

```
# Makefile for the kernel image.
```

```
u=/usr
CC=      exec cc
CFLAGS=-O -D_MINIX -D_POSIX_SOURCE
MDEC=    /usr/mdec
```

```
# Specify the programs that are part of the system image.
```

```
PROGRAMS=  ../kernel/kernel \
  ../servers/pm/pm \
  ../servers/fs/fs \
  ../servers/rs/rs \
  ../servers/ds/ds \
  ../drivers/tty/tty \
  ../drivers/memory/memory \
  ../drivers/log/log \
  ../servers/init/init
```

usage:

```
@echo " " >&2
@echo "Master Makefile to create new MINIX configuration." >& 2
@echo "Root privileges are required." >&2
@echo " " >&2
@echo "Usage:" >&2
@echo " make includes # Install include files" >&2
@echo " make depend  # Generate dependency files" >&2
@echo " make libraries # Make system libraries" >&2
@echo " make services # Compile and install all services" >&2
@echo " make fresh   # Make clean, libraries, and services" >&2
@echo " make image   # Make needed services and create boot image" >&2
@echo " make install # Make image, and install to hard disk" >&2
@echo " make hdboot  # Make image, and install to hard disk" >&2
@echo " make fdboot  # Make image, and install to floppy disk" >&2
@echo " make bootable # Make hard disk bootable" >&2
@echo " make clean   # Remove all compiler results, except libs" >&2
@echo " " >&2
@echo "To create a fresh MINIX configuration, try:" >&2
@echo " make clean install # new boot image" >&2
@echo " make fresh install # new everything" >&2
@echo " " >&2
```

```
# create a fresh configuration or system image
```

fresh:

```
cd ../lib && $(MAKE) clean
$(MAKE) clean
$(MAKE) libraries services
```

all: services image

image: includes

```
cd ../kernel && $(MAKE)
cd ../servers && $(MAKE) image
cd ../drivers && $(MAKE) image
installboot -image $@ $(PROGRAMS)
```

image_small: includes

```
cd ../kernel && $(MAKE)
cd ../servers && $(MAKE) EXTRA_OPTS=-D_MINIX_SMALL=1 image
cd ../drivers && $(MAKE) EXTRA_OPTS=$(EXTRA_OPTS) image
installboot -image $@ $(PROGRAMS)
```

```
# rebuild the program or system libraries
```

includes:

```
cd ../include && $(MAKE) install
```

depend: includes

```
cd ../ && $(MAKE) depend
```

services: includes

```
cd ../kernel && $(MAKE)
cd ../servers && $(MAKE) install
cd ../drivers && $(MAKE) install
```

libraries: includes

```
cd ../lib && $(MAKE) clean
cd ../lib && $(MAKE) all
cd ../lib && $(MAKE) install
```

make bootable and place system images

bootable:

```
exec su root mkboot bootable
```

hdboot: image

```
exec sh mkboot $@
@sync
```

fdboot: image

```
exec su root mkboot $@
@sync
```

install: includes services hdboot

clean up compile results

clean:

```
cd ../kernel && $(MAKE) $@
cd ../servers && $(MAKE) $@
cd ../drivers && $(MAKE) $@
rm -rf *.bak image image_small *.iso *.iso.gz cdfdimage rootimage src
```

```
#!/bin/sh
set -e
export PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
export SHELL=/bin/sh
cd /usr/src
make etcfiles
su bin -c 'make world install'
cd tools
rm revision
rm /boot/image/*
make install
cp /boot/image/* /boot/image_big # Make big image accessible by this name
cp ../boot/boot /boot/boot
make clean
make image_small
cp image_small /boot
cd /usr/src
make clean
# Let man find the manpages
su bin -c 'makewhatis /usr/man'
su bin -c 'makewhatis /usr/gnu/man'
su bin -c 'makewhatis /usr/local/man'
mv /usr/src/commands /usr/src.commands
binsizes normal
exit 0
```

Welcome to MINIX 3.1.2. The system is now running and many commands work normally. To use MINIX 3.1.2 in a serious way, you need to install it to your hard disk, which you can do by typing 'setup' while logged in as root. Then just follow the on-screen directions.

After setup is complete, type 'shutdown' and when the boot monitor starts, boot your new system by following the instructions at the end of setup. Keep the CD-ROM in the drive, login as root and type 'packman' to begin installing the many software packages available. After you have installed the packages, type 'xdm' to start X Windows if you have installed it.

It is strongly recommended that you print and read the 'setup.pdf' file on the CD-ROM before starting the installation. The file is also available at www.minix3.org/doc under the 'Installation' heading.

```
#!/bin/sh
#
#      mkboot 2.0 - make boot floppy, make root device bootable, etc.
#                                     Author: Kees J. Bot

trap 'e=$?; rm -f /tmp/mkb.$$; exit $e' 0 2

mdec=/usr/mdec # bootstraps

# Check arguments.
case "$#:1" in
1:bootable | 1:hdboot | [12]:fdboot | [12]:cdfdboot )
    action=$1 dev=$2
    ;;
*)
    echo "Usage: $0 [bootable | hdboot | cdfdboot | fdboot [device]] | cdfdboot [device]" >&2
    exit 1
esac

# Get the device table.
. /etc/fstab

# The real root device may be the RAM disk.
realroot='printroot -r'

# If it's an initial fstab, pretend root is real root
if [ $root = "/dev/ROOT" ]
then
    root=$realroot
fi

case $action in
bootable | hdboot)
    # We need the root device.

    if [ $realroot = $root ]
    then
        rootdir=
    else
        umount $root 2>/dev/null
        mount $root /mnt || exit
        rootdir=/mnt
    fi
esac

case $action in
bootable)
    # Install the boot monitor on the root device and make it bootable.
    install -cs -m 644 $mdec/boot $rootdir/boot/boot || exit
    sync
    installboot -device $root $mdec/bootblock /boot/boot || exit
    test $realroot != $root && umount $root
    ;;
hdboot)
    # Install a new image on the root device.
    if [ ! -d $rootdir/boot/image ]
    then
        /boot/image is not yet a directory! Fix it.
        su root -c \
            "exec mv $rootdir/boot/image/M"
        install -d $rootdir/boot/image
        su root -c \
            "exec mv $rootdir/M $rootdir/boot/image/'uname -r'.'uname -v'"
    fi

    sh tell_config OS_RELEASE . OS_VERSION >/tmp/mkb.$$
    version=`sed 's/[ "]/g;/^$/d' </tmp/mkb.$$`

    revision=`cat revision 2>/dev/null`

    if [ -z "$revision" ]
    then
        rrevision=""
    else
        rrevision=r$revision
    fi

    oldrev=$revision
```

```

target="${version}r${revision}"

if [ -z "$revision" ]
then
    revision=0
    rrevision=""
elif [ -f $rootdir/boot/image/$target ]
then
    if [ $rootdir/boot/image/$target -newer image ]
    then
        echo "$root:/boot/image/$target is up to date"
        test $realroot != $root && umount $root
        exit 0
    fi
    revision=`expr $revision + 1`
    rrevision=r$revision
fi
target="${version}${rrevision}"

set -- `ls -t $rootdir/boot/image`

case $# in
0|1|2|3)
    # Not much there, do not remove a thing.
    ;;
*)
    # Remove the third-newest image in /boot/image, but
    # only if there's an older one (which is kept).
    echo "rm $root:/boot/image/$3"
    rm -f "$rootdir/boot/image/$3"
esac

# Install the new image.
echo "install image $root:/boot/image/$target"
install -o root -m 600 image $rootdir/boot/image/$target || exit

# Save the revision number.
test "$revision" != "$oldrev" && echo $revision >revision

test $realroot != $root && umount $root
echo "Done."
;;

fdboot)
    # fdboot: Make a boot floppy.

    if [ -z "$dev" ]
    then
        echo -n \
"Finish the name of the floppy device to write (by default 'fd0'): /dev/" ;
        read dev
        case "$dev" in
            '') dev=/dev/fd0
                ;;
            /dev/*)
                ;;
            *) dev=/dev/$dev
        esac
    fi

    # Make a file system.
    umount $dev 2>/dev/null
    if mkfs -B 1024 -i 512 $dev
    then
        :
    else
        echo "mkfs of $dev failed."
        exit 1;
    fi

    # Install /dev, /boot/boot and /boot/image.
    mount $dev /mnt || exit
    mkdir -p /mnt/boot/image || exit
    cpdir /dev /mnt/dev || exit
    cp -p $mdec/boot /mnt/boot/boot || exit

```

```

cp -p image /mnt/boot/image/ || exit
umount $dev || exit

# Make bootable and copy the boot parameters.
installboot -d $dev $mdec/bootblock /boot/boot || exit
pfile=fdbootparams
if [ -f $pfile ]
then    echo "Using floppy boot parameters from file $pfile."
        edparams $dev "cat $pfile" || exit
else    echo "Copying floppy boot parameters from $root."
        dd if=$root of=$dev skip=1 seek=1 count=1 conv=silent || exit
fi
edparams $dev 'main(){delay 2000;boot}; save' || exit
echo "Test kernel installed on $dev"
;;

cdfdboot)
# cdfdbboot: Make a boot floppy image to go on a CD for booting from.
if [ -z "$dev" ]
then
    ramdisk `expr 1440 \* 1024` 2>/dev/null
    dev=/dev/ram
fi
umount $dev 2>/dev/null
if mkfs -B 1024 -b 1440 -i 512 $dev || exit 1
then :
else
    echo "mkfs of $dev failed."
    exit 1;
fi

# Install /dev, /boot
mount $dev /mnt || exit 1
mkdir /mnt/dev
mkdir /mnt/boot
mkdir /mnt/boot/image
( cd /mnt/dev && sh /usr/src/commands/scripts/MAKEDEV.sh std )
cp -p image image_* /mnt/boot/image || exit 1
cp -p ../boot/boot /mnt/boot/boot || exit 1
umount $dev || exit 1
installboot -d $dev ../boot/bootblock boot/boot || exit 1
edparams $dev 'unset bootopts; unset servers; disable=inet; image=/boot/image/image; bootbig(1, Regular MINIX
3 (requires at least 16 MB RAM)) { image=/boot/image/image ; boot } bootsmall(2, Small MINIX 3 (intended for 8 MB systems)) {
image=/boot/image/image_small ; boot } cdproberoot=1; unset rootdev; unset leader; leader() { echo \n--- Welcome to MINIX 3. T
his is the boot monitor. ---\n\nChoose an option from the menu or press ESC if you need to do anything special.\nOtherwise I will
boot with my defaults in 10 seconds.\n\n }; bootcd=1; main(){trap 10000 boot; menu; }; save' || exit

# copy image
dd if=$dev of=cdfdimage bs=8192 count=180
esac
sync
exit 0

```

```
#!/bin/sh

set -e

PACKAGEDIR=/usr/bigports/Packages
PACKAGESOURCEDIR=/usr/bigports/Sources
secs='expr 32 '*' 64'
export SHELL=/bin/sh

make_hdimage()
{
    dd if=$TMPDISK of=usrimage bs=$BS count=$USRBLOCKS

    rootsize='stat -size rootimage'
    usrszsize='stat -size usrimage'

    rootsects='expr $rootsize / 512'
    usrsects='expr $usrszsize / 512'

    # installboot -m needs at least 1KB
    dd < /dev/zero >tmpimage count=2
    partition -fm tmpimage 2 81:$rootsects* 0:0 81:$usrsects
    installboot -m tmpimage /usr/mdec/masterboot
    dd < tmpimage > subpart count=1

    primsects='expr 1 + $rootsects + $usrsects'
    cyl='expr '(' $primsects ')' / $secs + 1'
    padsects='expr $cyl \* $secs - 1 - $primsects'

    { dd < /dev/zero count=1
      cat subpart
      cat rootimage
      cat usrimage
      dd < /dev/zero count=$padsects
    } > hdimage
    partition -m hdimage 81:'expr $primsects + $padsects'*
    installboot -m hdimage /usr/mdec/masterboot
}

hdemu_root_changes()
{
    $RELEASEDIR/usr/bin/installboot -d $TMPDISK3 \
        $RELEASEDIR/usr/mdec/bootblock boot/boot
    echo \

    'bootcd=2
    disable=inet
    bios_wini=yes
    bios_remap_first=1
    ramimage=dev=c0d7p0s0
    bootbig(1, Regular MINIX 3) { image=/boot/image/image; boot }
    bootsmall(2, Small MINIX 3 (<16MB)) { image=/boot/image/image_small; boot }
    main() { trap 10000 boot ; menu; }
    save' | $RELEASEDIR/usr/bin/edparams $TMPDISK3

    echo \
    'root=/dev/c0d7p0s0
    usr=/dev/c0d7p0s2
    usr_roflag="-r"' > $RELEASEDIR/etc/fstab
}

usb_root_changes()
{
    $RELEASEDIR/usr/bin/installboot -d $TMPDISK3 \
        $RELEASEDIR/usr/mdec/bootblock boot/boot
    echo \

    'bios_wini=yes
    disable=inet
    bios_remap_first=1
    rootdev=c0d7p0s0
    save' | $RELEASEDIR/usr/bin/edparams $TMPDISK3

    echo \
    'root=/dev/c0d7p0s0
    usr=/dev/c0d7p0s2
```



```
' > $RELEASEDIR/etc/fstab
}

COPYITEMS="usr/bin bin usr/lib"
RELEASEDIR=/usr/r
RELEASEPACKAGE=${RELEASEDIR}/usr/install/packages
RELEASEPACKAGESOURCES=${RELEASEDIR}/usr/install/package-sources
IMAGE=cdfdimage
ROOTIMAGE=rootimage
CDFILES=/usr/tmp/cdreleasefiles
sh tell_config OS_RELEASE . OS_VERSION >/tmp/rel.$$
version_pretty=`sed 's/[" ]//g;/^$/d' </tmp/rel.$$`
version=`sed 's/[" ]//g;/^$/d' </tmp/rel.$$ | tr . _`
subfn="subreleasen.${version}"
if [ -f "$subfn" ]
then
    sub="cat $subfn"
else
    sub=""
fi
echo "`expr $sub + 1`" >$subfn
IMG_BASE=minix${version}_ide_build$sub
BS=4096

HDEMU=0
COPY=0
CVSTAG=HEAD
PACKAGES=1

while getopts "pchu?" c
do
    case "$c" in
        \?)
            echo "Usage: $0 [-p] [-c] [-h] [-r <tag>] [-u]" >&2
            exit 1
            ;;
        h)
            echo " * Making HD image"
            IMG_BASE=minix${version}_bios_build$sub
            HDEMU=1
            ;;
        c)
            echo " * Copying, not CVS"
            COPY=1
            ;;
        p)
            PACKAGES=0
            ;;
        r)
            CVSTAG=$OPTARG
            ;;
        u)
            echo " * Making live USB-stick image"
            IMG_BASE=minix${version}_usb_build$sub
            HDEMU=1
            USB=1
            ;;
    esac
done

if [ "$USB" -ne 0 ]; then
    IMG=${IMG_BASE}.img
else
    IMG=${IMG_BASE}.iso
fi
IMGBZ=${IMG}.bz2
echo "Making $IMGBZ"

USRMB=400

USRBLOCKS=`expr $USRMB \* 1024 \* 1024 / $BS`
USRSECTS=`expr $USRMB \* 1024 \* 2`
ROOTKB=4096
ROOTSECTS=`expr $ROOTKB \* 2`
ROOTBLOCKS=`expr $ROOTKB \* 1024 / $BS`
```

```
if [ "$COPY" -ne 1 ]
then
    echo "Note: this script wants to do cvs operations, so it's necessary"
    echo "to have \$CVSROOT set and cvs login done."
    echo ""
fi

TD1=.td1
TD2=.td2
TD3=.td3

if [ -f $TD1 ]
then
    TMPDISK="cat $TD1"
    echo " * Warning: I'm going to overwrite $TMPDISK!"
else
    echo "Temporary (sub)partition to use to make the /usr FS image? "
    echo "I need $USRMB MB. It will be mkfsed!"
    echo -n "Device: /dev/"
    read dev || exit 1
    TMPDISK=/dev/$dev
fi

if [ -b $TMPDISK ]
then :
else
    echo "$TMPDISK is not a block device.."
    exit 1
fi

echo $TMPDISK >$TD1

if [ -f $TD2 ]
then
    TMPDISK2="cat $TD2"
    echo " * Warning: I'm going to overwrite $TMPDISK2!"
else
    echo "Temporary (sub)partition to use for /tmp? "
    echo "It will be mkfsed!"
    echo -n "Device: /dev/"
    read dev || exit 1
    TMPDISK2=/dev/$dev
fi

if [ -b $TMPDISK2 ]
then :
else
    echo "$TMPDISK2 is not a block device.."
    exit 1
fi

echo $TMPDISK2 >$TD2

if [ -f $TD3 ]
then
    TMPDISK3="cat $TD3"
    echo " * Warning: I'm going to overwrite $TMPDISK3!"
else
    echo "It has to be at least $ROOTKB KB."
    echo ""
    echo "Temporary (sub)partition to use to make the root FS image? "
    echo "It will be mkfsed!"
    echo -n "Device: /dev/"
    read dev || exit 1
    TMPDISK3=/dev/$dev
fi

if [ -b $TMPDISK3 ]
then :
else
    echo "$TMPDISK3 is not a block device.."
    exit 1
fi

echo $TMPDISK3 >$TD3

umount $TMPDISK | true
umount $TMPDISK2 | true
umount $TMPDISK3 | true
```

```

if [ $TMPDISK = $TMPDISK2 -o $TMPDISK = $TMPDISK3 -o $TMPDISK2 = $TMPDISK3 ]
then
    echo "Temporary devices can't be equal."
    exit
fi

echo " * Cleanup old files"
rm -rf $RELEASEDIR $IMG $IMAGE $ROOTIMAGE $IMGBZ $CDFSFILES image*
mkdir -p $CDFSFILES || exit
mkdir -p $RELEASEDIR
mkfs -B $BS -b $ROOTBLOCKS $TMPDISK3 || exit
mkfs $TMPDISK2 || exit
echo " * mounting $TMPDISK3 as $RELEASEDIR"
mount $TMPDISK3 $RELEASEDIR || exit
mkdir -m 755 $RELEASEDIR/usr
mkdir -m 1777 $RELEASEDIR/tmp
mount $TMPDISK2 $RELEASEDIR/tmp

mkfs -B $BS -b $USRBLOCKS $TMPDISK || exit
echo " * Mounting $TMPDISK as $RELEASEDIR/usr"
mount $TMPDISK $RELEASEDIR/usr || exit
mkdir -p $RELEASEDIR/tmp
mkdir -p $RELEASEDIR/usr/tmp
mkdir -p $RELEASEPACKAGE
mkdir -p $RELEASEPACKAGESOURCES

echo " * Transferring $COPYITEMS to $RELEASEDIR"
( cd / && tar cf - $COPYITEMS ) | ( cd $RELEASEDIR && tar xf - ) || exit 1

if [ -d $PACKAGEDIR -a -d $PACKAGESOURCEDIR -a $PACKAGES -ne 0 ]
then
    echo " * Indexing packages"
    bintotal=0
    ( cd $PACKAGEDIR
      for p in *.tar.bz2
      do
          echo $p >&2
          p="`echo $p | sed 's/.tar.bz2//`"
          descr="`./$p.descr`"
          if [ -f "$descr" ]
          then
              echo "$p"cat $descr"
          fi
      done >List
    )
    for d in $PACKAGEDIR $PACKAGESOURCEDIR
    do
        echo Counting size of $d
        f=$d/SizeMB
        if [ ! -f $f ]
        then
            b="`bzip2 -dc $d/*.bz2 | wc -c`"
            echo "`expr 1 + $b / 1024 / 1024`" >$f
        fi
        echo "`cat $f` MB."
    done
    echo " * Transferring $PACKAGEDIR to $RELEASEPACKAGE"
    cp $PACKAGEDIR/* $RELEASEPACKAGE/
    echo " * Transferring $PACKAGESOURCEDIR to $RELEASEPACKAGESOURCES"
    cp $PACKAGESOURCEDIR/* $RELEASEPACKAGESOURCES/ || true
fi

# Make sure compilers and libraries are bin-owned
chown -R bin $RELEASEDIR/usr/lib
chmod -R u+w $RELEASEDIR/usr/lib

if [ "$COPY" -ne 1 ]
then
    echo " * Doing new cvs export"
    ( cd $RELEASEDIR/usr && mkdir src && cvs export -r$CVSTAG src )
else
    ( cd .. && make depend && make clean )
    srcdir=/usr/src
    ( cd $srcdir && tar cf - . ) | ( cd $RELEASEDIR/usr && mkdir src && cd src && tar
    xf - )
fi

```

```

echo " * Fixups for owners and modes of dirs and files"
chown -R bin $RELEASEDIR/usr/src
chmod -R u+w $RELEASEDIR/usr/src
find $RELEASEDIR/usr/src -type d | xargs chmod 755
find $RELEASEDIR/usr/src -type f | xargs chmod 644
find $RELEASEDIR/usr/src -name configure | xargs chmod 755
find $RELEASEDIR/usr/src/commands -name build | xargs chmod 755
# Bug tracking system not for on cd
rm -rf $RELEASEDIR/usr/src/doc/bugs

# Make sure the CD knows it's a CD
date >$RELEASEDIR/CD
echo " * Chroot build"
chroot $RELEASEDIR "/bin/sh -x /usr/src/tools/chrootmake.sh" || exit 1
echo " * Chroot build done"
# The build process leaves some file in src as root.
chown -R bin $RELEASEDIR/usr/src*
cp issue.install $RELEASEDIR/etc/issue

if [ "$USB" -ne 0 ]
then
    usb_root_changes
elif [ "$HDEMU" -ne 0 ]
then
    hdemu_root_changes
fi

echo $version_pretty >$RELEASEDIR/etc/version
echo " * Counting files"
extrakb=`du -s $RELEASEDIR/usr/install | awk '{ print $1 }'`
expr `df $TMPDISK | tail -1 | awk '{ print $4 }'` - $extrakb >$RELEASEDIR/.usrkb
du -s $RELEASEDIR/usr/src.* | awk '{ t+= $1 } END { print t }' >$RELEASEDIR/.extrasrckb
( for d in $RELEASEDIR/usr/src.*; do find $d; done ) | wc -l >$RELEASEDIR/.extrasrcfiles
find $RELEASEDIR/usr | fgrep -v /install/ | wc -l >$RELEASEDIR/.usrfiles
find $RELEASEDIR -xdev | wc -l >$RELEASEDIR/.rootfiles
echo " * Zeroing remainder of temporary areas"
df $TMPDISK
df $TMPDISK3
cp /dev/zero $RELEASEDIR/usr/.x 2>/dev/null || true
rm $RELEASEDIR/usr/.x
cp /dev/zero $RELEASEDIR/.x 2>/dev/null || true
rm $RELEASEDIR/.x

umount $TMPDISK || exit
umount $TMPDISK2 || exit
umount $TMPDISK3 || exit
(cd ../boot && make)
(cd .. && make depend)
make clean
make image || exit 1
mv image image_big
make clean
make image_small || exit 1
dd if=$TMPDISK3 of=$ROOTIMAGE bs=$BS count=$ROOTBLOCKS
# Prepare image and image_small for cdfdboot
mv image_big image
sh mkboot cdfdboot $TMPDISK3
cp $IMAGE $CDFILES/bootflop.img
cp release/cd/* $CDFILES || true
echo "This is Minix version $version_pretty prepared 'date'." >$CDFILES/VERSION.TXT

h_opt=
bootimage=$IMAGE
if [ "$HDEMU" -ne 0 ]; then
    make_hdimage
    h_opt='-h'
    bootimage=hdimage
fi

if [ "$USB" -ne 0 ]; then
    mv $bootimage $IMG
else
    writeisofs -l MINIX -b $bootimage $h_opt $CDFILES $IMG || exit 1

```

```
if [ "$HDEMU" -eq 0 ]
then
    echo "Appending Minix root and usr filesystem"
    # Pad ISO out to cylinder boundary
    isobytes=`stat -size $IMG`
    isosects=`expr $isobytes / 512`
    isopad=`expr $secs - '(' $isosects % $secs ')`
    dd if=/dev/zero count=$isopad >>$IMG
    # number of sectors
    isosects=`expr $isosects + $isopad`
    ( cat $IMG $ROOTIMAGE ;
      dd if=$TMPDISK bs=$BS count=$USRBLOCKS ) >m
    mv m $IMG
    # Make CD partition table
    installboot -m $IMG /usr/mdec/masterboot
    # Make sure there is no hole...! Otherwise the ISO format is
    # unreadable.
    partition -m $IMG 0 81:$isosects 81:$ROOTSECTS 81:$USRSECTS
fi
fi
```

```
#!/bin/sh
#
#      tellconfig - Tell the value of a <minix/config.h> parameter
#                      Author: Kees J. Bot

echo "
#include <minix/config.h>
$*
" >/tmp/tell.$$
exec </tmp/tell.$$
rm /tmp/tell.$$

exec cc -P -E -
```

Table of Contents

1	<i>Makefile</i>	sheets	1 to	2 (2)	pages	1-	2	101 lines
2	<i>chrootmake.sh</i>	sheets	3 to	3 (1)	pages	3-	3	27 lines
3	<i>issue.install</i>	sheets	4 to	4 (1)	pages	4-	4	18 lines
4	<i>mkboot</i>	sheets	5 to	7 (3)	pages	5-	7	197 lines
5	<i>release.sh</i>	sheets	8 to	13 (6)	pages	8-	13	391 lines
6	<i>tell_config</i>	sheets	14 to	14 (1)	pages	14-	14	14 lines