^?ðPé=%^[7þM-^JiÖf^E•^_5=M-^P\7;M-^AêâV,4^GM-^K
^DM-^SÓ@GzJM-^V'è3ÝìÄsÝ-^Y?9¥È ê]ÄY7PZ±M-^¨¤;úM-^C^K^Xs%ðüWùM-^N±
æyZf,5«M-^RM-^E°M-^RU^@xü*~*qS-¼gM-^Qi^\pÛM-^IÛM-^XM-^MRß-EM-^_M-^Sð^R@&^KJ³TÖM-^V^EM-^U
^@²M-^JÖM-^WÎ©¹;Ôaw:^RM-^Pd4Ä^aP>-Æÿ^Q'^U^WÄ#^Nç¶^a²ÿÉ×¾M-^E\$ (+îM-^Bí^\\cÏ^WþM-^Uþ^aÿJÿÛ,¹º¹r
=^Q^Zà^a[
hM-^M£t%Ð'>^a^Z!MD].:~M-^F\$^GE^M-^Kð-£.ÄtøM-^M⊗È-e^?^aòì^X+pM-^V¹^TÆw%^Væé/^]ütMÓM-^B×M-^¨^È
ÆM-^Wâé-9bâ
M-^U¿/ÏM-^UM-^\\M-^I^[â¥hM-^R^VÄ°K~ /ðùùM-^S#Ñ7Ôð\$^[~¥O^aM-^_^[[Ïýµ@l,í#çÊ]@M-^Q^Nn^A4m2Ônä
ÛM-^R^^^PM-^_M-^\\M-^C/h{^N^[M-^M÷M-^J4?à¶^Q&ð^T^a!^@ü"~^?6<^Açið-:ÛM-^Z@i^UM-^P-^XM-^B
q'fM-^DM-^@^µc|ÿ^SúhM-^Ctç¥öo:M-^A5WÓÄf^]KôM-^IÖQèTj4f^D-^_""[(;ä\$Kÿ^@N^[a!

^UæM-^]

î^W230ađM-^G ìî°Đ^Q"t^SM-^H÷^C^KM-^KodİöM-^]"D*à^2M-^@M-^Fãm-^\\-«F;ÝM-^Kk¶)VÑbëü6|^Tâ»R
iM-^\
M-^AXq%Ů±^@^AZ»ô50¼M-^\\^B.%ôîÆ^@M-^M¼18>÷à, M-^R|M-^Y=ôŮ^Hî®¶^@«pöö.ˆF:ˆXěĂ".ãm-^Ro(
M-^ACM-^McM-^@^^^M^U^]A-+v'^^D^[ö³M-^Q,öKùgö.ˆEpöý;¼«Ý^Op^AâCÊb^^U w\$îì©M-^@öc9ím-^IjM-^Y
M-^AQT²^Yö^G^D,--cÉ^W#^]ŮöXß(^D^\\.^³ÆGjáVİ^@^X»M-^\\~eô2\yÇtAAĂĂ\$^R:'b^@M-^OĂám-^_Ă•De7M-^J
çf^YZÇĂb%ˆB°28çó+îsîM-^@É?i^S^D]^Cø@-'^[;§^Fm^G¹M-^CĂ 'M-^_ĂM-^NM-^_M-^Nb@Z0s- D
5ôo+~^ZM-^O[M-^@M-^J^]M-^ZM-^TM-^IM-^[ŮĐ^CÿN6RMò^AM-^_M-^P^DDüM-^\\^S:M-^MŸ^\\^AM-^JM-^Y^Y6
Ů}öM-^X}\\¥Ká"Ů:öJM-^Tçİµú:M-^GBM-^"°MĐZM-^Sç\$#LsM-^S^[ŮŸŮöM-^A\$gxxM-^M"M-^AM^Vøu^D|M-^V~
%ŮİyßSM-^_go^T^NM-^Orl~Q±/"è[QÑhM-^_M-^MđîM-^UîM-^M-^Pœäl^S{^NÂM-^JěXí3^]ĂŮ-ă3HM-^FĂ_D«
^B^[ßM-^NĐ;gM-^Z\$M-^M"ta^[ø^E?ññ?^Sİ*kěÊ>(qm-^X^DUM-^Z^^³ø8DM-^Q°M-^SM-^D¶:#"•¶ZE^?Ůñ
M-^_M-^\\ð'^G^M-^FÝIM-^TlM-^P^Nşðè^H^S0^SçĂ»-1øĂzÓ=/^*^@²cs!0D¹QçD7M-^[w[nèĚí]Fµ¶@^D¼ ²
hù°{M-^JtM-^GáVhî-TöM-^HŮ^^Gæ.98éô[^S.WM-^ZM-^DěŸM-^U^K^Cò#: .M-^GM-^TtÔnµ^@³ĂĐ³r-N»è!Y
M-^VĂîM-^[&ĂđS|M-^_çdw:M-^MH^]ª@@^SăM-^TSo0
l~«m:)q^OĚ|îM-^[°lM-^[çM-^]7û>QjM-^T!²^O3M-^EúImørv³Q 7FsÆ»M-^]V^QŮNĚµæø»M-^O@Ó'ăĚĂĂŸèr
°^FİöİÇ[M-^W^YĂeîM-^VÉDíă%
Ă<^O^^M-^LM-^[M-^YW^ZŸM-^OñM-^IM-^C^4³[Ůçûâû@gŮô]ĐM-^RYM-^P-M-^\\^aM-^\\òM-^BRM-^Npíz×M-^PgI
M-^Eí"°"«M-^JİîM-^]ß'MÔJ²M-^Fás÷±°M-^[v!³^WßŮ¼^W»rM-^[^VM-^Bf^\\i^Y
ñjª@/]^;~-®-M-^GXoM-^Qô
M-^WêL^SY{²-a^G^S^Cö6^*15Ůpîw°
µô0'B!^E°^G2hkĂM-^O çŮI&î^N^QZuă{eâm-^XJ!4'Ă^S'±M-^DM-^OcÓ^DÇŮ¿'p~^R¹Í±×M-^T×BgîeŮ@2^G^[ò
æÇüð^ZM-^@M-^Zøěágøð^B»^@QĂ^S,|R7M-^P«ĐH¼û"q^@GP^A-°âtø^SŮ^_}OĂcM-^GCM-^BM-^PC9ăC^NM-^Xi
ĚZĚøç^VM-^HM-^E0M-^WA¶-ú^Awf¿ìö^Ff^XQăŮ7eĂ'

¶oQY*M-^WêÖM-^WúM-^_ ^B^C>P}M-^@•Ÿ7^ØÖM-^Qxßî^AçÇ^QbÂâ">^D¨&ç^@M-^} ; /Ê^W±•¼ü^ZM-^T^T^Aö
M-^GM-^Lø^Hx?|åo^EM-^A£M-^R²vw 6M-^J^@M-^F!EqŸñóPM-^DĀ^OøcDĚ-ñŌ¨ë^] ^F^S8^QÔ^OyM-^Y^@^UÑD-
øM-^C¿M-^]x{yùAîM-^S^@M-^M^CM-^JM-^Zð°M-^_ ^U^AĀG0Ō¨M-^@qZ\$Xù^BŮ\&ĚgJEÀùñāw^Bø^O+ÓM-^J*Ø\
^S9ÓG^Po"<M-^QM-^D^ZŌĚ^SÇM-^K'MX,´" ^Bø^O.YĚM-^FP'ÀM-^S:Mß\$×M-^O| \$q°^[9M-^Jém-^]^NM-^C^ [
^]*ŮM-^@ĀUM-^M-M-^M-^] ?Ůîð{M-^N èM-^]M-^ZH×÷^?ª*M-^]çM-^]>b&M-^TŮM-^F|kM-^Hi-«ä?£
M-^AM-^T+M-^JF'iu^-J^PW^RIu]Rö^QYİM-^C^Q^C;ÍWÎTc°M-^OWŃT«q®^Qá^G^PÊ_JĀ'g]²^C^OYç|M-^AÇK
M-^KZ\9m4
^QkŌuKOí1^[M-^@hM-^Z;¹M-^^\üÊ;ßp-M-^D]µk«Bæ¿öæ`Ā^SM-^M¨^ĚßßM-^FNEŸM-^M-^B^_Ê\i9!OM-^RŎè&X
V3Q÷¶°" ^X^W½nNĀIŮEŸM-^C{h÷8İ

nóuYÂçb^A°ZOMÄF»M-^Ts´;³C⁻w`g7^ADÜ²]M(^GIIO]2^SM-^Hp-^YöÖM-^^Ç÷ôû<ý?uM-^XM-^WÔ+M-^KcB^A±¾
 Î^]M-^G^H³M-^Q³M-^X®<^?´´Î~:=^?#7^S< ùóM-^M

^@^OJLögM-^ZP»ÛnÈnÕñkaM-^G°>LG#U½n^W^TH^T
,M-^_M-^Ik¼ðtLðð¼ra\$yg#5×nfK
M-^PM-^_ÍM-^YpM-^NaÍÍ^asÛT^C%üÖQïM•_ÓM-^\\) }MYáã1ñ¼ðñ³<ÛðM-^EQ^BM-^DM-^NEÝ%;2^[P{fóÂ;¶6_
;É@SDÑZÝË: `çý^@æ^Q²M-^O'ÛM-^B»B^OæE2DÆÿTM-^YäÍ^\\opM-^WL^FPÿE-ÛÓM-^Z{¾ I#`çÍßqKM-^CM-^]uÓ?
^NM-^R¥ø^B^[]Ì»eþÖKD^]`5-1ÀM-^L^A"ãM-^ZèM-^M-^ZM-^[a?M-^G²o¶^F}^Rph^]6M-^EÝtq¶Í° Û
M-^@R^\\^^qb"6M-^_FßvÛS?^P>àBÖM¶<?×;kálãýæÂUfðL|f×çM-^[ð|Ê÷;7ß|^BuqM-^VN!X^DÿaM-^J-ð^Pj°Èf
ÍM-^R^?4æyb^PÍ¾7ëM-^I2
Iï^X^OÉ^X&Ñ^FÌ5róuM-^PqFM-^V6^Z&>ó9)Íð)^Q«5:Ûµ^AM-^^ÎJøM-^S^BE¶Í6:1þÇümz^S\$uÍ^YðñÛ[]M-^Su
M-^Fçä°^O9ã¶ß0#¹^_¶ç^CM-^KM-^GâM-^Nt^Bú0Ãr, áxÍ¥
ê9{@wM^Hù%h^DçÖ
ÝpM-^_ýM-^O^@M-^[M-^\\ù^QaÛ³öM-^Jã^G½CtqM-^NnM-^QN1^GáM-^^!Ã^ZX»ð7Ã~^@Å;1^_îËM-^GÛ|M-^N^W
M-^IêYxM-^G(^EñúM-^GM-^Zá/ndW)&2^?^?ÊM-^Qjl^WM-^HM-^IU^NM-^Uû@#P^QēÎ` 0im-^N>;î:îhî^WF¿
9QYM-^^övm-^SM-^Lè^GdÐ@P^]^XÿM-^@Z*×x
mzÐM-^N5 ôim-^BÈM-^Prá9mÐélè\$KM-^Q¹½^_HM-^Kôav^K^FM-^Iñâ³{æM-^A^B÷^S^[dulM-^EM-^PðM-^Qe|
^NûûM-^E@

úîZÄö^QýM-^_#>j^Q| %ÅfM-^]<Ô^]ÖËn/GM-^BW\Eê¹5M-^ \ ;M-^P

^Nê^TâÔ:«oé~»e^V*fMâ#Qg^Yl,x•?M-^DM-^F^BM-^NM-^C ¶!:v¼M-^Rs[o
Ê^B×³)M-^L4 ^D^M-^^ûâIJXó^^]ÃpÑ_»M-^OtëÏEø ^Q×M-^Oé^WM-^M1M-^OÝ!^Q^H
^XáÝ^reM-^C^Zð°M-^\\^WxM-^M^NÊÃ,M-^@+á\$X ^ú£ËñM-^S^W<_X«^_M-^Qä\KùPα^OÏÏ@ã|çÒSZk¾ÿM-^J³iÁ
C
M-^\\^SÑ;ðM-^OhM-^WM-^B+jÖM-^X|oM-^AiÆ!M-^TM-^UÓ^AαJÛ^U^_s^PÀÛÂ¼Êð _KgN5 ^PÉB^Vjð]#!
i^A°ÛfÂäCR.M-^N
lûM-^BM-^Y%^\]d¥97dÊÀM-^C!M-^^AGEuM-^HÀUWCô^\q9|=OM-^YM-^A
é^N<^EâM-^NìÒM-^N²^Xp°^O^E<M-^SxĐǾÑçn^[Aâ^YêM-^Gz>q%^RP^:^W¿ÂM-^Jç¾¹Í^GM-^BT^\mαf^f-J^WJÕ
ìb³Û{Ö^NvË^U¾)|^Ck-M-^V•@x-Y^_ik5¼Q^EäÖr¾oãM-^DM-^FM-^PÉM-^Hc£@^Y]HLM-^IM-^RM-^GM-^UÇ-&@Î
c¬ÒNM-^P`;eM-^LÛzæ|oM-^Y5÷aÌM-^S3c^BM-^^¹YM-^LFM-^LM-^X~^^SÝM-^LêðLñbõwLÛ3WM-^KgM-^B5^N
^èl^PÝα
IÆ«pó~Ïαî,cÂÛUé M-^Xí¼^[/?M-^SB:Ý^A§<^?M-^JñÝM-^E;M-^V}=ÉM-^DM-^_M÷M-^RõN-^D¥³M-^@lé° ``
¥zpç^Uè^G*¬¿köwV0è^XM-^HôM-^FM-^Gh\$mM-^QBM-^A
ØÛRÔR^U^Ró^HW^M-^HöB^F^S^ZÉM-^FO³mR1Y^CM-^]1^[}ÀxM-^LTËÖc§^PÚ^^ðoM-^OM-^SM-^\\M-^K%^^|^Q
M-^Zg;^E)^ZßR]M-^TJ,ÆK^HeÏð k^]Bú"M-^Q^KdFM-^N2ö%^Cæç^KáOæØFgÄ>^Z^ndM-^ZM-^M´^FM-^Ké}^Z
;â

ØJgæ/Möú5Qæo\$M-^L?½»ÊM-^N&' ^RCLÓm×M-^L¶Öä•mCÂèÏC^_ý!h'Æ"bf»äKYk^BZ_^^\$ìQ^^7.M-^R\}Iÿ÷M-^P
M-^FÐÛr°Û¶û
^V^@
A´Âß¥7¶^Ö´M-^XÎß1¿;M-^AÁÂI#M-^F1H4)l¹E©M-^[ýÑu1#M-^QÂEM-^^"M-^DM-^AM-^DM-^Lëiz.ZÈQL"z&
M-^JrR^UM-^LPG0íuM-^TM-^^ªM-^EçT1êtfM-^XM-^YR<jM-^[Ý<êÐßöfM-^@íòQ%M-^V~øç^FBM-^I^NM-^E
M-^Sx)¬M-^AM-^CW¹l&¥;^K^Y
çÈvÝ^H^NT^U^FÿÎM-^Ggu´6²^[^X^Pöôm:4Ø' Ö°P^M-^SÅ^PeÖëùíh^HM-^Z%u+î7k^Z6dhÐÍnA:ÐM-^WM-^W
yÛ7\$^?\$M-^E^Co^@îqIMèA&Õm:M-^9×M-^Z^RÕ2V«^Xb¥Í7;½M-^YM-^Pg.½HNÅM-^M{78g^_ ^SM-^H•ñM-^Oí^S
Å2â°¬áOM-^IÀ,M-^N)^XM-^I^G½^Û¶RAM-^U#çÓM-^GÝâM-^FF;^\\^T"¥ÎM-^M-EM-^Q^ÛAM-^ZR¬û

M-^I4I^HÈÈuM-^ZaîüM-^JÈĩ©±
¹^NVNRĐ³ŷ[M-^X£^DÀ-µM-^F^Hú|;?ŷM-^C□m¶Õİ

<M-^IM-^@©-íFM-^[p^A^1*^Nö
[Ů}øYÄ´°^FC^FÛdîM-^LC¿^Be½jÓŸâXM@Z«XÈë]-\M-^N: ^H^U1~M-^NM-^M-^E.M-^Ah´
+Ů^Wu}^VJCEââôâPQ½0ôGQM-^BM-^Q=ê½J#G^T ^Xy£.P-AAŸŸç^a^T´î^F»TM-^S\PM-^Jh1 AÑM-^Ru
M-^PlZE»M-^Yq İŮ^UeQĐM-^A\óM-^RhG,h M-^Fa\îP«CY±M-^Gh|aM-^M#npĚîM-^SVò9M-^SĐçei^@´çJ;^^"Ů
yM-^_eô^P;kÇ^\\?øyDXM-^Sn-GĐwIM-^N^SM-^NM-^U9M-^@]ià^H½+¬>yOô^PÛ^EU`ĚM-^IM-^RaD.^@ăudxi
M-^Jt,?1|M-^W^Q^Táq´3^\\Ÿ<,Vtód•0Đæ/^XM-^RšM-^_M-^L=ŸsB(êIfœe8Ă1 gR@ě´^OM-^OM-^[Ç
M-^H>cîøâ^Qüq^_M-^Où^S^Y°ĂŮÆĂkZ2ñ¹î²p¬ùM-^G Ÿ!Oİ•üăăăôáóM-^KŮà@OrŮ^W^W^OM-^T^OM-^AM-^Y¿Y0
á^TØM-^M-^M-^WÂðĎ^\\^Y´Sü@šM-^^á-øÍY!óŸcŸeĚ
ùđM-^07 koi7¼÷/d¹|M-^S^XÇõ{šŸdë,raŸM-^KM-^F¬ŸM-^C~ôĬî^\\Ÿă^YM-^W<ú´_û^C³pñðĚ~\Ů/đŮ^XĬĐ@æ2^
^\\½]Æ^NA´]Ůúš^[\ð%½-â\30X^Y[>Ă^AM-^Hđæ!¬^ÇM-^WM-^^M-^Wš#½eîÇ¿û~_0ă^^Yîo>yŮwM-^p^O^^^C^Sü
M-^Es@ĂM-^VAHçšİ=M-^J•0æM-^OŮ{çcũæ÷š~păf|M-^P^C0æšM-^Y°;àg?KŮM-^Tûš?Ů0ŸwŮĚİ<J^]~^W^KŸ:ðö
>_G,
^?M-^TŮĚ,^T?ăŮôø^[^K¿¼M-^BŮp^ODİçšk<^XX0f¼öSRŮ^SM-^YçM-^Cræİ,^W´Â@ŮŸ^U^SyM-^UqĂg^_M-^A
M-^J

p
-ÿðÂß_°ðM-^K^@~M-^C_ÿìßCï5ó^P>Pðïÿ8íWÛïM-^AOë1¼oþmM-^VgóÃ '0þN\ã^UMM-^^ûïM-^^x^C^N:÷^^Oó
M-^_Ö}p^Aóú(qEfÛ^T^_YèÉ%î^D3t9ÛM-^_rh^?M-^H/M-^cM-^XÃ|(á'?'%ó{y^SîÁM-^JÐó^P3»M-^^Â^KráïØz
pð^[oøóRøM-^TM-^S8µþÝæáÃ} ÈðkyXè+þñØ×<~Ûp-M-^FùÝç3¶ÝøÇ«#^?/áwÛ1=óíï¼M-^J|[ûîð÷S=|ÿµ°ð3°¼8
;M-^BàÛM-^8^O¶M-^D p/<M-^P^@^@ ^Pv{^@^@#2'4^@^@_@niëÛm-ÛÖÖÖ?µ[jM-^]kù°ïµM-^K
M-^V¶w>êÛZVu^OM-^HînsÛn{»×°µ[-Q-DXWM-^Q^Dà%IM-^LM-^@M-^DM-^M@M-^E^WM-^D^Aé^GQM-^^ÄúM-^EàC
M-^HdBO^OI M-^P^R(@J^F^@ffÀM-^J

^@î^A|M-^Rî½jó{÷Ÿ6@°§M-^SjâÅ M-^H«µd{M-^D^BU[kknnαfmuoIÖ¹ö}GÜÚk/{«Ö¶ÚÜäêÖÅM-^KÛSmdM-^K
M-^KM-^[odG^V^Q/^ÖM-^N^U^S>x^R^O`M-^F^@

Ǿ^F^T@M-^@~!^@;^@^@!^RDM-^Z+^@ë={ŷ÷°ěŷ~Ö,ÛËM-^VM-^S»Öf»±WJ7²M-^[5^S
S
^P^QÄ3ø ~Dü0ǒÑ^BM-^H^GûM-^OääM-^Tp^G^^î£}^YßCŸmM-^JÍøM-^^•M-^ZYâî54^^Öx08^^{JZe^B^Bh^_lh»
«^W\$öM-^Jô^N^±îî¼Í^NñJZu³M-^I•upîÝögó]¶Üí^E•M-^Lî^UpF^[uÛøâi?øè«jM-^AAM-^V,mËÄ@7b>#VUv
M-^\/±M-^R/ÄGÔ6à^DÄ•Ä=!ø#-î{OLçYçM-^IqM-^MM-^H¶làêø#M-^Xp±j|jáÄñT;,pM-^GæŸM-^^M-^]9±ÄnùVÖ}
jÝuçà^Â,^]'îî!{ÆªÛêi^GFñÛL/yQEYÄÛÜ¶LbXM-^G!ûM-^KM-^KaM-^[M^G^C=âM-^O/Y5U³EM-^[Lăî5{»_tşû
^[sM-^Oö1qf^Gîà^Wi÷M-^Z\W ^O,JÓ5B+î5 |DM-^Fo°ĐÎM-^M~UËM-^Oç æôù@^_üĂM-^MfăM-^Eþ,ö#
M-^]r71ð

~§^D^[^S6^[^ZM-^HM-^[^Zy6ül^H?&•¼ð^Pø5^PlHØ¿Î^P75ðløø^PÝH7Áð^Pø5^PlLø^HM-^K^Ynkäøð°!;M-^R
ëm-^QvÎ7dα^]ÍÖID)'#M-^\'Ph^Qs^P^TM-^L ¿^?!(AMM-^K^RrÆ8M-^E M-^H^DM-^E!(
AQ^HÄM-^S,M-^RiÊM-^Y^[qVIØÈ^]"öùGM-^B^Q4GÈCNÖM-^HM-^VhM-^LN! ("ÄM-^V^?M-^LN!Q^]^A2£DÃQQ~^Z
M-^BM-^Z^]Ý|^3^_bDlë^M-^ÖM-^MM-^T,+ßCFM-^VDM^R^RM-^QÄEM-^AG+^QQM-^Zé\$d©ód
M-^Mâ^TM-^B"^P^TM-^E C^DM^T"^@|3áµ¼&>MM-^OyçÛ^R12[M-^U%+#^Y¹HVÜM-^S=^R?"M-^LÈ9^R0çÖM-^Y3ó
^R³üäñ^R^A^Z&

M-^V{ç?íçïiM-^L^XÖì-¹§^YûÑ#M-^Z^_8\$32M-^KîÛË^V#^UM-^W¼M-^]páo^C2-Ñ8E,jÁÔ^FM-^]M-^FÛ
 •M-^L¹2ö|!ðQp^Dy£¼QpÀox7L^[-M-^MÛM-^FîüÂS^NM-^A^CääÔÆµM-^P@d¥à9İ<-^T~?ri^P¹M-^I5;@ðbM-^Xê
 ½M-^-[P6M-^KM-^S|Û^^àm-^QM-^KvöMß.çly9F>Sÿo²ët^UoA´5´^^ÝñM-^LM-^@;M-^E¾Æ;¼Ä.yM-^B^W9â^N^N
 M-^Eun7^Wò-M-^W¼A÷ø´^OyË^Yİ?¿;¿1M-^[ñÉ>xÖ´,àm-^NWú3^]ùÛÄ/Û/»µM-^YêÖ«ÍM-^Ná^]îx^P¼M-^Zpq^O
 ¼/^BÄ¨!zM-^R´M-^Ayª,^Z^EÐM-^EÝÄÿîEî^E^A×Ë^Z,^^TN÷Ö0ÛÜ¼)*^E´M-^W}M-^Xqx^V7ÇM-^TÛ^] ^GYÇ^Gøq
 M-^F^FM-^UúÛbÔÄpM-^LË^Tsm-^]ðM-^V§^\f´M-^UñmJ-V´ÊþÄkøª
 e«ðÄ?M-^\ß¨îZ£•/¼;»ñk]»Í_hús/M-^C^^[ßM-^Béf^K;¹×ñ{úM-^M-^UÄ±M-^Wðýû³*»~ð^YipM-^XÚM-^Zâ
 M-^Z;éÛÛ µääö`nÑ§ðM-^KM-^F´^EßzD^Bó^Z~M-^GM-^LÄÓ£À!M-^FS{ù^?ù^OM-^FË^^M-^Qz2ý6Ñ^Q
 ^N^K^-^]M-^G;^^RHM-^BÓ);÷M-^]O^FM-^KpêÛÄþL´1hã^OM-^_Ø¿qÄM-^BM-^@-^]\ëM-^Xl^@Ñé:M-^_äU9]|x
 M-^FDh^DÛ±ù^ »ä,
 q& xÄn^^^^3¶ÖÄM-^QÛìðñÈ^Wv¥M-^IVM-^MÑM-^J•M-^Hî´ùè^PäÄ¥^WM-^_w^S9M-^P^QÄM-^MM-^Q_iID
 ê?@M-^^»mM-^MÆ±M-^M•µM-^WÊ|^P½M-^^\³zy0ê
 þ/;ä^Z{,UM-^Q ;ñÖ^Zëê\ì<M-^]]ÿ¬ÛM-^YM-^YM-^Q7^Z³(m"îÖVç°jÛ°kç^HzM-^Y^BM-^FXK_ ÄM-^[^T
 M-^REM-^P-8Lð096§^Yam;1^SÄP¬´M-^SÖ;½M-^AçÓM-^HM-^E&ÛM-^XM-^O0M-^[9H9YXÛ2þÄM-^O5?¿pF^S-d^^
 ,¶12ª^Vw9Ëm^S^\5M-^K^KM-^Z^K+W&^FþçË^DM-^LUj¹ëÖ&eB{^U»0^?Ä-M-^_W.¬oU1 ðUM-^GdÄ^M-^PM-^[W£L
 ää±^Cç }ö0èF=^O±±ÛÎ

ÁM-^UÛpXRM-^G2LM-^T~^]_É^Q\MzM-^S\¬ØßM-^V'vÖ^G^BM-^P~M-^A!^Yn©^M-^MLM-^M,]DÆ^Yâ^Rø4^r'
83íqlíè^Zâ.Ñó-{M-^]6^[0YÇèM-^@\M-^W8?ì#\lÑaj%İ•İ^AØéM-^_M-^FM-^XM-^]
ÜW£"v[Ä^ÇM-^Uα}øõñM-^E\$ñfHÓ^B@ö!M-^M,M-^X÷M-^LSîiðit¹Óg^VS^N]Ê,^YxîM-^C/^ZÀ^M-^NÞ|õ^S
α¬Ö^W4íİ[A^S^BM-^L£³éα»uXEM-^M-^Z's*è^SÃ©M-^[7^[]•Ä^Y÷Cx¥7+Ö-°{^K°M-^X,Ö\$B.^[â~\êM-^U>
#T^Sð:nM-^KùÔ1H^_ôM-^@M-^Wv_^A¿Đ;-4M-^]~fÊ^Nμ-^OM-^UM@ÝêÄs^RM-^]Ç°

Ů:ô¹^Pªxu^ZŸ<´M-^@İ;Q;(M-^KĤ4ê^]Đpİ;Ê^RM-^WŪM-^C@^M-^GzPM-^@^TM-^W"→*æM-^@^VôM-^]~Ůα
/N©/h^D¨ÈZ^AR2İÄe^Csð:^\^FuA^W ÂystÔM-^C°h|GêèM-^LM-^P^Oó¹M-^\.Ōİ}M-^Y\;M-^CfM-^GM-^Uê: ^U
|ŮæM-^@NcNZv^[^RM-^KzZó8^\¹^H=I\$^_M-^TîJM-^FO3M-^GíDç;ĤPo^\ÈdôfZM-^ ^û%M-^K/á´ô<tM-^SM-^Ih
<^VwH^X;P^B\rM-^Båæð; * !iM-^MÖM-^N^@wg0Dâ| %M-^M^DñíTM-^]Mç+īª.úM-^DM-^F³ M-^ ^d,*ī,:]ª1
M-^GÊÄĜŮ^P©*d_M-^@g¬ÄŌĀÇsM-^Q÷^]g^©TÔŌp M-^XSM-^QYö^K^H±2WđŌ´°M-^LM-^KM-^[/M-^Wb"Ā@-13©İ¹
RŸä^OcM-^Tαâ^H~øŌ^_úE®)^A|D-î^Pİ óOp^NM-^\m3M-^JM-^XRİ&ùAcç@û©ĤöM-^FİM-^XM-^R«wçç
^DŹ¨¨*`b±"M-^M-^M×M-^]; ^C; ^H/^GWpμ×, | ÷³i^Kçñ^Cp)=Dgsø#íŸ

kâ0C;^SêÝ1^GûóŁM-^RÑ|QM-^[

M-^\\ðÁM-^K¶äô\$
Qy@u²>PjM-^P^KÕÈóM-^@-öòM-^I´^?ðMþ^@P´êíaôøìBîr+^Nè;M-^I^UM-^Aþ¨0÷xöU>M-^L^_~M-^P;;~P?´.«
Õ`XÛM-^@^^
EBI}çPM-^KM-^YM-^AòSâ)³M-^]ë^TM-^C`|Qgvþj^Z"âíwM-^SðM-^MGM-^AM-^X@y;Ó^]V£´tiðu´ó^]aÕÄaßiñ
M-^U9^^, _tz\$^[^Z¿w^Zdô^@½Yì^^ÄèM-^W@vPzðê^K^Y¥ðþM-^ZLkM-^JÜĐçã6KM-^\\M-^MkM-^G£ýM-^Qâ¥g«
fÉëÖÝÝM-^N>ã½M-^Xûh÷îfM-^\\ø\$ð-fM-^DX5^\\öM-^F¥_
jM-^A^°Voð^GúKñ^Hn^E+T×År^^ð^OM-^VCy«uàm-^_M-^^Èvı¹««M-^^½jÝª]Êei*]ÜbìÝM-^A´^Fëç^AM-^U?eð
ßĐ£|ÄM-^CM-^W3K°©, æèCM-^Võñ^C•îM-^[^PÛGÕ^@\$EM-^RB1M-^Vè"êþ^S4:pCD¹½kĒd^Gãm-^BZ, ÷ð_M-^^^C
M-^Yæ&´^TM-^GM-^\\;ð^RxM-^Uê"Û^Tj;2M-^RM-^V¶ĒĒé9M-^UM-^YðM-^YM-^\\ěðæ£ðM-^R}îY^Yî©Ô^Pă-Õ

sM-^CPM-^DM-^V,twòfM-^XM-^ZÄ:~U±{ÄM-^C}{ãÑAP,ÝM-^LwM-^XM-^UM-^_<èîM-^Kz^UèLwÖM-^?M-^L
M-^ZÇÚ3gðkGiM-^OBQ\û]Úñ\¼î^^áÄîè(ÛßR•İDD;^DÄ^U tM-^SM-^KHS5^E~y|M-^LÝM-^@î~^]M-^VéhÄÖ{Ç~
eâ&M-^XøM-^R26ÿÈa~ðM-^G-|Hùoi6=K^SM-^N^^pøîs7'íîM-^O^Tú^^\$XM-^]È5V]AZ^BQáöa³ÖJM-^XdM-^EA¥
öM-^GM-^XM-^ZM-^]5^?^Z(&H^Tİ+H°o&öæM-^Tİµ9'dM-^RîM-^!Ö}}QM-^~^~^YM-^IM-^~M-^B½M-^KaM-^Bj
FUv£^]¥~î^@pî^?XE²HÇP»M-^G9\èîM-^]M-^GöM-^TÛSÔ(MÚ;W^X6ÆM-^[{M-^HM-^]ÉOfM-^Bö•M-^IM-^KÖM^D
Ñ,Ä~xçûB^QâùÄ^SV^Y
M-^CÖ?OM-^G-8;çêPM-^YY' Ý+ù^QWM-^_x^CM-^Vçd÷xç)Ä^[Çà^RM-^JM-^X.şÅ^T°l«M-^XÍêî^Düüm×M-^T5È
É½¶~ÈQ{¼UmEðsı~Kræ8M-^VéçÈÈhMçgÝ¹>RKü'~Kmé;da^Cu^FJ^[Ää^BæM-^K(
"M-^OÖôÊ/dM-^BM-^GSd;¼ÈîM-^[«ö¶eláaM-^O;^@M-^DM-^E M-^WðsáÝP_x|M-^J©^E{÷K¶^KPQÚ9~³ø%îÁîXS
M-^D2Hde!M-^~^GM-^[b ñ^KM-^EîÜOeÍ(@µ¹'M-^~^j^UZ,owÓwîl5pGM-^Fb½î'öÑîñ~ñ±^~^ä^~ç3äM-^MS\$ÖF
^Vkö> ö- ^YM-^YXîC°wM-^]¹^H°lpM-^HßWð£^K*è}ä¶ö±ßçM-^~^XM-^Q2øµûî|x)I[Æ\$^Fu:1Ö
^Pô^Pô^Aâ~M-^N¾^D2î^]^^C^TÝdÄ' M-^L#c:¹M-^_sPZóÐDrxz^^^EËqM-^~M-^F_H^P÷"ÇM-^PM-^NÝM-^[j
M-^LS)A5JÝä'
ŪáCM-^CíM-^UÈ[S?^W^V^AFnx=xöðÛ"~V^O^C]>v¼ä
fûx,ÁîÄ^Epz^G]M-^J'÷°i+M-^D££²³£\çM-^Oøi^S(î³çoc²M-^Fi^H.|{³ß4;İ+ÁßW~şÈBi»rª0tµUM-^Rbî*~
LM-^H5^Q-¥^AîXC×ßO,679D5;g¥^]M-^H 9ää÷\$ÆM-^Q6-M-^E^HµM-^WØv
Ñ)ë+Æ@ M-^L£M-^GÉM-^R~^F~¥w¥íøÄêóG{^]Ö&Eø~MurOu0
ÊlX^XjZPiM-^Jýd_~lgý

«mßÖîM-^IÄS:Ü¥M-^P²eİ;^Q^ECø\$Â^HM-^X^X"M-^M-^PIíM-^R[iíIlil@n9oM-^BÊM-^H^EM-^IoM-^LV^Sab
M-^NâãÅ?Èd{M-^DÛ^P=LM-^VØu\$lfWM-^T-W{M-^Hùl#^W.JA.#î\$Ã×/ì÷ífM-^IirOë*

-êM-^XuÛµøæÑÊ¶^^ðì4: + ^GM-^AÖRRM-^EÝÌ*xH^FAM-^AÌ¹M-^@1M-^O;M-^UÝ^V«+=^]Û
 ^RÂ! .>M-^TÇøç,¾½'M-^V^Bæmö\$Å^^^R¬xíùaÌ^\ÿ0ÄÐÈ"Ì^QM-^DT ¶^O ûfO\$+ÈPÄG x -á^Eh^R^ZûM-^E0\$`
 Í%ÀÄ³fæq^AûSαM-^KM-^RÂùQ^WoÐ^^-CÝ@¶^TÚM-^HKM-^NCM-^è
 3C
 G^NFæÄWxM-^@yp÷!M-^Xf^D[«»'j']F údªÕM-^BE?q^]Ý¹Çää±eM-^@x°y prÿóM-^D¾äÇ.ùaKÉM-^S-M-^C
 ÄM-^EÖXBªÛM-^Ce^P^?éÓÃ|cÖq¶u;þfîa»a°ü]Q*^G2_M-^@PqB1.PM-^Z¶¶êPiç`""ÝcM-^Yø8^Yß^U°ç^P^Hf^GÄu
 \$VÉ>V0â:^[s^yZûTË^[û=M-^W0Yu^R\cmM-^CµBñM-^MÖ^VM-^HÛ^Qv>ðM-^Ckñ4u;^V5íEÜËM-^QÄ^Dâ¾^W<Û0^^
 ÊM-^C#[)ªgT^SÔM-^R^CèÛM-^E¶«M-^D{^K^KÔM-^Z àÀ.e/9÷ØM-^F|Äv^Nö_^Dµ8!^CªY}ÄÑ^Tû^A; \bíM-^B
 ^Q*Û0"µ#LîM-^AM-^\/{àªM-^RÝfÛD^[M-^PM-^UM-^Eq µløääÄKZää-M-^CÏR")ø^S^Wääe[^P^F^F^?M-^R
 ^\8[M-^A=^D~zaThB»ÖZα]à^^ ¿Ä0XEÄ^G^PæJwp3eW\M-^TM-^VÐM-^UËªáo-`^X#(ª,þæâ_ ;ÛÄ>c^^@úQ
 UÝ*èé^@µç2X^T<_´M-^B&¹cM-^Y^DM-^NØ¹¬tÝðFM-^E47°ÖdYö6Ä nÌª±¿#Äà,mÇM-^X^R^U´çv^EÌ^^zQHÍ[
 ^A^@M-^}\þ§Ó^U1¾Ä^A^HM-^OËM-^@5j%7ààm-^QÎ4XuM-^IÍÄ^SäM-^DîeM-^Y?c^BM-^EÇ´^N.èÊM-^Mz1^GÄ^W
 ôSM-^OM-^Sþ^PαF"ÄM-^AM-^KM-^NÄ_ (?ëäó
 à
 M-^BâcFM-^Tr^Oâ! ;M-^W•:αM-^PB+Nôàm-^S£P^M-^J*^WÐ^ \-øM-^XAM-^}r^X³^Gj^_ìÝM-^@M-^V;{M-^NÉþ
 äs+0M-^Xífm_ØM-^Fs
 î@æ±`v^@+á*µM| |ZM-^CÐÓ^Q^PóeÖ^A^[5M-^@N^F®0+

ã×û¬ûb

cM-^NM-^B³Ä;^F°l;T^K^_YxM-^Y^C/d¹M-^ZPM-^Aw,Ö¶]P^XÚ^AÈö]!M-^@ém-^N,M-^@M-^FYëM-^M^_HÆ^Vùt
^VÂWÖ^BM-^DñDMP#5U-M-^G»¶|_DM-^IÇøM-^E,M ^_:^VM-^Kñs İü^Dz§«\$]_ý ^NM-^GáÄe^S\M-^²^Y`M-^V
~Eì^E`M-^UαÑ<s?,sl?ãlç^a´ôëaM-^C³æM-^OM-^Bc®^C«M-^HM-^Jv^C^Q799£7•
“4^R×İr§ÖEİ@7V"M-^Sü^Bíá^Çö×-Ö÷ì^^½M-^F«^GJÖM-^K^FMaDù£l´M-^V^AöL°HÈ*ÝÑÄs^DM-^]çsÑ½^@[^RŮ
^?2^C {B`í^KŮ´.M-^WöÖj´9^^ü,^AB^V@O^Oþ^R^P~ÂöM-^V^U^M-^^YRG7
ý^X+^@M-^[M-^P»&iþe^Q71é³^[JM-^GVP<qİM-^L^X@w´öäfÜÄç¥_ê!Q~ y;8^C°8.M-^QM-^V7M-^G^^êM-^NüF
Đ^X2Y©|=ò¬^Y\uÅ^AÖ±^OĐM-^[£^aM-^P:9g¹M-^FÖ^C+HJ^XgþrM-^BM-^C,-M-^SaHM-^TÉ°Úýy©#^FM-^GE^@
^@•ñ{¬ö^B?ÄÄ^Esα-YxM-^ORM-^BQ^¼F^EM-^AeðM-^H
s*ö^[•M-^K^PgM-^L@M-^E,^@Æ^s^E^a; ôËËÛİ^Q§^ARÔ"z^FêÇ_aM-^^m

IdËnêM-^Ye`^^^ô^PZ^U*M-^KEEp.^Bç&^?ñôWgê
M-^^àüî@iô9^M-^Y:K|)o&\$?ô ¶Jp»^Qj?dÖ1ÃØQ@^_*ííÂûT©)9 =ÅTñûø°ÈmÂ^_s:,^_Óâ=ðvM-^P^H1\$
j=^HãÈ-çABM-^Ji4 ÁpÐªb`M-^YÑJ{ÖM-^T^Y;9ð^@Æ«wX^U°téÍfQM-^E^Q`lM-^FM-^AbM-^_RË9àCM-^M&ßß3í
ÿì»M-^QËäý^S M-^OM-^ZM-^J+Q`É^@ n»øI+nËæ^Fè9M-^O±^FM-^U> {EözM-^LHNLpßOïï3mv^O^Rg
^S^EM-^]ùÃSã9a^F£çM-^FÈ10M-^M-^WÆ£•G~^Pq±M-^OÃM-^R.´¤æ^QM-^MM-^GM-^]M-^MY@ííP^NöM-^P
M-^^'Wz ãqÜçM-^B]n¼öM-^@1ã^BÄÜë«êçM-^KX`#^ãÖ^YM-^XM-^ZíËM-^BéQ^^x<ñmM-^TC•°^T^T^-
XM-^@f^F`Q9^UsAM-^G

XaM-^T^YÍ^Gµ-c¿Ýð

ĚDM-^E-M-^A° }Ø,M-^WǺÓĐú60Øà^C-£É[M-^PM-^O^1`¶^FÖYÎM-^X□×£ù

æŃM-^T^Fî>¶vM-^CM-^D^^qM-^S5XøØc/§É["M-^[m7°îÄîr^WøpM-^@^I.5ßîö,ÂM-^_Ö|SÚM-^Rö^CAÃô-àaMÎ
M-^HÛ0`!°_^\3O»ô9M-^_±Ÿ^EM-^BG M-^B ^GQëëM-^GóM-^WóŌú)M-^C¿:ù'«^PjG^?äTføiAf°j'5 ù5><Ã
M-^HTSWÆ^B»£ÆM-^@^^2^SŸX-Èé)öŸiHk*M-^@äZp'eMùM-^VS-M-^K3aø§'I ^W^G[^[3 ^W^M-^FuÃó^]:LM-^M
μ
M-^F ~,2İM-^MŸM-^^¥ăŮ^F;_M-^D8jò

ô^CM-^MÛ^Fàn^GpóØKM-^OM-^S.^Fx^Ypn;^İ
|¼²ÆM-^C^CM-^O^YÉ^NÉM-^YóN´^U^K^FYôpð ĄcĐv1Ůoâ@oêðFM-^U^RŎĐŸ*ØTL(sFO^[5ûJ^CøM-^V &kM-^_P3
Êë^\RNŸ@xH¨b^EM-^VM-^CŸúQ^SM-^H0e>JéÇ^F²3M-^VM-^XÍM-^KUđ^N2c^@h^Uău
*ªM-^CefmÄ2ŮN

äóî^[9F^P!•fM-^Y^^HæM-^N@>-^QöeÿK\$ M-^@Xö^YÜM-^[^QyI
7qü^Xým ùÿÊ6!Ô7M-^Rt}jw ^@.d½^U¼^R'pñQfwô N%*S^[``ç7X^YM-^Y^VM-^V^[çf ÍM-^_b;x^HM-^MiÒG=^T
dpï"M-^Y6{CM-^YQâiP^?^FTczM-^QM-^JM-^T`Á|õ^X)M-^XM-^UáM-^[LeÃ³^EÇ

mp

^YiÖM-^EÄb^C,×1³^N#M-^Fç.c;ö^HfÄ{>è^F^P^Y ³¥ª^_M-^@M-^O^ZÜ½^R^[Öµ4,'^D@[%ððh^P^?óy
M-^Aª<^@^^V^\\YöCÉ
M-^IgM-^JI5ãV-p~M-^L^K Òß9ªº«ºî>|¨^T^?OM-^Ué7Ü.ñãt^Dæí/»!M-^X|M-^Y^-^NÜ_2M-^FHXÌ W'5Ô^Zø
1ÓýVðªç
ÖM-^_¶I^?M-^PñgÖM-^W|Ô:~V-^^YçM-^\\M-^V,wM-^B^T^Gw M-^U4M-^LDÓİgT,º¼º10 ç^H^YM-^BM-^^M-^^
¼Ñ^F\$M-^DZ"M-^U,ÁQCRê°M-^Vfİİs ©M-^Fbªª^WøçB^^åM-^U~Ö.<êðém-^A¿Hú\L^O^Aüý4coÈûM-^AûM-^_ù
À M-^A'Å^Wå^TÍÀM-^_YM-^Iû£^Hf^@@M^]~4ºÄEÖDò¶TD^?Öèÿ"CM-^D+^_M-^XB:'Üám-^G;ÿp-Ö^G]Ýðám-^N
;ÛpM-^G^]î,Ã^NwÜáî?îîd•³ÿ3^DgyÖ}¹^Eá÷^CM-^Zîéßÿ-M-^XiÖ^GÊß^YİM-^NÍTëA^Yðf^HF^]ídivwå

²üð¬¹p×êzÂSİðQa7Û8M-^[M-^Opü2V¿©ùú¹^OM-^WÇ M-^Z^NM-^UæùÊ´M-^GMM-^W²ª^W!JÐ²b)²!)ÑâBPM-^Q6
M-^[ϣulM-^H
µ¹P(^["b-"M-^V-^[M-^Trn.b3ib^Sª|fM-^[Pl^X^SJSB§È*ÕÔM-^D+^HÆÿMM-^LQ M-^ER@M-^DQ^P55C
^Dã{^[ÉKM-^_5¶X(©Jg^E)M-^_M-^F(TÍM-^ER^UÆ u6;
Rf´M-^ZP°^RDATA
^QYØ(¥OXA^Dn]"M-^Eß^[-%¨ ðJçM-^U^\;M-^D"#,ÜM-^\Q(+?«öM-^Hi_ØáªRSÄM-^)|@ëÆ ZT(Ê?-n@¨W@p/36
h=@Mp^VÜM-^EôFM-^¨ÐI&³^PM-^NM-^FiM-^H76^R^_M-^Z¾^Vreê6ÅÝ0¹M-^L%&M-^W!Ä4Fî^S]¿^SuM-^YOLf^F
M-^XÄ\$M-^_Ûe<1M-^]A1±38^T6ñÄ^?M-^FM-^T2à\$%p^WM-^N#^XÈ^RM-^XM-^XX)#
Ôï^Ho^_Q`M-^Xi^[ÊRÂoÉmàM-^I³^DM-^¨&ôvMæM-^Q»M-^Gÿ2^Pgö^X#Ó{-´^R^DÔØðv;AI;Që¹^D½´ÈÚÓ&ùð 1»
M-^[ëK^DÄ´¿¶M-^_NM-^P§^Aðv^S◊M-^Zã^[o^DyM-^Zà[P´M-^H´Î[¿´È´ÖÜ¿´^H4Ú;OM-^Pi¿¹M-^@M^Pª=,óM
^P@iî^U]M-^BM-^QM-^Z^HÝÂ´\$µM-^P°M-^HOM-^Aj yMM-^T«W^D±-î½WM-^BÄM-^K| ^VTAä´?|ÖËÜM-^O´Ýjääî
^PMM-^DàÉð<;M-^[pD¿?iðM-^Dcâ

^U^X@Yy^EjM^B^SÈM^-[7VÂL^V+BMM^R(^TM^SDIM^XM^TJç¤\$^RÂ%Q(&M^IM^R1)M^TDII\$M^JJçQL
^R%bR(M^IM^SM^RI^TM^TDçM^X\$JÂ¤Q^R&%M^R(< ;M^[,DÃ;ù@/âM^IbÃ^SSDçM^Sbç^SçM^D' &^R
M^M^O^NM^EO Ç¿?/ü+?^C|ûø}^RM^-[P÷b^PÈM^D^F2M^Rp5^w^G^@^@SãXCc^Z^R ^F^CMöD^Z^Zðnø
=M^PM^M^M^-[?cw6^øðM^Lîð^VpßM^F^]dç,ûM^BîM^OHÚW+^YM^_Ñ^_Jlv±ðHñ¶VM^Q>,Jñ[<y,,kúi×
ÔM^RÎÛo[çM^MM^V^WfÉM^U*#³ª³Y"8,çfM^U|1",V\M^FªªM^F0-ó^[ã~^\\^H 8F^XtkóÍM^PÃ^Z^E
^D?,!~^V\$ Ñ¤T5^T5R]iîÁr@s^]PUß]sßIèM^_)M^RM^FîUß@îM^SÚ4à@ªc"i^ZðîûM^-[M^\\M^U
S^Y^Hî{îÛÂqçóÿzÃÿ üÄèÛM^XÀM^_YgM^M^@^A^T^U^A^@^@^Q
G^AAM^TcM^HÃÃb^XOM^Lùª*«¼I«Û^WîißAðQ×^EM^RæîûA\$9@k@è-z#1ó]p^R3õ|srÛM^GúÿçFõÉùoµçà
Ã9>^FEM^IÃ×M^Pá@]-ÛXðÈ^GÕHÈM^S!p^Eßiß^Cð^C²^[M^H¼^R^A¤=^[H"^[´Ó¼M^@?^ø,xù^E óæÈM^Vð
nÂi[M^-[M^YrDMUiðèðóú»ÄO¼sEîñÑ;ù«ðíÝçãçPù*ë|i°)¹)Ç»J
}^Uá¼M^_0^U½^û
ª¹çbîRoÖ\$øÈR?WûiA^E^G]sA^Wú.ßi°|<¤0<!ÃptE^O=M^FVü?Ñ^Bÿ^?q=
ÆvÄ¼Ñ-M^@^D^Nm±M^HEøQhlùM^PÃs,XM^G^WÃÃ>ªM^LM^EkXÑ^[<M^FM^CM^H;IWqú;¶êGzêÖM^XXa
.\\!C•"9M^JM^N^N¼M^Q~°^[6^BwæM^L8^EM^ª,m^DVð^Q^_Q^QC\$SÈM^QM^S»ðÃM^VM^OXM^MÓEçgb
^[?áw-X^wçð-M^Oÿ{^DîKWqú ½M^-[@?|k\$@M^H\$M^__{bôcM^EM^Q1ZQ'M^Ln-Ã7"îQW9ôè¹^E^]@^_ÛÛ
^_fääSÃ^C«@×ªpM^_]v~';M^JËßÖM^Y,~?&ø@T««M^@M^@^@?^?Jj<|j^OCM^OSYGM¹»ÜÖ Jè¼îM^A';ßö
M^_]b]^O<dnß_ø'~ãM^A;CdÖÛ³Ní?OÖM^_ø^G
½ÖM^D<²^WM^Rëj^CM^P²~pêÉ^K
M^UShÑ9µM^VlSKHM^Wò],r~+îëö#avî^Q*-!î½;M^Y¹M^DM^DM^VÔM^EM^Eæô0M^R,{M^Aß^Döæi¹s
òP;-f^TçM^_] \$#^^•DÖM^_j^RM^_j:RM^Ljn]M^W°|æó~^?ÂSD^@U#M^R^BUûM^QM^_]U%IîªM^H\$gô
5ã^DZÛM^O
--I^RTM^MHM^JVø3îZ²öÖM^C(ûíH*TíI2T-I:TM^MHM^NV~ÿ0ô²^C²î}ùBLí3JTíIM^QV;½¥SMIM^Jª¼öS
•í|),M^_ÿögm^E%öèM^A^YM^V^]ß×½ã;M^KU;}-L¼X^^^M^Q9ç05ö-W 7k¶/Vèð^KÃgÿ/sM^_]PöM^_][ß
M^Znû^Eíäi±î|M^Iö
¶M^Oùjÿ;*ø÷ç9M^FÖM^Q6ÿPëo2svÖÛ9nîûi|k^Cx--~ÿÛ~ÆM^_]»°v~í.M^XM^J^[,²jzôãm^EMkW,hH©
^N|ZP^VäXC²^XjA8¼ð!*+-Ë*M^KbvM^EcM^FÛ^M^N^G4Neê É2ª%êÈÀ^R#kBÖH³^N^CÔM^R<> 9-çÇou^@fE
RF@M^TM^_]Ó^N M^-[M^Q^TM^V#ü#³M^Va~,?#^EÀl£5rßM^YÚ'Ýãm^@Bc^]NñM^HXÖÈÛ^NéM^O:3>r0
æ¹]-~ 9í2pM^IH³»M^GZLM^ª¼îaVâM^NL°^^(-ªCö^QM^HM^VM^PM^_]sk5_ã'H^H^t¶ÓM^_]í)ûô>
él'[^_M^Eî@qmM^N^M^RHQY"i!ÿÄmó!êr¶^NíLip'^[0M^G(M^X¹ÖM^IM^WîÃ;-Zé÷M
M^_]M^B~sæM^NGödiM^PM^C^Xé~²ZÛ-M^F{Ô"M^I/s^Y'awq(¼)°dM^Pø1¹'[P^

#©8^ZM-^]ÖM-^FM-^Eİ^[^\\^1M-^_6.P2æaíà^^GØ²M-^SM-^N

ã11^SdM-^@Å-ª346\\^@-M-^O°SD^^Ê~j^K©f,î*eİçd^GiÛ^Av¶M-^GÅÛM-^EeM-^HÂ^?i"^[

İH•M-^PuF

o^XM-^Sp®^BİwM-^Im^_'-F|dìĐa%j

ıÝ«<PÉQ0Ø

e_æ'p^NÌM-^Yă^\ÿM-^ZÆM-^K²â^OÍª¼M-^Q^_b¨M-^J§òL66l',°<^*^C}M-^O^QM-^O¨à¾M-^Oö
ã^-^Gî`§U^TIDÚ\9ÿM-^Z<ÜÄ¼^WÎ£M-^JM°«^UM-^VM-^Cæ^[1øvqÿrPä»ĐÛÅgóİîÊu;-^BXY:^^[)ê%çBcí^T,ê
M-^T|Tî#ÛJjM-^V,â^D^TOêñ¿^Kv^WM-^WsØ^WĚ1^?!M-^TTÑ³M-^G):,^X°
ý\3ö^NĐ>8iIM-^P^BM-^F*ÿúr^Q]ă^_xvM-^E•MĚqPM-^PÖM-^D'pš^W^Nm)¿^?FÉM-^Vv^DI
7p<^Q¶iMßw^??İÛßf</»M-^F^B•M'M-^G²âM-^W
FM-^S2VC¹=M-^BM-^Q^M-^P-â³M-^X°Ç^Z^^û

W,M-^D^\ùúÉM-^_XñÅL|áëø^KÂÉAM-^_ ;ÊÈ^\M-^\M-^T«^GDîu?M-^N^?M^C/^NpP^QüM-^[. ^Hôâ ^OM-^@|
M-^IM-^P¬ Þ¤|M-^G]^^\bâ@^D^S^HöQǎÝM-^V]`â*¥Y~M-^CÔ^NǺĬó3M-^W
¬M-^LM-^F-6^\^@ÚöðM-^H½M-^EvHÆáôâÞÝ/%foH^H"zæÞçÝ^PM-^Z^HL:d{áiÉçM-^LcěĂĬ^HM-^L%|JmzǺHM-^A
\$d/M-^MJ\$YpV¿U+ZM-^S¬-M-^ES^D
^YîM-^MýÝíñ9 ^YôâÐÐÀ×ö§'W^H 7M-^\^_M-^Jc|M-^]NM-^H^YÆ
úKâH5:-³soÇmà àÔ®!£]²¼¬JúM-^K^Zî%âM-^F -4M-^ZÛ@¹-(®×

^]M-^Jy^]5M-^QM-^OŸ^U^@M-^U©M-^D°p>^Zî,åM;ĂÆM-^WŸ^OÎM-^CÚĐH^FaM-^CØ^Aâ§^^^S<¹M-^JÑ

^@âM-^DFĐQ#8Ë%%M-^YŒ^G-^Y

M-^Vuía[M-^Z
Zd³^^Î¥©DÄ¿k*ÄW":à~NŪÝLM-^W&ÀËTRÐÓ\M-^Cë^K^S\$E\$ûC_.:&Ä(3!^[^RpÃgcEq^B!^U1"Y^YÚÂGQP^K^D

ÄäM-^I^\$ ^Gü^CãTKUö^_!KM-^Rn.dp^XÂ±J^YyêM-^M²èB^]ÖM-^WaKM-^SR¹^îFV)-ÇËÁ1^^\-f)0IãM-^UM-^OL
M-^V%ZM-^YM-^M-^]É^A:Û»^YÈágÉe)a^V^P¿bÇM-^HÜ¼1¿M-^N^D[^R°^Y×^G28._M-^R;^Z';Ä^NRzYÛM-^ELÑ
N^Z^OM-^U

©^BZÎM-^RæM-^VÔ) -wpÅ3©: -oM-^ \ ð ^ \ M-^ YÖêM-^TD; ^A¹ / Ó ^B" \ M-^TÃ 'M-^KYçí?M-^^^GËM-^KóM-^QúNßU6p
:t 'É^Gt; Ð, M-^RGM-^OÄ5M-^V@±@Xù1M-^@L} éM-^EoM-^S^D' èµTÿ| vM-^YðS× 83/kmæM-^[, M-^YÄM-^ZÛ£6tQ
Ô9M-^J, ë@M-^GM-^ [øþM-^@^YTÖÊ°0Úþ^G^D, ®*çM-^_¼{1v^?ó5M-^PNRnã^C×çø3 %1ø-ïl. ^] èñ^ ^HσùÖXT a
¾: ^W*T{ ^Xd°S
^O^PM-^F^W÷M-^JM-^R÷M-^X/ ¾¼ÜO@û; Ô^Vé] Ä«Ó^E^Z¶Î×<j/q^_M-^] Ä^O[¿öÈM-^@À(, M-^FVM-^Sp>n1^T-^G
àm-^_ ^WÖFÔÛ M-^T°á• ^V^GCC' ^?RM-^ [^ [°ëÝM-^PÝ^K»M-^C\$Luò1M-^RÎ¼M2ä^Q^YM-^LM-^U<ÄB^] M-^ [öùF
' | äyM-^JH! ^_i©

BñîHX[
GA^U¶
1^]α ñM-^Jlt@HÌM-^M#^RÉM-^Fiç^S8^PA;3^AM-^Fî^[^XĂM-^Qt<Jh!M-^GçM-^BØM-^_î^¼÷Ĥ@È-^YM-^I^PO
#sM-^F^Tî &î^]WîpM-^î^x^YM-^_pM-^R!×
αÓ\$÷"!M-^SM-^Xiî^^^B×£)^-û^q¼wM-^SĐt"@ĤM-^WM-^GS^[^YM-^A¶gðÖ(!I^?°^K
h8M-^Bú^U^?pm^\%'²2^AĐc#nIU¶Â^UM-^YFÛ\ûêM-^FM-^D'~Ó>rM-^U"KM-^Z^[^R/ÿ=^[²Ü°)öX-Â°¿7ÇM-^G
A°p\$đo]^GM-^VPM-^E°£@^Xô^N^EUÁM-^V ^G#Ô~^E;^B\$bM-^B^^M-^]"-ăĂÔM-^Aα^FîbbÛi^T"lM-^R«
M-^_)^NµÛ3WóOÛ"~

ı *Ö{ùŰ<"ùéB_4I"ØM-^RM-^Xa\=-u^WëzÄ^Q~^O[^°Gò<M-^Cı °M-^Cbàé0ŸÖM-^ZJgŸăĂÈGM-^XVWEçİá^ZM-^Aù
^A^[>ĐM-^NM-^DkM-^U/[z1Űà²PèÈ'ìM-^Jı^O-^c^Gı^Fxcđ±M-^YM-^W:M-^C^aM-^R55wM-^W^XM-^@²+μL^a]
M-^LÖy!úáyâ^K'ßù>M-^OX2/ŌŸx^DhmĤ":İ^S\^-İîM-^GúQ^NM-^R4"ŰKHBz"μ-èz¼¶İ^YbkıM-^Q\$î^HxPTuM-^J
M-^BM-^[^Ma•İ;M-^MĂ^Rù^CM-^SAP^QM-^XI 3.^X^B^]M-^UmŰ ŰİÇæ¼'9-êĐ^D^C*á^O(½QM-^U^_jı^B
M-^N^^k>C^CM-^QVc^S»^KQ^Vî0³pŌŰM-^HS''-£K:"PáxóªØ6PlUe□^AOß³|"épªűM-^U³d%|t|běí"fSM-^G^]ò
qŸ^_ĂM-^YŰ ^TùŰTáM-^Fùù°_EAM-^Q'^Np×^_^O
íXo°I

|M-^FsÄØ^Nù^CCM-^^5M-^LFy^?İM-^Md÷2M-^R]ß?¹GY3İû^\ıı^EuûAM-^DM-^Rt^Dö^U^Zë2^N½=Ä•©IZ2)+\$g
:óÔM-^^VZM-^GZ\$'¼hðxĚûÔM-^W0<^]ıt5ök^UM-^@2:M-^]®M-^CM-^Z8M-^GöI

AM-^MFO^?0[^Psd^DM-^Rv^WÓ| 4èxðÈ^\^^Ým^G9^Wî-rK^?ù^^^Zðn/ÉM-^P)èM-^F¥Wu¾³M-^ü9vàm-^CÿG^Oð
¼^_D©Ñ-üM-^F@^D^U, 7^^íM-^UW^Pj^FM-^SyÌé^ZM-^Brñÿä^_^]Íç^PÍM-^NM-^R

M-^IôSô´8M-^F2Â^V

=sM-^C"i^O;^2íä×"ÑB{ôM-^J;M-^SĖĖĖĖ(bÊÓ^U^G
M-^^>+G>M-^T¼u%fM-^XFĀ" BŮM-^H^ES^SM-^D&H[M-^U\MM-^^°N¼3è^Z>^YIú|úQ^ZC#Ñ°5M-^B^aG°k^2^V¶p~
M-^_[M-^^\^Z^_{@Í+µ õð.Đ^M-^XMĪ,%#p7VM-^LQ^O^V^DÍhM-^VJ|ÁyĀăô&uM-^D&,-îy7Ī^Z^2;^2ð5=M-^}yâ
LIM-^\a^Z|^Đúû=ds¶M-^E(^Pëx^NÊŮ

,^S^Aw\^e}gËxBQuu´ÖM-^B, ,M-^R^@>»ÿ=^TÓ!¹M-^K59^Dôï5^PuM-^WM-^DH´:bM-^Z#4³àí^TEö% ^B_È¹M-^E=
Zäm=M-^D^K^PuM-^DM-^UM-^AöÆM-^CÆËP|-^Hád^K½+®|¶Ó»HŮÍc
WÇ&V2\$¼ÉM-^W?M-^_Ë@ÛëÇM-^YÓ^C^[^ [u^ZE\Á/^CO0°M-^NŸ:M-^O×¶jç7NiÖi4^?ªd3²Ú´¹¼M-^NM-^M-^F
M-^^s]Ö ^^^HÆ¿Ÿo±^,\ÊJW^N,7M-^P VÂÎ^O:İÄ^OŸ´H(RM-^C8~M-^Mİ÷PÎdî^S´M-^TM-^NÊM-^G^BW-ëë#Ëhf
XfM-^Pçffgø^\x«^YM-^YU!û_üÜV^NeÖ>M-^J12M-^LgNÂM-^P^V/ãa³®BjÈM-^Afdß°ßM-^Pcâ¥äjm-^ZM-^P
M-^]sp¶^YÇT9´iM-^FÈ^AØM-^X^F^]M-^X^^@YêÄ^_M-^AzM-^TM-^W°zøPW=þj\$Ö°^K ^P-ø; ,ãæf6M-^Lİ5ŮÓ
Äö|M-^Xr´z2/M-^TM-^QM-^NñôùBÖâM-^Q²&4^O^\spđj}^-^O^: :y^Q>^WM-^R^S÷á´İŮNÉ^D^RÓŸ^YqăñÍŃ´qy^Y
Đ^O°è0ăJèWĂ^C»ÎM-^K9;cÝIM-^Q^?M¿M-^GßÂ^Zß´K_Âİă3ı^Tt2^^uaM-^E^]ËÆæ²¥4óó "İM-^K_&¿à°

M-^OJ^NK¼^DÜm(Få<^NCM-^[¥ßë@@2•AMM-^IM-^XÜlM-^CG^O ZM-^[M-^C_/pÇ
^]ð¼.àÃî^R¼Ô´,A-ðÃw´-ýM-^\\ C=M-^EGM-^XlRqCåðHM-^]Öúü¿IM-^]YM-^I^H ^Y^UÃËM-^@V?CqSyH_m^aüÌm
Ó°^\\ðõz;4M-^PñM-^SãM-^JVjJ:CM-^\\M-^Fu@æñB!dx-¹¬3ÑR#^FÄzÎnáØÏ£¼^YàÏ²ÆM-^E'.M-^F~¶ì';Ëez^Sd
hãZM^Q1´´ó1íç(J<ùæ²¹^XcU÷^N&Ëÿrá|ds^a×÷M-^@î.ìM-^C^KÛ3FDÜM-^Iúc'ÿ¹^_çM-^[cM-^Q#wM-^V¶ËM-^BQ
M-^IM-^Ió°±tÛ§úá@ðf}GØÌM-^Uñ^Ar¿M-^R~Ä[Ö´qÆM-^H¶lvqM-^CM-^[÷;kpÌM-^X\\M-^P.¿^G9@^_æ2ä^S^?f
M-^XM-^M^a iÁ´´M-^_ùß^D^Uy,¥ìX¿³ÍÎ^PÐ^\\ö~Æ£D-öÄñ²/çgk^Tòóév^m^S^N´Ë^GÏù|ge^KM-^B^_M-^IM-^B
M-^J^XM-^Rã©NA.ÏÁ+M-^OSbËy^VN

^_[M-^IÓØÄ^PM-^\\ikqŸ^NW(ÄîM-^KM-^^
M-^U?\\^C?Ôî| \$M-^T^^^XæDM-^M^àÑ#ÍCh`Üjì waÊM-^BfoQ^@ÎO°&ö^\\yÄÄ[@èç; : îx?b\$D^rM-^Y^C%îM-^A@
¾, fM-^\\}\\^aÄ^DM-^M<Ö>hN]6ü><U&dM-^B^GM-^JVË, Eù5D1^E°@Ë(|ÚM-^PM-^W%ß±ØBM-^Lg)3´^QZ´V^EM-^G
^@r^[kV8äTM-^S_P#M-^Ntð^Dá ¶&TÓ^Qî@b^Z
^\\°?5^b¼M-^EÇ`ÖU \\Ä00B^YèD^YùÖà^KóÄ^CÛŸ7Û, M-^GnË; ^T2ldM-^R^_ \$I^R-KM-^@^NM-^TM-^_ ^N^^Ê^R
M-^NX^[If9>^Vl¾^P^G^NoYa^N<8><8, ; \\M-^Yã8w9LÓ|M-^\\û .; É^a£>^BM-^Y[MM-^O*°*KsÓÊÊö.þ²îc} |[+
M-^NM-^Vg¼Aî!M-^Iç×M-^O&^GŸè ò^PâM-^XÖÄîÆM-^I^^7î^@ H²M-^KM-^\\%î^D^?ôF#M-^Q-^]¿<Bîþ#Â
âm-^F°é~rM-^JM-^E,K|ê3;M-^I^Té@vM-^E@{^U"úwß^Xà±M-^Jé?kHþ-M-^Wæó^Y^OÎÖK M-^F-òtBo^D~:Ÿð´Q
¥M-^GM-^IM-^X)ÖM-^Ecq-Ûç^DX"âî@ã, (^KM-^GZαçM-^Sã^Uşóðs@ÄËÖBvq^GVM-^ ^KMH££^U7ÃÖ, ^VðHaëZî
M-^\\ Ä «M-^CJ|N!@2fM-^\\qM-^W^NmM-^D\\α¨-M-^YM-^S-Ÿ`vqB^RrM-^YÚ+ÖÄÄIh±ÂM-^ [^^öV:²M-^MS
M-^_ îM-^Lb^]Êû^W^]K26DàÄ^TîîÊw; ÔM-^Fs´í^HkJ´#^@^_éhY°M-^L•vó5~^R69±²^FM-^N³Ÿe;µ^O:^[Ø]Ÿ
-n^D^Q×äè´M-^Ië^aMM-^NM-^U^QÜÇŸ^CM-^ZŸŸM-^BUûæRw¹]]M-^Y¨ä^CÑ^a_&ü-M-^J6ÖÖ×¾M-^L\\òC^GÁ/ãÊ00
µk(M-^Zjç»«3¨?R^ZâñÖWUèM-^_5á^n5cîËIZOŸ)<^YÖ^Vö¥;l`îúšM-^Hô|¥d4M-^AM-^Q; ^D\$šM-^VM-^DÑ *
^SM-^X ^HkFîáY!|M-^O^OM-^]^E++Ô?@ÄX-{9RaTûJö^H%M-^An»ŸxÔ>*^AâMq^_Cûuj^Nô¶^EûM-^[×(\$W
M-^S^B÷n\$X!M-^E`Ü¥-3M-^XM?^G^Z5ùGpê-¼tB}Nî½^Z^AM-^H±|M-^A°Ÿ5ÛöíNM-^AM-^O÷U4ŸŸg>M-^Co×Sö1
^\\|âm-^\\^Z^Y°RM-^VhM-^Dûs^E
ã{cµ,3ê»M-^[
âm-^_ - |Ÿ}ŸÂM-^]êQOŸM-^@^@x<M-^B
M-^Lß÷î*ÜQË^F8x¾Øñú½ëeÜ³ú^Y
ß/÷g6b?äôûSÖ¶ÔM-^E²Ä^X`Äé|^XM-^Ry>M-^EÛαçkuqVmn\$_^KÈ½M-^]î©ÉM-^[c^K^XpdÆb&áqRí:´It^K²m
M-^SYM-^B^XSÚðMr×xüM-^@M-^EÖM-^HM-^C|^EMözM-^bëö@4Cb-M-^W@¶ÉM-^Rûq•ÄîîM-^JîJWM-^\\^Y^\\yÄP
g¨LÛ^BQxTâV|M-^L^B^\\y)ÜÜÖvş¬^CM-^LócN1ñT°@^V

ÆF{çør•kM-^FÿAMÁ^Q×M-^Fq1^ZM-^I^CSyM-^T»ïR^SxKÇM-^\\@ùó?ùà3M-^O^\\iÖ¼æ^RµaPÝM-^_ ^E3İ90(fÄeF
+Oe];Ûèô^HM-^QaÃ@¼uM-^@!^_I)È^NwM-^U^VOHxM-^I^K%^CVüö"^PM-^FM-^YÝ}M-^B÷v;p@c~øİ^OáfM-^EtÔ
M-^RĒäö^FÁTÅ^F\ (Öô².W8^HM-^P«@µM-^M84â^C^O^YM-^BĒ^U| òÐGM-^J, ^VeFIİZDè-1^N7fJ0ăđó.^V?#!
M-^Y+Mµ¬M-^Z¼{_NHM-^IM-^Uè(LQÿðø□M-^_M-^J^E^YL^FM•ăM-^GE^RÖŸ

ĩ ù¹úÇ?M-^LM-^Y¶wě§üçÛùÌM-^AØ<u2M-^_NÜò_|-ö«²ÊM-^^ě^EM-^M^F&^æë´óîD^Kö-^FQM-^FÝi¾4Bü<M-^H
M-^Yd£ð^S^DM-^F\$§M-^ZX^Z7dÔsÀM-^C&cÚAð^Q^CO
αð@μ`j^PÑÕ^W§M-^Cè^V,2ç±o<°M-^]Leþúú9k•M-^Wǎ^S°ÑwM-^VçM-^H^\\BŮú^U!\\M-^N9\\êö
Ç
)\$ê|çÒ^^K)h^BM-^Q-Îúð?μgW.ÁÉUM-^AØ úǺûBM-^U¨^Zÿ71^C;M-^\\^Y«B^\\«M-^Rŷ-}émöÁO±^UĚpQ,úî,^G^G
_høXíM-^MoM-^U4´^CMp³_ð¿Ê¥§ÈM-^_M-^Hŷ~ñǺ}+|§Y{?Õ>}Õù #°ŷ^_ŷM-^Wypô|Ô5£¨zM-^X,5ŷz0^AB^Kð`u
M-^J^O^OÍUDHoM-^ZMĚ»¹Ě^]1^ZM-^Eα¶²?Ÿ^B|8M-^ZJ0Û:ÛnAMÚ|oǻ

"ô^R^]zr^D;U•ÛÛàÆÂÇÚÍ@y^_Û^A:³-^UM-^I[B •éÓ^Bèà^^^aÝøM-^V«SRþélþiñf)IM-^P^O`Vø{^YÝP-xÚhf^Z
M-^D^\\M-^RÁ9M-^B•#W^YMQ^AHälÊC¬üîM-^L,^?^_Tð½^CÚM-^K@M-^S5ÿ¬ÿîM-^Kc6êÛaÑÐJ²ÄY^SiéM-^U@æ
>JþM^K5^R~^UÛëM-^V?^@xÛm@QA¬Û;y``*Â•´xýM-^Û^FpêM-^WäôhÝÖæñCÊßû]M-^]Ä«« ^?M-^MÖÖM-^Rm¹©İ-mG
©Ç5´sM-^GZvÈ^\\þûfM-^F@. ^GU^\\¬HJYðM-^NNøbM-^O_ÓM-^HÜ!ó/M-^TX¼-^N/. •-M-^\\pM-^VqÛÔô^VYèM-^B
6o^TþÔ¥M-^K&M-^Qx^F^_ú?=EM-^Z^QM-^Zny¹M-^\\M-^Jf•Ã!LM-^PÛ ^Bjp^R@M-^G7Rsù(T
è#M-^J¼\\¼ô# xÉÕ
ÉóM-^BbÇVeM-^Zèz7Åð\$WÄþM-^WÖM-^OM-^O½«èö}ª" |Qô ÍGM-^F "[ØI□^QâUM-^L2Ñ M-^H^Eµ^Vöİôß^[
&&é^X

wè^KÖpíC3-ö©& -ÊM-^Ia^?~ðpU;^X^[L^QăM-^IM-^NÉÂbM-^KM-^@èM-^QM-^R?<M-^WVL ÝéÄe^^Fô~
ŷ^]^[ú
M-^Ux'AK^KM-^UM-^I-T:TD<ÚE^RsØ=B,rxÁ!šă3M-^D#ŃĂeŮjÁ7³ŷyiü
h%^[mØ -A^ZM-^S2_W-M-^Y^RĐM-^^@^HÖ-°jM-^LvM-^ZSFM°HUM-^M-^VM-^WM-^_nñ^DĐĂ,Uă<æm^G^CM-^W
çý5^KM-^KEiM-^Nàbè +^PJ^C^K'Ê[îăM-^JkĂpR B°HèM-^XÔ^OCT^aHØŷiM-^M-^X0•M-^C^F^[^UM-^H9
M-^LpuTúĚøÛ&£[+Ůf»'?M-^ZJîêĂ{ Ç(Ô^Pp^?ă¹éÎ•M-^OyM-^TL^OM-^CHZM-^M^GM-^W^^Qu^FiôCM-^aS9
M-^WRM-^@
Pă°Ńo M-^H^a«Çpt^O¹]0M-^PnŷUOkM-^I|=r0VdMİlÂ°đ3´`M-^P¹M-^W/@^CP=Í'âÉ^GpKZđ7JM-^M+ŷ^B23>5ă
M-^TV*M-^R.İİo'M-^QĂ,zá^Qì`T1M-^Q M-^Rç^Fñ_M-^I

jbcj^BM-^^ε<^?iZ`â:2ëx»M-^JAM-^KyE¬,^W²|OÀ,^M-^GF^Zë¶GM-^KÿÝó}4M-^M@M-^_±k¥óM-^[èª¬º°,':
ã:CÛytò(^C^AM-^K/Ê:~P³F3ö^WßÝ^Oû33vD×MM-^Hâ|à>°M-^Tð-UÝ?FcN`úM-^AM-^Têp^]ù'i^VM-^@#^Z Ã
M-^\\ í<uë(ÀM-^VlËá4;FÌó1Õ¥-+s-d6ýM-^SG´^GA^FzM-^E!M-^E^Z!Đ•qðóæ6Jw3Ý^U-Ë¥÷M-^Y3í<^H>4|;i#
¬k^?ß{w\$Ì,nM-^Ncmß=+ý^Kc?.é0Jßý^÷Ê;XÃø•Öñzi`¶'ó|^1K^UçðÖM-^FßýÎ+M-^ZGîM-^[Ã²ðB×áxÀm©ðM-^Sá
^VûM-^WãæM-^MIM-^\\~Ëg×,í÷ên^S¹³M-^IÑý~ pü+ÃM-^ZøM-^RpÃó?^]ö+F^Y©ö8^_çæM-^D^^^PM-^MxM-^Q
ËM-^I%~O=È±

^^U,n•M-^_M-^WèÖsìVM-^Y%!WaFôÊKµR^VbÕ^\oÖαÈM-^LmM-^S^TM-^]M-^_M-^Q0M-^FÎó-pâM-^Q^Aë¹=â°L¨
^VxiÛ42r/M-^YKM-^S/T,ë)Î@ô["^]M-^LJ^D=^RM-^M'^^C-£]ô×%ĂçâM-^G^WDÂM-^Uàç-±M-^G=M-^\\%RĬM-^Tk
s^@uô÷M-^YR;ăbFÝfÛ,ÎÛM-^E|M-^AM-^]đM-^N^F[M-^]¬;M-^GgM-^D&ürĬM-^Qæ-^Oø pǎJûúM-^MM-^S³²éb
M-^Fú£M-^]M-^LøÝ.¿MÓ+ âM-^Q3W^V^Kx^NM-^TM-^Z^RÝP7^?^R±) AÓ^FAC_M-^R°Y\$xc¶Ů@

qÇ^E^T^O^UT^V>ËÿM-^F^RÛ^WØ

¹£%Z¿IoR

qM-^GâM-^W5+,M-^P4§M-^TM-^P^D~H^T2´b0qI^XM-^]*bG

M-^WW%î^W b\$:^T

gKM-^T- { ^_8^AÛsFö^W¿M-^YâM-^L'M-^U2û^PS39d5ó?Y^PM-^MM-^X+ û'í-^PM-^X(½Ã, ^XdpôÕÑM-^F
ý!ò-;LM-^Y^V^Tpxý¹ò8^Bİ^@WıiÕ2M-^RØÊt^Uμý'-^Y_jM-^JmóşyM-^Qăí 24wŪM-^BĚwD^QĂM-^_¬^Er+¾;²Ô
Wİ^DY^RwĐ]ªÛÂÖ±+£Y^^6[^Ogô^D@^HĂŮēm-^VM-^Găs]¿M-^J:M-^Dq-
0ĂuM-^G=ôĚû8^BO^WP^?ŮàDvĂ&^[ĂÿM-^[M-^PÖÔ4í[¿İ0|ù2ýësAA/M-^WJ^FL][Ů°l'BØe-^YæzM-^@M-^DĐHc
DM-^Ck Ůşy½÷»g0QÆHè^EéÆÑĀG^WĂÇM-^JtrŇŌëö^OĂÓM-^@ŮamAŮ ^TİU&X<P¶M-^EM-^C³ăi.İş'^^^Oq^E
M-^]ĚM-^]£ØbM-^V>z^PBçZjĚ.M-^]ëk@M-^@M-^XM-^G^?\M-^E<Æb@M-^HŸ+fM-^Z"ýĐO`9M-^Fİ^VİSùBĂFø^H
Ă^z

Ø, M-^MM-^IÆ

PM-^YÖÑ`^A^@M-^Yw"x\^^fM-^QÝ«ägİóÂ^@v@M-^_M-^Uh^H|WL^A xNä{TM-^M-^@PóİjTM-^Eμ^a□*
<M-^Ev÷
tM-^Z^a\M-^]
é^[^Z]İa;TM-^^Ê³M-^XÀN»´fmöü^B+_uw7BÑ
ÅαăÉM-^T«è:;tô-M-^E}M-^]âx^ZrèM-^YM-^DM-^CXñM-^Ky¹`Å-³ó\©Z@¶14ĂM-^X~^KÖgL3¶@-^VÛ^äÜİOç.^E
Y±ıxM-^]ßsòP÷ĂM-^EM-^J|^=^ûİM-^R^?^Bİ M-^MM-^RăßI^SÓÔNM-^Fð8¾'ûü'M-^WÛNJ³M0²_ Ø□¶M-^CYr
AĂM-^Ks' ^a^^&1Ó4úÂ&îM-^D*uöökÛ¶= ÔIÛL»>^UM-^SBM-^]M-^B;ü'M-^S;Û"NBÉbçØM-^Xlm{Ö^QcMİi«Ói
M-^]LÈM-^^ôÊ^ZM-^M\$M-^X cðM-^_²)-mów,ĂăM-^DÛðM-^S4ð-¾8.M-^BM-^DM-^L=M-^K^^-W°lPİt□M-^SzĚĂ
Ý'â3v^NİM-^YÂ]^RÉ@mVðZ<)÷DİP,«M-^KJM-^M|rM-^UM-^D^[1^HöÛ³ŷİÛ"x}^Cy:_=]h!_İİhÿZM-^E§&çM-^I
VM-^KM-^X^XÝĂwÚĂwøS^OuM-^FM-^X~ÛM-^RÇÆBİăMçç^?hVİØZç^YM-^X; >4M-^K^E]¹

}6^]j0^_@M-^^éÁF×Å´M-^Q⁻³M-^QÚÛAüfðmè©0M-^Hn²
Pİ q¿{ i vM-^D^@t"ì^Rmz.Ý~Ç³M-^S
OÚ^P°EJM-^QÇUÜÂ|G^Eü%M-^UF²M-^KSOO¶L^PFM-^E4bÖÉèæM-^['â%%ÑOF+g}ðcN^Pó°ûWÊh¶, "î0g£lDM-^EsG
Ñ(v°•M-^Gx'´+øX6Äšmq76^RíY^K10@□ q°`îîµM-^K)O i -pÆÅg#M-^V4¶×îîM-^[pě

M-^Om^-
çM-^SùV
cë-êÿzM-^VÛµîĐ&ö Ç^N, ^DM-^@d[Ý´^V°ÿöM-^Eª÷28©®UÆÖ{M-^B°a<1µUM-^BM-^AÖZsw©6M-^IIòì
^H6M-^UO«ĐøM-^A=^]M-^RM-^Jàn©±M-^RM-^^
Çe<]^E\÷SÑM-^Y^SBĐ%¹£,Ö

@±ßç?óZ&|ăiû⁻²S^TJ_ŮûM-^D¼ë\Í¿âñM-^B. M-^NàM-^P\$g^C¥M-^HM-^PŮP÷

³M-^@ßα^YM-^X^H6M-^MÑ^\^U^O^UM-^_\$_|09[^Z^Q1ú^DM-^]M-^X¶/Í^Sjv^FuçHúüÊévÍçÝ(M-^_Ûø}/2AO\$í
M-^^\^?M-^GM-^V^Pìü^EÔ,M-^IMu^^6¶%M-^Mù^KEÄ=SM-^^\^Úb
Ý[M-^PM-^ZÜËR¥M-^S72ÑçÿM-^F÷^[©H}¬-3M-^Ss±B-^VdM-^^\^%ÿ^KwM-^L^CÉM-^QÛÂóÃM-^[PEO^EÔ*)^OM-^Y
hÛ@ªix (ä=^_M-^@•M-^Mãm-^SM-^U ó ^KÖ\$(S!g¬<ú\)tCM-^L±â;æ
|&)îM-^^\^Ó+ódM-^^\^ûþM-^Y¿ûQ}éln^?B%EKþäi^KM-^H*^["M-^[Â>*âr'ùn;Û-ÂM-^RM-^P\$AqM-^O5^R^[
M-^Q_^Yôÿ;

ÅM-^[iûðî, ,½M-^@Ö'6^E©M-^Y>M-^N^[pÄrM-^J)è©XÖ+M-^]í)M-^¼D\$«ojú^WX>uÆÖðÇM-^Q7M-^[^@^T± IP
 È©^VÆ^SİM-^R²^OM-^Tx-è^GÆ;KŞR•M-^WM-^Yç|½(ÝðJfÖ@Ýj©M-^Êf'xkëí&¶³M-^D/Ý;t4^VÖ@st¼âÔ^Sø^Zð
 1^[q;u±M-^J\$M-^_ÿ½M-^&Üâáp^R^T¿ 3èâ|'-tiËM-^Uf.æM-^W~M-^X,IJ Gs/35×çM-^SéaË&lI¼ksYµM-^Eü
 =,è^O^_]Ê[bæM-^@^?iqW^Bÿ-i(M÷K^W^^H1M-^S3nð°9^EM-^R='Üx

ôM-^F^Y*«-A¼ü¼mö³ðy1M-^E{7^RM-^XM-^DFM-^\\øO*i&M-^Nßä^Z)ßM-^A@,gM-^_ ;ö¼ì÷"ìlÆM-^QQ^S.ç»
M-^A|uM-^Ró1T^H9ç^DÈM-^T^SÍM-^Cèumx}ÄÈrÐðÍI¹h^[L^W)Cu^?M-^BM-^_~4æ¼yàT¼ZM-^UMF"*&^HÚM-^SÓ
>1s-
âÿŮ^_ ^YM-^JM-^N÷O^QU^Q4bLM-^F}êé^Qh-Pê"è*^YÊjM-^WTÖ²^Z3^[æi«U»M-^QM-^MM\$%íM-^LpŮM-^Vfg
M-^OtçM-^M^US6uQ@µ^QZ`Xè?M-^OM-^Z4ð%M-^αD]M-^U\$M-^KîiM-^R^D^R1-^SËÿzzM-^@´¿TNøÓL^Og»^@
Ů[áM-^VM-^FðuÖ^R^ÛrWÀAshê¹¿M-^PM-^Wç^]cM-^QİE:¼^_3^XZPŮB^Uþ^U²ê%İè^KÔM-^V^S^Q^XMèÔM-^@³
TÔ¼Psâ¹3j^]"?³¥M-^PØ@Öh=
[^S|F-0XŮM-^Bí<pÖ^O!M-^XŮä>ðM-^Z^@¼;M-^EtöLä3ðQªúèM-^ZM-^Jh´^V\$)ðM-^LÈM-^Q•ÿ^V6´e±M-^W
M-^AM-^Vc>£9M-^GI^[é¼Zî^AİæçŮmUM-^IŸFM-^\\>M-^A5@û^T±tM-^E
ÚÄÍpè0ç|ÈçM-^kOM-^K^?öP_ðM-^YTì^@ðhM-^Z}²İ;ÿmM-^R)7ÐÉ^H^ZæM-^P.M-^EM-^Sz-ðÖ^XM-^_ ^WñDSìs
çÂK^[G^WµJàLß(q^TC"Åpã°ö|)èM-^XçåðèM-^YÆÆ0&@UqR^Kp2ŸM-^K»ú^Yd¶mİM-^IðM-^T²rC"m-^Q-M-^Z´
M-^R*,ÑGd^VðŮ)ù)Sh^_]T [üİ¼M-^OJŮām-^QDÍ{ù_ÉÄ^YUM-^[,µ^@^Wà¼M-^Z]M-^JM-^SM-^RM-^U^?M-^JÆ
M-^G_@éq?_»¥^?}dö@M-^En[-•M-^N^B/§ÅM-^MLy¼¿<§ÆM-^KM-^[İüüwaM-^_OWöŸ2MVŮpæM-^S&|fnit#UòtK
M-^G´°Lð»ê#^Rİn-I÷M-^BM-^B²ø^V8#i~ðŮ,F\$^N \$¥•İäz´ÈM-^_M-^THák]ç•OM-^J¼8iM-^_ÖiŮ&ÄôM-^E«^R
ô4ÄŮ-yM-^H½»½M-^[ÆhÔ?Đú^[GíM-^UzŮİ^?g|M-^DKŮŮM-^Qām-^Gİ^WVM-^JÑë<âf-M-^FäxÓ^Djó(t}5M-^_´_
Â\$ÈŮdM-^ZÔmòÓİÄÿh=^DQM-^@{h^_³²«çM-^KŸ^_èM-^\\@G^aM-^Rôm-^S^E^\\xYðøèúÈ^Fþ~ŸÄZÿéÿ^E^GQ%k£^A
^î|pM-^FxÔ@OM-^U:ªÖ11Ä_æS´^Z,þí-6Ů_èâ;Ÿó□iİŸäè@ç^R³M-^LB±^GÈ¥9}.ŮM-^Qv+^Q-^OÈx^OæqÔ^Q³
M-^It_¬jw@ìð?íJOÄÔM-^_ ^Tµ^X^Q\$í^X;+^Uİä+1xM-^D\$&\\İ£¼¬¬¿SM-^A

øM-^OM-^CLM-^W§Rç¼M-^EM-^Q~Ë
xÔM-^L÷a¶ÑM-^@Ê4ÿj;)M-^oM-^[æ&yM-^CãŸ_M-^AM-^RÓèM-^RZÂ@19“çÅ÷¶Y[^?^Wµö|M-^B*jèÉü6løM-^WÊ
|°áxM-^UM-^@ØéhëKÑĚllM-^KM-^O24^?cãã?VfÖéöM-^[Ééhþù&Æ#
£NÛ?;éM-^R^P'cür*:UM-^[~M-^FæÖ^?nJMR;FÂM-^Db«ûù,^CíM-^FbLôî^W^XFÎ)ÝÛU£M-^Q@mÛÓSODfB^SM-^U
?^TâÃGM-^S'K'M-^Rm^_M-^H^A=%\$M-^[^XùN\$M

n^\.Í°Ç;*Ôyâ©M-^Uâ(±#5!kİM-^WôM-^UH¹nH-xPoM-^C^UM-^S^U<½C¾Ó|M-^hðÇÑÑÈà»M-^T?M-^[üS×M-^H¾
 ZRf>¤JM-^T^H^S5^[^Y=ù^Wv?ý*"y|«Aî^D}M-^@E«k^KOÞ±fM-^Pñ^_öôÐY\$@M-^HM-^P,¤mÉE•°¶¶Pî3=Z M-^J
 b¼-Ç,Í\EİRÃë^

=O^^3&^C53IDÇØ
M-^Mg%^U^Q•Ês*Ã~M-^M²®M-^@

;@ØM-^M8M-^S°ÁP^Ug ^WÖM-^M KM-^L .M-^SgfÀ
ÉM-^VV^C°«KM-^E#M-^SznM-^SÎ^Rlÿα§^P_-? M-^SÄuüc\$f»^TiéÝo^?3NøYZ^CM-^N^-^N|oĐǾMÉM-^RM-^@ÿB
9^XÎ#æv@â`8^N^NĀM-^F¬^^_6Đó/7•Ā^HōîđçXP-^FM-^NŌđŌĬD^Y^\+^@é"q°ĤĀ7Đ6]=oM-^XĀ z^aàØ-6+rö\$z|
=X^QĚ' ;Ā^Něxx,9²u^UpM-^R03Ār-æ's%CM-^[iKlĀM-^R{M-^HK^aăw9îēĒĀM-^O°¼P>M-^M^Gâ+D^Kfw)ó\^SĪ°•
M-^KŌ!ĀĀM-^[Īû´;0³QýŁM-^B ŪW>è^Q,æ]^]jZV^Np;^G^?^C^UN³M-^GĐĪŪŪF;Ē-àtíwÔM-^IYVM-^BgĐý!
wM-^H^Yě¥=!^Uàm-^F½^]ÖyM-^OŪyĀ^KM-^PHM-^O(#" 'M-^E,M-^\7&;Āx[vĀ^Kö29^@=ê^ñŎ¶-a;ĒĒúM-^W<h
M-^YM-^VM-^B4IĀ•^Q M-^FEă/ĪĀ!IŪ°M-^T\$ŌPă/+Ūô uCjŪr@^@^W,µĒk^W^\Wè©Ē^? ÉM-^CXBM-^Kîl;
12¹•^a{RM-^O3HM-^Uhàø^\M-^G^W Çø'M-^T^M-^XêŪ»R òŌê§"Āî^CŊM-^WĪG44onG^QM-^W°^VákB^?T^@
M-^A%ă;²ièDY^_mă^F¼ZµM-^@Ŋ©c^[.Ç

^[iöQ^DM-^Y^^BH^P^÷Lµ>p49orT^Dñ}n^U8HÈ'»vØ«M-^ZßÚOT÷îû
5^Y]^½û×^T^TMuï2M-^@ÿ#^Q´Ñ^QÌNM-^WM-^NNÛM-^U^@72}jçÌôzÚ¶: ^EM-^[ðÉî^NĖV-Âà¼Æ@çM-^Z^OÍM-^BD
;^EÉ^OKÊwÑæh3^Bû¿§b{,M-^I'M-^Pð9±M-^F¬^TM-^H^K^_□^C^ChM-^N{^Dðn_M-^QM-^Y^E¹^X^KiM-^Ueû@ª; ß•p¨âpM-^\\nO<^V^Nµ8^X^K8'Æ3çû94M-^\\Yæ□±Kz^GCg^CHð.äm-^_LúM-^@^K^M-^LR-cěăŌ_Çó&0ÝM-^DÐØ¼ð
^C|[89M-^_,<ÿ^UÄ~^G

M-^WM-^BÔ'ŷ°-ŷŮ^C^@âA^PÔM-^Xμ+´
ŮeI¹0Í"̂f+M-^DNăM-^F,M-^B¬M-^EA}4ĐăĂ:
"̂;M-^Sđøpŷê,^PŇ^Tx'
M-^\\&,5^@Zó^\\Æ+fIèé3^TM-^OM-^Ußx(Y\$M-^NYŸ[M-^Q¬ ßcFO2ěöøMkšŷ½Ů¾M-^LlO'tM-^YμŷĐVtçæ-M-^C^?
^RbÍ^F°^\\ĚĂŎ\M¬ièŮM-^[ŷİUM-^SM-^YŮ@z|M-^MμM-^WlM-^HđŎlt;Dl[Í^P^6/k^G\ç_³Ňz42Ĥ~Ÿs)K?´M-^]
M-^FM-^Fv¬û&dŎ>*u-^@ç^_\\ð^Ejø3|#
é^_cdWĩ¬M-^Pr,:^\\3qM-^Fè^\\¬M-^VúÍM-^MM-^K<H>@y[,M-^Qj
=laLμFM-^ZF\$M-^IĐđ¼r]^\\M-^[FŮ0'^@c^ZgZ;ă
çà!đŮĂÇUAŮPM-^JM-^U'J6Cd[#M-^M»M-^M²a^î²©Ň^E^_M-^_M-^X^@ùI÷û^V2^]'«M-^P~Q4nM-^P

öïŒ^-M-^a+7M-^Bï~xé8^EM-^[úÍÑ@AF\°JótCM-^FqM-^Twj^NpqǺ|ú-Ç7àè-Sf"/B±#Ó{2§^HgG9ǺÑGDěň
M-^D14ÇÓJÔß O#T_^QM-^OM-^MX^Q¿M-^G>¼^_By^X=x^Q%«jkM-^LjU^T^K^Dí2qt&-^M-^HM-^E|/ZwÂkþð÷ê`
.Ö]Ǽ½SÎ]ǺM-^Y>QI°ñ¹.GQ^[{ÖhM-^Gǻ^[M-^N^[oM-^MŒÉu5\$7¼M-^@ñ,} [ō:ð¹ĐkM-^S'\ŨtM-^Vµk&j^\
ò(^TM-^JE^K^AM-^Záě^_V#M-^IÊR

å)M-^LpM-^I[ôYM-^Eyj{.ÔµY^Rúc9ûM-^P5Û^Q^M-^FÃ-¼-òM-^^±ía^XwQÛ`ê^ú-ÿ0lĐúM-^Y"M-^M^ZM-^@¼Ãé
lw^ZoM-^Y^A¥^Bh^]fçà^HM-^TM-^B,M-^F^
M-^ZìcÂË|M-^RM-^UM-^EyÕ%Ê¹^P^X^^^O^BÛ^Q^-Ađùê^T^\\^]M-^^M-^^ (CäSM-^X´¼^TVÕ£*3M-^N³M-^]FM-^L
ËtSD ^Eª (°M-^@^^;^@M-^[gM-^J^B/´ÚM-^@|Ú^UÚÝĐ¼çôFüW±#v(F¼ëm-^C4^HÃâD~ÝM-^AM-^\\Û@"^OM-^Y^Xæ
áU"WM-^D»M-^QXiÀDM-^PýóáúxM-^I^Z^@MÈªÈOM-^KM-^C|çOM-^Fz´^V9đ^:dM-^Aw^[b^N³@´ °M-^H÷ÕM-^BL
^BXLfµËM-^G3çlM-^ZM-^DÖ^KM-^\
Ç^Q¥ßÄh2xb×M-^JM-^\\pÎ=,ÀM-^Y^RWÎûearPÉSÆpö^Qr¬||©_M-^Ihæ□M-^GCM-^P~^R°À£çM-^ZM-^OĐÀM-^^ÕÆ
Îu^W0,©ÃİÄOk•ÊM-^GÊG÷¹[/÷^ZM-^IÄFEUn |¥ÎÖM-^Yó,~î02M-^OC=^S ñÉX#Ãi5\$XâÂTHM-^DM-^UiN^?
[M-^Zµ³`@^\\^B¶^Q

| (gS^Gôâ+Å^R²öú^RçÛ

..,Ÿ
1l¼ð¼ñ•^ZX¿Ê-~lŸ'í\Ÿ@1^¼ää1°Ü{ðM-^R-M-^X%n.ceİ-èhÞ{M-^Tßð^O»•eUNP´èèM-^G±ÛTM-^Zð{4Gf^BQ
M-^VløJ^TxM-^IbM-^LM-^W*Æð^Y´-!l^A]:yÔ^[®&M-^J+Ô^?kªð-ñOÞUUÜáe zŸj¿M-^DM-^RË^F#^Dj
©x¶^@2<eÉTĀM-^X/ÛÓ#RM-^QAëàM-^N^k^RpM-^DÎM-^X³Ÿ\$!ÊÎð)M-^P Ä?j^?M-^KŸ¶d6Z▯^EkOM-^FQø
M-^y&µ«#Û ĩP?M-^^»M-^]Pnøcĭ^CcM-^Wû^YìM-^]M-^W

d/ö=¼?«Ë¹uÝ,{ïâM-^YvtOS¿^M-^CÛðêK¨^]&\çM-^Q^K
 }(M-^A-î½q@÷^ZOÍv9M-^E"j@M-^U£¬M-^MM-^K9lî°ÕM-^FÍwcùï'ãÛ^F
 è^RÔ/&çµãM-^Gú;ð

g^Uî5hö^TU+à¼M-^^«u Ô
;M-^M-^O ²M-^J^R,V4İ#M-^B?^UM-^WrZDRy\$îM-^@îTk^K@\$m,ıiÖê<^_4C¶ó^?îM-^S-^Q^\\Ö5ÈÈS
lm^Gí/WäûBM-^TT-êÖD òEsM-^H.Öä^QaX\$M-^L=^\\= ^T^OØµî,ÇôM-^Oİ/Öbtsò¼\\Á^\\M-^[y\$ @,^FM-^I
â½çÖİ6M-^ZÓ^EWM-^U;^\\•5^_M-^\\ÚË-M-^HRZ^NM-^OøM-^Z\$VNþÇT^[,Y|ÔM-^V M-^O^Xz½
^?^P^] \ý¬På4#gÄî;^?mM-^I-M-^[)gö`ÐTM-^GM-^SÝM-^YXM-^BÔÚ°ém-^SM-^@Ë^EM-^ZÆ4L[#%^TlÐ!ØJk½
7I^^`Ûj,F__M-^F<@^\\\$°ñe5a-õM-^U^Qz@pÜYy)?<O3M-^Seòm°ñýx^H^WM-^AH&GñM-^Zx^BöGM-^OÁ5Pðv-µ'
Ó¼YAMq!ëp*xø^VM-^G^WÇM-^J"^^%[M-^@

¬±»©M-^M-^J £©ú^K´ì9,È7M-^RpO)HâC¬¼9ÇM-^TM-^@^B\$^_û^OERÜM-^K^^!cü^\\ëm-^\\iſ^GM-^M\{Ã¼ô^A
^UÏ÷«=½{\äçîM-^Xö^Y^^Ç5ÂM-^HZâ
M-^¾³âY|9ÛÛPαM-^CZ²G^ZM-^RP[TÍM-^U^VM-^]Æ
q^BÂM-^]iÃé^W*¥¼î^TM-^DD%XöÖ°qM-^]xM-^CM-^WÜ÷M-^O^\\ü^Yª²M-^@^S}ÔM-^X^Wdë^^O^YIæ^OËD8æîzÛa
±M-^FM-^Wç^]M-^Nèr•îë°ATã=^QÑ^O9 <°Öc?H#ÇåC
þ^DM-^KôÊkÂM-^[Uhu,Ô^Xíä»Ô0ÂR^^'q#-G^_M-^DÕ
M-^I^WùJ'3M-^RLM-^DM-^X^X^^M-^C^DVgixÂ-zIã^\\&0Ö@øóÈÛ±¬+CEçzØ<ØM-^\\M-^FM-^Nè?ÿ^GÆ³M-^Mîâ
ÚBÁÂu-Xî^SnM-^PÂBÝîñîLM-^M*½¼M-^T^W^@ë^UM-^YÔ^O|M-^H»ööÄFySV^Z
M-^WPe^F½úM-^Ec> k^RIlM-^PÝM-^EI°M-^Bwt'øfAM|QĐà|Íî^BmþM-^D}^SM-^CÂç~ÚþM-^@M-^W;T¼M-^U;
ð¥M-^SlM-^J¿]M-^D°Yüë¶} PQM-^\\ö^R^U^CM-^RM-^@^Olð^WM|^[@ê²Oëî^V=ÃîK,ñM-^NM-^Kôúí'M-^HM^?-
M-^][5M-^H*@æ^Y çw^Av"^ZM-^IM-^¬Í{TBÒèM-^RîS^QYDl^X«BJM-^IM-^AÑb-âu°ä@é^Wþv|^ [VÂ"

M-^WM-^A±raî2[VĐ*M-^B£,Ô^^ÖYBa_U,âmM-^C16irM-^Uw»nTÚM-^AlM-^HÉóK%f61^X^<WiÛi|Ûi±^X^GM-^Yv
fÉM-^NÈQ1ăíéz§}M-^Wg^KC6óÛÛsM-^J?^Uæ^XiÉ[o§lâ°~~M-^tÂM-^N¶«M-^SyÝçZ0aĂÖĂzÛ{M-^@M-^AôîĐV
^ROİ}M-^V^[/Â [«M-^TlÆt?És'^Uαá*M-^¶Ö^Nú6^R@Ñ^VĚ^F^]ă!M-^LÇM-^V8^R^@.Z¿±"kî]^CX@Dp^E
M-^[^FPM-^IM-^_rp'!αz2îEëi"íõnÛM-^M-yW>jêÿ%αÛöç}f+ÔÛM-^H³>48M-^AwĂĭ_Pm^K5ěM-^[°SôM-^V«Dás
*Q^],Z^U^FM-^Y[û=M-^M-^æ ^[x×g,y5^A½^_»ăÆ^Gv:¿Sd^UBVM-^QpM-^[çM-^R¼^oM-^BqM-^A³

OøèM-^NßüàèÈ^G}ÖM-^T´^]`R×QM-^J(nM-^]J+1`ÄM-^C^VÅÅ°{L M-^RcE¹ßx<Ïoôû{^?¥M-^I^E

^P^E^òM-^D !OÈ%P8Å?ps¹ò^\ìg&"¥ãð'ò^P^Sö^G¶h+ñ3èö©>³\pÀŮu^Ec4^@°eD×QßBÂM-^ZQz~^R•<æqM-^K
>æ=M-^WbM-^Rô^OÖ7-#M-^O6^As§§è ^PKdm<ÑiM-^L^a {q□Xë□|Zç
æ^|b1² |M-^M-^HÃE(N
GM-^@;Ö{M-^ZôÇM-^WM-^KfN)M-^G(Äy388
M-^Qx1pð^PM-^_i`y²^_a eM-^^ÖÍÈnQ^GzŮ^?M-^P-U\$rdÉçL8rHM-^Z)~Ä'M-^Mö?{÷ßfëM-^@% c1^qI÷òM-^U
C^K].M-^H^BÐ5
^A*ã^V÷s/^P6HÉÐ^K½ÉM-^DM-^@ofÖM-^C\Ů^G«UM-^Z¼f©1ë#=ZÀð^GOHM-^KSú»^^ÄVYËgM-^B•^?Ê•M-^@W¹¼
-oM-^A|Iü
ð^DvÅw^A÷'5#ð^XM-^C+M-^Tã^HM-^M+áx^M-^Kgs+ÁqM-^JíòY%ÝóÊp%²À2-2ðM-^Y[^YYÖ mē¼aŮ^OM-^P^CzSi
3SWpØëð|\ ÖM-^Ií^YÁM-^RÆ^?;^QÁç^[g^_z)'«&M-^PwŮM-^@^FqÀDô}
¿OòTpM-^NAFG8X^Q^VÆNÊeEL^Sy©h5EzÄÈCDFü{¹ÿ.Ø^R±^OM-^RM-^N^PM-^WJ4 àu
-È;wæDè(X¹^R
g-FCŸ=M

áw,i»^HM-^U"M-^L^Q^P/M-^K@M-^Wäß FÃL^GsÔÖç&{òèM-^YM-^MâÔM-^WM-^Dëø³£ú^A^H»CNTì^M-^]-îI^Vi
Ã^K<=M9W! ,^@

1~¿&#M-^NM-^V~ÖP´^]OGZÂU1î;awbó^]M-^BM-^Wÿ/^DÃM-^l¹MÖÖÜÔbS^UØ(v&|îÑ_ÊI÷Öµ5^@M-^IÚ¼Ãö-"
^Wâ~^]¿Û^M-^\\^^QH^^|'|M-^D^Q¥\'M-^_9Bèi^GóóÀÔ%Ú}¥9^YC²M-^LF•İa>^%M-^CÃ"^^?Öp-M-^L^Y~JİÝ^ [
^!^^j}hÃ"ÿøİ~½{M-^XÃû^SÃ_ü^Cß:¼4M-^D²M-^UÖÝ_4^H^BÐâûM-^MÚ^U1İJuÛ^DM-^K|¹ÿ\ÿQ^K[M-^ZTû
Q^S^SäVÿ^KuûM-^@HkM-^[ís]"(ÑM-^S2WrÈ%óá3âÃ^KâÛ^F^Z:ìUM-^^i~CÛbM-^HWQUP¼ð^P^B^G)b^BJM-^T»@
^Y÷ßM-^W@^_M-^IM-^B^V úØpª¼Ãç;^Aµ+¨Ö(Ês^GÃO^D1@8^YKM-^LOM-^HHM-^)*ã[^WáDkM-^E÷^VÊe^_M-^^
¬3Ó¬û;M-^N^T^YM-^W¹İB[°+°ÇÃUÛfÎM-^JNM-^Bîİø|µ¶Ûç']@51^Sø/GÑEM-^M²ñ>Ûwr._U#ö¼ÜAö#ùØ
òM-^LössmM-^Pñý5^@ÌM-^M«9cèöPâP,^Wl¾8Gã%,M-^[£jm^NK^TM-^Aa^[M-^FM-^UwV,ÊIâñfAÃ«M-^M^Cà¼8
M-^C M-^G}d-çM-^@÷Í^YM-^O^EM-^WÓsRÿqfÔ£4°M-^] ^R<^C^K_vÉkGPM-^^£2NM-^SÊ>mM-^Gs•M-^^¥àã
^Ecð<M-^VªÃ1/^Ds£°/q^\TÓé^Zé^T^Tt^DçM-^@şë-pØ¥ã^FÐ¹f-Ç^RÖ4UúÖhÆ¥Kü¼z•
ôEM-^S%iþM-^PFÿ´M-^Zª^V^N"ç

P-^DÎ*çM-^TM-^Q^Pâç<^SH^DW-M-^H^[á^\iç¼^V´-~klZ^RM-^Väv^y|óXBupRGî7§M-^K§d%ç-3>p0İóæZëîÓÊ
 Ø>\M-^\ÖâŰ=ŌM-^JjøîÉŸ°lM-^_ ^P,
 <M-^MªÖ ^PM-^[p×İĂcîĂŌáŰ^\α÷;Z½^C!^Q|:ûø`Ý\$^D²,6^\!^C×DM-^Mv1Ű±^]ä"Ö±M-^]ò

|äIûäF²@Úl^_ä«r]^SM-^Y|U!M-^NBE¹W^UJjM-^VO2M-^@M-^M^Y; *mU15[-M-^SM-^VÊ~^G9M-^CM-^Z"Đjðè©Ö
«M-^KúÄk8!µ~⁂dny÷Ô-2(à)a æáF, \$^Vİ®M-^^s!M-^XÖ^Of2^QM-^J\$ë¶<ÍI²M-^X^WM-^U9cM-^BTÖ4^F^ZG9

Ç^WM-^Dù ÎâP^P#M-^A/ ^D^@h
&\$¶Èf`ó@ M-^Qö^G^Nf/M-^RM-^QB\^G^R/~M-^LdoCÝ^AM-^PnHM-^ ^Hß(M-^N*M-^R^\
a)^O^NU^]M-^V,^AØh!L72^QM-^@n»M-^F;B p!f\$^?°î^KÝ°{XM-^OZx^SÔ5^A^NøÝ*3uM-^C-M-^D°psÁM-^^\È¥
ôêM-^DM-^S^CäÇ>?¹î([M-^Aðw¥M-^Q^PM-^Vß^@Éû¹M-^JWÔD;öèÛM-^R^S4^B^?ñ^G,^@Npi]¥^E^Qİ¥Áh
M-^M-^FöÉÚQà\$×<°ùM-^YG;ÀM-^],|ûªË±ÇIM-^M-^K»LW \D^SM-^N=h°M-^YñK|
¹ü

^K

x]B-^XùöÖÝ'`5¿^Â^WM-^X°M-^WM-^_ ;t^A ;M-^UÍØM-^_ö- ;İ^GYÄë , ^M-^]LµŸ^pßèóÚÅh¨·â^QöM-^Yld=|)V#Ů
M-^I~iyò^B-M-^GÊÆ^B¶NñxëM-^F^Q^@Ō^Y^_çrsÇOM-^F± t£,é^[+M-^CóÇ%ä ;í@^NÜŸ^B}jn!LĂ^XNuBM ßu+
M-^VÁÄM-^V\XìM-^T*/iWÃ°M-^U^XJÍ^NèáP^V^?^UÉM-^@s^-^CM-^NùØ/^EJM-^OM-^C^_Ä,M-^_ØË,^Äü@@B°V
M-^NMM-^IM-^SBM-^LnOM-^FMEw^°v0^YM-^YM-^_ \$ñM-^ZCM-^U¹ |ö7M-^YGLÄ^Q*Q¾³m%Â¾PøM-^SÉŮ
Ë^Z`MM-^D^ZM-^H^B` ;ü(næf»İo~Ýl@M-^P^]¹M-^VÆ^BM-^T^GM-^G^ZEM-^JÊâóM-^EbŌŌ:M-^Jü\µM-^K*3Gøè
Wlê6ÑîÍ÷ŸuXM-^@¿vd[^](øØM-^Aiª0uM-^Pã|^YØ÷IGâ

M-^_«^aspíëµp^Cc^1GM-^SéKoË^EĬ^^V>8|éM-^^^2Ã-
r¿øM-^[O@'«âÄ¼L^aEM-^]sM-^JĤbÝ@d×M-^Cðê^WcM-^S^G^K7^HV'ÄM-^El\$MÄ^SM-^W«Ĥ^P-¿'ù^B' BôôMË

²M-^M-^U_q"¼TÛÔÉ[»ăûtıâ@kM-^\şñĂªk^_M-^SM-^XÝ4Z4æ³èR`]? . ,Ă
ı6z¹]ĂM-^Q£Ç^Gs? '¬3â<M-^YÛ^DĐ~w½ê]øM-^[äñ®&^AÿM-^CŮăobM-^Fç, ^Q^[LjM-^AĂM-^M½M-^O^]^N½i
M-^Eviu, mAùÿ¼M-^AĂ«.kO
]Ă' | :Ă"À | m^PĂ^XüOĂ]HÖ^F^R^HM-^Oà^_ÇĂ;ÖÖM-^@, /P, ^]M-^B^Y7ûIİö^?ă^QuÊQÓE@KáJM-^IØ-EM-^@SC^Z
■M-^[G÷KM-^M^A1. !p]ç^E?è. §3Iôôà9M-^ZăPİ-6uWM-^GM-^Ifph^QôM-^G9. °5

M-^S•I¶M-^K¼Øİ/5^∧â&M-^∖öİØ^Y½NaM-^Ku^Z-M-^YM-^∖0ÅÆ4M-^E¾'Ä^XM-^DÆ+^R]pM-^TM-^EÖ,zuç 5
á»Ūn^H½ía3vU^K^∖@ìVĤp^O°ÔM-^EIt±÷^R
ÄM*M-^Af'^ZðM-^WY]^CM-^R^HM-^D¶M-^Hİ;§RM-^]¾è^M-^F8eylM-^B ^EBM-^SlâÊðeAr`DM-^O}M-^MD^^[
M-^∖M-^SM-^FM-^PM-^MàT^ZÖRE
híM-^WràPM-^F æ>EÛ^_ |A~p¶ç±•X^T^Y-3^Qi2pDİfbçñ%zM-^Fb¥ó5^WFM-^YÖM-^Lfx¶æ@M-^∖^@A^OÁú^G?±
S*ÖÄGiìVM-^Lì°^V@&Û^[¥àB^Ar^ajJM-^Bb^EØ}UM-^Z¼xÄM-^Nö°ĤM-^]M-^∖@øM-^WÍÖØM-^E^S±/9M-^PñW^^?
ÿIëÈGÃà^EéLM-^QlİM-^Zs^YÖ^U]^ [d^ZUìM-^B-üÿ |^∖,M-^UgM-^FOμ²&ðñÇ^NmÛTÀ:í³°^SİM-^RÖûôÛgÛÄÑ^
íμM-^VM-^X^VM-^@) ^ToåV]V^OCÔkþÆðæpjêí |Óo{Tμ^A&¾ÛO□÷-?gM-^SM-^P~+WqI¿M-^SjM-^RM-^Le
M-^]Ø6zÀ»I-H\é'/J6Û(|M-^GM-^WI^? *M-^F 8Û*^XM-^]Đ¾ð^Dç^Z^∖ |M-^LÉM-^O^_)M-^QİðpÛUð6.\ĖEí2
ç
5=\$-TRröM-^V§
É•M-^@'ÛM-^LM-^W« |hkFÊWE&MéôÍczú^R?^Zj!-^SÄ±Q
4Lÿ-ÔM-^J%¿Æ&]ÛÔM-^KoM-^Eâ~ª~ªÿùvó9 ŪÄ?P!cüĐM-^DM-^FM-^JzyİĤÖM-^KM-^J; \$M-^Xª•è^q#M-^M.Öí
M-^Uä@ÝÔ3è

OtIH^?^S\$ð+^[JM-^M-^G¶Ä^B"6^Bqø³Sp´\$§¹!i•JvùO^EzûY^Xt½îM-^N^\\UM-^_M-^JMĚM-^_M-^_.M-^R~^`
gM-^SÛR{ {M-^[x´M-^[^^þC-M-^Ex*¿3ÇÖ1,M-^V^HÛîÛàM-^_ô^Y-ÛöVVC1ªi`þM-^@sM-^Aq^O\\^K•aO>¼^T-
M-^KM-^I±;^Ru^YM-^JÄÄ4x3è^[ân«?¼ô@§@M-^O^Vÿ^A++RôêTÖUÆuÖc^_,ê¶M-^LmÇBøÇ/q\$NĚÓĚ´:ä°(M-^^°
M-^VH•cÛNē-ſê½]Wð^âù½nM-^\\M-^Df=ôôf^FM-^Sbó¿M-^C^W^A^Rä!±¹>C¼^_îŬM-^N.÷ÿF@^@^RM-^Mþ±ÄM-^N
α@ŎŬbî^hJ:mfxám.^Uñ÷M-^Y<M-^YM-^TÿßüŬo- òp`wM-^FùM-^I§ôö'M-^KÇ@}+¶μzä{⑥éLX^Z^VwM-^F0ðÖB^]
V^Uf¹M-^TpªÛ^Vn¿)6ĚŎxB.¨ÁñW£HĚÑ»KÎQM-^\\çfFŬ~^?αoÄ^QüÄ±ç&M-^AçCñ´ä;égm Ŭ^Y½×éâ¿αáIĚ²M-^]
M-^^îÿ¥F[^B¼Í3M-^Yªn^^U¼MM-^I^[^Hú^Pgâäªα^_B?îLßM-^X4^[Ě^YtÍjM-^X|iM-^R&â÷[x^Eþ`z/ÿM-^D•-
l^\\Â"^Y~M-^Q^[MM-^K¥Ŭ^R_~Ä!ü±M-^LJĚαs/Ŏ¿Äßü»»¿ô^P^Hî@^SĚ^CĚð}îĚ^A½é^B^G[çĚ^HM-^M Ŏ.wKw
M-^BîÄw^GPîŬ@ç^H ^G^@q^S^@°t^@5!^@p^@^ÿoi}f{æŬoøðfRä²
A»Ä~&|`L-Ä½M-^[½ŬŎóþŬŬŬŬ1\$wä'M-^SûĚ^G;wÑ-°ŎÑiîV-Ŏ¹².
ŎEM-^M£%!!K[Y^@C3^@4#^@fP^@ÿ
OŬZ`nkðM6M4ðM-^@M-^DDqáç¼V¶ ò: ëu/•
x¶iM-^M^Y!±Ŏ^NŬ^T%PαîŬ^ÊM-^NM-^A¶¶jdRM-^E^PNT^QA^E\$2Ff|øJM-^ITM-^SøM-^AGM-^BP|iŎM-^[^@#
^@^@4^F^F"D^@xS^UÄ0Ä^TEçÆØÇĚhM-^[i÷^RM-^Rä´þ^Y¼ryĚî?¥î^VXÆRà:þ^PM-^Xä(f^Fåð^XJV^H^K5@^ZL#
¹{§bM-^ZB/uÄ^WcJ^KŬ^[´ð4M-^PYY³ÆRM-^Wîd4M-^\\!7¥äēâám-^Tîe.! Äw^B!M-^J"iM-^Uq8ß^_6gM-^Lð#Ç
ü\$Ä^CXØP|X«^Wôþ8e^K¼M-^Eé%éVÉO?+]Øø^N%!H!%M-^Elp!Tfæ/1
-M-^Xö^KŸû^QM-^S±RM-^[B#PĚ^E\$Ěoßá1þhM-^F/!xîM-^[_X^Tß4ð²Q_ÆðUÄ^CéþĚuGM-^N^T^N^EünM-^KâBÓB
ñM-^_ ^G^WªM-^I•M-^@gÿb/Mî

ŷ<ÂI^QĂ^YtM-^NZ9O
 xKĂKary°GM:
 M-^Zİt^a^Oİ^GN°^NM-^M^WM-^S^?M-^^Üc&4Å7M-^]ÖØ¼:TM-^DM-^UJ^_@4^X'S^=M-^K!^Hăf|ă|çM-^C^[M-^P
 M M-^SM-^QsİB↵

ÇX *ðTM-^FÖ;5^\^1^^BknL^Ou£^A^Zl:B%wCWJ^F6M-^B5RrÖ'İBăöC;M-^T^Vt 1^Kă
z^Ruă^EÎ^SP#rM-^I^OÖæ\$^[M-^I<^1M-^@?^B2d;^ZÉvöç6.D!yq^^ÑM-^TtÆ&ıM-^^ëpR ²QM-^KI^P|ÑGM-^Z¶
ÿÊø^XR^VF"FÂBÊM-^VØM-^G^P2/¼M-^ZæPUP¹#-0Ă^?@2M-^Q12óXz²F\M-^UM-^JÑ,ná^H7ĂlÖx}İM-^^ÊM-^Dđ
M-^M©P!0M-^Wă¹^X^?'n^?!đF,¥ª|^^È^OSTM-^[*^?I"0l>ı,APĂTM-^Qì

QM-^B•\$}úÊßÖM-^E?d^OM-^RâPSĐM-^Caâ^ZM-^HùM-^PALM-^Q^\Çn*]^2^Rz,\;n|&¹M-^UèM-^W^Df&p ^K±F
M-^V^KM-^YM-^\\¥@o«âfM-^7ÖËâ^[TM-^P
½ßá[â_ùèC¿Ë`M-^TË§lM-^P^F»M-^F@sVN°~ÖM-^Lñ¿^K,M-^J^]këM-^D2÷å.ðtóM-^YXM-^NM-^Bß^T^Vh^?ÎÊ
M-^X^RM-^WĐó»M-^E^_pZ^QÁT^VLÂÇ3#~!ÀWM-^STvd`M-^J7Qf-¿FÁMAM-^T!Mh;9BM-^RRæeM.\$#%]M-^] ^VyPÎ
yâ¥^[7xQM-^E
£¼EjJË§M-^HM-^Rp³B•HM-^Kê5M-^TÂ¼{M-^T^})M-^V` 7qŮ×¿QôO2U`§^Z:ô_÷
M-^_J!Ůx^FäaM-^ZôM-^M-^D^WM-^MM-^IèíÉThPĐBáM-^_í^KfS^\ŮHv^V8 M-^xñ½~jM-^_M-^Pw**ã2oq Ó¿
çèM-^[M-^T»²^AqEM-^Z^[9-p^UpaáéŮâÂM-^_M-^Z¹ S}M-^[i^R^[34)^RS-^NðwM-^OOM-^WTP^FM-^Yf|M-^E
{½øÅS)^\\Éα9M-^Yæd§z¶âM^R^XM-^JhM-^MMαêûG^XOÑ!|αVM-^C^[Ê,üó¿M-^SyÄd^\\}M-^Văîr>Ñs4lÔ^YtHæé
[_^KM-^PÔ2p^?M-^ZM-^GM-^X©M-^A}/M-^IíM-^Pb^H^anÝM-^IŮÊ|ö«^RĂKúçÃ"(ĂI³
+^V !M-^xŮ5^_°ÓT°@^\gGu%;M-^S:ÔèYÇb³M-^@i@ð*Ă^A³M-^@^SŮĂ"ĂkpŮ^OP^RÉÎ^X'`5M-^Fí°M-^Pñ
^XM-^KIIÎŮ.M-^JĐBZf±^OôñM-^VN;M-^_M-^MØĚG¿3pCM-^\\ ð^R;^S\M-^@ÔM-^O0G\$ôM-^[ŮM-^K
•^QM-^Fç^TñM-^NJOé G^i^oİÑçÔI^AîîIM-^OêàM-^HþM-^U\$î#XÇ§ó>^F^Y»»^SCzÝ^H?M-^TP;`xR-M-^R^
M-^] ^ZM-^_Ô^K^G8M-^SóM-^OM-^Açvα"&Ů¥^A}*1^Y
K?%^]dçM-^CÀ

¥'^YÂÛ^_xDl!M-^Eû\ /ÜøÛÇñ^TNßİM-^^áéQ¹ÃåfTeô_[j~^[î'â=þM-^K¶Öä^PUüHFGæíM-^\\î[M-^KM-^[
 î@ð_@M-^UI³áNÎóÅ^VHM-^J,qåÄîiNª¶½U+Uðßø^P^EPáĚ2M-^ZM-^SM-^]JdM-^MKM-^Z^XªM-^_¬?è ò
 ŮŮ0#ÃîüUòX³M-^OO\$M-^C^R^R!sù ċKÆû|,(:[M-^S7~¬M-^H^@7M-^Kü]stĐ

ý R>7ŇÇ^T^}âðaz^AŞG^Tc&M-^GÂ°°XM-^DM-^MN]?ámF^?ñ-^_+ÈªăX;?ñY^^ě²CýN^Om÷M-^Qö/J^\4•M-^L^U¹
V~Å¿M-^D^^pn^PĤÇ±-Ã^e#ðkĚăªĀkwM-^UA¶|Ø4Ío¥¿Ø¼>`^[\±M-^EL^G/©hDM-^M(^oQĐxv``bM-^Tg?M-^Y\$S³
M-^BİM-^KM-^Q^[-'°M-^ZBy/ö^K)}zªdZ(M-^SÆ4^N``»`M-^:ÿ|`M-^ZM-^CĂM-^Bd8^D!aei6q•
OM-^B#¹ü åsÍ4%M-^RH°Ā£<^^×Y\$[M-^L©ÎM-^LĀÆ³RûM-^Z°
M-^YqM-^EOăw³¿LµlM-^EÇM-^S^DöÍèM-^RĀ)M-^G]²Ögëð¼úvÔùM-^ZăøpM-^@•´o¹^Rð-M-^Fw}M-^NqM-^]È|;|
és6|
^WTM-^PBĚAĀ^^ðĀ2M-^V«`µöÍ^Cî^HçM-^Uă«UTM-^Htâ|^PépM-^QŮ^U*ñÓM-^_LvM-^SM2,«
>UÍñM-^NµWŌ4^Kêd±ç^^7ĚàÇmĚǫÆö×ăM-^QM-^^-È}íd"lîÖÖ ?j1{°ö7|¥ŮÍ-!³--^Cð=]±ìY\$^N"ö}íM-^SĀn
M-^Cð.Ûă^^ÆðÇ^Odîv7M-^U3'WØ§51:M-^C<^XaôÔ/M-^|\|^]M-^Eg1Æ0Z¼^Opär8-ÇX^N¥@^CM-^K ^?Ö^[½{ØĀ
M-^PĀŸÿ×°M-^Iî6pd¹¹^UM-^D^N^NŮ°_X-M-^EM-^LĀŎ«M-^[>M-^G^[^Vă^ZĀíŇß^Wót7Āvv^SéeèĚÓ{³öfM-^F^S
ó

°M-^EÜæ.Ü_p+ÖPM-^WéM-^Pk@út(ÇÔçpqM-^L]6çÑ¹ðIö7ö|^[^G^B_ô^Uæñ°];&@ßAİ@ZÚ6±ib<<J^Vûw•éöM-^Q
İ8æµÛ¼ÓQ÷UF¿-Ñúİi'@M-^_pçM-^RαxÜÄçM-^M-^[8M-^Zèç¹M-^Qk+æ&a^U,öet»^Né±Æ99~^EßT-M-^B-M-^V
-ñÛ-^WM-^^\ø+j]R^WØM-^ZÖ°M-^RßæİÊ¿a/±úM-^ôd^N±øOeÝ÷i\±è³Nik^Dö-xM-^SæiK~s0îÿİ{ÈÜdnÊM-^^\#
M-^OÄ9M-^UĐj: ^G^SÇNSM-^@¿7¼^MñEÝ^Eİ^[àèégX&Zèiá M-^D°èM-^G°s^G[Í^PYÄK~M-^_mM-^Pûp
-OçÇEöÄ]©ÖN=gü¥Kw%M-^E^GM-^RZ6M-^ZM-^RonXİÜk^EgqÛ±GM-^SM-^2^F^UM-^D}w¼è]È¼3ð,ùVv^A@
M-^G^F--Ü¼xç92,»5pÛM-^I7M-^^[F{M-^S¼èPsM-^α°_¹iM-^EÖÆM-^Hç•ç^_M-^C^T^z-M-^XiçM-^U8M-^]:
M-^[pM-^W- t^OM-^R@o M-^ZOM-^Q°±Ú7@^Vkvön{cdm~hÛä0£!ÝM-^VM-^YM-^YAö7¼İ÷•|#:ha^VÖi
uB!hèr»^]Öâgc-M-^I9M-^Mû¼²,uâ+Ö¥zM-^C1oZ_éLpJâM-^[T]Û=M-^P^W]UË^E^_êRf^BM-^O@é^Y¼M-^
M-^HvM-^Hîî^RÑİÆX×M-^Y-Aî^@!Rî^BM-^DÔM-^HlvD°^X¼ð_K7&Ö^NÚ³¹\M-^]°M-^D^a^]ûÖÑM-^FÄWdM-^V<C
KÊM-^M^QM-^KmM-^L^S@_M-^R.cê.WM-^Xj^W<Ñ3Ä°¼αqûöÚM-^XM-^Us@ÜÿyÄ^]v^RxâÄæuöÄ|^GçØ^Sp¼^E^rîW
%hÛö^O
üs/YiCM-^Jùkó¬³Sù{
\aqM-^R
rÈÛ^K^EM-^_û{^XÖ?M-^Zp^E(nfß6¼]k-
~JB)Ô^idM-^]İö-÷^?ú'\Ö(%{^AUĐV¼\ö; eD+ñ^HM-^YM-^x^ëM-^NhM-^Aèİ.^UÍiM-^LtM-^]Ä^Zö^N^P"Í.GH
M-^Fàm-^DÈaGÛM-^L-Û¹ùµM-^AÛÖ;QM-^j9İM-^WÄ^SPiRşuo¹ÝM-^]İM-^Y_M-^İİ;¼DuOM-^W@vÄ°
wb²°?M-^İÖ|^DM-^Pm¬şL/İ,%ùM-^Mýİy^CÉý8çM-^JGM-^VM-^Z,M-^[^BİMM-^Wäm-^NM-^F;ökCRèÄM-^H'[Sù
AzM-^K-²^Yy1/CC

^Z«l;{M-^K^YS M-^[Lý&÷ÈÊfİM-^Cä|\$/^?M-^AÄİ÷EM-^Uú´ó@
YçYaxM-^Q<^^ç&=M-^Fÿ'^\1%kökK^[^Pq|nË!¨^EÄ•lössÈÜM-^Æ;HÄîHM-^s^@M-^MôuÛ^Rİ÷RM-^H°÷M-^S;ô
,î^T^UT;M-^@¼s@²ld^O²(°°OqKÛ^ø^VßCîxç
M-^X°M-^J¬9şM-^IYM-^O,M-^JM-^Vî~M-^Cá@^_M-^ZA^[¨A^YM-^NM-^Hû*;^^2ç İW+RôÆ>ş^Qfg ?*M-^_e\$3
êp^a
#4
M-^EM-^BM-^[ÈR-Â^YM-^BM-^KcßP^CM-^R9LR© İM-^Yp'Ü&hØÛÿ^SM-^JM-^TB^NÇsöweößZM-^Lq.nË^H~□^_
^_x
iÂp¥M-^TM-^Q^FÄ
²ÄX)|NC¼x^HÆÝM-^_İBZ@|^KæÈ±µ^_ie\$)M-^KVÜÜâqn<M-^LðM-^O¨Ñ<M-^]M-^RM-^KlM-^]M-^C^M-^PA^YÇ
M-^İÆ^?¿ß3È^C.M-^@^A^Ä^X0^F

^A^CÄM-^@¼^?¬Ô^M-^HÑÍ) Pîüg»^@s5+)Å
u²w6e7^] °^KM-^@^A`Ä^X0^F

^A^CÄM-^@0`

^Xü^GM-^]UT/M-^IxØÍ£PWÖM-^I¬ü/GS| ^I^Xp^F

^A^CÄM-^@0`

^X^C^FM-^@^]^G

Kö°'|ûM-^S' ^^ñ«^EWc8\ËÝ±CM-^CM-^MI1ë8à4

[^@»M-^WM-^OđaM-^G}ÁîðM-^C^KÅÕîðÈ^KM-^Sç1^Z^CM-^DOXæ&]qhM-^UxHM-^KYî^Fp
^

đOf àİw^AA^EİîŸ^?á^[-W>Ç^@}M-^Y^Eh•M-^PÂ^\\íÖ^W^_ŸøM-^Iô^[² ¼^B#^K^Fßü
 Ç^TM-^P«M-^@/zP^Fo^X8,í□*ăċ\M-^\\ÀđM-^[M-^R5¼×ñ;3%~^ZM-^Z^Fċ±/ç`ÓM-^BM-^CŸ^XM-^PvM-^AM-^Kİ
 AP^D'ăċ^?^\\M-^@á8l>0M-^A.îAÂM-^LW9 ĚKŸ1^Z^E_\\æš&Ăù^Yom-^Dò@^Gċ`ÿâM-^I^\\M-^Dê@VK

Uñ§^CÃM-^Wö©^ÄM-^IM-^^^SM>mèy)Û;^TVi7^^8ö

!^OP^AVøøÄÄ@½;L¶k^î^^\bt
æM-^R7 M-^FİMüM-^XEp4FM-^V^2ø^U^E^\^]P{M-^F ^BÚ^X^C9ÜE,^T¥İM-^Nðè^Fg^AÇM-^Kí^YOM-^M^O&,
M-^\^?^U^R\$|×ð M-^^bM-^Lß6`1øM-^FpÄM-^Jf^OM-^@[^FĐpø^@Ză,4U¾ °M-^Aï^V

&l^Z^E1VM-^Sÿ^@^K^Reî^[p7M-^@¶÷á'^KxNïC°^EM-^Nnü^SM-^W~^Wòì;jù2ïM-^JM-^R^GM-^S^E^Hã²;@~Í
^OM-^MM-^_éá^[M-^H'M-^@PcM-^BN @M-^Fw`^V^_4{mM-^NkÛ^FÂã[

iM-^ApwO-° ^@ăöè~w@0Öq|M-^PCçP^BM-^C-ěâCñ^WM-^@hHM-^C¹«^]ü >_ ^FôÁ=Ů
(^XM-^@^A, ^F×^W©£øM-^Xo ÷^X^G^VA=}M-^Oš#*□¹* M-^W^E^^Öë| ^BμăăM-^L ß¹^XM-^F÷øM-^JM-^NP
M-^G^UÀ-^AăzM-^@@\$^WîÂĬ□@âM-^G
p, ^XM-^VxM-^AHŌ^Uš^A(^Bš8zp, P^W¿9°(}Ů^â@, ô°| 3^US5øü½^N^_1^Aßzf^O: ^FM-^P^[/M-^\\^O^VM-^HĀ^
^\\
šrúñ^G³à^CĀAqM-^BŌŸM-^EăĂM-^ZúM-^LtM-^\\^YŸ^A8½-ø^C/ôôüă|M-^PîNŮ)^@^RM-^_K M-^FM-^^KAM-^UÊ
jM-^JA.0³ Xj?]^Ay|^W^@ĂM-^@;šEM^V^@^Ax>g4M-^^epü'øÑ=°M-^A]^FBí% ^Z@^@^]M-^YÝH•ôÑ¹@±«^C£ă^S
«©üà^AM{pM-^KZ^\\^EX^Uâ^@M-^P~v^KĀ|u^CŮ^EĐé

÷^Z^Bw^?°Ö^GM-^GçâM-^P^K³çÀî^Zf~ö^CÈ^ApfBñï5M-^AM-^BM-^GM-^o(xM-^VŮg(xp^(^DM-^Z)^dÄ^R
M-^C{é£¹=á¿ªà^[þ^ZÐÉÖǺ

^CÓ^SÂä#UK×PM-^O\
M-^@à^A½WGð«0Õ•%|İó^RİÜpckî^@öb«^F×^@n|^2¿úç|UFM^ALRÅbM-^D□^Uü2B*C
^RM-^Wİ5•âmk^Nxf^a,M-^]*>S^KñM-^ZçM-^IuAÖÖİ"j{ûM-^L^D%×)ç
ÛM-^VMö^Nm}^R\$|NB@^]M-^îâ'M-^AM-^D^Wz@Ü\$|n¿Xİ^N-sÂê
'M-^MM-^Hù'dî¿Js&M-^Xbñ^]á³f vðbð&çjM-^U8ÓM-^B¼^EçîîM-^@\V^WíU-__İG\$İ^@^?}È«¬İo'öLÂµUü±jVp
>^Y%^^]}à÷^□^N•M-^ZM-^KÝW@Á^NL=ñ,M-^_OWNñ"ÊM-^Ks
«ùÇàøgK^T@M-^D?M-^Z^N- ó,M-^O{+îG¶C>|rMøÖPßü•ÝM-^P*20'T-¹ç¿É3ÃÛM-^YVáM-^Q@èääM-^N Wøç
M-^J+ê|ív«YµóhÚ@}æ*Hçí'M-^PÔ^[s^P^Y÷[M-^î^AM-^Q^Pø
lmM-^Nwt
I^E0M-^UÛÊM-^MÆ;mð'ÖM-^X^?^OM.#'|O{NÃús!* ,^\e5/^[gpp{ j) È ^UO,*â½+ç%þ¥B^B¶ù^ð°^KK^C¶ØðsdU
İ|M-^Wâ@äçµ±«^O^Go½ù/ bçÔ*r M-^D ØøM-^JE\½^Z^]-hçM-^RÁ^ZL0M-^SaÊù/»M-^AÊMrM-^_B" ^D
M-^HRø^Eg±C_)M-^HAðLK^^æÀibM-^U¼^EM-^Gfþ²Êİİ"ü°çømUæâ#ÊÊÊñK'Rî kð¶Méèÿ-'ÖÊðM-^Z2ÿ=½: ^Cµb
íÖÄäJ°Ää| jM-^I@^Z°M-^SáæqUjÐúr,ÖBÊ
Û@M-^YM-^DM-^H-M-^]ÖþðÐt@lØ□NÜðÓHð¹±K3M-^TM-^Z,AdE-ÄöÝíi^?îM-^L')È«Ø¹' #Ã^U•PlMwM-^PG;BSÄ-
÷^Sèä" ^Gz{.UzÑÓM-^F^V3ZM-^NM-^U6LÚa1¼@Ê-±^X>NM-^NvêfyµÝ#ÆO°Ã^?M-^HÜfM-^Bµf»^Vû^FM-^_½Ãµ^[
!R}iÖääJGM-^BÃZ° æIþ+^Pî#^D¼7M-^[_; ^Wp^Zk
M-^UM-^AM-^Kw³^E42M-^J_ ^F"ýfU7pÊ^PM-^U^QoYÃîÊ«
éM-^GEÛM-^OM-^NÜÊ *ääÄÄÛ-qx{M-^Z«½"M-^GåVÓdM-^XM-^UfCßr²^@M-^J^K^SÍ M-^JÖÊM-^NM-^QY^Vñ¼
M-^CM-^T(öÊ@WpæU?^K#"£; .Sç^QâÛhÄèİ.â=2çÓM-^MW,ÿn9³M-^%\^]ÓM-^N+
,röæuÜ0z}M-^UM*N-^]M-^EM-^bM-^H^NøX^Y&çþÁíæ^[Ö¼<^_,C;¬Wp½M-^B^W½M-^@Öé+ûM-^BsÖÿ@ \$mP^ ^¹ Y
M-^I^V½^] '_½iAM^EM}Ê½z{(•"Êd×lu¹dðM-^J^AQÃtîYPäs:±µ kC-ð\$ÄM-^_YßM-^DM-^AvM-^K7e_¿'iÄM-^\
û^N|É^YçûM-^Yj^Zq^UM-^OûÛî,ü¾Lí'ç^Wµñ°Á,Éöê' '¿~;²M-^SoFûM-^] ^B{"«^Eð-^û^Oé^UM-^WM-^M-^T
«M-^N`M-^CM-^KM-^BN^WÅ^U _^Z&7M-^KBø±îMÉ^NM-^Dî¼iÄÿêiJM-^X^_M-^OÂs ÿò=:m|^C^T¶^F,
Z|±ääÊM-^AM-^S\$&□S°ZM-^Sð'ÿêÑ
ÛM-^Vôn^PZÐL·éj^@(M-^Kh¹ wm^_¥C';\
İ|sM-^[kÝb&ä<M-^Wàd^Q^X5ÖI~ç@&y^\;Ä ^?¿eêö½M-^RM-^LÄ¹u,M-^@G%í1^SpæUoÔXJ^_M-^DýM-^S
M-^T(-1^NeYä, ^îM-^F
M-^VrM-^DðÆzc0Rdu¹^vy³SýM-^RTjÿfU¥µiîÂ^Ft¬c^Bæ+Ð?R²ë¥^\É{ ßÔG{KøM-^O%ðM-^T2ÎhÜÔT[^Z
9+ÄpBxâ°íî:M-^Pið:P*ëßM-^Z0/pM-^D^VøM-^H^U•G^TÚÆÖÜQ¿H[â_þKXé^R^Ãµó=#«
M-^@ûM-^I'ün"Sÿ;HTM-^PVûû-G^Sp0þáðµ9ùM-^@^OW`M-^UIí½ÛMXçÿB^ZM-^RnSM-^EøM-^Dh¥oOM-^G+0½c¹
M-^Z

úo^FÔv^_FM-^IÂN²n^QM-^^iðèµMø^İñM-^^^_<Óù^?ùNIÝUÊöWyÝ÷â)M-^T}òM-^TÊ¿wJ%gM-^Eÿ,^Rİ¿ G
kM-^RU"İ÷MâuâV^?M-^W^WbÃdm*ü[nü| ÛG05M-^R¹

ídÂ¶M-^Tk-î^H-ßoíH

Búre^DlµM-^SM-^[ZUØ^CM-^[M-^YM-^^2R°J^H¥î•pb¿\$σ^2´j1¼*Å^M-^Q|dý ÑYo-T¥M-^F
|M-^Mj´bÂ^\ M-^Uàó´µ}a±±M-^Z%±@â¬wó4Z;ãî©ã^SOIOM-^Té¹1;ÔûÎU^R3Rñ¬sÄâÑ£ôÓMvQöM-^N÷V
M-^D¬M-^TM-^HmF^?För^W^Z¶60Æ^_\$_¥T+,îøÖË³îM-^Y^R[(NhZ M-^G^SM-^G+=O%>Ö3ÖIÖ^SX}jHÖM-^Pí
M-^]jã!ÁM-^YM-^E#L\$ám-^QZEçSµM-^A^^¬¿M-^Q(M-^EmPN°;M-^NØîM-^AM-^DLj\$<Smj¹hâµâ¼ÄRðM-^PM-^I
ÝG/ûM-^UþM´¬uû½wñM-^FS^Y!Ö^_ÿM-^T3¹µM-^Y^aM-^P©ÅJÅâÄ«°Öjs±Øk´îM-^^ M-^P.?þM-^Z^Y>
M-^\"a@~/ËÖ^_^RFÖSM-^DM-^RM-^X¥Öi^ORM-^TOM-^ZMaM-^SfM-^O@ÿ¼îcM-^GM-^Wífîj´ÆM-^L<u¼î\$^îK^Rl
)¬M-^P6©7´îØFOUÛM-^Að¼M-^V=:^S Ë^ZM-^CM-^LEM-^RM-^OQãVöak;M-^X^VÆQM-^\"l;^[³vM)2?Ðþe2©¼Öî
þM-^_M-^D^?M-^WfM-^OM-^\"M-^DS«^BóËj^M-^[ËUM-^Uð¿´^RzM-^Oúj^DPöËÖt^Xv?^UK¿G÷NÄÖî^R
¥^aM-^Yè* N^SZ+M-^GçM-^F,-þÿG^WfÿM-^K*^ ``w2´\$^Y0^V^Rm>)âç,^X^OM-^F}hÛ^YmaS^Oéä6Fü
´f^KM-^H¿>M-^]µLÛM-^Wv÷ßî^\"äKÿ!Ü<7Q^HM-^^ë´ä3K^O^W.p®^_M-^Rv½

Ø{k2³ewE-ÜöûĐM-^L°^]?\$^?M-^_M-^SM-^_N@Ů5£°çM-^_z¬yEä§ZJÊçM-^YD;éŮ#;ä_^Q.%DûàŸ=î;û\$î0
M-^D×{M-^Bû;ô^S

f=M-^A°+ŨM-^]sS^PêM-^_M-^]'bM-^T½¬¿ú ^B!,Ó^R-´ñ^QĐM-^Køb\î^M-^G§b=¼^X};^u^W¿M-^YŒêu!Ý Ç
M-^D#÷>^OZĂŨANM-^@P^Hßçoi/^DM-^K^Fú
Ø½÷y1.wM-^F-[^E~M-^CM-^DwM-^MSM-^DxDÉ M-^WM-^Iüs*z^QB#»j.jô»¥1M-^MäËpî>üm-^EK 4P^PvM-^D¿
t^GD8M-^Vu2~21VM-^I-ñGŨ¼{pŨmM-^Og6Ũ²§nM-^HöøM-^[/M-^BGÖ;DÔÿîŨŨ@7M-^P"İh^KxæM-^Oä9Ũ"@M-^LU
ŨM-^IŞE^SM5Ũç2égM-^IPŨo^Q^Ai2İM-^S^H|^D¶M-^NöZD´-^Qr\M-^W÷}M-^QPr^Vî^B²-ÇûM-^M-^Z\í%*è^Q
°^G•Yçîp÷M-^_M-^SM-^S•OöŨM-^U4alre³a=-')M-^[N4Œ¼M-^I

Ú»ÑM-^^ý;¹©û^UçXM-^[½Ýbì^W^QÖi"¼ãvÑ°Í&"M-^OÿM-^Kvw¶H½|M-^[" ^Q°j/öM-^]dM-^u5{Ýk¶£^OÿËX³
M-^JföÿdM-^B²NhuýM-^X9 &^QÀÍM-^H^XíwM-^Qý½\M\jâM-^X¿þM-^FÎO^RM-^SêyûM-^T@Ä:§k[×B@^OÝM-^G
-^HOøGßFðM-^ID^^^?]M-^]|É,þW^^Î^S¬m°^CM-^AoÚ{qéi\$¶i^Oo"fiÆèçwM-^RM-^QÓ«DÓP=^Q%~6@M-^]M-^D
³§÷øM-^W^VM-^T^Siû¿^X^@^F

^A^CÄM-^@0`

^Xü^G^@^P

```
Microsoft Visual Studio Solution File, Format Version 8.00
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "DotZLib", "DotZLib\DotZLib.csproj",
"{BB1EE0B1-1808-46CB-B786-949D91117FC5}"
    ProjectSection(ProjectDependencies) = postProject
        EndProjectSection
EndProject
Global
    GlobalSection(SolutionConfiguration) = preSolution
        Debug = Debug
        Release = Release
    EndGlobalSection
    GlobalSection(ProjectConfiguration) = postSolution
        {BB1EE0B1-1808-46CB-B786-949D91117FC5}.Debug.ActiveCfg = Debug|.NET
        {BB1EE0B1-1808-46CB-B786-949D91117FC5}.Debug.Build.0 = Debug|.NET
        {BB1EE0B1-1808-46CB-B786-949D91117FC5}.Release.ActiveCfg = Release|.NET
        {BB1EE0B1-1808-46CB-B786-949D91117FC5}.Release.Build.0 = Release|.NET
    EndGlobalSection
    GlobalSection(ExtensibilityGlobals) = postSolution
    EndGlobalSection
    GlobalSection(ExtensibilityAddIns) = postSolution
    EndGlobalSection
EndGlobal
```


Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This directory contains a .Net wrapper class library for the ZLib1.dll

The wrapper includes support for inflating/deflating memory buffers, .Net streaming wrappers for the gz streams part of zlib, and wrappers for the checksum parts of zlib. See DotZLib/UnitTests.cs for examples.

Directory structure:

LICENSE_1_0.txt	- License file.
readme.txt	- This file.
DotZLib.chm	- Class library documentation
DotZLib.build	- NAnt build file
DotZLib.sln	- Microsoft Visual Studio 2003 solution file
DotZLib*.cs	- Source files for the class library

Unit tests:

The file DotZLib/UnitTests.cs contains unit tests for use with NUnit 2.1 or higher. To include unit tests in the build, define nunit before building.

Build instructions:

1. Using Visual Studio.Net 2003:

Open DotZLib.sln in VS.Net and build from there. Output file (DotZLib.dll) will be found ./DotZLib/bin/release or ./DotZLib/bin/debug, depending on you are building the release or debug version of the library. Check DotZLib/UnitTests.cs for instructions on how to include unit tests in the build.

2. Using NAnt:

Open a command prompt with access to the build environment and run nant in the same directory as the DotZLib.build file. You can define 2 properties on the nant command-line to control the build: debug={true|false} to toggle between release/debug builds (default=true). nunit={true|false} to include or esclude unit tests (default=true). Also the target clean will remove binaries. Output file (DotZLib.dll) will be found in either ./DotZLib/bin/release or ./DotZLib/bin/debug, depending on whether you are building the release or debug version of the library.

Examples:

```
nant -D:debug=false -D:nunit=false
    will build a release mode version of the library without unit tests.
nant
    will build a debug version of the library with unit tests
nant clean
    will remove all previously built files.
```

Copyright (c) Henrik Ravn 2004

Use, modification and distribution are subject to the Boost Software License, Version 1.0
(See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)

```
using System.Reflection;
using System.Runtime.CompilerServices;

//
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
//
[assembly: AssemblyTitle("DotZLib")]
[assembly: AssemblyDescription(".Net bindings for ZLib compression dll 1.2.x")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("Henrik Ravn")]
[assembly: AssemblyProduct("")]
[assembly: AssemblyCopyright("(c) 2004 by Henrik Ravn")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

//
// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Revision and Build Numbers
// by using the '*' as shown below:

[assembly: AssemblyVersion("1.0.*")]

//
// In order to sign your assembly you must specify a key to use. Refer to the
// Microsoft .NET Framework documentation for more information on assembly signing.
//
// Use the attributes below to control which key is used for signing.
//
// Notes:
//      (*) If no key is specified, the assembly is not signed.
//      (*) KeyName refers to a key that has been installed in the Crypto Service
//      Provider (CSP) on your machine. KeyFile refers to a file which contains
//      a key.
//      (*) If the KeyFile and the KeyName values are both specified, the
//      following processing occurs:
//      (1) If the KeyName can be found in the CSP, that key is used.
//      (2) If the KeyName does not exist and the KeyFile does exist, the key
//          in the KeyFile is installed into the CSP and used.
//      (*) In order to create a KeyFile, you can use the sn.exe (Strong Name) utility.
//      When specifying the KeyFile, the location of the KeyFile should be
//      relative to the project output directory which is
//      %Project Directory%\obj\

```

```
//
// © Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License, Version
1.0.
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt
//
//

using System;
using System.Runtime.InteropServices;
using System.Text;

namespace DotZLib
{
    #region ChecksumGeneratorBase
    /// <summary>
    /// Implements the common functionality needed for all <see cref="ChecksumGenerator"/>
>s    /// </summary>
    /// <example></example>
    public abstract class ChecksumGeneratorBase : ChecksumGenerator
    {
        /// <summary>
        /// The value of the current checksum
        /// </summary>
        protected uint _current;

        /// <summary>
        /// Initializes a new instance of the checksum generator base - the current check
sum is
        /// set to zero
        /// </summary>
        public ChecksumGeneratorBase()
        {
            _current = 0;
        }

        /// <summary>
        /// Initializes a new instance of the checksum generator base with a specified val
ue
        /// </summary>
        /// <param name="initialValue">The value to set the current checksum to</param>
        public ChecksumGeneratorBase(uint initialValue)
        {
            _current = initialValue;
        }

        /// <summary>
        /// Resets the current checksum to zero
        /// </summary>
        public void Reset() { _current = 0; }

        /// <summary>
        /// Gets the current checksum value
        /// </summary>
        public uint Value { get { return _current; } }

        /// <summary>
        /// Updates the current checksum with part of an array of bytes
        /// </summary>
        /// <param name="data">The data to update the checksum with</param>
        /// <param name="offset">Where in <c>data</c> to start updating</param>
        /// <param name="count">The number of bytes from <c>data</c> to use</param>
        /// <exception cref="ArgumentException">The sum of offset and count is larger tha
n the length of <c>data</c></exception>
        /// <exception cref="NullReferenceException"><c>data</c> is a null reference</exc
eption>
        /// <exception cref="ArgumentOutOfRangeException">Offset or count is negative.</e
xception>
        /// <remarks>All the other <c>Update</c> methods are implmeneted in terms of this
one.
        /// This is therefore the only method a derived class has to implement</remarks>
    }
    #endregion
}
```

```

public abstract void Update(byte[] data, int offset, int count);

/// <summary>
/// Updates the current checksum with an array of bytes.
/// </summary>
/// <param name="data">The data to update the checksum with</param>
public void Update(byte[] data)
{
    Update(data, 0, data.Length);
}

/// <summary>
/// Updates the current checksum with the data from a string
/// </summary>
/// <param name="data">The string to update the checksum with</param>
/// <remarks>The characters in the string are converted by the UTF-8 encoding</re
marks>
public void Update(string data)
{
    Update(Encoding.UTF8.GetBytes(data));
}

/// <summary>
/// Updates the current checksum with the data from a string, using a specific en
coding
/// </summary>
/// <param name="data">The string to update the checksum with</param>
/// <param name="encoding">The encoding to use</param>
public void Update(string data, Encoding encoding)
{
    Update(encoding.GetBytes(data));
}

}
#endregion

#region CRC32
/// <summary>
/// Implements a CRC32 checksum generator
/// </summary>
public sealed class CRC32Checksum : ChecksumGeneratorBase
{
    #region DLL imports

    [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
    private static extern uint crc32(uint crc, int data, uint length);

    #endregion

    /// <summary>
    /// Initializes a new instance of the CRC32 checksum generator
    /// </summary>
    public CRC32Checksum() : base() {}

    /// <summary>
    /// Initializes a new instance of the CRC32 checksum generator with a specified v
alue
    /// </summary>
    /// <param name="initialValue">The value to set the current checksum to</param>
    public CRC32Checksum(uint initialValue) : base(initialValue) {}

    /// <summary>
    /// Updates the current checksum with part of an array of bytes
    /// </summary>
    /// <param name="data">The data to update the checksum with</param>
    /// <param name="offset">Where in <c>data</c> to start updating</param>
    /// <param name="count">The number of bytes from <c>data</c> to use</param>
    /// <exception cref="ArgumentException">The sum of offset and count is larger tha
n the length of <c>data</c></exception>
    /// <exception cref="NullReferenceException"><c>data</c> is a null reference</exc
eption>
    /// <exception cref="ArgumentOutOfRangeException">Offset or count is negative.</e
xception>
    public override void Update(byte[] data, int offset, int count)

```

```

        {
            if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException();
            if ((offset+count) > data.Length) throw new ArgumentException();
            GCHandle hData = GCHandle.Alloc(data, GCHandleType.Pinned);
            try
            {
                _current = crc32(_current, hData.AddrOfPinnedObject().ToInt32()+offset, (
uint)count);
            }
            finally
            {
                hData.Free();
            }
        }
    }
}
#endregion

#region Adler
/// <summary>
/// Implements a checksum generator that computes the Adler checksum on data
/// </summary>
public sealed class AdlerChecksum : ChecksumGeneratorBase
{
    #region DLL imports

    [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
    private static extern uint adler32(uint adler, int data, uint length);

    #endregion

    /// <summary>
    /// Initializes a new instance of the Adler checksum generator
    /// </summary>
    public AdlerChecksum() : base() {}

    /// <summary>
    /// Initializes a new instance of the Adler checksum generator with a specified v
alue
    /// </summary>
    /// <param name="initialValue">The value to set the current checksum to</param>
    public AdlerChecksum(uint initialValue) : base(initialValue) {}

    /// <summary>
    /// Updates the current checksum with part of an array of bytes
    /// </summary>
    /// <param name="data">The data to update the checksum with</param>
    /// <param name="offset">Where in <c>data</c> to start updating</param>
    /// <param name="count">The number of bytes from <c>data</c> to use</param>
    /// <exception cref="ArgumentException">The sum of offset and count is larger tha
n the length of <c>data</c></exception>
    /// <exception cref="NullReferenceException"><c>data</c> is a null reference</exc
eption>
    /// <exception cref="ArgumentOutOfRangeException">Offset or count is negative.</e
xception>
    public override void Update(byte[] data, int offset, int count)
    {
        if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException();
        if ((offset+count) > data.Length) throw new ArgumentException();
        GCHandle hData = GCHandle.Alloc(data, GCHandleType.Pinned);
        try
        {
            _current = adler32(_current, hData.AddrOfPinnedObject().ToInt32()+offset,
(uint)count);
        }
        finally
        {
            hData.Free();
        }
    }
}
}
#endregion

```

```
}
```

```
//
// © Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License, Version
// 1.0.
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt
// )
//

using System;
using System.Diagnostics;

namespace DotZLib
{
    /// <summary>
    /// This class implements a circular buffer
    /// </summary>
    internal class CircularBuffer
    {
        #region Private data
        private int _capacity;
        private int _head;
        private int _tail;
        private int _size;
        private byte[] _buffer;
        #endregion

        public CircularBuffer(int capacity)
        {
            Debug.Assert( capacity > 0 );
            _buffer = new byte[capacity];
            _capacity = capacity;
            _head = 0;
            _tail = 0;
            _size = 0;
        }

        public int Size { get { return _size; } }

        public int Put(byte[] source, int offset, int count)
        {
            Debug.Assert( count > 0 );
            int trueCount = Math.Min(count, _capacity - Size);
            for (int i = 0; i < trueCount; ++i)
                _buffer[( _tail+i ) % _capacity] = source[offset+i];
            _tail += trueCount;
            _tail %= _capacity;
            _size += trueCount;
            return trueCount;
        }

        public bool Put(byte b)
        {
            if (Size == _capacity) // no room
                return false;
            _buffer[_tail++] = b;
            _tail %= _capacity;
            ++_size;
            return true;
        }

        public int Get(byte[] destination, int offset, int count)
        {
            int trueCount = Math.Min(count, Size);
            for (int i = 0; i < trueCount; ++i)
                destination[offset + i] = _buffer[( _head+i ) % _capacity];
            _head += trueCount;
            _head %= _capacity;
            _size -= trueCount;
            return trueCount;
        }

        public int Get()

```



```
        {  
            if (Size == 0)  
                return -1;  
  
            int result = (int)_buffer[_head++ % _capacity];  
            --_size;  
            return result;  
        }  
    }  
}
```

```
//
// © Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License, Version
// 1.0.
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt
// )
//

using System;
using System.Runtime.InteropServices;

namespace DotZLib
{
    /// <summary>
    /// Implements the common functionality needed for all <see cref="Codec"/>s
    /// </summary>
    public abstract class CodecBase : Codec, IDisposable
    {
        #region Data members

        /// <summary>
        /// Instance of the internal zlib buffer structure that is
        /// passed to all functions in the zlib dll
        /// </summary>
        internal ZStream _zstream = new ZStream();

        /// <summary>
        /// True if the object instance has been disposed, false otherwise
        /// </summary>
        protected bool _isDisposed = false;

        /// <summary>
        /// The size of the internal buffers
        /// </summary>
        protected const int kBufferSize = 16384;

        private byte[] _outBuffer = new byte[kBufferSize];
        private byte[] _inBuffer = new byte[kBufferSize];

        private GCHandle _hInput;
        private GCHandle _hOutput;

        private uint _checksum = 0;

        #endregion

        /// <summary>
        /// Initializes a new instance of the <c>CodeBase</c> class.
        /// </summary>
        public CodecBase()
        {
            try
            {
                _hInput = GCHandle.Alloc(_inBuffer, GCHandleType.Pinned);
                _hOutput = GCHandle.Alloc(_outBuffer, GCHandleType.Pinned);
            }
            catch (Exception)
            {
                Cleanup(false);
                throw;
            }
        }

        #region Codec Members

        /// <summary>
        /// Occurs when more processed data are available.
        /// </summary>
        public event DataAvailableHandler DataAvailable;

        /// <summary>
```

```

    /// Fires the <see cref="DataAvailable"/> event
    /// </summary>
    protected void OnDataAvailable()
    {
        if (_zstream.total_out > 0)
        {
            if (DataAvailable != null)
                DataAvailable( _outBuffer, 0, (int)_zstream.total_out);
            resetOutput();
        }
    }

    /// <summary>
    /// Adds more data to the codec to be processed.
    /// </summary>
    /// <param name="data">Byte array containing the data to be added to the codec</p
aram>
    /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c> event</r
emarks>
    public void Add(byte[] data)
    {
        Add(data, 0, data.Length);
    }

    /// <summary>
    /// Adds more data to the codec to be processed.
    /// </summary>
    /// <param name="data">Byte array containing the data to be added to the codec</p
aram>
    /// <param name="offset">The index of the first byte to add from <c>data</c></par
am>
    /// <param name="count">The number of bytes to add</param>
    /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c> event</r
emarks>
    /// <remarks>This must be implemented by a derived class</remarks>
    public abstract void Add(byte[] data, int offset, int count);

    /// <summary>
    /// Finishes up any pending data that needs to be processed and handled.
    /// </summary>
    /// <remarks>This must be implemented by a derived class</remarks>
    public abstract void Finish();

    /// <summary>
    /// Gets the checksum of the data that has been added so far
    /// </summary>
    public uint Checksum { get { return _checksum; } }

    #endregion

    #region Destructor & IDisposable stuff

    /// <summary>
    /// Destroys this instance
    /// </summary>
    ~CodecBase()
    {
        CleanUp(false);
    }

    /// <summary>
    /// Releases any unmanaged resources and calls the <see cref="CleanUp()"/> method
of the derived class
    /// </summary>
    public void Dispose()
    {
        CleanUp(true);
    }

    /// <summary>
    /// Performs any codec specific cleanup
    /// </summary>
    /// <remarks>This must be implemented by a derived class</remarks>
    protected abstract void CleanUp();

```

```
// performs the release of the handles and calls the dereived CleanUp()
private void CleanUp(bool isDisposing)
{
    if (!_isDisposed)
    {
        CleanUp();
        if (_hInput.IsAllocated)
            _hInput.Free();
        if (_hOutput.IsAllocated)
            _hOutput.Free();

        _isDisposed = true;
    }
}

#endregion

#region Helper methods

/// <summary>
/// Copies a number of bytes to the internal codec buffer - ready for proccesing
/// </summary>
/// <param name="data">The byte array that contains the data to copy</param>
/// <param name="startIndex">The index of the first byte to copy</param>
/// <param name="count">The number of bytes to copy from <c>data</c></param>
protected void copyInput(byte[] data, int startIndex, int count)
{
    Array.Copy(data, startIndex, _inBuffer, 0, count);
    _zstream.next_in = _hInput.AddrOfPinnedObject();
    _zstream.total_in = 0;
    _zstream.avail_in = (uint)count;
}

/// <summary>
/// Resets the internal output buffers to a known state - ready for processing
/// </summary>
protected void resetOutput()
{
    _zstream.total_out = 0;
    _zstream.avail_out = kBufferSize;
    _zstream.next_out = _hOutput.AddrOfPinnedObject();
}

/// <summary>
/// Updates the running checksum property
/// </summary>
/// <param name="newSum">The new checksum value</param>
protected void setChecksum(uint newSum)
{
    _checksum = newSum;
}
#endregion
}
}
```

```

//
// © Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License, Version
1.0.
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt
)
//

using System;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace DotZLib
{
    /// <summary>
    /// Implements a data compressor, using the deflate algorithm in the ZLib dll
    /// </summary>
    public sealed class Deflater : CodecBase
    {
        #region Dll imports
        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl, CharSet=CharSe
t.Ansi)]
        private static extern int deflateInit_(ref ZStream sz, int level, string vs, int
size);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int deflate(ref ZStream sz, int flush);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int deflateReset(ref ZStream sz);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int deflateEnd(ref ZStream sz);
        #endregion

        /// <summary>
        /// Constructs an new instance of the <c>Deflater</c>
        /// </summary>
        /// <param name="level">The compression level to use for this <c>Deflater</c></pa
ram>
        public Deflater(CompressLevel level) : base()
        {
            int retval = deflateInit_(ref _zstream, (int)level, Info.Version, Marshal.Size
Of(_zstream));
            if (retval != 0)
                throw new ZLibException(retval, "Could not initialize deflater");

            resetOutput();
        }

        /// <summary>
        /// Adds more data to the codec to be processed.
        /// </summary>
        /// <param name="data">Byte array containing the data to be added to the codec</p
aram>
        /// <param name="offset">The index of the first byte to add from <c>data</c></par
am>
        /// <param name="count">The number of bytes to add</param>
        /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c> event</r
emarks>
        public override void Add(byte[] data, int offset, int count)
        {
            if (data == null) throw new ArgumentNullException();
            if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException();
            if ((offset+count) > data.Length) throw new ArgumentException();

            int total = count;
            int inputIndex = offset;
            int err = 0;

            while (err >= 0 && inputIndex < total)
            {

```

```
        copyInput(data, inputIndex, Math.Min(total - inputIndex, kBufferSize));
        while (err >= 0 && _zstream.avail_in > 0)
        {
            err = deflate(ref _zstream, (int)FlushTypes.None);
            if (err == 0)
                while (_zstream.avail_out == 0)
                {
                    OnDataAvailable();
                    err = deflate(ref _zstream, (int)FlushTypes.None);
                }
            inputIndex += (int)_zstream.total_in;
        }
    }
    setChecksum( _zstream.adler );
}
```

```
/// <summary>
/// Finishes up any pending data that needs to be processed and handled.
/// </summary>
```

```
public override void Finish()
```

```
{
    int err;
    do
    {
        err = deflate(ref _zstream, (int)FlushTypes.Finish);
        OnDataAvailable();
    }
    while (err == 0);
    setChecksum( _zstream.adler );
    deflateReset(ref _zstream);
    resetOutput();
}
```

```
/// <summary>
/// Closes the internal zlib deflate stream
/// </summary>
```

```
protected override void CleanUp() { deflateEnd(ref _zstream); }
```

```
    }
}
```

```
//  
// © Copyright Henrik Ravn 2004  
//  
// Use, modification and distribution are subject to the Boost Software License, Version  
1.0.  
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt  
)  
//  
  
using System;  
using System.IO;  
using System.Runtime.InteropServices;  
using System.Text;  
  
namespace DotZLib  
{  
  
    #region Internal types  
  
    /// <summary>  
    /// Defines constants for the various flush types used with zlib  
    /// </summary>  
    internal enum FlushTypes  
    {  
        None, Partial, Sync, Full, Finish, Block  
    }  
  
    #region ZStream structure  
    // internal mapping of the zlib zstream structure for marshalling  
    [StructLayoutAttribute(LayoutKind.Sequential, Pack=4, Size=0, CharSet=CharSet.Ansi)]  
    internal struct ZStream  
    {  
        public IntPtr next_in;  
        public uint avail_in;  
        public uint total_in;  
  
        public IntPtr next_out;  
        public uint avail_out;  
        public uint total_out;  
  
        [MarshalAs(UnmanagedType.LPStr)]  
        string msg;  
        uint state;  
  
        uint zalloc;  
        uint zfree;  
        uint opaque;  
  
        int data_type;  
        public uint Adler;  
        uint reserved;  
    }  
  
    #endregion  
  
    #endregion  
  
    #region Public enums  
    /// <summary>  
    /// Defines constants for the available compression levels in zlib  
    /// </summary>  
    public enum CompressLevel : int  
    {  
        /// <summary>  
        /// The default compression level with a reasonable compromise between compressio  
n and speed  
        /// </summary>  
        Default = -1,  
        /// <summary>  
        /// No compression at all. The data are passed straight through.  
        /// </summary>  
        None = 0,  
        /// <summary>
```

```

    /// The maximum compression rate available.
    /// </summary>
    Best = 9,
    /// <summary>
    /// The fastest available compression level.
    /// </summary>
    Fastest = 1
}
#endregion

#region Exception classes
/// <summary>
/// The exception that is thrown when an error occurs on the zlib dll
/// </summary>
public class ZLibException : ApplicationException
{
    /// <summary>
    /// Initializes a new instance of the <see cref="ZLibException"/> class with a sp
ecified
    /// error message and error code
    /// </summary>
    /// <param name="errorCode">The zlib error code that caused the exception</param>
    /// <param name="msg">A message that (hopefully) describes the error</param>
    public ZLibException(int errorCode, string msg) : base(String.Format("ZLib error
{0} {1}", errorCode, msg))
    {
    }

    /// <summary>
    /// Initializes a new instance of the <see cref="ZLibException"/> class with a sp
ecified
    /// error code
    /// </summary>
    /// <param name="errorCode">The zlib error code that caused the exception</param>
    public ZLibException(int errorCode) : base(String.Format("ZLib error {0}", errorC
ode))
    {
    }
}
#endregion

#region Interfaces
/// <summary>
/// Declares methods and properties that enables a running checksum to be calculated
/// </summary>
public interface ChecksumGenerator
{
    /// <summary>
    /// Gets the current value of the checksum
    /// </summary>
    uint Value { get; }

    /// <summary>
    /// Clears the current checksum to 0
    /// </summary>
    void Reset();

    /// <summary>
    /// Updates the current checksum with an array of bytes
    /// </summary>
    /// <param name="data">The data to update the checksum with</param>
    void Update(byte[] data);

    /// <summary>
    /// Updates the current checksum with part of an array of bytes
    /// </summary>
    /// <param name="data">The data to update the checksum with</param>
    /// <param name="offset">Where in <c>data</c> to start updating</param>
    /// <param name="count">The number of bytes from <c>data</c> to use</param>
    /// <exception cref="ArgumentException">The sum of offset and count is larger tha
n the length of <c>data</c></exception>
    /// <exception cref="ArgumentNullException"><c>data</c> is a null reference</exce
ption>

```



```
    /// <exception cref="ArgumentOutOfRangeException">Offset or count is negative.</e
exception>
    void Update(byte[] data, int offset, int count);

    /// <summary>
    /// Updates the current checksum with the data from a string
    /// </summary>
    /// <param name="data">The string to update the checksum with</param>
    /// <remarks>The characters in the string are converted by the UTF-8 encoding</re
marks>
    void Update(string data);

    /// <summary>
    /// Updates the current checksum with the data from a string, using a specific en
coding
    /// </summary>
    /// <param name="data">The string to update the checksum with</param>
    /// <param name="encoding">The encoding to use</param>
    void Update(string data, Encoding encoding);
}

    /// <summary>
    /// Represents the method that will be called from a codec when new data
    /// are available.
    /// </summary>
    /// <paramref name="data">The byte array containing the processed data</paramref>
    /// <paramref name="startIndex">The index of the first processed byte in <c>data</c><
/paramref>
    /// <paramref name="count">The number of processed bytes available</paramref>
    /// <remarks>On return from this method, the data may be overwritten, so grab it whil
e you can.
    /// You cannot assume that startIndex will be zero.
    /// </remarks>
    public delegate void DataAvailableHandler(byte[] data, int startIndex, int count);

    /// <summary>
    /// Declares methods and events for implementing compressors/decompressors
    /// </summary>
    public interface Codec
    {
        /// <summary>
        /// Occurs when more processed data are available.
        /// </summary>
        event DataAvailableHandler DataAvailable;

        /// <summary>
        /// Adds more data to the codec to be processed.
        /// </summary>
        /// <param name="data">Byte array containing the data to be added to the codec</p
aram>
        /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c> event</r
emarks>
        void Add(byte[] data);

        /// <summary>
        /// Adds more data to the codec to be processed.
        /// </summary>
        /// <param name="data">Byte array containing the data to be added to the codec</p
aram>
        /// <param name="offset">The index of the first byte to add from <c>data</c></par
am>
        /// <param name="count">The number of bytes to add</param>
        /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c> event</r
emarks>
        void Add(byte[] data, int offset, int count);

        /// <summary>
        /// Finishes up any pending data that needs to be processed and handled.
        /// </summary>
        void Finish();

        /// <summary>
        /// Gets the checksum of the data that has been added so far
```

```

    /// </summary>
    uint Checksum { get; }

}

#endregion

#region Classes
/// <summary>
/// Encapsulates general information about the ZLib library
/// </summary>
public class Info
{
    #region DLL imports
    [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
    private static extern uint zlibCompileFlags();

    [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
    private static extern string zlibVersion();
    #endregion

    #region Private stuff
    private uint _flags;

    // helper function that unpacks a bitsize mask
    private static int bitSize(uint bits)
    {
        switch (bits)
        {
            case 0: return 16;
            case 1: return 32;
            case 2: return 64;
        }
        return -1;
    }
    #endregion

    /// <summary>
    /// Constructs an instance of the <c>Info</c> class.
    /// </summary>
    public Info()
    {
        _flags = zlibCompileFlags();
    }

    /// <summary>
    /// True if the library is compiled with debug info
    /// </summary>
    public bool HasDebugInfo { get { return 0 != (_flags & 0x100); } }

    /// <summary>
    /// True if the library is compiled with assembly optimizations
    /// </summary>
    public bool UsesAssemblyCode { get { return 0 != (_flags & 0x200); } }

    /// <summary>
    /// Gets the size of the unsigned int that was compiled into Zlib
    /// </summary>
    public int SizeOfUInt { get { return bitSize(_flags & 3); } }

    /// <summary>
    /// Gets the size of the unsigned long that was compiled into Zlib
    /// </summary>
    public int SizeOfULong { get { return bitSize((_flags >> 2) & 3); } }

    /// <summary>
    /// Gets the size of the pointers that were compiled into Zlib
    /// </summary>
    public int SizeOfPointer { get { return bitSize((_flags >> 4) & 3); } }

    /// <summary>
    /// Gets the size of the z_off_t type that was compiled into Zlib
    /// </summary>

```

```
public int SizeOfOffset { get { return bitSize((_flags >> 6) & 3); } }

/// <summary>
/// Gets the version of ZLib as a string, e.g. "1.2.1"
/// </summary>
public static string Version { get { return zlibVersion(); } }
}

#endregion
}
```

```
<VisualStudioProject>
  <CSHARP
    ProjectType = "Local"
    ProductVersion = "7.10.3077"
    SchemaVersion = "2.0"
    ProjectGuid = "{BB1EE0B1-1808-46CB-B786-949D91117FC5}"
  >
    <Build>
      <Settings
        ApplicationIcon = ""
        AssemblyKeyContainerName = ""
        AssemblyName = "DotZLib"
        AssemblyOriginatorKeyFile = ""
        DefaultClientScript = "JScript"
        DefaultHTMLPageLayout = "Grid"
        DefaultTargetSchema = "IE50"
        DelaySign = "false"
        OutputType = "Library"
        PreBuildEvent = ""
        PostBuildEvent = ""
        RootNamespace = "DotZLib"
        RunPostBuildEvent = "OnBuildSuccess"
        StartupObject = ""
      >
        <Config
          Name = "Debug"
          AllowUnsafeBlocks = "false"
          BaseAddress = "285212672"
          CheckForOverflowUnderflow = "false"
          ConfigurationOverrideFile = ""
          DefineConstants = "DEBUG;TRACE"
          DocumentationFile = "docs\DotZLib.xml"
          DebugSymbols = "true"
          FileAlignment = "4096"
          IncrementalBuild = "false"
          NoStdLib = "false"
          NoWarn = "1591"
          Optimize = "false"
          OutputPath = "bin\Debug\"
          RegisterForComInterop = "false"
          RemoveIntegerChecks = "false"
          TreatWarningsAsErrors = "false"
          WarningLevel = "4"
        />
        <Config
          Name = "Release"
          AllowUnsafeBlocks = "false"
          BaseAddress = "285212672"
          CheckForOverflowUnderflow = "false"
          ConfigurationOverrideFile = ""
          DefineConstants = "TRACE"
          DocumentationFile = "docs\DotZLib.xml"
          DebugSymbols = "false"
          FileAlignment = "4096"
          IncrementalBuild = "false"
          NoStdLib = "false"
          NoWarn = ""
          Optimize = "true"
          OutputPath = "bin\Release\"
          RegisterForComInterop = "false"
          RemoveIntegerChecks = "false"
          TreatWarningsAsErrors = "false"
          WarningLevel = "4"
        />
      </Settings>
      <References>
        <Reference
          Name = "System"
          AssemblyName = "System"
          HintPath = "C:\WINNT\Microsoft.NET\Framework\v1.1.4322\System.dll"
        />
        <Reference
          Name = "System.Data"
          AssemblyName = "System.Data"
        />
      </References>
    </Build>
  </CSHARP>
</VisualStudioProject>
```

```

1"          HintPath = "C:\WINNT\Microsoft.NET\Framework\v1.1.4322\System.Data.dll"
        />
        <Reference
            Name = "System.XML"
            AssemblyName = "System.Xml"
            HintPath = "C:\WINNT\Microsoft.NET\Framework\v1.1.4322\System.XML.dll"
        />
        <Reference
            Name = "nunit.framework"
            AssemblyName = "nunit.framework"
            HintPath = "E:\apps\NUnit V2.1\bin\nunit.framework.dll"
            AssemblyFolderKey = "hkml\dn\nunit.framework"
        />
    </References>
</Build>
<Files>
    <Include>
        <File
            RelPath = "AssemblyInfo.cs"
            SubType = "Code"
            BuildAction = "Compile"
        />
        <File
            RelPath = "ChecksumImpl.cs"
            SubType = "Code"
            BuildAction = "Compile"
        />
        <File
            RelPath = "CircularBuffer.cs"
            SubType = "Code"
            BuildAction = "Compile"
        />
        <File
            RelPath = "CodecBase.cs"
            SubType = "Code"
            BuildAction = "Compile"
        />
        <File
            RelPath = "Deflater.cs"
            SubType = "Code"
            BuildAction = "Compile"
        />
        <File
            RelPath = "DotZLib.cs"
            SubType = "Code"
            BuildAction = "Compile"
        />
        <File
            RelPath = "GZipStream.cs"
            SubType = "Code"
            BuildAction = "Compile"
        />
        <File
            RelPath = "Inflater.cs"
            SubType = "Code"
            BuildAction = "Compile"
        />
        <File
            RelPath = "UnitTests.cs"
            SubType = "Code"
            BuildAction = "Compile"
        />
    </Include>
</Files>
</CSHARP>
</VisualStudioProject>

```

```
//
// © Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License, Version
// 1.0.
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt
//)
//

using System;
using System.IO;
using System.Runtime.InteropServices;

namespace DotZLib
{
    /// <summary>
    /// Implements a compressed <see cref="Stream"/>, in GZip (.gz) format.
    /// </summary>
    public class GZipStream : Stream, IDisposable
    {
        #region Dll Imports
        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl, CharSet=CharSe
t.Ansi)]
        private static extern IntPtr gzopen(string name, string mode);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int gzclose(IntPtr gzFile);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int gzwrite(IntPtr gzFile, int data, int length);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int gzread(IntPtr gzFile, int data, int length);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int gzgetc(IntPtr gzFile);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int gzputc(IntPtr gzFile, int c);

        #endregion

        #region Private data
        private IntPtr _gzFile;
        private bool _isDisposed = false;
        private bool _isWriting;
        #endregion

        #region Constructors
        /// <summary>
        /// Creates a new file as a writeable GZipStream
        /// </summary>
        /// <param name="fileName">The name of the compressed file to create</param>
        /// <param name="level">The compression level to use when adding data</param>
        /// <exception cref="ZLibException">If an error occurred in the internal zlib fun
ction</exception>
        public GZipStream(string fileName, CompressLevel level)
        {
            _isWriting = true;
            _gzFile = gzopen(fileName, String.Format("wb{0}", (int)level));
            if (_gzFile == IntPtr.Zero)
                throw new ZLibException(-1, "Could not open " + fileName);
        }

        /// <summary>
        /// Opens an existing file as a readable GZipStream
        /// </summary>
        /// <param name="fileName">The name of the file to open</param>
        /// <exception cref="ZLibException">If an error occurred in the internal zlib fun
ction</exception>
        public GZipStream(string fileName)
        {
            _isWriting = false;
            _gzFile = gzopen(fileName, "rb");
        }
    }
}
```

```

        if (_gzFile == IntPtr.Zero)
            throw new ZLibException(-1, "Could not open " + fileName);
    }
#endregion

#region Access properties
/// <summary>
/// Returns true of this stream can be read from, false otherwise
/// </summary>
public override bool CanRead
{
    get
    {
        return !_isWriting;
    }
}

/// <summary>
/// Returns false.
/// </summary>
public override bool CanSeek
{
    get
    {
        return false;
    }
}

/// <summary>
/// Returns true if this tstream is writeable, false otherwise
/// </summary>
public override bool CanWrite
{
    get
    {
        return _isWriting;
    }
}
#endregion

#region Destructor & IDispose stuff

/// <summary>
/// Destroys this instance
/// </summary>
~GZipStream()
{
    cleanUp(false);
}

/// <summary>
/// Closes the external file handle
/// </summary>
public void Dispose()
{
    cleanUp(true);
}

// Does the actual closing of the file handle.
private void cleanUp(bool isDisposing)
{
    if (!_isDisposed)
    {
        gzclose(_gzFile);
        _isDisposed = true;
    }
}
#endregion

#region Basic reading and writing
/// <summary>
/// Attempts to read a number of bytes from the stream.

```

```

    /// </summary>
    /// <param name="buffer">The destination data buffer</param>
    /// <param name="offset">The index of the first destination byte in <c>buffer</c>
</param>
    /// <param name="count">The number of bytes requested</param>
    /// <returns>The number of bytes read</returns>
    /// <exception cref="ArgumentNullException">If <c>buffer</c> is null</exception>
    /// <exception cref="ArgumentOutOfRangeException">If <c>count</c> or <c>offset</c>
> are negative</exception>
    /// <exception cref="ArgumentException">If <c>offset</c> + <c>count</c> is >
buffer.Length</exception>
    /// <exception cref="NotSupportedException">If this stream is not readable.</exce
ption>
    /// <exception cref="ObjectDisposedException">If this stream has been disposed.</
exception>
    public override int Read(byte[] buffer, int offset, int count)
    {
        if (!CanRead) throw new NotSupportedException();
        if (buffer == null) throw new ArgumentNullException();
        if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException();
        if ((offset+count) > buffer.Length) throw new ArgumentException();
        if (_isDisposed) throw new ObjectDisposedException("GZipStream");

        GCHandle h = GCHandle.Alloc(buffer, GCHandleType.Pinned);
        int result;
        try
        {
            result = gzread(_gzFile, h.AddrOfPinnedObject().ToInt32() + offset, count
);
            if (result < 0)
                throw new IOException();
        }
        finally
        {
            h.Free();
        }
        return result;
    }

    /// <summary>
    /// Attempts to read a single byte from the stream.
    /// </summary>
    /// <returns>The byte that was read, or -1 in case of error or End-Of-File</retur
ns>
    public override int ReadByte()
    {
        if (!CanRead) throw new NotSupportedException();
        if (_isDisposed) throw new ObjectDisposedException("GZipStream");
        return gzgetc(_gzFile);
    }

    /// <summary>
    /// Writes a number of bytes to the stream
    /// </summary>
    /// <param name="buffer"></param>
    /// <param name="offset"></param>
    /// <param name="count"></param>
    /// <exception cref="ArgumentNullException">If <c>buffer</c> is null</exception>
    /// <exception cref="ArgumentOutOfRangeException">If <c>count</c> or <c>offset</c>
> are negative</exception>
    /// <exception cref="ArgumentException">If <c>offset</c> + <c>count</c> is >
buffer.Length</exception>
    /// <exception cref="NotSupportedException">If this stream is not writeable.</exc
eption>
    /// <exception cref="ObjectDisposedException">If this stream has been disposed.</
exception>
    public override void Write(byte[] buffer, int offset, int count)
    {
        if (!CanWrite) throw new NotSupportedException();
        if (buffer == null) throw new ArgumentNullException();
        if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException();
        if ((offset+count) > buffer.Length) throw new ArgumentException();
        if (_isDisposed) throw new ObjectDisposedException("GZipStream");

```



```

        GCHandle h = GCHandle.Alloc(buffer, GCHandleType.Pinned);
        try
        {
            int result = gzwrite(_gzFile, h.AddrOfPinnedObject().ToInt32() + offset,
count);
            if (result < 0)
                throw new IOException();
        }
        finally
        {
            h.Free();
        }
    }

    /// <summary>
    /// Writes a single byte to the stream
    /// </summary>
    /// <param name="value">The byte to add to the stream.</param>
    /// <exception cref="NotSupportedException">If this stream is not writeable.</exc
exception>
    /// <exception cref="ObjectDisposedException">If this stream has been disposed.</
exception>
    public override void WriteByte(byte value)
    {
        if (!CanWrite) throw new NotSupportedException();
        if (_isDisposed) throw new ObjectDisposedException("GZipStream");

        int result = gzputc(_gzFile, (int)value);
        if (result < 0)
            throw new IOException();
    }
#endregion

#region Position & length stuff
    /// <summary>
    /// Not supported.
    /// </summary>
    /// <param name="value"></param>
    /// <exception cref="NotSupportedException">Always thrown</exception>
    public override void SetLength(long value)
    {
        throw new NotSupportedException();
    }

    /// <summary>
    /// Not supported.
    /// </summary>
    /// <param name="offset"></param>
    /// <param name="origin"></param>
    /// <returns></returns>
    /// <exception cref="NotSupportedException">Always thrown</exception>
    public override long Seek(long offset, SeekOrigin origin)
    {
        throw new NotSupportedException();
    }

    /// <summary>
    /// Flushes the <c>GZipStream</c>.
    /// </summary>
    /// <remarks>In this implementation, this method does nothing. This is because ex
cessive
    /// flushing may degrade the achievable compression rates.</remarks>
    public override void Flush()
    {
        // left empty on purpose
    }

    /// <summary>
    /// Gets/sets the current position in the <c>GZipStream</c>. Not supported.
    /// </summary>
    /// <remarks>In this implementation this property is not supported</remarks>
    /// <exception cref="NotSupportedException">Always thrown</exception>
    public override long Position
    {

```

```

        get
        {
            throw new NotSupportedException();
        }
        set
        {
            throw new NotSupportedException();
        }
    }

    /// <summary>
    /// Gets the size of the stream. Not supported.
    /// </summary>
    /// <remarks>In this implementation this property is not supported</remarks>
    /// <exception cref="NotSupportedException">Always thrown</exception>
    public override long Length
    {
        get
        {
            throw new NotSupportedException();
        }
    }
    #endregion
}

```

```

//
// © Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License, Version
1.0.
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt
)
//

using System;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace DotZLib
{
    /// <summary>
    /// Implements a data decompressor, using the inflate algorithm in the ZLib dll
    /// </summary>
    public class Inflater : CodecBase
    {
        #region Dll imports
        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl, CharSet=CharSe
t.Ansi)]
        private static extern int inflateInit_(ref ZStream sz, string vs, int size);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int inflate(ref ZStream sz, int flush);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int inflateReset(ref ZStream sz);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int inflateEnd(ref ZStream sz);
        #endregion

        /// <summary>
        /// Constructs an new instance of the <c>Inflater</c>
        /// </summary>
        public Inflater() : base()
        {
            int retval = inflateInit_(ref _zstream, Info.Version, Marshal.SizeOf(_zstream))
;
            if (retval != 0)
                throw new ZLibException(retval, "Could not initialize inflater");

            resetOutput();
        }

        /// <summary>
        /// Adds more data to the codec to be processed.
        /// </summary>
        /// <param name="data">Byte array containing the data to be added to the codec</p
aram>
        /// <param name="offset">The index of the first byte to add from <c>data</c></par
am>
        /// <param name="count">The number of bytes to add</param>
        /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c> event</r
emarks>
        public override void Add(byte[] data, int offset, int count)
        {
            if (data == null) throw new ArgumentNullException();
            if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException();
            if ((offset+count) > data.Length) throw new ArgumentException();

            int total = count;
            int inputIndex = offset;
            int err = 0;

            while (err >= 0 && inputIndex < total)
            {
                copyInput(data, inputIndex, Math.Min(total - inputIndex, kBufferSize));
                err = inflate(ref _zstream, (int)FlushTypes.None);
            }
        }
    }
}

```

```
        if (err == 0)
            while (_zstream.avail_out == 0)
            {
                OnDataAvailable();
                err = inflate(ref _zstream, (int)FlushTypes.None);
            }

        inputIndex += (int)_zstream.total_in;
    }
    setChecksum( _zstream.adler );
}

/// <summary>
/// Finishes up any pending data that needs to be processed and handled.
/// </summary>
public override void Finish()
{
    int err;
    do
    {
        err = inflate(ref _zstream, (int)FlushTypes.Finish);
        OnDataAvailable();
    }
    while (err == 0);
    setChecksum( _zstream.adler );
    inflateReset(ref _zstream);
    resetOutput();
}

/// <summary>
/// Closes the internal zlib inflate stream
/// </summary>
protected override void CleanUp() { inflateEnd(ref _zstream); }

}
```

```
//
// © Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License, Version
1.0.
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt
//
//

using System;
using System.Collections;
using System.IO;

// uncomment the define below to include unit tests
// #define nunit
// #if nunit
using NUnit.Framework;

// Unit tests for the DotZLib class library
// -----
//
// Use this with NUnit 2 from http://www.nunit.org
//

namespace DotZLibTests
{
    using DotZLib;

    // helper methods
    internal class Utils
    {
        public static bool byteArrEqual( byte[] lhs, byte[] rhs )
        {
            if (lhs.Length != rhs.Length)
                return false;
            for (int i = lhs.Length-1; i >= 0; --i)
                if (lhs[i] != rhs[i])
                    return false;
            return true;
        }
    }

    [TestFixture]
    public class CircBufferTests
    {
        #region Circular buffer tests
        [Test]
        public void SinglePutGet()
        {
            CircularBuffer buf = new CircularBuffer(10);
            Assert.AreEqual( 0, buf.Size );
            Assert.AreEqual( -1, buf.Get() );

            Assert.IsTrue(buf.Put( 1 ));
            Assert.AreEqual( 1, buf.Size );
            Assert.AreEqual( 1, buf.Get() );
            Assert.AreEqual( 0, buf.Size );
            Assert.AreEqual( -1, buf.Get() );
        }

        [Test]
        public void BlockPutGet()
        {
            CircularBuffer buf = new CircularBuffer(10);
            byte[] arr = {1,2,3,4,5,6,7,8,9,10};
            Assert.AreEqual( 10, buf.Put(arr,0,10) );
            Assert.AreEqual( 10, buf.Size );
            Assert.IsFalse( buf.Put(11) );
            Assert.AreEqual( 1, buf.Get() );
            Assert.IsTrue( buf.Put(11) );

            byte[] arr2 = (byte[])arr.Clone();
        }
    }
}
```

```

        Assert.AreEqual( 9, buf.Get(arr2,1,9) );
        Assert.IsTrue( Utils.byteArrEqual(arr,arr2) );
    }

    #endregion
}

[TestFixture]
public class ChecksumTests
{
    #region CRC32 Tests
    [Test]
    public void CRC32_Null()
    {
        CRC32Checksum crc32 = new CRC32Checksum();
        Assert.AreEqual( 0, crc32.Value );

        crc32 = new CRC32Checksum(1);
        Assert.AreEqual( 1, crc32.Value );

        crc32 = new CRC32Checksum(556);
        Assert.AreEqual( 556, crc32.Value );
    }

    [Test]
    public void CRC32_Data()
    {
        CRC32Checksum crc32 = new CRC32Checksum();
        byte[] data = { 1,2,3,4,5,6,7 };
        crc32.Update(data);
        Assert.AreEqual( 0x70e46888, crc32.Value );

        crc32 = new CRC32Checksum();
        crc32.Update("penguin");
        Assert.AreEqual( 0x0e5c1a120, crc32.Value );

        crc32 = new CRC32Checksum(1);
        crc32.Update("penguin");
        Assert.AreEqual(0x43b6aa94, crc32.Value);
    }
    #endregion

    #region Adler tests

    [Test]
    public void Adler_Null()
    {
        AdlerChecksum adler = new AdlerChecksum();
        Assert.AreEqual(0, adler.Value);

        adler = new AdlerChecksum(1);
        Assert.AreEqual( 1, adler.Value );

        adler = new AdlerChecksum(556);
        Assert.AreEqual( 556, adler.Value );
    }

    [Test]
    public void Adler_Data()
    {
        AdlerChecksum adler = new AdlerChecksum(1);
        byte[] data = { 1,2,3,4,5,6,7 };
        adler.Update(data);
        Assert.AreEqual( 0x5b001d, adler.Value );

        adler = new AdlerChecksum();
        adler.Update("penguin");
        Assert.AreEqual(0x0bcf02f6, adler.Value );

        adler = new AdlerChecksum(1);
        adler.Update("penguin");
        Assert.AreEqual(0x0bd602f7, adler.Value);
    }
    #endregion
}

```

```

    }
    #endregion
}

[TestFixture]
public class InfoTests
{
    #region Info tests
    [Test]
    public void Info_Version()
    {
        Info info = new Info();
        Assert.AreEqual("1.2.3", Info.Version);
        Assert.AreEqual(32, info.SizeOfUInt);
        Assert.AreEqual(32, info.SizeOfULong);
        Assert.AreEqual(32, info.SizeOfPointer);
        Assert.AreEqual(32, info.SizeOfOffset);
    }
    #endregion
}

[TestFixture]
public class DeflateInflateTests
{
    #region Deflate tests
    [Test]
    public void Deflate_Init()
    {
        using (Deflater def = new Deflater(CompressLevel.Default))
        {
        }
    }

    private ArrayList compressedData = new ArrayList();
    private uint adler1;

    private ArrayList uncompressedData = new ArrayList();
    private uint adler2;

    public void CDataAvail(byte[] data, int startIndex, int count)
    {
        for (int i = 0; i < count; ++i)
            compressedData.Add(data[i+startIndex]);
    }

    [Test]
    public void Deflate_Compress()
    {
        compressedData.Clear();

        byte[] testData = new byte[35000];
        for (int i = 0; i < testData.Length; ++i)
            testData[i] = 5;

        using (Deflater def = new Deflater((CompressLevel)5))
        {
            def.DataAvailable += new DataAvailableHandler(CDataAvail);
            def.Add(testData);
            def.Finish();
            adler1 = def.Checksum;
        }
    }
    #endregion

    #region Inflate tests
    [Test]
    public void Inflate_Init()
    {
        using (Inflater inf = new Inflater())
        {
        }
    }

    private void DDataAvail(byte[] data, int startIndex, int count)

```

```
{
    for (int i = 0; i < count; ++i)
        uncompressedData.Add(data[i+startIndex]);
}

[Test]
public void Inflate_Expand()
{
    uncompressedData.Clear();

    using (Inflater inf = new Inflater())
    {
        inf.DataAvailable += new DataAvailableHandler(DDataAvail);
        inf.Add((byte[])compressedData.ToArray(typeof(byte)));
        inf.Finish();
        adler2 = inf.Checksum;
    }
    Assert.AreEqual( adler1, adler2 );
}
#endregion

[TestFixture]
public class GZipStreamTests
{
    #region GZipStream test
    [Test]
    public void GZipStream_WriteRead()
    {
        using (GZipStream gzOut = new GZipStream("gzstream.gz", CompressLevel.Best))
        {
            BinaryWriter writer = new BinaryWriter(gzOut);
            writer.Write("hi there");
            writer.Write(Math.PI);
            writer.Write(42);
        }

        using (GZipStream gzIn = new GZipStream("gzstream.gz"))
        {
            BinaryReader reader = new BinaryReader(gzIn);
            string s = reader.ReadString();
            Assert.AreEqual("hi there",s);
            double d = reader.ReadDouble();
            Assert.AreEqual(Math.PI, d);
            int i = reader.ReadInt32();
            Assert.AreEqual(42,i);
        }
    }
    #endregion
}

#endif
```


See infback9.h for what this is and how to use it.

```

/* inflate9.c -- inflate deflate64 data using a call-back interface
 * Copyright (C) 1995-2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "infbck9.h"
#include "inftree9.h"
#include "inflate9.h"

#define WSIZE 65536UL

/*
 * strm provides memory allocation functions in zalloc and zfree, or
 * Z_NULL to use the library memory allocation functions.
 *
 * window is a user-supplied window and output buffer that is 64K bytes.
 */
int ZEXPORT inflateBack9Init_(strm, window, version, stream_size)
z_stream FAR *strm;
unsigned char FAR *window;
const char *version;
int stream_size;
{
    struct inflate_state FAR *state;

    if (version == Z_NULL || version[0] != ZLIB_VERSION[0] ||
        stream_size != (int)(sizeof(z_stream)))
        return Z_VERSION_ERROR;
    if (strm == Z_NULL || window == Z_NULL)
        return Z_STREAM_ERROR;
    strm->msg = Z_NULL; /* in case we return an error */
    if (strm->zalloc == (alloc_func)0) {
        strm->zalloc = zcalloc;
        strm->opaque = (voidpf)0;
    }
    if (strm->zfree == (free_func)0) strm->zfree = zcfree;
    state = (struct inflate_state FAR *)ZALLOC(strm, 1,
                                                sizeof(struct inflate_state));
    if (state == Z_NULL) return Z_MEM_ERROR;
    Tracev((stderr, "inflate: allocated\n"));
    strm->state = (voidpf)state;
    state->window = window;
    return Z_OK;
}

/*
 * Build and output length and distance decoding tables for fixed code
 * decoding.
 */
#ifdef MAKEFIXED
#include <stdio.h>

void makefixed9(void)
{
    unsigned sym, bits, low, size;
    code *next, *lenfix, *distfix;
    struct inflate_state state;
    code fixed[544];

    /* literal/length table */
    sym = 0;
    while (sym < 144) state.lens[sym++] = 8;
    while (sym < 256) state.lens[sym++] = 9;
    while (sym < 280) state.lens[sym++] = 7;
    while (sym < 288) state.lens[sym++] = 8;
    next = fixed;
    lenfix = next;
    bits = 9;
    inflate_table9(LENS, state.lens, 288, &(next), &(bits), state.work);

    /* distance table */
    sym = 0;
    while (sym < 32) state.lens[sym++] = 5;

```

```

distfix = next;
bits = 5;
inflate_table9(DISTS, state.lens, 32, &(next), &(bits), state.work);

/* write tables */
puts(" /* inflate9.h --- table for decoding deflate64 fixed codes */");
puts(" * Generated automatically by makefixed9().");
puts(" */");
puts("");
puts(" /* WARNING: this file should *not* be used by applications. */");
puts(" /* It is part of the implementation of this library and is */");
puts(" /* subject to change. Applications should only use zlib.h. */");
puts(" */");
puts("");
size = 1U << 9;
printf(" static const code lenfix[%u] = {", size);
low = 0;
for (;;) {
    if ((low % 6) == 0) printf("\n");
    printf("{%u,%u,%d}", lenfix[low].op, lenfix[low].bits,
        lenfix[low].val);
    if (++low == size) break;
    putchar(',');
}
puts("\n};");
size = 1U << 5;
printf("\n static const code distfix[%u] = {", size);
low = 0;
for (;;) {
    if ((low % 5) == 0) printf("\n");
    printf("{%u,%u,%d}", distfix[low].op, distfix[low].bits,
        distfix[low].val);
    if (++low == size) break;
    putchar(',');
}
puts("\n};");
}
#endif /* MAKEFIXED */

/* Macros for inflateBack(): */

/* Clear the input bit accumulator */
#define INITBITS() \
do { \
    hold = 0; \
    bits = 0; \
} while (0)

/* Assure that some input is available. If input is requested, but denied,
then return a Z_BUF_ERROR from inflateBack(). */
#define PULL() \
do { \
    if (have == 0) { \
        have = in(in_desc, &next); \
        if (have == 0) { \
            next = Z_NULL; \
            ret = Z_BUF_ERROR; \
            goto inf_leave; \
        } \
    } \
} while (0)

/* Get a byte of input into the bit accumulator, or return from inflateBack()
with an error if there is no input available. */
#define PULLBYTE() \
do { \
    PULL(); \
    have--; \
    hold += (unsigned long)(*next++) << bits; \
    bits += 8; \
} while (0)

/* Assure that there are at least n bits in the bit accumulator. If there is
not enough available input to do that, then return from inflateBack() with

```

```

    an error. */
#define NEEDBITS(n) \
    do { \
        while (bits < (unsigned)(n)) \
            PULLBYTE(); \
    } while (0)

/* Return the low n bits of the bit accumulator (n <= 16) */
#define BITS(n) \
    ((unsigned)hold & ((1U << (n)) - 1))

/* Remove n bits from the bit accumulator */
#define DROPBITS(n) \
    do { \
        hold >>= (n); \
        bits -= (unsigned)(n); \
    } while (0)

/* Remove zero to seven bits as needed to go to a byte boundary */
#define BYTEBITS() \
    do { \
        hold >>= bits & 7; \
        bits -= bits & 7; \
    } while (0)

/* Assure that some output space is available, by writing out the window
   if it's full. If the write fails, return from inflateBack() with a
   Z_BUF_ERROR. */
#define ROOM() \
    do { \
        if (left == 0) { \
            put = window; \
            left = WSIZE; \
            wrap = 1; \
            if (out(out_desc, put, (unsigned)left)) { \
                ret = Z_BUF_ERROR; \
                goto inf_leave; \
            } \
        } \
    } while (0)

/*
   strm provides the memory allocation functions and window buffer on input,
   and provides information on the unused input on return. For Z_DATA_ERROR
   returns, strm will also provide an error message.

   in() and out() are the call-back input and output functions. When
   inflateBack() needs more input, it calls in(). When inflateBack() has
   filled the window with output, or when it completes with data in the
   window, it calls out() to write out the data. The application must not
   change the provided input until in() is called again or inflateBack()
   returns. The application must not change the window/output buffer until
   inflateBack() returns.

   in() and out() are called with a descriptor parameter provided in the
   inflateBack() call. This parameter can be a structure that provides the
   information required to do the read or write, as well as accumulated
   information on the input and output such as totals and check values.

   in() should return zero on failure. out() should return non-zero on
   failure. If either in() or out() fails, then inflateBack() returns a
   Z_BUF_ERROR. strm->next_in can be checked for Z_NULL to see whether it
   was in() or out() that caused in the error. Otherwise, inflateBack()
   returns Z_STREAM_END on success, Z_DATA_ERROR for an deflate format
   error, or Z_MEM_ERROR if it could not allocate memory for the state.
   inflateBack() can also return Z_STREAM_ERROR if the input parameters
   are not correct, i.e. strm is Z_NULL or the state was not initialized.
   */
int ZEXPORT inflateBack9(strm, in, in_desc, out, out_desc)
z_stream FAR *strm;
in_func in;
void FAR *in_desc;
out_func out;
void FAR *out_desc;

```

```

{
    struct inflate_state FAR *state;
    unsigned char FAR *next;      /* next input */
    unsigned char FAR *put;       /* next output */
    unsigned have;                /* available input */
    unsigned long left;           /* available output */
    inflate_mode mode;             /* current inflate mode */
    int lastblock;                /* true if processing last block */
    int wrap;                      /* true if the window has wrapped */
    unsigned long write;           /* window write index */
    unsigned char FAR *window;    /* allocated sliding window, if needed */
    unsigned long hold;           /* bit buffer */
    unsigned bits;                /* bits in bit buffer */
    unsigned extra;               /* extra bits needed */
    unsigned long length;         /* literal or length of data to copy */
    unsigned long offset;         /* distance back to copy string from */
    unsigned long copy;           /* number of stored or match bytes to copy */
    unsigned char FAR *from;      /* where to copy match bytes from */
    code const FAR *lencode;      /* starting table for length/literal codes */
    code const FAR *distcode;     /* starting table for distance codes */
    unsigned lenbits;             /* index bits for lencode */
    unsigned distbits;            /* index bits for distcode */
    code this;                    /* current decoding table entry */
    code last;                    /* parent table entry */
    unsigned len;                 /* length to copy for repeats, bits to drop */
    int ret;                      /* return code */
    static const unsigned short order[19] = /* permutation of code lengths */
        {16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15};
#include "infix9.h"

    /* Check that the strm exists and that the state was initialized */
    if (strm == Z_NULL || strm->state == Z_NULL)
        return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;

    /* Reset the state */
    strm->msg = Z_NULL;
    mode = TYPE;
    lastblock = 0;
    write = 0;
    wrap = 0;
    window = state->window;
    next = strm->next_in;
    have = next != Z_NULL ? strm->avail_in : 0;
    hold = 0;
    bits = 0;
    put = window;
    left = WSIZE;
    lencode = Z_NULL;
    distcode = Z_NULL;

    /* Inflate until end of block marked as last */
    for (;;)
        switch (mode) {
            case TYPE:
                /* determine and dispatch block type */
                if (lastblock) {
                    BYTEBITS();
                    mode = DONE;
                    break;
                }
                NEEDBITS(3);
                lastblock = BITS(1);
                DROPBITS(1);
                switch (BITS(2)) {
                    case 0: /* stored block */
                        Tracev((stderr, "inflate: stored block%s\n",
                                lastblock ? "(last)" : ""));
                        mode = STORED;
                        break;
                    case 1: /* fixed block */
                        lencode = lenfix;
                        lenbits = 9;
                        distcode = distfix;

```

```

        distbits = 5;
        Tracev((stderr, "inflate: fixed codes block%s\n",
            lastblock ? "(last)" : ""));
        mode = LEN; /* decode codes */
        break;
    case 2: /* dynamic block */
        Tracev((stderr, "inflate: dynamic codes block%s\n",
            lastblock ? "(last)" : ""));
        mode = TABLE;
        break;
    case 3:
        strm->msg = (char *) "invalid block type";
        mode = BAD;
    }
    DROPBITS(2);
    break;

case STORED:
    /* get and verify stored block length */
    BYTEBITS(); /* go to byte boundary */
    NEEDBITS(32);
    if ((hold & 0xffff) != ((hold >> 16) ^ 0xffff)) {
        strm->msg = (char *) "invalid stored block lengths";
        mode = BAD;
        break;
    }
    length = (unsigned)hold & 0xffff;
    Tracev((stderr, "inflate: stored length %lu\n",
        length));
    INITBITS();

    /* copy stored block from input to output */
    while (length != 0) {
        copy = length;
        PULL();
        ROOM();
        if (copy > have) copy = have;
        if (copy > left) copy = left;
        zmemcpy(put, next, copy);
        have -= copy;
        next += copy;
        left -= copy;
        put += copy;
        length -= copy;
    }
    Tracev((stderr, "inflate: stored end\n"));
    mode = TYPE;
    break;

case TABLE:
    /* get dynamic table entries descriptor */
    NEEDBITS(14);
    state->nlen = BITS(5) + 257;
    DROPBITS(5);
    state->ndist = BITS(5) + 1;
    DROPBITS(5);
    state->ncode = BITS(4) + 4;
    DROPBITS(4);
    if (state->nlen > 286) {
        strm->msg = (char *) "too many length symbols";
        mode = BAD;
        break;
    }
    Tracev((stderr, "inflate: table sizes ok\n"));

    /* get code length code lengths (not a typo) */
    state->have = 0;
    while (state->have < state->ncode) {
        NEEDBITS(3);
        state->lens[order[state->have++]] = (unsigned short)BITS(3);
        DROPBITS(3);
    }
    while (state->have < 19)
        state->lens[order[state->have++]] = 0;

```

```

state->next = state->codes;
lencode = (code const FAR *)(state->next);
lenbits = 7;
ret = inflate_table9(CODES, state->lens, 19, &(state->next),
                    &(lenbits), state->work);
if (ret) {
    strm->msg = (char *) "invalid code lengths set";
    mode = BAD;
    break;
}
Tracev((stderr, "inflate:   code lengths ok\n"));

/* get length and distance code code lengths */
state->have = 0;
while (state->have < state->nlen + state->ndist) {
    for (;;) {
        this = lencode[BITS(lenbits)];
        if ((unsigned)(this.bits) <= bits) break;
        PULLBYTE();
    }
    if (this.val < 16) {
        NEEDBITS(this.bits);
        DROPBITS(this.bits);
        state->lens[state->have++] = this.val;
    }
    else {
        if (this.val == 16) {
            NEEDBITS(this.bits + 2);
            DROPBITS(this.bits);
            if (state->have == 0) {
                strm->msg = (char *) "invalid bit length repeat";
                mode = BAD;
                break;
            }
            len = (unsigned)(state->lens[state->have - 1]);
            copy = 3 + BITS(2);
            DROPBITS(2);
        }
        else if (this.val == 17) {
            NEEDBITS(this.bits + 3);
            DROPBITS(this.bits);
            len = 0;
            copy = 3 + BITS(3);
            DROPBITS(3);
        }
        else {
            NEEDBITS(this.bits + 7);
            DROPBITS(this.bits);
            len = 0;
            copy = 11 + BITS(7);
            DROPBITS(7);
        }
        if (state->have + copy > state->nlen + state->ndist) {
            strm->msg = (char *) "invalid bit length repeat";
            mode = BAD;
            break;
        }
        while (copy--)
            state->lens[state->have++] = (unsigned short)len;
    }
}

/* handle error breaks in while */
if (mode == BAD) break;

/* build code tables */
state->next = state->codes;
lencode = (code const FAR *)(state->next);
lenbits = 9;
ret = inflate_table9(LENS, state->lens, state->nlen,
                    &(state->next), &(lenbits), state->work);
if (ret) {
    strm->msg = (char *) "invalid literal/lengths set";
    mode = BAD;

```

```

        break;
    }
    distcode = (code const FAR *) (state->next);
    distbits = 6;
    ret = inflate_table9(DISTS, state->lens + state->nlen,
                        state->ndist, &(state->next), &(distbits),
                        state->work);

    if (ret) {
        strm->msg = (char *) "invalid distances set";
        mode = BAD;
        break;
    }
    Tracev((stderr, "inflate:   codes ok\n"));
    mode = LEN;

case LEN:
    /* get a literal, length, or end-of-block code */
    for (;;) {
        this = lencode[BITS(lenbits)];
        if ((unsigned)(this.bits) <= bits) break;
        PULLBYTE();
    }
    if (this.op && (this.op & 0xf0) == 0) {
        last = this;
        for (;;) {
            this = lencode[last.val +
                        (BITS(last.bits + last.op) >> last.bits)];
            if ((unsigned)(last.bits + this.bits) <= bits) break;
            PULLBYTE();
        }
        DROPBITS(last.bits);
    }
    DROPBITS(this.bits);
    length = (unsigned)this.val;

    /* process literal */
    if (this.op == 0) {
        Tracevv((stderr, this.val >= 0x20 && this.val < 0x7f ?
                "inflate:   literal '%c'\n" :
                "inflate:   literal 0x%02x\n", this.val));
        ROOM();
        *put++ = (unsigned char)(length);
        left--;
        mode = LEN;
        break;
    }

    /* process end of block */
    if (this.op & 32) {
        Tracevv((stderr, "inflate:   end of block\n"));
        mode = TYPE;
        break;
    }

    /* invalid code */
    if (this.op & 64) {
        strm->msg = (char *) "invalid literal/length code";
        mode = BAD;
        break;
    }

    /* length code -- get extra bits, if any */
    extra = (unsigned)(this.op) & 31;
    if (extra != 0) {
        NEEDBITS(extra);
        length += BITS(extra);
        DROPBITS(extra);
    }
    Tracevv((stderr, "inflate:   length %lu\n", length));

    /* get distance code */
    for (;;) {
        this = distcode[BITS(distbits)];
        if ((unsigned)(this.bits) <= bits) break;

```



```

        PULLBYTE();
    }
    if ((this.op & 0xf0) == 0) {
        last = this;
        for (;;) {
            this = distcode[last.val +
                (BITS(last.bits + last.op) >> last.bits)];
            if ((unsigned)(last.bits + this.bits) <= bits) break;
            PULLBYTE();
        }
        DROPBITS(last.bits);
    }
    DROPBITS(this.bits);
    if (this.op & 64) {
        strm->msg = (char *) "invalid distance code";
        mode = BAD;
        break;
    }
    offset = (unsigned)this.val;

    /* get distance extra bits, if any */
    extra = (unsigned)(this.op) & 15;
    if (extra != 0) {
        NEEDBITS(extra);
        offset += BITS(extra);
        DROPBITS(extra);
    }
    if (offset > WSIZE - (wrap ? 0 : left)) {
        strm->msg = (char *) "invalid distance too far back";
        mode = BAD;
        break;
    }
    Tracevv((stderr, "inflate: distance %lu\n", offset));

    /* copy match from window to output */
    do {
        ROOM();
        copy = WSIZE - offset;
        if (copy < left) {
            from = put + copy;
            copy = left - copy;
        }
        else {
            from = put - offset;
            copy = left;
        }
        if (copy > length) copy = length;
        length -= copy;
        left -= copy;
        do {
            *put++ = *from++;
        } while (--copy);
    } while (length != 0);
    break;

case DONE:
    /* inflate stream terminated properly -- write leftover output */
    ret = Z_STREAM_END;
    if (left < WSIZE) {
        if (out(out_desc, window, (unsigned)(WSIZE - left)))
            ret = Z_BUF_ERROR;
    }
    goto inf_leave;

case BAD:
    ret = Z_DATA_ERROR;
    goto inf_leave;

default:
    /* can't happen, but makes compilers happy */
    ret = Z_STREAM_ERROR;
    goto inf_leave;
}

/* Return unused input */

```

```
inf_leave:
    strm->next_in = next;
    strm->avail_in = have;
    return ret;
}

int ZEXPORT inflateBack9End(strm)
z_stream FAR *strm;
{
    if (strm == Z_NULL || strm->state == Z_NULL || strm->zfree == (free_func)0)
        return Z_STREAM_ERROR;
    ZFREE(strm, strm->state);
    strm->state = Z_NULL;
    Tracev((stderr, "inflate: end\n"));
    return Z_OK;
}
```

```
/* inflateBack9.h -- header for using inflateBack9 functions
 * Copyright (C) 2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 * This header file and associated patches provide a decoder for PKWare's
 * undocumented deflate64 compression method (method 9). Use with inflateBack9.c,
 * inftree9.h, inftree9.c, and inffix9.h. These patches are not supported.
 * This should be compiled with zlib, since it uses zutil.h and zutil.o.
 * This code has not yet been tested on 16-bit architectures. See the
 * comments in zlib.h for inflateBack() usage. These functions are used
 * identically, except that there is no windowBits parameter, and a 64K
 * window must be provided. Also if int's are 16 bits, then a zero for
 * the third parameter of the "out" function actually means 65536UL.
 * zlib.h must be included before this header file.
 */

#ifdef __cplusplus
extern "C" {
#endif

ZEXTERN int ZEXPORT inflateBack9 OF((z_stream FAR *strm,
                                     in_func in, void FAR *in_desc,
                                     out_func out, void FAR *out_desc));
ZEXTERN int ZEXPORT inflateBack9End OF((z_stream FAR *strm));
ZEXTERN int ZEXPORT inflateBack9Init_ OF((z_stream FAR *strm,
                                          unsigned char FAR *window,
                                          const char *version,
                                          int stream_size));

#define inflateBack9Init(strm, window) \
    inflateBack9Init_((strm), (window), \
        ZLIB_VERSION, sizeof(z_stream))

#ifdef __cplusplus
}
#endif
#endif
```

```
/* inffix9.h -- table for decoding deflate64 fixed codes
 * Generated automatically by makefixed9().
 */
```

```
/* WARNING: this file should *not* be used by applications.
 * It is part of the implementation of this library and is
 * subject to change. Applications should only use zlib.h.
 */
```

```
static const code lenfix[512] = {
    {96,7,0},{0,8,80},{0,8,16},{132,8,115},{130,7,31},{0,8,112},
    {0,8,48},{0,9,192},{128,7,10},{0,8,96},{0,8,32},{0,9,160},
    {0,8,0},{0,8,128},{0,8,64},{0,9,224},{128,7,6},{0,8,88},
    {0,8,24},{0,9,144},{131,7,59},{0,8,120},{0,8,56},{0,9,208},
    {129,7,17},{0,8,104},{0,8,40},{0,9,176},{0,8,8},{0,8,136},
    {0,8,72},{0,9,240},{128,7,4},{0,8,84},{0,8,20},{133,8,227},
    {131,7,43},{0,8,116},{0,8,52},{0,9,200},{129,7,13},{0,8,100},
    {0,8,36},{0,9,168},{0,8,4},{0,8,132},{0,8,68},{0,9,232},
    {128,7,8},{0,8,92},{0,8,28},{0,9,152},{132,7,83},{0,8,124},
    {0,8,60},{0,9,216},{130,7,23},{0,8,108},{0,8,44},{0,9,184},
    {0,8,12},{0,8,140},{0,8,76},{0,9,248},{128,7,3},{0,8,82},
    {0,8,18},{133,8,163},{131,7,35},{0,8,114},{0,8,50},{0,9,196},
    {129,7,11},{0,8,98},{0,8,34},{0,9,164},{0,8,2},{0,8,130},
    {0,8,66},{0,9,228},{128,7,7},{0,8,90},{0,8,26},{0,9,148},
    {132,7,67},{0,8,122},{0,8,58},{0,9,212},{130,7,19},{0,8,106},
    {0,8,42},{0,9,180},{0,8,10},{0,8,138},{0,8,74},{0,9,244},
    {128,7,5},{0,8,86},{0,8,22},{65,8,0},{131,7,51},{0,8,118},
    {0,8,54},{0,9,204},{129,7,15},{0,8,102},{0,8,38},{0,9,172},
    {0,8,6},{0,8,134},{0,8,70},{0,9,236},{128,7,9},{0,8,94},
    {0,8,30},{0,9,156},{132,7,99},{0,8,126},{0,8,62},{0,9,220},
    {130,7,27},{0,8,110},{0,8,46},{0,9,188},{0,8,14},{0,8,142},
    {0,8,78},{0,9,252},{96,7,0},{0,8,81},{0,8,17},{133,8,131},
    {130,7,31},{0,8,113},{0,8,49},{0,9,194},{128,7,10},{0,8,97},
    {0,8,33},{0,9,162},{0,8,1},{0,8,129},{0,8,65},{0,9,226},
    {128,7,6},{0,8,89},{0,8,25},{0,9,146},{131,7,59},{0,8,121},
    {0,8,57},{0,9,210},{129,7,17},{0,8,105},{0,8,41},{0,9,178},
    {0,8,9},{0,8,137},{0,8,73},{0,9,242},{128,7,4},{0,8,85},
    {0,8,21},{144,8,3},{131,7,43},{0,8,117},{0,8,53},{0,9,202},
    {129,7,13},{0,8,101},{0,8,37},{0,9,170},{0,8,5},{0,8,133},
    {0,8,69},{0,9,234},{128,7,8},{0,8,93},{0,8,29},{0,9,154},
    {132,7,83},{0,8,125},{0,8,61},{0,9,218},{130,7,23},{0,8,109},
    {0,8,45},{0,9,186},{0,8,13},{0,8,141},{0,8,77},{0,9,250},
    {128,7,3},{0,8,83},{0,8,19},{133,8,195},{131,7,35},{0,8,115},
    {0,8,51},{0,9,198},{129,7,11},{0,8,99},{0,8,35},{0,9,166},
    {0,8,3},{0,8,131},{0,8,67},{0,9,230},{128,7,7},{0,8,91},
    {0,8,27},{0,9,150},{132,7,67},{0,8,123},{0,8,59},{0,9,214},
    {130,7,19},{0,8,107},{0,8,43},{0,9,182},{0,8,11},{0,8,139},
    {0,8,75},{0,9,246},{128,7,5},{0,8,87},{0,8,23},{77,8,0},
    {131,7,51},{0,8,119},{0,8,55},{0,9,206},{129,7,15},{0,8,103},
    {0,8,39},{0,9,174},{0,8,7},{0,8,135},{0,8,71},{0,9,238},
    {128,7,9},{0,8,95},{0,8,31},{0,9,158},{132,7,99},{0,8,127},
    {0,8,63},{0,9,222},{130,7,27},{0,8,111},{0,8,47},{0,9,190},
    {0,8,15},{0,8,143},{0,8,79},{0,9,254},{96,7,0},{0,8,80},
    {0,8,16},{132,8,115},{130,7,31},{0,8,112},{0,8,48},{0,9,193},
    {128,7,10},{0,8,96},{0,8,32},{0,9,161},{0,8,0},{0,8,128},
    {0,8,64},{0,9,225},{128,7,6},{0,8,88},{0,8,24},{0,9,145},
    {131,7,59},{0,8,120},{0,8,56},{0,9,209},{129,7,17},{0,8,104},
    {0,8,40},{0,9,177},{0,8,8},{0,8,136},{0,8,72},{0,9,241},
    {128,7,4},{0,8,84},{0,8,20},{133,8,227},{131,7,43},{0,8,116},
    {0,8,52},{0,9,201},{129,7,13},{0,8,100},{0,8,36},{0,9,169},
    {0,8,4},{0,8,132},{0,8,68},{0,9,233},{128,7,8},{0,8,92},
    {0,8,28},{0,9,153},{132,7,83},{0,8,124},{0,8,60},{0,9,217},
    {130,7,23},{0,8,108},{0,8,44},{0,9,185},{0,8,12},{0,8,140},
    {0,8,76},{0,9,249},{128,7,3},{0,8,82},{0,8,18},{133,8,163},
    {131,7,35},{0,8,114},{0,8,50},{0,9,197},{129,7,11},{0,8,98},
    {0,8,34},{0,9,165},{0,8,2},{0,8,130},{0,8,66},{0,9,229},
    {128,7,7},{0,8,90},{0,8,26},{0,9,149},{132,7,67},{0,8,122},
    {0,8,58},{0,9,213},{130,7,19},{0,8,106},{0,8,42},{0,9,181},
    {0,8,10},{0,8,138},{0,8,74},{0,9,245},{128,7,5},{0,8,86},
    {0,8,22},{65,8,0},{131,7,51},{0,8,118},{0,8,54},{0,9,205},
    {129,7,15},{0,8,102},{0,8,38},{0,9,173},{0,8,6},{0,8,134},
    {0,8,70},{0,9,237},{128,7,9},{0,8,94},{0,8,30},{0,9,157},
    {132,7,99},{0,8,126},{0,8,62},{0,9,221},{130,7,27},{0,8,110},
    {0,8,46},{0,9,189},{0,8,14},{0,8,142},{0,8,78},{0,9,253},

```

```
{96,7,0},{0,8,81},{0,8,17},{133,8,131},{130,7,31},{0,8,113},
{0,8,49},{0,9,195},{128,7,10},{0,8,97},{0,8,33},{0,9,163},
{0,8,1},{0,8,129},{0,8,65},{0,9,227},{128,7,6},{0,8,89},
{0,8,25},{0,9,147},{131,7,59},{0,8,121},{0,8,57},{0,9,211},
{129,7,17},{0,8,105},{0,8,41},{0,9,179},{0,8,9},{0,8,137},
{0,8,73},{0,9,243},{128,7,4},{0,8,85},{0,8,21},{144,8,3},
{131,7,43},{0,8,117},{0,8,53},{0,9,203},{129,7,13},{0,8,101},
{0,8,37},{0,9,171},{0,8,5},{0,8,133},{0,8,69},{0,9,235},
{128,7,8},{0,8,93},{0,8,29},{0,9,155},{132,7,83},{0,8,125},
{0,8,61},{0,9,219},{130,7,23},{0,8,109},{0,8,45},{0,9,187},
{0,8,13},{0,8,141},{0,8,77},{0,9,251},{128,7,3},{0,8,83},
{0,8,19},{133,8,195},{131,7,35},{0,8,115},{0,8,51},{0,9,199},
{129,7,11},{0,8,99},{0,8,35},{0,9,167},{0,8,3},{0,8,131},
{0,8,67},{0,9,231},{128,7,7},{0,8,91},{0,8,27},{0,9,151},
{132,7,67},{0,8,123},{0,8,59},{0,9,215},{130,7,19},{0,8,107},
{0,8,43},{0,9,183},{0,8,11},{0,8,139},{0,8,75},{0,9,247},
{128,7,5},{0,8,87},{0,8,23},{77,8,0},{131,7,51},{0,8,119},
{0,8,55},{0,9,207},{129,7,15},{0,8,103},{0,8,39},{0,9,175},
{0,8,7},{0,8,135},{0,8,71},{0,9,239},{128,7,9},{0,8,95},
{0,8,31},{0,9,159},{132,7,99},{0,8,127},{0,8,63},{0,9,223},
{130,7,27},{0,8,111},{0,8,47},{0,9,191},{0,8,15},{0,8,143},
{0,8,79},{0,9,255}
};

static const code distfix[32] = {
{128,5,1},{135,5,257},{131,5,17},{139,5,4097},{129,5,5},
{137,5,1025},{133,5,65},{141,5,16385},{128,5,3},{136,5,513},
{132,5,33},{140,5,8193},{130,5,9},{138,5,2049},{134,5,129},
{142,5,32769},{128,5,2},{135,5,385},{131,5,25},{139,5,6145},
{129,5,7},{137,5,1537},{133,5,97},{141,5,24577},{128,5,4},
{136,5,769},{132,5,49},{140,5,12289},{130,5,13},{138,5,3073},
{134,5,193},{142,5,49153}
};
```

```
/* inflate9.h -- internal inflate state definition
 * Copyright (C) 1995-2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
 * part of the implementation of the compression library and is
 * subject to change. Applications should only use zlib.h.
 */

/* Possible inflate modes between inflate() calls */
typedef enum {
    TYPE,          /* i: waiting for type bits, including last-flag bit */
    STORED,        /* i: waiting for stored size (length and complement) */
    TABLE,        /* i: waiting for dynamic block table lengths */
    LEN,           /* i: waiting for length/lit code */
    DONE,          /* finished check, done -- remain here until reset */
    BAD            /* got a data error -- remain here until reset */
} inflate_mode;

/*
 * State transitions between above modes -
 *
 * (most modes can go to the BAD mode -- not shown for clarity)
 *
 * Read deflate blocks:
 *     TYPE -> STORED or TABLE or LEN or DONE
 *     STORED -> TYPE
 *     TABLE -> LENLENS -> CODELENS -> LEN
 *
 * Read deflate codes:
 *     LEN -> LEN or TYPE
 */

/* state maintained between inflate() calls.  Approximately 7K bytes. */
struct inflate_state {
    /* sliding window */
    unsigned char FAR *window; /* allocated sliding window, if needed */
    /* dynamic table building */
    unsigned ncode; /* number of code length code lengths */
    unsigned nlen; /* number of length code lengths */
    unsigned ndist; /* number of distance code lengths */
    unsigned have; /* number of code lengths in lens[] */
    code FAR *next; /* next available space in codes[] */
    unsigned short lens[320]; /* temporary storage for code lengths */
    unsigned short work[288]; /* work area for code table building */
    code codes[ENOUGH]; /* space for code tables */
};
```

```

/* inftree9.c -- generate Huffman trees for efficient decoding
 * Copyright (C) 1995-2005 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "inftree9.h"

#define MAXBITS 15

const char inflate9_copyright[] =
    " inflate9 1.2.3 Copyright 1995-2005 Mark Adler ";
/*
 * If you use the zlib library in a product, an acknowledgment is welcome
 * in the documentation of your product. If for some reason you cannot
 * include such an acknowledgment, I would appreciate that you keep this
 * copyright string in the executable of your product.
 */

/*
 * Build a set of tables to decode the provided canonical Huffman code.
 * The code lengths are lens[0..codes-1]. The result starts at *table,
 * whose indices are 0..2^bits-1. work is a writable array of at least
 * lens shorts, which is used as a work area. type is the type of code
 * to be generated, CODES, LENS, or DISTS. On return, zero is success,
 * -1 is an invalid code, and +1 means that ENOUGH isn't enough. table
 * on return points to the next available entry's address. bits is the
 * requested root table index bits, and on return it is the actual root
 * table index bits. It will differ if the request is greater than the
 * longest code or if it is less than the shortest code.
 */
int inflate_table9(type, lens, codes, table, bits, work)
codetype type;
unsigned short FAR *lens;
unsigned codes;
code FAR * FAR *table;
unsigned FAR *bits;
unsigned short FAR *work;
{
    unsigned len;                /* a code's length in bits */
    unsigned sym;                /* index of code symbols */
    unsigned min, max;           /* minimum and maximum code lengths */
    unsigned root;               /* number of index bits for root table */
    unsigned curr;               /* number of index bits for current table */
    unsigned drop;               /* code bits to drop for sub-table */
    int left;                    /* number of prefix codes available */
    unsigned used;               /* code entries in table used */
    unsigned huff;               /* Huffman code */
    unsigned incr;               /* for incrementing code, index */
    unsigned fill;               /* index for replicating entries */
    unsigned low;                /* low bits for current root entry */
    unsigned mask;               /* mask for low root bits */
    code this;                   /* table entry for duplication */
    code FAR *next;              /* next available space in table */
    const unsigned short FAR *base; /* base value table to use */
    const unsigned short FAR *extra; /* extra bits table to use */
    int end;                     /* use base and extra for symbol > end */
    unsigned short count[MAXBITS+1]; /* number of codes of each length */
    unsigned short offs[MAXBITS+1]; /* offsets in table for each length */
    static const unsigned short lbase[31] = { /* Length codes 257..285 base */
        3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17,
        19, 23, 27, 31, 35, 43, 51, 59, 67, 83, 99, 115,
        131, 163, 195, 227, 3, 0, 0};
    static const unsigned short lext[31] = { /* Length codes 257..285 extra */
        128, 128, 128, 128, 128, 128, 128, 128, 129, 129, 129, 129,
        130, 130, 130, 130, 131, 131, 131, 131, 132, 132, 132, 132,
        133, 133, 133, 133, 144, 201, 196};
    static const unsigned short dbase[32] = { /* Distance codes 0..31 base */
        1, 2, 3, 4, 5, 7, 9, 13, 17, 25, 33, 49,
        65, 97, 129, 193, 257, 385, 513, 769, 1025, 1537, 2049, 3073,
        4097, 6145, 8193, 12289, 16385, 24577, 32769, 49153};
    static const unsigned short dext[32] = { /* Distance codes 0..31 extra */
        128, 128, 128, 128, 129, 129, 130, 130, 131, 131, 132, 132,
        133, 133, 134, 134, 135, 135, 136, 136, 137, 137, 138, 138,

```

```
139, 139, 140, 140, 141, 141, 142, 142};
```

```
/*
Process a set of code lengths to create a canonical Huffman code. The
code lengths are lens[0..codes-1]. Each length corresponds to the
symbols 0..codes-1. The Huffman code is generated by first sorting the
symbols by length from short to long, and retaining the symbol order
for codes with equal lengths. Then the code starts with all zero bits
for the first code of the shortest length, and the codes are integer
increments for the same length, and zeros are appended as the length
increases. For the deflate format, these bits are stored backwards
from their more natural integer increment ordering, and so when the
decoding tables are built in the large loop below, the integer codes
are incremented backwards.
```

```
This routine assumes, but does not check, that all of the entries in
lens[] are in the range 0..MAXBITS. The caller must assure this.
1..MAXBITS is interpreted as that code length. zero means that that
symbol does not occur in this code.
```

```
The codes are sorted by computing a count of codes for each length,
creating from that a table of starting indices for each length in the
sorted table, and then entering the symbols in order in the sorted
table. The sorted table is work[], with that space being provided by
the caller.
```

```
The length counts are used for other purposes as well, i.e. finding
the minimum and maximum length codes, determining if there are any
codes at all, checking for a valid set of lengths, and looking ahead
at length counts to determine sub-table sizes when building the
decoding tables.
```

```
*/
```

```
/* accumulate lengths for codes (assumes lens[] all in 0..MAXBITS) */
```

```
for (len = 0; len <= MAXBITS; len++)
    count[len] = 0;
for (sym = 0; sym < codes; sym++)
    count[lens[sym]]++;
```

```
/* bound code lengths, force root to be within code lengths */
```

```
root = *bits;
for (max = MAXBITS; max >= 1; max--)
    if (count[max] != 0) break;
if (root > max) root = max;
if (max == 0) return -1; /* no codes! */
for (min = 1; min <= MAXBITS; min++)
    if (count[min] != 0) break;
if (root < min) root = min;
```

```
/* check for an over-subscribed or incomplete set of lengths */
```

```
left = 1;
for (len = 1; len <= MAXBITS; len++) {
    left <= 1;
    left -= count[len];
    if (left < 0) return -1; /* over-subscribed */
}
if (left > 0 && (type == CODES || max != 1))
    return -1; /* incomplete set */
```

```
/* generate offsets into symbol table for each length for sorting */
```

```
offs[1] = 0;
for (len = 1; len < MAXBITS; len++)
    offs[len + 1] = offs[len] + count[len];
```

```
/* sort symbols by length, by symbol order within each length */
```

```
for (sym = 0; sym < codes; sym++)
    if (lens[sym] != 0) work[offs[lens[sym]]++] = (unsigned short)sym;
```

```
/*
```

```
Create and fill in decoding tables. In this loop, the table being
filled is at next and has curr index bits. The code being used is huff
with length len. That code is converted to an index by dropping drop
bits off of the bottom. For codes where len is less than drop + curr,
those top drop + curr - len bits are incremented through all values to
```


fill the table with replicated entries.

root is the number of index bits for the root table. When len exceeds root, sub-tables are created pointed to by the root entry with an index of the low root bits of huff. This is saved in low to check for when a new sub-table should be started. drop is zero when the root table is being filled, and drop is root when sub-tables are being filled.

When a new sub-table is needed, it is necessary to look ahead in the code lengths to determine what size sub-table is needed. The length counts are used for this, and so count[] is decremented as codes are entered in the tables.

used keeps track of how many table entries have been allocated from the provided *table space. It is checked when a LENS table is being made against the space in *table, ENOUGH, minus the maximum space needed by the worst case distance code, MAXD. This should never happen, but the sufficiency of ENOUGH has not been proven exhaustively, hence the check. This assumes that when type == LENS, bits == 9.

sym increments through all symbols, and the loop terminates when all codes of length max, i.e. all codes, have been processed. This routine permits incomplete codes, so another loop after this one fills in the rest of the decoding tables with invalid code markers.

*/

/* set up for code type */

```
switch (type) {
case CODES:
    base = extra = work;    /* dummy value--not used */
    end = 19;
    break;
case LENS:
    base = lbase;
    base -= 257;
    extra = lext;
    extra -= 257;
    end = 256;
    break;
default:                /* DISTS */
    base = dbase;
    extra = dext;
    end = -1;
}
```

/* initialize state for loop */

```
huff = 0;                /* starting code */
sym = 0;                 /* starting code symbol */
len = min;               /* starting code length */
next = *table;           /* current table to fill in */
curr = root;             /* current table index bits */
drop = 0;                /* current bits to drop from code for index */
low = (unsigned)(-1);     /* trigger new sub-table when len > root */
used = 1U << root;        /* use root table entries */
mask = used - 1;         /* mask for comparing low */
```

/* check available table space */

```
if (type == LENS && used >= ENOUGH - MAXD)
    return 1;
```

/* process all codes and make table entries */

```
for (;;) {
    /* create table entry */
    this.bits = (unsigned char)(len - drop);
    if ((int)(work[sym]) < end) {
        this.op = (unsigned char)0;
        this.val = work[sym];
    }
    else if ((int)(work[sym]) > end) {
        this.op = (unsigned char)(extra[work[sym]]);
        this.val = base[work[sym]];
    }
    else {
        this.op = (unsigned char)(32 + 64);    /* end of block */
    }
}
```

```

        this.val = 0;
    }

    /* replicate for those indices with low len bits equal to huff */
    incr = 1U << (len - drop);
    fill = 1U << curr;
    do {
        fill -= incr;
        next[(huff >> drop) + fill] = this;
    } while (fill != 0);

    /* backwards increment the len-bit code huff */
    incr = 1U << (len - 1);
    while (huff & incr)
        incr >>= 1;
    if (incr != 0) {
        huff &= incr - 1;
        huff += incr;
    }
    else
        huff = 0;

    /* go to next symbol, update count, len */
    sym++;
    if (--(count[len]) == 0) {
        if (len == max) break;
        len = lens[work[sym]];
    }

    /* create new sub-table if needed */
    if (len > root && (huff & mask) != low) {
        /* if first time, transition to sub-tables */
        if (drop == 0)
            drop = root;

        /* increment past last table */
        next += 1U << curr;

        /* determine length of next table */
        curr = len - drop;
        left = (int)(1 << curr);
        while (curr + drop < max) {
            left -= count[curr + drop];
            if (left <= 0) break;
            curr++;
            left <=& 1;
        }

        /* check for enough space */
        used += 1U << curr;
        if (type == LENS && used >= ENOUGH - MAXD)
            return 1;

        /* point entry in root table to sub-table */
        low = huff & mask;
        (*table)[low].op = (unsigned char)curr;
        (*table)[low].bits = (unsigned char)root;
        (*table)[low].val = (unsigned short)(next - *table);
    }
}

/*
Fill in rest of table for incomplete codes. This loop is similar to the
loop above in incrementing huff for table indices. It is assumed that
len is equal to curr + drop, so there is no loop needed to increment
through high index bits. When the current sub-table is filled, the loop
drops back to the root table to fill in any remaining entries there.
*/
this.op = (unsigned char)64; /* invalid code marker */
this.bits = (unsigned char)(len - drop);
this.val = (unsigned short)0;
while (huff != 0) {
    /* when done with sub-table, drop back to root table */
    if (drop != 0 && (huff & mask) != low) {

```

```
        drop = 0;
        len = root;
        next = *table;
        curr = root;
        this.bits = (unsigned char)len;
    }

    /* put invalid code marker in table */
    next[huff >> drop] = this;

    /* backwards increment the len-bit code huff */
    incr = 1U << (len - 1);
    while (huff & incr)
        incr >>= 1;
    if (incr != 0) {
        huff &= incr - 1;
        huff += incr;
    }
    else
        huff = 0;
}

/* set return parameters */
*table += used;
*bits = root;
return 0;
}
```

```
/* inftree9.h -- header to use inftree9.c
 * Copyright (C) 1995-2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
 * part of the implementation of the compression library and is
 * subject to change. Applications should only use zlib.h.
 */

/* Structure for decoding tables. Each entry provides either the
 * information needed to do the operation requested by the code that
 * indexed that table entry, or it provides a pointer to another
 * table that indexes more bits of the code. op indicates whether
 * the entry is a pointer to another table, a literal, a length or
 * distance, an end-of-block, or an invalid code. For a table
 * pointer, the low four bits of op is the number of index bits of
 * that table. For a length or distance, the low four bits of op
 * is the number of extra bits to get after the code. bits is
 * the number of bits in this code or part of the code to drop off
 * of the bit buffer. val is the actual byte to output in the case
 * of a literal, the base length or distance, or the offset from
 * the current table to the next table. Each entry is four bytes. */
typedef struct {
    unsigned char op;          /* operation, extra bits, table bits */
    unsigned char bits;       /* bits in this part of the code */
    unsigned short val;        /* offset in table or code value */
} code;

/* op values as set by inflate_table():
   00000000 - literal
   0000tttt - table link, tttt != 0 is the number of table index bits
   100eeeeee - length or distance, eeee is the number of extra bits
   01100000 - end of block
   01000000 - invalid code
 */

/* Maximum size of dynamic tree. The maximum found in a long but non-
 * exhaustive search was 1444 code structures (852 for length/literals
 * and 592 for distances, the latter actually the result of an
 * exhaustive search). The true maximum is not known, but the value
 * below is more than safe. */
#define ENOUGH 2048
#define MAXD 592

/* Type of code to build for inftable() */
typedef enum {
    CODES,
    LENS,
    DISTS
} codetype;

extern int inflate_table9 OF((codetype type, unsigned short FAR *lens,
                             unsigned codes, code FAR * FAR *table,
                             unsigned FAR *bits, unsigned short FAR *work));
```

```

/* inffas86.c is a hand tuned assembler version of
*
* inffast.c -- fast decoding
* Copyright (C) 1995-2003 Mark Adler
* For conditions of distribution and use, see copyright notice in zlib.h
*
* Copyright (C) 2003 Chris Anderson <christop@charm.net>
* Please use the copyright conditions above.
*
* Dec-29-2003 -- I added AMD64 inflate asm support. This version is also
* slightly quicker on x86 systems because, instead of using rep movsb to copy
* data, it uses rep movsw, which moves data in 2-byte chunks instead of single
* bytes. I've tested the AMD64 code on a Fedora Core 1 + the x86_64 updates
* from http://fedora.linux.duke.edu/fc1_x86_64
* which is running on an Athlon 64 3000+ / Gigabyte GA-K8VT800M system with
* 1GB ram. The 64-bit version is about 4% faster than the 32-bit version,
* when decompressing mozilla-source-1.3.tar.gz.
*
* Mar-13-2003 -- Most of this is derived from inffast.S which is derived from
* the gcc -S output of zlib-1.2.0/inffast.c. Zlib-1.2.0 is in beta release at
* the moment. I have successfully compiled and tested this code with gcc2.96,
* gcc3.2, icc5.0, msvc6.0. It is very close to the speed of inffast.S
* compiled with gcc -DNO_MMX, but inffast.S is still faster on the P3 with MMX
* enabled. I will attempt to merge the MMX code into this version. Newer
* versions of this and inffast.S can be found at
* http://www.eetbeetee.com/zlib/ and http://www.charm.net/~christop/zlib/
*/

#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"

/* Mark Adler's comments from inffast.c: */

/*
Decode literal, length, and distance codes and write out the resulting
literal and match bytes until either not enough input or output is
available, an end-of-block is encountered, or a data error is encountered.
When large enough input and output buffers are supplied to inflate(), for
example, a 16K input buffer and a 64K output buffer, more than 95% of the
inflate execution time is spent in this routine.

Entry assumptions:
    state->mode == LEN
    strm->avail_in >= 6
    strm->avail_out >= 258
    start >= strm->avail_out
    state->bits < 8

On return, state->mode is one of:
    LEN -- ran out of enough output space or enough available input
    TYPE -- reached end of block code, inflate() to interpret next block
    BAD -- error in block data

Notes:
    - The maximum input bits used by a length/distance pair is 15 bits for the
      length code, 5 bits for the length extra, 15 bits for the distance code,
      and 13 bits for the distance extra. This totals 48 bits, or six bytes.
      Therefore if strm->avail_in >= 6, then there is enough input to avoid
      checking for available input while decoding.
    - The maximum bytes that a single length/distance pair can output is 258
      bytes, which is the maximum length that can be coded. inflate_fast()
      requires strm->avail_out >= 258 for each loop to avoid checking for
      output space.
*/
void inflate_fast(strm, start)
z_streamp strm;
unsigned start;          /* inflate()'s starting value for strm->avail_out */
{

```

```

    struct inflate_state FAR *state;
    struct inffast_ar {
/* 64      32                                x86  x86_64 */
/* ar offset                                register */
/* 0       0 */ void *esp;                  /* esp save */
/* 8       4 */ void *ebp;                  /* ebp save */
/* 16      8 */ unsigned char FAR *in;      /* esi rsi  local strm->next_in */
/* 24     12 */ unsigned char FAR *last;    /* r9       while in < last */
/* 32     16 */ unsigned char FAR *out;     /* edi rdi  local strm->next_out */
/* 40     20 */ unsigned char FAR *beg;     /*         inflate()'s init next_out */
/* 48     24 */ unsigned char FAR *end;     /* r10      while out < end */
/* 56     28 */ unsigned char FAR *window; /*         size of window, wsize!=0 */
/* 64     32 */ code const FAR *lcode;     /* ebp rbp  local strm->lencode */
/* 72     36 */ code const FAR *dcode;     /* r11      local strm->distcode */
/* 80     40 */ unsigned long hold;        /* edx rdx  local strm->hold */
/* 88     44 */ unsigned bits;             /* ebx rbx  local strm->bits */
/* 92     48 */ unsigned wsize;            /*         window size */
/* 96     52 */ unsigned write;            /*         window write index */
/*100     56 */ unsigned lmask;            /* r12      mask for lcode */
/*104     60 */ unsigned dmask;            /* r13      mask for dcode */
/*108     64 */ unsigned len;              /* r14      match length */
/*112     68 */ unsigned dist;             /* r15      match distance */
/*116     72 */ unsigned status;          /*         set when state chng*/
    } ar;

```

```

#if defined( __GNUC__ ) && defined( __amd64__ ) && ! defined( __i386__ )

```

```

#define PAD_AVAIL_IN 6

```

```

#define PAD_AVAIL_OUT 258

```

```

#else

```

```

#define PAD_AVAIL_IN 5

```

```

#define PAD_AVAIL_OUT 257

```

```

#endif

```

```

/* copy state to local variables */

```

```

state = (struct inflate_state FAR *)strm->state;

```

```

ar.in = strm->next_in;

```

```

ar.last = ar.in + (strm->avail_in - PAD_AVAIL_IN);

```

```

ar.out = strm->next_out;

```

```

ar.beg = ar.out - (start - strm->avail_out);

```

```

ar.end = ar.out + (strm->avail_out - PAD_AVAIL_OUT);

```

```

ar.wsize = state->wsize;

```

```

ar.write = state->write;

```

```

ar.window = state->window;

```

```

ar.hold = state->hold;

```

```

ar.bits = state->bits;

```

```

ar.lcode = state->lencode;

```

```

ar.dcode = state->distcode;

```

```

ar.lmask = (1U << state->lenbits) - 1;

```

```

ar.dmask = (1U << state->distbits) - 1;

```

```

/* decode literals and length/distances until end-of-block or not enough
   input data or output space */

```

```

/* align in on 1/2 hold size boundary */

```

```

while (((unsigned long)(void *)ar.in & (sizeof(ar.hold) / 2 - 1)) != 0) {
    ar.hold += (unsigned long)*ar.in++ << ar.bits;
    ar.bits += 8;
}

```

```

#if defined( __GNUC__ ) && defined( __amd64__ ) && ! defined( __i386__ )

```

```

    __asm__ __volatile__ (

```

```

"    leaq  %0, %%rax\n"

```

```

"    movq  %%rbp, 8(%%rax)\n"          /* save regs rbp and rsp */

```

```

"    movq  %%rsp, (%%rax)\n"

```

```

"    movq  %%rax, %%rsp\n"            /* make rsp point to &ar */

```

```

"    movq  16(%%rsp), %%rsi\n"        /* rsi = in */

```

```

"    movq  32(%%rsp), %%rdi\n"        /* rdi = out */

```

```

"    movq  24(%%rsp), %%r9\n"         /* r9 = last */

```

```

"    movq  48(%%rsp), %%r10\n"        /* r10 = end */

```

```

"    movq  64(%%rsp), %%rbp\n"        /* rbp = lcode */

```

```

"    movq  72(%%rsp), %%r11\n"        /* r11 = dcode */

```

```

"    movq  80(%%rsp), %%rdx\n"        /* rdx = hold */

```

```

"    movl  88(%%rsp), %%ebx\n"        /* ebx = bits */

```

```

"    movl  100(%%rsp), %%r12d\n"      /* r12d = lmask */

```

```
"    movl    104(%%rsp), %%r13d\n"                /* r13d = dmask */\n"                                /* r14d = len */\n"                                /* r15d = dist */\n\n"    cld\n"    cmpq    %%rdi, %%r10\n"    je      .L_one_time\n"                /* if only one decode left */\n"    cmpq    %%rsi, %%r9\n"    je      .L_one_time\n"    jmp     .L_do_loop\n\n.L_one_time:\n"    movq    %%r12, %%r8\n"                /* r8 = lmask */\n"    cmpb    $32, %%bl\n"    ja      .L_get_length_code_one_time\n\n"    lodsl\n"                /* eax = *(uint *)in++ */\n"    movb    %%bl, %%cl\n"                /* cl = bits, needs it for shifting */\n"    addb    $32, %%bl\n"                /* bits += 32 */\n"    shlq    %%cl, %%rax\n"    orq     %%rax, %%rdx\n"                /* hold |= *((uint *)in)++ << bits */\n"    jmp     .L_get_length_code_one_time\n\n.align 32,0x90\n.L_while_test:\n"    cmpq    %%rdi, %%r10\n"    jbe     .L_break_loop\n"    cmpq    %%rsi, %%r9\n"    jbe     .L_break_loop\n\n.L_do_loop:\n"    movq    %%r12, %%r8\n"                /* r8 = lmask */\n"    cmpb    $32, %%bl\n"    ja      .L_get_length_code\n"                /* if (32 < bits) */\n\n"    lodsl\n"                /* eax = *(uint *)in++ */\n"    movb    %%bl, %%cl\n"                /* cl = bits, needs it for shifting */\n"    addb    $32, %%bl\n"                /* bits += 32 */\n"    shlq    %%cl, %%rax\n"    orq     %%rax, %%rdx\n"                /* hold |= *((uint *)in)++ << bits */\n\n.L_get_length_code:\n"    andq    %%rdx, %%r8\n"                /* r8 &= hold */\n"    movl    (%rbp,%%r8,4), %%eax\n"                /* eax = lcode[hold & lmask] */\n\n"    movb    %%ah, %%cl\n"                /* cl = this.bits */\n"    subb    %%ah, %%bl\n"                /* bits -= this.bits */\n"    shrq    %%cl, %%rdx\n"                /* hold >= this.bits */\n\n"    testb   %%al, %%al\n"    jnz     .L_test_for_length_base\n"                /* if (op != 0) 45.7% */\n\n"    movq    %%r12, %%r8\n"                /* r8 = lmask */\n"    shrl    $16, %%eax\n"                /* output this.val char */\n"    stosb\n\n.L_get_length_code_one_time:\n"    andq    %%rdx, %%r8\n"                /* r8 &= hold */\n"    movl    (%rbp,%%r8,4), %%eax\n"                /* eax = lcode[hold & lmask] */\n\n.L_dolen:\n"    movb    %%ah, %%cl\n"                /* cl = this.bits */\n"    subb    %%ah, %%bl\n"                /* bits -= this.bits */\n"    shrq    %%cl, %%rdx\n"                /* hold >= this.bits */\n\n"    testb   %%al, %%al\n"    jnz     .L_test_for_length_base\n"                /* if (op != 0) 45.7% */\n\n"    shrl    $16, %%eax\n"                /* output this.val char */\n"    stosb\n"    jmp     .L_while_test\n\n.align 32,0x90\n.L_test_for_length_base:\n"    movl    %%eax, %%r14d\n"                /* len = this */
```

```
"    shr    $16, %%r14d\n"                /* len = this.val */
"    movb   %%al, %%cl\n"

"    testb  $16, %%al\n"
"    jz     .L_test_for_second_level_length\n" /* if ((op & 16) == 0) 8% */
"    andb   $15, %%cl\n"                /* op &= 15 */
"    jz     .L_decode_distance\n"        /* if (!op) */

.L_add_bits_to_len:\n"
"    subb   %%cl, %%bl\n"
"    xorl   %%eax, %%eax\n"
"    incl   %%eax\n"
"    shll   %%cl, %%eax\n"
"    decl   %%eax\n"
"    andl   %%edx, %%eax\n"                /* eax &= hold */
"    shrq   %%cl, %%rdx\n"
"    addl   %%eax, %%r14d\n"                /* len += hold & mask[op] */

.L_decode_distance:\n"
"    movq   %%r13, %%r8\n"                /* r8 = dmask */
"    cmpb   $32, %%bl\n"
"    ja     .L_get_distance_code\n"        /* if (32 < bits) */

"    lodsl\n"                            /* eax = *(uint *)in++ */
"    movb   %%bl, %%cl\n"                /* cl = bits, needs it for shifting */
"    addb   $32, %%bl\n"                /* bits += 32 */
"    shlq   %%cl, %%rax\n"
"    orq    %%rax, %%rdx\n"                /* hold |= *((uint *)in)++ << bits */

.L_get_distance_code:\n"
"    andq   %%rdx, %%r8\n"                /* r8 &= hold */
"    movl   (%r11, %%r8, 4), %%eax\n"      /* eax = dcode[hold & dmask] */

.L_dodist:\n"
"    movl   %%eax, %%r15d\n"                /* dist = this */
"    shr    $16, %%r15d\n"                /* dist = this.val */
"    movb   %%ah, %%cl\n"
"    subb   %%ah, %%bl\n"                /* bits -= this.bits */
"    shrq   %%cl, %%rdx\n"                /* hold >>= this.bits */
"    movb   %%al, %%cl\n"                /* cl = this.op */

"    testb  $16, %%al\n"                /* if ((op & 16) == 0) */
"    jz     .L_test_for_second_level_dist\n"
"    andb   $15, %%cl\n"                /* op &= 15 */
"    jz     .L_check_dist_one\n"

.L_add_bits_to_dist:\n"
"    subb   %%cl, %%bl\n"
"    xorl   %%eax, %%eax\n"
"    incl   %%eax\n"
"    shll   %%cl, %%eax\n"
"    decl   %%eax\n"                /* (1 << op) - 1 */
"    andl   %%edx, %%eax\n"                /* eax &= hold */
"    shrq   %%cl, %%rdx\n"
"    addl   %%eax, %%r15d\n"                /* dist += hold & ((1 << op) - 1) */

.L_check_window:\n"
"    movq   %%rsi, %%r8\n"                /* save in so from can use it's reg */
"    movq   %%rdi, %%rax\n"
"    subq   40(%%rsp), %%rax\n"            /* nbytes = out - beg */

"    cmpl   %%r15d, %%eax\n"
"    jb     .L_clip_window\n"            /* if (dist > nbytes) 4.2% */

"    movl   %%r14d, %%ecx\n"                /* ecx = len */
"    movq   %%rdi, %%rsi\n"
"    subq   %%r15, %%rsi\n"                /* from = out - dist */

"    sarl   %%ecx\n"
"    jnc    .L_copy_two\n"                /* if len % 2 == 0 */

"    rep    movsw\n"
"    movb   (%%rsi), %%al\n"
"    movb   %%al, (%%rdi)\n"
```



```
"    incq    %%rdi\n"

"    movq    %%r8, %%rsi\n"          /* move in back to %rsi, toss from */
"    jmp     .L_while_test\n"

".L_copy_two:\n"
"    rep     movsw\n"
"    movq    %%r8, %%rsi\n"          /* move in back to %rsi, toss from */
"    jmp     .L_while_test\n"

".align 32,0x90\n"
".L_check_dist_one:\n"
"    cmpl    $1, %%r15d\n"          /* if dist 1, is a memset */
"    jne     .L_check_window\n"
"    cmpq    %%rdi, 40(%%rsp)\n"     /* if out == beg, outside window */
"    je      .L_check_window\n"

"    movl    %%r14d, %%ecx\n"        /* ecx = len */
"    movb    -1(%%rdi), %%al\n"
"    movb    %%al, %%ah\n"

"    sarl    %%ecx\n"
"    jnc     .L_set_two\n"
"    movb    %%al, (%%rdi)\n"
"    incq    %%rdi\n"

".L_set_two:\n"
"    rep     stosw\n"
"    jmp     .L_while_test\n"

".align 32,0x90\n"
".L_test_for_second_level_length:\n"
"    testb    $64, %%al\n"
"    jnz      .L_test_for_end_of_block\n" /* if ((op & 64) != 0) */

"    xorl     %%eax, %%eax\n"
"    incl     %%eax\n"
"    shll     %%cl, %%eax\n"
"    decl     %%eax\n"
"    andl     %%edx, %%eax\n"        /* eax &= hold */
"    addl     %%r14d, %%eax\n"       /* eax += len */
"    movl     (%%rbp, %%rax, 4), %%eax\n" /* eax = lcode[val+(hold&mask[op])] */
"    jmp      .L_dolen\n"

".align 32,0x90\n"
".L_test_for_second_level_dist:\n"
"    testb    $64, %%al\n"
"    jnz      .L_invalid_distance_code\n" /* if ((op & 64) != 0) */

"    xorl     %%eax, %%eax\n"
"    incl     %%eax\n"
"    shll     %%cl, %%eax\n"
"    decl     %%eax\n"
"    andl     %%edx, %%eax\n"        /* eax &= hold */
"    addl     %%r15d, %%eax\n"       /* eax += dist */
"    movl     (%%r11, %%rax, 4), %%eax\n" /* eax = dcode[val+(hold&mask[op])] */
"    jmp      .L_dodist\n"

".align 32,0x90\n"
".L_clip_window:\n"
"    movl     %%eax, %%ecx\n"        /* ecx = nbytes */
"    movl     92(%%rsp), %%eax\n"     /* eax = wsize, prepare for dist cmp */
"    negl     %%ecx\n"              /* nbytes = -nbytes */

"    cmpl     %%r15d, %%eax\n"
"    jb       .L_invalid_distance_too_far\n" /* if (dist > wsize) */

"    addl     %%r15d, %%ecx\n"        /* nbytes = dist - nbytes */
"    cmpl     $0, 96(%%rsp)\n"
"    jne      .L_wrap_around_window\n" /* if (write != 0) */

"    movq     56(%%rsp), %%rsi\n"     /* from = window */
"    subl     %%ecx, %%eax\n"         /* eax -= nbytes */
"    addq     %%rax, %%rsi\n"         /* from += wsize - nbytes */
```

```

"    movl    %%r14d, %%eax\n"           /* eax = len */
"    cmpl    %%ecx, %%r14d\n"
"    jbe     .L_do_copy\n"             /* if (nbytes >= len) */

"    subl    %%ecx, %%eax\n"           /* eax -= nbytes */
"    rep     movsb\n"
"    movq     %%rdi, %%rsi\n"
"    subq     %%r15, %%rsi\n"         /* from = &out[ -dist ] */
"    jmp     .L_do_copy\n"

".align 32,0x90\n"
".L_wrap_around_window:\n"
"    movl    96(%%rsp), %%eax\n"       /* eax = write */
"    cmpl    %%eax, %%ecx\n"
"    jbe     .L_contiguous_in_window\n" /* if (write >= nbytes) */

"    movl    92(%%rsp), %%esi\n"       /* from = wsize */
"    addq     56(%%rsp), %%rsi\n"       /* from += window */
"    addq     %%rax, %%rsi\n"          /* from += write */
"    subq     %%rcx, %%rsi\n"          /* from -= nbytes */
"    subl    %%eax, %%ecx\n"           /* nbytes -= write */

"    movl    %%r14d, %%eax\n"           /* eax = len */
"    cmpl    %%ecx, %%eax\n"
"    jbe     .L_do_copy\n"             /* if (nbytes >= len) */

"    subl    %%ecx, %%eax\n"           /* len -= nbytes */
"    rep     movsb\n"
"    movq     56(%%rsp), %%rsi\n"       /* from = window */
"    movl    96(%%rsp), %%ecx\n"       /* nbytes = write */
"    cmpl    %%ecx, %%eax\n"
"    jbe     .L_do_copy\n"             /* if (nbytes >= len) */

"    subl    %%ecx, %%eax\n"           /* len -= nbytes */
"    rep     movsb\n"
"    movq     %%rdi, %%rsi\n"
"    subq     %%r15, %%rsi\n"         /* from = out - dist */
"    jmp     .L_do_copy\n"

".align 32,0x90\n"
".L_contiguous_in_window:\n"
"    movq     56(%%rsp), %%rsi\n"       /* rsi = window */
"    addq     %%rax, %%rsi\n"
"    subq     %%rcx, %%rsi\n"         /* from += write - nbytes */

"    movl    %%r14d, %%eax\n"           /* eax = len */
"    cmpl    %%ecx, %%eax\n"
"    jbe     .L_do_copy\n"             /* if (nbytes >= len) */

"    subl    %%ecx, %%eax\n"           /* len -= nbytes */
"    rep     movsb\n"
"    movq     %%rdi, %%rsi\n"
"    subq     %%r15, %%rsi\n"         /* from = out - dist */
"    jmp     .L_do_copy\n"           /* if (nbytes >= len) */

".align 32,0x90\n"
".L_do_copy:\n"
"    movl    %%eax, %%ecx\n"           /* ecx = len */
"    rep     movsb\n"

"    movq     %%r8, %%rsi\n"           /* move in back to %esi, toss from */
"    jmp     .L_while_test\n"

".L_test_for_end_of_block:\n"
"    testb    $32, %%al\n"
"    jz       .L_invalid_literal_length_code\n"
"    movl     $1, 116(%%rsp)\n"
"    jmp     .L_break_loop_with_status\n"

".L_invalid_literal_length_code:\n"
"    movl     $2, 116(%%rsp)\n"
"    jmp     .L_break_loop_with_status\n"

```

```

.L_invalid_distance_code:\n"
"    movl    $3, 116(%%rsp)\n"
"    jmp     .L_break_loop_with_status\n"

.L_invalid_distance_too_far:\n"
"    movl    $4, 116(%%rsp)\n"
"    jmp     .L_break_loop_with_status\n"

.L_break_loop:\n"
"    movl    $0, 116(%%rsp)\n"

.L_break_loop_with_status:\n"
/* put in, out, bits, and hold back into ar and pop esp */
"    movq    %rsi, 16(%%rsp)\n"        /* in */
"    movq    %rdi, 32(%%rsp)\n"        /* out */
"    movl    %%ebx, 88(%%rsp)\n"        /* bits */
"    movq    %%rdx, 80(%%rsp)\n"        /* hold */
"    movq    (%%rsp), %%rax\n"          /* restore rbp and rsp */
"    movq    8(%%rsp), %%rbp\n"
"    movq    %%rax, %%rsp\n"
"    :
"    : "m" (ar)
"    : "memory", "%rax", "%rbx", "%rcx", "%rdx", "%rsi", "%rdi",
      "%r8", "%r9", "%r10", "%r11", "%r12", "%r13", "%r14", "%r15"
"    );
#elif ( defined( __GNUC__ ) || defined( __ICC ) ) && defined( __i386 )
"    asm volatile (
"        leal    %0, %%eax\n"
"        movl    %%esp, (%%eax)\n"        /* save esp, ebp */
"        movl    %%ebp, 4(%%eax)\n"
"        movl    %%eax, %%esp\n"
"        movl    8(%%esp), %%esi\n"        /* esi = in */
"        movl    16(%%esp), %%edi\n"        /* edi = out */
"        movl    40(%%esp), %%edx\n"        /* edx = hold */
"        movl    44(%%esp), %%ebx\n"        /* ebx = bits */
"        movl    32(%%esp), %%ebp\n"        /* ebp = lcode */
"
"        cld\n"
"        jmp     .L_do_loop\n"

.align 32,0x90\n"
.L_while_test:\n"
"    cmpl    %%edi, 24(%%esp)\n"        /* out < end */
"    jbe     .L_break_loop\n"
"    cmpl    %%esi, 12(%%esp)\n"        /* in < last */
"    jbe     .L_break_loop\n"

.L_do_loop:\n"
"    cmpb    $15, %%bl\n"
"    ja      .L_get_length_code\n"        /* if (15 < bits) */

"    xorl    %%eax, %%eax\n"
"    lodsw\n"                            /* al = *(ushort *)in++ */
"    movb    %%bl, %%cl\n"                /* cl = bits, needs it for shifting */
"    addb    $16, %%bl\n"                /* bits += 16 */
"    shll    %%cl, %%eax\n"
"    orl     %%eax, %%edx\n"                /* hold |= *((ushort *)in)++ << bits */

.L_get_length_code:\n"
"    movl    56(%%esp), %%eax\n"        /* eax = lmask */
"    andl    %%edx, %%eax\n"            /* eax &= hold */
"    movl    (%%ebp, %%eax, 4), %%eax\n" /* eax = lcode[hold & lmask] */

.L_dolen:\n"
"    movb    %%ah, %%cl\n"                /* cl = this.bits */
"    subb    %%ah, %%bl\n"                /* bits -= this.bits */
"    shr     %%cl, %%edx\n"                /* hold >>= this.bits */

"    testb   %%al, %%al\n"
"    jnz     .L_test_for_length_base\n" /* if (op != 0) 45.7% */

"    shr     $16, %%eax\n"                /* output this.val char */
"    stosb\n"
"    jmp     .L_while_test\n"

```

```

".align 32,0x90\n"
".L_test_for_length_base:\n"
"    movl    %%eax, %%ecx\n"           /* len = this */
"    shr     $16, %%ecx\n"           /* len = this.val */
"    movl    %%ecx, 64(%%esp)\n"      /* save len */
"    movb    %%al, %%cl\n"

"    testb   $16, %%al\n"
"    jz      .L_test_for_second_level_length\n" /* if ((op & 16) == 0) 8% */
"    andb    $15, %%cl\n"           /* op &= 15 */
"    jz      .L_decode_distance\n"    /* if (!op) */
"    cmpb    %%cl, %%bl\n"
"    jae     .L_add_bits_to_len\n"    /* if (op <= bits) */

"    movb    %%cl, %%ch\n"           /* stash op in ch, freeing cl */
"    xorl    %%eax, %%eax\n"
"    lodsw\n"                         /* al = *(ushort *)in++ */
"    movb    %%bl, %%cl\n"           /* cl = bits, needs it for shifting */
"    addb    $16, %%bl\n"           /* bits += 16 */
"    shll    %%cl, %%eax\n"
"    orl     %%eax, %%edx\n"         /* hold |= *((ushort *)in)++ << bits */
"    movb    %%ch, %%cl\n"           /* move op back to ecx */

".L_add_bits_to_len:\n"
"    subb    %%cl, %%bl\n"
"    xorl    %%eax, %%eax\n"
"    incl    %%eax\n"
"    shll    %%cl, %%eax\n"
"    decl    %%eax\n"
"    andl    %%edx, %%eax\n"         /* eax &= hold */
"    shr     %%cl, %%edx\n"
"    addl    %%eax, 64(%%esp)\n"     /* len += hold & mask[op] */

".L_decode_distance:\n"
"    cmpb    $15, %%bl\n"
"    ja      .L_get_distance_code\n" /* if (15 < bits) */

"    xorl    %%eax, %%eax\n"
"    lodsw\n"                         /* al = *(ushort *)in++ */
"    movb    %%bl, %%cl\n"           /* cl = bits, needs it for shifting */
"    addb    $16, %%bl\n"           /* bits += 16 */
"    shll    %%cl, %%eax\n"
"    orl     %%eax, %%edx\n"         /* hold |= *((ushort *)in)++ << bits */

".L_get_distance_code:\n"
"    movl    60(%%esp), %%eax\n"      /* eax = dmask */
"    movl    36(%%esp), %%ecx\n"      /* ecx = dcode */
"    andl    %%edx, %%eax\n"         /* eax &= hold */
"    movl    (%%ecx, %%eax, 4), %%eax\n" /* eax = dcode[hold & dmask] */

".L_dodist:\n"
"    movl    %%eax, %%ebp\n"          /* dist = this */
"    shr     $16, %%ebp\n"           /* dist = this.val */
"    movb    %%ah, %%cl\n"
"    subb    %%ah, %%bl\n"           /* bits -= this.bits */
"    shr     %%cl, %%edx\n"         /* hold >>= this.bits */
"    movb    %%al, %%cl\n"           /* cl = this.op */

"    testb   $16, %%al\n"           /* if ((op & 16) == 0) */
"    jz      .L_test_for_second_level_dist\n"
"    andb    $15, %%cl\n"           /* op &= 15 */
"    jz      .L_check_dist_one\n"
"    cmpb    %%cl, %%bl\n"
"    jae     .L_add_bits_to_dist\n"  /* if (op <= bits) 97.6% */

"    movb    %%cl, %%ch\n"           /* stash op in ch, freeing cl */
"    xorl    %%eax, %%eax\n"
"    lodsw\n"                         /* al = *(ushort *)in++ */
"    movb    %%bl, %%cl\n"           /* cl = bits, needs it for shifting */
"    addb    $16, %%bl\n"           /* bits += 16 */
"    shll    %%cl, %%eax\n"
"    orl     %%eax, %%edx\n"         /* hold |= *((ushort *)in)++ << bits */
"    movb    %%ch, %%cl\n"           /* move op back to ecx */

```

```

.L_add_bits_to_dist:\n"
"    subb    %%cl, %%bl\n"
"    xorl    %%eax, %%eax\n"
"    incl    %%eax\n"
"    shll    %%cl, %%eax\n"
"    decl    %%eax\n"
"    andl    %%edx, %%eax\n"
"    shr     %%cl, %%edx\n"
"    addl    %%eax, %%ebp\n"
"                                     /* (1 << op) - 1 */
"                                     /* eax &= hold */
"                                     /* dist += hold & ((1 << op) - 1) */

.L_check_window:\n"
"    movl    %%esi, 8(%%esp)\n"
"    movl    %%edi, %%eax\n"
"    subl    20(%%esp), %%eax\n"
"                                     /* save in so from can use it's reg */
"                                     /* nbytes = out - beg */

"    cmpl    %%ebp, %%eax\n"
"    jb      .L_clip_window\n"
"                                     /* if (dist > nbytes) 4.2% */

"    movl    64(%%esp), %%ecx\n"
"    movl    %%edi, %%esi\n"
"    subl    %%ebp, %%esi\n"
"                                     /* ecx = len */
"                                     /* from = out - dist */

"    sarl    %%ecx\n"
"    jnc     .L_copy_two\n"
"                                     /* if len % 2 == 0 */

"    rep     movsw\n"
"    movb    (%%esi), %%al\n"
"    movb    %%al, (%%edi)\n"
"    incl    %%edi\n"

"    movl    8(%%esp), %%esi\n"
"    movl    32(%%esp), %%ebp\n"
"    jmp     .L_while_test\n"
"                                     /* move in back to %esi, toss from */
"                                     /* ebp = lcode */

.L_copy_two:\n"
"    rep     movsw\n"
"    movl    8(%%esp), %%esi\n"
"    movl    32(%%esp), %%ebp\n"
"    jmp     .L_while_test\n"
"                                     /* move in back to %esi, toss from */
"                                     /* ebp = lcode */

.align 32,0x90\n"
.L_check_dist_one:\n"
"    cmpl    $1, %%ebp\n"
"    jne     .L_check_window\n"
"    cmpl    %%edi, 20(%%esp)\n"
"    je      .L_check_window\n"
"                                     /* if dist 1, is a memset */
"                                     /* out == beg, if outside window */

"    movl    64(%%esp), %%ecx\n"
"    movb    -1(%%edi), %%al\n"
"    movb    %%al, %%ah\n"
"                                     /* ecx = len */

"    sarl    %%ecx\n"
"    jnc     .L_set_two\n"
"    movb    %%al, (%%edi)\n"
"    incl    %%edi\n"

.L_set_two:\n"
"    rep     stosw\n"
"    movl    32(%%esp), %%ebp\n"
"    jmp     .L_while_test\n"
"                                     /* ebp = lcode */

.align 32,0x90\n"
.L_test_for_second_level_length:\n"
"    testb    $64, %%al\n"
"    jnz     .L_test_for_end_of_block\n"
"                                     /* if ((op & 64) != 0) */

"    xorl    %%eax, %%eax\n"
"    incl    %%eax\n"
"    shll    %%cl, %%eax\n"
"    decl    %%eax\n"
"    andl    %%edx, %%eax\n"
"    addl    64(%%esp), %%eax\n"
"    movl    (%%ebp, %%eax, 4), %%eax\n"
"                                     /* eax &= hold */
"                                     /* eax += len */
"                                     /* eax = lcode[val+(hold&mask[op])]*/

```

```
"    jmp    .L_dolen\n"

".align 32,0x90\n"
".L_test_for_second_level_dist:\n"
"    testb  $64,%%eax\n"
"    jnz    .L_invalid_distance_code\n" /* if ((op & 64) != 0) */

"    xorl   %%eax,%%eax\n"
"    incl   %%eax\n"
"    shll   %%cl,%%eax\n"
"    decl   %%eax\n"
"    andl   %%edx,%%eax\n"          /* eax &= hold */
"    addl   %%ebp,%%eax\n"          /* eax += dist */
"    movl   36(%%esp),%%ecx\n"      /* ecx = dcode */
"    movl   (%%ecx,%%eax,4),%%eax\n" /* eax = dcode[val+(hold&mask[op])] */
"    jmp    .L_dodist\n"

".align 32,0x90\n"
".L_clip_window:\n"
"    movl   %%eax,%%ecx\n"
"    movl   48(%%esp),%%eax\n"      /* eax = wsize */
"    negl   %%ecx\n"                /* nbytes = -nbytes */
"    movl   28(%%esp),%%esi\n"      /* from = window */

"    cmpl   %%ebp,%%eax\n"
"    jb     .L_invalid_distance_too_far\n" /* if (dist > wsize) */

"    addl   %%ebp,%%ecx\n"          /* nbytes = dist - nbytes */
"    cmpl   $0,52(%%esp)\n"
"    jne    .L_wrap_around_window\n" /* if (write != 0) */

"    subl   %%ecx,%%eax\n"
"    addl   %%eax,%%esi\n"          /* from += wsize - nbytes */

"    movl   64(%%esp),%%eax\n"      /* eax = len */
"    cmpl   %%ecx,%%eax\n"
"    jbe    .L_do_copy\n"          /* if (nbytes >= len) */

"    subl   %%ecx,%%eax\n"          /* len -= nbytes */
"    rep    movsb\n"
"    movl   %%edi,%%esi\n"
"    subl   %%ebp,%%esi\n"          /* from = out - dist */
"    jmp    .L_do_copy\n"

".align 32,0x90\n"
".L_wrap_around_window:\n"
"    movl   52(%%esp),%%eax\n"      /* eax = write */
"    cmpl   %%eax,%%ecx\n"
"    jbe    .L_contiguous_in_window\n" /* if (write >= nbytes) */

"    addl   48(%%esp),%%esi\n"      /* from += wsize */
"    addl   %%eax,%%esi\n"          /* from += write */
"    subl   %%ecx,%%esi\n"          /* from -= nbytes */
"    subl   %%eax,%%ecx\n"          /* nbytes -= write */

"    movl   64(%%esp),%%eax\n"      /* eax = len */
"    cmpl   %%ecx,%%eax\n"
"    jbe    .L_do_copy\n"          /* if (nbytes >= len) */

"    subl   %%ecx,%%eax\n"          /* len -= nbytes */
"    rep    movsb\n"
"    movl   28(%%esp),%%esi\n"      /* from = window */
"    movl   52(%%esp),%%ecx\n"      /* nbytes = write */
"    cmpl   %%ecx,%%eax\n"
"    jbe    .L_do_copy\n"          /* if (nbytes >= len) */

"    subl   %%ecx,%%eax\n"          /* len -= nbytes */
"    rep    movsb\n"
"    movl   %%edi,%%esi\n"
"    subl   %%ebp,%%esi\n"          /* from = out - dist */
"    jmp    .L_do_copy\n"

".align 32,0x90\n"
".L_contiguous_in_window:\n"
```

```

"    addl    %%eax, %%esi\n"
"    subl    %%ecx, %%esi\n"                /* from += write - nbytes */

"    movl    64(%%esp), %%eax\n"            /* eax = len */
"    cmpl    %%ecx, %%eax\n"
"    jbe     .L_do_copy\n"                /* if (nbytes >= len) */

"    subl    %%ecx, %%eax\n"                /* len -= nbytes */
"    rep     movsb\n"
"    movl    %%edi, %%esi\n"
"    subl    %%ebp, %%esi\n"                /* from = out - dist */
"    jmp     .L_do_copy\n"                /* if (nbytes >= len) */

".align 32,0x90\n"
".L_do_copy:\n"
"    movl    %%eax, %%ecx\n"
"    rep     movsb\n"

"    movl    8(%%esp), %%esi\n"            /* move in back to %esi, toss from */
"    movl    32(%%esp), %%ebp\n"            /* ebp = lcode */
"    jmp     .L_while_test\n"

".L_test_for_end_of_block:\n"
"    testb   $32, %%al\n"
"    jz      .L_invalid_literal_length_code\n"
"    movl    $1, 72(%%esp)\n"
"    jmp     .L_break_loop_with_status\n"

".L_invalid_literal_length_code:\n"
"    movl    $2, 72(%%esp)\n"
"    jmp     .L_break_loop_with_status\n"

".L_invalid_distance_code:\n"
"    movl    $3, 72(%%esp)\n"
"    jmp     .L_break_loop_with_status\n"

".L_invalid_distance_too_far:\n"
"    movl    8(%%esp), %%esi\n"
"    movl    $4, 72(%%esp)\n"
"    jmp     .L_break_loop_with_status\n"

".L_break_loop:\n"
"    movl    $0, 72(%%esp)\n"

".L_break_loop_with_status:\n"
/* put in, out, bits, and hold back into ar and pop esp */
"    movl    %%esi, 8(%%esp)\n"            /* save in */
"    movl    %%edi, 16(%%esp)\n"           /* save out */
"    movl    %%ebx, 44(%%esp)\n"           /* save bits */
"    movl    %%edx, 40(%%esp)\n"           /* save hold */
"    movl    4(%%esp), %%ebp\n"            /* restore esp, ebp */
"    movl    (%%esp), %%esp\n"
"    :\n"
"    : "m" (ar)\n"
"    : "memory", "%eax", "%ebx", "%ecx", "%edx", "%esi", "%edi"
);
#elif defined( _MSC_VER ) && ! defined( _M_AMD64 )
__asm {
    lea     eax, ar
    mov     [eax], esp                /* save esp, ebp */
    mov     [eax+4], ebp
    mov     esp, eax
    mov     esi, [esp+8]              /* esi = in */
    mov     edi, [esp+16]             /* edi = out */
    mov     edx, [esp+40]             /* edx = hold */
    mov     ebx, [esp+44]             /* ebx = bits */
    mov     ebp, [esp+32]             /* ebp = lcode */

    cld
    jmp     L_do_loop

ALIGN 4
L_while_test:
    cmp     [esp+24], edi

```

```

    jbe     L_break_loop
    cmp     [esp+12], esi
    jbe     L_break_loop

L_do_loop:
    cmp     bl, 15
    ja      L_get_length_code    /* if (15 < bits) */

    xor     eax, eax
    lodsw                     /* al = *(ushort *)in++ */
    mov     cl, bl             /* cl = bits, needs it for shifting */
    add     bl, 16             /* bits += 16 */
    shl     eax, cl
    or      edx, eax           /* hold |= *((ushort *)in)++ << bits */

L_get_length_code:
    mov     eax, [esp+56]       /* eax = lmask */
    and     eax, edx           /* eax &= hold */
    mov     eax, [ebp+eax*4]     /* eax = lcode[hold & lmask] */

L_dolen:
    mov     cl, ah             /* cl = this.bits */
    sub     bl, ah             /* bits -= this.bits */
    shr     edx, cl            /* hold >>= this.bits */

    test    al, al
    jnz     L_test_for_length_base /* if (op != 0) 45.7% */

    shr     eax, 16             /* output this.val char */
    stosb
    jmp     L_while_test

ALIGN 4
L_test_for_length_base:
    mov     ecx, eax           /* len = this */
    shr     ecx, 16            /* len = this.val */
    mov     [esp+64], ecx       /* save len */
    mov     cl, al

    test    al, 16
    jz      L_test_for_second_level_length /* if ((op & 16) == 0) 8% */
    and     cl, 15             /* op &= 15 */
    jz      L_decode_distance  /* if (!op) */
    cmp     bl, cl
    jae     L_add_bits_to_len   /* if (op <= bits) */

    mov     ch, cl             /* stash op in ch, freeing cl */
    xor     eax, eax
    lodsw                     /* al = *(ushort *)in++ */
    mov     cl, bl             /* cl = bits, needs it for shifting */
    add     bl, 16             /* bits += 16 */
    shl     eax, cl
    or      edx, eax           /* hold |= *((ushort *)in)++ << bits */
    mov     cl, ch             /* move op back to ecx */

L_add_bits_to_len:
    sub     bl, cl
    xor     eax, eax
    inc     eax
    shl     eax, cl
    dec     eax
    and     eax, edx           /* eax &= hold */
    shr     edx, cl
    add     [esp+64], eax       /* len += hold & mask[op] */

L_decode_distance:
    cmp     bl, 15
    ja      L_get_distance_code /* if (15 < bits) */

    xor     eax, eax
    lodsw                     /* al = *(ushort *)in++ */
    mov     cl, bl             /* cl = bits, needs it for shifting */
    add     bl, 16             /* bits += 16 */
    shl     eax, cl

```



```

    or     edx, eax          /* hold |= *((ushort *)in)++ << bits */

L_get_distance_code:
    mov     eax, [esp+60]     /* eax = dmask */
    mov     ecx, [esp+36]     /* ecx = dcode */
    and     eax, edx         /* eax &= hold */
    mov     eax, [ecx+eax*4] /* eax = dcode[hold & dmask] */

L_dodist:
    mov     ebp, eax         /* dist = this */
    shr     ebp, 16          /* dist = this.val */
    mov     cl, ah
    sub     bl, ah           /* bits -= this.bits */
    shr     edx, cl          /* hold >>= this.bits */
    mov     cl, al           /* cl = this.op */

    test    al, 16           /* if ((op & 16) == 0) */
    jz      L_test_for_second_level_dist
    and     cl, 15           /* op &= 15 */
    jz      L_check_dist_one
    cmp     bl, cl
    jae     L_add_bits_to_dist /* if (op <= bits) 97.6% */

    mov     ch, cl           /* stash op in ch, freeing cl */
    xor     eax, eax
    lodsw                     /* al = *((ushort *)in)++ */
    mov     cl, bl           /* cl = bits, needs it for shifting */
    add     bl, 16           /* bits += 16 */
    shl     eax, cl
    or      edx, eax         /* hold |= *((ushort *)in)++ << bits */
    mov     cl, ch           /* move op back to ecx */

L_add_bits_to_dist:
    sub     bl, cl
    xor     eax, eax
    inc     eax
    shl     eax, cl
    dec     eax              /* (1 << op) - 1 */
    and     eax, edx         /* eax &= hold */
    shr     edx, cl
    add     ebp, eax         /* dist += hold & ((1 << op) - 1) */

L_check_window:
    mov     [esp+8], esi     /* save in so from can use it's reg */
    mov     eax, edi
    sub     eax, [esp+20]    /* nbytes = out - beg */

    cmp     eax, ebp
    jb      L_clip_window   /* if (dist > nbytes) 4.2% */

    mov     ecx, [esp+64]    /* ecx = len */
    mov     esi, edi
    sub     esi, ebp         /* from = out - dist */

    sar     ecx, 1
    jnc     L_copy_two

    rep     movsw
    mov     al, [esi]
    mov     [edi], al
    inc     edi

    mov     esi, [esp+8]     /* move in back to %esi, toss from */
    mov     ebp, [esp+32]    /* ebp = lcode */
    jmp     L_while_test

L_copy_two:
    rep     movsw
    mov     esi, [esp+8]     /* move in back to %esi, toss from */
    mov     ebp, [esp+32]    /* ebp = lcode */
    jmp     L_while_test

```

```

ALIGN 4
L_check_dist_one:

```

```

    cmp     ebp, 1          /* if dist 1, is a memset */
    jne     L_check_window
    cmp     [esp+20], edi
    je      L_check_window  /* out == beg, if outside window */

    mov     ecx, [esp+64]    /* ecx = len */
    mov     al, [edi-1]
    mov     ah, al

    sar     ecx, 1
    jnc     L_set_two
    mov     [edi], al        /* memset out with from[-1] */
    inc     edi

L_set_two:
    rep     stosw
    mov     ebp, [esp+32]    /* ebp = lcode */
    jmp     L_while_test

ALIGN 4
L_test_for_second_level_length:
    test    al, 64
    jnz     L_test_for_end_of_block /* if ((op & 64) != 0) */

    xor     eax, eax
    inc     eax
    shl     eax, cl
    dec     eax
    and     eax, edx         /* eax &= hold */
    add     eax, [esp+64]     /* eax += len */
    mov     eax, [ebp+eax*4] /* eax = lcode[val+(hold&mask[op])] */
    jmp     L_dolen

ALIGN 4
L_test_for_second_level_dist:
    test    al, 64
    jnz     L_invalid_distance_code /* if ((op & 64) != 0) */

    xor     eax, eax
    inc     eax
    shl     eax, cl
    dec     eax
    and     eax, edx         /* eax &= hold */
    add     eax, ebp         /* eax += dist */
    mov     ecx, [esp+36]     /* ecx = dcode */
    mov     eax, [ecx+eax*4] /* eax = dcode[val+(hold&mask[op])] */
    jmp     L_dodist

ALIGN 4
L_clip_window:
    mov     ecx, eax
    mov     eax, [esp+48]     /* eax = wsize */
    neg     ecx              /* nbytes = -nbytes */
    mov     esi, [esp+28]     /* from = window */

    cmp     eax, ebp
    jb      L_invalid_distance_too_far /* if (dist > wsize) */

    add     ecx, ebp         /* nbytes = dist - nbytes */
    cmp     dword ptr [esp+52], 0
    jne     L_wrap_around_window /* if (write != 0) */

    sub     eax, ecx
    add     esi, eax         /* from += wsize - nbytes */

    mov     eax, [esp+64]    /* eax = len */
    cmp     eax, ecx
    jbe     L_do_copy        /* if (nbytes >= len) */

    sub     eax, ecx         /* len -= nbytes */
    rep     movsb
    mov     esi, edi
    sub     esi, ebp         /* from = out - dist */
    jmp     L_do_copy

```

ALIGN 4

L_wrap_around_window:

```
    mov     eax, [esp+52]    /* eax = write */
    cmp     ecx, eax
    jbe     L_contiguous_in_window /* if (write >= nbytes) */

    add     esi, [esp+48]    /* from += wsize */
    add     esi, eax        /* from += write */
    sub     esi, ecx        /* from -= nbytes */
    sub     ecx, eax        /* nbytes -= write */

    mov     eax, [esp+64]    /* eax = len */
    cmp     eax, ecx
    jbe     L_do_copy       /* if (nbytes >= len) */

    sub     eax, ecx        /* len -= nbytes */
    rep     movsb
    mov     esi, [esp+28]    /* from = window */
    mov     ecx, [esp+52]    /* nbytes = write */
    cmp     eax, ecx
    jbe     L_do_copy       /* if (nbytes >= len) */

    sub     eax, ecx        /* len -= nbytes */
    rep     movsb
    mov     esi, edi
    sub     esi, ebp        /* from = out - dist */
    jmp     L_do_copy
```

ALIGN 4

L_contiguous_in_window:

```
    add     esi, eax
    sub     esi, ecx        /* from += write - nbytes */

    mov     eax, [esp+64]    /* eax = len */
    cmp     eax, ecx
    jbe     L_do_copy       /* if (nbytes >= len) */

    sub     eax, ecx        /* len -= nbytes */
    rep     movsb
    mov     esi, edi
    sub     esi, ebp        /* from = out - dist */
    jmp     L_do_copy
```

ALIGN 4

L_do_copy:

```
    mov     ecx, eax
    rep     movsb

    mov     esi, [esp+8]    /* move in back to %esi, toss from */
    mov     ebp, [esp+32]   /* ebp = lcode */
    jmp     L_while_test
```

L_test_for_end_of_block:

```
    test    al, 32
    jz      L_invalid_literal_length_code
    mov     dword ptr [esp+72], 1
    jmp     L_break_loop_with_status
```

L_invalid_literal_length_code:

```
    mov     dword ptr [esp+72], 2
    jmp     L_break_loop_with_status
```

L_invalid_distance_code:

```
    mov     dword ptr [esp+72], 3
    jmp     L_break_loop_with_status
```

L_invalid_distance_too_far:

```
    mov     esi, [esp+4]
    mov     dword ptr [esp+72], 4
    jmp     L_break_loop_with_status
```

L_break_loop:

```
    mov     dword ptr [esp+72], 0
```

```

L_break_loop_with_status:
/* put in, out, bits, and hold back into ar and pop esp */
    mov     [esp+8], esi      /* save in */
    mov     [esp+16], edi     /* save out */
    mov     [esp+44], ebx     /* save bits */
    mov     [esp+40], edx     /* save hold */
    mov     ebp, [esp+4]      /* restore esp, ebp */
    mov     esp, [esp]
}
#else
#error "x86 architecture not defined"
#endif

    if (ar.status > 1) {
        if (ar.status == 2)
            strm->msg = "invalid literal/length code";
        else if (ar.status == 3)
            strm->msg = "invalid distance code";
        else
            strm->msg = "invalid distance too far back";
        state->mode = BAD;
    }
    else if ( ar.status == 1 ) {
        state->mode = TYPE;
    }

/* return unused bytes (on entry, bits < 8, so in won't go too far back) */
ar.len = ar.bits >> 3;
ar.in -= ar.len;
ar.bits -= ar.len << 3;
ar.hold &= (1U << ar.bits) - 1;

/* update state and return */
strm->next_in = ar.in;
strm->next_out = ar.out;
strm->avail_in = (unsigned)(ar.in < ar.last ?
                           PAD_AVAIL_IN + (ar.last - ar.in) :
                           PAD_AVAIL_IN - (ar.in - ar.last));
strm->avail_out = (unsigned)(ar.out < ar.end ?
                           PAD_AVAIL_OUT + (ar.end - ar.out) :
                           PAD_AVAIL_OUT - (ar.out - ar.end));

state->hold = ar.hold;
state->bits = ar.bits;
return;
}

```

```
/*
 * inffast.S is a hand tuned assembler version of:
 *
 * inffast.c -- fast decoding
 * Copyright (C) 1995-2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 *
 * Copyright (C) 2003 Chris Anderson <christop@charm.net>
 * Please use the copyright conditions above.
 *
 * This version (Jan-23-2003) of inflate_fast was coded and tested under
 * GNU/Linux on a pentium 3, using the gcc-3.2 compiler distribution.  On that
 * machine, I found that gzip style archives decompressed about 20% faster than
 * the gcc-3.2 -O3 -fomit-frame-pointer compiled version.  Your results will
 * depend on how large of a buffer is used for z_stream.next_in & next_out
 * (8K-32K worked best for my 256K cpu cache) and how much overhead there is in
 * stream processing I/O and crc32/addler32.  In my case, this routine used
 * 70% of the cpu time and crc32 used 20%.
 *
 * I am confident that this version will work in the general case, but I have
 * not tested a wide variety of datasets or a wide variety of platforms.
 *
 * Jan-24-2003 -- Added -DUSE_MMX define for slightly faster inflating.
 * It should be a runtime flag instead of compile time flag...
 *
 * Jan-26-2003 -- Added runtime check for MMX support with cpuid instruction.
 * With -DUSE_MMX, only MMX code is compiled.  With -DNO_MMX, only non-MMX code
 * is compiled.  Without either option, runtime detection is enabled.  Runtime
 * detection should work on all modern cpus and the recommended algorithm (flip
 * ID bit on eflags and then use the cpuid instruction) is used in many
 * multimedia applications.  Tested under win2k with gcc-2.95 and gas-2.12
 * distributed with cygwin3.  Compiling with gcc-2.95 -c inffast.S -o
 * inffast.obj generates a COFF object which can then be linked with MSVC++
 * compiled code.  Tested under FreeBSD 4.7 with gcc-2.95.
 *
 * Jan-28-2003 -- Tested Athlon XP... MMX mode is slower than no MMX (and
 * slower than compiler generated code).  Adjusted cpuid check to use the MMX
 * code only for Pentiums < P4 until I have more data on the P4.  Speed
 * improvement is only about 15% on the Athlon when compared with code generated
 * with MSVC++.  Not sure yet, but I think the P4 will also be slower using the
 * MMX mode because many of it's x86 ALU instructions execute in .5 cycles and
 * have less latency than MMX ops.  Added code to buffer the last 11 bytes of
 * the input stream since the MMX code grabs bits in chunks of 32, which
 * differs from the inffast.c algorithm.  I don't think there would have been
 * read overruns where a page boundary was crossed (a segfault), but there
 * could have been overruns when next_in ends on unaligned memory (uninitialized
 * memory read).
 *
 * Mar-13-2003 -- P4 MMX is slightly slower than P4 NO_MMX.  I created a C
 * version of the non-MMX code so that it doesn't depend on zstrm and zstate
 * structure offsets which are hard coded in this file.  This was last tested
 * with zlib-1.2.0 which is currently in beta testing, newer versions of this
 * and inffas86.c can be found at http://www.eetbeetee.com/zlib/ and
 * http://www.charm.net/~christop/zlib/
 */

/*
 * if you have underscore linking problems (_inflate_fast undefined), try
 * using -DGAS_COFF
 */
#if ! defined( GAS_COFF ) && ! defined( GAS_ELF )

#if defined( WIN32 ) || defined( __CYGWIN__ )
#define GAS_COFF /* windows object format */
#else
#define GAS_ELF
#endif

#endif /* ! GAS_COFF && ! GAS_ELF */

#if defined( GAS_COFF )
```

```
/* coff externals have underscores */
#define inflate_fast _inflate_fast
#define inflate_fast_use_mmx _inflate_fast_use_mmx

#endif /* GAS_COFF */

.file "inffast.S"

.globl inflate_fast

.text
.align 4,0
.L_invalid_literal_length_code_msg:
.string "invalid literal/length code"

.align 4,0
.L_invalid_distance_code_msg:
.string "invalid distance code"

.align 4,0
.L_invalid_distance_too_far_msg:
.string "invalid distance too far back"

#if ! defined( NO_MMX )
.align 4,0
.L_mask: /* mask[N] = ( 1 << N ) - 1 */
.long 0
.long 1
.long 3
.long 7
.long 15
.long 31
.long 63
.long 127
.long 255
.long 511
.long 1023
.long 2047
.long 4095
.long 8191
.long 16383
.long 32767
.long 65535
.long 131071
.long 262143
.long 524287
.long 1048575
.long 2097151
.long 4194303
.long 8388607
.long 16777215
.long 33554431
.long 67108863
.long 134217727
.long 268435455
.long 536870911
.long 1073741823
.long 2147483647
.long 4294967295
#endif /* NO_MMX */

.text

/*
 * struct z_stream offsets, in zlib.h
 */
#define next_in_strm 0 /* strm->next_in */
#define avail_in_strm 4 /* strm->avail_in */
#define next_out_strm 12 /* strm->next_out */
#define avail_out_strm 16 /* strm->avail_out */
#define msg_strm 24 /* strm->msg */
#define state_strm 28 /* strm->state */
```

```

/*
 * struct inflate_state offsets, in inflate.h
 */
#define mode_state      0    /* state->mode */
#define wsize_state     32   /* state->wsize */
#define write_state     40   /* state->write */
#define window_state    44   /* state->window */
#define hold_state      48   /* state->hold */
#define bits_state      52   /* state->bits */
#define lencode_state   68   /* state->lencode */
#define distcode_state  72   /* state->distcode */
#define lenbits_state   76   /* state->lenbits */
#define distbits_state  80   /* state->distbits */

/*
 * inflate_fast's activation record
 */
#define local_var_size  64 /* how much local space for vars */
#define strm_sp         88 /* first arg: z_stream * (local_var_size + 24) */
#define start_sp        92 /* second arg: unsigned int (local_var_size + 28) */

/*
 * offsets for local vars on stack
 */
#define out              60 /* unsigned char* */
#define window          56 /* unsigned char* */
#define wsize           52 /* unsigned int */
#define write           48 /* unsigned int */
#define in              44 /* unsigned char* */
#define beg             40 /* unsigned char* */
#define buf             28 /* char[ 12 ] */
#define len             24 /* unsigned int */
#define last            20 /* unsigned char* */
#define end             16 /* unsigned char* */
#define dcode           12 /* code* */
#define lcode           8  /* code* */
#define dmask           4  /* unsigned int */
#define lmask           0  /* unsigned int */

/*
 * typedef enum inflate_mode consts, in inflate.h
 */
#define INFLATE_MODE_TYPE 11 /* state->mode flags enum-ed in inflate.h */
#define INFLATE_MODE_BAD 26

#if ! defined( USE_MMX ) && ! defined( NO_MMX )

#define RUN_TIME_MMX

#define CHECK_MMX      1
#define DO_USE_MMX     2
#define DONT_USE_MMX  3

.globl inflate_fast_use_mmx

.data

.align 4,0
inflate_fast_use_mmx: /* integer flag for run time control 1=check,2=mmx,3=no */
.long CHECK_MMX

#if defined( GAS_ELF )
/* elf info */
.type    inflate_fast_use_mmx,@object
.size    inflate_fast_use_mmx,4
#endif

#endif /* RUN_TIME_MMX */

#if defined( GAS_COFF )
/* coff info: scl 2 = extern, type 32 = function */
.def inflate_fast; .scl 2; .type 32; .endef
#endif

```

```
.text

.align 32,0x90
inflate_fast:
    pushl    %edi
    pushl    %esi
    pushl    %ebp
    pushl    %ebx
    pushf    /* save eflags (strm_sp, state_sp assumes this is 32 bits) */
    subl     $local_var_size, %esp
    cld

#define strm_r    %esi
#define state_r    %edi

    movl     strm_sp(%esp), strm_r
    movl     state_strm(strm_r), state_r

    /* in = strm->next_in;
     * out = strm->next_out;
     * last = in + strm->avail_in - 11;
     * beg = out - (start - strm->avail_out);
     * end = out + (strm->avail_out - 257);
     */
    movl     avail_in_strm(strm_r), %edx
    movl     next_in_strm(strm_r), %eax

    addl     %eax, %edx      /* avail_in += next_in */
    subl     $11, %edx      /* avail_in -= 11 */

    movl     %eax, in(%esp)
    movl     %edx, last(%esp)

    movl     start_sp(%esp), %ebp
    movl     avail_out_strm(strm_r), %ecx
    movl     next_out_strm(strm_r), %ebx

    subl     %ecx, %ebp      /* start -= avail_out */
    negl     %ebp           /* start = -start */
    addl     %ebx, %ebp      /* start += next_out */

    subl     $257, %ecx      /* avail_out -= 257 */
    addl     %ebx, %ecx      /* avail_out += out */

    movl     %ebx, out(%esp)
    movl     %ebp, beg(%esp)
    movl     %ecx, end(%esp)

    /* wsize = state->wsize;
     * write = state->write;
     * window = state->window;
     * hold = state->hold;
     * bits = state->bits;
     * lcode = state->lencode;
     * dcode = state->distcode;
     * lmask = ( 1 << state->lenbits ) - 1;
     * dmask = ( 1 << state->distbits ) - 1;
     */

    movl     lencode_state(state_r), %eax
    movl     distcode_state(state_r), %ecx

    movl     %eax, lcode(%esp)
    movl     %ecx, dcode(%esp)

    movl     $1, %eax
    movl     lenbits_state(state_r), %ecx
    shll     %cl, %eax
    decl     %eax
    movl     %eax, lmask(%esp)

    movl     $1, %eax
    movl     distbits_state(state_r), %ecx
```



```

    shll    %cl, %eax
    decl    %eax
    movl    %eax, dmask(%esp)

    movl    wsize_state(state_r), %eax
    movl    write_state(state_r), %ecx
    movl    window_state(state_r), %edx

    movl    %eax, wsize(%esp)
    movl    %ecx, write(%esp)
    movl    %edx, window(%esp)

    movl    hold_state(state_r), %ebp
    movl    bits_state(state_r), %ebx

#undef strm_r
#undef state_r

#define in_r      %esi
#define from_r    %esi
#define out_r     %edi

    movl    in(%esp), in_r
    movl    last(%esp), %ecx
    cmpl    in_r, %ecx
    ja      .L_align_long          /* if in < last */

    addl    $11, %ecx              /* ecx = &in[ avail_in ] */
    subl    in_r, %ecx            /* ecx = avail_in */
    movl    $12, %eax
    subl    %ecx, %eax            /* eax = 12 - avail_in */
    leal    buf(%esp), %edi
    rep     movsb                  /* memcpy( buf, in, avail_in ) */
    movl    %eax, %ecx
    xorl    %eax, %eax
    rep     stosb                  /* memset( &buf[ avail_in ], 0, 12 - avail_in ) */
    leal    buf(%esp), in_r       /* in = buf */
    movl    in_r, last(%esp)      /* last = in, do just one iteration */
    jmp     .L_is_aligned

    /* align in_r on long boundary */
.L_align_long:
    testl    $3, in_r
    jz       .L_is_aligned
    xorl     %eax, %eax
    movb     (in_r), %al
    incl     in_r
    movl     %ebx, %ecx
    addl     $8, %ebx
    shll     %cl, %eax
    orl      %eax, %ebp
    jmp      .L_align_long

.L_is_aligned:
    movl     out(%esp), out_r

#if defined( NO_MMX )
    jmp      .L_do_loop
#endif

#if defined( USE_MMX )
    jmp      .L_init_mmx
#endif

/**** Runtime MMX check ****/

#if defined( RUN_TIME_MMX )
.L_check_mmx:
    cmpl     $DO_USE_MMX, inflate_fast_use_mmx
    je       .L_init_mmx
    ja       .L_do_loop /* > 2 */

    pushl    %eax
    pushl    %ebx

```

```

    pushl    %ecx
    pushl    %edx
    pushf
    movl     (%esp), %eax    /* copy eflags to eax */
    xorl     $0x200000, (%esp) /* try toggling ID bit of eflags (bit 21)
                                * to see if cpu supports cpuid...
                                * ID bit method not supported by NexGen but
                                * bios may load a cpuid instruction and
                                * cpuid may be disabled on Cyrix 5-6x86 */

    popf
    pushf
    popl     %edx            /* copy new eflags to edx */
    xorl     %eax, %edx      /* test if ID bit is flipped */
    jz       .L_dont_use_mmx /* not flipped if zero */
    xorl     %eax, %eax
    cpuid
    cmpl     $0x756e6547, %ebx /* check for GenuineIntel in ebx,ecx,edx */
    jne      .L_dont_use_mmx
    cmpl     $0x6c65746e, %ecx
    jne      .L_dont_use_mmx
    cmpl     $0x49656e69, %edx
    jne      .L_dont_use_mmx
    movl     $1, %eax
    cpuid                    /* get cpu features */
    shrl     $8, %eax
    andl     $15, %eax
    cmpl     $6, %eax        /* check for Pentium family, is 0xf for P4 */
    jne      .L_dont_use_mmx
    testl    $0x800000, %edx /* test if MMX feature is set (bit 23) */
    jnz      .L_use_mmx
    jmp      .L_dont_use_mmx
.L_use_mmx:
    movl     $DO_USE_MMX, inflate_fast_use_mmx
    jmp      .L_check_mmx_pop
.L_dont_use_mmx:
    movl     $DONT_USE_MMX, inflate_fast_use_mmx
.L_check_mmx_pop:
    popl     %edx
    popl     %ecx
    popl     %ebx
    popl     %eax
    jmp      .L_check_mmx
#endif

/**** Non-MMX code ****/

#if defined ( NO_MMX ) || defined( RUN_TIME_MMX )

#define hold_r    %ebp
#define bits_r    %bl
#define bitslong_r %ebx

.align 32,0x90
.L_while_test:
    /* while (in < last && out < end)
    */
    cmpl     out_r, end(%esp)
    jbe      .L_break_loop    /* if (out ≥ end) */

    cmpl     in_r, last(%esp)
    jbe      .L_break_loop

.L_do_loop:
    /* regs: %esi = in, %ebp = hold, %bl = bits, %edi = out
    *
    * do {
    *   if (bits < 15) {
    *     hold |= *((unsigned short *)in)++ << bits;
    *     bits += 16
    *   }
    *   this = lcode[hold & lmask]
    */
    cmpb     $15, bits_r

```

```

ja      .L_get_length_code      /* if (15 < bits) */

xorl    %eax, %eax
lodsw
movb    bits_r, %cl             /* al = *(ushort *)in++ */
addb    $16, bits_r             /* cl = bits, needs it for shifting */
shll    %cl, %eax              /* bits += 16 */
orl     %eax, hold_r            /* hold |= *((ushort *)in)++ << bits */

.L_get_length_code:
movl    lmask(%esp), %edx        /* edx = lmask */
movl    lcode(%esp), %ecx        /* ecx = lcode */
andl    hold_r, %edx            /* edx &= hold */
movl    (%ecx,%edx,4), %eax      /* eax = lcode[hold & lmask] */

.L_dolen:
/* regs: %esi = in, %ebp = hold, %bl = bits, %edi = out
 *
 * dolen:
 *   bits -= this.bits;
 *   hold >=> this.bits
 */
movb    %ah, %cl                /* cl = this.bits */
subb    %ah, bits_r             /* bits -= this.bits */
shrl    %cl, hold_r            /* hold >=> this.bits */

/* check if op is a literal
 * if (op == 0) {
 *   PUP(out) = this.val;
 * }
 */
testb   %al, %al
jnz     .L_test_for_length_base /* if (op != 0) 45.7% */

shrl    $16, %eax               /* output this.val char */
stosb
jmp     .L_while_test

.L_test_for_length_base:
/* regs: %esi = in, %ebp = hold, %bl = bits, %edi = out, %edx = len
 *
 * else if (op & 16) {
 *   len = this.val
 *   op &= 15
 *   if (op) {
 *     if (op > bits) {
 *       hold |= *((unsigned short *)in)++ << bits;
 *       bits += 16
 *     }
 *     len += hold & mask[op];
 *     bits -= op;
 *     hold >=> op;
 *   }
 */
#define len_r %edx
movl    %eax, len_r             /* len = this */
shrl    $16, len_r              /* len = this.val */
movb    %al, %cl

testb   $16, %al
jz      .L_test_for_second_level_length /* if ((op & 16) == 0) 8% */
andb    $15, %cl                /* op &= 15 */
jz      .L_save_len             /* if (!op) */
cmpb    %cl, bits_r
jae     .L_add_bits_to_len      /* if (op <= bits) */

movb    %cl, %ch                /* stash op in ch, freeing cl */
xorl    %eax, %eax
lodsw
movb    bits_r, %cl             /* al = *(ushort *)in++ */
addb    $16, bits_r             /* cl = bits, needs it for shifting */
shll    %cl, %eax              /* bits += 16 */
orl     %eax, hold_r            /* hold |= *((ushort *)in)++ << bits */
movb    %ch, %cl                /* move op back to ecx */

```

```

.L_add_bits_to_len:
    movl    $1, %eax
    shll    %cl, %eax
    decl    %eax
    subb    %cl, bits_r
    andl    hold_r, %eax          /* eax &= hold */
    shrll   %cl, hold_r
    addl    %eax, len_r          /* len += hold & mask[op] */

.L_save_len:
    movl    len_r, len(%esp)     /* save len */
#undef     len_r

.L_decode_distance:
    /* regs: %esi = in, %ebp = hold, %bl = bits, %edi = out, %edx = dist
    *
    *   if (bits < 15) {
    *       hold |= *((unsigned short *)in)++ << bits;
    *       bits += 16
    *   }
    *   this = dcode[hold & dmask];
    * dodist:
    *   bits -= this.bits;
    *   hold >>= this.bits;
    *   op = this.op;
    */

    cmpb    $15, bits_r
    ja      .L_get_distance_code /* if (15 < bits) */

    xorl     %eax, %eax
    lodsw
    movb     bits_r, %cl          /* al = *(ushort *)in++ */
    addb     $16, bits_r          /* cl = bits, needs it for shifting */
    shll     %cl, %eax            /* bits += 16 */
    orl      %eax, hold_r         /* hold |= *((ushort *)in)++ << bits */

.L_get_distance_code:
    movl     dmask(%esp), %edx    /* edx = dmask */
    movl     dcode(%esp), %ecx    /* ecx = dcode */
    andl     hold_r, %edx         /* edx &= hold */
    movl     (%ecx,%edx,4), %eax   /* eax = dcode[hold & dmask] */

#define dist_r %edx
.L_dodist:
    movl     %eax, dist_r         /* dist = this */
    shrll    $16, dist_r          /* dist = this.val */
    movb     %ah, %cl
    subb     %ah, bits_r          /* bits -= this.bits */
    shrll    %cl, hold_r          /* hold >>= this.bits */

    /* if (op & 16) {
    *   dist = this.val
    *   op &= 15
    *   if (op > bits) {
    *       hold |= *((unsigned short *)in)++ << bits;
    *       bits += 16
    *   }
    *   dist += hold & mask[op];
    *   bits -= op;
    *   hold >>= op;
    */

    movb     %al, %cl             /* cl = this.op */

    testb    $16, %al             /* if ((op & 16) == 0) */
    jz       .L_test_for_second_level_dist
    andb     $15, %cl             /* op &= 15 */
    jz       .L_check_dist_one
    cmpb     %cl, bits_r
    jae      .L_add_bits_to_dist  /* if (op <= bits) 97.6% */

    movb     %cl, %ch             /* stash op in ch, freeing cl */
    xorl     %eax, %eax

```

```

    lodsw                                /* al = *(ushort *)in++ */
    movb  bits_r, %cl                    /* cl = bits, needs it for shifting */
    addb  $16, bits_r                    /* bits += 16 */
    shll  %cl, %eax
    orl   %eax, hold_r                    /* hold |= *((ushort *)in)++ << bits */
    movb  %ch, %cl                        /* move op back to ecx */

.L_add_bits_to_dist:
    movl  $1, %eax
    shll  %cl, %eax
    decl  %eax                            /* (1 << op) - 1 */
    subb  %cl, bits_r
    andl  hold_r, %eax                    /* eax &= hold */
    shrll %cl, hold_r
    addl  %eax, dist_r                    /* dist += hold & ((1 << op) - 1) */
    jmp   .L_check_window

.L_check_window:
/* regs: %esi = from, %ebp = hold, %bl = bits, %edi = out, %edx = dist
 *      %ecx = nbytes
 *
 * nbytes = out - beg;
 * if (dist <= nbytes) {
 *   from = out - dist;
 *   do {
 *     PUP(out) = PUP(from);
 *   } while (--len > 0) {
 * }
 */

    movl  in_r, in(%esp)                  /* save in so from can use it's reg */
    movl  out_r, %eax
    subl  beg(%esp), %eax                  /* nbytes = out - beg */

    cmpl  dist_r, %eax
    jnb   .L_clip_window                  /* if (dist > nbytes) 4.2% */

    movl  len(%esp), %ecx
    movl  out_r, from_r
    subl  dist_r, from_r                  /* from = out - dist */

    subl  $3, %ecx
    movb  (from_r), %al
    movb  %al, (out_r)
    movb  1(from_r), %al
    movb  2(from_r), %dl
    addl  $3, from_r
    movb  %al, 1(out_r)
    movb  %dl, 2(out_r)
    addl  $3, out_r
    rep   movsb

    movl  in(%esp), in_r                  /* move in back to %esi, toss from */
    jmp   .L_while_test

.align 16,0x90
.L_check_dist_one:
    cmpl  $1, dist_r
    jne   .L_check_window
    cmpl  out_r, beg(%esp)
    je    .L_check_window

    decl  out_r
    movl  len(%esp), %ecx
    movb  (out_r), %al
    subl  $3, %ecx

    movb  %al, 1(out_r)
    movb  %al, 2(out_r)
    movb  %al, 3(out_r)
    addl  $4, out_r
    rep   stosb

    jmp   .L_while_test

```

```

.align 16,0x90
.L_test_for_second_level_length:
    /* else if ((op & 64) == 0) {
    *   this = lcode[this.val + (hold & mask[op])];
    * }
    */
    testb    $64, %al
    jnz      .L_test_for_end_of_block /* if ((op & 64) != 0) */

    movl     $1, %eax
    shll     %cl, %eax
    decl     %eax
    andl     hold_r, %eax             /* eax &= hold */
    addl     %edx, %eax              /* eax += this.val */
    movl     lcode(%esp), %edx       /* edx = lcode */
    movl     (%edx,%eax,4), %eax     /* eax = lcode[val + (hold&mask[op])] */
    jmp      .L_dolen

.align 16,0x90
.L_test_for_second_level_dist:
    /* else if ((op & 64) == 0) {
    *   this = dcode[this.val + (hold & mask[op])];
    * }
    */
    testb    $64, %al
    jnz      .L_invalid_distance_code /* if ((op & 64) != 0) */

    movl     $1, %eax
    shll     %cl, %eax
    decl     %eax
    andl     hold_r, %eax             /* eax &= hold */
    addl     %edx, %eax              /* eax += this.val */
    movl     dcode(%esp), %edx       /* edx = dcode */
    movl     (%edx,%eax,4), %eax     /* eax = dcode[val + (hold&mask[op])] */
    jmp      .L_dodist

.align 16,0x90
.L_clip_window:
    /* regs: %esi = from, %ebp = hold, %bl = bits, %edi = out, %edx = dist
    *   %ecx = nbytes
    *
    * else {
    *   if (dist > wsize) {
    *     invalid distance
    *   }
    *   from = window;
    *   nbytes = dist - nbytes;
    *   if (write == 0) {
    *     from += wsize - nbytes;
    *   }
    */
#define nbytes_r %ecx
    movl     %eax, nbytes_r
    movl     wsize(%esp), %eax        /* prepare for dist compare */
    negl     nbytes_r                /* nbytes = -nbytes */
    movl     window(%esp), from_r    /* from = window */

    cmpl     dist_r, %eax
    jb       .L_invalid_distance_too_far /* if (dist > wsize) */

    addl     dist_r, nbytes_r         /* nbytes = dist - nbytes */
    cmpl     $0, write(%esp)
    jne      .L_wrap_around_window  /* if (write != 0) */

    subl     nbytes_r, %eax
    addl     %eax, from_r            /* from += wsize - nbytes */

    /* regs: %esi = from, %ebp = hold, %bl = bits, %edi = out, %edx = dist
    *   %ecx = nbytes, %eax = len
    *
    * if (nbytes < len) {
    *   len -= nbytes;
    *   do {
    *     PUP(out) = PUP(from);

```

```

*      } while (--nbytes);
*      from = out - dist;
*  }
*/
#define len_r %eax
movl    len(%esp), len_r
cmpl    nbytes_r, len_r
jbe     .L_do_copy1      /* if (nbytes ≥ len) */

subl    nbytes_r, len_r  /* len -= nbytes */
rep     movsb
movl    out_r, from_r
subl    dist_r, from_r   /* from = out - dist */
jmp     .L_do_copy1

cmpl    nbytes_r, len_r
jbe     .L_do_copy1      /* if (nbytes ≥ len) */

subl    nbytes_r, len_r  /* len -= nbytes */
rep     movsb
movl    out_r, from_r
subl    dist_r, from_r   /* from = out - dist */
jmp     .L_do_copy1

.L_wrap_around_window:
/* regs: %esi = from, %ebp = hold, %bl = bits, %edi = out, %edx = dist
*      %ecx = nbytes, %eax = write, %eax = len
*
*      else if (write < nbytes) {
*          from += wsize + write - nbytes;
*          nbytes -= write;
*          if (nbytes < len) {
*              len -= nbytes;
*              do {
*                  PUP(out) = PUP(from);
*              } while (--nbytes);
*              from = window;
*              nbytes = write;
*              if (nbytes < len) {
*                  len -= nbytes;
*                  do {
*                      PUP(out) = PUP(from);
*                  } while(--nbytes);
*                  from = out - dist;
*              }
*          }
*      }
*/
#define write_r %eax
movl    write(%esp), write_r
cmpl    write_r, nbytes_r
jbe     .L_contiguous_in_window /* if (write ≥ nbytes) */

addl    wsize(%esp), from_r
addl    write_r, from_r
subl    nbytes_r, from_r      /* from += wsize + write - nbytes */
subl    write_r, nbytes_r     /* nbytes -= write */
#undef write_r

movl    len(%esp), len_r
cmpl    nbytes_r, len_r
jbe     .L_do_copy1      /* if (nbytes ≥ len) */

subl    nbytes_r, len_r    /* len -= nbytes */
rep     movsb
movl    window(%esp), from_r /* from = window */
movl    write(%esp), nbytes_r /* nbytes = write */
cmpl    nbytes_r, len_r
jbe     .L_do_copy1      /* if (nbytes ≥ len) */

subl    nbytes_r, len_r    /* len -= nbytes */
rep     movsb
movl    out_r, from_r

```

```

    subl    dist_r, from_r          /* from = out - dist */
    jmp     .L_do_cpy1

.L_contiguous_in_window:
    /* regs: %esi = from, %ebp = hold, %bl = bits, %edi = out, %edx = dist
    *        %ecx = nbytes, %eax = write, %eax = len
    *
    *      else {
    *          from += write - nbytes;
    *          if (nbytes < len) {
    *              len -= nbytes;
    *              do {
    *                  PUP(out) = PUP(from);
    *              } while (--nbytes);
    *              from = out - dist;
    *          }
    *      }
    */
#define write_r %eax
    addl    write_r, from_r
    subl    nbytes_r, from_r        /* from += write - nbytes */
#undef write_r

    movl    len(%esp), len_r
    cmpl    nbytes_r, len_r
    jbe     .L_do_cpy1             /* if (nbytes ≥ len) */

    subl    nbytes_r, len_r        /* len -= nbytes */
    rep     movsb
    movl    out_r, from_r
    subl    dist_r, from_r        /* from = out - dist */

.L_do_cpy1:
    /* regs: %esi = from, %esi = in, %ebp = hold, %bl = bits, %edi = out
    *        %eax = len
    *
    *      while (len > 0) {
    *          PUP(out) = PUP(from);
    *          len--;
    *      }
    * } while (in < last && out < end);
    */
#undef nbytes_r
#define in_r %esi
    movl    len_r, %ecx
    rep     movsb

    movl    in(%esp), in_r        /* move in back to %esi, toss from */
    jmp     .L_while_test

#undef len_r
#undef dist_r

#endif /* NO_MMX || RUN_TIME_MMX */

/**** MMX code ****/

#if defined( USE_MMX ) || defined( RUN_TIME_MMX )

.align 32,0x90
.L_init_mmx:
    emms

#undef bits_r
#undef bitslong_r
#define bitslong_r %ebp
#define hold_mm %mm0
    movd    %ebp, hold_mm
    movl    %ebx, bitslong_r

#define used_mm %mm1
#define dmask2_mm %mm2

```



```

#define lmask2_mm %mm3
#define lmask_mm %mm4
#define dmask_mm %mm5
#define tmp_mm %mm6

    movd    lmask(%esp), lmask_mm
    movq    lmask_mm, lmask2_mm
    movd    dmask(%esp), dmask_mm
    movq    dmask_mm, dmask2_mm
    pxor    used_mm, used_mm
    movl    lcode(%esp), %ebx    /* ebx = lcode */
    jmp     .L_do_loop_mmx

.align 32,0x90
.L_while_test_mmx:
    /* while (in < last && out < end)
    */
    cmpl    out_r, end(%esp)
    jbe     .L_break_loop    /* if (out ≥ end) */

    cmpl    in_r, last(%esp)
    jbe     .L_break_loop

.L_do_loop_mmx:
    psrlq   used_mm, hold_mm    /* hold_mm >= last bit length */

    cmpl    $32, bitslong_r
    ja      .L_get_length_code_mmx /* if (32 < bits) */

    movd    bitslong_r, tmp_mm
    movd    (in_r), %mm7
    addl    $4, in_r
    psllq   tmp_mm, %mm7
    addl    $32, bitslong_r
    por     %mm7, hold_mm    /* hold_mm |= *((uint *)in)++ << bits */

.L_get_length_code_mmx:
    pand    hold_mm, lmask_mm
    movd    lmask_mm, %eax
    movq    lmask2_mm, lmask_mm
    movl    (%ebx,%eax,4), %eax    /* eax = lcode[hold & lmask] */

.L_dolen_mmx:
    movzbl  %ah, %ecx    /* ecx = this.bits */
    movd    %ecx, used_mm
    subl    %ecx, bitslong_r    /* bits -= this.bits */

    testb   %al, %al
    jnz     .L_test_for_length_base_mmx /* if (op ≠ 0) 45.7% */

    shrl    $16, %eax    /* output this.val char */
    stosb
    jmp     .L_while_test_mmx

.L_test_for_length_base_mmx:
#define len_r %edx
    movl    %eax, len_r    /* len = this */
    shrl    $16, len_r    /* len = this.val */

    testb   $16, %al
    jz      .L_test_for_second_level_length_mmx /* if ((op & 16) ≡ 0) 8% */
    andl    $15, %eax    /* op &= 15 */
    jz      .L_decode_distance_mmx /* if (!op) */

    psrlq   used_mm, hold_mm    /* hold_mm >= last bit length */
    movd    %eax, used_mm
    movd    hold_mm, %ecx
    subl    %eax, bitslong_r
    andl    .L_mask(,%eax,4), %ecx
    addl    %ecx, len_r    /* len += hold & mask[op] */

.L_decode_distance_mmx:
    psrlq   used_mm, hold_mm    /* hold_mm >= last bit length */

```

```

    cmpl    $32, bitslong_r
    ja      .L_get_dist_code_mmx    /* if (32 < bits) */

    movd    bitslong_r, tmp_mm
    movd    (in_r), %mm7
    addl    $4, in_r
    psllq   tmp_mm, %mm7
    addl    $32, bitslong_r
    por     %mm7, hold_mm           /* hold_mm |= *((uint *)in)++ << bits */

.L_get_dist_code_mmx:
    movl    dcode(%esp), %ebx       /* ebx = dcode */
    pand    hold_mm, dmask_mm
    movd    dmask_mm, %eax
    movq    dmask2_mm, dmask_mm
    movl    (%ebx,%eax,4), %eax     /* eax = dcode[hold & lmask] */

.L_dodist_mmx:
#define dist_r %ebx
    movzbl  %ah, %ecx               /* ecx = this.bits */
    movl    %eax, dist_r
    shr     $16, dist_r             /* dist = this.val */
    subl    %ecx, bitslong_r       /* bits -= this.bits */
    movd    %ecx, used_mm

    testb   $16, %al               /* if ((op & 16) == 0) */
    jz      .L_test_for_second_level_dist_mmx
    andl    $15, %eax             /* op &= 15 */
    jz      .L_check_dist_one_mmx

.L_add_bits_to_dist_mmx:
    psrlq   used_mm, hold_mm       /* hold_mm >>= last bit length */
    movd    %eax, used_mm          /* save bit length of current op */
    movd    hold_mm, %ecx          /* get the next bits on input stream */
    subl    %eax, bitslong_r       /* bits -= op bits */
    andl    .L_mask(,%eax,4), %ecx /* ecx = hold & mask[op] */
    addl    %ecx, dist_r           /* dist += hold & mask[op] */

.L_check_window_mmx:
    movl    in_r, in(%esp)         /* save in so from can use it's reg */
    movl    out_r, %eax
    subl    beg(%esp), %eax        /* nbytes = out - beg */

    cmpl    dist_r, %eax           /* if (dist > nbytes) 4.2% */
    jb      .L_clip_window_mmx

    movl    len_r, %ecx
    movl    out_r, from_r
    subl    dist_r, from_r         /* from = out - dist */

    subl    $3, %ecx
    movb    (from_r), %al
    movb    %al, (out_r)
    movb    1(from_r), %al
    movb    2(from_r), %dl
    addl    $3, from_r
    movb    %al, 1(out_r)
    movb    %dl, 2(out_r)
    addl    $3, out_r
    rep     movsb

    movl    in(%esp), in_r         /* move in back to %esi, toss from */
    movl    lcode(%esp), %ebx      /* move lcode back to %ebx, toss dist */
    jmp     .L_while_test_mmx

.align 16,0x90
.L_check_dist_one_mmx:
    cmpl    $1, dist_r
    jne     .L_check_window_mmx
    cmpl    out_r, beg(%esp)
    je      .L_check_window_mmx

    decl    out_r
    movl    len_r, %ecx

```

```

    movb    (out_r), %al
    subl    $3, %ecx

    movb    %al, 1(out_r)
    movb    %al, 2(out_r)
    movb    %al, 3(out_r)
    addl    $4, out_r
    rep     stosb

    movl    lcode(%esp), %ebx      /* move lcode back to %ebx, toss dist */
    jmp     .L_while_test_mmx

.align 16,0x90
.L_test_for_second_level_length_mmx:
    testb    $64, %al
    jnz     .L_test_for_end_of_block /* if ((op & 64) != 0) */

    andl     $15, %eax
    psrlq    used_mm, hold_mm      /* hold_mm >= last bit length */
    movd     hold_mm, %ecx
    andl     .L_mask(,%eax,4), %ecx
    addl     len_r, %ecx
    movl     (%ebx,%ecx,4), %eax    /* eax = lcode[hold & lmask] */
    jmp     .L_dolen_mmx

.align 16,0x90
.L_test_for_second_level_dist_mmx:
    testb    $64, %al
    jnz     .L_invalid_distance_code /* if ((op & 64) != 0) */

    andl     $15, %eax
    psrlq    used_mm, hold_mm      /* hold_mm >= last bit length */
    movd     hold_mm, %ecx
    andl     .L_mask(,%eax,4), %ecx
    movl     dcode(%esp), %eax     /* ecx = dcode */
    addl     dist_r, %ecx
    movl     (%eax,%ecx,4), %eax    /* eax = lcode[hold & lmask] */
    jmp     .L_dodist_mmx

.align 16,0x90
.L_clip_window_mmx:
#define nbytes_r %ecx
    movl     %eax, nbytes_r
    movl     wsize(%esp), %eax     /* prepare for dist compare */
    negl     nbytes_r              /* nbytes = -nbytes */
    movl     window(%esp), from_r /* from = window */

    cmpl     dist_r, %eax
    jb       .L_invalid_distance_too_far /* if (dist > wsize) */

    addl     dist_r, nbytes_r      /* nbytes = dist - nbytes */
    cmpl     $0, write(%esp)
    jne      .L_wrap_around_window_mmx /* if (write != 0) */

    subl     nbytes_r, %eax
    addl     %eax, from_r          /* from += wsize - nbytes */

    cmpl     nbytes_r, len_r
    jbe      .L_do_cpy1_mmx       /* if (nbytes >= len) */

    subl     nbytes_r, len_r      /* len -= nbytes */
    rep     movsb
    movl     out_r, from_r
    subl     dist_r, from_r        /* from = out - dist */
    jmp     .L_do_cpy1_mmx

    cmpl     nbytes_r, len_r
    jbe      .L_do_cpy1_mmx       /* if (nbytes >= len) */

    subl     nbytes_r, len_r      /* len -= nbytes */
    rep     movsb
    movl     out_r, from_r
    subl     dist_r, from_r        /* from = out - dist */
    jmp     .L_do_cpy1_mmx

```

```

.L_wrap_around_window_mmx:
#define write_r %eax
    movl    write(%esp), write_r
    cmpl    write_r, nbytes_r
    jbe     .L_contiguous_in_window_mmx /* if (write ≥ nbytes) */

    addl    wsize(%esp), from_r
    addl    write_r, from_r
    subl    nbytes_r, from_r          /* from += wsize + write - nbytes */
    subl    write_r, nbytes_r        /* nbytes -= write */
#undef write_r

    cmpl    nbytes_r, len_r
    jbe     .L_do_cpyl_mmx          /* if (nbytes ≥ len) */

    subl    nbytes_r, len_r          /* len -= nbytes */
    rep     movsb
    movl    window(%esp), from_r     /* from = window */
    movl    write(%esp), nbytes_r    /* nbytes = write */
    cmpl    nbytes_r, len_r
    jbe     .L_do_cpyl_mmx          /* if (nbytes ≥ len) */

    subl    nbytes_r, len_r          /* len -= nbytes */
    rep     movsb
    movl    out_r, from_r
    subl    dist_r, from_r           /* from = out - dist */
    jmp     .L_do_cpyl_mmx

.L_contiguous_in_window_mmx:
#define write_r %eax
    addl    write_r, from_r
    subl    nbytes_r, from_r          /* from += write - nbytes */
#undef write_r

    cmpl    nbytes_r, len_r
    jbe     .L_do_cpyl_mmx          /* if (nbytes ≥ len) */

    subl    nbytes_r, len_r          /* len -= nbytes */
    rep     movsb
    movl    out_r, from_r
    subl    dist_r, from_r           /* from = out - dist */

.L_do_cpyl_mmx:
#undef nbytes_r
#define in_r %esi
    movl    len_r, %ecx
    rep     movsb

    movl    in(%esp), in_r           /* move in back to %esi, toss from */
    movl    lcode(%esp), %ebx        /* move lcode back to %ebx, toss dist */
    jmp     .L_while_test_mmx

#undef hold_r
#undef bitslong_r

#endif /* USE_MMX || RUN_TIME_MMX */

/** USE_MMX, NO_MMX, and RUNTIME_MMX from here on */

.L_invalid_distance_code:
/* else {
 *   strm→msg = "invalid distance code";
 *   state→mode = BAD;
 * }
 */
    movl    $.L_invalid_distance_code_msg, %ecx
    movl    $INFLATE_MODE_BAD, %edx
    jmp     .L_update_stream_state

.L_test_for_end_of_block:
/* else if (op & 32) {
 *   state→mode = TYPE;

```

```

    *   break;
    * }
    */
testb    $32, %al
jz       .L_invalid_literal_length_code    /* if ((op & 32) == 0) */

movl     $0, %ecx
movl     $INFLATE_MODE_TYPE, %edx
jmp      .L_update_stream_state

.L_invalid_literal_length_code:
/* else {
 *   strm->msg = "invalid literal/length code";
 *   state->mode = BAD;
 * }
 */
movl     $.L_invalid_literal_length_code_msg, %ecx
movl     $INFLATE_MODE_BAD, %edx
jmp      .L_update_stream_state

.L_invalid_distance_too_far:
/* strm->msg = "invalid distance too far back";
 * state->mode = BAD;
 */
movl     in(%esp), in_r                    /* from_r has in's reg, put in back */
movl     $.L_invalid_distance_too_far_msg, %ecx
movl     $INFLATE_MODE_BAD, %edx
jmp      .L_update_stream_state

.L_update_stream_state:
/* set strm->msg = %ecx, strm->state->mode = %edx */
movl     strm_sp(%esp), %eax
testl    %ecx, %ecx                      /* if (msg != NULL) */
jz       .L_skip_msg
movl     %ecx, msg_strm(%eax)             /* strm->msg = msg */
.L_skip_msg:
movl     state_strm(%eax), %eax           /* state = strm->state */
movl     %edx, mode_state(%eax)           /* state->mode = edx (BAD | TYPE) */
jmp      .L_break_loop

.align 32,0x90
.L_break_loop:

/*
 * Regs:
 *
 * bits = %ebp when mmx, and in %ebx when non-mmx
 * hold = %hold_mm when mmx, and in %ebp when non-mmx
 * in   = %esi
 * out  = %edi
 */

#if defined( USE_MMX ) || defined( RUN_TIME_MMX )
#if defined( RUN_TIME_MMX )

    cmpl    $DO_USE_MMX, inflate_fast_use_mmx
    jne     .L_update_next_in

#endif /* RUN_TIME_MMX */

    movl    %ebp, %ebx

.L_update_next_in:
#endif

#define strm_r %eax
#define state_r %edx

/* len = bits >> 3;
 * in -= len;
 * bits -= len << 3;
 * hold &= (1U << bits) - 1;

```

```

    * state→hold = hold;
    * state→bits = bits;
    * strm→next_in = in;
    * strm→next_out = out;
    */
    movl    strm_sp(%esp), strm_r
    movl    %ebx, %ecx
    movl    state_strm(strm_r), state_r
    shr     $3, %ecx
    subl    %ecx, in_r
    shll    $3, %ecx
    subl    %ecx, %ebx
    movl    out_r, next_out_strm(strm_r)
    movl    %ebx, bits_state(state_r)
    movl    %ebx, %ecx

    leal    buf(%esp), %ebx
    cmpl    %ebx, last(%esp)
    jne     .L_buf_not_used        /* if buf ≠ last */

    subl    %ebx, in_r             /* in -= buf */
    movl    next_in_strm(strm_r), %ebx
    movl    %ebx, last(%esp)      /* last = strm→next_in */
    addl    %ebx, in_r            /* in += strm→next_in */
    movl    avail_in_strm(strm_r), %ebx
    subl    $11, %ebx
    addl    %ebx, last(%esp)      /* last = &strm→next_in[ avail_in - 11 ] */

.L_buf_not_used:
    movl    in_r, next_in_strm(strm_r)

    movl    $1, %ebx
    shll    %cl, %ebx
    decl    %ebx

#if defined( USE_MMX ) || defined( RUN_TIME_MMX )
#if defined( RUN_TIME_MMX )

    cmpl    $DO_USE_MMX, inflate_fast_use_mmx
    jne     .L_update_hold

#endif /* RUN_TIME_MMX */

    psrlq   used_mm, hold_mm      /* hold_mm >= last bit length */
    movd    hold_mm, %ebp

    emms

.L_update_hold:
#endif /* USE_MMX || RUN_TIME_MMX */

    andl    %ebx, %ebp
    movl    %ebp, hold_state(state_r)

#define last_r %ebx

    /* strm→avail_in = in < last ? 11 + (last - in) : 11 - (in - last) */
    movl    last(%esp), last_r
    cmpl    in_r, last_r
    jbe     .L_last_is_smaller    /* if (in ≥ last) */

    subl    in_r, last_r          /* last -= in */
    addl    $11, last_r          /* last += 11 */
    movl    last_r, avail_in_strm(strm_r)
    jmp     .L_fixup_out
.L_last_is_smaller:
    subl    last_r, in_r          /* in -= last */
    negl    in_r                 /* in = -in */
    addl    $11, in_r            /* in += 11 */
    movl    in_r, avail_in_strm(strm_r)

#undef last_r

```

```
#define end_r %ebx

.L_fixup_out:
/* strm->avail_out = out < end ? 257 + (end - out) : 257 - (out - end) */
movl    end(%esp), end_r
cmpl    out_r, end_r
jbe     .L_end_is_smaller    /* if (out ≥ end) */

    subl    out_r, end_r      /* end -= out */
    addl    $257, end_r      /* end += 257 */
    movl    end_r, avail_out_strm(strm_r)
    jmp     .L_done

.L_end_is_smaller:
    subl    end_r, out_r      /* out -= end */
    negl    out_r            /* out = -out */
    addl    $257, out_r      /* out += 257 */
    movl    out_r, avail_out_strm(strm_r)

#undef end_r
#undef strm_r
#undef state_r

.L_done:
    addl    $local_var_size, %esp
    popf
    popl    %ebx
    popl    %ebp
    popl    %esi
    popl    %edi
    ret

#if defined( GAS_ELF )
/* elf info */
.type inflate_fast,@function
.size inflate_fast,.-inflate_fast
#endif
```

```
#include "zfstream.h"

int main() {

    // Construct a stream object with this filebuffer.  Anything sent
    // to this stream will go to standard out.
    gzofstream os( 1, ios::out );

    // This text is getting compressed and sent to stdout.
    // To prove this, run 'test | zcat'.
    os << "Hello, Mommy" << endl;

    os << setcompressionlevel( Z_NO_COMPRESSION );
    os << "hello, hello, hi, ho!" << endl;

    setcompressionlevel( os, Z_DEFAULT_COMPRESSION )
        << "I'm compressing again" << endl;

    os.close();

    return 0;
}
```



```
#include "zfstream.h"

gzfilebuf::gzfilebuf() :
    file(NULL),
    mode(0),
    own_file_descriptor(0)
{ }

gzfilebuf::~gzfilebuf() {
    sync();
    if ( own_file_descriptor )
        close();
}

gzfilebuf *gzfilebuf::open( const char *name,
                             int io_mode ) {

    if ( is_open() )
        return NULL;

    char char_mode[10];
    char *p = char_mode;

    if ( io_mode & ios::in ) {
        mode = ios::in;
        *p++ = 'r';
    } else if ( io_mode & ios::app ) {
        mode = ios::app;
        *p++ = 'a';
    } else {
        mode = ios::out;
        *p++ = 'w';
    }

    if ( io_mode & ios::binary ) {
        mode |= ios::binary;
        *p++ = 'b';
    }

    // Hard code the compression level
    if ( io_mode & (ios::out|ios::app) ) {
        *p++ = '9';
    }

    // Put the end-of-string indicator
    *p = '\0';

    if ( (file = gzopen(name, char_mode)) == NULL )
        return NULL;

    own_file_descriptor = 1;

    return this;
}

gzfilebuf *gzfilebuf::attach( int file_descriptor,
                               int io_mode ) {

    if ( is_open() )
        return NULL;

    char char_mode[10];
    char *p = char_mode;

    if ( io_mode & ios::in ) {
        mode = ios::in;
        *p++ = 'r';
    } else if ( io_mode & ios::app ) {
        mode = ios::app;
        *p++ = 'a';
    }
```

```
    } else {
        mode = ios::out;
        *p++ = 'w';
    }

    if ( io_mode & ios::binary ) {
        mode |= ios::binary;
        *p++ = 'b';
    }

    // Hard code the compression level
    if ( io_mode & (ios::out|ios::app) ) {
        *p++ = '9';
    }

    // Put the end-of-string indicator
    *p = '\0';

    if ( (file = gzopen(file_descriptor, char_mode)) == NULL )
        return NULL;

    own_file_descriptor = 0;

    return this;
}

gzfilebuf *gzfilebuf::close() {

    if ( is_open() ) {

        sync();
        gzclose( file );
        file = NULL;

    }

    return this;
}

int gzfilebuf::setcompressionlevel( int comp_level ) {

    return gzsetparams(file, comp_level, -2);
}

int gzfilebuf::setcompressionstrategy( int comp_strategy ) {

    return gzsetparams(file, -2, comp_strategy);
}

streampos gzfilebuf::seekoff( streamoff off, ios::seek_dir dir, int which ) {

    return streampos(EOF);
}

int gzfilebuf::underflow() {

    // If the file hasn't been opened for reading, error.
    if ( !is_open() || !(mode & ios::in) )
        return EOF;

    // if a buffer doesn't exists, allocate one.
    if ( !base() ) {

        if ( (allocate()) == EOF )
            return EOF;
        setp(0,0);

    } else {
```

```
    if ( in_avail() )
        return (unsigned char) *gptr();

    if ( out_waiting() ) {
        if ( flushbuf() == EOF )
            return EOF;
    }

}

// Attempt to fill the buffer.

int result = fillbuf();
if ( result == EOF ) {
    // disable get area
    setg(0,0,0);
    return EOF;
}

return (unsigned char) *gptr();
}

int gzfilebuf::overflow( int c ) {

    if ( !is_open() || !(mode & ios::out) )
        return EOF;

    if ( !base() ) {
        if ( allocate() == EOF )
            return EOF;
        setg(0,0,0);
    } else {
        if ( in_avail() ) {
            return EOF;
        }
        if ( out_waiting() ) {
            if ( flushbuf() == EOF )
                return EOF;
        }
    }

    int bl = blen();
    setp( base(), base() + bl);

    if ( c != EOF ) {

        *pptr() = c;
        pbump(1);

    }

    return 0;
}

int gzfilebuf::sync() {

    if ( !is_open() )
        return EOF;

    if ( out_waiting() )
        return flushbuf();

    return 0;
}

int gzfilebuf::flushbuf() {

    int n;
    char *q;
```

```
q = pbase();
n = pptr() - q;

if ( gzwrite( file, q, n) < n )
    return EOF;

setp(0,0);

return 0;
}

int gzfilebuf::fillbuf() {

    int required;
    char *p;

    p = base();

    required = blen();

    int t = gzread( file, p, required );

    if ( t <= 0 ) return EOF;

    setg( base(), base(), base()+t);

    return t;
}

gzfilestream_common::gzfilestream_common() :
    ios( gzfilestream_common::rdbuf() )
{ }

gzfilestream_common::~gzfilestream_common()
{ }

void gzfilestream_common::attach( int fd, int io_mode ) {

    if ( !buffer.attach( fd, io_mode) )
        clear( ios::failbit | ios::badbit );
    else
        clear();
}

void gzfilestream_common::open( const char *name, int io_mode ) {

    if ( !buffer.open( name, io_mode ) )
        clear( ios::failbit | ios::badbit );
    else
        clear();
}

void gzfilestream_common::close() {

    if ( !buffer.close() )
        clear( ios::failbit | ios::badbit );
}

gzfilebuf *gzfilestream_common::rdbuf()
{
    return &buffer;
}

gzifstream::gzifstream() :
    ios( gzfilestream_common::rdbuf() )
{
    clear( ios::badbit );
}
```

```
gzifstream::gzifstream( const char *name, int io_mode ) :
    ios( gzfilestream_common::rdbuf() )
{
    gzfilestream_common::open( name, io_mode );
}

gzifstream::gzifstream( int fd, int io_mode ) :
    ios( gzfilestream_common::rdbuf() )
{
    gzfilestream_common::attach( fd, io_mode );
}

gzifstream::~gzifstream() { }

gzofstream::gzofstream() :
    ios( gzfilestream_common::rdbuf() )
{
    clear( ios::badbit );
}

gzofstream::gzofstream( const char *name, int io_mode ) :
    ios( gzfilestream_common::rdbuf() )
{
    gzfilestream_common::open( name, io_mode );
}

gzofstream::gzofstream( int fd, int io_mode ) :
    ios( gzfilestream_common::rdbuf() )
{
    gzfilestream_common::attach( fd, io_mode );
}

gzofstream::~gzofstream() { }
```

```
#ifndef zfstream_h
#define zfstream_h

#include <fstream.h>
#include "zlib.h"

class gzfilebuf : public streambuf {
public:
    gzfilebuf( );
    virtual ~gzfilebuf();

    gzfilebuf *open( const char *name, int io_mode );
    gzfilebuf *attach( int file_descriptor, int io_mode );
    gzfilebuf *close();

    int setcompressionlevel( int comp_level );
    int setcompressionstrategy( int comp_strategy );

    inline int is_open() const { return (file !=NULL); }

    virtual streampos seekoff( streamoff, ios::seek_dir, int );

    virtual int sync();
protected:
    virtual int underflow();
    virtual int overflow( int = EOF );
private:
    gzFile file;
    short mode;
    short own_file_descriptor;

    int flushbuf();
    int fillbuf();
};

class gzfilestream_common : virtual public ios {
    friend class gzifstream;
    friend class gzofstream;
    friend gzofstream &setcompressionlevel( gzofstream &, int );
    friend gzofstream &setcompressionstrategy( gzofstream &, int );
public:
    virtual ~gzfilestream_common();

    void attach( int fd, int io_mode );
    void open( const char *name, int io_mode );
    void close();
protected:
    gzfilestream_common();
private:
    gzfilebuf *rdbuf();

    gzfilebuf buffer;
};

class gzifstream : public gzfilestream_common, public istream {
public:
    gzifstream();
    gzifstream( const char *name, int io_mode = ios::in );
    gzifstream( int fd, int io_mode = ios::in );
```

```
    virtual ~gzifstream();
};

class gzofstream : public gzfilestream_common, public ostream {
public:

    gzofstream();
    gzofstream( const char *name, int io_mode = ios::out );
    gzofstream( int fd, int io_mode = ios::out );

    virtual ~gzofstream();
};

template<class T> class gzomanip {
    friend gzofstream &operator<<(gzofstream &, const gzomanip<T> &);
public:
    gzomanip(gzofstream &(*f)(gzofstream &, T), T v) : func(f), val(v) { }
private:
    gzofstream &(*func)(gzofstream &, T);
    T val;
};

template<class T> gzofstream &operator<<(gzofstream &s, const gzomanip<T> &m)
{
    return (*m.func)(s, m.val);
}

inline gzofstream &setcompressionlevel( gzofstream &s, int l )
{
    (s.rdbuf())->setcompressionlevel(l);
    return s;
}

inline gzofstream &setcompressionstrategy( gzofstream &s, int l )
{
    (s.rdbuf())->setcompressionstrategy(l);
    return s;
}

inline gzomanip<int> setcompressionlevel(int l)
{
    return gzomanip<int>(&setcompressionlevel,l);
}

inline gzomanip<int> setcompressionstrategy(int l)
{
    return gzomanip<int>(&setcompressionstrategy,l);
}

#endif
```

```

/*
 *
 * Copyright (c) 1997
 * Christian Michelsen Research AS
 * Advanced Computing
 * Fantoftvegen 38, 5036 BERGEN, Norway
 * http://www.cmr.no
 *
 * Permission to use, copy, modify, distribute and sell this software
 * and its documentation for any purpose is hereby granted without fee,
 * provided that the above copyright notice appear in all copies and
 * that both that copyright notice and this permission notice appear
 * in supporting documentation. Christian Michelsen Research AS makes no
 * representations about the suitability of this software for any
 * purpose. It is provided "as is" without express or implied warranty.
 */

#ifndef ZSTREAM__H
#define ZSTREAM__H

/*
 * zstream.h - C++ interface to the 'zlib' general purpose compression library
 * $Id: zstream.h,v 1.1 2005/09/23 22:39:03 beng Exp $
 */

#include <strstream.h>
#include <string.h>
#include <stdio.h>
#include "zlib.h"

#if defined(_WIN32)
#   include <fcntl.h>
#   include <io.h>
#   define SET_BINARY_MODE(file) setmode(fileno(file), O_BINARY)
#else
#   define SET_BINARY_MODE(file)
#endif

class zstringlen {
public:
    zstringlen(class izstream&);
    zstringlen(class ozstream&, const char*);
    size_t value() const { return val.word; }
private:
    struct Val { unsigned char byte; size_t word; } val;
};

// ----- izstream -----

class izstream
{
public:
    izstream() : m_fp(0) {}
    izstream(FILE* fp) : m_fp(0) { open(fp); }
    izstream(const char* name) : m_fp(0) { open(name); }
    ~izstream() { close(); }

    /* Opens a gzip (.gz) file for reading.
     * open() can be used to read a file which is not in gzip format;
     * in this case read() will directly read from the file without
     * decompression. errno can be checked to distinguish two error
     * cases (if errno is zero, the zlib error is Z_MEM_ERROR).
     */
    void open(const char* name) {
        if (m_fp) close();
        m_fp = ::gzopen(name, "rb");
    }

    void open(FILE* fp) {
        SET_BINARY_MODE(fp);
        if (m_fp) close();
        m_fp = ::gzdopen(fileno(fp), "rb");
    }
}

```



```

    /* Flushes all pending input if necessary, closes the compressed file
    * and deallocates all the (de)compression state. The return value is
    * the zlib error number (see function error() below).
    */
    int close() {
        int r = ::gzclose(m_fp);
        m_fp = 0; return r;
    }

    /* Binary read the given number of bytes from the compressed file.
    */
    int read(void* buf, size_t len) {
        return ::gzread(m_fp, buf, len);
    }

    /* Returns the error message for the last error which occurred on the
    * given compressed file. errnum is set to zlib error number. If an
    * error occurred in the file system and not in the compression library,
    * errnum is set to Z_ERRNO and the application may consult errno
    * to get the exact error code.
    */
    const char* error(int* errnum) {
        return ::gzerror(m_fp, errnum);
    }

    gzFile fp() { return m_fp; }

private:
    gzFile m_fp;
};

/*
    * Binary read the given (array of) object(s) from the compressed file.
    * If the input file was not in gzip format, read() copies the objects number
    * of bytes into the buffer.
    * returns the number of uncompressed bytes actually read
    * (0 for end of file, -1 for error).
    */
template <class T, class Items>
inline int read(izstream& zs, T* x, Items items) {
    return ::gzread(zs.fp(), x, items*sizeof(T));
}

/*
    * Binary input with the '>' operator.
    */
template <class T>
inline izstream& operator>(izstream& zs, T& x) {
    ::gzread(zs.fp(), &x, sizeof(T));
    return zs;
}

inline zstringlen::zstringlen(izstream& zs) {
    zs > val.byte;
    if (val.byte == 255) zs > val.word;
    else val.word = val.byte;
}

/*
    * Read length of string + the string with the '>' operator.
    */
inline izstream& operator>(izstream& zs, char* x) {
    zstringlen len(zs);
    ::gzread(zs.fp(), x, len.value());
    x[len.value()] = '\0';
    return zs;
}

inline char* read_string(izstream& zs) {
    zstringlen len(zs);
    char* x = new char[len.value()+1];
    ::gzread(zs.fp(), x, len.value());

```

```

    x[len.value()] = '\0';
    return x;
}

// ----- ozstream -----

class ozstream
{
public:
    ozstream() : m_fp(0), m_os(0) {
    }
    ozstream(FILE* fp, int level = Z_DEFAULT_COMPRESSION)
        : m_fp(0), m_os(0) {
        open(fp, level);
    }
    ozstream(const char* name, int level = Z_DEFAULT_COMPRESSION)
        : m_fp(0), m_os(0) {
        open(name, level);
    }
    ~ozstream() {
        close();
    }

    /* Opens a gzip (.gz) file for writing.
     * The compression level parameter should be in 0..9
     * errno can be checked to distinguish two error cases
     * (if errno is zero, the zlib error is Z_MEM_ERROR).
     */
    void open(const char* name, int level = Z_DEFAULT_COMPRESSION) {
        char mode[4] = "wb0";
        if (level != Z_DEFAULT_COMPRESSION) mode[2] = '0'+level;
        if (m_fp) close();
        m_fp = ::gzopen(name, mode);
    }

    /* open from a FILE pointer.
     */
    void open(FILE* fp, int level = Z_DEFAULT_COMPRESSION) {
        SET_BINARY_MODE(fp);
        char mode[4] = "wb0";
        if (level != Z_DEFAULT_COMPRESSION) mode[2] = '0'+level;
        if (m_fp) close();
        m_fp = ::gzdopen(fileno(fp), mode);
    }

    /* Flushes all pending output if necessary, closes the compressed file
     * and deallocates all the (de)compression state. The return value is
     * the zlib error number (see function error() below).
     */
    int close() {
        if (m_os) {
            ::gzwrite(m_fp, m_os->str(), m_os->pcount());
            delete[] m_os->str(); delete m_os; m_os = 0;
        }
        int r = ::gzcclose(m_fp); m_fp = 0; return r;
    }

    /* Binary write the given number of bytes into the compressed file.
     */
    int write(const void* buf, size_t len) {
        return ::gzwrite(m_fp, (voidp) buf, len);
    }

    /* Flushes all pending output into the compressed file. The parameter
     * _flush is as in the deflate() function. The return value is the zlib
     * error number (see function gzerror below). flush() returns Z_OK if
     * the flush_parameter is Z_FINISH and all output could be flushed.
     * flush() should be called only when strictly necessary because it can
     * degrade compression.
     */
    int flush(int _flush) {
        os_flush();
        return ::gzflush(m_fp, _flush);
    }
}

```

```

    /* Returns the error message for the last error which occurred on the
    * given compressed file. errnum is set to zlib error number. If an
    * error occurred in the file system and not in the compression library,
    * errnum is set to Z_ERRNO and the application may consult errno
    * to get the exact error code.
    */
    const char* error(int* errnum) {
        return ::gzerror(m_fp, errnum);
    }

    gzFile fp() { return m_fp; }

    ostream& os() {
        if (m_os == 0) m_os = new ostrstream;
        return *m_os;
    }

    void os_flush() {
        if (m_os && m_os->pcount() > 0) {
            ostrstream* oss = new ostrstream;
            oss->fill(m_os->fill());
            oss->flags(m_os->flags());
            oss->precision(m_os->precision());
            oss->width(m_os->width());
            ::gzwrite(m_fp, m_os->str(), m_os->pcount());
            delete[] m_os->str(); delete m_os; m_os = oss;
        }
    }

private:
    gzFile m_fp;
    ostrstream* m_os;
};

/*
    * Binary write the given (array of) object(s) into the compressed file.
    * returns the number of uncompressed bytes actually written
    * (0 in case of error).
    */
template <class T, class Items>
inline int write(ozstream& zs, const T* x, Items items) {
    return ::gzwrite(zs.fp(), (voidp) x, items*sizeof(T));
}

/*
    * Binary output with the '<' operator.
    */
template <class T>
inline ozstream& operator<(ozstream& zs, const T& x) {
    ::gzwrite(zs.fp(), (voidp) &x, sizeof(T));
    return zs;
}

inline zstringlen::zstringlen(ozstream& zs, const char* x) {
    val.byte = 255; val.word = ::strlen(x);
    if (val.word < 255) zs < (val.byte = val.word);
    else zs < val;
}

/*
    * Write length of string + the string with the '<' operator.
    */
inline ozstream& operator<(ozstream& zs, const char* x) {
    zstringlen len(zs, x);
    ::gzwrite(zs.fp(), (voidp) x, len.value());
    return zs;
}

#ifdef _MSC_VER
inline ozstream& operator<(ozstream& zs, char* const& x) {
    return zs < (const char*) x;
}
#endif

```

```
/*
 * Ascii write with the << operator;
 */
template <class T>
inline ostream& operator<<(ozstream& zs, const T& x) {
    zs.os_flush();
    return zs.os() << x;
}

#endif
```

```

#include "zstream.h"
#include <math.h>
#include <stdlib.h>
#include <iomanip.h>

void main() {
    char h[256] = "Hello";
    char* g = "Goodbye";
    ostream out("temp.gz");
    out < "This works well" < h < g;
    out.close();

    istream in("temp.gz"); // read it back
    char *x = read_string(in), *y = new char[256], z[256];
    in > y > z;
    in.close();
    cout << x << endl << y << endl << z << endl;

    out.open("temp.gz"); // try ascii output; zcat temp.gz to see the results
    out << setw(50) << setfill('#') << setprecision(20) << x << endl << y << endl << z <<
endl;
    out << z << endl << y << endl << x << endl;
    out << 1.1234567890123456789 << endl;

    delete[] x; delete[] y;
}

```

These classes provide a C++ stream interface to the zlib library. It allows you to do things like:

```
gzofstream outf("blah.gz");  
outf << "These go into the gzip file " << 123 << endl;
```

It does this by deriving a specialized stream buffer for gzipped files, which is the way Stroustrup would have done it. :->

The gzifstream and gzofstream classes were originally written by Kevin Ruland and made available in the zlib contrib/iostream directory. The older version still compiles under gcc 2.xx, but not under gcc 3.xx, which sparked the development of this version.

The new classes are as standard-compliant as possible, closely following the approach of the standard library's fstream classes. It compiles under gcc versions 3.2 and 3.3, but not under gcc 2.xx. This is mainly due to changes in the standard library naming scheme. The new version of gzifstream/gzofstream/gzfilebuf differs from the previous one in the following respects:

- added showmanyc
- added setbuf, with support for unbuffered output via setbuf(0,0)
- a few bug fixes of stream behavior
- gzipped output file opened with default compression level instead of maximum level
- setcompressionlevel()/strategy() members replaced by single setcompression()

The code is provided "as is", with the permission to use, copy, modify, distribute and sell it for any purpose without fee.

Ludwig Schwardt
<schwardt@sun.ac.za>

DSP Lab
Electrical & Electronic Engineering Department
University of Stellenbosch
South Africa

Possible upgrades to gzfilebuf:

- The ability to do putback (e.g. putbackfail)
- The ability to seek (zlib supports this, but could be slow/tricky)
- Simultaneous read/write access (does it make sense?)
- Support for ios_base::ate open mode
- Locale support?
- Check public interface to see which calls give problems (due to dependence on library internals)
- Override operator<<(ostream&, gzfilebuf*) to allow direct copying of stream buffer to stream (i.e. os << is.rdbuf();)

```
/*
 * Test program for gzifstream and gzofstream
 *
 * by Ludwig Schwardt <schwardt@sun.ac.za>
 * original version by Kevin Ruland <kevin@rodin.wustl.edu>
 */

#include "zfstream.h"
#include <iostream>          // for cout

int main() {

    gzofstream outf;
    gzifstream inf;
    char buf[80];

    outf.open("test1.txt.gz");
    outf << "The quick brown fox sidestepped the lazy canine\n"
        << 1.3 << "\nPlan " << 9 << std::endl;
    outf.close();
    std::cout << "Wrote the following message to 'test1.txt.gz' (check with zcat or zless):\n"
        << "The quick brown fox sidestepped the lazy canine\n"
        << 1.3 << "\nPlan " << 9 << std::endl;

    std::cout << "\nReading 'test1.txt.gz' (buffered) produces:\n";
    inf.open("test1.txt.gz");
    while (inf.getline(buf, 80, '\n')) {
        std::cout << buf << "\t(" << inf.rdbuf()->in_avail() << " chars left in buffer)\n";
    }
    inf.close();

    outf.rdbuf()->pubsetbuf(0, 0);
    outf.open("test2.txt.gz");
    outf << setcompression(Z_NO_COMPRESSION)
        << "The quick brown fox sidestepped the lazy canine\n"
        << 1.3 << "\nPlan " << 9 << std::endl;
    outf.close();
    std::cout << "\nWrote the same message to 'test2.txt.gz' in uncompressed form";

    std::cout << "\nReading 'test2.txt.gz' (unbuffered) produces:\n";
    inf.rdbuf()->pubsetbuf(0, 0);
    inf.open("test2.txt.gz");
    while (inf.getline(buf, 80, '\n')) {
        std::cout << buf << "\t(" << inf.rdbuf()->in_avail() << " chars left in buffer)\n";
    }
    inf.close();

    return 0;
}
```



```
/*
 * A C++ I/O streams interface to the zlib gz* functions
 *
 * by Ludwig Schwardt <schwardt@sun.ac.za>
 * original version by Kevin Ruland <kevin@rodin.wustl.edu>
 *
 * This version is standard-compliant and compatible with gcc 3.x.
 */

#include "zfstream.h"
#include <cstring>           // for strcpy, strcat, strlen (mode strings)
#include <cstdio>           // for BUFSIZ

// Internal buffer sizes (default and "unbuffered" versions)
#define BIGBUFSIZE BUFSIZ
#define SMALLBUFSIZE 1

/*****

// Default constructor
gzfilebuf::gzfilebuf()
: file(NULL), io_mode(std::ios_base::openmode(0)), own_fd(false),
  buffer(NULL), buffer_size(BIGBUFSIZE), own_buffer(true)
{
    // No buffers to start with
    this->disable_buffer();
}

// Destructor
gzfilebuf::~gzfilebuf()
{
    // Sync output buffer and close only if responsible for file
    // (i.e. attached streams should be left open at this stage)
    this->sync();
    if (own_fd)
        this->close();
    // Make sure internal buffer is deallocated
    this->disable_buffer();
}

// Set compression level and strategy
int
gzfilebuf::setcompression(int comp_level,
                          int comp_strategy)
{
    return gzsetparams(file, comp_level, comp_strategy);
}

// Open gzipped file
gzfilebuf*
gzfilebuf::open(const char *name,
                std::ios_base::openmode mode)
{
    // Fail if file already open
    if (this->is_open())
        return NULL;
    // Don't support simultaneous read/write access (yet)
    if ((mode & std::ios_base::in) && (mode & std::ios_base::out))
        return NULL;

    // Build mode string for gzopen and check it [27.8.1.3.2]
    char char_mode[6] = "\0\0\0\0\0";
    if (!this->open_mode(mode, char_mode))
        return NULL;

    // Attempt to open file
    if ((file = gzopen(name, char_mode)) == NULL)
        return NULL;

    // On success, allocate internal buffer and set flags
    this->enable_buffer();
    io_mode = mode;
    own_fd = true;
    return this;
}
*****/
```

```

}

// Attach to gzipped file
gzfilebuf*
gzfilebuf::attach(int fd,
                  std::ios_base::openmode mode)
{
    // Fail if file already open
    if (this->is_open())
        return NULL;
    // Don't support simultaneous read/write access (yet)
    if ((mode & std::ios_base::in) && (mode & std::ios_base::out))
        return NULL;

    // Build mode string for gzdopen and check it [27.8.1.3.2]
    char char_mode[6] = "\0\0\0\0\0";
    if (!this->open_mode(mode, char_mode))
        return NULL;

    // Attempt to attach to file
    if ((file = gzdopen(fd, char_mode)) == NULL)
        return NULL;

    // On success, allocate internal buffer and set flags
    this->enable_buffer();
    io_mode = mode;
    own_fd = false;
    return this;
}

// Close gzipped file
gzfilebuf*
gzfilebuf::close()
{
    // Fail immediately if no file is open
    if (!this->is_open())
        return NULL;
    // Assume success
    gzfilebuf* retval = this;
    // Attempt to sync and close gzipped file
    if (this->sync() == -1)
        retval = NULL;
    if (gzclose(file) < 0)
        retval = NULL;
    // File is now gone anyway (postcondition [27.8.1.3.8])
    file = NULL;
    own_fd = false;
    // Destroy internal buffer if it exists
    this->disable_buffer();
    return retval;
}

/* * * * * * */

// Convert int open mode to mode string
bool
gzfilebuf::open_mode(std::ios_base::openmode mode,
                    char* c_mode) const
{
    bool testb = mode & std::ios_base::binary;
    bool testi = mode & std::ios_base::in;
    bool testo = mode & std::ios_base::out;
    bool testt = mode & std::ios_base::trunc;
    bool testa = mode & std::ios_base::app;

    // Check for valid flag combinations - see [27.8.1.3.2] (Table 92)
    // Original zfstream hardcoded the compression level to maximum here...
    // Double the time for less than 1% size improvement seems
    // excessive though - keeping it at the default level
    // To change back, just append "9" to the next three mode strings
    if (!testi && testo && !testt && !testa)
        strcpy(c_mode, "w");
    if (!testi && testo && !testt && testa)
        strcpy(c_mode, "a");

```

```
if (!testi && testo && testt && !testa)
    strcpy(c_mode, "w");
if (testi && !testo && !testt && !testa)
    strcpy(c_mode, "r");
// No read/write mode yet
// if (testi && testo && !testt && !testa)
//     strcpy(c_mode, "r+");
// if (testi && testo && testt && !testa)
//     strcpy(c_mode, "w+");

// Mode string should be empty for invalid combination of flags
if (strlen(c_mode) == 0)
    return false;
if (testb)
    strcat(c_mode, "b");
return true;
}

// Determine number of characters in internal get buffer
std::streamsize
gzfilebuf::showmanyc()
{
    // Calls to underflow will fail if file not opened for reading
    if (!this->is_open() || !(io_mode & std::ios_base::in))
        return -1;
    // Make sure get area is in use
    if (this->gptr() && (this->gptr() < this->egptr()))
        return std::streamsize(this->egptr() - this->gptr());
    else
        return 0;
}

// Fill get area from gzipped file
gzfilebuf::int_type
gzfilebuf::underflow()
{
    // If something is left in the get area by chance, return it
    // (this shouldn't normally happen, as underflow is only supposed
    // to be called when gptr >= egptr, but it serves as error check)
    if (this->gptr() && (this->gptr() < this->egptr()))
        return traits_type::to_int_type(*(this->gptr()));

    // If the file hasn't been opened for reading, produce error
    if (!this->is_open() || !(io_mode & std::ios_base::in))
        return traits_type::eof();

    // Attempt to fill internal buffer from gzipped file
    // (buffer must be guaranteed to exist...)
    int bytes_read = gzread(file, buffer, buffer_size);
    // Indicates error or EOF
    if (bytes_read <= 0)
    {
        // Reset get area
        this->setg(buffer, buffer, buffer);
        return traits_type::eof();
    }
    // Make all bytes read from file available as get area
    this->setg(buffer, buffer, buffer + bytes_read);

    // Return next character in get area
    return traits_type::to_int_type(*(this->gptr()));
}

// Write put area to gzipped file
gzfilebuf::int_type
gzfilebuf::overflow(int_type c)
{
    // Determine whether put area is in use
    if (this->pbase())
    {
        // Double-check pointer range
        if (this->pptr() > this->epptr() || this->pptr() < this->pbase())
            return traits_type::eof();
        // Add extra character to buffer if not EOF
    }
}
```

```
if (!traits_type::eq_int_type(c, traits_type::eof()))
{
    *(this->pptr()) = traits_type::to_char_type(c);
    this->pbump(1);
}
// Number of characters to write to file
int bytes_to_write = this->pptr() - this->pbase();
// Overflow doesn't fail if nothing is to be written
if (bytes_to_write > 0)
{
    // If the file hasn't been opened for writing, produce error
    if (!this->is_open() || !(io_mode & std::ios_base::out))
        return traits_type::eof();
    // If gzipped file won't accept all bytes written to it, fail
    if (gzwrite(file, this->pbase(), bytes_to_write) != bytes_to_write)
        return traits_type::eof();
    // Reset next pointer to point to pbase on success
    this->pbump(-bytes_to_write);
}
}
// Write extra character to file if not EOF
else if (!traits_type::eq_int_type(c, traits_type::eof()))
{
    // If the file hasn't been opened for writing, produce error
    if (!this->is_open() || !(io_mode & std::ios_base::out))
        return traits_type::eof();
    // Impromptu char buffer (allows "unbuffered" output)
    char_type last_char = traits_type::to_char_type(c);
    // If gzipped file won't accept this character, fail
    if (gzwrite(file, &last_char, 1) != 1)
        return traits_type::eof();
}

// If you got here, you have succeeded (even if c was EOF)
// The return value should therefore be non-EOF
if (traits_type::eq_int_type(c, traits_type::eof()))
    return traits_type::not_eof(c);
else
    return c;
}

// Assign new buffer
std::streambuf*
gzfilebuf::setbuf(char_type* p,
                  std::streamsize n)
{
    // First make sure stuff is sync'ed, for safety
    if (this->sync() == -1)
        return NULL;
    // If buffering is turned off on purpose via setbuf(0,0), still allocate one...
    // "Unbuffered" only really refers to put [27.8.1.4.10], while get needs at
    // least a buffer of size 1 (very inefficient though, therefore make it bigger?)
    // This follows from [27.5.2.4.3]/12 (gp_ptr needs to point at something, it seems)
    if (!p || !n)
    {
        // Replace existing buffer (if any) with small internal buffer
        this->disable_buffer();
        buffer = NULL;
        buffer_size = 0;
        own_buffer = true;
        this->enable_buffer();
    }
    else
    {
        // Replace existing buffer (if any) with external buffer
        this->disable_buffer();
        buffer = p;
        buffer_size = n;
        own_buffer = false;
        this->enable_buffer();
    }
    return this;
}
```

```
// Write put area to gzipped file (i.e. ensures that put area is empty)
int
gzfilebuf::sync()
{
    return traits_type::eq_int_type(this->overflow(), traits_type::eof()) ? -1 : 0;
}

/* * * * * * */

// Allocate internal buffer
void
gzfilebuf::enable_buffer()
{
    // If internal buffer required, allocate one
    if (own_buffer && !buffer)
    {
        // Check for buffered vs. "unbuffered"
        if (buffer_size > 0)
        {
            // Allocate internal buffer
            buffer = new char_type[buffer_size];
            // Get area starts empty and will be expanded by underflow as need arises
            this->setg(buffer, buffer, buffer);
            // Setup entire internal buffer as put area.
            // The one-past-end pointer actually points to the last element of the buffer,
            // so that overflow(c) can safely add the extra character c to the sequence.
            // These pointers remain in place for the duration of the buffer
            this->setp(buffer, buffer + buffer_size - 1);
        }
        else
        {
            // Even in "unbuffered" case, (small?) get buffer is still required
            buffer_size = SMALLBUFSIZE;
            buffer = new char_type[buffer_size];
            this->setg(buffer, buffer, buffer);
            // "Unbuffered" means no put buffer
            this->setp(0, 0);
        }
    }
    else
    {
        // If buffer already allocated, reset buffer pointers just to make sure no
        // stale chars are lying around
        this->setg(buffer, buffer, buffer);
        this->setp(buffer, buffer + buffer_size - 1);
    }
}

// Destroy internal buffer
void
gzfilebuf::disable_buffer()
{
    // If internal buffer exists, deallocate it
    if (own_buffer && buffer)
    {
        // Preserve unbuffered status by zeroing size
        if (!this->pbase())
            buffer_size = 0;
        delete[] buffer;
        buffer = NULL;
        this->setg(0, 0, 0);
        this->setp(0, 0);
    }
    else
    {
        // Reset buffer pointers to initial state if external buffer exists
        this->setg(buffer, buffer, buffer);
        if (buffer)
            this->setp(buffer, buffer + buffer_size - 1);
        else
            this->setp(0, 0);
    }
}
```

```
/*
// Default constructor initializes stream buffer
gzifstream::gzifstream()
: std::istream(NULL), sb()
{ this->init(&sb); }

// Initialize stream buffer and open file
gzifstream::gzifstream(const char* name,
                        std::ios_base::openmode mode)
: std::istream(NULL), sb()
{
    this->init(&sb);
    this->open(name, mode);
}

// Initialize stream buffer and attach to file
gzifstream::gzifstream(int fd,
                        std::ios_base::openmode mode)
: std::istream(NULL), sb()
{
    this->init(&sb);
    this->attach(fd, mode);
}

// Open file and go into fail() state if unsuccessful
void
gzifstream::open(const char* name,
                 std::ios_base::openmode mode)
{
    if (!sb.open(name, mode | std::ios_base::in))
        this->setstate(std::ios_base::failbit);
    else
        this->clear();
}

// Attach to file and go into fail() state if unsuccessful
void
gzifstream::attach(int fd,
                   std::ios_base::openmode mode)
{
    if (!sb.attach(fd, mode | std::ios_base::in))
        this->setstate(std::ios_base::failbit);
    else
        this->clear();
}

// Close file
void
gzifstream::close()
{
    if (!sb.close())
        this->setstate(std::ios_base::failbit);
}

/*
// Default constructor initializes stream buffer
gzofstream::gzofstream()
: std::ostream(NULL), sb()
{ this->init(&sb); }

// Initialize stream buffer and open file
gzofstream::gzofstream(const char* name,
                       std::ios_base::openmode mode)
: std::ostream(NULL), sb()
{
    this->init(&sb);
    this->open(name, mode);
}

// Initialize stream buffer and attach to file
gzofstream::gzofstream(int fd,
                       std::ios_base::openmode mode)
```

```
: std::ostream(NULL), sb()
{
    this->init(&sb);
    this->attach(fd, mode);
}

// Open file and go into fail() state if unsuccessful
void
gzofstream::open(const char* name,
                 std::ios_base::openmode mode)
{
    if (!sb.open(name, mode | std::ios_base::out))
        this->setstate(std::ios_base::failbit);
    else
        this->clear();
}

// Attach to file and go into fail() state if unsuccessful
void
gzofstream::attach(int fd,
                   std::ios_base::openmode mode)
{
    if (!sb.attach(fd, mode | std::ios_base::out))
        this->setstate(std::ios_base::failbit);
    else
        this->clear();
}

// Close file
void
gzofstream::close()
{
    if (!sb.close())
        this->setstate(std::ios_base::failbit);
}
```

```

/*
 * A C++ I/O streams interface to the zlib gz* functions
 *
 * by Ludwig Schwardt <schwardt@sun.ac.za>
 * original version by Kevin Ruland <kevin@rodin.wustl.edu>
 *
 * This version is standard-compliant and compatible with gcc 3.x.
 */

#ifndef ZFSTREAM_H
#define ZFSTREAM_H

#include <istream> // not iostream, since we don't need cin/cout
#include <ostream>
#include "zlib.h"

/*****
**
 * @brief Gzipped file stream buffer class.
 *
 * This class implements basic_filebuf for gzipped files. It doesn't yet support
 * seeking (allowed by zlib but slow/limited), putback and read/write access
 * (tricky). Otherwise, it attempts to be a drop-in replacement for the standard
 * file streambuf.
 */
class gzfilebuf : public std::streambuf
{
public:
    // Default constructor.
    gzfilebuf();

    // Destructor.
    virtual
    ~gzfilebuf();

    /**
     * @brief Set compression level and strategy on the fly.
     * @param comp_level Compression level (see zlib.h for allowed values)
     * @param comp_strategy Compression strategy (see zlib.h for allowed values)
     * @return Z_OK on success, Z_STREAM_ERROR otherwise.
     *
     * Unfortunately, these parameters cannot be modified separately, as the
     * previous zfstream version assumed. Since the strategy is seldom changed,
     * it can default and setcompression(level) then becomes like the old
     * setcompressionlevel(level).
     */
    int
    setcompression(int comp_level,
                  int comp_strategy = Z_DEFAULT_STRATEGY);

    /**
     * @brief Check if file is open.
     * @return True if file is open.
     */
    bool
    is_open() const { return (file != NULL); }

    /**
     * @brief Open gzipped file.
     * @param name File name.
     * @param mode Open mode flags.
     * @return @c this on success, NULL on failure.
     */
    gzfilebuf*
    open(const char* name,
         std::ios_base::openmode mode);

    /**
     * @brief Attach to already open gzipped file.
     * @param fd File descriptor.
     * @param mode Open mode flags.
     * @return @c this on success, NULL on failure.
     */

```



```

gzfilebuf*
attach(int fd,
      std::ios_base::openmode mode);

/**
 * @brief Close gzipped file.
 * @return @c this on success, NULL on failure.
 */
gzfilebuf*
close();

protected:
/**
 * @brief Convert ios open mode int to mode string used by zlib.
 * @return True if valid mode flag combination.
 */
bool
open_mode(std::ios_base::openmode mode,
          char* c_mode) const;

/**
 * @brief Number of characters available in stream buffer.
 * @return Number of characters.
 *
 * This indicates number of characters in get area of stream buffer.
 * These characters can be read without accessing the gzipped file.
 */
virtual std::streamsize
showmanyc();

/**
 * @brief Fill get area from gzipped file.
 * @return First character in get area on success, EOF on error.
 *
 * This actually reads characters from gzipped file to stream
 * buffer. Always buffered.
 */
virtual int_type
underflow();

/**
 * @brief Write put area to gzipped file.
 * @param c Extra character to add to buffer contents.
 * @return Non-EOF on success, EOF on error.
 *
 * This actually writes characters in stream buffer to
 * gzipped file. With unbuffered output this is done one
 * character at a time.
 */
virtual int_type
overflow(int_type c = traits_type::eof());

/**
 * @brief Installs external stream buffer.
 * @param p Pointer to char buffer.
 * @param n Size of external buffer.
 * @return @c this on success, NULL on failure.
 *
 * Call setbuf(0,0) to enable unbuffered output.
 */
virtual std::streambuf*
setbuf(char_type* p,
      std::streamsize n);

/**
 * @brief Flush stream buffer to file.
 * @return 0 on success, -1 on error.
 *
 * This calls underflow(EOF) to do the job.
 */
virtual int
sync();

//

```

```
// Some future enhancements
//
// virtual int_type uflow();
// virtual int_type pbackfail(int_type c = traits_type::eof());
// virtual pos_type
// seekoff(off_type off,
//         std::ios_base::seekdir way,
//         std::ios_base::openmode mode = std::ios_base::in|std::ios_base::out);
// virtual pos_type
// seekpos(pos_type sp,
//         std::ios_base::openmode mode = std::ios_base::in|std::ios_base::out);

private:
/**
 * @brief Allocate internal buffer.
 *
 * This function is safe to call multiple times. It will ensure
 * that a proper internal buffer exists if it is required. If the
 * buffer already exists or is external, the buffer pointers will be
 * reset to their original state.
 */
void
enable_buffer();

/**
 * @brief Destroy internal buffer.
 *
 * This function is safe to call multiple times. It will ensure
 * that the internal buffer is deallocated if it exists. In any
 * case, it will also reset the buffer pointers.
 */
void
disable_buffer();

/**
 * Underlying file pointer.
 */
gzFile file;

/**
 * Mode in which file was opened.
 */
std::ios_base::openmode io_mode;

/**
 * @brief True if this object owns file descriptor.
 *
 * This makes the class responsible for closing the file
 * upon destruction.
 */
bool own_fd;

/**
 * @brief Stream buffer.
 *
 * For simplicity this remains allocated on the free store for the
 * entire life span of the gzfilebuf object, unless replaced by setbuf.
 */
char_type* buffer;

/**
 * @brief Stream buffer size.
 *
 * Defaults to system default buffer size (typically 8192 bytes).
 * Modified by setbuf.
 */
std::streamsize buffer_size;

/**
 * @brief True if this object owns stream buffer.
 *
 * This makes the class responsible for deleting the buffer
 * upon destruction.
 */
```

```
bool own_buffer;
};

/*****

/**
 * @brief Gzipped file input stream class.
 *
 * This class implements ifstream for gzipped files. Seeking and putback
 * is not supported yet.
 */
class gzifstream : public std::istream
{
public:
    // Default constructor
    gzifstream();

    /**
     * @brief Construct stream on gzipped file to be opened.
     * @param name File name.
     * @param mode Open mode flags (forced to contain ios::in).
     */
    explicit
    gzifstream(const char* name,
               std::ios_base::openmode mode = std::ios_base::in);

    /**
     * @brief Construct stream on already open gzipped file.
     * @param fd File descriptor.
     * @param mode Open mode flags (forced to contain ios::in).
     */
    explicit
    gzifstream(int fd,
               std::ios_base::openmode mode = std::ios_base::in);

    /**
     * Obtain underlying stream buffer.
     */
    gzfilebuf*
    rdbuf() const
    { return const_cast<gzfilebuf*>(&sb); }

    /**
     * @brief Check if file is open.
     * @return True if file is open.
     */
    bool
    is_open() { return sb.is_open(); }

    /**
     * @brief Open gzipped file.
     * @param name File name.
     * @param mode Open mode flags (forced to contain ios::in).
     *
     * Stream will be in state good() if file opens successfully;
     * otherwise in state fail(). This differs from the behavior of
     * ifstream, which never sets the state to good() and therefore
     * won't allow you to reuse the stream for a second file unless
     * you manually clear() the state. The choice is a matter of
     * convenience.
     */
    void
    open(const char* name,
         std::ios_base::openmode mode = std::ios_base::in);

    /**
     * @brief Attach to already open gzipped file.
     * @param fd File descriptor.
     * @param mode Open mode flags (forced to contain ios::in).
     *
     * Stream will be in state good() if attach succeeded; otherwise
     * in state fail().
     */
    void
```

```

attach(int fd,
        std::ios_base::openmode mode = std::ios_base::in);

/**
 * @brief Close gzipped file.
 *
 * Stream will be in state fail() if close failed.
 */
void
close();

private:
/**
 * Underlying stream buffer.
 */
gzfilebuf sb;
};

/*****/

/**
 * @brief Gzipped file output stream class.
 *
 * This class implements ostream for gzipped files. Seeking and putback
 * is not supported yet.
 */
class gzofstream : public std::ostream
{
public:
    // Default constructor
    gzofstream();

    /**
     * @brief Construct stream on gzipped file to be opened.
     * @param name File name.
     * @param mode Open mode flags (forced to contain ios::out).
     */
    explicit
    gzofstream(const char* name,
               std::ios_base::openmode mode = std::ios_base::out);

    /**
     * @brief Construct stream on already open gzipped file.
     * @param fd File descriptor.
     * @param mode Open mode flags (forced to contain ios::out).
     */
    explicit
    gzofstream(int fd,
               std::ios_base::openmode mode = std::ios_base::out);

    /**
     * Obtain underlying stream buffer.
     */
    gzfilebuf*
    rdbuf() const
    { return const_cast<gzfilebuf*>(&sb); }

    /**
     * @brief Check if file is open.
     * @return True if file is open.
     */
    bool
    is_open() { return sb.is_open(); }

    /**
     * @brief Open gzipped file.
     * @param name File name.
     * @param mode Open mode flags (forced to contain ios::out).
     *
     * Stream will be in state good() if file opens successfully;
     * otherwise in state fail(). This differs from the behavior of
     * ostream, which never sets the state to good() and therefore
     * won't allow you to reuse the stream for a second file unless
     * you manually clear() the state. The choice is a matter of

```

```

    * convenience.
    */
void
open(const char* name,
      std::ios_base::openmode mode = std::ios_base::out);

/**
 * @brief Attach to already open gzipped file.
 * @param fd File descriptor.
 * @param mode Open mode flags (forced to contain ios::out).
 *
 * Stream will be in state good() if attach succeeded; otherwise
 * in state fail().
 */
void
attach(int fd,
       std::ios_base::openmode mode = std::ios_base::out);

/**
 * @brief Close gzipped file.
 *
 * Stream will be in state fail() if close failed.
 */
void
close();

private:
    /**
     * Underlying stream buffer.
     */
    gzfilebuf sb;
};

/*****

/**
 * @brief Gzipped file output stream manipulator class.
 *
 * This class defines a two-argument manipulator for gzofstream. It is used
 * as base for the setcompression(int,int) manipulator.
 */
template<typename T1, typename T2>
class gzomanip2
{
public:
    // Allows inserter to peek at internals
    template <typename Ta, typename Tb>
    friend gzofstream&
    operator<<(gzofstream&,
              const gzomanip2<Ta,Tb>&);

    // Constructor
    gzomanip2(gzofstream& (*f)(gzofstream&, T1, T2),
              T1 v1,
              T2 v2);

private:
    // Underlying manipulator function
    gzofstream&
    (*func)(gzofstream&, T1, T2);

    // Arguments for manipulator function
    T1 val1;
    T2 val2;
};

*****/

// Manipulator function thunks through to stream buffer
inline gzofstream&
setcompression(gzofstream &gzs, int l, int s = Z_DEFAULT_STRATEGY)
{
    (gzs.rdbuf()->setcompression(l, s));
    return gzs;
}

```

```
// Manipulator constructor stores arguments
template<typename T1, typename T2>
inline
gzomanip2<T1,T2>::gzomanip2(gzofstream &(*f)(gzofstream &, T1, T2),
                           T1 v1,
                           T2 v2)
: func(f), val1(v1), val2(v2)
{ }

// Insertor applies underlying manipulator function to stream
template<typename T1, typename T2>
inline gzofstream&
operator<<(gzofstream& s, const gzomanip2<T1,T2>& m)
{ return (*m.func)(s, m.val1, m.val2); }

// Insert this onto stream to simplify setting of compression level
inline gzomanip2<int,int>
setcompression(int l, int s = Z_DEFAULT_STRATEGY)
{ return gzomanip2<int,int>(&setcompression, l, s); }

#endif // ZFSTREAM_H
```

```

; match.asm -- Pentium-Pro optimized version of longest_match()
;
; Updated for zlib 1.1.3 and converted to MASM 6.1x
; Copyright (C) 2000 Dan Higdon <hdan@kinesoft.com>
;               and Chuck Walbourn <chuckw@kinesoft.com>
; Corrections by Cosmin Truta <cosmint@cs.ubbcluj.ro>
;
; This is free software; you can redistribute it and/or modify it
; under the terms of the GNU General Public License.

; Based on match.S
; Written for zlib 1.1.2
; Copyright (C) 1998 Brian Raiter <breadbox@muppetlabs.com>
;
; Modified by Gilles Vollant (2005) for add gzhead and gzindex

        .686P
        .MODEL    FLAT

;=====
; EQUATES
;=====

MAX_MATCH      EQU 258
MIN_MATCH      EQU 3
MIN_LOOKAHEAD  EQU (MAX_MATCH + MIN_MATCH + 1)
MAX_MATCH_8    EQU ((MAX_MATCH + 7) AND (NOT 7))

;=====
; STRUCTURES
;=====

; This STRUCT assumes a 4-byte alignment

DEFLATE_STATE  STRUCT
ds_strm                dd ?
ds_status              dd ?
ds_pending_buf         dd ?
ds_pending_buf_size    dd ?
ds_pending_out         dd ?
ds_pending            dd ?
ds_wrap               dd ?
; gzhead and gzindex are added in zlib 1.2.2.2 (see deflate.h)
ds_gzhead              dd ?
ds_gzindex             dd ?
ds_data_type           db ?
ds_method              db ?
                      db ?      ; padding
                      db ?      ; padding
ds_last_flush          dd ?
ds_w_size              dd ?      ; used
ds_w_bits              dd ?
ds_w_mask              dd ?      ; used
ds_window             dd ?      ; used
ds_window_size        dd ?
ds_prev               dd ?      ; used
ds_head              dd ?
ds_ins_h              dd ?
ds_hash_size          dd ?
ds_hash_bits          dd ?
ds_hash_mask          dd ?
ds_hash_shift         dd ?
ds_block_start        dd ?
ds_match_length       dd ?      ; used
ds_prev_match         dd ?      ; used
ds_match_available    dd ?
ds_strstart           dd ?      ; used
ds_match_start        dd ?      ; used
ds_lookahead          dd ?      ; used
ds_prev_length        dd ?      ; used
ds_max_chain_length   dd ?      ; used
ds_max_laxy_match     dd ?
ds_level              dd ?

```

```

ds_strategy          dd ?
ds_good_match        dd ?      ; used
ds_nice_match        dd ?      ; used

; Don't need anymore of the struct for match
DEFLATE_STATE        ENDS

;=====
; CODE
;=====
_TEXT                SEGMENT

;-----
; match_init
;-----
        ALIGN      4
PUBLIC  _match_init
_match_init          PROC
        ; no initialization needed
        ret
_match_init          ENDP

;-----
; uInt longest_match(deflate_state *deflatestate, IPos curmatch)
;-----
        ALIGN      4

PUBLIC  _longest_match
_longest_match        PROC

; Since this code uses EBP for a scratch register, the stack frame must
; be manually constructed and referenced relative to the ESP register.

; Stack image
; Variables
chainlenwmask        = 0      ; high word: current chain len
                        ; low word: s->wmask
window               = 4      ; local copy of s->window
windowbestlen        = 8      ; s->window + bestlen
scanend              = 12     ; last two bytes of string
scanstart            = 16     ; first two bytes of string
scanalign            = 20     ; dword-misalignment of string
nicematch            = 24     ; a good enough match size
bestlen              = 28     ; size of best match so far
scan                 = 32     ; ptr to string wanting match
varsize              = 36     ; number of bytes (also offset to last saved register)

; Saved Registers (actually pushed into place)
ebx_save             = 36
edi_save             = 40
esi_save             = 44
ebp_save             = 48

; Parameters
retaddr              = 52
deflatestate         = 56
curmatch             = 60

; Save registers that the compiler may be using
        push      ebp
        push      edi
        push      esi
        push      ebx

; Allocate local variable space
        sub       esp,varsize

; Retrieve the function arguments. ecx will hold cur_match
; throughout the entire function. edx will hold the pointer to the
; deflate_state structure during the function's setup (before
; entering the main loop).

        mov       edx,[esp+deflatestate]
ASSUME  edx:PTR DEFLATE_STATE

```



```

        mov     ecx, [esp+curmatch]

; uInt wmask = s->w_mask;
; unsigned chain_length = s->max_chain_length;
; if (s->prev_length >= s->good_match) {
;     chain_length >>= 2;
; }

        mov     eax, [edx].ds_prev_length
        mov     ebx, [edx].ds_good_match
        cmp     eax, ebx
        mov     eax, [edx].ds_w_mask
        mov     ebx, [edx].ds_max_chain_length
        jl      SHORT LastMatchGood
        shr     ebx, 2
LastMatchGood:

; chainlen is decremented once beforehand so that the function can
; use the sign flag instead of the zero flag for the exit test.
; It is then shifted into the high word, to make room for the wmask
; value, which it will always accompany.

        dec     ebx
        shl     ebx, 16
        or      ebx, eax
        mov     [esp+chainlenwmask], ebx

; if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;

        mov     eax, [edx].ds_nice_match
        mov     ebx, [edx].ds_lookahead
        cmp     ebx, eax
        jl      SHORT LookaheadLess
        mov     ebx, eax
LookaheadLess:
        mov     [esp+nicematch], ebx

; /* register Bytef *scan = s->window + s->strstart; */

        mov     esi, [edx].ds_window
        mov     [esp+window], esi
        mov     ebp, [edx].ds_strstart
        lea     edi, [esi+ebp]
        mov     [esp+scan], edi

; /* Determine how many bytes the scan ptr is off from being */
; /* dword-aligned. */

        mov     eax, edi
        neg     eax
        and     eax, 3
        mov     [esp+scanalign], eax

; /* IPos limit = s->strstart > (IPos)MAX_DIST(s) ? */
; /*     s->strstart - (IPos)MAX_DIST(s) : NIL; */

        mov     eax, [edx].ds_w_size
        sub     eax, MIN_LOOKAHEAD
        sub     ebp, eax
        jg      SHORT LimitPositive
        xor     ebp, ebp
LimitPositive:

; /* int best_len = s->prev_length; */

        mov     eax, [edx].ds_prev_length
        mov     [esp+bestlen], eax

; /* Store the sum of s->window + best_len in %esi locally, and in %esi. */

        add     esi, eax
        mov     [esp+windowbestlen], esi

```

```

/* register ush scan_start = *(ushf*)scan; */
/* register ush scan_end   = *(ushf*)(scan+best_len-1); */
/* Posf *prev = s->prev; */

    movzx    ebx, WORD PTR[edi]
    mov      [esp+scanstart], ebx
    movzx    ebx, WORD PTR[eax+edi-1]
    mov      [esp+scanend], ebx
    mov      edi, [edx].ds_prev

/* Jump into the main loop. */

    mov      edx, [esp+chainlenwmask]
    jmp      SHORT LoopEntry

/* do {
 *   match = s->window + cur_match;
 *   if (*(ushf*)(match+best_len-1) != scan_end ||
 *       *(ushf*)match != scan_start) continue;
 *   [...]
 * } while ((cur_match = prev[cur_match & wmask]) > limit
 *         && --chain_length != 0);
 *
 * Here is the inner loop of the function. The function will spend the
 * majority of its time in this loop, and majority of that time will
 * be spent in the first ten instructions.
 *
 * Within this loop:
 * %ebx = scanend
 * %ecx = curmatch
 * %edx = chainlenwmask - i.e., ((chainlen << 16) | wmask)
 * %esi = windowbestlen - i.e., (window + bestlen)
 * %edi = prev
 * %ebp = limit
 */

    ALIGN    4
LookupLoop:
    and      ecx, edx
    movzx    ecx, WORD PTR[edi+ecx*2]
    cmp      ecx, ebp
    jbe      LeaveNow
    sub      edx, 000010000H
    js       LeaveNow

LoopEntry:
    movzx    eax, WORD PTR[esi+ecx-1]
    cmp      eax, ebx
    jnz      SHORT LookupLoop

    mov      eax, [esp+window]
    movzx    eax, WORD PTR[eax+ecx]
    cmp      eax, [esp+scanstart]
    jnz      SHORT LookupLoop

/* Store the current value of chainlen. */

    mov      [esp+chainlenwmask], edx

/* Point %edi to the string under scrutiny, and %esi to the string we
 * are hoping to match it up with. In actuality, %esi and %edi are
 * both pointed (MAX_MATCH_8 - scanalign) bytes ahead, and %edx is
 * initialized to -(MAX_MATCH_8 - scanalign). */

    mov      esi, [esp+window]
    mov      edi, [esp+scan]
    add      esi, ecx
    mov      eax, [esp+scanalign]
    mov      edx, -MAX_MATCH_8
    lea      edi, [edi+eax+MAX_MATCH_8]
    lea      esi, [esi+eax+MAX_MATCH_8]

/* Test the strings for equality, 8 bytes at a time. At the end,
 * adjust %edx so that it is offset to the exact byte that mismatched.

```

```
; *
; * We already know at this point that the first three bytes of the
; * strings match each other, and they can be safely passed over before
; * starting the compare loop. So what this code does is skip over 0-3
; * bytes, as much as necessary in order to dword-align the %edi
; * pointer. (%esi will still be misaligned three times out of four.)
; *
; * It should be confessed that this loop usually does not represent
; * much of the total running time. Replacing it with a more
; * straightforward "rep cmpsb" would not drastically degrade
; * performance.
; */

LoopCmps:
    mov     eax, DWORD PTR[esi+edx]
    xor     eax, DWORD PTR[edi+edx]
    jnz     SHORT LeaveLoopCmps

    mov     eax, DWORD PTR[esi+edx+4]
    xor     eax, DWORD PTR[edi+edx+4]
    jnz     SHORT LeaveLoopCmps4

    add     edx, 8
    jnz     SHORT LoopCmps
    jmp     LenMaximum
    ALIGN   4

LeaveLoopCmps4:
    add     edx, 4

LeaveLoopCmps:
    test    eax, 00000FFFFH
    jnz     SHORT LenLower

    add     edx, 2
    shr     eax, 16

LenLower:
    sub     al, 1
    adc     edx, 0

; /* Calculate the length of the match. If it is longer than MAX_MATCH, */
; /* then automatically accept it as the best possible match and leave. */

    lea     eax, [edi+edx]
    mov     edi, [esp+scan]
    sub     eax, edi
    cmp     eax, MAX_MATCH
    jge     SHORT LenMaximum

; /* If the length of the match is not longer than the best match we */
; /* have so far, then forget it and return to the lookup loop. */

    mov     edx, [esp+deflatestate]
    mov     ebx, [esp+bestlen]
    cmp     eax, ebx
    jg      SHORT LongerMatch
    mov     esi, [esp+windowbestlen]
    mov     edi, [edx].ds_prev
    mov     ebx, [esp+scanend]
    mov     edx, [esp+chainlenwmask]
    jmp     LookupLoop
    ALIGN   4

; /*      s->match_start = cur_match; */
; /*      best_len = len; */
; /*      if (len >= nice_match) break; */
; /*      scan_end = *(ushf*)(scan+best_len-1); */

LongerMatch:
    mov     ebx, [esp+nicematch]
    mov     [esp+bestlen], eax
    mov     [edx].ds_match_start, ecx
    cmp     eax, ebx
```

```
jge      SHORT LeaveNow
mov      esi, [esp+window]
add      esi, eax
mov      [esp+windowbestlen], esi
movzx    ebx, WORD PTR[edi+eax-1]
mov      edi, [edx].ds_prev
mov      [esp+scanend], ebx
mov      edx, [esp+chainlenwmask]
jmp      LookupLoop
ALIGN    4

; /* Accept the current string, with the maximum possible length.          */
LenMaximum:
    mov      edx, [esp+deflatestate]
    mov      DWORD PTR[esp+bestlen], MAX_MATCH
    mov      [edx].ds_match_start, ecx

; /* if ((uInt)best_len <= s->lookahead) return (uInt)best_len;          */
; /* return s->lookahead;                                                */
LeaveNow:
    mov      edx, [esp+deflatestate]
    mov      ebx, [esp+bestlen]
    mov      eax, [edx].ds_lookahead
    cmp      ebx, eax
    jg       SHORT LookaheadRet
    mov      eax, ebx
LookaheadRet:

; Restore the stack and return from whence we came.

    add      esp, varsize
    pop      ebx
    pop      esi
    pop      edi
    pop      ebp
    ret

_longest_match ENDP

_TEXT     ENDS
END
```

```
ml64.exe /Flinffasx64 /c /Zi inffasx64.asm  
ml64.exe /Flgvmat64 /c /Zi gvmat64.asm
```

```
;uInt longest_match_x64(
;    deflate_state *s,
;    IPos cur_match);                                /* current match */

; gvmat64.asm -- Asm portion of the optimized longest_match for 32 bits x86
; Copyright (C) 1995-2005 Jean-loup Gailly, Brian Raiter and Gilles Vollant.
;
; File written by Gilles Vollant, by converting to assembly the longest_match
; from Jean-loup Gailly in deflate.c of zlib and infoZip zip.
;
; and by taking inspiration on asm686 with masm, optimised assembly code
; from Brian Raiter, written 1998
;
;    http://www.zlib.net
;    http://www.winimage.com/zLibDll
;    http://www.muppetlabs.com/~breadbox/software/assembly.html
;
; to compile this file for infozip Zip, I use option:
; ml64.exe /Flgvmat64 /c /Zi /DINFOZIP gvmat64.asm
;
; to compile this file for zlib, I use option:
; ml64.exe /Flgvmat64 /c /Zi gvmat64.asm
; Be carrefull to adapt zlib1222add below to your version of zlib
; (if you use a version of zlib before 1.0.4 or after 1.2.2.2, change
; value of zlib1222add later)
;
; This file compile with Microsoft Macro Assembler (x64) for AMD64
;
; ml64.exe is given with Visual Studio 2005 and Windows 2003 server DDK
;
; (you can get Windows 2003 server DDK with ml64 and cl for AMD64 from
; http://www.microsoft.com/whdc/devtools/ddk/default.mspx for low price)
;

;uInt longest_match(s, cur_match)
;    deflate_state *s;
;    IPos cur_match;                                /* current match */
.code
longest_match PROC

;LocalVarsSize    equ 88
;LocalVarsSize    equ 72

; register used : rax,rbx,rcx,rdx,rsi,rdi,r8,r9,r10,r11,r12
; free register : r14,r15
; register can be saved : rsp

    chainlenwmask    equ    rsp + 8 - LocalVarsSize    ; high word: current chain len
                                                            ; low word: s->wmask
;window            equ    rsp + xx - LocalVarsSize    ; local copy of s->window ; stored in r1
0
;windowbestlen      equ    rsp + xx - LocalVarsSize    ; s->window + bestlen , use r10+r11
;scanstart          equ    rsp + xx - LocalVarsSize    ; first two bytes of string ; stored in
r12w
;scanend            equ    rsp + xx - LocalVarsSize    ; last two bytes of string use ebx
;scanalign          equ    rsp + xx - LocalVarsSize    ; dword-misalignment of string r13
;bestlen            equ    rsp + xx - LocalVarsSize    ; size of best match so far -> r11d
;scan              equ    rsp + xx - LocalVarsSize    ; ptr to string wanting match -> r9
IFDEF INFOZIP
ELSE
    nicematch        equ    (rsp + 16 - LocalVarsSize) ; a good enough match size
ENDIF

save_rdi            equ    rsp + 24 - LocalVarsSize
save_rsi            equ    rsp + 32 - LocalVarsSize
save_rbx            equ    rsp + 40 - LocalVarsSize
save_rbp            equ    rsp + 48 - LocalVarsSize
save_r12            equ    rsp + 56 - LocalVarsSize
save_r13            equ    rsp + 64 - LocalVarsSize
;save_r14            equ    rsp + 72 - LocalVarsSize
;save_r15            equ    rsp + 80 - LocalVarsSize
```

```
; all the +4 offsets are due to the addition of pending_buf_size (in zlib
; in the deflate_state structure since the asm code was first written
; (if you compile with zlib 1.0.4 or older, remove the +4).
; Note : these value are good with a 8 bytes boundary pack structure

MAX_MATCH          equ      258
MIN_MATCH          equ      3
MIN_LOOKAHEAD      equ      (MAX_MATCH+MIN_MATCH+1)

;;; Offsets for fields in the deflate_state structure. These numbers
;;; are calculated from the definition of deflate_state, with the
;;; assumption that the compiler will dword-align the fields. (Thus,
;;; changing the definition of deflate_state could easily cause this
;;; program to crash horribly, without so much as a warning at
;;; compile time. Sigh.)

; all the +zlib1222add offsets are due to the addition of fields
; in zlib in the deflate_state structure since the asm code was first written
; (if you compile with zlib 1.0.4 or older, use "zlib1222add equ (-4)").
; (if you compile with zlib between 1.0.5 and 1.2.2.1, use "zlib1222add equ 0").
; if you compile with zlib 1.2.2.2 or later , use "zlib1222add equ 8").
```

```
IFDEF INFOZIP
```

```
_DATA    SEGMENT
COMM     window_size:DWORD
; WMask ; 7fff
COMM     window:BYTE:010040H
COMM     prev:WORD:08000H
; MatchLen : unused
; PrevMatch : unused
COMM     strstart:DWORD
COMM     match_start:DWORD
; Lookahead : ignore
COMM     prev_length:DWORD ; PrevLen
COMM     max_chain_length:DWORD
COMM     good_match:DWORD
COMM     nice_match:DWORD
prev_ad equ OFFSET prev
window_ad equ OFFSET window
nicematch equ nice_match
_DATA ENDS
WMask equ 07ffff
```

```
ELSE
```

```
    IFNDEF zlib1222add
        zlib1222add equ 8
    ENDIF
dsWSize          equ 56+zlib1222add+(zlib1222add/2)
dsWMask          equ 64+zlib1222add+(zlib1222add/2)
dsWindow         equ 72+zlib1222add
dsPrev           equ 88+zlib1222add
dsMatchLen       equ 128+zlib1222add
dsPrevMatch      equ 132+zlib1222add
dsStrStart       equ 140+zlib1222add
dsMatchStart     equ 144+zlib1222add
dsLookahead      equ 148+zlib1222add
dsPrevLen        equ 152+zlib1222add
dsMaxChainLen    equ 156+zlib1222add
dsGoodMatch      equ 172+zlib1222add
dsNiceMatch      equ 176+zlib1222add

window_size      equ [ rcx + dsWSize]
WMask            equ [ rcx + dsWMask]
window_ad        equ [ rcx + dsWindow]
prev_ad          equ [ rcx + dsPrev]
strstart         equ [ rcx + dsStrStart]
match_start      equ [ rcx + dsMatchStart]
```

```

Lookahead      equ [ rcx + dsLookahead] ; 0ffffffffh on infozip
prev_length    equ [ rcx + dsPrevLen]
max_chain_length equ [ rcx + dsMaxChainLen]
good_match     equ [ rcx + dsGoodMatch]
nice_match     equ [ rcx + dsNiceMatch]
ENDIF

; parameter 1 in r8(deflate state s), param 2 in rdx (cur match)

; see http://weblogs.asp.net/oldnewthing/archive/2004/01/14/58579.aspx and
; http://msdn.microsoft.com/library/en-us/kmarch/hh/kmarch/64bitAMD_8e951dd2-ee77-4728-87
02-55ce4b5dd24a.xml.asp
;
; All registers must be preserved across the call, except for
;   rax, rcx, rdx, r8, r9, r10, and r11, which are scratch.

;;; Save registers that the compiler may be using, and adjust esp to
;;; make room for our stack frame.

;;; Retrieve the function arguments. r8d will hold cur_match
;;; throughout the entire function. edx will hold the pointer to the
;;; deflate_state structure during the function's setup (before
;;; entering the main loop.

; parameter 1 in rcx (deflate_state* s), param 2 in edx -> r8 (cur match)
; this clear high 32 bits of r8, which can be garbage in both r8 and rdx

        mov [save_rdi],rdi
        mov [save_rsi],rsi
        mov [save_rbx],rbx
        mov [save_rbp],rbp
IFDEF INFOZIP
        mov r8d,ecx
ELSE
        mov r8d,edx
ENDIF
        mov [save_r12],r12
        mov [save_r13],r13
;        mov [save_r14],r14
;        mov [save_r15],r15

;;; uInt wmask = s->w_mask;
;;; unsigned chain_length = s->max_chain_length;
;;; if (s->prev_length >= s->good_match) {
;;;     chain_length >= 2;
;;; }

        mov edi, prev_length
        mov esi, good_match
        mov eax, WMask
        mov ebx, max_chain_length
        cmp edi, esi
        jl  LastMatchGood
        shr ebx, 2
LastMatchGood:

;;; chainlen is decremented once beforehand so that the function can
;;; use the sign flag instead of the zero flag for the exit test.
;;; It is then shifted into the high word, to make room for the wmask
;;; value, which it will always accompany.

        dec ebx
        shl ebx, 16
        or  ebx, eax

;;; on zlib only
;;; if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;

IFDEF INFOZIP

```



```
        mov [chainlenwmask], ebx
; on infozip nice_match = [nice_match]
ELSE
        mov eax, nice_match
        mov [chainlenwmask], ebx
        mov r10d, Lookahead
        cmp r10d, eax
        cmovnl r10d, eax
        mov [nicematch], r10d
ENDIF

;;; register Bytef *scan = s->window + s->strstart;
        mov r10, window_ad
        mov ebp, strstart
        lea r13, [r10 + rbp]

;;; Determine how many bytes the scan ptr is off from being
;;; dword-aligned.

        mov r9, r13
        neg r13
        and r13, 3

;;; IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
;;; s->strstart - (IPos)MAX_DIST(s) : NIL;
IFDEF INFOZIP
        mov eax, 07efah ; MAX_DIST = (WSIZE-MIN_LOOKAHEAD) (0x8000-(3+8+1))
ELSE
        mov eax, window_size
        sub eax, MIN_LOOKAHEAD
ENDIF
        xor edi, edi
        sub ebp, eax

        mov r11d, prev_length
        cmovng ebp, edi

;;; int best_len = s->prev_length;

;;; Store the sum of s->window + best_len in esi locally, and in esi.

        lea rsi, [r10+r11]

;;; register ush scan_start = *(ushf*)scan;
;;; register ush scan_end  = *(ushf*)(scan+best_len-1);
;;; Posf *prev = s->prev;

        movzx r12d, word ptr [r9]
        movzx ebx, word ptr [r9 + r11 - 1]

        mov rdi, prev_ad

;;; Jump into the main loop.

        mov edx, [chainlenwmask]

        cmp bx, word ptr [rsi + r8 - 1]
        jz  LookupLoopIsZero

LookupLoop1:
        and r8d, edx

        movzx r8d, word ptr [rdi + r8*2]
        cmp r8d, ebp
        jbe LeaveNow
        sub edx, 00010000h
        js  LeaveNow

LoopEntry1:
        cmp bx, word ptr [rsi + r8 - 1]
        jz  LookupLoopIsZero
```

```
LookupLoop2:
    and r8d, edx

    movzx    r8d, word ptr [rdi + r8*2]
    cmp r8d, ebp
    jbe LeaveNow
    sub edx, 00010000h
    js  LeaveNow

LoopEntry2:
    cmp bx,word ptr [rsi + r8 - 1]
    jz  LookupLoopIsZero

LookupLoop4:
    and r8d, edx

    movzx    r8d, word ptr [rdi + r8*2]
    cmp r8d, ebp
    jbe LeaveNow
    sub edx, 00010000h
    js  LeaveNow

LoopEntry4:

    cmp bx,word ptr [rsi + r8 - 1]
    jnz LookupLoop1
    jmp LookupLoopIsZero

;;; do {
;;;     match = s->window + cur_match;
;;;     if (*(ushf*)(match+best_len-1) != scan_end ||
;;;         *(ushf*)match != scan_start) continue;
;;;     [...]
;;; } while ((cur_match = prev[cur_match & wmask]) > limit
;;;         && --chain_length != 0);
;;;
;;; Here is the inner loop of the function. The function will spend the
;;; majority of its time in this loop, and majority of that time will
;;; be spent in the first ten instructions.
;;;
;;; Within this loop:
;;; ebx = scanend
;;; r8d = curmatch
;;; edx = chainlenwmask - i.e., ((chainlen << 16) | wmask)
;;; esi = windowbestlen - i.e., (window + bestlen)
;;; edi = prev
;;; ebp = limit

LookupLoop:
    and r8d, edx

    movzx    r8d, word ptr [rdi + r8*2]
    cmp r8d, ebp
    jbe LeaveNow
    sub edx, 00010000h
    js  LeaveNow

LoopEntry:

    cmp bx,word ptr [rsi + r8 - 1]
    jnz LookupLoop1

LookupLoopIsZero:
    cmp     r12w, word ptr [r10 + r8]
    jnz LookupLoop1

;;; Store the current value of chainlen.
    mov [chainlenwmask], edx

;;; Point edi to the string under scrutiny, and esi to the string we
;;; are hoping to match it up with. In actuality, esi and edi are
;;; both pointed (MAX_MATCH_8 - scanalign) bytes ahead, and edx is
;;; initialized to -(MAX_MATCH_8 - scanalign).
```

```

    lea rsi,[r8+r10]
    mov rdx, 0ffffffffffffef8h; -(MAX_MATCH_8)
    lea rsi, [rsi + r13 + 0108h] ;MAX_MATCH_8]
    lea rdi, [r9 + r13 + 0108h] ;MAX_MATCH_8]

```

```

    prefetcht1 [rsi+rdx]
    prefetcht1 [rdi+rdx]

```

```

;;; Test the strings for equality, 8 bytes at a time. At the end,
;;; adjust rdx so that it is offset to the exact byte that mismatched.
;;;
;;; We already know at this point that the first three bytes of the
;;; strings match each other, and they can be safely passed over before
;;; starting the compare loop. So what this code does is skip over 0-3
;;; bytes, as much as necessary in order to dword-align the edi
;;; pointer. (rsi will still be misaligned three times out of four.)
;;;
;;; It should be confessed that this loop usually does not represent
;;; much of the total running time. Replacing it with a more
;;; straightforward "rep cmpsb" would not drastically degrade
;;; performance.

```

LoopCmps:

```

    mov rax, [rsi + rdx]
    xor rax, [rdi + rdx]
    jnz LeaveLoopCmps

    mov rax, [rsi + rdx + 8]
    xor rax, [rdi + rdx + 8]
    jnz LeaveLoopCmps8

    mov rax, [rsi + rdx + 8+8]
    xor rax, [rdi + rdx + 8+8]
    jnz LeaveLoopCmps16

    add rdx,8+8+8

```

```

    jmp short LoopCmps

```

LeaveLoopCmps16: add rdx,8

LeaveLoopCmps8: add rdx,8

LeaveLoopCmps:

```

    test    eax, 0000FFFFh
    jnz LenLower

```

```

    test eax,0ffffffffh

```

```

    jnz LenLower32

```

```

    add rdx,4
    shr rax,32
    or ax,ax
    jnz LenLower

```

LenLower32:

```

    shr eax,16
    add rdx,2

```

LenLower: sub al, 1

```

    adc rdx, 0

```

```

;;; Calculate the length of the match. If it is longer than MAX_MATCH,
;;; then automatically accept it as the best possible match and leave.

```

```

    lea rax, [rdi + rdx]
    sub rax, r9
    cmp eax, MAX_MATCH
    jge LenMaximum

```

```

;;; If the length of the match is not longer than the best match we
;;; have so far, then forget it and return to the lookup loop.
;////////////////////////////////////

```

```
    cmp eax, r11d
    jg  LongerMatch

    lea rsi,[r10+r11]

    mov rdi, prev_ad
    mov edx, [chainlenwmask]
    jmp LookupLoop

;;;      s->match_start = cur_match;
;;;      best_len = len;
;;;      if (len >= nice_match) break;
;;;      scan_end = *(ushf*)(scan+best_len-1);
LongerMatch:
    mov r11d, eax
    mov match_start, r8d
    cmp eax, [nicematch]
    jge LeaveNow

    lea rsi,[r10+rax]

    movzx ebx, word ptr [r9 + rax - 1]
    mov rdi, prev_ad
    mov edx, [chainlenwmask]
    jmp LookupLoop

;;; Accept the current string, with the maximum possible length.
LenMaximum:
    mov r11d,MAX_MATCH
    mov match_start, r8d

;;; if ((uInt)best_len <= s->lookahead) return (uInt)best_len;
;;; return s->lookahead;
LeaveNow:
IFDEF INFOZIP
    mov eax,r11d
ELSE
    mov eax, Lookahead
    cmp r11d, eax
    cmovng eax, r11d
ENDIF

;;; Restore the stack and return from whence we came.

    mov rsi,[save_rsi]
    mov rdi,[save_rdi]
    mov rbx,[save_rbx]
    mov rbp,[save_rbp]
    mov r12,[save_r12]
    mov r13,[save_r13]
;    mov r14,[save_r14]
;    mov r15,[save_r15]

    ret 0
; please don't remove this string !
; Your can freely use gvmat64 in any free or commercial app
; but it is far better don't remove the string in the binary!
    db      0dh,0ah,"asm686 with masm, optimised assembly code from Brian Raiter, written
1998, converted to amd 64 by Gilles Vollant 2005",0dh,0ah,0
longest_match    ENDP

match_init PROC
    ret 0
match_init ENDP

END
```

```
/* inffas8664.c is a hand tuned assembler version of inffast.c - fast decoding
 * version for AMD64 on Windows using Microsoft C compiler
 *
 * Copyright (C) 1995-2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 *
 * Copyright (C) 2003 Chris Anderson <christop@charm.net>
 * Please use the copyright conditions above.
 *
 * 2005 - Adaptation to Microsoft C Compiler for AMD64 by Gilles Vollant
 *
 * inffas8664.c call function inffas8664fnc in inffasx64.asm
 * inffasx64.asm is automatically convert from AMD64 portion of inffas86.c
 *
 * Dec-29-2003 -- I added AMD64 inflate asm support. This version is also
 * slightly quicker on x86 systems because, instead of using rep movsb to copy
 * data, it uses rep movsw, which moves data in 2-byte chunks instead of single
 * bytes. I've tested the AMD64 code on a Fedora Core 1 + the x86_64 updates
 * from http://fedora.linux.duke.edu/fc1\_x86\_64
 * which is running on an Athlon 64 3000+ / Gigabyte GA-K8VT800M system with
 * 1GB ram. The 64-bit version is about 4% faster than the 32-bit version,
 * when decompressing mozilla-source-1.3.tar.gz.
 *
 * Mar-13-2003 -- Most of this is derived from inffast.S which is derived from
 * the gcc -S output of zlib-1.2.0/inffast.c. Zlib-1.2.0 is in beta release at
 * the moment. I have successfully compiled and tested this code with gcc2.96,
 * gcc3.2, icc5.0, msvc6.0. It is very close to the speed of inffast.S
 * compiled with gcc -DNO_MMX, but inffast.S is still faster on the P3 with MMX
 * enabled. I will attempt to merge the MMX code into this version. Newer
 * versions of this and inffast.S can be found at
 * http://www.eetbeetee.com/zlib/ and http://www.charm.net/~christop/zlib/
 */
```

```
#include <stdio.h>
#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"
```

```
/* Mark Adler's comments from inffast.c: */
```

```
/*
Decode literal, length, and distance codes and write out the resulting
literal and match bytes until either not enough input or output is
available, an end-of-block is encountered, or a data error is encountered.
When large enough input and output buffers are supplied to inflate(), for
example, a 16K input buffer and a 64K output buffer, more than 95% of the
inflate execution time is spent in this routine.
```

Entry assumptions:

```
state->mode == LEN
strm->avail_in >= 6
strm->avail_out >= 258
start >= strm->avail_out
state->bits < 8
```

On return, state->mode is one of:

```
LEN -- ran out of enough output space or enough available input
TYPE -- reached end of block code, inflate() to interpret next block
BAD -- error in block data
```

Notes:

- The maximum input bits used by a length/distance pair is 15 bits for the length code, 5 bits for the length extra, 15 bits for the distance code, and 13 bits for the distance extra. This totals 48 bits, or six bytes. Therefore if strm->avail_in >= 6, then there is enough input to avoid checking for available input while decoding.
- The maximum bytes that a single length/distance pair can output is 258 bytes, which is the maximum length that can be coded. inflate_fast()

```

requires strm->avail_out >= 258 for each loop to avoid checking for
output space.
*/

typedef struct inffast_ar {
/* 64 32                                x86 x86_64 */
/* ar offset                            register */
/* 0 0 */ void *esp; /* esp save */
/* 8 4 */ void *ebp; /* ebp save */
/* 16 8 */ unsigned char FAR *in; /* esi rsi local strm->next_in */
/* 24 12 */ unsigned char FAR *last; /* r9 while in < last */
/* 32 16 */ unsigned char FAR *out; /* edi rdi local strm->next_out */
/* 40 20 */ unsigned char FAR *beg; /* inflate()'s init next_out */
/* 48 24 */ unsigned char FAR *end; /* r10 while out < end */
/* 56 28 */ unsigned char FAR *window; /* size of window, wsize!=0 */
/* 64 32 */ code const FAR *lcode; /* ebp rbp local strm->lencode */
/* 72 36 */ code const FAR *dcode; /* r11 local strm->distcode */
/* 80 40 */ size_t /*unsigned long */hold; /* edx rdx local strm->hold */
/* 88 44 */ unsigned bits; /* ebx rbx local strm->bits */
/* 92 48 */ unsigned wsize; /* window size */
/* 96 52 */ unsigned write; /* window write index */
/*100 56 */ unsigned lmask; /* r12 mask for lcode */
/*104 60 */ unsigned dmask; /* r13 mask for dcode */
/*108 64 */ unsigned len; /* r14 match length */
/*112 68 */ unsigned dist; /* r15 match distance */
/*116 72 */ unsigned status; /* set when state chng */
} type_ar;
#ifdef ASMINF

void inflate_fast(strm, start)
z_streamp strm;
unsigned start; /* inflate()'s starting value for strm->avail_out */
{
    struct inflate_state FAR *state;
    type_ar ar;
    void inffas8664fnc(struct inffast_ar * par);

#if (defined( __GNUC__ ) && defined( __amd64__ ) && ! defined( __i386__ )) || (defined(_MSC_
_VER) && defined(_M_AMD64))
#define PAD_AVAIL_IN 6
#define PAD_AVAIL_OUT 258
#else
#define PAD_AVAIL_IN 5
#define PAD_AVAIL_OUT 257
#endif

    /* copy state to local variables */
    state = (struct inflate_state FAR *)strm->state;

    ar.in = strm->next_in;
    ar.last = ar.in + (strm->avail_in - PAD_AVAIL_IN);
    ar.out = strm->next_out;
    ar.beg = ar.out - (start - strm->avail_out);
    ar.end = ar.out + (strm->avail_out - PAD_AVAIL_OUT);
    ar.wsize = state->wsize;
    ar.write = state->write;
    ar.window = state->window;
    ar.hold = state->hold;
    ar.bits = state->bits;
    ar.lcode = state->lencode;
    ar.dcode = state->distcode;
    ar.lmask = (1U << state->lenbits) - 1;
    ar.dmask = (1U << state->distbits) - 1;

    /* decode literals and length/distances until end-of-block or not enough
input data or output space */

    /* align in on 1/2 hold size boundary */
    while (((size_t)(void *)ar.in & (sizeof(ar.hold) / 2 - 1)) != 0) {
        ar.hold += (unsigned long)*ar.in++ << ar.bits;

```

```
    ar.bits += 8;
}

inffas8664fnc(&ar);

if (ar.status > 1) {
    if (ar.status == 2)
        strm->msg = "invalid literal/length code";
    else if (ar.status == 3)
        strm->msg = "invalid distance code";
    else
        strm->msg = "invalid distance too far back";
    state->mode = BAD;
}
else if ( ar.status == 1 ) {
    state->mode = TYPE;
}

/* return unused bytes (on entry, bits < 8, so in won't go too far back) */
ar.len = ar.bits >> 3;
ar.in -= ar.len;
ar.bits -= ar.len << 3;
ar.hold &= (1U << ar.bits) - 1;

/* update state and return */
strm->next_in = ar.in;
strm->next_out = ar.out;
strm->avail_in = (unsigned)(ar.in < ar.last ?
                           PAD_AVAIL_IN + (ar.last - ar.in) :
                           PAD_AVAIL_IN - (ar.in - ar.last));
strm->avail_out = (unsigned)(ar.out < ar.end ?
                             PAD_AVAIL_OUT + (ar.end - ar.out) :
                             PAD_AVAIL_OUT - (ar.out - ar.end));
state->hold = (unsigned long)ar.hold;
state->bits = ar.bits;
return;
}

#endif
```

```
; inffasx64.asm is a hand tuned assembler version of inffast.c - fast decoding
; version for AMD64 on Windows using Microsoft C compiler
;
; inffasx64.asm is automatically convert from AMD64 portion of inffas86.c
; inffasx64.asm is called by inffas8664.c, which contain more info.

; to compile this file, I use option
; ml64.exe /Flinffasx64 /c /Zi inffasx64.asm
; with Microsoft Macro Assembler (x64) for AMD64
;
; ml64.exe is given with Visual Studio 2005, Windows 2003 server DDK
;
; (you can get Windows 2003 server DDK with ml64 and cl.exe for AMD64 from
; http://www.microsoft.com/whdc/devtools/ddk/default.mspx for low price)
;

.code
inffas8664fnc PROC

; see http://weblogs.asp.net/oldnewthing/archive/2004/01/14/58579.aspx and
; http://msdn.microsoft.com/library/en-us/kmarch/hh/kmarch/64bitAMD_8e951dd2-ee77-4728-87
02-55ce4b5dd24a.xml.asp
;
; All registers must be preserved across the call, except for
; rax, rcx, rdx, r8, r-9, r10, and r11, which are scratch.

    mov [rsp-8],rsi
    mov [rsp-16],rdi
    mov [rsp-24],r12
    mov [rsp-32],r13
    mov [rsp-40],r14
    mov [rsp-48],r15
    mov [rsp-56],rbx

    mov rax,rcx

    mov [rax+8],rbp      ; /* save regs rbp and rsp */
    mov [rax],rsp

    mov rsp,rax          ; /* make rsp point to &ar */

    mov rsi,[rsp+16]      ; /* rsi = in */
    mov rdi,[rsp+32]      ; /* rdi = out */
    mov r9,[rsp+24]       ; /* r9 = last */
    mov r10,[rsp+48]      ; /* r10 = end */
    mov rbp,[rsp+64]      ; /* rbp = lcode */
    mov r11,[rsp+72]      ; /* r11 = dcode */
    mov rdx,[rsp+80]      ; /* rdx = hold */
    mov ebx,[rsp+88]      ; /* ebx = bits */
    mov r12d,[rsp+100]    ; /* r12d = lmask */
    mov r13d,[rsp+104]    ; /* r13d = dmask */
                          ; /* r14d = len */
                          ; /* r15d = dist */

    cld
    cmp r10,rdi
    je L_one_time         ; /* if only one decode left */
    cmp r9,rsi

jne L_do_loop

L_one_time:
    mov r8,r12            ; /* r8 = lmask */
    cmp bl,32
    ja L_get_length_code_one_time

    lodsd                 ; /* eax = *(uint *)in++ */
    mov cl,bl             ; /* cl = bits, needs it for shifting */
    add bl,32             ; /* bits += 32 */
    shl rax,cl
```



```
    or     rdx, rax          ; /* hold |= *((uint *)in)++ << bits */
    jmp    L_get_length_code_one_time

ALIGN 4
L_while_test:
    cmp     r10, rdi
    jbe     L_break_loop
    cmp     r9, rsi
    jbe     L_break_loop

L_do_loop:
    mov     r8, r12          ; /* r8 = lmask */
    cmp     bl, 32
    ja      L_get_length_code ; /* if (32 < bits) */

    lodsd                     ; /* eax = *(uint *)in++ */
    mov     cl, bl            ; /* cl = bits, needs it for shifting */
    add     bl, 32            ; /* bits += 32 */
    shl     rax, cl
    or      rdx, rax          ; /* hold |= *((uint *)in)++ << bits */

L_get_length_code:
    and     r8, rdx          ; /* r8 &= hold */
    mov     eax, [rbp+r8*4]   ; /* eax = lcode[hold & lmask] */

    mov     cl, ah            ; /* cl = this.bits */
    sub     bl, ah            ; /* bits -= this.bits */
    shr     rdx, cl           ; /* hold >>= this.bits */

    test    al, al
    jnz     L_test_for_length_base ; /* if (op != 0) 45.7% */

    mov     r8, r12          ; /* r8 = lmask */
    shr     eax, 16           ; /* output this.val char */
    stosb

L_get_length_code_one_time:
    and     r8, rdx          ; /* r8 &= hold */
    mov     eax, [rbp+r8*4] ; /* eax = lcode[hold & lmask] */

L_dolen:
    mov     cl, ah            ; /* cl = this.bits */
    sub     bl, ah            ; /* bits -= this.bits */
    shr     rdx, cl           ; /* hold >>= this.bits */

    test    al, al
    jnz     L_test_for_length_base ; /* if (op != 0) 45.7% */

    shr     eax, 16           ; /* output this.val char */
    stosb
    jmp     L_while_test

ALIGN 4
L_test_for_length_base:
    mov     r14d, eax         ; /* len = this */
    shr     r14d, 16          ; /* len = this.val */
    mov     cl, al

    test    al, 16
    jz      L_test_for_second_level_length ; /* if ((op & 16) == 0) 8% */
    and     cl, 15             ; /* op &= 15 */
    jz      L_decode_distance ; /* if (!op) */

L_add_bits_to_len:
    sub     bl, cl
    xor     eax, eax
    inc     eax
    shl     eax, cl
    dec     eax
    and     eax, edx          ; /* eax &= hold */
    shr     rdx, cl
    add     r14d, eax          ; /* len += hold & mask[op] */

L_decode_distance:
```

```

    mov     r8, r13                ; /* r8 = dmask */
    cmp     bl, 32
    ja      L_get_distance_code    ; /* if (32 < bits) */

    lodsd                                ; /* eax = *(uint *)in++ */
    mov     cl, bl                  ; /* cl = bits, needs it for shifting */
    add     bl, 32                  ; /* bits += 32 */
    shl     rax, cl
    or      rdx, rax               ; /* hold |= *((uint *)in)++ << bits */

L_get_distance_code:
    and     r8, rdx                ; /* r8 &= hold */
    mov     eax, [r11+r8*4] ; /* eax = dcode[hold & dmask] */

L_dodist:
    mov     r15d, eax              ; /* dist = this */
    shr     r15d, 16               ; /* dist = this.val */
    mov     cl, ah
    sub     bl, ah                 ; /* bits -= this.bits */
    shr     rdx, cl                ; /* hold >>= this.bits */
    mov     cl, al                ; /* cl = this.op */

    test    al, 16                ; /* if ((op & 16) == 0) */
    jz      L_test_for_second_level_dist
    and     cl, 15                ; /* op &= 15 */
    jz      L_check_dist_one

L_add_bits_to_dist:
    sub     bl, cl
    xor     eax, eax
    inc     eax
    shl     eax, cl
    dec     eax                   ; /* (1 << op) - 1 */
    and     eax, edx              ; /* eax &= hold */
    shr     rdx, cl
    add     r15d, eax             ; /* dist += hold & ((1 << op) - 1) */

L_check_window:
    mov     r8, rsi                ; /* save in so from can use it's reg */
    mov     rax, rdi
    sub     rax, [rsp+40]          ; /* nbytes = out - beg */

    cmp     eax, r15d
    jb      L_clip_window         ; /* if (dist > nbytes) 4.2% */

    mov     ecx, r14d             ; /* ecx = len */
    mov     rsi, rdi
    sub     rsi, r15              ; /* from = out - dist */

    sar     ecx, 1
    jnc     L_copy_two            ; /* if len % 2 == 0 */

    rep     movsw
    mov     al, [rsi]
    mov     [rdi], al
    inc     rdi

    mov     rsi, r8               ; /* move in back to %rsi, toss from */
    jmp     L_while_test

L_copy_two:
    rep     movsw
    mov     rsi, r8               ; /* move in back to %rsi, toss from */
    jmp     L_while_test

ALIGN 4
L_check_dist_one:
    cmp     r15d, 1               ; /* if dist 1, is a memset */
    jne     L_check_window
    cmp     [rsp+40], rdi         ; /* if out == beg, outside window */
    je      L_check_window

    mov     ecx, r14d             ; /* ecx = len */
    mov     al, [rdi-1]

```

```
        mov     ah, al

        sar     ecx, 1
        jnc     L_set_two
        mov     [rdi], al
        inc     rdi

L_set_two:
        rep     stosw
        jmp     L_while_test

ALIGN 4
L_test_for_second_level_length:
        test    al, 64
        jnz     L_test_for_end_of_block ; /* if ((op & 64) != 0) */

        xor     eax, eax
        inc     eax
        shl     eax, cl
        dec     eax
        and     eax, edx ; /* eax &= hold */
        add     eax, r14d ; /* eax += len */
        mov     eax, [rbp+rax*4] ; /* eax = lcode[val+(hold&mask[op])]* */
        jmp     L_dolen

ALIGN 4
L_test_for_second_level_dist:
        test    al, 64
        jnz     L_invalid_distance_code ; /* if ((op & 64) != 0) */

        xor     eax, eax
        inc     eax
        shl     eax, cl
        dec     eax
        and     eax, edx ; /* eax &= hold */
        add     eax, r15d ; /* eax += dist */
        mov     eax, [r11+rax*4] ; /* eax = dcode[val+(hold&mask[op])]* */
        jmp     L_dodist

ALIGN 4
L_clip_window:
        mov     ecx, eax ; /* ecx = nbytes */
        mov     eax, [rsp+92] ; /* eax = wsize, prepare for dist cmp */
        neg     ecx ; /* nbytes = -nbytes */

        cmp     eax, r15d
        jnb     L_invalid_distance_too_far ; /* if (dist > wsize) */

        add     ecx, r15d ; /* nbytes = dist - nbytes */
        cmp     dword ptr [rsp+96], 0
        jne     L_wrap_around_window ; /* if (write != 0) */

        mov     rsi, [rsp+56] ; /* from = window */
        sub     eax, ecx ; /* eax -= nbytes */
        add     rsi, rax ; /* from += wsize - nbytes */

        mov     eax, r14d ; /* eax = len */
        cmp     r14d, ecx
        jbe     L_do_copy ; /* if (nbytes >= len) */

        sub     eax, ecx ; /* eax -= nbytes */
        rep     movsb
        mov     rsi, rdi
        sub     rsi, r15 ; /* from = &out[ -dist ] */
        jmp     L_do_copy

ALIGN 4
L_wrap_around_window:
        mov     eax, [rsp+96] ; /* eax = write */
        cmp     ecx, eax
        jbe     L_contiguous_in_window ; /* if (write >= nbytes) */

        mov     esi, [rsp+92] ; /* from = wsize */
        add     rsi, [rsp+56] ; /* from += window */
```

```
add     rsi, rax           ; /* from += write */
sub     rsi, rcx           ; /* from -= nbytes */
sub     ecx, eax           ; /* nbytes -= write */

mov     eax, r14d          ; /* eax = len */
cmp     eax, ecx
jbe     L_do_copy          ; /* if (nbytes >= len) */

sub     eax, ecx           ; /* len -= nbytes */
rep     movsb
mov     rsi, [rsp+56]      ; /* from = window */
mov     ecx, [rsp+96]      ; /* nbytes = write */
cmp     eax, ecx
jbe     L_do_copy          ; /* if (nbytes >= len) */

sub     eax, ecx           ; /* len -= nbytes */
rep     movsb
mov     rsi, rdi
sub     rsi, r15           ; /* from = out - dist */
jmp     L_do_copy
```

ALIGN 4

L_contiguous_in_window:

```
mov     rsi, [rsp+56]      ; /* rsi = window */
add     rsi, rax
sub     rsi, rcx           ; /* from += write - nbytes */

mov     eax, r14d          ; /* eax = len */
cmp     eax, ecx
jbe     L_do_copy          ; /* if (nbytes >= len) */

sub     eax, ecx           ; /* len -= nbytes */
rep     movsb
mov     rsi, rdi
sub     rsi, r15           ; /* from = out - dist */
jmp     L_do_copy          ; /* if (nbytes >= len) */
```

ALIGN 4

L_do_copy:

```
mov     ecx, eax           ; /* ecx = len */
rep     movsb

mov     rsi, r8            ; /* move in back to %esi, toss from */
jmp     L_while_test
```

L_test_for_end_of_block:

```
test    al, 32
jz      L_invalid_literal_length_code
mov     dword ptr [rsp+116], 1
jmp     L_break_loop_with_status
```

L_invalid_literal_length_code:

```
mov     dword ptr [rsp+116], 2
jmp     L_break_loop_with_status
```

L_invalid_distance_code:

```
mov     dword ptr [rsp+116], 3
jmp     L_break_loop_with_status
```

L_invalid_distance_too_far:

```
mov     dword ptr [rsp+116], 4
jmp     L_break_loop_with_status
```

L_break_loop:

```
mov     dword ptr [rsp+116], 0
```

L_break_loop_with_status:

; /* put in, out, bits, and hold back into ar and pop esp */

```
mov     [rsp+16], rsi      ; /* in */
mov     [rsp+32], rdi      ; /* out */
mov     [rsp+88], ebx      ; /* bits */
mov     [rsp+80], rdx      ; /* hold */
```

```
mov     rax, [rsp]        ; /* restore rbp and rsp */
```

```
    mov     rbp, [rsp+8]
    mov     rsp, rax

    mov     rsi,[rsp-8]
    mov     rdi,[rsp-16]
    mov     r12,[rsp-24]
    mov     r13,[rsp-32]
    mov     r14,[rsp-40]
    mov     r15,[rsp-48]
    mov     rbx,[rsp-56]

    ret 0
;          :
;          : "m" (ar)
;          : "memory", "%rax", "%rbx", "%rcx", "%rdx", "%rsi", "%rdi",
;          : "%r8", "%r9", "%r10", "%r11", "%r12", "%r13", "%r14", "%r15"
;          );

inffas8664fnc    ENDP
;_TEXT    ENDS
END
```

Summary

This directory contains ASM implementations of the functions longest_match() and inflate_fast(), for 64 bits x86 (both AMD64 and Intel EM64t), for use with Microsoft Macro Assembler (x64) for AMD64 and Microsoft C++ 64 bits.

gvmat64.asm is written by Gilles Vollant (2005), by using Brian Raiter 686/32 bits assembly optimized version from Jean-loup Gailly original longest_match function

inffasx64.asm and inffas8664.c were written by Chris Anderson, by optimizing original function from Mark Adler

Use instructions

Copy these files into the zlib source directory.

define ASMV and ASMINF in your project. Include inffas8664.c in your source tree, and inffasx64.obj and gvmat64.obj as object to link.

Build instructions

run bld_64.bat with Microsoft Macro Assembler (x64) for AMD64 (ml64.exe)

ml64.exe is given with Visual Studio 2005, Windows 2003 server DDK

You can get Windows 2003 server DDK with ml64 and cl for AMD64 from
<http://www.microsoft.com/whdc/devtools/ddk/default.mspx> for low price)

```
ml /coff /Zi /c /Flgvmat32.lst gvmat32.asm  
ml /coff /Zi /c /Flinffas32.lst inffas32.asm
```

```
; gvmat32.asm -- Asm portion of the optimized longest_match for 32 bits x86
; Copyright (C) 1995-1996 Jean-loup Gailly and Gilles Vollant.
; File written by Gilles Vollant, by modifying the longest_match
;   from Jean-loup Gailly in deflate.c
;
;   http://www.zlib.net
;   http://www.winimage.com/zLibDll
;   http://www.muppetlabs.com/~breadbox/software/assembly.html
;
; For Visual C++ 4.x and higher and ML 6.x and higher
;   ml.exe is in directory \MASM611C of Win95 DDK
;   ml.exe is also distributed in http://www.masm32.com/masmdl.htm
;   and in VC++2003 toolkit at http://msdn.microsoft.com/visualc/vctoolkit2003/
;
; this file contain two implementation of longest_match
;
;   longest_match_7fff : written 1996 by Gilles Vollant optimized for
;       first Pentium. Assume s->w_mask == 0x7fff
;   longest_match_686 : written by Brian raiter (1998), optimized for Pentium Pro
;
; for using an seembly version of longest_match, you need define ASMV in project
; There is two way in using gvmat32.asm
;
; A) Suggested method
;   if you want include both longest_match_7fff and longest_match_686
;   compile the asm file running
;       ml /coff /Zi /Flgvmat32.lst /c gvmat32.asm
;   and include gvmat32c.c in your project
;   if you have an old cpu (386,486 or first Pentium) and s->w_mask==0x7fff,
;       longest_match_7fff will be used
;   if you have a more modern CPU (Pentium Pro, II and higher)
;       longest_match_686 will be used
;   on old cpu with s->w_mask!=0x7fff, longest_match_686 will be used,
;       but this is not a sitation you'll find often
;
; B) Alternative
;   if you are not interresed in old cpu performance and want the smaller
;       binaries possible
;
;   compile the asm file running
;       ml /coff /Zi /c /Flgvmat32.lst /DNOOLDPENTIUMCODE gvmat32.asm
;   and do not include gvmat32c.c in your project (ou define also
;       NOOLDPENTIUMCODE)
;
; note : as I known, longest_match_686 is very faster than longest_match_7fff
;       on pentium Pro/II/III, faster (but less) in P4, but it seem
;       longest_match_7fff can be faster (very very litte) on AMD Athlon64/K8
;
; see below : zlib1222add must be adjuster if you use a zlib version < 1.2.2.2

;uInt longest_match_7fff(s, cur_match)
;   deflate_state *s;
;   IPos cur_match;                               /* current match */

      NbStack      equ      76
      cur_match     equ      dword ptr[esp+NbStack-0]
      str_s         equ      dword ptr[esp+NbStack-4]
; 5 dword on top (ret,ebp,esi,edi,ebx)
      adrret        equ      dword ptr[esp+NbStack-8]
      pushebp       equ      dword ptr[esp+NbStack-12]
      pushedi       equ      dword ptr[esp+NbStack-16]
      pushesesi     equ      dword ptr[esp+NbStack-20]
      pushebx       equ      dword ptr[esp+NbStack-24]

      chain_length  equ      dword ptr [esp+NbStack-28]
      limit         equ      dword ptr [esp+NbStack-32]
      best_len      equ      dword ptr [esp+NbStack-36]
      window        equ      dword ptr [esp+NbStack-40]
      prev          equ      dword ptr [esp+NbStack-44]
      scan_start    equ      word  ptr [esp+NbStack-48]
      wmask         equ      dword ptr [esp+NbStack-52]
      match_start_ptr equ      dword ptr [esp+NbStack-56]
      nice_match    equ      dword ptr [esp+NbStack-60]
      scan          equ      dword ptr [esp+NbStack-64]
```



```
windowlen      equ      dword ptr [esp+NbStack-68]
match_start    equ      dword ptr [esp+NbStack-72]
strend         equ      dword ptr [esp+NbStack-76]
NbStackAdd     equ      (NbStack-24)

.386p

name          gvmatch
.MODEL        FLAT

; all the +zlib1222add offsets are due to the addition of fields
; in zlib in the deflate_state structure since the asm code was first written
; (if you compile with zlib 1.0.4 or older, use "zlib1222add equ (-4)").
; (if you compile with zlib between 1.0.5 and 1.2.2.1, use "zlib1222add equ 0").
; if you compile with zlib 1.2.2.2 or later , use "zlib1222add equ 8").

zlib1222add     equ      8

; Note : these value are good with a 8 bytes boundary pack structure
dep_chain_length equ      74h+zlib1222add
dep_window      equ      30h+zlib1222add
dep_strstart    equ      64h+zlib1222add
dep_prev_length equ      70h+zlib1222add
dep_nice_match   equ      88h+zlib1222add
dep_w_size      equ      24h+zlib1222add
dep_prev        equ      38h+zlib1222add
dep_w_mask      equ      2ch+zlib1222add
dep_good_match   equ      84h+zlib1222add
dep_match_start equ      68h+zlib1222add
dep_lookahead   equ      6ch+zlib1222add

_TEXT          segment

IFDEF NOUNDERLINE
    IFDEF NOOLDPENTIUMCODE
        public longest_match
        public match_init
    ELSE
        public longest_match_7fff
        public cpudetect32
        public longest_match_686
    ENDIF
ELSE
    IFDEF NOOLDPENTIUMCODE
        public _longest_match
        public _match_init
    ELSE
        public _longest_match_7fff
        public _cpudetect32
        public _longest_match_686
    ENDIF
ENDIF

MAX_MATCH      equ      258
MIN_MATCH      equ      3
MIN_LOOKAHEAD  equ      (MAX_MATCH+MIN_MATCH+1)

IFDEF NOOLDPENTIUMCODE
IFDEF NOUNDERLINE
longest_match_7fff proc near
ELSE
_longest_match_7fff proc near
ENDIF
    mov     edx,[esp+4]
```

```

    push    ebp
    push    edi
    push    esi
    push    ebx

    sub     esp,NbStackAdd

; initialize or check the variables used in match.asm.
    mov     ebp,edx

; chain_length = s->max_chain_length
; if (prev_length>=good_match) chain_length >= 2
    mov     edx,[ebp+dep_chain_length]
    mov     ebx,[ebp+dep_prev_length]
    cmp     [ebp+dep_good_match],ebx
    ja      noshr
    shr     edx,2
noshr:
; we increment chain_length because in the asm, the --chain_lenght is in the beginning of
the loop
    inc     edx
    mov     edi,[ebp+dep_nice_match]
    mov     chain_length,edx
    mov     eax,[ebp+dep_lookahead]
    cmp     eax,edi
; if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;
    jae     noloookaheadnicematch
    mov     edi,eax
noloookaheadnicematch:
; best_len = s->prev_length
    mov     best_len,ebx

; window = s->window
    mov     esi,[ebp+dep_window]
    mov     ecx,[ebp+dep_strstart]
    mov     window,esi

    mov     nice_match,edi
; scan = window + strstart
    add     esi,ecx
    mov     scan,esi
; dx = *window
    mov     dx,word ptr [esi]
; bx = *(window+best_len-1)
    mov     bx,word ptr [esi+ebx-1]
    add     esi,MAX_MATCH-1
; scan_start = *scan
    mov     scan_start,dx
; strend = scan + MAX_MATCH-1
    mov     strend,esi
; bx = scan_end = *(window+best_len-1)

;   IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
;   s->strstart - (IPos)MAX_DIST(s) : NIL;

    mov     esi,[ebp+dep_w_size]
    sub     esi,MIN_LOOKAHEAD
; here esi = MAX_DIST(s)
    sub     ecx,esi
    ja      nodist
    xor     ecx,ecx
nodist:
    mov     limit,ecx

; prev = s->prev
    mov     edx,[ebp+dep_prev]
    mov     prev,edx

;
    mov     edx,dword ptr [ebp+dep_match_start]
    mov     bp,scan_start
    mov     eax,cur_match
    mov     match_start,edx

```

```

    mov     edx,window
    mov     edi,edx
    add     edi,best_len
    mov     esi,prev
    dec     edi
; windowlen = window + best_len -1
    mov     windowlen,edi

    jmp     beginloop2
    align   4

; here, in the loop
;     eax = ax = cur_match
;     ecx = limit
;     bx = scan_end
;     bp = scan_start
;     edi = windowlen (window + best_len -1)
;     esi = prev

; // here; chain_length <=16
normalbeg0add16:
    add     chain_length,16
    jz      exitloop
normalbeg0:
    cmp     word ptr[edi+eax],bx
    je      normalbeg2noroll
rcontlabnoroll:
; cur_match = prev[cur_match & wmask]
    and     eax,7fffh
    mov     ax,word ptr[esi+eax*2]
; if cur_match > limit, go to exitloop
    cmp     ecx,eax
    jnb     exitloop
; if --chain_length != 0, go to exitloop
    dec     chain_length
    jnz     normalbeg0
    jmp     exitloop

normalbeg2noroll:
; if (scan_start==(cur_match+window)) goto normalbeg2
    cmp     bp,word ptr[edx+eax]
    jne     rcontlabnoroll
    jmp     normalbeg2

contloop3:
    mov     edi>windowlen

; cur_match = prev[cur_match & wmask]
    and     eax,7fffh
    mov     ax,word ptr[esi+eax*2]
; if cur_match > limit, go to exitloop
    cmp     ecx,eax
jnbexitloopshort1:
    jnb     exitloop
; if --chain_length != 0, go to exitloop

; begin the main loop
beginloop2:
    sub     chain_length,16+1
; if chain_length <=16, don't use the unrolled loop
    jna     normalbeg0add16

do16:
    cmp     word ptr[edi+eax],bx
    je      normalbeg2dc0

macrocn    MACRO    lab
    and     eax,7fffh
    mov     ax,word ptr[esi+eax*2]
    cmp     ecx,eax
    jnb     exitloop
    cmp     word ptr[edi+eax],bx

```



```
normalbeg2dc12:
    normbeg short rcontloop12,12

normalbeg2dc13:
    normbeg short rcontloop13,13

normalbeg2dc14:
    normbeg short rcontloop14,14

normalbeg2dc15:
    normbeg short rcontloop15,15

normalbeg2dc10:
    normbeg rcontloop10,10

normalbeg2dc9:
    normbeg rcontloop9,9

normalbeg2dc8:
    normbeg rcontloop8,8

normalbeg2dc7:
    normbeg rcontloop7,7

normalbeg2dc6:
    normbeg rcontloop6,6

normalbeg2dc5:
    normbeg rcontloop5,5

normalbeg2dc4:
    normbeg rcontloop4,4

normalbeg2dc3:
    normbeg rcontloop3,3

normalbeg2dc2:
    normbeg rcontloop2,2

normalbeg2dc1:
    normbeg rcontloop1,1

normalbeg2dc0:
    normbeg rcontloop0,0

; we go in normalbeg2 because *(ushf*)(match+best_len-1) == scan_end

normalbeg2:
    mov     edi,window

    cmp     bp,word ptr[edi+eax]
    jne     contloop3                ; if *(ushf*)match != scan_start, continue

iseq:
; if we are here, we know that *(match+best_len-1) == scan_end
; and (match == scan_start)

    mov     edi,edx
    mov     esi,scan                ; esi = scan
    add     edi,eax                 ; edi = window + cur_match = match

    mov     edx,[esi+3]             ; compare manually dword at match+3
    xor     edx,[edi+3]             ; and scan +3

    jz      begincompare            ; if equal, go to long compare

; we will determine the unmatched byte and calculate len (in esi)
    or      dl,dl
    je      eqlrr
    mov     esi,3
    jmp     trfinval
eqlrr:
    or      dx,dx
```

```
je      eq1
mov     esi,4
jmp     trfinval
eq1:
and     edx,0fffffffh
jz      eq11
mov     esi,5
jmp     trfinval
eq11:
mov     esi,6
jmp     trfinval

begincompare:
; here we now scan and match begin same
add     edi,6
add     esi,6
mov     ecx,(MAX_MATCH-(2+4))/4      ; scan for at most MAX_MATCH bytes
repe    cmpsd                       ; loop until mismatch

je      trfin                       ; go to trfin if not unmatched
; we determine the unmatched byte
sub     esi,4
mov     edx,[edi-4]
xor     edx,[esi]

or      dl,dl
jnz     trfin
inc     esi

or      dx,dx
jnz     trfin
inc     esi

and     edx,0fffffffh
jnz     trfin
inc     esi

trfin:
sub     esi,scan                    ; esi = len
trfinval:
; here we have finished compare, and esi contain len of equal string
cmp     esi,best_len                ; if len > best_len, go newbestlen
ja      short newbestlen
; now we restore edx, ecx and esi, for the big loop
mov     esi,prev
mov     ecx,limit
mov     edx>window
jmp     contloop3

newbestlen:
mov     best_len,esi                ; len become best_len

mov     match_start,eax              ; save new position as match_start
cmp     esi,nice_match               ; if best_len >= nice_match, exit
jae     exitloop
mov     ecx,scan
mov     edx>window                  ; restore edx=window
add     ecx,esi
add     esi,edx

dec     esi
mov     windowlen,esi               ; windowlen = window + best_len-1
mov     bx,[ecx-1]                  ; bx = *(scan+best_len-1) = scan_end

; now we restore ecx and esi, for the big loop :
mov     esi,prev
mov     ecx,limit
jmp     contloop3

exitloop:
; exit : s->match_start=match_start
mov     ebx,match_start
mov     ebp,str_s
```

```

    mov     ecx,best_len
    mov     dword ptr [ebp+dep_match_start],ebx
    mov     eax,dword ptr [ebp+dep_lookahead]
    cmp     ecx,eax
    ja      minexlo
    mov     eax,ecx
minexlo:
; return min(best_len,s->lookahead)

; restore stack and register ebx,esi,edi,ebp
    add     esp,NbStackAdd

    pop     ebx
    pop     esi
    pop     edi
    pop     ebp
    ret
InfoAuthor:
; please don't remove this string !
; Your are free use gvmat32 in any fre or commercial apps if you don't remove the string
in the binary!
    db      0dh,0ah,"GVMat32 optimised assembly code written 1996-98 by Gilles Vollant",0d
h,0ah

IFDEF NOUNDERLINE
longest_match_7fff    endp
ELSE
_longest_match_7fff    endp
ENDIF

IFDEF NOUNDERLINE
cpudetect32          proc near
ELSE
_cpudetect32          proc near
ENDIF

    push     ebx

    pushfd                    ; push original EFLAGS
    pop      eax              ; get original EFLAGS
    mov     ecx, eax          ; save original EFLAGS
    xor     eax, 40000h        ; flip AC bit in EFLAGS
    push     eax              ; save new EFLAGS value on stack
    popfd                    ; replace current EFLAGS value
    pushfd                    ; get new EFLAGS
    pop      eax              ; store new EFLAGS in EAX
    xor     eax, ecx           ; canM-^Rt toggle AC bit, processor=80386
    jz      end_cpu_is_386    ; jump if 80386 processor
    push     ecx
    popfd                    ; restore AC bit in EFLAGS first

    pushfd
    pushfd
    pop      ecx

    mov     eax, ecx          ; get original EFLAGS
    xor     eax, 200000h      ; flip ID bit in EFLAGS
    push     eax              ; save new EFLAGS value on stack
    popfd                    ; replace current EFLAGS value
    pushfd                    ; get new EFLAGS
    pop      eax              ; store new EFLAGS in EAX
    popfd                    ; restore original EFLAGS
    xor     eax, ecx           ; canM-^Rt toggle ID bit,
    je      is_old_486        ; processor=old

    mov     eax,1
    db      0fh,0a2h          ;CPUID

exitcpudetect:
    pop     ebx
    ret

```

```
end_cpu_is_386:
    mov     eax,0300h
    jmp     exitcpudetect

is_old_486:
    mov     eax,0400h
    jmp     exitcpudetect

IFDEF NOUNDERLINE
cpudetect32     endp
ELSE
_cpudetect32    endp
ENDIF
ENDIF

MAX_MATCH      equ     258
MIN_MATCH      equ     3
MIN_LOOKAHEAD  equ     (MAX_MATCH + MIN_MATCH + 1)
MAX_MATCH_8_   equ     ((MAX_MATCH + 7) AND 0FFF0h)

;;; stack frame offsets

chainlenwmask  equ     esp + 0      ; high word: current chain len
                ; low word: s->wmask
window         equ     esp + 4      ; local copy of s->window
windowbestlen  equ     esp + 8      ; s->window + bestlen
scanstart      equ     esp + 16     ; first two bytes of string
scanend        equ     esp + 12     ; last two bytes of string
scanalign      equ     esp + 20     ; dword-misalignment of string
nicematch      equ     esp + 24     ; a good enough match size
bestlen        equ     esp + 28     ; size of best match so far
scan           equ     esp + 32     ; ptr to string wanting match

LocalVarsSize  equ 36
;   saved ebx    byte esp + 36
;   saved edi    byte esp + 40
;   saved esi    byte esp + 44
;   saved ebp    byte esp + 48
;   return address byte esp + 52
deflatestate   equ     esp + 56     ; the function arguments
curmatch       equ     esp + 60

;;; Offsets for fields in the deflate_state structure. These numbers
;;; are calculated from the definition of deflate_state, with the
;;; assumption that the compiler will dword-align the fields. (Thus,
;;; changing the definition of deflate_state could easily cause this
;;; program to crash horribly, without so much as a warning at
;;; compile time. Sigh.)

dsWSize        equ 36+zlib1222add
dsWMask        equ 44+zlib1222add
dsWindow       equ 48+zlib1222add
dsPrev         equ 56+zlib1222add
dsMatchLen     equ 88+zlib1222add
dsPrevMatch    equ 92+zlib1222add
dsStrStart     equ 100+zlib1222add
dsMatchStart   equ 104+zlib1222add
dsLookahead    equ 108+zlib1222add
dsPrevLen      equ 112+zlib1222add
dsMaxChainLen  equ 116+zlib1222add
dsGoodMatch    equ 132+zlib1222add
dsNiceMatch    equ 136+zlib1222add

;;; match.asm -- Pentium-Pro-optimized version of longest_match()
;;; Written for zlib 1.1.2
;;; Copyright (C) 1998 Brian Raiter <breadbox@muppetlabs.com>
;;; You can look at http://www.muppetlabs.com/~breadbox/software/assembly.html
;;;
;;; This is free software; you can redistribute it and/or modify it
;;; under the terms of the GNU General Public License.
```



```
;GLOBAL _longest_match, _match_init

;SECTION      .text

;;; uInt longest_match(deflate_state *deflatestate, IPos curmatch)

;_longest_match:
IFDEF NOOLDPENTIUMCODE
    IFDEF NOUNDERLINE
        longest_match      proc near
    ELSE
        _longest_match      proc near
    ENDIF
ELSE
    IFDEF NOUNDERLINE
        longest_match_686    proc near
    ELSE
        _longest_match_686    proc near
    ENDIF
ENDIF
ENDIF

;;; Save registers that the compiler may be using, and adjust esp to
;;; make room for our stack frame.

        push    ebp
        push    edi
        push    esi
        push    ebx
        sub     esp, LocalVarsSize

;;; Retrieve the function arguments. ecx will hold cur_match
;;; throughout the entire function. edx will hold the pointer to the
;;; deflate_state structure during the function's setup (before
;;; entering the main loop.

        mov     edx, [deflatestate]
        mov     ecx, [curmatch]

;;; uInt wmask = s->w_mask;
;;; unsigned chain_length = s->max_chain_length;
;;; if (s->prev_length >= s->good_match) {
;;;     chain_length >>= 2;
;;; }

        mov     eax, [edx + dsPrevLen]
        mov     ebx, [edx + dsGoodMatch]
        cmp     eax, ebx
        mov     eax, [edx + dsWMask]
        mov     ebx, [edx + dsMaxChainLen]
        jl      LastMatchGood
        shr     ebx, 2
LastMatchGood:

;;; chainlen is decremented once beforehand so that the function can
;;; use the sign flag instead of the zero flag for the exit test.
;;; It is then shifted into the high word, to make room for the wmask
;;; value, which it will always accompany.

        dec     ebx
        shl     ebx, 16
        or      ebx, eax
        mov     [chainlenwmask], ebx

;;; if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;

        mov     eax, [edx + dsNiceMatch]
        mov     ebx, [edx + dsLookahead]
        cmp     ebx, eax
        jl      LookaheadLess
        mov     ebx, eax
LookaheadLess:  mov     [nicematch], ebx

;;; register Bytef *scan = s->window + s->strstart;
```

```
    mov esi, [edx + dsWindow]
    mov [window], esi
    mov ebp, [edx + dsStrStart]
    lea edi, [esi + ebp]
    mov [scan], edi

;;; Determine how many bytes the scan ptr is off from being
;;; dword-aligned.

    mov eax, edi
    neg eax
    and eax, 3
    mov [scanalign], eax

;;; IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
;;;     s->strstart - (IPos)MAX_DIST(s) : NIL;

    mov eax, [edx + dsWSzSize]
    sub eax, MIN_LOOKAHEAD
    sub ebp, eax
    jg LimitPositive
    xor ebp, ebp
LimitPositive:

;;; int best_len = s->prev_length;

    mov eax, [edx + dsPrevLen]
    mov [bestlen], eax

;;; Store the sum of s->window + best_len in esi locally, and in esi.

    add esi, eax
    mov [windowbestlen], esi

;;; register ush scan_start = *(ushf*)scan;
;;; register ush scan_end   = *(ushf*)(scan+best_len-1);
;;; Posf *prev = s->prev;

    movzx ebx, word ptr [edi]
    mov [scanstart], ebx
    movzx ebx, word ptr [edi + eax - 1]
    mov [scanend], ebx
    mov edi, [edx + dsPrev]

;;; Jump into the main loop.

    mov edx, [chainlenwmask]
    jmp short LoopEntry

align 4

;;; do {
;;;     match = s->window + cur_match;
;;;     if (*(ushf*)(match+best_len-1) != scan_end ||
;;;         *(ushf*)match != scan_start) continue;
;;;     [...]
;;; } while ((cur_match = prev[cur_match & wmask]) > limit
;;;         && --chain_length != 0);
;;;
;;; Here is the inner loop of the function. The function will spend the
;;; majority of its time in this loop, and majority of that time will
;;; be spent in the first ten instructions.
;;;
;;; Within this loop:
;;; ebx = scanend
;;; ecx = curmatch
;;; edx = chainlenwmask - i.e., ((chainlen << 16) | wmask)
;;; esi = windowbestlen - i.e., (window + bestlen)
;;; edi = prev
;;; ebp = limit

LookupLoop:
    and ecx, edx
```

```
    movzx    ecx, word ptr [edi + ecx*2]
    cmp ecx, ebp
    jbe LeaveNow
    sub edx, 00010000h
    js  LeaveNow
LoopEntry:  movzx    eax, word ptr [esi + ecx - 1]
    cmp eax, ebx
    jnz LookupLoop
    mov eax, [window]
    movzx    eax, word ptr [eax + ecx]
    cmp eax, [scanstart]
    jnz LookupLoop

;;; Store the current value of chainlen.

    mov [chainlenwmask], edx

;;; Point edi to the string under scrutiny, and esi to the string we
;;; are hoping to match it up with. In actuality, esi and edi are
;;; both pointed (MAX_MATCH_8 - scanalign) bytes ahead, and edx is
;;; initialized to -(MAX_MATCH_8 - scanalign).

    mov esi, [window]
    mov edi, [scan]
    add esi, ecx
    mov eax, [scanalign]
    mov edx, 0fffffff8h; -(MAX_MATCH_8)
    lea edi, [edi + eax + 0108h] ;MAX_MATCH_8]
    lea esi, [esi + eax + 0108h] ;MAX_MATCH_8]

;;; Test the strings for equality, 8 bytes at a time. At the end,
;;; adjust edx so that it is offset to the exact byte that mismatched.
;;;
;;; We already know at this point that the first three bytes of the
;;; strings match each other, and they can be safely passed over before
;;; starting the compare loop. So what this code does is skip over 0-3
;;; bytes, as much as necessary in order to dword-align the edi
;;; pointer. (esi will still be misaligned three times out of four.)
;;;
;;; It should be confessed that this loop usually does not represent
;;; much of the total running time. Replacing it with a more
;;; straightforward "rep cmpsb" would not drastically degrade
;;; performance.

LoopCmps:
    mov eax, [esi + edx]
    xor eax, [edi + edx]
    jnz LeaveLoopCmps
    mov eax, [esi + edx + 4]
    xor eax, [edi + edx + 4]
    jnz LeaveLoopCmps4
    add edx, 8
    jnz LoopCmps
    jmp short LenMaximum
LeaveLoopCmps4: add edx, 4
LeaveLoopCmps: test    eax, 0000FFFFh
    jnz LenLower
    add edx, 2
    shr eax, 16
LenLower:    sub al, 1
    adc edx, 0

;;; Calculate the length of the match. If it is longer than MAX_MATCH,
;;; then automatically accept it as the best possible match and leave.

    lea eax, [edi + edx]
    mov edi, [scan]
    sub eax, edi
    cmp eax, MAX_MATCH
    jge LenMaximum

;;; If the length of the match is not longer than the best match we
;;; have so far, then forget it and return to the lookup loop.
```

```

    mov edx, [deflatestate]
    mov ebx, [bestlen]
    cmp eax, ebx
    jg LongerMatch
    mov esi, [windowbestlen]
    mov edi, [edx + dsPrev]
    mov ebx, [scanend]
    mov edx, [chainlenwmask]
    jmp LookupLoop

;;;      s->match_start = cur_match;
;;;      best_len = len;
;;;      if (len >= nice_match) break;
;;;      scan_end = *(ushf*)(scan+best_len-1);
LongerMatch:    mov ebx, [nicematch]
                mov [bestlen], eax
                mov [edx + dsMatchStart], ecx
                cmp eax, ebx
                jge LeaveNow
                mov esi, [window]
                add esi, eax
                mov [windowbestlen], esi
                movzx ebx, word ptr [edi + eax - 1]
                mov edi, [edx + dsPrev]
                mov [scanend], ebx
                mov edx, [chainlenwmask]
                jmp LookupLoop

;;; Accept the current string, with the maximum possible length.
LenMaximum:    mov edx, [deflatestate]
                mov dword ptr [bestlen], MAX_MATCH
                mov [edx + dsMatchStart], ecx

;;; if ((uInt)best_len <= s->lookahead) return (uInt)best_len;
;;; return s->lookahead;
LeaveNow:
    mov edx, [deflatestate]
    mov ebx, [bestlen]
    mov eax, [edx + dsLookahead]
    cmp ebx, eax
    jg LookaheadRet
    mov eax, ebx
LookaheadRet:

;;; Restore the stack and return from whence we came.

    add esp, LocalVarsSize
    pop ebx
    pop esi
    pop edi
    pop ebp

    ret
; please don't remove this string !
; Your can freely use gvmat32 in any free or commercial app if you don't remove the string
; in the binary!
    db      0dh,0ah,"asm686 with masm, optimised assembly code from Brian Raiter, written
1998",0dh,0ah

IFDEF NOOLDPENTIUMCODE
    IFDEF NOUNDERLINE
        longest_match        endp
    ELSE
        _longest_match        endp
    ENDIF

    IFDEF NOUNDERLINE
        match_init            proc near
                                ret
        match_init            endp

```

```
ELSE
    _match_init    proc near
                    ret
    _match_init    endp
ENDIF
ELSE
    IFDEF NOUNDERLINE
        longest_match_686    endp
    ELSE
        _longest_match_686    endp
    ENDIF
ENDIF
_TEXT    ends
end
```

```
L^A^D^M~I.^FB~^]@^@i^@^@^@^@^@^@^.text^@^@^@^@^@^@^@^@^@D^G^@^@' ^@^@^@^@^@^@^@.  
^G^@^@^@^@]^A ^@0`.data^@^@^@D^G^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@  
À.debug$S^D^G^@^@^@^@^@^@|^G^@^@æ^O^@^@b^W^@^@^@^@^@^@M~T^@^@^@^@^@PB.debug$TM~^@N^@^@  
^@^@^@^@^@L^@^@^@^@^@*^]]^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@PBM~KT$^DUWVSM~Cî4M~KÊM~KU|M~K|x9  
M~^|M~L^L^@^@^@w^CÂêBBM~K½M~P^@^@M~IT$OM~Ket;çsBM~KØM~I\$(M~Ku8M~KMlM~It$$  
M~^I|$^P^CñM~^It$
```

fM-^K^VfM-^K\3ÿM-^AÆ^A^A^@^@fM-^IT\$^M-^I4\$M-^Ku,M-^Aî^F^A^@^@+îw^B3ÉM-^IL\$,M-^KU@M-^IT\$
M-^KUpfM-^Kl\$^M-^KD\$LM-^IT\$^DM-^KT\$M-^Kú^C|\$ (M-^Kt\$ OM-^I|\$^HëNM-^PM-^CD\$0^P^OM-^DÑ^C^@
^@f9^8t^%\%ÿ^?^@^@fM-^K^DF;È^OM-^C^C^@^@ÿL\$0uãé-^C^@^@f; , ^PuBéã^B^@^@M-^K|\$^H%ÿ^?^@^@f
M-^K^DF;È^OM-^CM-^O^C^@^@M-^Cl\$0Qv-f9^8^OM-^D-^B^@^@%ÿ^?^@^@fM-^K^DF;È^OM-^Cm^C^@^@f9^8^OM-^DM-^@^B^@^@%ÿ^?^@^@fM-^K^DF;È^OM-^CR^C^@^@f9^8^OM-^DT^B^@^@%ÿ^?^@^@fM-^K^DF;È^O
M-^C7^C^@^@f9^8^OM-^D(^B^@^@%ÿ^?^@^@fM-^K^DF;È^OM-^C^C^@^@f9^8^OM-^Dü^A^@^@%ÿ^?^@^@f
M-^K^DF;È^OM-^C^A^C^@^@f9^8^OM-^Dð^A^@^@%ÿ^?^@^@fM-^K^DF;È^OM-^Cæ^B^@^@f9^8^OM-^D□^A^@
^@%ÿ^?^@^@fM-^K^DF;È^OM-^CÈ^B^@^@f9^8^OM-^Du^A^@^@%ÿ^?^@^@fM-^K^DF;È^OM-^C°^B^@^@f9^8^O
M-^DF^A^@^@%ÿ^?^@^@fM-^K^DF;È^OM-^CM-^U^B^@^@f9^8^OM-^D^W^A^@^@%ÿ^?^@^@fM-^K^DF;È^OM-^Cz
^B^@^@f9^8^OM-^Dè^@^@%ÿ^?^@^@fM-^K^DF;È^OM-^C_ ^B^@^@f9^8t}%ÿ^?^@^@fM-^K^DF;È^OM-^CH^B
^@^@f9^8tz%ÿ^?^@^@fM-^K^DF;È^OM-^C1^B^@^@f9^8ts%ÿ^?^@^@fM-^K^DF;È^OM-^C^Z^B^@^@f9^8t1%
ÿ^?^@^@fM-^K^DF;È^OM-^C^C^B^@^@f9^8te%ÿ^?^@^@fM-^K^DF;È^OM-^Cì^A^@^@M-^Cl\$0^P^OM-^GYpÿÿé
^@pÿÿf; , ^P^OM-^EyÿÿÿM-^CD\$0^Eé^U^A^@^@f; , ^PuM-^@M-^CD\$0^Dé^E^A^@^@f; , ^PuM-^GM-^CD\$0^Céó^@
^@^@f; , ^PuM-^NM-^CD\$0^Béå^@^@^@f; , ^PuM-^UM-^CD\$0^AéÕ^@^@^@f; , ^P^OM-^E^NÿÿÿM-^CD\$0^FéÁ^@^@
^@f; , ^P^OM-^EßpÿÿM-^CD\$0^Gé-^@^@^@f; , ^P^OM-^E°pÿÿM-^CD\$0^HÉM-^Y^@^@^@f; , ^P^OM-^EM-^Abÿÿ
M-^CD\$0 éM-^E^@^@^@f; , ^P^OM-^ERpÿÿM-^CD\$0
ët f; , ^P^OM-^E&pÿÿM-^CD\$0^Këcf; , ^P^OM-^EúÿÿÿM-^CD\$0

```

ëRf ; , ^P^OM-^EîÿÿÿM-^CD$0
ëAf ; , ^P^OM-^EçÿÿÿM-^CD$0^Në0f ; , ^P^OM-^EvÿÿÿM-^CD$0^Oë^_f ; , ^P^OM-^EJÿÿÿM-^CD$0^Pë^NM-^K| $$
f ; , 8^OM-^E^OÿÿÿM-^KúM-^Kt$
    
```


^CøM-^KV^C3W^Ct-

Ôt^G¼^C^@^@^@ëQf^KÛt^G¼^D^@^@^@ëEM-^Aâÿÿÿ^@t^G¼^E^@^@^@ë6¼^F^@^@^@ë/M-^CÇ^FM-^CÆ^F¹?^@^@
^@ó§t^M-^Cî^DM-^KWü3^V

Ôu^Pff^KÛu

FM-^Aâÿÿÿ^@u^AF+t\$

```
;t$(w^QM-^Kt$ M-^KL$,M-^KT$$éM-^LüÿÿM-^It$(M-^ID$^D;t$^Ps"M-^KL$
```

M-^KT\$\$^CÎ^CòNM-^It\$^HfM-^KYÿM-^Kt\$ M-^KL\$,é\üÿÿM-^K\^\$DM-^Kl\$HM-^KL\$(M-^I]pM-^KEt;Ëw^B
M-^KÂM-^CÄ4[^_]Ã
GVMat32 optimised assembly code written 1996-98 by Gilles Vollant
SM-^XM-^KÈ5^@@^D^@PM-^]M-^X3Ãt^^QM-^]M-^M-^YM-^KÁ5^@@^ @PM-^]M-^XM-^]3Ãt^P,^A^@@
^@^Oç[Ã,^@^C^@^@ë÷,^@^D^@^@ëðUWVSM-^Cì\$M-^KT\$8M-^KL\$<M-^KBxM-^KM-^ZM-^L^@^@^@;ÃM-^KB4M-^K
Z|^CÃë^BKÃã^P^KØM-^I^\\$M-^KM-^BM-^P^@^@^@M-^KZt;Ø|^BM-^KØM-^I\\$^XM-^Kr8M-^It\$^DM-^Kjl
M-^M|5^@M-^I|\$ M-^KÇ÷ØM-^Cà^CM-^ID\$^TM-^KB,-^F^A^@^@+è^?^B3íM-^KBxM-^ID\$^\^CöM-^It\$^H^O•
^_M-^I\\$^P^O•\8ÿM-^I\\$

M-^Kz@M-^K^T\$ë^Z#Ê^O•

O;Í^OM-^Fà^@@^@M-^Aê^@@^A^@^OM-^HÔ^@@^@^O•D1ÿ;ÃuÝM-^KD\$^D^O•^D^A;D\$^PuİM-^I^T\$M-^Kt\$^D
M-^K|\$ ^CñM-^KD\$^T°øþÿÿM-^M¼8^H^A^@@M-^M´0^H^A^@@M-^K^D23^D:u^TM-^KD2^D3D:^Du^GM-^CÂ^Hu
éëqM-^CÂ^D@ÿÿ^@@u^FM-^CÂ^BÁè^P,^AM-^CÔ^@M-^M^D:M-^K|\$ +Ç=^B^A^@@^}LM-^KT\$8M-^K\\$\$^;Ã^?^S
M-^Kt\$^HM-^Kz@M-^K\\$\$

M-^K^T\$éNÿÿÿM-^K\,\$^XM-^ID\$^M-^IJp;Ã}-M-^Kt\$^D^CÖM-^It\$^H^O•\8ÿM-^Kz@M-^I\,\$

[illegible]

^@^K^@^@^@O^@
^@^@^@S^@P^@^@^@T^@S^@^@^@U^@Y^@^@^@V^@[^@^@^@W^@^@^@Z^@^@_ ^@^@^@[^@% ^@^@^@
^ \^@)^ ^@^@^@] ^@, ^@^@^@^@. ^@^@^@ ^@0^@^@^@! ^@2^@^@^@\$^@6^@^@^@' ^@9^@^@^@ (^@<^@^@^@) ^@^@^@
^@^@^@+^@D^@^@^@-^@F^@^@^@. ^@J^@^@^@0^@M^@^@^@2^@R^@^@^@3^@X^@^@^@5^@] ^@^@^@7^@' ^@^@^@=^@c
^@^@^@>^@i^@^@^@^@k^@^@^@A^@m^@^@^@B^@o^@^@^@D^@s^@^@^@G^@v^@^@^@H^@z^@^@^@K^@} ^@^@^@L^@
M-^@B^@^@^@M^@M-^@F^@^@^@N^@M-^@J^@^@^@P^@M-^@N^@^@^@Q^@M-^@P^@^@^@R^@M-^@T^@^@^@S^@M-^@X^@^@^@T
^@M-^@Y^@^@^@V^@M-^@] ^@^@^@X^@ ^@^@^@f^@¥^@^@^@g^@« ^@^@^@i^@-^@^@^@j^@±^@^@^@m^@¶^@^@^@n^@°
^@^@^@p^@4^@^@^@q^@Â^@^@^@s^@Æ^@^@^@t^@È^@^@^@u^@Í^@^@^@y^@Ñ^@^@^@z^@Ó^@^@^@{ ^@Ø^@^@^@~^@
Û^@^@^@M-^@A^@á^@^@^@M-^@B^@â^@^@^@M-^@D^@ç^@^@^@M-^@F^@í^@^@^@M-^@L^@ð^@^@^@M-^@N^@ô^@^@^@M-^@Q
^@ø^@^@^@M-^@R^@p^@^@^@M-^@^@^@Y^@A^@^@; ^@4^@A^@^@^@O^@A^@^@§^@j^@A^@^@^@M-^@E^@A^@^@-^@ ^@A^@^@
°^@» ^@A^@^@^@3^@Ö^@A^@^@¶^@ñ^@A^@^@^@1^@


```

^B^@^@1/4^@#^B^@^@;^@:^B^@^@Å^@Q^B^@^@Å^@h^B^@^@È^@?^B^@^@Ë^@M-^D^B^@^@Ì^@M-^H^B^@^@Í^@
M-^J^B^@^@Î^@M-^P^B^@^@Ë^@M-^U^B^@^@Ñ^@M-[^B^@^@Ï^@ ^B^@^@â^@ ^B^@^@ã^@Ä^B^@^@ä^@Ï^B^@^@
ë^@ä^B^@^@î^@ô^B^@^@ñ^@^H^C^@^@ô^@^C^@^@÷^@0^C^@^@ú^@D^C^@^@ý^@U^C^@^@^@Af^C^@^@^C^Aw
^C^@^@^F^AM-^H^C^@^@ ^AM-^Y^C^@^@

```

^A^C^@^@O^A»^C^@^@U^A¿^C^@^@W^AÃ^C^@^@X^AÉ^C^@^@^^^AË^C^@^@_ ^AĬ^C^@^@ ^AÑ^C^@^@"^AÔ
^C^@^@#^A×^C^@^@%^AÛ^C^@^@(^AÛ^C^@^@)^AÝ^C^@^@*^Aâ^C^@^@+^Aä^C^@^@-^Aç^C^@^@.^Aé^C^@^@0^A
î^C^@^@1^Að^C^@^@3^Aö^C^@^@4^Aø^C^@^@5^Aÿ^C^@^@6^Aÿ^C^@^@8^A^D^D^@^@9^A^F^D^@^@=^A
^D^@^@>^A

^D^@^@?^A^Q^D^@^@^A^S^D^@^@B^A^U^D^@^@D^A^X^D^@^@E^A^ [^D^@^@F^A^] ^D^@^@H^A^_ ^D^@^@I^A! ^D
^@^@J^A" ^D^@^@L^A% ^D^@^@M^A' ^D^@^@N^A(^D^@^@P^A. ^D^@^@Q^A0 ^D^@^@R^A1 ^D^@^@U^A5 ^D^@^@X^A9
^D^@^@Y^A; ^D^@^@[^A? ^D^@^@ \ ^AC^D^@^@] ^AG^D^@^@^AL^D^@^@a^AP^D^@^@c^AT^D^@^@d^AX^D^@^@e^A
Z^D^@^@f^A^D^@^@g^Ab^D^@^@h^Ad^D^@^@i^Af^D^@^@k^Ag^D^@^@l^Ak^D^@^@m^Ao^D^@^@p^As^D^@^@q
^Aw^D^@^@r^A | ^D^@^@v^AM- ^@^D^@^@w^AM- ^D^D^@^@x^AM- ^H^D^@^@y^AM- ^K^D^@^@z^AM- ^N^D^@^@ { ^A
M- ^P^D^@^@ | ^AM- ^R^D^@^@ } ^AM- ^T^D^@^@M- ^B^AM- ^W^D^@^@M- ^D^AM- ^X^D^@^@M- ^E^AM- ^Y^D^@^@M- ^F
^AM- ^Z^D^@^@M- ^G^AM- ^ [^D^@^@M- ^H^A^T^@^@^@^@á^D^@^@C^@â^D^@^@E^@ã^D^@^@F^@ä^D^@^@G
^@æ^D^@^@ë^D^@^@ ^@î^D^@^@
^@í^D^@^@K^@î^D^@^@

```

^@i^D^@^@
^@n^D^@^@N^@6^D^@^@O^@8^D^@^@P^@8^D^@^@R^@8^D^@^@S^@÷^D^@^@T^@8^D^@^@V^@ú^D^@^@W
^@ÿ^D^@^@X^@^@E^@^@Y^@^@A^@E^@^@Z^@^@B^@E^@^@[^@C^@E^@^@\^@D^@E^@^@]^@F^@E^@^@H^@E
^@^@^@O^@E^@^@S^@^@P^@E^@^@%^@Q^@E^@^@(^@V^@E^@^@)^@X^@E^@^@,^@^@]^@E^@^@-^@^@[^@^@^@^@^@_E
^@^@G^@^@E^@^@!^@E^@^@^@^@E^@^@
^@#^@E^@^@K^@^@&^@E^@^@R^@^@*^@E^@^@S^@^@.^@E^@^@[^@1^@E^@^@\^@7^@E^@^@]^@9^@E^@^@^@^@<^@E^@^@_
^@?^@E^@^@^@A^@E^@^@!^@D^@E^@^@)^@E^@^@*^@H^@E^@^@+^@J^@E^@^@,^@M^@E^@^@0^@S^@E^@^@1^@V^@E^@^@
2^@X^@E^@^@3^@Z^@E^@^@4^@^@E^@^@5^@^@E^@^@9^@c^@E^@^@:^@g^@E^@^@;^@j^@E^@^@<^@n^@E^@^@=^@r^@E^@
^@B^@t^@E^@^@C^@v^@E^@^@D^@y^@E^@^@E^@^@}^@E^@^@J^@M^@-^@E^@^@K^@M^@-^@E^@^@L^@M^@-^@G^@E^@^@M^@M^@-^@I
^@E^@^@N^@M^@-^@K^@E^@^@S^@M^@-^@N^@E^@^@T^@M^@-^@R^@E^@^@X^@M^@-^@T^@E^@^@Y^@M^@-^@X^@E^@^@_^@M^@-^@[^@E^@^@^@^@
M^@-^@_E^@^@a^@^@E^@^@b^@^@E^@^@c^@^@«^@E^@^@g^@^@E^@^@h^@^@E^@^@M^@-^@A^@^@2^@E^@^@M^@-^@B^@^@E^@^@
M^@-^@C^@^@,^@E^@^@M^@-^@D^@^@4^@E^@^@M^@-^@E^@^@Ä^@E^@^@M^@-^@F^@^@Ê^@E^@^@M^@-^@G^@^@Ï^@E^@^@M^@-^@H^@^@Ñ^@E^@^@M^@-^@I^@^@Ó^@E^@
^@M^@-^@J^@^@x^@E^@^@M^@-^@K^@^@Û^@E^@^@M^@-^@L^@^@ß^@E^@^@M^@-^@M^@^@á^@E^@^@M^@-^@Q^@^@ä^@E^@^@M^@-^@X^@^@è^@E^@^@M^@-^@Y^@^@ì^@E
^@^@M^@-^@Z^@^@î^@E^@^@M^@-^@[^@ð^@E^@^@M^@-^@\^@÷^@E^@^@M^@-^@]^@p^@E^@^@M^@-^@^@^@E^@F^@^@-^@^@H^@F^@^@^@^@K^@F
^@^@±^@^@
^@F^@^@2^@^@Q^@F^@^@3^@^@U^@F^@^@^@^@W^@F^@^@μ^@^@Z^@F^@^@¶^@^@\^@F^@^@•^@^@F^@^@,^@!^@F^@^@1^@^@&^@F
^@^@^@(^@F^@^@»^@+^@F^@^@4^@^@.^@F^@^@½^@0^@F^@^@¾^@3^@F^@^@Ä^@6^@F^@^@Ä^@:^@F^@^@Ä^@<^@F^@^@E^@A
^@F^@^@Ç^@C^@F^@^@Î^@G^@F^@^@Î^@K^@F^@^@Î^@M^@F^@^@Ï^@O^@F^@^@Ð^@S^@F^@^@Ñ^@V^@F^@^@Ò^@Z^@F^@^@Ó^@
]^@F^@^@Ô^@b^@F^@^@Û^@f^@F^@^@Ü^@j^@F^@^@Ý^@m^@F^@^@Þ^@o^@F^@^@ß^@q^@F^@^@à^@u^@F^@^@á^@w^@F^@^@â
^@{^@F^@^@ã^@M^@-^@F^@^@ä^@M^@-^@C^@F^@^@å^@M^@-^@G^@F^@^@æ^@M^@-^@J^@F^@^@ç^@M^@-^@O^@F^@^@è^@M^@-^@S^@F^@^@ì
^@M^@-^@[^@F^@^@í^@M^@-^@F^@^@ö^@ç^@F^@^@ô^@|^@F^@^@õ^@C^@F^@^@ö^@«^@F^@^@÷^@-^@F^@^@ø^@-^@F^@^@ý^@2
^@F^@^@p^@3^@F^@^@ÿ^@^@F^@^@^@Aμ^@F^@^@A¶^@F^@^@C^@A^@D^@^@^@ñ^@^@^@^@^@^@,^@^@A^@Q^@^@^@
^@O:\updasm\contrib\masmx86\gvmat32.obj^@4^@^@V^@Q^@C^@^@^@A^@^@^@^@^@^@G^@^@
^@^@E
    
```

1045/1462

```
^C^@^@7^@^@^@K^@P^C^@^@7^@^@^@
^@&^C^@^@8^@^@^@K^@*^C^@^@8^@^@^@
^@B^C^@^@9^@^@^@K^@F^C^@^@9^@^@^@
^@W^C^@^@: ^@^@^@K^@[ ^C^@^@: ^@^@^@
^@t^C^@^@; ^@^@^@K^@x^C^@^@; ^@^@^@
^@M-^J^C^@^@<^@^@^@K^@M-^N^C^@^@<^@^@^@
^@M-^Z^C^@^@=^@^@^@K^@M-^^^C^@^@=^@^@^@
^@^C^@^@>^@^@^@K^@' ^C^@^@>^@^@^@
^@Æ^C^@^@? ^@^@^@K^@Ê^C^@^@? ^@^@^@
^@Û^C^@^@^@^@^@K^@à^C^@^@^@^@^@
^@ö^C^@^@A^@^@^@K^@ö^C^@^@A^@^@^@
^@H^D^@^@B^@^@^@K^@
```

```
^D^@^@B^@^@^@
^@^@^@D^@^@C^@^@^@K^@"^D^@^@C^@^@^@
^@4^D^@^@D^@^@^@K^@8^D^@^@D^@^@^@
^@J^D^@^@E^@^@^@K^@N^D^@^@E^@^@^@
^@^D^@^@F^@^@^@K^@D^D^@^@F^@^@^@
^@v^D^@^@G^@^@^@K^@z^D^@^@G^@^@^@
^@M-^M^D^@^@H^@^@^@K^@M-^Q^D^@^@H^@^@^@
^@α^D^@^@I^@^@^@K^@^D^@^@I^@^@^@
^@»^D^@^@J^@^@^@K^@;^D^@^@J^@^@^@
^@Ö^D^@^@K^@^@^@K^@Ö^D^@^@K^@^@^@
^@é^D^@^@L^@^@^@K^@í^D^@^@L^@^@^@
^@^E^@^@M^@^@^@K^@D^E^@^@M^@^@^@
^@^Z^E^@^@N^@^@^@K^@^E^@^@N^@^@^@
^@4^E^@^@O^@^@^@K^@8^E^@^@O^@^@^@
^@N^E^@^@P^@^@^@K^@R^E^@^@P^@^@^@
^@h^E^@^@Q^@^@^@K^@l^E^@^@Q^@^@^@
^@M-^B^E^@^@R^@^@^@K^@M-^F^E^@^@R^@^@^@
^@M-^E^@^@S^@^@^@K^@ ^E^@^@S^@^@^@
^@u^E^@^@T^@^@^@K^@1^E^@^@T^@^@^@
^@Î^E^@^@U^@^@^@K^@Ï^E^@^@U^@^@^@
^@ç^E^@^@V^@^@^@K^@ë^E^@^@V^@^@^@
^@^F^@^@W^@^@^@K^@D^F^@^@W^@^@^@
^@Y^F^@^@X^@^@^@K^@^] ^F^@^@X^@^@^@
^@2^F^@^@Y^@^@^@K^@6^F^@^@Y^@^@^@
^@K^F^@^@Z^@^@^@K^@O^F^@^@Z^@^@^@
^@d^F^@^@[ ^@^@^@K^@h^F^@^@[ ^@^@^@
^@} ^F^@^@\ ^@^@^@K^@M-^A^F^@^@\ ^@^@^@
^@M-^V^F^@^@] ^@^@^@K^@M-^Z^F^@^@] ^@^@^@
^@¬^F^@^@^@^@^@K^@°^F^@^@^@^@^@
^@¼^F^@^@_ ^@^@^@K^@À^F^@^@_ ^@^@^@
^@Í^F^@^@` ^@^@^@K^@Ñ^F^@^@` ^@^@^@
^@Û^F^@^@a^@^@^@K^@à^F^@^@a^@^@^@
^@ì^F^@^@b^@^@^@K^@ð^F^@^@b^@^@^@
^@D^G^@^@c^@^@^@K^@H^G^@^@c^@^@^@
^@U^G^@^@d^@^@^@K^@Y^G^@^@d^@^@^@
^@)^G^@^@e^@^@^@K^@-^G^@^@e^@^@^@
^@?^G^@^@f^@^@^@K^@C^G^@^@f^@^@^@
^@S^G^@^@g^@^@^@K^@W^G^@^@g^@^@^@
^@f^G^@^@h^@^@^@K^@j^G^@^@h^@^@^@
^@D^@^@^@F^@^@N^@^@^@ðñ^N^@^@H^P^C^@^@M-^^^@^@^@B^P^@^@F^@^@A^R^@^@^@^@N^@^@H^P^C^@^@
^@} ^@^@^@D^P^@^@F^@^@A^R^@^@^@N^@^@H^P^C^@^@^@\ ^@^@^@F^P^@^@F^@^@A^R^@^@^@.file^@^@
^@^@^@^@^@pÿ^@^@g^CO:\updasm\contrib\masmx86\gvmat32.asm^@^@^@^@^@^@^@^@^@^@^@^@^@
^@comp.id^E
```

```

^O^@y^@^@C^@.text^@^@^@^@^@^@^@^@A^@^@^@^@C^@A^@D^@G^@^@^@^@]^@A^@^@^@^@^@^@^@^@^@^@.data^@^@
^@^@y^@^@^@B^@^@^@C^@A^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@.debug$$^@^@^@^@C^@^@^@^@C^@A|^@G
^@^@M^-^T^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@.debug$T^@^@^@^@D^@^@^@^@C^@A^@T^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@D^@^@^@^@^@^@^@^@A^@^@^@^@B^@A^@O^@^@^@^@á^@D^@^@^@^@G^@^@^@^@T^@^@^@^@^@^@^@^@.bf^@^@^@^@
^@^@^@^@^@^@A^@^@^@^@e^@A^@^@^@^@M^-^N^@^@^@^@^@^@^@^@V^@^@^@^@^@^@^@.lf^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@.ef^@^@^@^@^@^@á^@D^@^@^@^@A^@^@^@^@e^@A^@^@^@^@^@^@^@^@V^@B^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@X^@^@^@^@á^@D
^@^@^@A^@^@^@B^@A^@V^@^@^@^@^@^@^@^@^@^@^@Z

```


^@^@^ [^@^@^@^@^@.bf^@^@^@^@^@á^D^@^@^A^@^@^e^A^@^@^@^@ (^B^@^@^@^@^@^@^)^@^@^@^@^@.lf^@^@
^@^@^@!^@^@^@A^@^@^e^@.ef^@^@^@^@^@^ _^E^@^@^A^@^@^e^A^@^@^@^@U^B^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@% ^@^@^@^ _^E^@^@^A^@ ^@^B^A^]^@^@^@â^A^@^@à

^@^@^@^@^@^@^@.bf^@^@^@^@^@^@_E^@^@^@A^@^@^@e^A^@^@^@^@^@^@^@^@^@.lf^@
^@^@^@^@M-^A^@^@^@^@^@^@e^@.ef^@^@^@^@^@^@D^G^@^@^@A^@^@^@e^A^@^@^@^@-^C^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@8^@^@^@D^E^@^@^@A^@^@^@F^@^@^@^@F^@^@^@^@^@^@^@^@T^@^@
^@M-^K^E^@^@^@A^@^@^@F^@^@^@^@b^@^@^@^@E^@^@^@A^@^@^@F^@^@^@^@m^@^@^@E^@^@^@A^@^@^@
^F^@LoopCmps^E^F^@^@^@A^@^@^@F^@^@^@^@w^@^@^@^@F^@^@^@A^@^@^@F^@^@^@^@M-^F^@^@^@!^F
^@^@^@A^@^@^@F^@LenLower.^F^@^@^@A^@^@^@F^@^@^@^@M-^T^@^@^@b^F^@^@^@A^@^@^@F^@^@^@^@
^@^@^@M-^O^F^@^@^@A^@^@^@F^@LeaveNowM-^F^@^@^@A^@^@^@F^@^@^@^@^@^@^@^@^@F^@^@^@A^@^@^@
^F^@^@^@^@^@^@,^@^@^@O^E^@^@^@A^@^@^@F^@^@^@^@^@^@^@^@Q^E^@^@^@A^@^@^@F^@^@^@^@^@^@^@
^X^E^@^@^@A^@^@^@F^@noshr^@^@^@^@^@^@A^@^@^@F^@^@^@^@^@^@^@2^@^@^@A^@^@^@F^@^@^@^@
st^@^@^@^@^@^@A^@^@^@F^@^@^@^@^@^@^@^@^@^@^@A^@^@^@F^@^@^@^@^@^@^@^@E^@^@^@^@^@^@^@
^@F^@^@^@^@^@^@P^A^@^@^@^@^@^@A^@^@^@F^@^@^@^@^@^@_A^@^@^@I^@^@^@^@A^@^@^@F^@^@^@^@^@^@0^A^@
^@0^@^@^@A^@^@^@F^@^@^@^@^@^@^@^@^@^@^@A^@^@^@F^@^@^@^@^@^@L^A^@^@^@i^@^@^@A^@^@^@F^@
do16^@^@^@^@^@^@A^@^@^@F^@^@^@^@^@^@W^A^@^@^@p^@^@^@^@A^@^@^@F^@^@^@^@^@^@b^A^@^@^@Y^A^@^@
^A^@^@^@F^@^@^@^@^@^@m^A^@^@^@4^A^@^@^@A^@^@^@F^@^@^@^@^@^@x^A^@^@^@O^A^@^@^@A^@^@^@F^@^@^@^@^@
M-^C^A^@^@^@j^A^@^@^@A^@^@^@F^@^@^@^@^@^@M-^N^A^@^@M-^E^A^@^@^@A^@^@^@F^@^@^@^@^@^@M-^Y^A^@^@
^A^@^@^@A^@^@^@F^@^@^@^@^@^@A^@^@^@»^A^@^@^@A^@^@^@F^@^@^@^@^@^@^@^@O^A^@^@^@A^@^@^@F^@^@^@
^@^@^@^@A^@^@^@ñ^A^@^@^@A^@^@^@F^@^@^@^@^@^@A^@^@^@

```
^B^@^@^A^@^@^F^@^@^@Ñ^A^@^@#^B^@^@^A^@^@^F^@^@^@Ý^A^@^@: ^B^@^@^A^@^@^F^@^@^@
^@^@^@É^A^@^@Q^B^@^@^A^@^@^F^@^@^@Ö^A^@^@h^B^@^@^A^@^@^F^@^@^@^A^B^@^@^?^B^@
^@^A^@^@^F^@^@^@^@^@
^B^@^@ ^B^@^@^A^@^@^F^@^@^@^@^@^B^@^@^B^@^@^A^@^@^F^@^@^@^@^@^B^@^@^Ä^B^@^@^A^@^@
^@^F^@^@^@^@^@: ^B^@^@Ö^B^@^@^A^@^@^F^@^@^@^@^@I^B^@^@ä^B^@^@^A^@^@^F^@^@^@^@^@X^B^@^@
ö^B^@^@^A^@^@^F^@^@^@^@^@g^B^@^@H^C^@^@^A^@^@^F^@^@^@^@^@u^B^@^@^C^@^@^A^@^@^F^@
^@^@^@M^-^C^B^@^@0^C^@^@^A^@^@^F^@^@^@^@^@M^-^Q^B^@^@D^C^@^@^A^@^@^F^@^@^@^@^@M^-^_B
^@^@U^C^@^@^A^@^@^F^@^@^@^@^@-^B^@^@f^C^@^@^A^@^@^F^@^@^@^@^@»^B^@^@w^C^@^@^A^@^@^F
^@^@^@^@^@É^B^@^@M^-^H^C^@^@^A^@^@^F^@^@^@^@^@x^B^@^@M^-^Y^C^@^@^A^@^@^F^@^@^@^@^@ä^B^@
^@^C^@^@^A^@^@^F^@^@^@^@^@ö^B^@^@»^C^@^@^A^@^@^F^@^@^@^@^@É^C^@^@^A^@^@^F^@^@^@eq1
rr^@^@^@ä^C^@^@^A^@^@^F^@^@^@eq1^@^@^@^@^@ö^C^@^@^A^@^@^F^@^@^@eq11^@^@^@^@^@ÿ^C^@^@^A^@^@^F
^@^@^@^@^@p^B^@^@^F^D^@^@^A^@^@^F^@^@^@^@^@1^D^@^@^A^@^@^F^@^@^@^@^@5^D^@^@^A^@^@
^@^F^@^@^@^@^@K^C^@^@L^D^@^@^A^@^@^F^@^@^@^@^@exitloop|^D^@^@^A^@^@^F^@^@^@^@^@minexlo^M^-^T^D^@^@^A
^@^@^@^F^@^@^@^@^@V^C^@^@M^-^D^@^@^A^@^@^F^@^@^@^@^@_longest_match_7fff^@_cpudetect32
^@_longest_match_686^@LastMatchGood^@LookaheadLess^@LimitPositive^@LookupLoop^@LoopEntry
^@LeaveLoopCmps4^@LeaveLoopCmps^@LongerMatch^@LenMaximum^@LookaheadRet^@exitcpudetect^@en
d_cpu_is_386^@is_old_486^@nolookaheadnicematch^@normalbeg0add16^@normalbeg0^@rcontlabnoro
ll^@normalbeg2noroll^@contloop3^@jnbexitloopshort1^@beginloop2^@rcontloop0^@rcontloop1^@r
contloop2^@rcontloop3^@rcontloop4^@rcontloop5^@rcontloop6^@rcontloop7^@rcontloop8^@rcontl
oop9^@rcontloop10^@rcontloop11^@rcontloop12^@rcontloop13^@rcontloop14^@rcontloop15^@norma
lbeg2dc11^@normalbeg2dc12^@normalbeg2dc13^@normalbeg2dc14^@normalbeg2dc15^@normalbeg2dc10
^@normalbeg2dc9^@normalbeg2dc8^@normalbeg2dc7^@normalbeg2dc6^@normalbeg2dc5^@normalbeg2dc
4^@normalbeg2dc3^@normalbeg2dc2^@normalbeg2dc1^@normalbeg2dc0^@normalbeg2^@begincompare^@
newbestlen^@InfoAuthor^@
```

```
/* gvmat32.c -- C portion of the optimized longest_match for 32 bits x86
 * Copyright (C) 1995-1996 Jean-loup Gailly and Gilles Vollant.
 * File written by Gilles Vollant, by modifying the longest_match
 * from Jean-loup Gailly in deflate.c
 * it prepare all parameters and call the assembly longest_match_gvasm
 * longest_match execute standard C code is wmask != 0x7fff
 * (assembly code is faster with a fixed wmask)
 *
 * Read comment at beginning of gvmat32.asm for more information
 */
```

```
#if defined(ASMV) && (!defined(NOOLDPENTIUMCODE))
#include "deflate.h"
```

```
/* if your C compiler don't add underline before function name,
   define ADD_UNDERLINE_ASMFUNC */
```

```
#ifndef ADD_UNDERLINE_ASMFUNC
#define longest_match_7fff _longest_match_7fff
#define longest_match_686 _longest_match_686
#define cpudetect32 _cpudetect32
#endif
```

```
unsigned long cpudetect32();
```

```
uInt longest_match_c(
    deflate_state *s,
    IPos cur_match); /* current match */
```

```
uInt longest_match_7fff(
    deflate_state *s,
    IPos cur_match); /* current match */
```

```
uInt longest_match_686(
    deflate_state *s,
    IPos cur_match); /* current match */
```

```
static uInt iIsPPro=2;
```

```
void match_init ()
{
    iIsPPro = (((cpudetect32()/0x100)&0xf)>=6) ? 1 : 0;
}
```

```
uInt longest_match(
    deflate_state *s,
    IPos cur_match) /* current match */
{
    if (iIsPPro!=0)
        return longest_match_686(s,cur_match);

    if (s->w_mask != 0x7fff)
        return longest_match_686(s,cur_match);

    /* now ((s->w_mask == 0x7fff) && (iIsPPro==0)) */
    return longest_match_7fff(s,cur_match);
}
```

```
#endif /* defined(ASMV) && (!defined(NOOLDPENTIUMCODE)) */
```

```
;/* inffas32.asm is a hand tuned assembler version of inffast.c -- fast decoding
; *
; * inffas32.asm is derivated from inffas86.c, with translation of assembly code
; *
; * Copyright (C) 1995-2003 Mark Adler
; * For conditions of distribution and use, see copyright notice in zlib.h
; *
; * Copyright (C) 2003 Chris Anderson <christop@charm.net>
; * Please use the copyright conditions above.
; *
; * Mar-13-2003 -- Most of this is derived from inffast.S which is derived from
; * the gcc -S output of zlib-1.2.0/inffast.c. Zlib-1.2.0 is in beta release at
; * the moment. I have successfully compiled and tested this code with gcc2.96,
; * gcc3.2, icc5.0, msvc6.0. It is very close to the speed of inffast.S
; * compiled with gcc -DNO_MMX, but inffast.S is still faster on the P3 with MMX
; * enabled. I will attempt to merge the MMX code into this version. Newer
; * versions of this and inffast.S can be found at
; * http://www.eetbeetee.com/zlib/ and http://www.charm.net/~christop/zlib/
; *
; * 2005 : modification by Gilles Vollant
; */
; For Visual C++ 4.x and higher and ML 6.x and higher
; ml.exe is in directory \MASM611C of Win95 DDK
; ml.exe is also distributed in http://www.masm32.com/masmdl.htm
; and in VC++2003 toolkit at http://msdn.microsoft.com/visualc/vctoolkit2003/
;
; compile with command line option
; ml /coff /Zi /c /Flinffas32.lst inffas32.asm
;
; if you define NO_GZIP (see inflate.h), compile with
; ml /coff /Zi /c /Flinffas32.lst /DNO_GUNZIP inffas32.asm

; zlib122sup is 0 for zlib 1.2.2.1 and lower
; zlib122sup is 8 for zlib 1.2.2.2 and more (with addition of dmax and head
; in inflate_state in inflate.h)
zlib122sup equ 8

IFDEF GUNZIP
    INFLATE_MODE_TYPE equ 11
    INFLATE_MODE_BAD equ 26
ELSE
    IFNDEF NO_GUNZIP
        INFLATE_MODE_TYPE equ 11
        INFLATE_MODE_BAD equ 26
    ELSE
        INFLATE_MODE_TYPE equ 3
        INFLATE_MODE_BAD equ 17
    ENDIF
ENDIF

; 75 "inffast.S"
;FILE "inffast.S"

;;;GLOBAL _inflate_fast

;;;SECTION .text

    .586p
    .mmx

    name inflate_fast_x86
    .MODEL FLAT

_DATA segment
inflate_fast_use_mmx:
    dd 1
```

```
_TEXT                                segment
PUBLIC _inflate_fast

ALIGN 4
_inflate_fast:
    jmp inflate_fast_entry

ALIGN 4
    db      'Fast decoding Code from Chris Anderson'
    db      0

ALIGN 4
invalid_literal_length_code_msg:
    db      'invalid literal/length code'
    db      0

ALIGN 4
invalid_distance_code_msg:
    db      'invalid distance code'
    db      0

ALIGN 4
invalid_distance_too_far_msg:
    db      'invalid distance too far back'
    db      0

ALIGN 4
inflate_fast_mask:
dd      0
dd      1
dd      3
dd      7
dd      15
dd      31
dd      63
dd      127
dd      255
dd      511
dd      1023
dd      2047
dd      4095
dd      8191
dd      16383
dd      32767
dd      65535
dd      131071
dd      262143
dd      524287
dd      1048575
dd      2097151
dd      4194303
dd      8388607
dd      16777215
dd      33554431
dd      67108863
dd      134217727
dd      268435455
dd      536870911
dd      1073741823
dd      2147483647
dd      4294967295

mode_state      equ      0          ;/* state->mode */
wsizestate      equ      (32+zlib1222sup)    ;/* state->wsizestate */
write_state     equ      (36+4+zlib1222sup)  ;/* state->write */
window_state    equ      (40+4+zlib1222sup)  ;/* state->window */
hold_state      equ      (44+4+zlib1222sup)  ;/* state->hold */
bits_state      equ      (48+4+zlib1222sup)  ;/* state->bits */
lencode_state   equ      (64+4+zlib1222sup)  ;/* state->lencode */
distcode_state  equ      (68+4+zlib1222sup)  ;/* state->distcode */
```

```
lenbits_state    equ    (72+4+zlib1222sup)    ;/* state->lenbits */
distbits_state   equ    (76+4+zlib1222sup)    ;/* state->distbits */

;;SECTION .text
; 205 "inffast.S"
;GLOBAL inflate_fast_use_mmx

;SECTION .data

; GLOBAL inflate_fast_use_mmx:object
;.size inflate_fast_use_mmx, 4
; 226 "inffast.S"
;SECTION .text

ALIGN 4
inflate_fast_entry:
    push    edi
    push    esi
    push    ebp
    push    ebx
    pushfd
    sub     esp,64
    cld

    mov     esi, [esp+88]
    mov     edi, [esi+28]

    mov     edx, [esi+4]
    mov     eax, [esi+0]

    add     edx,eax
    sub     edx,11

    mov     [esp+44],eax
    mov     [esp+20],edx

    mov     ebp, [esp+92]
    mov     ecx, [esi+16]
    mov     ebx, [esi+12]

    sub     ebp,ecx
    neg     ebp
    add     ebp,ebx

    sub     ecx,257
    add     ecx,ebx

    mov     [esp+60],ebx
    mov     [esp+40],ebp
    mov     [esp+16],ecx
; 285 "inffast.S"
    mov     eax, [edi+lencode_state]
    mov     ecx, [edi+distcode_state]

    mov     [esp+8],eax
    mov     [esp+12],ecx

    mov     eax,1
    mov     ecx, [edi+lenbits_state]
    shl     eax,cl
    dec     eax
    mov     [esp+0],eax
```

```

    mov     eax,1
    mov     ecx,[edi+distbits_state]
    shl     eax,cl
    dec     eax
    mov     [esp+4],eax

    mov     eax,[edi+wsizestate]
    mov     ecx,[edi+write_state]
    mov     edx,[edi>window_state]

    mov     [esp+52],eax
    mov     [esp+48],ecx
    mov     [esp+56],edx

    mov     ebp,[edi+hold_state]
    mov     ebx,[edi+bits_state]
; 321 "inffast.S"
    mov     esi,[esp+44]
    mov     ecx,[esp+20]
    cmp     ecx,esi
    ja      L_align_long

    add     ecx,11
    sub     ecx,esi
    mov     eax,12
    sub     eax,ecx
    lea     edi,[esp+28]
    rep     movsb
    mov     ecx,eax
    xor     eax,eax
    rep     stosb
    lea     esi,[esp+28]
    mov     [esp+20],esi
    jmp     L_is_aligned

L_align_long:
    test    esi,3
    jz      L_is_aligned
    xor     eax,eax
    mov     al,[esi]
    inc     esi
    mov     ecx,ebx
    add     ebx,8
    shl     eax,cl
    or      ebp,eax
    jmp     L_align_long

L_is_aligned:
    mov     edi,[esp+60]
; 366 "inffast.S"
L_check_mmx:
    cmp     dword ptr [inflate_fast_use_mmx],2
    je      L_init_mmx
    ja      L_do_loop

    push    eax
    push    ebx
    push    ecx
    push    edx
    pushfd
    mov     eax,[esp]
    xor     dword ptr [esp],0200000h

    popfd
    pushfd
    pop     edx
    xor     edx,eax
    jz      L_dont_use_mmx
    xor     eax,eax
    cpuid

```



```
    cmp     ebx,0756e6547h
    jne     L_dont_use_mmx
    cmp     ecx,06c65746eh
    jne     L_dont_use_mmx
    cmp     edx,049656e69h
    jne     L_dont_use_mmx
    mov     eax,1
    cpuid
    shr     eax,8
    and     eax,15
    cmp     eax,6
    jne     L_dont_use_mmx
    test    edx,0800000h
    jnz     L_use_mmx
    jmp     L_dont_use_mmx
L_use_mmx:
    mov     dword ptr [inflate_fast_use_mmx],2
    jmp     L_check_mmx_pop
L_dont_use_mmx:
    mov     dword ptr [inflate_fast_use_mmx],3
L_check_mmx_pop:
    pop     edx
    pop     ecx
    pop     ebx
    pop     eax
    jmp     L_check_mmx
; 426 "inffast.S"
ALIGN 4
L_do_loop:
; 437 "inffast.S"
    cmp     bl,15
    ja      L_get_length_code

    xor     eax,eax
    lodsw
    mov     cl,bl
    add     bl,16
    shl     eax,cl
    or      ebp,eax

L_get_length_code:
    mov     edx,[esp+0]
    mov     ecx,[esp+8]
    and     edx,ebp
    mov     eax,[ecx+edx*4]

L_dolen:

    mov     cl,ah
    sub     bl,ah
    shr     ebp,cl

    test    al,al
    jnz     L_test_for_length_base

    shr     eax,16
    stosb

L_while_test:

    cmp     [esp+16],edi
    jbe     L_break_loop
```

```

    cmp [esp+20],esi
    ja  L_do_loop
    jmp L_break_loop

```

L_test_for_length_base:

```

; 502 "inffast.S"
    mov edx,eax
    shr edx,16
    mov cl,al

    test al,16
    jz   L_test_for_second_level_length
    and cl,15
    jz   L_save_len
    cmp  bl,cl
    jae  L_add_bits_to_len

    mov ch,cl
    xor eax,eax
    lodsw
    mov cl,bl
    add bl,16
    shl eax,cl
    or  ebp,eax
    mov cl,ch

```

L_add_bits_to_len:

```

    mov eax,1
    shl eax,cl
    dec eax
    sub bl,cl
    and eax,ebp
    shr ebp,cl
    add edx,eax

```

L_save_len:

```

    mov [esp+24],edx

```

L_decode_distance:

```

; 549 "inffast.S"
    cmp bl,15
    ja  L_get_distance_code

    xor eax,eax
    lodsw
    mov cl,bl
    add bl,16
    shl eax,cl
    or  ebp,eax

```

L_get_distance_code:

```

    mov edx,[esp+4]
    mov ecx,[esp+12]
    and edx,ebp
    mov eax,[ecx+edx*4]

```

L_dodist:

```

    mov edx,eax
    shr edx,16
    mov cl,ah
    sub bl,ah
    shr ebp,cl
; 584 "inffast.S"
    mov cl,al

    test al,16
    jz   L_test_for_second_level_dist
    and cl,15
    jz   L_check_dist_one
    cmp  bl,cl
    jae  L_add_bits_to_dist

```

```
    mov  ch,cl
    xor  eax,eax
    lodsw
    mov  cl,bl
    add  bl,16
    shl  eax,cl
    or   ebp,eax
    mov  cl,ch
```

L_add_bits_to_dist:

```
    mov  eax,1
    shl  eax,cl
    dec  eax
    sub  bl,cl
    and  eax,ebp
    shr  ebp,cl
    add  edx,eax
    jmp  L_check_window
```

L_check_window:

; 625 "inffast.S"

```
    mov  [esp+44],esi
    mov  eax,edi
    sub  eax,[esp+40]
```

```
    cmp  eax,edx
    jb   L_clip_window
```

```
    mov  ecx,[esp+24]
    mov  esi,edi
    sub  esi,edx
```

```
    sub  ecx,3
    mov  al,[esi]
    mov  [edi],al
    mov  al,[esi+1]
    mov  dl,[esi+2]
    add  esi,3
    mov  [edi+1],al
    mov  [edi+2],dl
    add  edi,3
    rep movsb
```

```
    mov  esi,[esp+44]
    jmp  L_while_test
```

ALIGN 4

L_check_dist_one:

```
    cmp  edx,1
    jne  L_check_window
    cmp  [esp+40],edi
    je   L_check_window
```

```
    dec  edi
    mov  ecx,[esp+24]
    mov  al,[edi]
    sub  ecx,3
```

```
    mov  [edi+1],al
    mov  [edi+2],al
    mov  [edi+3],al
    add  edi,4
    rep stosb
```

```
    jmp  L_while_test
```

ALIGN 4

L_test_for_second_level_length:

```
    test al,64
    jnz  L_test_for_end_of_block
```

```
    mov     eax,1
    shl     eax,cl
    dec     eax
    and     eax,ebp
    add     eax,edx
    mov     edx,[esp+8]
    mov     eax,[edx+eax*4]
    jmp     L_dolen
```

ALIGN 4

L_test_for_second_level_dist:

```
    test    al,64
    jnz     L_invalid_distance_code

    mov     eax,1
    shl     eax,cl
    dec     eax
    and     eax,ebp
    add     eax,edx
    mov     edx,[esp+12]
    mov     eax,[edx+eax*4]
    jmp     L_dodist
```

ALIGN 4

L_clip_window:

; 721 "inffast.S"

```
    mov     ecx,eax
    mov     eax,[esp+52]
    neg     ecx
    mov     esi,[esp+56]
```

```
    cmp     eax,edx
    jb      L_invalid_distance_too_far
```

```
    add     ecx,edx
    cmp     dword ptr [esp+48],0
    jne     L_wrap_around_window
```

```
    sub     eax,ecx
    add     esi,eax
```

; 749 "inffast.S"

```
    mov     eax,[esp+24]
    cmp     eax,ecx
    jbe     L_do_copy1
```

```
    sub     eax,ecx
    rep movsb
    mov     esi,edi
    sub     esi,edx
    jmp     L_do_copy1
```

```
    cmp     eax,ecx
    jbe     L_do_copy1
```

```
    sub     eax,ecx
    rep movsb
    mov     esi,edi
    sub     esi,edx
    jmp     L_do_copy1
```

L_wrap_around_window:

; 793 "inffast.S"

```
    mov     eax,[esp+48]
    cmp     ecx,eax
    jbe     L_contiguous_in_window
```

```
    add     esi,[esp+52]
    add     esi,eax
    sub     esi,ecx
```

```
    sub    ecx,eax

    mov    eax,[esp+24]
    cmp    eax,ecx
    jbe    L_do_copy1

    sub    eax,ecx
    rep    movsb
    mov    esi,[esp+56]
    mov    ecx,[esp+48]
    cmp    eax,ecx
    jbe    L_do_copy1

    sub    eax,ecx
    rep    movsb
    mov    esi,edi
    sub    esi,edx
    jmp    L_do_copy1

L_contiguous_in_window:
; 836 "inffast.S"
    add    esi,eax
    sub    esi,ecx

    mov    eax,[esp+24]
    cmp    eax,ecx
    jbe    L_do_copy1

    sub    eax,ecx
    rep    movsb
    mov    esi,edi
    sub    esi,edx

L_do_copy1:
; 862 "inffast.S"
    mov    ecx,eax
    rep    movsb

    mov    esi,[esp+44]
    jmp    L_while_test
; 878 "inffast.S"
ALIGN 4
L_init_mmx:
    emms

    movd   mm0,ebp
    mov    ebp,ebx
; 896 "inffast.S"
    movd   mm4,[esp+0]
    movq   mm3,mm4
    movd   mm5,[esp+4]
    movq   mm2,mm5
    pxor   mm1,mm1
    mov    ebx,[esp+8]
    jmp    L_do_loop_mmx

ALIGN 4
L_do_loop_mmx:
    psrlq  mm0,mm1

    cmp    ebp,32
    ja     L_get_length_code_mmx

    movd   mm6,ebp
    movd   mm7,[esi]
    add    esi,4
    psllq  mm7,mm6
    add    ebp,32
```

```

        por mm0,mm7

L_get_length_code_mmx:
        pand mm4,mm0
        movd eax,mm4
        movq mm4,mm3
        mov  eax, [ebx+eax*4]

L_dolen_mmx:
        movzx ecx,ah
        movd mml,ecx
        sub  ebp,ecx

        test al,al
        jnz L_test_for_length_base_mmx

        shr  eax,16
        stosb

L_while_test_mmx:

        cmp  [esp+16],edi
        jbe  L_break_loop

        cmp  [esp+20],esi
        ja   L_do_loop_mmx
        jmp  L_break_loop

L_test_for_length_base_mmx:

        mov  edx,eax
        shr  edx,16

        test al,16
        jz   L_test_for_second_level_length_mmx
        and  eax,15
        jz   L_decode_distance_mmx

        psrlq mm0,mm1
        movd mml,eax
        movd ecx,mm0
        sub  ebp,eax
        and  ecx, [inflate_fast_mask+eax*4]
        add  edx,ecx

L_decode_distance_mmx:
        psrlq mm0,mm1

        cmp  ebp,32
        ja   L_get_dist_code_mmx

        movd mm6,ebp
        movd mm7,[esi]
        add  esi,4
        psllq mm7,mm6
        add  ebp,32
        por  mm0,mm7

L_get_dist_code_mmx:
        mov  ebx, [esp+12]
        pand mm5,mm0
        movd eax,mm5
        movq mm5,mm2
        mov  eax, [ebx+eax*4]

L_dodist_mmx:

        movzx ecx,ah
        mov  ebx,eax
        shr  ebx,16
        sub  ebp,ecx
        movd mml,ecx
    
```

```
test al,16
jz L_test_for_second_level_dist_mmx
and eax,15
jz L_check_dist_one_mmx
```

L_add_bits_to_dist_mmx:

```
psrlq mm0,mm1
movd mm1,eax
movd ecx,mm0
sub ebp,eax
and ecx,[inflate_fast_mask+eax*4]
add ebx,ecx
```

L_check_window_mmx:

```
mov [esp+44],esi
mov eax,edi
sub eax,[esp+40]

cmp eax,ebx
jb L_clip_window_mmx
```

```
mov ecx,edx
mov esi,edi
sub esi,ebx
```

```
sub ecx,3
mov al,[esi]
mov [edi],al
mov al,[esi+1]
mov dl,[esi+2]
add esi,3
mov [edi+1],al
mov [edi+2],dl
add edi,3
rep movsb
```

```
mov esi,[esp+44]
mov ebx,[esp+8]
jmp L_while_test_mmx
```

ALIGN 4

L_check_dist_one_mmx:

```
cmp ebx,1
jne L_check_window_mmx
cmp [esp+40],edi
je L_check_window_mmx
```

```
dec edi
mov ecx,edx
mov al,[edi]
sub ecx,3
```

```
mov [edi+1],al
mov [edi+2],al
mov [edi+3],al
add edi,4
rep stosb
```

```
mov ebx,[esp+8]
jmp L_while_test_mmx
```

ALIGN 4

L_test_for_second_level_length_mmx:

```
test al,64
jnz L_test_for_end_of_block
```

```
and eax,15
psrlq mm0,mm1
movd ecx,mm0
and ecx,[inflate_fast_mask+eax*4]
add ecx,edx
mov eax,[ebx+ecx*4]
jmp L_dolen_mmx
```

```
ALIGN 4
L_test_for_second_level_dist_mmx:
    test    al,64
    jnz     L_invalid_distance_code

    and     eax,15
    psrlq   mm0,mm1
    movd    ecx,mm0
    and     ecx,[inflate_fast_mask+eax*4]
    mov     eax,[esp+12]
    add     ecx,ebx
    mov     eax,[eax+ecx*4]
    jmp     L_dodist_mmx
```

```
ALIGN 4
L_clip_window_mmx:

    mov     ecx,eax
    mov     eax,[esp+52]
    neg     ecx
    mov     esi,[esp+56]

    cmp     eax,ebx
    jb      L_invalid_distance_too_far

    add     ecx,ebx
    cmp     dword ptr [esp+48],0
    jne     L_wrap_around_window_mmx

    sub     eax,ecx
    add     esi,eax

    cmp     edx,ecx
    jbe     L_do_cpy1_mmx

    sub     edx,ecx
    rep movsb
    mov     esi,edi
    sub     esi,ebx
    jmp     L_do_cpy1_mmx

    cmp     edx,ecx
    jbe     L_do_cpy1_mmx

    sub     edx,ecx
    rep movsb
    mov     esi,edi
    sub     esi,ebx
    jmp     L_do_cpy1_mmx
```

```
L_wrap_around_window_mmx:

    mov     eax,[esp+48]
    cmp     ecx,eax
    jbe     L_contiguous_in_window_mmx

    add     esi,[esp+52]
    add     esi,eax
    sub     esi,ecx
    sub     ecx,eax

    cmp     edx,ecx
    jbe     L_do_cpy1_mmx

    sub     edx,ecx
    rep movsb
    mov     esi,[esp+56]
    mov     ecx,[esp+48]
    cmp     edx,ecx
    jbe     L_do_cpy1_mmx

    sub     edx,ecx
    rep movsb
```



```
    mov     esi,edi
    sub     esi,ebx
    jmp     L_do_copy1_mmx
```

L_contiguous_in_window_mmx:

```
    add     esi,eax
    sub     esi,ecx
```

```
    cmp     edx,ecx
    jbe     L_do_copy1_mmx
```

```
    sub     edx,ecx
    rep     movsb
    mov     esi,edi
    sub     esi,ebx
```

L_do_copy1_mmx:

```
    mov     ecx,edx
    rep     movsb
```

```
    mov     esi, [esp+44]
    mov     ebx, [esp+8]
    jmp     L_while_test_mmx
```

; 1174 "inffast.S"

L_invalid_distance_code:

```
    mov     ecx, invalid_distance_code_msg
    mov     edx, INFLATE_MODE_BAD
    jmp     L_update_stream_state
```

L_test_for_end_of_block:

```
    test    al,32
    jz      L_invalid_literal_length_code
```

```
    mov     ecx,0
    mov     edx, INFLATE_MODE_TYPE
    jmp     L_update_stream_state
```

L_invalid_literal_length_code:

```
    mov     ecx, invalid_literal_length_code_msg
    mov     edx, INFLATE_MODE_BAD
    jmp     L_update_stream_state
```

L_invalid_distance_too_far:

```
    mov     esi, [esp+44]
    mov     ecx, invalid_distance_too_far_msg
    mov     edx, INFLATE_MODE_BAD
    jmp     L_update_stream_state
```

L_update_stream_state:

```
    mov     eax, [esp+88]
    test     ecx,ecx
```

```
        jz     L_skip_msg
        mov     [eax+24],ecx
L_skip_msg:
        mov     eax, [eax+28]
        mov     [eax+mode_state],edx
        jmp     L_break_loop

ALIGN 4
L_break_loop:
; 1243 "inffast.S"
        cmp     dword ptr [inflate_fast_use_mmx],2
        jne     L_update_next_in

        mov     ebx,ebp

L_update_next_in:
; 1266 "inffast.S"
        mov     eax, [esp+88]
        mov     ecx,ebx
        mov     edx, [eax+28]
        shr     ecx,3
        sub     esi,ecx
        shl     ecx,3
        sub     ebx,ecx
        mov     [eax+12],edi
        mov     [edx+bits_state],ebx
        mov     ecx,ebx

        lea     ebx, [esp+28]
        cmp     [esp+20],ebx
        jne     L_buf_not_used

        sub     esi,ebx
        mov     ebx, [eax+0]
        mov     [esp+20],ebx
        add     esi,ebx
        mov     ebx, [eax+4]
        sub     ebx,11
        add     [esp+20],ebx

L_buf_not_used:
        mov     [eax+0],esi

        mov     ebx,1
        shl     ebx,cl
        dec     ebx

        cmp     dword ptr [inflate_fast_use_mmx],2
        jne     L_update_hold

        psrlq   mm0,mm1
        movd    ebp,mm0

        emms

L_update_hold:

        and     ebp,ebx
        mov     [edx+hold_state],ebp

        mov     ebx, [esp+20]
```

```
        cmp     ebx,esi
        jbe     L_last_is_smaller

        sub     ebx,esi
        add     ebx,11
        mov     [eax+4],ebx
        jmp     L_fixup_out
L_last_is_smaller:
        sub     esi,ebx
        neg     esi
        add     esi,11
        mov     [eax+4],esi

L_fixup_out:
        mov     ebx, [esp+16]
        cmp     ebx,edi
        jbe     L_end_is_smaller

        sub     ebx,edi
        add     ebx,257
        mov     [eax+16],ebx
        jmp     L_done
L_end_is_smaller:
        sub     edi,ebx
        neg     edi
        add     edi,257
        mov     [eax+16],edi

L_done:
        add     esp,64
        popfd
        pop     ebx
        pop     ebp
        pop     esi
        pop     edi
        ret

_TEXT   ends
end
```

L^A^D^@vM-^N^EBÖ/^@@R^@@@@@@@@.text^@@@@@@@@@@@@@@@@M-^[^G^@@^'@@^@P^H^@@E
^H^@@@

```
@Q^@B ^@P`.data^@@^@M-^[^G^@@^@^@^@^@^@D^@^@^@R^V^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@PA.debug
$SM-^_G^@^@^@^@^@^@X^O^@^@^2^V^@^@E%^^@^@^@^@^@^@p^@^@^@^@^@PB.debug$T.^V^@^@^@^@^@^@^@
^@^@^@||/^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@PBÉC^A^@^@M-^MI^@Fast decoding Code from Chris And
erson^@M-^Pinvalid literal/length code^@invalid distance code^@M-^Kÿinvalid distance too
far back^@M-^Kÿ^@^@^@^@^@A^@^@^@C^@^@^@G^@^@^@O^@^@^@_@^@^@?^@^@^@?^@^@^@ÿ^@^@^@ÿ^A
^@^@^@ÿC^@^@^@ÿG^@^@^@ÿO^@^@^@ÿ_@^@^@ÿ?^@^@^@ÿ?^@^@^@ÿ^@^@^@ÿ^A^@ÿÿC^@ÿÿG^@ÿÿO^@ÿÿ_@ÿÿ?^@ÿÿ
^?^@ÿÿÿ^@ÿÿÿAÿÿÿCÿÿÿGÿÿÿOÿÿÿ_ÿÿÿÿÿÿÿWVUSM-^\M-^CiüM-^Kt$XM-^K~\M-^KV^DM-^K^F
^ÇBM-^CêKM-^IDŞ,M-^ITŞ^TM-^KlŞ\M-^KN^PM-^K^
```

+é÷Ÿ^CëM-^Aé^A^A^@^@^CëM-^I\,\$<M-^I l \$(M-^IL\$^PM-^KGLM-^KOPM-^ID\$^HM-^IL\$

,^A^@^@M-^KOTÓàHM-^I^D\$,^A^@^@M-^KXÓàHM-^ID\$^DM-^KG(M-^K00M-^KW4M-^ID\$4M-^IL\$0M-^IT\$8M-^K08M-^K_<M-^Kt\$,M-^KL\$^T;Îw"M-^CÁ^K+Î,

```
^@@@+ÃM-^M| $^\óM-^KÈ3Ãó^M-^Mt$^\M-^It$^Të^X÷Æ^C^@@@t^P3ÀM-^J^FFM-^KËM-^CÃ^HÓà^Kèèè
M-^K| $<M-^C=^@@@^@@^B^OM-^DM-^I^B^@@wkPSQRM-^\M-^K^D$M-^A4$^@@ ^@M-^]M-^\Z3ĐtD3Ã^Oç
M-^AûGenuu8M-^Aùntelu0M-^AúineIu(, ^A^@@@^OçÃè^HM-^Cà^OM-^Cø^Fu^V÷Â^@@M-^@@u^Bë
```


Ç^E^@^@^@^@B^@^@^ë

Ç^E^@^@^@^@C^@^@^ZY[XëM-^GM-^PM-^@û^Ow

3Àf-M-^JëM-^@Ã^PÓà^KèM-^K^T\$M-^KL\$^H#ÕM-^K^DM-^QM-^JÌ*ÜÓíM-^DÀu^YÁè^P^9| \$^P^OM-^Fb^D^@^@9
t\$^TwÄéW^D^@^@M-^KĐÁê^PM-^JÈ^P^OM-^Dô^@^@^@M-^@á^Ot%8Ës^QM-^Jé3Àf-M-^JëM-^@Ã^PÓà^KèM-^JÍ

^A^@^@^@ÓàH*Û#ÅÓí^CĐM-^IT\$^XM-^@û^Ow

3Àf-M-^JëM-^@Ã^PÓà^KèM-^KT\$^DM-^KL\$

#ÖM-^K^DM-^QM-^KĐĂê^PM-^Jİ*ÜÓıM-^JÈ^P^OM-^D^2^@^@^@M-^@á^Ote8Ěs^QM-^Jé3Ăf-M-^JĚM-^@Ă^PÓà
^KèM-^JÍ,^A^@^@^@ÓàH*Û#ĂÓı^CĐě^@M-^It\$,M-^KÇ+D\$(;Ă^OM-^BM-^T^@^@^@M-^KL\$^XM-^K÷+öM-^Cé^C
M-^J^FM-^H^GM-^JF^AM-^JV^BM-^CĚ^CM-^HG^AM-^HW^BM-^CÇ^CÓóM-^Kt\$,é^PÿÿÿM-^Cú^Au¼9|\$(t•OM-^K
L\$^XM-^J^GM-^Cé^CM-^HG^AM-^HG^BM-^HG^CM-^CÇ^Dó^aéèpÿÿÿ^@OM-^E^N^C^@^@,^A^@^@^@ÓàH#Ă^CĂM-^K
T\$^DM-^Bé°pÿÿÿ^@OM-^Eă^B^@^@,^A^@^@^@ÓàH#Ă^CĂM-^KT\$

M-^K^DM-^Bé^YÿÿÿM-^KÈM-^KD\$4÷ÛM-^Kt\$8;Â^OM-^BÞ^B^@^@^CÈM-^C|\$0^@u\$+Á^CðM-^KD\$^X;Áv`+Áó□
M-^K÷+ðëV;ÁvR+Áó□M-^K÷+ðëHM-^KD\$0;Èv,^Ct\$4^Cð+ñ+ÈM-^KD\$^X;Áv.+Áó□M-^Kt\$8M-^KL\$0;Áv^^+Áó□
M-^K÷+ðë^T^Cð+ñM-^KD\$^X;ÁvÁó□M-^K÷+ðM-^KÈó□M-^Kt\$,é^RþÿÿM-^Kÿ^Ow^OnÅM-^Kë^On\$^O^?ã^Onl\$
^D^O^?ê^OïÈM-^K\ \$^AM-^P^OÓÁM-^CÝ w^R^Onö^On>M-^CÆ^D^OóþM-^CÅ ^OëÇ^OÛà^O~à^O^?ÛM-^K^DM-^C
^O¶Î^OnÉ+éM-^DÀu^YÁè^Pª9|\$^P^OM-^F^\^B^@^@9t\$^Tw°éQ^B^@^@M-^KĐÁê^P^^P^OM-^Dà^@^@^@M-^Cà
^Ot^T^OÓÁ^OnÈ^O~Á+è#

M-^E^@^@^@^@^@CÑ^OÓÁM-^CŸ w^R^Onõ^On>M-^CÆ^D^OópM-^CĀ ^OëÇM-^K\\$_

^OÛè^O~è^O^?ÕM-^K^DM-^C^O¶ÌM-^KØÁě^P+é^OnÉ''^P^OM-^D¬^@^@^@M-^Cà^OtW^OÓÁ^OnÈ^O~Á+è#

M-^E^@@@^@^@^CÛM-^It\$,M-^KÇ+D\$(;Ã^OM-^B©^@^@^@M-^KÊM-^K÷+óM-^Cé^CM-^J^FM-^H^GM-^JF^AM-^JV
^BM-^CÆ^CM-^HG^AM-^HW^BM-^CÇ^CÓ◻M-^Kt\$,M-^K\\$_-ÿÿÿM-^MI^@M-^Cû^Au,9|\$(t^2OM-^KÊM-^J^GM-^Cé
^CM-^HG^AM-^HG^BM-^HG^CM-^CÇ^DóªM-^K\\$_-ÿÿÿM-^Kÿ"@OM-^Eþ^@^@^@M-^Cà^O^OÓÁ^O~Á#

M-^E^@^@^@^@^CÊM-^K^DM-^KéİpÿÿM-^Kÿ" @^OM-^E®^@^@^M-^Cà^O^OÓÁ^O~Á#

M-^E^@^@^@M-^KD\$

^CĚM-^K^DM-^Hé^YÿÿM-^KÿM-^KĚM-^KD\$4÷ÛM-^Kt\$8;Ã^OM-^Bç^@@^@^CĚM-^C|\$0^@u +Á^Cđ;ÑvX+ÑóM-^K÷+óĚN;ÑvJ+ÑóM-^K÷+óĚ@M-^KD\$0;Ěv(^Ct\$4^Cđ+ñ+Ě;Ñv*+ÑóM-^Kt\$8M-^KL\$0;Ñv^Z+ÑóM-^K÷+óĚ^P^Cđ+ñ;ÑvÑóM-^K÷+óM-^KĚóM-^Kt\$,M-^K\\$\$pÿÿ^1^@@^@^@^@^Z^@^@^@ě," t

1^@@@^@°^K^@@@ë\1^@@@^@°^Z^@@@ë^PM-^Kt\$,1^@@@^@°^Z^@@@ë^M-^KD\$XM-^EÉt^CM-^I
H^XM-^K@^M-^I^Pë^M-^C=^@@@@@^Bu^BM-^KÝM-^KD\$XM-^KËM-^KP^Áé^C+ñÁá^C+ÛM-^Ix

M-^IZ<M-^KĚM-^M\\$/9\\$/^Tu^T+óM-^K^XM-^I\\$/^T^Cóm-^KX^DM-^Cě^K^A\\$/^TM-^I0»^A^@^@^@ŌăKM-^C=
^@^@^@^@^BuŌĂ^O~Ă^Ow#ĚM-^Ij8M-^K\\$/^T;Ǽv
+ǼM-^CĂ^KM-^IX^Dě
+ó÷ǼM-^CĚ^KM-^Ip^DM-^K\\$/^P;Ǽv
+ǼM-^AĂ^A^A^@^@M-^IX^PĚ
+û÷ǼM-^AÇ^A^A^@^@M-^Ix^PM-^CĂ@M-^][]^_Ă^@à^A^@^@^O^@^@^@F^@=^B^@^@^O^@^@^@F^@I^B^@^@^O
^@^@^@F^@Ÿ^D^@^@
^@^@^@F^@W^E^@^@
^@^@^@F^@à^E^@^@
^@^@^@F^@D^F^@^@
^@^@^@F^@Ÿ^F^@^@K^@^@^@F^@Ă^F^@^@N^@^@^@F^@Ō^F^@^@P^@^@^@F^@ò^F^@^@O^@^@^@F^@^@G
^@^@^@O^@^@^@F^@Q^@^@^@^@^@H^A^@^@W^@ ^A^@^@X^@
^A^@^@Y^@^K^A^@^@Z^@

Tuesday June 13, 2006

```

^BE^D^@^@
^BG^D^@^@^O^BI^D^@^@^P^BK^D^@^@^Q^BM^D^@^@^R^BO^D^@^@^S^BQ^D^@^@^W^BS^D^@^@^X^BU^D^@^@^
^BY^D^@^@^B[ ^D^@^@^ ]^B]^D^@^@^_ ^B_ ^D^@^@^ ^Ba^D^@^@! ^Bc^D^@^@" ^Be^D^@^@& ^Bg^D^@^@' ^Bi^D
^@^@) ^Bm^D^@^@* ^Bt^D^@^@. ^Bv^D^@^@4 ^By^D^@^@5 ^B{ ^D^@^@7 ^B? ^D^@^@8 ^BM- ^B^D^@^@9 ^BM- ^G^D^@
^@: ^BM- ^J^D^@^@; ^BM- ^M^D^@^@< ^BM- ^Q^D^@^@= ^BM- ^T^D^@^@A ^BM- ^W^D^@^@C ^BM- ^Z^D^@^@D ^BM- ^\ ^D
^@^@F ^BM- ^_ ^D^@^@G ^Bç^D^@^@H ^B¥^D^@^@I ^B" ^D^@^@J ^B« ^D^@^@K ^B® ^D^@^@N ^B± ^D^@^@O ^B´ ^D^@^@P
^B• ^D^@^@Q ^B° ^D^@^@T ^B½ ^D^@^@U ^BÀ ^D^@^@V ^BĀ ^D^@^@X ^BǺ ^D^@^@Y ^BÆ ^D^@^@[ ^BÉ ^D^@^@\ ^BÊ ^D^@^@
a ^BÎ ^D^@^@b ^BÔ ^D^@^@d ^BØ ^D^@^@e ^BÚ ^D^@^@f ^BŜ ^D^@^@j ^Bá ^D^@^@k ^Bă ^D^@^@m ^Bæ ^D^@^@n ^Bi ^D^@
^@o ^Bi ^D^@^@p ^Bñ ^D^@^@r ^Bô ^D^@^@s ^B÷ ^D^@^@t ^Bú ^D^@^@u ^Bü ^D^@^@v ^B^C ^E^@^@w ^B^E ^E^@^@z ^B^H
^E^@^@| ^B^K ^E^@^@} ^B
^E^@^@? ^B^P ^E^@^@M- ^@ ^B^S ^E^@^@M- ^A ^B^V ^E^@^@M- ^B ^B^Y ^E^@^@M- ^C ^B^ \ ^E^@^@M- ^D ^B^ _ ^E^@^@
M- ^G ^B^# ^E^@^@M- ^H ^B^& ^E^@^@M- ^I ^B^ ) ^E^@^@M- ^J ^B^ , ^E^@^@M- ^K ^B^ / ^E^@^@M- ^O ^B^2 ^E^@^@M- ^P ^B^4 ^E^@
^@M- ^Q ^B^7 ^E^@^@M- ^R ^B^9 ^E^@^@M- ^S ^B^< ^E^@^@M- ^U ^B^> ^E^@^@M- ^V ^B^D ^E^@^@M- ^W ^B^G ^E^@^@M- ^X ^B^I ^E
^@^@M- ^[ ^B^L ^E^@^@M- ^\ ^B^O ^E^@^@M- ^ ^B^R ^E^@^@M- ^ ^B^T ^E^@^@M- ^ _ ^B[ ^E^@^@ ^B ] ^E^@^@f ^B^a ^E^@^@
^Bc ^E^@^@¥ ^B^g ^E^@^@§ ^B^i ^E^@^@" ^B^o ^E^@^@ª ^B^q ^E^@^@« ^B^s ^E^@^@¬ ^B^u ^E^@^@® ^B^x ^E^@^@¯ ^B^z ^E^@
^@ ^B | ^E^@^@± ^B? ^E^@^@² ^B- ^B^E^@^@³ ^B- ^E^@^@´ ^B- ^H ^E^@^@µ ^B- ^K ^E^@^@¶ ^B- ^N ^E^@^@•
^B- ^P ^E^@^@¹ ^B- ^T ^E^@^@º ^B- ^X ^E^@^@» ^B ^E^@^@¿ ^B^f ^E^@^@À ^B^¥ ^E^@^@Ā ^B^© ^E^@^@Ė ^B^« ^E^@^@Ħ
^B¬ ^E^@^@Ħ ^B^® ^E^@^@Æ ^B^° ^E^@^@Ç ^B³ ^E^@^@É ^B¶ ^E^@^@Ê ^B¹ ^E^@^@Ë ^B¼ ^E^@^@İ ^B¿ ^E^@^@Í ^BĀ ^E^@^@
İ ^BĖ ^E^@^@Đ ^BÌ ^E^@^@Ï ^BÎ ^E^@^@Ö ^BÔ ^E^@^@× ^B× ^E^@^@Ø ^BÚ ^E^@^@Ù ^BÝ ^E^@^@Ů ^Bă ^E^@^@Ű ^Bæ ^E^@
^@Ű ^Bé ^E^@^@Ÿ ^Bő ^E^@^@á ^Bò ^E^@^@â ^Bø ^E^@^@ă ^Bû ^E^@^@ă ^Bþ ^E^@^@æ ^B^A ^F^@^@ç ^B^H ^F^@^@è ^B

```

^F^@^@é^B^N^F^@^@ê^B^Q^F^@^@ë^B^X^F^@^@ö^B^Z^F^@^@ñ^B^^^F^@^@ò^B ^F^@^@ó^B\$^F^@^@ö^B&^F^@
^@ö^B,^F^@^@ø^B.^F^@^@ù^B3^F^@^@ú^B5^F^@^@ü^B7^F^@^@ý^B9^F^@^@ÿ^B;^F^@^@^@C=^F^@^@^B^C?
^F^@^@^C^CA^F^@^@^D^CC^F^@^@^E^CE^F^@^@^F^CG^F^@^@^H^CI^F^@^@ ^CK^F^@^@^K^CM^F^@^@

^CO^F^@^@
^CQ^F^@^@^N^CS^F^@^@^O^CU^F^@^@^S^CY^F^@^@^T^C[^F^@^@^U^C]^F^@^@^W^Ca^F^@^@^X^Cc^F^@^@^Y
^Ce^F^@^@^Z^Cg^F^@^@^] ^Ci^F^@^@^Ck^F^@^@ ^Cm^F^@^@! ^Co^F^@^@" ^Cs^F^@^@# ^Cw^F^@^@ \$ ^Cy^F
^@^@% ^C{ ^F^@^@' ^C} ^F^@^@ (^C? ^F^@^@) ^CM- ^A^F^@^@* ^CM- ^C^F^@^@+ ^CM- ^E^F^@^@/ ^CM- ^G^F^@^@0
^CM- ^I^F^@^@3 ^CM- ^K^F^@^@4 ^CM- ^M^F^@^@6 ^CM- ^O^F^@^@7 ^CM- ^Q^F^@^@8 ^CM- ^S^F^@^@9 ^CM- ^U^F^@
^@> ^CM- ^W^F^@^@? ^CM- ^Y^F^@^@A ^CM- ^] ^F^@^@B^C; ^F^@^@C^C| ^F^@^@K^C< ^F^@^@L^C° ^F^@^@M^C² ^F^@
^@U^C' ^F^@^@V^C¶ ^F^@^@X^C» ^F^@^@Y^CÀ ^F^@^@Z^CÂ ^F^@^@b^CÇ ^F^@^@c^CÌ ^F^@^@d^CÎ ^F^@^@j^CÔ ^F
^@^@k^C× ^F^@^@l^CÛ ^F^@^@m^Cᐁ ^F^@^@q^Câ ^F^@^@r^Cä ^F^@^@s^Cæ ^F^@^@t^Cé ^F^@^@v^Cì ^F^@^@w^Cî
^F^@^@x^Cḏ ^F^@^@} ^C÷ ^F^@^@~ ^Cù ^F^@^@M- ^B^Cû ^F^@^@M- ^F^Cÿ ^F^@^@M- ^G^C^A^G^@^@M- ^H^C^D^G^@
^@M- ^I^C^G^G^@^@M- ^J^C ^G^@^@M- ^K^C

```

^G^@^@M-^L^C^N^G^@^@M-^M^C^Q^G^@^@M-^N^C^T^G^@^@M-^O^C^V^G^@^@M-^Q^C^Z^G^@^@M-^R^C^G^@
^@M-^S^C ^G^@^@M-^U^C" ^G^@^@M-^V^C$^G^@^@M-^W^C( ^G^@^@M-^X^C*^G^@^@M-^Y^C-^G^@^@M-^Z^C0^G
^@^@M-^[^C4^G^@^@M-^^^C6^G^@^@ ^C;^G^@^@j^C=^G^@^@ç^C>^G^@^@"^CE^G^@^@C^G^@^@-^CJ^G^@
^@8^CM^G^@^@°CO^G^@^@¶^CQ^G^@^@•^CT^G^@^@¼^CX^G^@^@½^CZ^G^@^@¾^C\^G^@^@À^C^G^@^@Ã^Ca^G
^@^@Â^Cd^G^@^@Ã^Cf^G^@^@Ä^Ch^G^@^@Å^Cj^G^@^@Ç^Cm^G^@^@È^Cp^G^@^@Ì^Ct^G^@^@Ð^Cv^G^@^@Ñ^Cx
^G^@^@Ó^Cz^G^@^@Ô^CM-^@G^@^@Õ^CM-^C^G^@^@Ö^CM-^E^G^@^@Ø^CM-^G^G^@^@Û^CM-^I^G^@^@Ü^CM-^O
^G^@^@Û^CM-^R^G^@^@â^CM-^U^G^@^@ä^CM-^V^G^@^@ä^CM-^W^G^@^@å^CM-^X^G^@^@æ^CM-^Y^G^@^@ç^C
M-^Z^G^@^@è^C^A^@^@^@A^@^@^@S^@ ^@A^@^@^@

```



```
inffas32.obj6^A^A^C^@/Microsoft (R) Macro Assembler Version 6.14.8444 ^@ ^B^@^@^@
^@^@^@^@VL_test_for_length_base$^@ ^B^@^@^@^@^@^@^@ZL_invalid_distance_too_far^^^@
^B^@^@^@^@^@^@^@TL_check_dist_one_mmx^X^@ ^B^@^@^@^@^@^@^@NL_do_cpy1_mmx^^^@
^B^@^@^@^@^@^@^@TL_wrap_around_window^T^@ ^B^@^@^@^@^@^@^@
L_do_cpy1[^@ ^B^@^@^@^@^@^@^@QL_add_bits_to_len^S^@ ^B^@^@^@^@^@^@^@ L_do_loop
'^@ ^B^@^@^@^@^@^@^@]L_invalid_literal_length_code#^@ ^B^@^@^@^@^@^@^@Yinvalid
_distance_code_msg^_@ ^B^@^@^@^@^@^@^@UL_get_length_code_mmx^W^@ ^B^@^@^@^@^@^@^@
_inflate_fast^T^@ ^B^@^@^@^@^@^@^@
L_init_mmx(^@ ^B^@^@^@^@^@^@^@L_test_for_second_level_length^T^@ ^B^@^@^@^@^@^@^@
L_save_len ^@ ^B^@^@^@^@^@^@^@VL_add_bits_to_dist_mmx[^@ ^B^@^@^@^@^@^@^@Qinflate
_fast_mask^Z^@ ^B^@^@^@^@^@^@^@PL_update_next_in^R^@ ^B^@^@^@^@^@^@^@HL_dodist^T^@
^B^@^@^@^@^@^@^@
L_skip_msg^]^@ ^B^@^@^@^@^@^@^@SL_get_distance_code"^@ ^B^@^@^@^@^@^@^@XL_wrap_
around_window_mmx^W^@ ^B^@^@^@^@^@^@^@
L_do_loop_mmx!^@ ^B^@^@^@^@^@^@^@WL_invalid_distance_code&^@ ^B^@^@^@^@^@^@^@
^L_test_for_second_level_dist!^@ ^B^@^@^@^@^@^@^@WL_test_for_end_of_block)^@
^B^@^@^@^@^@^@^@_invalid_literal_length_code_msg*^@ ^B^@^@^@^@^@^@^@ L_test_for_secon
d_level_dist_mmx[^@ ^B^@^@^@^@^@^@^@QL_clip_window_mmx^U^@ ^B^@^@^@^@^@^@^@KL_dolen
_mmx^S^@ ^B^@^@^@^@^@^@^@ L_use_mmx^V^@ ^B^@^@^@^@^@^@^@
```

```
L_break_loop^[^@      ^B@@@@@@@@@QL_get_length_code^\^@ ^B@@@@@@@@@RL_add_b
its_to_dist^U^@ ^B@@@@@@@@@KL_fixup_out^]^@      ^B@@@@@@@@@SL_get_dist_code
_mmx^W^@      ^B@@@@@@@@@
L_clip_window,^@D^B@@@@@@@@@@@@@@@@@M-^[^G@@@@D@@@@M-^[^G@@@@@@@@@@@@A^P
^H$$$00001)^@ ^B@@@@@@@@@_invalid_literal_length_code_msg#^@      ^B@@@@@@@@@
^Yinvalid_distance_code_msg&^@ ^B@@@@@@@@@invalid_distance_too_far_msg^[^@
^B@@@@@@@@@Qinflate_fast_mask^\^@ ^B@@@@@@@@@Rinflate_fast_entry^V^@
^B@@@@@@@@@
```

L_align_long^V^@

^B^@^@^@^@^@^@

```
L_is_aligned^U^@      ^B^@@^@@^@@^@@^@@KL_check_mmx^S^@      ^B^@@^@@^@@^@@^@@      L
_use_mmx^X^@      ^B^@@^@@^@@^@@^@@NL_dont_use_mmx^Y^@      ^B^@@^@@^@@^@@^@@OL_check_mmx_pop
^S^@      ^B^@@^@@^@@^@@^@@      L_do_loop^[^@      ^B^@@^@@^@@^@@^@@QL_get_length_code^Q^@
^B^@@^@@^@@^@@^@@GL_dolen^V^@      ^B^@@^@@^@@^@@^@@
```

```
L_while_test ^@ ^B@@@@@^VL_test_for_length_base^[^@ ^B@@@@@^QL_add_b
its_to_len^T^@ ^B@@@@@^
L_save_len^[^@ ^B@@@@@^QL_decode_distance]^@ ^B@@@@@^SL_get_distance_
code^R^@ ^B@@@@@^HL_dodist^\^@ ^B@@@@@^RL_add_bits_to_dist^X^@
^B@@@@@^NL_check_window^Z^@ ^B@@@@@^PL_check_dist_one(^@ ^B@@@@@
^@@@@^L_test_for_second_level_length&^@ ^B@@@@@^L_test_for_second_level
_dist^W^@ ^B@@@@@
L_clip_window^^^@ ^B@@@@@^TL_wrap_around_window ^@ ^B@@@@@
^VL_contiguous_in_window^T^@ ^B@@@@@
L_do_copy1^T^@ ^B@@@@@
L_init_mmx^W^@ ^B@@@@@
L_do_loop_mmx^_@ ^B@@@@@^UL_get_length_code_mmx^U^@ ^B@@@@@
^KL_dolen_mmx^Z^@ ^B@@@@@^PL_while_test_mmx$^@ ^B@@@@@^ZL_test_
for_length_base_mmx^_@ ^B@@@@@^UL_decode_distance_mmx]^@ ^B@@@@@
^SL_get_dist_code_mmx^V^@ ^B@@@@@
```

```
L_dodist_mmx ^@ ^B^@@^@@^@@^@@^@@VL_add_bits_to_dist_mmx^\^@ ^B^@@^@@^@@^@@^@@RL_check
_window_mmx^^^@ ^B^@@^@@^@@^@@^@@TL_check_dist_one_mmx,^@ ^B^@@^@@^@@^@@^@@L_test_f
or_second_level_length_mmx*^@ ^B^@@^@@^@@^@@^@@ L_test_for_second_level_dist_mmx^[^@
^B^@@^@@^@@^@@^@@QL_clip_window_mmx"^@ ^B^@@^@@^@@^@@^@@^@XL_wrap_around_window_mmx$^@
^B^@@^@@^@@^@@^@@ZL_contiguous_in_window_mmx^X^@ ^B^@@^@@^@@^@@^@@^@NL_do_copy1_mmx!
^@ ^B^@@^@@^@@^@@^@@^@WL_invalid_distance_code!^@ ^B^@@^@@^@@^@@^@@^@WL_test_for_end_
of_block'^@ ^B^@@^@@^@@^@@^@@^@]L_invalid_literal_length_code$^@ ^B^@@^@@^@@^@@^@@^@
^ZL_invalid_distance_too_far^_ ^@ ^B^@@^@@^@@^@@^@@^@UL_update_stream_state^T^@
^B^@@^@@^@@^@@^@@^@
L_skip_msg^V^@ ^B^@@^@@^@@^@@^@@^@
```

```
L_break_loop^Z^@      ^B^@@^@@^@@^@@^@@PL_update_next_in^X^@  ^B^@@^@@^@@^@@^@@NL_buf_n
ot_used^W^@      ^B^@@^@@^@@^@@^@@
L_update_hold^[^@      ^B^@@^@@^@@^@@^@@QL_last_is_smaller^U^@  ^B^@@^@@^@@^@@^@@KL_fixup
_out^Z^@      ^B^@@^@@^@@^@@^@@PL_end_is_smaller^P^@  ^B^@@^@@^@@^@@^@@FL_done^B^@^F^@
^U^@      ^B^@@^@@^@@^@@^@@KL_check_mmx^_^@      ^B^@@^@@^@@^@@^@@UL_decode_distance_mmx^X
^@      ^B^@@^@@^@@^@@^@@NL_dont_use_mmx^V^@      ^B^@@^@@^@@^@@^@@
```

L_is_aligned^X^@	^B^@^@^@^@^@^@^@^@NL_check_window ^@	^B^@^@^@^@^@^@^@^@VL_conti
guous_in_window^X^@	^B^@^@^@^@^@^@^@^@NL_buf_not_used^_ ^@	^B^@^@^@^@^@^@^@^@UL_updat
e_stream_state^[^@	^B^@^@^@^@^@^@^@^@QL_decode_distance^V^@	^B^@^@^@^@^@^@^@^@


```
L_while_test^^^@      ^B^@@@@@@@@^Tinflate_fast_use_mmx,^@      ^B^@@@@@@@@^"  
L_test_for_second_level_length_mmx^Z^@  ^B^@@@@@@@@^PL_check_dist_one^Z^@  ^B^@@@@@  
^@@@@^PL_end_is_smaller^\^@  ^B^@@@@@@@@^RL_check_window_mmx^Z^@  ^B^@@@@@  
^@@@@^PL_while_test_mmx^W^@  ^B^@@@@@@@@^  
L_update_hold$^@      ^B^@@@@@@@@^ZL_test_for_length_base_mmx^V^@      ^B^@@@@@  
^@@@@^@
```

L_dodist_mmx^V^@

^B^@^@^@^@^@^@

```

L_align_long^\^@      ^B^@^@^@^@^@^@^@Rinflate_fast_entry^Q^@      ^B^@^@^@^@^@^@^@
^GL_dolen^[^@      ^B^@^@^@^@^@^@^@QL_last_is_smaller^Y^@ ^B^@^@^@^@^@^@^@OL_check_mmx_pop
^P^@      ^B^@^@^@^@^@^@^@FL_done$^@      ^B^@^@^@^@^@^@^@ZL_contiguous_in_window_mmx&^@
^B^@^@^@^@^@^@^@^@invalid_distance_too_far_msgU^@^@^@X^@^@^@K^@Y^@^@^@X^@^@^@
^w^@^@^@^@Y^@^@^@K^@{^@^@^@Y^@^@^@
^@M-^@]^@^@^@Z^@^@^@K^@;^@^@^@Z^@^@^@
^@½^@^@^@^@[^@^@^@K^@Á^@^@^@[^@^@^@
^@x^@^@^@^@^\^@^@^@K^@Û^@^@^@^\^@^@^@
^@÷^@^@^@^@]^@^@^@K^@û^@^@^@]^@^@^@
^@
^A^@^@^@^@^@^@K^@Q^A^@^@^@^@^@^@
^@*^A^@^@^@_ ^@^@^@K^@.^A^@^@^@_ ^@^@^@
^@?^A^@^@ ^@^@^@K^@C^A^@^@ ^@^@^@
^@h^A^@^@K^@^@^@K^@l^A^@^@K^@^@^@
^@M-^M^A^@^@! ^@^@^@K^@M-^Q^A^@^@! ^@^@^@
^@®^A^@^@

```

^@^@^@^K^@^2^A^@^@

^@^@^@
^@C^A^@^@"^@^@^@K^@E^A^@^@"^@^@^@
^@Y^A^@^@#^@^@^@K^@á^A^@^@#^@^@^@
^@G^B^@^@\$^@^@^@K^@K^B^@^@\$^@^@^@
^@^]^B^@^@%^@^@^@K^@!^B^@^@%^@^@^@
^@?^B^@^@
^@^@^@K^@C^B^@^@
^@^@^@
^@\^B^@^@&^@^@^@K^@'\^B^@^@&^@^@^@
^@x^B^@^@/'^@^@^@K^@|^B^@^@/'^@^@^@
^@M-^L^B^@^@(^@^@^@K^@M-^P^B^@^@(^@^@^@
^@ç^B^@^@)^@^@^@K^@|^B^@^@)^@^@^@
^@Ā^B^@^@*^@^@^@K^@Ā^B^@^@*^@^@^@
^@ă^B^@^@+^@^@^@K^@é^B^@^@+^@^@^@
^@p^B^@^@,^@^@^@K^@B^C^@^@,^@^@^@
^@!^C^@^@-^@^@^@K^@%C^@^@-^@^@^@
^@I^C^@^@.^@^@^@K^@M^C^@^@.^@^@^@
^@l^C^@^@N^@^@^@K^@p^C^@^@N^@^@^@
^@M-^W^C^@^@/^@^@^@K^@M-^[^C^@^@/^@^@^@
^@Ā^C^@^@0^@^@^@K^@Ç^C^@^@0^@^@^@
^@à^C^@^@1^@^@^@K^@ă^C^@^@1^@^@^@
^@÷^C^@^@2^@^@^@K^@û^C^@^@2^@^@^@
^@

```
^D^@^@3^@^@^@K^@P^D^@^@3^@^@^@
^@$^D^@^@4^@^@^@K^@(^D^@^@4^@^@^@
^@A^D^@^@5^@^@^@K^@E^D^@^@5^@^@^@
^@_ ^D^@^@6^@^@^@K^@c^D^@^@6^@^@^@
^@v^D^@^@7^@^@^@K^@z^D^@^@7^@^@^@
^@M-^U^D^@^@8^@^@^@K^@M-^Y^D^@^@8^@^@^@
^@Æ^D^@^@Q^@^@^@K^@Ê^D^@^@Q^@^@^@
^@Û^D^@^@N^@^@^@K^@â^D^@^@N^@^@^@
^@G^E^@^@K^@^@^@K^@K^E^@^@K^@^@^@
^@, ^E^@^@P^@^@^@K^@0^E^@^@P^@^@^@
^@T^E^@^@
^@^@^@K^@X^E^@^@
^@^@^@
^@q^E^@^@9^@^@^@K^@u^E^@^@9^@^@^@
^@M-^O^E^@^@:^@^@^@K^@M-^S^E^@^@:^@^@^@
^@§^E^@^@;^@^@^@K^@«^E^@^@;^@^@^@
^@¿^E^@^@<^@^@^@K^@Ã^E^@^@<^@^@^@
^@Ö^E^@^@2^@^@^@K^@Û^E^@^@2^@^@^@
^@ë^E^@^@=^@^@^@K^@ï^E^@^@=^@^@^@
^@^E^F^@^@>^@^@^@K^@^F^@^@>^@^@^@
^@ ^F^@^@_ ^@^@^@K^@$^F^@^@_ ^@^@^@
^@5^F^@^@4^@^@^@K^@9^F^@^@4^@^@^@
^@R^F^@^@?^@^@^@K^@V^F^@^@?^@^@^@
^@e^F^@^@^@^@^@K^@i^F^@^@^@^@^@
^@}^F^@^@X^@^@^@K^@M-^A^F^@^@X^@^@^@
^@M-^ _^F^@^@^@^@^@K^@f^F^@^@^@^@^@
^@¼^F^@^@$^@^@^@K^@À^F^@^@$^@^@^@
^@Ö^F^@^@A^@^@^@K^@Ï^F^@^@A^@^@^@
^@i^F^@^@)^@^@^@K^@ó^F^@^@)^@^@^@
^@^N^G^@^@/^@^@^@K^@R^G^@^@/^@^@^@
^@"^G^@^@5^@^@^@K^@&^G^@^@5^@^@^@
^@^G^@^@B^@^@^@K^@D^G^@^@B^@^@^@
^@Z^G^@^@C^@^@^@K^@^G^@^@C^@^@^@
^@v^G^@^@#^@^@^@K^@z^G^@^@#^@^@^@
^@ ^G^@^@-^@^@^@K^@q^G^@^@-^@^@^@
^@È^G^@^@8^@^@^@K^@Ì^G^@^@8^@^@^@
^@á^G^@^@\\^@^@^@K^@â^G^@^@\\^@^@^@
^@^A^H^@^@D^@^@^@K^@E^H^@^@D^@^@^@
^@#^H^@^@^@]^@^@^@K^@/^H^@^@^@]^@^@^@
^@9^H^@^@^@"^@^@^@K^@=^H^@^@^@"^@^@^@
^@O^H^@^@+^@^@^@K^@S^H^@^@+^@^@^@
^@h^H^@^@!^@^@^@K^@l^H^@^@!^@^@^@
^@M-^I^H^@^@l^@^@^@K^@M-^M^H^@^@l^@^@^@
^@ ^H^@^@E^@^@^@K^@q^H^@^@E^@^@^@
^@¼^H^@^@F^@^@^@K^@À^H^@^@F^@^@^@
^@â^H^@^@G^@^@^@K^@æ^H^@^@G^@^@^@
^@^C      ^@^@7^@^@^@K^@G      ^@^@7^@^@^@
^@"      ^@^@H^@^@^@K^@&      ^@^@H^@^@^@
^@:      ^@^@% ^@^@^@K^@>      ^@^@% ^@^@^@
^@\\      ^@^@I^@^@^@K^@`      ^@^@I^@^@^@
^@z      ^@^@Z^@^@^@K^@~      ^@^@Z^@^@^@
^@M-^Z      ^@^@J^@^@^@K^@M-^      ^@^@J^@^@^@
^@È      ^@^@/^@^@^@K^@Ì      ^@^@/^@^@^@
^@ô      ^@^@0^@^@^@K^@ø      ^@^@0^@^@^@
^@^Q
^@^@*^@^@^@K^@^U
^@^@*^@^@^@
^@5
^@^@K^@^@^@K^@9
^@^@K^@^@^@
^@[
^@^@^@[^@^@^@K^@_
^@^@^@[^@^@^@
^@u
^@^@,^@^@^@K^@y
^@^@,^@^@^@
^@M-^X
^@^@.^@^@^@K^@M-^\\
^@^@.^@^@^@
^@»
^@^@ ^@^@^@K^@¿
^@^@ ^@^@^@
^@â
^@^@^@Y^@^@^@K^@è
```

```
^@^@^Y^@^@^@
^@
^K^@^@L^@^@^@K^@^N^K^@^@L^@^@^@
^@+^K^@^@(^@^@^@K^@/^K^@^@(^@^@^@
^@A^K^@^@3^@^@^@K^@E^K^@^@3^@^@^@
^@Y^K^@^@&^@^@^@K^@]^K^@^@&^@^@^@
^@u^K^@^@M^@^@^@K^@y^K^@^@M^@^@^@
^@M-^O^K^@^@N^@^@^@K^@M-^S^K^@^@N^@^@^@
^@^K^@^@O^@^@^@K^@^K^@^@O^@^@^@
^@Å^K^@^@6^@^@^@K^@É^K^@^@6^@^@^@
^@Û^K^@^@P^@^@^@K^@à^K^@^@P^@^@^@
^@Ø^K^@^@Q^@^@^@K^@ü^K^@^@Q^@^@^@
^@^N
```

^@^@<^@^@^@K^@^R

^@^@<^@^@^@
^@%

^@^@G^@^@^K^@)

^@^@G^@^@^@
^@F

^@^@=^@^@^@^K^@J

^@^@=^@^@^@
^@`

^@^@;^@^@^@^K^@d

^@^@;^@^@^@
^@x

^@^@B^@^@^K^@|

^@^@B^@^@^@
^@M-^R

^@^@D^@^@^K^@M-^V

^@^@D^@^@^@
^@`

^@^@M^@^@^@K^@,

^@^@M^@^@^@
^@Î

^@^@L^@^@^K^@Ï

^@^@L^@^@^@
^@i

^@^@A^@^@^K^@ó

^@^@A^@^@^@
^@

^@^@^@^@^@K^@P
^@^@^@^@^@
^@
^@^@O^@^@^@K^@(
^@^@O^@^@^@
^@D
^@^@J^@^@^@K^@H
^@^@J^@^@^@
^@r
^@^@C^@^@^@K^@v
^@^@C^@^@^@
^@M-^N
^@^@P^@^@^@K^@M-^R
^@^@P^@^@^@
^@a
^@^@I^@^@^@K^@R
^@^@I^@^@^@
^@E
^@^@E^@^@^@K^@I
^@^@E^@^@^@
^@ä
^@^@N^@^@^@K^@è
^@^@N^@^@^@
^@ý
^@^@F^@^@^@K^@A^N^@^@F^@^@^@
^@#^N^@^@H^@^@^@K^@,'^N^@^@H^@^@^@
^@;^N^@^@:^@^@^@K^@?^N^@^@:^@^@^@
^@S^N^@^@9^@^@^@K^@W^N^@^@9^@^@^@
^@q^N^@^@?^@^@^@K^@u^N^@^@?^@^@^@
^@M-^D^N^@^@O^@^@^@K^@M-^H^N^@^@O^@^@^@
^@;^N^@^@>^@^@^@K^@¥^N^@^@>^@^@^@
^@4^N^@^@Q^@^@^@K^@À^N^@^@Q^@^@^@
^@î^N^@^@K^@^@^@K^@Ï^N^@^@K^@^@^@
^@ô^N^@^@P^@^@^@K^@ø^N^@^@P^@^@^@
^@A^@^@^@F^@^N^@^@^@ðñ
^@C^@^@^@^@B^P^F^@A^B^@^@ðñ.file^@^@^@^@^@^@by^@^@g^Ainffas32.asm^@^@^@^@^@^@comp
.idü ^R^@ÿÿ^@^@C^@.text^@^@^@^@^@^@A^@^@^@C^AM-^[^G^@^@

[illegible]

```
cl /DASMV /I..\..\ /O2 /c gvmat32c.c
ml /coff /Zi /c /Flgvmat32.lst gvmat32.asm
ml /coff /Zi /c /Flinffas32.lst inffas32.asm
```

Summary

This directory contains ASM implementations of the functions
longest_match() and inflate_fast().

Use instructions

Copy these files into the zlib source directory, then run the
appropriate makefile, as suggested below.

Build instructions

* With Microsoft C and MASM:

```
nmake -f win32/Makefile.msc LOC="-DASMV -DASMINF" OBJA="gvmat32c.obj gvmat32.obj inffas32.obj"
```

* With Borland C and TASM:

```
make -f win32/Makefile.bor LOCAL_ZLIB="-DASMV -DASMINF" OBJA="gvmat32c.obj gvmat32.obj inffas32.obj" OBJPA="+gvmat32c.obj+gvmat32.obj+inffas32.obj"
```

Change in 1.01e (12 feb 05)

- Fix in zipOpen2 for globalcomment (Rolf Kalbermatter)
- Fix possible memory leak in unzip.c (Zoran Stevanovic)

Change in 1.01b (20 may 04)

- Integrate patch from Debian package (submitted by Mark Brown)
- **Add** tools mzttools from Xavier Roche

Change in 1.01 (8 may 04)

- fix buffer overrun risk in unzip.c (Xavier Roche)
- fix a minor buffer insecurity in minizip.c (Mike Whittaker)

Change in 1.00: (10 sept 03)

- rename to 1.00
- cosmetic code change

Change in 0.22: (19 May 03)

- crypting support (unless you define NOCRYPT)
- append file in existing zipfile

Change in 0.21: (10 Mar 03)

- bug fixes

Change in 0.17: (27 Jan 02)

- bug fixes

Change in 0.16: (19 Jan 02)

- Support of ioapi for virtualize zip file access

Change in 0.15: (19 Mar 98)

- fix memory leak in minizip.c

Change in 0.14: (10 Mar 98)

- fix bugs in minizip.c sample for zipping big file
- fix problem in month in date handling
- fix bug in unzlocal_GetCurrentFileInfoInternal in unzip.c for comment handling

Change in 0.13: (6 Mar 98)

- fix bugs in zip.c
- **add** real minizip sample

Change in 0.12: (4 Mar 98)

- **add** zip.c and zip.h for creates .zip file
- fix change_file_date in miniunz.c for Unix (Jean-loup Gailly)
- fix miniunz.c for file without specific record for directory

Change in 0.11: (3 Mar 98)

- fix bug in unzGetCurrentFileInfo for get extra field and comment
- enhance miniunz sample, **remove** the bad unztst.c sample

Change in 0.10: (2 Mar 98)

- fix bug in unzReadCurrentFile
- rename unzip* **to** unz* **function** and structure
- **remove** Windows-like hungary notation variable name
- modify some structure in unzip.h
- **add** some comment in source
- **remove** unzipGetcCurrentFile function
- replace ZUNZEXPORT by ZEXPORT
- **add** unzGetLocalExtrafield for get the local extrafield info
- **add** a new sample, miniunz.c

Change in 0.4: (25 Feb 98)

- suppress the type unzipFileInZip.
Only one file in the zipfile can be open at the same time
- fix some typo in code
- **added** tm_unz structure in unzip_file_info (date/time in readable format)

```
CC=cc
CFLAGS=-O -I../..

UNZ_OBJS = miniunz.o unzip.o ioapi.o ../../libz.a
ZIP_OBJS = minizip.o zip.o   ioapi.o ../../libz.a

.c.o:
    $(CC) -c $(CFLAGS) $.c

all: miniunz minizip

miniunz: $(UNZ_OBJS)
    $(CC) $(CFLAGS) -o $@ $(UNZ_OBJS)

minizip: $(ZIP_OBJS)
    $(CC) $(CFLAGS) -o $@ $(ZIP_OBJS)

test: miniunz minizip
    ./minizip test readme.txt
    ./miniunz -l test.zip
    mv readme.txt readme.old
    ./miniunz test.zip

clean:
    /bin/rm -f *.o *~ minizip miniunz
```

```
/* crypt.h -- base code for crypt/uncrypt ZIPfile
```

Version 1.01e, February 12th, 2005

Copyright (C) 1998-2005 Gilles Vollant

This code is a modified version of crypting code in Infozip distribution

The encryption/decryption parts of this source code (as opposed to the non-echoing password parts) were originally written in Europe. The whole source package can be freely distributed, including from the USA. (Prior to January 2000, re-export from the US was a violation of US law.)

This encryption code is a direct transcription of the algorithm from Roger Schlafly, described by Phil Katz in the file appnote.txt. This file (appnote.txt) is distributed with the PKZIP program (even in the version without encryption capabilities).

If you don't need crypting in your application, just define symbols NOCRYPT and NOUNCRYPT.

This code support the "Traditional PKWARE Encryption".

The new AES encryption added on Zip format by Winzip (see the page http://www.winzip.com/aes_info.htm) and PKWare PKZip 5.x Strong Encryption is not supported.

```
*/
```

```
#define CRC32(c, b) (((pcrc_32_tab+(((int)(c) ^ (b)) & 0xff))) ^ ((c) >> 8))
```

```
/*
```

```
 * Return the next byte in the pseudo-random sequence
```

```
*/
```

```
static int decrypt_byte(unsigned long* pkeys, const unsigned long* pcrc_32_tab)
```

```
{
```

```
    unsigned temp; /* POTENTIAL BUG: temp*(temp^1) may overflow in an
                     * unpredictable manner on 16-bit systems; not a problem
                     * with any known compiler so far, though */
```

```
    temp = ((unsigned)(*(pkeys+2)) & 0xffff) | 2;
    return (int)(((temp * (temp ^ 1)) >> 8) & 0xff);
```

```
}
```

```
/*
```

```
 * Update the encryption keys with the next byte of plain text
```

```
*/
```

```
static int update_keys(unsigned long* pkeys, const unsigned long* pcrc_32_tab, int c)
```

```
{
```

```
    (*(pkeys+0)) = CRC32(*(pkeys+0), c);
    (*(pkeys+1)) += (*(pkeys+0)) & 0xff;
    (*(pkeys+1)) = (*(pkeys+1)) * 134775813L + 1;
    {
        register int keyshift = (int)((*(pkeys+1)) >> 24);
        (*(pkeys+2)) = CRC32(*(pkeys+2), keyshift);
    }
```

```
    return c;
}
```

```
/*
```

```
 * Initialize the encryption keys and the random header according to
 * the given password.
```

```
*/
```

```
static void init_keys(const char* passwd, unsigned long* pkeys, const unsigned long* pcrc_32_tab)
```

```
{
```

```
    *(pkeys+0) = 305419896L;
    *(pkeys+1) = 591751049L;
    *(pkeys+2) = 878082192L;
    while (*passwd != '\0') {
        update_keys(pkeys, pcrc_32_tab, (int)*passwd);
        passwd++;
    }
```



```

}

#define zdecode(pkeys,pcrc_32_tab,c) \
    (update_keys(pkeys,pcrc_32_tab,c ^= decrypt_byte(pkeys,pcrc_32_tab)))

#define zencode(pkeys,pcrc_32_tab,c,t) \
    (t=decrypt_byte(pkeys,pcrc_32_tab), update_keys(pkeys,pcrc_32_tab,c), t^(c))

#ifndef INCLUDECRYPTINGCODE_IFCRYPTALLOWED

#define RAND_HEAD_LEN 12
    /* "last resort" source for second part of crypt seed pattern */
#define ZCR_SEED2
    /* use PI as default pattern */
    #define ZCR_SEED2 3141592654UL
    #endif

static int crypthead(passwd, buf, bufSize, pkeys, pcrc_32_tab, crcForCrypting)
    const char *passwd; /* password string */
    unsigned char *buf; /* where to write header */
    int bufSize;
    unsigned long* pkeys;
    const unsigned long* pcrc_32_tab;
    unsigned long crcForCrypting;
{
    int n; /* index in random header */
    int t; /* temporary */
    int c; /* random byte */
    unsigned char header[RAND_HEAD_LEN-2]; /* random header */
    static unsigned calls = 0; /* ensure different random header each time */

    if (bufSize<RAND_HEAD_LEN)
        return 0;

    /* First generate RAND_HEAD_LEN-2 random bytes. We encrypt the
     * output of rand() to get less predictability, since rand() is
     * often poorly implemented.
     */
    if (++calls == 1)
    {
        srand((unsigned)(time(NULL) ^ ZCR_SEED2));
    }
    init_keys(passwd, pkeys, pcrc_32_tab);
    for (n = 0; n < RAND_HEAD_LEN-2; n++)
    {
        c = (rand() >> 7) & 0xff;
        header[n] = (unsigned char)zencode(pkeys, pcrc_32_tab, c, t);
    }
    /* Encrypt random header (last two bytes is high word of crc) */
    init_keys(passwd, pkeys, pcrc_32_tab);
    for (n = 0; n < RAND_HEAD_LEN-2; n++)
    {
        buf[n] = (unsigned char)zencode(pkeys, pcrc_32_tab, header[n], t);
    }
    buf[n++] = zencode(pkeys, pcrc_32_tab, (int)(crcForCrypting >> 16) & 0xff, t);
    buf[n++] = zencode(pkeys, pcrc_32_tab, (int)(crcForCrypting >> 24) & 0xff, t);
    return n;
}

#endif

```

```
/* ioapi.c -- IO base function header for compress/uncompress .zip
   files using zlib + zip or unzip API

   Version 1.01e, February 12th, 2005

   Copyright (C) 1998-2005 Gilles Vollant
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "zlib.h"
#include "ioapi.h"

/* I've found an old Unix (a SunOS 4.1.3_U1) without all SEEK_* defined.... */

#ifndef SEEK_CUR
#define SEEK_CUR    1
#endif

#ifndef SEEK_END
#define SEEK_END    2
#endif

#ifndef SEEK_SET
#define SEEK_SET    0
#endif

voidpf ZCALLBACK fopen_file_func OF((
    voidpf opaque,
    const char* filename,
    int mode));

uLong ZCALLBACK fread_file_func OF((
    voidpf opaque,
    voidpf stream,
    void* buf,
    uLong size));

uLong ZCALLBACK fwrite_file_func OF((
    voidpf opaque,
    voidpf stream,
    const void* buf,
    uLong size));

long ZCALLBACK ftell_file_func OF((
    voidpf opaque,
    voidpf stream));

long ZCALLBACK fseek_file_func OF((
    voidpf opaque,
    voidpf stream,
    uLong offset,
    int origin));

int ZCALLBACK fclose_file_func OF((
    voidpf opaque,
    voidpf stream));

int ZCALLBACK ferror_file_func OF((
    voidpf opaque,
    voidpf stream));

voidpf ZCALLBACK fopen_file_func (opaque, filename, mode)
    voidpf opaque;
    const char* filename;
    int mode;
{
    FILE* file = NULL;
    const char* mode_fopen = NULL;
```

```
if ((mode & ZLIB_FILEFUNC_MODE_READWRITEFILTER)==ZLIB_FILEFUNC_MODE_READ)
    mode_fopen = "rb";
else
if (mode & ZLIB_FILEFUNC_MODE_EXISTING)
    mode_fopen = "r+b";
else
if (mode & ZLIB_FILEFUNC_MODE_CREATE)
    mode_fopen = "wb";

if ((filename!=NULL) && (mode_fopen != NULL))
    file = fopen(filename, mode_fopen);
return file;
}
```

```
uLong ZCALLBACK fread_file_func (opaque, stream, buf, size)
voidpf opaque;
voidpf stream;
void* buf;
uLong size;
{
    uLong ret;
    ret = (uLong)fread(buf, 1, (size_t)size, (FILE *)stream);
    return ret;
}
```

```
uLong ZCALLBACK fwrite_file_func (opaque, stream, buf, size)
voidpf opaque;
voidpf stream;
const void* buf;
uLong size;
{
    uLong ret;
    ret = (uLong)fwrite(buf, 1, (size_t)size, (FILE *)stream);
    return ret;
}
```

```
long ZCALLBACK ftell_file_func (opaque, stream)
voidpf opaque;
voidpf stream;
{
    long ret;
    ret = ftell((FILE *)stream);
    return ret;
}
```

```
long ZCALLBACK fseek_file_func (opaque, stream, offset, origin)
voidpf opaque;
voidpf stream;
uLong offset;
int origin;
{
    int fseek_origin=0;
    long ret;
    switch (origin)
    {
        case ZLIB_FILEFUNC_SEEK_CUR :
            fseek_origin = SEEK_CUR;
            break;
        case ZLIB_FILEFUNC_SEEK_END :
            fseek_origin = SEEK_END;
            break;
        case ZLIB_FILEFUNC_SEEK_SET :
            fseek_origin = SEEK_SET;
            break;
        default: return -1;
    }
    ret = 0;
    fseek((FILE *)stream, offset, fseek_origin);
    return ret;
}
```

```
int ZCALLBACK fclose_file_func (opaque, stream)
```

```
voidpf opaque;
voidpf stream;
{
    int ret;
    ret = fclose((FILE *)stream);
    return ret;
}

int ZCALLBACK ferror_file_func (opaque, stream)
voidpf opaque;
voidpf stream;
{
    int ret;
    ret = ferror((FILE *)stream);
    return ret;
}

void fill_fopen_filefunc (pzlib_filefunc_def)
zlib_filefunc_def* pzlib_filefunc_def;
{
    pzlib_filefunc_def->zopen_file = fopen_file_func;
    pzlib_filefunc_def->zread_file = fread_file_func;
    pzlib_filefunc_def->zwrite_file = fwrite_file_func;
    pzlib_filefunc_def->ztell_file = ftell_file_func;
    pzlib_filefunc_def->zseek_file = fseek_file_func;
    pzlib_filefunc_def->zclose_file = fclose_file_func;
    pzlib_filefunc_def->zerror_file = ferror_file_func;
    pzlib_filefunc_def->opaque = NULL;
}
```

```

/* ioapi.h -- IO base function header for compress/uncompress .zip
   files using zlib + zip or unzip API

   Version 1.01e, February 12th, 2005

   Copyright (C) 1998-2005 Gilles Vollant
*/

#ifndef _ZLIBIOAPI_H
#define _ZLIBIOAPI_H

#define ZLIB_FILEFUNC_SEEK_CUR (1)
#define ZLIB_FILEFUNC_SEEK_END (2)
#define ZLIB_FILEFUNC_SEEK_SET (0)

#define ZLIB_FILEFUNC_MODE_READ (1)
#define ZLIB_FILEFUNC_MODE_WRITE (2)
#define ZLIB_FILEFUNC_MODE_READWRITEFILTER (3)

#define ZLIB_FILEFUNC_MODE_EXISTING (4)
#define ZLIB_FILEFUNC_MODE_CREATE (8)

#ifndef ZCALLBACK
#define ZCALLBACK
#endif

#if (defined(WIN32) || defined(WINDOWS) || defined(_WINDOWS)) && defined(CALLBACK) && defined(USEWINDOWS_CALLBACK)
#define ZCALLBACK CALLBACK
#else
#define ZCALLBACK
#endif

#ifdef __cplusplus
extern "C" {
#endif

typedef voidpf (ZCALLBACK *open_file_func) OF((voidpf opaque, const char* filename, int mode));
typedef uLong (ZCALLBACK *read_file_func) OF((voidpf opaque, voidpf stream, void* buf, uLong size));
typedef uLong (ZCALLBACK *write_file_func) OF((voidpf opaque, voidpf stream, const void* buf, uLong size));
typedef long (ZCALLBACK *tell_file_func) OF((voidpf opaque, voidpf stream));
typedef long (ZCALLBACK *seek_file_func) OF((voidpf opaque, voidpf stream, uLong offset, int origin));
typedef int (ZCALLBACK *close_file_func) OF((voidpf opaque, voidpf stream));
typedef int (ZCALLBACK *testerror_file_func) OF((voidpf opaque, voidpf stream));

typedef struct zlib_filefunc_def_s
{
    open_file_func      zopen_file;
    read_file_func      zread_file;
    write_file_func     zwrite_file;
    tell_file_func      ztell_file;
    seek_file_func      zseek_file;
    close_file_func     zclose_file;
    testerror_file_func zerror_file;
    voidpf              opaque;
} zlib_filefunc_def;

void fill_fopen_filefunc OF((zlib_filefunc_def* pzlib_filefunc_def));

#define ZREAD(filefunc, filestream, buf, size) ((*((filefunc).zread_file))((filefunc).opaque, filestream, buf, size))
#define ZWRITE(filefunc, filestream, buf, size) ((*((filefunc).zwrite_file))((filefunc).opaque, filestream, buf, size))
#define ZTELL(filefunc, filestream) ((*((filefunc).ztell_file))((filefunc).opaque, filestream))
#define ZSEEK(filefunc, filestream, pos, mode) ((*((filefunc).zseek_file))((filefunc).opaque, filestream, pos, mode))

```

```
#define ZCLOSE(filefunc,filestream) ((*((filefunc).zclose_file))((filefunc).opaque,filestream))
#define ZERROR(filefunc,filestream) ((*((filefunc).zerror_file))((filefunc).opaque,filestream))

#ifdef __cplusplus
}
#endif

#endif
```

```
/* iowin32.c -- IO base function header for compress/uncompress .zip
   files using zlib + zip or unzip API
   This IO API version uses the Win32 API (for Microsoft Windows)

   Version 1.01e, February 12th, 2005

   Copyright (C) 1998-2005 Gilles Vollant
*/

#include <stdlib.h>

#include "zlib.h"
#include "ioapi.h"
#include "iowin32.h"

#ifndef INVALID_HANDLE_VALUE
#define INVALID_HANDLE_VALUE (0xFFFFFFFF)
#endif

#ifndef INVALID_SET_FILE_POINTER
#define INVALID_SET_FILE_POINTER ((DWORD)-1)
#endif

voidpf ZCALLBACK win32_open_file_func OF((
    voidpf opaque,
    const char* filename,
    int mode));

uLong ZCALLBACK win32_read_file_func OF((
    voidpf opaque,
    voidpf stream,
    void* buf,
    uLong size));

uLong ZCALLBACK win32_write_file_func OF((
    voidpf opaque,
    voidpf stream,
    const void* buf,
    uLong size));

long ZCALLBACK win32_tell_file_func OF((
    voidpf opaque,
    voidpf stream));

long ZCALLBACK win32_seek_file_func OF((
    voidpf opaque,
    voidpf stream,
    uLong offset,
    int origin));

int ZCALLBACK win32_close_file_func OF((
    voidpf opaque,
    voidpf stream));

int ZCALLBACK win32_error_file_func OF((
    voidpf opaque,
    voidpf stream));

typedef struct
{
    HANDLE hf;
    int error;
} WIN32FILE_IOWIN;

voidpf ZCALLBACK win32_open_file_func (opaque, filename, mode)
    voidpf opaque;
    const char* filename;
    int mode;
{
    const char* mode_fopen = NULL;
    DWORD dwDesiredAccess, dwCreationDisposition, dwShareMode, dwFlagsAndAttributes ;
    HANDLE hFile = 0;
    voidpf ret=NULL;
```

```

dwDesiredAccess = dwShareMode = dwFlagsAndAttributes = 0;

if ((mode & ZLIB_FILEFUNC_MODE_READWRITEFILTER)==ZLIB_FILEFUNC_MODE_READ)
{
    dwDesiredAccess = GENERIC_READ;
    dwCreationDisposition = OPEN_EXISTING;
    dwShareMode = FILE_SHARE_READ;
}
else
if (mode & ZLIB_FILEFUNC_MODE_EXISTING)
{
    dwDesiredAccess = GENERIC_WRITE | GENERIC_READ;
    dwCreationDisposition = OPEN_EXISTING;
}
else
if (mode & ZLIB_FILEFUNC_MODE_CREATE)
{
    dwDesiredAccess = GENERIC_WRITE | GENERIC_READ;
    dwCreationDisposition = CREATE_ALWAYS;
}

if ((filename!=NULL) && (dwDesiredAccess != 0))
    hFile = CreateFile((LPCTSTR)filename, dwDesiredAccess, dwShareMode, NULL,
                      dwCreationDisposition, dwFlagsAndAttributes, NULL);

if (hFile == INVALID_HANDLE_VALUE)
    hFile = NULL;

if (hFile != NULL)
{
    WIN32FILE_IOWIN w32fiow;
    w32fiow.hf = hFile;
    w32fiow.error = 0;
    ret = malloc(sizeof(WIN32FILE_IOWIN));
    if (ret==NULL)
        CloseHandle(hFile);
    else *((WIN32FILE_IOWIN*)ret) = w32fiow;
}
return ret;
}

uLong ZCALLBACK win32_read_file_func (opaque, stream, buf, size)
voidpf opaque;
voidpf stream;
void* buf;
uLong size;
{
    uLong ret=0;
    HANDLE hFile = NULL;
    if (stream!=NULL)
        hFile = ((WIN32FILE_IOWIN*)stream) -> hf;
    if (hFile != NULL)
        if (!ReadFile(hFile, buf, size, &ret, NULL))
        {
            DWORD dwErr = GetLastError();
            if (dwErr == ERROR_HANDLE_EOF)
                dwErr = 0;
            ((WIN32FILE_IOWIN*)stream) -> error=(int)dwErr;
        }

    return ret;
}

uLong ZCALLBACK win32_write_file_func (opaque, stream, buf, size)
voidpf opaque;
voidpf stream;
const void* buf;
uLong size;
{
    uLong ret=0;
    HANDLE hFile = NULL;
    if (stream!=NULL)

```



```
hFile = ((WIN32FILE_IOWIN*)stream) -> hf;

if (hFile !=NULL)
    if (!WriteFile(hFile, buf, size, &ret, NULL))
    {
        DWORD dwErr = GetLastError();
        if (dwErr == ERROR_HANDLE_EOF)
            dwErr = 0;
        ((WIN32FILE_IOWIN*)stream) -> error=(int)dwErr;
    }

return ret;
}

long ZCALLBACK win32_tell_file_func (opaque, stream)
voidpf opaque;
voidpf stream;
{
    long ret=-1;
    HANDLE hFile = NULL;
    if (stream!=NULL)
        hFile = ((WIN32FILE_IOWIN*)stream) -> hf;
    if (hFile != NULL)
    {
        DWORD dwSet = SetFilePointer(hFile, 0, NULL, FILE_CURRENT);
        if (dwSet == INVALID_SET_FILE_POINTER)
        {
            DWORD dwErr = GetLastError();
            ((WIN32FILE_IOWIN*)stream) -> error=(int)dwErr;
            ret = -1;
        }
        else
            ret=(long)dwSet;
    }
    return ret;
}

long ZCALLBACK win32_seek_file_func (opaque, stream, offset, origin)
voidpf opaque;
voidpf stream;
uLong offset;
int origin;
{
    DWORD dwMoveMethod=0xFFFFFFFF;
    HANDLE hFile = NULL;

    long ret=-1;
    if (stream!=NULL)
        hFile = ((WIN32FILE_IOWIN*)stream) -> hf;
    switch (origin)
    {
        case ZLIB_FILEFUNC_SEEK_CUR :
            dwMoveMethod = FILE_CURRENT;
            break;
        case ZLIB_FILEFUNC_SEEK_END :
            dwMoveMethod = FILE_END;
            break;
        case ZLIB_FILEFUNC_SEEK_SET :
            dwMoveMethod = FILE_BEGIN;
            break;
        default: return -1;
    }

    if (hFile != NULL)
    {
        DWORD dwSet = SetFilePointer(hFile, offset, NULL, dwMoveMethod);
        if (dwSet == INVALID_SET_FILE_POINTER)
        {
            DWORD dwErr = GetLastError();
            ((WIN32FILE_IOWIN*)stream) -> error=(int)dwErr;
            ret = -1;
        }
        else
            ret=0;
    }
}
```

```
    }
    return ret;
}

int ZCALLBACK win32_close_file_func (opaque, stream)
voidpf opaque;
voidpf stream;
{
    int ret=-1;

    if (stream!=NULL)
    {
        HANDLE hFile;
        hFile = ((WIN32FILE_IOWIN*)stream) -> hf;
        if (hFile != NULL)
        {
            CloseHandle(hFile);
            ret=0;
        }
        free(stream);
    }
    return ret;
}

int ZCALLBACK win32_error_file_func (opaque, stream)
voidpf opaque;
voidpf stream;
{
    int ret=-1;
    if (stream!=NULL)
    {
        ret = ((WIN32FILE_IOWIN*)stream) -> error;
    }
    return ret;
}

void fill_win32_filefunc (pzlib_filefunc_def)
zlib_filefunc_def* pzlib_filefunc_def;
{
    pzlib_filefunc_def->zopen_file = win32_open_file_func;
    pzlib_filefunc_def->zread_file = win32_read_file_func;
    pzlib_filefunc_def->zwrite_file = win32_write_file_func;
    pzlib_filefunc_def->ztell_file = win32_tell_file_func;
    pzlib_filefunc_def->zseek_file = win32_seek_file_func;
    pzlib_filefunc_def->zclose_file = win32_close_file_func;
    pzlib_filefunc_def->zerror_file = win32_error_file_func;
    pzlib_filefunc_def->opaque=NULL;
}
```

```
/* iowin32.h -- IO base function header for compress/uncompress .zip
   files using zlib + zip or unzip API
   This IO API version uses the Win32 API (for Microsoft Windows)

   Version 1.01e, February 12th, 2005

   Copyright (C) 1998-2005 Gilles Vollant
*/

#include <windows.h>

#ifdef __cplusplus
extern "C" {
#endif

void fill_win32_filefunc OF((zlib_filefunc_def* pzlib_filefunc_def));

#ifdef __cplusplus
}
#endif
```

```
/*
    miniunz.c
    Version 1.01e, February 12th, 2005

    Copyright (C) 1998-2005 Gilles Vollant
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <fcntl.h>

#ifdef unix
# include <unistd.h>
# include <utime.h>
#else
# include <direct.h>
# include <io.h>
#endif

#include "unzip.h"

#define CASESENSITIVITY (0)
#define WRITEBUFFERSIZE (8192)
#define MAXFILENAME (256)

#ifdef WIN32
#define USEWIN32IOAPI
#include "iowin32.h"
#endif
/*
    mini unzip, demo of unzip package

    usage :
    Usage : miniunz [-exvlo] file.zip [file_to_extract] [-d extractdir]

    list the file in the zipfile, and print the content of FILE_ID.ZIP or README.TXT
    if it exists
*/

/*
    change_file_date : change the date/time of a file
    filename : the filename of the file where date/time must be modified
    dosdate : the new date at the MSDos format (4 bytes)
    tmu_date : the SAME new date at the tm_unz format */
void change_file_date(filename,dosdate,tmu_date)
    const char *filename;
    uLong dosdate;
    tm_unz tmu_date;
{
#ifdef WIN32
    HANDLE hFile;
    FILETIME ftm,ftLocal,ftCreate,ftLastAcc,ftLastWrite;

    hFile = CreateFile(filename,GENERIC_READ | GENERIC_WRITE,
        0,NULL,OPEN_EXISTING,0,NULL);
    GetFileTime(hFile,&ftCreate,&ftLastAcc,&ftLastWrite);
    DosDateTimeToFileTime((WORD)(dosdate>>16),(WORD)dosdate,&ftLocal);
    LocalFileTimeToFileTime(&ftLocal,&ftm);
    SetFileTime(hFile,&ftm,&ftLastAcc,&ftm);
    CloseHandle(hFile);
#else
#ifdef unix
    struct utimbuf ut;
    struct tm newdate;
    newdate.tm_sec = tmu_date.tm_sec;
    newdate.tm_min=tmu_date.tm_min;
    newdate.tm_hour=tmu_date.tm_hour;
    newdate.tm_mday=tmu_date.tm_mday;
    newdate.tm_mon=tmu_date.tm_mon;
    if (tmu_date.tm_year > 1900)

```

```
    newdate.tm_year=tmu_date.tm_year - 1900;
else
    newdate.tm_year=tmu_date.tm_year ;
newdate.tm_isdst=-1;

    ut.actime=ut.modtime=mktime(&newdate);
    utime(filename,&ut);
#endif
#endif
}

/* mymkdir and change_file_date are not 100 % portable
   As I don't know well Unix, I wait feedback for the unix portion */

int mymkdir(dirname)
    const char* dirname;
{
    int ret=0;
#ifdef WIN32
    ret = mkdir(dirname);
#else
#ifdef unix
    ret = mkdir (dirname,0775);
#endif
#endif
    return ret;
}

int makedir (newdir)
    char *newdir;
{
    char *buffer ;
    char *p;
    int len = (int)strlen(newdir);

    if (len <= 0)
        return 0;

    buffer = (char*)malloc(len+1);
    strcpy(buffer,newdir);

    if (buffer[len-1] == '/') {
        buffer[len-1] = '\0';
    }
    if (mymkdir(buffer) == 0)
    {
        free(buffer);
        return 1;
    }

    p = buffer+1;
    while (1)
    {
        char hold;

        while(*p && *p != '\\' && *p != '/')
            p++;
        hold = *p;
        *p = 0;
        if ((mymkdir(buffer) == -1) && (errno == ENOENT))
        {
            printf("couldn't create directory %s\n",buffer);
            free(buffer);
            return 0;
        }
        if (hold == 0)
            break;
        *p++ = hold;
    }
    free(buffer);
    return 1;
}
```

```

void do_banner()
{
    printf( "MiniUnz 1.01b, demo of zLib + Unz package written by Gilles Vollant\n" );
    printf( "more info at http://www.winimage.com/zLibDll/unzip.html\n\n" );
}

void do_help()
{
    printf( "Usage : miniunz [-e] [-x] [-v] [-l] [-o] [-p password] file.zip [file_to_extr.] [-d extractdir]\n\n" \
        "  -e Extract without pathname (junk paths)\n" \
        "  -x Extract with pathname\n" \
        "  -v list files\n" \
        "  -l list files\n" \
        "  -d directory to extract into\n" \
        "  -o overwrite files without prompting\n" \
        "  -p extract crypted file using password\n\n" );
}

int do_list(uf)
    unzFile uf;
{
    uLong i;
    unz_global_info gi;
    int err;

    err = unzGetGlobalInfo (uf,&gi);
    if (err!=UNZ_OK)
        printf( "error %d with zipfile in unzGetGlobalInfo\n",err );
    printf( " Length Method Size Ratio Date   Time   CRC-32   Name\n" );
    printf( "-----" );
    for (i=0;i<gi.number_entry;i++)
    {
        char filename_inzip[256];
        unz_file_info file_info;
        uLong ratio=0;
        const char *string_method;
        char charCrypt=' ';
        err = unzGetCurrentFileInfo(uf,&file_info,filename_inzip,sizeof(filename_inzip),N
ULL,0,NULL,0);
        if (err!=UNZ_OK)
        {
            printf( "error %d with zipfile in unzGetCurrentFileInfo\n",err );
            break;
        }
        if (file_info.uncompressed_size>0)
            ratio = (file_info.compressed_size*100)/file_info.uncompressed_size;

        /* display a '*' if the file is crypted */
        if ((file_info.flag & 1) != 0)
            charCrypt='*';

        if (file_info.compression_method==0)
            string_method="Stored";
        else
            if (file_info.compression_method==Z_DEFLATED)
            {
                uInt iLevel=(uInt)((file_info.flag & 0x6)/2);
                if (iLevel==0)
                    string_method="Defl:N";
                else if (iLevel==1)
                    string_method="Defl:X";
                else if ((iLevel==2) || (iLevel==3))
                    string_method="Defl:F"; /* 2:fast , 3 : extra fast*/
            }
        else
            string_method="Unkn. ";

        printf( "%7lu %6s%c%7lu %3lu%% %2.2lu-%2.2lu-%2.2lu %2.2lu:%2.2lu %8.8lx %s\n",
            file_info.uncompressed_size,string_method,
            charCrypt,
            file_info.compressed_size,
            ratio,
            (uLong)file_info.tmu_date.tm_mon + 1,

```

```
        (uLong)file_info.tmu_date.tm_mday,
        (uLong)file_info.tmu_date.tm_year % 100,
        (uLong)file_info.tmu_date.tm_hour, (uLong)file_info.tmu_date.tm_min,
        (uLong)file_info.crc, filename_inzip);
    if ((i+1)<gi.number_entry)
    {
        err = unzGoToNextFile(uf);
        if (err!=UNZ_OK)
        {
            printf("error %d with zipfile in unzGoToNextFile\n",err);
            break;
        }
    }
}

return 0;
}

int do_extract_currentfile(uf,popt_extract_without_path,popt_overwrite,password)
unzFile uf;
const int* popt_extract_without_path;
int* popt_overwrite;
const char* password;
{
    char filename_inzip[256];
    char* filename_withoutpath;
    char* p;
    int err=UNZ_OK;
    FILE *fout=NULL;
    void* buf;
    uInt size_buf;

    unz_file_info file_info;
    uLong ratio=0;
    err = unzGetCurrentFileInfo(uf,&file_info,filename_inzip,sizeof(filename_inzip),NULL,
0,NULL,0);

    if (err!=UNZ_OK)
    {
        printf("error %d with zipfile in unzGetCurrentFileInfo\n",err);
        return err;
    }

    size_buf = WRITEBUFFERSIZE;
    buf = (void*)malloc(size_buf);
    if (buf==NULL)
    {
        printf("Error allocating memory\n");
        return UNZ_INTERNALERROR;
    }

    p = filename_withoutpath = filename_inzip;
    while ((*p) != '\0')
    {
        if (((*p)=='/') || ((*p)=='\\'))
            filename_withoutpath = p+1;
        p++;
    }

    if ((*filename_withoutpath)=='\0')
    {
        if ((*popt_extract_without_path)==0)
        {
            printf("creating directory: %s\n",filename_inzip);
            mymkdir(filename_inzip);
        }
    }
    else
    {
        const char* write_filename;
        int skip=0;

        if ((*popt_extract_without_path)==0)
```

```
        write_filename = filename_inzip;
    else
        write_filename = filename_withoutpath;

    err = unzOpenCurrentFilePassword(uf,password);
    if (err!=UNZ_OK)
    {
        printf("error %d with zipfile in unzOpenCurrentFilePassword\n",err);
    }

    if (((*popt_overwrite)==0) && (err==UNZ_OK))
    {
        char rep=0;
        FILE* ftestexist;
        ftestexist = fopen(write_filename,"rb");
        if (ftestexist!=NULL)
        {
            fclose(ftestexist);
            do
            {
                char answer[128];
                int ret;

                printf("The file %s exists. Overwrite ? [y]es, [n]o, [A]ll: ",write_filename);
                ret = scanf("%ls",answer);
                if (ret != 1)
                {
                    exit(EXIT_FAILURE);
                }
                rep = answer[0];
                if ((rep>='a') && (rep<='z'))
                    rep -= 0x20;
            } while ((rep!='Y') && (rep!='N') && (rep!='A'));
        }

        if (rep == 'N')
            skip = 1;

        if (rep == 'A')
            *popt_overwrite=1;
    }

    if ((skip==0) && (err==UNZ_OK))
    {
        fout=fopen(write_filename,"wb");

        /* some zipfile don't contain directory alone before file */
        if ((fout==NULL) && ((*popt_extract_without_path)==0) &&
            (filename_withoutpath!=(char*)filename_inzip))
        {
            char c=(filename_withoutpath-1);
            *(filename_withoutpath-1)='\0';
            mkdir(write_filename);
            *(filename_withoutpath-1)=c;
            fout=fopen(write_filename,"wb");
        }

        if (fout==NULL)
        {
            printf("error opening %s\n",write_filename);
        }
    }

    if (fout!=NULL)
    {
        printf(" extracting: %s\n",write_filename);

        do
        {
            err = unzReadCurrentFile(uf,buf,size_buf);
            if (err<0)
            {
                printf("error %d with zipfile in unzReadCurrentFile\n",err);
            }
        }
    }
}
```



```

        break;
    }
    if (err>0)
        if (fwrite(buf,err,1,fout)!=1)
        {
            printf("error in writing extracted file\n");
            err=UNZ_ERRNO;
            break;
        }
    }
    while (err>0);
    if (fout)
        fclose(fout);

    if (err==0)
        change_file_date(write_filename,file_info.dosDate,
                        file_info.tmu_date);
}

if (err==UNZ_OK)
{
    err = unzCloseCurrentFile (uf);
    if (err!=UNZ_OK)
    {
        printf("error %d with zipfile in unzCloseCurrentFile\n",err);
    }
}
else
    unzCloseCurrentFile(uf); /* don't lose the error */
}

free(buf);
return err;
}

int do_extract(uf,opt_extract_without_path,opt_overwrite,password)
unzFile uf;
int opt_extract_without_path;
int opt_overwrite;
const char* password;
{
    uLong i;
    unz_global_info gi;
    int err;
    FILE* fout=NULL;

    err = unzGetGlobalInfo (uf,&gi);
    if (err!=UNZ_OK)
        printf("error %d with zipfile in unzGetGlobalInfo \n",err);

    for (i=0;i<gi.number_entry;i++)
    {
        if (do_extract_currentfile(uf,&opt_extract_without_path,
                                &opt_overwrite,
                                password) != UNZ_OK)

            break;

        if ((i+1)<gi.number_entry)
        {
            err = unzGoToNextFile(uf);
            if (err!=UNZ_OK)
            {
                printf("error %d with zipfile in unzGoToNextFile\n",err);
                break;
            }
        }
    }

    return 0;
}

int do_extract_onefile(uf,filename,opt_extract_without_path,opt_overwrite,password)
unzFile uf;

```

```
const char* filename;
int opt_extract_without_path;
int opt_overwrite;
const char* password;
{
    int err = UNZ_OK;
    if (unzLocateFile(uf, filename, CASESENSITIVITY) != UNZ_OK)
    {
        printf("file %s not found in the zipfile\n", filename);
        return 2;
    }

    if (do_extract_currentfile(uf, &opt_extract_without_path,
                                &opt_overwrite,
                                password) == UNZ_OK)

        return 0;
    else
        return 1;
}

int main(argc, argv)
    int argc;
    char *argv[];
{
    const char *zipfilename=NULL;
    const char *filename_to_extract=NULL;
    const char *password=NULL;
    char filename_try[MAXFILENAME+16] = "";
    int i;
    int opt_do_list=0;
    int opt_do_extract=1;
    int opt_do_extract_withoutpath=0;
    int opt_overwrite=0;
    int opt_extractdir=0;
    const char *dirname=NULL;
    unzFile uf=NULL;

    do_banner();
    if (argc==1)
    {
        do_help();
        return 0;
    }
    else
    {
        for (i=1; i<argc; i++)
        {
            if ((*argv[i])=='-')
            {
                const char *p=argv[i]+1;

                while ((*p)!='\0')
                {
                    char c=*(p++);
                    if ((c=='l') || (c=='L'))
                        opt_do_list = 1;
                    if ((c=='v') || (c=='V'))
                        opt_do_list = 1;
                    if ((c=='x') || (c=='X'))
                        opt_do_extract = 1;
                    if ((c=='e') || (c=='E'))
                        opt_do_extract = opt_do_extract_withoutpath = 1;
                    if ((c=='o') || (c=='O'))
                        opt_overwrite=1;
                    if ((c=='d') || (c=='D'))
                    {
                        opt_extractdir=1;
                        dirname=argv[i+1];
                    }

                    if (((c=='p') || (c=='P')) && (i+1<argc))
                    {
                        password=argv[i+1];
                    }
                }
            }
        }
    }
}
```

```

        i++;
    }
}
else
{
    if (zipfilename == NULL)
        zipfilename = argv[i];
    else if ((filename_to_extract==NULL) && (!opt_extractdir))
        filename_to_extract = argv[i] ;
}
}

if (zipfilename!=NULL)
{
#       ifdef USEWIN32IOAPI
#       zlib_filefunc_def ffunc;
#       endif

    strncpy(filename_try, zipfilename,MAXFILENAME-1);
    /* strncpy doesnt append the trailing NULL, of the string is too long. */
    filename_try[ MAXFILENAME ] = '\0';

#       ifdef USEWIN32IOAPI
#       fill_win32_filefunc(&ffunc);
#       uf = unzOpen2(zipfilename,&ffunc);
#       else
#       uf = unzOpen(zipfilename);
#       endif
#       if (uf==NULL)
#       {
#           strcat(filename_try, ".zip");
#           ifdef USEWIN32IOAPI
#           uf = unzOpen2(filename_try,&ffunc);
#           else
#           uf = unzOpen(filename_try);
#           endif
#       }

    if (uf==NULL)
    {
        printf("Cannot open %s or %s.zip\n", zipfilename, zipfilename);
        return 1;
    }
    printf("%s opened\n", filename_try);

    if (opt_do_list==1)
        return do_list(uf);
    else if (opt_do_extract==1)
    {
        if (opt_extractdir && chdir(dirname))
        {
            printf("Error changing into %s, aborting\n",  dirname);
            exit(-1);
        }

        if (filename_to_extract == NULL)
            return do_extract(uf,opt_do_extract_withoutpath,opt_overwrite,password);
        else
            return do_extract_onefile(uf,filename_to_extract,
                                     opt_do_extract_withoutpath,opt_overwrite,password);
    }
    unzCloseCurrentFile(uf);

    return 0;
}

```

```
/*
  minizip.c
  Version 1.01e, February 12th, 2005

  Copyright (C) 1998-2005 Gilles Vollant
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <fcntl.h>

#ifdef unix
# include <unistd.h>
# include <utime.h>
# include <sys/types.h>
# include <sys/stat.h>
#else
# include <direct.h>
# include <io.h>
#endif

#include "zip.h"

#ifdef WIN32
#define USEWIN32IOAPI
#include "iowin32.h"
#endif

#define WRITEBUFFERSIZE (16384)
#define MAXFILENAME (256)

#ifdef WIN32
uLong filetime(f, tmzip, dt)
    char *f; /* name of file to get info on */
    tm_zip *tmzip; /* return value: access, modific. and creation times */
    uLong *dt; /* dostime */
{
    int ret = 0;
    {
        FILETIME ftLocal;
        HANDLE hFind;
        WIN32_FIND_DATA ff32;

        hFind = FindFirstFile(f,&ff32);
        if (hFind != INVALID_HANDLE_VALUE)
        {
            FileTimeToLocalFileTime(&(ff32.ftLastWriteTime),&ftLocal);
            FileTimeToDosDateTime(&ftLocal,((LPWORD)dt)+1,((LPWORD)dt)+0);
            FindClose(hFind);
            ret = 1;
        }
    }
    return ret;
}
#else
#define unix
uLong filetime(f, tmzip, dt)
    char *f; /* name of file to get info on */
    tm_zip *tmzip; /* return value: access, modific. and creation times */
    uLong *dt; /* dostime */
{
    int ret=0;
    struct stat s; /* results of stat() */
    struct tm* filedate;
    time_t tm_t=0;

    if (strcmp(f,"-")!=0)
    {
        char name[MAXFILENAME+1];

```

```

    int len = strlen(f);
    if (len > MAXFILENAME)
        len = MAXFILENAME;

    strncpy(name, f, MAXFILENAME-1);
    /* strncpy doesnt append the trailing NULL, of the string is too long. */
    name[ MAXFILENAME ] = '\0';

    if (name[len - 1] == '/')
        name[len - 1] = '\0';
    /* not all systems allow stat'ing a file with / appended */
    if (stat(name, &s) == 0)
    {
        tm_t = s.st_mtime;
        ret = 1;
    }
}
filedate = localtime(&tm_t);

tmzip->tm_sec = filedate->tm_sec;
tmzip->tm_min = filedate->tm_min;
tmzip->tm_hour = filedate->tm_hour;
tmzip->tm_mday = filedate->tm_mday;
tmzip->tm_mon = filedate->tm_mon;
tmzip->tm_year = filedate->tm_year;

return ret;
}
#else
uLong filetime(f, tmzip, dt)
    char *f; /* name of file to get info on */
    tm_zip *tmzip; /* return value: access, modific. and creation times */
    uLong *dt; /* dostime */
{
    return 0;
}
#endif
#endif

int check_exist_file(filename)
    const char* filename;
{
    FILE* ftestexist;
    int ret = 1;
    ftestexist = fopen(filename, "rb");
    if (ftestexist == NULL)
        ret = 0;
    else
        fclose(ftestexist);
    return ret;
}

void do_banner()
{
    printf("MiniZip 1.01b, demo of zlib + Zip package written by Gilles Vollant\n");
    printf("more info at http://www.winimage.com/zLibDll/unzip.html\n\n");
}

void do_help()
{
    printf("Usage : minizip [-o] [-a] [-0 to -9] [-p password] file.zip [files_to_add]\n\n" \
        " -o Overwrite existing file.zip\n" \
        " -a Append to existing file.zip\n" \
        " -0 Store only\n" \
        " -1 Compress faster\n" \
        " -9 Compress better\n\n");
}

/* calculate the CRC32 of a file,
   because to encrypt a file, we need known the CRC32 of the file before */
int getFileCrc(const char* filename_inzip, void*buf, unsigned long size_buf, unsigned long* r

```

```
result_crc)
{
    unsigned long calculate_crc=0;
    int err=ZIP_OK;
    FILE * fin = fopen(filenameinzip,"rb");
    unsigned long size_read = 0;
    unsigned long total_read = 0;
    if (fin==NULL)
    {
        err = ZIP_ERRNO;
    }

    if (err == ZIP_OK)
        do
        {
            err = ZIP_OK;
            size_read = (int)fread(buf,1,size_buf,fin);
            if (size_read < size_buf)
                if (feof(fin)==0)
                {
                    printf("error in reading %s\n",filenameinzip);
                    err = ZIP_ERRNO;
                }

            if (size_read>0)
                calculate_crc = crc32(calculate_crc,buf,size_read);
            total_read += size_read;

        } while ((err == ZIP_OK) && (size_read>0));

    if (fin)
        fclose(fin);

    *result_crc=calculate_crc;
    printf("file %s crc %x\n",filenameinzip,calculate_crc);
    return err;
}

int main(argc,argv)
int argc;
char *argv[];
{
    int i;
    int opt_overwrite=0;
    int opt_compress_level=Z_DEFAULT_COMPRESSION;
    int zipfilenamearg = 0;
    char filename_try[MAXFILENAME+16];
    int zipok;
    int err=0;
    int size_buf=0;
    void* buf=NULL;
    const char* password=NULL;

    do_banner();
    if (argc==1)
    {
        do_help();
        return 0;
    }
    else
    {
        for (i=1;i<argc;i++)
        {
            if ((*argv[i])=='-')
            {
                const char *p=argv[i]+1;

                while ((*p)!='\0')
                {
                    char c=*(p++);
                    if ((c=='o') || (c=='O'))
                        opt_overwrite = 1;
                    if ((c=='a') || (c=='A'))

```

```

        opt_overwrite = 2;
        if ((c>='0') && (c<='9'))
            opt_compress_level = c-'0';

        if (((c=='p') || (c=='P')) && (i+1<argc))
        {
            password=argv[i+1];
            i++;
        }
    }
    else
        if (zipfilenamearg == 0)
            zipfilenamearg = i ;
}

size_buf = WRITEBUFFERSIZE;
buf = (void*)malloc(size_buf);
if (buf==NULL)
{
    printf("Error allocating memory\n");
    return ZIP_INTERNALERROR;
}

if (zipfilenamearg==0)
    zipok=0;
else
{
    int i,len;
    int dot_found=0;

    zipok = 1 ;
    strncpy(filename_try, argv[zipfilenamearg],MAXFILENAME-1);
    /* strncpy doesnt append the trailing NULL, of the string is too long. */
    filename_try[ MAXFILENAME ] = '\0';

    len=(int)strlen(filename_try);
    for (i=0;i<len;i++)
        if (filename_try[i]=='.')
            dot_found=1;

    if (dot_found==0)
        strcat(filename_try, ".zip");

    if (opt_overwrite==2)
    {
        /* if the file don't exist, we not append file */
        if (check_exist_file(filename_try)==0)
            opt_overwrite=1;
    }
    else
        if (opt_overwrite==0)
            if (check_exist_file(filename_try)!=0)
            {
                char rep=0;
                do
                {
                    char answer[128];
                    int ret;
                    printf("The file %s exists. Overwrite ? [y]es, [n]o, [a]ppend : ", filename_try);
                    ret = scanf("%ls", answer);
                    if (ret != 1)
                    {
                        exit(EXIT_FAILURE);
                    }
                    rep = answer[0] ;
                    if ((rep>='a') && (rep<='z'))
                        rep -= 0x20;
                }
                while ((rep!='Y') && (rep!='N') && (rep!='A'));
                if (rep=='N')
                    zipok = 0;
                if (rep=='A')

```

```

        opt_overwrite = 2;
    }
}

if (zipok==1)
{
    zipFile zf;
    int errclose;
#   ifdef USEWIN32IOAPI
        zlib_filefunc_def ffunc;
        fill_win32_filefunc(&ffunc);
        zf = zipOpen2(filename_try,(opt_overwrite==2) ? 2 : 0,NULL,&ffunc);
#   else
        zf = zipOpen(filename_try,(opt_overwrite==2) ? 2 : 0);
#   endif

    if (zf == NULL)
    {
        printf("error opening %s\n",filename_try);
        err= ZIP_ERRNO;
    }
    else
        printf("creating %s\n",filename_try);

    for (i=zipfilenamearg+1;(i<argc) && (err==ZIP_OK);i++)
    {
        if (!((((*(argv[i]))=='-') || ((*(argv[i]))=='/')) &&
            ((argv[i][1]=='o') || (argv[i][1]=='O') ||
             (argv[i][1]=='a') || (argv[i][1]=='A') ||
             (argv[i][1]=='p') || (argv[i][1]=='P') ||
             ((argv[i][1]>='0') || (argv[i][1]<='9')))) &&
            (strlen(argv[i]) == 2)))
        {
            FILE * fin;
            int size_read;
            const char* filenameinzip = argv[i];
            zip_fileinfo zi;
            unsigned long crcFile=0;

            zi.tmz_date.tm_sec = zi.tmz_date.tm_min = zi.tmz_date.tm_hour =
            zi.tmz_date.tm_mday = zi.tmz_date.tm_mon = zi.tmz_date.tm_year = 0;
            zi.dosDate = 0;
            zi.internal_fa = 0;
            zi.external_fa = 0;
            filetime(filenameinzip,&zi.tmz_date,&zi.dosDate);

/*
            err = zipOpenNewFileInZip(zf,filenameinzip,&zi,
                                     NULL,0,NULL,0,NULL /* comment */ ,
                                     (opt_compress_level != 0) ? Z_DEFLATED : 0,
                                     opt_compress_level);
*/

            if ((password != NULL) && (err==ZIP_OK))
                err = getFileCrc(filenameinzip,buf,size_buf,&crcFile);

            err = zipOpenNewFileInZip3(zf,filenameinzip,&zi,
                                     NULL,0,NULL,0,NULL /* comment*/,
                                     (opt_compress_level != 0) ? Z_DEFLATED : 0,
                                     opt_compress_level,0,
                                     /* -MAX_WBITS, DEF_MEM_LEVEL, Z_DEFAULT_STRATEGY, */
                                     -MAX_WBITS, DEF_MEM_LEVEL, Z_DEFAULT_STRATEGY,
                                     password,crcFile);

            if (err != ZIP_OK)
                printf("error in opening %s in zipfile\n", filenameinzip);
            else
            {
                fin = fopen(filenameinzip,"rb");
                if (fin==NULL)
                {
                    err=ZIP_ERRNO;
                    printf("error in opening %s for reading\n", filenameinzip);
                }
            }
        }
    }
}

```



```
        if (err == ZIP_OK)
            do
            {
                err = ZIP_OK;
                size_read = (int)fread(buf,1,size_buf,fin);
                if (size_read < size_buf)
                    if (feof(fin)==0)
                    {
                        printf("error in reading %s\n",filenameinzip);
                        err = ZIP_ERRNO;
                    }

                if (size_read>0)
                {
                    err = zipWriteInFileInZip (zf,buf,size_read);
                    if (err<0)
                    {
                        printf("error in writing %s in the zipfile\n",
                               filenameinzip);
                    }
                }
            } while ((err == ZIP_OK) && (size_read>0));

        if (fin)
            fclose(fin);

        if (err<0)
            err=ZIP_ERRNO;
        else
        {
            err = zipCloseFileInZip(zf);
            if (err!=ZIP_OK)
                printf("error in closing %s in the zipfile\n",
                       filenameinzip);
        }
    }
}
errclose = zipClose(zf,NULL);
if (errclose != ZIP_OK)
    printf("error in closing %s\n",filename_try);
}
else
{
    do_help();
}

free(buf);
return 0;
}
```

```

/*
  Additional tools for Minizip
  Code: Xavier Roche '2004
  License: Same as ZLIB (www.gzip.org)
*/

/* Code */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "zlib.h"
#include "unzip.h"

#define READ_8(adr) ((unsigned char)*(adr))
#define READ_16(adr) ( READ_8(adr) | (READ_8(adr+1) << 8) )
#define READ_32(adr) ( READ_16(adr) | (READ_16((adr)+2) << 16) )

#define WRITE_8(buff, n) do { \
    *((unsigned char*)(buff)) = (unsigned char) ((n) & 0xff); \
} while(0)
#define WRITE_16(buff, n) do { \
    WRITE_8((unsigned char*)(buff), n); \
    WRITE_8(((unsigned char*)(buff)) + 1, (n) >> 8); \
} while(0)
#define WRITE_32(buff, n) do { \
    WRITE_16((unsigned char*)(buff), (n) & 0xffff); \
    WRITE_16((unsigned char*)(buff) + 2, (n) >> 16); \
} while(0)

extern int ZEXPORT unzRepair(file, fileOut, fileOutTmp, nRecovered, bytesRecovered)
const char* file;
const char* fileOut;
const char* fileOutTmp;
uLong* nRecovered;
uLong* bytesRecovered;
{
    int err = Z_OK;
    FILE* fpZip = fopen(file, "rb");
    FILE* fpOut = fopen(fileOut, "wb");
    FILE* fpOutCD = fopen(fileOutTmp, "wb");
    if (fpZip != NULL && fpOut != NULL) {
        int entries = 0;
        uLong totalBytes = 0;
        char header[30];
        char filename[256];
        char extra[1024];
        int offset = 0;
        int offsetCD = 0;
        while ( fread(header, 1, 30, fpZip) == 30 ) {
            int currentOffset = offset;

            /* File entry */
            if (READ_32(header) == 0x04034b50) {
                unsigned int version = READ_16(header + 4);
                unsigned int gpflag = READ_16(header + 6);
                unsigned int method = READ_16(header + 8);
                unsigned int filetime = READ_16(header + 10);
                unsigned int filedate = READ_16(header + 12);
                unsigned int crc = READ_32(header + 14); /* crc */
                unsigned int cpsize = READ_32(header + 18); /* compressed size */
                unsigned int uncpsize = READ_32(header + 22); /* uncompressed sz */
                unsigned int fnsize = READ_16(header + 26); /* file name length */
                unsigned int extsize = READ_16(header + 28); /* extra field length */
                filename[0] = extra[0] = '\0';

                /* Header */
                if (fwrite(header, 1, 30, fpOut) == 30) {
                    offset += 30;
                } else {
                    err = Z_ERRNO;
                    break;
                }

                /* Filename */

```

```
if (fnsiz > 0) {
    if (fread(filename, 1, fnsiz, fpZip) == fnsiz) {
        if (fwrite(filename, 1, fnsiz, fpOut) == fnsiz) {
            offset += fnsiz;
        } else {
            err = Z_ERRNO;
            break;
        }
    } else {
        err = Z_ERRNO;
        break;
    }
} else {
    err = Z_STREAM_ERROR;
    break;
}

/* Extra field */
if (extsiz > 0) {
    if (fread(extra, 1, extsiz, fpZip) == extsiz) {
        if (fwrite(extra, 1, extsiz, fpOut) == extsiz) {
            offset += extsiz;
        } else {
            err = Z_ERRNO;
            break;
        }
    } else {
        err = Z_ERRNO;
        break;
    }
}

/* Data */
{
    int dataSize = cpsiz;
    if (dataSize == 0) {
        dataSize = uncpsiz;
    }
    if (dataSize > 0) {
        char* data = malloc(dataSize);
        if (data != NULL) {
            if ((int)fread(data, 1, dataSize, fpZip) == dataSize) {
                if ((int)fwrite(data, 1, dataSize, fpOut) == dataSize) {
                    offset += dataSize;
                    totalBytes += dataSize;
                } else {
                    err = Z_ERRNO;
                }
            } else {
                err = Z_ERRNO;
            }
            free(data);
            if (err != Z_OK) {
                break;
            }
        } else {
            err = Z_MEM_ERROR;
            break;
        }
    }
}

/* Central directory entry */
{
    char header[46];
    char* comment = "";
    int comsiz = (int) strlen(comment);
    WRITE_32(header, 0x02014b50);
    WRITE_16(header + 4, version);
    WRITE_16(header + 6, version);
    WRITE_16(header + 8, gpflag);
    WRITE_16(header + 10, method);
    WRITE_16(header + 12, filetime);
    WRITE_16(header + 14, filedate);
}
```

```
WRITE_32(header + 16, crc);
WRITE_32(header + 20, cpsize);
WRITE_32(header + 24, uncpsize);
WRITE_16(header + 28, fnsz);
WRITE_16(header + 30, extsz);
WRITE_16(header + 32, comsz);
WRITE_16(header + 34, 0); /* disk # */
WRITE_16(header + 36, 0); /* int attrb */
WRITE_32(header + 38, 0); /* ext attrb */
WRITE_32(header + 42, currentOffset);
/* Header */
if (fwrite(header, 1, 46, fpOutCD) == 46) {
    offsetCD += 46;

    /* Filename */
    if (fnsz > 0) {
        if (fwrite(filename, 1, fnsz, fpOutCD) == fnsz) {
            offsetCD += fnsz;
        } else {
            err = Z_ERRNO;
            break;
        }
    } else {
        err = Z_STREAM_ERROR;
        break;
    }

    /* Extra field */
    if (extsz > 0) {
        if (fwrite(extra, 1, extsz, fpOutCD) == extsz) {
            offsetCD += extsz;
        } else {
            err = Z_ERRNO;
            break;
        }
    }

    /* Comment field */
    if (comsz > 0) {
        if ((int)fwrite(comment, 1, comsz, fpOutCD) == comsz) {
            offsetCD += comsz;
        } else {
            err = Z_ERRNO;
            break;
        }
    }

    } else {
        err = Z_ERRNO;
        break;
    }
}

/* Success */
entries++;

} else {
    break;
}

}

/* Final central directory */
{
    int entriesZip = entries;
    char header[22];
    char* comment = ""; /* "ZIP File recovered by zlib/minizip/mztools" */
    int comsz = (int) strlen(comment);
    if (entriesZip > 0xffff) {
        entriesZip = 0xffff;
    }
    WRITE_32(header, 0x06054b50);
    WRITE_16(header + 4, 0); /* disk # */
    WRITE_16(header + 6, 0); /* disk # */
}
```

```
WRITE_16(header + 8, entriesZip); /* hack */
WRITE_16(header + 10, entriesZip); /* hack */
WRITE_32(header + 12, offsetCD); /* size of CD */
WRITE_32(header + 16, offset); /* offset to CD */
WRITE_16(header + 20, comsize); /* comment */

/* Header */
if (fwrite(header, 1, 22, fpOutCD) == 22) {

    /* Comment field */
    if (comsize > 0) {
        if ((int)fwrite(comment, 1, comsize, fpOutCD) != comsize) {
            err = Z_ERRNO;
        }
    }

    } else {
        err = Z_ERRNO;
    }
}

/* Final merge (file + central directory) */
fclose(fpOutCD);
if (err == Z_OK) {
    fpOutCD = fopen(fileOutTmp, "rb");
    if (fpOutCD != NULL) {
        int nRead;
        char buffer[8192];
        while ( (nRead = (int)fread(buffer, 1, sizeof(buffer), fpOutCD)) > 0) {
            if ((int)fwrite(buffer, 1, nRead, fpOut) != nRead) {
                err = Z_ERRNO;
                break;
            }
        }
        fclose(fpOutCD);
    }
}

/* Close */
fclose(fpZip);
fclose(fpOut);

/* Wipe temporary file */
(void)remove(fileOutTmp);

/* Number of recovered entries */
if (err == Z_OK) {
    if (nRecovered != NULL) {
        *nRecovered = entries;
    }
    if (bytesRecovered != NULL) {
        *bytesRecovered = totalBytes;
    }
}
} else {
    err = Z_STREAM_ERROR;
}
return err;
}
```

```
/*
  Additional tools for Minizip
  Code: Xavier Roche '2004
  License: Same as ZLIB (www.gzip.org)
*/

#ifndef _zip_tools_H
#define _zip_tools_H

#ifdef __cplusplus
extern "C" {
#endif

#ifndef _ZLIB_H
#include "zlib.h"
#endif

#include "unzip.h"

/* Repair a ZIP file (missing central directory)
  file: file to recover
  fileOut: output file after recovery
  fileOutTmp: temporary file name used for recovery
*/
extern int ZEXPORT unzRepair(const char* file,
                             const char* fileOut,
                             const char* fileOutTmp,
                             uLong* nRecovered,
                             uLong* bytesRecovered);

#endif
```

```
/* unzip.c -- IO for uncompress .zip files using zlib
   Version 1.01e, February 12th, 2005

   Copyright (C) 1998-2005 Gilles Vollant

   Read unzip.h for more info
*/

/* Decryption code comes from crypt.c by Info-ZIP but has been greatly reduced in terms o
f compatibility with older software. The following is from the original crypt.c. Code
woven in by Terry Thorsen 1/2003.
*/
/*
   Copyright (c) 1990-2000 Info-ZIP.  All rights reserved.

   See the accompanying file LICENSE, version 2000-Apr-09 or later
   (the contents of which are also included in zip.h) for terms of use.
   If, for some reason, all these files are missing, the Info-ZIP license
   also may be found at:  ftp://ftp.info-zip.org/pub/infozip/license.html
*/
/*
   crypt.c (full version) by Info-ZIP.      Last revised:  [see crypt.h]

   The encryption/decryption parts of this source code (as opposed to the
   non-echoing password parts) were originally written in Europe.  The
   whole source package can be freely distributed, including from the USA.
   (Prior to January 2000, re-export from the US was a violation of US law.)
*/
/*
   This encryption code is a direct transcription of the algorithm from
   Roger Schlafly, described by Phil Katz in the file appnote.txt.  This
   file (appnote.txt) is distributed with the PKZIP program (even in the
   version without encryption capabilities).
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "zlib.h"
#include "unzip.h"

#ifdef STDC
# include <stddef.h>
# include <string.h>
# include <stdlib.h>
#endif
#ifdef NO_ERRNO_H
extern int errno;
#else
# include <errno.h>
#endif

#ifdef local
# define local static
#endif
/* compile with -Dlocal if your debugger can't find static symbols */

#ifdef CASESENSITIVITYDEFAULT_NO
# if !defined(unix) && !defined(CASESENSITIVITYDEFAULT_YES)
#   define CASESENSITIVITYDEFAULT_NO
# endif
#endif

#ifdef UNZ_BUFSIZE
#define UNZ_BUFSIZE (16384)
#else
#define UNZ_BUFSIZE 16384
#endif

#ifdef UNZ_MAXFILENAMEINZIP
```

```

#define UNZ_MAXFILENAMEINZIP (256)
#endif

#ifndef ALLOC
#define ALLOC(size) (malloc(size))
#endif
#ifndef TRYFREE
#define TRYFREE(p) {if (p) free(p);}
#endif

#define SIZECENTRALDIRITEM (0x2e)
#define SIZEZIPLOCALHEADER (0x1e)

const char unz_copyright[] =
    " unzip 1.01 Copyright 1998-2004 Gilles Vollant - http://www.winimage.com/zLibDll";

/* unz_file_info_interntal contain internal info about a file in zipfile*/
typedef struct unz_file_info_internal_s
{
    uLong offset_curfile; /* relative offset of local header 4 bytes */
} unz_file_info_internal;

/* file_in_zip_read_info_s contain internal information about a file in zipfile,
when reading and decompress it */
typedef struct
{
    char *read_buffer; /* internal buffer for compressed data */
    z_stream stream; /* zlib stream structure for inflate */

    uLong pos_in_zipfile; /* position in byte on the zipfile, for fseek*/
    uLong stream_initialised; /* flag set if stream structure is initialised*/

    uLong offset_local_extrafield; /* offset of the local extra field */
    uInt size_local_extrafield; /* size of the local extra field */
    uLong pos_local_extrafield; /* position in the local extra field in read*/

    uLong crc32; /* crc32 of all data uncompressed */
    uLong crc32_wait; /* crc32 we must obtain after decompress all */
    uLong rest_read_compressed; /* number of byte to be decompressed */
    uLong rest_read_uncompressed; /* number of byte to be obtained after decomp*/
    zlib_filefunc_def z_filefunc;
    voidpf filestream; /* io structure of the zipfile */
    uLong compression_method; /* compression method (0==store) */
    uLong byte_before_the_zipfile; /* byte before the zipfile, (>0 for sfx)*/
    int raw;
} file_in_zip_read_info_s;

/* unz_s contain internal information about the zipfile
*/
typedef struct
{
    zlib_filefunc_def z_filefunc;
    voidpf filestream; /* io structure of the zipfile */
    unz_global_info gi; /* public global information */
    uLong byte_before_the_zipfile; /* byte before the zipfile, (>0 for sfx)*/
    uLong num_file; /* number of the current file in the zipfile*/
    uLong pos_in_central_dir; /* pos of the current file in the central dir*/
    uLong current_file_ok; /* flag about the usability of the current file*/
    uLong central_pos; /* position of the beginning of the central dir*/

    uLong size_central_dir; /* size of the central directory */
    uLong offset_central_dir; /* offset of start of central directory with
respect to the starting disk number */

    unz_file_info cur_file_info; /* public info about the current file in zip*/
    unz_file_info_internal cur_file_info_internal; /* private info about it*/
    file_in_zip_read_info_s* pfile_in_zip_read; /* structure about the current
file if we are decompressing it */
    int encrypted;

```



```

#   ifndef NOUNCRYPT
    unsigned long keys[3];      /* keys defining the pseudo-random sequence */
    const unsigned long* pcr_32_tab;
#   endif
} unz_s;

#ifndef NOUNCRYPT
#include "crypt.h"
#endif

/* =====
   Read a byte from a gz_stream; update next_in and avail_in. Return EOF
   for end of file.
   IN assertion: the stream s has been sucessfully opened for reading.
*/

local int unzlocal_getByte OF((
    const zlib_filefunc_def* pzlib_filefunc_def,
    voidpf filestream,
    int *pi));

local int unzlocal_getByte(pzlib_filefunc_def,filestream,pi)
    const zlib_filefunc_def* pzlib_filefunc_def;
    voidpf filestream;
    int *pi;
{
    unsigned char c;
    int err = (int)ZREAD(*pzlib_filefunc_def,filestream,&c,1);
    if (err==1)
    {
        *pi = (int)c;
        return UNZ_OK;
    }
    else
    {
        if (ZERROR(*pzlib_filefunc_def,filestream))
            return UNZ_ERRNO;
        else
            return UNZ_EOF;
    }
}

/* =====
   Reads a long in LSB order from the given gz_stream. Sets
*/
local int unzlocal_getShort OF((
    const zlib_filefunc_def* pzlib_filefunc_def,
    voidpf filestream,
    uLong *pX));

local int unzlocal_getShort (pzlib_filefunc_def,filestream,pX)
    const zlib_filefunc_def* pzlib_filefunc_def;
    voidpf filestream;
    uLong *pX;
{
    uLong x ;
    int i;
    int err;

    err = unzlocal_getByte(pzlib_filefunc_def,filestream,&i);
    x = (uLong)i;

    if (err==UNZ_OK)
        err = unzlocal_getByte(pzlib_filefunc_def,filestream,&i);
    x += ((uLong)i)<<8;

    if (err==UNZ_OK)
        *pX = x;
    else
        *pX = 0;
    return err;
}

```

```
}

local int unzlocal_getLong OF((
    const zlib_filefunc_def* pzlib_filefunc_def,
    voidpf filestream,
    uLong *pX));

local int unzlocal_getLong (pzlib_filefunc_def,filestream,pX)
    const zlib_filefunc_def* pzlib_filefunc_def;
    voidpf filestream;
    uLong *pX;
{
    uLong x ;
    int i;
    int err;

    err = unzlocal_getByte(pzlib_filefunc_def,filestream,&i);
    x = (uLong)i;

    if (err==UNZ_OK)
        err = unzlocal_getByte(pzlib_filefunc_def,filestream,&i);
    x += ((uLong)i)<<8;

    if (err==UNZ_OK)
        err = unzlocal_getByte(pzlib_filefunc_def,filestream,&i);
    x += ((uLong)i)<<16;

    if (err==UNZ_OK)
        err = unzlocal_getByte(pzlib_filefunc_def,filestream,&i);
    x += ((uLong)i)<<24;

    if (err==UNZ_OK)
        *pX = x;
    else
        *pX = 0;
    return err;
}

/* My own strcmpi / strcasecmp */
local int strcmpcasenosensitive_internal (fileName1,fileName2)
    const char* fileName1;
    const char* fileName2;
{
    for (;;)
    {
        char c1=*(fileName1++);
        char c2=*(fileName2++);
        if ((c1>='a') && (c1<='z'))
            c1 -= 0x20;
        if ((c2>='a') && (c2<='z'))
            c2 -= 0x20;
        if (c1=='\0')
            return ((c2=='\0') ? 0 : -1);
        if (c2=='\0')
            return 1;
        if (c1<c2)
            return -1;
        if (c1>c2)
            return 1;
    }
}

#ifdef CASESENSITIVITYDEFAULT_NO
#define CASESENSITIVITYDEFAULTVALUE 2
#else
#define CASESENSITIVITYDEFAULTVALUE 1
#endif

#ifndef STRCMPCASENOSENTIVEFUNCTION
#define STRCMPCASENOSENTIVEFUNCTION strcmpcasenosensitive_internal
#endif
```

```

/*
  Compare two filename (fileName1,fileName2).
  If iCaseSensitivity = 1, comparison is case sensitivity (like strcmp)
  If iCaseSensitivity = 2, comparison is not case sensitivity (like strcmpi
                                     or strcasecmp)
  If iCaseSensitivity = 0, case sensitivity is default of your operating system
    (like 1 on Unix, 2 on Windows)
*/
extern int ZEXPORT unzStringFileNameCompare (fileName1,fileName2,iCaseSensitivity)
const char* fileName1;
const char* fileName2;
int iCaseSensitivity;
{
    if (iCaseSensitivity==0)
        iCaseSensitivity=CASESENSITIVITYDEFAULTVALUE;

    if (iCaseSensitivity==1)
        return strcmp(fileName1,fileName2);

    return STRCMPCASENOSENTIVEFUNCTION(fileName1,fileName2);
}

#ifdef BUFREADCOMMENT
#define BUFREADCOMMENT (0x400)
#endif

/*
  Locate the Central directory of a zipfile (at the end, just before
  the global comment)
*/
local uLong unzlocal_SearchCentralDir OF((
    const zlib_filefunc_def* pzlib_filefunc_def,
    voidpf filestream));

local uLong unzlocal_SearchCentralDir(pzlib_filefunc_def,filestream)
const zlib_filefunc_def* pzlib_filefunc_def;
voidpf filestream;
{
    unsigned char* buf;
    uLong uSizeFile;
    uLong uBackRead;
    uLong uMaxBack=0xffff; /* maximum size of global comment */
    uLong uPosFound=0;

    if (ZSEEK(*pzlib_filefunc_def,filestream,0,ZLIB_FILEFUNC_SEEK_END) != 0)
        return 0;

    uSizeFile = ZTELL(*pzlib_filefunc_def,filestream);

    if (uMaxBack>uSizeFile)
        uMaxBack = uSizeFile;

    buf = (unsigned char*)ALLOC(BUFREADCOMMENT+4);
    if (buf==NULL)
        return 0;

    uBackRead = 4;
    while (uBackRead<uMaxBack)
    {
        uLong uReadSize,uReadPos ;
        int i;
        if (uBackRead+BUFREADCOMMENT>uMaxBack)
            uBackRead = uMaxBack;
        else
            uBackRead+=BUFREADCOMMENT;
        uReadPos = uSizeFile-uBackRead ;

        uReadSize = ((BUFREADCOMMENT+4) < (uSizeFile-uReadPos)) ?
            (BUFREADCOMMENT+4) : (uSizeFile-uReadPos);
        if (ZSEEK(*pzlib_filefunc_def,filestream,uReadPos,ZLIB_FILEFUNC_SEEK_SET)!=0)
            break;

```

```

    if (ZREAD(*pzlib_filefunc_def, filestream, buf, uReadSize) != uReadSize)
        break;

    for (i = (int)uReadSize - 3; (i--) > 0;)
        if (((*(buf + i)) == 0x50) && ((*(buf + i + 1)) == 0x4b) &&
            ((*(buf + i + 2)) == 0x05) && ((*(buf + i + 3)) == 0x06))
        {
            uPosFound = uReadPos + i;
            break;
        }

    if (uPosFound != 0)
        break;
}
TRYFREE(buf);
return uPosFound;
}

/*
  Open a Zip file. path contain the full pathname (by example,
  on a Windows NT computer "c:\\test\\zlib114.zip" or on an Unix computer
  "zlib/zlib114.zip".
  If the zipfile cannot be opened (file doesn't exist or in not valid), the
  return value is NULL.
  Else, the return value is a unzFile Handle, usable with other function
  of this unzip package.
*/
extern unzFile ZEXPORT unzOpen2 (path, pzlib_filefunc_def)
const char *path;
zlib_filefunc_def* pzlib_filefunc_def;
{
    unz_s us;
    unz_s *s;
    uLong central_pos, uL;

    uLong number_disk;           /* number of the current dist, used for
                                   spanning ZIP, unsupported, always 0 */
    uLong number_disk_with_CD;   /* number the the disk with central dir, used
                                   for spanning ZIP, unsupported, always 0 */
    uLong number_entry_CD;       /* total number of entries in
                                   the central dir
                                   (same than number_entry on nospan) */

    int err = UNZ_OK;

    if (unz_copyright[0] != ' ')
        return NULL;

    if (pzlib_filefunc_def == NULL)
        fill_fopen_filefunc(&us.z_filefunc);
    else
        us.z_filefunc = *pzlib_filefunc_def;

    us.filestream = (*(us.z_filefunc.zopen_file))(us.z_filefunc.opaque,
                                                  path,
                                                  ZLIB_FILEFUNC_MODE_READ |
                                                  ZLIB_FILEFUNC_MODE_EXISTING);

    if (us.filestream == NULL)
        return NULL;

    central_pos = unzlocal_SearchCentralDir(&us.z_filefunc, us.filestream);
    if (central_pos == 0)
        err = UNZ_ERRNO;

    if (ZSEEK(us.z_filefunc, us.filestream,
              central_pos, ZLIB_FILEFUNC_SEEK_SET) != 0)
        err = UNZ_ERRNO;

    /* the signature, already checked */
    if (unzlocal_getLong(&us.z_filefunc, us.filestream, &uL) != UNZ_OK)
        err = UNZ_ERRNO;

    /* number of this disk */
    if (unzlocal_getShort(&us.z_filefunc, us.filestream, &number_disk) != UNZ_OK)

```

```

    err=UNZ_ERRNO;

    /* number of the disk with the start of the central directory */
    if (unzlocal_getShort(&us.z_filefunc, us.filestream,&number_disk_with_CD)!=UNZ_OK)
        err=UNZ_ERRNO;

    /* total number of entries in the central dir on this disk */
    if (unzlocal_getShort(&us.z_filefunc, us.filestream,&us.gi.number_entry)!=UNZ_OK)
        err=UNZ_ERRNO;

    /* total number of entries in the central dir */
    if (unzlocal_getShort(&us.z_filefunc, us.filestream,&number_entry_CD)!=UNZ_OK)
        err=UNZ_ERRNO;

    if ((number_entry_CD!=us.gi.number_entry) ||
        (number_disk_with_CD!=0) ||
        (number_disk!=0))
        err=UNZ_BADZIPFILE;

    /* size of the central directory */
    if (unzlocal_getLong(&us.z_filefunc, us.filestream,&us.size_central_dir)!=UNZ_OK)
        err=UNZ_ERRNO;

    /* offset of start of central directory with respect to the
       starting disk number */
    if (unzlocal_getLong(&us.z_filefunc, us.filestream,&us.offset_central_dir)!=UNZ_OK)
        err=UNZ_ERRNO;

    /* zipfile comment length */
    if (unzlocal_getShort(&us.z_filefunc, us.filestream,&us.gi.size_comment)!=UNZ_OK)
        err=UNZ_ERRNO;

    if ((central_pos<us.offset_central_dir+us.size_central_dir) &&
        (err==UNZ_OK))
        err=UNZ_BADZIPFILE;

    if (err!=UNZ_OK)
    {
        ZCLOSE(us.z_filefunc, us.filestream);
        return NULL;
    }

    us.byte_before_the_zipfile = central_pos -
                                (us.offset_central_dir+us.size_central_dir);
    us.central_pos = central_pos;
    us.pfile_in_zip_read = NULL;
    us.encrypted = 0;

    s=(unz_s*)ALLOC(sizeof(unz_s));
    *s=us;
    unzGoToFirstFile((unzFile)s);
    return (unzFile)s;
}

extern unzFile ZEXPORT unzOpen (path)
    const char *path;
{
    return unzOpen2(path, NULL);
}

/*
  Close a ZipFile opened with unzipOpen.
  If there is files inside the .Zip opened with unzipOpenCurrentFile (see later),
  these files MUST be closed with unzipCloseCurrentFile before call unzipClose.
  return UNZ_OK if there is no problem. */
extern int ZEXPORT unzClose (file)
    unzFile file;
{
    unz_s* s;
    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;

```

```

    if (s->pfile_in_zip_read!=NULL)
        unzCloseCurrentFile(file);

    ZCLOSE(s->z_filefunc, s->filestream);
    TRYFREE(s);
    return UNZ_OK;
}

/*
    Write info about the ZipFile in the *pglobal_info structure.
    No preparation of the structure is needed
    return UNZ_OK if there is no problem. */
extern int ZEXPORT unzGetGlobalInfo (file,pglobal_info)
    unzFile file;
    unz_global_info *pglobal_info;
{
    unz_s* s;
    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    *pglobal_info=s->gi;
    return UNZ_OK;
}

/*
    Translate date/time from Dos format to tm_unz (readable more easilty)
*/
local void unzlocal_DosDateToTmuDate (ulDosDate, ptm)
    uLong ulDosDate;
    tm_unz* ptm;
{
    uLong uDate;
    uDate = (uLong)(ulDosDate>>16);
    ptm->tm_mday = (uInt)(uDate&0x1f) ;
    ptm->tm_mon = (uInt)((((uDate)&0x1E0)/0x20)-1) ;
    ptm->tm_year = (uInt)((((uDate)&0x0FE0)/0x0200)+1980) ;

    ptm->tm_hour = (uInt) ((ulDosDate &0xF800)/0x800);
    ptm->tm_min = (uInt) ((ulDosDate&0x7E0)/0x20) ;
    ptm->tm_sec = (uInt) (2*(ulDosDate&0x1f)) ;
}

/*
    Get Info about the current file in the zipfile, with internal only info
*/
local int unzlocal_GetCurrentFileInfoInternal OF((unzFile file,
                                                    unz_file_info *pfile_info,
                                                    unz_file_info_internal
                                                    *pfile_info_internal,
                                                    char *szFileName,
                                                    uLong fileNameBufferSize,
                                                    void *extraField,
                                                    uLong extraFieldBufferSize,
                                                    char *szComment,
                                                    uLong commentBufferSize));

local int unzlocal_GetCurrentFileInfoInternal (file,
                                                pfile_info,
                                                pfile_info_internal,
                                                szFileName, fileNameBufferSize,
                                                extraField, extraFieldBufferSize,
                                                szComment, commentBufferSize)

    unzFile file;
    unz_file_info *pfile_info;
    unz_file_info_internal *pfile_info_internal;
    char *szFileName;
    uLong fileNameBufferSize;
    void *extraField;
    uLong extraFieldBufferSize;
    char *szComment;
    uLong commentBufferSize;

```

```
{
    unz_s* s;
    unz_file_info file_info;
    unz_file_info_internal file_info_internal;
    int err=UNZ_OK;
    uLong uMagic;
    long lSeek=0;

    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    if (ZSEEK(s->z_filefunc, s->filestream,
             s->pos_in_central_dir+s->byte_before_the_zipfile,
             ZLIB_FILEFUNC_SEEK_SET)!=0)
        err=UNZ_ERRNO;

    /* we check the magic */
    if (err==UNZ_OK)
        if (unzlocal_getLong(&s->z_filefunc, s->filestream,&uMagic) != UNZ_OK)
            err=UNZ_ERRNO;
        else if (uMagic!=0x02014b50)
            err=UNZ_BADZIPFILE;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&file_info.version) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&file_info.version_needed) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&file_info.flag) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&file_info.compression_method) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getLong(&s->z_filefunc, s->filestream,&file_info.dosDate) != UNZ_OK)
        err=UNZ_ERRNO;

    unzlocal_DosDateToTmuDate(file_info.dosDate,&file_info.tmu_date);

    if (unzlocal_getLong(&s->z_filefunc, s->filestream,&file_info.crc) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getLong(&s->z_filefunc, s->filestream,&file_info.compressed_size) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getLong(&s->z_filefunc, s->filestream,&file_info.uncompressed_size) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&file_info.size_filename) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&file_info.size_file_extra) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&file_info.size_file_comment) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&file_info.disk_num_start) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&file_info.internal_fa) != UNZ_OK)
        err=UNZ_ERRNO;
}
```

```
if (unzlocal_getLong(&s->z_filefunc, s->filestream,&file_info.external_fa) != UNZ_OK)
    err=UNZ_ERRNO;

if (unzlocal_getLong(&s->z_filefunc, s->filestream,&file_info_internal.offset_curfile)
) != UNZ_OK)
    err=UNZ_ERRNO;

lSeek+=file_info.size_filename;
if ((err==UNZ_OK) && (szFileName!=NULL))
{
    uLong uSizeRead ;
    if (file_info.size_filename<fileNameBufferSize)
    {
        *(szFileName+file_info.size_filename)='\0';
        uSizeRead = file_info.size_filename;
    }
    else
        uSizeRead = fileNameBufferSize;

    if ((file_info.size_filename>0) && (fileNameBufferSize>0))
        if (ZREAD(s->z_filefunc, s->filestream,szFileName,uSizeRead)!=uSizeRead)
            err=UNZ_ERRNO;
    lSeek -= uSizeRead;
}

if ((err==UNZ_OK) && (extraField!=NULL))
{
    uLong uSizeRead ;
    if (file_info.size_file_extra<extraFieldBufferSize)
        uSizeRead = file_info.size_file_extra;
    else
        uSizeRead = extraFieldBufferSize;

    if (lSeek!=0)
        if (ZSEEK(s->z_filefunc, s->filestream,lSeek,ZLIB_FILEFUNC_SEEK_CUR)==0)
            lSeek=0;
        else
            err=UNZ_ERRNO;
    if ((file_info.size_file_extra>0) && (extraFieldBufferSize>0))
        if (ZREAD(s->z_filefunc, s->filestream,extraField,uSizeRead)!=uSizeRead)
            err=UNZ_ERRNO;
    lSeek += file_info.size_file_extra - uSizeRead;
}
else
    lSeek+=file_info.size_file_extra;

if ((err==UNZ_OK) && (szComment!=NULL))
{
    uLong uSizeRead ;
    if (file_info.size_file_comment<commentBufferSize)
    {
        *(szComment+file_info.size_file_comment)='\0';
        uSizeRead = file_info.size_file_comment;
    }
    else
        uSizeRead = commentBufferSize;

    if (lSeek!=0)
        if (ZSEEK(s->z_filefunc, s->filestream,lSeek,ZLIB_FILEFUNC_SEEK_CUR)==0)
            lSeek=0;
        else
            err=UNZ_ERRNO;
    if ((file_info.size_file_comment>0) && (commentBufferSize>0))
        if (ZREAD(s->z_filefunc, s->filestream,szComment,uSizeRead)!=uSizeRead)
            err=UNZ_ERRNO;
    lSeek+=file_info.size_file_comment - uSizeRead;
}
else
    lSeek+=file_info.size_file_comment;

if ((err==UNZ_OK) && (pfile_info!=NULL))
    *pfile_info=file_info;
```



```

    if ((err==UNZ_OK) && (pfile_info_internal!=NULL))
        *pfile_info_internal=file_info_internal;

    return err;
}

/*
Write info about the ZipFile in the *pglobal_info structure.
No preparation of the structure is needed
return UNZ_OK if there is no problem.
*/
extern int ZEXPORT unzGetCurrentFileInfo (file,
                                          pfile_info,
                                          szFileName, fileNameBufferSize,
                                          extraField, extraFieldBufferSize,
                                          szComment, commentBufferSize)

unzFile file;
unz_file_info *pfile_info;
char *szFileName;
uLong fileNameBufferSize;
void *extraField;
uLong extraFieldBufferSize;
char *szComment;
uLong commentBufferSize;
{
    return unzlocal_GetCurrentFileInfoInternal(file,pfile_info,NULL,
                                              szFileName,fileNameBufferSize,
                                              extraField,extraFieldBufferSize,
                                              szComment,commentBufferSize);
}

/*
Set the current file of the zipfile to the first file.
return UNZ_OK if there is no problem
*/
extern int ZEXPORT unzGoToFirstFile (file)
unzFile file;
{
    int err=UNZ_OK;
    unz_s* s;
    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    s->pos_in_central_dir=s->offset_central_dir;
    s->num_file=0;
    err=unzlocal_GetCurrentFileInfoInternal(file,&s->cur_file_info,
                                              &s->cur_file_info_internal,
                                              NULL,0,NULL,0,NULL,0);
    s->current_file_ok = (err == UNZ_OK);
    return err;
}

/*
Set the current file of the zipfile to the next file.
return UNZ_OK if there is no problem
return UNZ_END_OF_LIST_OF_FILE if the actual file was the latest.
*/
extern int ZEXPORT unzGoToNextFile (file)
unzFile file;
{
    unz_s* s;
    int err;

    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    if (!s->current_file_ok)
        return UNZ_END_OF_LIST_OF_FILE;
    if (s->gi.number_entry != 0xffff) /* 2^16 files overflow hack */
        if (s->num_file+1==s->gi.number_entry)
            return UNZ_END_OF_LIST_OF_FILE;
    return UNZ_OK;
}

```

```

s->pos_in_central_dir += SIZECENTRALDIRITEM + s->cur_file_info.size_filename +
    s->cur_file_info.size_file_extra + s->cur_file_info.size_file_comment ;
s->num_file++;
err = unzlocal_GetCurrentFileInfoInternal(file,&s->cur_file_info,
                                          &s->cur_file_info_internal,
                                          NULL,0,NULL,0,NULL,0);

s->current_file_ok = (err == UNZ_OK);
return err;
}

/*
Try locate the file szFileName in the zipfile.
For the iCaseSensitivity signification, see unzipStringFileNameCompare

return value :
UNZ_OK if the file is found. It becomes the current file.
UNZ_END_OF_LIST_OF_FILE if the file is not found
*/
extern int ZEXPORT unzLocateFile (file, szFileName, iCaseSensitivity)
    unzFile file;
    const char *szFileName;
    int iCaseSensitivity;
{
    unz_s* s;
    int err;

    /* We remember the 'current' position in the file so that we can jump
    * back there if we fail.
    */
    unz_file_info cur_file_infoSaved;
    unz_file_info_internal cur_file_info_internalSaved;
    uLong num_fileSaved;
    uLong pos_in_central_dirSaved;

    if (file==NULL)
        return UNZ_PARAMERROR;

    if (strlen(szFileName)>=UNZ_MAXFILENAMEINZIP)
        return UNZ_PARAMERROR;

    s=(unz_s*)file;
    if (!s->current_file_ok)
        return UNZ_END_OF_LIST_OF_FILE;

    /* Save the current state */
    num_fileSaved = s->num_file;
    pos_in_central_dirSaved = s->pos_in_central_dir;
    cur_file_infoSaved = s->cur_file_info;
    cur_file_info_internalSaved = s->cur_file_info_internal;

    err = unzGoToFirstFile(file);

    while (err == UNZ_OK)
    {
        char szCurrentFileName[UNZ_MAXFILENAMEINZIP+1];
        err = unzGetCurrentFileInfo(file,NULL,
                                   szCurrentFileName,sizeof(szCurrentFileName)-1,
                                   NULL,0,NULL,0);

        if (err == UNZ_OK)
        {
            if (unzStringFileNameCompare(szCurrentFileName,
                                       szFileName,iCaseSensitivity)==0)
                return UNZ_OK;
            err = unzGoToNextFile(file);
        }
    }

    /* We failed, so restore the state of the 'current file' to where we
    * were.
    */
    s->num_file = num_fileSaved ;

```

```

s->pos_in_central_dir = pos_in_central_dirSaved ;
s->cur_file_info = cur_file_infoSaved;
s->cur_file_info_internal = cur_file_info_internalSaved;
return err;
}

/*
////////////////////////////////////
// Contributed by Ryan Haksi (mailto://cryogen@infoserve.net)
// I need random access
//
// Further optimization could be realized by adding an ability
// to cache the directory in memory. The goal being a single
// comprehensive file read to put the file I need in a memory.
*/

/*
typedef struct unz_file_pos_s
{
    uLong pos_in_zip_directory;    // offset in file
    uLong num_of_file;            // # of file
} unz_file_pos;
*/

extern int ZEXPORT unzGetFilePos(file, file_pos)
unzFile file;
unz_file_pos* file_pos;
{
    unz_s* s;

    if (file==NULL || file_pos==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    if (!s->current_file_ok)
        return UNZ_END_OF_LIST_OF_FILE;

    file_pos->pos_in_zip_directory  = s->pos_in_central_dir;
    file_pos->num_of_file          = s->num_file;

    return UNZ_OK;
}

extern int ZEXPORT unzGoToFilePos(file, file_pos)
unzFile file;
unz_file_pos* file_pos;
{
    unz_s* s;
    int err;

    if (file==NULL || file_pos==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;

    /* jump to the right spot */
    s->pos_in_central_dir = file_pos->pos_in_zip_directory;
    s->num_file           = file_pos->num_of_file;

    /* set the current file */
    err = unzlocal_GetCurrentFileInfoInternal(file,&s->cur_file_info,
                                              &s->cur_file_info_internal,
                                              NULL,0,NULL,0,NULL,0);

    /* return results */
    s->current_file_ok = (err == UNZ_OK);
    return err;
}

/*
// Unzip Helper Functions - should be here?
////////////////////////////////////
*/

/*
Read the local header of the current zipfile

```

```

    Check the coherency of the local header and info in the end of central
    directory about this file
    store in *piSizeVar the size of extra info in local header
    (filename and size of extra field data)
*/
local int unzlocal_CheckCurrentFileCoherencyHeader (s,piSizeVar,
                                                    poffset_local_extrafield,
                                                    psize_local_extrafield)
{
    unz_s* s;
    uInt* piSizeVar;
    uLong *poffset_local_extrafield;
    uInt *psize_local_extrafield;

    uLong uMagic,uData,uFlags;
    uLong size_filename;
    uLong size_extra_field;
    int err=UNZ_OK;

    *piSizeVar = 0;
    *poffset_local_extrafield = 0;
    *psize_local_extrafield = 0;

    if (ZSEEK(s->z_filefunc, s->filestream,s->cur_file_info.internal.offset_curfile +
              s->byte_before_the_zipfile,ZLIB_FILEFUNC_SEEK_SET)!=0)
        return UNZ_ERRNO;

    if (err==UNZ_OK)
        if (unzlocal_getLong(&s->z_filefunc, s->filestream,&uMagic) != UNZ_OK)
            err=UNZ_ERRNO;
        else if (uMagic!=0x04034b50)
            err=UNZ_BADZIPFILE;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&uData) != UNZ_OK)
        err=UNZ_ERRNO;
/*
    else if ((err==UNZ_OK) && (uData!=s->cur_file_info.wVersion))
        err=UNZ_BADZIPFILE;
*/
    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&uFlags) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&uData) != UNZ_OK)
        err=UNZ_ERRNO;
    else if ((err==UNZ_OK) && (uData!=s->cur_file_info.compression_method))
        err=UNZ_BADZIPFILE;

    if ((err==UNZ_OK) && (s->cur_file_info.compression_method!=0) &&
        (s->cur_file_info.compression_method!=Z_DEFLATED))
        err=UNZ_BADZIPFILE;

    if (unzlocal_getLong(&s->z_filefunc, s->filestream,&uData) != UNZ_OK) /* date/time */
        err=UNZ_ERRNO;

    if (unzlocal_getLong(&s->z_filefunc, s->filestream,&uData) != UNZ_OK) /* crc */
        err=UNZ_ERRNO;
    else if ((err==UNZ_OK) && (uData!=s->cur_file_info.crc) &&
        ((uFlags & 8)==0))
        err=UNZ_BADZIPFILE;

    if (unzlocal_getLong(&s->z_filefunc, s->filestream,&uData) != UNZ_OK) /* size compr */
        err=UNZ_ERRNO;
    else if ((err==UNZ_OK) && (uData!=s->cur_file_info.compressed_size) &&
        ((uFlags & 8)==0))
        err=UNZ_BADZIPFILE;

    if (unzlocal_getLong(&s->z_filefunc, s->filestream,&uData) != UNZ_OK) /* size uncompr */
        err=UNZ_ERRNO;
    else if ((err==UNZ_OK) && (uData!=s->cur_file_info.uncompressed_size) &&
        ((uFlags & 8)==0))
        err=UNZ_BADZIPFILE;
}

```

```

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&size_filename) != UNZ_OK)
        err=UNZ_ERRNO;
    else if ((err==UNZ_OK) && (size_filename!=s->cur_file_info.size_filename))
        err=UNZ_BADZIPFILE;

    *piSizeVar += (uInt)size_filename;

    if (unzlocal_getShort(&s->z_filefunc, s->filestream,&size_extra_field) != UNZ_OK)
        err=UNZ_ERRNO;
    *poffset_local_extrafield= s->cur_file_info_internal.offset_curfile +
                                SIZEZIPLOCALHEADER + size_filename;
    *psize_local_extrafield = (uInt)size_extra_field;

    *piSizeVar += (uInt)size_extra_field;

    return err;
}

/*
    Open for reading data the current file in the zipfile.
    If there is no error and the file is opened, the return value is UNZ_OK.
*/
extern int ZEXPORT unzOpenCurrentFile3 (file, method, level, raw, password)
    unzFile file;
    int* method;
    int* level;
    int raw;
    const char* password;
{
    int err=UNZ_OK;
    uInt iSizeVar;
    unz_s* s;
    file_in_zip_read_info_s* pfile_in_zip_read_info;
    uLong offset_local_extrafield; /* offset of the local extra field */
    uInt size_local_extrafield; /* size of the local extra field */
#   ifndef NOUNCRYPT
    char source[12];
#   else
    if (password != NULL)
        return UNZ_PARAMERROR;
#   endif

    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    if (!s->current_file_ok)
        return UNZ_PARAMERROR;

    if (s->pfile_in_zip_read != NULL)
        unzCloseCurrentFile(file);

    if (unzlocal_CheckCurrentFileCoherencyHeader(s,&iSizeVar,
        &offset_local_extrafield,&size_local_extrafield)!=UNZ_OK)
        return UNZ_BADZIPFILE;

    pfile_in_zip_read_info = (file_in_zip_read_info_s*)
        ALLOC(sizeof(file_in_zip_read_info_s));
    if (pfile_in_zip_read_info==NULL)
        return UNZ_INTERNALERROR;

    pfile_in_zip_read_info->read_buffer=(char*)ALLOC(UNZ_BUFSIZE);
    pfile_in_zip_read_info->offset_local_extrafield = offset_local_extrafield;
    pfile_in_zip_read_info->size_local_extrafield = size_local_extrafield;
    pfile_in_zip_read_info->pos_local_extrafield=0;
    pfile_in_zip_read_info->raw=raw;

    if (pfile_in_zip_read_info->read_buffer==NULL)
    {
        TRYFREE(pfile_in_zip_read_info);
        return UNZ_INTERNALERROR;
    }

    pfile_in_zip_read_info->stream_initialised=0;

```

```

if (method!=NULL)
    *method = (int)s->cur_file_info.compression_method;

if (level!=NULL)
{
    *level = 6;
    switch (s->cur_file_info.flag & 0x06)
    {
        case 6 : *level = 1; break;
        case 4 : *level = 2; break;
        case 2 : *level = 9; break;
    }
}

if ((s->cur_file_info.compression_method!=0) &&
    (s->cur_file_info.compression_method!=Z_DEFLATED))
    err=UNZ_BADZIPFILE;

pfile_in_zip_read_info->crc32_wait=s->cur_file_info.crc;
pfile_in_zip_read_info->crc32=0;
pfile_in_zip_read_info->compression_method =
    s->cur_file_info.compression_method;
pfile_in_zip_read_info->filestream=s->filestream;
pfile_in_zip_read_info->z_filefunc=s->z_filefunc;
pfile_in_zip_read_info->byte_before_the_zipfile=s->byte_before_the_zipfile;

pfile_in_zip_read_info->stream.total_out = 0;

if ((s->cur_file_info.compression_method==Z_DEFLATED) &&
    (!raw))
{
    pfile_in_zip_read_info->stream.zalloc = (alloc_func)0;
    pfile_in_zip_read_info->stream.zfree = (free_func)0;
    pfile_in_zip_read_info->stream.opaque = (voidpf)0;
    pfile_in_zip_read_info->stream.next_in = (voidpf)0;
    pfile_in_zip_read_info->stream.avail_in = 0;

    err=inflateInit2(&pfile_in_zip_read_info->stream, -MAX_WBITS);
    if (err == Z_OK)
        pfile_in_zip_read_info->stream.initialised=1;
    else
    {
        TRYFREE(pfile_in_zip_read_info);
        return err;
    }
    /* windowBits is passed < 0 to tell that there is no zlib header.
     * Note that in this case inflate *requires* an extra "dummy" byte
     * after the compressed stream in order to complete decompression and
     * return Z_STREAM_END.
     * In unzip, i don't wait absolutely Z_STREAM_END because I known the
     * size of both compressed and uncompressed data
     */
}
pfile_in_zip_read_info->rest_read_compressed =
    s->cur_file_info.compressed_size ;
pfile_in_zip_read_info->rest_read_uncompressed =
    s->cur_file_info.uncompressed_size ;

pfile_in_zip_read_info->pos_in_zipfile =
    s->cur_file_info.internal.offset_curfile + SIZEZIPLOCALHEADER +
    iSizeVar;

pfile_in_zip_read_info->stream.avail_in = (uInt)0;

s->pfile_in_zip_read = pfile_in_zip_read_info;

#    ifndef NOUNCRYPT
if (password != NULL)
{
    int i;
    s->pcrc_32_tab = get_crc_table();
    init_keys(password,s->keys,s->pcrc_32_tab);

```

```

        if (ZSEEK(s->z_filefunc, s->filestream,
            s->pfile_in_zip_read->pos_in_zipfile +
            s->pfile_in_zip_read->byte_before_the_zipfile,
            SEEK_SET)!=0)
            return UNZ_INTERNALERROR;
        if(ZREAD(s->z_filefunc, s->filestream,source, 12)<12)
            return UNZ_INTERNALERROR;

        for (i = 0; i<12; i++)
            zdecode(s->keys,s->pcrc_32_tab,source[i]);

        s->pfile_in_zip_read->pos_in_zipfile+=12;
        s->encrypted=1;
    }
#   endif

    return UNZ_OK;
}

extern int ZEXPORT unzOpenCurrentFile (file)
    unzFile file;
{
    return unzOpenCurrentFile3(file, NULL, NULL, 0, NULL);
}

extern int ZEXPORT unzOpenCurrentFilePassword (file, password)
    unzFile file;
    const char* password;
{
    return unzOpenCurrentFile3(file, NULL, NULL, 0, password);
}

extern int ZEXPORT unzOpenCurrentFile2 (file,method,level,raw)
    unzFile file;
    int* method;
    int* level;
    int raw;
{
    return unzOpenCurrentFile3(file, method, level, raw, NULL);
}

/*
    Read bytes from the current file.
    buf contain buffer where data must be copied
    len the size of buf.

    return the number of byte copied if somes bytes are copied
    return 0 if the end of file was reached
    return <0 with error code if there is an error
        (UNZ_ERRNO for IO error, or zLib error for uncompress error)
*/
extern int ZEXPORT unzReadCurrentFile (file, buf, len)
    unzFile file;
    voidp buf;
    unsigned len;
{
    int err=UNZ_OK;
    uInt iRead = 0;
    unz_s* s;
    file_in_zip_read_info_s* pfile_in_zip_read_info;
    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    pfile_in_zip_read_info=s->pfile_in_zip_read;

    if (pfile_in_zip_read_info==NULL)
        return UNZ_PARAMERROR;

    if ((pfile_in_zip_read_info->read_buffer == NULL))
        return UNZ_END_OF_LIST_OF_FILE;
    if (len==0)
        return 0;

```

```

pfile_in_zip_read_info->stream.next_out = (Bytef*)buf;
pfile_in_zip_read_info->stream.avail_out = (uInt)len;

if ((len>pfile_in_zip_read_info->rest_read_uncompressed) &&
    (!(pfile_in_zip_read_info->raw)))
    pfile_in_zip_read_info->stream.avail_out =
        (uInt)pfile_in_zip_read_info->rest_read_uncompressed;

if ((len>pfile_in_zip_read_info->rest_read_compressed+
    pfile_in_zip_read_info->stream.avail_in) &&
    (pfile_in_zip_read_info->raw))
    pfile_in_zip_read_info->stream.avail_out =
        (uInt)pfile_in_zip_read_info->rest_read_compressed+
        pfile_in_zip_read_info->stream.avail_in;

while (pfile_in_zip_read_info->stream.avail_out>0)
{
    if ((pfile_in_zip_read_info->stream.avail_in==0) &&
        (pfile_in_zip_read_info->rest_read_compressed>0))
    {
        uInt uReadThis = UNZ_BUFSIZE;
        if (pfile_in_zip_read_info->rest_read_compressed<uReadThis)
            uReadThis = (uInt)pfile_in_zip_read_info->rest_read_compressed;
        if (uReadThis == 0)
            return UNZ_EOF;
        if (ZSEEK(pfile_in_zip_read_info->z_filefunc,
            pfile_in_zip_read_info->filestream,
            pfile_in_zip_read_info->pos_in_zipfile +
            pfile_in_zip_read_info->byte_before_the_zipfile,
            ZLIB_FILEFUNC_SEEK_SET)!=0)
            return UNZ_ERRNO;
        if (ZREAD(pfile_in_zip_read_info->z_filefunc,
            pfile_in_zip_read_info->filestream,
            pfile_in_zip_read_info->read_buffer,
            uReadThis)!=uReadThis)
            return UNZ_ERRNO;

        #ifdef NOUNCRYPT
        if(s->encrypted)
        {
            uInt i;
            for(i=0;i<uReadThis;i++)
                pfile_in_zip_read_info->read_buffer[i] =
                    zdecode(s->keys,s->pcrc_32_tab,
                        pfile_in_zip_read_info->read_buffer[i]);
        }
        #endif

        pfile_in_zip_read_info->pos_in_zipfile += uReadThis;
        pfile_in_zip_read_info->rest_read_compressed-=uReadThis;

        pfile_in_zip_read_info->stream.next_in =
            (Bytef*)pfile_in_zip_read_info->read_buffer;
        pfile_in_zip_read_info->stream.avail_in = (uInt)uReadThis;
    }

    if ((pfile_in_zip_read_info->compression_method==0) || (pfile_in_zip_read_info->r
aw))
    {
        uInt uDoCopy,i ;

        if ((pfile_in_zip_read_info->stream.avail_in == 0) &&
            (pfile_in_zip_read_info->rest_read_compressed == 0))
            return (iRead==0) ? UNZ_EOF : iRead;

        if (pfile_in_zip_read_info->stream.avail_out <
            pfile_in_zip_read_info->stream.avail_in)
            uDoCopy = pfile_in_zip_read_info->stream.avail_out ;
        else

```



```

        uDoCopy = pfile_in_zip_read_info->stream.avail_in ;

        for (i=0;i<uDoCopy;i++)
            *(pfile_in_zip_read_info->stream.next_out+i) =
                *(pfile_in_zip_read_info->stream.next_in+i);

        pfile_in_zip_read_info->crc32 = crc32(pfile_in_zip_read_info->crc32,
            pfile_in_zip_read_info->stream.next_out,
            uDoCopy);
        pfile_in_zip_read_info->rest_read_uncompressed-=uDoCopy;
        pfile_in_zip_read_info->stream.avail_in -= uDoCopy;
        pfile_in_zip_read_info->stream.avail_out -= uDoCopy;
        pfile_in_zip_read_info->stream.next_out += uDoCopy;
        pfile_in_zip_read_info->stream.next_in += uDoCopy;
        pfile_in_zip_read_info->stream.total_out += uDoCopy;
        iRead += uDoCopy;
    }
    else
    {
        uLong uTotalOutBefore,uTotalOutAfter;
        const Bytef *bufBefore;
        uLong uOutThis;
        int flush=Z_SYNC_FLUSH;

        uTotalOutBefore = pfile_in_zip_read_info->stream.total_out;
        bufBefore = pfile_in_zip_read_info->stream.next_out;

        /*
        if ((pfile_in_zip_read_info->rest_read_uncompressed ==
            pfile_in_zip_read_info->stream.avail_out) &&
            (pfile_in_zip_read_info->rest_read_compressed == 0))
            flush = Z_FINISH;
        */
        err=inflate(&pfile_in_zip_read_info->stream,flush);

        if ((err>=0) && (pfile_in_zip_read_info->stream.msg!=NULL))
            err = Z_DATA_ERROR;

        uTotalOutAfter = pfile_in_zip_read_info->stream.total_out;
        uOutThis = uTotalOutAfter-uTotalOutBefore;

        pfile_in_zip_read_info->crc32 =
            crc32(pfile_in_zip_read_info->crc32,bufBefore,
                (uInt)(uOutThis));

        pfile_in_zip_read_info->rest_read_uncompressed -=
            uOutThis;

        iRead += (uInt)(uTotalOutAfter - uTotalOutBefore);

        if (err==Z_STREAM_END)
            return (iRead==0) ? UNZ_EOF : iRead;
        if (err!=Z_OK)
            break;
    }
}

if (err==Z_OK)
    return iRead;
return err;
}

/*
    Give the current position in uncompressed data
*/
extern z_off_t ZEXPORT unztell (file)
    unzFile file;
{
    unz_s* s;
    file_in_zip_read_info_s* pfile_in_zip_read_info;
    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;

```

```
pfile_in_zip_read_info=s->pfile_in_zip_read;

if (pfile_in_zip_read_info==NULL)
    return UNZ_PARAMERROR;

return (z_off_t)pfile_in_zip_read_info->stream.total_out;
}

/*
    return 1 if the end of file was reached, 0 elsewhere
*/
extern int ZEXPORT unzEOF (file)
    unzFile file;
{
    unz_s* s;
    file_in_zip_read_info_s* pfile_in_zip_read_info;
    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    pfile_in_zip_read_info=s->pfile_in_zip_read;

    if (pfile_in_zip_read_info==NULL)
        return UNZ_PARAMERROR;

    if (pfile_in_zip_read_info->rest_read_uncompressed == 0)
        return 1;
    else
        return 0;
}

/*
    Read extra field from the current file (opened by unzOpenCurrentFile)
    This is the local-header version of the extra field (sometimes, there is
    more info in the local-header version than in the central-header)

    if buf==NULL, it return the size of the local extra field that can be read

    if buf!=NULL, len is the size of the buffer, the extra header is copied in
    buf.
    the return value is the number of bytes copied in buf, or (if <0)
    the error code
*/
extern int ZEXPORT unzGetLocalExtrafield (file,buf,len)
    unzFile file;
    voidp buf;
    unsigned len;
{
    unz_s* s;
    file_in_zip_read_info_s* pfile_in_zip_read_info;
    uInt read_now;
    uLong size_to_read;

    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    pfile_in_zip_read_info=s->pfile_in_zip_read;

    if (pfile_in_zip_read_info==NULL)
        return UNZ_PARAMERROR;

    size_to_read = (pfile_in_zip_read_info->size_local_extrafield -
        pfile_in_zip_read_info->pos_local_extrafield);

    if (buf==NULL)
        return (int)size_to_read;

    if (len>size_to_read)
        read_now = (uInt)size_to_read;
    else
        read_now = (uInt)len ;
}
```

```

    if (read_now==0)
        return 0;

    if (ZSEEK(pfile_in_zip_read_info->z_filefunc,
              pfile_in_zip_read_info->filestream,
              pfile_in_zip_read_info->offset_local_extrafield +
              pfile_in_zip_read_info->pos_local_extrafield,
              ZLIB_FILEFUNC_SEEK_SET)!=0)
        return UNZ_ERRNO;

    if (ZREAD(pfile_in_zip_read_info->z_filefunc,
              pfile_in_zip_read_info->filestream,
              buf,read_now)!=read_now)
        return UNZ_ERRNO;

    return (int)read_now;
}

/*
  Close the file in zip opened with unzipOpenCurrentFile
  Return UNZ_CRCERROR if all the file was read but the CRC is not good
*/
extern int ZEXPORT unzCloseCurrentFile (file)
    unzFile file;
{
    int err=UNZ_OK;

    unz_s* s;
    file_in_zip_read_info_s* pfile_in_zip_read_info;
    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    pfile_in_zip_read_info=s->pfile_in_zip_read;

    if (pfile_in_zip_read_info==NULL)
        return UNZ_PARAMERROR;

    if ((pfile_in_zip_read_info->rest_read_uncompressed == 0) &&
        (!pfile_in_zip_read_info->raw))
    {
        if (pfile_in_zip_read_info->crc32 != pfile_in_zip_read_info->crc32_wait)
            err=UNZ_CRCERROR;
    }

    TRYFREE(pfile_in_zip_read_info->read_buffer);
    pfile_in_zip_read_info->read_buffer = NULL;
    if (pfile_in_zip_read_info->stream_initialised)
        inflateEnd(&pfile_in_zip_read_info->stream);

    pfile_in_zip_read_info->stream_initialised = 0;
    TRYFREE(pfile_in_zip_read_info);

    s->pfile_in_zip_read=NULL;

    return err;
}

/*
  Get the global comment string of the ZipFile, in the szComment buffer.
  uSizeBuf is the size of the szComment buffer.
  return the number of byte copied or an error code <0
*/
extern int ZEXPORT unzGetGlobalComment (file, szComment, uSizeBuf)
    unzFile file;
    char *szComment;
    uLong uSizeBuf;
{
    int err=UNZ_OK;
    unz_s* s;
    uLong uReadThis ;
    if (file==NULL)

```

```
    return UNZ_PARAMERROR;
    s=(unz_s*)file;

    uReadThis = uSizeBuf;
    if (uReadThis>s->gi.size_comment)
        uReadThis = s->gi.size_comment;

    if (ZSEEK(s->z_filefunc,s->filestream,s->central_pos+22,ZLIB_FILEFUNC_SEEK_SET)!=0)
        return UNZ_ERRNO;

    if (uReadThis>0)
    {
        *szComment='\0';
        if (ZREAD(s->z_filefunc,s->filestream,szComment,uReadThis)!=uReadThis)
            return UNZ_ERRNO;
    }

    if ((szComment != NULL) && (uSizeBuf > s->gi.size_comment))
        *(szComment+s->gi.size_comment)='\0';
    return (int)uReadThis;
}

/* Additions by RX '2004 */
extern uLong ZEXPORT unzGetOffset (file)
    unzFile file;
{
    unz_s* s;

    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;
    if (!s->current_file_ok)
        return 0;
    if (s->gi.number_entry != 0 && s->gi.number_entry != 0xffff)
        if (s->num_file==s->gi.number_entry)
            return 0;
    return s->pos_in_central_dir;
}

extern int ZEXPORT unzSetOffset (file, pos)
    unzFile file;
    uLong pos;
{
    unz_s* s;
    int err;

    if (file==NULL)
        return UNZ_PARAMERROR;
    s=(unz_s*)file;

    s->pos_in_central_dir = pos;
    s->num_file = s->gi.number_entry; /* hack */
    err = unzlocal_GetCurrentFileInfoInternal(file,&s->cur_file_info,
                                                &s->cur_file_info_internal,
                                                NULL,0,NULL,0,NULL,0);

    s->current_file_ok = (err == UNZ_OK);
    return err;
}
```

```
/* unzip.h -- IO for uncompress .zip files using zlib
Version 1.01e, February 12th, 2005
```

Copyright (C) 1998-2005 Gilles Vollant

This unzip package allow extract file from .ZIP file, compatible with PKZip 2.04g
WinZip, InfoZip tools and compatible.

Multi volume ZipFile (span) are not supported.
Encryption compatible with pkzip 2.04g only supported
Old compressions used by old PKZip 1.x are not supported

I WAIT FEEDBACK at mail info@winimage.com
Visit also <http://www.winimage.com/zLibDll/unzip.htm> for evolution

Condition of use and distribution are the same than zlib :

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
claim that you wrote the original software. If you use this software
in a product, an acknowledgment in the product documentation would be
appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

```
*/
```

```
/* for more info about .ZIP format, see
   http://www.info-zip.org/pub/infozip/doc/appnote-981119-iz.zip
   http://www.info-zip.org/pub/infozip/doc/
   PkWare has also a specification at :
   ftp://ftp.pkware.com/probdesc.zip
*/
```

```
#ifndef _unz_H
#define _unz_H
```

```
#ifdef __cplusplus
extern "C" {
#endif
```

```
#ifndef _ZLIB_H
#include "zlib.h"
#endif
```

```
#ifndef _ZLIBIOAPI_H
#include "ioapi.h"
#endif
```

```
#if defined(STRICTUNZIP) || defined(STRICTZIPUNZIP)
/* like the STRICT of WIN32, we define a pointer that cannot be converted
   from (void*) without cast */
typedef struct TagunzFile__ { int unused; } unzFile__;
typedef unzFile__ *unzFile;
#else
typedef voidp unzFile;
#endif
```

```
#define UNZ_OK (0)
#define UNZ_END_OF_LIST_OF_FILE (-100)
#define UNZ_ERRNO (Z_ERRNO)
#define UNZ_EOF (0)
#define UNZ_PARAMERROR (-102)
```

```

#define UNZ_BADZIPFILE                (-103)
#define UNZ_INTERNALERROR            (-104)
#define UNZ_CRCERROR                 (-105)

/* tm_unz contain date/time info */
typedef struct tm_unz_s
{
    uInt tm_sec;           /* seconds after the minute - [0,59] */
    uInt tm_min;           /* minutes after the hour - [0,59] */
    uInt tm_hour;          /* hours since midnight - [0,23] */
    uInt tm_mday;          /* day of the month - [1,31] */
    uInt tm_mon;           /* months since January - [0,11] */
    uInt tm_year;          /* years - [1980..2044] */
} tm_unz;

/* unz_global_info structure contain global data about the ZIPfile
   These data comes from the end of central dir */
typedef struct unz_global_info_s
{
    uLong number_entry;     /* total number of entries in
                           the central dir on this disk */
    uLong size_comment;     /* size of the global comment of the zipfile */
} unz_global_info;

/* unz_file_info contain information about a file in the zipfile */
typedef struct unz_file_info_s
{
    uLong version;          /* version made by                2 bytes */
    uLong version_needed;   /* version needed to extract    2 bytes */
    uLong flag;             /* general purpose bit flag     2 bytes */
    uLong compression_method; /* compression method          2 bytes */
    uLong dosDate;          /* last mod file date in Dos fmt 4 bytes */
    uLong crc;              /* crc-32                       4 bytes */
    uLong compressed_size;  /* compressed size              4 bytes */
    uLong uncompressed_size; /* uncompressed size            4 bytes */
    uLong size_filename;    /* filename length              2 bytes */
    uLong size_file_extra;  /* extra field length           2 bytes */
    uLong size_file_comment; /* file comment length          2 bytes */

    uLong disk_num_start;   /* disk number start            2 bytes */
    uLong internal_fa;      /* internal file attributes     2 bytes */
    uLong external_fa;      /* external file attributes     4 bytes */

    tm_unz tmu_date;
} unz_file_info;

extern int ZEXPORT unzStringFileNameCompare OF ((const char* fileName1,
                                                const char* fileName2,
                                                int iCaseSensitivity));

/*
   Compare two filename (fileName1,fileName2).
   If iCaseSensitivity = 1, comparison is case sensitivity (like strcmp)
   If iCaseSensitivity = 2, comparison is not case sensitivity (like strcmpi
                           or strcasecmp)
   If iCaseSensitivity = 0, case sensitivity is default of your operating system
   (like 1 on Unix, 2 on Windows)
*/

extern unzFile ZEXPORT unzOpen OF((const char *path));

/*
   Open a Zip file. path contain the full pathname (by example,
   on a Windows XP computer "c:\zlib\zlib113.zip" or on an Unix computer
   "zlib/zlib113.zip".
   If the zipfile cannot be opened (file don't exist or in not valid), the
   return value is NULL.
   Else, the return value is a unzFile Handle, usable with other function
   of this unzip package.
*/

extern unzFile ZEXPORT unzOpen2 OF((const char *path,
                                   zlib_filefunc_def* pzlib_filefunc_def));

/*

```

```

    Open a Zip file, like unzOpen, but provide a set of file low level API
    for read/write the zip file (see ioapi.h)
*/

extern int ZEXPORT unzClose OF((unzFile file));
/*
    Close a ZipFile opened with unzipOpen.
    If there is files inside the .Zip opened with unzOpenCurrentFile (see later),
    these files MUST be closed with unzipCloseCurrentFile before call unzipClose.
    return UNZ_OK if there is no problem. */

extern int ZEXPORT unzGetGlobalInfo OF((unzFile file,
                                         unz_global_info *pglobal_info));
/*
    Write info about the ZipFile in the *pglobal_info structure.
    No preparation of the structure is needed
    return UNZ_OK if there is no problem. */

extern int ZEXPORT unzGetGlobalComment OF((unzFile file,
                                           char *szComment,
                                           uLong uSizeBuf));
/*
    Get the global comment string of the ZipFile, in the szComment buffer.
    uSizeBuf is the size of the szComment buffer.
    return the number of byte copied or an error code <0
*/

/*****
/* Unzip package allow you browse the directory of the zipfile */

extern int ZEXPORT unzGoToFirstFile OF((unzFile file));
/*
    Set the current file of the zipfile to the first file.
    return UNZ_OK if there is no problem
*/

extern int ZEXPORT unzGoToNextFile OF((unzFile file));
/*
    Set the current file of the zipfile to the next file.
    return UNZ_OK if there is no problem
    return UNZ_END_OF_LIST_OF_FILE if the actual file was the latest.
*/

extern int ZEXPORT unzLocateFile OF((unzFile file,
                                     const char *szFileName,
                                     int iCaseSensitivity));
/*
    Try locate the file szFileName in the zipfile.
    For the iCaseSensitivity signification, see unzStringFileNameCompare

    return value :
    UNZ_OK if the file is found. It becomes the current file.
    UNZ_END_OF_LIST_OF_FILE if the file is not found
*/

/* ***** */
/* Ryan supplied functions */
/* unz_file_info contain information about a file in the zipfile */
typedef struct unz_file_pos_s
{
    uLong pos_in_zip_directory; /* offset in zip file directory */
    uLong num_of_file;         /* # of file */
} unz_file_pos;

extern int ZEXPORT unzGetFilePos(
    unzFile file,
    unz_file_pos* file_pos);

extern int ZEXPORT unzGoToFilePos(
    unzFile file,
    unz_file_pos* file_pos);

```

```

/* ***** */

extern int ZEXPORT unzGetCurrentFileInfo OF((unzFile file,
      unz_file_info *pfile_info,
      char *szFileName,
      uLong fileNameBufferSize,
      void *extraField,
      uLong extraFieldBufferSize,
      char *szComment,
      uLong commentBufferSize));

/*
  Get Info about the current file
  if pfile_info!=NULL, the *pfile_info structure will contain some info about
  the current file
  if szFileName!=NULL, the filename string will be copied in szFileName
    (fileNameBufferSize is the size of the buffer)
  if extraField!=NULL, the extra field information will be copied in extraField
    (extraFieldBufferSize is the size of the buffer).
    This is the Central-header version of the extra field
  if szComment!=NULL, the comment string of the file will be copied in szComment
    (commentBufferSize is the size of the buffer)
*/

/*
  *****
  for reading the content of the current zip file, you can open it, read data
  from it, and close it (you can close it before reading all the file)
  */

extern int ZEXPORT unzOpenCurrentFile OF((unzFile file));
/*
  Open for reading data the current file in the zip file.
  If there is no error, the return value is UNZ_OK.
*/

extern int ZEXPORT unzOpenCurrentFilePassword OF((unzFile file,
      const char* password));
/*
  Open for reading data the current file in the zip file.
  password is a crypting password
  If there is no error, the return value is UNZ_OK.
*/

extern int ZEXPORT unzOpenCurrentFile2 OF((unzFile file,
      int* method,
      int* level,
      int raw));
/*
  Same than unzOpenCurrentFile, but open for read raw the file (not uncompress)
  if raw==1
  *method will receive method of compression, *level will receive level of
  compression
  note : you can set level parameter as NULL (if you did not want known level,
  but you CANNOT set method parameter as NULL
*/

extern int ZEXPORT unzOpenCurrentFile3 OF((unzFile file,
      int* method,
      int* level,
      int raw,
      const char* password));
/*
  Same than unzOpenCurrentFile, but open for read raw the file (not uncompress)
  if raw==1
  *method will receive method of compression, *level will receive level of
  compression
  note : you can set level parameter as NULL (if you did not want known level,
  but you CANNOT set method parameter as NULL
*/

extern int ZEXPORT unzCloseCurrentFile OF((unzFile file));
/*
  Close the file in zip opened with unzOpenCurrentFile

```



```
Return UNZ_CRCERROR if all the file was read but the CRC is not good
*/

extern int ZEXPORT unzReadCurrentFile OF((unzFile file,
                                         voidp buf,
                                         unsigned len));

/*
  Read bytes from the current file (opened by unzOpenCurrentFile)
  buf contain buffer where data must be copied
  len the size of buf.

  return the number of byte copied if somes bytes are copied
  return 0 if the end of file was reached
  return <0 with error code if there is an error
  (UNZ_ERRNO for IO error, or zlib error for uncompress error)
*/

extern z_off_t ZEXPORT unzTell OF((unzFile file));

/*
  Give the current position in uncompressed data
*/

extern int ZEXPORT unzEOF OF((unzFile file));

/*
  return 1 if the end of file was reached, 0 elsewhere
*/

extern int ZEXPORT unzGetLocalExtrafield OF((unzFile file,
                                             voidp buf,
                                             unsigned len));

/*
  Read extra field from the current file (opened by unzOpenCurrentFile)
  This is the local-header version of the extra field (sometimes, there is
  more info in the local-header version than in the central-header)

  if buf==NULL, it return the size of the local extra field

  if buf!=NULL, len is the size of the buffer, the extra header is copied in
  buf.
  the return value is the number of bytes copied in buf, or (if <0)
  the error code
*/

/*****

/* Get the current file offset */
extern uLong ZEXPORT unzGetOffset (unzFile file);

/* Set the current file offset */
extern int ZEXPORT unzSetOffset (unzFile file, uLong pos);

#ifdef __cplusplus
}
#endif

#endif /* _unz_H */
```

```
/* zip.c -- IO on .zip files using zlib
   Version 1.01e, February 12th, 2005

   27 Dec 2004 Rolf Kalbermatter
   Modification to zipOpen2 to support globalComment retrieval.

   Copyright (C) 1998-2005 Gilles Vollant

   Read zip.h for more info
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "zlib.h"
#include "zip.h"

#ifdef STDC
# include <stddef.h>
# include <string.h>
# include <stdlib.h>
#endif
#ifdef NO_ERRNO_H
extern int errno;
#else
# include <errno.h>
#endif

#ifdef local
# define local static
#endif
/* compile with -Dlocal if your debugger can't find static symbols */

#ifdef VERSIONMADEBY
# define VERSIONMADEBY (0x0) /* platform dependant */
#endif

#ifdef Z_BUFSIZE
# define Z_BUFSIZE (16384)
#endif

#ifdef Z_MAXFILENAMEINZIP
# define Z_MAXFILENAMEINZIP (256)
#endif

#ifdef ALLOC
# define ALLOC(size) (malloc(size))
#endif
#ifdef TRYFREE
# define TRYFREE(p) {if (p) free(p);}
#endif

/*
#define SIZECENTRALDIRITEM (0x2e)
#define SIZEZIPLOCALHEADER (0x1e)
*/

/* I've found an old Unix (a SunOS 4.1.3_U1) without all SEEK_* defined.... */

#ifndef SEEK_CUR
#define SEEK_CUR    1
#endif

#ifndef SEEK_END
#define SEEK_END    2
#endif

#ifndef SEEK_SET
#define SEEK_SET    0
#endif
```

```
#ifndef DEF_MEM_LEVEL
#define MAX_MEM_LEVEL 8
#define DEF_MEM_LEVEL 8
#else
#define DEF_MEM_LEVEL MAX_MEM_LEVEL
#endif
#endif
const char zip_copyright[] =
    " zip 1.01 Copyright 1998-2004 Gilles Vollant - http://www.winimage.com/zLibDll " ;

#define SIZEDATA_INDATABLOCK (4096-(4*4))

#define LOCALHEADERMAGIC (0x04034b50)
#define CENTRALHEADERMAGIC (0x02014b50)
#define ENDHEADERMAGIC (0x06054b50)

#define FLAG_LOCALHEADER_OFFSET (0x06)
#define CRC_LOCALHEADER_OFFSET (0x0e)

#define SIZECENTRALHEADER (0x2e) /* 46 */

typedef struct linkedlist_datablock_internal_s
{
    struct linkedlist_datablock_internal_s* next_datablock;
    uLong avail_in_this_block;
    uLong filled_in_this_block;
    uLong unused; /* for future use and alignement */
    unsigned char data[SIZEDATA_INDATABLOCK];
} linkedlist_datablock_internal;

typedef struct linkedlist_data_s
{
    linkedlist_datablock_internal* first_block;
    linkedlist_datablock_internal* last_block;
} linkedlist_data;

typedef struct
{
    z_stream stream; /* zLib stream structure for inflate */
    int stream_initialised; /* 1 is stream is initialised */
    uInt pos_in_buffered_data; /* last written byte in buffered_data */

    uLong pos_local_header; /* offset of the local header of the file
                           currently writing */
    char* central_header; /* central header data for the current file */
    uLong size_centralheader; /* size of the central header for cur file */
    uLong flag; /* flag of the file currently writing */

    int method; /* compression method of file currently wr.*/
    int raw; /* 1 for directly writing raw data */
    Byte buffered_data[Z_BUFSIZE]; /* buffer contain compressed data to be writ*/
    uLong dosDate;
    uLong crc32;
    int encrypt;
#ifndef NOCRYPT
    unsigned long keys[3]; /* keys defining the pseudo-random sequence */
    const unsigned long* pcrc32_tab;
    int crypt_header_size;
#endif
} curfile_info;

typedef struct
{
    zlib_filefunc_def z_filefunc;
    voidpf filestream; /* io structore of the zipfile */
    linkedlist_data central_dir; /* datablock with central dir in construction*/
    int in_opened_file_inzip; /* 1 if a file in the zip is currently writ.*/
    curfile_info ci; /* info on the file curretly writing */

    uLong begin_pos; /* position of the beginning of the zipfile */
    uLong add_position_when_writting_offset;
    uLong number_entry;
```

```
#ifndef NO_ADDFILEINEXISTINGZIP
    char *globalcomment;
#endif
} zip_internal;

#ifndef NOCRYPT
#define INCLUDECRYPTINGCODE_IFCRYPTALLOWED
#include "crypt.h"
#endif

local linkedlist_datablock_internal* allocate_new_datablock()
{
    linkedlist_datablock_internal* ldi;
    ldi = (linkedlist_datablock_internal*)
        ALLOC(sizeof(linkedlist_datablock_internal));
    if (ldi!=NULL)
    {
        ldi->next_datablock = NULL ;
        ldi->filled_in_this_block = 0 ;
        ldi->avail_in_this_block = SIZEDATA_INDATABLOCK ;
    }
    return ldi;
}

local void free_datablock(ldi)
    linkedlist_datablock_internal* ldi;
{
    while (ldi!=NULL)
    {
        linkedlist_datablock_internal* ldinext = ldi->next_datablock;
        TRYFREE(ldi);
        ldi = ldinext;
    }
}

local void init_linkedlist(ll)
    linkedlist_data* ll;
{
    ll->first_block = ll->last_block = NULL;
}

local void free_linkedlist(ll)
    linkedlist_data* ll;
{
    free_datablock(ll->first_block);
    ll->first_block = ll->last_block = NULL;
}

local int add_data_in_datablock(ll,buf,len)
    linkedlist_data* ll;
    const void* buf;
    uLong len;
{
    linkedlist_datablock_internal* ldi;
    const unsigned char* from_copy;

    if (ll==NULL)
        return ZIP_INTERNALERROR;

    if (ll->last_block == NULL)
    {
        ll->first_block = ll->last_block = allocate_new_datablock();
        if (ll->first_block == NULL)
            return ZIP_INTERNALERROR;
    }

    ldi = ll->last_block;
    from_copy = (unsigned char*)buf;

    while (len>0)
    {
```

```

    uInt copy_this;
    uInt i;
    unsigned char* to_copy;

    if (ldi->avail_in_this_block==0)
    {
        ldi->next_datablock = allocate_new_datablock();
        if (ldi->next_datablock == NULL)
            return ZIP_INTERNALERROR;
        ldi = ldi->next_datablock ;
        ll->last_block = ldi;
    }

    if (ldi->avail_in_this_block < len)
        copy_this = (uInt)ldi->avail_in_this_block;
    else
        copy_this = (uInt)len;

    to_copy = &(ldi->data[ldi->filled_in_this_block]);

    for (i=0;i<copy_this;i++)
        *(to_copy+i)=*(from_copy+i);

    ldi->filled_in_this_block += copy_this;
    ldi->avail_in_this_block -= copy_this;
    from_copy += copy_this ;
    len -= copy_this;
}
return ZIP_OK;
}

/*****
#ifdef NO_ADDFILEINEXISTINGZIP
/* =====
Inputs a long in LSB order to the given file
nbByte == 1, 2 or 4 (byte, short or long)
*/

local int ziplocal_putValue OF((const zlib_filefunc_def* pzlib_filefunc_def,
                                voidpf filestream, uLong x, int nbByte));
local int ziplocal_putValue (pzlib_filefunc_def, filestream, x, nbByte)
    const zlib_filefunc_def* pzlib_filefunc_def;
    voidpf filestream;
    uLong x;
    int nbByte;
{
    unsigned char buf[4];
    int n;
    for (n = 0; n < nbByte; n++)
    {
        buf[n] = (unsigned char)(x & 0xff);
        x >>= 8;
    }
    if (x != 0)
    {
        /* data overflow - hack for ZIP64 (X Roche) */
        for (n = 0; n < nbByte; n++)
        {
            buf[n] = 0xff;
        }
    }

    if (ZWRITE(*pzlib_filefunc_def, filestream, buf, nbByte) != (uLong)nbByte)
        return ZIP_ERRNO;
    else
        return ZIP_OK;
}

local void ziplocal_putValue_inmemory OF((void* dest, uLong x, int nbByte));
local void ziplocal_putValue_inmemory (dest, x, nbByte)
    void* dest;
    uLong x;

```

```

    int nbByte;
{
    unsigned char* buf=(unsigned char*)dest;
    int n;
    for (n = 0; n < nbByte; n++) {
        buf[n] = (unsigned char)(x & 0xff);
        x >>= 8;
    }

    if (x != 0)
    {
        /* data overflow - hack for ZIP64 */
        for (n = 0; n < nbByte; n++)
        {
            buf[n] = 0xff;
        }
    }
}

/*****/

local uLong ziplocal_TmzDateToDosDate(ptm,dosDate)
    const tm_zip* ptm;
    uLong dosDate;
{
    uLong year = (uLong)ptm->tm_year;
    if (year>1980)
        year-=1980;
    else if (year>80)
        year-=80;
    return
        (uLong) (((ptm->tm_mday) + (32 * (ptm->tm_mon+1)) + (512 * year)) << 16) |
        ((ptm->tm_sec/2) + (32* ptm->tm_min) + (2048 * (uLong)ptm->tm_hour));
}

/*****/

local int ziplocal_getByte OF((
    const zlib_filefunc_def* pzlib_filefunc_def,
    voidpf filestream,
    int *pi));

local int ziplocal_getByte(pzlib_filefunc_def,filestream,pi)
    const zlib_filefunc_def* pzlib_filefunc_def;
    voidpf filestream;
    int *pi;
{
    unsigned char c;
    int err = (int)ZREAD(*pzlib_filefunc_def,filestream,&c,1);
    if (err==1)
    {
        *pi = (int)c;
        return ZIP_OK;
    }
    else
    {
        if (ZERROR(*pzlib_filefunc_def,filestream))
            return ZIP_ERRNO;
        else
            return ZIP_EOF;
    }
}

/* =====
   Reads a long in LSB order from the given gz_stream. Sets
*/
local int ziplocal_getShort OF((
    const zlib_filefunc_def* pzlib_filefunc_def,
    voidpf filestream,
    uLong *pX));

local int ziplocal_getShort (pzlib_filefunc_def,filestream,pX)

```

```

    const zlib_filefunc_def* pzlib_filefunc_def;
    voidpf filestream;
    uLong *pX;
{
    uLong x ;
    int i;
    int err;

    err = ziplocal_getByte(pzlib_filefunc_def,filestream,&i);
    x = (uLong)i;

    if (err==ZIP_OK)
        err = ziplocal_getByte(pzlib_filefunc_def,filestream,&i);
    x += ((uLong)i)<<8;

    if (err==ZIP_OK)
        *pX = x;
    else
        *pX = 0;
    return err;
}

local int ziplocal_getLong OF((
    const zlib_filefunc_def* pzlib_filefunc_def,
    voidpf filestream,
    uLong *pX));

local int ziplocal_getLong (pzlib_filefunc_def,filestream,pX)
    const zlib_filefunc_def* pzlib_filefunc_def;
    voidpf filestream;
    uLong *pX;
{
    uLong x ;
    int i;
    int err;

    err = ziplocal_getByte(pzlib_filefunc_def,filestream,&i);
    x = (uLong)i;

    if (err==ZIP_OK)
        err = ziplocal_getByte(pzlib_filefunc_def,filestream,&i);
    x += ((uLong)i)<<8;

    if (err==ZIP_OK)
        err = ziplocal_getByte(pzlib_filefunc_def,filestream,&i);
    x += ((uLong)i)<<16;

    if (err==ZIP_OK)
        err = ziplocal_getByte(pzlib_filefunc_def,filestream,&i);
    x += ((uLong)i)<<24;

    if (err==ZIP_OK)
        *pX = x;
    else
        *pX = 0;
    return err;
}

#ifdef BUFREADCOMMENT
#define BUFREADCOMMENT (0x400)
#endif
/*
    Locate the Central directory of a zipfile (at the end, just before
    the global comment)
*/
local uLong ziplocal_SearchCentralDir OF((
    const zlib_filefunc_def* pzlib_filefunc_def,
    voidpf filestream));

local uLong ziplocal_SearchCentralDir(pzlib_filefunc_def,filestream)
    const zlib_filefunc_def* pzlib_filefunc_def;
    voidpf filestream;
{
    unsigned char* buf;

```

```

uLong uSizeFile;
uLong uBackRead;
uLong uMaxBack=0xffff; /* maximum size of global comment */
uLong uPosFound=0;

if (ZSEEK(*pzlib_filefunc_def, filestream, 0, ZLIB_FILEFUNC_SEEK_END) != 0)
    return 0;

uSizeFile = ZTELL(*pzlib_filefunc_def, filestream);

if (uMaxBack > uSizeFile)
    uMaxBack = uSizeFile;

buf = (unsigned char*)ALLOC(BUFREADCOMMENT+4);
if (buf == NULL)
    return 0;

uBackRead = 4;
while (uBackRead < uMaxBack)
{
    uLong uReadSize, uReadPos;
    int i;
    if (uBackRead + BUFREADCOMMENT > uMaxBack)
        uBackRead = uMaxBack;
    else
        uBackRead += BUFREADCOMMENT;
    uReadPos = uSizeFile - uBackRead;

    uReadSize = ((BUFREADCOMMENT+4) < (uSizeFile - uReadPos)) ?
        (BUFREADCOMMENT+4) : (uSizeFile - uReadPos);
    if (ZSEEK(*pzlib_filefunc_def, filestream, uReadPos, ZLIB_FILEFUNC_SEEK_SET) != 0)
        break;

    if (ZREAD(*pzlib_filefunc_def, filestream, buf, uReadSize) != uReadSize)
        break;

    for (i = (int)uReadSize - 3; i-- > 0;)
        if (((*(buf+i)) == 0x50) && ((*(buf+i+1)) == 0x4b) &&
            ((*(buf+i+2)) == 0x05) && ((*(buf+i+3)) == 0x06))
        {
            uPosFound = uReadPos + i;
            break;
        }

    if (uPosFound != 0)
        break;
}
TRYFREE(buf);
return uPosFound;
}
#endif /* !NO_ADDFILEINEXISTINGZIP */

/*****/
extern zipFile ZEXPORT zipOpen2 (pathname, append, globalcomment, pzlib_filefunc_def)
    const char *pathname;
    int append;
    zipcharpc globalcomment;
    zlib_filefunc_def* pzlib_filefunc_def;
{
    zip_internal ziinit;
    zip_internal* zi;
    int err = ZIP_OK;

    if (pzlib_filefunc_def == NULL)
        fill_fopen_filefunc(&ziinit.z_filefunc);
    else
        ziinit.z_filefunc = *pzlib_filefunc_def;

    ziinit.filestream = (*(ziinit.z_filefunc.zopen_file))
        (ziinit.z_filefunc.opaque,
         pathname,
         (append == APPEND_STATUS_CREATE) ?

```



```

(ZLIB_FILEFUNC_MODE_READ | ZLIB_FILEFUNC_MODE_WRITE | ZLIB_FILEFUNC_MOD
E_CREATE) :
(ZLIB_FILEFUNC_MODE_READ | ZLIB_FILEFUNC_MODE_WRITE | ZLIB_FILEFUNC_M
ODE_EXISTING));

    if (ziinit.filestream == NULL)
        return NULL;
    ziinit.begin_pos = ZTELL(ziinit.z_filefunc,ziinit.filestream);
    ziinit.in_opened_file_inzip = 0;
    ziinit.ci.stream_initialised = 0;
    ziinit.number_entry = 0;
    ziinit.add_position_when_writting_offset = 0;
    init_linkedlist(&(ziinit.central_dir));

    zi = (zip_internal*)ALLOC(sizeof(zip_internal));
    if (zi==NULL)
    {
        ZCLOSE(ziinit.z_filefunc,ziinit.filestream);
        return NULL;
    }

    /* now we add file in a zipfile */
#   ifndef NO_ADDFILEINEXISTINGZIP
    ziinit.globalcomment = NULL;
    if (append == APPEND_STATUS_ADDINZIP)
    {
        uLong byte_before_the_zipfile; /* byte before the zipfile, (>0 for sfx)*/

        uLong size_central_dir;        /* size of the central directory */
        uLong offset_central_dir;      /* offset of start of central directory */
        uLong central_pos,uL;

        uLong number_disk;              /* number of the current dist, used for
                                           spanning ZIP, unsupported, always 0*/
        uLong number_disk_with_CD;      /* number the the disk with central dir, used
                                           for spanning ZIP, unsupported, always 0*/

        uLong number_entry;
        uLong number_entry_CD;          /* total number of entries in
                                           the central dir
                                           (same than number_entry on nospan) */

        uLong size_comment;

        central_pos = ziplocal_SearchCentralDir(&ziinit.z_filefunc,ziinit.filestream);
        if (central_pos==0)
            err=ZIP_ERRNO;

        if (ZSEEK(ziinit.z_filefunc, ziinit.filestream,
                   central_pos,ZLIB_FILEFUNC_SEEK_SET)!=0)
            err=ZIP_ERRNO;

        /* the signature, already checked */
        if (ziplocal_getLong(&ziinit.z_filefunc, ziinit.filestream,&uL)!=ZIP_OK)
            err=ZIP_ERRNO;

        /* number of this disk */
        if (ziplocal_getShort(&ziinit.z_filefunc, ziinit.filestream,&number_disk)!=ZIP_OK
)
            err=ZIP_ERRNO;

        /* number of the disk with the start of the central directory */
        if (ziplocal_getShort(&ziinit.z_filefunc, ziinit.filestream,&number_disk_with_CD)
!=ZIP_OK)
            err=ZIP_ERRNO;

        /* total number of entries in the central dir on this disk */
        if (ziplocal_getShort(&ziinit.z_filefunc, ziinit.filestream,&number_entry)!=ZIP_O
K)
            err=ZIP_ERRNO;

        /* total number of entries in the central dir */
        if (ziplocal_getShort(&ziinit.z_filefunc, ziinit.filestream,&number_entry_CD)!=ZI
P_OK)
            err=ZIP_ERRNO;
    }

```

```

    if ((number_entry_CD!=number_entry) ||
        (number_disk_with_CD!=0) ||
        (number_disk!=0))
        err=ZIP_BADZIPFILE;

    /* size of the central directory */
P_OK) if (ziplocal_getLong(&ziinit.z_filefunc, ziinit.filestream,&size_central_dir)!=ZI
        err=ZIP_ERRNO;

    /* offset of start of central directory with respect to the
       starting disk number */
ZIP_OK) if (ziplocal_getLong(&ziinit.z_filefunc, ziinit.filestream,&offset_central_dir)!=
        err=ZIP_ERRNO;

    /* zipfile global comment length */
K) if (ziplocal_getShort(&ziinit.z_filefunc, ziinit.filestream,&size_comment)!=ZIP_O
        err=ZIP_ERRNO;

    if ((central_pos<offset_central_dir+size_central_dir) &&
        (err==ZIP_OK))
        err=ZIP_BADZIPFILE;

    if (err!=ZIP_OK)
    {
        ZCLOSE(ziinit.z_filefunc, ziinit.filestream);
        return NULL;
    }

    if (size_comment>0)
    {
        ziinit.globalcomment = ALLOC(size_comment+1);
        if (ziinit.globalcomment)
        {
            size_comment = ZREAD(ziinit.z_filefunc, ziinit.filestream,ziinit.globalcom
ment,size_comment);
            ziinit.globalcomment[size_comment]=0;
        }
    }

    byte_before_the_zipfile = central_pos -
        (offset_central_dir+size_central_dir);
    ziinit.add_position_when_writting_offset = byte_before_the_zipfile;

    {
        uLong size_central_dir_to_read = size_central_dir;
        size_t buf_size = SIZEDATA_INDATABLOCK;
        void* buf_read = (void*)ALLOC(buf_size);
        if (ZSEEK(ziinit.z_filefunc, ziinit.filestream,
            offset_central_dir + byte_before_the_zipfile,
            ZLIB_FILEFUNC_SEEK_SET) != 0)
            err=ZIP_ERRNO;

        while ((size_central_dir_to_read>0) && (err==ZIP_OK))
        {
            uLong read_this = SIZEDATA_INDATABLOCK;
            if (read_this > size_central_dir_to_read)
                read_this = size_central_dir_to_read;
            if (ZREAD(ziinit.z_filefunc, ziinit.filestream,buf_read,read_this) != rea
d_this)
                err=ZIP_ERRNO;

            if (err==ZIP_OK)
                err = add_data_in_datablock(&ziinit.central_dir,buf_read,
                    (uLong)read_this);
            size_central_dir_to_read-=read_this;
        }
        TRYFREE(buf_read);
    }
    ziinit.begin_pos = byte_before_the_zipfile;
    ziinit.number_entry = number_entry_CD;

```

```

        if (ZSEEK(ziinit.z_filefunc, ziinit.filestream,
            offset_central_dir+byte_before_the_zipfile,ZLIB_FILEFUNC_SEEK_SET)!=0)
            err=ZIP_ERRNO;
    }

    if (globalcomment)
    {
        *globalcomment = ziinit.globalcomment;
    }
#    endif /* !NO_ADDFILEINEXISTINGZIP*/

    if (err != ZIP_OK)
    {
#        ifndef NO_ADDFILEINEXISTINGZIP
            TRYFREE(ziinit.globalcomment);
#        endif /* !NO_ADDFILEINEXISTINGZIP*/
            TRYFREE(zi);
            return NULL;
        }
    else
    {
        *zi = ziinit;
        return (zipFile)zi;
    }
}

extern zipFile ZEXPORT zipOpen (pathname, append)
    const char *pathname;
    int append;
{
    return zipOpen2(pathname,append,NULL,NULL);
}

extern int ZEXPORT zipOpenNewFileInZip3 (file, filename, zipfi,
                                         extrafield_local, size_extrafield_local,
                                         extrafield_global, size_extrafield_global,
                                         comment, method, level, raw,
                                         windowBits, memLevel, strategy,
                                         password, crcForCrypting)

    zipFile file;
    const char* filename;
    const zip_fileinfo* zipfi;
    const void* extrafield_local;
    uInt size_extrafield_local;
    const void* extrafield_global;
    uInt size_extrafield_global;
    const char* comment;
    int method;
    int level;
    int raw;
    int windowBits;
    int memLevel;
    int strategy;
    const char* password;
    uLong crcForCrypting;
{
    zip_internal* zi;
    uInt size_filename;
    uInt size_comment;
    uInt i;
    int err = ZIP_OK;

#    ifdef NOCRYPT
    if (password != NULL)
        return ZIP_PARAMERROR;
#    endif

    if (file == NULL)
        return ZIP_PARAMERROR;
    if ((method!=0) && (method!=Z_DEFLATED))
        return ZIP_PARAMERROR;

    zi = (zip_internal*)file;

```

```
if (zi->in_opened_file_inzip == 1)
{
    err = zipCloseFileInZip (file);
    if (err != ZIP_OK)
        return err;
}

if (filename==NULL)
    filename="-";

if (comment==NULL)
    size_comment = 0;
else
    size_comment = (uInt)strlen(comment);

size_filename = (uInt)strlen(filename);

if (zipfi == NULL)
    zi->ci.dosDate = 0;
else
{
    if (zipfi->dosDate != 0)
        zi->ci.dosDate = zipfi->dosDate;
    else zi->ci.dosDate = ziplocal_TmzDateToDosDate(&zipfi->tmz_date,zipfi->dosDate);
}

zi->ci.flag = 0;
if ((level==8) || (level==9))
    zi->ci.flag |= 2;
if ((level==2))
    zi->ci.flag |= 4;
if ((level==1))
    zi->ci.flag |= 6;
if (password != NULL)
    zi->ci.flag |= 1;

zi->ci.crc32 = 0;
zi->ci.method = method;
zi->ci.encrypt = 0;
zi->ci.stream_initialised = 0;
zi->ci.pos_in_buffered_data = 0;
zi->ci.raw = raw;
zi->ci.pos_local_header = ZTELL(zi->z_filefunc,zi->filestream) ;
zi->ci.size_centralheader = SIZECENTRALHEADER + size_filename +
                        size_extrafield_global + size_comment;
zi->ci.central_header = (char*)ALLOC((uInt)zi->ci.size_centralheader);

ziplocal_putValue_inmemory(zi->ci.central_header,(uLong)CENTRALHEADERMAGIC,4);
/* version info */
ziplocal_putValue_inmemory(zi->ci.central_header+4,(uLong)VERSIONMADEBY,2);
ziplocal_putValue_inmemory(zi->ci.central_header+6,(uLong)20,2);
ziplocal_putValue_inmemory(zi->ci.central_header+8,(uLong)zi->ci.flag,2);
ziplocal_putValue_inmemory(zi->ci.central_header+10,(uLong)zi->ci.method,2);
ziplocal_putValue_inmemory(zi->ci.central_header+12,(uLong)zi->ci.dosDate,4);
ziplocal_putValue_inmemory(zi->ci.central_header+16,(uLong)0,4); /*crc*/
ziplocal_putValue_inmemory(zi->ci.central_header+20,(uLong)0,4); /*compr size*/
ziplocal_putValue_inmemory(zi->ci.central_header+24,(uLong)0,4); /*uncompr size*/
ziplocal_putValue_inmemory(zi->ci.central_header+28,(uLong)size_filename,2);
ziplocal_putValue_inmemory(zi->ci.central_header+30,(uLong)size_extrafield_global,2);
ziplocal_putValue_inmemory(zi->ci.central_header+32,(uLong)size_comment,2);
ziplocal_putValue_inmemory(zi->ci.central_header+34,(uLong)0,2); /*disk nm start*/

if (zipfi==NULL)
    ziplocal_putValue_inmemory(zi->ci.central_header+36,(uLong)0,2);
else
    ziplocal_putValue_inmemory(zi->ci.central_header+36,(uLong)zipfi->internal_fa,2);

if (zipfi==NULL)
    ziplocal_putValue_inmemory(zi->ci.central_header+38,(uLong)0,4);
else
    ziplocal_putValue_inmemory(zi->ci.central_header+38,(uLong)zipfi->external_fa,4);
```

```

ziplocal_putValue_inmemory(zi->ci.central_header+42,(uLong)zi->ci.pos_local_header- z
i->add_position_when_writting_offset,4);

for (i=0;i<size_filename;i++)
    *(zi->ci.central_header+SIZECENTRALHEADER+i) = *(filename+i);

for (i=0;i<size_extrafield_global;i++)
    *(zi->ci.central_header+SIZECENTRALHEADER+size_filename+i) =
        *(((const char*)extrafield_global)+i);

for (i=0;i<size_comment;i++)
    *(zi->ci.central_header+SIZECENTRALHEADER+size_filename+
        size_extrafield_global+i) = *(comment+i);
if (zi->ci.central_header == NULL)
    return ZIP_INTERNALERROR;

/* write the local header */
err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)LOCALHEADERMAGIC,4);

if (err==ZIP_OK)
    err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)20,2);/* version ne
eded to extract */
if (err==ZIP_OK)
    err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)zi->ci.flag,2);

if (err==ZIP_OK)
    err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)zi->ci.method,2);

if (err==ZIP_OK)
    err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)zi->ci.dosDate,4);

if (err==ZIP_OK)
    err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,4); /* crc 32, un
known */
if (err==ZIP_OK)
    err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,4); /* compressed
size, unknown */
if (err==ZIP_OK)
    err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,4); /* uncompress
ed size, unknown */

if (err==ZIP_OK)
    err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)size_filename,2);

if (err==ZIP_OK)
    err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)size_extrafield_loc
al,2);

if ((err==ZIP_OK) && (size_filename>0))
    if (ZWRITE(zi->z_filefunc,zi->filestream,filename,size_filename)!=size_filename)
        err = ZIP_ERRNO;

if ((err==ZIP_OK) && (size_extrafield_local>0))
    if (ZWRITE(zi->z_filefunc,zi->filestream,extrafield_local,size_extrafield_local)
        !=size_extrafi
eld_local)
        err = ZIP_ERRNO;

zi->ci.stream.avail_in = (uInt)0;
zi->ci.stream.avail_out = (uInt)Z_BUFSIZE;
zi->ci.stream.next_out = zi->ci.buffered_data;
zi->ci.stream.total_in = 0;
zi->ci.stream.total_out = 0;

if ((err==ZIP_OK) && (zi->ci.method == Z_DEFLATED) && (!zi->ci.raw))
{
    zi->ci.stream.zalloc = (alloc_func)0;
    zi->ci.stream.zfree = (free_func)0;
    zi->ci.stream.opaque = (voidpf)0;

    if (windowBits>0)
        windowBits = -windowBits;

    err = deflateInit2(&zi->ci.stream, level,

```

```

        Z_DEFLATED, windowBits, memLevel, strategy);

    if (err==Z_OK)
        zi->ci.stream_initialised = 1;
}
#   ifndef NOCRYPT
zi->ci.crypt_header_size = 0;
if ((err==Z_OK) && (password != NULL))
{
    unsigned char bufHead[RAND_HEAD_LEN];
    unsigned int sizeHead;
    zi->ci.encrypt = 1;
    zi->ci.pcrc_32_tab = get_crc_table();
    /*init_keys(password,zi->ci.keys,zi->ci.pcrc_32_tab);*/

    sizeHead=cryptthead(password,bufHead,RAND_HEAD_LEN,zi->ci.keys,zi->ci.pcrc_32_tab,
crcForCryption);
    zi->ci.crypt_header_size = sizeHead;

    if (ZWRITE(zi->z_filefunc,zi->filestream,bufHead,sizeHead) != sizeHead)
        err = ZIP_ERRNO;
}
#   endif

if (err==Z_OK)
    zi->in_opened_file_inzip = 1;
return err;
}

extern int ZEXPORT zipOpenNewFileInZip2(file, filename, zipfi,
                                       extrafield_local, size_extrafield_local,
                                       extrafield_global, size_extrafield_global,
                                       comment, method, level, raw)

zipFile file;
const char* filename;
const zip_fileinfo* zipfi;
const void* extrafield_local;
uInt size_extrafield_local;
const void* extrafield_global;
uInt size_extrafield_global;
const char* comment;
int method;
int level;
int raw;
{
    return zipOpenNewFileInZip3 (file, filename, zipfi,
                                extrafield_local, size_extrafield_local,
                                extrafield_global, size_extrafield_global,
                                comment, method, level, raw,
                                -MAX_WBITS, DEF_MEM_LEVEL, Z_DEFAULT_STRATEGY,
                                NULL, 0);
}

extern int ZEXPORT zipOpenNewFileInZip (file, filename, zipfi,
                                       extrafield_local, size_extrafield_local,
                                       extrafield_global, size_extrafield_global,
                                       comment, method, level)

zipFile file;
const char* filename;
const zip_fileinfo* zipfi;
const void* extrafield_local;
uInt size_extrafield_local;
const void* extrafield_global;
uInt size_extrafield_global;
const char* comment;
int method;
int level;
{
    return zipOpenNewFileInZip2 (file, filename, zipfi,
                                extrafield_local, size_extrafield_local,
                                extrafield_global, size_extrafield_global,
                                comment, method, level, 0);
}

```

```

local int zipFlushWriteBuffer zi)
zip_internal* zi;
{
    int err=ZIP_OK;

    if (zi->ci.encrypt != 0)
    {
#ifdef NOCRYPT
        uInt i;
        int t;
        for (i=0;i<zi->ci.pos_in_buffered_data;i++)
            zi->ci.buffered_data[i] = zencode(zi->ci.keys, zi->ci.pcrc_32_tab,
                                                zi->ci.buffered_data[i],t);
#endif
    }
    if (ZWRITE(zi->z_filefunc,zi->filestream,zi->ci.buffered_data,zi->ci.pos_in_buffered_
data)
                                                    !=zi->ci.pos_in_buffe
red_data)
        err = ZIP_ERRNO;
    zi->ci.pos_in_buffered_data = 0;
    return err;
}

extern int ZEXPORT zipWriteInFileInZip (file, buf, len)
zipFile file;
const void* buf;
unsigned len;
{
    zip_internal* zi;
    int err=ZIP_OK;

    if (file == NULL)
        return ZIP_PARAMERROR;
    zi = (zip_internal*)file;

    if (zi->in_opened_file_inzip == 0)
        return ZIP_PARAMERROR;

    zi->ci.stream.next_in = (void*)buf;
    zi->ci.stream.avail_in = len;
    zi->ci.crc32 = crc32(zi->ci.crc32,buf,len);

    while ((err==ZIP_OK) && (zi->ci.stream.avail_in>0))
    {
        if (zi->ci.stream.avail_out == 0)
        {
            if (zipFlushWriteBuffer(zi) == ZIP_ERRNO)
                err = ZIP_ERRNO;
            zi->ci.stream.avail_out = (uInt)Z_BUFSIZE;
            zi->ci.stream.next_out = zi->ci.buffered_data;
        }

        if(err != ZIP_OK)
            break;

        if ((zi->ci.method == Z_DEFLATED) && (!zi->ci.raw))
        {
            uLong uTotalOutBefore = zi->ci.stream.total_out;
            err=deflate(&zi->ci.stream, Z_NO_FLUSH);
            zi->ci.pos_in_buffered_data += (uInt)(zi->ci.stream.total_out - uTotalOutBefo
re) ;

        }
        else
        {
            uInt copy_this,i;
            if (zi->ci.stream.avail_in < zi->ci.stream.avail_out)
                copy_this = zi->ci.stream.avail_in;
            else
                copy_this = zi->ci.stream.avail_out;
            for (i=0;i<copy_this;i++)
                *(((char*)zi->ci.stream.next_out)+i) =

```

```

        *(((const char*)zi->ci.stream.next_in)+i);
    {
        zi->ci.stream.avail_in -= copy_this;
        zi->ci.stream.avail_out -= copy_this;
        zi->ci.stream.next_in += copy_this;
        zi->ci.stream.next_out += copy_this;
        zi->ci.stream.total_in += copy_this;
        zi->ci.stream.total_out += copy_this;
        zi->ci.pos_in_buffered_data += copy_this;
    }
}

return err;
}

extern int ZEXPORT zipCloseFileInZipRaw (file, uncompressed_size, crc32)
    zipFile file;
    uLong uncompressed_size;
    uLong crc32;
{
    zip_internal* zi;
    uLong compressed_size;
    int err=ZIP_OK;

    if (file == NULL)
        return ZIP_PARAMERROR;
    zi = (zip_internal*)file;

    if (zi->in_opened_file_inzip == 0)
        return ZIP_PARAMERROR;
    zi->ci.stream.avail_in = 0;

    if ((zi->ci.method == Z_DEFLATED) && (!zi->ci.raw))
        while (err==ZIP_OK)
        {
            uLong uTotalOutBefore;
            if (zi->ci.stream.avail_out == 0)
            {
                if (zipFlushWriteBuffer(zi) == ZIP_ERRNO)
                    err = ZIP_ERRNO;
                zi->ci.stream.avail_out = (uInt)Z_BUFSIZE;
                zi->ci.stream.next_out = zi->ci.buffered_data;
            }
            uTotalOutBefore = zi->ci.stream.total_out;
            err=deflate(&zi->ci.stream, Z_FINISH);
            zi->ci.pos_in_buffered_data += (uInt)(zi->ci.stream.total_out - uTotalOutBefore)
;
        }

    if (err==Z_STREAM_END)
        err=ZIP_OK; /* this is normal */

    if ((zi->ci.pos_in_buffered_data>0) && (err==ZIP_OK))
        if (zipFlushWriteBuffer(zi)==ZIP_ERRNO)
            err = ZIP_ERRNO;

    if ((zi->ci.method == Z_DEFLATED) && (!zi->ci.raw))
    {
        err=deflateEnd(&zi->ci.stream);
        zi->ci.stream_initialised = 0;
    }

    if (!zi->ci.raw)
    {
        crc32 = (uLong)zi->ci.crc32;
        uncompressed_size = (uLong)zi->ci.stream.total_in;
    }
    compressed_size = (uLong)zi->ci.stream.total_out;
#   ifndef NOCRYPT
    compressed_size += zi->ci.crypt_header_size;
#   endif

    ziplocal_putValue_inmemory(zi->ci.central_header+16,crc32,4); /*crc*/

```



```

ziplocal_putValue_inmemory(zi->ci.central_header+20,
                           compressed_size,4); /*compr size*/
if (zi->ci.stream.data_type == Z_ASCII)
    ziplocal_putValue_inmemory(zi->ci.central_header+36,(uLong)Z_ASCII,2);
ziplocal_putValue_inmemory(zi->ci.central_header+24,
                           uncompressed_size,4); /*uncompr size*/

if (err==ZIP_OK)
    err = add_data_in_datablock(&zi->central_dir,zi->ci.central_header,
                               (uLong)zi->ci.size_centralheader);
free(zi->ci.central_header);

if (err==ZIP_OK)
{
    long cur_pos_inzip = ZTELL(zi->z_filefunc,zi->filestream);
    if (ZSEEK(zi->z_filefunc,zi->filestream,
              zi->ci.pos_local_header + 14,ZLIB_FILEFUNC_SEEK_SET)!=0)
        err = ZIP_ERRNO;

    if (err==ZIP_OK)
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,crc32,4); /* crc 32, u
known */

    if (err==ZIP_OK) /* compressed size, unknown */
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,compressed_size,4);

    if (err==ZIP_OK) /* uncompressed size, unknown */
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,uncompressed_size,4);

    if (ZSEEK(zi->z_filefunc,zi->filestream,
              cur_pos_inzip,ZLIB_FILEFUNC_SEEK_SET)!=0)
        err = ZIP_ERRNO;
}

zi->number_entry ++;
zi->in_opened_file_inzip = 0;

return err;
}

extern int ZEXPORT zipCloseFileInZip (file)
zipFile file;
{
    return zipCloseFileInZipRaw (file,0,0);
}

extern int ZEXPORT zipClose (file, global_comment)
zipFile file;
const char* global_comment;
{
    zip_internal* zi;
    int err = 0;
    uLong size_centraldir = 0;
    uLong centraldir_pos_inzip;
    uInt size_global_comment;
    if (file == NULL)
        return ZIP_PARAMERROR;
    zi = (zip_internal*)file;

    if (zi->in_opened_file_inzip == 1)
    {
        err = zipCloseFileInZip (file);
    }

#ifdef NO_ADDFILEINEXISTINGZIP
    if (global_comment==NULL)
        global_comment = zi->globalcomment;
#endif
    if (global_comment==NULL)
        size_global_comment = 0;
    else
        size_global_comment = (uInt)strlen(global_comment);

    centraldir_pos_inzip = ZTELL(zi->z_filefunc,zi->filestream);

```

```

    if (err==ZIP_OK)
    {
        linkedlist_datablock_internal* ldi = zi->central_dir.first_block ;
        while (ldi!=NULL)
        {
            if ((err==ZIP_OK) && (ldi->filled_in_this_block>0))
                if (ZWRITE(zi->z_filefunc,zi->filestream,
                    ldi->data,ldi->filled_in_this_block)
                    !=ldi->filled_in_this_block )
                    err = ZIP_ERRNO;

            size_centraldir += ldi->filled_in_this_block;
            ldi = ldi->next_datablock;
        }
    }
    free_datablock(zi->central_dir.first_block);

    if (err==ZIP_OK) /* Magic End */
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)ENDHEADERMAGIC,4);

    if (err==ZIP_OK) /* number of this disk */
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,2);

    if (err==ZIP_OK) /* number of the disk with the start of the central directory */
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,2);

    if (err==ZIP_OK) /* total number of entries in the central dir on this disk */
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)zi->number_entry,2)
;

    if (err==ZIP_OK) /* total number of entries in the central dir */
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)zi->number_entry,2)
;

    if (err==ZIP_OK) /* size of the central directory */
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)size_centraldir,4);

    if (err==ZIP_OK) /* offset of start of central directory with respect to the
        starting disk number */
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,
            (uLong)(centraldir_pos_inzip - zi->add_position_when_writing_offset),4);

    if (err==ZIP_OK) /* zipfile comment length */
        err = ziplocal_putValue(&zi->z_filefunc,zi->filestream,(uLong)size_global_comment,2);

    if ((err==ZIP_OK) && (size_global_comment>0))
        if (ZWRITE(zi->z_filefunc,zi->filestream,
            global_comment,size_global_comment) != size_global_comment)
            err = ZIP_ERRNO;

    if (ZCLOSE(zi->z_filefunc,zi->filestream) != 0)
        if (err == ZIP_OK)
            err = ZIP_ERRNO;

#ifdef NO_ADDFILEINEXISTINGZIP
    TRYFREE(zi->globalcomment);
#endif
    TRYFREE(zi);

    return err;
}

```

```
/* zip.h -- IO for compress .zip files using zlib
Version 1.01e, February 12th, 2005
```

```
Copyright (C) 1998-2005 Gilles Vollant
```

```
This unzip package allow creates .ZIP file, compatible with PKZip 2.04g
WinZip, InfoZip tools and compatible.
```

```
Multi volume ZipFile (span) are not supported.
```

```
Encryption compatible with pkzip 2.04g only supported
```

```
Old compressions used by old PKZip 1.x are not supported
```

```
For uncompress .zip file, look at unzip.h
```

```
I WAIT FEEDBACK at mail info@winimage.com
```

```
Visit also http://www.winimage.com/zLibDll/unzip.html for evolution
```

```
Condition of use and distribution are the same than zlib :
```

```
This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.
```

```
Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:
```

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

```
*/
```

```
/* for more info about .ZIP format, see
http://www.info-zip.org/pub/infozip/doc/appnote-981119-iz.zip
http://www.info-zip.org/pub/infozip/doc/
PkWare has also a specification at :
ftp://ftp.pkware.com/probdesc.zip
*/
```

```
#ifndef _zip_H
```

```
#define _zip_H
```

```
#ifdef __cplusplus
```

```
extern "C" {
```

```
#endif
```

```
#ifndef _ZLIB_H
```

```
#include "zlib.h"
```

```
#endif
```

```
#ifndef _ZLIBIOAPI_H
```

```
#include "ioapi.h"
```

```
#endif
```

```
#if defined(STRICTZIP) || defined(STRICTZIPUNZIP)
```

```
/* like the STRICT of WIN32, we define a pointer that cannot be converted
from (void*) without cast */
```

```
typedef struct TagzipFile__ { int unused; } zipFile__;
```

```
typedef zipFile__ *zipFile;
```

```
#else
```

```
typedef voidp zipFile;
```

```
#endif
```

```
#define ZIP_OK (0)
```

```
#define ZIP_EOF (0)
```

```
#define ZIP_ERRNO (Z_ERRNO)
```

```
#define ZIP_PARAMERROR (-102)
```

```
#define ZIP_BADZIPFILE (-103)
```

```

#define ZIP_INTERNALERROR                (-104)

#ifndef DEF_MEM_LEVEL
# if MAX_MEM_LEVEL >= 8
#   define DEF_MEM_LEVEL 8
# else
#   define DEF_MEM_LEVEL  MAX_MEM_LEVEL
# endif
#endif
/* default memLevel */

/* tm_zip contain date/time info */
typedef struct tm_zip_s
{
    uInt tm_sec;           /* seconds after the minute - [0,59] */
    uInt tm_min;           /* minutes after the hour - [0,59] */
    uInt tm_hour;          /* hours since midnight - [0,23] */
    uInt tm_mday;          /* day of the month - [1,31] */
    uInt tm_mon;           /* months since January - [0,11] */
    uInt tm_year;          /* years - [1980..2044] */
} tm_zip;

typedef struct
{
    tm_zip      tmz_date;      /* date in understandable format           */
    uLong       dosDate;       /* if dos_date == 0, tm_date is used        */
    /* uLong     flag;          /* general purpose bit flag                2 bytes */

    uLong       internal_fa;   /* internal file attributes                 2 bytes */
    uLong       external_fa;   /* external file attributes                 4 bytes */
} zip_fileinfo;

typedef const char* zipcharpc;

#define APPEND_STATUS_CREATE        (0)
#define APPEND_STATUS_CREATEAFTER  (1)
#define APPEND_STATUS_ADDINZIP     (2)

extern zipFile ZEXPORT zipOpen OF((const char *pathname, int append));
/*
    Create a zipfile.
    pathname contain on Windows XP a filename like "c:\\zlib\\zlib113.zip" or on
    an Unix computer "zlib/zlib113.zip".
    if the file pathname exist and append==APPEND_STATUS_CREATEAFTER, the zip
    will be created at the end of the file.
    (useful if the file contain a self extractor code)
    if the file pathname exist and append==APPEND_STATUS_ADDINZIP, we will
    add files in existing zip (be sure you don't add file that doesn't exist)
    If the zipfile cannot be opened, the return value is NULL.
    Else, the return value is a zipFile Handle, usable with other function
    of this zip package.
*/

/* Note : there is no delete function into a zipfile.
    If you want delete file into a zipfile, you must open a zipfile, and create another
    Of course, you can use RAW reading and writing to copy the file you did not delete
*/

extern zipFile ZEXPORT zipOpen2 OF((const char *pathname,
                                   int append,
                                   zipcharpc* globalcomment,
                                   zlib_filefunc_def* pzlib_filefunc_def));

extern int ZEXPORT zipOpenNewFileInZip OF((zipFile file,
                                           const char* filename,
                                           const zip_fileinfo* zipfi,
                                           const void* extrafield_local,
                                           uInt size_extrafield_local,
                                           const void* extrafield_global,
                                           uInt size_extrafield_global,
                                           const char* comment,
                                           int method,
                                           int level));

```

```
/*
  Open a file in the ZIP for writing.
  filename : the filename in zip (if NULL, '-' without quote will be used
  *zipfi contain supplemental information
  if extrafield_local!=NULL and size_extrafield_local>0, extrafield_local
    contains the extrafield data the the local header
  if extrafield_global!=NULL and size_extrafield_global>0, extrafield_global
    contains the extrafield data the the local header
  if comment != NULL, comment contain the comment string
  method contain the compression method (0 for store, Z_DEFLATED for deflate)
  level contain the level of compression (can be Z_DEFAULT_COMPRESSION)
*/

extern int ZEXPORT zipOpenNewFileInZip2 OF((zipFile file,
                                           const char* filename,
                                           const zip_fileinfo* zipfi,
                                           const void* extrafield_local,
                                           uInt size_extrafield_local,
                                           const void* extrafield_global,
                                           uInt size_extrafield_global,
                                           const char* comment,
                                           int method,
                                           int level,
                                           int raw));

/*
  Same than zipOpenNewFileInZip, except if raw=1, we write raw file
*/

extern int ZEXPORT zipOpenNewFileInZip3 OF((zipFile file,
                                           const char* filename,
                                           const zip_fileinfo* zipfi,
                                           const void* extrafield_local,
                                           uInt size_extrafield_local,
                                           const void* extrafield_global,
                                           uInt size_extrafield_global,
                                           const char* comment,
                                           int method,
                                           int level,
                                           int raw,
                                           int windowBits,
                                           int memLevel,
                                           int strategy,
                                           const char* password,
                                           uLong crcForCtypting));

/*
  Same than zipOpenNewFileInZip2, except
  windowBits,memLevel,,strategy : see parameter strategy in deflateInit2
  password : crypting password (NULL for no crypting)
  crcForCtypting : crc of file to compress (needed for crypting)
*/

extern int ZEXPORT zipWriteInFileInZip OF((zipFile file,
                                           const void* buf,
                                           unsigned len));

/*
  Write data in the zipfile
*/

extern int ZEXPORT zipCloseFileInZip OF((zipFile file));

/*
  Close the current file in the zipfile
*/

extern int ZEXPORT zipCloseFileInZipRaw OF((zipFile file,
                                           uLong uncompressed_size,
                                           uLong crc32));

/*
  Close the current file in the zipfile, for fiel opened with
  parameter raw=1 in zipOpenNewFileInZip2
  uncompressed_size and crc32 are value for the uncompressed size
*/
```

```
*/  
  
extern int ZEXPORT zipClose OF((zipFile file,  
                                const char* global_comment));  
/*  
    Close the zipfile  
*/  
  
#ifdef __cplusplus  
}  
#endif  
  
#endif /* _zip_H */
```

```
(* example.c -- usage example of the zlib compression library
* Copyright (C) 1995-2003 Jean-loup Gailly.
* For conditions of distribution and use, see copyright notice in zlib.h
*
* Pascal translation
* Copyright (C) 1998 by Jacques Nomssi Nzali.
* For conditions of distribution and use, see copyright notice in readme.txt
*
* Adaptation to the zlibpas interface
* Copyright (C) 2003 by Cosmin Truta.
* For conditions of distribution and use, see copyright notice in readme.txt
*)
```

```
program example;
```

```
{ $DEFINE TEST_COMPRESS }
{ DO NOT $DEFINE TEST_GZIO }
{ $DEFINE TEST_DEFLATE }
{ $DEFINE TEST_INFLATE }
{ $DEFINE TEST_FLUSH }
{ $DEFINE TEST_SYNC }
{ $DEFINE TEST_DICT }
```

```
uses SysUtils, zlibpas;
```

```
const TESTFILE = 'foo.gz';
```

```
(* "hello world" would be more standard, but the repeated "hello"
* stresses the compression code better, sorry...
*)
```

```
const hello: PChar = 'hello,hello!';
```

```
const dictionary: PChar = 'hello';
```

```
var dictId: LongInt; (* Adler32 value of the dictionary *)
```

```
procedure CHECK_ERR(err: Integer; msg: String);
begin
  if err <> Z_OK then
  begin
    WriteLn(msg, ' error: ', err);
    Halt(1);
  end;
end;
```

```
procedure EXIT_ERR(const msg: String);
begin
  WriteLn('Error: ', msg);
  Halt(1);
end;
```

```
(* =====
* Test compress and uncompress
*)
```

```
{ $IFDEF TEST_COMPRESS }
procedure test_compress(compr: Pointer; comprLen: LongInt;
                       uncompr: Pointer; uncomprLen: LongInt);
```

```
var err: Integer;
    len: LongInt;
```

```
begin
  len := StrLen(hello)+1;

  err := compress(compr, comprLen, hello, len);
  CHECK_ERR(err, 'compress');

  StrCopy(PChar(uncompr), 'garbage');
```

```
  err := uncompress(uncompr, uncomprLen, compr, comprLen);
  CHECK_ERR(err, 'uncompress');
```

```
  if StrComp(PChar(uncompr), hello) <> 0 then
    EXIT_ERR('bad uncompress')
```

```
  else
    WriteLn('uncompress(): ', PChar(uncompr));
```

```
end;
{$ENDIF}

(* =====
 * Test read/write of .gz files
 *)
{$IFDEF TEST_GZIO}
procedure test_gzio(const fname: PChar; (* compressed file name *)
                    uncompr: Pointer;
                    uncomprLen: LongInt);
var err: Integer;
    len: Integer;
    zfile: gzFile;
    pos: LongInt;
begin
    len := StrLen(hello)+1;

    zfile := gzopen(fname, 'wb');
    if zfile = NIL then
    begin
        WriteLn('gzopen error');
        Halt(1);
    end;
    gzputc(zfile, 'h');
    if gzputs(zfile, 'ello') <> 4 then
    begin
        WriteLn('gzputs err: ', gzerror(zfile, err));
        Halt(1);
    end;
    {$IFDEF GZ_FORMAT_STRING}
    if gzprintf(zfile, '%s!', 'hello') <> 8 then
    begin
        WriteLn('gzprintf err: ', gzerror(zfile, err));
        Halt(1);
    end;
    {$ELSE}
    if gzputs(zfile, ',hello!') <> 8 then
    begin
        WriteLn('gzputs err: ', gzerror(zfile, err));
        Halt(1);
    end;
    {$ENDIF}
    gzseek(zfile, 1, SEEK_CUR); (* add one zero byte *)
    gzclose(zfile);

    zfile := gzopen(fname, 'rb');
    if zfile = NIL then
    begin
        WriteLn('gzopen error');
        Halt(1);
    end;

    StrCopy(PChar(uncompr), 'garbage');

    if gzread(zfile, uncompr, uncomprLen) <> len then
    begin
        WriteLn('gzread err: ', gzerror(zfile, err));
        Halt(1);
    end;
    if StrComp(PChar(uncompr), hello) <> 0 then
    begin
        WriteLn('bad gzread: ', PChar(uncompr));
        Halt(1);
    end
    else
        WriteLn('gzread(): ', PChar(uncompr));

    pos := gzseek(zfile, -8, SEEK_CUR);
    if (pos <> 6) or (gtell(zfile) <> pos) then
    begin
        WriteLn('gzseek error, pos=', pos, ', gtell=', gtell(zfile));
        Halt(1);
    end;
end;
```



```
if gzgetc(zfile) <> ' ' then
begin
  WriteLn('gzgetc error');
  Halt(1);
end;

if gzungetc(' ', zfile) <> ' ' then
begin
  WriteLn('gzungetc error');
  Halt(1);
end;

gzgets(zfile, PChar(uncompr), uncomprLen);
uncomprLen := StrLen(PChar(uncompr));
if uncomprLen <> 7 then (* "hello!" *)
begin
  WriteLn('gzgets err after gzseek: ', gzerror(zfile, err));
  Halt(1);
end;
if StrComp(PChar(uncompr), hello + 6) <> 0 then
begin
  WriteLn('bad gzgets after gzseek');
  Halt(1);
end
else
  WriteLn('gzgets() after gzseek: ', PChar(uncompr));

  gzclose(zfile);
end;
{$ENDIF}

(* =====
 * Test deflate with small buffers
 *)
{$IFDEF TEST_DEFLATE}
procedure test_deflate(compr: Pointer; comprLen: LongInt);
var c_stream: z_stream; (* compression stream *)
    err: Integer;
    len: LongInt;
begin
  len := StrLen(hello)+1;

  c_stream.zalloc := NIL;
  c_stream.zfree := NIL;
  c_stream.opaque := NIL;

  err := deflateInit(c_stream, Z_DEFAULT_COMPRESSION);
  CHECK_ERR(err, 'deflateInit');

  c_stream.next_in := hello;
  c_stream.next_out := compr;

  while (c_stream.total_in <> len) and
        (c_stream.total_out < comprLen) do
  begin
    c_stream.avail_out := 1; { force small buffers }
    c_stream.avail_in := 1;
    err := deflate(c_stream, Z_NO_FLUSH);
    CHECK_ERR(err, 'deflate');
  end;

  (* Finish the stream, still forcing small buffers: *)
  while TRUE do
  begin
    c_stream.avail_out := 1;
    err := deflate(c_stream, Z_FINISH);
    if err = Z_STREAM_END then
      break;
    CHECK_ERR(err, 'deflate');
  end;

  err := deflateEnd(c_stream);
  CHECK_ERR(err, 'deflateEnd');
end;
```

```
{ $ENDIF }

(* =====
 * Test inflate with small buffers
 *)
{$IFDEF TEST_INFLATE}
procedure test_inflate(compr: Pointer; comprLen : LongInt;
                     uncompr: Pointer; uncomprLen : LongInt);
var err: Integer;
    d_stream: z_stream; (* decompression stream *)
begin
    StrCopy(PChar(uncompr), 'garbage');

    d_stream.zalloc := NIL;
    d_stream.zfree  := NIL;
    d_stream.opaque  := NIL;

    d_stream.next_in := compr;
    d_stream.avail_in := 0;
    d_stream.next_out := uncompr;

    err := inflateInit(d_stream);
    CHECK_ERR(err, 'inflateInit');

    while (d_stream.total_out < uncomprLen) and
        (d_stream.total_in < comprLen) do
    begin
        d_stream.avail_out := 1; (* force small buffers *)
        d_stream.avail_in := 1;
        err := inflate(d_stream, Z_NO_FLUSH);
        if err = Z_STREAM_END then
            break;
        CHECK_ERR(err, 'inflate');
    end;

    err := inflateEnd(d_stream);
    CHECK_ERR(err, 'inflateEnd');

    if StrComp(PChar(uncompr), hello) <> 0 then
        EXIT_ERR('bad inflate')
    else
        WriteLn('inflate(): ', PChar(uncompr));
    end;
{$ENDIF}

(* =====
 * Test deflate with large buffers and dynamic change of compression level
 *)
{$IFDEF TEST_DEFLATE}
procedure test_large_deflate(compr: Pointer; comprLen: LongInt;
                           uncompr: Pointer; uncomprLen: LongInt);
var c_stream: z_stream; (* compression stream *)
    err: Integer;
begin
    c_stream.zalloc := NIL;
    c_stream.zfree  := NIL;
    c_stream.opaque  := NIL;

    err := deflateInit(c_stream, Z_BEST_SPEED);
    CHECK_ERR(err, 'deflateInit');

    c_stream.next_out := compr;
    c_stream.avail_out := Integer(comprLen);

    (* At this point, uncompr is still mostly zeroes, so it should compress
     * very well:
     *)
    c_stream.next_in := uncompr;
    c_stream.avail_in := Integer(uncomprLen);
    err := deflate(c_stream, Z_NO_FLUSH);
    CHECK_ERR(err, 'deflate');
    if c_stream.avail_in <> 0 then
        EXIT_ERR('deflate not greedy');
```

```

(* Feed in already compressed data and switch to no compression: *)
deflateParams(c_stream, Z_NO_COMPRESSION, Z_DEFAULT_STRATEGY);
c_stream.next_in := compr;
c_stream.avail_in := Integer(comprLen div 2);
err := deflate(c_stream, Z_NO_FLUSH);
CHECK_ERR(err, 'deflate');

(* Switch back to compressing mode: *)
deflateParams(c_stream, Z_BEST_COMPRESSION, Z_FILTERED);
c_stream.next_in := uncompr;
c_stream.avail_in := Integer(uncomprLen);
err := deflate(c_stream, Z_NO_FLUSH);
CHECK_ERR(err, 'deflate');

err := deflate(c_stream, Z_FINISH);
if err <> Z_STREAM_END then
    EXIT_ERR('deflate should report Z_STREAM_END');

err := deflateEnd(c_stream);
CHECK_ERR(err, 'deflateEnd');
end;
{$ENDIF}

(* =====
* Test inflate with large buffers
*)
{$IFDEF TEST_INFLATE}
procedure test_large_inflate(compr: Pointer; comprLen: LongInt;
                             uncompr: Pointer; uncomprLen: LongInt);
var err: Integer;
    d_stream: z_stream; (* decompression stream *)
begin
    StrCopy(PChar(uncompr), 'garbage');

    d_stream.zalloc := NIL;
    d_stream.zfree := NIL;
    d_stream.opaque := NIL;

    d_stream.next_in := compr;
    d_stream.avail_in := Integer(comprLen);

    err := inflateInit(d_stream);
    CHECK_ERR(err, 'inflateInit');

    while TRUE do
    begin
        d_stream.next_out := uncompr; (* discard the output *)
        d_stream.avail_out := Integer(uncomprLen);
        err := inflate(d_stream, Z_NO_FLUSH);
        if err = Z_STREAM_END then
            break;
        CHECK_ERR(err, 'large inflate');
    end;

    err := inflateEnd(d_stream);
    CHECK_ERR(err, 'inflateEnd');

    if d_stream.total_out <> 2 * uncomprLen + comprLen div 2 then
    begin
        WriteLn('bad large inflate: ', d_stream.total_out);
        Halt(1);
    end
    else
        WriteLn('large_inflate(): OK');
    end;
{$ENDIF}

(* =====
* Test deflate with full flush
*)
{$IFDEF TEST_FLUSH}
procedure test_flush(compr: Pointer; var comprLen : LongInt);
var c_stream: z_stream; (* compression stream *)
    err: Integer;

```

```

    len: Integer;
begin
    len := StrLen(hello)+1;

    c_stream.zalloc := NIL;
    c_stream.zfree  := NIL;
    c_stream.opaque  := NIL;

    err := deflateInit(c_stream, Z_DEFAULT_COMPRESSION);
    CHECK_ERR(err, 'deflateInit');

    c_stream.next_in := hello;
    c_stream.next_out := compr;
    c_stream.avail_in := 3;
    c_stream.avail_out := Integer(comprLen);
    err := deflate(c_stream, Z_FULL_FLUSH);
    CHECK_ERR(err, 'deflate');

    Inc(PByteArray(compr)^[3]); (* force an error in first compressed block *)
    c_stream.avail_in := len - 3;

    err := deflate(c_stream, Z_FINISH);
    if err <> Z_STREAM_END then
        CHECK_ERR(err, 'deflate');

    err := deflateEnd(c_stream);
    CHECK_ERR(err, 'deflateEnd');

    comprLen := c_stream.total_out;
end;
{$ENDIF}

(* =====
 * Test inflateSync()
 *)
{$IFDEF TEST_SYNC}
procedure test_sync(compr: Pointer; comprLen: LongInt;
                   uncompr: Pointer; uncomprLen : LongInt);
var err: Integer;
    d_stream: z_stream; (* decompression stream *)
begin
    StrCopy(PChar(uncompr), 'garbage');

    d_stream.zalloc := NIL;
    d_stream.zfree  := NIL;
    d_stream.opaque  := NIL;

    d_stream.next_in := compr;
    d_stream.avail_in := 2; (* just read the zlib header *)

    err := inflateInit(d_stream);
    CHECK_ERR(err, 'inflateInit');

    d_stream.next_out := uncompr;
    d_stream.avail_out := Integer(uncomprLen);

    inflate(d_stream, Z_NO_FLUSH);
    CHECK_ERR(err, 'inflate');

    d_stream.avail_in := Integer(comprLen-2); (* read all compressed data *)
    err := inflateSync(d_stream);           (* but skip the damaged part *)
    CHECK_ERR(err, 'inflateSync');

    err := inflate(d_stream, Z_FINISH);
    if err <> Z_DATA_ERROR then
        EXIT_ERR('inflate should report DATA_ERROR');
        (* Because of incorrect adler32 *)

    err := inflateEnd(d_stream);
    CHECK_ERR(err, 'inflateEnd');

    WriteLn('after inflateSync(): hel', PChar(uncompr));
end;
{$ENDIF}

```

```

(* =====
 * Test deflate with preset dictionary
 *)
{$IFDEF TEST_DICT}
procedure test_dict_deflate(compr: Pointer; comprLen: LongInt);
var c_stream: z_stream; (* compression stream *)
    err: Integer;
begin
    c_stream.zalloc := NIL;
    c_stream.zfree := NIL;
    c_stream.opaque := NIL;

    err := deflateInit(c_stream, Z_BEST_COMPRESSION);
    CHECK_ERR(err, 'deflateInit');

    err := deflateSetDictionary(c_stream, dictionary, StrLen(dictionary));
    CHECK_ERR(err, 'deflateSetDictionary');

    dictId := c_stream.adler;
    c_stream.next_out := compr;
    c_stream.avail_out := Integer(comprLen);

    c_stream.next_in := hello;
    c_stream.avail_in := StrLen(hello)+1;

    err := deflate(c_stream, Z_FINISH);
    if err <> Z_STREAM_END then
        EXIT_ERR('deflate should report Z_STREAM_END');

    err := deflateEnd(c_stream);
    CHECK_ERR(err, 'deflateEnd');
end;
{$ENDIF}

(* =====
 * Test inflate with a preset dictionary
 *)
{$IFDEF TEST_DICT}
procedure test_dict_inflate(compr: Pointer; comprLen: LongInt;
                           uncompr: Pointer; uncomprLen: LongInt);
var err: Integer;
    d_stream: z_stream; (* decompression stream *)
begin
    StrCopy(PChar(uncompr), 'garbage');

    d_stream.zalloc := NIL;
    d_stream.zfree := NIL;
    d_stream.opaque := NIL;

    d_stream.next_in := compr;
    d_stream.avail_in := Integer(comprLen);

    err := inflateInit(d_stream);
    CHECK_ERR(err, 'inflateInit');

    d_stream.next_out := uncompr;
    d_stream.avail_out := Integer(uncomprLen);

    while TRUE do
        begin
            err := inflate(d_stream, Z_NO_FLUSH);
            if err = Z_STREAM_END then
                break;
            if err = Z_NEED_DICT then
                begin
                    if d_stream.adler <> dictId then
                        EXIT_ERR('unexpected dictionary');
                    err := inflateSetDictionary(d_stream, dictionary, StrLen(dictionary));
                end;
                CHECK_ERR(err, 'inflate with dict');
            end;

            err := inflateEnd(d_stream);

```

```
CHECK_ERR(err, 'inflateEnd');

if StrComp(PChar(uncompr), hello) <> 0 then
  EXIT_ERR('bad inflate with dict')
else
  WriteLn('inflate with dictionary: ', PChar(uncompr));
end;
{$ENDIF}

var compr, uncompr: Pointer;
    comprLen, uncomprLen: LongInt;

begin
  if zlibVersion^ <> ZLIB_VERSION[1] then
    EXIT_ERR('Incompatible zlib version');

  WriteLn('zlib version: ', zlibVersion);
  WriteLn('zlib compile flags: ', Format('0x%x', [zlibCompileFlags]));

  comprLen := 10000 * SizeOf(Integer); (* don't overflow on MSDOS *)
  uncomprLen := comprLen;
  GetMem(compr, comprLen);
  GetMem(uncompr, uncomprLen);
  if (compr = NIL) or (uncompr = NIL) then
    EXIT_ERR('Out of memory');
  (* compr and uncompr are cleared to avoid reading uninitialized
   * data and to ensure that uncompr compresses well.
   *)
  FillChar(compr^, comprLen, 0);
  FillChar(uncompr^, uncomprLen, 0);

  {$IFDEF TEST_COMPRESS}
  WriteLn('** Testing compress');
  test_compress(compr, comprLen, uncompr, uncomprLen);
  {$ENDIF}

  {$IFDEF TEST_GZIO}
  WriteLn('** Testing gzio');
  if ParamCount >= 1 then
    test_gzio(ParamStr(1), uncompr, uncomprLen)
  else
    test_gzio(TESTFILE, uncompr, uncomprLen);
  {$ENDIF}

  {$IFDEF TEST_DEFLATE}
  WriteLn('** Testing deflate with small buffers');
  test_deflate(compr, comprLen);
  {$ENDIF}
  {$IFDEF TEST_INFLATE}
  WriteLn('** Testing inflate with small buffers');
  test_inflate(compr, comprLen, uncompr, uncomprLen);
  {$ENDIF}

  {$IFDEF TEST_DEFLATE}
  WriteLn('** Testing deflate with large buffers');
  test_large_deflate(compr, comprLen, uncompr, uncomprLen);
  {$ENDIF}
  {$IFDEF TEST_INFLATE}
  WriteLn('** Testing inflate with large buffers');
  test_large_inflate(compr, comprLen, uncompr, uncomprLen);
  {$ENDIF}

  {$IFDEF TEST_FLUSH}
  WriteLn('** Testing deflate with full flush');
  test_flush(compr, comprLen);
  {$ENDIF}
  {$IFDEF TEST_SYNC}
  WriteLn('** Testing inflateSync');
  test_sync(compr, comprLen, uncompr, uncomprLen);
  {$ENDIF}
  comprLen := uncomprLen;

  {$IFDEF TEST_DICT}
  WriteLn('** Testing deflate and inflate with preset dictionary');
```

```
test_dict_deflate(compr, comprLen);
test_dict_inflate(compr, comprLen, uncompr, uncomprLen);
{$ENDIF}

FreeMem(compr, comprLen);
FreeMem(uncompr, uncomprLen);
end.
```

This directory contains a Pascal (Delphi, Kylix) interface to the zlib data compression library.

Directory listing

=====

zlibd32.mak	makefile for Borland C++
example.pas	usage example of zlib
zlibpas.pas	the Pascal interface to zlib
readme.txt	this file

Compatibility notes

=====

- Although the name "zlib" would have been more normal for the zlibpas unit, this name is already taken by Borland's ZLib unit. This is somehow unfortunate, because that unit is not a genuine interface to the full-fledged zlib functionality, but a suite of class wrappers around zlib streams. Other essential features, such as checksums, are missing. It would have been more appropriate for that unit to have a name like "ZStreams", or something similar.
- The C and zlib-supplied types int, uInt, long, uLong, etc. are translated directly into Pascal types of similar sizes (Integer, LongInt, etc.), to avoid namespace pollution. In particular, there is no conversion of unsigned int into a Pascal unsigned integer. The Word type is non-portable and has the same size (16 bits) both in a 16-bit and in a 32-bit environment, unlike Integer. Even if there is a 32-bit Cardinal type, there is no real need for unsigned int in zlib under a 32-bit environment.
- Except for the callbacks, the zlib function interfaces are assuming the calling convention normally used in Pascal (__pascal for DOS and Windows16, __fastcall for Windows32). Since the cdecl keyword is used, the old Turbo Pascal does not work with this interface.
- The gz* function interfaces are not translated, to avoid interfacing problems with the C runtime library. Besides, gzprintf(gzFile file, const char *format, ...) cannot be translated into Pascal.

Legal issues

=====

The zlibpas interface is:

Copyright (C) 1995-2003 Jean-loup Gailly and Mark Adler.
Copyright (C) 1998 by Bob Dellaca.
Copyright (C) 2003 by Cosmin Truta.

The example program is:

Copyright (C) 1995-2003 by Jean-loup Gailly.
Copyright (C) 1998,1999,2000 by Jacques Nomssi Nzali.
Copyright (C) 2003 by Cosmin Truta.

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

```
# Makefile for zlib
# For use with Delphi and C++ Builder under Win32
# Updated for zlib 1.2.x by Cosmin Truta

# ----- Borland C++ -----

# This project uses the Delphi (fastcall/register) calling convention:
LOC = -DZEXPORT=__fastcall -DZEXPORTVA=__cdecl

CC = bcc32
LD = bcc32
AR = tlib
# do not use "-pr" in CFLAGS
CFLAGS = -a -d -k- -O2 $(LOC)
LDFLAGS =

# variables
ZLIB_LIB = zlib.lib

OBJ1 = Adler32.obj compress.obj crc32.obj deflate.obj gzio.obj inffast.obj
OBJ2 = inffast.obj inflate.obj inftrees.obj trees.obj uncompress.obj zutil.obj
OBJP1 = +Adler32.obj+compress.obj+crc32.obj+deflate.obj+gzio.obj+inffast.obj
OBJP2 = +inffast.obj+inflate.obj+inftrees.obj+trees.obj+uncompress.obj+zutil.obj

# targets
all: $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
    $(CC) -c $(CFLAGS) *.c

Adler32.obj: Adler32.c zlib.h zconf.h
compress.obj: compress.c zlib.h zconf.h
crc32.obj: crc32.c zlib.h zconf.h crc32.h
deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h
gzio.obj: gzio.c zutil.h zlib.h zconf.h
inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h inffix.h
inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h inffix.h
inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h
trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h
uncompress.obj: uncompress.c zlib.h zconf.h
zutil.obj: zutil.c zutil.h zlib.h zconf.h
example.obj: example.c zlib.h zconf.h
minigzip.obj: minigzip.c zlib.h zconf.h

# For the sake of the old Borland make,
# the command line is cut to fit in the MS-DOS 128 byte limit:
$(ZLIB_LIB): $(OBJ1) $(OBJ2)
    -del $(ZLIB_LIB)
    $(AR) $(ZLIB_LIB) $(OBJP1)
    $(AR) $(ZLIB_LIB) $(OBJP2)

# testing
test: example.exe minigzip.exe
```

```
example
echo hello world | minigzip | minigzip -d

example.exe: example.obj $(ZLIB_LIB)
$(LD) $(LDFLAGS) example.obj $(ZLIB_LIB)

minigzip.exe: minigzip.obj $(ZLIB_LIB)
$(LD) $(LDFLAGS) minigzip.obj $(ZLIB_LIB)

# cleanup
clean:
    -del *.obj
    -del *.exe
    -del *.lib
    -del *.tds
    -del zlib.bak
    -del foo.gz
```

```

(* zlibpas -- Pascal interface to the zlib data compression library
 *
 * Copyright (C) 2003 Cosmin Truta.
 * Derived from original sources by Bob Dellaca.
 * For conditions of distribution and use, see copyright notice in readme.txt
 *)

unit zlibpas;

interface

const
  ZLIB_VERSION = '1.2.3';

type
  alloc_func = function(opaque: Pointer; items, size: Integer): Pointer;
               cdecl;
  free_func  = procedure(opaque, address: Pointer);
               cdecl;

  in_func    = function(opaque: Pointer; var buf: PByte): Integer;
               cdecl;
  out_func   = function(opaque: Pointer; buf: PByte; size: Integer): Integer;
               cdecl;

  z_streamp = ^z_stream;
  z_stream = packed record
    next_in: PChar;      (* next input byte *)
    avail_in: Integer;   (* number of bytes available at next_in *)
    total_in: LongInt;   (* total nb of input bytes read so far *)

    next_out: PChar;     (* next output byte should be put there *)
    avail_out: Integer;  (* remaining free space at next_out *)
    total_out: LongInt;  (* total nb of bytes output so far *)

    msg: PChar;          (* last error message, NULL if no error *)
    state: Pointer;      (* not visible by applications *)

    zalloc: alloc_func;  (* used to allocate the internal state *)
    zfree: free_func;    (* used to free the internal state *)
    opaque: Pointer;     (* private data object passed to zalloc and zfree *)

    data_type: Integer;  (* best guess about the data type: ascii or binary *)
    Adler: LongInt;      (* Adler32 value of the uncompressed data *)
    reserved: LongInt;   (* reserved for future use *)
  end;

(* constants *)
const
  Z_NO_FLUSH      = 0;
  Z_PARTIAL_FLUSH = 1;
  Z_SYNC_FLUSH    = 2;
  Z_FULL_FLUSH    = 3;
  Z_FINISH        = 4;

  Z_OK            = 0;
  Z_STREAM_END    = 1;
  Z_NEED_DICT     = 2;
  Z_ERRNO         = -1;
  Z_STREAM_ERROR  = -2;
  Z_DATA_ERROR    = -3;
  Z_MEM_ERROR     = -4;
  Z_BUF_ERROR     = -5;
  Z_VERSION_ERROR = -6;

  Z_NO_COMPRESSION      = 0;
  Z_BEST_SPEED          = 1;
  Z_BEST_COMPRESSION    = 9;
  Z_DEFAULT_COMPRESSION = -1;

  Z_FILTERED          = 1;
  Z_HUFFMAN_ONLY       = 2;
  Z_RLE               = 3;
  Z_DEFAULT_STRATEGY  = 0;

```

```

Z_BINARY      = 0;
Z_ASCII       = 1;
Z_UNKNOWN     = 2;

Z_DEFLATED    = 8;

(* basic functions *)
function zlibVersion: PChar;
function deflateInit(var strm: z_stream; level: Integer): Integer;
function deflate(var strm: z_stream; flush: Integer): Integer;
function deflateEnd(var strm: z_stream): Integer;
function inflateInit(var strm: z_stream): Integer;
function inflate(var strm: z_stream; flush: Integer): Integer;
function inflateEnd(var strm: z_stream): Integer;

(* advanced functions *)
function deflateInit2(var strm: z_stream; level, method, windowBits,
                     memLevel, strategy: Integer): Integer;
function deflateSetDictionary(var strm: z_stream; const dictionary: PChar;
                             dictLength: Integer): Integer;
function deflateCopy(var dest, source: z_stream): Integer;
function deflateReset(var strm: z_stream): Integer;
function deflateParams(var strm: z_stream; level, strategy: Integer): Integer;
function deflateBound(var strm: z_stream; sourceLen: LongInt): LongInt;
function deflatePrime(var strm: z_stream; bits, value: Integer): Integer;
function inflateInit2(var strm: z_stream; windowBits: Integer): Integer;
function inflateSetDictionary(var strm: z_stream; const dictionary: PChar;
                             dictLength: Integer): Integer;
function inflateSync(var strm: z_stream): Integer;
function inflateCopy(var dest, source: z_stream): Integer;
function inflateReset(var strm: z_stream): Integer;
function inflateBackInit(var strm: z_stream;
                        windowBits: Integer; window: PChar): Integer;
function inflateBack(var strm: z_stream; in_fn: in_func; in_desc: Pointer;
                    out_fn: out_func; out_desc: Pointer): Integer;
function inflateBackEnd(var strm: z_stream): Integer;
function zlibCompileFlags: LongInt;

(* utility functions *)
function compress(dest: PChar; var destLen: LongInt;
                 const source: PChar; sourceLen: LongInt): Integer;
function compress2(dest: PChar; var destLen: LongInt;
                  const source: PChar; sourceLen: LongInt;
                  level: Integer): Integer;
function compressBound(sourceLen: LongInt): LongInt;
function uncompress(dest: PChar; var destLen: LongInt;
                   const source: PChar; sourceLen: LongInt): Integer;

(* checksum functions *)
function Adler32(adler: LongInt; const buf: PChar; len: Integer): LongInt;
function CRC32(crc: LongInt; const buf: PChar; len: Integer): LongInt;

(* various hacks, don't look :) *)
function deflateInit_(var strm: z_stream; level: Integer;
                     const version: PChar; stream_size: Integer): Integer;
function inflateInit_(var strm: z_stream; const version: PChar;
                     stream_size: Integer): Integer;
function deflateInit2_(var strm: z_stream;
                      level, method, windowBits, memLevel, strategy: Integer;
                      const version: PChar; stream_size: Integer): Integer;
function inflateInit2_(var strm: z_stream; windowBits: Integer;
                      const version: PChar; stream_size: Integer): Integer;
function inflateBackInit_(var strm: z_stream;
                         windowBits: Integer; window: PChar;
                         const version: PChar; stream_size: Integer): Integer;

implementation

{$L adler32.obj}
{$L compress.obj}
{$L crc32.obj}
{$L deflate.obj}

```

```
{ $L infback.obj }
{ $L inffast.obj }
{ $L inflate.obj }
{ $L inftrees.obj }
{ $L trees.obj }
{ $L uncompr.obj }
{ $L zutil.obj }

function Adler32; external;
function compress; external;
function compress2; external;
function compressBound; external;
function crc32; external;
function deflate; external;
function deflateBound; external;
function deflateCopy; external;
function deflateEnd; external;
function deflateInit_; external;
function deflateInit2_; external;
function deflateParams; external;
function deflatePrime; external;
function deflateReset; external;
function deflateSetDictionary; external;
function inflate; external;
function inflateBack; external;
function inflateBackEnd; external;
function inflateBackInit_; external;
function inflateCopy; external;
function inflateEnd; external;
function inflateInit_; external;
function inflateInit2_; external;
function inflateReset; external;
function inflateSetDictionary; external;
function inflateSync; external;
function uncompress; external;
function zlibCompileFlags; external;
function zlibVersion; external;

function deflateInit(var strm: z_stream; level: Integer): Integer;
begin
    Result := deflateInit_(strm, level, ZLIB_VERSION, sizeof(z_stream));
end;

function deflateInit2(var strm: z_stream; level, method, windowBits, memLevel,
                      strategy: Integer): Integer;
begin
    Result := deflateInit2_(strm, level, method, windowBits, memLevel, strategy,
                           ZLIB_VERSION, sizeof(z_stream));
end;

function inflateInit(var strm: z_stream): Integer;
begin
    Result := inflateInit_(strm, ZLIB_VERSION, sizeof(z_stream));
end;

function inflateInit2(var strm: z_stream; windowBits: Integer): Integer;
begin
    Result := inflateInit2_(strm, windowBits, ZLIB_VERSION, sizeof(z_stream));
end;

function inflateBackInit(var strm: z_stream;
                        windowBits: Integer; window: PChar): Integer;
begin
    Result := inflateBackInit_(strm, windowBits, window,
                               ZLIB_VERSION, sizeof(z_stream));
end;

function _malloc(Size: Integer): Pointer; cdecl;
begin
    GetMem(Result, Size);
end;

procedure _free(Block: Pointer); cdecl;
begin
```

```
    FreeMem(Block);  
end;  
  
procedure _memset(P: Pointer; B: Byte; count: Integer); cdecl;  
begin  
    FillChar(P^, count, B);  
end;  
  
procedure _memcpy(dest, source: Pointer; count: Integer); cdecl;  
begin  
    Move(source^, dest^, count);  
end;  
  
end.
```

```
puff: puff.c puff.h  
      cc -DTEST -o puff puff.c
```

```
test: puff  
      puff zeros.raw
```

```
clean:  
      rm -f puff puff.o
```


Puff -- A Simple Inflate
3 Mar 2003
Mark Adler
madler@alumni.caltech.edu

What this is --

puff.c provides the routine puff() to decompress the deflate data format. It does so more slowly than zlib, but the code is about one-fifth the size of the inflate code in zlib, and written to be very easy to read.

Why I wrote this --

puff.c was written to document the deflate format unambiguously, by virtue of being working C code. It is meant to supplement RFC 1951, which formally describes the deflate format. I have received many questions on details of the deflate format, and I hope that reading this code will answer those questions. puff.c is heavily commented with details of the deflate format, especially those little nooks and crannies of the format that might not be obvious from a specification.

puff.c may also be useful in applications where code size or memory usage is a very limited resource, and speed is not as important.

How to use it --

Well, most likely you should just be reading puff.c and using zlib for actual applications, but if you must ...

Include puff.h in your code, which provides this prototype:

```
int puff(unsigned char *dest,          /* pointer to destination pointer */
         unsigned long *destlen,       /* amount of output space */
         unsigned char *source,        /* pointer to source data pointer */
         unsigned long *sourcelen);    /* amount of input available */
```

Then you can call puff() to decompress a deflate stream that is in memory in its entirety at source, to a sufficiently sized block of memory for the decompressed data at dest. puff() is the only external symbol in puff.c. The only C library functions that puff.c needs are setjmp() and longjmp(), which are used to simplify error checking in the code to improve readability. puff.c does no memory allocation, and uses less than 2K bytes off of the stack.

If destlen is not enough space for the uncompressed data, then inflate will return an error without writing more than destlen bytes. Note that this means that in order to decompress the deflate data successfully, you need to know the size of the uncompressed data ahead of time.

If needed, puff() can determine the size of the uncompressed data with no output space. This is done by passing dest equal to (unsigned char *)0. Then the initial value of *destlen is ignored and *destlen is set to the length of the uncompressed data. So if the size of the uncompressed data is not known, then two passes of puff() can be used--first to determine the size, and second to do the actual inflation after allocating the appropriate memory. Not pretty, but it works. (This is one of the reasons you should be using zlib.)

The deflate format is self-terminating. If the deflate stream does not end in *sourcelen bytes, puff() will return an error without reading at or past endsource.

On return, *sourcelen is updated to the amount of input data consumed, and *destlen is updated to the size of the uncompressed data. See the comments in puff.c for the possible return codes for puff().

```

/*
 * puff.c
 * Copyright (C) 2002-2004 Mark Adler
 * For conditions of distribution and use, see copyright notice in puff.h
 * version 1.8, 9 Jan 2004
 *
 * puff.c is a simple inflate written to be an unambiguous way to specify the
 * deflate format. It is not written for speed but rather simplicity. As a
 * side benefit, this code might actually be useful when small code is more
 * important than speed, such as bootstrap applications. For typical deflate
 * data, zlib's inflate() is about four times as fast as puff(). zlib's
 * inflate compiles to around 20K on my machine, whereas puff.c compiles to
 * around 4K on my machine (a PowerPC using GNU cc). If the faster decode()
 * function here is used, then puff() is only twice as slow as zlib's
 * inflate().
 *
 * All dynamically allocated memory comes from the stack. The stack required
 * is less than 2K bytes. This code is compatible with 16-bit int's and
 * assumes that long's are at least 32 bits. puff.c uses the short data type,
 * assumed to be 16 bits, for arrays in order to to conserve memory. The code
 * works whether integers are stored big endian or little endian.
 *
 * In the comments below are "Format notes" that describe the inflate process
 * and document some of the less obvious aspects of the format. This source
 * code is meant to supplement RFC 1951, which formally describes the deflate
 * format:
 *
 *   http://www.zlib.org/rfc-deflate.html
 */

/*
 * Change history:
 *
 * 1.0  10 Feb 2002    - First version
 * 1.1  17 Feb 2002    - Clarifications of some comments and notes
 *                      - Update puff() dest and source pointers on negative
 *                      - errors to facilitate debugging deflators
 *                      - Remove longest from struct huffman -- not needed
 *                      - Simplify offs[] index in construct()
 *                      - Add input size and checking, using longjmp() to
 *                      - maintain easy readability
 *                      - Use short data type for large arrays
 *                      - Use pointers instead of long to specify source and
 *                      - destination sizes to avoid arbitrary 4 GB limits
 * 1.2  17 Mar 2002    - Add faster version of decode(), doubles speed (!),
 *                      - but leave simple version for readability
 *                      - Make sure invalid distances detected if pointers
 *                      - are 16 bits
 *                      - Fix fixed codes table error
 *                      - Provide a scanning mode for determining size of
 *                      - uncompressed data
 * 1.3  20 Mar 2002    - Go back to lengths for puff() parameters [Jean-loup]
 *                      - Add a puff.h file for the interface
 *                      - Add braces in puff() for else do [Jean-loup]
 *                      - Use indexes instead of pointers for readability
 * 1.4  31 Mar 2002    - Simplify construct() code set check
 *                      - Fix some comments
 *                      - Add FIXLCODES #define
 * 1.5   6 Apr 2002    - Minor comment fixes
 * 1.6   7 Aug 2002    - Minor format changes
 * 1.7   3 Mar 2003    - Added test code for distribution
 *                      - Added zlib-like license
 * 1.8   9 Jan 2004    - Added some comments on no distance codes case
 */

#include <setjmp.h>          /* for setjmp(), longjmp(), and jmp_buf */
#include "puff.h"           /* prototype for puff() */

#define local static        /* for local function definitions */
#define NIL ((unsigned char *)0) /* for no output option */

/*
 * Maximums for allocations and loops. It is not useful to change these --
 * they are fixed by the deflate format.
 */

```

```

*/
#define MAXBITS 15                /* maximum bits in a code */
#define MAXLCODES 286             /* maximum number of literal/length codes */
#define MAXDCODES 30             /* maximum number of distance codes */
#define MAXCODES (MAXLCODES+MAXDCODES) /* maximum codes lengths to read */
#define FIXLCODES 288            /* number of fixed literal/length codes */

/* input and output state */
struct state {
    /* output state */
    unsigned char *out;           /* output buffer */
    unsigned long outlen;         /* available space at out */
    unsigned long outcnt;         /* bytes written to out so far */

    /* input state */
    unsigned char *in;           /* input buffer */
    unsigned long inlen;         /* available input at in */
    unsigned long incnt;         /* bytes read so far */
    int bitbuf;                  /* bit buffer */
    int bitcnt;                  /* number of bits in bit buffer */

    /* input limit error return state for bits() and decode() */
    jmp_buf env;
};

/*
 * Return need bits from the input stream. This always leaves less than
 * eight bits in the buffer. bits() works properly for need == 0.
 *
 * Format notes:
 *
 * - Bits are stored in bytes from the least significant bit to the most
 *   significant bit. Therefore bits are dropped from the bottom of the bit
 *   buffer, using shift right, and new bytes are appended to the top of the
 *   bit buffer, using shift left.
 */
local int bits(struct state *s, int need)
{
    long val;                    /* bit accumulator (can use up to 20 bits) */

    /* load at least need bits into val */
    val = s->bitbuf;
    while (s->bitcnt < need) {
        if (s->incnt == s->inlen) longjmp(s->env, 1); /* out of input */
        val |= (long)(s->in[s->incnt++]) << s->bitcnt; /* load eight bits */
        s->bitcnt += 8;
    }

    /* drop need bits and update buffer, always zero to seven bits left */
    s->bitbuf = (int)(val >> need);
    s->bitcnt -= need;

    /* return need bits, zeroing the bits above that */
    return (int)(val & ((1L << need) - 1));
}

/*
 * Process a stored block.
 *
 * Format notes:
 *
 * - After the two-bit stored block type (00), the stored block length and
 *   stored bytes are byte-aligned for fast copying. Therefore any leftover
 *   bits in the byte that has the last bit of the type, as many as seven, are
 *   discarded. The value of the discarded bits are not defined and should not
 *   be checked against any expectation.
 *
 * - The second inverted copy of the stored block length does not have to be
 *   checked, but it's probably a good idea to do so anyway.
 *
 * - A stored block can have zero length. This is sometimes used to byte-align
 *   subsets of the compressed data for random access or partial recovery.
 */
local int stored(struct state *s)

```

```

{
    unsigned len;          /* length of stored block */

    /* discard leftover bits from current byte (assumes s->bitcnt < 8) */
    s->bitbuf = 0;
    s->bitcnt = 0;

    /* get length and check against its one's complement */
    if (s->incnt + 4 > s->inlen) return 2;      /* not enough input */
    len = s->in[s->incnt++];
    len |= s->in[s->incnt++] << 8;
    if (s->in[s->incnt++] != (~len & 0xff) ||
        s->in[s->incnt++] != ((~len >> 8) & 0xff))
        return -2;                          /* didn't match complement! */

    /* copy len bytes from in to out */
    if (s->incnt + len > s->inlen) return 2;    /* not enough input */
    if (s->out != NIL) {
        if (s->outcnt + len > s->outlen)
            return 1;                          /* not enough output space */
        while (len--)
            s->out[s->outcnt++] = s->in[s->incnt++];
    }
    else {
        /* just scanning */
        s->outcnt += len;
        s->incnt += len;
    }

    /* done with a valid stored block */
    return 0;
}

/*
 * Huffman code decoding tables.  count[1..MAXBITS] is the number of symbols of
 * each length, which for a canonical code are stepped through in order.
 * symbol[] are the symbol values in canonical order, where the number of
 * entries is the sum of the counts in count[].  The decoding process can be
 * seen in the function decode() below.
 */
struct huffman {
    short *count;          /* number of symbols of each length */
    short *symbol;         /* canonically ordered symbols */
};

/*
 * Decode a code from the stream s using huffman table h.  Return the symbol or
 * a negative value if there is an error.  If all of the lengths are zero, i.e.
 * an empty code, or if the code is incomplete and an invalid code is received,
 * then -9 is returned after reading MAXBITS bits.
 *
 * Format notes:
 *
 * - The codes as stored in the compressed data are bit-reversed relative to
 *   a simple integer ordering of codes of the same lengths.  Hence below the
 *   bits are pulled from the compressed data one at a time and used to
 *   build the code value reversed from what is in the stream in order to
 *   permit simple integer comparisons for decoding.  A table-based decoding
 *   scheme (as used in zlib) does not need to do this reversal.
 *
 * - The first code for the shortest length is all zeros.  Subsequent codes of
 *   the same length are simply integer increments of the previous code.  When
 *   moving up a length, a zero bit is appended to the code.  For a complete
 *   code, the last code of the longest length will be all ones.
 *
 * - Incomplete codes are handled by this decoder, since they are permitted
 *   in the deflate format.  See the format notes for fixed() and dynamic().
 */
#ifdef SLOW
local int decode(struct state *s, struct huffman *h)
{
    int len;                /* current number of bits in code */
    int code;              /* len bits being decoded */
    int first;             /* first code of length len */
    int count;             /* number of codes of length len */

```

```

    int index;                /* index of first code of length len in symbol table */

    code = first = index = 0;
    for (len = 1; len <= MAXBITS; len++) {
        code |= bits(s, 1);    /* get next bit */
        count = h->count[len];
        if (code < first + count) /* if length len, return symbol */
            return h->symbol[index + (code - first)];
        index += count;        /* else update for next length */
        first += count;
        first <= 1;
        code <= 1;
    }
    return -9;                /* ran out of codes */
}

/*
 * A faster version of decode() for real applications of this code.  It's not
 * as readable, but it makes puff() twice as fast.  And it only makes the code
 * a few percent larger.
 */
#else /* !SLOW */
local int decode(struct state *s, struct huffman *h)
{
    int len;                  /* current number of bits in code */
    int code;                 /* len bits being decoded */
    int first;                /* first code of length len */
    int count;                /* number of codes of length len */
    int index;                /* index of first code of length len in symbol table */
    int bitbuf;               /* bits from stream */
    int left;                 /* bits left in next or left to process */
    short *next;              /* next number of codes */

    bitbuf = s->bitbuf;
    left = s->bitcnt;
    code = first = index = 0;
    len = 1;
    next = h->count + 1;
    while (1) {
        while (left-- > 0) {
            code |= bitbuf & 1;
            bitbuf >>= 1;
            count = *next++;
            if (code < first + count) { /* if length len, return symbol */
                s->bitbuf = bitbuf;
                s->bitcnt = (s->bitcnt - len) & 7;
                return h->symbol[index + (code - first)];
            }
            index += count;        /* else update for next length */
            first += count;
            first <= 1;
            code <= 1;
            len++;
        }
        left = (MAXBITS+1) - len;
        if (left == 0) break;
        if (s->incnt == s->inlen) longjmp(s->env, 1); /* out of input */
        bitbuf = s->in[s->incnt++];
        if (left > 8) left = 8;
    }
    return -9;                /* ran out of codes */
}
#endif /* SLOW */

/*
 * Given the list of code lengths length[0..n-1] representing a canonical
 * Huffman code for n symbols, construct the tables required to decode those
 * codes.  Those tables are the number of codes of each length, and the symbols
 * sorted by length, retaining their original order within each length.  The
 * return value is zero for a complete code set, negative for an over-
 * subscribed code set, and positive for an incomplete code set.  The tables
 * can be used if the return value is zero or positive, but they cannot be used
 * if the return value is negative.  If the return value is zero, it is not
 * possible for decode() using that table to return an error--any stream of

```

```

* enough bits will resolve to a symbol. If the return value is positive, then
* it is possible for decode() using that table to return an error for received
* codes past the end of the incomplete lengths.
*
* Not used by decode(), but used for error checking, h->count[0] is the number
* of the n symbols not in the code. So n - h->count[0] is the number of
* codes. This is useful for checking for incomplete codes that have more than
* one symbol, which is an error in a dynamic block.
*
* Assumption: for all i in 0..n-1, 0 <= length[i] <= MAXBITS
* This is assured by the construction of the length arrays in dynamic() and
* fixed() and is not verified by construct().
*
* Format notes:
*
* - Permitted and expected examples of incomplete codes are one of the fixed
* codes and any code with a single symbol which in deflate is coded as one
* bit instead of zero bits. See the format notes for fixed() and dynamic().
*
* - Within a given code length, the symbols are kept in ascending order for
* the code bits definition.
*/
local int construct(struct huffman *h, short *length, int n)
{
    int symbol;          /* current symbol when stepping through length[] */
    int len;             /* current length when stepping through h->count[] */
    int left;            /* number of possible codes left of current length */
    short offs[MAXBITS+1]; /* offsets in symbol table for each length */

    /* count number of codes of each length */
    for (len = 0; len <= MAXBITS; len++)
        h->count[len] = 0;
    for (symbol = 0; symbol < n; symbol++)
        (h->count[length[symbol]])++; /* assumes lengths are within bounds */
    if (h->count[0] == n) /* no codes! */
        return 0; /* complete, but decode() will fail */

    /* check for an over-subscribed or incomplete set of lengths */
    left = 1; /* one possible code of zero length */
    for (len = 1; len <= MAXBITS; len++) {
        left <= 1; /* one more bit, double codes left */
        left -= h->count[len]; /* deduct count from possible codes */
        if (left < 0) return left; /* over-subscribed--return negative */
    } /* left > 0 means incomplete */

    /* generate offsets into symbol table for each length for sorting */
    offs[1] = 0;
    for (len = 1; len < MAXBITS; len++)
        offs[len + 1] = offs[len] + h->count[len];

    /*
     * put symbols in table sorted by length, by symbol order within each
     * length
     */
    for (symbol = 0; symbol < n; symbol++)
        if (length[symbol] != 0)
            h->symbol[offs[length[symbol]]++] = symbol;

    /* return zero for complete set, positive for incomplete set */
    return left;
}

/*
 * Decode literal/length and distance codes until an end-of-block code.
 */
* Format notes:
*
* - Compressed data that is after the block type if fixed or after the code
* description if dynamic is a combination of literals and length/distance
* pairs terminated by and end-of-block code. Literals are simply Huffman
* coded bytes. A length/distance pair is a coded length followed by a
* coded distance to represent a string that occurs earlier in the
* uncompressed data that occurs again at the current location.

```

```

* - Literals, lengths, and the end-of-block code are combined into a single
*   code of up to 286 symbols. They are 256 literals (0..255), 29 length
*   symbols (257..285), and the end-of-block symbol (256).
*
* - There are 256 possible lengths (3..258), and so 29 symbols are not enough
*   to represent all of those. Lengths 3..10 and 258 are in fact represented
*   by just a length symbol. Lengths 11..257 are represented as a symbol and
*   some number of extra bits that are added as an integer to the base length
*   of the length symbol. The number of extra bits is determined by the base
*   length symbol. These are in the static arrays below, lens[] for the base
*   lengths and lext[] for the corresponding number of extra bits.
*
* - The reason that 258 gets its own symbol is that the longest length is used
*   often in highly redundant files. Note that 258 can also be coded as the
*   base value 227 plus the maximum extra value of 31. While a good deflate
*   should never do this, it is not an error, and should be decoded properly.
*
* - If a length is decoded, including its extra bits if any, then it is
*   followed a distance code. There are up to 30 distance symbols. Again
*   there are many more possible distances (1..32768), so extra bits are added
*   to a base value represented by the symbol. The distances 1..4 get their
*   own symbol, but the rest require extra bits. The base distances and
*   corresponding number of extra bits are below in the static arrays dist[]
*   and dext[].
*
* - Literal bytes are simply written to the output. A length/distance pair is
*   an instruction to copy previously uncompressed bytes to the output. The
*   copy is from distance bytes back in the output stream, copying for length
*   bytes.
*
* - Distances pointing before the beginning of the output data are not
*   permitted.
*
* - Overlapped copies, where the length is greater than the distance, are
*   allowed and common. For example, a distance of one and a length of 258
*   simply copies the last byte 258 times. A distance of four and a length of
*   twelve copies the last four bytes three times. A simple forward copy
*   ignoring whether the length is greater than the distance or not implements
*   this correctly. You should not use memcpy() since its behavior is not
*   defined for overlapped arrays. You should not use memmove() or bcopy()
*   since though their behavior -is- defined for overlapping arrays, it is
*   defined to do the wrong thing in this case.
*/
local int codes(struct state *s,
                struct huffman *lencode,
                struct huffman *distcode)
{
    int symbol;          /* decoded symbol */
    int len;             /* length for copy */
    unsigned dist;       /* distance for copy */
    static const short lens[29] = { /* Size base for length codes 257..285 */
        3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 19, 23, 27, 31,
        35, 43, 51, 59, 67, 83, 99, 115, 131, 163, 195, 227, 258};
    static const short lext[29] = { /* Extra bits for length codes 257..285 */
        0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2,
        3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 0};
    static const short dists[30] = { /* Offset base for distance codes 0..29 */
        1, 2, 3, 4, 5, 7, 9, 13, 17, 25, 33, 49, 65, 97, 129, 193,
        257, 385, 513, 769, 1025, 1537, 2049, 3073, 4097, 6145,
        8193, 12289, 16385, 24577};
    static const short dext[30] = { /* Extra bits for distance codes 0..29 */
        0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6,
        7, 7, 8, 8, 9, 9, 10, 10, 11, 11,
        12, 12, 13, 13};

    /* decode literals and length/distance pairs */
    do {
        symbol = decode(s, lencode);
        if (symbol < 0) return symbol; /* invalid symbol */
        if (symbol < 256) { /* literal: symbol is the byte */
            /* write out the literal */
            if (s->out != NIL) {
                if (s->outcnt == s->outlen) return 1;
                s->out[s->outcnt] = symbol;

```

```

    }
    s->outcnt++;
}
else if (symbol > 256) { /* length */
    /* get and compute length */
    symbol -= 257;
    if (symbol >= 29) return -9; /* invalid fixed code */
    len = lens[symbol] + bits(s, ltext[symbol]);

    /* get and check distance */
    symbol = decode(s, distcode);
    if (symbol < 0) return symbol; /* invalid symbol */
    dist = dists[symbol] + bits(s, dext[symbol]);
    if (dist > s->outcnt)
        return -10; /* distance too far back */

    /* copy length bytes from distance bytes back */
    if (s->out != NIL) {
        if (s->outcnt + len > s->outlen) return 1;
        while (len--) {
            s->out[s->outcnt] = s->out[s->outcnt - dist];
            s->outcnt++;
        }
    }
    else
        s->outcnt += len;
}
} while (symbol != 256); /* end of block symbol */

/* done with a valid fixed or dynamic block */
return 0;
}

/*
 * Process a fixed codes block.
 *
 * Format notes:
 *
 * - This block type can be useful for compressing small amounts of data for
 *   which the size of the code descriptions in a dynamic block exceeds the
 *   benefit of custom codes for that block. For fixed codes, no bits are
 *   spent on code descriptions. Instead the code lengths for literal/length
 *   codes and distance codes are fixed. The specific lengths for each symbol
 *   can be seen in the "for" loops below.
 *
 * - The literal/length code is complete, but has two symbols that are invalid
 *   and should result in an error if received. This cannot be implemented
 *   simply as an incomplete code since those two symbols are in the "middle"
 *   of the code. They are eight bits long and the longest literal/length\
 *   code is nine bits. Therefore the code must be constructed with those
 *   symbols, and the invalid symbols must be detected after decoding.
 *
 * - The fixed distance codes also have two invalid symbols that should result
 *   in an error if received. Since all of the distance codes are the same
 *   length, this can be implemented as an incomplete code. Then the invalid
 *   codes are detected while decoding.
 */
local int fixed(struct state *s)
{
    static int virgin = 1;
    static short lencnt[MAXBITS+1], lensym[FIXLCODES];
    static short distcnt[MAXBITS+1], distsym[MAXDCODES];
    static struct huffman lencode = {lencnt, lensym};
    static struct huffman distcode = {distcnt, distsym};

    /* build fixed huffman tables if first call (may not be thread safe) */
    if (virgin) {
        int symbol;
        short lengths[FIXLCODES];

        /* literal/length table */
        for (symbol = 0; symbol < 144; symbol++)
            lengths[symbol] = 8;
        for (; symbol < 256; symbol++)

```



```
        lengths[symbol] = 9;
    for (; symbol < 280; symbol++)
        lengths[symbol] = 7;
    for (; symbol < FIXLCODES; symbol++)
        lengths[symbol] = 8;
    construct(&lencode, lengths, FIXLCODES);

    /* distance table */
    for (symbol = 0; symbol < MAXDCODES; symbol++)
        lengths[symbol] = 5;
    construct(&distcode, lengths, MAXDCODES);

    /* do this just once */
    virgin = 0;
}

/* decode data until end-of-block code */
return codes(s, &lencode, &distcode);
}

/*
 * Process a dynamic codes block.
 *
 * Format notes:
 *
 * - A dynamic block starts with a description of the literal/length and
 *   distance codes for that block. New dynamic blocks allow the compressor to
 *   rapidly adapt to changing data with new codes optimized for that data.
 *
 * - The codes used by the deflate format are "canonical", which means that
 *   the actual bits of the codes are generated in an unambiguous way simply
 *   from the number of bits in each code. Therefore the code descriptions
 *   are simply a list of code lengths for each symbol.
 *
 * - The code lengths are stored in order for the symbols, so lengths are
 *   provided for each of the literal/length symbols, and for each of the
 *   distance symbols.
 *
 * - If a symbol is not used in the block, this is represented by a zero as
 *   as the code length. This does not mean a zero-length code, but rather
 *   that no code should be created for this symbol. There is no way in the
 *   deflate format to represent a zero-length code.
 *
 * - The maximum number of bits in a code is 15, so the possible lengths for
 *   any code are 1..15.
 *
 * - The fact that a length of zero is not permitted for a code has an
 *   interesting consequence. Normally if only one symbol is used for a given
 *   code, then in fact that code could be represented with zero bits. However
 *   in deflate, that code has to be at least one bit. So for example, if
 *   only a single distance base symbol appears in a block, then it will be
 *   represented by a single code of length one, in particular one 0 bit. This
 *   is an incomplete code, since if a 1 bit is received, it has no meaning,
 *   and should result in an error. So incomplete distance codes of one symbol
 *   should be permitted, and the receipt of invalid codes should be handled.
 *
 * - It is also possible to have a single literal/length code, but that code
 *   must be the end-of-block code, since every dynamic block has one. This
 *   is not the most efficient way to create an empty block (an empty fixed
 *   block is fewer bits), but it is allowed by the format. So incomplete
 *   literal/length codes of one symbol should also be permitted.
 *
 * - If there are only literal codes and no lengths, then there are no distance
 *   codes. This is represented by one distance code with zero bits.
 *
 * - The list of up to 286 length/literal lengths and up to 30 distance lengths
 *   are themselves compressed using Huffman codes and run-length encoding. In
 *   the list of code lengths, a 0 symbol means no code, a 1..15 symbol means
 *   that length, and the symbols 16, 17, and 18 are run-length instructions.
 *   Each of 16, 17, and 18 are followed by extra bits to define the length of
 *   the run. 16 copies the last length 3 to 6 times. 17 represents 3 to 10
 *   zero lengths, and 18 represents 11 to 138 zero lengths. Unused symbols
 *   are common, hence the special coding for zero lengths.
 */
```

```

* - The symbols for 0..18 are Huffman coded, and so that code must be
*   described first. This is simply a sequence of up to 19 three-bit values
*   representing no code (0) or the code length for that symbol (1..7).
*
* - A dynamic block starts with three fixed-size counts from which is computed
*   the number of literal/length code lengths, the number of distance code
*   lengths, and the number of code length code lengths (ok, you come up with
*   a better name!) in the code descriptions. For the literal/length and
*   distance codes, lengths after those provided are considered zero, i.e. no
*   code. The code length code lengths are received in a permuted order (see
*   the order[] array below) to make a short code length code length list more
*   likely. As it turns out, very short and very long codes are less likely
*   to be seen in a dynamic code description, hence what may appear initially
*   to be a peculiar ordering.
*
* - Given the number of literal/length code lengths (nlen) and distance code
*   lengths (ndist), then they are treated as one long list of nlen + ndist
*   code lengths. Therefore run-length coding can and often does cross the
*   boundary between the two sets of lengths.
*
* - So to summarize, the code description at the start of a dynamic block is
*   three counts for the number of code lengths for the literal/length codes,
*   the distance codes, and the code length codes. This is followed by the
*   code length code lengths, three bits each. This is used to construct the
*   code length code which is used to read the remainder of the lengths. Then
*   the literal/length code lengths and distance lengths are read as a single
*   set of lengths using the code length codes. Codes are constructed from
*   the resulting two sets of lengths, and then finally you can start
*   decoding actual compressed data in the block.
*
* - For reference, a "typical" size for the code description in a dynamic
*   block is around 80 bytes.
*/

```

```

local int dynamic(struct state *s)
{
    int nlen, ndist, ncode;           /* number of lengths in descriptor */
    int index;                        /* index of lengths[] */
    int err;                          /* construct() return value */
    short lengths[MAXCODES];          /* descriptor code lengths */
    short lencnt[MAXBITS+1], lensym[MAXLCODES]; /* lencode memory */
    short distcnt[MAXBITS+1], distsym[MAXDCODES]; /* distcode memory */
    struct huffman lencode = {lencnt, lensym}; /* length code */
    struct huffman distcode = {distcnt, distsym}; /* distance code */
    static const short order[19] = /* permutation of code length codes */
        {16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15};

    /* get number of lengths in each table, check lengths */
    nlen = bits(s, 5) + 257;
    ndist = bits(s, 5) + 1;
    ncode = bits(s, 4) + 4;
    if (nlen > MAXLCODES || ndist > MAXDCODES)
        return -3; /* bad counts */

    /* read code length code lengths (really), missing lengths are zero */
    for (index = 0; index < ncode; index++)
        lengths[order[index]] = bits(s, 3);
    for (; index < 19; index++)
        lengths[order[index]] = 0;

    /* build huffman table for code lengths codes (use lencode temporarily) */
    err = construct(&lencode, lengths, 19);
    if (err != 0) return -4; /* require complete code set here */

    /* read length/literal and distance code length tables */
    index = 0;
    while (index < nlen + ndist) {
        int symbol; /* decoded value */
        int len; /* last length to repeat */

        symbol = decode(s, &lencode);
        if (symbol < 16) /* length in 0..15 */
            lengths[index++] = symbol;
        else { /* repeat instruction */
            len = 0; /* assume repeating zeros */

```

```

        if (symbol == 16) { /* repeat last length 3..6 times */
            if (index == 0) return -5; /* no last length! */
            len = lengths[index - 1]; /* last length */
            symbol = 3 + bits(s, 2);
        }
        else if (symbol == 17) /* repeat zero 3..10 times */
            symbol = 3 + bits(s, 3);
        else /* == 18, repeat zero 11..138 times */
            symbol = 11 + bits(s, 7);
        if (index + symbol > nlen + ndist)
            return -6; /* too many lengths! */
        while (symbol--) /* repeat last or zero symbol times */
            lengths[index++] = len;
    }
}

/* build huffman table for literal/length codes */
err = construct(&lencode, lengths, nlen);
if (err < 0 || (err > 0 && nlen - lencode.count[0] != 1))
    return -7; /* only allow incomplete codes if just one code */

/* build huffman table for distance codes */
err = construct(&distcode, lengths + nlen, ndist);
if (err < 0 || (err > 0 && ndist - distcode.count[0] != 1))
    return -8; /* only allow incomplete codes if just one code */

/* decode data until end-of-block code */
return codes(s, &lencode, &distcode);
}

/*
 * Inflate source to dest. On return, destlen and sourcelen are updated to the
 * size of the uncompressed data and the size of the deflate data respectively.
 * On success, the return value of puff() is zero. If there is an error in the
 * source data, i.e. it is not in the deflate format, then a negative value is
 * returned. If there is not enough input available or there is not enough
 * output space, then a positive error is returned. In that case, destlen and
 * sourcelen are not updated to facilitate retrying from the beginning with the
 * provision of more input data or more output space. In the case of invalid
 * inflate data (a negative error), the dest and source pointers are updated to
 * facilitate the debugging of deflators.
 *
 * puff() also has a mode to determine the size of the uncompressed output with
 * no output written. For this dest must be (unsigned char *)0. In this case,
 * the input value of *destlen is ignored, and on return *destlen is set to the
 * size of the uncompressed output.
 *
 * The return codes are:
 *
 * 2: available inflate data did not terminate
 * 1: output space exhausted before completing inflate
 * 0: successful inflate
 * -1: invalid block type (type == 3)
 * -2: stored block length did not match one's complement
 * -3: dynamic block code description: too many length or distance codes
 * -4: dynamic block code description: code lengths codes incomplete
 * -5: dynamic block code description: repeat lengths with no first length
 * -6: dynamic block code description: repeat more than specified lengths
 * -7: dynamic block code description: invalid literal/length code lengths
 * -8: dynamic block code description: invalid distance code lengths
 * -9: invalid literal/length or distance code in fixed or dynamic block
 * -10: distance is too far back in fixed or dynamic block
 *
 * Format notes:
 *
 * - Three bits are read for each block to determine the kind of block and
 * whether or not it is the last block. Then the block is decoded and the
 * process repeated if it was not the last block.
 *
 * - The leftover bits in the last byte of the deflate data after the last
 * block (if it was a fixed or dynamic block) are undefined and have no
 * expected values to check.
 */
int puff(unsigned char *dest, /* pointer to destination pointer */

```

```

        unsigned long *destlen,          /* amount of output space */
        unsigned char *source,          /* pointer to source data pointer */
        unsigned long *sourcelen)       /* amount of input available */
{
    struct state s;                      /* input/output state */
    int last, type;                      /* block information */
    int err;                             /* return value */

    /* initialize output state */
    s.out = dest;
    s.outlen = *destlen;                  /* ignored if dest is NIL */
    s.outcnt = 0;

    /* initialize input state */
    s.in = source;
    s.inlen = *sourcelen;
    s.incnt = 0;
    s.bitbuf = 0;
    s.bitcnt = 0;

    /* return if bits() or decode() tries to read past available input */
    if (setjmp(s.env) != 0)              /* if came back here via longjmp() */
        err = 2;                          /* then skip do-loop, return error */
    else {
        /* process blocks until last block or error */
        do {
            last = bits(&s, 1);            /* one if last block */
            type = bits(&s, 2);            /* block type 0..3 */
            err = type == 0 ? stored(&s) :
                (type == 1 ? fixed(&s) :
                 (type == 2 ? dynamic(&s) :
                  -1));                    /* type == 3, invalid */
            if (err != 0) break;           /* return with error */
        } while (!last);
    }

    /* update the lengths and return */
    if (err <= 0) {
        *destlen = s.outcnt;
        *sourcelen = s.incnt;
    }
    return err;
}

#ifdef TEST
/* Example of how to use puff() */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>

local unsigned char *yank(char *name, unsigned long *len)
{
    unsigned long size;
    unsigned char *buf;
    FILE *in;
    struct stat s;

    *len = 0;
    if (stat(name, &s)) return NULL;
    if ((s.st_mode & S_IFMT) != S_IFREG) return NULL;
    size = (unsigned long)(s.st_size);
    if (size == 0 || (off_t)size != s.st_size) return NULL;
    in = fopen(name, "r");
    if (in == NULL) return NULL;
    buf = malloc(size);
    if (buf != NULL && fread(buf, 1, size, in) != size) {
        free(buf);
        buf = NULL;
    }
    fclose(in);
    *len = size;
    return buf;
}

```

```
int main(int argc, char **argv)
{
    int ret;
    unsigned char *source;
    unsigned long len, sourcelen, destlen;

    if (argc < 2) return 2;
    source = yank(argv[1], &len);
    if (source == NULL) return 2;
    sourcelen = len;
    ret = puff(NIL, &destlen, source, &sourcelen);
    if (ret)
        printf("puff() failed with return code %d\n", ret);
    else {
        printf("puff() succeeded uncompressing %lu bytes\n", destlen);
        if (sourcelen < len) printf("%lu compressed bytes unused\n",
                                   len - sourcelen);
    }
    free(source);
    return ret;
}
#endif
```

```
/* puff.h
```

```
Copyright (C) 2002, 2003 Mark Adler, all rights reserved  
version 1.7, 3 Mar 2002
```

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

- 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.*
- 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.*
- 3. This notice may not be removed or altered from any source distribution.*

Mark Adler madler@alumni.caltech.edu

```
*/
```

```
/*
```

```
 * See puff.c for purpose and usage.
```

```
*/
```

```
int puff(unsigned char *dest,          /* pointer to destination pointer */  
          unsigned long *destlen,      /* amount of output space */  
          unsigned char *source,       /* pointer to source data pointer */  
          unsigned long *sourcelen);   /* amount of input available */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

#include "zlib.h"

void MyDoMinus64(LARGE_INTEGER *R,LARGE_INTEGER A,LARGE_INTEGER B)
{
    R->HighPart = A.HighPart - B.HighPart;
    if (A.LowPart >= B.LowPart)
        R->LowPart = A.LowPart - B.LowPart;
    else
    {
        R->LowPart = A.LowPart - B.LowPart;
        R->HighPart --;
    }
}

#ifdef _M_X64
// see http://msdn2.microsoft.com/library/twchhe95\(en-us,vs.80\).aspx for __rdtsc
unsigned __int64 __rdtsc(void);
void BeginCountRdtsc(LARGE_INTEGER * pbeginTime64)
{
    // printf("rdtsc = %I64x\n",__rdtsc());
    pbeginTime64->QuadPart=__rdtsc();
}

LARGE_INTEGER GetResRdtsc(LARGE_INTEGER beginTime64,BOOL fComputeTimeQueryPerf)
{
    LARGE_INTEGER LIres;
    unsigned __int64 res=__rdtsc()-((unsigned __int64)(beginTime64.QuadPart));
    LIres.QuadPart=res;
    // printf("rdtsc = %I64x\n",__rdtsc());
    return LIres;
}
#else
#ifdef _M_IX86
void myGetRDTSC32(LARGE_INTEGER * pbeginTime64)
{
    DWORD dwEdx,dwEax;
    _asm
    {
        rdtsc
        mov dwEax,eax
        mov dwEdx,edx
    }
    pbeginTime64->LowPart=dwEax;
    pbeginTime64->HighPart=dwEdx;
}

void BeginCountRdtsc(LARGE_INTEGER * pbeginTime64)
{
    myGetRDTSC32(pbeginTime64);
}

LARGE_INTEGER GetResRdtsc(LARGE_INTEGER beginTime64,BOOL fComputeTimeQueryPerf)
{
    LARGE_INTEGER LIres,endTime64;
    myGetRDTSC32(&endTime64);

    LIres.LowPart=LIres.HighPart=0;
    MyDoMinus64(&LIres,endTime64,beginTime64);
    return LIres;
}
#else
void myGetRDTSC32(LARGE_INTEGER * pbeginTime64)
{
}

void BeginCountRdtsc(LARGE_INTEGER * pbeginTime64)
{
}
```

```
LARGE_INTEGER GetResRdtsc(LARGE_INTEGER beginTime64, BOOL fComputeTimeQueryPerf)
{
    LARGE_INTEGER lr;
    lr.QuadPart=0;
    return lr;
}
#endif
#endif

void BeginCountPerfCounter(LARGE_INTEGER * pbeginTime64, BOOL fComputeTimeQueryPerf)
{
    if ((!fComputeTimeQueryPerf) || (!QueryPerformanceCounter(pbeginTime64)))
    {
        pbeginTime64->LowPart = GetTickCount();
        pbeginTime64->HighPart = 0;
    }
}

DWORD GetMsecSincePerfCounter(LARGE_INTEGER beginTime64, BOOL fComputeTimeQueryPerf)
{
    LARGE_INTEGER endTime64, ticksPerSecond, ticks;
    DWORDLONG ticksShifted, tickSecShifted;
    DWORD dwLog=16+0;
    DWORD dwRet;
    if ((!fComputeTimeQueryPerf) || (!QueryPerformanceCounter(&endTime64)))
        dwRet = (GetTickCount() - beginTime64.LowPart)*1;
    else
    {
        MyDoMinus64(&ticks, endTime64, beginTime64);
        QueryPerformanceFrequency(&ticksPerSecond);

        {
            ticksShifted = Int64Shr1Mod32(*(DWORDLONG*)&ticks, dwLog);
            tickSecShifted = Int64Shr1Mod32(*(DWORDLONG*)&ticksPerSecond, dwLog);
        }

        dwRet = (DWORD)((((DWORD)ticksShifted)*1000)/(DWORD)(tickSecShifted));
        dwRet *=1;
    }
    return dwRet;
}

int ReadFileMemory(const char* filename, long* plFileSize, void** pFilePtr)
{
    FILE* stream;
    void* ptr;
    int retVal=1;
    stream=fopen(filename, "rb");
    if (stream==NULL)
        return 0;

    fseek(stream, 0, SEEK_END);

    *plFileSize=ftell(stream);
    fseek(stream, 0, SEEK_SET);
    ptr=malloc((*plFileSize)+1);
    if (ptr==NULL)
        retVal=0;
    else
    {
        if (fread(ptr, 1, *plFileSize, stream) != (*plFileSize))
            retVal=0;
    }
    fclose(stream);
    *pFilePtr=ptr;
    return retVal;
}

int main(int argc, char *argv[])
{
    int BlockSizeCompress=0x8000;
    int BlockSizeUncompress=0x8000;
```



```
int cprLevel=Z_DEFAULT_COMPRESSION ;
long lFileSize;
unsigned char* FilePtr;
long lBufferSizeCpr;
long lBufferSizeUncpr;
long lCompressedSize=0;
unsigned char* CprPtr;
unsigned char* UncprPtr;
long lSizeCpr,lSizeUncpr;
DWORD dwGetTick,dwMsecQP;
LARGE_INTEGER li_qp,li_rdtsc,dwResRdtsc;

if (argc<=1)
{
    printf("run TestZlib <File> [BlockSizeCompress] [BlockSizeUncompress] [compress. level]\n");
    return 0;
}

if (ReadFileMemory(argv[1],&lFileSize,&FilePtr)==0)
{
    printf("error reading %s\n",argv[1]);
    return 1;
}
else printf("file %s read, %u bytes\n",argv[1],lFileSize);

if (argc>=3)
    BlockSizeCompress=atol(argv[2]);

if (argc>=4)
    BlockSizeUncompress=atol(argv[3]);

if (argc>=5)
    cprLevel=(int)atol(argv[4]);

lBufferSizeCpr = lFileSize + (lFileSize/0x10) + 0x200;
lBufferSizeUncpr = lBufferSizeCpr;

CprPtr=(unsigned char*)malloc(lBufferSizeCpr + BlockSizeCompress);

BeginCountPerfCounter(&li_qp,TRUE);
dwGetTick=GetTickCount();
BeginCountRdtsc(&li_rdtsc);
{
    z_stream zcpr;
    int ret=Z_OK;
    long lOrigToDo = lFileSize;
    long lOrigDone = 0;
    int step=0;
    memset(&zcpr,0,sizeof(z_stream));
    deflateInit(&zcpr,cprLevel);

    zcpr.next_in = FilePtr;
    zcpr.next_out = CprPtr;

    do
    {
        long all_read_before = zcpr.total_in;
        zcpr.avail_in = min(lOrigToDo,BlockSizeCompress);
        zcpr.avail_out = BlockSizeCompress;
        ret=deflate(&zcpr,(zcpr.avail_in==lOrigToDo) ? Z_FINISH : Z_SYNC_FLUSH);
        lOrigDone += (zcpr.total_in-all_read_before);
        lOrigToDo -= (zcpr.total_in-all_read_before);
        step++;
    } while (ret==Z_OK);

    lSizeCpr=zcpr.total_out;
    deflateEnd(&zcpr);
    dwGetTick=GetTickCount()-dwGetTick;
    dwMsecQP=GetMsecSincePerfCounter(li_qp,TRUE);
    dwResRdtsc=GetResRdtsc(li_rdtsc,TRUE);
    printf("total compress size = %u, in %u step\n",lSizeCpr,step);
    printf("time = %u msec = %f sec\n",dwGetTick,dwGetTick/(double)1000.);
    printf("defcpr time QP = %u msec = %f sec\n",dwMsecQP,dwMsecQP/(double)1000.);
```

```
    printf("defcpr result rdtsc = %I64x\n\n", dwResRdtsc.QuadPart);
}

CprPtr=(unsigned char*)realloc(CprPtr,lSizeCpr);
UncprPtr=(unsigned char*)malloc(lBufferSizeUncpr + BlockSizeUncompress);

BeginCountPerfCounter(&li_qp,TRUE);
dwGetTick=GetTickCount();
BeginCountRdtsc(&li_rdtsc);
{
    z_stream zcpr;
    int ret=Z_OK;
    long lOrigToDo = lSizeCpr;
    long lOrigDone = 0;
    int step=0;
    memset(&zcpr,0,sizeof(z_stream));
    inflateInit(&zcpr);

    zcpr.next_in = CprPtr;
    zcpr.next_out = UncprPtr;

    do
    {
        long all_read_before = zcpr.total_in;
        zcpr.avail_in = min(lOrigToDo,BlockSizeUncompress);
        zcpr.avail_out = BlockSizeUncompress;
        ret=inflate(&zcpr,Z_SYNC_FLUSH);
        lOrigDone += (zcpr.total_in-all_read_before);
        lOrigToDo -= (zcpr.total_in-all_read_before);
        step++;
    } while (ret==Z_OK);

    lSizeUncpr=zcpr.total_out;
    inflateEnd(&zcpr);
    dwGetTick=GetTickCount()-dwGetTick;
    dwMsecQP=GetMsecSincePerfCounter(li_qp,TRUE);
    dwResRdtsc=GetResRdtsc(li_rdtsc,TRUE);
    printf("total uncompress size = %u, in %u step\n",lSizeUncpr,step);
    printf("time = %u msec = %f sec\n",dwGetTick,dwGetTick/(double)1000.);
    printf("uncpr time QP = %u msec = %f sec\n",dwMsecQP,dwMsecQP/(double)1000.);
    printf("uncpr result rdtsc = %I64x\n\n",dwResRdtsc.QuadPart);
}

if (lSizeUncpr==lFileSize)
{
    if (memcmp(FilePtr,UncprPtr,lFileSize)==0)
        printf("compare ok\n");
}

return 0;
}
```

To build testzLib with Visual Studio 2005:

copy to a directory file from :

- root of zLib tree
- contrib/testzlib
- contrib/masmx86
- contrib/masmx64
- contrib/vstudio/vc7

and open testzlib8.sln

```
CC=cc  
CFLAGS=-g
```

```
untgz: untgz.o ../../libz.a  
        $(CC) $(CFLAGS) -o untgz untgz.o -L../../ -lz
```

```
untgz.o: untgz.c ../../zlib.h  
        $(CC) $(CFLAGS) -c -I../../ untgz.c
```

```
../../libz.a:  
        cd ../../; ./configure; make
```

```
clean:  
        rm -f untgz untgz.o *~
```

```
CC=cl
CFLAGS=-MD
```

```
untgz.exe: untgz.obj ..\..\zlib.lib
$(CC) $(CFLAGS) untgz.obj ..\..\zlib.lib
```

```
untgz.obj: untgz.c ..\..\zlib.h
$(CC) $(CFLAGS) -c -I..\.. untgz.c
```

```
..\..\zlib.lib:
cd ..\..
$(MAKE) -f win32\makefile.msc
cd contrib\untgz
```

```
clean:
-del untgz.obj
-del untgz.exe
```

```
/*
 * untgz.c -- Display contents and extract files from a gzip'd TAR file
 *
 * written by Pedro A. Aranda Gutierrez <paag@tid.es>
 * adaptation to Unix by Jean-loup Gailly <jloup@gzip.org>
 * various fixes by Cosmin Truta <cosmint@cs.ubbcluj.ro>
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <errno.h>

#include "zlib.h"

#ifdef unix
# include <unistd.h>
#else
# include <direct.h>
# include <io.h>
#endif

#ifdef WIN32
#include <windows.h>
# ifndef F_OK
#   define F_OK 0
# endif
# define mkdir(dirname,mode) _mkdir(dirname)
# ifdef _MSC_VER
#   define access(path,mode) _access(path,mode)
#   define chmod(path,mode) _chmod(path,mode)
#   define strdup(str) _strdup(str)
# endif
#else
# include <utime.h>
#endif

/* values used in typeflag field */

#define REGTYPE '0' /* regular file */
#define AREGTYPE '\0' /* regular file */
#define LNKTYPE '1' /* link */
#define SYMTYPE '2' /* reserved */
#define CHRTYPE '3' /* character special */
#define BLKTYPE '4' /* block special */
#define DIRTYPE '5' /* directory */
#define FIFOTYPE '6' /* FIFO special */
#define CONTTYPE '7' /* reserved */

/* GNU tar extensions */

#define GNUTYPE_DUMPDIR 'D' /* file names from dumped directory */
#define GNUTYPE_LONGLINK 'K' /* long link name */
#define GNUTYPE_LONGNAME 'L' /* long file name */
#define GNUTYPE_MULTIVOL 'M' /* continuation of file from another volume */
#define GNUTYPE_NAMES 'N' /* file name that does not fit into main hdr */
#define GNUTYPE_SPARSE 'S' /* sparse file */
#define GNUTYPE_VOLHDR 'V' /* tape/volume header */

/* tar header */

#define BLOCKSIZE 512
#define SHORTNAME_SIZE 100

struct tar_header
{
    /* byte offset */
    char name[100]; /* 0 */
    char mode[8]; /* 100 */
    char uid[8]; /* 108 */
    char gid[8]; /* 116 */
    char size[12]; /* 124 */
}
```

```
char mtime[12];          /* 136 */
char chksum[8];          /* 148 */
char typeflag;           /* 156 */
char linkname[100];      /* 157 */
char magic[6];           /* 257 */
char version[2];        /* 263 */
char uname[32];         /* 265 */
char gname[32];         /* 297 */
char devmajor[8];       /* 329 */
char devminor[8];       /* 337 */
char prefix[155];       /* 345 */
                          /* 500 */
};

union tar_buffer
{
    char          buffer[BLOCKSIZE];
    struct tar_header header;
};

struct attr_item
{
    struct attr_item *next;
    char          *fname;
    int           mode;
    time_t        time;
};

enum { TGZ_EXTRACT, TGZ_LIST, TGZ_INVALID };

char *TGZfname          OF((const char *));
void TGZnotfound        OF((const char *));

int getoct              OF((char *, int));
char *strtime           OF((time_t *));
int setfiletime         OF((char *, time_t));
void push_attr          OF((struct attr_item **, char *, int, time_t));
void restore_attr       OF((struct attr_item **));

int ExprMatch           OF((char *, char *));

int mkdir               OF((char *));
int matchname           OF((int, int, char **, char *));

void error              OF((const char *));
int tar                 OF((gzFile, int, int, int, char **));

void help               OF((int));
int main                OF((int, char **));

char *prog;

const char *TGZsuffix[] = { "\0", ".tar", ".tar.gz", ".taz", ".tgz", NULL };

/* return the file name of the TGZ archive */
/* or NULL if it does not exist */

char *TGZfname (const char *arcname)
{
    static char buffer[1024];
    int origlen,i;

    strcpy(buffer,arcname);
    origlen = strlen(buffer);

    for (i=0; TGZsuffix[i]; i++)
    {
        strcpy(buffer+origlen,TGZsuffix[i]);
        if (access(buffer,F_OK) == 0)
            return buffer;
    }
    return NULL;
}
```

```

/* error message for the filename */
void TGZnotfound (const char *arcname)
{
    int i;

    fprintf(stderr, "%s: Couldn't find ", prog);
    for (i=0; TGZsuffix[i]; i++)
        fprintf(stderr, (TGZsuffix[i+1]) ? "%s%s, " : "or %s%s\n",
                    arcname,
                    TGZsuffix[i]);
    exit(1);
}

/* convert octal digits to int */
/* on error return -1 */
int getoct (char *p, int width)
{
    int result = 0;
    char c;

    while (width--)
    {
        c = *p++;
        if (c == 0)
            break;
        if (c == ' ')
            continue;
        if (c < '0' || c > '7')
            return -1;
        result = result * 8 + (c - '0');
    }
    return result;
}

/* convert time_t to string */
/* use the "YYYY/MM/DD hh:mm:ss" format */
char *strtime (time_t *t)
{
    struct tm *local;
    static char result[32];

    local = localtime(t);
    sprintf(result, "%4d/%02d/%02d %02d:%02d:%02d",
            local->tm_year+1900, local->tm_mon+1, local->tm_mday,
            local->tm_hour, local->tm_min, local->tm_sec);
    return result;
}

/* set file time */
int setfiletime (char *fname, time_t ftime)
{
#ifdef WIN32
    static int isWinNT = -1;
    SYSTEMTIME st;
    FILETIME locft, modft;
    struct tm *loctm;
    HANDLE hFile;
    int result;

    loctm = localtime(&ftime);
    if (loctm == NULL)
        return -1;

    st.wYear = (WORD)loctm->tm_year + 1900;
    st.wMonth = (WORD)loctm->tm_mon + 1;
    st.wDayOfWeek = (WORD)loctm->tm_wday;

```



```

st.wDay          = (WORD)loctm->tm_mday;
st.wHour         = (WORD)loctm->tm_hour;
st.wMinute       = (WORD)loctm->tm_min;
st.wSecond       = (WORD)loctm->tm_sec;
st.wMilliseconds = 0;
if (!SystemTimeToFileTime(&st, &locft) ||
    !LocalFileTimeToFileTime(&locft, &modft))
    return -1;

if (isWinNT < 0)
    isWinNT = (GetVersion() < 0x80000000) ? 1 : 0;
hFile = CreateFile(fname, GENERIC_WRITE, 0, NULL, OPEN_EXISTING,
    (isWinNT ? FILE_FLAG_BACKUP_SEMANTICS : 0),
    NULL);
if (hFile == INVALID_HANDLE_VALUE)
    return -1;
result = SetFileTime(hFile, NULL, NULL, &modft) ? 0 : -1;
CloseHandle(hFile);
return result;
#else
    struct utimbuf settime;

    settime.actime = settime.modtime = ftime;
    return utime(fname, &settime);
#endif
}

/* push file attributes */
void push_attr(struct attr_item **list, char *fname, int mode, time_t time)
{
    struct attr_item *item;

    item = (struct attr_item *)malloc(sizeof(struct attr_item));
    if (item == NULL)
        error("Out of memory");
    item->fname = strdup(fname);
    item->mode = mode;
    item->time = time;
    item->next = *list;
    *list = item;
}

/* restore file attributes */
void restore_attr(struct attr_item **list)
{
    struct attr_item *item, *prev;

    for (item = *list; item != NULL; )
    {
        setfiletime(item->fname, item->time);
        chmod(item->fname, item->mode);
        prev = item;
        item = item->next;
        free(prev);
    }
    *list = NULL;
}

/* match regular expression */
#define ISSPECIAL(c) (((c) == '*') || ((c) == '/'))

int ExprMatch (char *string, char *expr)
{
    while (1)
    {
        if (ISSPECIAL(*expr))
        {
            if (*expr == '/')

```

```

        {
            if (*string != '\\' && *string != '/')
                return 0;
            string++; expr++;
        }
        else if (*expr == '*')
        {
            if (*expr++ == 0)
                return 1;
            while (*++string != *expr)
                if (*string == 0)
                    return 0;
        }
    }
    else
    {
        if (*string != *expr)
            return 0;
        if (*expr++ == 0)
            return 1;
        string++;
    }
}

/* recursive mkdir */
/* abort on ENOENT; ignore other errors like "directory already exists" */
/* return 1 if OK */
/*      0 on error */

int mkdir (char *newdir)
{
    char *buffer = strdup(newdir);
    char *p;
    int len = strlen(buffer);

    if (len <= 0) {
        free(buffer);
        return 0;
    }
    if (buffer[len-1] == '/') {
        buffer[len-1] = '\0';
    }
    if (mkdir(buffer, 0755) == 0)
    {
        free(buffer);
        return 1;
    }

    p = buffer+1;
    while (1)
    {
        char hold;

        while(*p && *p != '\\' && *p != '/')
            p++;
        hold = *p;
        *p = 0;
        if ((mkdir(buffer, 0755) == -1) && (errno == ENOENT))
        {
            fprintf(stderr, "%s: Couldn't create directory %s\n", prog, buffer);
            free(buffer);
            return 0;
        }
        if (hold == 0)
            break;
        *p++ = hold;
    }
    free(buffer);
    return 1;
}

```

```

int matchname (int arg,int argc,char **argv,char *fname)
{
    if (arg == argc)          /* no arguments given (untgz tgzarchive) */
        return 1;

    while (arg < argc)
        if (ExprMatch(fname,argv[arg++]))
            return 1;

    return 0; /* ignore this for the moment being */
}

/* tar file list or extract */

int tar (gzFile in,int action,int arg,int argc,char **argv)
{
    union tar_buffer buffer;
    int len;
    int err;
    int getheader = 1;
    int remaining = 0;
    FILE *outfile = NULL;
    char fname[BLOCKSIZE];
    int tarmode;
    time_t tartime;
    struct attr_item *attributes = NULL;

    if (action == TGZ_LIST)
        printf("  date    time    size           file\n"
               "  -----\n");
    while (1)
    {
        len = gzread(in, &buffer, BLOCKSIZE);
        if (len < 0)
            error(gzerror(in, &err));
        /*
         * Always expect complete blocks to process
         * the tar information.
         */
        if (len != BLOCKSIZE)
        {
            action = TGZ_INVALID; /* force error exit */
            remaining = 0;        /* force I/O cleanup */
        }

        /*
         * If we have to get a tar header
         */
        if (getheader >= 1)
        {
            /*
             * if we met the end of the tar
             * or the end-of-tar block,
             * we are done
             */
            if (len == 0 || buffer.header.name[0] == 0)
                break;

            tarmode = getoct(buffer.header.mode,8);
            tartime = (time_t)getoct(buffer.header.mtime,12);
            if (tarmode == -1 || tartime == (time_t)-1)
            {
                buffer.header.name[0] = 0;
                action = TGZ_INVALID;
            }

            if (getheader == 1)
            {
                strncpy(fname,buffer.header.name,SHORTNAMESIZE);
                if (fname[SHORTNAMESIZE-1] != 0)
                    fname[SHORTNAMESIZE] = 0;
            }
            else

```

```

    {
        /*
         * The file name is longer than SHORTNAMESIZE
         */
        if (strncmp(fname,buffer.header.name,SHORTNAMESIZE-1) != 0)
            error("bad long name");
        getheader = 1;
    }

/*
 * Act according to the type flag
 */
switch (buffer.header.typeflag)
{
    case DIRTYPE:
        if (action == TGZ_LIST)
            printf(" %s <dir> %s\n",strtime(&tartime),fname);
        if (action == TGZ_EXTRACT)
        {
            mkdir(fname);
            push_attr(&attributes,fname,tarmode,tartime);
        }
        break;
    case REGTYPE:
    case AREGTYPE:
        remaining = getoct(buffer.header.size,12);
        if (remaining == -1)
        {
            action = TGZ_INVALID;
            break;
        }
        if (action == TGZ_LIST)
            printf(" %s %9d %s\n",strtime(&tartime),remaining,fname);
        else if (action == TGZ_EXTRACT)
        {
            if (matchname(arg,argc,argv,fname))
            {
                outfile = fopen(fname,"wb");
                if (outfile == NULL) {
                    /* try creating directory */
                    char *p = strrchr(fname, '/');
                    if (p != NULL) {
                        *p = '\0';
                        mkdir(fname);
                        *p = '/';
                        outfile = fopen(fname,"wb");
                    }
                }
                if (outfile != NULL)
                    printf("Extracting %s\n",fname);
                else
                    fprintf(stderr, "%s: Couldn't create %s",prog,fname);
            }
            else
                outfile = NULL;
        }
        getheader = 0;
        break;
    case GNUTYPE_LONGLINK:
    case GNUTYPE_LONGNAME:
        remaining = getoct(buffer.header.size,12);
        if (remaining < 0 || remaining >= BLOCKSIZE)
        {
            action = TGZ_INVALID;
            break;
        }
        len = gzread(in, fname, BLOCKSIZE);
        if (len < 0)
            error(gzerror(in, &err));
        if (fname[BLOCKSIZE-1] != 0 || (int)strlen(fname) > remaining)
        {
            action = TGZ_INVALID;
            break;
        }
}

```

```

        getheader = 2;
        break;
    default:
        if (action == TGZ_LIST)
            printf(" %s  <---> %s\n", strtime(&tartime), fname);
        break;
    }
}
else
{
    unsigned int bytes = (remaining > BLOCKSIZE) ? BLOCKSIZE : remaining;

    if (outfile != NULL)
    {
        if (fwrite(&buffer, sizeof(char), bytes, outfile) != bytes)
        {
            fprintf(stderr,
                "%s: Error writing %s -- skipping\n", prog, fname);
            fclose(outfile);
            outfile = NULL;
            remove(fname);
        }
        remaining -= bytes;
    }

    if (remaining == 0)
    {
        getheader = 1;
        if (outfile != NULL)
        {
            fclose(outfile);
            outfile = NULL;
            if (action != TGZ_INVALID)
                push_attr(&attributes, fname, tarmode, tartime);
        }
    }

    /*
     * Abandon if errors are found
     */
    if (action == TGZ_INVALID)
    {
        error("broken archive");
        break;
    }
}

/*
 * Restore file modes and time stamps
 */
restore_attr(&attributes);

if (gzclose(in) != Z_OK)
    error("failed gzclose");

return 0;
}

/* ===== */

void help(int exitval)
{
    printf("untgz version 0.2.1\n"
        "  using zlib version %s\n",
        zlibVersion());
    printf("Usage: untgz file.tgz      extract all files\n"
        "       untgz file.tgz fname ... extract selected files\n"
        "       untgz -l file.tgz    list archive contents\n"
        "       untgz -h             display this help\n");
    exit(exitval);
}

```

```
void error(const char *msg)
{
    fprintf(stderr, "%s: %s\n", prog, msg);
    exit(1);
}

/* ===== */

#ifdef WIN32 && defined(__GNUC__)
int _CRT_glob = 0; /* disable argument globbing in MinGW */
#endif

int main(int argc, char **argv)
{
    int          action = TGZ_EXTRACT;
    int          arg = 1;
    char         *TGZfile;
    gzFile       *f;

    prog = strrchr(argv[0], '\\');
    if (prog == NULL)
    {
        prog = strrchr(argv[0], '/');
        if (prog == NULL)
        {
            prog = strrchr(argv[0], ':');
            if (prog == NULL)
                prog = argv[0];
            else
                prog++;
        }
        else
            prog++;
    }
    else
        prog++;

    if (argc == 1)
        help(0);

    if (strcmp(argv[arg], "-l") == 0)
    {
        action = TGZ_LIST;
        if (argc == ++arg)
            help(0);
    }
    else if (strcmp(argv[arg], "-h") == 0)
    {
        help(0);
    }

    if ((TGZfile = TGZfname(argv[arg])) == NULL)
        TGZnotfound(argv[arg]);

    ++arg;
    if ((action == TGZ_LIST) && (arg != argc))
        help(1);

    /*
     * Process the TGZ file
     */
    switch(action)
    {
        case TGZ_LIST:
        case TGZ_EXTRACT:
            f = gzopen(TGZfile, "rb");
            if (f == NULL)
            {
                fprintf(stderr, "%s: Couldn't gzopen %s\n", prog, TGZfile);
                return 1;
            }
            exit(tar(f, action, arg, argc, argv));
        break;
    }
```

```
    default:
        error( "Unknown option" );
        exit(1);
    }

    return 0;
}
```

Building instructions for the DLL versions of Zlib 1.2.3

This directory contains projects that build zlib and minizip using Microsoft Visual C++ 7.0/7.1, and Visual C++ .

You don't need to build these projects yourself. You can download the binaries from:

<http://www.winimage.com/zLibDll>

More information can be found at this site.

Build instructions for Visual Studio 7.x (32 bits)

- Uncompress current zlib, including all contrib/* files
- Download the crtdll library from
<http://www.winimage.com/zLibDll/crtdll.zip>
Unzip crtdll.zip to extract crtdll.lib on contrib\vstudio\vc7.
- Open contrib\vstudio\vc7\zlibvc.sln with Microsoft Visual C++ 7.x (Visual Studio .Net 2002 or 2003).

Build instructions for Visual Studio 2005 (32 bits or 64 bits)

- Uncompress current zlib, including all contrib/* files
- For 32 bits only: download the crtdll library from
<http://www.winimage.com/zLibDll/crtdll.zip>
Unzip crtdll.zip to extract crtdll.lib on contrib\vstudio\vc8.
- Open contrib\vstudio\vc8\zlibvc.sln with Microsoft Visual C++ 8.0

Build instructions for Visual Studio 2005 64 bits, PSDK compiler

at the time of writing this text file, Visual Studio 2005 (and Microsoft Visual C++ 8.0) is on the beta 2 stage.
Using you can get the free 64 bits compiler from Platform SDK, which is NOT a beta, and compile using the Visual studio 2005 IDE see <http://www.winimage.com/misc/sdk64onvs2005/> for instruction

- Uncompress current zlib, including all contrib/* files
- start Visual Studio 2005 from a platform SDK command prompt, using the /useenv switch
- Open contrib\vstudio\vc8\zlibvc.sln with Microsoft Visual C++ 8.0

Important

- To use zlibwapi.dll in your application, you must define the macro ZLIB_WINAPI when compiling your application's source files.

Additional notes

- This DLL, named zlibwapi.dll, is compatible to the old zlib.dll built by Gilles Vollant from the zlib 1.1.x sources, and distributed at <http://www.winimage.com/zLibDll>
It uses the WINAPI calling convention for the exported functions, and includes the minizip functionality. If your application needs that particular build of zlib.dll, you can rename zlibwapi.dll to zlib.dll.
- The new DLL was renamed because there exist several incompatible versions of zlib.dll on the Internet.
- There is also an official DLL build of zlib, named zlib1.dll. This one is exporting the functions using the CDECL convention. See the file win32\DLL_FAQ.txt found in this zlib distribution.
- There used to be a ZLIB_DLL macro in zlib 1.1.x, but now this symbol has a slightly different effect. To avoid compatibility problems, do not define it here.

Gilles Vollant
info@winimage.com


```

<?xml version="1.0" encoding = "Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="7.00"
  Name="miniunz"
  ProjectGUID="{C52F9E7B-498A-42BE-8DB4-85A15694382A}"
  Keyword="Win32Proj">
  <Platforms>
    <Platform
      Name="Win32" />
  </Platforms>
  <Configurations>
    <Configuration
      Name="Debug|Win32"
      OutputDirectory="Debug"
      IntermediateDirectory="Debug"
      ConfigurationType="1"
      CharacterSet="2">
      <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\..\minizip"
        PreprocessorDefinitions="WIN32;ZLIB_WINAPI;_DEBUG;_CONSOLE"
        MinimalRebuild="TRUE"
        BasicRuntimeChecks="3"
        RuntimeLibrary="5"
        UsePrecompiledHeader="0"
        WarningLevel="3"
        Detect64BitPortabilityProblems="TRUE"
        DebugInformationFormat="4" />
      <Tool
        Name="VCCustomBuildTool" />
      <Tool
        Name="VCLinkerTool"
        OutputFile="$(OutDir)/miniunz.exe"
        LinkIncremental="2"
        GenerateDebugInformation="TRUE"
        ProgramDatabaseFile="$(OutDir)/miniunz.pdb"
        SubSystem="1"
        TargetMachine="1" />
      <Tool
        Name="VCMTDLTool" />
      <Tool
        Name="VCPPostBuildEventTool" />
      <Tool
        Name="VCPPreBuildEventTool" />
      <Tool
        Name="VCPPreLinkEventTool" />
      <Tool
        Name="VCResourceCompilerTool" />
      <Tool
        Name="VCWebServiceProxyGeneratorTool" />
      <Tool
        Name="VCWebDeploymentTool" />
    </Configuration>
    <Configuration
      Name="Release|Win32"
      OutputDirectory="Release"
      IntermediateDirectory="Release"
      ConfigurationType="1"
      CharacterSet="2">
      <Tool
        Name="VCCLCompilerTool"
        Optimization="2"
        InlineFunctionExpansion="1"
        OmitFramePointers="TRUE"
        AdditionalIncludeDirectories="..\..\..\..\..\minizip"
        PreprocessorDefinitions="WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE"
        StringPooling="TRUE"
        RuntimeLibrary="4"
        EnableFunctionLevelLinking="TRUE"
        UsePrecompiledHeader="0"

```

```
WarningLevel="3"
Detect64BitPortabilityProblems="TRUE"
DebugInformationFormat="3"/>
<Tool
  Name="VCCustomBuildTool"/>
<Tool
  Name="VCLinkerTool"
  OutputFile="$(OutDir)/miniunz.exe"
  LinkIncremental="1"
  GenerateDebugInformation="TRUE"
  SubSystem="1"
  OptimizeReferences="2"
  EnableCOMDATFolding="2"
  OptimizeForWindows98="1"
  TargetMachine="1"/>
<Tool
  Name="VCIDLTool"/>
<Tool
  Name="VCPePostBuildEventTool"/>
<Tool
  Name="VCPePreBuildEventTool"/>
<Tool
  Name="VCPePreLinkEventTool"/>
<Tool
  Name="VCResourceCompilerTool"/>
<Tool
  Name="VCWebServiceProxyGeneratorTool"/>
<Tool
  Name="VCWebDeploymentTool"/>
</Configuration>
</Configurations>
<Files>
  <Filter
    Name="Source Files"
    Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm">
    <File
      RelativePath="..\..\minizip\miniunz.c">
    </File>
    </Filter>
    <Filter
      Name="Header Files"
      Filter="h;hpp;hxx;hm;inl;inc">
    </Filter>
    <Filter
      Name="Resource Files"
      Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe">
    </Filter>
    <File
      RelativePath="ReleaseDll\zlibwapi.lib">
    </File>
  </Files>
  <Globals>
  </Globals>
</VisualStudioProject>
```

```

<?xml version="1.0" encoding = "Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="7.00"
  Name="minizip"
  ProjectGUID="{48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}"
  Keyword="Win32Proj">
  <Platforms>
    <Platform
      Name="Win32" />
  </Platforms>
  <Configurations>
    <Configuration
      Name="Debug|Win32"
      OutputDirectory="Debug"
      IntermediateDirectory="Debug"
      ConfigurationType="1"
      CharacterSet="2">
      <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\minizip"
        PreprocessorDefinitions="WIN32;ZLIB_WINAPI;_DEBUG;_CONSOLE"
        MinimalRebuild="TRUE"
        BasicRuntimeChecks="3"
        RuntimeLibrary="5"
        UsePrecompiledHeader="0"
        WarningLevel="3"
        Detect64BitPortabilityProblems="TRUE"
        DebugInformationFormat="4" />
      <Tool
        Name="VCCustomBuildTool" />
      <Tool
        Name="VCLinkerTool"
        OutputFile="$(OutDir)/minizip.exe"
        LinkIncremental="2"
        GenerateDebugInformation="TRUE"
        ProgramDatabaseFile="$(OutDir)/minizip.pdb"
        SubSystem="1"
        TargetMachine="1" />
      <Tool
        Name="VCMTDLTool" />
      <Tool
        Name="VCPPostBuildEventTool" />
      <Tool
        Name="VCPPreBuildEventTool" />
      <Tool
        Name="VCPPreLinkEventTool" />
      <Tool
        Name="VCResourceCompilerTool" />
      <Tool
        Name="VCWebServiceProxyGeneratorTool" />
      <Tool
        Name="VCWebDeploymentTool" />
    </Configuration>
    <Configuration
      Name="Release|Win32"
      OutputDirectory="Release"
      IntermediateDirectory="Release"
      ConfigurationType="1"
      CharacterSet="2">
      <Tool
        Name="VCCLCompilerTool"
        Optimization="2"
        InlineFunctionExpansion="1"
        OmitFramePointers="TRUE"
        AdditionalIncludeDirectories="..\..\..\..\minizip"
        PreprocessorDefinitions="WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE"
        StringPooling="TRUE"
        RuntimeLibrary="4"
        EnableFunctionLevelLinking="TRUE"
        UsePrecompiledHeader="0"

```

```
WarningLevel="3"
Detect64BitPortabilityProblems="TRUE"
DebugInformationFormat="3"/>
<Tool
  Name="VCCustomBuildTool"/>
<Tool
  Name="VCLinkerTool"
  OutputFile="$(OutDir)/minizip.exe"
  LinkIncremental="1"
  GenerateDebugInformation="TRUE"
  SubSystem="1"
  OptimizeReferences="2"
  EnableCOMDATFolding="2"
  OptimizeForWindows98="1"
  TargetMachine="1"/>
<Tool
  Name="VCMidlTool"/>
<Tool
  Name="VCPostBuildEventTool"/>
<Tool
  Name="VCPreBuildEventTool"/>
<Tool
  Name="VCPreLinkEventTool"/>
<Tool
  Name="VCResourceCompilerTool"/>
<Tool
  Name="VCWebServiceProxyGeneratorTool"/>
<Tool
  Name="VCWebDeploymentTool"/>
</Configuration>
</Configurations>
<Files>
  <Filter
    Name="Source Files"
    Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm">
    <File
      RelativePath="../../minizip/minizip.c">
    </File>
  </Filter>
  <Filter
    Name="Header Files"
    Filter="h;hpp;hxx;hm;inl;inc">
  </Filter>
  <Filter
    Name="Resource Files"
    Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe">
  </Filter>
  <File
    RelativePath="ReleaseDll\zlibwapi.lib">
  </File>
</Files>
<Globals>
</Globals>
</VisualStudioProject>
```

```

<?xml version="1.0" encoding = "Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="7.00"
  Name="testZlibDll"
  ProjectGUID="{AA6666AA-E09F-4135-9C0C-4FE50C3C654C}"
  Keyword="Win32Proj">
  <Platforms>
    <Platform
      Name="Win32" />
  </Platforms>
  <Configurations>
    <Configuration
      Name="Debug|Win32"
      OutputDirectory="Debug"
      IntermediateDirectory="Debug"
      ConfigurationType="1"
      CharacterSet="2">
      <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\.."
        PreprocessorDefinitions="WIN32;ZLIB_WINAPI;_DEBUG;_CONSOLE"
        MinimalRebuild="TRUE"
        BasicRuntimeChecks="3"
        RuntimeLibrary="5"
        UsePrecompiledHeader="0"
        WarningLevel="3"
        Detect64BitPortabilityProblems="TRUE"
        DebugInformationFormat="4" />
      <Tool
        Name="VCCustomBuildTool" />
      <Tool
        Name="VCLinkerTool"
        OutputFile="$(OutDir)/testzlib.exe"
        LinkIncremental="2"
        GenerateDebugInformation="TRUE"
        ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
        SubSystem="1"
        TargetMachine="1" />
      <Tool
        Name="VCMTIDLTool" />
      <Tool
        Name="VCPostBuildEventTool" />
      <Tool
        Name="VCPreBuildEventTool" />
      <Tool
        Name="VCPreLinkEventTool" />
      <Tool
        Name="VCResourceCompilerTool" />
      <Tool
        Name="VCWebServiceProxyGeneratorTool" />
      <Tool
        Name="VCWebDeploymentTool" />
    </Configuration>
    <Configuration
      Name="Release|Win32"
      OutputDirectory="Release"
      IntermediateDirectory="Release"
      ConfigurationType="1"
      CharacterSet="2">
      <Tool
        Name="VCCLCompilerTool"
        Optimization="2"
        InlineFunctionExpansion="1"
        OmitFramePointers="TRUE"
        AdditionalIncludeDirectories="..\..\.."
        PreprocessorDefinitions="WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE"
        StringPooling="TRUE"
        RuntimeLibrary="4"
        EnableFunctionLevelLinking="TRUE"
        UsePrecompiledHeader="0"

```

```
WarningLevel="3"
Detect64BitPortabilityProblems="TRUE"
DebugInformationFormat="3"/>
<Tool
  Name="VCCustomBuildTool"/>
<Tool
  Name="VCLinkerTool"
  OutputFile="$(OutDir)/testzlib.exe"
  LinkIncremental="1"
  GenerateDebugInformation="TRUE"
  SubSystem="1"
  OptimizeReferences="2"
  EnableCOMDATFolding="2"
  OptimizeForWindows98="1"
  TargetMachine="1"/>
<Tool
  Name="VCIDLTool"/>
<Tool
  Name="VCPostBuildEventTool"/>
<Tool
  Name="VCPreBuildEventTool"/>
<Tool
  Name="VCPreLinkEventTool"/>
<Tool
  Name="VCResourceCompilerTool"/>
<Tool
  Name="VCWebServiceProxyGeneratorTool"/>
<Tool
  Name="VCWebDeploymentTool"/>
</Configuration>
</Configurations>
<Files>
  <Filter
    Name="Source Files"
    Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm">
    <File
      RelativePath="../../testzlib/testzlib.c">
    </File>
    </Filter>
    <Filter
      Name="Header Files"
      Filter="h;hpp;hxx;hm;inl;inc">
    </Filter>
    <Filter
      Name="Resource Files"
      Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe">
    </Filter>
    <File
      RelativePath="ReleaseDll\zlibwapi.lib">
    </File>
  </Files>
  <Globals>
  </Globals>
</VisualStudioProject>
```

```
#include <windows.h>

#define IDR_VERSION1 1
IDR_VERSION1 VERSIONINFO MOVEABLE IMPURE LOADONCALL DISCARDABLE
    FILEVERSION 1,2,3,0
    PRODUCTVERSION 1,2,3,0
    FILEFLAGSMASK VS_FFI_FILEFLAGSMASK
    FILEFLAGS 0
    FILEOS VOS_DOS_WINDOWS32
    FILETYPE VFT_DLL
    FILESUBTYPE 0 // not used
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904E4"
        //language ID = U.S. English, char set = Windows, Multilingual

        BEGIN
            VALUE "FileDescription", "zlib data compression library\0"
            VALUE "FileVersion", "1.2.3.0\0"
            VALUE "InternalName", "zlib\0"
            VALUE "OriginalFilename", "zlib.dll\0"
            VALUE "ProductName", "ZLib.DLL\0"
            VALUE "Comments", "DLL support by Alessandro Iacopetti & Gilles Vollant\0"
            VALUE "LegalCopyright", "(C) 1995-2003 Jean-loup Gailly & Mark Adler\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x0409, 1252
    END
END
```

```
<?xml version="1.0" encoding = "Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="7.00"
  Name="zlibstat"
  SccProjectName=""
  SccLocalPath="">
  <Platforms>
    <Platform
      Name="Win32" />
  </Platforms>
  <Configurations>
    <Configuration
      Name="Debug|Win32"
      OutputDirectory=".\zlibstatDebug"
      IntermediateDirectory=".\zlibstatDebug"
      ConfigurationType="4"
      UseOfMFC="0"
      ATLMinimizesCRunTimeLibraryUsage="FALSE">
      <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
        PreprocessorDefinitions="WIN32;ZLIB_WINAPI"
        ExceptionHandling="FALSE"
        RuntimeLibrary="5"
        PrecompiledHeaderFile=".\zlibstatDebug/zlibstat.pch"
        AssemblerListingLocation=".\zlibstatDebug/"
        ObjectFile=".\zlibstatDebug/"
        ProgramDataBaseFileName=".\zlibstatDebug/"
        WarningLevel="3"
        SuppressStartupBanner="TRUE"
        DebugInformationFormat="1" />
      <Tool
        Name="VCCustomBuildTool" />
      <Tool
        Name="VCLibrarianTool"
        AdditionalOptions="/NODEFAULTLIB"
        OutputFile=".\zlibstatDebug\zlibstat.lib"
        SuppressStartupBanner="TRUE" />
      <Tool
        Name="VCIDLTool" />
      <Tool
        Name="VCPostBuildEventTool" />
      <Tool
        Name="VCPreBuildEventTool" />
      <Tool
        Name="VCPreLinkEventTool" />
      <Tool
        Name="VCResourceCompilerTool"
        Culture="1036" />
      <Tool
        Name="VCWebServiceProxyGeneratorTool" />
    </Configuration>
    <Configuration
      Name="ReleaseAxp|Win32"
      OutputDirectory=".\zlibsta0"
      IntermediateDirectory=".\zlibsta0"
      ConfigurationType="4"
      UseOfMFC="0"
      ATLMinimizesCRunTimeLibraryUsage="FALSE">
      <Tool
        Name="VCCLCompilerTool"
        InlineFunctionExpansion="1"
        AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
        PreprocessorDefinitions="WIN32;ZLIB_WINAPI"
        StringPooling="TRUE"
        ExceptionHandling="FALSE"
        RuntimeLibrary="4"
        EnableFunctionLevelLinking="TRUE"
        PrecompiledHeaderFile=".\zlibsta0/zlibstat.pch"
        AssemblerListingLocation=".\zlibsta0/"
        ObjectFile=".\zlibsta0/"
        ProgramDataBaseFileName=".\zlibsta0/"
```



```

        WarningLevel="3"
        SuppressStartupBanner="TRUE" />
    <Tool
        Name="VCCustomBuildTool" />
    <Tool
        Name="VCLibrarianTool"
        AdditionalOptions="/NODEFAULTLIB "
        OutputFile=".\\zlibsta0\\zlibstat.lib"
        SuppressStartupBanner="TRUE" />
    <Tool
        Name="VCIDLTool" />
    <Tool
        Name="VCPostBuildEventTool" />
    <Tool
        Name="VCPreBuildEventTool" />
    <Tool
        Name="VCPreLinkEventTool" />
    <Tool
        Name="VCResourceCompilerTool" />
    <Tool
        Name="VCWebServiceProxyGeneratorTool" />
</Configuration>
<Configuration
    Name="Release|Win32"
    OutputDirectory=".\\zlibstat"
    IntermediateDirectory=".\\zlibstat"
    ConfigurationType="4"
    UseOfMFC="0"
    ATLMinimizesCRunTimeLibraryUsage="FALSE">
    <Tool
        Name="VCCLCompilerTool"
        InlineFunctionExpansion="1"
        AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
        PreprocessorDefinitions="WIN32;ZLIB_WINAPI;ASMV;ASMINF"
        StringPooling="TRUE"
        ExceptionHandling="FALSE"
        RuntimeLibrary="4"
        EnableFunctionLevelLinking="TRUE"
        PrecompiledHeaderFile=".\\zlibstat\\zlibstat.pch"
        AssemblerListingLocation=".\\zlibstat/"
        ObjectFile=".\\zlibstat/"
        ProgramDataBaseFileName=".\\zlibstat/"
        WarningLevel="3"
        SuppressStartupBanner="TRUE" />
    <Tool
        Name="VCCustomBuildTool" />
    <Tool
        Name="VCLibrarianTool"
        AdditionalOptions="..\..\masmx86\\gvm32.obj ....\masmx8
6\\inffas32.obj /NODEFAULTLIB "
        OutputFile=".\\zlibstat\\zlibstat.lib"
        SuppressStartupBanner="TRUE" />
    <Tool
        Name="VCIDLTool" />
    <Tool
        Name="VCPostBuildEventTool" />
    <Tool
        Name="VCPreBuildEventTool" />
    <Tool
        Name="VCPreLinkEventTool" />
    <Tool
        Name="VCResourceCompilerTool"
        Culture="1036" />
    <Tool
        Name="VCWebServiceProxyGeneratorTool" />
</Configuration>
<Configuration
    Name="ReleaseWithoutAsm|Win32"
    OutputDirectory="zlibstatWithoutAsm"
    IntermediateDirectory="zlibstatWithoutAsm"
    ConfigurationType="4"
    UseOfMFC="0"
    ATLMinimizesCRunTimeLibraryUsage="FALSE">
    <Tool

```

```
Name="VCCLCompilerTool"
InlineFunctionExpansion="1"
AdditionalIncludeDirectories="..\..\..\..\masmx86"
PreprocessorDefinitions="WIN32;ZLIB_WINAPI"
StringPooling="TRUE"
ExceptionHandling="FALSE"
RuntimeLibrary="4"
EnableFunctionLevelLinking="TRUE"
PrecompiledHeaderFile="..\zlibstat\zlibstat.pch"
AssemblerListingLocation="..\zlibstatWithoutAsm/"
ObjectFile="..\zlibstatWithoutAsm/"
ProgramDataBaseFileName="..\zlibstatWithoutAsm/"
WarningLevel="3"
SuppressStartupBanner="TRUE" />
<Tool
  Name="VCCustomBuildTool" />
<Tool
  Name="VCLibrarianTool"
  AdditionalOptions=" /NODEFAULTLIB "
  OutputFile="..\zlibstatWithoutAsm\zlibstat.lib"
  SuppressStartupBanner="TRUE" />
<Tool
  Name="VCIDLTool" />
<Tool
  Name="VCPostBuildEventTool" />
<Tool
  Name="VCPreBuildEventTool" />
<Tool
  Name="VCPreLinkEventTool" />
<Tool
  Name="VCResourceCompilerTool"
  Culture="1036" />
<Tool
  Name="VCWebServiceProxyGeneratorTool" />
</Configuration>
</Configurations>
<Files>
  <Filter
    Name="Source Files"
    Filter="">
    <File
      RelativePath="..\..\..\..\adler32.c">
    </File>
    <File
      RelativePath="..\..\..\..\compress.c">
    </File>
    <File
      RelativePath="..\..\..\..\crc32.c">
    </File>
    <File
      RelativePath="..\..\..\..\deflate.c">
    </File>
    <File
      RelativePath="..\..\..\..\masmx86\gvm32c.c">
    </File>
    <File
      RelativePath="..\..\..\..\gzio.c">
    </File>
    <File
      RelativePath="..\..\..\..\inffast.c">
    </File>
    <File
      RelativePath="..\..\..\..\inflate.c">
    </File>
    <File
      RelativePath="..\..\..\..\inftrees.c">
    </File>
    <File
      RelativePath="..\..\..\..\minizip\ioapi.c">
    </File>
  </Filter>
</Files>
```

```
        RelativePath="..\..\..\trees.c">
    </File>
    <File
        RelativePath="..\..\..\uncompr.c">
    </File>
    <File
        RelativePath="..\..\minizip\unzip.c">
    </File>
    <File
        RelativePath="..\..\minizip\zip.c">
    </File>
    <File
        RelativePath=".\zlib.rc">
    </File>
    <File
        RelativePath=".\zlibvc.def">
    </File>
    <File
        RelativePath="..\..\..\zutil.c">
    </File>
</Filter>
</Files>
<Globals>
</Globals>
</VisualStudioProject>
```

VERSION 1.23

HEAPSIZE 1048576,8192

EXPORTS

adler32	@1
compress	@2
crc32	@3
deflate	@4
deflateCopy	@5
deflateEnd	@6
deflateInit2_	@7
deflateInit_	@8
deflateParams	@9
deflateReset	@10
deflateSetDictionary	@11
gzclose	@12
gzdopen	@13
gzerror	@14
gzflush	@15
gzopen	@16
gzread	@17
gzwrite	@18
inflate	@19
inflateEnd	@20
inflateInit2_	@21
inflateInit_	@22
inflateReset	@23
inflateSetDictionary	@24
inflateSync	@25
uncompress	@26
zlibVersion	@27
gzprintf	@28
gzputc	@29
gzgetc	@30
gzseek	@31
gzrewind	@32
gtell	@33
gzeof	@34
gzsetparams	@35
zError	@36
inflateSyncPoint	@37
get_crc_table	@38
compress2	@39
gzputs	@40
gzgets	@41
inflateCopy	@42
inflateBackInit_	@43
inflateBack	@44
inflateBackEnd	@45
compressBound	@46
deflateBound	@47
gzclearerr	@48
gzungetc	@49
zlibCompileFlags	@50
deflatePrime	@51
unzOpen	@61
unzClose	@62
unzGetGlobalInfo	@63
unzGetCurrentFileInfo	@64
unzGoToFirstFile	@65
unzGoToNextFile	@66
unzOpenCurrentFile	@67
unzReadCurrentFile	@68
unzOpenCurrentFile3	@69
unztell	@70
unzeof	@71
unzCloseCurrentFile	@72
unzGetGlobalComment	@73
unzStringFileNameCompare	@74
unzLocateFile	@75
unzGetLocalExtrafield	@76

unzOpen2	@77
unzOpenCurrentFile2	@78
unzOpenCurrentFilePassword	@79
zipOpen	@80
zipOpenNewFileInZip	@81
zipWriteInFileInZip	@82
zipCloseFileInZip	@83
zipClose	@84
zipOpenNewFileInZip2	@86
zipCloseFileInZipRaw	@87
zipOpen2	@88
zipOpenNewFileInZip3	@89
unzGetFilePos	@100
unzGoToFilePos	@101
fill_win32_filefunc	@110

```
Microsoft Visual Studio Solution File, Format Version 7.00
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "zlibstat", "zlibstat.vcproj", "{745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "zlibvc", "zlibvc.vcproj", "{8FD826F8-3739-44E6-8CC8-997122E53B8D}"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "minizip", "minizip.vcproj", "{48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "miniunz", "miniunz.vcproj", "{C52F9E7B-498A-42BE-8DB4-85A15694382A}"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "testZlibDll", "testzlib.vcproj", "{A6666AA-E09F-4135-9C0C-4FE50C3C654C}"
EndProject
Global
    GlobalSection(SolutionConfiguration) = preSolution
        ConfigName.0 = Debug
        ConfigName.1 = Release
        ConfigName.2 = ReleaseAxp
        ConfigName.3 = ReleaseWithoutAsm
        ConfigName.4 = ReleaseWithoutCrtdll
    EndGlobalSection
    GlobalSection(ProjectDependencies) = postSolution
    EndGlobalSection
    GlobalSection(ProjectConfiguration) = postSolution
        {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug.ActiveCfg = Debug|Win32
        {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug.Build.0 = Debug|Win32
        {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release.ActiveCfg = Release|Win32
        {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release.Build.0 = Release|Win32
        {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseAxp.ActiveCfg = ReleaseAxp|Win32
        {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseAxp.Build.0 = ReleaseAxp|Win32
        {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm.ActiveCfg = ReleaseWithoutAsm|Win32
        {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm.Build.0 = ReleaseWithoutAsm|Win32
        {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutCrtdll.ActiveCfg = ReleaseWithoutCrtdll|Win32
        {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutCrtdll.Build.0 = ReleaseWithoutCrtdll|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug.ActiveCfg = Debug|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug.Build.0 = Debug|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release.ActiveCfg = Release|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release.Build.0 = Release|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseAxp.ActiveCfg = ReleaseAxp|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseAxp.Build.0 = ReleaseAxp|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm.ActiveCfg = ReleaseWithoutAsm|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm.Build.0 = ReleaseWithoutAsm|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutCrtdll.ActiveCfg = ReleaseWithoutCrtdll|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutCrtdll.Build.0 = ReleaseWithoutCrtdll|Win32
        {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug.ActiveCfg = Debug|Win32
        {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug.Build.0 = Debug|Win32
        {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release.ActiveCfg = Release|Win32
        {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release.Build.0 = Release|Win32
        {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseAxp.ActiveCfg = ReleaseAxp|Win32
        {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseAxp.Build.0 = ReleaseAxp|Win32
        {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm.ActiveCfg = ReleaseWithoutAsm|Win32
        {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm.Build.0 = ReleaseWithoutAsm|Win32
        {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutCrtdll.ActiveCfg = ReleaseWithoutCrtdll|Win32
        {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutCrtdll.Build.0 = ReleaseWithoutCrtdll|Win32
```

```
32      {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug.ActiveCfg = Debug|Win32
      {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug.Build.0 = Debug|Win32
      {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release.ActiveCfg = Release|Win32
      {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release.Build.0 = Release|Win32
      {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseAxp.ActiveCfg = Release|Win
ase|Win32
      {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseAxp.Build.0 = Release|Win32
      {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm.ActiveCfg = Rel
e|Win32
      {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm.Build.0 = Releas
e|Win32
      {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutCrtdll.ActiveCfg = R
ease|Win32
      {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutCrtdll.Build.0 = Rel
ease|Win32
32      {AA6666AA-E09F-4135-9C0C-4FE50C3C654C}.Debug.ActiveCfg = Debug|Win32
      {AA6666AA-E09F-4135-9C0C-4FE50C3C654C}.Debug.Build.0 = Debug|Win32
      {AA6666AA-E09F-4135-9C0C-4FE50C3C654C}.Release.ActiveCfg = Release|Win32
      {AA6666AA-E09F-4135-9C0C-4FE50C3C654C}.Release.Build.0 = Release|Win32
      {AA6666AA-E09F-4135-9C0C-4FE50C3C654C}.ReleaseAxp.ActiveCfg = Release|Win
ase|Win32
      {AA6666AA-E09F-4135-9C0C-4FE50C3C654C}.ReleaseAxp.Build.0 = Release|Win32
      {AA6666AA-E09F-4135-9C0C-4FE50C3C654C}.ReleaseWithoutAsm.ActiveCfg = Rel
e|Win32
      {AA6666AA-E09F-4135-9C0C-4FE50C3C654C}.ReleaseWithoutAsm.Build.0 = Releas
e|Win32
      {AA6666AA-E09F-4135-9C0C-4FE50C3C654C}.ReleaseWithoutCrtdll.ActiveCfg = R
ease|Win32
      {AA6666AA-E09F-4135-9C0C-4FE50C3C654C}.ReleaseWithoutCrtdll.Build.0 = Rel
ease|Win32
      EndGlobalSection
      GlobalSection(ExtensibilityGlobals) = postSolution
      EndGlobalSection
      GlobalSection(ExtensibilityAddIns) = postSolution
      EndGlobalSection
EndGlobal
```

```

<?xml version="1.0" encoding = "Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="7.00"
  Name="zlibvc"
  SccProjectName=""
  SccLocalPath="">
  <Platforms>
    <Platform
      Name="Win32" />
  </Platforms>
  <Configurations>
    <Configuration
      Name="Debug|Win32"
      OutputDirectory=".\\DebugDll"
      IntermediateDirectory=".\\DebugDll"
      ConfigurationType="2"
      UseOfMFC="0"
      ATLMinimizesCRunTimeLibraryUsage="FALSE">
      <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
        PreprocessorDefinitions="WIN32,ZLIB_WINAPI,ASMV,ASMINF"
        ExceptionHandling="FALSE"
        RuntimeLibrary="1"
        PrecompiledHeaderFile=".\\DebugDll/zlibvc.pch"
        AssemblerListingLocation=".\\DebugDll/"
        ObjectFile=".\\DebugDll/"
        ProgramDataBaseFileName=".\\DebugDll/"
        WarningLevel="3"
        SuppressStartupBanner="TRUE"
        DebugInformationFormat="4" />
      <Tool
        Name="VCCustomBuildTool" />
      <Tool
        Name="VCLinkerTool"
        AdditionalOptions="/MACHINE:I386"
        AdditionalDependencies="..\..\masmx86\gvm32.obj ..\..\m
asmx86\inffas32.obj"
        OutputFile=".\\DebugDll\zlibwapi.dll"
        LinkIncremental="2"
        SuppressStartupBanner="TRUE"
        ModuleDefinitionFile=".\\zlibvc.def"
        GenerateDebugInformation="TRUE"
        ProgramDatabaseFile=".\\DebugDll\zlibwapi.pdb"
        SubSystem="2"
        ImportLibrary=".\\DebugDll\zlibwapi.lib" />
      <Tool
        Name="VCIDLTool"
        PreprocessorDefinitions="_DEBUG"
        MkTypLibCompatible="TRUE"
        SuppressStartupBanner="TRUE"
        TargetEnvironment="1"
        TypeLibraryName=".\\DebugDll\zlibvc.tlb" />
      <Tool
        Name="VCPePostBuildEventTool" />
      <Tool
        Name="VCPePreBuildEventTool" />
      <Tool
        Name="VCPePreLinkEventTool" />
      <Tool
        Name="VCResourceCompilerTool"
        PreprocessorDefinitions="_DEBUG"
        Culture="1036" />
      <Tool
        Name="VCWebServiceProxyGeneratorTool" />
      <Tool
        Name="VCWebDeploymentTool" />
    </Configuration>
    <Configuration
      Name="ReleaseWithoutAsm|Win32"
      OutputDirectory=".\\zlibDllWithoutAsm"
      IntermediateDirectory=".\\zlibDllWithoutAsm"

```



```
ConfigurationType="2"
UseOfMFC="0"
ATLMinimizesCRunTimeLibraryUsage="FALSE"
WholeProgramOptimization="TRUE">
<Tool
  Name="VCCLCompilerTool"
  InlineFunctionExpansion="1"
  AdditionalIncludeDirectories="..\..\..\..\masmx86"
  PreprocessorDefinitions="WIN32,ZLIB_WINAPI"
  StringPooling="TRUE"
  ExceptionHandling="FALSE"
  RuntimeLibrary="0"
  EnableFunctionLevelLinking="TRUE"
  PrecompiledHeaderFile=".\zlibDllWithoutAsm/zlibvc.pch"
  AssemblerOutput="2"
  AssemblerListingLocation=".\zlibDllWithoutAsm/"
  ObjectFile=".\zlibDllWithoutAsm/"
  ProgramDataBaseFileName=".\zlibDllWithoutAsm/"
  BrowseInformation="1"
  WarningLevel="3"
  SuppressStartupBanner="TRUE"/>
<Tool
  Name="VCCustomBuildTool"/>
<Tool
  Name="VCLinkerTool"
  AdditionalOptions="/MACHINE:I386"
  AdditionalDependencies="crtdll.lib"
  OutputFile=".\zlibDllWithoutAsm\zlibwapi.dll"
  LinkIncremental="1"
  SuppressStartupBanner="TRUE"
  IgnoreAllDefaultLibraries="TRUE"
  ModuleDefinitionFile=".\zlibvc.def"
  ProgramDatabaseFile=".\zlibDllWithoutAsm/zlibwapi.pdb"
  GenerateMapFile="TRUE"
  MapFileName=".\zlibDllWithoutAsm/zlibwapi.map"
  SubSystem="2"
  OptimizeForWindows98="1"
  ImportLibrary=".\zlibDllWithoutAsm/zlibwapi.lib"/>
<Tool
  Name="VCIDLTool"
  PreprocessorDefinitions="NDEBUG"
  MkTypLibCompatible="TRUE"
  SuppressStartupBanner="TRUE"
  TargetEnvironment="1"
  TypeLibraryName=".\zlibDllWithoutAsm/zlibvc.tlb"/>
<Tool
  Name="VCPostBuildEventTool"/>
<Tool
  Name="VCPreBuildEventTool"/>
<Tool
  Name="VCPreLinkEventTool"/>
<Tool
  Name="VCResourceCompilerTool"
  PreprocessorDefinitions="NDEBUG"
  Culture="1036"/>
<Tool
  Name="VCWebServiceProxyGeneratorTool"/>
<Tool
  Name="VCWebDeploymentTool"/>
</Configuration>
<Configuration
  Name="ReleaseWithoutCrtDll|Win32"
  OutputDirectory=".\zlibDllWithoutCrtDll"
  IntermediateDirectory=".\zlibDllWithoutCrtDll"
  ConfigurationType="2"
  UseOfMFC="0"
  ATLMinimizesCRunTimeLibraryUsage="FALSE"
  WholeProgramOptimization="TRUE">
<Tool
  Name="VCCLCompilerTool"
  InlineFunctionExpansion="1"
  AdditionalIncludeDirectories="..\..\..\..\masmx86"
  PreprocessorDefinitions="WIN32,ZLIB_WINAPI,ASMV,ASMINF"
  StringPooling="TRUE"
```

```

        ExceptionHandling="FALSE"
        RuntimeLibrary="0"
        EnableFunctionLevelLinking="TRUE"
        PrecompiledHeaderFile=".\\zlibDllWithoutCrtDll/zlibvc.pch"
        AssemblerOutput="2"
        AssemblerListingLocation=".\\zlibDllWithoutCrtDll/"
        ObjectFile=".\\zlibDllWithoutCrtDll/"
        ProgramDataBaseFileName=".\\zlibDllWithoutCrtDll/"
        BrowseInformation="1"
        WarningLevel="3"
        SuppressStartupBanner="TRUE" />
    <Tool
        Name="VCCustomBuildTool" />
    <Tool
        Name="VCLinkerTool"
        AdditionalOptions="/MACHINE:I386"
        AdditionalDependencies="..\..\masmx86\gvmat32.obj ....\m
asmx86\inffas32.obj "
        OutputFile=".\\zlibDllWithoutCrtDll\zlibwapi.dll"
        LinkIncremental="1"
        SuppressStartupBanner="TRUE"
        IgnoreAllDefaultLibraries="FALSE"
        ModuleDefinitionFile=".\\zlibvc.def"
        ProgramDatabaseFile=".\\zlibDllWithoutCrtDll/zlibwapi.pdb"
        GenerateMapFile="TRUE"
        MapFileName=".\\zlibDllWithoutCrtDll/zlibwapi.map"
        SubSystem="2"
        OptimizeForWindows98="1"
        ImportLibrary=".\\zlibDllWithoutCrtDll/zlibwapi.lib" />
    <Tool
        Name="VCMidlTool"
        PreprocessorDefinitions="NDEBUG"
        MkTypLibCompatible="TRUE"
        SuppressStartupBanner="TRUE"
        TargetEnvironment="1"
        TypeLibraryName=".\\zlibDllWithoutCrtDll/zlibvc.tlb" />
    <Tool
        Name="VCPostBuildEventTool" />
    <Tool
        Name="VCPreBuildEventTool" />
    <Tool
        Name="VCPreLinkEventTool" />
    <Tool
        Name="VCResourceCompilerTool"
        PreprocessorDefinitions="NDEBUG"
        Culture="1036" />
    <Tool
        Name="VCWebServiceProxyGeneratorTool" />
    <Tool
        Name="VCWebDeploymentTool" />
</Configuration>
<Configuration
    Name="ReleaseAxp|Win32"
    OutputDirectory=".\\zlibvc__"
    IntermediateDirectory=".\\zlibvc__"
    ConfigurationType="2"
    UseOfMFC="0"
    ATLMinimizesCRunTimeLibraryUsage="FALSE"
    WholeProgramOptimization="TRUE">
    <Tool
        Name="VCCLCompilerTool"
        InlineFunctionExpansion="1"
        AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
        PreprocessorDefinitions="WIN32,ZLIB_WINAPI"
        StringPooling="TRUE"
        ExceptionHandling="FALSE"
        RuntimeLibrary="0"
        EnableFunctionLevelLinking="TRUE"
        PrecompiledHeaderFile=".\\zlibvc__/zlibvc.pch"
        AssemblerOutput="2"
        AssemblerListingLocation=".\\zlibvc__/"
        ObjectFile=".\\zlibvc__/"
        ProgramDataBaseFileName=".\\zlibvc__/"
        BrowseInformation="1"

```

```

WarningLevel="3"
SuppressStartupBanner="TRUE" />
<Tool
  Name="VCCustomBuildTool" />
<Tool
  Name="VCLinkerTool"
  AdditionalDependencies="crtddll.lib"
  OutputFile="zlibvc__\zlibwapi.dll"
  LinkIncremental="1"
  SuppressStartupBanner="TRUE"
  IgnoreAllDefaultLibraries="TRUE"
  ModuleDefinitionFile=".\\zlibvc.def"
  ProgramDatabaseFile=".\\zlibvc__\zlibwapi.pdb"
  GenerateMapFile="TRUE"
  MapFileName=".\\zlibvc__\zlibwapi.map"
  SubSystem="2"
  ImportLibrary=".\\zlibvc__\zlibwapi.lib" />
<Tool
  Name="VCMDLTool"
  PreprocessorDefinitions="NDEBUG"
  MkTypLibCompatible="TRUE"
  SuppressStartupBanner="TRUE"
  TargetEnvironment="1"
  TypeLibraryName=".\\zlibvc__\zlibvc.tlb" />
<Tool
  Name="VCPPostBuildEventTool" />
<Tool
  Name="VCPPreBuildEventTool" />
<Tool
  Name="VCPPreLinkEventTool" />
<Tool
  Name="VCResourceCompilerTool"
  PreprocessorDefinitions="NDEBUG"
  Culture="1036" />
<Tool
  Name="VCWebServiceProxyGeneratorTool" />
<Tool
  Name="VCWebDeploymentTool" />
</Configuration>
<Configuration
  Name="Release|Win32"
  OutputDirectory=".\\ReleaseDll"
  IntermediateDirectory=".\\ReleaseDll"
  ConfigurationType="2"
  UseOfMFC="0"
  ATLMinimizesCRunTimeLibraryUsage="FALSE"
  WholeProgramOptimization="TRUE">
<Tool
  Name="VCCLCompilerTool"
  InlineFunctionExpansion="1"
  AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
  PreprocessorDefinitions="WIN32,ZLIB_WINAPI,AS MV,ASMINF"
  StringPooling="TRUE"
  ExceptionHandling="FALSE"
  RuntimeLibrary="0"
  EnableFunctionLevelLinking="TRUE"
  PrecompiledHeaderFile=".\\ReleaseDll\zlibvc.pch"
  AssemblerOutput="2"
  AssemblerListingLocation=".\\ReleaseDll/"
  ObjectFile=".\\ReleaseDll/"
  ProgramDataBaseFileName=".\\ReleaseDll/"
  BrowseInformation="1"
  WarningLevel="3"
  SuppressStartupBanner="TRUE" />
<Tool
  Name="VCCustomBuildTool" />
<Tool
  Name="VCLinkerTool"
  AdditionalOptions="/MACHINE:I386"
  AdditionalDependencies="..\..\masmx86\gvmat32.obj ..\..\m
asmx86\inffas32.obj crtddll.lib"
  OutputFile=".\\ReleaseDll\zlibwapi.dll"
  LinkIncremental="1"
  SuppressStartupBanner="TRUE"

```

```

        IgnoreAllDefaultLibraries="TRUE"
        ModuleDefinitionFile=".\\zlibvc.def"
        ProgramDatabaseFile=".\\ReleaseDll/zlibwapi.pdb"
        GenerateMapFile="TRUE"
        MapFileName=".\\ReleaseDll/zlibwapi.map"
        SubSystem="2"
        OptimizeForWindows98="1"
        ImportLibrary=".\\ReleaseDll/zlibwapi.lib"/>
    <Tool
        Name="VCMIDLTool"
        PreprocessorDefinitions="NDEBUG"
        MkTypLibCompatible="TRUE"
        SuppressStartupBanner="TRUE"
        TargetEnvironment="1"
        TypeLibraryName=".\\Release/zlibvc.tlb"/>
    <Tool
        Name="VCPostBuildEventTool"/>
    <Tool
        Name="VCPreBuildEventTool"/>
    <Tool
        Name="VCPreLinkEventTool"/>
    <Tool
        Name="VCResourceCompilerTool"
        PreprocessorDefinitions="NDEBUG"
        Culture="1036"/>
    <Tool
        Name="VCWebServiceProxyGeneratorTool"/>
    <Tool
        Name="VCWebDeploymentTool"/>
</Configuration>
</Configurations>
<Files>
    <Filter
        Name="Source Files"
        Filter="cpp;c;cxx;rc;def;r;odl;hpj;bat;for;f90">
        <File
            RelativePath="..\\..\\..\\adler32.c">
        </File>
        <File
            RelativePath="..\\..\\..\\compress.c">
        </File>
        <File
            RelativePath="..\\..\\..\\crc32.c">
        </File>
        <File
            RelativePath="..\\..\\..\\deflate.c">
        </File>
        <File
            RelativePath="..\\..\\..\\masmx86\\gvm32c.c">
            <FileConfiguration
                Name="ReleaseWithoutAsm|Win32"
                ExcludedFromBuild="TRUE">
                <Tool
                    Name="VCCLCompilerTool"/>
            </FileConfiguration>
        </File>
        <File
            RelativePath="..\\..\\..\\gzio.c">
        </File>
        <File
            RelativePath="..\\..\\..\\inffast.c">
        </File>
        <File
            RelativePath="..\\..\\..\\inffast.c">
        </File>
        <File
            RelativePath="..\\..\\..\\inflate.c">
        </File>
        <File
            RelativePath="..\\..\\..\\inftrees.c">
        </File>
        <File
            RelativePath="..\\..\\..\\minizip\\ioapi.c">
        </File>
    </Filter>
</Files>

```

```
<File
    RelativePath="..\..\minizip\iowin32.c">
</File>
<File
    RelativePath="..\..\..\trees.c">
</File>
<File
    RelativePath="..\..\..\uncompr.c">
</File>
<File
    RelativePath="..\..\minizip\unzip.c">
    <FileConfiguration
        Name="Release|Win32">
        <Tool
            Name="VCCLCompilerTool"
            AdditionalIncludeDirectories=""
            PreprocessorDefinitions="ZLIB_INTERNAL"/>
        </FileConfiguration>
</File>
<File
    RelativePath="..\..\minizip\zip.c">
    <FileConfiguration
        Name="Release|Win32">
        <Tool
            Name="VCCLCompilerTool"
            AdditionalIncludeDirectories=""
            PreprocessorDefinitions="ZLIB_INTERNAL"/>
        </FileConfiguration>
</File>
<File
    RelativePath=".\zlib.rc">
</File>
<File
    RelativePath=".\zlibvc.def">
</File>
<File
    RelativePath="..\..\..\zutil.c">
</File>
</Filter>
<Filter
    Name="Header Files"
    Filter="h;hpp;hxx;hm;inl;fi;fd">
    <File
        RelativePath="..\..\..\deflate.h">
    </File>
    <File
        RelativePath="..\..\..\infblock.h">
    </File>
    <File
        RelativePath="..\..\..\infcodes.h">
    </File>
    <File
        RelativePath="..\..\..\inffast.h">
    </File>
    <File
        RelativePath="..\..\..\inftrees.h">
    </File>
    <File
        RelativePath="..\..\..\infutil.h">
    </File>
    <File
        RelativePath="..\..\..\zconf.h">
    </File>
    <File
        RelativePath="..\..\..\zlib.h">
    </File>
    <File
        RelativePath="..\..\..\zutil.h">
    </File>
</Filter>
<Filter
    Name="Resource Files"
    Filter="ico;cur;bmp;dlg;rc2;rct;bin;cnt;rtf;gif;jpg;jpeg;jpe">
</Filter>
```

```
</Files>  
<Globals>  
</Globals>  
</VisualStudioProject>
```

```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="8,00"
  Name="miniunz"
  ProjectGUID="{C52F9E7B-498A-42BE-8DB4-85A15694382A}"
  Keyword="Win32Proj"
>
  <Platforms>
    <Platform
      Name="Win32"
    />
    <Platform
      Name="x64"
    />
    <Platform
      Name="Itanium"
    />
  </Platforms>
  <ToolFiles>
  </ToolFiles>
  <Configurations>
    <Configuration
      Name="Debug|Win32"
      OutputDirectory="x86\MiniUnzip$(ConfigurationName)"
      IntermediateDirectory="x86\MiniUnzip$(ConfigurationName)\Tmp"
      ConfigurationType="1"
      InheritedPropertySheets="UpgradeFromVC70.v.props"
      CharacterSet="2"
    >
      <Tool
        Name="VCPreBuildEventTool"
      />
      <Tool
        Name="VCCustomBuildTool"
      />
      <Tool
        Name="VCXMLDataGeneratorTool"
      />
      <Tool
        Name="VCWebServiceProxyGeneratorTool"
      />
      <Tool
        Name="VCMIDLTool"
      />
      <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\..\minizip"
        PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_DEPRECATED;
_CRT_SECURE_NO_DEPRECATED;ZLIB_WINAPI;_DEBUG;_CONSOLE"
        MinimalRebuild="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="1"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="4"
      />
      <Tool
        Name="VCManagedResourceCompilerTool"
      />
      <Tool
        Name="VCResourceCompilerTool"
      />
      <Tool
        Name="VCPreLinkEventTool"
      />
      <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="x86\ZlibDllDebug\zlibwapi.lib"
        OutputFile="$(OutDir)/miniunz.exe"
      />
    </Configuration>
  </Configurations>
</VisualStudioProject>
```

```

        LinkIncremental="2"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        ProgramDatabaseFile="$(OutDir)/miniunz.pdb"
        SubSystem="1"
        TargetMachine="1"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCAAppVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Debug|x64"
    OutputDirectory="x64\MiniUnzip$(ConfigurationName)"
    IntermediateDirectory="x64\MiniUnzip$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.vsprops"
    CharacterSet="2"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCMIDLTool"
        TargetEnvironment="3"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\..\minizip"
        PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;DEBUG;_CONSOLE;WIN64"
        MinimalRebuild="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="3"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\
        WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"

```



```

        />
        <Tool
            Name="VCResourceCompilerTool "
        />
        <Tool
            Name="VCPreLinkEventTool "
        />
        <Tool
            Name="VCLinkerTool "
            AdditionalDependencies="x64\ZlibDllDebug\zlibwapi.lib"
            OutputFile="$(OutDir)/miniunz.exe"
            LinkIncremental="2"
            GenerateManifest="false"
            GenerateDebugInformation="true"
            ProgramDatabaseFile="$(OutDir)/miniunz.pdb"
            SubSystem="1"
            TargetMachine="17"
        />
        <Tool
            Name="VCALinkTool "
        />
        <Tool
            Name="VCManifestTool "
        />
        <Tool
            Name="VCXDCMakeTool "
        />
        <Tool
            Name="VCBscMakeTool "
        />
        <Tool
            Name="VCFxCopTool "
        />
        <Tool
            Name="VCApVerifierTool "
        />
        <Tool
            Name="VCWebDeploymentTool "
        />
        <Tool
            Name="VCPostBuildEventTool "
        />
    </Configuration>
    <Configuration
        Name="Debug | Itanium"
        OutputDirectory="ia64\MiniUnzip$(ConfigurationName) "
        IntermediatedDirectory="ia64\MiniUnzip$(ConfigurationName)\Tmp "
        ConfigurationType="1"
        InheritedPropertySheets="UpgradeFromVC70.vsprops"
        CharacterSet="2"
    >
        <Tool
            Name="VCPreBuildEventTool "
        />
        <Tool
            Name="VCCustomBuildTool "
        />
        <Tool
            Name="VCXMLDataGeneratorTool "
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool "
        />
        <Tool
            Name="VCIDLTool "
            TargetEnvironment="2"
        />
        <Tool
            Name="VCCLCompilerTool "
            Optimization="0"
            AdditionalIncludeDirectories="..\..\..\..\..\minizip"
            PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;_DEBUG;_CONSOLE;WIN64"
            MinimalRebuild="true"

```

```
        BasicRuntimeChecks="0"
        RuntimeLibrary="3"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\
        WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="ia64\ZlibDllDebug\zlibwapi.lib"
        OutputFile="$(OutDir)/miniunz.exe"
        LinkIncremental="2"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        ProgramDatabaseFile="$(OutDir)/miniunz.pdb"
        SubSystem="1"
        TargetMachine="5"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Release|Win32"
    OutputDirectory="x86\MiniUnzip$(ConfigurationName)"
    IntermediateDirectory="x86\MiniUnzip$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.vsprops"
    CharacterSet="2"
    >
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
```

```
<Tool
    Name="VCMDLTool"
/>
<Tool
    Name="VCCLCompilerTool"
    Optimization="2"
    InlineFunctionExpansion="1"
    OmitFramePointers="true"
    AdditionalIncludeDirectories="..\..\..\..\..\minizip"
    PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_DEPRECATED;
_CRT_SECURE_NO_DEPRECATED;ZLIB_WINAPI;NDEBUG;_CONSOLE"
    StringPooling="true"
    BasicRuntimeChecks="0"
    RuntimeLibrary="0"
    BufferSecurityCheck="false"
    EnableFunctionLevelLinking="true"
    UsePrecompiledHeader="0"
    AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
    Detect64BitPortabilityProblems="true"
    DebugInformationFormat="3"
/>
<Tool
    Name="VCManagedResourceCompilerTool"
/>
<Tool
    Name="VCResourceCompilerTool"
/>
<Tool
    Name="VCPreLinkEventTool"
/>
<Tool
    Name="VCLinkerTool"
    AdditionalDependencies="x86\ZlibDllRelease\zlibwapi.lib"
    OutputFile="$(OutDir)/miniunz.exe"
    LinkIncremental="1"
    GenerateManifest="false"
    GenerateDebugInformation="true"
    SubSystem="1"
    OptimizeReferences="2"
    EnableCOMDATFolding="2"
    OptimizeForWindows98="1"
    TargetMachine="1"
/>
<Tool
    Name="VCALinkTool"
/>
<Tool
    Name="VCManifestTool"
/>
<Tool
    Name="VCXDCMakeTool"
/>
<Tool
    Name="VCBscMakeTool"
/>
<Tool
    Name="VCFxCopTool"
/>
<Tool
    Name="VCApVerifierTool"
/>
<Tool
    Name="VCWebDeploymentTool"
/>
<Tool
    Name="VCPostBuildEventTool"
/>
</Configuration>
<Configuration
    Name="Release|x64"
    OutputDirectory="x64\MiniUnzip$(ConfigurationName)"
    IntermediateDirectory="x64\MiniUnzip$(ConfigurationName)\Tmp"
    ConfigurationType="1"
```

```
InheritedPropertySheets="UpgradeFromVC70.vsprops"
CharacterSet="2"
>
<Tool
    Name="VCPreBuildEventTool"
/>
<Tool
    Name="VCCustomBuildTool"
/>
<Tool
    Name="VCXMLDataGeneratorTool"
/>
<Tool
    Name="VCWebServiceProxyGeneratorTool"
/>
<Tool
    Name="VCIDLTool"
    TargetEnvironment="3"
/>
<Tool
    Name="VCCLCompilerTool"
    Optimization="2"
    InlineFunctionExpansion="1"
    OmitFramePointers="true"
    AdditionalIncludeDirectories="..\..\..\..\..\minizip"
    PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;NDEBUG;_CONSOLE;WIN64"
    StringPooling="true"
    BasicRuntimeChecks="0"
    RuntimeLibrary="2"
    BufferSecurityCheck="false"
    EnableFunctionLevelLinking="true"
    UsePrecompiledHeader="0"
    AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
    Detect64BitPortabilityProblems="true"
    DebugInformationFormat="3"
/>
<Tool
    Name="VCManagedResourceCompilerTool"
/>
<Tool
    Name="VCResourceCompilerTool"
/>
<Tool
    Name="VCPreLinkEventTool"
/>
<Tool
    Name="VCLinkerTool"
    AdditionalDependencies="x64\ZlibDllRelease\zlibwapi.lib"
    OutputFile="$(OutDir)/miniunz.exe"
    LinkIncremental="1"
    GenerateManifest="false"
    GenerateDebugInformation="true"
    SubSystem="1"
    OptimizeReferences="2"
    EnableCOMDATFolding="2"
    OptimizeForWindows98="1"
    TargetMachine="17"
/>
<Tool
    Name="VCALinkTool"
/>
<Tool
    Name="VCManifestTool"
/>
<Tool
    Name="VCXDCMakeTool"
/>
<Tool
    Name="VCBscMakeTool"
/>
<Tool
    Name="VCFxCopTool"
```

```

        />
        <Tool
            Name="VCApVerifierTool "
        />
        <Tool
            Name="VCWebDeploymentTool "
        />
        <Tool
            Name="VCPstBuildEventTool "
        />
    </Configuration>
    <Configuration
        Name="Release|Itanium"
        OutputDirectory="ia64\MiniUnzip$(ConfigurationName)"
        IntermediateDirectory="ia64\MiniUnzip$(ConfigurationName)\Tmp"
        ConfigurationType="1"
        InheritedPropertySheets="UpgradeFromVC70.vsprops"
        CharacterSet="2"
    >
        <Tool
            Name="VCPreBuildEventTool "
        />
        <Tool
            Name="VCCustomBuildTool "
        />
        <Tool
            Name="VCXMLDataGeneratorTool "
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool "
        />
        <Tool
            Name="VCMIDLTool "
            TargetEnvironment="2"
        />
        <Tool
            Name="VCCLCompilerTool "
            Optimization="2"
            InlineFunctionExpansion="1"
            OmitFramePointers="true"
            AdditionalIncludeDirectories="..\..\..\..\..\minizip"
            PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;NDEBUG;_CONSOLE;WIN64"
            StringPooling="true"
            BasicRuntimeChecks="0"
            RuntimeLibrary="2"
            BufferSecurityCheck="false"
            EnableFunctionLevelLinking="true"
            UsePrecompiledHeader="0"
            AssemblerListingLocation="$(IntDir)\
            WarningLevel="3"
            Detect64BitPortabilityProblems="true"
            DebugInformationFormat="3"
        />
        <Tool
            Name="VCManagedResourceCompilerTool "
        />
        <Tool
            Name="VCResourceCompilerTool "
        />
        <Tool
            Name="VCPreLinkEventTool "
        />
        <Tool
            Name="VCLinkerTool "
            AdditionalDependencies="ia64\ZlibDllRelease\zlibwapi.lib"
            OutputFile="$(OutDir)/miniunz.exe"
            LinkIncremental="1"
            GenerateManifest="false"
            GenerateDebugInformation="true"
            SubSystem="1"
            OptimizeReferences="2"
            EnableCOMDATFolding="2"
            OptimizeForWindows98="1"

```

```

        TargetMachine="5"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
</Configurations>
<References>
</References>
<Files>
    <Filter
        Name="Source Files"
        Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm"
    >
        <File
            RelativePath="..\..\minizip\miniunz.c"
        >
        </File>
    </Filter>
    <Filter
        Name="Header Files"
        Filter="h;hpp;hxx;hm;inl;inc"
    >
    </Filter>
    <Filter
        Name="Resource Files"
        Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe"
    >
    </Filter>
</Files>
<Globals>
</Globals>
</VisualStudioProject>
```

```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="8,00"
  Name="minizip"
  ProjectGUID="{48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}"
  Keyword="Win32Proj"
>
  <Platforms>
    <Platform
      Name="Win32"
    />
    <Platform
      Name="x64"
    />
    <Platform
      Name="Itanium"
    />
  </Platforms>
  <ToolFiles>
  </ToolFiles>
  <Configurations>
    <Configuration
      Name="Debug|Win32"
      OutputDirectory="x86\MiniZip$(ConfigurationName)"
      IntermediateDirectory="x86\MiniZip$(ConfigurationName)\Tmp"
      ConfigurationType="1"
      InheritedPropertySheets="UpgradeFromVC70.v.props"
      CharacterSet="2"
    >
      <Tool
        Name="VCPreBuildEventTool"
      />
      <Tool
        Name="VCCustomBuildTool"
      />
      <Tool
        Name="VCXMLDataGeneratorTool"
      />
      <Tool
        Name="VCWebServiceProxyGeneratorTool"
      />
      <Tool
        Name="VCIDLTool"
      />
      <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\..\minizip"
        PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_DEPRECATED;
_CRT_SECURE_NO_DEPRECATED;ZLIB_WINAPI;_DEBUG;_CONSOLE"
        MinimalRebuild="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="1"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="4"
      />
      <Tool
        Name="VCManagedResourceCompilerTool"
      />
      <Tool
        Name="VCResourceCompilerTool"
      />
      <Tool
        Name="VCPreLinkEventTool"
      />
      <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="x86\ZlibDllDebug\zlibwapi.lib"
        OutputFile="$(OutDir)/minizip.exe"
      />
    </Configuration>
  </Configurations>
</VisualStudioProject>
```

```
        LinkIncremental="2"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        ProgramDatabaseFile="$(OutDir)/minizip.pdb"
        SubSystem="1"
        TargetMachine="1"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApplibTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Debug|x64"
    OutputDirectory="x64\\$(ConfigurationName)"
    IntermediateDirectory="x64\\$(ConfigurationName)"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.vsprops"
    CharacterSet="2"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCIDLTool"
        TargetEnvironment="3"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\minizip"
        PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;DEBUG;_CONSOLE;WIN64"
        MinimalRebuild="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="3"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
```



```

    />
    <Tool
        Name="VCResourceCompilerTool"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="x64\ZlibDllDebug\zlibwapi.lib"
        OutputFile="$(OutDir)/minizip.exe"
        LinkIncremental="2"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        ProgramDatabaseFile="$(OutDir)/minizip.pdb"
        SubSystem="1"
        TargetMachine="17"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Debug | Itanium"
    OutputDirectory="ia64\$(ConfigurationName)"
    IntermediateDirectory="ia64\$(ConfigurationName)"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.vsprops"
    CharacterSet="2"
    >
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCIDLTool"
        TargetEnvironment="2"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\minizip"
        PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;_DEBUG;_CONSOLE;WIN64"
        MinimalRebuild="true"

```

```
        BasicRuntimeChecks="0"
        RuntimeLibrary="3"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\ "
        WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="ia64\ZlibDllDebug\zlibwapi.lib"
        OutputFile="$(OutDir)/minizip.exe"
        LinkIncremental="2"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        ProgramDatabaseFile="$(OutDir)/minizip.pdb"
        SubSystem="1"
        TargetMachine="5"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Release|Win32"
    OutputDirectory="x86\MiniZip$(ConfigurationName)"
    IntermediateDirectory="x86\MiniZip$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.vsprops"
    CharacterSet="2"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
```

```
<Tool
    Name="VCMIDLTool"
/>
<Tool
    Name="VCCLCompilerTool"
    Optimization="2"
    InlineFunctionExpansion="1"
    OmitFramePointers="true"
    AdditionalIncludeDirectories="..\..\..\..\minizip"
    PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_DEPRECATED;
_CRT_SECURE_NO_DEPRECATED;ZLIB_WINAPI;NDEBUG;_CONSOLE"
    StringPooling="true"
    BasicRuntimeChecks="0"
    RuntimeLibrary="0"
    BufferSecurityCheck="false"
    EnableFunctionLevelLinking="true"
    UsePrecompiledHeader="0"
    AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
    Detect64BitPortabilityProblems="true"
    DebugInformationFormat="3"
/>
<Tool
    Name="VCManagedResourceCompilerTool"
/>
<Tool
    Name="VCResourceCompilerTool"
/>
<Tool
    Name="VCPreLinkEventTool"
/>
<Tool
    Name="VCLinkerTool"
    AdditionalDependencies="x86\ZlibDllRelease\zlibwapi.lib"
    OutputFile="$(OutDir)/minizip.exe"
    LinkIncremental="1"
    GenerateDebugInformation="true"
    SubSystem="1"
    OptimizeReferences="2"
    EnableCOMDATFolding="2"
    OptimizeForWindows98="1"
    TargetMachine="1"
/>
<Tool
    Name="VCALinkTool"
/>
<Tool
    Name="VCManifestTool"
/>
<Tool
    Name="VCXDCMakeTool"
/>
<Tool
    Name="VCBscMakeTool"
/>
<Tool
    Name="VCFxCopTool"
/>
<Tool
    Name="VCApVerifierTool"
/>
<Tool
    Name="VCWebDeploymentTool"
/>
<Tool
    Name="VCPostBuildEventTool"
/>
</Configuration>
<Configuration
    Name="Release|x64"
    OutputDirectory="x64\$(ConfigurationName)"
    IntermediateDirectory="x64\$(ConfigurationName)"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.v.props"
/>
```

```
CharacterSet="2"
>
<Tool
    Name="VCPreBuildEventTool"
/>
<Tool
    Name="VCCustomBuildTool"
/>
<Tool
    Name="VCXMLDataGeneratorTool"
/>
<Tool
    Name="VCWebServiceProxyGeneratorTool"
/>
<Tool
    Name="VCMIDLTool"
    TargetEnvironment="3"
/>
<Tool
    Name="VCCLCompilerTool"
    Optimization="2"
    InlineFunctionExpansion="1"
    OmitFramePointers="true"
    AdditionalIncludeDirectories="..\..\..\..\minizip"
    PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;NDEBUG;_CONSOLE;WIN64"
    StringPooling="true"
    BasicRuntimeChecks="0"
    RuntimeLibrary="2"
    BufferSecurityCheck="false"
    EnableFunctionLevelLinking="true"
    UsePrecompiledHeader="0"
    AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
    Detect64BitPortabilityProblems="true"
    DebugInformationFormat="3"
/>
<Tool
    Name="VCManagedResourceCompilerTool"
/>
<Tool
    Name="VCResourceCompilerTool"
/>
<Tool
    Name="VCPreLinkEventTool"
/>
<Tool
    Name="VCLinkerTool"
    AdditionalDependencies="x64\ZlibDllRelease\zlibwapi.lib"
    OutputFile="$(OutDir)/minizip.exe"
    LinkIncremental="1"
    GenerateDebugInformation="true"
    SubSystem="1"
    OptimizeReferences="2"
    EnableCOMDATFolding="2"
    OptimizeForWindows98="1"
    TargetMachine="17"
/>
<Tool
    Name="VCALinkTool"
/>
<Tool
    Name="VCManifestTool"
/>
<Tool
    Name="VCXDCMakeTool"
/>
<Tool
    Name="VCBscMakeTool"
/>
<Tool
    Name="VCFxCopTool"
/>
<Tool
```

```
        Name="VCApplibTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Release|Itanium"
    OutputDirectory="$(ConfigurationName)"
    IntermediateDirectory="$(ConfigurationName)"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.vprops"
    CharacterSet="2"
>
    <Tool
        Name="VCPPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCIDLTool"
        TargetEnvironment="2"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="2"
        InlineFunctionExpansion="1"
        OmitFramePointers="true"
        AdditionalIncludeDirectories="..\..\..\..\..\minizip"
        PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;NDEBUG;_CONSOLE;WIN64"
        StringPooling="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="2"
        BufferSecurityCheck="false"
        EnableFunctionLevelLinking="true"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\
        WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
    />
    <Tool
        Name="VCPPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="ia64\ZlibDllRelease\zlibwapi.lib"
        OutputFile="$(OutDir)/minizip.exe"
        LinkIncremental="1"
        GenerateDebugInformation="true"
        SubSystem="1"
        OptimizeReferences="2"
        EnableCOMDATFolding="2"
        OptimizeForWindows98="1"
        TargetMachine="5"
    />
    <Tool
```

```

        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
</Configurations>
<References>
</References>
<Files>
    <Filter
        Name="Source Files"
        Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm"
    >
        <File
            RelativePath="..\..\minizip\minizip.c"
        >
        </File>
    </Filter>
    <Filter
        Name="Header Files"
        Filter="h;hpp;hxx;hm;inl;inc"
    >
    </Filter>
    <Filter
        Name="Resource Files"
        Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe"
    >
    </Filter>
</Files>
<Globals>
</Globals>
</VisualStudioProject>
```

```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="8,00"
  Name="testzlib"
  ProjectGUID="{AA6666AA-E09F-4135-9C0C-4FE50C3C654B}"
  RootNamespace="testzlib"
  Keyword="Win32Proj"
  >
  <Platforms>
    <Platform
      Name="Win32"
    />
    <Platform
      Name="x64"
    />
    <Platform
      Name="Itanium"
    />
  </Platforms>
  <ToolFiles>
  </ToolFiles>
  <Configurations>
    <Configuration
      Name="Debug|Win32"
      OutputDirectory="x86\TestZlib$(ConfigurationName)"
      IntermediateDirectory="x86\TestZlib$(ConfigurationName)\Tmp"
      ConfigurationType="1"
      CharacterSet="2"
    >
      <Tool
        Name="VCPreBuildEventTool"
      />
      <Tool
        Name="VCCustomBuildTool"
      />
      <Tool
        Name="VCXMLDataGeneratorTool"
      />
      <Tool
        Name="VCWebServiceProxyGeneratorTool"
      />
      <Tool
        Name="VCMIDLTool"
      />
      <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\.."
        PreprocessorDefinitions="ASMV;ASMINF;WIN32;ZLIB_WINAPI;_D
EBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRECATED;_CRT_SECURE_NO_DEPRECATED"
        MinimalRebuild="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="1"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerOutput="4"
        AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="4"
      />
      <Tool
        Name="VCManagedResourceCompilerTool"
      />
      <Tool
        Name="VCResourceCompilerTool"
      />
      <Tool
        Name="VCPreLinkEventTool"
      />
      <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="..\..\masmx86\gvm32.obj ....\m
```

asmx86\inffas32.obj"

```
OutputFile="$(OutDir)/testzlib.exe"
LinkIncremental="2"
GenerateManifest="false"
GenerateDebugInformation="true"
ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
SubSystem="1"
TargetMachine="1"
```

```
</>
```

```
<Tool
```

```
  Name="VCALinkTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCManifestTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCXDCMakeTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCBscMakeTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCFxCopTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCApVerifierTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCWebDeploymentTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCPostBuildEventTool"
```

```
</>
```

```
</Configuration>
```

```
<Configuration
```

```
  Name="Debug|x64"
```

```
  OutputDirectory="x64\TestZlib$(ConfigurationName)"
```

```
  IntermediateDirectory="x64\TestZlib$(ConfigurationName)\Tmp"
```

```
  ConfigurationType="1"
```

```
>
```

```
<Tool
```

```
  Name="VCPreBuildEventTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCCustomBuildTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCXMLDataGeneratorTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCWebServiceProxyGeneratorTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCIDLTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCCLCompilerTool"
```

```
  AdditionalIncludeDirectories="..\..\\"
```

```
  PreprocessorDefinitions="ASMV;ASMINF;WIN32;ZLIB_WINAPI;_D
```

```
EBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRECATED;_CRT_SECURE_NO_DEPRECATED"
```

```
  BasicRuntimeChecks="0"
```

```
  RuntimeLibrary="3"
```

```
  BufferSecurityCheck="false"
```

```
  AssemblerListingLocation="$(IntDir)\"
```

```
</>
```

```
<Tool
```

```
  Name="VCManagedResourceCompilerTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCResourceCompilerTool"
```

```
</>
```

```
<Tool
```

```
  Name="VCPreLinkEventTool"
```

```
</>
```



```

asmx64\inffasx64.obj"
    <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="..\..\masmx64\gvmat64.obj ....\m
        GenerateManifest="false"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Debug | Itanium"
    OutputDirectory="ia64\TestZlib$(ConfigurationName)"
    IntermediateDirectory="ia64\TestZlib$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    CharacterSet="2"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCIDLTool"
        TargetEnvironment="2"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\.."
        PreprocessorDefinitions="ZLIB_WINAPI;_DEBUG;_CONSOLE;_CRT
_NONSTDC_NO_DEPRECATED;_CRT_SECURE_NO_DEPRECATED;WIN64"
        MinimalRebuild="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="3"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerOutput="4"
        AssemblerListingLocation="$(IntDir)\
        WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />

```

```

        <Tool
            Name="VCResourceCompilerTool"
        />
        <Tool
            Name="VCPreLinkEventTool"
        />
        <Tool
            Name="VCLinkerTool"
            OutputFile="$(OutDir)/testzlib.exe"
            LinkIncremental="2"
            GenerateManifest="false"
            GenerateDebugInformation="true"
            ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
            SubSystem="1"
            TargetMachine="5"
        />
        <Tool
            Name="VCALinkTool"
        />
        <Tool
            Name="VCManifestTool"
        />
        <Tool
            Name="VCXDCMakeTool"
        />
        <Tool
            Name="VCBscMakeTool"
        />
        <Tool
            Name="VCFxCopTool"
        />
        <Tool
            Name="VCApVerifierTool"
        />
        <Tool
            Name="VCWebDeploymentTool"
        />
        <Tool
            Name="VCPostBuildEventTool"
        />
    </Configuration>
    <Configuration
        Name="ReleaseWithoutAsm|Win32"
        OutputDirectory="x86\TestZlib$(ConfigurationName)"
        IntermediateDirectory="x86\TestZlib$(ConfigurationName)\Tmp"
        ConfigurationType="1"
        CharacterSet="2"
        WholeProgramOptimization="1"
    >
        <Tool
            Name="VCPreBuildEventTool"
        />
        <Tool
            Name="VCCustomBuildTool"
        />
        <Tool
            Name="VCXMLDataGeneratorTool"
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool"
        />
        <Tool
            Name="VCIDLTool"
        />
        <Tool
            Name="VCCLCompilerTool"
            Optimization="2"
            InlineFunctionExpansion="1"
            OmitFramePointers="true"
            AdditionalIncludeDirectories="..\..\.."
            PreprocessorDefinitions="WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRECATED;_CRT_SECURE_NO_DEPRECATED"
            StringPooling="true"
            BasicRuntimeChecks="0"

```

```
RuntimeLibrary="0"
BufferSecurityCheck="false"
EnableFunctionLevelLinking="true"
UsePrecompiledHeader="0"
AssemblerListingLocation="$(IntDir)\ "
WarningLevel="3"
Detect64BitPortabilityProblems="true"
DebugInformationFormat="3"
/>
<Tool
  Name="VCManagedResourceCompilerTool"
/>
<Tool
  Name="VCResourceCompilerTool"
/>
<Tool
  Name="VCPreLinkEventTool"
/>
<Tool
  Name="VCLinkerTool"
  OutputFile="$(OutDir)/testzlib.exe"
  LinkIncremental="1"
  GenerateManifest="false"
  GenerateDebugInformation="true"
  SubSystem="1"
  OptimizeReferences="2"
  EnableCOMDATFolding="2"
  OptimizeForWindows98="1"
  TargetMachine="1"
/>
<Tool
  Name="VCALinkTool"
/>
<Tool
  Name="VCManifestTool"
/>
<Tool
  Name="VCXDCMakeTool"
/>
<Tool
  Name="VCBscMakeTool"
/>
<Tool
  Name="VCFxCopTool"
/>
<Tool
  Name="VCApVerifierTool"
/>
<Tool
  Name="VCWebDeploymentTool"
/>
<Tool
  Name="VCPostBuildEventTool"
/>
</Configuration>
<Configuration
  Name="ReleaseWithoutAsm|x64"
  OutputDirectory="x64\TestZlib$(ConfigurationName)"
  IntermediateDirectory="x64\TestZlib$(ConfigurationName)\Tmp"
  ConfigurationType="1"
  WholeProgramOptimization="1"
>
<Tool
  Name="VCPreBuildEventTool"
/>
<Tool
  Name="VCCustomBuildTool"
/>
<Tool
  Name="VCXMLDataGeneratorTool"
/>
<Tool
  Name="VCWebServiceProxyGeneratorTool"
/>
```

```

        <Tool
            Name="VCIDLTool "
        />
        <Tool
            Name="VCCLCompilerTool "
            AdditionalIncludeDirectories="..\..\.."
            PreprocessorDefinitions="WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRECATED;_CRT_SECURE_NO_DEPRECATED"
            BasicRuntimeChecks="0"
            RuntimeLibrary="2"
            BufferSecurityCheck="false"
            AssemblerListingLocation="$(IntDir)\ "
        />
        <Tool
            Name="VCManagedResourceCompilerTool "
        />
        <Tool
            Name="VCResourceCompilerTool "
        />
        <Tool
            Name="VCPreLinkEventTool "
        />
        <Tool
            Name="VCLinkerTool "
            AdditionalDependencies=""
            GenerateManifest="false"
        />
        <Tool
            Name="VCALinkTool "
        />
        <Tool
            Name="VCManifestTool "
        />
        <Tool
            Name="VCXDCMakeTool "
        />
        <Tool
            Name="VCBscMakeTool "
        />
        <Tool
            Name="VCFxCopTool "
        />
        <Tool
            Name="VCApVerifierTool "
        />
        <Tool
            Name="VCWebDeploymentTool "
        />
        <Tool
            Name="VCPostBuildEventTool "
        />
    </Configuration>
    <Configuration
        Name="ReleaseWithoutAsm|Itanium"
        OutputDirectory="ia64\TestZlib$(ConfigurationName)"
        IntermediateDirectory="ia64\TestZlib$(ConfigurationName)\Tmp"
        ConfigurationType="1"
        CharacterSet="2"
        WholeProgramOptimization="1"
    >
        <Tool
            Name="VCPreBuildEventTool "
        />
        <Tool
            Name="VCCustomBuildTool "
        />
        <Tool
            Name="VCXMLDataGeneratorTool "
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool "
        />
        <Tool
            Name="VCIDLTool "

```

```

        TargetEnvironment="2"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="2"
        InlineFunctionExpansion="1"
        OmitFramePointers="true"
        AdditionalIncludeDirectories="..\..\.."
        PreprocessorDefinitions="ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT
_NONSTDC_NO_DEPRECATED;_CRT_SECURE_NO_DEPRECATED;WIN64"
        StringPooling="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="2"
        BufferSecurityCheck="false"
        EnableFunctionLevelLinking="true"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\ "
        WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        OutputFile="$(OutDir)/testzlib.exe"
        LinkIncremental="1"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        SubSystem="1"
        OptimizeReferences="2"
        EnableCOMDATFolding="2"
        OptimizeForWindows98="1"
        TargetMachine="5"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Release|Win32"
    OutputDirectory="x86\TestZlib$(ConfigurationName)"
    IntermediateDirectory="x86\TestZlib$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    CharacterSet="2"
    WholeProgramOptimization="1"

```

```
>
<Tool
    Name="VCPreBuildEventTool"
/>
<Tool
    Name="VCCustomBuildTool"
/>
<Tool
    Name="VCXMLDataGeneratorTool"
/>
<Tool
    Name="VCWebServiceProxyGeneratorTool"
/>
<Tool
    Name="VCIDLTool"
/>
<Tool
    Name="VCCLCompilerTool"
    Optimization="2"
    InlineFunctionExpansion="1"
    OmitFramePointers="true"
    AdditionalIncludeDirectories="..\..\.."
    PreprocessorDefinitions="ASMV;ASMINF;WIN32;ZLIB_WINAPI;ND
EBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRECATED;_CRT_SECURE_NO_DEPRECATED"
    StringPooling="true"
    BasicRuntimeChecks="0"
    RuntimeLibrary="0"
    BufferSecurityCheck="false"
    EnableFunctionLevelLinking="true"
    UsePrecompiledHeader="0"
    AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
    Detect64BitPortabilityProblems="true"
    DebugInformationFormat="3"
/>
<Tool
    Name="VCManagedResourceCompilerTool"
/>
<Tool
    Name="VCResourceCompilerTool"
/>
<Tool
    Name="VCPreLinkEventTool"
/>
<Tool
    Name="VCLinkerTool"
    AdditionalDependencies="..\..\masmx86\gvm32.obj ....\m
asmx86\inffas32.obj"
    OutputFile="$(OutDir)/testzlib.exe"
    LinkIncremental="1"
    GenerateManifest="false"
    GenerateDebugInformation="true"
    SubSystem="1"
    OptimizeReferences="2"
    EnableCOMDATFolding="2"
    OptimizeForWindows98="1"
    TargetMachine="1"
/>
<Tool
    Name="VCALinkTool"
/>
<Tool
    Name="VCManifestTool"
/>
<Tool
    Name="VCXDCMakeTool"
/>
<Tool
    Name="VCBscMakeTool"
/>
<Tool
    Name="VCFxCopTool"
/>
<Tool
```

```

        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Release|x64"
    OutputDirectory="x64\TestZlib$(ConfigurationName)"
    IntermediateDirectory="x64\TestZlib$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    WholeProgramOptimization="1"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCIDLTool"
    />
    <Tool
        Name="VCCLCompilerTool"
        AdditionalIncludeDirectories="..\..\.."
        PreprocessorDefinitions="ASMV;ASMINF;WIN32;ZLIB_WINAPI;ND
EBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRECATED;_CRT_SECURE_NO_DEPRECATED"
        BasicRuntimeChecks="0"
        RuntimeLibrary="2"
        BufferSecurityCheck="false"
        AssemblerListingLocation="$(IntDir)\\"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="..\..\masmx64\gvm64.obj ....\m
asmx64\inffasx64.obj"
        GenerateManifest="false"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />

```

```
<Tool
    Name="VCWebDeploymentTool"
/>
<Tool
    Name="VCPostBuildEventTool"
/>
</Configuration>
<Configuration
    Name="Release|Itanium"
    OutputDirectory="ia64\TestZlib$(ConfigurationName)"
    IntermediateDirectory="ia64\TestZlib$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    CharacterSet="2"
    WholeProgramOptimization="1"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCMIDLTool"
        TargetEnvironment="2"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="2"
        InlineFunctionExpansion="1"
        OmitFramePointers="true"
        AdditionalIncludeDirectories="..\..\.."
        PreprocessorDefinitions="ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT
_NONSTDC_NO_DEPRECATED;_CRT_SECURE_NO_DEPRECATED;WIN64"
        StringPooling="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="2"
        BufferSecurityCheck="false"
        EnableFunctionLevelLinking="true"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        OutputFile="$(OutDir)/testzlib.exe"
        LinkIncremental="1"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        SubSystem="1"
        OptimizeReferences="2"
        EnableCOMDATFolding="2"
        OptimizeForWindows98="1"
        TargetMachine="5"
    />
    <Tool
        Name="VCALinkTool"
    />
```



```
<Tool
    Name="VCManifestTool"
/>
<Tool
    Name="VCXDCMakeTool"
/>
<Tool
    Name="VCBscMakeTool"
/>
<Tool
    Name="VCFxCopTool"
/>
<Tool
    Name="VCApVerifierTool"
/>
<Tool
    Name="VCWebDeploymentTool"
/>
<Tool
    Name="VCPostBuildEventTool"
/>
</Configuration>
</Configurations>
<References>
</References>
<Files>
    <Filter
        Name="Source Files"
        Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm"
    >
        <File
            RelativePath="..\..\..\adler32.c"
        >
        </File>
        <File
            RelativePath="..\..\..\compress.c"
        >
        </File>
        <File
            RelativePath="..\..\..\crc32.c"
        >
        </File>
        <File
            RelativePath="..\..\..\deflate.c"
        >
        </File>
        <File
            RelativePath="..\..\..\masmx86\gvm32c.c"
        >
            <FileConfiguration
                Name="Debug|x64"
                ExcludedFromBuild="true"
            >
                <Tool
                    Name="VCCLCompilerTool"
                />
            </FileConfiguration>
            <FileConfiguration
                Name="Debug|Itanium"
                ExcludedFromBuild="true"
            >
                <Tool
                    Name="VCCLCompilerTool"
                />
            </FileConfiguration>
            <FileConfiguration
                Name="ReleaseWithoutAsm|x64"
                ExcludedFromBuild="true"
            >
                <Tool
                    Name="VCCLCompilerTool"
                />
            </FileConfiguration>
        </FileConfiguration>
    </Filter>
</Files>
```

```
        Name="ReleaseWithoutAsm|Itanium"
        ExcludedFromBuild="true"
    >
    <Tool
        Name="VCCLCompilerTool"
    />
</FileConfiguration>
<FileConfiguration
    Name="Release|x64"
    ExcludedFromBuild="true"
>
    <Tool
        Name="VCCLCompilerTool"
    />
</FileConfiguration>
<FileConfiguration
    Name="Release|Itanium"
    ExcludedFromBuild="true"
>
    <Tool
        Name="VCCLCompilerTool"
    />
</FileConfiguration>
<FileConfiguration
    Name="Debug|Win64 (AMD64)"
    ExcludedFromBuild="TRUE"
>
    <Tool
        Name="VCCLCompilerTool"
    />
</FileConfiguration>
<FileConfiguration
    Name="Release|Win64 (AMD64)"
    ExcludedFromBuild="TRUE"
>
    <Tool
        Name="VCCLCompilerTool"
    />
</FileConfiguration>
<FileConfiguration
    Name="ReleaseAsm|Win64 (AMD64)"
    ExcludedFromBuild="TRUE"
>
    <Tool
        Name="VCCLCompilerTool"
    />
</FileConfiguration>
</File>
<File
    RelativePath="../../..\infback.c"
>
</File>
<File
    RelativePath="../../..\masmx64\inffas8664.c"
>
    <FileConfiguration
        Name="Debug|Win32"
        ExcludedFromBuild="true"
    >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
    <FileConfiguration
        Name="Debug|Itanium"
        ExcludedFromBuild="true"
    >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
    <FileConfiguration
        Name="ReleaseWithoutAsm|Win32"
        ExcludedFromBuild="true"
```

```
>
<Tool
    Name="VCCLCompilerTool"
/>
</FileConfiguration>
<FileConfiguration
    Name="ReleaseWithoutAsm|Itanium"
    ExcludedFromBuild="true"
>
<Tool
    Name="VCCLCompilerTool"
/>
</FileConfiguration>
<FileConfiguration
    Name="Release|Win32"
    ExcludedFromBuild="true"
>
<Tool
    Name="VCCLCompilerTool"
/>
</FileConfiguration>
<FileConfiguration
    Name="Release|Itanium"
    ExcludedFromBuild="true"
>
<Tool
    Name="VCCLCompilerTool"
/>
</FileConfiguration>
</File>
<File
    RelativePath="..\..\..\inffast.c"
>
</File>
<File
    RelativePath="..\..\..\inflate.c"
>
</File>
<File
    RelativePath="..\..\..\inftrees.c"
>
</File>
<File
    RelativePath="..\..\testzlib\testzlib.c"
>
</File>
<File
    RelativePath="..\..\..\trees.c"
>
</File>
<File
    RelativePath="..\..\..\uncompr.c"
>
</File>
<File
    RelativePath="..\..\..\zutil.c"
>
</File>
</Filter>
<Filter
    Name="Header Files"
    Filter="h;hpp;hxx;hm;inl;inc"
>
</Filter>
<Filter
    Name="Resource Files"
    Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe"
>
</Filter>
</Files>
<Globals>
</Globals>
</VisualStudioProject>
```

```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="8,00"
  Name="TestZlibDll"
  ProjectGUID="{C52F9E7B-498A-42BE-8DB4-85A15694366A}"
  Keyword="Win32Proj"
  SignManifests="true"
>
  <Platforms>
    <Platform
      Name="Win32"
    />
    <Platform
      Name="x64"
    />
    <Platform
      Name="Itanium"
    />
  </Platforms>
  <ToolFiles>
  </ToolFiles>
  <Configurations>
    <Configuration
      Name="Debug|Win32"
      OutputDirectory="x86\TestZlibDll$(ConfigurationName)"
      IntermediateDirectory="x86\TestZlibDll$(ConfigurationName)\Tmp"
      ConfigurationType="1"
      InheritedPropertySheets="UpgradeFromVC70.vsprops"
      CharacterSet="2"
    >
      <Tool
        Name="VCPreBuildEventTool"
      />
      <Tool
        Name="VCCustomBuildTool"
      />
      <Tool
        Name="VCXMLDataGeneratorTool"
      />
      <Tool
        Name="VCWebServiceProxyGeneratorTool"
      />
      <Tool
        Name="VCIDLTool"
      />
      <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\..\minizip"
        PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_DEPRECATED;
_CRT_SECURE_NO_DEPRECATED;ZLIB_WINAPI;_DEBUG;_CONSOLE"
        MinimalRebuild="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="1"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="4"
      />
      <Tool
        Name="VCManagedResourceCompilerTool"
      />
      <Tool
        Name="VCResourceCompilerTool"
      />
      <Tool
        Name="VCPreLinkEventTool"
      />
      <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="x86\ZlibDllDebug\zlibwapi.lib"
      />
    </Configuration>
  </Configurations>
</VisualStudioProject>
```

```

        OutputFile="$(OutDir)/testzlib.exe"
        LinkIncremental="2"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
        SubSystem="1"
        TargetMachine="1"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Debug|x64"
    OutputDirectory="x64\TestZlibDll$(ConfigurationName)"
    IntermediateDirectory="x64\TestZlibDll$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.v.props"
    CharacterSet="2"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCMIDLTool"
        TargetEnvironment="3"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\minizip"
        PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;_DEBUG;_CONSOLE;WIN64"
        MinimalRebuild="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="3"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool

```

```
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="x64\ZlibDllDebug\zlibwapi.lib"
        OutputFile="$(OutDir)/testzlib.exe"
        LinkIncremental="2"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
        SubSystem="1"
        TargetMachine="17"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Debug|Itanium"
    OutputDirectory="ia64\TestZlibDll$(ConfigurationName)"
    IntermediateDirectory="ia64\TestZlibDll$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.vsprops"
    CharacterSet="2"
    >
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCIDLTool"
        TargetEnvironment="2"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\..\minizip"
        PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;_DEBUG;_CONSOLE;WIN64"
```

```
        MinimalRebuild="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="3"
        BufferSecurityCheck="false"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\ "
        WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="ia64\ZlibDllDebug\zlibwapi.lib"
        OutputFile="$(OutDir)/testzlib.exe"
        LinkIncremental="2"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
        SubSystem="1"
        TargetMachine="5"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Release|Win32"
    OutputDirectory="x86\TestZlibDll$(ConfigurationName)"
    IntermediateDirectory="x86\TestZlibDll$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.vsprops"
    CharacterSet="2"
    >
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
```

```
</>
<Tool
    Name="VCMidlTool"
/>
<Tool
    Name="VCCLCompilerTool"
    Optimization="2"
    InlineFunctionExpansion="1"
    OmitFramePointers="true"
    AdditionalIncludeDirectories="..\..\..\..\minizip"
    PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_DEPRECATED;
_CRT_SECURE_NO_DEPRECATED;ZLIB_WINAPI;NDEBUG;_CONSOLE"
    StringPooling="true"
    BasicRuntimeChecks="0"
    RuntimeLibrary="0"
    BufferSecurityCheck="false"
    EnableFunctionLevelLinking="true"
    UsePrecompiledHeader="0"
    AssemblerListingLocation="$(IntDir)\
    WarningLevel="3"
    Detect64BitPortabilityProblems="true"
    DebugInformationFormat="3"
/>
<Tool
    Name="VCManagedResourceCompilerTool"
/>
<Tool
    Name="VCResourceCompilerTool"
/>
<Tool
    Name="VCPreLinkEventTool"
/>
<Tool
    Name="VCLinkerTool"
    AdditionalDependencies="x86\ZlibDllRelease\zlibwapi.lib"
    OutputFile="$(OutDir)/testzlib.exe"
    LinkIncremental="1"
    GenerateManifest="false"
    GenerateDebugInformation="true"
    SubSystem="1"
    OptimizeReferences="2"
    EnableCOMDATFolding="2"
    OptimizeForWindows98="1"
    TargetMachine="1"
/>
<Tool
    Name="VCALinkTool"
/>
<Tool
    Name="VCManifestTool"
/>
<Tool
    Name="VCXDCMakeTool"
/>
<Tool
    Name="VCBscMakeTool"
/>
<Tool
    Name="VCFxCopTool"
/>
<Tool
    Name="VCApVerifierTool"
/>
<Tool
    Name="VCWebDeploymentTool"
/>
<Tool
    Name="VCPostBuildEventTool"
/>
</Configuration>
<Configuration
    Name="Release|x64"
    OutputDirectory="x64\TestZlibDll$(ConfigurationName)"
    IntermediateDirectory="x64\TestZlibDll$(ConfigurationName)\Tmp"
```



```
ConfigurationType="1"
InheritedPropertySheets="UpgradeFromVC70.vsprops"
CharacterSet="2"
>
<Tool
    Name="VCPreBuildEventTool"
/>
<Tool
    Name="VCCustomBuildTool"
/>
<Tool
    Name="VCXMLDataGeneratorTool"
/>
<Tool
    Name="VCWebServiceProxyGeneratorTool"
/>
<Tool
    Name="VCMIDLTool"
    TargetEnvironment="3"
/>
<Tool
    Name="VCCLCompilerTool"
    Optimization="2"
    InlineFunctionExpansion="1"
    OmitFramePointers="true"
    AdditionalIncludeDirectories="..\..\..\..\minizip"
    PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;NDEBUG;_CONSOLE;WIN64"
    StringPooling="true"
    BasicRuntimeChecks="0"
    RuntimeLibrary="2"
    BufferSecurityCheck="false"
    EnableFunctionLevelLinking="true"
    UsePrecompiledHeader="0"
    AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
    Detect64BitPortabilityProblems="true"
    DebugInformationFormat="3"
/>
<Tool
    Name="VCManagedResourceCompilerTool"
/>
<Tool
    Name="VCResourceCompilerTool"
/>
<Tool
    Name="VCPreLinkEventTool"
/>
<Tool
    Name="VCLinkerTool"
    AdditionalDependencies="x64\ZlibDllRelease\zlibwapi.lib"
    OutputFile="$(OutDir)/testzlib.exe"
    LinkIncremental="1"
    GenerateManifest="false"
    GenerateDebugInformation="true"
    SubSystem="1"
    OptimizeReferences="2"
    EnableCOMDATFolding="2"
    OptimizeForWindows98="1"
    TargetMachine="17"
/>
<Tool
    Name="VCALinkTool"
/>
<Tool
    Name="VCManifestTool"
/>
<Tool
    Name="VCXDCMakeTool"
/>
<Tool
    Name="VCBscMakeTool"
/>
<Tool
```

```
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPstBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Release|Itanium"
    OutputDirectory="ia64\TestZlibDll$(ConfigurationName)"
    IntermediateDirectory="ia64\TestZlibDll$(ConfigurationName)\Tmp"
    ConfigurationType="1"
    InheritedPropertySheets="UpgradeFromVC70.vsprops"
    CharacterSet="2"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCMIDLTool"
        TargetEnvironment="2"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="2"
        InlineFunctionExpansion="1"
        OmitFramePointers="true"
        AdditionalIncludeDirectories="..\..\..\..\minizip"
        PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECATED;_CRT_S
ECURE_NO_DEPRECATED;ZLIB_WINAPI;NDEBUG;_CONSOLE;WIN64"
        StringPooling="true"
        BasicRuntimeChecks="0"
        RuntimeLibrary="2"
        BufferSecurityCheck="false"
        EnableFunctionLevelLinking="true"
        UsePrecompiledHeader="0"
        AssemblerListingLocation="$(IntDir)\
WarningLevel="3"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="3"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="ia64\ZlibDllRelease\zlibwapi.lib"
        OutputFile="$(OutDir)/testzlib.exe"
        LinkIncremental="1"
        GenerateManifest="false"
        GenerateDebugInformation="true"
        SubSystem="1"
        OptimizeReferences="2"
        EnableCOMDATFolding="2"
```

```
        OptimizeForWindows98="1"
        TargetMachine="5"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
</Configurations>
<References>
</References>
<Files>
    <Filter
        Name="Source Files"
        Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm"
    >
        <File
            RelativePath="..\..\testzlib\testzlib.c"
        >
    </File>
    </Filter>
    <Filter
        Name="Header Files"
        Filter="h;hpp;hxx;hm;inl;inc"
    >
    </Filter>
    <Filter
        Name="Resource Files"
        Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe"
    >
    </Filter>
</Files>
<Globals>
</Globals>
</VisualStudioProject>
```

```
#include <windows.h>

#define IDR_VERSION1 1
IDR_VERSION1 VERSIONINFO MOVEABLE IMPURE LOADONCALL DISCARDABLE
    FILEVERSION 1,2,3,0
    PRODUCTVERSION 1,2,3,0
    FILEFLAGSMASK VS_FFI_FILEFLAGSMASK
    FILEFLAGS 0
    FILEOS VOS_DOS_WINDOWS32
    FILETYPE VFT_DLL
    FILESUBTYPE 0 // not used
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904E4"
        //language ID = U.S. English, char set = Windows, Multilingual

        BEGIN
            VALUE "FileDescription", "zlib data compression library\0"
            VALUE "FileVersion", "1.2.3.0\0"
            VALUE "InternalName", "zlib\0"
            VALUE "OriginalFilename", "zlib.dll\0"
            VALUE "ProductName", "ZLib.DLL\0"
            VALUE "Comments", "DLL support by Alessandro Iacopetti & Gilles Vollant\0"
            VALUE "LegalCopyright", "(C) 1995-2003 Jean-loup Gailly & Mark Adler\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x0409, 1252
    END
END
```

```

<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="8,00"
  Name="zlibstat"
  ProjectGUID="{745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}"
  >
    <Platforms>
      <Platform
        Name="Win32"
      />
      <Platform
        Name="x64"
      />
      <Platform
        Name="Itanium"
      />
    </Platforms>
    <ToolFiles>
    </ToolFiles>
    <Configurations>
      <Configuration
        Name="Debug|Win32"
        OutputDirectory="x86\ZlibStat$(ConfigurationName)"
        IntermediateDirectory="x86\ZlibStat$(ConfigurationName)\Tmp"
        ConfigurationType="4"
        InheritedPropertySheets="UpgradeFromVC70.vsprops"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
      >
        <Tool
          Name="VCPreBuildEventTool"
        />
        <Tool
          Name="VCCustomBuildTool"
        />
        <Tool
          Name="VCXMLDataGeneratorTool"
        />
        <Tool
          Name="VCWebServiceProxyGeneratorTool"
        />
        <Tool
          Name="VCIDLTool"
        />
        <Tool
          Name="VCCLCompilerTool"
          Optimization="0"
          AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
          PreprocessorDefinitions="WIN32;ZLIB_WINAPI;_CRT_NONSTDC_N
O_DEPRECATED;_CRT_SECURE_NO_DEPRECATED"
          ExceptionHandling="0"
          RuntimeLibrary="1"
          BufferSecurityCheck="false"
          PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
          AssemblerListingLocation="$(IntDir)\\"
          ObjectFile="$(IntDir)\\"
          ProgramDataBaseFileName="$(OutDir)\\"
          WarningLevel="3"
          SuppressStartupBanner="true"
          Detect64BitPortabilityProblems="true"
          DebugInformationFormat="1"
        />
        <Tool
          Name="VCManagedResourceCompilerTool"
        />
        <Tool
          Name="VCResourceCompilerTool"
          Culture="1036"
        />
        <Tool
          Name="VCPreLinkEventTool"
        />
        <Tool

```

```

        Name="VCLibrarianTool"
        AdditionalOptions="/MACHINE:X86 /NODEFAULTLIB"
        OutputFile="$(OutDir)\zlibstat.lib"
        SuppressStartupBanner="true"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Debug|x64"
    OutputDirectory="x64\ZlibStat$(ConfigurationName)"
    IntermediateDirectory="x64\ZlibStat$(ConfigurationName)\Tmp"
    ConfigurationType="4"
    InheritedPropertySheets="UpgradeFromVC70.vsprops"
    UseOfMFC="0"
    ATLMinimizesCRunTimeLibraryUsage="false"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCMIDLTool"
        TargetEnvironment="3"
    />
    <Tool
        Name="VCCLCompilerTool"
        Optimization="0"
        AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
        PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPR
ECATE;_CRT_SECURE_NO_DEPRECATED;WIN64"
        ExceptionHandling="0"
        RuntimeLibrary="3"
        BufferSecurityCheck="false"
        PrecompiledHeaderFile="$(IntDir)\zlibstat.pch"
        AssemblerListingLocation="$(IntDir)\\"
        ObjectFile="$(IntDir)\\"
        ProgramDataBaseFileName="$(OutDir)\\"
        WarningLevel="3"
        SuppressStartupBanner="true"
        Detect64BitPortabilityProblems="true"
        DebugInformationFormat="1"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
        Culture="1036"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />

```

```

        <Tool
            Name="VCLibrarianTool"
            AdditionalOptions="/MACHINE:AMD64 /NODEFAULTLIB"
            OutputFile="$(OutDir)\zlibstat.lib"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCALinkTool"
        />
        <Tool
            Name="VCXDCMakeTool"
        />
        <Tool
            Name="VCBscMakeTool"
        />
        <Tool
            Name="VCFxCopTool"
        />
        <Tool
            Name="VCPostBuildEventTool"
        />
    </Configuration>
    <Configuration
        Name="Debug|Itanium"
        OutputDirectory="ia64\ZlibStat$(ConfigurationName)"
        IntermediateDirectory="ia64\ZlibStat$(ConfigurationName)\Tmp"
        ConfigurationType="4"
        InheritedPropertySheets="UpgradeFromVC70.v.props"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
    >
        <Tool
            Name="VCPreBuildEventTool"
        />
        <Tool
            Name="VCCustomBuildTool"
        />
        <Tool
            Name="VCXMLDataGeneratorTool"
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool"
        />
        <Tool
            Name="VCIDLTool"
            TargetEnvironment="2"
        />
        <Tool
            Name="VCCLCompilerTool"
            Optimization="0"
            AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
            PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPR
ECATE;_CRT_SECURE_NO_DEPRECATED;WIN64"
            ExceptionHandling="0"
            RuntimeLibrary="3"
            BufferSecurityCheck="false"
            PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
            AssemblerListingLocation="$(IntDir)\\"
            ObjectFile="$(IntDir)\\"
            ProgramDataBaseFileName="$(OutDir)\\"
            WarningLevel="3"
            SuppressStartupBanner="true"
            Detect64BitPortabilityProblems="true"
            DebugInformationFormat="1"
        />
        <Tool
            Name="VCManagedResourceCompilerTool"
        />
        <Tool
            Name="VCResourceCompilerTool"
            Culture="1036"
        />
        <Tool
            Name="VCPreLinkEventTool"

```

```

        />
        <Tool
            Name="VCLibrarianTool "
            AdditionalOptions="/MACHINE:IA64 /NODEFAULTLIB "
            OutputFile="$(OutDir)\zlibstat.lib"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCALinkTool "
        />
        <Tool
            Name="VCXDCMakeTool "
        />
        <Tool
            Name="VCBscMakeTool "
        />
        <Tool
            Name="VCFxCopTool "
        />
        <Tool
            Name="VCPostBuildEventTool "
        />
    </Configuration>
    <Configuration
        Name="Release|Win32"
        OutputDirectory="x86\ZlibStat$(ConfigurationName)"
        IntermediateDirectory="x86\ZlibStat$(ConfigurationName)\Tmp"
        ConfigurationType="4"
        InheritedPropertySheets="UpgradeFromVC70.vprops"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
    >
        <Tool
            Name="VCPreBuildEventTool "
        />
        <Tool
            Name="VCCustomBuildTool "
        />
        <Tool
            Name="VCXMLDataGeneratorTool "
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool "
        />
        <Tool
            Name="VCMIDLTool "
        />
        <Tool
            Name="VCCLCompilerTool "
            InlineFunctionExpansion="1"
            AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
            PreprocessorDefinitions="WIN32;ZLIB_WINAPI;_CRT_NONSTDC_N
O_DEPRECATED;_CRT_SECURE_NO_DEPRECATED;ASMV;ASMINF"
            StringPooling="true"
            ExceptionHandling="0"
            RuntimeLibrary="0"
            BufferSecurityCheck="false"
            EnableFunctionLevelLinking="true"
            PrecompiledHeaderFile="$(IntDir)\zlibstat.pch"
            AssemblerListingLocation="$(IntDir)\
            ObjectFile="$(IntDir)\
            ProgramDataBaseFileName="$(OutDir)\
            WarningLevel="3"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCManagedResourceCompilerTool "
        />
        <Tool
            Name="VCResourceCompilerTool "
            Culture="1036"
        />
        <Tool
            Name="VCPreLinkEventTool "

```



```

        />
        <Tool
            Name="VCLibrarianTool"
            AdditionalOptions="/MACHINE:X86 /NODEFAULTLIB"
            AdditionalDependencies="..\..\masmx86\gvmat32.obj ..\..\m
asmx86\inffas32.obj "
            OutputFile="$(OutDir)\zlibstat.lib"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCALinkTool"
        />
        <Tool
            Name="VCXDCMakeTool"
        />
        <Tool
            Name="VCBscMakeTool"
        />
        <Tool
            Name="VCFxCopTool"
        />
        <Tool
            Name="VCPostBuildEventTool"
        />
    </Configuration>
    <Configuration
        Name="Release|x64"
        OutputDirectory="x64\ZlibStat$(ConfigurationName)"
        IntermediateDirectory="x64\ZlibStat$(ConfigurationName)\Tmp"
        ConfigurationType="4"
        InheritedPropertySheets="UpgradeFromVC70.vsprops"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
    >
        <Tool
            Name="VCPreBuildEventTool"
        />
        <Tool
            Name="VCCustomBuildTool"
        />
        <Tool
            Name="VCXMLDataGeneratorTool"
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool"
        />
        <Tool
            Name="VCIDLTool"
            TargetEnvironment="3"
        />
        <Tool
            Name="VCCLCompilerTool"
            InlineFunctionExpansion="1"
            AdditionalIncludeDirectories="..\..\..\..\masmx86"
            PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPR
ECATE;_CRT_SECURE_NO_DEPRECATED;ASMV;ASMINF;WIN64"
            StringPooling="true"
            ExceptionHandling="0"
            RuntimeLibrary="2"
            BufferSecurityCheck="false"
            EnableFunctionLevelLinking="true"
            PrecompiledHeaderFile="$(IntDir)\zlibstat.pch"
            AssemblerListingLocation="$(IntDir)\\"
            ObjectFile="$(IntDir)\\"
            ProgramDataBaseFileName="$(OutDir)\\"
            WarningLevel="3"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCManagedResourceCompilerTool"
        />
        <Tool
            Name="VCResourceCompilerTool"
            Culture="1036"
    </Configuration>

```

```

        />
        <Tool
            Name="VCPreLinkEventTool "
        />
        <Tool
            Name="VCLibrarianTool "
            AdditionalOptions="/MACHINE:AMD64 /NODEFAULTLIB"
            AdditionalDependencies="..\..\masmx64\gvmat64.obj ....\m
asmx64\inffasx64.obj "
            OutputFile="$(OutDir)\zlibstat.lib"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCALinkTool "
        />
        <Tool
            Name="VCXDCMakeTool "
        />
        <Tool
            Name="VCBscMakeTool "
        />
        <Tool
            Name="VCFxCopTool "
        />
        <Tool
            Name="VCPostBuildEventTool "
        />
    </Configuration>
    <Configuration
        Name="Release|Itanium"
        OutputDirectory="ia64\ZlibStat$(ConfigurationName)"
        IntermediateDirectory="ia64\ZlibStat$(ConfigurationName)\Tmp"
        ConfigurationType="4"
        InheritedPropertySheets="UpgradeFromVC70.vsprops"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
    >
        <Tool
            Name="VCPreBuildEventTool "
        />
        <Tool
            Name="VCCustomBuildTool "
        />
        <Tool
            Name="VCXMLDataGeneratorTool "
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool "
        />
        <Tool
            Name="VCIDLTool "
            TargetEnvironment="2"
        />
        <Tool
            Name="VCCLCompilerTool "
            InlineFunctionExpansion="1"
            AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
            PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPR
ECATE;_CRT_SECURE_NO_DEPRECATED;WIN64"
            StringPooling="true"
            ExceptionHandling="0"
            RuntimeLibrary="2"
            BufferSecurityCheck="false"
            EnableFunctionLevelLinking="true"
            PrecompiledHeaderFile="$(IntDir)\zlibstat.pch"
            AssemblerListingLocation="$(IntDir)\
            ObjectFile="$(IntDir)\
            ProgramDataBaseFileName="$(OutDir)\
            WarningLevel="3"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCManagedResourceCompilerTool "
        />

```

```

        <Tool
            Name="VCResourceCompilerTool"
            Culture="1036"
        />
        <Tool
            Name="VCPreLinkEventTool"
        />
        <Tool
            Name="VCLibrarianTool"
            AdditionalOptions="/MACHINE:IA64 /NODEFAULTLIB"
            OutputFile="$(OutDir)\zlibstat.lib"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCALinkTool"
        />
        <Tool
            Name="VCXDCMakeTool"
        />
        <Tool
            Name="VCBscMakeTool"
        />
        <Tool
            Name="VCFxCopTool"
        />
        <Tool
            Name="VCPostBuildEventTool"
        />
    </Configuration>
    <Configuration
        Name="ReleaseWithoutAsm|Win32"
        OutputDirectory="x86\ZlibStat$(ConfigurationName)"
        IntermediateDirectory="x86\ZlibStat$(ConfigurationName)\Tmp"
        ConfigurationType="4"
        InheritedPropertySheets="UpgradeFromVC70.v.props"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
    >
        <Tool
            Name="VCPreBuildEventTool"
        />
        <Tool
            Name="VCCustomBuildTool"
        />
        <Tool
            Name="VCXMLDataGeneratorTool"
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool"
        />
        <Tool
            Name="VCIDLTool"
        />
        <Tool
            Name="VCCLCompilerTool"
            InlineFunctionExpansion="1"
            AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
            PreprocessorDefinitions="WIN32;ZLIB_WINAPI;_CRT_NONSTDC_N
O_DEPRECATED;_CRT_SECURE_NO_DEPRECATED"
            StringPooling="true"
            ExceptionHandling="0"
            RuntimeLibrary="0"
            BufferSecurityCheck="false"
            EnableFunctionLevelLinking="true"
            PrecompiledHeaderFile="$(IntDir)\zlibstat.pch"
            AssemblerListingLocation="$(IntDir)\
            ObjectFile="$(IntDir)\
            ProgramDataBaseFileName="$(OutDir)\
            WarningLevel="3"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCManagedResourceCompilerTool"
        />

```

```

        <Tool
            Name="VCResourceCompilerTool"
            Culture="1036"
        />
        <Tool
            Name="VCPreLinkEventTool"
        />
        <Tool
            Name="VCLibrarianTool"
            AdditionalOptions="/MACHINE:X86 /NODEFAULTLIB"
            OutputFile="$(OutDir)\zlibstat.lib"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCALinkTool"
        />
        <Tool
            Name="VCXDCMakeTool"
        />
        <Tool
            Name="VCBscMakeTool"
        />
        <Tool
            Name="VCFxCopTool"
        />
        <Tool
            Name="VCPostBuildEventTool"
        />
    </Configuration>
    <Configuration
        Name="ReleaseWithoutAsm|x64"
        OutputDirectory="x64\ZlibStat$(ConfigurationName)"
        IntermediateDirectory="x64\ZlibStat$(ConfigurationName)\Tmp"
        ConfigurationType="4"
        InheritedPropertySheets="UpgradeFromVC70.vprops"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
    >
        <Tool
            Name="VCPreBuildEventTool"
        />
        <Tool
            Name="VCCustomBuildTool"
        />
        <Tool
            Name="VCXMLDataGeneratorTool"
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool"
        />
        <Tool
            Name="VCMIDLTool"
            TargetEnvironment="3"
        />
        <Tool
            Name="VCCLCompilerTool"
            InlineFunctionExpansion="1"
            AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
            PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPR
ECATE;_CRT_SECURE_NO_DEPRECATED;WIN64"
            StringPooling="true"
            ExceptionHandling="0"
            RuntimeLibrary="2"
            BufferSecurityCheck="false"
            EnableFunctionLevelLinking="true"
            PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
            AssemblerListingLocation="$(IntDir)\
            ObjectFile="$(IntDir)\
            ProgramDataBaseFileName="$(OutDir)\
            WarningLevel="3"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCManagedResourceCompilerTool"

```

```

        />
        <Tool
            Name="VCResourceCompilerTool"
            Culture="1036"
        />
        <Tool
            Name="VCPreLinkEventTool"
        />
        <Tool
            Name="VCLibrarianTool"
            AdditionalOptions="/MACHINE:AMD64 /NODEFAULTLIB"
            OutputFile="$(OutDir)\zlibstat.lib"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCALinkTool"
        />
        <Tool
            Name="VCXDCMakeTool"
        />
        <Tool
            Name="VCBscMakeTool"
        />
        <Tool
            Name="VCFxCopTool"
        />
        <Tool
            Name="VCPostBuildEventTool"
        />
    </Configuration>
    <Configuration
        Name="ReleaseWithoutAsm|Itanium"
        OutputDirectory="ia64\ZlibStat$(ConfigurationName)"
        IntermediateDirectory="ia64\ZlibStat$(ConfigurationName)\Tmp"
        ConfigurationType="4"
        InheritedPropertySheets="UpgradeFromVC70.v.props"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
    >
        <Tool
            Name="VCPreBuildEventTool"
        />
        <Tool
            Name="VCCustomBuildTool"
        />
        <Tool
            Name="VCXMLDataGeneratorTool"
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool"
        />
        <Tool
            Name="VCIDLTool"
            TargetEnvironment="2"
        />
        <Tool
            Name="VCCLCompilerTool"
            InlineFunctionExpansion="1"
            AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
            PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPR
ECATE;_CRT_SECURE_NO_DEPRECATED;WIN64"
            StringPooling="true"
            ExceptionHandling="0"
            RuntimeLibrary="2"
            BufferSecurityCheck="false"
            EnableFunctionLevelLinking="true"
            PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
            AssemblerListingLocation="$(IntDir)\
            ObjectFile="$(IntDir)\
            ProgramDataBaseFile="$(OutDir)\
            WarningLevel="3"
            SuppressStartupBanner="true"
        />
        <Tool

```

```
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
        Culture="1036"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLibrarianTool"
        AdditionalOptions="/MACHINE:IA64 /NODEFAULTLIB"
        OutputFile="$(OutDir)\zlibstat.lib"
        SuppressStartupBanner="true"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
</Configurations>
<References>
</References>
<Files>
    <Filter
        Name="Source Files"
    >
        <File
            RelativePath="..\..\..\adler32.c"
        >
        </File>
        <File
            RelativePath="..\..\..\compress.c"
        >
        </File>
        <File
            RelativePath="..\..\..\crc32.c"
        >
        </File>
        <File
            RelativePath="..\..\..\deflate.c"
        >
        </File>
        <File
            RelativePath="..\..\..\masmx86\gvm32c.c"
        >
            <FileConfiguration
                Name="Debug|x64"
                ExcludedFromBuild="true"
            >
                <Tool
                    Name="VCCLCompilerTool"
                />
            </FileConfiguration>
            <FileConfiguration
                Name="Debug|Itanium"
                ExcludedFromBuild="true"
            >
                <Tool
                    Name="VCCLCompilerTool"
                />
            </FileConfiguration>
        </FileConfiguration>
    </Filter>
</Files>
```

```
        Name="Release|x64"
        ExcludedFromBuild="true"
    >
    <Tool
        Name="VCCLCompilerTool"
    />
</FileConfiguration>
<FileConfiguration
    Name="Release|Itanium"
    ExcludedFromBuild="true"
    >
    <Tool
        Name="VCCLCompilerTool"
    />
</FileConfiguration>
<FileConfiguration
    Name="ReleaseWithoutAsm|x64"
    ExcludedFromBuild="true"
    >
    <Tool
        Name="VCCLCompilerTool"
    />
</FileConfiguration>
<FileConfiguration
    Name="ReleaseWithoutAsm|Itanium"
    ExcludedFromBuild="true"
    >
    <Tool
        Name="VCCLCompilerTool"
    />
</FileConfiguration>
</File>
<File
    RelativePath="..\..\..\gzio.c"
    >
</File>
<File
    RelativePath="..\..\..\inffback.c"
    >
</File>
<File
    RelativePath="..\..\masmx64\inffas8664.c"
    >
    <FileConfiguration
        Name="Debug|Win32"
        ExcludedFromBuild="true"
        >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
    <FileConfiguration
        Name="Debug|Itanium"
        ExcludedFromBuild="true"
        >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
    <FileConfiguration
        Name="Release|Win32"
        ExcludedFromBuild="true"
        >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
    <FileConfiguration
        Name="Release|Itanium"
        ExcludedFromBuild="true"
        >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
</File>
```

```
        </FileConfiguration>
        <FileConfiguration
            Name="ReleaseWithoutAsm|Win32"
            ExcludedFromBuild="true"
        >
            <Tool
                Name="VCCLCompilerTool"
            />
        </FileConfiguration>
        <FileConfiguration
            Name="ReleaseWithoutAsm|Itanium"
            ExcludedFromBuild="true"
        >
            <Tool
                Name="VCCLCompilerTool"
            />
        </FileConfiguration>
    </File>
    <File
        RelativePath="..\..\..\inffast.c"
    >
    </File>
    <File
        RelativePath="..\..\..\inflate.c"
    >
    </File>
    <File
        RelativePath="..\..\..\inftrees.c"
    >
    </File>
    <File
        RelativePath="..\..\minizip\ioapi.c"
    >
    </File>
    <File
        RelativePath="..\..\..\trees.c"
    >
    </File>
    <File
        RelativePath="..\..\..\uncompr.c"
    >
    </File>
    <File
        RelativePath="..\..\minizip\unzip.c"
    >
    </File>
    <File
        RelativePath="..\..\minizip\zip.c"
    >
    </File>
    <File
        RelativePath=".\zlib.rc"
    >
    </File>
    <File
        RelativePath=".\zlibvc.def"
    >
    </File>
    <File
        RelativePath="..\..\..\zutil.c"
    >
    </File>
</Filter>
</Files>
<Globals>
</Globals>
</VisualStudioProject>
```


VERSION 1.23

HEAPSIZE 1048576,8192

EXPORTS

adler32	@1
compress	@2
crc32	@3
deflate	@4
deflateCopy	@5
deflateEnd	@6
deflateInit2_	@7
deflateInit_	@8
deflateParams	@9
deflateReset	@10
deflateSetDictionary	@11
gzclose	@12
gzdopen	@13
gzerror	@14
gzflush	@15
gzopen	@16
gzread	@17
gzwrite	@18
inflate	@19
inflateEnd	@20
inflateInit2_	@21
inflateInit_	@22
inflateReset	@23
inflateSetDictionary	@24
inflateSync	@25
uncompress	@26
zlibVersion	@27
gzprintf	@28
gzputc	@29
gzgetc	@30
gzseek	@31
gzrewind	@32
gtell	@33
gzeof	@34
gzsetparams	@35
zError	@36
inflateSyncPoint	@37
get_crc_table	@38
compress2	@39
gzputs	@40
gzgets	@41
inflateCopy	@42
inflateBackInit_	@43
inflateBack	@44
inflateBackEnd	@45
compressBound	@46
deflateBound	@47
gzclearerr	@48
gzungetc	@49
zlibCompileFlags	@50
deflatePrime	@51
unzOpen	@61
unzClose	@62
unzGetGlobalInfo	@63
unzGetCurrentFileInfo	@64
unzGoToFirstFile	@65
unzGoToNextFile	@66
unzOpenCurrentFile	@67
unzReadCurrentFile	@68
unzOpenCurrentFile3	@69
unztell	@70
unzeof	@71
unzCloseCurrentFile	@72
unzGetGlobalComment	@73
unzStringFileNameCompare	@74
unzLocateFile	@75
unzGetLocalExtrafield	@76

unzOpen2	@77
unzOpenCurrentFile2	@78
unzOpenCurrentFilePassword	@79
zipOpen	@80
zipOpenNewFileInZip	@81
zipWriteInFileInZip	@82
zipCloseFileInZip	@83
zipClose	@84
zipOpenNewFileInZip2	@86
zipCloseFileInZipRaw	@87
zipOpen2	@88
zipOpenNewFileInZip3	@89
unzGetFilePos	@100
unzGoToFilePos	@101
fill_win32_filefunc	@110

```
ï»¿
Microsoft Visual Studio Solution File, Format Version 9.00
# Visual Studio 2005
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "zlibvc", "zlibvc.vcproj", "{8FD826F8-3739-44E6-8CC8-997122E53B8D}"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "zlibstat", "zlibstat.vcproj", "{745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "testzlib", "testzlib.vcproj", "{AA6666AA-E09F-4135-9C0C-4FE50C3C654B}"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "TestZlibDll", "testzlibdll.vcproj", "{C52F9E7B-498A-42BE-8DB4-85A15694366A}"
    ProjectSection(ProjectDependencies) = postProject
        {8FD826F8-3739-44E6-8CC8-997122E53B8D} = {8FD826F8-3739-44E6-8CC8-997122E53B8D}
    EndProjectSection
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "minizip", "minizip.vcproj", "{48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}"
    ProjectSection(ProjectDependencies) = postProject
        {8FD826F8-3739-44E6-8CC8-997122E53B8D} = {8FD826F8-3739-44E6-8CC8-997122E53B8D}
    EndProjectSection
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "miniunz", "miniunz.vcproj", "{C52F9E7B-498A-42BE-8DB4-85A15694382A}"
    ProjectSection(ProjectDependencies) = postProject
        {8FD826F8-3739-44E6-8CC8-997122E53B8D} = {8FD826F8-3739-44E6-8CC8-997122E53B8D}
    EndProjectSection
EndProject
Global
    GlobalSection(SolutionConfigurationPlatforms) = preSolution
        Debug|Itanium = Debug|Itanium
        Debug|Win32 = Debug|Win32
        Debug|x64 = Debug|x64
        Release|Itanium = Release|Itanium
        Release|Win32 = Release|Win32
        Release|x64 = Release|x64
        ReleaseWithoutAsm|Itanium = ReleaseWithoutAsm|Itanium
        ReleaseWithoutAsm|Win32 = ReleaseWithoutAsm|Win32
        ReleaseWithoutAsm|x64 = ReleaseWithoutAsm|x64
    EndGlobalSection
    GlobalSection(ProjectConfigurationPlatforms) = postSolution
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Itanium.ActiveCfg = Debug|Itanium
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Itanium.Build.0 = Debug|Itanium
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Win32.ActiveCfg = Debug|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Win32.Build.0 = Debug|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|x64.ActiveCfg = Debug|x64
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|x64.Build.0 = Debug|x64
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Itanium.ActiveCfg = Release|Itanium
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Itanium.Build.0 = Release|Itanium
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Win32.ActiveCfg = Release|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Win32.Build.0 = Release|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|x64.ActiveCfg = ReleaseWithoutAsm|x64
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|x64.Build.0 = ReleaseWithoutAsm|x64
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Itanium.ActiveCfg = ReleaseWithoutAsm|Itanium
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Itanium.Build.0 = ReleaseWithoutAsm|Itanium
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Win32.ActiveCfg = ReleaseWithoutAsm|Win32
        {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Win32.Build.0 = ReleaseWithoutAsm|Win32
```

```

ReleaseWithoutAsm|Win32
    {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|x64.ActiveCfg =
ReleaseWithoutAsm|x64
    {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|x64.Build.0 = Re
leaseWithoutAsm|x64
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Itanium.ActiveCfg = Debug|It
anium
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Itanium.Build.0 = Debug|Itan
ium
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Win32.ActiveCfg = Debug|Win3
2
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Win32.Build.0 = Debug|Win32
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|x64.ActiveCfg = Debug|x64
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|x64.Build.0 = Debug|x64
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Itanium.ActiveCfg = Releas
e|Itanium
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Itanium.Build.0 = Release|
Itanium
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Win32.ActiveCfg = Release|
Win32
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Win32.Build.0 = Release|Wi
n32
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|x64.ActiveCfg = Release|x6
4
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|x64.Build.0 = Release|x64
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Itanium.ActiveCf
g = ReleaseWithoutAsm|Itanium
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Itanium.Build.0
= ReleaseWithoutAsm|Itanium
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Win32.ActiveCfg
= ReleaseWithoutAsm|Win32
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Win32.Build.0 =
ReleaseWithoutAsm|Win32
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|x64.ActiveCfg =
ReleaseWithoutAsm|x64
    {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|x64.Build.0 = Re
leaseWithoutAsm|x64
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.ActiveCfg = Debug|It
anium
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.Build.0 = Debug|Itan
ium
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.ActiveCfg = Debug|Win3
2
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.Build.0 = Debug|Win32
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.ActiveCfg = Debug|x64
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.Build.0 = Debug|x64
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.ActiveCfg = Releas
e|Itanium
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.Build.0 = Release|
Itanium
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.ActiveCfg = Release|
Win32
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.Build.0 = Release|Wi
n32
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.ActiveCfg = Release|x6
4
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.Build.0 = Release|x64
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium.ActiveCf
g = ReleaseWithoutAsm|Itanium
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium.Build.0
= ReleaseWithoutAsm|Itanium
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.ActiveCfg
= ReleaseWithoutAsm|Win32
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.Build.0 =
ReleaseWithoutAsm|Win32
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.ActiveCfg =
ReleaseWithoutAsm|x64
    {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.Build.0 = Re
leaseWithoutAsm|x64
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Itanium.ActiveCfg = Debug|It
anium
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Itanium.Build.0 = Debug|Itan
ium
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Win32.ActiveCfg = Debug|Win3

```

```
2
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Win32.Build.0 = Debug|Win32
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|x64.ActiveCfg = Debug|x64
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|x64.Build.0 = Debug|x64
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Itanium.ActiveCfg = Release
e|Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Itanium.Build.0 = Release|
Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Win32.ActiveCfg = Release|
Win32
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Win32.Build.0 = Release|Wi
n32
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|x64.ActiveCfg = Release|x6
4
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|x64.Build.0 = Release|x64
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Itanium.ActiveCf
g = Release|Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Itanium.Build.0
= Release|Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Win32.ActiveCfg
= Release|Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|x64.ActiveCfg =
Release|Itanium
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.ActiveCfg = Debug|It
anium
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.Build.0 = Debug|Itan
ium
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.ActiveCfg = Debug|Win3
2
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.Build.0 = Debug|Win32
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.ActiveCfg = Debug|x64
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.Build.0 = Debug|x64
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.ActiveCfg = Release
e|Itanium
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.Build.0 = Release|
Itanium
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.ActiveCfg = Release|
Win32
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.Build.0 = Release|Wi
n32
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.ActiveCfg = Release|x6
4
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.Build.0 = Release|x64
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium.ActiveCf
g = Release|Itanium
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium.Build.0
= Release|Itanium
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.ActiveCfg
= Release|Itanium
    {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.ActiveCfg =
Release|Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Itanium.ActiveCfg = Debug|It
anium
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Itanium.Build.0 = Debug|Itan
ium
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Win32.ActiveCfg = Debug|Win3
2
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Win32.Build.0 = Debug|Win32
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|x64.ActiveCfg = Debug|x64
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|x64.Build.0 = Debug|x64
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Itanium.ActiveCfg = Release
e|Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Itanium.Build.0 = Release|
Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Win32.ActiveCfg = Release|
Win32
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Win32.Build.0 = Release|Wi
n32
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|x64.ActiveCfg = Release|x6
4
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|x64.Build.0 = Release|x64
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Itanium.ActiveCf
g = Release|Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Itanium.Build.0
```

```
= Release|Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Win32.ActiveCfg
= Release|Itanium
    {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|x64.ActiveCfg =
Release|Itanium
    EndGlobalSection
    GlobalSection(SolutionProperties) = preSolution
        HideSolutionNode = FALSE
    EndGlobalSection
EndGlobal
```

```

<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
  ProjectType="Visual C++"
  Version="8,00"
  Name="zlibvc"
  ProjectGUID="{8FD826F8-3739-44E6-8CC8-997122E53B8D}"
  >
    <Platforms>
      <Platform
        Name="Win32"
      />
      <Platform
        Name="x64"
      />
      <Platform
        Name="Itanium"
      />
    </Platforms>
    <ToolFiles>
    </ToolFiles>
    <Configurations>
      <Configuration
        Name="Debug|Win32"
        OutputDirectory="x86\ZlibDll$(ConfigurationName)"
        IntermediateDirectory="x86\ZlibDll$(ConfigurationName)\Tmp"
        ConfigurationType="2"
        InheritedPropertySheets="UpgradeFromVC70.v.props"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
      >
        <Tool
          Name="VCPreBuildEventTool"
        />
        <Tool
          Name="VCCustomBuildTool"
        />
        <Tool
          Name="VCXMLDataGeneratorTool"
        />
        <Tool
          Name="VCWebServiceProxyGeneratorTool"
        />
        <Tool
          Name="VCIDLTool"
          PreprocessorDefinitions="_DEBUG"
          MkTypLibCompatible="true"
          SuppressStartupBanner="true"
          TargetEnvironment="1"
          TypeLibraryName="$(OutDir)/zlibvc.tlb"
        />
        <Tool
          Name="VCCLCompilerTool"
          Optimization="0"
          AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
          PreprocessorDefinitions="WIN32,_CRT_SECURE_NO_DEPRECATED,Z
LIB_WINAPI,ASMV,ASMINF"
          ExceptionHandling="0"
          RuntimeLibrary="1"
          BufferSecurityCheck="false"
          PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
          AssemblerListingLocation="$(IntDir)\\"
          ObjectFile="$(IntDir)\\"
          ProgramDataBaseFileName="$(OutDir)\\"
          BrowseInformation="0"
          WarningLevel="3"
          SuppressStartupBanner="true"
          DebugInformationFormat="4"
        />
        <Tool
          Name="VCManagedResourceCompilerTool"
        />
        <Tool
          Name="VCResourceCompilerTool"
          PreprocessorDefinitions="_DEBUG"

```

```

        Culture="1036"
    />
    <Tool
        Name="VCPreLinkEventTool "
    />
    <Tool
        Name="VCLinkerTool "
        AdditionalOptions="/MACHINE:I386"
        AdditionalDependencies="..\..\masmx86\gvmat32.obj ..\..\m
asmx86\inffas32.obj"
        OutputFile="$(OutDir)\zlibwapi.dll"
        LinkIncremental="2"
        SuppressStartupBanner="true"
        GenerateManifest="false"
        ModuleDefinitionFile="..\zlibvc.def"
        GenerateDebugInformation="true"
        ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
        GenerateMapFile="true"
        MapFileName="$(OutDir)/zlibwapi.map"
        SubSystem="2"
        ImportLibrary="$(OutDir)/zlibwapi.lib"
    />
    <Tool
        Name="VCALinkTool "
    />
    <Tool
        Name="VCManifestTool "
    />
    <Tool
        Name="VCXDCMakeTool "
    />
    <Tool
        Name="VCBscMakeTool "
    />
    <Tool
        Name="VCFxCopTool "
    />
    <Tool
        Name="VCApVerifierTool "
    />
    <Tool
        Name="VCWebDeploymentTool "
    />
    <Tool
        Name="VCPostBuildEventTool "
    />
</Configuration>
<Configuration
    Name="Debug|x64"
    OutputDirectory="x64\ZlibDll$(ConfigurationName)"
    IntermediateDirectory="x64\ZlibDll$(ConfigurationName)\Tmp"
    ConfigurationType="2"
    InheritedPropertySheets="UpgradeFromVC70.vsprops"
    UseOfMFC="0"
    ATLMinimizesCRunTimeLibraryUsage="false"
>
    <Tool
        Name="VCPreBuildEventTool "
    />
    <Tool
        Name="VCCustomBuildTool "
    />
    <Tool
        Name="VCXMLDataGeneratorTool "
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool "
    />
    <Tool
        Name="VCMIDLTool "
        PreprocessorDefinitions="_DEBUG"
        MkTypLibCompatible="true"
        SuppressStartupBanner="true"
        TargetEnvironment="3"
    />

```



```

TypeLibraryName="$(OutDir)/zlibvc.tlb"
/>
<Tool
  Name="VCCLCompilerTool"
  Optimization="0"
  AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
  PreprocessorDefinitions="WIN32,_CRT_SECURE_NO_DEPRECATED,Z
LIB_WINAPI,ASMV,ASMINF;WIN64"
  ExceptionHandling="0"
  RuntimeLibrary="3"
  BufferSecurityCheck="false"
  PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
  AssemblerListingLocation="$(IntDir)\
  ObjectFile="$(IntDir)\
  ProgramDataBaseFileName="$(OutDir)\
  BrowseInformation="0"
  WarningLevel="3"
  SuppressStartupBanner="true"
  DebugInformationFormat="3"
/>
<Tool
  Name="VCManagedResourceCompilerTool"
/>
<Tool
  Name="VCResourceCompilerTool"
  PreprocessorDefinitions="_DEBUG"
  Culture="1036"
/>
<Tool
  Name="VCPreLinkEventTool"
/>
<Tool
  Name="VCLinkerTool"
  AdditionalDependencies="..\..\masmx64\gvm64.obj ....\m
asmx64\inffasx64.obj "
  OutputFile="$(OutDir)\zlibwapi.dll"
  LinkIncremental="2"
  SuppressStartupBanner="true"
  GenerateManifest="false"
  ModuleDefinitionFile="..\zlibvc.def"
  GenerateDebugInformation="true"
  ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
  GenerateMapFile="true"
  MapFileName="$(OutDir)/zlibwapi.map"
  SubSystem="2"
  ImportLibrary="$(OutDir)/zlibwapi.lib"
  TargetMachine="17"
/>
<Tool
  Name="VCALinkTool"
/>
<Tool
  Name="VCManifestTool"
/>
<Tool
  Name="VCXDCMakeTool"
/>
<Tool
  Name="VCBscMakeTool"
/>
<Tool
  Name="VCFxCopTool"
/>
<Tool
  Name="VCAAppVerifierTool"
/>
<Tool
  Name="VCWebDeploymentTool"
/>
<Tool
  Name="VCPPostBuildEventTool"
/>
</Configuration>
</Configuration>

```

```
Name="Debug|Itanium"
OutputDirectory="ia64\ZlibDll$(ConfigurationName)"
IntermediateDirectory="ia64\ZlibDll$(ConfigurationName)\Tmp"
ConfigurationType="2"
InheritedPropertySheets="UpgradeFromVC70.vsprops"
UseOfMFC="0"
ATLMinimizesCRunTimeLibraryUsage="false"
>
<Tool
    Name="VCPreBuildEventTool"
/>
<Tool
    Name="VCCustomBuildTool"
/>
<Tool
    Name="VCXMLDataGeneratorTool"
/>
<Tool
    Name="VCWebServiceProxyGeneratorTool"
/>
<Tool
    Name="VCMIDLTool"
    PreprocessorDefinitions="_DEBUG"
    MkTypLibCompatible="true"
    SuppressStartupBanner="true"
    TargetEnvironment="2"
    TypeLibraryName="$(OutDir)/zlibvc.tlb"
/>
<Tool
    Name="VCCLCompilerTool"
    Optimization="0"
    AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
    PreprocessorDefinitions="WIN32;_CRT_SECURE_NO_DEPRECATED;Z
LIB_WINAPI;WIN64"
    ExceptionHandling="0"
    RuntimeLibrary="3"
    BufferSecurityCheck="false"
    PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
    AssemblerListingLocation="$(IntDir)\\"
    ObjectFile="$(IntDir)\\"
    ProgramDataBaseFileName="$(OutDir)\\"
    BrowseInformation="0"
    WarningLevel="3"
    SuppressStartupBanner="true"
    DebugInformationFormat="3"
/>
<Tool
    Name="VCManagedResourceCompilerTool"
/>
<Tool
    Name="VCResourceCompilerTool"
    PreprocessorDefinitions="_DEBUG"
    Culture="1036"
/>
<Tool
    Name="VCPreLinkEventTool"
/>
<Tool
    Name="VCLinkerTool"
    OutputFile="$(OutDir)\zlibwapi.dll"
    LinkIncremental="2"
    SuppressStartupBanner="true"
    GenerateManifest="false"
    ModuleDefinitionFile="..\zlibvc.def"
    GenerateDebugInformation="true"
    ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
    GenerateMapFile="true"
    MapFileName="$(OutDir)/zlibwapi.map"
    SubSystem="2"
    ImportLibrary="$(OutDir)/zlibwapi.lib"
    TargetMachine="5"
/>
<Tool
    Name="VCALinkTool"
```

```

        />
        <Tool
            Name="VCManifestTool "
        />
        <Tool
            Name="VCXDCMakeTool "
        />
        <Tool
            Name="VCBscMakeTool "
        />
        <Tool
            Name="VCFxCopTool "
        />
        <Tool
            Name="VCApVerifierTool "
        />
        <Tool
            Name="VCWebDeploymentTool "
        />
        <Tool
            Name="VCPostBuildEventTool "
        />
    </Configuration>
    <Configuration
        Name="ReleaseWithoutAsm|Win32"
        OutputDirectory="x86\ZlibDll$(ConfigurationName)"
        IntermediateDirectory="x86\ZlibDll$(ConfigurationName)\Tmp"
        ConfigurationType="2"
        InheritedPropertySheets="UpgradeFromVC70.v.props"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
        WholeProgramOptimization="1"
    >
        <Tool
            Name="VCPreBuildEventTool "
        />
        <Tool
            Name="VCCustomBuildTool "
        />
        <Tool
            Name="VCXMLDataGeneratorTool "
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool "
        />
        <Tool
            Name="VCMIDLTool "
            PreprocessorDefinitions="NDEBUG"
            MkTypLibCompatible="true"
            SuppressStartupBanner="true"
            TargetEnvironment="1"
            TypeLibraryName="$(OutDir)/zlibvc.tlb"
        />
        <Tool
            Name="VCCLCompilerTool "
            InlineFunctionExpansion="1"
            AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
            PreprocessorDefinitions="WIN32,_CRT_SECURE_NO_DEPRECATED,Z
LIB_WINAPI "
            StringPooling="true"
            ExceptionHandling="0"
            RuntimeLibrary="2"
            BufferSecurityCheck="false"
            EnableFunctionLevelLinking="true"
            PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
            AssemblerOutput="2"
            AssemblerListingLocation="$(IntDir)\ "
            ObjectFile="$(IntDir)\ "
            ProgramDataBaseFileName="$(OutDir)\ "
            BrowseInformation="0"
            WarningLevel="3"
            SuppressStartupBanner="true"
        />
        <Tool

```

```
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
        PreprocessorDefinitions="NDEBUG"
        Culture="1036"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalOptions="/MACHINE:I386"
        OutputFile="$(OutDir)\zlibwapi.dll"
        LinkIncremental="1"
        SuppressStartupBanner="true"
        GenerateManifest="false"
        IgnoreAllDefaultLibraries="false"
        ModuleDefinitionFile=".\\zlibvc.def"
        ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
        GenerateMapFile="true"
        MapFileName="$(OutDir)/zlibwapi.map"
        SubSystem="2"
        OptimizeForWindows98="1"
        ImportLibrary="$(OutDir)/zlibwapi.lib"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="ReleaseWithoutAsm|x64"
    OutputDirectory="x64\ZlibDll$(ConfigurationName)"
    IntermediateDirectory="x64\ZlibDll$(ConfigurationName)\Tmp"
    ConfigurationType="2"
    InheritedPropertySheets="UpgradeFromVC70.vsp"
    UseOfMFC="0"
    ATLMinimizesCRunTimeLibraryUsage="false"
    WholeProgramOptimization="1"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
```

LIB_WINAPI;WIN64"

```
        Name="VCMIDLTool"
        PreprocessorDefinitions="NDEBUG"
        MkTypLibCompatible="true"
        SuppressStartupBanner="true"
        TargetEnvironment="3"
        TypeLibraryName="$(OutDir)/zlibvc.tlb"
    />
    <Tool
        Name="VCCLCompilerTool"
        InlineFunctionExpansion="1"
        AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
        PreprocessorDefinitions="WIN32,_CRT_SECURE_NO_DEPRECATED,Z
LIB_WINAPI;WIN64"
        StringPooling="true"
        ExceptionHandling="0"
        RuntimeLibrary="2"
        BufferSecurityCheck="false"
        EnableFunctionLevelLinking="true"
        PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
        AssemblerOutput="2"
        AssemblerListingLocation="$(IntDir)\
ObjectFile="$(IntDir)\
ProgramDataBaseFileName="$(OutDir)\
BrowseInformation="0"
WarningLevel="3"
        SuppressStartupBanner="true"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
        PreprocessorDefinitions="NDEBUG"
        Culture="1036"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        OutputFile="$(OutDir)\zlibwapi.dll"
        LinkIncremental="1"
        SuppressStartupBanner="true"
        GenerateManifest="false"
        IgnoreAllDefaultLibraries="false"
        ModuleDefinitionFile="..\zlibvc.def"
        ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
        GenerateMapFile="true"
        MapFileName="$(OutDir)/zlibwapi.map"
        SubSystem="2"
        OptimizeForWindows98="1"
        ImportLibrary="$(OutDir)/zlibwapi.lib"
        TargetMachine="17"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCAAppVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
```

```

        />
        <Tool
            Name="VCPostBuildEventTool"
        />
    </Configuration>
    <Configuration
        Name="ReleaseWithoutAsm|Itanium"
        OutputDirectory="$(IntDir)\ZlibDll$(ConfigurationName)"
        IntermediateDirectory="$(IntDir)\ZlibDll$(ConfigurationName)\Tmp"
        ConfigurationType="2"
        InheritedPropertySheets="$(VCProjectDir)\..\..\..\VC70\vsprops"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
        WholeProgramOptimization="1"
    >
        <Tool
            Name="VCPreBuildEventTool"
        />
        <Tool
            Name="VCCustomBuildTool"
        />
        <Tool
            Name="VCXMLDataGeneratorTool"
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool"
        />
        <Tool
            Name="VCMIDLTool"
            PreprocessorDefinitions="NDEBUG"
            MkTypLibCompatible="true"
            SuppressStartupBanner="true"
            TargetEnvironment="2"
            TypeLibraryName="$(OutDir)\zlibvc.tlb"
        />
        <Tool
            Name="VCCLCompilerTool"
            InlineFunctionExpansion="1"
            AdditionalIncludeDirectories="$(IntDir)\..\..\..\masmx86"
            PreprocessorDefinitions="WIN32,_CRT_SECURE_NO_DEPRECATED,Z
LIB_WINAPI;WIN64"
            StringPooling="true"
            ExceptionHandling="0"
            RuntimeLibrary="2"
            BufferSecurityCheck="false"
            EnableFunctionLevelLinking="true"
            PrecompiledHeaderFile="$(IntDir)\zlibvc.pch"
            AssemblerOutput="2"
            AssemblerListingLocation="$(IntDir)\\"
            ObjectFile="$(IntDir)\\"
            ProgramDataBaseFileName="$(OutDir)\\"
            BrowseInformation="0"
            WarningLevel="3"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCManagedResourceCompilerTool"
        />
        <Tool
            Name="VCResourceCompilerTool"
            PreprocessorDefinitions="NDEBUG"
            Culture="1036"
        />
        <Tool
            Name="VCPreLinkEventTool"
        />
        <Tool
            Name="VCLinkerTool"
            OutputFile="$(OutDir)\zlibwapi.dll"
            LinkIncremental="1"
            SuppressStartupBanner="true"
            GenerateManifest="false"
            IgnoreAllDefaultLibraries="false"
            ModuleDefinitionFile="$(IntDir)\zlibvc.def"
        />
    </Configuration>

```

```

        ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
        GenerateMapFile="true"
        MapFileName="$(OutDir)/zlibwapi.map"
        SubSystem="2"
        OptimizeForWindows98="1"
        ImportLibrary="$(OutDir)/zlibwapi.lib"
        TargetMachine="5"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Release|Win32"
    OutputDirectory="x86\ZlibDll$(ConfigurationName)"
    IntermediateDirectory="x86\ZlibDll$(ConfigurationName)\Tmp"
    ConfigurationType="2"
    InheritedPropertySheets="UpgradeFromVC70.v.props"
    UseOfMFC="0"
    ATLMinimizesCRunTimeLibraryUsage="false"
    WholeProgramOptimization="1"
>
    <Tool
        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCIDLTool"
        PreprocessorDefinitions="NDEBUG"
        MkTypLibCompatible="true"
        SuppressStartupBanner="true"
        TargetEnvironment="1"
        TypeLibraryName="$(OutDir)/zlibvc.tlb"
    />
    <Tool
        Name="VCCLCompilerTool"
        InlineFunctionExpansion="1"
        AdditionalIncludeDirectories="..\..\..\..\masmx86"
        PreprocessorDefinitions="WIN32;_CRT_SECURE_NO_DEPRECATED;Z
LIB_WINAPI;ASMV;ASMINF"
        StringPooling="true"
        ExceptionHandling="0"
        RuntimeLibrary="2"
        BufferSecurityCheck="false"
        EnableFunctionLevelLinking="true"

```

```

        PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
        AssemblerOutput="2"
        AssemblerListingLocation="$(IntDir)\\"
        ObjectFile="$(IntDir)\\"
        ProgramDataBaseFileName="$(OutDir)\\"
        BrowseInformation="0"
        WarningLevel="3"
        SuppressStartupBanner="true"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
        PreprocessorDefinitions="NDEBUG"
        Culture="1036"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalOptions="/MACHINE:I386"
        AdditionalDependencies="..\..\masmx86\gvm32.obj ....\m
asmx86\inffas32.obj "
        OutputFile="$(OutDir)\zlibwapi.dll"
        LinkIncremental="1"
        SuppressStartupBanner="true"
        GenerateManifest="false"
        IgnoreAllDefaultLibraries="false"
        ModuleDefinitionFile="..\zlibvc.def"
        ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
        GenerateMapFile="true"
        MapFileName="$(OutDir)/zlibwapi.map"
        SubSystem="2"
        OptimizeForWindows98="1"
        ImportLibrary="$(OutDir)/zlibwapi.lib"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
<Configuration
    Name="Release|x64"
    OutputDirectory="x64\ZlibDll$(ConfigurationName)"
    IntermediateDirectory="x64\ZlibDll$(ConfigurationName)\Tmp"
    ConfigurationType="2"
    InheritedPropertySheets="UpgradeFromVC70.vsp"
    UseOfMFC="0"
    ATLMinimizesCRunTimeLibraryUsage="false"
    WholeProgramOptimization="1"
>
    <Tool

```



```

        Name="VCPreBuildEventTool"
    />
    <Tool
        Name="VCCustomBuildTool"
    />
    <Tool
        Name="VCXMLDataGeneratorTool"
    />
    <Tool
        Name="VCWebServiceProxyGeneratorTool"
    />
    <Tool
        Name="VCMIDLTool"
        PreprocessorDefinitions="NDEBUG"
        MkTypLibCompatible="true"
        SuppressStartupBanner="true"
        TargetEnvironment="3"
        TypeLibraryName="$(OutDir)/zlibvc.tlb"
    />
    <Tool
        Name="VCCLCompilerTool"
        InlineFunctionExpansion="1"
        AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
        PreprocessorDefinitions="_CRT_SECURE_NO_DEPRECATED;ZLIB_WI
NAPI;ASMV;ASMINF;WIN64"
        StringPooling="true"
        ExceptionHandling="0"
        RuntimeLibrary="2"
        BufferSecurityCheck="false"
        EnableFunctionLevelLinking="true"
        PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
        AssemblerOutput="2"
        AssemblerListingLocation="$(IntDir)\
"
        ObjectFile="$(IntDir)\
"
        ProgramDataBaseFileName="$(OutDir)\
"
        BrowseInformation="0"
        WarningLevel="3"
        SuppressStartupBanner="true"
    />
    <Tool
        Name="VCManagedResourceCompilerTool"
    />
    <Tool
        Name="VCResourceCompilerTool"
        PreprocessorDefinitions="NDEBUG"
        Culture="1036"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        AdditionalDependencies="..\..\masmx64\gvm64.obj ..\..\m
asmx64\inffasx64.obj "
        OutputFile="$(OutDir)\zlibwapi.dll"
        LinkIncremental="1"
        SuppressStartupBanner="true"
        GenerateManifest="false"
        IgnoreAllDefaultLibraries="false"
        ModuleDefinitionFile=".\zlibvc.def"
        ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
        GenerateMapFile="true"
        MapFileName="$(OutDir)/zlibwapi.map"
        SubSystem="2"
        OptimizeForWindows98="1"
        ImportLibrary="$(OutDir)/zlibwapi.lib"
        TargetMachine="17"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />

```

```

        <Tool
            Name="VCXDCMakeTool "
        />
        <Tool
            Name="VCBscMakeTool "
        />
        <Tool
            Name="VCFxCopTool "
        />
        <Tool
            Name="VCApVerifierTool "
        />
        <Tool
            Name="VCWebDeploymentTool "
        />
        <Tool
            Name="VCPstBuildEventTool "
        />
    </Configuration>
    <Configuration
        Name="Release|Itanium"
        OutputDirectory="ia64\ZlibDll$(ConfigurationName) "
        IntermediateDirectory="ia64\ZlibDll$(ConfigurationName)\Tmp "
        ConfigurationType="2"
        InheritedPropertySheets="UpgradeFromVC70.vsprops"
        UseOfMFC="0"
        ATLMinimizesCRunTimeLibraryUsage="false"
        WholeProgramOptimization="1"
    >
        <Tool
            Name="VCPreBuildEventTool "
        />
        <Tool
            Name="VCCustomBuildTool "
        />
        <Tool
            Name="VCXMLDataGeneratorTool "
        />
        <Tool
            Name="VCWebServiceProxyGeneratorTool "
        />
        <Tool
            Name="VCMIDLTool "
            PreprocessorDefinitions="NDEBUG"
            MkTypLibCompatible="true"
            SuppressStartupBanner="true"
            TargetEnvironment="2"
            TypeLibraryName="$(OutDir)/zlibvc.tlb"
        />
        <Tool
            Name="VCCLCompilerTool "
            InlineFunctionExpansion="1"
            AdditionalIncludeDirectories="..\..\..\..\..\masmx86"
            PreprocessorDefinitions="_CRT_SECURE_NO_DEPRECATED;ZLIB_WI
NAPI;WIN64"
            StringPooling="true"
            ExceptionHandling="0"
            RuntimeLibrary="2"
            BufferSecurityCheck="false"
            EnableFunctionLevelLinking="true"
            PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
            AssemblerOutput="2"
            AssemblerListingLocation="$(IntDir)\ "
            ObjectFile="$(IntDir)\ "
            ProgramDataBaseFileName="$(OutDir)\ "
            BrowseInformation="0"
            WarningLevel="3"
            SuppressStartupBanner="true"
        />
        <Tool
            Name="VCManagedResourceCompilerTool "
        />
        <Tool
            Name="VCResourceCompilerTool "

```

```
        PreprocessorDefinitions="NDEBUG"
        Culture="1036"
    />
    <Tool
        Name="VCPreLinkEventTool"
    />
    <Tool
        Name="VCLinkerTool"
        OutputFile="$(OutDir)\zlibwapi.dll"
        LinkIncremental="1"
        SuppressStartupBanner="true"
        GenerateManifest="false"
        IgnoreAllDefaultLibraries="false"
        ModuleDefinitionFile=".\\zlibvc.def"
        ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
        GenerateMapFile="true"
        MapFileName="$(OutDir)/zlibwapi.map"
        SubSystem="2"
        OptimizeForWindows98="1"
        ImportLibrary="$(OutDir)/zlibwapi.lib"
        TargetMachine="5"
    />
    <Tool
        Name="VCALinkTool"
    />
    <Tool
        Name="VCManifestTool"
    />
    <Tool
        Name="VCXDCMakeTool"
    />
    <Tool
        Name="VCBscMakeTool"
    />
    <Tool
        Name="VCFxCopTool"
    />
    <Tool
        Name="VCApVerifierTool"
    />
    <Tool
        Name="VCWebDeploymentTool"
    />
    <Tool
        Name="VCPostBuildEventTool"
    />
</Configuration>
</Configurations>
<References>
</References>
<Files>
    <Filter
        Name="Source Files"
        Filter="cpp;c;cxx;rc;def;r;odl;hpj;bat;for;f90"
    >
        <File
            RelativePath="..\..\..\adler32.c"
        >
        </File>
        <File
            RelativePath="..\..\..\compress.c"
        >
        </File>
        <File
            RelativePath="..\..\..\crc32.c"
        >
        </File>
        <File
            RelativePath="..\..\..\deflate.c"
        >
        </File>
        <File
            RelativePath="..\..\..\masmx86\gvm32c.c"
        >
    </Filter>
</Files>
```

```
<FileConfiguration
  Name="Debug|x64"
  ExcludedFromBuild="true"
>
  <Tool
    Name="VCCLCompilerTool"
  />
</FileConfiguration>
<FileConfiguration
  Name="Debug|Itanium"
  ExcludedFromBuild="true"
>
  <Tool
    Name="VCCLCompilerTool"
  />
</FileConfiguration>
<FileConfiguration
  Name="ReleaseWithoutAsm|Win32"
  ExcludedFromBuild="true"
>
  <Tool
    Name="VCCLCompilerTool"
  />
</FileConfiguration>
<FileConfiguration
  Name="ReleaseWithoutAsm|x64"
  ExcludedFromBuild="true"
>
  <Tool
    Name="VCCLCompilerTool"
  />
</FileConfiguration>
<FileConfiguration
  Name="ReleaseWithoutAsm|Itanium"
  ExcludedFromBuild="true"
>
  <Tool
    Name="VCCLCompilerTool"
  />
</FileConfiguration>
<FileConfiguration
  Name="Release|x64"
  ExcludedFromBuild="true"
>
  <Tool
    Name="VCCLCompilerTool"
  />
</FileConfiguration>
<FileConfiguration
  Name="Release|Itanium"
  ExcludedFromBuild="true"
>
  <Tool
    Name="VCCLCompilerTool"
  />
</FileConfiguration>
</File>
<File
  RelativePath="..\..\..\gzio.c"
>
</File>
<File
  RelativePath="..\..\..\inffback.c"
>
</File>
<File
  RelativePath="..\..\masmx64\inffas8664.c"
>
  <FileConfiguration
    Name="Debug|Win32"
    ExcludedFromBuild="true"
  >
    <Tool
      Name="VCCLCompilerTool"
```

```
        />
    </FileConfiguration>
    <FileConfiguration
        Name="Debug|Itanium"
        ExcludedFromBuild="true"
    >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
    <FileConfiguration
        Name="ReleaseWithoutAsm|Win32"
        ExcludedFromBuild="true"
    >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
    <FileConfiguration
        Name="ReleaseWithoutAsm|Itanium"
        ExcludedFromBuild="true"
    >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
    <FileConfiguration
        Name="Release|Win32"
        ExcludedFromBuild="true"
    >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
    <FileConfiguration
        Name="Release|Itanium"
        ExcludedFromBuild="true"
    >
        <Tool
            Name="VCCLCompilerTool"
        />
    </FileConfiguration>
</File>
<File
    RelativePath="..\..\..\inffast.c"
>
</File>
<File
    RelativePath="..\..\..\inflate.c"
>
</File>
<File
    RelativePath="..\..\..\inftrees.c"
>
</File>
<File
    RelativePath="..\..\minizip\ioapi.c"
>
</File>
<File
    RelativePath="..\..\minizip\iowin32.c"
>
</File>
<File
    RelativePath="..\..\..\trees.c"
>
</File>
<File
    RelativePath="..\..\..\uncompr.c"
>
</File>
<File
    RelativePath="..\..\minizip\unzip.c"
>
```

```
<FileConfiguration
  Name="Release|Win32"
>
  <Tool
    Name="VCCLCompilerTool"
    AdditionalIncludeDirectories=""
    PreprocessorDefinitions="ZLIB_INTERNAL"
  />
</FileConfiguration>
<FileConfiguration
  Name="Release|x64"
>
  <Tool
    Name="VCCLCompilerTool"
    AdditionalIncludeDirectories=""
    PreprocessorDefinitions="ZLIB_INTERNAL"
  />
</FileConfiguration>
<FileConfiguration
  Name="Release|Itanium"
>
  <Tool
    Name="VCCLCompilerTool"
    AdditionalIncludeDirectories=""
    PreprocessorDefinitions="ZLIB_INTERNAL"
  />
</FileConfiguration>
</File>
<File
  RelativePath="..\..\minizip\zip.c"
>
  <FileConfiguration
    Name="Release|Win32"
  >
    <Tool
      Name="VCCLCompilerTool"
      AdditionalIncludeDirectories=""
      PreprocessorDefinitions="ZLIB_INTERNAL"
    />
  </FileConfiguration>
  <FileConfiguration
    Name="Release|x64"
  >
    <Tool
      Name="VCCLCompilerTool"
      AdditionalIncludeDirectories=""
      PreprocessorDefinitions="ZLIB_INTERNAL"
    />
  </FileConfiguration>
  <FileConfiguration
    Name="Release|Itanium"
  >
    <Tool
      Name="VCCLCompilerTool"
      AdditionalIncludeDirectories=""
      PreprocessorDefinitions="ZLIB_INTERNAL"
    />
  </FileConfiguration>
</File>
<File
  RelativePath=".\zlib.rc"
>
</File>
<File
  RelativePath=".\zlibvc.def"
>
</File>
<File
  RelativePath="..\..\..\zutil.c"
>
</File>
</Filter>
<Filter
  Name="Header Files"
```

```
        Filter="h;hpp;hxx;hm;inl;fi;fd"
    >
    <File
        RelativePath="..\..\..\deflate.h"
    >
</File>
    <File
        RelativePath="..\..\..\infblock.h"
    >
</File>
    <File
        RelativePath="..\..\..\infcodes.h"
    >
</File>
    <File
        RelativePath="..\..\..\inffast.h"
    >
</File>
    <File
        RelativePath="..\..\..\inftrees.h"
    >
</File>
    <File
        RelativePath="..\..\..\infutil.h"
    >
</File>
    <File
        RelativePath="..\..\..\zconf.h"
    >
</File>
    <File
        RelativePath="..\..\..\zlib.h"
    >
</File>
    <File
        RelativePath="..\..\..\zutil.h"
    >
    </File>
</Filter>
<Filter
    Name="Resource Files"
    Filter="ico;cur;bmp;dlg;rc2;rct;bin;cnt;rtf;gif;jpg;jpeg;jpe"
    >
    </Filter>
</Files>
<Globals>
</Globals>
</VisualStudioProject>
```

This directory contains examples of the use of zlib.

`fitblk.c`

- compress just enough input to nearly fill a requested output size
- zlib isn't designed to do this, but `fitblk` does it anyway

`gun.c`

- uncompress a gzip file
- illustrates the use of `inflateBack()` for high speed file-to-file decompression using call-back functions
- is approximately twice as fast as `gzip -d`
- also provides Unix uncompress functionality, again twice as fast

`gzappend.c`

- append to a gzip file
- illustrates the use of the `Z_BLOCK` flush parameter for `inflate()`
- illustrates the use of `deflatePrime()` to start at any bit

`gzjoin.c`

- join gzip files without recalculating the crc or recompressing
- illustrates the use of the `Z_BLOCK` flush parameter for `inflate()`
- illustrates the use of `crc32_combine()`

`gzlog.c`

`gzlog.h`

- efficiently maintain a message log file in gzip format
- illustrates use of raw deflate and `Z_SYNC_FLUSH`
- illustrates use of gzip header extra field

`zlib_how.html`

- painfully comprehensive description of `zpipe.c` (see below)
- describes in excruciating detail the use of `deflate()` and `inflate()`

`zpipe.c`

- reads and writes zlib streams from `stdin` to `stdout`
- illustrates the proper use of `deflate()` and `inflate()`
- deeply commented in `zlib_how.html` (see above)

`zran.c`

- index a zlib or gzip stream and randomly access it
- illustrates the use of `Z_BLOCK`, `inflatePrime()`, and `inflateSetDictionary()` to provide random access


```
/* fitblk.c: example of fitting compressed output to a specified size
   Not copyrighted -- provided to the public domain
   Version 1.1 25 November 2004 Mark Adler */

/* Version history:
   1.0 24 Nov 2004 First version
   1.1 25 Nov 2004 Change deflateInit2() to deflateInit()
                   Use fixed-size, stack-allocated raw buffers
                   Simplify code moving compression to subroutines
                   Use assert() for internal errors
                   Add detailed description of approach
*/

/* Approach to just fitting a requested compressed size:

   fitblk performs three compression passes on a portion of the input
   data in order to determine how much of that input will compress to
   nearly the requested output block size. The first pass generates
   enough deflate blocks to produce output to fill the requested
   output size plus a specified excess amount (see the EXCESS define
   below). The last deflate block may go quite a bit past that, but
   is discarded. The second pass decompresses and recompresses just
   the compressed data that fit in the requested plus excess sized
   buffer. The deflate process is terminated after that amount of
   input, which is less than the amount consumed on the first pass.
   The last deflate block of the result will be of a comparable size
   to the final product, so that the header for that deflate block and
   the compression ratio for that block will be about the same as in
   the final product. The third compression pass decompresses the
   result of the second step, but only the compressed data up to the
   requested size minus an amount to allow the compressed stream to
   complete (see the MARGIN define below). That will result in a
   final compressed stream whose length is less than or equal to the
   requested size. Assuming sufficient input and a requested size
   greater than a few hundred bytes, the shortfall will typically be
   less than ten bytes.

   If the input is short enough that the first compression completes
   before filling the requested output size, then that compressed
   stream is return with no recompression.

   EXCESS is chosen to be just greater than the shortfall seen in a
   two pass approach similar to the above. That shortfall is due to
   the last deflate block compressing more efficiently with a smaller
   header on the second pass. EXCESS is set to be large enough so
   that there is enough uncompressed data for the second pass to fill
   out the requested size, and small enough so that the final deflate
   block of the second pass will be close in size to the final deflate
   block of the third and final pass. MARGIN is chosen to be just
   large enough to assure that the final compression has enough room
   to complete in all cases.
*/

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "zlib.h"

#define local static

/* print nastygram and leave */
local void quit(char *why)
{
    fprintf(stderr, "fitblk abort: %s\n", why);
    exit(1);
}

#define RAWLEN 4096 /* intermediate uncompressed buffer size */

/* compress from file to def until provided buffer is full or end of
   input reached; return last deflate() return value, or Z_ERRNO if
   there was read error on the file */
local int partcompress(FILE *in, z_stream def)
{

```

```

int ret, flush;
unsigned char raw[RAWLEN];

flush = Z_NO_FLUSH;
do {
    def->avail_in = fread(raw, 1, RAWLEN, in);
    if (ferror(in))
        return Z_ERRNO;
    def->next_in = raw;
    if (feof(in))
        flush = Z_FINISH;
    ret = deflate(def, flush);
    assert(ret != Z_STREAM_ERROR);
} while (def->avail_out != 0 && flush == Z_NO_FLUSH);
return ret;
}

/* recompress from inf's input to def's output; the input for inf and
the output for def are set in those structures before calling;
return last deflate() return value, or Z_MEM_ERROR if inflate()
was not able to allocate enough memory when it needed to */
local int recompress(z_stream inf, z_stream def)
{
    int ret, flush;
    unsigned char raw[RAWLEN];

    flush = Z_NO_FLUSH;
    do {
        /* decompress */
        inf->avail_out = RAWLEN;
        inf->next_out = raw;
        ret = inflate(inf, Z_NO_FLUSH);
        assert(ret != Z_STREAM_ERROR && ret != Z_DATA_ERROR &&
            ret != Z_NEED_DICT);
        if (ret == Z_MEM_ERROR)
            return ret;

        /* compress what was decompressed until done or no room */
        def->avail_in = RAWLEN - inf->avail_out;
        def->next_in = raw;
        if (inf->avail_out != 0)
            flush = Z_FINISH;
        ret = deflate(def, flush);
        assert(ret != Z_STREAM_ERROR);
    } while (ret != Z_STREAM_END && def->avail_out != 0);
    return ret;
}

#define EXCESS 256      /* empirically determined stream overage */
#define MARGIN 8        /* amount to back off for completion */

/* compress from stdin to fixed-size block on stdout */
int main(int argc, char **argv)
{
    int ret;                /* return code */
    unsigned size;          /* requested fixed output block size */
    unsigned have;          /* bytes written by deflate() call */
    unsigned char *blk;     /* intermediate and final stream */
    unsigned char *tmp;     /* close to desired size stream */
    z_stream def, inf;      /* zlib deflate and inflate states */

    /* get requested output size */
    if (argc != 2)
        quit("need one argument: size of output block");
    ret = strtoul(argv[1], argv + 1, 10);
    if (argv[1][0] != 0)
        quit("argument must be a number");
    if (ret < 8)             /* 8 is minimum zlib stream size */
        quit("need positive size of 8 or greater");
    size = (unsigned)ret;

    /* allocate memory for buffers and compression engine */
    blk = malloc(size + EXCESS);
    def.zalloc = Z_NULL;

```

```
def.zfree = Z_NULL;
def.opaque = Z_NULL;
ret = deflateInit(&def, Z_DEFAULT_COMPRESSION);
if (ret != Z_OK || blk == NULL)
    quit("out of memory");

/* compress from stdin until output full, or no more input */
def.avail_out = size + EXCESS;
def.next_out = blk;
ret = partcompress(stdin, &def);
if (ret == Z_ERRNO)
    quit("error reading input");

/* if it all fit, then size was undersubscribed -- done! */
if (ret == Z_STREAM_END && def.avail_out >= EXCESS) {
    /* write block to stdout */
    have = size + EXCESS - def.avail_out;
    if (fwrite(blk, 1, have, stdout) != have || ferror(stdout))
        quit("error writing output");

    /* clean up and print results to stderr */
    ret = deflateEnd(&def);
    assert(ret != Z_STREAM_ERROR);
    free(blk);
    fprintf(stderr,
        "%u bytes unused out of %u requested (all input)\n",
        size - have, size);
    return 0;
}

/* it didn't all fit -- set up for recompression */
inf.zalloc = Z_NULL;
inf.zfree = Z_NULL;
inf.opaque = Z_NULL;
inf.avail_in = 0;
inf.next_in = Z_NULL;
ret = inflateInit(&inf);
tmp = malloc(size + EXCESS);
if (ret != Z_OK || tmp == NULL)
    quit("out of memory");
ret = deflateReset(&def);
assert(ret != Z_STREAM_ERROR);

/* do first recompression close to the right amount */
inf.avail_in = size + EXCESS;
inf.next_in = blk;
def.avail_out = size + EXCESS;
def.next_out = tmp;
ret = recompress(&inf, &def);
if (ret == Z_MEM_ERROR)
    quit("out of memory");

/* set up for next recompression */
ret = inflateReset(&inf);
assert(ret != Z_STREAM_ERROR);
ret = deflateReset(&def);
assert(ret != Z_STREAM_ERROR);

/* do second and final recompression (third compression) */
inf.avail_in = size - MARGIN; /* assure stream will complete */
inf.next_in = tmp;
def.avail_out = size;
def.next_out = blk;
ret = recompress(&inf, &def);
if (ret == Z_MEM_ERROR)
    quit("out of memory");
assert(ret == Z_STREAM_END); /* otherwise MARGIN too small */

/* done -- write block to stdout */
have = size - def.avail_out;
if (fwrite(blk, 1, have, stdout) != have || ferror(stdout))
    quit("error writing output");

/* clean up and print results to stderr */
```

```
    free(tmp);
    ret = inflateEnd(&inf);
    assert(ret != Z_STREAM_ERROR);
    ret = deflateEnd(&def);
    assert(ret != Z_STREAM_ERROR);
    free(blk);
    fprintf(stderr,
        "%u bytes unused out of %u requested (%lu input)\n",
        size - have, size, def.total_in);
    return 0;
}
```

```

/* gun.c -- simple gunzip to give an example of the use of inflateBack()
 * Copyright (C) 2003, 2005 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 * Version 1.3 12 June 2005 Mark Adler */

/* Version history:
1.0 16 Feb 2003 First version for testing of inflateBack()
1.1 21 Feb 2005 Decompress concatenated gzip streams
Remove use of "this" variable (C++ keyword)
Fix return value for in()
Improve allocation failure checking
Add typecasting for void * structures
Add -h option for command version and usage
Add a bunch of comments
1.2 20 Mar 2005 Add Unix compress (LZW) decompression
Copy file attributes from input file to output file
1.3 12 Jun 2005 Add casts for error messages [Oberhumer]
*/

/*
gun [ -t ] [ name ... ]

decompresses the data in the named gzip files. If no arguments are given,
gun will decompress from stdin to stdout. The names must end in .gz, -gz,
.z, -z, _z, or .Z. The uncompressed data will be written to a file name
with the suffix stripped. On success, the original file is deleted. On
failure, the output file is deleted. For most failures, the command will
continue to process the remaining names on the command line. A memory
allocation failure will abort the command. If -t is specified, then the
listed files or stdin will be tested as gzip files for integrity (without
checking for a proper suffix), no output will be written, and no files
will be deleted.

Like gzip, gun allows concatenated gzip streams and will decompress them,
writing all of the uncompressed data to the output. Unlike gzip, gun allows
an empty file on input, and will produce no error writing an empty output
file.

gun will also decompress files made by Unix compress, which uses LZW
compression. These files are automatically detected by virtue of their
magic header bytes. Since the end of Unix compress stream is marked by the
end-of-file, they cannot be concatenated. If a Unix compress stream is
encountered in an input file, it is the last stream in that file.

Like gunzip and uncompress, the file attributes of the original compressed
file are maintained in the final uncompressed file, to the extent that the
user permissions allow it.

On my Mac OS X PowerPC G4, gun is almost twice as fast as gunzip (version
1.2.4) is on the same file, when gun is linked with zlib 1.2.2. Also the
LZW decompression provided by gun is about twice as fast as the standard
Unix uncompress command.
*/

/* external functions and related types and constants */
#include <stdio.h> /* fprintf() */
#include <stdlib.h> /* malloc(), free() */
#include <string.h> /* strerror(), strcmp(), strlen(), memcpy() */
#include <errno.h> /* errno */
#include <fcntl.h> /* open() */
#include <unistd.h> /* read(), write(), close(), chown(), unlink() */
#include <sys/types.h>
#include <sys/stat.h> /* stat(), chmod() */
#include <utime.h> /* utime() */
#include "zlib.h" /* inflateBackInit(), inflateBack(), */
/* inflateBackEnd(), crc32() */

/* function declaration */
#define local static

/* buffer constants */
#define SIZE 32768U /* input and output buffer sizes */
#define PIECE 16384 /* limits i/o chunks for 16-bit int case */

```

```
/* structure for infbac() to pass to input function in() -- it maintains the
input file and a buffer of size SIZE */
struct ind {
    int infile;
    unsigned char *inbuf;
};

/* Load input buffer, assumed to be empty, and return bytes loaded and a
pointer to them. read() is called until the buffer is full, or until it
returns end-of-file or error. Return 0 on error. */
local unsigned in(void *in_desc, unsigned char **buf)
{
    int ret;
    unsigned len;
    unsigned char *next;
    struct ind *me = (struct ind *)in_desc;

    next = me->inbuf;
    *buf = next;
    len = 0;
    do {
        ret = PIECE;
        if ((unsigned)ret > SIZE - len)
            ret = (int)(SIZE - len);
        ret = (int)read(me->infile, next, ret);
        if (ret == -1) {
            len = 0;
            break;
        }
        next += ret;
        len += ret;
    } while (ret != 0 && len < SIZE);
    return len;
}

/* structure for infbac() to pass to output function out() -- it maintains the
output file, a running CRC-32 check on the output and the total number of
bytes output, both for checking against the gzip trailer. (The length in
the gzip trailer is stored modulo 2^32, so it's ok if a long is 32 bits and
the output is greater than 4 GB.) */
struct outd {
    int outfile;
    int check; /* true if checking crc and total */
    unsigned long crc;
    unsigned long total;
};

/* Write output buffer and update the CRC-32 and total bytes written. write()
is called until all of the output is written or an error is encountered.
On success out() returns 0. For a write failure, out() returns 1. If the
output file descriptor is -1, then nothing is written.
*/
local int out(void *out_desc, unsigned char *buf, unsigned len)
{
    int ret;
    struct outd *me = (struct outd *)out_desc;

    if (me->check) {
        me->crc = crc32(me->crc, buf, len);
        me->total += len;
    }
    if (me->outfile != -1)
        do {
            ret = PIECE;
            if ((unsigned)ret > len)
                ret = (int)len;
            ret = (int)write(me->outfile, buf, ret);
            if (ret == -1)
                return 1;
            buf += ret;
            len -= ret;
        } while (len != 0);
    return 0;
}
```

```

/* next input byte macro for use inside lunpipe() and gunpipe() */
#define NEXT() (have ? 0 : (have = in(indp, &next)), \
    last = have ? (have--, (int)(*next++)) : -1)

/* memory for gunpipe() and lunpipe() --
   the first 256 entries of prefix[] and suffix[] are never used, could
   have offset the index, but it's faster to waste the memory */
unsigned char inbuf[SIZE];          /* input buffer */
unsigned char outbuf[SIZE];         /* output buffer */
unsigned short prefix[65536];       /* index to LZW prefix string */
unsigned char suffix[65536];        /* one-character LZW suffix */
unsigned char match[65280 + 2];     /* buffer for reversed match or gzip
                                     32K sliding window */

/* throw out what's left in the current bits byte buffer (this is a vestigial
   aspect of the compressed data format derived from an implementation that
   made use of a special VAX machine instruction!) */
#define FLUSHCODE() \
do { \
    left = 0; \
    rem = 0; \
    if (chunk > have) { \
        chunk -= have; \
        have = 0; \
        if (NEXT() == -1) \
            break; \
        chunk--; \
        if (chunk > have) { \
            chunk = have = 0; \
            break; \
        } \
    } \
    have -= chunk; \
    next += chunk; \
    chunk = 0; \
} while (0)

/* Decompress a compress (LZW) file from indp to outfile. The compress magic
   header (two bytes) has already been read and verified. There are have bytes
   of buffered input at next. strm is used for passing error information back
   to gunpipe().

   lunpipe() will return Z_OK on success, Z_BUF_ERROR for an unexpected end of
   file, read error, or write error (a write error indicated by strm->next_in
   not equal to Z_NULL), or Z_DATA_ERROR for invalid input.
*/
local int lunpipe(unsigned have, unsigned char *next, struct ind *indp,
    int outfile, z_stream *strm)
{
    int last;          /* last byte read by NEXT(), or -1 if EOF */
    int chunk;         /* bytes left in current chunk */
    int left;          /* bits left in rem */
    unsigned rem;      /* unused bits from input */
    int bits;          /* current bits per code */
    unsigned code;      /* code, table traversal index */
    unsigned mask;      /* mask for current bits codes */
    int max;           /* maximum bits per code for this stream */
    int flags;          /* compress flags, then block compress flag */
    unsigned end;       /* last valid entry in prefix/suffix tables */
    unsigned temp;      /* current code */
    unsigned prev;      /* previous code */
    unsigned final;     /* last character written for previous code */
    unsigned stack;     /* next position for reversed string */
    unsigned outcnt;    /* bytes in output buffer */
    struct outd outd;   /* output structure */

    /* set up output */
    outd.outfile = outfile;
    outd.check = 0;

    /* process remainder of compress header -- a flags byte */
    flags = NEXT();
    if (last == -1)

```

```

    return Z_BUF_ERROR;
if (flags & 0x60) {
    strm->msg = (char *) "unknown lzw flags set";
    return Z_DATA_ERROR;
}
max = flags & 0x1f;
if (max < 9 || max > 16) {
    strm->msg = (char *) "lzw bits out of range";
    return Z_DATA_ERROR;
}
if (max == 9) /* 9 doesn't really mean 9 */
    max = 10;
flags &= 0x80; /* true if block compress */

/* clear table */
bits = 9;
mask = 0x1ff;
end = flags ? 256 : 255;

/* set up: get first 9-bit code, which is the first decompressed byte, but
   don't create a table entry until the next code */
if (NEXT() == -1) /* no compressed data is ok */
    return Z_OK;
final = prev = (unsigned)last; /* low 8 bits of code */
if (NEXT() == -1) /* missing a bit */
    return Z_BUF_ERROR;
if (last & 1) { /* code must be < 256 */
    strm->msg = (char *) "invalid lzw code";
    return Z_DATA_ERROR;
}
rem = (unsigned)last >> 1; /* remaining 7 bits */
left = 7;
chunk = bits - 2; /* 7 bytes left in this chunk */
outbuf[0] = (unsigned char)final; /* write first decompressed byte */
outcnt = 1;

/* decode codes */
stack = 0;
for (;;) {
    /* if the table will be full after this, increment the code size */
    if (end >= mask && bits < max) {
        FLUSHCODE();
        bits++;
        mask <= 1;
        mask++;
    }

    /* get a code of length bits */
    if (chunk == 0) /* decrement chunk modulo bits */
        chunk = bits;
    code = rem; /* low bits of code */
    if (NEXT() == -1) { /* EOF is end of compressed data */
        /* write remaining buffered output */
        if (outcnt && out(&outd, outbuf, outcnt)) {
            strm->next_in = outbuf; /* signal write error */
            return Z_BUF_ERROR;
        }
        return Z_OK;
    }
    code += (unsigned)last << left; /* middle (or high) bits of code */
    left += 8;
    chunk--;
    if (bits > left) { /* need more bits */
        if (NEXT() == -1) /* can't end in middle of code */
            return Z_BUF_ERROR;
        code += (unsigned)last << left; /* high bits of code */
        left += 8;
        chunk--;
    }
    code &= mask; /* mask to current code length */
    left -= bits; /* number of unused bits */
    rem = (unsigned)last >> (8 - left); /* unused bits from last byte */

    /* process clear code (256) */

```



```

if (code == 256 && flags) {
    FLUSHCODE();
    bits = 9;                                /* initialize bits and mask */
    mask = 0x1fff;
    end = 255;                                /* empty table */
    continue;                                /* get next code */
}

/* special code to reuse last match */
temp = code;                                /* save the current code */
if (code > end) {
    /* Be picky on the allowed code here, and make sure that the code
       we drop through (prev) will be a valid index so that random
       input does not cause an exception. The code != end + 1 check is
       empirically derived, and not checked in the original uncompress
       code. If this ever causes a problem, that check could be safely
       removed. Leaving this check in greatly improves gun's ability
       to detect random or corrupted input after a compress header.
       In any case, the prev > end check must be retained. */
    if (code != end + 1 || prev > end) {
        strm->msg = (char *)"invalid lzw code";
        return Z_DATA_ERROR;
    }
    match[stack++] = (unsigned char)final;
    code = prev;
}

/* walk through linked list to generate output in reverse order */
while (code >= 256) {
    match[stack++] = suffix[code];
    code = prefix[code];
}
match[stack++] = (unsigned char)code;
final = code;

/* link new table entry */
if (end < mask) {
    end++;
    prefix[end] = (unsigned short)prev;
    suffix[end] = (unsigned char)final;
}

/* set previous code for next iteration */
prev = temp;

/* write output in forward order */
while (stack > SIZE - outcnt) {
    while (outcnt < SIZE)
        outbuf[outcnt++] = match[--stack];
    if (out(&outd, outbuf, outcnt)) {
        strm->next_in = outbuf; /* signal write error */
        return Z_BUF_ERROR;
    }
    outcnt = 0;
}
do {
    outbuf[outcnt++] = match[--stack];
} while (stack);

/* loop for next code with final and prev as the last match, rem and
   left provide the first 0..7 bits of the next code, end is the last
   valid table entry */
}
}

```

/* Decompress a gzip file from infile to outfile. strm is assumed to have been successfully initialized with inflateBackInit(). The input file may consist of a series of gzip streams, in which case all of them will be decompressed to the output file. If outfile is -1, then the gzip stream(s) integrity is checked and nothing is written.

The return value is a zlib error code: Z_MEM_ERROR if out of memory, Z_DATA_ERROR if the header or the compressed data is invalid, or if the trailer CRC-32 check or length doesn't match, Z_BUF_ERROR if the input ends

```

    prematurely or a write error occurs, or Z_ERRNO if junk (not a another gzip
    stream) follows a valid gzip stream.
*/
local int gunpipe(z_stream *strm, int infile, int outfile)
{
    int ret, first, last;
    unsigned have, flags, len;
    unsigned char *next;
    struct ind ind, *indp;
    struct outd outd;

    /* setup input buffer */
    ind.infile = infile;
    ind.inbuf = inbuf;
    indp = &ind;

    /* decompress concatenated gzip streams */
    have = 0;
    first = 1;
    strm->next_in = Z_NULL;
    for (;;) {
        /* look for the two magic header bytes for a gzip stream */
        if (NEXT() == -1) {
            ret = Z_OK;
            break;
        }
        if (last != 31 || (NEXT() != 139 && last != 157)) {
            strm->msg = (char *)"incorrect header check";
            ret = first ? Z_DATA_ERROR : Z_ERRNO;
            break;
        }
        first = 0;

        /* process a compress (LZW) file -- can't be concatenated after this */
        if (last == 157) {
            ret = lunpipe(have, next, indp, outfile, strm);
            break;
        }

        /* process remainder of gzip header */
        ret = Z_BUF_ERROR;
        if (NEXT() != 8) {
            if (last == -1) break;
            strm->msg = (char *)"unknown compression method";
            ret = Z_DATA_ERROR;
            break;
        }
        flags = NEXT();
        NEXT();
        NEXT();
        NEXT();
        NEXT();
        NEXT();
        NEXT();
        if (last == -1) break;
        if (flags & 0xe0) {
            strm->msg = (char *)"unknown header flags set";
            ret = Z_DATA_ERROR;
            break;
        }
        if (flags & 4) {
            len = NEXT();
            len += (unsigned)(NEXT()) << 8;
            if (last == -1) break;
            while (len > have) {
                len -= have;
                have = 0;
                if (NEXT() == -1) break;
                len--;
            }
            if (last == -1) break;
            have -= len;
            next += len;
        }
    }
}

```

```

    if (flags & 8) /* file name */
        while (NEXT() != 0 && last != -1)
            ;
    if (flags & 16) /* comment */
        while (NEXT() != 0 && last != -1)
            ;
    if (flags & 2) { /* header crc */
        NEXT();
        NEXT();
    }
    if (last == -1) break;

    /* set up output */
    outd.outfile = outfile;
    outd.check = 1;
    outd.crc = crc32(0L, Z_NULL, 0);
    outd.total = 0;

    /* decompress data to output */
    strm->next_in = next;
    strm->avail_in = have;
    ret = inflateBack(strm, in, indp, out, &outd);
    if (ret != Z_STREAM_END) break;
    next = strm->next_in;
    have = strm->avail_in;
    strm->next_in = Z_NULL; /* so Z_BUF_ERROR means EOF */

    /* check trailer */
    ret = Z_BUF_ERROR;
    if (NEXT() != (outd.crc & 0xff) ||
        NEXT() != ((outd.crc >> 8) & 0xff) ||
        NEXT() != ((outd.crc >> 16) & 0xff) ||
        NEXT() != ((outd.crc >> 24) & 0xff)) {
        /* crc error */
        if (last != -1) {
            strm->msg = (char *) "incorrect data check";
            ret = Z_DATA_ERROR;
        }
        break;
    }
    if (NEXT() != (outd.total & 0xff) ||
        NEXT() != ((outd.total >> 8) & 0xff) ||
        NEXT() != ((outd.total >> 16) & 0xff) ||
        NEXT() != ((outd.total >> 24) & 0xff)) {
        /* length error */
        if (last != -1) {
            strm->msg = (char *) "incorrect length check";
            ret = Z_DATA_ERROR;
        }
        break;
    }

    /* go back and look for another gzip stream */
}

/* clean up and return */
return ret;
}

/* Copy file attributes, from -> to, as best we can. This is best effort, so
no errors are reported. The mode bits, including suid, sgid, and the sticky
bit are copied (if allowed), the owner's user id and group id are copied
(again if allowed), and the access and modify times are copied. */
local void copymeta(char *from, char *to)
{
    struct stat was;
    struct utimbuf when;

    /* get all of from's Unix meta data, return if not a regular file */
    if (stat(from, &was) != 0 || (was.st_mode & S_IFMT) != S_IFREG)
        return;

    /* set to's mode bits, ignore errors */
    (void)chmod(to, was.st_mode & 07777);

```

```

/* copy owner's user and group, ignore errors */
(void)chown(to, was.st_uid, was.st_gid);

/* copy access and modify times, ignore errors */
when.actime = was.st_atime;
when.modtime = was.st_mtime;
(void)utime(to, &when);
}

/* Decompress the file inname to the file outname, of if test is true, just
decompress without writing and check the gzip trailer for integrity. If
inname is NULL or an empty string, read from stdin. If outname is NULL or
an empty string, write to stdout. strm is a pre-initialized inflateBack
structure. When appropriate, copy the file attributes from inname to
outname.

gunzip() returns 1 if there is an out-of-memory error or an unexpected
return code from gunpipe(). Otherwise it returns 0.
*/
local int gunzip(z_stream *strm, char *inname, char *outname, int test)
{
    int ret;
    int infile, outfile;

    /* open files */
    if (inname == NULL || *inname == 0) {
        inname = "-";
        infile = 0;      /* stdin */
    }
    else {
        infile = open(inname, O_RDONLY, 0);
        if (infile == -1) {
            fprintf(stderr, "gun cannot open %s\n", inname);
            return 0;
        }
    }
    if (test)
        outfile = -1;
    else if (outname == NULL || *outname == 0) {
        outname = "-";
        outfile = 1;     /* stdout */
    }
    else {
        outfile = open(outname, O_CREAT | O_TRUNC | O_WRONLY, 0666);
        if (outfile == -1) {
            close(infile);
            fprintf(stderr, "gun cannot create %s\n", outname);
            return 0;
        }
    }
    errno = 0;

    /* decompress */
    ret = gunpipe(strm, infile, outfile);
    if (outfile > 2) close(outfile);
    if (infile > 2) close(infile);

    /* interpret result */
    switch (ret) {
    case Z_OK:
    case Z_ERRNO:
        if (infile > 2 && outfile > 2) {
            copymeta(inname, outname);      /* copy attributes */
            unlink(inname);
        }
        if (ret == Z_ERRNO)
            fprintf(stderr, "gun warning: trailing garbage ignored in %s\n",
                    inname);
        break;
    case Z_DATA_ERROR:
        if (outfile > 2) unlink(outname);
        fprintf(stderr, "gun data error on %s: %s\n", inname, strm->msg);
        break;
    }
}

```

```

    case Z_MEM_ERROR:
        if (outfile > 2) unlink(outname);
        fprintf(stderr, "gun out of memory error--aborting\n");
        return 1;
    case Z_BUF_ERROR:
        if (outfile > 2) unlink(outname);
        if (strm->next_in != Z_NULL) {
            fprintf(stderr, "gun write error on %s: %s\n",
                    outname, strerror(errno));
        }
        else if (errno) {
            fprintf(stderr, "gun read error on %s: %s\n",
                    inname, strerror(errno));
        }
        else {
            fprintf(stderr, "gun unexpected end of file on %s\n",
                    inname);
        }
        break;
    default:
        if (outfile > 2) unlink(outname);
        fprintf(stderr, "gun internal error--aborting\n");
        return 1;
}
return 0;
}

/* Process the gun command line arguments. See the command syntax near the
beginning of this source file. */
int main(int argc, char **argv)
{
    int ret, len, test;
    char *outname;
    unsigned char *window;
    z_stream strm;

    /* initialize inflateBack state for repeated use */
    window = match; /* reuse LZW match buffer */
    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    ret = inflateBackInit(&strm, 15, window);
    if (ret != Z_OK) {
        fprintf(stderr, "gun out of memory error--aborting\n");
        return 1;
    }

    /* decompress each file to the same name with the suffix removed */
    argc--;
    argv++;
    test = 0;
    if (argc && strcmp(*argv, "-h") == 0) {
        fprintf(stderr, "gun 1.3 (12 Jun 2005)\n");
        fprintf(stderr, "Copyright (c) 2005 Mark Adler\n");
        fprintf(stderr, "usage: gun [-t] [file1.gz [file2.Z ...]]\n");
        return 0;
    }
    if (argc && strcmp(*argv, "-t") == 0) {
        test = 1;
        argc--;
        argv++;
    }
    if (argc)
        do {
            if (test)
                outname = NULL;
            else {
                len = (int)strlen(*argv);
                if (strcmp(*argv + len - 3, ".gz") == 0 ||
                    strcmp(*argv + len - 3, "-gz") == 0)
                    len -= 3;
                else if (strcmp(*argv + len - 2, ".Z") == 0 ||
                         strcmp(*argv + len - 2, "-Z") == 0 ||
                         strcmp(*argv + len - 2, "_Z") == 0 ||

```

```
        strcmp(*argv + len - 2, ".Z") == 0)
        len -= 2;
    else {
        fprintf(stderr, "gun error: no gz type on %s--skipping\n",
                *argv);
        continue;
    }
    outname = malloc(len + 1);
    if (outname == NULL) {
        fprintf(stderr, "gun out of memory error--aborting\n");
        ret = 1;
        break;
    }
    memcpy(outname, *argv, len);
    outname[len] = 0;
}
ret = gunzip(&strm, *argv, outname, test);
if (outname != NULL) free(outname);
if (ret) break;
} while (argv++, --argc);
else
    ret = gunzip(&strm, NULL, NULL, test);

/* clean up */
inflateBackEnd(&strm);
return ret;
}
```

```
/* gzappend -- command to append to a gzip file
```

```
Copyright (C) 2003 Mark Adler, all rights reserved  
version 1.1, 4 Nov 2003
```

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Mark Adler madler@alumni.caltech.edu

```
*/
```

```
/*
```

```
* Change history:
```

```
*  
* 1.0   19 Oct 2003   - First version  
* 1.1   4 Nov 2003   - Expand and clarify some comments and notes  
*                   - Add version and copyright to help  
*                   - Send help to stdout instead of stderr  
*                   - Add some preemptive typecasts  
*                   - Add L to constants in lseek() calls  
*                   - Remove some debugging information in error messages  
*                   - Use new data_type definition for zlib 1.2.1  
*                   - Simplify and unify file operations  
*                   - Finish off gzip file in gztack()  
*                   - Use deflatePrime() instead of adding empty blocks  
*                   - Keep gzip file clean on appended file read errors  
*                   - Use in-place rotate instead of auxiliary buffer  
*                   (Why you ask? Because it was fun to write!)
```

```
*/
```

```
/*
```

gzappend takes a gzip file and appends to it, compressing files from the command line or data from stdin. The gzip file is written to directly, to avoid copying that file, in case it's large. Note that this results in the unfriendly behavior that if gzappend fails, the gzip file is corrupted.

This program was written to illustrate the use of the new Z_BLOCK option of zlib 1.2.x's inflate() function. This option returns from inflate() at each block boundary to facilitate locating and modifying the last block bit at the start of the final deflate block. Also whether using Z_BLOCK or not, another required feature of zlib 1.2.x is that inflate() now provides the number of unused bits in the last input byte used. gzappend will not work with versions of zlib earlier than 1.2.1.

gzappend first decompresses the gzip file internally, discarding all but the last 32K of uncompressed data, and noting the location of the last block bit and the number of unused bits in the last byte of the compressed data. The gzip trailer containing the CRC-32 and length of the uncompressed data is verified. This trailer will be later overwritten.

Then the last block bit is cleared by seeking back in the file and rewriting the byte that contains it. Seeking forward, the last byte of the compressed data is saved along with the number of unused bits to initialize deflate.

A deflate process is initialized, using the last 32K of the uncompressed data from the gzip file to initialize the dictionary. If the total uncompressed data was less than 32K, then all of it is used to initialize the dictionary. The deflate output bit buffer is also initialized with the last bits from the original deflate stream. From here on, the data to append is simply compressed using deflate, and written to the gzip file. When that is complete, the new CRC-32 and uncompressed length are written

```
as the trailer of the gzip file.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include "zlib.h"

#define local static
#define LGCHUNK 14
#define CHUNK (1U << LGCHUNK)
#define DSIZE 32768U

/* print an error message and terminate with extreme prejudice */
local void bye(char *msg1, char *msg2)
{
    fprintf(stderr, "gzappend error: %s%s\n", msg1, msg2);
    exit(1);
}

/* return the greatest common divisor of a and b using Euclid's algorithm,
   modified to be fast when one argument much greater than the other, and
   coded to avoid unnecessary swapping */
local unsigned gcd(unsigned a, unsigned b)
{
    unsigned c;

    while (a && b)
        if (a > b) {
            c = b;
            while (a - c >= c)
                c <=<= 1;
            a -= c;
        }
        else {
            c = a;
            while (b - c >= c)
                c <=<= 1;
            b -= c;
        }
    return a + b;
}

/* rotate list[0..len-1] left by rot positions, in place */
local void rotate(unsigned char *list, unsigned len, unsigned rot)
{
    unsigned char tmp;
    unsigned cycles;
    unsigned char *start, *last, *to, *from;

    /* normalize rot and handle degenerate cases */
    if (len < 2) return;
    if (rot >= len) rot %= len;
    if (rot == 0) return;

    /* pointer to last entry in list */
    last = list + (len - 1);

    /* do simple left shift by one */
    if (rot == 1) {
        tmp = *list;
        memcpy(list, list + 1, len - 1);
        *last = tmp;
        return;
    }

    /* do simple right shift by one */
    if (rot == len - 1) {
        tmp = *last;
        memmove(list + 1, list, len - 1);
        *list = tmp;
        return;
    }
}
```



```

    }

    /* otherwise do rotate as a set of cycles in place */
    cycles = gcd(len, rot);          /* number of cycles */
    do {
        start = from = list + cycles; /* start index is arbitrary */
        tmp = *from;                  /* save entry to be overwritten */
        for (;;) {
            to = from;                /* next step in cycle */
            from += rot;              /* go right rot positions */
            if (from > last) from -= len; /* (pointer better not wrap) */
            if (from == start) break; /* all but one shifted */
            *to = *from;              /* shift left */
        }
        *to = tmp;                    /* complete the circle */
    } while (--cycles);
}

/* structure for gzip file read operations */
typedef struct {
    int fd;                /* file descriptor */
    int size;              /* 1 << size is bytes in buf */
    unsigned left;         /* bytes available at next */
    unsigned char *buf;    /* buffer */
    unsigned char *next;   /* next byte in buffer */
    char *name;           /* file name for error messages */
} file;

/* reload buffer */
local int readin(file *in)
{
    int len;

    len = read(in->fd, in->buf, 1 << in->size);
    if (len == -1) bye("error reading ", in->name);
    in->left = (unsigned)len;
    in->next = in->buf;
    return len;
}

/* read from file in, exit if end-of-file */
local int readmore(file *in)
{
    if (readin(in) == 0) bye("unexpected end of ", in->name);
    return 0;
}

#define readl(in) (in->left == 0 ? readmore(in) : 0, \
    in->left--, *(in->next)++)

/* skip over n bytes of in */
local void skip(file *in, unsigned n)
{
    unsigned bypass;

    if (n > in->left) {
        n -= in->left;
        bypass = n & ~((1U << in->size) - 1);
        if (bypass) {
            if (lseek(in->fd, (off_t)bypass, SEEK_CUR) == -1)
                bye("seeking ", in->name);
            n -= bypass;
        }
        readmore(in);
        if (n > in->left)
            bye("unexpected end of ", in->name);
    }
    in->left -= n;
    in->next += n;
}

/* read a four-byte unsigned integer, little-endian, from in */
unsigned long read4(file *in)
{

```

```
    unsigned long val;

    val = read1(in);
    val += (unsigned)read1(in) << 8;
    val += (unsigned long)read1(in) << 16;
    val += (unsigned long)read1(in) << 24;
    return val;
}

/* skip over gzip header */
local void gzheader(file *in)
{
    int flags;
    unsigned n;

    if (read1(in) != 31 || read1(in) != 139) bye(in->name, " not a gzip file");
    if (read1(in) != 8) bye("unknown compression method in", in->name);
    flags = read1(in);
    if (flags & 0xe0) bye("unknown header flags set in", in->name);
    skip(in, 6);
    if (flags & 4) {
        n = read1(in);
        n += (unsigned)(read1(in)) << 8;
        skip(in, n);
    }
    if (flags & 8) while (read1(in) != 0) ;
    if (flags & 16) while (read1(in) != 0) ;
    if (flags & 2) skip(in, 2);
}

/* decompress gzip file "name", return strm with a deflate stream ready to
   continue compression of the data in the gzip file, and return a file
   descriptor pointing to where to write the compressed data -- the deflate
   stream is initialized to compress using level "level" */
local int gzscan(char *name, z_stream *strm, int level)
{
    int ret, lastbit, left, full;
    unsigned have;
    unsigned long crc, tot;
    unsigned char *window;
    off_t lastoff, end;
    file gz;

    /* open gzip file */
    gz.name = name;
    gz.fd = open(name, O_RDWR, 0);
    if (gz.fd == -1) bye("cannot open ", name);
    gz.buf = malloc(CHUNK);
    if (gz.buf == NULL) bye("out of memory", "");
    gz.size = LGCHUNK;
    gz.left = 0;

    /* skip gzip header */
    gzheader(&gz);

    /* prepare to decompress */
    window = malloc(DSIZE);
    if (window == NULL) bye("out of memory", "");
    strm->zalloc = Z_NULL;
    strm->zfree = Z_NULL;
    strm->opaque = Z_NULL;
    ret = inflateInit2(strm, -15);
    if (ret != Z_OK) bye("out of memory", " or library mismatch");

    /* decompress the deflate stream, saving append information */
    lastbit = 0;
    lastoff = lseek(gz.fd, 0L, SEEK_CUR) - gz.left;
    left = 0;
    strm->avail_in = gz.left;
    strm->next_in = gz.next;
    crc = crc32(0L, Z_NULL, 0);
    have = full = 0;
    do {
        /* if needed, get more input */

```

```

    if (strm->avail_in == 0) {
        readmore(&gz);
        strm->avail_in = gz.left;
        strm->next_in = gz.next;
    }

    /* set up output to next available section of sliding window */
    strm->avail_out = DSIZE - have;
    strm->next_out = window + have;

    /* inflate and check for errors */
    ret = inflate(strm, Z_BLOCK);
    if (ret == Z_STREAM_ERROR) bye("internal stream error!", "");
    if (ret == Z_MEM_ERROR) bye("out of memory", "");
    if (ret == Z_DATA_ERROR)
        bye("invalid compressed data--format violated in", name);

    /* update crc and sliding window pointer */
    crc = crc32(crc, window + have, DSIZE - have - strm->avail_out);
    if (strm->avail_out)
        have = DSIZE - strm->avail_out;
    else {
        have = 0;
        full = 1;
    }

    /* process end of block */
    if (strm->data_type & 128) {
        if (strm->data_type & 64)
            left = strm->data_type & 0x1f;
        else {
            lastbit = strm->data_type & 0x1f;
            lastoff = lseek(gz.fd, 0L, SEEK_CUR) - strm->avail_in;
        }
    }
} while (ret != Z_STREAM_END);
inflateEnd(strm);
gz.left = strm->avail_in;
gz.next = strm->next_in;

/* save the location of the end of the compressed data */
end = lseek(gz.fd, 0L, SEEK_CUR) - gz.left;

/* check gzip trailer and save total for deflate */
if (crc != read4(&gz))
    bye("invalid compressed data--crc mismatch in", name);
tot = strm->total_out;
if ((tot & 0xffffffffUL) != read4(&gz))
    bye("invalid compressed data--length mismatch in", name);

/* if not at end of file, warn */
if (gz.left || readin(&gz))
    fprintf(stderr,
        "gzappend warning: junk at end of gzip file overwritten\n");

/* clear last block bit */
lseek(gz.fd, lastoff - (lastbit != 0), SEEK_SET);
if (read(gz.fd, gz.buf, 1) != 1) bye("reading after seek on", name);
*gz.buf = (unsigned char)(*gz.buf ^ (1 << ((8 - lastbit) & 7)));
lseek(gz.fd, -1L, SEEK_CUR);
if (write(gz.fd, gz.buf, 1) != 1) bye("writing after seek to", name);

/* if window wrapped, build dictionary from window by rotating */
if (full) {
    rotate(window, DSIZE, have);
    have = DSIZE;
}

/* set up deflate stream with window, crc, total_in, and leftover bits */
ret = deflateInit2(strm, level, Z_DEFLATED, -15, 8, Z_DEFAULT_STRATEGY);
if (ret != Z_OK) bye("out of memory", "");
deflateSetDictionary(strm, window, have);
strm->adler = crc;
strm->total_in = tot;

```

```
if (left) {
    lseek(gz.fd, --end, SEEK_SET);
    if (read(gz.fd, gz.buf, 1) != 1) bye("reading after seek on ", name);
    deflatePrime(strm, 8 - left, *gz.buf);
}
lseek(gz.fd, end, SEEK_SET);

/* clean up and return */
free(window);
free(gz.buf);
return gz.fd;
}

/* append file "name" to gzip file gd using deflate stream strm -- if last
   is true, then finish off the deflate stream at the end */
local void gztack(char *name, int gd, z_stream *strm, int last)
{
    int fd, len, ret;
    unsigned left;
    unsigned char *in, *out;

    /* open file to compress and append */
    fd = 0;
    if (name != NULL) {
        fd = open(name, O_RDONLY, 0);
        if (fd == -1)
            fprintf(stderr, "gzappend warning: %s not found, skipping ...\n",
                    name);
    }

    /* allocate buffers */
    in = fd == -1 ? NULL : malloc(CHUNK);
    out = malloc(CHUNK);
    if (out == NULL) bye("out of memory", "");

    /* compress input file and append to gzip file */
    do {
        /* get more input */
        len = fd == -1 ? 0 : read(fd, in, CHUNK);
        if (len == -1) {
            fprintf(stderr,
                    "gzappend warning: error reading %s, skipping rest ...\n",
                    name);
            len = 0;
        }
        strm->avail_in = (unsigned)len;
        strm->next_in = in;
        if (len) strm->adler = crc32(strm->adler, in, (unsigned)len);

        /* compress and write all available output */
        do {
            strm->avail_out = CHUNK;
            strm->next_out = out;
            ret = deflate(strm, last && len == 0 ? Z_FINISH : Z_NO_FLUSH);
            left = CHUNK - strm->avail_out;
            while (left) {
                len = write(gd, out + CHUNK - strm->avail_out - left, left);
                if (len == -1) bye("writing gzip file", "");
                left -= (unsigned)len;
            }
        } while (strm->avail_out == 0 && ret != Z_STREAM_END);
    } while (len != 0);

    /* write trailer after last entry */
    if (last) {
        deflateEnd(strm);
        out[0] = (unsigned char)(strm->adler);
        out[1] = (unsigned char)(strm->adler >> 8);
        out[2] = (unsigned char)(strm->adler >> 16);
        out[3] = (unsigned char)(strm->adler >> 24);
        out[4] = (unsigned char)(strm->total_in);
        out[5] = (unsigned char)(strm->total_in >> 8);
        out[6] = (unsigned char)(strm->total_in >> 16);
        out[7] = (unsigned char)(strm->total_in >> 24);
    }
}
```

```
len = 8;
do {
    ret = write(gd, out + 8 - len, len);
    if (ret == -1) bye("writing gzip file", "");
    len -= ret;
} while (len);
close(gd);
}

/* clean up and return */
free(out);
if (in != NULL) free(in);
if (fd > 0) close(fd);
}

/* process the compression level option if present, scan the gzip file, and
append the specified files, or append the data from stdin if no other file
names are provided on the command line -- the gzip file must be writable
and seekable */
int main(int argc, char **argv)
{
    int gd, level;
    z_stream strm;

    /* ignore command name */
    argv++;

    /* provide usage if no arguments */
    if (*argv == NULL) {
        printf("gzappend 1.1 (4 Nov 2003) Copyright (C) 2003 Mark Adler\n");
        printf("usage: gzappend [-level] file.gz [ addthis [ andthis ... ]]\n");
        return 0;
    }

    /* set compression level */
    level = Z_DEFAULT_COMPRESSION;
    if (argv[0][0] == '-') {
        if (argv[0][1] < '0' || argv[0][1] > '9' || argv[0][2] != 0)
            bye("invalid compression level", "");
        level = argv[0][1] - '0';
        if (*++argv == NULL) bye("no gzip file name after options", "");
    }

    /* prepare to append to gzip file */
    gd = gzscan(*argv++, &strm, level);

    /* append files on command line, or from stdin if none */
    if (*argv == NULL)
        gztack(NULL, gd, &strm, 1);
    else
        do {
            gztack(*argv, gd, &strm, argv[1] == NULL);
        } while (*++argv != NULL);
    return 0;
}
```

```
/* gzjoin -- command to join gzip files into one gzip file
```

```
Copyright (C) 2004 Mark Adler, all rights reserved  
version 1.0, 11 Dec 2004
```

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Mark Adler madler@alumni.caltech.edu

```
*/
```

```
/*
```

```
* Change history:
```

```
*
```

```
* 1.0 11 Dec 2004      - First version
```

```
* 1.1 12 Jun 2005      - Changed ssize_t to long for portability
```

```
*/
```

```
/*
```

gzjoin takes one or more gzip files on the command line and writes out a single gzip file that will uncompress to the concatenation of the uncompressed data from the individual gzip files. gzjoin does this without having to recompress any of the data and without having to calculate a new crc32 for the concatenated uncompressed data. gzjoin does however have to decompress all of the input data in order to find the bits in the compressed data that need to be modified to concatenate the streams.

gzjoin does not do an integrity check on the input gzip files other than checking the gzip header and decompressing the compressed data. They are otherwise assumed to be complete and correct.

Each joint between gzip files removes at least 18 bytes of previous trailer and subsequent header, and inserts an average of about three bytes to the compressed data in order to connect the streams. The output gzip file has a minimal ten-byte gzip header with no file name or modification time.

This program was written to illustrate the use of the Z_BLOCK option of inflate() and the crc32_combine() function. gzjoin will not compile with versions of zlib earlier than 1.2.3.

```
*/
```

```
#include <stdio.h>            /* fputs(), fprintf(), fwrite(), putc() */  
#include <stdlib.h>          /* exit(), malloc(), free() */  
#include <fcntl.h>           /* open() */  
#include <unistd.h>          /* close(), read(), lseek() */  
#include "zlib.h"            /* crc32(), crc32_combine(), inflateInit2(), inflate(), inflateEnd() */  
    /* crc32(), crc32_combine(), inflateInit2(), inflate(), inflateEnd() */
```

```
#define local static
```

```
/* exit with an error (return a value to allow use in an expression) */
```

```
local int bail(char *why1, char *why2)
```

```
{  
    fprintf(stderr, "gzjoin error: %s%s, output incomplete\n", why1, why2);  
    exit(1);  
    return 0;  
}
```

```
/* -- simple buffered file input with access to the buffer -- */
```

```
#define CHUNK 32768           /* must be a power of two and fit in unsigned */
```

```

/* bin buffered input file type */
typedef struct {
    char *name;           /* name of file for error messages */
    int fd;               /* file descriptor */
    unsigned left;        /* bytes remaining at next */
    unsigned char *next;  /* next byte to read */
    unsigned char *buf;   /* allocated buffer of length CHUNK */
} bin;

/* close a buffered file and free allocated memory */
local void bclose(bin *in)
{
    if (in != NULL) {
        if (in->fd != -1)
            close(in->fd);
        if (in->buf != NULL)
            free(in->buf);
        free(in);
    }
}

/* open a buffered file for input, return a pointer to type bin, or NULL on
   failure */
local bin *bopen(char *name)
{
    bin *in;

    in = malloc(sizeof(bin));
    if (in == NULL)
        return NULL;
    in->buf = malloc(CHUNK);
    in->fd = open(name, O_RDONLY, 0);
    if (in->buf == NULL || in->fd == -1) {
        bclose(in);
        return NULL;
    }
    in->left = 0;
    in->next = in->buf;
    in->name = name;
    return in;
}

/* load buffer from file, return -1 on read error, 0 or 1 on success, with
   1 indicating that end-of-file was reached */
local int blood(bin *in)
{
    long len;

    if (in == NULL)
        return -1;
    if (in->left != 0)
        return 0;
    in->next = in->buf;
    do {
        len = (long)read(in->fd, in->buf + in->left, CHUNK - in->left);
        if (len < 0)
            return -1;
        in->left += (unsigned)len;
    } while (len != 0 && in->left < CHUNK);
    return len == 0 ? 1 : 0;
}

/* get a byte from the file, bail if end of file */
#define bget(in) (in->left ? 0 : blood(in), \
                 in->left ? (in->left--, *(in->next)++) : \
                 bail("unexpected end of file on ", in->name))

/* get a four-byte little-endian unsigned integer from file */
local unsigned long bget4(bin *in)
{
    unsigned long val;

    val = bget(in);

```

```
    val += (unsigned long)(bget(in)) << 8;
    val += (unsigned long)(bget(in)) << 16;
    val += (unsigned long)(bget(in)) << 24;
    return val;
}

/* skip bytes in file */
local void bskip(bin *in, unsigned skip)
{
    /* check pointer */
    if (in == NULL)
        return;

    /* easy case -- skip bytes in buffer */
    if (skip <= in->left) {
        in->left -= skip;
        in->next += skip;
        return;
    }

    /* skip what's in buffer, discard buffer contents */
    skip -= in->left;
    in->left = 0;

    /* seek past multiples of CHUNK bytes */
    if (skip > CHUNK) {
        unsigned left;

        left = skip & (CHUNK - 1);
        if (left == 0) {
            /* exact number of chunks: seek all the way minus one byte to check
             for end-of-file with a read */
            lseek(in->fd, skip - 1, SEEK_CUR);
            if (read(in->fd, in->buf, 1) != 1)
                bail("unexpected end of file on ", in->name);
            return;
        }

        /* skip the integral chunks, update skip with remainder */
        lseek(in->fd, skip - left, SEEK_CUR);
        skip = left;
    }

    /* read more input and skip remainder */
    bload(in);
    if (skip > in->left)
        bail("unexpected end of file on ", in->name);
    in->left -= skip;
    in->next += skip;
}

/* -- end of buffered input functions -- */

/* skip the gzip header from file in */
local void gzhead(bin *in)
{
    int flags;

    /* verify gzip magic header and compression method */
    if (bget(in) != 0x1f || bget(in) != 0x8b || bget(in) != 8)
        bail(in->name, " is not a valid gzip file");

    /* get and verify flags */
    flags = bget(in);
    if ((flags & 0xe0) != 0)
        bail("unknown reserved bits set in ", in->name);

    /* skip modification time, extra flags, and os */
    bskip(in, 6);

    /* skip extra field if present */
    if (flags & 4) {
        unsigned len;
```



```

        len = bget(in);
        len += (unsigned)(bget(in)) << 8;
        bskip(in, len);
    }

    /* skip file name if present */
    if (flags & 8)
        while (bget(in) != 0)
            ;

    /* skip comment if present */
    if (flags & 16)
        while (bget(in) != 0)
            ;

    /* skip header crc if present */
    if (flags & 2)
        bskip(in, 2);
}

/* write a four-byte little-endian unsigned integer to out */
local void put4(unsigned long val, FILE *out)
{
    putc(val & 0xff, out);
    putc((val >> 8) & 0xff, out);
    putc((val >> 16) & 0xff, out);
    putc((val >> 24) & 0xff, out);
}

/* Load up zlib stream from buffered input, bail if end of file */
local void zpull(z_stream strm, bin *in)
{
    if (in->left == 0)
        bload(in);
    if (in->left == 0)
        bail("unexpected end of file on ", in->name);
    strm->avail_in = in->left;
    strm->next_in = in->next;
}

/* Write header for gzip file to out and initialize trailer. */
local void gzinit(unsigned long *crc, unsigned long *tot, FILE *out)
{
    fwrite("\x1f\x8b\x08\x00\x00\x00\xff", 1, 10, out);
    *crc = crc32(0L, Z_NULL, 0);
    *tot = 0;
}

/* Copy the compressed data from name, zeroing the last block bit of the last
block if clr is true, and adding empty blocks as needed to get to a byte
boundary. If clr is false, then the last block becomes the last block of
the output, and the gzip trailer is written. crc and tot maintains the
crc and length (modulo 2^32) of the output for the trailer. The resulting
gzip file is written to out. gzinit() must be called before the first call
of gzcopy() to write the gzip header and to initialize crc and tot. */
local void gzcopy(char *name, int clr, unsigned long *crc, unsigned long *tot,
FILE *out)
{
    int ret;                /* return value from zlib functions */
    int pos;                /* where the "last block" bit is in byte */
    int last;               /* true if processing the last block */
    bin *in;                /* buffered input file */
    unsigned char *start;   /* start of compressed data in buffer */
    unsigned char *junk;    /* buffer for uncompressed data -- discarded */
    z_off_t len;            /* length of uncompressed data (support > 4 GB) */
    z_stream strm;          /* zlib inflate stream */

    /* open gzip file and skip header */
    in = bopen(name);
    if (in == NULL)
        bail("could not open ", name);
    gzhead(in);

    /* allocate buffer for uncompressed data and initialize raw inflate

```

```

    stream */
    junk = malloc(CHUNK);
    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    strm.avail_in = 0;
    strm.next_in = Z_NULL;
    ret = inflateInit2(&strm, -15);
    if (junk == NULL || ret != Z_OK)
        bail("out of memory", "");

    /* inflate and copy compressed data, clear last-block bit if requested */
    len = 0;
    zpull(&strm, in);
    start = strm.next_in;
    last = start[0] & 1;
    if (last && clr)
        start[0] &= ~1;
    strm.avail_out = 0;
    for (;;) {
        /* if input used and output done, write used input and get more */
        if (strm.avail_in == 0 && strm.avail_out != 0) {
            fwrite(start, 1, strm.next_in - start, out);
            start = in->buf;
            in->left = 0;
            zpull(&strm, in);
        }

        /* decompress -- return early when end-of-block reached */
        strm.avail_out = CHUNK;
        strm.next_out = junk;
        ret = inflate(&strm, Z_BLOCK);
        switch (ret) {
            case Z_MEM_ERROR:
                bail("out of memory", "");
            case Z_DATA_ERROR:
                bail("invalid compressed data in ", in->name);
        }

        /* update length of uncompressed data */
        len += CHUNK - strm.avail_out;

        /* check for block boundary (only get this when block copied out) */
        if (strm.data_type & 128) {
            /* if that was the last block, then done */
            if (last)
                break;

            /* number of unused bits in last byte */
            pos = strm.data_type & 7;

            /* find the next last-block bit */
            if (pos != 0) {
                /* next last-block bit is in last used byte */
                pos = 0x100 >> pos;
                last = strm.next_in[-1] & pos;
                if (last && clr)
                    strm.next_in[-1] &= ~pos;
            }
            else {
                /* next last-block bit is in next unused byte */
                if (strm.avail_in == 0) {
                    /* don't have that byte yet -- get it */
                    fwrite(start, 1, strm.next_in - start, out);
                    start = in->buf;
                    in->left = 0;
                    zpull(&strm, in);
                }
                last = strm.next_in[0] & 1;
                if (last && clr)
                    strm.next_in[0] &= ~1;
            }
        }
    }
}

```

```

/* update buffer with unused input */
in->left = strm.avail_in;
in->next = strm.next_in;

/* copy used input, write empty blocks to get to byte boundary */
pos = strm.data_type & 7;
fwrite(start, 1, in->next - start - 1, out);
last = in->next[-1];
if (pos == 0 || !clr)
    /* already at byte boundary, or last file: write last byte */
    putc(last, out);
else {
    /* append empty blocks to last byte */
    last &= ((0x100 >> pos) - 1); /* assure unused bits are zero */
    if (pos & 1) {
        /* odd -- append an empty stored block */
        putc(last, out);
        if (pos == 1)
            putc(0, out); /* two more bits in block header */
        fwrite("\0\0\xff\xff", 1, 4, out);
    }
    else {
        /* even -- append 1, 2, or 3 empty fixed blocks */
        switch (pos) {
            case 6:
                putc(last | 8, out);
                last = 0;
            case 4:
                putc(last | 0x20, out);
                last = 0;
            case 2:
                putc(last | 0x80, out);
                putc(0, out);
            }
    }
}

/* update crc and tot */
*crc = crc32_combine(*crc, bget4(in), len);
*tot += (unsigned long)len;

/* clean up */
inflateEnd(&strm);
free(junk);
bclose(in);

/* write trailer if this is the last gzip file */
if (!clr) {
    put4(*crc, out);
    put4(*tot, out);
}
}

/* join the gzip files on the command line, write result to stdout */
int main(int argc, char **argv)
{
    unsigned long crc, tot; /* running crc and total uncompressed length */

    /* skip command name */
    argc--;
    argv++;

    /* show usage if no arguments */
    if (argc == 0) {
        fputs("gzjoin usage: gzjoin f1.gz [f2.gz [f3.gz ...]] > fjoin.gz\n",
            stderr);
        return 0;
    }

    /* join gzip files on command line and write to stdout */
    gzinit(&crc, &tot, stdout);
    while (argc--)
        gzcopyp(*argv++, argc, &crc, &tot, stdout);
}

```

```
    /* done */  
    return 0;  
}
```

```

/*
 * gzlog.c
 * Copyright (C) 2004 Mark Adler
 * For conditions of distribution and use, see copyright notice in gzlog.h
 * version 1.0, 26 Nov 2004
 */

#include <string.h>          /* memcmp() */
#include <stdlib.h>          /* malloc(), free(), NULL */
#include <sys/types.h>       /* size_t, off_t */
#include <unistd.h>          /* read(), close(), sleep(), ftruncate(), */
                           /* lseek() */
#include <fcntl.h>           /* open() */
#include <sys/file.h>        /* flock() */
#include "zlib.h"            /* deflateInit2(), deflate(), deflateEnd() */

#include "gzlog.h"           /* interface */
#define local static

/* log object structure */
typedef struct {
    int id;                  /* object identifier */
    int fd;                  /* log file descriptor */
    off_t extra;             /* offset of extra "ap" subfield */
    off_t mark_off;          /* offset of marked data */
    off_t last_off;          /* offset of last block */
    unsigned long crc;       /* uncompressed crc */
    unsigned long len;       /* uncompressed length (modulo 2^32) */
    unsigned stored;         /* length of current stored block */
} gz_log;

#define GZLOGID 19334        /* gz_log object identifier */

#define LOCK_RETRY 1         /* retry lock once a second */
#define LOCK_PATIENCE 1200  /* try about twenty minutes before forcing */

/* acquire a lock on a file */
local int lock(int fd)
{
    int patience;

    /* try to lock every LOCK_RETRY seconds for LOCK_PATIENCE seconds */
    patience = LOCK_PATIENCE;
    do {
        if (flock(fd, LOCK_EX + LOCK_NB) == 0)
            return 0;
        (void)sleep(LOCK_RETRY);
        patience -= LOCK_RETRY;
    } while (patience > 0);

    /* we've run out of patience -- give up */
    return -1;
}

/* release lock */
local void unlock(int fd)
{
    (void)flock(fd, LOCK_UN);
}

/* release a log object */
local void log_clean(gz_log *log)
{
    unlock(log->fd);
    (void)close(log->fd);
    free(log);
}

/* read an unsigned long from a byte buffer little-endian */
local unsigned long make_ulg(unsigned char *buf)
{
    int n;
    unsigned long val;

```

```

    val = (unsigned long)(*buf++);
    for (n = 8; n < 32; n += 8)
        val += (unsigned long)(*buf++) << n;
    return val;
}

/* read an off_t from a byte buffer little-endian */
local off_t make_off(unsigned char *buf)
{
    int n;
    off_t val;

    val = (off_t)(*buf++);
    for (n = 8; n < 64; n += 8)
        val += (off_t)(*buf++) << n;
    return val;
}

/* write an unsigned long little-endian to byte buffer */
local void dice_ulg(unsigned long val, unsigned char *buf)
{
    int n;

    for (n = 0; n < 4; n++) {
        *buf++ = val & 0xff;
        val >>= 8;
    }
}

/* write an off_t little-endian to byte buffer */
local void dice_off(off_t val, unsigned char *buf)
{
    int n;

    for (n = 0; n < 8; n++) {
        *buf++ = val & 0xff;
        val >>= 8;
    }
}

/* initial, empty gzip file for appending */
local char empty_gz[] = {
    0x1f, 0x8b,                /* magic gzip id */
    8,                        /* compression method is deflate */
    4,                        /* there is an extra field */
    0, 0, 0, 0,              /* no modification time provided */
    0, 0xff,                 /* no extra flags, no OS */
    20, 0, 'a', 'p', 16, 0,  /* extra field with "ap" subfield */
    32, 0, 0, 0, 0, 0, 0, 0, /* offset of uncompressed data */
    32, 0, 0, 0, 0, 0, 0, 0, /* offset of last block */
    1, 0, 0, 0xff, 0xff,     /* empty stored block (last) */
    0, 0, 0, 0,             /* crc */
    0, 0, 0, 0              /* uncompressed length */
};

/* initialize a log object with locking */
void *gzlog_open(char *path)
{
    unsigned xlen;
    unsigned char temp[20];
    unsigned sub_len;
    int good;
    gz_log *log;

    /* allocate log structure */
    log = malloc(sizeof(gz_log));
    if (log == NULL)
        return NULL;
    log->id = GZLOGID;

    /* open file, creating it if necessary, and locking it */
    log->fd = open(path, O_RDWR | O_CREAT, 0600);
    if (log->fd < 0) {

```

```

    free(log);
    return NULL;
}
if (lock(log->fd)) {
    close(log->fd);
    free(log);
    return NULL;
}

/* if file is empty, write new gzip stream */
if (lseek(log->fd, 0, SEEK_END) == 0) {
    if (write(log->fd, empty_gz, sizeof(empty_gz)) != sizeof(empty_gz)) {
        log_clean(log);
        return NULL;
    }
}

/* check gzip header */
(void)lseek(log->fd, 0, SEEK_SET);
if (read(log->fd, temp, 12) != 12 || temp[0] != 0x1f ||
    temp[1] != 0x8b || temp[2] != 8 || (temp[3] & 4) == 0) {
    log_clean(log);
    return NULL;
}

/* process extra field to find "ap" sub-field */
xlen = temp[10] + (temp[11] << 8);
good = 0;
while (xlen) {
    if (xlen < 4 || read(log->fd, temp, 4) != 4)
        break;
    sub_len = temp[2];
    sub_len += temp[3] << 8;
    xlen -= 4;
    if (memcmp(temp, "ap", 2) == 0 && sub_len == 16) {
        good = 1;
        break;
    }
    if (xlen < sub_len)
        break;
    (void)lseek(log->fd, sub_len, SEEK_CUR);
    xlen -= sub_len;
}
if (!good) {
    log_clean(log);
    return NULL;
}

/* read in "ap" sub-field */
log->extra = lseek(log->fd, 0, SEEK_CUR);
if (read(log->fd, temp, 16) != 16) {
    log_clean(log);
    return NULL;
}
log->mark_off = make_off(temp);
log->last_off = make_off(temp + 8);

/* get crc, length of gzip file */
(void)lseek(log->fd, log->last_off, SEEK_SET);
if (read(log->fd, temp, 13) != 13 ||
    memcmp(temp, "\001\000\000\377\377", 5) != 0) {
    log_clean(log);
    return NULL;
}
log->crc = make_ulg(temp + 5);
log->len = make_ulg(temp + 9);

/* set up to write over empty last block */
(void)lseek(log->fd, log->last_off + 5, SEEK_SET);
log->stored = 0;
return (void *)log;
}

/* maximum amount to put in a stored block before starting a new one */

```

```

#define MAX_BLOCK 16384

/* write a block to a log object */
int gzlog_write(void *obj, char *data, size_t len)
{
    size_t some;
    unsigned char temp[5];
    gz_log *log;

    /* check object */
    log = (gz_log *)obj;
    if (log == NULL || log->id != GZLOGID)
        return 1;

    /* write stored blocks until all of the input is written */
    do {
        some = MAX_BLOCK - log->stored;
        if (some > len)
            some = len;
        if (write(log->fd, data, some) != some)
            return 1;
        log->crc = crc32(log->crc, data, some);
        log->len += some;
        len -= some;
        data += some;
        log->stored += some;

        /* if the stored block is full, end it and start another */
        if (log->stored == MAX_BLOCK) {
            (void)lseek(log->fd, log->last_off, SEEK_SET);
            temp[0] = 0;
            dice_ulg(log->stored + ((unsigned long)(~log->stored) << 16),
                    temp + 1);
            if (write(log->fd, temp, 5) != 5)
                return 1;
            log->last_off = lseek(log->fd, log->stored, SEEK_CUR);
            (void)lseek(log->fd, 5, SEEK_CUR);
            log->stored = 0;
        }
    } while (len);
    return 0;
}

/* recompress the remaining stored deflate data in place */
local int recomp(gz_log *log)
{
    z_stream strm;
    size_t len, max;
    unsigned char *in;
    unsigned char *out;
    unsigned char temp[16];

    /* allocate space and read it all in (it's around 1 MB) */
    len = log->last_off - log->mark_off;
    max = len + (len >> 12) + (len >> 14) + 11;
    out = malloc(max);
    if (out == NULL)
        return 1;
    in = malloc(len);
    if (in == NULL) {
        free(out);
        return 1;
    }
    (void)lseek(log->fd, log->mark_off, SEEK_SET);
    if (read(log->fd, in, len) != len) {
        free(in);
        free(out);
        return 1;
    }

    /* recompress in memory, decoding stored data as we go */
    /* note: this assumes that unsigned is four bytes or more */
    /* consider not making that assumption */
    strm.zalloc = Z_NULL;

```



```

    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    if (deflateInit2(&strm, Z_BEST_COMPRESSION, Z_DEFLATED, -15, 8,
        Z_DEFAULT_STRATEGY) != Z_OK) {
        free(in);
        free(out);
        return 1;
    }
    strm.next_in = in;
    strm.avail_out = max;
    strm.next_out = out;
    while (len >= 5) {
        if (strm.next_in[0] != 0)
            break;
        strm.avail_in = strm.next_in[1] + (strm.next_in[2] << 8);
        strm.next_in += 5;
        len -= 5;
        if (strm.avail_in != 0) {
            if (len < strm.avail_in)
                break;
            len -= strm.avail_in;
            (void)deflate(&strm, Z_NO_FLUSH);
            if (strm.avail_in != 0 || strm.avail_out == 0)
                break;
        }
    }
    (void)deflate(&strm, Z_SYNC_FLUSH);
    (void)deflateEnd(&strm);
    free(in);
    if (len != 0 || strm.avail_out == 0) {
        free(out);
        return 1;
    }

    /* overwrite stored data with compressed data */
    (void)lseek(log->fd, log->mark_off, SEEK_SET);
    len = max - strm.avail_out;
    if (write(log->fd, out, len) != len) {
        free(out);
        return 1;
    }
    free(out);

    /* write last empty block, crc, and length */
    log->mark_off = log->last_off = lseek(log->fd, 0, SEEK_CUR);
    temp[0] = 1;
    dice_ulg(0xffffL << 16, temp + 1);
    dice_ulg(log->crc, temp + 5);
    dice_ulg(log->len, temp + 9);
    if (write(log->fd, temp, 13) != 13)
        return 1;

    /* truncate file to discard remaining stored data and old trailer */
    ftruncate(log->fd, lseek(log->fd, 0, SEEK_CUR));

    /* update extra field to point to new last empty block */
    (void)lseek(log->fd, log->extra, SEEK_SET);
    dice_off(log->mark_off, temp);
    dice_off(log->last_off, temp + 8);
    if (write(log->fd, temp, 16) != 16)
        return 1;
    return 0;
}

/* maximum accumulation of stored blocks before compressing */
#define MAX_STORED 1048576

/* close log object */
int gzlog_close(void *obj)
{
    unsigned char temp[8];
    gz_log *log;

    /* check object */

```

```
log = (gz_log *)obj;
if (log == NULL || log->id != GZLOGID)
    return 1;

/* go to start of most recent block being written */
(void)lseek(log->fd, log->last_off, SEEK_SET);

/* if some stuff was put there, update block */
if (log->stored) {
    temp[0] = 0;
    dice_ulg(log->stored + ((unsigned long)(~log->stored) << 16),
             temp + 1);
    if (write(log->fd, temp, 5) != 5)
        return 1;
    log->last_off = lseek(log->fd, log->stored, SEEK_CUR);
}

/* write last block (empty) */
if (write(log->fd, "\001\000\000\377\377", 5) != 5)
    return 1;

/* write updated crc and uncompressed length */
dice_ulg(log->crc, temp);
dice_ulg(log->len, temp + 4);
if (write(log->fd, temp, 8) != 8)
    return 1;

/* put offset of that last block in gzip extra block */
(void)lseek(log->fd, log->extra + 8, SEEK_SET);
dice_off(log->last_off, temp);
if (write(log->fd, temp, 8) != 8)
    return 1;

/* if more than 1 MB stored, then time to compress it */
if (log->last_off - log->mark_off > MAX_STORED) {
    if (recomp(log))
        return 1;
}

/* unlock and close file */
log_clean(log);
return 0;
}
```

```
/* gzlog.h
Copyright (C) 2004 Mark Adler, all rights reserved
version 1.0, 26 Nov 2004

This software is provided 'as-is', without any express or implied
warranty. In no event will the author be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Mark Adler      madler@alumni.caltech.edu
*/

/*
The gzlog object allows writing short messages to a gzipped log file,
opening the log file locked for small bursts, and then closing it. The log
object works by appending stored data to the gzip file until 1 MB has been
accumulated. At that time, the stored data is compressed, and replaces the
uncompressed data in the file. The log file is truncated to its new size at
that time. After closing, the log file is always valid gzip file that can
decompressed to recover what was written.

A gzip header "extra" field contains two file offsets for appending. The
first points to just after the last compressed data. The second points to
the last stored block in the deflate stream, which is empty. All of the
data between those pointers is uncompressed.
*/

/* Open a gzlog object, creating the log file if it does not exist. Return
   NULL on error. Note that gzlog_open() could take a long time to return if
   there is difficulty in locking the file. */
void *gzlog_open(char *path);

/* Write to a gzlog object. Return non-zero on error. This function will
   simply write data to the file uncompressed. Compression of the data
   will not occur until gzlog_close() is called. It is expected that
   gzlog_write() is used for a short message, and then gzlog_close() is
   called. If a large amount of data is to be written, then the application
   should write no more than 1 MB at a time with gzlog_write() before
   calling gzlog_close() and then gzlog_open() again. */
int gzlog_write(void *log, char *data, size_t len);

/* Close a gzlog object. Return non-zero on error. The log file is locked
   until this function is called. This function will compress stored data
   at the end of the gzip file if at least 1 MB has been accumulated. Note
   that the file will not be a valid gzip file until this function completes.
   */
int gzlog_close(void *log);
```

```

/* zpipe.c: example of proper use of zlib's inflate() and deflate()
Not copyrighted -- provided to the public domain
Version 1.2 9 November 2004 Mark Adler */

/* Version history:
1.0 30 Oct 2004 First version
1.1 8 Nov 2004 Add void casting for unused return values
               Use switch statement for inflate() return values
1.2 9 Nov 2004 Add assertions to document zlib guarantees
1.3 6 Apr 2005 Remove incorrect assertion in inf()
*/

#include <stdio.h>
#include <string.h>
#include <assert.h>
#include "zlib.h"

#define CHUNK 16384

/* Compress from file source to file dest until EOF on source.
def() returns Z_OK on success, Z_MEM_ERROR if memory could not be
allocated for processing, Z_STREAM_ERROR if an invalid compression
level is supplied, Z_VERSION_ERROR if the version of zlib.h and the
version of the library linked do not match, or Z_ERRNO if there is
an error reading or writing the files. */
int def(FILE *source, FILE *dest, int level)
{
    int ret, flush;
    unsigned have;
    z_stream strm;
    char in[CHUNK];
    char out[CHUNK];

    /* allocate deflate state */
    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    ret = deflateInit(&strm, level);
    if (ret != Z_OK)
        return ret;

    /* compress until end of file */
    do {
        strm.avail_in = fread(in, 1, CHUNK, source);
        if (ferror(source)) {
            (void)deflateEnd(&strm);
            return Z_ERRNO;
        }
        flush = feof(source) ? Z_FINISH : Z_NO_FLUSH;
        strm.next_in = in;

        /* run deflate() on input until output buffer not full, finish
        compression if all of source has been read in */
        do {
            strm.avail_out = CHUNK;
            strm.next_out = out;
            ret = deflate(&strm, flush); /* no bad return value */
            assert(ret != Z_STREAM_ERROR); /* state not clobbered */
            have = CHUNK - strm.avail_out;
            if (fwrite(out, 1, have, dest) != have || ferror(dest)) {
                (void)deflateEnd(&strm);
                return Z_ERRNO;
            }
        } while (strm.avail_out == 0);
        assert(strm.avail_in == 0); /* all input will be used */

        /* done when last data in file processed */
    } while (flush != Z_FINISH);
    assert(ret == Z_STREAM_END); /* stream will be complete */

    /* clean up and return */
    (void)deflateEnd(&strm);
    return Z_OK;
}

```

```

/* Decompress from file source to file dest until stream ends or EOF.
   inf() returns Z_OK on success, Z_MEM_ERROR if memory could not be
   allocated for processing, Z_DATA_ERROR if the deflate data is
   invalid or incomplete, Z_VERSION_ERROR if the version of zlib.h and
   the version of the library linked do not match, or Z_ERRNO if there
   is an error reading or writing the files. */
int inf(FILE *source, FILE *dest)
{
    int ret;
    unsigned have;
    z_stream strm;
    char in[CHUNK];
    char out[CHUNK];

    /* allocate inflate state */
    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    strm.avail_in = 0;
    strm.next_in = Z_NULL;
    ret = inflateInit(&strm);
    if (ret != Z_OK)
        return ret;

    /* decompress until deflate stream ends or end of file */
    do {
        strm.avail_in = fread(in, 1, CHUNK, source);
        if (ferror(source)) {
            (void)inflateEnd(&strm);
            return Z_ERRNO;
        }
        if (strm.avail_in == 0)
            break;
        strm.next_in = in;

        /* run inflate() on input until output buffer not full */
        do {
            strm.avail_out = CHUNK;
            strm.next_out = out;
            ret = inflate(&strm, Z_NO_FLUSH);
            assert(ret != Z_STREAM_ERROR); /* state not clobbered */
            switch (ret) {
                case Z_NEED_DICT:
                    ret = Z_DATA_ERROR; /* and fall through */
                case Z_DATA_ERROR:
                case Z_MEM_ERROR:
                    (void)inflateEnd(&strm);
                    return ret;
            }
            have = CHUNK - strm.avail_out;
            if (fwrite(out, 1, have, dest) != have || ferror(dest)) {
                (void)inflateEnd(&strm);
                return Z_ERRNO;
            }
        } while (strm.avail_out == 0);

        /* done when inflate() says it's done */
    } while (ret != Z_STREAM_END);

    /* clean up and return */
    (void)inflateEnd(&strm);
    return ret == Z_STREAM_END ? Z_OK : Z_DATA_ERROR;
}

/* report a zlib or i/o error */
void zerr(int ret)
{
    fputs("zpipe: ", stderr);
    switch (ret) {
        case Z_ERRNO:
            if (ferror(stdin))
                fputs("error reading stdin\n", stderr);
            if (ferror(stdout))

```

```
        fputs("error writing stdout\n", stderr);
        break;
    case Z_STREAM_ERROR:
        fputs("invalid compression level\n", stderr);
        break;
    case Z_DATA_ERROR:
        fputs("invalid or incomplete deflate data\n", stderr);
        break;
    case Z_MEM_ERROR:
        fputs("out of memory\n", stderr);
        break;
    case Z_VERSION_ERROR:
        fputs("zlib version mismatch!\n", stderr);
    }
}

/* compress or decompress from stdin to stdout */
int main(int argc, char **argv)
{
    int ret;

    /* do compression if no arguments */
    if (argc == 1) {
        ret = def(stdin, stdout, Z_DEFAULT_COMPRESSION);
        if (ret != Z_OK)
            zerr(ret);
        return ret;
    }

    /* do decompression if -d specified */
    else if (argc == 2 && strcmp(argv[1], "-d") == 0) {
        ret = inf(stdin, stdout);
        if (ret != Z_OK)
            zerr(ret);
        return ret;
    }

    /* otherwise, report usage */
    else {
        fputs("zpipe usage: zpipe [-d] <source> dest\n", stderr);
        return 1;
    }
}
```

```

/* zran.c -- example of zlib/gzip stream indexing and random access
 * Copyright (C) 2005 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 * Version 1.0 29 May 2005 Mark Adler */

/* Illustrate the use of Z_BLOCK, inflatePrime(), and inflateSetDictionary()
for random access of a compressed file. A file containing a zlib or gzip
stream is provided on the command line. The compressed stream is decoded in
its entirety, and an index built with access points about every SPAN bytes
in the uncompressed output. The compressed file is left open, and can then
be read randomly, having to decompress on the average SPAN/2 uncompressed
bytes before getting to the desired block of data.

An access point can be created at the start of any deflate block, by saving
the starting file offset and bit of that block, and the 32K bytes of
uncompressed data that precede that block. Also the uncompressed offset of
that block is saved to provide a reference for locating a desired starting
point in the uncompressed stream. build_index() works by decompressing the
input zlib or gzip stream a block at a time, and at the end of each block
deciding if enough uncompressed data has gone by to justify the creation of
a new access point. If so, that point is saved in a data structure that
grows as needed to accommodate the points.

To use the index, an offset in the uncompressed data is provided, for which
the latest access point at or preceding that offset is located in the index.
The input file is positioned to the specified location in the index, and if
necessary the first few bits of the compressed data is read from the file.
inflate is initialized with those bits and the 32K of uncompressed data, and
the decompression then proceeds until the desired offset in the file is
reached. Then the decompression continues to read the desired uncompressed
data from the file.

Another approach would be to generate the index on demand. In that case,
requests for random access reads from the compressed data would try to use
the index, but if a read far enough past the end of the index is required,
then further index entries would be generated and added.

There is some fair bit of overhead to starting inflation for the random
access, mainly copying the 32K byte dictionary. So if small pieces of the
file are being accessed, it would make sense to implement a cache to hold
some lookahead and avoid many calls to extract() for small lengths.

Another way to build an index would be to use inflateCopy(). That would
not be constrained to have access points at block boundaries, but requires
more memory per access point, and also cannot be saved to file due to the
use of pointers in the state. The approach here allows for storage of the
index in a file.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "zlib.h"

#define local static

#define SPAN 1048576L /* desired distance between access points */
#define WINSIZE 32768U /* sliding window size */
#define CHUNK 16384 /* file input buffer size */

/* access point entry */
struct point {
    off_t out; /* corresponding offset in uncompressed data */
    off_t in; /* offset in input file of first full byte */
    int bits; /* number of bits (1-7) from byte at in - 1, or 0 */
    unsigned char window[WINSIZE]; /* preceding 32K of uncompressed data */
};

/* access point list */
struct access {
    int have; /* number of list entries filled in */
    int size; /* number of list entries allocated */
    struct point *list; /* allocated list */
};

```

```

/* Deallocate an index built by build_index() */
local void free_index(struct access *index)
{
    if (index != NULL) {
        free(index->list);
        free(index);
    }
}

/* Add an entry to the access point list. If out of memory, deallocate the
existing list and return NULL. */
local struct access *addpoint(struct access *index, int bits,
    off_t in, off_t out, unsigned left, unsigned char *window)
{
    struct point *next;

    /* if list is empty, create it (start with eight points) */
    if (index == NULL) {
        index = malloc(sizeof(struct access));
        if (index == NULL) return NULL;
        index->list = malloc(sizeof(struct point) << 3);
        if (index->list == NULL) {
            free(index);
            return NULL;
        }
        index->size = 8;
        index->have = 0;
    }

    /* if list is full, make it bigger */
    else if (index->have == index->size) {
        index->size <= 1;
        next = realloc(index->list, sizeof(struct point) * index->size);
        if (next == NULL) {
            free_index(index);
            return NULL;
        }
        index->list = next;
    }

    /* fill in entry and increment how many we have */
    next = index->list + index->have;
    next->bits = bits;
    next->in = in;
    next->out = out;
    if (left)
        memcpy(next->window, window + WINSIZE - left, left);
    if (left < WINSIZE)
        memcpy(next->window + left, window, WINSIZE - left);
    index->have++;

    /* return list, possibly reallocated */
    return index;
}

/* Make one entire pass through the compressed stream and build an index, with
access points about every span bytes of uncompressed output -- span is
chosen to balance the speed of random access against the memory requirements
of the list, about 32K bytes per access point. Note that data after the end
of the first zlib or gzip stream in the file is ignored. build_index()
returns the number of access points on success (>= 1), Z_MEM_ERROR for out
of memory, Z_DATA_ERROR for an error in the input file, or Z_ERRNO for a
file read error. On success, *built points to the resulting index. */
local int build_index(FILE *in, off_t span, struct access **built)
{
    int ret;
    off_t totin, totout;          /* our own total counters to avoid 4GB limit */
    off_t last;                  /* totout value of last access point */
    struct access *index;        /* access points being generated */
    z_stream strm;
    unsigned char input[CHUNK];
    unsigned char window[WINSIZE];

```



```

/* initialize inflate */
strm.zalloc = Z_NULL;
strm.zfree = Z_NULL;
strm.opaque = Z_NULL;
strm.avail_in = 0;
strm.next_in = Z_NULL;
ret = inflateInit2(&strm, 47);      /* automatic zlib or gzip decoding */
if (ret != Z_OK)
    return ret;

/* inflate the input, maintain a sliding window, and build an index -- this
   also validates the integrity of the compressed data using the check
   information at the end of the gzip or zlib stream */
totin = totout = last = 0;
index = NULL;                      /* will be allocated by first addpoint() */
strm.avail_out = 0;
do {
    /* get some compressed data from input file */
    strm.avail_in = fread(input, 1, CHUNK, in);
    if (ferror(in)) {
        ret = Z_ERRNO;
        goto build_index_error;
    }
    if (strm.avail_in == 0) {
        ret = Z_DATA_ERROR;
        goto build_index_error;
    }
    strm.next_in = input;

    /* process all of that, or until end of stream */
    do {
        /* reset sliding window if necessary */
        if (strm.avail_out == 0) {
            strm.avail_out = WINSIZE;
            strm.next_out = window;
        }

        /* inflate until out of input, output, or at end of block --
           update the total input and output counters */
        totin += strm.avail_in;
        totout += strm.avail_out;
        ret = inflate(&strm, Z_BLOCK);      /* return at end of block */
        totin -= strm.avail_in;
        totout -= strm.avail_out;
        if (ret == Z_NEED_DICT)
            ret = Z_DATA_ERROR;
        if (ret == Z_MEM_ERROR || ret == Z_DATA_ERROR)
            goto build_index_error;
        if (ret == Z_STREAM_END)
            break;

        /* if at end of block, consider adding an index entry (note that if
           data_type indicates an end-of-block, then all of the
           uncompressed data from that block has been delivered, and none
           of the compressed data after that block has been consumed,
           except for up to seven bits) -- the totout == 0 provides an
           entry point after the zlib or gzip header, and assures that the
           index always has at least one access point; we avoid creating an
           access point after the last block by checking bit 6 of data_type
           */
        if ((strm.data_type & 128) && !(strm.data_type & 64) &&
            (totout == 0 || totout - last > span)) {
            index = addpoint(index, strm.data_type & 7, totin,
                             totout, strm.avail_out, window);
            if (index == NULL) {
                ret = Z_MEM_ERROR;
                goto build_index_error;
            }
            last = totout;
        }
    } while (strm.avail_in != 0);
} while (ret != Z_STREAM_END);

/* clean up and return index (release unused entries in list) */

```

```

(void)inflateEnd(&strm);
index = realloc(index, sizeof(struct point) * index->have);
index->size = index->have;
*built = index;
return index->size;

/* return error */
build_index_error:
(void)inflateEnd(&strm);
if (index != NULL)
    free_index(index);
return ret;
}

/* Use the index to read len bytes from offset into buf, return bytes read or
negative for error (Z_DATA_ERROR or Z_MEM_ERROR). If data is requested past
the end of the uncompressed data, then extract() will return a value less
than len, indicating how much as actually read into buf. This function
should not return a data error unless the file was modified since the index
was generated. extract() may also return Z_ERRNO if there is an error on
reading or seeking the input file. */
local int extract(FILE *in, struct access *index, off_t offset,
    unsigned char *buf, int len)
{
    int ret, skip;
    z_stream strm;
    struct point *here;
    unsigned char input[CHUNK];
    unsigned char discard[WINSIZE];

    /* proceed only if something reasonable to do */
    if (len < 0)
        return 0;

    /* find where in stream to start */
    here = index->list;
    ret = index->have;
    while (--ret && here[1].out <= offset)
        here++;

    /* initialize file and inflate state to start there */
    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    strm.avail_in = 0;
    strm.next_in = Z_NULL;
    ret = inflateInit2(&strm, -15);          /* raw inflate */
    if (ret != Z_OK)
        return ret;
    ret = fseeko(in, here->in - (here->bits ? 1 : 0), SEEK_SET);
    if (ret == -1)
        goto extract_ret;
    if (here->bits) {
        ret = getc(in);
        if (ret == -1) {
            ret = ferror(in) ? Z_ERRNO : Z_DATA_ERROR;
            goto extract_ret;
        }
        (void)inflatePrime(&strm, here->bits, ret >> (8 - here->bits));
    }
    (void)inflateSetDictionary(&strm, here->window, WINSIZE);

    /* skip uncompressed bytes until offset reached, then satisfy request */
    offset -= here->out;
    strm.avail_in = 0;
    skip = 1;                                /* while skipping to offset */
    do {
        /* define where to put uncompressed data, and how much */
        if (offset == 0 && skip) {             /* at offset now */
            strm.avail_out = len;
            strm.next_out = buf;
            skip = 0;                          /* only do this once */
        }
        if (offset > WINSIZE) {                /* skip WINSIZE bytes */

```

```

        strm.avail_out = WINSIZE;
        strm.next_out = discard;
        offset -= WINSIZE;
    }
    else if (offset != 0) { /* last skip */
        strm.avail_out = (unsigned)offset;
        strm.next_out = discard;
        offset = 0;
    }

    /* uncompress until avail_out filled, or end of stream */
    do {
        if (strm.avail_in == 0) {
            strm.avail_in = fread(input, 1, CHUNK, in);
            if (ferror(in)) {
                ret = Z_ERRNO;
                goto extract_ret;
            }
            if (strm.avail_in == 0) {
                ret = Z_DATA_ERROR;
                goto extract_ret;
            }
            strm.next_in = input;
        }
        ret = inflate(&strm, Z_NO_FLUSH); /* normal inflate */
        if (ret == Z_NEED_DICT)
            ret = Z_DATA_ERROR;
        if (ret == Z_MEM_ERROR || ret == Z_DATA_ERROR)
            goto extract_ret;
        if (ret == Z_STREAM_END)
            break;
    } while (strm.avail_out != 0);

    /* if reach end of stream, then don't keep trying to get more */
    if (ret == Z_STREAM_END)
        break;

    /* do until offset reached and requested data read, or stream ends */
    } while (skip);

    /* compute number of uncompressed bytes read after offset */
    ret = skip ? 0 : len - strm.avail_out;

    /* clean up and return bytes read or error */
extract_ret:
    (void)inflateEnd(&strm);
    return ret;
}

/* Demonstrate the use of build_index() and extract() by processing the file
   provided on the command line, and the extracting 16K from about 2/3rds of
   the way through the uncompressed output, and writing that to stdout. */
int main(int argc, char **argv)
{
    int len;
    off_t offset;
    FILE *in;
    struct access *index;
    unsigned char buf[CHUNK];

    /* open input file */
    if (argc != 2) {
        fprintf(stderr, "usage: zran file.gz\n");
        return 1;
    }
    in = fopen(argv[1], "rb");
    if (in == NULL) {
        fprintf(stderr, "zran: could not open %s for reading\n", argv[1]);
        return 1;
    }

    /* build index */
    len = build_index(in, SPAN, &index);
    if (len < 0) {

```

```
    fclose(in);
    switch (len) {
    case Z_MEM_ERROR:
        fprintf(stderr, "zran: out of memory\n");
        break;
    case Z_DATA_ERROR:
        fprintf(stderr, "zran: compressed data error in %s\n", argv[1]);
        break;
    case Z_ERRNO:
        fprintf(stderr, "zran: read error on %s\n", argv[1]);
        break;
    default:
        fprintf(stderr, "zran: error %d while building index\n", len);
    }
    return 1;
}
fprintf(stderr, "zran: built index with %d access points\n", len);

/* use index by reading some bytes from an arbitrary offset */
offset = (index->list[index->have - 1].out << 1) / 3;
len = extract(in, index, offset, buf, CHUNK);
if (len < 0)
    fprintf(stderr, "zran: extraction failed: %s error\n",
            len == Z_MEM_ERROR ? "out of memory" : "input corrupted");
else {
    fwrite(buf, 1, len, stdout);
    fprintf(stderr, "zran: extracted %d bytes at %llu\n", len, offset);
}

/* clean up and exit */
free_index(index);
fclose(in);
return 0;
}
```

```
# Makefile for zlib
# Borland C++
# Last updated: 15-Mar-2003

# To use, do "make -fmakefile.bor"
# To compile in small model, set below: MODEL=s

# WARNING: the small model is supported but only for small values of
# MAX_WBITS and MAX_MEM_LEVEL. For example:
# -DMAX_WBITS=11 -DDEF_WBITS=11 -DMAX_MEM_LEVEL=3
# If you wish to reduce the memory requirements (default 256K for big
# objects plus a few K), you can add to the LOC macro below:
# -DMAX_MEM_LEVEL=7 -DMAX_WBITS=14
# See zconf.h for details about the memory requirements.

# ----- Turbo C++, Borland C++ -----

# Optional nonstandard preprocessor flags (e.g. -DMAX_MEM_LEVEL=7)
# should be added to the environment via "set LOCAL_ZLIB=DFOO" or added
# to the declaration of LOC here:
LOC = $(LOCAL_ZLIB)

# type for CPU required: 0: 8086, 1: 80186, 2: 80286, 3: 80386, etc.
CPU_TYP = 0

# memory model: one of s, m, c, l (small, medium, compact, large)
MODEL=l

# replace bcc with tcc for Turbo C++ 1.0, with bcc32 for the 32 bit version
CC=bcc
LD=bcc
AR=tlib

# compiler flags
# replace "-O2" by "-O -G -a -d" for Turbo C++ 1.0
CFLAGS=-O2 -Z -m$(MODEL) $(LOC)

LDFLAGS=-m$(MODEL) -f-

# variables
ZLIB_LIB = zlib_$(MODEL).lib

OBJ1 = Adler32.obj compress.obj crc32.obj deflate.obj gzio.obj inffast.obj
inftrees.obj trees.obj uncompress.obj zutil.obj
OBJ2 = inffast.obj inflate.obj inftrees.obj trees.obj uncompress.obj zutil.obj
OBJP1 = +Adler32.obj+compress.obj+crc32.obj+deflate.obj+gzio.obj+inffast.obj
+inftrees.obj+trees.obj+uncompress.obj+zutil.obj
OBJP2 = +inffast.obj+inflate.obj+inftrees.obj+trees.obj+uncompress.obj+zutil.obj

# targets
all: $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
$(CC) -c $(CFLAGS) *.c

adler32.obj: adler32.c zlib.h zconf.h
compress.obj: compress.c zlib.h zconf.h
crc32.obj: crc32.c zlib.h zconf.h crc32.h
deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h
gzio.obj: gzio.c zutil.h zlib.h zconf.h
inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
inffast.h
inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
inffast.h
```

```
inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h
trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h
uncompr.obj: uncompr.c zlib.h zconf.h
zutil.obj: zutil.c zutil.h zlib.h zconf.h
example.obj: example.c zlib.h zconf.h
minigzip.obj: minigzip.c zlib.h zconf.h

# the command line is cut to fit in the MS-DOS 128 byte limit:
$(ZLIB_LIB): $(OBJ1) $(OBJ2)
    -del $(ZLIB_LIB)
    $(AR) $(ZLIB_LIB) $(OBJP1)
    $(AR) $(ZLIB_LIB) $(OBJP2)

example.exe: example.obj $(ZLIB_LIB)
    $(LD) $(LDFLAGS) example.obj $(ZLIB_LIB)

minigzip.exe: minigzip.obj $(ZLIB_LIB)
    $(LD) $(LDFLAGS) minigzip.obj $(ZLIB_LIB)

test: example.exe minigzip.exe
    example
    echo hello world | minigzip | minigzip -d

clean:
    -del *.obj
    -del *.lib
    -del *.exe
    -del zlib_*.bak
    -del foo.gz
```

```
# Makefile for zlib. Modified for djgpp v2.0 by F. J. Donahoe, 3/15/96.
# Copyright (C) 1995-1998 Jean-loup Gailly.
# For conditions of distribution and use, see copyright notice in zlib.h

# To compile, or to compile and test, type:
#
#   make -fmakefile.dj2; make test -fmakefile.dj2
#
# To install libz.a, zconf.h and zlib.h in the djgpp directories, type:
#
#   make install -fmakefile.dj2
#
# after first defining LIBRARY_PATH and INCLUDE_PATH in djgpp.env as
# in the sample below if the pattern of the DJGPP distribution is to
# be followed. Remember that, while <sp>'es around <=> are ignored in
# makefiles, they are *not* in batch files or in djgpp.env.
# - - - - -
# [make]
# INCLUDE_PATH=%\>;INCLUDE_PATH%%\DJDIR%\include
# LIBRARY_PATH=%\>;LIBRARY_PATH%%\DJDIR%\lib
# BUTT=-m486
# - - - - -
# Alternately, these variables may be defined below, overriding the values
# in djgpp.env, as
# INCLUDE_PATH=c:\usr\include
# LIBRARY_PATH=c:\usr\lib

CC=gcc

#CFLAGS=-MMD -O
#CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
#CFLAGS=-MMD -g -DDEBUG
CFLAGS=-MMD -O3 $(BUTT) -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
        -Wstrict-prototypes -Wmissing-prototypes

# If cp.exe is available, replace "copy /Y" with "cp -fp" .
CP=copy /Y
# If gnu install.exe is available, replace $(CP) with ginstall.
INSTALL=$(CP)
# The default value of RM is "rm -f." If "rm.exe" is found, comment out:
RM=del
LDLIBS=-L. -lz
LD=$(CC) -s -o
LDSHARED=$(CC)

INCL=zlib.h zconf.h
LIBS=libz.a

AR=ar rcs

prefix=/usr/local
exec_prefix = $(prefix)

OBJS = adler32.o compress.o crc32.o gzio.o uncompr.o deflate.o trees.o \
        zutil.o inflate.o infback.o inftrees.o inffast.o

OBJA =
# to use the asm code: make OBJA=match.o

TEST_OBJS = example.o minigzip.o

all: example.exe minigzip.exe

check: test
test: all
        ./example
        echo hello world | .\minigzip | .\minigzip -d

%.o : %.c
        $(CC) $(CFLAGS) -c $< -o $@

libz.a: $(OBJS) $(OBJA)
        $(AR) $@ $(OBJS) $(OBJA)
```

```
% .exe : %.o $(LIBS)
        $(LD) $@ $< $(LDLIBS)

# INCLUDE_PATH and LIBRARY_PATH were set for [make] in djgpp.env .

.PHONY : uninstall clean

install: $(INCL) $(LIBS)
        -@if not exist $(INCLUDE_PATH)\nul mkdir $(INCLUDE_PATH)
        -@if not exist $(LIBRARY_PATH)\nul mkdir $(LIBRARY_PATH)
        $(INSTALL) zlib.h $(INCLUDE_PATH)
        $(INSTALL) zconf.h $(INCLUDE_PATH)
        $(INSTALL) libz.a $(LIBRARY_PATH)

uninstall:
        $(RM) $(INCLUDE_PATH)\zlib.h
        $(RM) $(INCLUDE_PATH)\zconf.h
        $(RM) $(LIBRARY_PATH)\libz.a

clean:
        $(RM) *.d
        $(RM) *.o
        $(RM) *.exe
        $(RM) libz.a
        $(RM) foo.gz

DEPS := $(wildcard *.d)
ifneq ($(DEPS),)
include $(DEPS)
endif
```



```
# Makefile for zlib. Modified for emx 0.9c by Chr. Spieler, 6/17/98.
# Copyright (C) 1995-1998 Jean-loup Gailly.
# For conditions of distribution and use, see copyright notice in zlib.h

# To compile, or to compile and test, type:
#
# make -fmakefile.emx; make test -fmakefile.emx
#

CC=gcc

#CFLAGS=-MMD -O
#CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
#CFLAGS=-MMD -g -DDEBUG
CFLAGS=-MMD -O3 $(BUTT) -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
        -Wstrict-prototypes -Wmissing-prototypes

# If cp.exe is available, replace "copy /Y" with "cp -fp" .
CP=copy /Y
# If gnu install.exe is available, replace $(CP) with ginstall.
INSTALL=$(CP)
# The default value of RM is "rm -f." If "rm.exe" is found, comment out:
RM=del
LDLIBS=-L. -lzlib
LD=$(CC) -s -o
LDLIBS=$(CC)

INCL=zlib.h zconf.h
LIBS=zlib.a

AR=ar rcs

prefix=/usr/local
exec_prefix = $(prefix)

OBJS = adler32.o compress.o crc32.o gzio.o uncompr.o deflate.o trees.o \
        zutil.o inflate.o infback.o inftrees.o inffast.o

TEST_OBJS = example.o minigzip.o

all: example.exe minigzip.exe

test: all
        ./example
        echo hello world | .\minigzip | .\minigzip -d

%.o : %.c
        $(CC) $(CFLAGS) -c $< -o $@

zlib.a: $(OBJS)
        $(AR) $@ $(OBJS)

%.exe : %.o $(LIBS)
        $(LD) $@ $< $(LDLIBS)

.PHONY : clean

clean:
        $(RM) *.d
        $(RM) *.o
        $(RM) *.exe
        $(RM) zlib.a
        $(RM) foo.gz

DEPS := $(wildcard *.d)
ifneq ($(DEPS),)
include $(DEPS)
endif
```

```
# Makefile for zlib
# Microsoft C 5.1 or later
# Last updated: 19-Mar-2003

# To use, do "make makefile.msc"
# To compile in small model, set below: MODEL=S

# If you wish to reduce the memory requirements (default 256K for big
# objects plus a few K), you can add to the LOC macro below:
#   -DMAX_MEM_LEVEL=7 -DMAX_WBITS=14
# See zconf.h for details about the memory requirements.

# ----- Microsoft C 5.1 and later -----

#   Optional nonstandard preprocessor flags (e.g. -DMAX_MEM_LEVEL=7)
#   should be added to the environment via "set LOCAL_ZLIB=DFOO" or added
#   to the declaration of LOC here:
LOC = $(LOCAL_ZLIB)

# Type for CPU required: 0: 8086, 1: 80186, 2: 80286, 3: 80386, etc.
CPU_TYP = 0

# Memory model: one of S, M, C, L (small, medium, compact, large)
MODEL=L

CC=c1
CFLAGS=-nologo -A$(MODEL) -G$(CPU_TYP) -W3 -Oait -Gs $(LOC)
#-Ox generates bad code with MSC 5.1
LIB_CFLAGS=-Zl $(CFLAGS)

LD=link
LDFLAGS=/noi/e/st:0x1500/noe/farcall/packcode
# "/farcall/packcode" are only useful for 'large code' memory models
# but should be a "no-op" for small code models.

# variables
ZLIB_LIB = zlib_$(MODEL).lib

OBJ1 = Adler32.obj compress.obj crc32.obj deflate.obj gzio.obj inffast.obj
OBJ2 = inflate.obj inftrees.obj trees.obj uncompress.obj zutil.obj

# targets
all: $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
    $(CC) -c $(LIB_CFLAGS) *.c

adler32.obj: adler32.c zlib.h zconf.h
compress.obj: compress.c zlib.h zconf.h
crc32.obj: crc32.c zlib.h zconf.h crc32.h
deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h
gzio.obj: gzio.c zutil.h zlib.h zconf.h
inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h inffix.h
inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h inffix.h
inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h
trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h
uncompress.obj: uncompress.c zlib.h zconf.h
```

zutil.obj: zutil.c zutil.h zlib.h zconf.h

example.obj: example.c zlib.h zconf.h
\$(CC) -c \$(CFLAGS) \$.c

minigzip.obj: minigzip.c zlib.h zconf.h
\$(CC) -c \$(CFLAGS) \$.c

the command line is cut to fit in the MS-DOS 128 byte limit:

\$(ZLIB_LIB): \$(OBJ1) \$(OBJ2)
 if exist \$(ZLIB_LIB) del \$(ZLIB_LIB)
 lib \$(ZLIB_LIB) \$(OBJ1);
 lib \$(ZLIB_LIB) \$(OBJ2);

example.exe: example.obj \$(ZLIB_LIB)
\$(LD) \$(LDFLAGS) example.obj,,, \$(ZLIB_LIB);

minigzip.exe: minigzip.obj \$(ZLIB_LIB)
\$(LD) \$(LDFLAGS) minigzip.obj,,, \$(ZLIB_LIB);

test: example.exe minigzip.exe
 example
 echo hello world | minigzip | minigzip -d

clean:
 -del *.obj
 -del *.lib
 -del *.exe
 -del *.map
 -del zlib_*.bak
 -del foo.gz

```
# Makefile for zlib
# Turbo C 2.01, Turbo C++ 1.01
# Last updated: 15-Mar-2003

# To use, do "make -fmakefile.tc"
# To compile in small model, set below: MODEL=s

# WARNING: the small model is supported but only for small values of
# MAX_WBITS and MAX_MEM_LEVEL. For example:
# -DMAX_WBITS=11 -DMAX_MEM_LEVEL=3
# If you wish to reduce the memory requirements (default 256K for big
# objects plus a few K), you can add to CFLAGS below:
# -DMAX_MEM_LEVEL=7 -DMAX_WBITS=14
# See zconf.h for details about the memory requirements.

# ----- Turbo C 2.01, Turbo C++ 1.01 -----
MODEL=1
CC=tcc
LD=tcc
AR=tlib
# CFLAGS=-O2 -G -Z -m$(MODEL) -DMAX_WBITS=11 -DMAX_MEM_LEVEL=3
CFLAGS=-O2 -G -Z -m$(MODEL)
LDFLAGS=-m$(MODEL) -f-

# variables
ZLIB_LIB = zlib_$(MODEL).lib

OBJ1 = Adler32.obj compress.obj crc32.obj deflate.obj gzio.obj infback.obj
OBJ2 = inffast.obj inflate.obj inftrees.obj trees.obj uncompr.obj zutil.obj
OBJP1 = +adler32.obj+compress.obj+crc32.obj+deflate.obj+gzio.obj+infback.obj
OBJP2 = +inffast.obj+inflate.obj+inftrees.obj+trees.obj+uncompr.obj+zutil.obj

# targets
all: $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
    $(CC) -c $(CFLAGS) $.c

adler32.obj: adler32.c zlib.h zconf.h
compress.obj: compress.c zlib.h zconf.h
crc32.obj: crc32.c zlib.h zconf.h crc32.h
deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h
gzio.obj: gzio.c zutil.h zlib.h zconf.h
infback.obj: infback.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h inffixed.h
inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h
inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h inffixed.h
inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h
trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h
uncompr.obj: uncompr.c zlib.h zconf.h
zutil.obj: zutil.c zutil.h zlib.h zconf.h
example.obj: example.c zlib.h zconf.h
minigzip.obj: minigzip.c zlib.h zconf.h

# the command line is cut to fit in the MS-DOS 128 byte limit:
$(ZLIB_LIB): $(OBJ1) $(OBJ2)
```

```
-del $(ZLIB_LIB)
$(AR) $(ZLIB_LIB) $(OBJP1)
$(AR) $(ZLIB_LIB) $(OBJP2)
```

```
example.exe: example.obj $(ZLIB_LIB)
$(LD) $(LDFLAGS) example.obj $(ZLIB_LIB)
```

```
minigzip.exe: minigzip.obj $(ZLIB_LIB)
$(LD) $(LDFLAGS) minigzip.obj $(ZLIB_LIB)
```

```
test: example.exe minigzip.exe
example
echo hello world | minigzip | minigzip -d
```

```
clean:
-del *.obj
-del *.lib
-del *.exe
-del zlib_*.bak
-del foo.gz
```

```

# Project:      zlib_1_03
# Patched for zlib 1.1.2 rw@shadow.org.uk 19980430
# test works out-of-the-box, installs 'somewhere' on demand

# Toolflags:
CCflags = -c -depend !Depend -IC: -g -throwback -DRISCOS -fah
C++flags = -c -depend !Depend -IC: -throwback
Linkflags = -aif -c++ -o $@
ObjAsmflags = -throwback -NoCache -depend !Depend
CMHGflags =
LibFileflags = -c -l -o $@
Squeezeflags = -o $@

# change the line below to where you want the library installed.
libdest = lib:zlib

# Final targets:
@.lib:      @.o.adler32 @.o.compress @.o.crc32 @.o.deflate @.o.gzio \
            @.o.infblock @.o.infcodes @.o.inffast @.o.inflate @.o.inftrees @.o.infutil @.o.tr
ees \
            @.o.uncompr @.o.zutil
LibFile $(LibFileflags) @.o.adler32 @.o.compress @.o.crc32 @.o.deflate \
            @.o.gzio @.o.infblock @.o.infcodes @.o.inffast @.o.inflate @.o.inftrees @.o.infut
il \
            @.o.trees @.o.uncompr @.o.zutil
test:
@.minigzip @.example @.lib
@copy @.lib @.libc A~C~DF~L~N~P~Q~RS~TV
@echo running tests: hang on.
@/@.minigzip -f -9 libc
@/@.minigzip -d libc-gz
@/@.minigzip -f -1 libc
@/@.minigzip -d libc-gz
@/@.minigzip -h -9 libc
@/@.minigzip -d libc-gz
@/@.minigzip -h -1 libc
@/@.minigzip -d libc-gz
@/@.minigzip -9 libc
@/@.minigzip -d libc-gz
@/@.minigzip -1 libc
@/@.minigzip -d libc-gz
@diff @.lib @.libc
@echo that should have reported ' @.lib and @.libc identical' if you have diff.
@/@.example @.fred @.fred
@echo that will have given lots of hello!'s.

@.minigzip: @.o.minigzip @.lib C:o.Stubs
Link $(Linkflags) @.o.minigzip @.lib C:o.Stubs
@.example: @.o.example @.lib C:o.Stubs
Link $(Linkflags) @.o.example @.lib C:o.Stubs

install: @.lib
        cd $(libdest)
        cd $(libdest).h
        @copy @.h.zlib $(libdest).h.zlib A~C~DF~L~N~P~Q~RS~TV
        @copy @.h.zconf $(libdest).h.zconf A~C~DF~L~N~P~Q~RS~TV
        @copy @.lib $(libdest).lib A~C~DF~L~N~P~Q~RS~TV
        @echo okay, installed zlib in $(libdest)

clean:; remove @.minigzip
        remove @.example
        remove @.libc
        -wipe @.o.* F~r~cV
        remove @.fred

# User-editable dependencies:
.c.o:
        cc $(ccflags) -o $@ $<

# Static dependencies:

# Dynamic dependencies:
o.example:      c.example
o.example:      h.zlib
o.example:      h.zconf

```

o.minigzip:	c.minigzip
o.minigzip:	h.zlib
o.minigzip:	h.zconf
o.adler32:	c.adler32
o.adler32:	h.zlib
o.adler32:	h.zconf
o.compress:	c.compress
o.compress:	h.zlib
o.compress:	h.zconf
o.crc32:	c.crc32
o.crc32:	h.zlib
o.crc32:	h.zconf
o.deflate:	c.deflate
o.deflate:	h.deflate
o.deflate:	h.zutil
o.deflate:	h.zlib
o.deflate:	h.zconf
o.gzio:	c.gzio
o.gzio:	h.zutil
o.gzio:	h.zlib
o.gzio:	h.zconf
o.infblock:	c.infblock
o.infblock:	h.zutil
o.infblock:	h.zlib
o.infblock:	h.zconf
o.infblock:	h.infblock
o.infblock:	h.inftrees
o.infblock:	h.infcodes
o.infblock:	h.infutil
o.infcodes:	c.infcodes
o.infcodes:	h.zutil
o.infcodes:	h.zlib
o.infcodes:	h.zconf
o.infcodes:	h.inftrees
o.infcodes:	h.infblock
o.infcodes:	h.infcodes
o.infcodes:	h.infutil
o.infcodes:	h.inffast
o.inffast:	c.inffast
o.inffast:	h.zutil
o.inffast:	h.zlib
o.inffast:	h.zconf
o.inffast:	h.inftrees
o.inffast:	h.infblock
o.inffast:	h.infcodes
o.inffast:	h.infutil
o.inffast:	h.inffast
o.inflate:	c.inflate
o.inflate:	h.zutil
o.inflate:	h.zlib
o.inflate:	h.zconf
o.inflate:	h.infblock
o.inftrees:	c.inftrees
o.inftrees:	h.zutil
o.inftrees:	h.zlib
o.inftrees:	h.zconf
o.inftrees:	h.inftrees
o.inftrees:	h.inffixed
o.infutil:	c.infutil
o.infutil:	h.zutil
o.infutil:	h.zlib
o.infutil:	h.zconf
o.infutil:	h.infblock
o.infutil:	h.inftrees
o.infutil:	h.infcodes
o.infutil:	h.infutil
o.trees:	c.trees
o.trees:	h.deflate
o.trees:	h.zutil
o.trees:	h.zlib
o.trees:	h.zconf
o.trees:	h.trees
o.uncompr:	c.uncompr
o.uncompr:	h.zlib

o.uncompr:	h.zconf
o.zutil:	c.zutil
o.zutil:	h.zutil
o.zutil:	h.zlib
o.zutil:	h.zconf

This directory contains files that have not been updated for zlib 1.2.x

(Volunteers are encouraged to help clean this up. Thanks.)

```
# descrip.mms: MMS description file for building zlib on VMS
# written by Martin P.J. Zinser <m.zinser@gsi.de>

cc_defs =
c_deb =

.ifdef __DECC__
pref = /prefix=all
.endif

OBS = Adler32.obj, compress.obj, crc32.obj, gzio.obj, uncompr.obj,\
      deflate.obj, trees.obj, zutil.obj, inflate.obj, infblock.obj,\
      inftrees.obj, infcodes.obj, infutil.obj, inffast.obj

CFLAGS= $(C_DEB) $(CC_DEFS) $(PREF)

all : example.exe minigzip.exe
      @ write sys$output " Example applications available"
libz.olb : libz.olb$(OBS)
      @ write sys$output " libz available"

example.exe : example.obj libz.olb
      link example,libz.olb/lib

minigzip.exe : minigzip.obj libz.olb
      link minigzip,libz.olb/lib,x11vms:xvmsutils.olb/lib

clean :
      delete *.obj;*,libz.olb;*

# Other dependencies.
adler32.obj : zutil.h zlib.h zconf.h
compress.obj : zlib.h zconf.h
crc32.obj : zutil.h zlib.h zconf.h
deflate.obj : deflate.h zutil.h zlib.h zconf.h
example.obj : zlib.h zconf.h
gzio.obj : zutil.h zlib.h zconf.h
infblock.obj : zutil.h zlib.h zconf.h infblock.h inftrees.h infcodes.h infutil.h
infcodes.obj : zutil.h zlib.h zconf.h inftrees.h infutil.h infcodes.h inffast.h
inffast.obj : zutil.h zlib.h zconf.h inftrees.h infutil.h inffast.h
inflate.obj : zutil.h zlib.h zconf.h infblock.h
inftrees.obj : zutil.h zlib.h zconf.h inftrees.h
infutil.obj : zutil.h zlib.h zconf.h inftrees.h infutil.h
minigzip.obj : zlib.h zconf.h
trees.obj : deflate.h zutil.h zlib.h zconf.h
uncompr.obj : zlib.h zconf.h
zutil.obj : zutil.h zlib.h zconf.h
```

See below some functions declarations for Visual Basic.

Frequently Asked Question:

Q: Each time I use the compress function I get the -5 error (not enough room in the output buffer).

A: Make sure that the length of the compressed buffer is passed by reference ("as any"), not by value ("as long"). Also check that before the call of compress this length is equal to the total size of the compressed buffer and not zero.

From: "Jon Caruana" <jon-net@usa.net>
Subject: Re: How to port zlib declares to vb?
Date: Mon, 28 Oct 1996 18:33:03 -0600

Got the answer! (I haven't had time to check this but it's what I got, and looks correct):

He has the following routines working:

```
compress
uncompress
gzopen
gzwrite
gzread
gzclose
```

Declares follow: (Quoted from Carlos Rios <c_rios@sonda.cl>, in Vb4 form)

```
#If Win16 Then 'Use Win16 calls.
Declare Function compress Lib "ZLIB.DLL" (ByVal compr As
String, comprLen As Any, ByVal buf As String, ByVal buflen
As Long) As Integer
Declare Function uncompress Lib "ZLIB.DLL" (ByVal uncompr
As String, uncomprLen As Any, ByVal compr As String, ByVal
lcompr As Long) As Integer
Declare Function gzopen Lib "ZLIB.DLL" (ByVal filePath As
String, ByVal mode As String) As Long
Declare Function gzread Lib "ZLIB.DLL" (ByVal file As
Long, ByVal uncompr As String, ByVal uncomprLen As Integer)
As Integer
Declare Function gzwrite Lib "ZLIB.DLL" (ByVal file As
Long, ByVal uncompr As String, ByVal uncomprLen As Integer)
As Integer
Declare Function gzclose Lib "ZLIB.DLL" (ByVal file As
Long) As Integer
#Else
Declare Function compress Lib "ZLIB32.DLL"
(ByVal compr As String, comprLen As Any, ByVal buf As
String, ByVal buflen As Long) As Integer
Declare Function uncompress Lib "ZLIB32.DLL"
(ByVal uncompr As String, uncomprLen As Any, ByVal compr As
String, ByVal lcompr As Long) As Long
Declare Function gzopen Lib "ZLIB32.DLL"
(ByVal file As String, ByVal mode As String) As Long
Declare Function gzread Lib "ZLIB32.DLL"
(ByVal file As Long, ByVal uncompr As String, ByVal
uncomprLen As Long) As Long
Declare Function gzwrite Lib "ZLIB32.DLL"
(ByVal file As Long, ByVal uncompr As String, ByVal
uncomprLen As Long) As Long
Declare Function gzclose Lib "ZLIB32.DLL"
(ByVal file As Long) As Long
#End If
```

-Jon Caruana
jon-net@usa.net
Microsoft Sitebuilder Network Level 1 Member - HTML Writer's Guild Member

Here is another example from Michael <michael_borgsys@hotmail.com> that he says conforms to the VB guidelines, and that solves the problem of not knowing the uncompressed size by storing it at the end of the file:

```

'Calling the functions:
'bracket meaning: <parameter> [optional] {Range of possible values}
'Call subCompressFile(<path with filename to compress> [, <path with
filename to write to>, [level of compression {1..9}]]])
'Call subUncompressFile(<path with filename to compress>)

Option Explicit
Private lngpvtPcnSml As Long 'Stores value for 'lngPercentSmaller'
Private Const SUCCESS As Long = 0
Private Const strFileExt As String = ".cpr"
Private Declare Function lngfncCpr Lib "zlib.dll" Alias "compress2" (ByRef
dest As Any, ByRef destLen As Any, ByRef src As Any, ByVal srcLen As Long,
ByVal level As Integer) As Long
Private Declare Function lngfncUcp Lib "zlib.dll" Alias "uncompress" (ByRef
dest As Any, ByRef destLen As Any, ByRef src As Any, ByVal srcLen As Long)
As Long

Public Sub subCompressFile(ByVal strargOriFilPth As String, Optional ByVal
strargCprFilPth As String, Optional ByVal intLvl As Integer = 9)
    Dim strCprPth As String
    Dim lngOriSiz As Long
    Dim lngCprSiz As Long
    Dim bytaryOri() As Byte
    Dim bytaryCpr() As Byte
    lngOriSiz = FileLen(strargOriFilPth)
    ReDim bytaryOri(lngOriSiz - 1)
    Open strargOriFilPth For Binary Access Read As #1
        Get #1, , bytaryOri()
    Close #1
    strCprPth = IIf(strargCprFilPth = "", strargOriFilPth, strargCprFilPth)
'Select file path and name
    strCprPth = strCprPth & IIf(Right(strCprPth, Len(strFileExt)) =
strFileExt, "", strFileExt) 'Add file extension if not exists
    lngCprSiz = (lngOriSiz * 1.01) + 12 'Compression needs temporary a bit
more space then original file size
    ReDim bytaryCpr(lngCprSiz - 1)
    If lngfncCpr(bytaryCpr(0), lngCprSiz, bytaryOri(0), lngOriSiz, intLvl) =
SUCCESS Then
        lngpvtPcnSml = (1# - (lngCprSiz / lngOriSiz)) * 100
        ReDim Preserve bytaryCpr(lngCprSiz - 1)
        Open strCprPth For Binary Access Write As #1
            Put #1, , bytaryCpr()
            Put #1, , lngOriSiz 'Add the the original size value to the end
(last 4 bytes)
        Close #1
    Else
        MsgBox "Compression error"
    End If
    Erase bytaryCpr
    Erase bytaryOri
End Sub

Public Sub subUncompressFile(ByVal strargFilPth As String)
    Dim bytaryCpr() As Byte
    Dim bytaryOri() As Byte
    Dim lngOriSiz As Long
    Dim lngCprSiz As Long
    Dim strOriPth As String
    lngCprSiz = FileLen(strargFilPth)
    ReDim bytaryCpr(lngCprSiz - 1)
    Open strargFilPth For Binary Access Read As #1
        Get #1, , bytaryCpr()
    Close #1
'Read the original file size value:
    lngOriSiz = bytaryCpr(lngCprSiz - 1) * (2 ^ 24) _
        + bytaryCpr(lngCprSiz - 2) * (2 ^ 16) _
        + bytaryCpr(lngCprSiz - 3) * (2 ^ 8) _
        + bytaryCpr(lngCprSiz - 4)
    ReDim Preserve bytaryCpr(lngCprSiz - 5) 'Cut of the original size value
    ReDim bytaryOri(lngOriSiz - 1)
    If lngfncUcp(bytaryOri(0), lngOriSiz, bytaryCpr(0), lngCprSiz) = SUCCESS
Then
        strOriPth = Left(strargFilPth, Len(strargFilPth) - Len(strFileExt))

```

```
        Open strOriPth For Binary Access Write As #1
        Put #1, , bytaryOri()
        Close #1
    Else
        MsgBox "Uncompression error"
    End If
    Erase bytaryCpr
    Erase bytaryOri
End Sub
Public Property Get lngPercentSmaller() As Long
    lngPercentSmaller = lngpvtPcnSml
End Property
```

```

# Makefile for zlib under OS/2 using GCC (PGCC)
# For conditions of distribution and use, see copyright notice in zlib.h

# To compile and test, type:
#   cp Makefile.os2 ..
#   cd ..
#   make -f Makefile.os2 test

# This makefile will build a static library z.lib, a shared library
# z.dll and a import library zdll.lib. You can use either z.lib or
# zdll.lib by specifying either -lz or -lzdll on gcc's command line

CC=gcc -Zomf -s

CFLAGS=-O6 -Wall
#CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
#CFLAGS=-g -DDEBUG
#CFLAGS=-O3 -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
#       -Wstrict-prototypes -Wmissing-prototypes

##### BUG WARNING: #####
## infcodes.c hits a bug in pgcc-1.0, so you have to use either
## -O# where # <= 4 or one of (-fno-omit-frame-pointer or -fno-force-mem)
## This bug is reportedly fixed in pgcc >1.0, but this was not tested
CFLAGS+=-fno-force-mem

LDFLAGS=-s -L. -lzdll -Zcrt.dll
LD_SHARED=${CC} -s -Zomf -Zdll -Zcrt.dll

VER=1.1.0
ZLIB=z.lib
SHARED_LIB=z.dll
SHARED_LIB_IMP=zdll.lib
LIBS=${ZLIB} ${SHARED_LIB} ${SHARED_LIB_IMP}

AR=emxomfar cr
IMPLIB=emximp
RANLIB=echo
TAR=tar
SHELL=bash

prefix=/usr/local
exec_prefix = ${prefix}

OBJS = adler32.o compress.o crc32.o gzio.o uncompress.o deflate.o trees.o \
       zutil.o inflate.o inffast.o inftrees.o infcodes.o infutil.o inffast.o

TEST_OBJS = example.o minigzip.o

DISTFILES = README INDEX ChangeLog configure Make*[a-z0-9] *.ch descrip.mms \
algorithm.txt zlib.3 msdos/Make*[a-z0-9] msdos/zlib.def msdos/zlib.rc \
nt/Makefile.nt nt/zlib.dnt contrib/README.contrib contrib/*.txt \
contrib/asm386/*.asm contrib/asm386/*.c \
contrib/asm386/*.bat contrib/asm386/zlibvc.d?? contrib/iostream/*.cpp \
contrib/iostream/*.h contrib/iostream2/*.h contrib/iostream2/*.cpp \
contrib/untgz/Makefile contrib/untgz/*.c contrib/untgz/*.w32

all: example.exe minigzip.exe

test: all
    @LD_LIBRARY_PATH=.:${LD_LIBRARY_PATH} ; export LD_LIBRARY_PATH; \
    echo hello world | ./minigzip | ./minigzip -d || \
    echo '*** minigzip test FAILED ***' ; \
    if ./example; then \
    echo '*** zlib test OK ***' ; \
    else \
    echo '*** zlib test FAILED ***' ; \
    fi

$(ZLIB): $(OBJS)
    $(AR) $@ $(OBJS)
    -@ ($(RANLIB) $@ || true) >/dev/null 2>&1

$(SHARED_LIB): $(OBJS) os2/z.def

```

```
$(LDSHARED) -o $@ $^
```

```
$(SHAREDLIBIMP): os2/z.def
```

```
$(IMPLIB) -o $@ $^
```

```
example.exe: example.o $(LIBS)
```

```
$(CC) $(CFLAGS) -o $@ example.o $(LDFLAGS)
```

```
minigzip.exe: minigzip.o $(LIBS)
```

```
$(CC) $(CFLAGS) -o $@ minigzip.o $(LDFLAGS)
```

clean:

```
rm -f *.o *~ example minigzip libz.a libz.so* foo.gz
```

distclean:

```
clean
```

zip:

```
mv Makefile Makefile~; cp -p Makefile.in Makefile
```

```
rm -f test.c ztest*.c
```

```
v='sed -n -e 's/\\.g' -e '/VERSION "/s/.*"(.*)".*/l/p' < zlib.h';\
```

```
zip -ul9 zlib$$v $(DISTFILES)
```

```
mv Makefile~ Makefile
```

dist:

```
mv Makefile Makefile~; cp -p Makefile.in Makefile
```

```
rm -f test.c ztest*.c
```

```
d=zlib-'sed -n '/VERSION "/s/.*"(.*)".*/l/p' < zlib.h';\
```

```
rm -f $$d.tar.gz; \
```

```
if test ! -d ../$$d; then rm -f ../$$d; ln -s 'pwd' ../$$d; fi; \
```

```
files=""; \
```

```
for f in $(DISTFILES); do files="$$files $$d/$$f"; done; \
```

```
cd ..; \
```

```
GZIP=-9 $(TAR) chofz $$d/$$d.tar.gz $$files; \
```

```
if test ! -d $$d; then rm -f $$d; fi
```

```
mv Makefile~ Makefile
```

tags:

```
etags *.ch
```

depend:

```
makedepend -- $(CFLAGS) -- *.ch
```

```
# DO NOT DELETE THIS LINE -- make depend depends on it.
```

```
adler32.o: zlib.h zconf.h
```

```
compress.o: zlib.h zconf.h
```

```
crc32.o: zlib.h zconf.h
```

```
deflate.o: deflate.h zutil.h zlib.h zconf.h
```

```
example.o: zlib.h zconf.h
```

```
gzio.o: zutil.h zlib.h zconf.h
```

```
infblock.o: infblock.h inftrees.h infcodes.h infutil.h zutil.h zlib.h zconf.h
```

```
infcodes.o: zutil.h zlib.h zconf.h
```

```
infcodes.o: inftrees.h infblock.h infcodes.h infutil.h inffast.h
```

```
inffast.o: zutil.h zlib.h zconf.h inftrees.h
```

```
inffast.o: infblock.h infcodes.h infutil.h inffast.h
```

```
inflate.o: zutil.h zlib.h zconf.h infblock.h
```

```
inftrees.o: zutil.h zlib.h zconf.h inftrees.h
```

```
infutil.o: zutil.h zlib.h zconf.h infblock.h inftrees.h infcodes.h infutil.h
```

```
minigzip.o: zlib.h zconf.h
```

```
trees.o: deflate.h zutil.h zlib.h zconf.h trees.h
```

```
uncompr.o: zlib.h zconf.h
```

```
zutil.o: zutil.h zlib.h zconf.h
```

```
;
; Slightly modified version of ../nt/zlib.dnt :-)
;
```

```
LIBRARY      Z
DESCRIPTION   "Zlib compression library for OS/2"
CODE          PRELOAD MOVEABLE DISCARDABLE
DATA          PRELOAD MOVEABLE MULTIPLE
```

```
EXPORTS
    Adler32
    compress
    crc32
    deflate
    deflateCopy
    deflateEnd
    deflateInit2_
    deflateInit_
    deflateParams
    deflateReset
    deflateSetDictionary
    gzclose
    gzdopen
    gzerror
    gzflush
    gzopen
    gzread
    gzwrite
    inflate
    inflateEnd
    inflateInit2_
    inflateInit_
    inflateReset
    inflateSetDictionary
    inflateSync
    uncompress
    zlibVersion
    gzprintf
    gzputc
    gzgetc
    gzseek
    gzrewind
    gztell
    gzeof
    gzsetparams
    zError
    inflateSyncPoint
    get_crc_table
    compress2
    gzputs
    gzgets
```


This directory contains project files for building zlib under various Integrated Development Environments (IDE).

If you wish to submit a new project to this directory, you should comply to the following requirements. Otherwise (e.g. if you wish to integrate a custom piece of code that changes the zlib interface or its behavior), please consider submitting the project to the contrib directory.

Requirements

=====

- The project must build zlib using the source files from the official zlib source distribution, exclusively.
- If the project produces redistributable builds (e.g. shared objects or DLL files), these builds must be compatible to those produced by makefiles, if such makefiles exist in the zlib distribution. In particular, if the project produces a DLL build for the Win32 platform, this build must comply to the officially-ammended Win32 DLL Application Binary Interface (ABI), described in win32/DLL_FAQ.txt.
- The project may provide additional build targets, which depend on 3rd-party (unofficially-supported) software, present in the contrib directory. For example, it is possible to provide an "ASM build", besides the officially-supported build, and have ASM source files among its dependencies.
- If there are significant differences between the project files created by different versions of an IDE (e.g. Visual C++ 6.0 vs. 7.0), the name of the project directory should contain the version number of the IDE for which the project is intended (e.g. "visualc6" for Visual C++ 6.0, or "visualc7" for Visual C++ 7.0 and 7.1).

Current projects

=====

visualc6/ by Simon-Pierre Cadieux <methodex@methodex.ca>
 and Cosmin Truta <cosmint@cs.ubbcluj.ro>
 Project for Microsoft Visual C++ 6.0

Microsoft Developer Studio Project Files, Format Version 6.00 for zlib.

Copyright (C) 2000-2004 Simon-Pierre Cadieux.

Copyright (C) 2004 Cosmin Truta.

For conditions of distribution and use, see copyright notice in zlib.h.

This project builds the zlib binaries as follows:

* Win32_DLL_Release\zlib1.dll	DLL build
* Win32_DLL_Debug\zlib1d.dll	DLL build (debug version)
* Win32_DLL_ASM_Release\zlib1.dll	DLL build using ASM code
* Win32_DLL_ASM_Debug\zlib1d.dll	DLL build using ASM code (debug version)
* Win32_LIB_Release\zlib.lib	static build
* Win32_LIB_Debug\zlibd.lib	static build (debug version)
* Win32_LIB_ASM_Release\zlib.lib	static build using ASM code
* Win32_LIB_ASM_Debug\zlibd.lib	static build using ASM code (debug version)

For more information regarding the DLL builds, please see the DLL FAQ in ..\..\win32\DLL_FAQ.txt.

To build and test:

- 1) On the main menu, select "File | Open Workspace".
Open "zlib.dsw".
- 2) Select "Build | Set Active Configuration".
Choose the configuration you wish to build.
- 3) Select "Build | Clean".
- 4) Select "Build | Build ... (F7)". Ignore warning messages about not being able to find certain include files (e.g. alloc.h).
- 5) If you built one of the sample programs (example or minigzip), select "Build | Execute ... (Ctrl+F5)".

To use:

- 1) Select "Project | Settings (Alt+F7)".
Make note of the configuration names used in your project.
Usually, these names are "Win32 Release" and "Win32 Debug".
- 2) In the Workspace window, select the "FileView" tab.
Right-click on the root item "Workspace '...'".
Select "Insert Project into Workspace".
Switch on the checkbox "Dependency of:", and select the name of your project. Open "zlib.dsp".
- 3) Select "Build | Configurations".
For each configuration of your project:
 - 3.1) Choose the zlib configuration you wish to use.
 - 3.2) Click on "Add".
 - 3.3) Set the new zlib configuration name to the name used by the configuration from the current iteration.
- 4) Select "Build | Set Active Configuration".
Choose the configuration you wish to build.
- 5) Select "Build | Build ... (F7)".
- 6) If you built an executable program, select
"Build | Execute ... (Ctrl+F5)".

Note:

To build the ASM-enabled code, you need Microsoft Assembler (ML.EXE). You can get it by downloading and installing the latest Processor Pack for Visual C++ 6.0.

```
# Microsoft Developer Studio Project File - Name="example" - Package Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version 6.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Console Application" 0x0103

CFG=example - Win32 LIB Debug
!MESSAGE This is not a valid makefile. To build this project using NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "example.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For example:
!MESSAGE
!MESSAGE NMAKE /f "example.mak" CFG="example - Win32 LIB Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "example - Win32 DLL Release" (based on "Win32 (x86) Console Application")
!MESSAGE "example - Win32 DLL Debug" (based on "Win32 (x86) Console Application")
!MESSAGE "example - Win32 DLL ASM Release" (based on "Win32 (x86) Console Application")
!MESSAGE "example - Win32 DLL ASM Debug" (based on "Win32 (x86) Console Application")
!MESSAGE "example - Win32 LIB Release" (based on "Win32 (x86) Console Application")
!MESSAGE "example - Win32 LIB Debug" (based on "Win32 (x86) Console Application")
!MESSAGE "example - Win32 LIB ASM Release" (based on "Win32 (x86) Console Application")
!MESSAGE "example - Win32 LIB ASM Debug" (based on "Win32 (x86) Console Application")
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""
# PROP Scc_LocalPath ""
CPP=cl.exe
RSC=rc.exe

!IF "$(CFG)" == "example - Win32 DLL Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "example__Win32_DLL_Release"
# PROP BASE Intermediate_Dir "example__Win32_DLL_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_DLL_Release"
# PROP Intermediate_Dir "Win32_DLL_Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /machine:I386
# ADD LINK32 /nologo /subsystem:console /machine:I386

!ELSEIF "$(CFG)" == "example - Win32 DLL Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "example__Win32_DLL_Debug"
# PROP BASE Intermediate_Dir "example__Win32_DLL_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_DLL_Debug"
```

```
# PROP Intermediate_Dir "Win32_DLL_Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /debug /machine:I386 /pdbtype:sept
# ADD LINK32 /nologo /subsystem:console /debug /machine:I386 /pdbtype:sept

!ELSEIF "$(CFG)" == "example - Win32 DLL ASM Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "example____Win32_DLL_ASM_Release"
# PROP BASE Intermediate_Dir "example____Win32_DLL_ASM_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_DLL_ASM_Release"
# PROP Intermediate_Dir "Win32_DLL_ASM_Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /machine:I386
# ADD LINK32 /nologo /subsystem:console /machine:I386

!ELSEIF "$(CFG)" == "example - Win32 DLL ASM Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "example____Win32_DLL_ASM_Debug"
# PROP BASE Intermediate_Dir "example____Win32_DLL_ASM_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_DLL_ASM_Debug"
# PROP Intermediate_Dir "Win32_DLL_ASM_Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /debug /machine:I386 /pdbtype:sept
# ADD LINK32 /nologo /subsystem:console /debug /machine:I386 /pdbtype:sept
```

```
!ELSEIF "$(CFG)" == "example - Win32 LIB Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "example__Win32_LIB_Release"
# PROP BASE Intermediate_Dir "example__Win32_LIB_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_LIB_Release"
# PROP Intermediate_Dir "Win32_LIB_Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /machine:I386
# ADD LINK32 /nologo /subsystem:console /machine:I386

!ELSEIF "$(CFG)" == "example - Win32 LIB Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "example__Win32_LIB_Debug"
# PROP BASE Intermediate_Dir "example__Win32_LIB_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_LIB_Debug"
# PROP Intermediate_Dir "Win32_LIB_Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /debug /machine:I386 /pdbtype:sept
# ADD LINK32 /nologo /subsystem:console /debug /machine:I386 /pdbtype:sept

!ELSEIF "$(CFG)" == "example - Win32 LIB ASM Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "example__Win32_LIB_ASM_Release"
# PROP BASE Intermediate_Dir "example__Win32_LIB_ASM_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_LIB_ASM_Release"
# PROP Intermediate_Dir "Win32_LIB_ASM_Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "NDEBUG"
```

```
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /machine:I386
# ADD LINK32 /nologo /subsystem:console /machine:I386

!ELSEIF "$(CFG)" == "example - Win32 LIB ASM Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "example___Win32_LIB_ASM_Debug"
# PROP BASE Intermediate_Dir "example___Win32_LIB_ASM_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_LIB_ASM_Debug"
# PROP Intermediate_Dir "Win32_LIB_ASM_Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /debug /machine:I386 /pdbtype:sept
# ADD LINK32 /nologo /subsystem:console /debug /machine:I386 /pdbtype:sept

!ENDIF

# Begin Target

# Name "example - Win32 DLL Release"
# Name "example - Win32 DLL Debug"
# Name "example - Win32 DLL ASM Release"
# Name "example - Win32 DLL ASM Debug"
# Name "example - Win32 LIB Release"
# Name "example - Win32 LIB Debug"
# Name "example - Win32 LIB ASM Release"
# Name "example - Win32 LIB ASM Debug"
# Begin Group "Source Files"

# PROP Default_Filter "cpp;c;cx;c;rc;def;r;odl;idl;hpp;bat"
# Begin Source File

SOURCE=..\..\example.c
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h;hpp;hxx;hm;inl"
# Begin Source File

SOURCE=..\..\zconf.h
# End Source File
# Begin Source File

SOURCE=..\..\zlib.h
# End Source File
# End Group
# End Target
# End Project
```

```
# Microsoft Developer Studio Project File - Name="minigzip" - Package Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version 6.00
# ** DO NOT EDIT **

# TARGETTYPE "Win32 (x86) Console Application" 0x0103

CFG=minigzip - Win32 LIB Debug
!MESSAGE This is not a valid makefile. To build this project using NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "minigzip.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For example:
!MESSAGE
!MESSAGE NMAKE /f "minigzip.mak" CFG="minigzip - Win32 LIB Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "minigzip - Win32 DLL Release" (based on "Win32 (x86) Console Application")
!MESSAGE "minigzip - Win32 DLL Debug" (based on "Win32 (x86) Console Application")
!MESSAGE "minigzip - Win32 DLL ASM Release" (based on "Win32 (x86) Console Application")
!MESSAGE "minigzip - Win32 DLL ASM Debug" (based on "Win32 (x86) Console Application")
!MESSAGE "minigzip - Win32 LIB Release" (based on "Win32 (x86) Console Application")
!MESSAGE "minigzip - Win32 LIB Debug" (based on "Win32 (x86) Console Application")
!MESSAGE "minigzip - Win32 LIB ASM Release" (based on "Win32 (x86) Console Application")
!MESSAGE "minigzip - Win32 LIB ASM Debug" (based on "Win32 (x86) Console Application")
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""
# PROP Scc_LocalPath ""
CPP=cl.exe
RSC=rc.exe

!IF "$(CFG)" == "minigzip - Win32 DLL Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "minigzip__Win32_DLL_Release"
# PROP BASE Intermediate_Dir "minigzip__Win32_DLL_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_DLL_Release"
# PROP Intermediate_Dir "Win32_DLL_Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /machine:I386
# ADD LINK32 /nologo /subsystem:console /machine:I386

!ELSEIF "$(CFG)" == "minigzip - Win32 DLL Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "minigzip__Win32_DLL_Debug"
# PROP BASE Intermediate_Dir "minigzip__Win32_DLL_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_DLL_Debug"
```

```
# PROP Intermediate_Dir "Win32_DLL_Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /debug /machine:I386 /pdbtype:sept
# ADD LINK32 /nologo /subsystem:console /debug /machine:I386 /pdbtype:sept

!ELSEIF "$(CFG)" == "minigzip - Win32 DLL ASM Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "minigzip___Win32_DLL_ASM_Release"
# PROP BASE Intermediate_Dir "minigzip___Win32_DLL_ASM_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_DLL_ASM_Release"
# PROP Intermediate_Dir "Win32_DLL_ASM_Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /machine:I386
# ADD LINK32 /nologo /subsystem:console /machine:I386

!ELSEIF "$(CFG)" == "minigzip - Win32 DLL ASM Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "minigzip___Win32_DLL_ASM_Debug"
# PROP BASE Intermediate_Dir "minigzip___Win32_DLL_ASM_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_DLL_ASM_Debug"
# PROP Intermediate_Dir "Win32_DLL_ASM_Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /debug /machine:I386 /pdbtype:sept
# ADD LINK32 /nologo /subsystem:console /debug /machine:I386 /pdbtype:sept
```



```
!ELSEIF "$(CFG)" == "minigzip - Win32 LIB Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "minigzip__Win32_LIB_Release"
# PROP BASE Intermediate_Dir "minigzip__Win32_LIB_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_LIB_Release"
# PROP Intermediate_Dir "Win32_LIB_Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /machine:I386
# ADD LINK32 /nologo /subsystem:console /machine:I386

!ELSEIF "$(CFG)" == "minigzip - Win32 LIB Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "minigzip__Win32_LIB_Debug"
# PROP BASE Intermediate_Dir "minigzip__Win32_LIB_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_LIB_Debug"
# PROP Intermediate_Dir "Win32_LIB_Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /debug /machine:I386 /pdbtype:sept
# ADD LINK32 /nologo /subsystem:console /debug /machine:I386 /pdbtype:sept

!ELSEIF "$(CFG)" == "minigzip - Win32 LIB ASM Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "minigzip__Win32_LIB_ASM_Release"
# PROP BASE Intermediate_Dir "minigzip__Win32_LIB_ASM_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_LIB_ASM_Release"
# PROP Intermediate_Dir "Win32_LIB_ASM_Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "NDEBUG"
```

```
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /machine:I386
# ADD LINK32 /nologo /subsystem:console /machine:I386

!ELSEIF "$(CFG)" == "minigzip - Win32 LIB ASM Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "minigzip__Win32_LIB_ASM_Debug"
# PROP BASE Intermediate_Dir "minigzip__Win32_LIB_ASM_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_LIB_ASM_Debug"
# PROP Intermediate_Dir "Win32_LIB_ASM_Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:
console /debug /machine:I386 /pdbtype:sept
# ADD LINK32 /nologo /subsystem:console /debug /machine:I386 /pdbtype:sept

!ENDIF

# Begin Target

# Name "minigzip - Win32 DLL Release"
# Name "minigzip - Win32 DLL Debug"
# Name "minigzip - Win32 DLL ASM Release"
# Name "minigzip - Win32 DLL ASM Debug"
# Name "minigzip - Win32 LIB Release"
# Name "minigzip - Win32 LIB Debug"
# Name "minigzip - Win32 LIB ASM Release"
# Name "minigzip - Win32 LIB ASM Debug"
# Begin Group "Source Files"

# PROP Default_Filter "cpp;c;cx;c;rc;def;r;odl;idl;hpj;bat"
# Begin Source File

SOURCE=..\..\minigzip.c
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h;hpp;hxx;hm;inl"
# Begin Source File

SOURCE=..\..\zconf.h
# End Source File
# Begin Source File

SOURCE=..\..\zlib.h
# End Source File
# End Group
# End Target
# End Project
```

```
# Microsoft Developer Studio Project File - Name="zlib" - Package Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version 6.00
# ** DO NOT EDIT **

# TARGETTYPE "Win32 (x86) Dynamic-Link Library" 0x0102
# TARGETTYPE "Win32 (x86) Static Library" 0x0104

CFG=zlib - Win32 LIB Debug
!MESSAGE This is not a valid makefile. To build this project using NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "zlib.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For example:
!MESSAGE
!MESSAGE NMAKE /f "zlib.mak" CFG="zlib - Win32 LIB Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "zlib - Win32 DLL Release" (based on "Win32 (x86) Dynamic-Link Library")
!MESSAGE "zlib - Win32 DLL Debug" (based on "Win32 (x86) Dynamic-Link Library")
!MESSAGE "zlib - Win32 DLL ASM Release" (based on "Win32 (x86) Dynamic-Link Library")
!MESSAGE "zlib - Win32 DLL ASM Debug" (based on "Win32 (x86) Dynamic-Link Library")
!MESSAGE "zlib - Win32 LIB Release" (based on "Win32 (x86) Static Library")
!MESSAGE "zlib - Win32 LIB Debug" (based on "Win32 (x86) Static Library")
!MESSAGE "zlib - Win32 LIB ASM Release" (based on "Win32 (x86) Static Library")
!MESSAGE "zlib - Win32 LIB ASM Debug" (based on "Win32 (x86) Static Library")
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""
# PROP Scc_LocalPath ""

!IF "$(CFG)" == "zlib - Win32 DLL Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "zlib__Win32_DLL_Release"
# PROP BASE Intermediate_Dir "zlib__Win32_DLL_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_DLL_Release"
# PROP Intermediate_Dir "Win32_DLL_Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
CPP=cl.exe
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX /Yc /Yu
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT CPP /YX /Yc /Yu
MTL=midl.exe
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /win32
RSC=rc.exe
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /dll /machi
ne:I386
# ADD LINK32 /nologo /dll /machine:I386 /out:"Win32_DLL_Release\zlib1.dll"

!ELSEIF "$(CFG)" == "zlib - Win32 DLL Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "zlib__Win32_DLL_Debug"
# PROP BASE Intermediate_Dir "zlib__Win32_DLL_Debug"
```

```
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_DLL_Debug"
# PROP Intermediate_Dir "Win32_DLL_Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
CPP=cl.exe
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX /Yc /Yu
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT CPP /YX /Yc /Yu
MTL=midl.exe
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /win32
RSC=rc.exe
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /dll /debug
/machine:I386 /pdbtype:sept
# ADD LINK32 /nologo /dll /debug /machine:I386 /out:"Win32_DLL_Debug\zlib1d.dll" /pdbtype
:sept

!ELSEIF "$(CFG)" == "zlib - Win32 DLL ASM Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "zlib___Win32_DLL_ASM_Release"
# PROP BASE Intermediate_Dir "zlib___Win32_DLL_ASM_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_DLL_ASM_Release"
# PROP Intermediate_Dir "Win32_DLL_ASM_Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
CPP=cl.exe
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX /Yc /Yu
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /D "ASMV" /D "ASMINF" /FD /c
# SUBTRACT CPP /YX /Yc /Yu
MTL=midl.exe
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /win32
RSC=rc.exe
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /dll /machi
ne:I386
# ADD LINK32 /nologo /dll /machine:I386 /out:"Win32_DLL_ASM_Release\zlib1.dll"

!ELSEIF "$(CFG)" == "zlib - Win32 DLL ASM Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "zlib___Win32_DLL_ASM_Debug"
# PROP BASE Intermediate_Dir "zlib___Win32_DLL_ASM_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_DLL_ASM_Debug"
# PROP Intermediate_Dir "Win32_DLL_ASM_Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
```

```
CPP=cl.exe
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX /Yc /Yu
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /D "ASMV" /D "ASMINF" /FD /
GZ /c
# SUBTRACT CPP /YX /Yc /Yu
MTL=midl.exe
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /win32
RSC=rc.exe
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.li
b shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /dll /debug
/machine:I386 /pdbtype:sept
# ADD LINK32 /nologo /dll /debug /machine:I386 /out:"Win32_DLL_ASM_Debug\zlib1d.dll" /pdb
type:sept

!ELSEIF "$(CFG)" == "zlib - Win32 LIB Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "zlib__Win32_LIB_Release"
# PROP BASE Intermediate_Dir "zlib__Win32_LIB_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_LIB_Release"
# PROP Intermediate_Dir "Win32_LIB_Release"
# PROP Target_Dir ""
CPP=cl.exe
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX /Yc /Yu
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT CPP /YX /Yc /Yu
RSC=rc.exe
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LIB32=link.exe -lib
# ADD BASE LIB32 /nologo
# ADD LIB32 /nologo

!ELSEIF "$(CFG)" == "zlib - Win32 LIB Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "zlib__Win32_LIB_Debug"
# PROP BASE Intermediate_Dir "zlib__Win32_LIB_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_LIB_Debug"
# PROP Intermediate_Dir "Win32_LIB_Debug"
# PROP Target_Dir ""
CPP=cl.exe
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX /Yc /Yu
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT CPP /YX /Yc /Yu
RSC=rc.exe
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LIB32=link.exe -lib
# ADD BASE LIB32 /nologo
```

```
# ADD LIB32 /nologo /out:"Win32_LIB_Debug\zlibd.lib"

!ELSEIF "$(CFG)" == "zlib - Win32 LIB ASM Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "zlib___Win32_LIB_ASM_Release"
# PROP BASE Intermediate_Dir "zlib___Win32_LIB_ASM_Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Win32_LIB_ASM_Release"
# PROP Intermediate_Dir "Win32_LIB_ASM_Release"
# PROP Target_Dir ""
CPP=cl.exe
# ADD BASE CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /FD /c
# SUBTRACT BASE CPP /YX /Yc /Yu
# ADD CPP /nologo /MD /W3 /O2 /D "WIN32" /D "NDEBUG" /D "ASMV" /D "ASMINF" /FD /c
# SUBTRACT CPP /YX /Yc /Yu
RSC=rc.exe
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LIB32=link.exe -lib
# ADD BASE LIB32 /nologo
# ADD LIB32 /nologo

!ELSEIF "$(CFG)" == "zlib - Win32 LIB ASM Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "zlib___Win32_LIB_ASM_Debug"
# PROP BASE Intermediate_Dir "zlib___Win32_LIB_ASM_Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Win32_LIB_ASM_Debug"
# PROP Intermediate_Dir "Win32_LIB_ASM_Debug"
# PROP Target_Dir ""
CPP=cl.exe
# ADD BASE CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /FD /GZ /c
# SUBTRACT BASE CPP /YX /Yc /Yu
# ADD CPP /nologo /MDd /W3 /Gm /ZI /Od /D "WIN32" /D "_DEBUG" /D "ASMV" /D "ASMINF" /FD /
GZ /c
# SUBTRACT CPP /YX /Yc /Yu
RSC=rc.exe
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LIB32=link.exe -lib
# ADD BASE LIB32 /nologo
# ADD LIB32 /nologo /out:"Win32_LIB_ASM_Debug\zlibd.lib"

!ENDIF

# Begin Target

# Name "zlib - Win32 DLL Release"
# Name "zlib - Win32 DLL Debug"
# Name "zlib - Win32 DLL ASM Release"
# Name "zlib - Win32 DLL ASM Debug"
# Name "zlib - Win32 LIB Release"
# Name "zlib - Win32 LIB Debug"
# Name "zlib - Win32 LIB ASM Release"
# Name "zlib - Win32 LIB ASM Debug"
# Begin Group "Source Files"

# PROP Default_Filter "cpp;c;cxx;rc;def;r;odl;idl;hpj;bat"
# Begin Source File
```

```
SOURCE=..\..\adler32.c
# End Source File
# Begin Source File

SOURCE=..\..\compress.c
# End Source File
# Begin Source File

SOURCE=..\..\crc32.c
# End Source File
# Begin Source File

SOURCE=..\..\deflate.c
# End Source File
# Begin Source File

SOURCE=..\..\gzio.c
# End Source File
# Begin Source File

SOURCE=..\..\inffback.c
# End Source File
# Begin Source File

SOURCE=..\..\inffast.c
# End Source File
# Begin Source File

SOURCE=..\..\inflate.c
# End Source File
# Begin Source File

SOURCE=..\..\inftrees.c
# End Source File
# Begin Source File

SOURCE=..\..\trees.c
# End Source File
# Begin Source File

SOURCE=..\..\uncompr.c
# End Source File
# Begin Source File

SOURCE=..\..\win32\zlib.def

!IF "$(CFG)" == "zlib - Win32 DLL Release"

!ELSEIF "$(CFG)" == "zlib - Win32 DLL Debug"

!ELSEIF "$(CFG)" == "zlib - Win32 DLL ASM Release"

!ELSEIF "$(CFG)" == "zlib - Win32 DLL ASM Debug"

!ELSEIF "$(CFG)" == "zlib - Win32 LIB Release"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 LIB Debug"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 LIB ASM Release"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 LIB ASM Debug"

# PROP Exclude_From_Build 1

!ENDIF

# End Source File
# Begin Source File
```

```
SOURCE=..\..\zutil.c
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h;hpp;hxx;hm;inl"
# Begin Source File

SOURCE=..\..\crc32.h
# End Source File
# Begin Source File

SOURCE=..\..\deflate.h
# End Source File
# Begin Source File

SOURCE=..\..\inffast.h
# End Source File
# Begin Source File

SOURCE=..\..\inffixed.h
# End Source File
# Begin Source File

SOURCE=..\..\inflate.h
# End Source File
# Begin Source File

SOURCE=..\..\inftrees.h
# End Source File
# Begin Source File

SOURCE=..\..\trees.h
# End Source File
# Begin Source File

SOURCE=..\..\zconf.h
# End Source File
# Begin Source File

SOURCE=..\..\zlib.h
# End Source File
# Begin Source File

SOURCE=..\..\zutil.h
# End Source File
# End Group
# Begin Group "Resource Files"

# PROP Default_Filter "ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe"
# Begin Source File

SOURCE=..\..\win32\zlib1.rc
# End Source File
# End Group
# Begin Group "Assembler Files (Unsupported)"

# PROP Default_Filter "asm;obj;c;cpp;cxx;h;hpp;hxx"
# Begin Source File

SOURCE=..\..\contrib\masmx86\gvm32.asm

!IF "$(CFG)" == "zlib - Win32 DLL Release"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 DLL Debug"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 DLL ASM Release"

# Begin Custom Build - Assembling...
```



```
IntDir=.\Win32_DLL_ASM_Release
InputPath=..\..\contrib\masmx86\gvmat32.asm
InputName=gvmat32

"$(IntDir)\$(InputName).obj" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
    ml.exe /nologo /c /coff /Cx /Fo"$(IntDir)\$(InputName).obj" "$(InputPath)"

# End Custom Build

!ELSEIF "$(CFG)" == "zlib - Win32 DLL ASM Debug"

# Begin Custom Build - Assembling...
IntDir=.\Win32_DLL_ASM_Debug
InputPath=..\..\contrib\masmx86\gvmat32.asm
InputName=gvmat32

"$(IntDir)\$(InputName).obj" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
    ml.exe /nologo /c /coff /Cx /Zi /Fo"$(IntDir)\$(InputName).obj" "$(InputPath)"

# End Custom Build

!ELSEIF "$(CFG)" == "zlib - Win32 LIB Release"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 LIB Debug"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 LIB ASM Release"

# Begin Custom Build - Assembling...
IntDir=.\Win32_LIB_ASM_Release
InputPath=..\..\contrib\masmx86\gvmat32.asm
InputName=gvmat32

"$(IntDir)\$(InputName).obj" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
    ml.exe /nologo /c /coff /Cx /Fo"$(IntDir)\$(InputName).obj" "$(InputPath)"

# End Custom Build

!ELSEIF "$(CFG)" == "zlib - Win32 LIB ASM Debug"

# Begin Custom Build - Assembling...
IntDir=.\Win32_LIB_ASM_Debug
InputPath=..\..\contrib\masmx86\gvmat32.asm
InputName=gvmat32

"$(IntDir)\$(InputName).obj" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
    ml.exe /nologo /c /coff /Cx /Zi /Fo"$(IntDir)\$(InputName).obj" "$(InputPath)"

# End Custom Build

!ENDIF

# End Source File
# Begin Source File

SOURCE=..\..\contrib\masmx86\gvmat32c.c

!IF "$(CFG)" == "zlib - Win32 DLL Release"

# PROP Exclude_From_Build 1
# ADD CPP /I "..\.."

!ELSEIF "$(CFG)" == "zlib - Win32 DLL Debug"

# PROP Exclude_From_Build 1
# ADD CPP /I "..\.."

!ELSEIF "$(CFG)" == "zlib - Win32 DLL ASM Release"

# ADD CPP /I "..\.."
```

```
!ELSEIF "$(CFG)" == "zlib - Win32 DLL ASM Debug"

# ADD CPP /I "..\.."

!ELSEIF "$(CFG)" == "zlib - Win32 LIB Release"

# PROP Exclude_From_Build 1
# ADD CPP /I "..\.."

!ELSEIF "$(CFG)" == "zlib - Win32 LIB Debug"

# PROP Exclude_From_Build 1
# ADD CPP /I "..\.."

!ELSEIF "$(CFG)" == "zlib - Win32 LIB ASM Release"

# ADD CPP /I "..\.."

!ELSEIF "$(CFG)" == "zlib - Win32 LIB ASM Debug"

# ADD CPP /I "..\.."

!ENDIF

# End Source File
# Begin Source File

SOURCE=..\..\contrib\masmx86\inffas32.asm

!IF "$(CFG)" == "zlib - Win32 DLL Release"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 DLL Debug"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 DLL ASM Release"

# Begin Custom Build - Assembling...
IntDir=.\Win32_DLL_ASM_Release
InputPath=..\..\contrib\masmx86\inffas32.asm
InputName=inffas32

"$(IntDir)\$(InputName).obj" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
    ml.exe /nologo /c /coff /Cx /Fo"$(IntDir)\$(InputName).obj" "$(InputPath)"

# End Custom Build

!ELSEIF "$(CFG)" == "zlib - Win32 DLL ASM Debug"

# Begin Custom Build - Assembling...
IntDir=.\Win32_DLL_ASM_Debug
InputPath=..\..\contrib\masmx86\inffas32.asm
InputName=inffas32

"$(IntDir)\$(InputName).obj" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
    ml.exe /nologo /c /coff /Cx /Zi /Fo"$(IntDir)\$(InputName).obj" "$(InputPath)"

# End Custom Build

!ELSEIF "$(CFG)" == "zlib - Win32 LIB Release"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 LIB Debug"

# PROP Exclude_From_Build 1

!ELSEIF "$(CFG)" == "zlib - Win32 LIB ASM Release"

# Begin Custom Build - Assembling...
IntDir=.\Win32_LIB_ASM_Release
InputPath=..\..\contrib\masmx86\inffas32.asm
```

```
InputName=inffas32

"$(IntDir)\$(InputName).obj" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
    ml.exe /nologo /c /coff /Cx /Fo"$(IntDir)\$(InputName).obj" "$(InputPath)"

# End Custom Build

!ELSEIF "$(CFG)" == "zlib - Win32 LIB ASM Debug"

# Begin Custom Build - Assembling...
IntDir=.\Win32_LIB_ASM_Debug
InputPath=..\..\contrib\masmx86\inffas32.asm
InputName=inffas32

"$(IntDir)\$(InputName).obj" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
    ml.exe /nologo /c /coff /Cx /Zi /Fo"$(IntDir)\$(InputName).obj" "$(InputPath)"

# End Custom Build

!ENDIF

# End Source File
# End Group
# Begin Source File

SOURCE=.\README.txt
# End Source File
# End Target
# End Project
```

Microsoft Developer Studio Workspace File, Format Version 6.00

WARNING: DO NOT EDIT OR DELETE THIS WORKSPACE FILE!

#####

Project: "example"=".\\example.dsp - Package Owner=<4>

Package=<5>

{{{
}}}

Package=<4>

{{{
 Begin Project Dependency
 Project_Dep_Name zlib
 End Project Dependency
}}}

#####

Project: "minigzip"=".\\minigzip.dsp - Package Owner=<4>

Package=<5>

{{{
}}}

Package=<4>

{{{
 Begin Project Dependency
 Project_Dep_Name zlib
 End Project Dependency
}}}

#####

Project: "zlib"=".\\zlib.dsp - Package Owner=<4>

Package=<5>

{{{
}}}

Package=<4>

{{{
}}}

#####

Global:

Package=<5>

{{{
}}}

Package=<3>

{{{
}}}

#####

```

<QPG:Generation>
  <QPG:Options>
    <QPG:User unattended="no" verbosity="2" listfiles="yes"/>
    <QPG:Defaults type="qnx_package"/>
    <QPG:Source></QPG:Source>
    <QPG:Release number="+"/>
    <QPG:Build></QPG:Build>
    <QPG:FileSorting strip="yes"/>
    <QPG:Package targets="combine"/>
    <QPG:Repository generate="yes"/>
    <QPG:FinalDir></QPG:FinalDir>
    <QPG:Cleanup></QPG:Cleanup>
  </QPG:Options>

  <QPG:Responsible>
    <QPG:Company></QPG:Company>
    <QPG:Department></QPG:Department>
    <QPG:Group></QPG:Group>
    <QPG:Team></QPG:Team>
    <QPG:Employee></QPG:Employee>
    <QPG:EmailAddress></QPG:EmailAddress>
  </QPG:Responsible>

  <QPG:Values>
    <QPG:Files>
      <QPG:Add file="../../zconf.h" install="/opt/include/" user="root:sys" permission="644"/>
      <QPG:Add file="../../zlib.h" install="/opt/include/" user="root:sys" permission="644"/>
      <QPG:Add file="../../libz.so.1.2.3" install="/opt/lib/" user="root:bin" permission="644"/>
      <QPG:Add file="libz.so" install="/opt/lib/" component="dev" filetype="symlink" linkto="libz.so.1.2.3"/>
      <QPG:Add file="libz.so.1" install="/opt/lib/" filetype="symlink" linkto="libz.so.1.2.3"/>
      <QPG:Add file="../../libz.so.1.2.3" install="/opt/lib/" component="slib"/>
    </QPG:Files>

    <QPG:PackageFilter>
      <QPM:PackageManifest>
        <QPM:PackageDescription>
          <QPM:PackageType>Library</QPM:PackageType>
          <QPM:PackageReleaseNotes></QPM:PackageReleaseNotes>
          <QPM:PackageReleaseUrgency>Medium</QPM:PackageReleaseUrgency>
          <QPM:PackageRepository></QPM:PackageRepository>
          <QPM:FileVersion>2.0</QPM:FileVersion>
        </QPM:PackageDescription>

        <QPM:ProductDescription>
          <QPM:ProductName>zlib</QPM:ProductName>
          <QPM:ProductIdentifier>zlib</QPM:ProductIdentifier>
          <QPM:ProductEmail>alain.bonnefoy@icbt.com</QPM:ProductEmail>
          <QPM:VendorName>Public</QPM:VendorName>
          <QPM:VendorInstallName>public</QPM:VendorInstallName>
          <QPM:VendorURL>www.gzip.org/zlib</QPM:VendorURL>
          <QPM:VendorEmbedURL></QPM:VendorEmbedURL>
          <QPM:VendorEmail></QPM:VendorEmail>
          <QPM:AuthorName>Jean-Loup Gailly, Mark Adler</QPM:AuthorName>
          <QPM:AuthorURL>www.gzip.org/zlib</QPM:AuthorURL>
          <QPM:AuthorEmbedURL></QPM:AuthorEmbedURL>
          <QPM:AuthorEmail>zlib@gzip.org</QPM:AuthorEmail>
          <QPM:ProductIconSmall></QPM:ProductIconSmall>
          <QPM:ProductIconLarge></QPM:ProductIconLarge>
          <QPM:ProductDescriptionShort>A massively spiffy yet delicately unobtrusive
compression library.</QPM:ProductDescriptionShort>
          <QPM:ProductDescriptionLong>zlib is designed to be a free, general-purpose
, legally unencumbered, lossless data compression library for use on virtually any comput
er hardware and operating system.</QPM:ProductDescriptionLong>
          <QPM:ProductDescriptionURL>http://www.gzip.org/zlib</QPM:ProductDescriptio
nURL>
          <QPM:ProductDescriptionEmbedURL></QPM:ProductDescriptionEmbedURL>
        </QPM:ProductDescription>

        <QPM:ReleaseDescription>

```

```

    <QPM:ReleaseVersion>1.2.3</QPM:ReleaseVersion>
    <QPM:ReleaseUrgency>Medium</QPM:ReleaseUrgency>
    <QPM:ReleaseStability>Stable</QPM:ReleaseStability>
    <QPM:ReleaseNoteMinor></QPM:ReleaseNoteMinor>
    <QPM:ReleaseNoteMajor></QPM:ReleaseNoteMajor>
    <QPM:ExcludeCountries>
      <QPM:Country></QPM:Country>
    </QPM:ExcludeCountries>

    <QPM:ReleaseCopyright>No License</QPM:ReleaseCopyright>
  </QPM:ReleaseDescription>

  <QPM:ContentDescription>
    <QPM:ContentTopic xmlmultiple="true">Software Development/Libraries and Ex
tensions/C Libraries</QPM:ContentTopic>
    <QPM:ContentKeyword>zlib,compression</QPM:ContentKeyword>
    <QPM:TargetOS>qnx6</QPM:TargetOS>
    <QPM:HostOS>qnx6</QPM:HostOS>
    <QPM:DisplayEnvironment xmlmultiple="true">None</QPM:DisplayEnvironment>
    <QPM:TargetAudience xmlmultiple="true">Developer</QPM:TargetAudience>
  </QPM:ContentDescription>
</QPM:PackageManifest>
</QPG:PackageFilter>

<QPG:PackageFilter proc="none" target="none">
  <QPM:PackageManifest>
    <QPM:ProductInstallationDependencies>
      <QPM:ProductRequirements></QPM:ProductRequirements>
    </QPM:ProductInstallationDependencies>

    <QPM:ProductInstallationProcedure>
      <QPM:Script xmlmultiple="true">
        <QPM:ScriptName></QPM:ScriptName>
        <QPM:ScriptType>Install</QPM:ScriptType>
        <QPM:ScriptTiming>Post</QPM:ScriptTiming>
        <QPM:ScriptBlocking>No</QPM:ScriptBlocking>
        <QPM:ScriptResult>Ignore</QPM:ScriptResult>
        <QPM:ShortDescription></QPM:ShortDescription>
        <QPM:UseBinaries>No</QPM:UseBinaries>
        <QPM:Priority>Optional</QPM:Priority>
      </QPM:Script>
    </QPM:ProductInstallationProcedure>
  </QPM:PackageManifest>

  <QPM:Launch>
  </QPM:Launch>
</QPG:PackageFilter>

<QPG:PackageFilter type="core" component="none">
  <QPM:PackageManifest>
    <QPM:ProductInstallationProcedure>
      <QPM:OrderDependency xmlmultiple="true">
        <QPM:Order>InstallOver</QPM:Order>
        <QPM:Product>zlib</QPM:Product>
      </QPM:OrderDependency>
    </QPM:ProductInstallationProcedure>
  </QPM:PackageManifest>

  <QPM:Launch>
  </QPM:Launch>
</QPG:PackageFilter>

<QPG:PackageFilter type="core" component="dev">
  <QPM:PackageManifest>
    <QPM:ProductInstallationProcedure>
      <QPM:OrderDependency xmlmultiple="true">
        <QPM:Order>InstallOver</QPM:Order>
        <QPM:Product>zlib-dev</QPM:Product>
      </QPM:OrderDependency>
    </QPM:ProductInstallationProcedure>
  </QPM:PackageManifest>

  <QPM:Launch>
  </QPM:Launch>

```

```
</QPG:PackageFilter>  
</QPG:Values>  
</QPG:Generation>
```

Frequently Asked Questions about ZLIB1.DLL

This document describes the design, the rationale, and the usage of the official DLL build of zlib, named ZLIB1.DLL. If you have general questions about zlib, you should see the file "FAQ" found in the zlib distribution, or at the following location:

http://www.gzip.org/zlib/zlib_faq.html

1. What is ZLIB1.DLL, and how can I get it?

- ZLIB1.DLL is the official build of zlib as a DLL.
(Please remark the character '1' in the name.)

Pointers to a precompiled ZLIB1.DLL can be found in the zlib web site at:

<http://www.zlib.org/>

Applications that link to ZLIB1.DLL can rely on the following specification:

- * The exported symbols are exclusively defined in the source files "zlib.h" and "zlib.def", found in an official zlib source distribution.
- * The symbols are exported by name, not by ordinal.
- * The exported names are undecorated.
- * The calling convention of functions is "C" (CDECL).
- * The ZLIB1.DLL binary is linked to MSVCRT.DLL.

The archive in which ZLIB1.DLL is bundled contains compiled test programs that must run with a valid build of ZLIB1.DLL. It is recommended to download the prebuilt DLL from the zlib web site, instead of building it yourself, to avoid potential incompatibilities that could be introduced by your compiler and build settings. If you do build the DLL yourself, please make sure that it complies with all the above requirements, and it runs with the precompiled test programs, bundled with the original ZLIB1.DLL distribution.

If, for any reason, you need to build an incompatible DLL, please use a different file name.

2. Why did you change the name of the DLL to ZLIB1.DLL?

What happened to the old ZLIB.DLL?

- The old ZLIB.DLL, built from zlib-1.1.4 or earlier, required compilation settings that were incompatible to those used by a static build. The DLL settings were supposed to be enabled by defining the macro ZLIB_DLL, before including "zlib.h". Incorrect handling of this macro was silently accepted at build time, resulting in two major problems:
 - * ZLIB_DLL was missing from the old makefile. When building the DLL, not all people added it to the build options. In consequence, incompatible incarnations of ZLIB.DLL started to circulate around the net.
 - * When switching from using the static library to using the DLL, applications had to define the ZLIB_DLL macro and to recompile all the sources that contained calls to zlib functions. Failure to do so resulted in creating binaries that were unable to run with the official ZLIB.DLL build.

The only possible solution that we could foresee was to make a binary-incompatible change in the DLL interface, in order to remove the dependency on the ZLIB_DLL macro, and to release the new DLL under a different name.

We chose the name ZLIB1.DLL, where '1' indicates the major zlib version number. We hope that we will not have to break the binary compatibility again, at least not as long as the

zlib-1.x series will last.

There is still a ZLIB_DLL macro, that can trigger a more efficient build and use of the DLL, but compatibility no longer depends on it.

3. Can I build ZLIB.DLL from the new zlib sources, and replace an old ZLIB.DLL, that was built from zlib-1.1.4 or earlier?

- In principle, you can do it by assigning calling convention keywords to the macros ZEXPORT and ZEXPORTVA. In practice, it depends on what you mean by "an old ZLIB.DLL", because the old DLL exists in several mutually-incompatible versions. You have to find out first what kind of calling convention is being used in your particular ZLIB.DLL build, and to use the same one in the new build. If you don't know what this is all about, you might be better off if you would just leave the old DLL intact.

4. Can I compile my application using the new zlib interface, and link it to an old ZLIB.DLL, that was built from zlib-1.1.4 or earlier?

- The official answer is "no"; the real answer depends again on what kind of ZLIB.DLL you have. Even if you are lucky, this course of action is unreliable.

If you rebuild your application and you intend to use a newer version of zlib (post- 1.1.4), it is strongly recommended to link it to the new ZLIB1.DLL.

5. Why are the zlib symbols exported by name, and not by ordinal?

- Although exporting symbols by ordinal is a little faster, it is risky. Any single glitch in the maintenance or use of the DEF file that contains the ordinals can result in incompatible builds and frustrating crashes. Simply put, the benefits of exporting symbols by ordinal do not justify the risks.

Technically, it should be possible to maintain ordinals in the DEF file, and still export the symbols by name. Ordinals exist in every DLL, and even if the dynamic linking performed at the DLL startup is searching for names, ordinals serve as hints, for a faster name lookup. However, if the DEF file contains ordinals, the Microsoft linker automatically builds an implib that will cause the executables linked to it to use those ordinals, and not the names. It is interesting to notice that the GNU linker for Win32 does not suffer from this problem.

It is possible to avoid the DEF file if the exported symbols are accompanied by a "__declspec(dllexport)" attribute in the source files. You can do this in zlib by predefining the ZLIB_DLL macro.

6. I see that the ZLIB1.DLL functions use the "C" (CDECL) calling convention. Why not use the STDCALL convention? STDCALL is the standard convention in Win32, and I need it in my Visual Basic project!

(For readability, we use CDECL to refer to the convention triggered by the "__cdecl" keyword, STDCALL to refer to the convention triggered by "__stdcall", and FASTCALL to refer to the convention triggered by "__fastcall".)

- Most of the native Windows API functions (without varargs) use indeed the WINAPI convention (which translates to STDCALL in Win32), but the standard C functions use CDECL. If a user application is intrinsically tied to the Windows API (e.g. it calls native Windows API functions such as CreateFile()),

sometimes it makes sense to decorate its own functions with WINAPI. But if ANSI C or POSIX portability is a goal (e.g. it calls standard C functions such as fopen()), it is not a sound decision to request the inclusion of <windows.h>, or to use non-ANSI constructs, for the sole purpose to make the user functions STDCALL-able.

The functionality offered by zlib is not in the category of "Windows functionality", but is more like "C functionality".

Technically, STDCALL is not bad; in fact, it is slightly faster than CDECL, and it works with variable-argument functions, just like CDECL. It is unfortunate that, in spite of using STDCALL in the Windows API, it is not the default convention used by the C compilers that run under Windows. The roots of the problem reside deep inside the unsafety of the K&R-style function prototypes, where the argument types are not specified; but that is another story for another day.

The remaining fact is that CDECL is the default convention. Even if an explicit convention is hard-coded into the function prototypes inside C headers, problems may appear. The necessity to expose the convention in users' callbacks is one of these problems.

The calling convention issues are also important when using zlib in other programming languages. Some of them, like Ada (GNAT) and Fortran (GNU G77), have C bindings implemented initially on Unix, and relying on the C calling convention. On the other hand, the pre-.NET versions of Microsoft Visual Basic require STDCALL, while Borland Delphi prefers, although it does not require, FASTCALL.

In fairness to all possible uses of zlib outside the C programming language, we choose the default "C" convention. Anyone interested in different bindings or conventions is encouraged to maintain specialized projects. The "contrib/" directory from the zlib distribution already holds a couple of foreign bindings, such as Ada, C++, and Delphi.

7. I need a DLL for my Visual Basic project. What can I do?

- Define the ZLIB_WINAPI macro before including "zlib.h", when building both the DLL and the user application (except that you don't need to define anything when using the DLL in Visual Basic). The ZLIB_WINAPI macro will switch on the WINAPI (STDCALL) convention. The name of this DLL must be different than the official ZLIB1.DLL.

Gilles Vollant has contributed a build named ZLIBWAPI.DLL, with the ZLIB_WINAPI macro turned on, and with the minizip functionality built in. For more information, please read the notes inside "contrib/vstudio/readme.txt", found in the zlib distribution.

8. I need to use zlib in my Microsoft .NET project. What can I do?

- Henrik Ravn has contributed a .NET wrapper around zlib. Look into contrib/dotzlib/, inside the zlib distribution.

9. If my application uses ZLIB1.DLL, should I link it to MSVCRT.DLL? Why?

- It is not required, but it is recommended to link your application to MSVCRT.DLL, if it uses ZLIB1.DLL.

The executables (.EXE, .DLL, etc.) that are involved in the same process and are using the C run-time library (i.e. they are calling standard C functions), must link to the same library. There are several libraries in the Win32 system:

CRTDLL.DLL, MSVCRT.DLL, the static C libraries, etc.
Since ZLIB1.DLL is linked to MSVCRT.DLL, the executables that depend on it should also be linked to MSVCRT.DLL.

10. Why are you saying that ZLIB1.DLL and my application should be linked to the same C run-time (CRT) library? I linked my application and my DLLs to different C libraries (e.g. my application to a static library, and my DLLs to MSVCRT.DLL), and everything works fine.
- If a user library invokes only pure Win32 API (accessible via <windows.h> and the related headers), its DLL build will work in any context. But if this library invokes standard C API, things get more complicated.

There is a single Win32 library in a Win32 system. Every function in this library resides in a single DLL module, that is safe to call from anywhere. On the other hand, there are multiple versions of the C library, and each of them has its own separate internal state. Standalone executables and user DLLs that call standard C functions must link to a C run-time (CRT) library, be it static or shared (DLL). Intermixing occurs when an executable (not necessarily standalone) and a DLL are linked to different CRTs, and both are running in the same process.

Intermixing multiple CRTs is possible, as long as their internal states are kept intact. The Microsoft Knowledge Base articles KB94248 "HOWTO: Use the C Run-Time" and KB140584 "HOWTO: Link with the Correct C Run-Time (CRT) Library" mention the potential problems raised by intermixing.

If intermixing works for you, it's because your application and DLLs are avoiding the corruption of each of the CRTs' internal states, maybe by careful design, or maybe by fortune.

Also note that linking ZLIB1.DLL to non-Microsoft CRTs, such as those provided by Borland, raises similar problems.

11. Why are you linking ZLIB1.DLL to MSVCRT.DLL?

- MSVCRT.DLL exists on every Windows 95 with a new service pack installed, or with Microsoft Internet Explorer 4 or later, and on all other Windows 4.x or later (Windows 98, Windows NT 4, or later). It is freely distributable; if not present in the system, it can be downloaded from Microsoft or from other software provider for free.

The fact that MSVCRT.DLL does not exist on a virgin Windows 95 is not so problematic. Windows 95 is scarcely found nowadays, Microsoft ended its support a long time ago, and many recent applications from various vendors, including Microsoft, do not even run on it. Furthermore, no serious user should run Windows 95 without a proper update installed.

12. Why are you not linking ZLIB1.DLL to <<my favorite C run-time library>> ?

- We considered and abandoned the following alternatives:
 - * Linking ZLIB1.DLL to a static C library (LIBC.LIB, or LIBCMT.LIB) is not a good option. People are using the DLL mainly to save disk space. If you are linking your program to a static C library, you may as well consider linking zlib in statically, too.
 - * Linking ZLIB1.DLL to CRTDLL.DLL looks appealing, because CRTDLL.DLL is present on every Win32 installation. Unfortunately, it has a series of problems: it does not work properly with Microsoft's C++ libraries, it does not provide support for 64-bit file offsets, (and so on...),

and Microsoft discontinued its support a long time ago.

- * Linking ZLIB1.DLL to MSVCR70.DLL or MSVCR71.DLL, supplied with the Microsoft .NET platform, and Visual C++ 7.0/7.1, raises problems related to the status of ZLIB1.DLL as a system component. According to the Microsoft Knowledge Base article KB326922 "INFO: Redistribution of the Shared C Runtime Component in Visual C++ .NET", MSVCR70.DLL and MSVCR71.DLL are not supposed to function as system DLLs, because they may clash with MSVCRT.DLL. Instead, the application's installer is supposed to put these DLLs (if needed) in the application's private directory. If ZLIB1.DLL depends on a non-system runtime, it cannot function as a redistributable system component.
- * Linking ZLIB1.DLL to non-Microsoft runtimes, such as Borland's, or Cygwin's, raises problems related to the reliable presence of these runtimes on Win32 systems. It's easier to let the DLL build of zlib up to the people who distribute these runtimes, and who may proceed as explained in the answer to Question 14.

13. If ZLIB1.DLL cannot be linked to MSVCR70.DLL or MSVCR71.DLL, how can I build/use ZLIB1.DLL in Microsoft Visual C++ 7.0 (Visual Studio .NET) or newer?

- Due to the problems explained in the Microsoft Knowledge Base article KB326922 (see the previous answer), the C runtime that comes with the VC7 environment is no longer considered a system component. That is, it should not be assumed that this runtime exists, or may be installed in a system directory. Since ZLIB1.DLL is supposed to be a system component, it may not depend on a non-system component.

In order to link ZLIB1.DLL and your application to MSVCRT.DLL in VC7, you need the library of Visual C++ 6.0 or older. If you don't have this library at hand, it's probably best not to use ZLIB1.DLL.

We are hoping that, in the future, Microsoft will provide a way to build applications linked to a proper system runtime, from the Visual C++ environment. Until then, you have a couple of alternatives, such as linking zlib in statically. If your application requires dynamic linking, you may proceed as explained in the answer to Question 14.

14. I need to link my own DLL build to a CRT different than MSVCRT.DLL. What can I do?

- Feel free to rebuild the DLL from the zlib sources, and link it the way you want. You should, however, clearly state that your build is unofficial. You should give it a different file name, and/or install it in a private directory that can be accessed by your application only, and is not visible to the others (e.g. it's not in the SYSTEM or the SYSTEM32 directory, and it's not in the PATH). Otherwise, your build may clash with applications that link to the official build.

For example, in Cygwin, zlib is linked to the Cygwin runtime CYGWIN1.DLL, and it is distributed under the name CYGZ.DLL.

15. May I include additional pieces of code that I find useful, link them in ZLIB1.DLL, and export them?

- No. A legitimate build of ZLIB1.DLL must not include code that does not originate from the official zlib source code. But you can make your own private DLL build, under a different file name, as suggested in the previous answer.

For example, zlib is a part of the VCL library, distributed with Borland Delphi and C++ Builder. The DLL build of VCL

is a redistributable file, named VCLxx.DLL.

16. May I remove some functionality out of ZLIB1.DLL, by enabling macros like NO_GZCOMPRESS or NO_GZIP at compile time?

- No. A legitimate build of ZLIB1.DLL must provide the complete zlib functionality, as implemented in the official zlib source code. But you can make your own private DLL build, under a different file name, as suggested in the previous answer.

17. I made my own ZLIB1.DLL build. Can I test it for compliance?

- We prefer that you download the official DLL from the zlib web site. If you need something peculiar from this DLL, you can send your suggestion to the zlib mailing list.

However, in case you do rebuild the DLL yourself, you can run it with the test programs found in the DLL distribution. Running these test programs is not a guarantee of compliance, but a failure can imply a detected problem.

**

This document is written and maintained by
Cosmin Truta <cosmint@cs.ubbcluj.ro>

```
# Makefile for zlib
# Borland C++ for Win32
#
# Updated for zlib 1.2.x by Cosmin Truta, 11-Mar-2003
# Last updated: 28-Aug-2003
#
# Usage:
# make -f win32/Makefile.bor
# make -f win32/Makefile.bor LOCAL_ZLIB=-DASMV OBJA=match.obj OBJPA=+match.obj

# ----- Borland C++ -----

# Optional nonstandard preprocessor flags (e.g. -DMAX_MEM_LEVEL=7)
# should be added to the environment via "set LOCAL_ZLIB=-DFOO" or
# added to the declaration of LOC here:
LOC = $(LOCAL_ZLIB)

CC = bcc32
AS = bcc32
LD = bcc32
AR = tlib
CFLAGS = -a -d -k- -O2 $(LOC)
ASFLAGS = $(LOC)
LDFLAGS = $(LOC)

# variables
ZLIB_LIB = zlib.lib

OBJ1 = adler32.obj compress.obj crc32.obj deflate.obj gzio.obj infback.obj
OBJ2 = inffast.obj inflate.obj inftrees.obj trees.obj uncompr.obj zutil.obj
#OBJA =
OBJP1 = +adler32.obj+compress.obj+crc32.obj+deflate.obj+gzio.obj+infback.obj
OBJP2 = +inffast.obj+inflate.obj+inftrees.obj+trees.obj+uncompr.obj+zutil.obj
#OBJPA=

# targets
all: $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
    $(CC) -c $(CFLAGS) $<

.asm.obj:
    $(AS) -c $(ASFLAGS) $<

adler32.obj: adler32.c zlib.h zconf.h
compress.obj: compress.c zlib.h zconf.h
crc32.obj: crc32.c zlib.h zconf.h crc32.h
deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h
gzio.obj: gzio.c zutil.h zlib.h zconf.h
infback.obj: infback.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h inffixed.h
inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h
inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
    inffast.h inffixed.h
inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h
trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h
uncompr.obj: uncompr.c zlib.h zconf.h
zutil.obj: zutil.c zutil.h zlib.h zconf.h
example.obj: example.c zlib.h zconf.h
```

minigzip.obj: minigzip.c zlib.h zconf.h

*# For the sake of the old Borland make,
the command line is cut to fit in the MS-DOS 128 byte limit:*

\$(ZLIB_LIB): \$(OBJ1) \$(OBJ2) \$(OBJA)
-del \$(ZLIB_LIB)
\$(AR) \$(ZLIB_LIB) \$(OBJP1)
\$(AR) \$(ZLIB_LIB) \$(OBJP2)
\$(AR) \$(ZLIB_LIB) \$(OBJPA)

testing

test: example.exe minigzip.exe
example
echo hello world | minigzip | minigzip -d

example.exe: example.obj \$(ZLIB_LIB)
\$(LD) \$(LDFLAGS) example.obj \$(ZLIB_LIB)

minigzip.exe: minigzip.obj \$(ZLIB_LIB)
\$(LD) \$(LDFLAGS) minigzip.obj \$(ZLIB_LIB)

cleanup

clean:
-del *.obj
-del *.lib
-del *.exe
-del *.tds
-del zlib.bak
-del foo.gz

```
# Makefile for zlib.  Modified for emx/rsxnt by Chr. Spieler, 6/16/98.
# Copyright (C) 1995-1998 Jean-loup Gailly.
# For conditions of distribution and use, see copyright notice in zlib.h

# To compile, or to compile and test, type:
#
#   make -fmakefile.emx;  make test -fmakefile.emx
#

CC=gcc -Zwin32

#CFLAGS=-MMD -O
#CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
#CFLAGS=-MMD -g -DDEBUG
CFLAGS=-MMD -O3 $(BUTT) -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
        -Wstrict-prototypes -Wmissing-prototypes

# If cp.exe is available, replace "copy /Y" with "cp -fp" .
CP=copy /Y
# If gnu install.exe is available, replace $(CP) with ginstall.
INSTALL=$(CP)
# The default value of RM is "rm -f."  If "rm.exe" is found, comment out:
RM=del
LDLIBS=-L. -lzlib
LD=$(CC) -s -o
LDSHARED=$(CC)

INCL=zlib.h zconf.h
LIBS=zlib.a

AR=ar rcs

prefix=/usr/local
exec_prefix = $(prefix)

OBJS = adler32.o compress.o crc32.o gzio.o uncompr.o deflate.o trees.o \
        zutil.o inflate.o infback.o inftrees.o inffast.o

TEST_OBJS = example.o minigzip.o

all: example.exe minigzip.exe

test: all
        ./example
        echo hello world | .\minigzip | .\minigzip -d

%.o : %.c
        $(CC) $(CFLAGS) -c $< -o $@

zlib.a: $(OBJS)
        $(AR) $@ $(OBJS)

%.exe : %.o $(LIBS)
        $(LD) $@ $< $(LDLIBS)

.PHONY : clean

clean:
        $(RM) *.d
        $(RM) *.o
        $(RM) *.exe
        $(RM) zlib.a
        $(RM) foo.gz

DEPS := $(wildcard *.d)
ifneq ($(DEPS),)
include $(DEPS)
endif
```



```
# Makefile for zlib, derived from Makefile.dj2.
# Modified for mingw32 by C. Spieler, 6/16/98.
# Updated for zlib 1.2.x by Christian Spieler and Cosmin Truta, Mar-2003.
# Last updated: 1-Aug-2003.
# Tested under Cygwin and MinGW.

# Copyright (C) 1995-2003 Jean-loup Gailly.
# For conditions of distribution and use, see copyright notice in zlib.h

# To compile, or to compile and test, type:
#
#   make -fmakefile.gcc; make test testdll -fmakefile.gcc
#
# To use the asm code, type:
#   cp contrib/asm?86/match.S ./match.S
#   make LOC=-DASMV OBJA=match.o -fmakefile.gcc
#
# To install libz.a, zconf.h and zlib.h in the system directories, type:
#
#   make install -fmakefile.gcc

# Note:
# If the platform is *not* MinGW (e.g. it is Cygwin or UWIN),
# the DLL name should be changed from "zlib1.dll".

STATICLIB = libz.a
SHAREDLIB = zlib1.dll
IMPLIB     = libzdll.a

#LOC = -DASMV
#LOC = -DDEBUG -g

CC = gcc
CFLAGS = $(LOC) -O3 -Wall

AS = $(CC)
ASFLAGS = $(LOC) -Wall

LD = $(CC)
LDFLAGS = $(LOC) -s

AR = ar
ARFLAGS = rcs

RC = windres
RCFLAGS = --define GCC_WINDRES

CP = cp -fp
# If GNU install is available, replace $(CP) with install.
INSTALL = $(CP)
RM = rm -f

prefix = /usr/local
exec_prefix = $(prefix)

OBJS = Adler32.o compress.o crc32.o deflate.o gzio.o infback.o \
      inffast.o inflate.o inftrees.o trees.o uncompress.o zutil.o
OBJA =

all: $(STATICLIB) $(SHAREDLIB) $(IMPLIB) example minigzip example_d minigzip_d

test: example minigzip
      ./example
      echo hello world | ./minigzip | ./minigzip -d

testdll: example_d minigzip_d
      ./example_d
      echo hello world | ./minigzip_d | ./minigzip_d -d

.c.o:
      $(CC) $(CFLAGS) -c -o $@ $<

.S.o:
      $(AS) $(ASFLAGS) -c -o $@ $<
```

```
$(STATICLIB): $(OBJS) $(OBJA)
    $(AR) $(ARFLAGS) $@ $(OBJS) $(OBJA)

$(IMPLIB): $(SHAREDLIB)

$(SHAREDLIB): win32/zlib.def $(OBJS) $(OBJA) zlibrc.o
    dllwrap --driver-name $(CC) --def win32/zlib.def \
        --implib $(IMPLIB) -o $@ $(OBJS) $(OBJA) zlibrc.o
    strip $@

example: example.o $(STATICLIB)
    $(LD) $(LDFLAGS) -o $@ example.o $(STATICLIB)

minigzip: minigzip.o $(STATICLIB)
    $(LD) $(LDFLAGS) -o $@ minigzip.o $(STATICLIB)

example_d: example.o $(IMPLIB)
    $(LD) $(LDFLAGS) -o $@ example.o $(IMPLIB)

minigzip_d: minigzip.o $(IMPLIB)
    $(LD) $(LDFLAGS) -o $@ minigzip.o $(IMPLIB)

zlibrc.o: win32/zlib1.rc
    $(RC) $(RCFLAGS) -o $@ win32/zlib1.rc

# INCLUDE_PATH and LIBRARY_PATH must be set.

.PHONY: install uninstall clean

install: zlib.h zconf.h $(LIB)
    -@if not exist $(INCLUDE_PATH)/nul mkdir $(INCLUDE_PATH)
    -@if not exist $(LIBRARY_PATH)/nul mkdir $(LIBRARY_PATH)
    -$(INSTALL) zlib.h $(INCLUDE_PATH)
    -$(INSTALL) zconf.h $(INCLUDE_PATH)
    -$(INSTALL) $(STATICLIB) $(LIBRARY_PATH)
    -$(INSTALL) $(IMPLIB) $(LIBRARY_PATH)

uninstall:
    -$(RM) $(INCLUDE_PATH)/zlib.h
    -$(RM) $(INCLUDE_PATH)/zconf.h
    -$(RM) $(LIBRARY_PATH)/$(STATICLIB)
    -$(RM) $(LIBRARY_PATH)/$(IMPLIB)

clean:
    -$(RM) $(STATICLIB)
    -$(RM) $(SHAREDLIB)
    -$(RM) $(IMPLIB)
    -$(RM) *.o
    -$(RM) *.exe
    -$(RM) foo.gz

adler32.o: zlib.h zconf.h
compress.o: zlib.h zconf.h
crc32.o: crc32.h zlib.h zconf.h
deflate.o: deflate.h zutil.h zlib.h zconf.h
example.o: zlib.h zconf.h
gzio.o: zutil.h zlib.h zconf.h
inffast.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
inflate.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
inffast.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
inftrees.o: zutil.h zlib.h zconf.h inftrees.h
minigzip.o: zlib.h zconf.h
trees.o: deflate.h zutil.h zlib.h zconf.h trees.h
uncompr.o: zlib.h zconf.h
zutil.o: zutil.h zlib.h zconf.h
```

```
# Makefile for zlib -- Microsoft (Visual) C
#
# Authors:
#   Cosmin Truta, 11-Mar-2003
#   Christian Spieler, 19-Mar-2003
#
# Last updated:
#   Cosmin Truta, 27-Aug-2003
#
# Usage:
#   nmake -f win32/Makefile.msc           (standard build)
#   nmake -f win32/Makefile.msc LOC=-DFOO (nonstandard build)
#   nmake -f win32/Makefile.msc LOC=-DASMV OBJA=match.obj (use ASM code)

# optional build flags
LOC =

# variables
STATICLIB = zlib.lib
SHAREDLIB = zlib1.dll
IMPLIB    = zdll.lib

CC = cl
AS = ml
LD = link
AR = lib
RC = rc
CFLAGS = -nologo -MD -O2 $(LOC)
ASFLAGS = -coff
LDFLAGS = -nologo -release
ARFLAGS = -nologo
RCFLAGS = /dWIN32 /r

OBJS = adler32.obj compress.obj crc32.obj deflate.obj gzio.obj infback.obj \
       inffast.obj inflate.obj inftrees.obj trees.obj uncompr.obj zutil.obj
OBJA =

# targets
all: $(STATICLIB) $(SHAREDLIB) $(IMPLIB) \
     example.exe minigzip.exe example_d.exe minigzip_d.exe

$(STATICLIB): $(OBJS) $(OBJA)
             $(AR) $(ARFLAGS) -out:$@ $(OBJS) $(OBJA)

$(IMPLIB): $(SHAREDLIB)

$(SHAREDLIB): win32/zlib.def $(OBJS) $(OBJA) zlib1.res
             $(LD) $(LDFLAGS) -def:win32/zlib.def -dll -implib:$$(IMPLIB) \
             -out:$@ $(OBJS) $(OBJA) zlib1.res

example.exe: example.obj $(STATICLIB)
             $(LD) $(LDFLAGS) example.obj $(STATICLIB)

minigzip.exe: minigzip.obj $(STATICLIB)
             $(LD) $(LDFLAGS) minigzip.obj $(STATICLIB)

example_d.exe: example.obj $(IMPLIB)
             $(LD) $(LDFLAGS) -out:$@ example.obj $(IMPLIB)

minigzip_d.exe: minigzip.obj $(IMPLIB)
             $(LD) $(LDFLAGS) -out:$@ minigzip.obj $(IMPLIB)

.c.obj:
       $(CC) -c $(CFLAGS) $<

.asm.obj:
       $(AS) -c $(ASFLAGS) $<

adler32.obj: adler32.c zlib.h zconf.h

compress.obj: compress.c zlib.h zconf.h
```

```
crc32.obj: crc32.c zlib.h zconf.h crc32.h
deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h
gzio.obj: gzio.c zutil.h zlib.h zconf.h
inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
             inffast.h inffixed.h
inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
             inffast.h
inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
             inffast.h inffixed.h
inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h
trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h
uncompr.obj: uncompr.c zlib.h zconf.h
zutil.obj: zutil.c zutil.h zlib.h zconf.h
example.obj: example.c zlib.h zconf.h
minigzip.obj: minigzip.c zlib.h zconf.h
zlib1.res: win32/zlib1.rc
           $(RC) $(RCFLAGS) /fo$@ win32/zlib1.rc

# testing
test: example.exe minigzip.exe
      example
      echo hello world | minigzip | minigzip -d

testdll: example_d.exe minigzip_d.exe
         example_d
         echo hello world | minigzip_d | minigzip_d -d

# cleanup
clean:
      -del $(STATICLIB)
      -del $(SHAREDLIB)
      -del $(IMPLIB)
      -del *.obj
      -del *.res
      -del *.exp
      -del *.exe
      -del foo.gz
```

To build zlib using the Microsoft Visual C++ environment,
use the appropriate project from the projects/ directory.

```
LIBRARY
; zlib data compression library

EXPORTS
; basic functions
    zlibVersion
    deflate
    deflateEnd
    inflate
    inflateEnd
; advanced functions
    deflateSetDictionary
    deflateCopy
    deflateReset
    deflateParams
    deflateBound
    deflatePrime
    inflateSetDictionary
    inflateSync
    inflateCopy
    inflateReset
    inflateBack
    inflateBackEnd
    zlibCompileFlags
; utility functions
    compress
    compress2
    compressBound
    uncompress
    gzopen
    gzdopen
    gzsetparams
    gzread
    gzwrite
    gzprintf
    gzputs
    gzgets
    gzputc
    gzgetc
    gzungetc
    gzflush
    gzseek
    gzrewind
    gztell
    gzeof
    gzclos
    gzerror
    gzcloserr
; checksum functions
    adler32
    crc32
; various hacks, don't look :)
    deflateInit_
    deflateInit2_
    inflateInit_
    inflateInit2_
    inflateBackInit_
    inflateSyncPoint
    get_crc_table
    zError
```

```
#include <windows.h>

#ifdef GCC_WINDRES
VS_VERSION_INFO          VERSIONINFO
#else
VS_VERSION_INFO          VERSIONINFO      MOVEABLE IMPURE LOADONCALL DISCARDABLE
#endif
    FILEVERSION           1,2,2,0
    PRODUCTVERSION        1,2,2,0
    FILEFLAGSMASK         VS_FFI_FILEFLAGSMASK
#ifdef _DEBUG
    FILEFLAGS              1
#else
    FILEFLAGS              0
#endif
    FILEOS                 VOS_DOS_WINDOWS32
    FILETYPE               VFT_DLL
    FILESUBTYPE            0              // not used
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904E4"
        //language ID = U.S. English, char set = Windows, Multilingual
        BEGIN
            VALUE "FileDescription",  "zlib data compression library\0"
            VALUE "FileVersion",      "1.2.3\0"
            VALUE "InternalName",     "zlib1.dll\0"
            VALUE "LegalCopyright",   "(C) 1995-2004 Jean-loup Gailly & Mark Adler\0"
            VALUE "OriginalFilename", "zlib1.dll\0"
            VALUE "ProductName",      "zlib\0"
            VALUE "ProductVersion",   "1.2.3\0"
            VALUE "Comments", "DLL support by Alessandro Iacopetti & Gilles Vollant\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x0409, 1252
    END
END
```

Table of Contents

1	<code>_fchown.c</code>	sheets	1 to	1 (1)	pages	1-	1	18	lines
2	<code>COPYRIGHT</code>	sheets	2 to	2 (1)	pages	2-	2	57	lines
3	<code>Makefile.in</code>	sheets	3 to	3 (1)	pages	3-	3	14	lines
4	<code>WHATSOEVER</code>	sheets	4 to	5 (2)	pages	4-	5	95	lines
5	<code>cclass.h</code>	sheets	6 to	6 (1)	pages	6-	6	71	lines
6	<code>cname.h</code>	sheets	7 to	8 (2)	pages	7-	8	142	lines
7	<code>engine.c</code>	sheets	9 to	23 (15)	pages	9-	23	1092	lines
8	<code>re_format.7</code>	sheets	23 to	26 (4)	pages	23-	26	1	lines
9	<code>regcomp.c</code>	sheets	27 to	50 (24)	pages	27-	50	1707	lines
10	<code>regerror.c</code>	sheets	51 to	53 (3)	pages	51-	53	189	lines
11	<code>regex.3</code>	sheets	53 to	57 (5)	pages	53-	57	1	lines
12	<code>regex2.h</code>	sheets	58 to	60 (3)	pages	58-	60	174	lines
13	<code>regexec.c</code>	sheets	61 to	63 (3)	pages	61-	63	190	lines
14	<code>regfree.c</code>	sheets	64 to	65 (2)	pages	64-	65	85	lines
15	<code>utils.h</code>	sheets	66 to	66 (1)	pages	66-	66	62	lines
16	<code>Makefile.in</code>	sheets	67 to	67 (1)	pages	67-	67	61	lines
17	<code>clearerr.c</code>	sheets	68 to	68 (1)	pages	68-	68	13	lines
18	<code>data.c</code>	sheets	69 to	69 (1)	pages	69-	69	29	lines
19	<code>doprnt.c</code>	sheets	70 to	74 (5)	pages	70-	74	311	lines
20	<code>doscan.c</code>	sheets	75 to	81 (7)	pages	75-	81	450	lines
21	<code>ecvt.c</code>	sheets	82 to	82 (1)	pages	82-	82	32	lines
22	<code>fclose.c</code>	sheets	83 to	83 (1)	pages	83-	83	32	lines
23	<code>feof.c</code>	sheets	84 to	84 (1)	pages	84-	84	13	lines
24	<code>ferror.c</code>	sheets	85 to	85 (1)	pages	85-	85	13	lines
25	<code>fflush.c</code>	sheets	86 to	87 (2)	pages	86-	87	87	lines
26	<code>fgetc.c</code>	sheets	88 to	88 (1)	pages	88-	88	13	lines
27	<code>fgetpos.c</code>	sheets	89 to	89 (1)	pages	89-	89	15	lines
28	<code>fgets.c</code>	sheets	90 to	90 (1)	pages	90-	90	28	lines
29	<code>fileno.c</code>	sheets	91 to	91 (1)	pages	91-	91	13	lines
30	<code>fillbuf.c</code>	sheets	92 to	92 (1)	pages	92-	92	70	lines
31	<code>flushbuf.c</code>	sheets	93 to	94 (2)	pages	93-	94	128	lines
32	<code>fopen.c</code>	sheets	95 to	96 (2)	pages	95-	96	130	lines
33	<code>fprintf.c</code>	sheets	97 to	97 (1)	pages	97-	97	24	lines
34	<code>fputc.c</code>	sheets	98 to	98 (1)	pages	98-	98	13	lines
35	<code>fputs.c</code>	sheets	99 to	99 (1)	pages	99-	99	19	lines
36	<code>fread.c</code>	sheets	100 to	100 (1)	pages	100-	100	30	lines
37	<code>freopen.c</code>	sheets	101 to	102 (2)	pages	101-	102	111	lines
38	<code>fscanf.c</code>	sheets	103 to	103 (1)	pages	103-	103	24	lines
39	<code>fseek.c</code>	sheets	104 to	104 (1)	pages	104-	104	45	lines
40	<code>fsetpos.c</code>	sheets	105 to	105 (1)	pages	105-	105	13	lines
41	<code>ftell.c</code>	sheets	106 to	106 (1)	pages	106-	106	39	lines
42	<code>fwrite.c</code>	sheets	107 to	107 (1)	pages	107-	107	30	lines
43	<code>getc.c</code>	sheets	108 to	108 (1)	pages	108-	108	13	lines
44	<code>getchar.c</code>	sheets	109 to	109 (1)	pages	109-	109	13	lines
45	<code>gets.c</code>	sheets	110 to	110 (1)	pages	110-	110	28	lines
46	<code>icompute.c</code>	sheets	111 to	111 (1)	pages	111-	111	22	lines
47	<code>loc_incl.h</code>	sheets	112 to	112 (1)	pages	112-	112	41	lines
48	<code>mktemp.c</code>	sheets	113 to	114 (2)	pages	113-	114	133	lines
49	<code>perror.c</code>	sheets	115 to	115 (1)	pages	115-	115	34	lines
50	<code>printf.c</code>	sheets	116 to	116 (1)	pages	116-	116	24	lines
51	<code>putc.c</code>	sheets	117 to	117 (1)	pages	117-	117	13	lines
52	<code>putchar.c</code>	sheets	118 to	118 (1)	pages	118-	118	13	lines
53	<code>puts.c</code>	sheets	119 to	119 (1)	pages	119-	119	21	lines
54	<code>remove.c</code>	sheets	120 to	120 (1)	pages	120-	120	14	lines
55	<code>rewind.c</code>	sheets	121 to	121 (1)	pages	121-	121	15	lines
56	<code>scanf.c</code>	sheets	122 to	122 (1)	pages	122-	122	26	lines
57	<code>setbuf.c</code>	sheets	123 to	123 (1)	pages	123-	123	14	lines
58	<code>setvbuf.c</code>	sheets	124 to	124 (1)	pages	124-	124	49	lines
59	<code>sprintf.c</code>	sheets	125 to	125 (1)	pages	125-	125	40	lines
60	<code>sscanf.c</code>	sheets	126 to	126 (1)	pages	126-	126	31	lines
61	<code>tmpfile.c</code>	sheets	127 to	127 (1)	pages	127-	127	32	lines
62	<code>tmpnam.c</code>	sheets	128 to	128 (1)	pages	128-	128	32	lines
63	<code>ungetc.c</code>	sheets	129 to	129 (1)	pages	129-	129	27	lines
64	<code>vfprintf.c</code>	sheets	130 to	130 (1)	pages	130-	130	15	lines
65	<code>vprintf.c</code>	sheets	131 to	131 (1)	pages	131-	131	15	lines
66	<code>vscanf.c</code>	sheets	132 to	132 (1)	pages	132-	132	14	lines
67	<code>vsprintf.c</code>	sheets	133 to	133 (1)	pages	133-	133	35	lines
68	<code>vsscanf.c</code>	sheets	134 to	134 (1)	pages	134-	134	22	lines
69	<code>Makefile.in</code>	sheets	135 to	136 (2)	pages	135-	136	117	lines
70	<code>_exit.s</code>	sheets	137 to	137 (1)	pages	137-	137	8	lines
71	<code>_pm_findproc.s</code>	sheets	138 to	138 (1)	pages	138-	138	8	lines
72	<code>access.s</code>	sheets	139 to	139 (1)	pages	139-	139	8	lines
73	<code>alarm.s</code>	sheets	140 to	140 (1)	pages	140-	140	8	lines

74	<i>sysuname.s</i>	sheets	141	to	141	(1)	pages	141-141	8	lines
75	<i>brk.s</i>	sheets	142	to	142	(1)	pages	142-142	8	lines
76	<i>cfgetispeed.s</i>	sheets	143	to	143	(1)	pages	143-143	8	lines
77	<i>cfgetospeed.s</i>	sheets	144	to	144	(1)	pages	144-144	8	lines
78	<i>cfsetispeed.s</i>	sheets	145	to	145	(1)	pages	145-145	8	lines
79	<i>cfsetospeed.s</i>	sheets	146	to	146	(1)	pages	146-146	8	lines
80	<i>chdir.s</i>	sheets	147	to	147	(1)	pages	147-147	12	lines
81	<i>chmod.s</i>	sheets	148	to	148	(1)	pages	148-148	8	lines
82	<i>chown.s</i>	sheets	149	to	149	(1)	pages	149-149	8	lines
83	<i>chroot.s</i>	sheets	150	to	150	(1)	pages	150-150	8	lines
84	<i>close.s</i>	sheets	151	to	151	(1)	pages	151-151	8	lines
85	<i>closedir.s</i>	sheets	152	to	152	(1)	pages	152-152	8	lines
86	<i>creat.s</i>	sheets	153	to	153	(1)	pages	153-153	8	lines
87	<i>devctl.s</i>	sheets	154	to	154	(1)	pages	154-154	8	lines
88	<i>dup.s</i>	sheets	155	to	155	(1)	pages	155-155	8	lines
89	<i>dup2.s</i>	sheets	156	to	156	(1)	pages	156-156	8	lines
90	<i>execl.s</i>	sheets	157	to	157	(1)	pages	157-157	8	lines
91	<i>execle.s</i>	sheets	158	to	158	(1)	pages	158-158	8	lines
92	<i>execvp.s</i>	sheets	159	to	159	(1)	pages	159-159	8	lines
93	<i>execv.s</i>	sheets	160	to	160	(1)	pages	160-160	8	lines
94	<i>execve.s</i>	sheets	161	to	161	(1)	pages	161-161	8	lines
95	<i>execvp.s</i>	sheets	162	to	162	(1)	pages	162-162	8	lines
96	<i>fcntl.s</i>	sheets	163	to	163	(1)	pages	163-163	8	lines
97	<i>fork.s</i>	sheets	164	to	164	(1)	pages	164-164	8	lines
98	<i>fpathconf.s</i>	sheets	165	to	165	(1)	pages	165-165	8	lines
99	<i>fstat.s</i>	sheets	166	to	166	(1)	pages	166-166	8	lines
100	<i>fstatfs.s</i>	sheets	167	to	167	(1)	pages	167-167	8	lines
101	<i>getcwd.s</i>	sheets	168	to	168	(1)	pages	168-168	8	lines
102	<i>getegid.s</i>	sheets	169	to	169	(1)	pages	169-169	8	lines
103	<i>geteuid.s</i>	sheets	170	to	170	(1)	pages	170-170	8	lines
104	<i>getgid.s</i>	sheets	171	to	171	(1)	pages	171-171	8	lines
105	<i>getgroups.s</i>	sheets	172	to	172	(1)	pages	172-172	8	lines
106	<i>getnpid.s</i>	sheets	173	to	173	(1)	pages	173-173	8	lines
107	<i>getnprocnr.s</i>	sheets	174	to	174	(1)	pages	174-174	8	lines
108	<i>getpgrp.s</i>	sheets	175	to	175	(1)	pages	175-175	8	lines
109	<i>getpid.s</i>	sheets	176	to	176	(1)	pages	176-176	8	lines
110	<i>getppid.s</i>	sheets	177	to	177	(1)	pages	177-177	8	lines
111	<i>getpprocnr.s</i>	sheets	178	to	178	(1)	pages	178-178	8	lines
112	<i>getprocnr.s</i>	sheets	179	to	179	(1)	pages	179-179	8	lines
113	<i>getsigset.s</i>	sheets	180	to	180	(1)	pages	180-180	8	lines
114	<i>getsysinfo.s</i>	sheets	181	to	181	(1)	pages	181-181	8	lines
115	<i>getuid.s</i>	sheets	182	to	182	(1)	pages	182-182	8	lines
116	<i>ioctl.s</i>	sheets	183	to	183	(1)	pages	183-183	8	lines
117	<i>isatty.s</i>	sheets	184	to	184	(1)	pages	184-184	8	lines
118	<i>kill.s</i>	sheets	185	to	185	(1)	pages	185-185	8	lines
119	<i>link.s</i>	sheets	186	to	186	(1)	pages	186-186	8	lines
120	<i>lseek.s</i>	sheets	187	to	187	(1)	pages	187-187	8	lines
121	<i>lstat.s</i>	sheets	188	to	188	(1)	pages	188-188	8	lines
122	<i>mkdir.s</i>	sheets	189	to	189	(1)	pages	189-189	8	lines
123	<i>mkfifo.s</i>	sheets	190	to	190	(1)	pages	190-190	8	lines
124	<i>mknod.s</i>	sheets	191	to	191	(1)	pages	191-191	8	lines
125	<i>mount.s</i>	sheets	192	to	192	(1)	pages	192-192	8	lines
126	<i>open.s</i>	sheets	193	to	193	(1)	pages	193-193	8	lines
127	<i>opendir.s</i>	sheets	194	to	194	(1)	pages	194-194	8	lines
128	<i>pathconf.s</i>	sheets	195	to	195	(1)	pages	195-195	8	lines
129	<i>pause.s</i>	sheets	196	to	196	(1)	pages	196-196	8	lines
130	<i>pipe.s</i>	sheets	197	to	197	(1)	pages	197-197	8	lines
131	<i>ptrace.s</i>	sheets	198	to	198	(1)	pages	198-198	8	lines
132	<i>read.s</i>	sheets	199	to	199	(1)	pages	199-199	8	lines
133	<i>readdir.s</i>	sheets	200	to	200	(1)	pages	200-200	8	lines
134	<i>readlink.s</i>	sheets	201	to	201	(1)	pages	201-201	8	lines
135	<i>reboot.s</i>	sheets	202	to	202	(1)	pages	202-202	8	lines
136	<i>rename.s</i>	sheets	203	to	203	(1)	pages	203-203	8	lines
137	<i>rewinddir.s</i>	sheets	204	to	204	(1)	pages	204-204	8	lines
138	<i>rmdir.s</i>	sheets	205	to	205	(1)	pages	205-205	8	lines
139	<i>sbrk.s</i>	sheets	206	to	206	(1)	pages	206-206	8	lines
140	<i>seekdir.s</i>	sheets	207	to	207	(1)	pages	207-207	8	lines
141	<i>setgid.s</i>	sheets	208	to	208	(1)	pages	208-208	12	lines
142	<i>setsid.s</i>	sheets	209	to	209	(1)	pages	209-209	8	lines
143	<i>setuid.s</i>	sheets	210	to	210	(1)	pages	210-210	12	lines
144	<i>sigaction.s</i>	sheets	211	to	211	(1)	pages	211-211	8	lines
145	<i>sigaddset.s</i>	sheets	212	to	212	(1)	pages	212-212	8	lines
146	<i>sigdelset.s</i>	sheets	213	to	213	(1)	pages	213-213	8	lines
147	<i>sigemptyset.s</i>	sheets	214	to	214	(1)	pages	214-214	8	lines

148	<i>sigfillset.s</i>	sheets	215	to	215	(1)	pages	215-215	8	lines
149	<i>sigismember.s</i>	sheets	216	to	216	(1)	pages	216-216	8	lines
150	<i>sigpending.s</i>	sheets	217	to	217	(1)	pages	217-217	8	lines
151	<i>sigprocmask.s</i>	sheets	218	to	218	(1)	pages	218-218	8	lines
152	<i>sigreturn.s</i>	sheets	219	to	219	(1)	pages	219-219	8	lines
153	<i>sigsuspend.s</i>	sheets	220	to	220	(1)	pages	220-220	8	lines
154	<i>sleep.s</i>	sheets	221	to	221	(1)	pages	221-221	8	lines
155	<i>stat.s</i>	sheets	222	to	222	(1)	pages	222-222	8	lines
156	<i>stime.s</i>	sheets	223	to	223	(1)	pages	223-223	8	lines
157	<i>svrctl.s</i>	sheets	224	to	224	(1)	pages	224-224	8	lines
158	<i>symlink.s</i>	sheets	225	to	225	(1)	pages	225-225	8	lines
159	<i>sync.s</i>	sheets	226	to	226	(1)	pages	226-226	8	lines
160	<i>tcdrain.s</i>	sheets	227	to	227	(1)	pages	227-227	8	lines
161	<i>tcflow.s</i>	sheets	228	to	228	(1)	pages	228-228	8	lines
162	<i>tcflush.s</i>	sheets	229	to	229	(1)	pages	229-229	8	lines
163	<i>tcgetattr.s</i>	sheets	230	to	230	(1)	pages	230-230	8	lines
164	<i>tcsendbreak.s</i>	sheets	231	to	231	(1)	pages	231-231	8	lines
165	<i>tcsetattr.s</i>	sheets	232	to	232	(1)	pages	232-232	8	lines
166	<i>time.s</i>	sheets	233	to	233	(1)	pages	233-233	8	lines
167	<i>times.s</i>	sheets	234	to	234	(1)	pages	234-234	8	lines
168	<i>truncate.s</i>	sheets	235	to	235	(1)	pages	235-235	14	lines
169	<i>umask.s</i>	sheets	236	to	236	(1)	pages	236-236	8	lines
170	<i>umount.s</i>	sheets	237	to	237	(1)	pages	237-237	8	lines
171	<i>uname.s</i>	sheets	238	to	238	(1)	pages	238-238	8	lines
172	<i>utime.s</i>	sheets	239	to	239	(1)	pages	239-239	8	lines
173	<i>unlink.s</i>	sheets	240	to	240	(1)	pages	240-240	8	lines
174	<i>wait.s</i>	sheets	241	to	241	(1)	pages	241-241	8	lines
175	<i>waitpid.s</i>	sheets	242	to	242	(1)	pages	242-242	8	lines
176	<i>write.s</i>	sheets	243	to	243	(1)	pages	243-243	8	lines
177	<i>fchmod.s</i>	sheets	244	to	244	(1)	pages	244-244	8	lines
178	<i>fchown.s</i>	sheets	245	to	245	(1)	pages	245-245	8	lines
179	<i>Makefile.in</i>	sheets	246	to	246	(1)	pages	246-246	64	lines
180	<i>assert.c</i>	sheets	247	to	247	(1)	pages	247-247	15	lines
181	<i>panic.c</i>	sheets	248	to	248	(1)	pages	248-248	46	lines
182	<i>pci.h</i>	sheets	249	to	249	(1)	pages	249-249	2	lines
183	<i>pci_attr_r16.c</i>	sheets	250	to	250	(1)	pages	250-250	33	lines
184	<i>pci_attr_r32.c</i>	sheets	251	to	251	(1)	pages	251-251	33	lines
185	<i>pci_attr_r8.c</i>	sheets	252	to	252	(1)	pages	252-252	33	lines
186	<i>pci_attr_w16.c</i>	sheets	253	to	253	(1)	pages	253-253	33	lines
187	<i>pci_attr_w32.c</i>	sheets	254	to	254	(1)	pages	254-254	33	lines
188	<i>pci_attr_w8.c</i>	sheets	255	to	255	(1)	pages	255-255	33	lines
189	<i>pci_dev_name.c</i>	sheets	256	to	256	(1)	pages	256-256	49	lines
190	<i>pci_find_dev.c</i>	sheets	257	to	257	(1)	pages	257-257	44	lines
191	<i>pci_first_dev.c</i>	sheets	258	to	258	(1)	pages	258-258	43	lines
192	<i>pci_ids.c</i>	sheets	259	to	259	(1)	pages	259-259	34	lines
193	<i>pci_init.c</i>	sheets	260	to	260	(1)	pages	260-260	16	lines
194	<i>pci_init1.c</i>	sheets	261	to	261	(1)	pages	261-261	44	lines
195	<i>pci_next_dev.c</i>	sheets	262	to	262	(1)	pages	262-262	44	lines
196	<i>pci_rescan_bus.c</i>	sheets	263	to	263	(1)	pages	263-263	32	lines
197	<i>pci_reserve.c</i>	sheets	264	to	264	(1)	pages	264-264	29	lines
198	<i>pci_slot_name.c</i>	sheets	265	to	265	(1)	pages	265-265	40	lines
199	<i>sys_abort.c</i>	sheets	266	to	266	(1)	pages	266-266	22	lines
200	<i>sys_endsig.c</i>	sheets	267	to	267	(1)	pages	267-267	17	lines
201	<i>sys_eniop.c</i>	sheets	268	to	268	(1)	pages	268-268	15	lines
202	<i>sys_exec.c</i>	sheets	269	to	269	(1)	pages	269-269	19	lines
203	<i>sys_exit.c</i>	sheets	270	to	270	(1)	pages	270-270	18	lines
204	<i>sys_fork.c</i>	sheets	271	to	271	(1)	pages	271-271	21	lines
205	<i>sys_getinfo.c</i>	sheets	272	to	272	(1)	pages	272-272	26	lines
206	<i>sys_getsig.c</i>	sheets	273	to	273	(1)	pages	273-273	19	lines
207	<i>sys_in.c</i>	sheets	274	to	274	(1)	pages	274-274	23	lines
208	<i>sys_int86.c</i>	sheets	275	to	275	(1)	pages	275-275	18	lines
209	<i>sys_irqctl.c</i>	sheets	276	to	276	(1)	pages	276-276	27	lines
210	<i>sys_kill.c</i>	sheets	277	to	277	(1)	pages	277-277	15	lines
211	<i>sys_memset.c</i>	sheets	278	to	278	(1)	pages	278-278	17	lines
212	<i>sys_newmap.c</i>	sheets	279	to	279	(1)	pages	279-279	15	lines
213	<i>sys_nice.c</i>	sheets	280	to	280	(1)	pages	280-280	14	lines
214	<i>sys_out.c</i>	sheets	281	to	281	(1)	pages	281-281	21	lines
215	<i>sys_physcopy.c</i>	sheets	282	to	282	(1)	pages	282-282	31	lines
216	<i>sys_privctl.c</i>	sheets	283	to	283	(1)	pages	283-283	14	lines
217	<i>sys_sdevio.c</i>	sheets	284	to	284	(1)	pages	284-284	27	lines
218	<i>sys_segctl.c</i>	sheets	285	to	285	(1)	pages	285-285	25	lines
219	<i>sys_setalarm.c</i>	sheets	286	to	286	(1)	pages	286-286	19	lines
220	<i>sys_sigreturn.c</i>	sheets	287	to	287	(1)	pages	287-287	19	lines
221	<i>sys_sigsend.c</i>	sheets	288	to	288	(1)	pages	288-288	19	lines

222	<i>sys_times.c</i>	sheets	289	to	289	(1)	pages	289-289	20	lines
223	<i>sys_svrctl.c</i>	sheets	290	to	290	(1)	pages	290-290	14	lines
224	<i>sys_trace.c</i>	sheets	291	to	291	(1)	pages	291-291	18	lines
225	<i>sys_umap.c</i>	sheets	292	to	292	(1)	pages	292-292	26	lines
226	<i>sys_vinb.c</i>	sheets	293	to	293	(1)	pages	293-293	19	lines
227	<i>sys_vinl.c</i>	sheets	294	to	294	(1)	pages	294-294	19	lines
228	<i>sys_vinw.c</i>	sheets	295	to	295	(1)	pages	295-295	20	lines
229	<i>sys_vircopy.c</i>	sheets	296	to	296	(1)	pages	296-296	30	lines
230	<i>sys_vm_map.c</i>	sheets	297	to	297	(1)	pages	297-297	26	lines
231	<i>sys_vm_setbuf.c</i>	sheets	298	to	298	(1)	pages	298-298	22	lines
232	<i>sys_voutb.c</i>	sheets	299	to	299	(1)	pages	299-299	19	lines
233	<i>sys_voutl.c</i>	sheets	300	to	300	(1)	pages	300-300	19	lines
234	<i>sys_voutw.c</i>	sheets	301	to	301	(1)	pages	301-301	20	lines
235	<i>syslib.h</i>	sheets	302	to	302	(1)	pages	302-302	8	lines
236	<i>taskcall.c</i>	sheets	303	to	303	(1)	pages	303-303	21	lines
237	<i>Makefile.in</i>	sheets	304	to	304	(1)	pages	304-304	22	lines
238	<i>env_get_prm.c</i>	sheets	305	to	306	(2)	pages	305-306	98	lines
239	<i>env_panic.c</i>	sheets	307	to	307	(1)	pages	307-307	20	lines
240	<i>env_parse.c</i>	sheets	308	to	309	(2)	pages	308-309	91	lines
241	<i>env_prefix.c</i>	sheets	310	to	310	(1)	pages	310-310	31	lines
242	<i>fkey_ctl.c</i>	sheets	311	to	311	(1)	pages	311-311	28	lines
243	<i>getuptime.c</i>	sheets	312	to	312	(1)	pages	312-312	23	lines
244	<i>kmalloc.c</i>	sheets	313	to	315	(3)	pages	313-315	178	lines
245	<i>kprintf.c</i>	sheets	316	to	318	(3)	pages	316-318	192	lines
246	<i>kputc.c</i>	sheets	319	to	319	(1)	pages	319-319	48	lines
247	<i>report.c</i>	sheets	320	to	320	(1)	pages	320-320	21	lines
248	<i>sysutil.h</i>	sheets	321	to	321	(1)	pages	321-321	9	lines
249	<i>taskcall.c</i>	sheets	322	to	322	(1)	pages	322-322	21	lines
250	<i>tickdelay.c</i>	sheets	323	to	323	(1)	pages	323-323	45	lines
251	<i>Makefile.in</i>	sheets	324	to	324	(1)	pages	324-324	13	lines
252	<i>timers.h</i>	sheets	325	to	325	(1)	pages	325-325	8	lines
253	<i>tmrs_clr.c</i>	sheets	326	to	326	(1)	pages	326-326	41	lines
254	<i>tmrs_exp.c</i>	sheets	327	to	327	(1)	pages	327-327	32	lines
255	<i>tmrs_set.c</i>	sheets	328	to	328	(1)	pages	328-328	40	lines
256	<i>Makefile.in</i>	sheets	329	to	329	(1)	pages	329-329	10	lines
257	<i>openpty.c</i>	sheets	330	to	331	(2)	pages	330-331	83	lines
258	<i>ChangeLog</i>	sheets	332	to	343	(12)	pages	332-343	856	lines
259	<i>FAQ</i>	sheets	344	to	348	(5)	pages	344-348	340	lines
260	<i>INDEX</i>	sheets	349	to	349	(1)	pages	349-349	52	lines
261	<i>Makefile.in</i>	sheets	350	to	350	(1)	pages	350-350	13	lines
262	<i>README</i>	sheets	351	to	352	(2)	pages	351-352	126	lines
263	<i>adler32.c</i>	sheets	353	to	355	(3)	pages	353-355	150	lines
264	<i>algorithm.txt</i>	sheets	356	to	358	(3)	pages	356-358	210	lines
265	<i>compress.c</i>	sheets	359	to	360	(2)	pages	359-360	80	lines
266	<i>configure</i>	sheets	361	to	367	(7)	pages	361-367	460	lines
267	<i>crc32.c</i>	sheets	368	to	373	(6)	pages	368-373	424	lines
268	<i>crc32.h</i>	sheets	374	to	379	(6)	pages	374-379	442	lines
269	<i>deflate.c</i>	sheets	380	to	403	(24)	pages	380-403	1737	lines
270	<i>deflate.h</i>	sheets	404	to	408	(5)	pages	404-408	332	lines
271	<i>example.c</i>	sheets	409	to	416	(8)	pages	409-416	566	lines
272	<i>gzio.c</i>	sheets	417	to	430	(14)	pages	417-430	1027	lines
273	<i>infbac.c</i>	sheets	431	to	439	(9)	pages	431-439	624	lines
274	<i>inffast.c</i>	sheets	440	to	444	(5)	pages	440-444	319	lines
275	<i>inffast.h</i>	sheets	445	to	445	(1)	pages	445-445	12	lines
276	<i>inffixed.h</i>	sheets	446	to	447	(2)	pages	446-447	95	lines
277	<i>inflate.c</i>	sheets	448	to	466	(19)	pages	448-466	1369	lines
278	<i>inflate.h</i>	sheets	467	to	468	(2)	pages	467-468	116	lines
279	<i>inftrees.c</i>	sheets	469	to	473	(5)	pages	469-473	330	lines
280	<i>inftrees.h</i>	sheets	474	to	474	(1)	pages	474-474	56	lines
281	<i>make_vms.com</i>	sheets	475	to	481	(7)	pages	475-481	462	lines
282	<i>trees.c</i>	sheets	482	to	498	(17)	pages	482-498	1220	lines
283	<i>minigzip.c</i>	sheets	499	to	503	(5)	pages	499-503	323	lines
284	<i>trees.h</i>	sheets	504	to	505	(2)	pages	504-505	129	lines
285	<i>uncompr.c</i>	sheets	506	to	506	(1)	pages	506-506	62	lines
286	<i>zconf.h</i>	sheets	507	to	511	(5)	pages	507-511	333	lines
287	<i>zconf.in.h</i>	sheets	512	to	516	(5)	pages	512-516	333	lines
288	<i>zlib.3</i>	sheets	517	to	519	(3)	pages	517-519	160	lines
289	<i>zlib.h</i>	sheets	520	to	538	(19)	pages	520-538	1358	lines
290	<i>zutil.c</i>	sheets	539	to	543	(5)	pages	539-543	319	lines
291	<i>zutil.h</i>	sheets	544	to	547	(4)	pages	544-547	270	lines
292	<i>Makefile.pup</i>	sheets	548	to	548	(1)	pages	548-548	67	lines
293	<i>Makefile.sas</i>	sheets	549	to	549	(1)	pages	549-549	66	lines
294	<i>bndsrc</i>	sheets	550	to	551	(2)	pages	550-551	133	lines
295	<i>compile.clp</i>	sheets	552	to	553	(2)	pages	552-553	124	lines

296	readme.txt.....	sheets	554	to	555	(2)	pages	554-555	112	lines
297	zlib.inc.....	sheets	556	to	562	(7)	pages	556-562	332	lines
298	README.contrib.....	sheets	563	to	563	(1)	pages	563-563	72	lines
299	buffer_demo.adb.....	sheets	564	to	565	(2)	pages	564-565	107	lines
300	mtest.adb.....	sheets	566	to	568	(3)	pages	566-568	157	lines
301	read.adb.....	sheets	569	to	571	(3)	pages	569-571	157	lines
302	readme.txt.....	sheets	572	to	572	(1)	pages	572-572	66	lines
303	test.adb.....	sheets	573	to	579	(7)	pages	573-579	464	lines
304	zlib-streams.adb....	sheets	580	to	583	(4)	pages	580-583	226	lines
305	zlib-streams.ads....	sheets	584	to	585	(2)	pages	584-585	115	lines
306	zlib-thin.adb.....	sheets	586	to	587	(2)	pages	586-587	142	lines
307	zlib-thin.ads.....	sheets	588	to	594	(7)	pages	588-594	451	lines
308	zlib.adb.....	sheets	595	to	604	(10)	pages	595-604	702	lines
309	zlib.ads.....	sheets	605	to	609	(5)	pages	605-609	329	lines
310	zlib.gpr.....	sheets	610	to	610	(1)	pages	610-610	21	lines
311	README.586.....	sheets	611	to	611	(1)	pages	611-611	44	lines
312	match.S.....	sheets	612	to	616	(5)	pages	612-616	365	lines
313	README.686.....	sheets	617	to	617	(1)	pages	617-617	35	lines
314	match.S.....	sheets	618	to	622	(5)	pages	618-622	330	lines
315	Makefile.....	sheets	623	to	623	(1)	pages	623-623	9	lines
316	README.....	sheets	624	to	624	(1)	pages	624-624	5	lines
317	blast.c.....	sheets	625	to	630	(6)	pages	625-630	445	lines
318	blast.h.....	sheets	631	to	631	(1)	pages	631-631	72	lines
319	test.pk.....	sheets	631	to	631	(1)	pages	631-631	1	lines
320	test.txt.....	sheets	632	to	632	(1)	pages	632-632	2	lines
321	ZLib.pas.....	sheets	633	to	640	(8)	pages	633-640	558	lines
322	ZLibConst.pas.....	sheets	641	to	641	(1)	pages	641-641	12	lines
323	readme.txt.....	sheets	642	to	643	(2)	pages	642-643	77	lines
324	zlibd32.mak.....	sheets	644	to	645	(2)	pages	644-645	94	lines
325	DotZLib.build.....	sheets	646	to	646	(1)	pages	646-646	34	lines
326	DotZLib.chm.....	sheets	647	to	871	(225)	pages	647-871	432	lines
327	DotZLib.sln.....	sheets	872	to	872	(1)	pages	872-872	22	lines
328	LICENSE_1_0.txt.....	sheets	873	to	873	(1)	pages	873-873	24	lines
329	readme.txt.....	sheets	874	to	874	(1)	pages	874-874	59	lines
330	AssemblyInfo.cs.....	sheets	875	to	875	(1)	pages	875-875	59	lines
331	ChecksumImpl.cs.....	sheets	876	to	879	(4)	pages	876-879	203	lines
332	CircularBuffer.cs...	sheets	880	to	881	(2)	pages	880-881	84	lines
333	CodecBase.cs.....	sheets	882	to	884	(3)	pages	882-884	199	lines
334	Deflater.cs.....	sheets	885	to	886	(2)	pages	885-886	107	lines
335	DotZLib.cs.....	sheets	887	to	891	(5)	pages	887-891	289	lines
336	DotZLib.csproj.....	sheets	892	to	893	(2)	pages	892-893	142	lines
337	GZipStream.cs.....	sheets	894	to	898	(5)	pages	894-898	302	lines
338	Inflater.cs.....	sheets	899	to	900	(2)	pages	899-900	106	lines
339	UnitTests.cs.....	sheets	901	to	904	(4)	pages	901-904	275	lines
340	README.....	sheets	905	to	905	(1)	pages	905-905	2	lines
341	infbck9.c.....	sheets	906	to	914	(9)	pages	906-914	609	lines
342	infbck9.h.....	sheets	915	to	915	(1)	pages	915-915	38	lines
343	inffix9.h.....	sheets	916	to	917	(2)	pages	916-917	108	lines
344	inflate9.h.....	sheets	918	to	918	(1)	pages	918-918	48	lines
345	inftree9.c.....	sheets	919	to	923	(5)	pages	919-923	324	lines
346	inftree9.h.....	sheets	924	to	924	(1)	pages	924-924	56	lines
347	inffas86.c.....	sheets	925	to	940	(16)	pages	925-940	1158	lines
348	inffast.S.....	sheets	941	to	959	(19)	pages	941-959	1369	lines
349	test.cpp.....	sheets	960	to	960	(1)	pages	960-960	25	lines
350	zfstream.cpp.....	sheets	961	to	965	(5)	pages	961-965	330	lines
351	zfstream.h.....	sheets	966	to	967	(2)	pages	966-967	129	lines
352	zstream.h.....	sheets	968	to	972	(5)	pages	968-972	308	lines
353	zstream_test.cpp....	sheets	973	to	973	(1)	pages	973-973	26	lines
354	README.....	sheets	974	to	974	(1)	pages	974-974	36	lines
355	TODO.....	sheets	975	to	975	(1)	pages	975-975	18	lines
356	test.cc.....	sheets	976	to	976	(1)	pages	976-976	51	lines
357	zfstream.cc.....	sheets	977	to	983	(7)	pages	977-983	480	lines
358	zfstream.h.....	sheets	984	to	990	(7)	pages	984-990	467	lines
359	match.asm.....	sheets	991	to	996	(6)	pages	991-996	414	lines
360	bld_ml64.bat.....	sheets	997	to	997	(1)	pages	997-997	3	lines
361	gvmat64.asm.....	sheets	998	to	1004	(7)	pages	998-1004	514	lines
362	gvmat64.obj.....	sheets	1004	to	1004	(1)	pages	1004-1004	1	lines
363	inffas8664.c.....	sheets	1005	to	1007	(3)	pages	1005-1007	187	lines
364	inffasx64.asm.....	sheets	1008	to	1013	(6)	pages	1008-1013	393	lines
365	inffasx64.obj.....	sheets	1013	to	1013	(1)	pages	1013-1013	1	lines
366	readme.txt.....	sheets	1014	to	1014	(1)	pages	1014-1014	29	lines
367	bld_ml32.bat.....	sheets	1015	to	1015	(1)	pages	1015-1015	3	lines
368	gvmat32.asm.....	sheets	1016	to	1029	(14)	pages	1016-1029	973	lines
369	gvmat32.obj.....	sheets	1030	to	1051	(22)	pages	1030-1051	97	lines

370	<i>gvmat32c.c</i>	sheets	1052	to	1052	(1)	pages	1052-1052	63	lines
371	<i>inffas32.asm</i>	sheets	1053	to	1067	(15)	pages	1053-1067	1084	lines
372	<i>inffas32.obj</i>	sheets	1068	to	1123	(56)	pages	1068-1123	203	lines
373	<i>mkasm.bat</i>	sheets	1124	to	1124	(1)	pages	1124-1124	4	lines
374	<i>readme.txt</i>	sheets	1125	to	1125	(1)	pages	1125-1125	22	lines
375	<i>ChangeLogUnzip</i>	sheets	1126	to	1126	(1)	pages	1126-1126	68	lines
376	<i>Makefile</i>	sheets	1127	to	1127	(1)	pages	1127-1127	26	lines
377	<i>crypt.h</i>	sheets	1128	to	1129	(2)	pages	1128-1129	133	lines
378	<i>ioapi.c</i>	sheets	1130	to	1132	(3)	pages	1130-1132	178	lines
379	<i>ioapi.h</i>	sheets	1133	to	1134	(2)	pages	1133-1134	76	lines
380	<i>iowin32.c</i>	sheets	1135	to	1138	(4)	pages	1135-1138	271	lines
381	<i>iowin32.h</i>	sheets	1139	to	1139	(1)	pages	1139-1139	22	lines
382	<i>miniunz.c</i>	sheets	1140	to	1147	(8)	pages	1140-1147	586	lines
383	<i>minizip.c</i>	sheets	1148	to	1153	(6)	pages	1148-1153	421	lines
384	<i>mztools.c</i>	sheets	1154	to	1157	(4)	pages	1154-1157	282	lines
385	<i>mztools.h</i>	sheets	1158	to	1158	(1)	pages	1158-1158	32	lines
386	<i>unzip.c</i>	sheets	1159	to	1180	(22)	pages	1159-1180	1599	lines
387	<i>unzip.h</i>	sheets	1181	to	1185	(5)	pages	1181-1185	355	lines
388	<i>zip.c</i>	sheets	1186	to	1202	(17)	pages	1186-1202	1220	lines
389	<i>zip.h</i>	sheets	1203	to	1206	(4)	pages	1203-1206	236	lines
390	<i>example.pas</i>	sheets	1207	to	1215	(9)	pages	1207-1215	600	lines
391	<i>readme.txt</i>	sheets	1216	to	1217	(2)	pages	1216-1217	77	lines
392	<i>zlibd32.mak</i>	sheets	1218	to	1219	(2)	pages	1218-1219	94	lines
393	<i>zlibpas.pas</i>	sheets	1220	to	1223	(4)	pages	1220-1223	237	lines
394	<i>Makefile</i>	sheets	1224	to	1224	(1)	pages	1224-1224	9	lines
395	<i>README</i>	sheets	1225	to	1225	(1)	pages	1225-1225	64	lines
396	<i>puff.c</i>	sheets	1226	to	1237	(12)	pages	1226-1237	838	lines
397	<i>puff.h</i>	sheets	1238	to	1238	(1)	pages	1238-1238	32	lines
398	<i>zeros.raw</i>	sheets	1238	to	1238	(1)	pages	1238-1238	1	lines
399	<i>testzlib.c</i>	sheets	1239	to	1242	(4)	pages	1239-1242	276	lines
400	<i>testzlib.txt</i>	sheets	1243	to	1243	(1)	pages	1243-1243	11	lines
401	<i>Makefile</i>	sheets	1244	to	1244	(1)	pages	1244-1244	15	lines
402	<i>Makefile.msc</i>	sheets	1245	to	1245	(1)	pages	1245-1245	18	lines
403	<i>untgz.c</i>	sheets	1246	to	1255	(10)	pages	1246-1255	675	lines
404	<i>readme.txt</i>	sheets	1256	to	1256	(1)	pages	1256-1256	74	lines
405	<i>miniunz.vcproj</i>	sheets	1257	to	1258	(2)	pages	1257-1258	127	lines
406	<i>minizip.vcproj</i>	sheets	1259	to	1260	(2)	pages	1259-1260	127	lines
407	<i>testzlib.vcproj</i>	sheets	1261	to	1262	(2)	pages	1261-1262	127	lines
408	<i>zlib.rc</i>	sheets	1263	to	1263	(1)	pages	1263-1263	33	lines
409	<i>zlibstat.vcproj</i>	sheets	1264	to	1267	(4)	pages	1264-1267	247	lines
410	<i>zlibvc.def</i>	sheets	1268	to	1269	(2)	pages	1268-1269	93	lines
411	<i>zlibvc.sln</i>	sheets	1270	to	1271	(2)	pages	1270-1271	79	lines
412	<i>zlibvc.vcproj</i>	sheets	1272	to	1278	(7)	pages	1272-1278	446	lines
413	<i>miniunz.vcproj</i>	sheets	1279	to	1286	(8)	pages	1279-1286	567	lines
414	<i>minizip.vcproj</i>	sheets	1287	to	1294	(8)	pages	1287-1294	564	lines
415	<i>testzlib.vcproj</i>	sheets	1295	to	1307	(13)	pages	1295-1307	949	lines
416	<i>testzlibdll.vcproj</i> ..	sheets	1308	to	1315	(8)	pages	1308-1315	568	lines
417	<i>zlib.rc</i>	sheets	1316	to	1316	(1)	pages	1316-1316	33	lines
418	<i>zlibstat.vcproj</i>	sheets	1317	to	1328	(12)	pages	1317-1328	871	lines
419	<i>zlibvc.def</i>	sheets	1329	to	1330	(2)	pages	1329-1330	93	lines
420	<i>zlibvc.sln</i>	sheets	1331	to	1334	(4)	pages	1331-1334	145	lines
421	<i>zlibvc.vcproj</i>	sheets	1335	to	1351	(17)	pages	1335-1351	1220	lines
422	<i>README.examples</i>	sheets	1352	to	1352	(1)	pages	1352-1352	43	lines
423	<i>fitblk.c</i>	sheets	1353	to	1356	(4)	pages	1353-1356	234	lines
424	<i>gun.c</i>	sheets	1357	to	1366	(10)	pages	1357-1366	694	lines
425	<i>gzappend.c</i>	sheets	1367	to	1373	(7)	pages	1367-1373	501	lines
426	<i>gzjoin.c</i>	sheets	1374	to	1380	(7)	pages	1374-1380	449	lines
427	<i>gzlog.c</i>	sheets	1381	to	1386	(6)	pages	1381-1386	414	lines
428	<i>gzlog.h</i>	sheets	1387	to	1387	(1)	pages	1387-1387	59	lines
429	<i>zlib_how.html</i>	sheets	1387	to	1387	(1)	pages	1387-1387	1	lines
430	<i>zpipe.c</i>	sheets	1388	to	1390	(3)	pages	1388-1390	192	lines
431	<i>zran.c</i>	sheets	1391	to	1396	(6)	pages	1391-1396	405	lines
432	<i>Makefile.bor</i>	sheets	1397	to	1398	(2)	pages	1397-1398	110	lines
433	<i>Makefile.dj2</i>	sheets	1399	to	1400	(2)	pages	1399-1400	105	lines
434	<i>Makefile.emx</i>	sheets	1401	to	1401	(1)	pages	1401-1401	70	lines
435	<i>Makefile.msc</i>	sheets	1402	to	1403	(2)	pages	1402-1403	107	lines
436	<i>Makefile.tc</i>	sheets	1404	to	1405	(2)	pages	1404-1405	95	lines
437	<i>Makefile.riscos</i>	sheets	1406	to	1408	(3)	pages	1406-1408	152	lines
438	<i>README</i>	sheets	1409	to	1409	(1)	pages	1409-1409	4	lines
439	<i>descrip.mms</i>	sheets	1410	to	1410	(1)	pages	1410-1410	49	lines
440	<i>visual-basic.txt</i>	sheets	1411	to	1413	(3)	pages	1411-1413	161	lines
441	<i>zlib.html</i>	sheets	1413	to	1413	(1)	pages	1413-1413	1	lines
442	<i>Makefile.os2</i>	sheets	1414	to	1415	(2)	pages	1414-1415	137	lines
443	<i>zlib.def</i>	sheets	1416	to	1416	(1)	pages	1416-1416	52	lines

444	<i>README.projects.....</i>	sheets	1417	to	1417	(1)	pages	1417-1417	42	lines
445	<i>README.txt.....</i>	sheets	1418	to	1418	(1)	pages	1418-1418	74	lines
446	<i>example.dsp.....</i>	sheets	1419	to	1422	(4)	pages	1419-1422	279	lines
447	<i>minigzip.dsp.....</i>	sheets	1423	to	1426	(4)	pages	1423-1426	279	lines
448	<i>zlib.dsp.....</i>	sheets	1427	to	1435	(9)	pages	1427-1435	610	lines
449	<i>zlib.dsw.....</i>	sheets	1436	to	1436	(1)	pages	1436-1436	60	lines
450	<i>package.qpg.....</i>	sheets	1437	to	1439	(3)	pages	1437-1439	142	lines
451	<i>DLL_FAQ.txt.....</i>	sheets	1440	to	1445	(6)	pages	1440-1445	398	lines
452	<i>Makefile.bor.....</i>	sheets	1446	to	1447	(2)	pages	1446-1447	108	lines
453	<i>Makefile.emx.....</i>	sheets	1448	to	1448	(1)	pages	1448-1448	70	lines
454	<i>Makefile.gcc.....</i>	sheets	1449	to	1450	(2)	pages	1449-1450	142	lines
455	<i>Makefile.msc.....</i>	sheets	1451	to	1452	(2)	pages	1451-1452	127	lines
456	<i>VisualC.txt.....</i>	sheets	1453	to	1453	(1)	pages	1453-1453	4	lines
457	<i>zlib.def.....</i>	sheets	1454	to	1454	(1)	pages	1454-1454	61	lines
458	<i>zlib1.rc.....</i>	sheets	1455	to	1455	(1)	pages	1455-1455	40	lines