Таблица
файлов
исходников
операционной системы
Minix3.1.5
stable

Часть 2
include

Предисловие к первому изданию
(версия 0.1)

Идея создания обзора исходников операционной системы Minix3 возникла при попытке разобраться в том, почему функция mmap в Minix3 не работает так, как она работает в GNU/Linux. Попытка сделать это просто так (используя grep и редакторы программиста) оказалась безуспешной - исходники такого большого проекта, как Minix3, напоминают своего рода лабиринт. Для того, чтобы вторая попытка была успешнее, было решено делать небольшой анонс каждого «коридора этого лабиринта», т.е. файла исходников. Для того, чтобы частично результаты этой работы можно было использовать и для анализа других системных вызовов Minix3, было решено все анонсы заносить в таблицу. Оформление кратких описаний в виде таблицы позволило бы свободно добавлять такие аннотации в произвольном порядке, а затем сортировать их как по расширенным именам («/kernel/proc.c , вместо proc.c»), так и по индексам. Постепенно бы формировалось описание исходников Minix3.1.5, ориентированных на потенциального разработчика Minix3.

Так как я предположил, что могу быть не одиноким в своих образовательных интересах, то я решил начало таблицы с описанием идеи разместить на форуме сайта www.minix3.ru . Попутно я хотел апробировать описания индексов, в котором я сомневался ибо не имел достаточной квалификации. Увы, но критики самих описаний индексов я не получил (вообще никакой, ни конструктивной, ни деструктивной). Вместо этого я получил предложение развить подобный обзор исходников Minix3.1.5 уже для русской версии Minix3, выход которой был запланирован на 23-е февраля 2010-го года. Это заставила кардинально изменить подход к формированию таблицы:
- требовалось охватить большое количество файлов (желательно – всю систему) в очень сжатые сроки,
- добавление аннотаций в таблицу должно было быть систематическим и планомерным для того, чтобы другой человек мог бы подключиться к заполнению таблицы, а главное – для того, чтобы к назначенному сроку получить полное описание части подсистем, вместо неполного описания всех подсистем,
- пришлось отложить практические задачи, которые были толчком для самой идеи создания обзора (что печально, так как снижает качество аннотаций).

После месяца работы стало возможным определить то количество файлов, которое реально успеть включить в таблицу. Стало ясно, что нереально охватить даже всю собственно систему, однако возможно довольно близко подойти к этому, включив в таблицу микроядро (подсистему /kernel/), серверы (подсистему /servers/), драйверы (подсистему /drivers/), включаемые файлы (подсистему /include/), системный загрузчик (подсистему /boot/), а также вспомогательные файлы (подсистемы /etc/ и /tools/). Из собственно системы пришло временно отказаться от системной и пользовательской библиотеки (подсистема /lib/). Тесты (подсистема /test/) и системные утилиты (подсистема /commands/) были исключены из приоритетного рассмотрения потому, что не являются частями собственно операционной системы minix3.1.5 (/man/ и /docs/ не содержат файлов исходников).

При более подробном рассмотрении оказалось, что обзор включаемых заголовочных файлов требует совсем иного подхода, нежели чем обзор микроядра, серверов и драйверов. Фактически стало понятно, что для подсистемы /include/ целесообразно иметь отдельную таблицу аннотаций, содержащую большее количество индексов для сортировки. Именно тогда (в контексте времени выхода первой русской версии Minix3 – в самый последний момент) и сформировался полный план обзора:

основные:
Часть 1 (/boot/, /kernel/, /servers/, /drivers/, /etc/, /tools/);

Часть 2 (/include/);
Часть 3 (/lib/);
дополнительные (возможно даже, что их обзор нецелесообразен):
Часть 4 (/commands/);
Часть 5 (/test/).

Вместе с первой русской версией Minix3 выйдут первые две части обзора исходников операционной системы Minix3.1.5 (на которой и будет основана русская версия). Вся эта «история создания» для того, чтобы потенциальный читатель подошёл к версии 0.1 обзора довольно критично, однако с пониманием:
- у меня не было времени согласовывать с сообществом переводы терминов, поэтому очень часто они будут, мягко говоря, довольно спорными, НО при этом я всегда старался в скобках привести исходный английский вариант;
- так как «лобовая» попытка уточнить термины в интернете наткнулась на серьёзные трудности, то я привёл список всех моих терминологических затруднений в «прелюдии» к таблице (хоть это и тема для отдельной статьи, а лучше, если бы в рамках русской wiki был создан соответствующий словарь);
- индексы также имеют пока только ориентировочное значение (их соответствие заявленным описанием требует отдельной проверки, на которую у меня не было времени), следует их рассматривать как попытку разделить все файлы исходников на несколько неравных частей с надеждой, что изучение более малой части существенно облегчит понимание большей;
- «прелюдии» к таблицам не содержат никаких рекомендаций по практическому использованию (на практическую апробацию не было времени);
- текст не выдержан в соответствии с правилами оформления, объявленными для документации русской версии Minix3.

Исправление всех этих недостатков и будет основной целью версии 0.2. Единственной отрадной мыслью является то, что обзора исходников Minix3.1.5 по всей видимости нет вообще – даже на английском языке. А изменения по сравнению с Minix3.1.1 уже довольно значительные: эти и виртуальная память, это и виртуальная файловая система, это и новый сервер ipc. Как говориться: «На безрыбье и рак – рыба!»

В данном обзоре *нет непосредственных цитат (*за одним исключением: цитируется ***«Траектория программного вызова» (Цилюрик О.И).****)*, поэтому в версии 0.1 *отсутствует* и *список используемой литературы*.В целом хочу сказать огромное спасибо русскому Minix-сообществу и особенно его фактическому руководителю - Роману Игнатову!

Часть
2
Таблица описаний заголовочных файлов

Введение

Таблица данной части обзора исходников операционной системы Minix3.1.5 создавалась в довольно сжатые сроки. Это не могло не сказаться отрицательно на обзоре, однако все сомнительные части как в плане перевода, так и в плане аннотации отмечены знаком: (?)

В таблице шрифтом Arial помечены материал моего помощника (*VS - Valery Solovey*). Огромное ему спасибо!

Включаемые файлы представляют собой самую «стандартную» часть проекта операционной системы. Именно поэтому подавляющее большинство файлов имеют индексы «1» - если это файл, предусмотренный «внешним» стандартом (ANSI C, POSIX, ...), «2» - для Minix – специфических файлов.

Следует заметить, что подсистема /include/ является своего рода фасадом подсистемы /lib/. Так как в первую русскую версию описание подсистемы /lib/ не войдёт, то в данном обзоре сделана попытка это компенсировать – большинство анонсов заголовочных файлов содержат список всех названий прототипов функций со ссылками на файлы их реализации.

Таблица предварена «прелюдией», состоящей из 3-х разделов.

В первом разделе изложены основные идеи создания обзора файлов исходников Minix3.1.5, в контексте описаний именно данной части обзора – заголовочных файлов(/include/). Кроме того дана очень краткая аннотация файлов описываемых подсистем в целом.

Во втором разделе даны пояснения значениям столбцов таблицы (за исключением самого столбца описаний – предполагается, что его значение понятно и так; а также столбца расширенного имени файла, который дан в виде абсолютного пути, хотя правильнее было бы убрать начальный «/», превратив абсолютный путь относительный с «префиксом» в файловой иерархии Minix3.1.5 «/usr/src/»).

В третьем разделе очень формально приведен список терминов, перевод которых у меня вызвал затруднения. Предполагается, что это поможет как в дальнейшем улучшении обзора, так и лучшему пониманию текста анонсов файлов исходников. *Нумерация списка продолжает соответствующий список 1-й части.*

1. Вводные замечания по поводу особенностей данной подсистемы

1.1. *Основная идея таблицы анонсов и подсистема /include/*

Главным преимуществом проектов с открытыми исходниками является возможность переиспользования исходного текста как для других проектов, так и путём видоизменения и адаптации данного проекта к нуждам конкретной задачи. Переиспользование кода конкретного проекта подразумевает под собой анализ этого проекта другим программистом или группой программистов (т.е. людьми, не принимавшими участия в создании исходного проекта).

Для проекта операционной системы наибольшее значение имеет как раз задача видоизменения и адаптации к нуждам конкретной задачи: это и портирование на новую архитектуру, это и добавление новых системных вызовов, это и изменение свойств самой системы с сохранением программного интерфейса (например, замена алгоритмов планирования , чтобы обеспечить нужды задач реального времени).

При видоизменении проекта главным вопросом становится: «Как добиться желаемого результата, потратив минимальное время на изучение проекта?» Одним подходом к решению данной проблемы является использование эффективных методов анализа программных проектов (в.т.ч. с применением программных средств, таких как grep, cflow, doxygen, OpenGrok). Другим подходом к решению этой проблемы является стиль и дисциплина программирования самих разработчиков исходного проекта (в.т.ч. с учётом, например, возможностей doxyen). Третьим подходом является написание документации проекта, ориентированной именно на его переиспользование (т.е. документации не для пользователя, а для разработчика).

Создание индексированной таблицы анонсов файлов исходников является одним из видов подобной документации, ориентированной на переиспользование проекта. (Более подробно об идее таблицы анонсов см. п. 1. первой части обзора)

Особенностью подсистемы /include/ является то, что она сама по себе не содержит самостоятельного программного кода (являясь своего рода «фасадом» для LIBC – подсистемы /lib/). Подсистема /include/ представляет собой программный интерфейс для разработчика прежде всего пользовательских программ. Особенностью микроядерной мультисерверной архитектуры Minix3.1.5 является то, что подсистема /include/ является и программным интерфесом системного разработчика, например, разработчика драйверов устройств. (В GNU/Linux для этих целей используется другой пакет – ***kernel headers***.) Поэтому, кроме традиционного значения внешнего интерфейса, подсистема /include/ также играет роль «координатора», объединяющего все процессы (микроядро, серверы, драйверы) в единое согласованное целое.

Другой особенностью подсистемы /include/ является её «стандартный характер» - именно здесь в наибольшей степени отражены как внутренние, так и внешние стандарты. Это также делает эту подсистему Minix3.1.5 наиболее знакомой для стороннего разработчика – знакомой по аналогии с другими операционными системами. «Стандартный характер» подсистемы /include/ также определяет индексы «1» или «2» для большинства файлов, означающие нежелательность их изменения при видоизменении или адаптации операционной системы minix3.1.5.

Таким образом, с точки зрения задачи видоизменения и адаптации опрерационой системы Minix3.1.5, следует отметить особенности подсистемы /include/:
(1) большинство файлов отражают стандарты, но

- не все стандартные файлы имеются в Minix3.1.5 (потенциальная цель видоизменения),
- стандарты могут меняться, да и может меняться ориентация операционной системы Minix3.1.5 на список конкретных стандартов;
(2) имеются файлы, объединяющие все подсистемы в единое целое, координирующие их работу, поэтому любое существенное изменение в операционной системе (например, новый системный вызов, изменение формата сообщений, новая поддерживаемая архитектура) непременно должно отражаться в этих файлах (выделить их - одна из основных задач этой части обзора);
(3) большинство файлов являются ссылками на подсистему /lib/, поэтому их анонсы должны помогать при видоизменении других подсистем облегчая поиск того конкретного кода, который стоит за библиотечной функцией.

1.2. Дополнительные индексные столбцы

С точки зрения индексов, содержимое /include/ также существенно отличается от всего остального. Если кратко, то даже при поверхностном рассмотрении возникает три дополнительных индекса, применимых только для include:
- связь конкретного заголовочного файла с конкретной библиотекой, а также связь конкретного заголовочного файла с конкретным файлом, в котором размещён код, реализующий функции, объявленные в данном заголовочном файле, (последнее является детализацией первого, поэтому это можно сделать отдельным - единым индексом);
- связь конкретного заголовочного файла с конкретным стандартом (сразу приходят на ум индексы: C-STD, UNIX, Minix3_SPEC);
- место в системной иерархии (clibc, ulibc, slibc (0, 1, 2, ...), klibc).

Последнее требует дополнительного пояснения...

Если сравнивать Minix3 и Linux, то следует сразу заметить
- ядро Linux вместо стандартных заголовочных файлов (и стандартной библиотеки) использует свои - внутренние заголовочные файлы (связанные с соответствующими подсистемами ядра Linux),
- функциональность, которая в Linux обеспечивается ядром, в Minix3 обеспечивается совокупностью (микроядро+серверы+драйверы), но при этом серверы и драйверы значительно ближе к пользовательским процессам, нежели чем к микроядру,
- в Minix3 даже микроядро ссылается на некоторые заголовочные файлы из общего каталога /includes/ , при этом требуется установка специфических констант препроцессора (фрагмент кода из /kernel/kernel.h) :

Код:

```
#define _POSIX_SOURCE      1    /* tell headers to include POSIX stuff */
#define _MINIX             1    /* tell headers to include MINIX stuff */
#define _SYSTEM            1    /* tell headers that this is the kernel */
```

Исходя из зравого смысла (и учитывая написанное в соседней ветке "Траектория программного вызова") можно все функции (объявляемые в заголовочных файлах /include/) разделить по группам:

clibc : общие функции, которые в принципе не выполняют никаких привилегированных операций и системных вызовов, их функционирование не зависит от режима процессора,

klibc : некоторые часто используемые функции, которым место исключительно в микроядре

(у меня есть такая гипотеза: можно установить требование - для помещения функции в klibc необходима архитектурная зависимость реализации (иначе непонятно, почему её нельзя разместить в clibc));

slibc : иерархическая система функций на которые могут ссылаться в.т.ч. серверы и драйверы (обычные процессы также могут на них ссылаться, но система скорее всего откажет в их выполнении ввиду недостатка полномочий, однако это не факт);

ulibc : функции, на которые смело может ссылаться любой непривилегированный процесс, но на которые ни в коем случае не должен ссылаться привилегированный процесс.

Замечу сразу - это довольно нетривиальная задача - определить место данной функции в системной иерархии для такой двольно развитой операционной системы, как Minix3.
Также следует заметить, что ***эта проблема возникает только в связи с микроядерной мультисерверной архитектурой*** Minix3.

Таким образом возникают дополнительные индексные столбцы:
L – для индекса связи с библиотекой,
STD – для индекса связи со стандартами,
SI – для индекса места в системной иерархии.

1.3. Подробнее об индексе L.

Разумеется, что если иметь в виду связь конкретного заголовочного файла (а не функции!) с исходниками системной библиотеки libc (в дальнейшем просто LIBC), то скорее всего в заголовочном файле, который во многом представляет собой и некоторые стандарты (а не особенности реализации), будут содержаться прототипы функций, реализация которых расположена в разных файлах исходников. Скорее всего правильно было бы ввести индекс связи с заголовочным файлов для конкретного файла исходников LIBC. Тем не менее было бы разумным предположить, что все функции, прототипы которых расположены в одном заголовочном файле, расположены также и в одной бинарной библиотеке. Бинарные библиотеки расположены в папке /usr/lib , а точнее – в /usr/lib/i386 (хотя каталог /usr/lib/i86 также не является пустым и также будет в поле зрения данного обзора).

Содержимое каталога /usr/lib/i86:

as	cg	**end.a**	**libc.a**	libd.a	libe.a
libfp.a	**libsys.a**				

Содержимое каталога /usr/lib/i386:

as	cg	crtso.o	**end.a**	libbas.a	**libc.a**
libcurses.a	libd.a	libe.a	**libedit.a**	libfl.a	libfp.a
libm.a	libm2.a	libmq.a	libocm.a	libp.a	**libsys.a**
libsysutil.a	**libtimers.a**	**libutil.a**	liby.a	libz.a	m2rtso.o
prtso.o					

[*Примечание:* жирным шрифтом отмечены статические библиотеки, упомянутые в таблице «*Таблица «реальных» индексов группы L*» (см. далее)]

При этом следует также сразу заметить, что существуют также
- заголовочные файлы, не содержащие прототипов функций (только определения типов, константы - определения препроцессора, макросы препроцессора, и.т.п.);
- заголовочные файлы фактически являющиеся ссылками на другие заголовочные файлы

(причём один заголовочный файл может включать в себя даже несколько других заголовочных файлов).

Выставление данного индекса на первый взгляд выглядит довольно нетривиальной задачей, особенно если пытаться «ухитриться» сделать это без анализа исходников LIBC (т.е. подсистемы /lib/).

Частично анализ LIBC начат в работе ***«Траектория программного вызова» (Цилюрик О.И).***

Позволю себе некоторые цитаты оттуда:

(1):

Все действия в
программе выполняют программные вызовы из стандартной библиотеки C
`/usr/lib/i386/libc.a :`

```
# ls /usr/lib/i386
as end.a libcurses.a libedit.a libm.a libocm.a libsysutil.a liby.a
prtso.o
cg libbas.a libd.a libfl.a libm2.a libp.a libtimers.a libz.a
crtso.o libc.a libe.a libfp.a libmq.a libsys.a libutil.a m2rtso.o
```

Примечание: обратим внимание, что

– здесь же находятся стандартные библиотеки периода исполнения для MODULA-2 (`libm2.a`) и Pascal (`libp.a`);

– и стартовые преамбулы, для C программ это `crtso.o` который обеспечивает выполнение что-то вроде следующего:

```
push ecx ! push envp
push edx ! push argv
push eax ! push argc
. . .
call _main ! main(argc, argv, envp)
push eax ! push exit status
call _exit
```

– который, в принципе, тоже может быть подменён (исходный код находится в `/usr/src/lib/i386/rts/crtso.s`), например, чтобы выполнить некоторые действия прежде вызова `main()`.

Исходные коды реализации всех программных вызовов находятся в подкаталогах каталога `/usr/src/lib`.

Простейшие программные вызовы стандартной библиотеки C начнём рассматривать с `/usr/src/lib/ansi`:

```
# ls /usr/src/lib/ansi
atof.c exit.c islower.c malloc.c qsort.c strcpy.c strstr.c
atoi.c ext
```

(2):

```
# cat /usr/src/lib/ansi/strlen.c
size_t strlen(const char *org)
{
register const char *s = org;
while (*s++)
/* EMPTY */ ;
```

```
return --s - org;
}
```
Здесь всё предельно ясно: программа выполняет вызов функции, которая полностью обслуживается объектным
кодом из библиотеки `libc.a`, как это показано на рисунке 1.

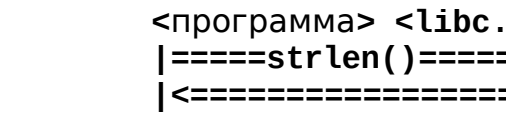

Рисунок 1. Обслуживание программного вызова библиотекой `/usr/lib/i386/libc.a` .

Таких (обслуживаемых по такой схеме, без привлечения служб операционной системы) программных вызовов (точек входа `libc.a`) — большинство. Но даже в подкаталоге `/usr/src/lib/ansi` (библиотека ANSI) есть
неожиданности:
...

Также необходимо заметить, что в числе заголовочных файлов встречаются файлы, не содержащие прототипов функций(и глобальных переменных, но наличие глобальных переменных в LIBC – не нонсенс ли?), и поэтому не отображающийся непосредственно в объектные файлы, а значит и файлы библиотек. Кроме того есть подозрение, что среди заголовочных файлов LIBC попадаются и заголовочные файлы, не относящиеся к LIBC. Отсюда получаем дополнительные индексы:
N – для заголовочных файлов без (в.т.ч.) косвенных прототипов;
EX – для заголовочных файлов, не относящихся к LIBC.

Для нахождения индексов заголовочных файлов группы L необходимо:
- определить нахождение кода всех функций, прототипы которых явно или неявно включены в данный заголовочный файл (прослеживается в описании соответствующего заголовочного файла, частично компенсирует отсутствие анализа подсистемы /lib/);
- определить индекс (возможно комплексный) исходя из каталогов подсистемы /lib/, в которых расположены файлы, содержащие код функций, прототипы которых явно или неявно содержатся в данном заголовочном файле (для этого используется таблица: «*Таблица «реальных» индексов группы L*» (см. далее)).

Список символов, используемых в конкретной библиотеке можно получить различным способом. Один из способов – это анализ соответствующих файлов Makefile , Makefile.in, находящихся в каждом из подкатилоге подсистемы /lib/ . Именно таким образом «*Таблица «реальных» индексов группы L*» (см. далее) и составлена.

Таблица «реальных» индексов группы L

Каталог подсистемы /lib/	Библиотека	Индекс L
ansi/	libc.a	C
curses/	libcurses.a	A
dummy/	libm.a	M
editline/	libedit.a	E
end/	end.a	B
float/	?	?
fphook/	?	?
gnu/	-	-
i86/	-	-
ack/	-	-
i386/	libc.a	C
ip/	libc.a	C
math/	libc.a	C
obj-ack/	-	-
other/	libc.a	C
posix/	libc.a	C
regex/	libc.a	C
stdio/	libc.a	C
stdtime/	libc.a	C
syscall/	libc.a	C
syslib/	libsys.a	S
sysutil/	libsys.a	S
sysvipc/	libc.a	C
timers/	libtimers.a	T
util/	libutil.a	U

Порядок индексов в комбинациях: **STUEABMC** .

Дополнительные индексы:
N – для заголовочных файлов без (в.т.ч.) косвенных прототипов;
EX – для заголовочных файлов, не относящихся к LIBC;
исключают возможность комбинирования с другими индексами и между собой.

1.4. Подробнее об индексе STD.

Здесь всё **на первый взгляд предельно просто – есть** , грубо говоря, **3 варианта**:

C – самый общий стандарт языка программирования C, предлагаемый всеми компиляторами, независимо от операционной системы;
U – функции, специфические для UNIX (т.е. которые есть во всех реализациях C для UNIX-подобных операционных систем, однако в остальных реализациях C их может и не быть; обычно эти функции, константы, типы оговорены в том или ином стандарте POSIX);
M – специфические для Minix функции, которые в реализациях C для других операционных систем отсутствуют, либо имеют другое значение.

Конечно можно предположить, что некоторый заголовочный файл будет содержать элементы из разных групп, но тогда также легко выстраивается порядок расположения букв комплексного индекса: M , U , C. При этом следует всё же заметить, что определения групп C, U, M планируется усовершенствовать так, чтобы на отдельных элементах заголовочных файлов они не пересекались (т.е. комплексный индекс заголовочный файл может иметь потому, что содержит несколько элементов - разнотипных элементов).

Проблема здесь в другом – в границе между C и U. Во-первых, все POSIX стандарты содержат не U, а объединение C и U; во-вторых, C сам по себе развивался в контексте UNIX, и оказал влияние (и продолжает его оказывать) на другие операционные системы; в-третьих, количество операционных систем настолько велико, что охватить их все на предмет наличия универсальных общих моментов в LIBC просто не представляется возможным.

Временно эта проблема будет решаться несколько упрощённо – то, что входит в стандарт ANSI – будет иметь индекс C, а то, что не входит в стандарт ANSI, но входит в стандарт POSIX - будет иметь индекс U.

Ещё один момент также нарушает эту схему – архитектурно зависимая часть, а также части зависимые от устройств. Имеется также множество заголовочных файлов, не содержащих прототипов функций (только константы и макросы). Всё это заставляет ввести ещё хоть один индекс. Пока (в версии 0.1) будет даже три индекса:
A – архитекрурно зависимый заголовочный файл, связанный с особенностями IBM PC совместимых персональных компьютеров;
D – заголовочный файл, связанный с конкретным устройством (вообще-то подобным заголовочным файлам не совсем место в заголовочных файлах LIBC, однако при микроядерной многосерверной архитектуре операционной системы это в принципе возможно);
N – заголовочный файл, не содержащий прототипов функций, и который затруднительно отнести к какой либо конкретной группе (такой запасной вариант).

1.5. Подробнее об индексе SI.

Здесь также довольно просто, ибо есть сразу исходящие из описания основные индексы
C – общие функции без системных вызовов,
U – функции с системными вызовами для всех непривилегированных процессов (подразумевается опора на полный набор работающих серверов и драйверов Minix3, но без трюков, хакерства и привелегий);

S – функции, содержащие системные вызовы, предназначенные для вызова исключительно серверами и/или драйверами (т.е. с одной стороны подразумевается отсутствие некоторой части функционала операционной системы, с другой стороны – наличие привилегий, позволяющих выполнить некоторые операции, недоступные для непривилегированных процессов);

Данные индексы могут быть комбинированы, но располагаются в порядке S , U , C.

Кроме того подразумевается также наличие некоторых дополнительных индексов:
N – для заголовочных файлов, не содержащих прототипов функций (и, разумеется, самих определений функций) ни непосредственно, ни косвенным образом (заголовочный файл также может включать в себя другие заголовочные файлы, которые могут таким образом косвенно подключать прототипы функций). Этот индекс не может комбинироваться ни с одним из вышеназванных ибо установка основного индекса подразумевает под собой наличие хоть одного прототипа функции (непосредственно или косвенно);
K – для заголовочных файлов, прямо включаемых подсистемой микроядра Minix3 (файлами внутри /kernel/); предполагается, что данный индекс будет комбинироваться с другими и тогда всегда должен быть на второй позиции (например SKUC, NK и.т.п.); в одиночке данный индекс может быть включён только если какой-то (гипотетически такой случай можно представить) заголовочный файл исключительно подключается только из исходников микроядра и не может быть рекомендован для подключения в пользовательских проектах (т.е. ему нельзя поставить (например) индекс C); следует также заметить, что индекс K является единственным индексом, который не наследуется посредством косвенного включения (т.е. пусть имеем файл **a.h** , который сам по себе не подключается ни непосредственно, ни косвенно ни к одному из исходников микроядра, но который начинается
#include <b.h>
#include <c.h>
#include <d.h>

и пусть теперь:
<b.h> имеет индекс **CK**,
<c.h> имеет индекс **UK**,
<d.h> имеет индекс **SK**, то
сам файл <a.h> **будет иметь индекс SUC**, а не SKUC).

Здесь также основная проблема – это найти грань между U *и* S. *Пока этот вопрос будет решаться несколько упрощённо – если функция содержит системный вызов (вызов микроядра) и находится в libc.a, то она будет иметь индекс* U*; если функция содержит системный вызов (точнее - вызов микроядра), но не находится в libc.a, то она будет иметь индекс* S.

2. Пояснения к таблице

2.1. *Смысл значений колонны* «pr.»:

1 – этот файл не стоит видоизменять, разве что в исключительном случае; (*файл содержит в себе информацию, связанную со стандартом UNIX, ANSI, соглашениями, принятыми в сообществе разработчиков Minix3; поэтому и цель изменить этот файл может ставиться исключительно только после согласования с сообществом разработчиков Minix3*)

2 – этот файл лучше не видоизменять, а если и видоизменять, то очень и очень осторожно; (*хоть файл и не содержит ничего, непосредственно связанного со стандартами и соглашениями, тем не менее изменения в API и/или структурах данных приведут к значительному количеству изменений в других местах, которые скорее всего следует оценить отрицательно*)

3 – видоизменять имеет смысл только если это непосредственно оправдано (т.е. ставится такая цель в формулировке: «перепишем функционал и структуры данного файла»); (*не стоит сюда добавлять что-то не связанное с уже имеющейся информацией – это будет противоречить принципу модульности; для нового элемента подсистемы лучше создать новый файл;*)

4 - видоизменять имеет смысл только если это непосредственно оправдано, но в файле используются функции, описания которых не ограничены стандартами и/или соглашениями, а также не имеют глобального значения (см. pr. 2); изменения прототипа любой из этих функций потребует внесения изменений и в этот файл; (*не стоит сюда добавлять что-то не связанное с уже имеющейся информацией ; однако может возникнуть необходимость внести изменения, связанные с изменением прототипов используемых функций; этот вид индекса наиболее характерен для «интегрирующих» файлов – содержащих, например, main(), init(), ... подсистемы;*)

[Примечание (1): *Если мы собираемся добавить новый элемент в подсистему, то мы автоматически ставим непосредственную цель изменить интегрирующий файл – иначе просто этот новый элемент так и останется неиспользованным!*]

5 – файл скорее всего также придётся менять, если вносятся изменения в подсистему (обычно это соседние файлы соответствующей папки исходников);(*данный файл содержит глобальные данные для подсистемы; если включение нового элемента также подразумевает свои глобальные данные, то их скорее всего надо размещать в этом файле;этот вид индекса также характерен для «интегрирующих» файлов – содержащих, например, main(), init(), ... подсистемы, однако только в том случае, если в его начале находятся ещё и глобальные данные;*)

6 – файл содержит данные используемые для связи данной подсистемы (какой – указано в аннотации) с остальными подсистемами; (*однако изменения в этот файл вносятся только при изменении данной подсистемы, но они могут повлиять и на другие подсистемы – в случае изменения этого файла скорее всего потребуется что-то поменять и в других подсистемах*)

7 – файл содержит общие данные хотя бы для двух разных подсистем; информация данного файла не может быть непосредственно связана с какой-то одной подсистемой; эти данные обычно не требуется менять при добавлении системных вызовов, новых сообщений,..., разве, что это непосредственно связано с данными подсистемами; (*если ставиться задача*

добавить системный вызов, новое сообщение, ... , то всё же не возникает автоматически вопрос о необходимых изменениях в данном файле; изменения в файле всё же инициированы внутренними изменениями в подсистемах, а не внешними изменениями в работе системы;)

8 – файл часто требуется менять при добавлении системных вызовов, новых сообщений, других существенных изменениях в работе OS Minix3, причём вне зависимости от изменений в подсистеме - соседних файлах в папке; (*если ставиться задача добавить системный вызов, новое сообщение, ... , то автоматически возникает вопрос о необходимых изменениях в данном файле;*)

9 – файл придётся менять почти всегда (за редким исключением);

2.2. *Смысл значений колонны* «aut.»:

FW – Борманис Виктор Янович (FireWall)

VS - Valery Solovey

2.3. *Смысл значений колонны* «L»

При создании однобуквенных индексов в основном использовалась первая буква статической библиотеки, следующая за приставкой lib. Однако такой подход не для всех случаев пригоден, поэтому были сделаны исключения для индексов:

A - libcurses.a
B - end.a

Остальные индексы:

C - libc.a
M - libm.a
E - libedit.a
S - libsys.a
T - libtimers.a
U - libutil.a

Порядок индексов в комбинациях: STUEABMC .

Дополнительные индексы:

N – для заголовочных файлов без (в.т.ч.) косвенных прототипов;
EX – для заголовочных файлов, не относящихся к LIBC;

исключают возможность комбинирования с другими индексами и между собой.

2.4. *Смысл значений колонны* «STD»

A – архитекрурно зависимый заголовочный файл, связанный с особенностями IBM PC совместимых персональных компьютеров;
C – *ANSI* стандарт языка программирования C;
D – заголовочный файл, связанный с конкретным устройством;
M – специфические для Minix функции;
N – заголовочный файл, не содержащий прототипов функций, который нельзя отнести к стандартам ANSI, POSIX, Minix (так как связан с прочими стандартами, например,

стандартами, принятыми в компьютерных сетях);
U – функции, специфические для UNIX (POSIX, FreeBSD, OGBSI6, ...);

Порядок индексов в комплексных индексах: D, A, M , U , C. Индекс N исключает все остальные и не может быть частью комплексного индекса.

2.5. *Смысл значений колонны* «SI»

C – общие функции без системных вызовов;
U – функции содержат системный вызов (вызов микроядра) и находятся в libc.a;
S – функции содержат системный вызов (вызов микроядра), но не находятся в libc.a.

Данные индексы могут быть комбинированы и располагаются в порядке: S , U , C.

N – для заголовочных файлов, не содержащих прототипов функций ни непосредственно, ни косвенным образом (не может комбинироваться ни с одним из вышеназванных);
K – для заголовочных файлов, прямо включаемых подсистемой микроядра Minix3 (файлами внутри /kernel/).

Индекс «K» почти всегда комбинируется и расположен на второй позиции.

3. *Список терминов и фраз, перевод которых может быть спорным*

(58) - **тип заголовка 01,** (PCI-to-PCI bridge devices (?))
[следующие регистры являются общими с типом 00:
/include/ibm/pci.h

(59) Операции (макросы) для манипулирования битами простых флаговых переменных ((?) simple mask variable).
/include/minix/bitmap.h

(60) Точка во вставке 'размере поколения' ((?) generation size) нужна для некоторых фиктивных конечных точек ((?) endpoint), таких как NONE, ANY, ...
/include/minix/endpoint.h

(61) Этот включаемый файл определяет некоторые магические числа ((?) magic process numbers) процессов, таких как ANY и NONE, и не должен быть допустимой конечной точкой ((?)valid endpoint number).
/include/minix/endpoint.h

/page 116/

(62) определения, обеспечивающие (более удобные) имена для наиболее часто используемых членов структуры (message;): m1_i1, ... m9_c2
система выполнения ((?)run-time system) Minix (IPC), (asynmsg_t;)
/include/minix/ipc.h

(63) Определения для работы с раскладами клавиатуры (т.е. в частности, таблицами символов) ((?) defines for keymapping).
/include/minix/keymap.h

(64) Параметры запрашивающей программы ((?) query program parameters).
/include/minix/queryparam.h

(65) На данный момент только значение возникшее в первом vty (conosole) – системной консоли ((?) Currently only the value of the origin of the first vty (console),) ,
/include/minix/tty.h

(66) Это сообщение записывается в конец видео памяти (база видео памяти + размер видео памяти - sizeof(struct boot_tty_info)). ((?) (video memory base + video memory size - sizeof(struct boot_tty_info)) .
/include/minix/tty.h

(67) Макросы с именами, написанными строчными буквами гарантируют об исполнении ((?) to evaluate their argument exactly once) аргументов ровно один раз. Функции макросов закодированы в их именах ((?) The function of the macros is encoded in their names); htons означает преобразовать (беззнаковый) short в порядке байтов хоста (компьютера) в сетевой порядок байтов.
/include/net/hton.h

(68) а также несколько констант препроцессора, в.т.ч. для полей контрольной информации VLAN и о метках приоритетов ((?) Priority tagging)
/include/net/gen/ether.h

(69) Конфигурационный файл системы разрешения имён и адресов ((?) Resolver configuration file. : маршрутизатора)
/include/net/gen/resolv.h

(70) (?)Definitions for the Routing Information Protocol (RFC-1058).
/include/net/gen/rip.h

(71) Это просто «фальшивая» ((?) fake) библиотека асинхронного ввода/вывода, которая использовалась для программ, написанных для Minix-vmd,
/include/sys/asynchio.h

(72) Системный вызов sigreturn() редко вызывается пользовательскими программами, но он используется внутри (ОС Minix3 – привилегированными процессами (?)) механизмом перехвата сигналов ((?)signal catching mechanism.)
/include/sys/sigcontext.h

(73) Эти вызовы являются единственным общепринятым ((?) approved way to inspect i-nodes) способом для контроля i-узлов ((?) i-node, inode).
/include/sys/stat.h

pr.	*L*	*STD*	*SI*	path	*an.*	*aut*
2	C	M	UK	**/include/a.out.h**	Похоже, что этот файл - нечто стандартное для UNIX. Но между различными версиями ОС он может конфликтовать, поэтому для MINIX 3 он ссылается на <minix/a.out.h>, а для других ОС выдаёт ошибку. **[include:]** ansi.h Является специфическим для Minix но фактически является ссылкой на <minix/a.out.h>. В свою очередь, <minix/a.out.h> описывает структуры и константы, связанные с форматом исполняемого файла minix. Описывает структуры и константы, связанные с форматом исполняемого файла minix.Определяется только одна функция: _PROTOTYPE(int nlist, (char *_file, struct nlist *_nl)); код которой находится в файле: /lib/other/nlist.c	VS FW
1	C	C	C	**/include/alloca.h**	Функция выделения памяти? (для gcc и ack) **[man:]** alloca(3) Содержит прототип функции: _PROTOTYPE(void *alloca, (size_t _size)); ассемблерный код которой находится в файле: /lib/i386/misc/alloca.s Эта функция осуществляет динамическое выделение памяти в стеке.	VS FW
1	N	C	N	**/include/ansi.h**	Макросы, задающие совместимость с ANSI или K&R версией языка	VS
2	SC	C	SU	**/include/assert.h**	Макрос, позволяющий добавлять в программу утверждения о её состоянии. Условие – проверка корректности работы программы, результат - аварийный останов. **[man:]** assert(3) **[include:]** ansi.h Содержит прототип функции _PROTOTYPE(void *__bad_assertion*, (const char *_mess)) код которой находится в файлах:	VS

					/lib/syslib/assert.c **/lib/ansi/assert.c**	
2	C	M	U	**/include/configfile.h**	Структура и функции для работы с конфигурационными файлами в универсальном формате. **[man:]** configfile(3) Содержит прототипы функций: config_t *_config_read_(const char *_file, int flags, config_t *_cfg); void _config_delete(_config_t *_cfg); int _config_renewed_(config_t *_cfg); size_t _config_length_(config_t *_cfg); код которых находится в файле: /lib/other/configfile.c	VS
1	C	C	CK	**/include/ctype.h**	character type? Функциии и макросы для работы с символами (диапазоны, классы: цифры, буквы большие, буквы маленькие). Завязан на chartab.c. **[man:]** ctype(3) **[include:]** ansi.h Содержит символы (прототипы функций): isalnum, isalpha, iscntrl, ... определённые в каталоге: /lib/ansi/ в файлах, соответственно: isalnum.c, isalpha.c, iscntrl.c, ... Довольно «остроумная» реализация, даже не верится, что так можно !.. chartab.c – это просто таблица индексов по однобайтовому коду символа (при смене локали или кодировки вполне может потребоваться поменять этот файл).	VS FW
2	A	M	U	**/include/curses.h**	Управление окнами. Графика? Псевдографика? **[man:]** curses(3) **[include:]** termios.h, stdarg.h, stdio.h Управление «символьными» окнами. Содержит символы (прототипы функций): unctrl, baudrate, beep, cbreak, clearok, clrscr, curs_set, delwin, doupdate,echo, endwin, erasechar, fatal, fixterm, flash, gettmode, idlok, initscr, keypad,	VS FW

					killchar, leaveok, longname, meta, mvcur, mvinch, mvprintw, mvscanw, mvwin, ... определённые в каталоге: /lib/curses/ в файлах, соответственно: unctrl.c, termmisc.c, beep.c, setterm.c, options.c, cursesio.c, curs_set.c, windel.c, update.c, setterm.c, endwin.c, termmisc.c, cursesio.c, termmisc.c, flash.c, cursesio.c, options.c, initscr.c, options.c, termmisc.c, options.c, longname.c, options.c, mvcursor.c, charpick.c, prntscan.c, prntscan.c, winmove.c, ... ***Данный заголовок собственно в системе не используется, а его функции включены не в libc.a, а в libcurses.a !***	
1	C	MU	U	**/include/dirent.h**	Нечто стандартное для UNIX. Перенаправляет на <minix/dirent.h> или выдаёт ошибку. Определения для процедур чтения файловых каталогов. Начало: #ifndef _TYPES_H #include <sys/types.h> #endif #include <sys/dir.h> Содержит символы (прототипы фнукий): closedir, opendir, readdir, rewinddir, seekdir, getdents определённые в каталоге: /lib/syscall/ в файлах, соответственно: closedir.s, opendir.s, readdir.s, rewinddir.s, seekdir.s, getdents.s telldir :: /lib/other/telldir.c	VS FW
2	S	MU	UC	**/include/env.h**	Работа с переменными окружения. Символы: env_parse :: /lib/sysutil/env_parse.c env_panic :: /lib/sysutil/env_panic.c env_prefix :: /lib/sysutil/env_prefix.c env_memory_parse :: /lib/sysutil/env_parse.c	VS FW
2	N	MU C	NK	**/include/errno.h**	Список ошибок (общие, сетевые, от ядра)	VS

					[man:] intro(2) **Не содержит прототипов функций!** Интересно то, что коды ошибок пользовательских программ и внутри MINIX различаются знаком, что обеспечивается определением препроцессора: #ifdef _SYSTEM # define _SIGN - # define OK 0 #else # define _SIGN #endif	FW
1	C	U	U	**/include/fcntl.h**	Флаги и функции для создания, открытия и получения общей информации о данных файла. fcntl, open, creat, flock. **[man:]** fcntl(2) **[include:]** minix/types.h Содержит символы (прототипы функций): creat, fcntl, open, определённые в каталоге: **/ib/posix/** в файлах, соответственно: ***_creat.c, _fcntl.c, _open.c,*** Функция *flock* определена в файле ***/lib/other/flock.c***	VS FW
2	N	C	N	**/include/float.h**	Параметры чисел с плавающей точкой.	VS
1	C	MU	U	**/include/fts.h**	FreeBSD. Структуры и функции для работы с деревом файлов. (file tree structure?) Содержит символы (прототипы функций): *fts_children*, *fts_close*, *fts_get_clientptr*, *fts_get_stream*, *fts_open*, *fts_read*, *fts_set*, *fts_set_clientptr*, определённые в файле: **/lib/other/fts.c**	VS FW
1	C	U	UC	**/include/glob.h**	FreeBSD. Какая-то глобальная статистика по директориям. Поиск вариантов пути по данному образцу (вроде he*.h). В системе нигде н используется.	VS FW
1	C	U	UC	**/include/grp.h**	Структура для работы с записями из /etc/group **[include:]** minix/types.h	VS

1	C	U	U	**/include/ifaddrs.h**	FreeBSD. Структура, предназначенная для хранения адреса, маски и целевого адреса, ещё одна структура и функции для работы с ними. Предположительное назначение - для работы с роутами.	VS
2	N	MC	N	**/include/inttypes.h**	Макросы для объявления целых чисел при печати с помощью printf **[include:]** stdint.h	VS
2	C	M	U	**/include/lib.h**	Пара констант, пара функций и туча включений. Чтобы при определённых задачах не забыть ничего включить достаточно использовать только этот заголовочный файл. Все исходники в поддиректориях lib его включают. **[include:]** minix/config.h, minix/types.h, limits.h, errno.h, ansi.h, minix/const.h, minix/com.h, minix/type.h, minix/callnr.h, minix/ipc.h	VS
1	C	U	C	**/include/libgen.h**	OGBSI6. **[include:]** ansi.h Содержит только один прототип функции: basename код которой находится в файле: /lib/other/basename.c	VS
2	U	M	S	**/include/libutil.h**	Функция открытия псевдотерминала **[include:]** termios.h Сама по себе содержит только один прототип функции: int openpty(int *, int *, char *, struct termios *, struct winsize *); код которой находится в файле: /lib/util/openpty.c Используется очень и очень незначительно: /test/select/test14.c:#include <libutil.h> /man/man3/openpty.3:#include <libutil.h> /lib/util/openpty.c:#include <libutil.h>	VS
2	N	MC	N	**/include/limits.h**	Пределы значений разных типов данных и элементов ОС (процессы, ссылки) **[include:]** minix/dir.h	VS
1	C	C	C	**/include/locale.h**	Локаль для Си. Имеется функция для задания локали **[include:]** ansi.h	VS
1	C	C	C	**/include/math.h**	математические функции **[include:]** ansi.h, mathconst.h	VS
2	N	C	C	**/include/mathconst.h**	математические константы (пи, е и т.д.)	VS

1	EX	D	U	**/include/midiparser.h**	часть OSS. Содержит функции для работы с форматом MIDI	VS
2	C	MU	UC	**/include/netdb.h**	OGBSI6. единственная полезная строка – включение другого заголовочного файла **[include:]** net/gen/netdb.h	VS
2	C	U	U	**/include/pwd.h**	Структура и функции для работы с password. **[include:]** minix/types.h	VS
2	C	U	UC	**/include/regex.h**	FreeBSD. Синтаксический разбор и отождествление регулярного выражения **[man:]** regex(3) **[include:]** minix/types.h	VS
2	C	MU	UC	**/include/regexp.h**	Синтаксический разбор и отождествление регулярного выражения **[include:]** ansi.h	VS
1	C	U	C	**/include/setjmp.h**	Функции для сохранения и восстановления контекста исполнения. (Скорее всего находится в rts.o а не в libc.a) **[man:]** setjmp(3) **[include:]** ansi.h Содержит символы (прототипы функций): __setjmp, longjmp, определённые (каждая) в двух файлах, в зависимости от компилятора: lib/ack/rts/setjmp.e или lib/gnu/rts/__setjmp.gs lib/gnu/rts/longjmp.gs	VS FW
2	C	MU	U	**/include/sgtty.h**	не должен нигде использоваться. Кандидат на удаление? В комментарии написано примерно следующее: «Не должен быть ни использован, ни расширен. Файл termios.h – его замена для tty функций, а код, заменяющий ioctl, должен быть перемещён в файл sys/ioctl.h и в специфические файлы для sys или minix каталогов » Однако реально используется в: /commands/zmodem/rbsb.c:#include <sgtty.h> /commands/elle/eesite.c:#include <sgtty.h> /commands/elle/eeterm.c:#include <sgtty.h> /commands/yap/term.h:# include <sgtty.h> /commands/mdb/ioctl.c:#include <sgtty.h>	VS FW

					/lib/editline/sysunix.c:#include <sgtty.h> Сам по себе содержит только определения типов и константы препроцессора. **[include:]** sys/ioctl.h	
1	C	C	UK	**/include/signal.h**	Определены все ANSI и POSIX сигналы и функции (kill, raise, sig*) **[man:]** sigaction(2) **[include:]** ansi.h, minix/types.h	VS
1	N	C	N	**/include/stdarg.h**	ANSI. Обработка аргументов для функций с переменным количеством параметров. **[man:]** stdarg(3) Опрделены макросы: va_start(ap, parmN) prepare to access parameters va_arg(ap, type) get next parameter value and type va_end(ap) access is finished Не содержит прототипов функций. Каким-тот очень косвенным путём имеется зависимость микроядра от этого заголовка: kernel/.depend:utility.o: /usr/include/stdarg.h	VS FW
2	N	C	NK	**/include/stddef.h**	Написано, что там пара часто используемых макросов В реальности только один (довольно странный, о нём написано, что он довольно не портируемый): #define offsetof(type, ident) ((size_t) (unsigned long) &((type *)0)->ident)	VS FW
1	N	C	N	**/include/stdint.h**	Названия типов-синонимов целых чисел, их размеры и границы **[include:]** minix/types.h, minix/sys_config.h, limits.h	VS
1	C	C	UC	**/include/stdlib.h**	OGBSI6. Преобразование строки в числа, выделение и возврат памяти, работа с переменными окружения, случайные числа. Другие общие функции **[include:]** ansi.h	VS
1	C	C	C	**/include/string.h**	OGBSI6. Обработка строк **[man:]** string(3) Общеизвестный стандартный файл, дополненный некоторыми функциями, специфичными для Minix3 (_MINIX). Начало: #define NULL ((void *)0) #ifndef _SIZE_T #define _SIZE_T typedef unsigned int size_t; /* type returned by sizeof */	VS FW

					#endif /*_SIZE_T */ /* Function Prototypes. */ #ifndef _ANSI_H #include <ansi.h> #endif Содержит символы (прототипы функций): memchr, memcmp, memcpy, memmove, memset, strcat, strchr, strncmp, strcmp, strcpy, определённые в каталоге: /lib/i386/string/ (а также: /lib/i86/string/) в файлах, соответственно: memchr.s, memcmp.s, memcpy.s, memmove.s, memset.s, strcat.s, strchr.s, strncmp.s, strcmp.s, strcpy.s, Кроме того содержится и варианты данных функций на языке программирования C в каталоге: /lib/ansi/ в файлах, соответственно: memchr.c, memcmp.c, memcpy.c, memmove.c, memset.c, strcat.c, strchr.c, strncmp.c, strcmp.c, strcpy.c, strcoll: /lib/ansi/strcoll.c	
2	C	C	C	**/include/strings.h**	OGBSI6. Обработка строк (Содержит OLDNAME функции (в терминологии MinGW))	VS FW
1	C	C	U	**/include/stdio.h**	Стандартный ввод-вывод: структура, константы и функции **[man:]** stdio(3) **[include:]** ansi.h, minix/dir.h	VS
1	N	U	N	**/include/sysexits.h**	FreeBSD. Константы кодов выхода из программы (OK, ошибка командной строки, ошибка ОС и т.д.)	VS
1	C	U	U	**/include/syslog.h**	FreeBSD. Для syslogd. Описывает важности сообщений (отладка, внимание, ошибка и т.д.) и их инициаторов (ядро, пользователь, почта, демоны, устройства, сеть, безопасность). В виде списков пар (код – имя). Объявление функций closelog, openlog, syslog. Флаги опций для функций.	VS

					Содержит прототипы функций: void closelog(void); void openlog(const char *, int, int); void syslog(int, const char *,...); определённые в файлах, соответственно:	
1	EX	D	U	**/include/tar.h**	Макросы флагов настроек и структура для утилиты tar. Что она делает в libc? Судя по wiki, является стандартным заголовочнымфайлом (POSIX). Используется в файле: /commands/simple/tar.c	VS FW
2	C	M	U	**/include/termcap.h**	Объявление функций для termcap, в основном, для получения его данных **[man:]** termcap(1), termcap(3), termcap(5) **[include:]** ansi.h	VS
1	C	U	U	**/include/termios.h**	Управляет режимами TTY. Объявлена структура управления терминалом, флаги режимов, настройки по умолчанию для stty **[man:]** termios(3), termios(4) **[include:]** ansi.h	VS
1	C	C	U	**/include/time.h**	Макросы, типы, структуры и функции (включая nanosleep) для работы со временем в разных форматах (от int до строки GMT) **[man:]** time(2) **[include:]** ansi.h	VS
2	T	M	SK	**/include/timers.h**	Структуры и функции для работы с таймерами **[include:]** limits.h, minix/types.h	VS
7	EX	M	SC	**/include/tools.h**	Содержит набор констант и прототипов для довольно низкоуровневого использования (в основном поиск по символам приводит к системному монитору). Ссылается на ассемблерный файл monhead.s. По всей видимости очень специфичен для MINIX. Используется в сервере виртуальной памяти (VM).	FW
2	C	M	U	**/include/ttyent.h**	Содержит структуру и функции для доступа к /etc/ttytab **[man:]** getttyent(3) **[include:]** ansi.h	VS
1	C	U	UKC	**/include/unistd.h**	OGBSI6. Множество констант и функций общего назначения. В основном POSIX. stdin, stdout,stderr; способы выхода из системы; информация о системе (её	VS

					работе), NULL; _exit, alarm; функции доступа к файлам и работы с их правами; запуска процесса и получения его pid gid; работы с конвейером, tty, сетью, устройствами; ещё кое-что **[include:]** minix/types.h, minix/type.h	
1	C	U	U	**/include/utime.h**	Структура, используемая системным вызовом utime при задании файлу времени последнего обращения и времени последнего обновления **[man:]** utime(2) **[include:]** minix/types.h	VS
2	N	M	N	**/include/utmp.h**	Структура, используемая для доступа к данным файлов /etc/utmp и /usr/adm/wtmp, содержащих активных пользователей, историю входа и выхода пользователей из системы. **[man:]** utmp(5) Содержит только определения типов и константы препроцессора. По всей видимости очень специфичен для MINIX.	VS
2	C	M	UC	**/include/arpa/inet.h**	OGBSI6. Содержит макросы и прототипы функций, связанных с сетевым функционалом. Есть упоминание о файле «netinet/in.h». Объявляет функции: htonl : lib/i86/misc/hton86.s htons : lib/i86/misc/hton86.s ntohl : lib/i86/misc/hton86.s ntohs : lib/i86/misc/hton86.s inet_ntoa : lib/ip/inet_ntoa.c inet_aton : (_MINIX) lib/ip/inet_addr.c **#include** <stdint.h>	FW
2	N	A	NK	**/include/ibm/bios.h**	Определения нескольких известных адресов BIOS. Перечисленные здесь адреса могут быть найдены в трёх областях памяти, которые определены в <ibm/memory.h>: - векторы прерываний BIOS, - область данных BIOS, - память BIOS на материнской плате.	FW
2	N	D	NK	**/include/ibm/cmos.h**	Определения для часов реального времени CMOS. Основана на данных для Dallas DS12887, совместима с Motorola MC146818.	FW

					Константы имеют префиксы: RTC_ CMOS_ CS_	
2	N	A	NK	**/include/ibm/cpu.h**	Определения для битов регистра состояний процессора i86. Например: #define X86_FLAGS_USER (X86_FLAG_C | X86_FLAG_P | X86_FLAG_A | X86_FLAG_Z | \ X86_FLAG_S | X86_FLAG_D | X86_FLAG_O)	FW
2	N	AM	N	**/include/ibm/diskparm.h**	Структуры PC (и AT) BIOS для хранения параметров диска. В Minix они используются в основном для форматирования. Конкретно (только одна структура): struct disk_parameter_s	FW
2	N	AM	NK	**/include/ibm/int86.h**	Типы прерываний 8086. Регистры, используемые в реальном режиме процессора PC для вызова сервиса BIOS или DOS. Драйвер вызывается через вектор если номер прерываний нулевой. Конкретно определены типы: union reg86 struct reg86u struct mio_int86 struct mio_ldt86	FW
2	N	AM	NK	**/include/ibm/interrupt.h**	Номера прерываний и аппаратные векторы. В.т.ч. порты контроллера прерываний 8259A; #define END_OF_INT 0x20 векторы прерываний, определённые/резервированные процессором фиксированные векторы системных вызовов (32, 33, 34!); Подходящие базы irq для аппаратных прерываний. Перепрограммирует 8259(s) со значений по умолчанию для PC BIOS, так как BIOS не поддерживает все резервируемые процессором векторы (от 0 до 31); номера аппаратных прерываний; макросы для преобразований номера прерываний в/из аппаратного вектора.	FW
2	N	AM	NK	**/include/ibm/memory.h**	Расположение физической памяти для IBM-совместимых персональных кмпьютеров (PC). Только основные – фиксированные области памяти описываются здесь. Известные адреса областей данных BIOS определены в <ibm/bios.h>. Карта верхней области памяти (UMA) определена только примерно, так как секции UMA могут различаться по размеру и расположению.	FW
3	N	DAM	N	**/include/ibm/partition.h**	Описание записей в таблице партиций. Конкретно – это структура struct part_entry	FW

					а также: #define ACTIVE_FLAG 0x80 #define NR_PARTITIONS 4 #define PART_TABLE_OFF 0x1BE /*смещение таблицы партиций в загрузочном секторе */ /* Типы партиций. */ #define NO_PART 0x00 /* неиспользуемая запись */ #define MINIX_PART 0x81 /* партиция Minix*/ *Список определений может быть дополнен по мере увеличения числа поддерживаемых ОС Minix3 типов партиций и файловых систем.*	
3	N	D	N	**/include/ibm/pci.h**	Содержит константы, важные при взаимодействии с устройствами PCI: - **тип заголовка 00, обычное устройство PCI;** - **тип заголовка 01,** (PCI-to-PCI bridge devices (?)) [следующие регистры являются общими с типом 00: PCI_VID, PCI_DID, PCI_CR, PCI_SR, PCI_REV, PCI_PIFR, PCI_SCR, PCI_BCR, PCI_CLS, PCI_LT, PCI_HEADT, PCI_BIST, PCI_BAR, PCI_BAR2, PCI_CAPPTR, PCI_ILR, PCI_IPR.]; - **тип заголовка 02,** (Cardbus bridge(?)) [общие регистры с типом 00: PCI_VID, PCI_DID, PCI_CR, PCI_SR, PCI_REV, PCI_PIFR, PCI_SCR, PCI_BCR, PCI_CLS, PCI_LT, PCI_HEADT, PCI_BIST, PCI_BAR, PCI_ILR, PCI_IPR; общие регистры с типом 01: PPB_PRIMBN, PPB_SECBN, PPB_SUBORDBN, PPB_SECBLT.] **Кроме того**, значения типов устройств как ([PCI_BCR] << 16)	([PCI_SCR] << 8)	[PCI_PIFR] *Скорее всего может быть дополнен по мере необходимости.*	FW
3	N	DAM	NK	**/include/ibm/ports.h**	Адреса и магические числа для различных портов: #define PCR 0x65 /* Planar Control Register */ #define PORT_B 0x61 /* I/O port for 8255 port B (kbd, beeper...) */ #define TIMER0 0x40 /* I/O port for timer channel 0 */ #define TIMER2 0x42 /* I/O port for timer channel 2 */ #define TIMER_MODE 0x43 /* I/O port for timer mode control */ *Скорее всего может быть дополнен по мере необходимости.*	FW
2	C	M	UK	**/include/minix/a.out.h**	формат исполняемого файла **[include:]** ansi.h	VS FW

					(!!!) В исходниках собственной подсистемы /include/ я не нашёл такого файла !!!	
3	N	M	NK	**/include/minix/bitmap.h**	Макросы для работы с битовыми полями Кроме того содержит определения, перемещённые сюда из **kernel/const.h**. *Этот файл можно осторожно дополнять (только вряд ли это будет нужно).*	VS FW
8	N	M	NK	**/include/minix/callnr.h**	Количество системных вызовов, их номера (работа с файлами, межпроцессное взаимодействие и т.д.) **[Стандарт:]** POSIX Список констант, связанных с системными вызовами Minix3. (Якобы всего 111 системных вызовов , но есть и 120, 121, 122).	FW
3				**/include/minix/cdrom.h**	Этот файл содержит некоторые структуры, используемые драйвером cdrom Mitsumi. Конкретно, определены константы: #define MINUTES 0 #define SECONDS 1 #define SECTOR 2 - и структуры: struct cd_play_mss struct cd_play_track struct cd_disk_info struct cd_toc_entry	FW
8	N	DAM	NK	**/include/minix/com.h**	com=communication. Является парой файлу <minix/ipc.h>. Сообщает, как трактовать элементы сообщений. Содержит диапазоны сообщений и объясняет для чего диапазон используется (ядро, RS, DS, PM, VM, IPC, устройства, шина). **[Стандарт:]** POSIX Этот файл определяет константы для использовании в обмене сообщениями (в основном) между системными поцессами. Определены номер протокола запроса сообщения и типы ответов. Для отладочных нужд, каждому протоколу присвоен его собственный уникальный диапазон номеров. Некоторые такие выделенные диапазоны номеров типов сообщений: * 1 - 0xFF POSIX requests (see callnr.h) * 0x200 - 0x2FF Data link layer requests and responses * 0x300 - 0x3FF Bus controller requests and responses * 0x400 - 0x4FF Block and character device requests * 0x500 - 0x5FF Block and character device responses * 0x600 - 0x6FF Kernel calls to SYSTEM task * 0x700 - 0x7FF Reincarnation Server (RS) requests * 0x800 - 0x8FF Data Store (DS) requests * 0x900 - 0x9FF Requests from PM to VFS, and responses * (0xA00 - 0xAFF old TTY and LOG requests, being phased out)	VS FW

					* 0xA00 - 0xAFF Requests from VFS to file systems (see vfsif.h) * 0xB00 - 0xBFF Requests from VM to VFS * 0xC00 - 0xCFF Virtual Memory (VM) requests * 0xD00 - 0xDFF IPC server requests * 0x1000 - 0x10FF Notify messages Нулевые и негативные значения широко используются для OK и ответов – сообщений об ошибках. Интерес представляют некоторые фрагменты: /* Kernel tasks. These all run in the same address space. */ #define IDLE -4 /* runs when no one else can run */ #define CLOCK -3 /* alarms and other clock functions */ #define SYSTEM -2 /* request system functionality */ #define KERNEL -1 /* pseudo-process for IPC and scheduling */ #define HARDWARE KERNEL /* for hardware interrupt handlers */ /* Number of tasks. Note that NR_PROCS is defined in <minix/config.h>. */ #define MAX_NR_TASKS 1023 #define NR_TASKS 4 /* User-space processes, that is, device drivers, servers, and INIT. */ #define PM_PROC_NR 0 /* process manager */ #define FS_PROC_NR 1 /* file system */ #define VFS_PROC_NR FS_PROC_NR /* FS has been renamed to VFS. */ #define RS_PROC_NR 2 /* memory driver (RAM disk, null, etc.) */ #define MEM_PROC_NR 3 /* memory driver (RAM disk, null, etc.) */ #define LOG_PROC_NR 4 /* log device driver */ #define TTY_PROC_NR 5 /* terminal (TTY) driver */ #define DS_PROC_NR 6 /* data store server */ #define MFS_PROC_NR 7 /* minix root filesystem */ #define VM_PROC_NR 8 /* memory server */ #define INIT_PROC_NR 9 /* init -- goes multiuser */ /* Number of processes contained in the system image. */ #define NR_BOOT_PROCS (NR_TASKS + INIT_PROC_NR + 1) **Содержит макросы, но не содержит ни одного прототипа функций.** **Является одним из глобальных интеграционных файлов**.	
7	N	AM	NK	**/include/minix/config.h**	Настройки ядра, ФС, диспетчера процессов; платформы, количества процессов, контроллеров, (системных|псевдо|RS) консолей и т.д. **[include :]** minix/sys_config.h **[man :]** config(8) Начинается с /* Minix release and version numbers. */ #define OS_RELEASE "3" #define OS_VERSION "1.5"	VS FW

					Этот файл задаёт конфигурационные параметры для микроядра MINIX., сервера файловой системы (VFS), и сервера управления процессами (PM). Он разделён на две основных секции. Первая содержит параметры, которые может устанавливать пользователь (user-settable). Во второй секции различные внутренние системные параметры устанавливаются по установленным пользователем параметрам из первой секции (user-settable parameters). Некоторые части config.h были перенесены в файл sys_config.h (***/include/minix/sys_config.h***), который может быть включён посредством других заголовочных файлов, если возникает желание получить доступ к конфигурационным данным но без излишнего «переполнения» пространства имён (пользователя). Некоторые редактируемые параметры перешли туда. Дальнейшее продолжение (совсем начало): #include <minix/sys_config.h> #define MACHINE _MINIX_MACHINE #define IBM_PC _MACHINE_IBM_PC #define SUN_4 _MACHINE_SUN_4 #define SUN_4_60 _MACHINE_SUN_4_60 #define ATARI _MACHINE_ATARI #define MACINTOSH _MACHINE_MACINTOSH	
8	N	AM	NK	**/include/minix/const.h**	Макросы по сегментам данных, кода, стека, PRIVATE, TRUE, размеры очередей запросов, max размер message, NIL_MESS, параметры сегментов, флаги файла для inode, атрибуты процессов; max номер inode, max смещение в файле (другие ограничения можно найти в файле limits.h) (Может возникнуть вопрос: «что такое clicks?» Если говорить совсем грубо – то это блоки разметки памяти, что-то вроде применяемого в Linux понятия slab.) Содержит множество констант и некоторые макросы, как общего назначения, так и специфические для архитектуры и OS Minix3. Некоторые фрагменты: (1) #ifndef CHIP #error CHIP is not defined #endif (2) #define TRUE 1 /* used for turning integers into Booleans */ #define FALSE 0 /* used for turning integers into Booleans */ #define DEFAULT_HZ 60 /* clock freq (software settable on IBM-PC) */	VS FW

					#define SUPER_USER (uid_t) 0 /* uid_t of superuser */ #define NULL ((void *)0) /* null pointer */ (3) /* Memory is allocated in clicks. */ #if (CHIP == INTEL) #define CLICK_SIZE 4096 /* unit in which memory is allocated */ #define CLICK_SHIFT 12 /* log2 of CLICK_SIZE */ #endif *Включение либо/или связь с кодом микроядра обозначена исходя из общих соображений – конкретно этого doxygen почему-то не обнаруживает.*	
2	C	M	CK	**/include/minix/cpufeature.h**	Константы и функция для определения технических характеристик процессора #define _CPUF_I386_PSE 1 /* Page Size Extension */ #define _CPUF_I386_PGE 2 /* Page Global Enable */ _PROTOTYPE(int _cpufeature, (int featureno)); Реализация содержится в файле: **/lib/i386/misc/_cpufeature.c**	VS FW
2	C	M	C	**/include/minix/crtso.h**	Содержит только: _PROTOTYPE(void _minix_unmapzero, (void)); Это прототип ассемблерной функции, расположенной в **/lib/i386/rts/crtso.s**	FW
3	N	M	NK	**/include/minix/debug.h**	Содержит только макрос FIXME(str) Для напоминания вещей, которые должны быть исправлены. Довольно странный макрос, опирающийся на printf ...	FW
7	N	M	NK	**/include/minix/devio.h**	Типы и константы, используемые для обращения из пользовательского пространства к устройствам через системные вызовы SYS_DEVIO и SYS_VDEVIO, макрос для задания размера порта (байт, слово, long) **[include :]** minix/sys_config.h, minix/types.h Этот файл обспечивает основные типы и некоторые константы для системных вызовов SYS_DEVIO и SYS_VDEVIO, которые позволяют пользовательским процессам (процессам ring3 (?)) выполнять ввод/вывод с устройствами (аппаратными устройствами(?)). ***#include <minix/sys_config.h> /* needed to include <minix/type.h> */*** ***#include <sys/types.h> /* u8_t, u16_t, u32_t needed */*** Связан также (семантически) с <ibm/portio.h>.	VS FW

					Не содержит прототипов функций, но определяет макрос pv_set(pv, p, v) . Содержит также некоторые константы, которые прокомментированы как больше не используемые.	
2	N	M	N	**/include/minix/dir.h**	Структура для i-node и имени директории. Задан размер блока директории. **[include :]** minix/types.h **[man :]** dir(5) *Этого файла нет в подсистеме /include/ Minix3.1.5 stable !*	VS
2	N	M	N	**/include/minix/dirent.h**	Функции чтения директорий. Структуры различных форматов (V7, flex) для элементов директорий. **[include :]** minix/types.h , minix/dir.h **[man :]** directory(3) *Этого файла нет в подсистеме /include/ Minix3.1.5 stable !*	VS
3	N	M	N	**/include/minix/dl_eth.h**	Структура eth_stat используется в запросе DL_GETSTAT для ehw_task. Только структура eth_stat_t; и определена.	FW
2	C	M	U	**/include/minix/dmap.h**	device map? Максимальное количество видов устройств, старшие (и младшие) номера устройств. **[include :]** minix/sys_config.h, minix/ipc.h **[man :]** dev(4) Начало: #include <minix/sys_config.h> #include <minix/ipc.h> Номера устройств (основных и дополнительных((?) Major and minor device numbers) Начало: /* Total number of different devices. */ #define NR_DEVICES 32 /* number of (major) devices */ Сам по себе не содержит прототипов функций. Однако заимсвует таковые из minix/ipc.h	VS FW
2	S	M	S	**/include/minix/ds.h**	Флаги, константы и функции для работы с DS **[include :]** minix/types.h Прототипы и определения для интерфейса сервера данных (DS).	VS FW

					Начало: #include <sys/types.h> Символы (прототипы функций): ds_subscribe, ds_publish_u32, ds_publish_str, ds_retrieve_u32, ds_retrieve_str, ds_check_u32, ds_check_str, расположены (судя по комментарию) в **/lib/syslib/ds.c** . Символы (прототипы функций): mini_ds_retrieve_u32 , расположены в **/lib/sysvipc/ds.c** .	
2	N	MU	NK	**/include/minix/endpoint.h**	Определяет макросы для вычисления так называемых конечных точек. Начало: #include <minix/sys_config.h> #include <minix/com.h> #include <limits.h> #include <minix/type.h> Точка во вставке 'размере поколения' ((?) generation size) нужна для некоторых фиктивных конечных точек ((?) endpoint), таких как NONE, ANY, ... _MAX_MAGIC_PROC определена в <minix/com.h>. Этот включаемый файл определяет некоторые магические числа ((?) magic process numbers) процессов, таких как ANY и NONE, и не должен быть допустимой конечной точкой ((?)valid endpoint number). Поэтому мы убеждаемся, что размер поколения ((?) generation size) достаточно велик, чтобы начать следующее поколение выше самого большого магического числа (magic number(?)). Определяет (файл сам по себе) <u>только</u> макросы: _ENDPOINT_GENERATION_SIZE _ENDPOINT_MAX_GENERATION _ENDPOINT(g, p) _ENDPOINT_G(e) _ENDPOINT_P(e)	FW
2	C	M	C	**/include/minix/fslib.h**	Функции, обеспечивающие (двухсторонний) обмен данными между диском с файловой системой Minix V1, V2 и памятью. Определяет символы (прототипы функций): bitmapsize, conv2, conv4, conv_inode, old_icopy, new_icopy, расположенные в файле **/lib/other/fslib.c** *Гипотеза: этот заголовок в самой системе не используется:* $ grep 'fslib.h' -r ./ ./commands/simple/origmkfs.c:#include <minix/fslib.h> ./commands/simple/fsck.c:#include <minix/fslib.h> ./commands/simple/fsck1.c:#include <minix/fslib.h> ./commands/simple/mkfs.c:#include <minix/fslib.h> ./commands/de/de_recover.c:#include <minix/fslib.h> ./commands/de/de_diskio.c:#include <minix/fslib.h>	FW

					./commands/de/de_stdout.c:#include <minix/fslib.h> ./lib/other/fslib.c:#include <minix/fslib.h>	
1	C	U	U	**/include/minix/ioctl.h**	Задаёт структуру запросов, управляющих вводом-выводом: команда, параметр, режим (приём, передача, и т.д.). Содержит константы и макросы для извлечения элементов запросов и функцию ioctl. Включается всеми заголовочными файлами, объявляющими что-либо, относящееся к управлению вводом-выводом. **[include :]** minix/types.h **[man :]** ioctl(2) Вспомогательные определения для IOCTL. Этот файл включён посредством каждого заголовочного файла, который определяет код IOCTL. Начало: #ifndef _TYPES_H #include <sys/types.h> #endif Символ (прототип функции): int ioctl(int _fd, int _request, void *_data); расположен в файле **/lib/posix/_ioctl.c**	VS FW
2	C	M	U	**/include/minix/ipc.h**	Задаёт структуру всех сообщений, выделяя их типы и некоторые операции над ними, но не сообщает, как трактовать элементы сообщений (для этого смотри com.h). **[include :]** minix/type.h Начало: #include <minix/type.h> Далее расположены: типы, связанные с сообщениями (mess_1; ... mess_9; message;), определения, обеспечивающие (более удобные) имена для наиболее часто используемых членов структуры (message;): m1_i1, ... m9_c2 система выполнения ((?)run-time system) Minix (IPC), (asynmsg_t;) определения для флаговых полей, а также скрытие имён во избежание «засорения» пространтсва имён: #define echo _echo ... Символы (сокрытые! (?)) - прототипы функций: echo, расположены в файле: **/lib/curses/setterm.c** Символы (сокрытые! (?)) - прототипы функций: notify, sendrec, receive, send, sendnb, senda, расположены в файле: **/lib/i386/rts/_ipc.s** Однако их можно искать и в папке /lib/i86/rts/	VS FW

					разбросанными по разным ассемблерным файлам (для 16-битной версии).	
2	N	M	N	**/include/minix/keymap.h**	Определения для работы с раскладами клавиатуры (т.е. в частности, таблицами символов) ((?) defines for keymapping). Файл содержит только макросы и определения. Интересно заключение: #define MAP_COLS 6 /* Number of columns in keymap */ #define NR_SCAN_CODES 0x80 /* Number of scan codes (rows in keymap) */ typedef unsigned short keymap_t[NR_SCAN_CODES * MAP_COLS]; #define KEY_MAGIC "KMAZ" /* Magic number of keymap file */	FW
2	C	MU	U	**/include/minix/minlib.h**	Начало: #ifndef _ANSI_H #include <ansi.h> #endif /* Miscellaneous **BSD**. */ Символы (прототипы функций): swab, itoa, getpass, расположены в файлах каталога **/lib/other/** , соответственно: **swab.c, itoa.c, getpass.c** /* Miscellaneous **MINIX**. */ Символы (прототипы функций): std_err, fsversion, расположены в файлахкаталога **/lib/other/** , соответственно: **stderr.c, fsversion.c** Символы (прототипы функций): load_mtab, rewrite_mtab, get_mtab_entry, put_mtab_entry, расположены в файле **/lib/other/mtab.c** **prints**:: **lib/ack/i386/em/em_print.s** опирается на _write(...) в **lib/stdio/fflush.c** и **lib/stdio/perror.c**. **getprocessor** :: **lib/i86/misc/getprocessor.s** **_cpuid**::**lib/i386/misc/_cpuid.s**	FW
2	EX	M	S	**/include/minix/mq.h**	inet/mq.h Содержит определение структуры mq_t;	FW

					и прототипы функций: mq_get, mq_free, mq_init, реализация которых находится в : **/drivers/libdriver/mq.c** или **/drivers/libdriver_asyn/mq.c** (т.е. не в системе LIBC!!!)	
2	N	M	N	**/include/minix/partition.h**	Дескриптор раздела диска. Используется подсистемами, работающие с диском напрямую. **[include :]** minix/types.h Место партиции на диске и геометрия диска, для использования с DIOCGETP и DIOCSETP (ioctl). Начало: #ifndef _TYPES_H #include <sys/types.h> #endif Содержит только определение структуры: struct partition { u64_t base; /* byte offset to the partition start */ u64_t size; /* number of bytes in the partition */ unsigned cylinders; /* disk geometry */ unsigned heads; unsigned sectors; };	VS FW
2	N	MU	N	**/include/minix/paths.h**	Только определяет константы (препроцессора) - всевозможные пути по умолчанию (настройки из /etc, пути к общедоступным ресурсам): #define _PATH_DHCPCONF "/etc/dhcp.conf" #define _PATH_DHCPPID "/usr/run/dhcpd.pid" #define _PATH_DHCPCACHE "/usr/adm/dhcp.cache" #define _PATH_DHCPPOOL "/usr/adm/dhcp.pool" #define _PATH_WTMP "/usr/adm/wtmp" #define _PATH_UTMP "/etc/utmp" #define _PATH_LASTLOG "/usr/adm/lastlog" #define _PATH_MOTD "/etc/motd" #define _PATH_HOSTS "/etc/hosts" #define _PATH_DEFTAPE "/dev/sa0" #define _PATH_RAMDISK "/dev/ram" #define _PATH_TMP "/tmp" #define _PATH_BSHELL "/bin/sh" #define _PATH_SERVICE "/bin/service" #define _PATH_DRIVERS_CONF "/etc/drivers.conf"	VS FW
2	C	M	U	**/include/minix/portio.h**	Функции чтения и записи данных в порт, разрешения и запрещения прерываний Начало: #ifndef _TYPES_H #include <sys/types.h> #endif	VS FW

					Содержит прототипы функций: *unsigned inb(U16_t _port);* *unsigned inw(U16_t _port);* *unsigned inl(U32_t _port);* *void outb(U16_t _port, U8_t _value);* *void outw(U16_t _port, U16_t _value);* *void outl(U16_t _port, U32_t _value);* *void insb(U16_t _port, void *_buf, size_t _count);* *void insw(U16_t _port, void *_buf, size_t _count);* *void insl(U16_t _port, void *_buf, size_t _count);* *void outsb(U16_t _port, void *_buf, size_t _count);* *void outsw(U16_t _port, void *_buf, size_t _count);* *void outsl(U16_t _port, void *_buf, size_t _count);* которые определены (каждая) в двух каталогах **/lib/i386/misc/** и **/lib/i86/misc/** в ассемблерных файлах, соответственно: **io_inb.s, io_inw.s, io_inl.s, io_outb.s, io_outw.s, io_outl.s, io_insb.s, io_insl.s, io_outsb.s, io_outsw.s, io_outsl.s** . Функции *void intr_disable(void);* *void intr_enable(void);* определены также в двух каталогах **/lib/i386/misc/** и **/lib/i86/misc/** в ассемблерном файле: **io_intr.s**	
3	C	M	U	**/include/minix/profile.h**	Начало: **#include <ansi.h>** Типы, относящиеся к системному профилированию. Типы предназначены как для статистического профилирования, так и для профилирования вызовов (statistical profiling and call profiling). Далее содержится также включение: **#include <sys/types.h>** Заголовочный файл содержит прототипы функций: *profile_get_tbl_size, profile_get_announce, profile_register,* определённые в **/lib/sysutil/profile_extern.c** прототипы функций: *sprofile,* **/lib/syscall/sprofile.s** **/lib/other/_sprofile.c** *cprofile,*	FW

					/lib/syscall/cprofile.s **/lib/other/_cprofile.c**	
2	EX	M	SU	**/ include/minix/queryparam. h**	Структура параметров запросов программы, структура списка параметров, макросы заполнения параметра, функции работы с параметрами запроса. Параметры запрашивающей программы ((?) query program parameters). Определения некоторых объявленных функций находятся вне /lib/ !!! Начало: #include <ansi.h> Содержит символы (прототипы функций): *qp_export*, *queryparam*, их определения находятся в : **/servers/inet/minix3/queryparam.c** *paramvalue*, определение находятся в **/lib/other/paramvalue.c**	VS FW
2	C	M	U	**/include/minix/rs.h**	Структуры (структура …, дескриптор запускаемого драйвера или сервера, структура ACL для PCI) и функция сервера реинкорнации. Интерфейс для сервера реинкарнации (RS). Начало: #include <minix/bitmap.h> Содержит символ (прототип функции): *minix_rs_lookup*, определение находится в **/lib/other/minix_rs.c**	VS FW
2	S	M	CK	**/include/minix/safecopies.h**	Начало: #include <minix/sys_config.h> #include <sys/types.h> #include <stdint.h> Содержит символы (прототипы функций): *cpf_grant_magic*, *cpf_revoke*, *cpf_lookup*, *cpf_getgrants*, *cpf_setgrant_direct*, *cpf_setgrant_indirect*, *cpf_setgrant_magic*, *cpf_setgrant_disable*, *cpf_reload*, определения находятся в: **/lib/syslib/safecopies.c**	FW
2	N	M	N	**/include/minix/sound.h**	Определения, используемые /dev/audio и	FW

					/dev/mixer. Содержит только определения типов: enum Device enum InputState struct volume_level struct inout_ctrl	
2	N	M	N	**/include/minix/swap.h**	Определяет суперблок партиций подкачки (super block of swap partitions) и некоторые полезные константы. /* Two possible layouts for a partition with swapspace: * * Sector Swap partition FS+swap partition * * 0 - 1 bootblock bootblock * 2 swap header FS header * 3 blank swap header * 4 - m swapspace file system * m+1 - n - swapspace */ Содержит только константы (препроцессора) и определение типа: swap_hdr_t;	FW
2	N	M	NK	**/include/minix/sys_config.h**	Настройки, задаваемые пользователем перед компиляцией ядра. Например, аппаратная платформа, количество процессоров. Содержит: - параметры, пригодные для установки пользователем, - размеры слова в байтах (константа эквивалентная sizeof(int)); - установка типа CHIP; - отладочные проверки микроядра (Kernel debug checks); - а также добавленное скриптом дистрибутива: /* Added by release script */ #ifndef _SVN_REVISION #define _SVN_REVISION "R3.1.5-r5612" #endif Не содержит прототипов функций.	VS FW
2	C	M	U	**/include/minix/sysinfo.h**	Функции по получению какой-то информации о системе. В том числе, частоту системных часов. Начало: #include <minix/endpoint.h> #include <minix/type.h> Содержит символы (прототипы функций): *getsysinfo*, *getsysinfo_up*, определённые в : **/lib/other/_getsysinfo.c**	VS FW

					/lib/syscall/getsysinfo.s	
7	SC	AM	CK U	**/include/minix/syslib.h**	Функции системной библиотеки (запуск и управление процессом, команды виртуальной памяти, получение времени, обращение к данным BIOS-а, управление прерываниями, ввод-вывод в устройства, копирование по физическим и виртуальным адресам; получение различной системной информации о процессе, пользователе, статусах; профилирование). **[include:]** minix/types.h, minix/ipc.h, minix/u64.h, minix/devio.h, minix/safecopies.h Прототипы для функций системной библиотеки. Начало: #ifndef _TYPES_H #include <sys/types.h> #endif #ifndef _IPC_H #include <minix/ipc.h> #endif #include <minix/u64.h> #ifndef _DEVIO_H #include <minix/devio.h> #endif #include <minix/safecopies.h> Непосредственно содержит символы (прототипы функций): *sys_abort, sys_enable_iop, sys_exec, sys_fork, sys_newmap, sys_exit, sys_trace, sys_runctl, sys_privctl, sys_setgrant, sys_nice, sys_int86, sys_vmctl, sys_vmctl_get_pagefault_i386, sys_vmctl_get_cr3_i386, sys_vmctl_get_memreq, sys_vmctl_enable_paging, sys_readbios, sys_stime, sys_sysctl, sys_sysctl_stacktrace, sys_sdevio, alloc_contig, sys_times, sys_setalarm, sys_vtimer, sys_irqctl, sys_vircopy, sys_physcopy, sys_safecopyfrom, sys_safecopyto, sys_vsafecopy, sys_memset, sys_umap, sys_umap_data_fb, sys_segctl, sys_getinfo, sys_whoami, sys_kill, sys_sigsend, sys_sigreturn, sys_getksig, sys_endksig, sys_voutb, sys_voutw, sys_voutl, sys_vinb, sys_vinw, sys_vinl, sys_out, sys_in, pci_init, pci_init1, pci_first_dev, pci_next_dev, pci_find_dev, pci_reserve, pci_reserve_ok, pci_ids, pci_rescan_bus, pci_attr_r8, pci_attr_r16, pci_attr_r32, pci_attr_w8, pci_attr_w16, pci_attr_w32, pci_dev_name, pci_slot_name, pci_set_acl, pci_del_acl, sys_sprof, sys_cprof, sys_profbuf, read_tsc_64,* определённые в файлах каталога **/lib/syslib/** **(некоторые также в**	FW

					/lib/other/: sys_enable_iop,) (соответственно): ***sys_abort.c, sys_eniop.c, sys_exec.c, sys_fork.c, sys_newmap.c, sys_exit.c, sys_trace.c, sys_runctl.c, sys_privctl.c, sys_setgrant.c, sys_nice.c, sys_int86.c, sys_vmctl.c, sys_vmctl.c, sys_vmctl.c, sys_vmctl.c, sys_vmctl.c, sys_readbios.c, sys_stime.c, sys_sysctl.c, sys_sysctl.c, sys_sdevio.c, alloc_util.c, sys_times.c, sys_setalarm.c, sys_vtimer.c, sys_irqctl.c, sys_vircopy.c, sys_physcopy.c, sys_safecopy.c, sys_safecopy.c, sys_vsafecopy.c, sys_memset.c, sys_umap.c, alloc_util.c, sys_segctl.c, sys_getinfo.c, sys_getinfo.c, sys_kill.c, sys_sigsend.c, sys_sigreturn, sys_getsig.c, sys_endsig.c, sys_voutb.c, sys_voutw.c, sys_voutl.c, sys_vinb.c, sys_vinw.c, sys_vinl.c, sys_out.c, sys_in.c, pci_init.c, pci_init1.c, pci_first_dev.c, pci_next_dev.c, pci_find_dev.c, pci_reserve.c, pci_reserve.c, pci_ids.c, pci_rescan_bus.c, pci_attr_r8.c, pci_attr_r16.c, pci_attr_r32.c, pci_attr_w8.c, pci_attr_w16.c, pci_attr_w32.c, pci_dev_name.c, pci_slot_name.c, pci_set_acl.c, pci_del_acl.c, sys_sprof.c, sys_cprof.c, sys_profbuf.c, read_tsc_64.c,*** *_taskcall*: **/lib/other/taskcall.c** *read_tsc:* **/lib/sysutil/read_tsc.s** Определения символов (прототипов функций): sys_vm_setbuf, sys_vm_map, найти не удалось !!! Также необходимо отметить: /* Vectored virtual / physical copy calls. */ #if DEAD_CODE /* library part not yet implemented */ _PROTOTYPE(int sys_virvcopy, (phys_cp_req *vec_ptr,int vec_size,int *nr_ok)); _PROTOTYPE(int sys_physvcopy, (phys_cp_req *vec_ptr,int vec_size,int *nr_ok)); #endif	
7	SC	AM	CK SU	**/include/minix/sysutil.h**	Дополнение к minix/syslib.h по части поддержки серверов и драйверов устройств. Функции для работы с переменными окружения, реагирование на функциональную клавишу(?), вывод на печать, много чего ещё. Начало: #include <minix/ipc.h> Особые определения системной библиотеки для поддержки драйверов устройств и серверов. Непосредственно содержит символы (прототипы функций): *env_setargs, env_get_param, env_prefix, env_panic, env_parse, fkey_ctl, report, getuptime, getuptime2, tickdelay, micro_delay_calibrate, sys_hz, util_stacktrace, util_nstrcat, util_stacktrace_strcat, micro_delay, micros_to_ticks, ser_putc,*	FW

					get_randomness, asynsend3 определённые в файлах каталога **/lib/sysutil/** (соответствекнно): ***env_get_prm.c, env_get_prm.c, env_prefix.c, env_panic.c, env_parse.c, fkey_ctl.c, report.c, getuptime.c, getuptime2.c, tickdelay.c, micro_delay.c, sys_hz.c, stacktrace.c, stacktrace.c, stacktrace.c, micro_delay.c, sys_hz.c, ser_putc.c, get_randomness.c, asynsend.c*** *printf*: **/lib/stdio/printf.c** **/lib/sysutil/kprintf.c** *kputc*: **/lib/sysutil/kputc.c** **/lib/sysutil/kprintf.c** *panic*: **/lib/syslib/panic.c**	
2	N	M	N	**/include/minix/tty.h**	Начало: #include <sys/types.h> #define TTYMAGIC 0xb105 Структура, которую драйвер терминала (TTY) может использовать для того, чтобы передать значения загрузочному монитору. На данный момент только значение возникшее в первом vty (conosole) – системной консоли ((?) Currently only the value of the origin of the first vty (console),) , таким образом загрузочный монитор может как полагается показать его когда возникает паника микроядра (when panicing) (терминал (TTY) не планируется чтобы переключиться на первый виртуальный терминал (1st vty)). Это сообщение записывается в конец видео памяти (база видео памяти + размер видео памяти - sizeof(struct boot_tty_info)). ((?) (video memory base + video memory size - sizeof(struct boot_tty_info)) . Не содержит прототипов функций ...	FW
2	N	M	NK	**/include/minix/type.h**	Структуры для работы менеджера памяти, менеджера процессов, структуры для информации по системе (ядру), информации по процессам и параметрам аппаратуры; типы endpoint, физический адрес, виртуальный адрес. Как я понимаю, все эти типы данных предназначены для работы с памятью и данными периода работы системы. **[include:]** minix/sys_config, minix/types.h, minix/config.h, ibm/interrupt.h Начало:	VS FW

					#ifndef _MINIX_SYS_CONFIG_H #include <minix/sys_config.h> #endif #ifndef _TYPES_H #include <sys/types.h> #endif Далее: #include <minix/config.h> #include <ibm/interrupt.h> Файл сам по себе не содержит прототипов функций – только определения типов, константы и макросы препроцессора.	
1	N	MU	NK	**/include/minix/types.h**	Различные типы данных-синонимов char, int и long; они нейтральные. Типы, используемые в структурах disk, inode и т.п. Макросы для вычисления MINOR и MAJOR номеров устройств. Как я понимаю, эти типы данных или полностью абстрактны, или имеют дело с файлами, файловой системой, доступом к файлам. **[include:]** ansi.h **[стандарт:]** OGBSI6 *Мне не удалось найти этот файл в исходниках Minix3.1.5 – stable !* (FW)	VS
2	C	M	CK	**/include/minix/u64.h**	Функции для манипуляций с 64-битными дисковыми адресами. Начало: #ifndef _TYPES_H #include <sys/types.h> #endif Содержит прототипы функций: *add64, add64u, add64ul, sub64, sub64u, sub64ul, diff64, cvu64, cvul64, cv64u, cv64ul, div64u, rem64u, mul64u, cmp64, cmp64u, cmp64ul, ex64lo, ex64hi, make64* определения которых находятся в каталоге: **/lib/i386/int64/** **/lib/i86/int64/** в файлах, соответственно: ***add64.s, add64u.s, add64u.s, sub64.s, sub64u.s, sub64u.s, diff64.s, cvu64.s, cvu64.s, cv64u.s, cv64u.s, div64u.s, div64u.s, mul64u.s, cmp64.s, cmp64.s, cmp64.s, ex64.s, ex64.s, make64.s***	FW
2	N	M	N	**/include/minix/vfsif.h**	Смысл такой же, как и у minix/com.h — задать семантику сообщений. Только в этом файле расположено всё, что касается VFS/FS Поля для сообщений-запросов сервера виртуальной файловой системы (файловой	VS FW

					системы) (VFS/FS). Содержит только определения препроцессора.	
2	SC	M	SU	**/include/minix/vm.h**	Прототипы (и определения – их как раз пока нет!) для интерфейса с сервером(?) виртуальной памяти (VM) . Начало: #include <sys/types.h> #include <minix/endpoint.h> Содержит прототипы функций: *vm_exit*, *vm_fork*, *vm_brk*, *vm_exec_newmem*, *vm_push_sig*, *vm_willexit*, *vm_map_phys*, *vm_unmap_phys*, *vm_allocmem*, *vm_notify_sig*, *vm_ctl*, определения которых расположены в каталоге: **/lib/syslib/** в файлах, соответственно: ***vm_exit.c, vm_fork.c, vm_brk.c, vm_exec_newmem.c, vm_push_sig.c, vm_exit.c, vm_map_phys.c, vm_map_phys.c, vm_allocmem.c, vm_notify_sig.c, vm_ctl.c*** А также: *vm_adddma*, *vm_deldma*, *vm_getdma*: **/lib/other/_vm_dmacalls.c** *vm_set_priv*: **/lib/other/_vm_set_priv.c** *vm_query_exit*: **/lib/other/_vm_query_exit.c**	FW
1	N	M	N	**/include/net/hton.h**	*Макроопределения преобразуют в/из порядка байтов, принятого сетевыми стандартами*. Макросы с именами, написанными строчными буквами гарантируют об исполнении ((?) to evaluate their argument exactly once) аргументов ровно один раз. Функции макросов закодированы в их именах ((?) The function of the macros is encoded in their names); htons означает преобразовать (беззнаковый) short в порядке байтов хоста (компьютера) в сетевой порядок байтов. Начало: #include <minix/sys_config.h> extern u16_t _tmp; extern u32_t _tmp_l; Содержит только макросы и определения препроцессора, но есть интересный фрагмент: #else /* _WORD_SIZE == 2 */	FW

					/* The above macros are too unwieldy for a 16-bit machine. */ u16_t htons(u16_t x); u16_t ntohs(u16_t x); u32_t htonl(u32_t x); u32_t ntohl(u32_t x); #endif /* _WORD_SIZE == 2 */	
3	N	U	N	**/include/net/if.h**	Пустой заголовочный файл! /* net/if.h */	FW
1	C	U	U	**/include/net/ioctl.h**	Команды-макросы для управления вводом-выводом в сети Коды команд сетевого ioctl(). Начало: #include <minix/ioctl.h> Содержит только макросы препроцессора. Существенно опирается на макросы _IOW, _IOR, _IORW, определённые в <minix/ioctl.h>	VS FW
2	C	M	UC	**/include/net/netlib.h**	Начало: #ifndef _ANSI #include <ansi.h> #endif Содержит прототипы функций: *iruserok, rcmd,* определения которых находятся в каталоге: **/lib/ip/** в файлах, соответственно: ***ruserok.c, rcmd.c*** Окончание: #define IPSTAT_DEVICE "/dev/ipstat" #define ETH_DEVICE "/dev/eth" #define IP_DEVICE "/dev/ip" #define TCP_DEVICE "/dev/tcp" #define UDP_DEVICE "/dev/udp"	FW
2	N	N	N	**/include/net/gen/arp_io.h**	Определяет только тип(структуру) nwio_arp_t; и константы препроцессора: #define NWAF_EMPTY 0 #define NWAF_INCOMPLETE 1 #define NWAF_DEAD 2 #define NWAF_PERM 4 #define NWAF_PUB 8	FW

2	C	M	C	**/include/net/gen/dhcp.h**	Формат пакетов DHCP. Определяет тип(структуру) dhcp_t; множество констант препроцессора. Содержит также прототипы функций: *dhcp_init*, *dhcp_settag*, *dhcp_gettag* определённые в каталлоге: **/lib/ip/** в файлах, соответственно: ***dhcp_settag.c, dhcp_settag.c, dhcp_gettag.c***	FW
2	N	N	N	**/include/net/gen/eth_hdr.h**	Определяет только тип (структуру): eth_hdr_t;	FW
2	N	N	N	**/include/net/gen/eth_io.h**	Определяет только типы (структуры): nwio_ethopt_t; eth_stat_t; nwio_ethstat_t; и множество констант препроцессора.	FW
2	N	N	N	**/include/net/gen/ether.h**	Определяет только тип (структур): ether_addr_t; а также: typedef u16_t ether_type_t; typedef U16_t Ether_type_t; а также несколько констант препроцессора, в.т.ч. для полей контрольной информации VLAN и о метках приоритетов ((?) Priority tagging)	FW
2	N	N	N	**/include/net/gen/icmp.h**	Определяет только константы препроцессора с префиксом ICMP_	FW
2	N	N	N	**/ include/net/gen/icmp_hdr.h**	Определяет только типы (структуры): icmp_id_seq_t; icmp_ip_id_t; icmp_ram_t; icmp_pp_t; icmp_mtu_t; icmp_hdr_t;	FW
2	C	M	C	**/include/net/gen/if_ether.h**	Начало: struct ether_addr; #define _PATH_ETHERS "/etc/ethers" Содержит символы (прототипы функций): *ether_ntoa*, *ether_aton*, *ether_ntohost*, *ether_hostton*,	FW

					ether_line определённые в каталоге: **/lib/ip/** в файлах, соответственно: ***ethere2a.c, ethera2n.c, ethern2h.c, etherh2n.c, ether_line.c***	
2	N	N	N	**/include/net/gen/in.h**	Содержит <u>только</u> константы препроцессора и определения типов: typedef u32_t ipaddr_t; typedef u8_t ipproto_t; typedef struct ip_hdropt { u8_t iho_opt_siz; u8_t iho_data[IP_MAX_HDR_SIZE-IP_MIN_HDR_SIZE]; } ip_hdropt_t;	FW
2	C	N	C	**/include/net/gen/inet.h**	Содержит символы (прототипы функций): *inet_addr, inet_network, inet_ntoa, inet_aton* определённые в каталоге: **/lib/ip/** в файлах, соответственно: ***inet_addr.c, inet_network.c, inet_ntoa.c, inet_addr.c***	FW
2	N	N	N	**/include/net/gen/ip_hdr.h**	Содержит <u>только</u> константы (препроцессора) и определение типа: ip_hdr_t;	FW
2	N	N	N	**/include/net/gen/ip_io.h**	Содержит <u>только</u> константы (препроцессора) и определение типов данных: nwio_ipconf2_t; nwio_ipconf_t; nwio_ipopt_t;	FW
2	N	N	N	**/include/net/gen/nameser.h**	Начинается с лицензионных соглашений (Copyright (c) 1983, 1989 Regents of the University of California.) Содержит <u>только</u> константы (препроцессора) и определение типов данных: dns_hdr_t; typedef dns_hdr_t HEADER;	FW
1	C	MU	CU	**/include/net/gen/netdb.h**	Структуры: запись БД, сеть сетевой сервис, протокол, функциии работы с БД. **[include:]** ansi.h **[стандарт:]** FreeBSD	VS FW

					Начинается с лицензионных соглашений (Copyright (c) 1980, 1983, 1988 Regents of the University of California.) Интересны стандартные «пути»: #define _PATH_HEQUIV "/etc/hosts.equiv" #define _PATH_HOSTS "/etc/hosts" #define _PATH_NETWORKS "/etc/networks" #define _PATH_PROTOCOLS "/etc/protocols" #define _PATH_SERVICES "/etc/services" #define _PATH_SERVACCES "/etc/serv.access" Содержит константы (препроцессора) и определение типов данных: struct hostent struct netent struct servent struct protoent Далее: #ifndef _ANSI_H #include <ansi.h> #endif Содержит также символы (прототипы функций): *endhostent, endnetent, endprotoent, endservent, gethostbyaddr, gethostbyname, gethostent, getnetbyaddr, getnetbyname, getnetent, getprotobyname, getprotobynumber, getprotoent, getservbyname, getservbyport, getservent,sethostent, setnetent, setprotoent, setservent, servxcheck, servxfile,* определённые в каталоге: **/lib/ip/** в файлах, соответственно: ***gethostent.c, getnetent.c, getprotoent.c, getservent.c, gethnmadr.c (gethostent.c), gethnmadr.c (gethostent.c), gethostent.c, getnetbyaddr.c, getnetbyname.c, getnetent.c, getprotoname.c, getproto.c, getprotoent.c, getsrvbyname.c,getsrvbyport.c, getservent.c, gethostent.c, getnetent.c, getprotoent.c, getservent.c, servxcheck.c servxcheck.c*** Не удалось найти определения функций: *void herror _ARGS((const char *));*	
1	C	N	C	**/include/net/gen/oneCsum.h**	Содержит прототип функции: *oneC_sum* определённой в файле:	FW

					/lib/ip/oneC_sum.c	
2	N	N	N	**/include/net/gen/psip_hdr.h**	Содержит только константы (препроцессора) и определение типа данных (структуры): psip_io_hdr_t;	FW
2	N	N	N	**/include/net/gen/psip_io.h**	Содержит только константы (препроцессора) и определение типа данных (структуры): nwio_psipopt_t;	FW
2	C	MU	CU	**/include/net/gen/resolv.h**	Начинается с лицензионных соглашений (Copyright (c) 1983, 1987, 1989 The Regents of the University of California.) Конфигурационный файл системы разрешения имён и адресов ((?) Resolver configuration file. : маршрутизатора) Обычно отсутствует, но может содержать адрес сервера изначальных имён для для поиска в очереди и списке доменов. Начало: #ifndef _PATH_RESCONF #define _PATH_RESCONF "/etc/resolv.conf" #endif Содержит также символы (прототипы функций): *res_init, res_mkquery, res_query, res_querydomain, res_search, res_send, _res_close, dn_comp, dn_expand, dn_skipname, __hostalias, _getshort, _getlong, __putshort, __putlong,* определённые в каталоге: **/lib/ip/** в файлах, соответственно: ***res_init.c, res_mkquery.c, res_query.c, res_query.c, res_query.c, res_send.c, res_send.c, res_comp.c, res_comp.c, res_comp.c, res_query.c, res_comp.c, res_comp.c, res_comp.c, res_comp.c*** Не удалось найти файл, в котором определяется символ (прототип функции): *p_query* !	FW
2	N	N	N	**/include/net/gen/rip.h**	(?) Definitions for the Routing Information Protocol (RFC-1058). Содержит только константы (препроцессора) и определение типов данных: rip_hdr_t; rip_entry_t;	FW
2	N	N	N	**/include/net/gen/route.h**	Содержит только константы (препроцессора) и определение типа данных (структуры):	FW

					nwio_route_t;	
2	N	N	N	**/include/net/gen/socket.h**	Начало: /* From SunOS: /usr/include/sys/socketh */ Содержит только константы препроцессора.	FW
2	N	N	N	**/include/net/gen/tcp.h**	#define TCP_MIN_HDR_SIZE 20 #define TCP_MAX_HDR_SIZE 60 #define TCPPORT_TELNET 23 #define TCPPORT_FINGER 79 #define TCPPORT_RESERVED 1024 typedef u16_t tcpport_t; typedef U16_t Tcpport_t; /* for use in prototypes */	FW
2	N	N	N	**/include/net/gen/tcp_hdr.h**	Содержит только константы (препроцессора) и определение типов данных: tcp_hdr_t; tcp_hdropt_t;	FW
2	N	N	N	**/include/net/gen/tcp_io.h**	Содержит только константы (препроцессора) и определение типов данных: nwio_tcpconf_t; nwio_tcpcl_t; nwio_tcpatt_t; nwio_tcpopt_t; tcp_cookie_t;	FW
2	N	N	N	**/include/net/gen/udp.h**	typedef u16_t udpport_t; typedef U16_t Udpport_t; #define UDP_HDR_SIZE 8 #define UDP_IO_HDR_SIZE 16	FW
2	N	N	N	**/include/net/gen/udp_hdr.h**	Содержит только определение типов данных: udp_hdr_t; udp_io_hdr_t;	FW
2	N	N	N	**/include/net/gen/udp_io.h**	Содержит только константы (препроцессора) и определение типа данных (структуры): nwio_udpopt_t;	FW
2	N	N	N	**/include/net/gen/vjhc.h**	Содержит константы препроцессора и множество макросов препроцессора !	FW
3	N	N	N	**/include/netinet/if_ether.h**	Пустой файл!!!	FW
2	N	U	N	**/include/netinet/in.h**	Начало: /* Can we include <stdint.h> here or do we need an additional header that is * safe to include? */ #include <stdint.h> /* Open Group Base Specifications Issue 6 (not complete) */ #define INADDR_ANY (uint32_t)0x00000000 #define INADDR_BROADCAST (uint32_t)0xFFFFFFFF #define INADDR_LOOPBACK (uint32_t)0x7F000001 ...	FW

					И продолжение в таком же духе – только определения типов, константы и макросы препроцессора.	
3	N	U	N	**/include/netinet/tcp.h**	#define TCP_NODELAY 0x01 /* Avoid coalescing of small segments */	FW
2	C	M	SU	**/include/sys/asynchio.h**	Асинхронный ввод/вывод. Это просто «фальшивая» ((?) fake) библиотека асинхронного ввода/вывода, которая использовалась для программ, написанных для Minix-vmd, которые должны были также работать и в стандартном Minix. Этот файл несколько ограничивает число некрасивых #ifdef - ов . Эти программы конечно должны быть ограничены выполнением только одного сервиса. Начало: `#ifndef _ANSI_H` `#include <ansi.h>` `#endif` `#include <sys/time.h>` `typedef struct {` `  char  state;` `  char  op;` `  char  fd;` `  char  req;` `  void  *data;` `  ssize_t  count;` `  int  errno;` `} asynchio_t;` `#define ASYN_NONBLOCK  0x01` `#define ASYN_INPROGRESS  EINPROGRESS` Файл содержит также символы (прототипы функций): *asyn_init*, *asyn_read*, *asyn_write*, *asyn_write*, *asyn_ioctl*, *asyn_wait*, *asyn_synch*, *asyn_close* определённые в файле: **lib/other/asynchio.c**	FW
2	N	M	NK	**/include/sys/dir.h**	Даёт макет каталогов. Его текст (содержит только структуру и константы препроцессора): `#include <sys/types.h>` `#define  DIRBLKSIZ  512  /* size of directory block */` `#ifndef DIRSIZ` `#define DIRSIZ  60` `#endif` `struct direct {` `  ino_t d_ino;` `  char d_name[DIRSIZ];` `};`	FW

2	N	N	N	**/include/sys/file.h**	Путой файл !!!	FW
2	N	M	N	**/include/sys/ioc_cmos.h**	Коды команд CMOS ioctl() Его текст (содержит только структуру и константы препроцессора): #include <minix/ioctl.h> #define CIOCGETTIME _IOR('c', 1, struct tm) #define CIOCGETTIMEY2K _IOR('c', 2, struct tm) #define CIOCSETTIME _IOW('c', 3, u32_t) #define CIOCSETTIMEY2K _IOW('c', 4, u32_t)	FW
2	C	MU	U	**/include/sys/ioc_disk.h**	Коды команд дискового ioctl() Его текст (содержит только структуру и константы препроцессора): #include <minix/ioctl.h> #define DIOCSETP _IOW('d', 3, struct partition) #define DIOCGETP _IOR('d', 4, struct partition) #define DIOCEJECT _IO ('d', 5) #define DIOCTIMEOUT _IORW('d', 6, int) #define DIOCOPENCT _IOR('d', 7, int)	FW
2	C	MU	U	**/include/sys/ioc_file.h**	Коды команд файлового ioctl(). Его текст (содержит только структуру и константы препроцессора): #include <minix/ioctl.h> #define FIONREAD _IOR('f', 1, int)	FW
2	C	MU	U	**/include/sys/ioc_memory.h**	Коды команд ioctl() памяти. Его текст (содержит только структуру и константы препроцессора): #include <minix/ioctl.h> #define MIOCRAMSIZE _IOW('m', 3, u32_t) #define MIOCMAP _IOW('m', 4, struct mapreq) #define MIOCUNMAP _IOW('m', 5, struct mapreq)	FW
2	C	MU	U	**/include/sys/ioc_scsi.h**	Коды команд SCSI ioctl(). Его текст (содержит только структуру и константы препроцессора): #include <minix/ioctl.h> #define SCIOCCMD _IOW('S', 1, struct scsicmd)	FW
2	C	MU	U	**/include/sys/ioc_sound.h**	Коды команд ioctl() для звука. Его текст (содержит только структуру и константы препроцессора): #include <minix/ioctl.h> /* Soundcard DSP ioctls. */ #define DSPIORATE _IOW('s', 1, unsigned int) #define DSPIOSTEREO _IOW('s', 2, unsigned int) #define DSPIOSIZE _IOW('s', 3, unsigned int) #define DSPIOBITS _IOW('s', 4, unsigned int)	FW

					#define DSPIOSIGN _IOW('s', 5, unsigned int) #define DSPIOMAX _IOR('s', 6, unsigned int) #define DSPIORESET _IO ('s', 7) #define DSPIOFREEBUF _IOR('s', 30, unsigned int) #define DSPIOSAMPLESINBUF _IOR('s', 31, unsigned int) #define DSPIOPAUSE _IO ('s', 32) #define DSPIORESUME _IO ('s', 33) /* Soundcard mixer ioctls. */ #define MIXIOGETVOLUME _IORW('s', 10, struct volume_level) #define MIXIOGETINPUTLEFT _IORW('s', 11, struct inout_ctrl) #define MIXIOGETINPUTRIGHT _IORW('s', 12, struct inout_ctrl) #define MIXIOGETOUTPUT _IORW('s', 13, struct inout_ctrl) #define MIXIOSETVOLUME _IORW('s', 20, struct volume_level) #define MIXIOSETINPUTLEFT _IORW('s', 21, struct inout_ctrl) #define MIXIOSETINPUTRIGHT _IORW('s', 22, struct inout_ctrl) #define MIXIOSETOUTPUT _IORW('s', 23, struct inout_ctrl)	
2	C	MU	U	**/include/sys/ioc_tape.h**	Коды команд ioctl() магнитофона (накопителя на магнитной ленте). #include <minix/ioctl.h> #define MTIOCTOP _IOW('M', 1, struct mtop) #define MTIOCGET _IOR('M', 2, struct mtget)	FW
2	C	MU	U	**/include/sys/ioc_tty.h**	Коды команд ioctl() терминала. #include <minix/ioctl.h> /* Terminal ioctls. */ #define TCGETS _IOR('T', 8, struct termios) /* tcgetattr */ #define TCSETS _IOW('T', 9, struct termios) /* tcsetattr, TCSANOW */ #define TCSETSW _IOW('T', 10, struct termios) /* tcsetattr, TCSADRAIN */ #define TCSETSF _IOW('T', 11, struct termios) /* tcsetattr, TCSAFLUSH */ #define TCSBRK _IOW('T', 12, int) /* tcsendbreak */ #define TCDRAIN _IO ('T', 13) /* tcdrain */ #define TCFLOW _IOW('T', 14, int) /* tcflow */ #define TCFLSH _IOW('T', 15, int) /* tcflush */ #define TIOCGWINSZ _IOR('T', 16, struct winsize) #define TIOCSWINSZ _IOW('T', 17, struct winsize) #define TIOCGPGRP _IOW('T', 18, int) #define TIOCSPGRP _IOW('T', 19, int) #define TIOCSFON_OLD _IOW('T', 20, u8_t [8192]) #define TIOCSFON _IOW_BIG(1, u8_t [8192]) /* Legacy <sgtty.h> */ #define TIOCGETP _IOR('t', 1, struct sgttyb) #define TIOCSETP _IOW('t', 2, struct sgttyb) #define TIOCGETC _IOR('t', 3, struct tchars) #define TIOCSETC _IOW('t', 4, struct tchars) /* Keyboard ioctls. */ #define KIOCBELL _IOW('k', 1, struct kio_bell) #define KIOCSLEDS _IOW('k', 2, struct kio_leds) #define KIOCSMAP _IOW('k', 3, keymap_t) /* /dev/video ioctls. */ #define TIOCMAPMEM _IORW('v', 1, struct mapreqvm) #define TIOCUNMAPMEM _IORW('v', 2, struct mapreqvm)	FW

5	C	MU	U	**/include/sys/ioctl.h**	Все коды команд ioctl(). Этот файл включает все другие заголовки кодов команд ioctl. Драйвер, использующий ioctl запрашивает символ для его серии команд. Например: #define TCGETS _IOR('T', 8, struct termios) Это ioctl терминала, которое использует символ 'T' . Символы, используемые в каждом заголовочном файле показаны в следующем за заголовком коментарии. #include <sys/ioc_tty.h> /* 'T' 't' 'k' */ #include <net/ioctl.h> /* 'n' */ #include <sys/ioc_disk.h> /* 'd' */ #include <sys/ioc_file.h> /* 'f' */ #include <sys/ioc_memory.h> /* 'm' */ #include <sys/ioc_cmos.h> /* 'c' */ #include <sys/ioc_tape.h> /* 'M' */ #include <sys/ioc_scsi.h> /* 'S' */ #include <sys/ioc_sound.h> /* 's' */	FW
1	C	U	U	**/include/sys/ipc.h**	Стандартный заголовочный файл (POSIX), содержащий необходимые константы, определения и прототип функции для межпроцессного взаимодействия. Функция ftok непосредственно связана с функцией lstat (обёрткой системного вызова). /* For gid_t, uid_t */ #include <sys/types.h> /* Mode bits for `msgget', `semget', and `shmget'. */ /* Create key if key does not exist. */ #define IPC_CREAT 01000 /* Fail if key exists. */ #define IPC_EXCL 02000 /* Return error on wait. */ #define IPC_NOWAIT 04000  /* Control commands for `msgctl', `semctl', and `shmctl'. */ /* Remove identifier. */ #define IPC_RMID 0 /* Set `ipc_perm' options. */ #define IPC_SET 1 /* Get `ipc_perm' options. */ #define IPC_STAT 2 #define IPC_INFO 3 /* See ipcs. */ /* Special key values. */ /* Private key. */ #define IPC_PRIVATE ((key_t) 0) /* Data structure used to pass permission information to IPC operations. */ struct ipc_perm { key_t key; /* Key. */ uid_t uid; /* Owner's user ID. */ gid_t gid; /* Owner's group ID. */ uid_t cuid; /* Creator's user ID. */ gid_t cgid; /* Creator's group ID. */ unsigned short int mode; /* Reader/write permission. */ unsigned short int __seq; /* Sequence number. */ }; *Функция :*	FW

					_PROTOTYPE(key_t ftok, (const char *__path, int __id)); *определена в файле*: **/lib/sysvipc/ftok.c**	
6	N	MU	N	**/include/sys/jmp_buf.h**	Этот файл содержит только константы препроцессора. Он рассчитан на включение в программный код (возможно даже написанный на языке ассемблера), манипулирующий структурами jmp_buf, sigjmp_buf. Значения являются смещениями (в байтах) внутри структур jmp_buf, sigjmp_buf. Тип jmp_buf определён в файле **/include/setjmp.h** Начало: #include <minix/config.h>	FW
2	C	U	U	**/include/sys/kbdio.h**	Определения, связанные с системным громкоговорителем и индикаторами клавиатуры (NumLock , CapsLock , ScrollLock). Сам по себе содержит только определения типов и определения препроцессора. Используется только в файле **/drivers/tty/keyboard.c** Начало: #include <sys/time.h>	FW
1	C	U	U	**/include/sys/mman.h**	Стандартный файл (POSIX) обеспечивающий определения и прототипы функций – системных вызовов, связанных с отображением файлоа в память (что может быть также использовано для организации разделяемой памяти). **Функционал этого файла не работает так, как он работает, например, в Linux.** Символы (прототипы функций): *mmap*, *munmap*, *munmap_text*, *vm_remap*, *vm_unmap*, *vm_getphys*, *vm_getrefcount* определены в файле: **/lib/posix/_mmap.c**	FW
2	C	M	U	**/include/sys/mount.h**	Определения для mount(2): Используется только в подсистеме /commands/simple/ (и конечно в /lib/). #define MS_RDONLY 0x001 /* Mount device read only */ #define MS_REUSE 0x002 /* Tell RS to try reusing binary from memory */	FW

					/* Function Prototypes. */ #ifndef _ANSI_H #include <ansi.h> #endif Содержит прототипы функций *mount, umount,* определённые в файле: **/lib/other/_mount.c**	
2	N	M	N	**/include/sys/mtio.h**	Команды магнитофона (носителя информации на магнитной ленте). Содержит только константы препроцессора и определения типов (структуры): struct mtop struct mtget	FW
3	N	M	N	**/include/sys/param.h**	#define MAXHOSTNAMELEN 256 /* max hostname size */	FW
2	C	M	UK	**/include/sys/ptrace.h**	Определения для ptrace(2): Содержит константы препроцессора, /* Function Prototypes. */ #ifndef _ANSI_H #include <ansi.h> #endif а также символ (прототип функции): *ptrace,* определённый в файле: **/lib/posix/_ptrace.c**	FW
2	N	M	N	**/include/sys/queue.h**	Начинается с лицензионных соглашений: * Copyright (c) 1991, 1993 * The Regents of the University of California. All rights reserved. Содержит только определения типов (структур), константы препроцессора и множество макросов (которые фактически являются inline функциями). Макросы и типы определяют работу со списками и очередями, в.т.ч. с двойной прошивкой. * For details on the use of these macros, see the **queue(3)** manual page. Включается через **/servers/mfs/inode.h** во все подсистемы, связанные с конкретной реализацией файловых систем: /servers/mfs/ /servers/iso9660fs/ /boot/	FW

					а также используется в файле: **/lib/other/fslib.c**	
1	C	U	UK	**/include/sys/resource.h**	Диапазон приоритетов для интерфейса get/setpriority(). Это не является отображением на приоритеты внутреннего планировщика Minix. #define PRIO_MIN -20 #define PRIO_MAX 20 #define PRIO_PROCESS 0 #define PRIO_PGRP 1 #define PRIO_USER 2 Содержит также прототипы функций: *int getpriority(int, int);* *int setpriority(int, int, int);* определённые в файле: **/lib/posix/priority.c**	FW
1	C	U	U	**/include/sys/select.h**	Начало: #ifndef _POSIX_SOURCE #define _POSIX_SOURCE 1 #endif #include <sys/time.h> #include <sys/types.h> #include <limits.h> #include <string.h> Содержит также прототип функции: _PROTOTYPE(int select, (int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds, struct timeval *timeout)); определённой в файле: **/lib/posix/_select.c**	FW
1	C	U	U	**/include/sys/sem.h**	Определения, структуры и прототипы функций для работы с семаформаи. Начало: #include <sys/types.h> #include <sys/ipc.h> Содержит символы (прототипы функций): *semctl*, *semget*, *semop*, определённые в файле: **/lib/sysvipc/sem.c**	FW
1	C	U	U	**/include/sys/shm.h**	Определения, структуры и прототипы функций для работы с разделяемой памятью (новое в Minix 3.1.5). #include <sys/types.h> #include <sys/ipc.h>	FW

					#include <unistd.h> Содержит символы (прототипы функций): *shmctl*, *shmget*, *shmat*, *shmdt*, определённые в файле: **/lib/sysvipc/shm.c**	
2	C	M	UK	**/include/sys/sigcontext.h**	Структура sigcontext используется для системного вызова sigreturn(2). Системный вызов sigreturn() редко вызывается пользовательскими программами, но он используется внутри (ОС Minix3 – привилегированными процессами (?)) механизмом перехвата сигналов ((?)signal catching mechanism.) Начало: #ifndef _ANSI_H #include <ansi.h> #endif #ifndef _MINIX_SYS_CONFIG_H #include <minix/sys_config.h> #endif #if !defined(_MINIX_CHIP) #include "error, configuration is not known" #endif Содержит, в частности, структуру: ***struct sigregs*** используемую для переключения контекста микроядром. (Сделано также замечание по поводу того, что регистры FPU должны быть добавлены в другую структуру.) Данный заголовок содержит поддержку как i386, так и _CHIP_M68000 Содержит прототип функции: _PROTOTYPE(int sigreturn, (struct sigcontext *_scp)); которая определена в файле: **/lib/posix/_sigreturn.c**	FW
2	C	M	U	**/include/sys/signal.h**	#include <signal.h>	FW
1	C	U	U	**/include/sys/socket.h**	Стандартный файл (POSIX).Содержит определения и прототипы функций для работы с файловыми объектами – сокетами, предназначенными для межпроцессного взаимодействия в.т.ч. между процессами на разных компьютерах - по сетевому соединению. Связан с файлами: **/include/netinet/in.h** **/include/sys/un.h**	FW

					Начало: /* Can we include <stdint.h> here or do we need an additional header that is * safe to include? */ #include <stdint.h> /* Open Group Base Specifications Issue 6 (not complete) */ #include <net/gen/socket.h> Содержит также символы (прототипы функций): accept, bind, connect, getpeername, getsockname, setsockopt, getsockopt, listen, recv, recvfrom, send, sendto, shutdown, socket, определённые в каталоге: **/lib/ip/** в файлах, соответственно: accept.c, bind.c, connect.c, getpeername.c, getsockname.c, setsockopt.c, getsockopt.c, listen.c, recv.c, recvfrom.c, send.c, sendto.c, shutdown.c, socket.c Заключение файла: /* The following constants are often used in applications, but are not defined * by POSIX. */ #define PF_UNIX AF_UNIX #define PF_INET AF_INET	
1	EX	N	U	**/include/sys/soundcard.h**	Заголовочный файл C/C++, который определяет программный интерфейс OSS API. Этот заголовочный файл содержит все объявления (declarations), необходимые для компиляции OSS программ. Последняя версия всегда устанавливается вместе с OSS, использование именно последней версии (данного файла) строго рекомендуется. Следует однако иметь в виду, что заголовочный файл содержит много устаревших определений (для совместимости со старыми приложениями, которые в них нуждаются). Не используйте этот файл как руководство по OSS – используйте руководство программиста OSS (OSS Programmer's guide) для описания деталей программного интерфейса (http://manuals.opensound.com/developer).	FW
1	C	U	U	**/include/sys/stat.h**	Структура, возвращаемая из stat() и fstat(), содержащая информацию об inode (включая данные о типе файла (dir, спецфайл) и правах доступа) **[include:]** minix/types.h **[стандарт:]** OGBSI6, POSIX	FW

					Этот заголовочный файл определяет структуру, которая используется в функциях stat(), fstat(). Информация в этих структурах получается из i-узла ((?) i-node, inode) некоторого файла. Эти вызовы являются единственным общепринятым ((?) approved way to inspect i-nodes) способом для контроля i-узлов ((?) i-node, inode). Начало: #ifndef _TYPES_H #include <sys/types.h> #endif Содержит символы (прототипы функций), *chmod, fchmod, fstat, mkdir, mkfifo, stat, umask, lstat,* определённые в каталоге: **/lib/posix/** в файлах, соответственно: ***_chmod.c, _fchmod.c, _fstat.c, _mkdir.c, _mkfifo.c, _stat.c, _umask.c, _lstat.c***	
1	C	MU	U	**/include/sys/statfs.h**	Данные для системного вызова fstatfs(). #ifndef _TYPES_H #include <sys/types.h> #endif struct statfs { off_t f_bsize; /* file system block size */ }; Содержит также прототип функции: _PROTOTYPE(int fstatfs, (int fd, struct statfs *st)); определённый в файле: **/lib/posix/_fstatfs.c**	FW
2	C	M	U	**/include/sys/svrctl.h**	Команды контроля сервера имеют ту же систему кодирования, что и команды для ioctl. Начало: #ifndef _TYPES_H #include <sys/types.h> #endif /* Server control commands have the same encoding as the commands for ioctls. */ #include <minix/ioctl.h> .... Далее: /* A proper system call must be created later. */ #include <minix/dmap.h> Содержит также прототип функции: _PROTOTYPE(int svrctl, (int _request, void *_data)	FW

					); определённый в файле: **/lib/other/_svrctl.c**	
1	C	U	U	**/include/sys/time.h**	Начало: #include <ansi.h> /* Open Group Base Specifications Issue 6 (not complete) */ Содержит также символы (прототипы функций): *gettimeofday, getitimer, setitimer,* определённые в каталоге: **/lib/posix/** в файлах, соответственно: ***gettimeofday.c, _getitimer.c, _setitimer.c*** Функция *settimeofday* содержится в файле: **/lib/other/settimeofday.c** Фукнция *gettimeofday* содержится <u>также</u> в файле: **/lib/ansi/misc.c**	FW
2	C	MU	U	**/include/sys/timeb.h**	Начало: #include <time.h> struct timeb { time_t time; /* Seconds since epoch, as from `time'. */ unsigned short int millitm; /* Additional milliseconds. */ short int timezone; /* Minutes west of GMT. */ short int dstflag; /* Nonzero if Daylight Savings Time used. */ }; Содержит также прототип функции, заполняющий вышеприведенную структуру информацией о текущем времени: _PROTOTYPE(int ftime, (struct timeb *__timebuf)); определённую в файле: **/lib/stdtime/ftime.c**	FW
1	C	U	U	**/include/sys/times.h**	Заголовочный файл для системного вызова times(). После /* Function Prototypes. */ #ifndef _ANSI_H #include <ansi.h> #endif содержит также прототип функции: _PROTOTYPE(clock_t times, (struct tms *_buffer));	FW

					определённой в файле: **/lib/posix/_times.c**	
1	N	U	NK	**/include/sys/types.h**	Этот заголовочный файл содержит определения важных типов данных. Считается хорошей практикой программирования использовать здесь определённые типы данных вместо соответствующих базовых типов. По соглашению все названия типов имеют суффикс «_t». Начало: #ifndef _ANSI_H #include <ansi.h> #endif Содержит только определения типов, а также некоторые макросы и константы препроцессора.	VS FW
1	C	MU	U	**/include/sys/uio.h**	Определения для векторных операций ввода/вывода. /* Open Group Base Specifications Issue 6 (not complete) */ Содержит также прототипы функций: #if 0 _PROTOTYPE(ssize_t readv, (int _fildes, const struct iovec *_iov, int _iovcnt)); _PROTOTYPE(ssize_t writev, (int _fildes, const struct iovec *_iov, int iovcnt)); #endif Функция *writev* определена в файле **/lib/other/writev.c** Определение функции *readv* найти не удалось.	FW
1	N	U	N	**/include/sys/un.h**	*Этот файл в системе нигде не используется!* /* Open Group Base Specifications Issue 6 */ #ifndef _SA_FAMILY_T #define _SA_FAMILY_T /* Should match corresponding typedef in <sys/socket.h> */ typedef uint8_t sa_family_t; #endif /* _SA_FAMILY_T */ struct sockaddr_un { sa_family_t sun_family; char sun_path[127]; }; /* Note: UNIX domain sockets are not implemented! */	FW
1	C	U	U	**/include/sys/utsname.h**	Этот заголовочный файл даёт имя системы. Начало: #ifndef _ANSI_H #include <ansi.h> #endif	FW

					Содержит также прототип функции: _PROTOTYPE(int uname, (struct utsname *_name)); определённой в файле: **/lib/posix/_uname.c** прототип функции: _PROTOTYPE(int sysuname, (int _req, int _field, char *_value, size_t _len)); определённой в файле: **/lib/other/_sysuname.c**	
2	N	A	N	**/include/sys/video.h**	Содержит: - определения, используемые для драйвера консоли; - константы, связанные с чипом контроллера; - константы для системного гудка; - определения для управления фонтами. Содержит только константы препроцессора.	FW
2	N	M	N	**/include/sys/vm.h**	/* MIOCMAP */ struct mapreq { void *base; size_t size; off_t offset; int readonly; }; /* used in ioctl to tty for mapvm map and unmap request. */ struct mapreqvm { int flags; /* reserved, must be 0 */ off_t phys_offset; size_t size; int readonly; char reserved[36]; /* reserved, must be 0 */ void *vaddr; void *vaddr_ret; };	FW
2	N	A	NK	**/include/sys/vm_i386.h**	Содержит только константы и макросы препроцессора, связанные со страничной организацией памяти и переключения режимов (защиты) процессора i386.	FW
1	C	U	U	**/include/sys/wait.h**	Этот заголовочный файл содержит макросы, связанные с системным вызовом wait(). Значение, возвращаемое wait() и waitpid() зависит от того, как процесс - был завершён системным вызовом exit(), - был завершён сигналом (KILL), - был остановлен в результате контроля заданий. Это показано в схеме: * High byte Low byte * +--------------------+ * exit(status)	status	0 | * +--------------------+ * killed by signal	0	signal | * +--------------------+ * stopped (job control)	signal	0177		FW

					* +----------+ */ Содержит также прототип функции: _PROTOTYPE(pid_t wait, (int *_stat_loc)); определённой в файле: **/lib/posix/_wait.c** и прототип функции: _PROTOTYPE(pid_t waitpid, (pid_t _pid, int *_stat_loc, int _options)); определённой в файле: **/lib/posix/_waitpid.c**	