MAN PAGES

5. Форматы файлов

ACCT

НАЗВАНИЕ acct - файл, содержащий учетные данные процессов

СИНТАКСИС **#include <sys/acct.h>**

ОПИСАНИЕ Ядро поддерживает структуру учетной информации обо всех процессах. При завершении процесса, если система учета запущена, ядро вызывает функцию **acct(2)** для того, чтобы подготовить, а затем записать данные об этом процессе в файл учетных данных. Учетная структура **struct acct** также описана в файле */usr/include/linux/acct.h*.

СМ. ТАКЖЕ **acct(2), sa(1)**

Debian/GNU/Linux 1995 October 31 ACCT**(5)**

ASPLDR.CONF

НАЗВАНИЕ aspldr.conf - конфигурационный файл для ASPLoader

ОПИСАНИЕ

Этот файл (по умолчанию */etc/aspldr.conf*), считывается установщиком загрузчика системы aspldr (см. **aspldr(8)**). Конфигурационный файл состоит из нескольких разделов, каждый из которых начинается с **[названия_раздела]**. Существуют два вида разделов: разделы операционных систем (ОС) и общие разделы.

ОБЩИЕ РАЗДЕЛЫ

Существуют два общих раздела - **[ACTIVATOR]** и **[BOOTMGR]**. Раздел **[ACTIVATOR]** содержит инструкции о том, куда записывается сектор загрузки, и может обрабатывать такие команды:

language *en|ru|ja*
Определяет язык ASPLoader (английский, русский, японский). По умолчанию - английский.

mbrdev *имя_устройства*
Позволяет вам указать устройство, на котором необходимо обновить запись начальной загрузки (MBR).

path *путь*
Для вывода своего меню ASPLoader требует наличия дополнительных файлов, расположенных по умолчанию в каталоге */boot/aspldr*. Данный параметр изменяет этот путь.

biosnum *номер*
Определяет номер жесткого диска (начиная с 1), на котором содержится файл загрузки.

bootfile *имя_файла*
Полный путь к файлу загрузки. По умолчанию это */boot/aspldr.map*.

writeboot *[on|off]*
Определяет, переписывать загрузочный сектор при работе, или нет. По умолчанию - не переписывать (off).

writembr *[on|off]*
Определяет, переписывать запись начальной загрузки при работе, или нет. По умолчанию - не переписывать (off).

Раздел **[BOOTMGR]** определяет поведение загрузчика ASPLoader. Он может содержать такие команды:

align *right|center|left*

Определяет выравнивание названий в меню при загрузке (по левому краю, по центру, по правому краю). По умолчанию названия выравниваются по левому краю.

automatic *on|off*

Если параметр включен (on), то меню выбора операционной системы ASPLoader при старте не показывается, а грузится определенная по умолчанию операционная система, пока не нажата клавиша "ESC". По умолчанию параметр выключен (off).

clock *12|24*

Определяет формат времени (12-часовой или 24-часовой) часов в ASPLoader. По умолчанию определен 24-часовой формат.

default *название_системы*

Определяет, какую операционную систему загружать по умолчанию. *Название_системы* должно соответствовать одному из названий разделов из операционных систем.

delay *число*

Определяет задержку, во время которой ожидается нажатие на клавишу "ESC" (в 1/10 секунды). По умолчанию - 5.

icons *on|off*

Если установлено в "on", то показывает значки в меню. По умолчанию - "on".

logo *on|off*

Если установлено в "on", то показывает логотип и информацию об авторских правах. По умолчанию - "on".

menu *on|off*

Позволяет Вам выключать подменю при выборе системы для загрузки в ASPLoader. По умолчанию - "on".

numbers *on|off*

Если установлено в "off", то меню выбора системы для загрузки не будет пронумеровано. По умолчанию - "on".

savelast *on|off*

Если установлено в "on", то ASPLoader запомнит последнюю выбранную систему загрузки и определит ее как систему для загрузки по умолчанию. По умолчанию - "off".

timeout *число*

Устанавливает количество секунд для выбора системы для загрузки; после истечения этого времени загрузится система по умолчанию. По умолчанию этот параметр равен 30 секундам.

video *pseudographics|graphics|text*

Определяет видеорежим меню ASPLoader (псевдографический, графический, текстовый). По умолчанию - графический. Заметим, что программа VMware немного некорректно работает с псевдографикой.

РАЗДЕЛ ОПЕРАЦИОННЫХ СИСТЕМ

Раздел операционных систем определяет пункты меню для загрузки во время работы ASPLoader. Среди всех разделов существует один особый элемент, называемый **[SEPARATOR].** При его использовании в меню загрузчика рисуется разделяющая линия.

Остальные разделы имеют такой формат: [метка@Нормально читаемое название] , где **метка** - уникальное внутреннее название этого раздела, а **Нормально читаемое название** - описание раздела, которое будет видно в меню загрузчика.

Следующие команды определены для использовании в описании разделов операционных систем:

initrd *имя_файла*

Полное имя файла начального электронного диска. Имеет значение только для разделов, описывающих пункты меню для систем Linux.

kernel *имя_файла***[аргументы]**

Полное имя файла для загрузки ядра Linux. *Аргументами* могут быть любые параметры для ядра Linux, в большинстве случаев вы будете использовать хотя-бы параметры root=/dev/<root_устройство> и ro.

lilo *аргументы*

Если у Вас есть другая версия Linux, установленная на машине, и эта версия использует загрузчик lilo, записанный в загрузочный сектор своего root-раздела, то этот параметр позволит вам запустить из загрузчика ASPLoader загрузчик lilo. Обычно используется вместе с параметром **sysboot**.

sysboot *[раздел]***[a:|b:]**

Параметр sysboot загружает и исполняет загрузочный сектор с указанного раздела диска (формат <номер-диска>-<номер-раздела>, например для второго раздела на первом диске - 1-2). Особые формы записи "a:" и "b:" используются для загрузки с первого или второго флоппи-дисковода соответственно. Это бывает удобно, если вы хотите грузится с дискет, но в BIOS загрузка с них отключена.

vga *ask|normal|number*

Устанавливает видеорежим при загрузке Linux. *ask* запросит номер видеорежима при загрузке, *number* позволяет Вам сразу указать номер видеорежима при загрузке. По умолчанию видеорежим обычный (normal).

ПРИМЕР

Пример, приведенный ниже, определяет конфигурацию из 4-х пунктов меню (два разных ядра linux на одном разделе, раздел Windows 2K и загрузку с дискеты). Windows установлена на /dev/hda1, а загрузчик записан в главной загрузочной записи (MBR). Время показывается в 12-часовом формате, язык загрузчика - русский.

```
[linux@ASPLinux]
icon linux
kernel /boot/vmlinuz mem=188M root=/dev/hda6 ro
initrd /boot/initrd.img

[linux1@ASPLinux Test Kernel]
icon linux
kernel /boot/bzImage root=/dev/hda6 ro
initrd /boot/initrd.img

[SEPARATOR]

[nt1@Windows 2001]
icon windows
sysboot 1-1

[SEPARATOR]

[floppy@Boot from floppy]
icon floppy
sysboot a:
```

```
[BOOTMGR]
video graphics
default linux
timeout 30
clock 12

[ACTIVATOR]
writeboot off
writembr on
biosnum 1
mbrdev /dev/hda
language ru
```

СМ. ТАКЖЕ **aspldr(8).** 7 January 2001
ASPLDR.CONF**(5)**

---

charmap

НАЗВАНИЕ charmap - набор символов, определяющий их кодировки

ОПИСАНИЕ Это набор символов (charmap) и их кодировка. Все поддерживаемые наборы символов должны иметь **совместимый набор символов** в качестве вспомогательного. для того, чтобы на него можно было потом сослаться.

СИНТАКСИС

Файл с набором символов начинается с заголовка, который может содержать следующие ключевые слова:

<codeset>,
: за которым следует имя кодировки.

<mb_cur_max>,
: за которым следует максимальное число байтов для многобайтового символа. Многобайтовые символы в данный момент не поддерживаются. По умолчанию это число равно 1.

<mb_cur_min>,
: за которым следует минимальное число байтов на символ. Данное значение может быть меньше или равно **mb_cur_max**. Если оно не задано, то по умолчанию принимается значение **mb_cur_max**.

<escape_char>;
: за ним следует символ, который должен использоваться как символ экранирования во всех нижеследующих строках, чтобы выделять символы, интерпретируемые специальным образом. По умолчанию таким символом является обратная косая черта (****).

<comment_char>,
: за которым следует символ, который будет использоваться в качестве символа комментария во всех нижеследующих строках. По умолчанию таким символом является знак решетки (**#**).

Само определение набора символов начинается с ключевого слова **CHARMAP** в начале строки. Последующие строки могут иметь два формата, предназначенных для задания кодировки символов:

<символьное-имя> <кодировка> <комментарии>
: Такая форма задает ровно один символ и его кодировку.

<символьное-имя>...<символьное-имя> <кодировка> <комментарии>
: Такая форма задает несколько символов. Она удобна только для многобайтовых символов, которые в данный момент не внесены в программы.

Последняя строка в файле определения набора символов

должна содержать **END CHARMAP**.

СИМВОЛЬНЫЕ НАЗВАНИЯ

Символьное имя содержит только символы из **совместимого набора символов.** Само имя заключается в угловые скобки. Символы, которые следуют за **<escape_char>**, теряют специальное значение и интерпретируются как сами символы; например, последовательность **'<\\\>>'** представляет собой символьное имя **'>'**, заключенное в угловые скобки.

КОДИРОВКА СИМВОЛОВ

Кодировка может быть любой из трех следующих:

<escape_char>d<число>
с десятичным числом;

<escape_char>x<число>
с шестнадцатеричным числом;

<escape_char><число>
с восьмеричным числом.

ФАЙЛЫ

*/usr/share/i18n/charmaps/**

АВТОР

Jochen Hein (jochen.hein@delphi.central.de)

СООТВЕТСТВИЕ POSIX.2

СМ. ТАКЖЕ locale(1), localedef(1), setlocale(3), localeconv(3), locale(5)

National Language Support 28 Nov 1994 **charmap(5)**

DIR_COLORS

НАЗВАНИЕ dir_colors - файл конфигурации **dircolors(1)**

ОПИСАНИЕ

Программа **ls(1)** использует переменную окружения **LS_COLORS** для определения цвета которым будут написаны названия файлов. Эта переменная окружения обычно устанавливается командой

eval `dircolors some_path/dir_colors`

находящейся в файле инициализации оболочки установленной по умолчанию, например в */etc/profile* или */etc/csh.cshrc*. (См. также **dircolors(1)**.) Обычно в этом случае используется файл */etc/DIR_COLORS* установки которого могут быть переназначены файлом *.dir_colors* в домашнем каталоге пользователя.

Этот конфигурационный файл состоит из нескольких операторов, по одному на строку. Если в строке встречается символ решетки (#), а этот символ находится в начале строки или ему предшествует хотя бы один пробел, то часть строки, находящаяся правее решетки, считается комментарием. Пустые строки игнорируются.

Секция *глобальных* настроек состоит из любых операторов до первого оператора **TERM**. Любой оператор в глобальной секции действителен для всех терминалов. За глобальной секцией следуют одна или более секций *терминалов*; эти секции начинаются с одного или нескольких операторов **TERM**, определяющих типы терминалов (в формате, аналогичном содержимому переменной окружения **TERM**). Операторы, заданные для конкретного терминала, имеют больший, по сравнению с глобальными, приоритет, поэтому некоторые глобальные значения могут быть изменены в секции терминалов.

Существуют следующие операторы (регистр неважен):

TERM *тип-терминала*
Определяет начало секции терминала и определяет, какому терминалу эта секция соответствует.

Несколько подряд идущих операторов **TERM** означают, что операторы этой секции относятся к нескольким видам терминалов.

COLOR yes|all|no|none|tty
(Только в Slackware; игнорируется GNU **dircolors(1)**.) Задает, что цвет всегда должен использоваться (*yes* или *all*), никогда не должен использоваться (*no* или *none*) или должен использоваться только в том случае, если вывод ведется на терминал (*tty*). По умолчанию используется *no*.

EIGHTBIT yes|no
(Только в Slackware; игнорируется GNU **dircolors(1)**.) Задает использование восьмибитных символов ISO 8859. Для совместимости допустимы также такие значения: 1 для yes (использовать) и 0 для no (не использовать). По умолчанию - *no*.

OPTIONS *опции*
(Только в Slackware; игнорируется GNU **dircolors(1)**.) Добавлять заданные опции к стандартной командной строке **ls**. Это могут быть любые опции команды **ls**, включая и ведущий знак -. Заметьте, что **dircolors** не проверяет правильность этих опций.

NORMAL *цвет*
Задает цвет обычного текста (не имени файла).

FILE *цвет*
Задает цвет имени обычного файла.

DIR *цвет*
Задает цвет имени каталога.

LINK *цвет*
Задает цвет имени символьной ссылки.

ORPHAN *цвет*
Задает цвет имени "разорванной" символьной ссылки (указывающей на несуществующий файл). Если эта опция не задана, то **ls** будет использовать цвет, определенный в **LINK**.

MISSING *цвет*
Задает цвет отсутствующего файла (несуществующего файла, на который указывает символьная ссылка). Если эта опция не задана, то **ls** будет использовать цвет, определенный в **FILE**.

FIFO *цвет*
Задает цвет имени FIFO (поименованного канала).

SOCK *цвет*
Задает цвет имени сокета.

DOOR *color-sequence*
(поддерживается начиная с file-utils 4.1) Задает цвет, используемый для двери (door) (Solaris 2.5 и более поздние версии). **BLK** *цвет* Задает цвет имени блочного устройства.

CHR *цвет*
Задает цвет имени символьного устройства.

EXEC *цвет*
Задает цвет имени запускаемого файла.

LEFTCODE *цвет*
Задает *левый код* не-ISO 6429 терминалам (см. ниже).

RIGHTCODE *цвет*
Задает *правый код* не-ISO 6429 терминалам (см. ниже).

ENDCODE *цвет*
Задает *конечный код* не-ISO 6429 терминалам (см. ниже).

*****_расширение цвет_

Задает цвет файлов, имя которых оканчивается на _расширение_.

.*расширение цвет*

То же самое, что и *****._расширение_. Задает цвет файлов, имя которых оканчивается на _расширение_. Заметьте, что точка включена в расширение: это делает невозможным задать расширение, начинающееся не с точки, например, **~** в резервных копиях файлов **emacs**. Эта форма устарела.

ПОСЛЕДОВАТЕЛЬНОСТЬ ЦВЕТОВ ISO 6429 (ANSI)

Многие цветные ASCII-терминалы сегодня используют цветовые последовательности ISO 6429 (ANSI), а многие нецветные терминалы, включая **xterm** и широко распространенный DEC VT100, распознают цветовые последовательности ISO 6429 и без проблем игнорируют или эмулируют их. **ls** использует по умолчанию коды ISO 6429, если цвет "включен". Цветовые последовательности ISO 6429 представляют собой числа, разделенные точками с запятой. Наиболее распространенные коды:

0	восстанавливает стандартные цвета
1	цвета повышенной яркости
4	подчеркнутый текст
5	"мигающий" текст
30	черный цвет текста
31	красный цвет текста
32	зеленый цвет текста
33	желтый (коричневый) цвет текста
34	синий цвет текста
35	фиолетовый цвет текста
36	голубой цвет текста
37	белый (серый) цвет текста
40	черный фон
41	красный фон
42	зеленый фон
43	желтый (коричневый) фон
44	синий фон
45	фиолетовый фон
46	голубой фон
47	белый (серый) фон

Не все команды будут работать во всех системах или устройствах вывода.

По умолчанию **ls** использует следующие значения:

NORMAL	0	Обычный (не имя файла) текст
FILE	0	Обычный файл
DIR	32	Каталог
LINK	36	Символьная ссылка
ORPHAN	не_определено	Разорванная ссылка
MISSING	не_определено	Отсутствующий файл
FIFO	31	Именованый канал (FIFO)
SOCK	33	Сокет
BLK	44;37	Блочное устройство
CHR	44;37	Символьное устройство
EXEC	35	Исполняемый файл

Некоторые терминалы неправильно обрабатывают код 0. Если все стало выводиться в цвете, то смените коды операторов **NORMAL** и **FILE** на цифровые коды стандартных цветов текста и фона вашего терминала.

ДРУГИЕ ТИПЫ ТЕРМИНАЛОВ (РАСШИРЕННАЯ КОНФИГУРАЦИЯ)

Если Ваш цветной (или подсвечивающий) терминал (или принтер!) использует другие цветовые последовательности, используйте операторы **LEFTCODE**, **RIGHTCODE** и **ENDCODE.**

При выводе имени файла **ls** генерирует следующую

последовательность: **LEFTCODE** *код_типа* **RIGHTCODE** *имя_файла* **ENDCODE**, где *код_типа* - это цветовая последовательность, зависящая от типа или имени файла. Если последовательность **ENDCODE** на задана, то вместо нее используется последовательность **LEFTCODE NORMAL RIGHTCODE**. Цель левых и правых кодов - уменьшить количество задаваемых последовательностей (и скрыть экранирование от пользователя). Если эти коды не подходят терминалу, то Вы можете отключить их, задав в секции Вашего терминала одно ключевое слово, без параметров.

ЗАМЕЧАНИЕ: Если последовательность **ENDCODE** задана в глобальной секции файла конфигурации, то ее *нельзя* пропустить в секциях терминалов. Это означает, что определение **NORMAL** не будет иметь никакого эффекта. Однако, можно задать другой оператор, **ENDCODE**, который будет работать как обычно.

ЭКРАНИРУЮЩИЕ ПОСЛЕДОВАТЕЛЬНОСТИ

Для задания управляющих символов или пробелов в цветовых последовательностях или расширениях файлов можно использовать \-последовательность (\-escaped notation) в стиле языка C или **stty**-совместимую ^-последовательность. C-совместимая последовательность включает в себя следующие символы:

\a	Звуковой сигнал (ASCII 7)
\b	Забой (ASCII 8)
\e	Escape (ASCII 27)
\f	Перевод формата (ASCII 12)
\n	Перевод строки (ASCII 10)
\r	Возврат каретки (ASCII 13)
\t	Табуляция (ASCII 9)
\v	Вертикальная табуляция (ASCII 11)
\?	Удаление символа (ASCII 127)
*****nnn*	Любой символ (восьмеричная нотация)
\x*nnn*	Любой символ (шестнадцатеричная нотация)
_	Пробел
****	Обратная косая черта (\)
\^	Галочка (^)
\#	Решетка (#)

Заметьте, что эти последовательности необходимы при использовании пробела, обратной косой черты, галочки или любого управляющего символа в любом месте строки.

ЗАМЕЧАНИЯ

По умолчанию **LEFTCODE** и **RIGHTCODE** соответствуют стандартам терминалов ISO 6429:

```
LEFTCODE  \e[
RIGHTCODE m
```

По умолчанию **ENDCODE** не определен.

СМ. ТАКЖЕ **dircolors(1), ls(1), stty(1), xterm(1)**

ФАЙЛЫ

/etc/DIR_COLORS
Общесистемный файл конфигурации.

~/.dir_colors
Пользовательские файлы конфигурации.

ЗАМЕЧАНИЯ

Данная страница описывает формат файла **dir_colors** в соответствии с пакетом fileutils-4.1; другие версии могут немного отличаться от этой. Исправления и дополнения присылайте по адресу: aeb@cwi.nl. Отчеты об ошибках в этой программе присылайте по адресу: fileutils-bugs@gnu.ai.mit.edu.

ENVIRON

НАЗВАНИЕ environ - переменные пользовательского окружения

СИНТАКСИС

extern char **environ;

ОПИСАНИЕ

Переменная *environ* указывает на массив строк, называемый "окружением". (Эта переменная должна быть указана в пользовательской программе, но, на самом деле, ее определение находится в файле *unistd.h*, если это libc4 или libc5, или glibc, если в ней задано определение **_GNU_SOURCE.**) Этот массив становится доступен процессу после его запуска функцией **exec(3).** Эти строки имеют форму `*название*=*значение*'. Наиболее распространенные переменные окружения:

USER Имя пользователя (используется некоторыми программами, перенесенными из BSD);

LOGNAME
Имя пользователя (используется некоторыми программами, перенесенными из System-V);

HOME Домашний каталог пользователя, в который он попадает при входе в систему. Устанавливается командой **login(1)** в соответствии с файлом **passwd(5).**

LANG Название локали, используемой по умолчанию, если ее значение не аннулировано **LC_ALL**, или другой переменной окружения, такой как **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**, **LC_MONETARY**, **LC_NUMERIC**, **LC_TIME**. См. также **locale(5).**

PATH Набор каталогов, в которых **sh(1)** и многие другие программы производят поиск файла, заданного неполным именем. Эти каталоги разделяются знаком `:'. (Так же определяются и другие подобные переменные: **CDPATH**, используемая некоторыми оболочками для поиска каталога при выполнении смены каталога; **MANPATH**, используемая командой **man(1)** для поиска страниц руководства и т.п.)

PWD Текущий рабочий каталог. Устанавливается некоторыми оболочками.

SHELL Имя файла пользовательской оболочки.

TERM Тип терминала, на который производится вывод.

PAGER Предпочитаемая пользователем утилита просмотра текстовых файлов.

EDITOR/**VISUAL**
Предпочитаемая пользователем утилита редактирования текстовых файлов.

BROWSER
Программа для просмотра URL предпочитаемая пользователем. Последовательность команд просмотра, разделенных двоеточием. Загляните на http://www.tuxedo.org/~esr/BROWSER/ .

Другие переменные могут быть помещены в окружение командами **export**, командой `название=значение'в **sh(1)** или командой **setenv**, если вы используете **csh(1).** Параметры могут быть помещены в окружение при исполнении функции **exec(3).** В программе, написаннной на языке C, Вы можете работать с окружением при помощи: **getenv(3)**, **putenv(3)**, **setenv(3)** и **unsetenv(3).** Заметьте, что поведение многих программ и библиотечных функций зависит от наличия или значения некоторых переменных окружения. Опишем некоторые

из них.

Переменные **LANG**, **LANGUAGE**, **NLSPATH**, **LOCPATH**, **LC_ALL**, **LC_MESSAGES**, и т.п. влияют на работу локалей. См. **locale(5)**.

TMPDIR влияет на префикс полного имени файла, создаваемого **tmpnam(3)** и другими подобными функциями, а также является временным каталогом для **sort(1)** и других программ.

LD_LIBRARY_PATH, **LD_PRELOAD** и другие переменные LD_* влияют на работу динамического загрузчика библиотек.

POSIXLY_CORRECT заставляет некоторые программы и библиотеки точно следовать требованиям POSIX.

Поведением функций **malloc(3)** управляют переменные MALLOC_*.

Переменная **HOSTALIASES** задает имя файла с псевдонимами машин; это имя используется **gethostbyname(3)**.

TZ и **TZDIR** содержат информацию о часовом поясе, используемую **tzset(3)**, и, через нее, функциями типа *ctime*(), *localtime*(), *mktime*(), *strftime*(). См. также **tzselect(1)**.

TERMCAP содержит информацию о работе с заданным терминалом (или задает имя файла, содержащего такую информацию).

.LP **COLUMNS** и **LINES** сообщают приложениям размер окна, которые могут изменить размеры установленные по умолчанию.

PRINTER или **LPDEST** могут указывать нужный принтер для печати. См. **lpr(1)**.

НАЙДЕННЫЕ ОШИБКИ

Очевидно, что это представляет угрозу системе безопасности. Многие системные команды могут быть "сбиты с толку" пользователем, задающим необычные значения **IFS** или **LD_LIBRARY_PATH**. Так же существует риск изменения имен файлов. Такие программы как *make* и *autoconf* позволяют переименовывать названия файлов, заменяя некоторые переменные на такие же в верхнем регистре. Поэтому некоторые используют перменную окружения **CC** для выбора нужного компилятора C (и соответствующих **MAKE**, **AR**, **AS**, **FC**, **LD**, **LEX**, **RM**, **YACC**, и т.д.). Иногда через переменные окружения передают параметры для программы. Для этого используют переменные **MORE**, **LESS**, и **GZIP**. это считается порочной практикой и избегается в новых программах. А авторы *gzip* должны оставлять опцию **GZIP_OPT**.

СМ. ТАКЖЕ

login(1), **sh(1)**, **bash(1)**, **csh(1)**, **tcsh(1)**, **execve(2)**, **exec(3)**, **getenv(3)**, **putenv(3)**, **setenv(3)**, **unsetenv(3)**, **locale(5)**

Linux 2001-12-14 ENVIRON**(5)**

FTPUSERS

НАЗВАНИЕ ftpusers - список пользователей, которым запрещен доступ на FTP-сервер

ОПИСАНИЕ

Текстовый файл **ftpusers** содержит список пользователей, которым запрещен доступ на FTP-сервер (File Transfer Protocol). Этот файл используется не только в целях администрирования, но и для улучшения системы безопасности в среде сети TCP/IP. Обычно файл содержит список пользователей которым не положено пользоваться этим сервисом или которые имеют слишком много прав. Обычно это пользователи: root, daemon, bin, uucp и news. Если ваш демон FTP не использует **ftpusers**, то рекомендуется прочесть документацию на тему безопасности и заблокировать доступ некоторым прользователям. Демон FTP-сервиса Вашингтонского Университета (wuftpd) и демон Professional

FTP (proftpd) используют именно файл **ftpusers.**

ФОРМАТ

Формат **ftpusers** очень прост. В каждой строке записывается одна учетная запись (или имя пользователя). Строки начинающиеся с символа `#' игнорируются.

ФАЙЛ

/etc/ftpusers

СМ. ТАКЖЕ **passwd(5), wuftpd(8), proftpd(8)**

Linux August 27, 2000 FTPUSERS**(5)**

GROUP

НАЗВАНИЕ group - файл, описывающий группы пользователей

ОПИСАНИЕ

/etc/group -- это ASCII-файл, описывающий группы, к которым принадлежат пользователи. В нем находятся однострочные записи в формате

group_name:passwd:GID:user_list

Используются следующие поля:

group_name
- название группы;

password
- пароль группы (зашифрованный). Если это поле пустое, пароля не требуется;

GID - цифровой идентификатор группы;

user_list
- имена всех пользователей, входящих в группу, разделенные запятыми.

ФАЙЛЫ /etc/group

СМ. ТАКЖЕ **login(1), newgrp(1), passwd(5)**

Linux December 29 1992 GROUP**(5)**

HOST.CONF

НАЗВАНИЕ host.conf - файл, содержащий настройки для резольвера

ОПИСАНИЕ

Файл */etc/host.conf* содержит настройки для библиотеки резольвера (resolver). Резольвером называется механизм преобразования имен узлов (обычно компьютеров) сети в IP-адреса и наоборот (так называемое прямое и обратное преобразование, прим. перев.) Данный файл должен содержать в каждой строке одно ключевое слово, за которым следует информация о настройке. Вот эти ключевые слова: *order*, *trim*, *multi*, *nospoof* и *reorder*. Каждое ключевое слово описывается ниже.

order Данное ключевое слово задает метод, с помощью которого будет осуществляться поиск адреса узла. За этим словом должно следовать одно или несколько названий методов, разделенных запятыми. Допустимы следующие названия методов: *bind*, *hosts* и *nis*.

trim Данное ключевое слово может быть использовано больше одного раза. Каждый раз, когда указывается это слово, за ним должно следовать одно имя домена, начинающееся с точки. Данная настройка приказывает resolv+ отсекать заданное имя домена в конце всех имен, которые преобразуются в адреса с помощью DNS. Этот параметр предназначен для использования с локальными узлами и доменами. (Замечание: отсекание не будет выполняться в случае с именами узлов, обрабатываемых с помощью NIS или файла

/etc/hosts). Вы должны позаботиться о том, чтобы первое имя узла для каждой записи в файле /etc/hosts являлось полностью заданным (имя узла с доменом) или неполностью заданным (только имя узла) в зависимости от того, как это необходимо Вашим локальным настройкам.

multi Допустимыми для этого ключевого слова являются значения *on* и *off*. Если задано *on*, то библиотека resolv+ будет возвращать все допустимые адреса узла, которые встретились в файле */etc/hosts*, а не только первый из них. По умолчанию стоит *off*, так как в противном случае возможно существенное снижение производительности сайтов с большими файлами узлов.

nospoof

Допустимыми для этого ключевого слова являются значения *on* и *off*. Если задано *on*, то библиотека resolv+ будет пытаться оградить систему от подложных имен узлов, тем самым обезопасив **rlogin** и **rsh**. Это выполняется так: после поиска адреса узла по имени резольвер будет искать имя узла по адресу. Если эти два имени не совпадут, то результат операции будет признан ошибочным.

spoofalert

Если установленное значение данного ключевого слова равно *on* , и при этом также установлено значение *nospoof*, то резольвер будет записывать в системный журнал предупреждения об ошибках посредством sys-log. По умолчанию установленное значение этого ключевого слова равно *off*.

reorder

Допустимыми значениями этого ключевого слова являются *on* и *off*. Если задано значение *on*, то резольвер будет пытаться перегруппировать адреса узлов так, чтобы локальные адреса (т.е., адреса в той же подсети) были выведены первыми, когда выполняется вызов **gethostbyname(3).** Перегруппировка выполняется для всех методов поиска. По умолчанию установленное значение равно *off*.

ФАЙЛЫ

/etc/host.conf

Файл, содержащий настройки резольвера

/etc/resolv.conf

Файл, содержащий настройки резольвера

/etc/hosts

Локальная база данных узлов

СМ. ТАКЖЕ gethostbyname(3), hostname(7), resolv+(8), named(8)

Debian/GNU/Linux 1997 January 2 HOST.CONF**(5)**

HOSTS

НАЗВАНИЕ hosts - статическая таблица преобразования IP-адресов в имена машин

СИНТАКСИС /etc/hosts

ОПИСАНИЕ На это странице содержится описание формата файла */etc/hosts*. Этот файл содержит текст, описывающий соответствие IP-адресов и имен машин (по одному адресу в строке). Для каждой машины в одной строке должна присутствовать следующая информация:

IP_адрес каноническое_имя алиасы

Поля этой записи отделяются друг от друга пробелами и/или табуляциями. Текст, начинающийся с символа "#", до конца строки считается комментарием и игнорируется. Имена машин могут содержать любой печатный символ, кроме разделителя полей, символа новой строки или символа комментария. Алиасы представляют собой измененные, альтернативные, укороченные или обобщенные формы имен машин (например, *localhost*). Формат таблицы имен машин описан в RFC 952.

Сервер доменных имен интернет Berkeley (BIND) содержит сервер интернет-имен для машин UNIX. Он заменяет файл */etc/hosts* или систему поиска машин и освобождает машину от необходимости полного подробного заполнения файла */etc/hosts*.

Несмотря на то, что функции этой таблицы давно исполняются DNS, она все еще используется для

начальной загрузки системы.
Во многих системах есть небольшая таблица имен машин, содержащая информацию об именах и адресах важных машин в локальной сети. Это полезно в том случае, если DNS не работает, например, при загрузке системы.

NIS Сайты, работающие с NIS, используют таблицу имен машин в качестве источника информации для базы данных машин NIS. Несмотря на то, что NIS может работать и с DNS, многие NIS-сайты используют таблицу имен машин со строками для всех машин локальной сети в качестве подстраховки.

изолированные узлы
Небольшие сайты, не подключенные к глобальной сети, используют таблицу имен машин вместо DNS. Если локальная информация меняется редко, а сеть не подключена к интернет, то DNS вряд ли необходим.

ПРИМЕР

```
127.0.0.1       localhost
192.168.1.10    foo.mydomain.org  foo
192.168.1.13    bar.mydomain.org  bar
216.234.231.5   master.debian.org      master
205.230.163.103 www.opensource.org
```

ИСТОРИЧЕСКАЯ СПРАВКА

Перед введением DNS, таблица имен машин была единственным средством для преобразования имен машин в адреса в растущей сети интернет. На самом деле, этот файл создавался на основе официальной базы данных машин, поддерживаемой в Сетевом информационном контрольном центре (NIC), а также на основе локальных исправлений, содержащих неофициальные алиасы и/или информацию о неизвестных машинах. NIC больше не поддерживает файлы hosts.txt, однако, во время написания этой страницы (около 2000 года) существовали старые файлы hosts.txt на WWW. Они были найдены; даты их размещения - 92, 94 и 95 годы.

ФАЙЛЫ */etc/hosts*

СМ. ТАКЖЕ **hostname(1)**, **resolver(3)**, **resolver(5)**, **hosts(5)**, **hostname(7)**, **named(8)**, Internet RFC 952

АВТОР Эта страница написана Manoj Srivastava <srivasta@debian.org> для системы Debian GNU/Linux.

Debian March 12 2000 HOSTS**(5)**

HOSTS.EQUIV

НАЗВАНИЕ /etc/hosts.equiv - выводит список узлов и пользователей, которым предоставляется "доверительный" доступ к Вашей

системе посредством **r**-команды

ОПИСАНИЕ

Файл **hosts.equiv** разрешает или запрещает узлам и пользователям использовать **r**-команды (например, **rlogin**, **rsh** или **rcp**) без ввода пароля.

Файл имеет следующий формат:

[+ | -] [hostname] [username]

Где *hostname* - это имя узла, которое логически эквивалентно локальному узлу. Пользователям, которые работают на этом узле, разрешается доступ к локальному узлу под их собственным именем без ввода пароля. Имени *hostname* может (необязательно) предшествовать знак плюс (+). Если в качестве имени узла используется одиночный плюс, то доступ к Вашей системе разрешен с любого узла. Вы можете явно запретить доступ какого-либо узла, если перед *hostname* поставите знак "минус" (-). Пользователи этого узла всегда будут должны вводить пароль. Из соображений безопасности Вы всегда должны указывать полные имена узлов (имя с доменом) и не использовать коротких имен.
Запись *username* предоставляет доступ к машине заданному пользователю (исключая root) без ввода пароля. Это означает, что пользователь может задавать любые имена в локальной машине. Имени *username* может (необязательно) предшествовать знак "плюс" (+). Вы можете явно запретить доступ к машине какому-либо пользователю, если перед *username* поставите знак "минус" (-).
С помощью знака @ может быть задана сетевая группа.
Будьте очень осторожны при указании знака "плюс" (+). Простая описка может привести к тому, что этот плюс будет стоять в строке в полном одиночестве, что означает "любой узел"!

ФАЙЛЫ /etc/hosts.equiv

ЗАМЕЧАНИЕ

Некоторые системы используют содержимое данного файла только тогда,когда его владельцем является root, и при этом отсутствуют права на запись для всех остальных. Некоторые "исключительно параноидальные" системы даже требуют, чтобы на данный файл не существовало других "жестких" ссылок.

СМ. ТАКЖЕ **rhosts(5), rshd(8), rlogind(8)**

Linux 29 Jan 1995 HOSTS.EQUIV**(5)**

INTRO

НАЗВАНИЕ intro - введение в раздел, описывающий форматы файлов

ОПИСАНИЕ

В этом разделе описываются различные форматы файлов и протоколов и используемые структуры языка C, если таковые существуют.

АВТОРЫ

Список авторов, авторских прав и условия распространения информации можно найти в заголовке каждой конкретной страницы руководства. Обратите внимание, что они могут отличаться от странице к странице!

Linux 24 July 1993 INTRO**(5)**

IPC

НАЗВАНИЕ ipc - межпроцессные коммуникационные механизмы System V

СИНТАКСИС

 # include <sys/types.h>

```
# include <sys/ipc.h>
# include <sys/msg.h>
# include <sys/sem.h>
# include <sys/shm.h>
```

ОПИСАНИЕ

На этой странице описываются межпроцессные коммуникационные механизмы System V, реализованные в Linux: очереди сообщений, наборы семафоров и сегменты разделяемой памяти. Ниже слово **ресурс** означает рабочую единицу одного из этих механизмов.

Права доступа к ресурсам

Для каждого ресурса система использует общую структуру типа **struct ipc_perm**, чтобы хранить информацию, необходимую для определения прав доступа при исполнении ipc-операции. Структура **ipc_perm**, заданная в файле системных заголовков *<sys/ipc.h>*, состоит из следующих полей:

```
ushort cuid; /* идентификатор пользователя-создателя */
ushort cgid; /* идентификатор группы-создателя */
ushort uid; /* идентификатор пользователя-владельца */
ushort gid; /* идентификатор группы-владельца */
ushort mode; /* права на чтение/запись */
```

Поле **mode** структуры **ipc_perm** определяет в своих младших 9-и битах права доступа процесса к ресурсу, вызывающему системную функцию ipc. Права определяются следующим образом:

```
0400 Чтение пользователем.
0200 Запись пользователем.
0040 Чтение группой.
0020 Запись группой.
0004 Чтение прочими.
0002 Запись прочими.
```

Биты 0100, 0010 и 0001 (права на исполнение) не используются. Более того, "Запись" на самом деле означает "Изменение" набора семафоров.

Тот же файл системных заголовков определяет и некоторые символьные константы:

IPC_CREAT	Создать ресурс, если такого ключа не существует.
IPC_EXCL	Вернуть ошибку, если ключ существует.
IPC_NOWAIT	Вернуть ошибку, если запрос требует ожидания.
IPC_PRIVATE	Индивидуальный ключ.
IPC_RMID	Удалить ресурс.
IPC_SET	Установить опции ресурса.
IPC_STAT	Получить опции ресурса.

Заметьте, что **IPC_PRIVATE** имеет тип **key_t**, а остальные символьные константы являются флагами, которые можно использовать в качестве операнда функции логического сложения с переменной типа **int**.

Очереди сообщений

Очереди сообщений указываются с помощью положительного числа, называемого (идентификатором очереди сообщений *msqid*). У каждой очереди существует структура **struct msquid_ds**, определенная в *<sys/msg.h>*, состоящая из следующих полей:

```
struct ipc_perm msg_perm;
ushort msg_qnum; /* количество сообщений в
```

очереди */
ushort msg_qbytes; /* максимальное количество байтов в очереди */
ushort msg_lspid; /* идентификатор процесса, последний раз вызвавшего функцию msgsnd */
ushort msg_lrpid; /* идентификатор процесса, последний раз вызвавшего функцию msgrcv */
time_t msg_stime; /* время последнего вызова msgsnd */
time_t msg_rtime; /* время последнего вызова msgrcv */
time_t msg_ctime; /* последнее время изменения состояния очереди */

msg_perm **ipc_perm** - это структура, определяющая права доступа к очереди сообщений.
msg_qnum Количество сообщений, находящихся на текущий момент в очереди.
msg_qbytes Максимальное количество байтов текста, которое может находиться в очереди сообщений.
msg_lspid Идентификатор процесса, последний раз вызвавшего системную функцию **msgsnd.**
msg_lrpid Идентификатор процесса, последний раз вызвавшего системную функцию **msgrcv.**
msg_stime Время последнего вызова системной функции **msgsnd.**
msg_rtime Время последнего вызова системной функции **msgrcv**
msg_ctime Время последнего вызова системной функции, изменившей поле структуры **msqid_ds.**

Наборы семафоров

Набор семафоров однозначно определяется положительным числом, называемым (идентификатором набора семафоров *semid*). У каждого набора семафоров имеется структура данных **struct semid_ds,** определенная в *<sys/sem.h>*и состоящая из следующих полей:

struct ipc_perm sem_perm;
time_t sem_otime; /* время последней операции */
time_t sem_ctime; /* время последнего изменения */
ushort sem_nsems; /* количество семафоров в наборе */

sem_perm **ipc_perm** - это структура, определяющая права доступа к набору семафоров.
sem_otime Время последнего вызова функции **semop.**
sem_ctime Время последнего вызова функции **semctl,** изменившего одно из полей вышеописанной структуры или состояние одного из семафоров этого набора.
sem_nsems Количество семафоров в наборе. Каждый семафор в наборе определяется неотрицательным числом диапазона от **0** до **sem_nsems-1.**

Семафор - это структура типа **struct sem,** состоящая из следующих полей:

ushort semval; /* значение семафора */
short sempid; /* идентификатор процесса, последний раз работавшего с семафором */
ushort semncnt; /* количество процессов, ожидающих увеличения значения семафора */
ushort semzcnt; /* количество процессов, ожидающих обнуления значения семафора */

semval Значение семафора: неотрицательное целое.
sempid Идентификатор процесса, последний раз

выполнявшего операцию с этим семафором.

semncnt Количество процессов, ожидающих увеличения значения **semval**.

semznt Количество процессов, ожидающих обнуления значения **semval** .

Сегменты разделяемой памяти

Сегмент разделяемой памяти однозначно определяется положительным числом, называемым (идентификатором сегмента разделяемой памяти *shmid*). У каждого сегмента разделяемой памяти имеется структура данных **struct shmid_ds**, определенная в *<sys/shm.h>*, состоящая из следующих полей:

```
struct ipc_perm shm_perm;
int shm_segsz;  /* размер сегмента */
ushort       shm_cpid;          /*      идентификатор
пользователя-создателя */
ushort  shm_lpid;      /*  идентификатор  процесса,
последний раз работавшего с сегментом */
short    shm_nattch;       /*    количество    текущих
подключений сегмента */
time_t shm_atime;    /* время последнего  подключения
сегмента */
time_t  shm_dtime;     /* время последнего отключения
сегмента */
time_t shm_ctime;    /*  время  последнего  изменения
структуры */
```

shm_perm **ipc_perm** - это структура, определяющая права доступа к сегменту разделяемой памяти.

shm_segsz Размер сегмента разделяемой памяти в байтах.

shm_cpid Идентификатор процесса, создавшего сегмент разделяемой памяти.

shm_lpid Идентификатор процесса, последний раз вызывавшего функции **shmat** или **shmdt**.

shm_nattch Количество текущих подключений сегмента разделяемой памяти.

shm_atime Время последнего вызова функции **shmat**.

shm_dtime Время последнего вызова функции **shmdt**.

shm_ctime Время последнего вызова функции **shmctl**, изменившего одно из полей **shmid_ds**.

СМ. ТАКЖЕ

ftok(3), msgctl(2), msgget(2), msgrcv(2), msgsnd(2), semctl(2), semget(2), semop(2), shmat(2), shmctl(2), shmget(2), shmdt(2).

Linux 0.99.13 November 1, 1993 IPC**(5)**

ISSUE

НАЗВАНИЕ

issue - файл идентификации, содержащий сообщение, выдаваемым перед приглашением в систему "login:"

ОПИСАНИЕ

Файл **/etc/issue** является текстовым файлом, который содержит какое-либо сообщение или информацию о системе, появляющиеся перед приглашением в систему "login:". Файл может содержать разные последовательности вида: **@***символ* и *****символ*, - если они поддерживаются программой **getty(1)**.

ФАЙЛЫ

/etc/issue

СМ. ТАКЖЕ **getty(1), motd(5)**

Linux 24 July 1993 ISSUE**(5)**

ISSUE

НАЗВАНИЕ issue - файл идентификации, содержащий сообщение,

выдаваемым перед приглашением в систему "login:"

ОПИСАНИЕ

Файл **/etc/issue** является текстовым файлом, который содержит какое-либо сообщение или информацию о системе, появляющиеся перед приглашением в систему "login:". Файл может содержать разные последовательности вида: **@***символ* и *****символ*, - если они поддерживаются программой **getty(1)**.

ФАЙЛЫ /etc/issue

СМ. ТАКЖЕ **getty(1), motd(5)**

Linux 24 July 1993 ISSUE**(5)**

LILO.CONF

НАЗВАНИЕ lilo.conf - конфигурационный файл lilo

ОПИСАНИЕ

Файл */etc/lilo.conf* по умолчанию считывается системным менеджером загрузки lilo (см. **lilo(8)**).
Он может выглядеть таким образом:

```
boot = /dev/hda
delay = 40
compact
vga = normal
root = /dev/hda1
read-only
image = /zImage-2.5.99
        label = try
image = /zImage-1.0.9
        label = 1.0.9
image = /tamu/vmlinuz
        label = tamu
        root = /dev/hdb2
        vga = ask
other = /dev/hda3
        label = dos
        table = /dev/hda
```

Данный конфигурационный файл определяет, что lilo использует основную загрузочную запись (MBR) на устройстве /dev/hda. (Для знакомства с различными способами использования lilo и его работы с другими операционными системами смотрите файл user.tex из документации о lilo.) В начале работы менеджер загрузки будет ожидать нажатия на клавишу Shift 4 секунды (40 децисекунд). Если на нее не нажимают, то начнет загружаться первый указанный образ ядра (/zImage-1.5.99, тот, который Вы, вероятнее всего, установили буквально пять минут назад). Если Вы нажали на Shift, то менеджер спросит, какой образ загружать. В случае, если Вы забыли возможные варианты загрузки, нажмите на [TAB] (или [?] - если клавиатура соответствует стандарту США), и Вам будет показано меню. Вы должны выбрать объект загрузки: новое ядро, старое (проверенное) ядро, ядро другой файловой системы (в случае, если Вы допустили какую-либо грубую ошибку при работе с обычной файловой системой), - или загружать другую операционную систему. В файле lilo.conf таким образом может быть указано до 16-и образов.
Как можно увидеть выше, конфигурационный файл начинается с некоторого числа общих параметров (верхние 6 строк в примере), сопровождаемых описаниями параметров разных образов. Параметры в описании каждого образа имеют приоритет над параметрами в общей секции.

ОБЩИЕ ПАРАМЕТРЫ

Существует множество подходящих для работы команд. Описания, приведенные ниже, просто взяты из файла user.tex (только немного урезаны).

backup=*резервный_файл*

Копировать загрузочный сектор в файл *резервный_файл* (который может быть устройством, например, */dev/null*) вместо */boot/boot.NNNN*).

boot=*устройство***загрузки**

Устанавливает имя устройства (например, раздел жесткого диска), которое ссылается на загрузочный сектор. Если эта команда пропущена, то загрузочный сектор считывается с устройства, которое в данный момент запущено как root, и, вероятнее всего, записывается на него.

change-rules

Определяет, возможно ли изменить количество видимых разделов при загрузке (`скрытие' разделов). Смотрите секцию "Правила изменения типов разделов" в user.tex для знакомства с подробностями.

compact

Пытается объединять запросы на чтение рядом расположенных секторов в единый запрос. Это очень уменьшает время загрузки и снижает объем карт (\map). Использование режима `compact' особенно рекомендуется при загрузке данных с дискет.

default=*название*

Использовать указанное название образа как выбираемое по умолчанию при загрузке. Если параметр `default' пропущен, то в качестве него принимается первый образ, описанный в конфигурационном файле.

delay=*децисекунд*

Определяет количество десятых долей секунд, на которые менеджер загрузки приостанавливает запуск выбранного образа. Это бывает необходимо в системах, которые начинают загрузку с жесткого диска сразу после того, как клавиатура инициализирована. Менеджер загрузки не будет приостанавливать запуск, если параметр `delay' пропущен или установленное значение его равно нулю.

disk=*название_устройства*

Определяет нестандартные параметры указанному диску. Для знакомства с подробностями смотрите раздел "Геометрия диска" в user.tex. Особенно полезен параметр `bios='. Вообще, BIOS нумерует ваши диски по порядку, например, 0x80, 0x81 и т.п., но, в целом, бывает невозможно определить соответствие между дисками в Linux и дисками в BIOS (так как это зависит от настроек BIOS и от типа BIOS), так что если в системе есть нестандартные настройки, Вам будет необходимо установить соответствие между Linux-дисками и дисками BIOS. Например,

```
disk=/dev/sda
        bios=0x80
disk=/dev/hda
        bios=0x81
```

определяет, что Ваш SCSI-диск является первым диском BIOS, а Ваш главный (первичный) IDE-диск является вторым диском BIOS.

disktab=*файл_disktab*

Определяет название таблицы дисковых параметров. Инсталлятор карты (\map installer) ищет файл */etc/disktab* в случае, если параметр `disktab' пропущен. Использовать этот параметр не

рекомендуется.

fix-table

Данная опция позволяет lilo править 3D-адреса в таблице разделов. Каждый элемент таблицы содержит 3D (сектор/головка/цилиндр) и линейные адреса первого и последнего секторов раздела. Если раздел не расположен в соответствии с цилиндрами и если какие-то другие операционные системы (например, PC/MS-DOS или OS/2) используют тот же диск, то это может изменить 3D-адреса. А lilo может сохранять свой сектор загрузки только в тех разделах, где оба типа адреса совпадают. Если установлен параметр `fix-table', то lilo может изменять (исправлять) некорректные 3D-адреса. ПРЕДУПРЕЖДЕНИЕ: данная опция не гарантирует, что другие операционные системы не попытаются в дальнейшем изменить какие-либо адреса. Также есть вероятность, что внесенные изменения (после запуска данной опции) приведут к неожиданным побочным эффектам. Исправить ошибки можно будет с помощью программы, которая перераспределяет разделы на диске, выравнивая разделы в соответствии с цилиндрами. На некоторых дисках (например, на некоторых больших EIDE-дисках с системой переопределения адресов) при некоторых обстоятельствах, возможно, всегда будет происходить конфликт в адресах разделов между элементами диска.

force-backup=*резервный_файл*

Работает так же, как `backup', но при этом переписывает старую резервную копию, если она существует.

ignore-table

указывает lilo на то, что следует игнорировать поврежденную таблицу разделов.

install=*сектор_загрузки*

Устанавливает указанный файл как новый сектор загрузки. Если `install' пропущено, то по умолчанию используется файл */boot/boot.b*
.

linear Генерировать линейные адреса секторов вместо 3D-адреса (сектор/головка/цилиндр). Линейные адреса преобразуются и не зависят от геометрии диска. Заметим, что загрузочные диски могут стать несовместимыми с другими системами, если использована опция `linear', так как сервисы BIOS, определяющие геометрию диска, ненадежно работают с флоппи-дисководами. При использовании `linear' с большими дисками, */sbin/lilo* может создать ссылки на недоступные области диска, так как 3D-адреса секторов перед загрузкой не до конца известны.

lock Включает автоматическое исполнение команд загрузки по умолчанию для далее указанного образа загрузки. Таким образом, lilo "закрепляет" данный выбор до тех пор, пока он не будет заново определен вручную.

map=*файл_карты***(map-file)**

Определяет расположение файла карты. Если параметр `map' пропущен, то по умолчанию используется файл */boot/map* .

message=*файл_сообщения*

определяет файл, содержащий сообщение, отображаемое перед приглашением загрузки. Во время ожидания нажатия клавиши [Shift] (после вывода строки "LILO ") на экран ничего не выводится. Символ FF сообщения([Ctrl L]) "очищает" экран. Размер файла

сообщения ограничен до 65535-и байтов. Файл карты (\map file) должен быть собран заново, если файл сообщений изменен или куда-то перемещен.

nowarn Не выводить предупреждения о возможных некорректных последствиях работы.

optional
Определенный в этом случае параметр каждого образа `optional' (смотрите ниже) относится ко всем образам.

password=*пароль*
Определенный в данном случае параметр каждого образа `password=...' (смотрите ниже) относится ко всем образам.

prompt Принудительно выводит приглашение загрузки, не ожидая никаких нажатий клавиатуры. Автоматическая перезагрузка невозможна, если `prompt' устанавлено, а `timeout' не включено.

restricted
Определенный в данном случае параметр каждого образа `restricted' (смотрите ниже) относится ко всем образам.

serial=*параметры*
позволяет управлять операцией с помощью последовательной линии связи. Указанный последовательный порт инициализируется, и менеджер загрузки начинает принимать посредством этого порта и клавиатуры PC входящие данные. Отправка сигнала прерывания в последовательную линию осуществляется нажатием клавиши [Shift] на консоли и служит для привлечения внимания менеджера загрузки. Все образы загрузки должны быть защищены паролем, если доступ к последовательной линии менее безопасен, чем доступ к консоли, например, если линия подключена к модему. Строка параметров имеет следующий вид:

<port>[,<bps>[<parity>[<bits>]]]

<port>: номер последовательного порта, начинающийся с нуля. 0 соответсвует COM1, оно же является /dev/ttyS0 и т.п. Могут быть использованы все четыре последовательных порта (если они существуют).
<bps>: скорость передачи данных последовательного порта. Поддерживаются следующие скорости передачи: 110, 150, 300, 600, 1200, 2400, 4800 и 9600 б/с. По умолчанию скорость устанавливается равной 2400 б/с.
<parity>: проверка четности в последовательной линии. Менеджер загрузки игнорирует входящую четность и укорачивает 8-ой бит. Последующие символы (в верхнем или в нижнем регистре) используются для описания четности: "n"- для снятия проверки, "e" - для проверки четности, "o" - для проверки нечетности.
<bits>: количество битов в символе. Поддерживаются только 7 или 8 битов. По умолчанию принимается размер, равный 8-и битам, если проверка снята, и 7-и битам, если проверка нечетности или четности запущена.
Если запущен параметр `serial', то значение `delay' автоматически увеличивается до 20-и.
Пример: serial=0,2400n8 инициализирует COM1 с параметрами по умолчанию.

timeout=*децисекунд*

устанавливает ограничение по времени (в десятых долях секунды) ввода команды с помощью клавиатуры. Если никаких клавиш в указанное время не нажато, то автоматически начинает запускаться первый образ. Ввод пароля также отменяется, если пользователь слишком долго не выполняет никаких операций. По умолчанию ограничение по времени бесконечно.

verbose=*уровень*

Запускает вывод большого количества сообщений при работе. Большие номера соответствуют большему количеству сообщений. Если в командной строке lilo дополнительно указан параметр -v , то уровень, соответственно, увеличивается. Максимальный уровень вывода сообщений равен 5-и.

Также такие параметры конфигурации ядра, как **append**, **ramdisk**, **read-only**, **read-write**, **root** и **vga** могут быть установлены в разделе общих параметров. Они используются по умолчанию, если дополнительно не указаны соответствующие образы загрузки в конфигурационных секциях.

СЕКЦИИ ДЛЯ КАЖДОГО РАЗДЕЛА

Секция для каждого раздела начинается со строки

image=*pathname*

(чтобы указать на файл или устройство, содержащее образ загрузки ядра Linux), либо со строки

other=*pathname*

для указания произвольной системы загрузки.
В первом случае, если строка **image** определяет загрузку с устройства, надо указать область секторов, которые должны отображаться, используя
range=*начало-конец*

Во втором случае (загрузка другой системы) можно указать еще три параметра:

loader=*последовательный_загрузчик***(**

Данный параметр определяет последовательный загрузчик, который должен использоваться. По умолчанию используется файл */boot/chain.b*. Последовательный загрузчик должен быть обязательно определен, если Вы хотите проделывать загрузку с устройства, отличного от первого жесткого диска или первого флоппи-дисковода.

table=*устройство*

Данный параметр определяет устройство, которое содержит таблицу разделов. Менеджер загрузки не передаст информацию об имеющихся разделах загружаемой операционной системе, если эта переменная пропущена. (Некоторые операционные системы имеют другие способы для определения раздела, с которого они были запущены. Например, MS-DOS обычно сохраняет геометрию загрузочного диска или раздела в своем загрузочном секторе). Заметим, что файл /sbin/lilo должен быть повторно запущен, если параметром `table' были изменены отображаемые ссылки в таблице разделов.

unsafe Не обращаться к загрузочному сектору во время создания карты (\map). Данный параметр снимает некоторые проверки, включая проверку таблицы разделов. Если загрузочный сектор находится на флоппи-диске фиксированного формата, то

использование параметра UNSAFE исключает необходимость устанавливать считываемый диск в дисковод во время запуска установки отображения. Параметры `unsafe' и `table' являются взаимоисключающими, несовместимыми.

В обоих случаях используются параметры, описанные ниже.

label=*название*
Менеджер загрузки использует имя файла (неполное) со спецификацией каждого образа для обозначения этого образа. С помощью изменения параметра `label' может быть использовано другое имя

alias=*название*
Может быть использовано и дополнительное название одного образа, если определить его псевдоним.

lock (смотрите выше).

optional
Пропустить образ, если он недоступен во время создания карты отображения. Это бывает полезно, когда нужно определять тестовые ядра, которые в действительности не всегда существуют.

password=*пароль*
Защитить данный образ паролем.

restricted
Пароль требуется только для загрузки образа, если в командной строке указаны дополнительные параметры (например, single).

ОПЦИИ ЯДРА

Если загружаемый образ является ядром Linux, то ему можно передать некоторые параметры командной строки для запуска.

append=*строка*
Учитывать данную строку как параметр, передаваемый ядру. Это обычно используется для указания параметров аппаратных средств, которые не могут автоматически определяться, или для которых очень опасно применять разные "пробные" конфигурации. Пример:

```
append = "hd=64,32,202"
```

literal=*строка*
Действует как параметр `append', но при этом "очищает" все другие параметры (например, определения корневого устройства). Так как при использовании параметра `literal' могут быть непреднамеренно удалены крайне важные настройки, этот параметр не должен быть указан в разделе общих параметров.

ramdisk=*размер*
Параметр определяет размер дополнительного виртуального ОЗУ-диска. Значение "ноль" определяет, что ОЗУ-диск вообще не должен быть создан. Если этот параметр пропущен, то используется ОЗУ-диск с размером загружаемого образа.

read-only
Параметр определяет, что корневая файловая система должна быть запущена с атрибутами "только для чтения". Обычно процедура загрузки системы заново запускает корневую файловую систему с атрибутами "чтение и запись" (например, после использования fsck).

read-write
Параметр определяет, что корневая файловая система должна быть запущена с атрибутами "чтение и

запись".

root=*корневое_устройство***(root-device)**

Параметр определяет устройство, которое должно запускаться как корневое. Если использовано особое название **current**, то устанавливаемое корневое устройство представляет собой устройство, к которому в данный момент подключена корневая файловая система. Если была использована переменная 'root' с ключом -r , то соответствующее устройство будет запущено. Если переменная `root' пропущена, то используются настройки корневого устройства в образе ядра. Они устанавливаются во время компиляции с помощью переменной ROOT_DEV в Makefile ядра и могут позднее изменяться программой **rdev(8)**.)

vga=*режим*

Определяет текстовый VGA-режим, который должен использоваться при загрузке. Могут использоваться следующие значения (регистр игнорируется):

normal: установить обычный текстовый режим 80x25.

extended (или **ext**): установить текстовый режим 80x50.

ask: остановиться (при загрузке) и спросить пользователя о его выборе.

<число>: использовать соответствующий текстовый режим. Список возможных режимов можно получить, установив vga=ask и нажав [Enter] при загрузке.

Если данный параметр пропущен, то используются VGA-режимы, содержащиеся в выбранном образе ядра. Они устанавливаются во время компиляции с помощью переменной SVGA_MODE в Makefile ядра и могут позднее изменяться программой **rdev(8)**.

СМ. ТАКЖЕ **lilo(8)**, **rdev(8)**.

К программе lilo прилагается очень подробная документация, и все описанное в этом файле является лишь кратким ее обзором.

28 July 1995 LILO.CONF**(5)**

---

locale

НАЗВАНИЕ locale - описание файла определения локали

ОПИСАНИЕ Файлы определения **locale** содержат в себе всю информацию, которую команда **localedef(1)** требует для преобразования в бинарную базу данных докали. Файлы определения состоят из разделов, каждый из которых подробно описывает некоторые категории локали.

СИНТАКСИС

Файлы определения локали начинаются с заголовка, который может состоять из следующих ключевых слов:

<escape_char>

сопровождается символом, который будет использоваться как символ экранирования всей оставшейся части файла и для обозначения символов, которые должны восприниматься особым образом. По умолчанию это символ обратной косой черты (**\f[R]).**

<comment_char>

сопровождается символом, который будет использоваться как символ комментария для всей оставшейся части файла. По умолчанию это символ номера (#).

Определение локали имеет часть для каждой категории локали. Каждая часть может быть скопирована из другой существующей локали или может быть скопирована и скорректирована по Вашему желанию. Если категория должна

быть скопирована, то в определении должно находиться единственное ключевое слово **copy** , сопровождаемое названием локали, из которой скопирована категория.

LC_CTYPE

Определение для категории **LC_CTYPE** начинается с команды *LC_CTYPE* в первой колонке. Существуют следующие разрешенные ключевые слова:

upper сопровождается списком символов в верхнем регистре. Символы от **A** до **Z** включаются в класс автоматически. Не разрешается использовать в списке символы классов: **cntrl**, **digit**, **punct** или **space**.

lower сопровождается списком символов в нижнем регистре. Символы от **a** до **z** включаются в класс автоматически. Не разрешается использовать в списке символы классов: **cntrl**, **digit**, **punct** или **space**.

alpha сопровождается списком символов. Все символы, указанные в классах **upper** или **lower**, автоматически включаются в данный класс. Не разрешается использовать в списке символы классов: **cntrl**, **digit**, **punct** или **space**.

digit сопровождается списком символов, классифицированных как цифры. Разрешается использовать в списке только цифры от **0** до **9** (они включаются в этот класс автоматически);

space сопровождается списком символов, классифицированных как пустые символы. Не разрешается использовать в списке символы классов: **upper**, **lower**, **alpha**, **digit**, **graph** или **xdigit**. Символы **<space>**, **<form-feed>**, **<newline>**, **<carriage-return>**, **<tab>** и **<vertical-tab>** (включаются в этот класс автоматически)

cntrl сопровождается списком символов, классифицированных как управляющие символы. Не разрешается использовать в списке символы из классов: **upper**, **lower**, **alpha**, **digit**, **punct**, **graph**, **print** или **xdigit**.

punct сопровождается списком символов, классифицированных как символы пунктуации. Не разрешается использовать в списке символы классов: **upper**, **lower**, **alpha**, **digit**, **cntrl**, **xdigit** или символ **<space>**.

graph сопровождается списком символов для печати, исключая символ **<space>**. Символы из классов **upper**, **lower**, **alpha**, **digit**, **xdigit** и **punct** (включаются в этот класс автоматически). Не разрешается использовать в списке символы из класса **cntrl**;

print сопровождается списком печатаемых символов, включая символ **<space>**. Символы из классов **upper**, **lower**, **alpha**, **digit**, **xdigit**, **punct** и символ **<space>** (включаются в этот класс автоматически). Не разрешается использовать в списке символы из класса **cntrl**.

xdigit сопровождается списком символов, классифицированных как шестнадцатеричные числа. Далее должны быть указаны десятичные числа, сопровождаемые одним или несколькими списками из шести символов в возрастающем порядке. Символы от **0** до **9**, от **a** до **f**, от **A** до **F** (включаются в этот список автоматически).

blank сопровождается списком символов, классифицированных как **blank**. Символы **<space>** и **<tab>** (включаются в этот класс автоматически).

toupper
сопровождается списком преобразований символов нижнего регистра в символы верхнего. Каждое преобразование выполняется для пары символов в нижнем и верхнем регистре, разделенных , и закрытых круглыми скобками. Элементы списка разделяются точками с запятой.

tolower
сопровождается списком преобразований символов верхнего регистра в символы нижнего. Если команда tolower не задана, то по умолчанию используется список, противоположный списку команды toupper.

Определение для категории **LC_CTYPE** заканчивается строкой *END LC_CYTPE.*

LC_COLLATE

Категория **LC_COLLATE** определяет правила сортировки символов. В libc (из-за ее несовершенства) не все параметры POSIX могут быть выполнимы. Определение начинается с команды **LC_COLLATE** в первой колонке. Для выполнения определения существуют следующие допустимые ключевые слова:

collating-element

collating-symbol

Определение упорядочивания начинается с команды

order_start
,

сопровождаемой списком команд: **forward**, **backward** или **position**. Описание упорядочивания состоит из строк, описывающих метод упорядочивания. Описание заканчивается командой

order_end.

Для более подробной информации смотрите исходные варианты в **/usr/lib/nls/src** , особенно, примеры **POSIX: Примеры (Example)** и **Примеры2 (Example2).**
Определение для категории **LC_COLLATE** заканчивается строкой *END LC_COLLATE.*

LC_MONETARY

Определение начинается с команды **LC_MONETARY** в первой колонке. Это выполняется с помощью следующих разрешенных ключевых слов:

int_curr_symbol
(сопровождаются символом международной валюты). Это должна быть строка из четырех символов, в которой указан символ международной валюты согласно стандарту ISO 4217: три символа с последующим разделителем.

currency_symbol
сопровождается локальным валютным символом.

mon_decimal_point
сопровождается строкой, которая будет использована в качестве разделителя (десятичной точки) при выводе денежных сумм.

mon_thousands_sep
сопровождается строкой, которая будет использована в качестве разделителя тысяч при выводе денежных

сумм.

mon_grouping
сопровождается строкой, которая описывает форматирование числовых количеств.

positive_sign
сопровождается строкой, которая используется для указания положительного знака денежных сумм.

negative_sign
сопровождается строкой, которая используется для указания отрицательного знака денежных сумм.

int_frac_digits
сопровождается количеством дробных цифр, которые должны использоваться при форматировании с **int_curr_symbol**.

frac_digits
сопровождается количеством дробных цифр, которые должны использоваться при форматировании с **currency_symbol**.

p_cs_precedes
сопровождается целым числом, установленное значение которого равно **1** , если символ *currency_symbol* или *int_curr_symbol* должен предшествовать форматированному денежному количеству; или равно **0** , если символ следует за его величиной.

p_sep_by_space
сопровождается целым числом.

0 означает, что между символом и величиной не должны пропечатываться никакие знаки.

1 означает, что между символом и величиной должен печататься пробел.

2 означает, что пробел должен печататься между символом и знаковой строкой, если они смежны.

n_cs_precedes

0 : символ следует за величиной

1: символ предшествует величине

n_sep_by_space
Устанавливаемое целое число равно **нулю** , если никакие пробелы не отделяют *currency_symbol* или *int_curr_symbol* от величины (в случае с отрицательным денежным значением); оно равно **единице**, если пробел отделяет символ от величины; и равно **двум**, если пробел отделяет символ от знаковой строки (в случае их смежности).

p_sign_posn

0 Круглые скобки включают в себя количество и *currency_symbol* или *int_curr_symbol.*

1 Знаковая строка предшествует количеству и *currency_symbol* или *int_curr_symbol.*

2 Знаковая строка следует за количеством и

currency_symbol или *int_curr_symbol.*

3 Знаковая строка предшествует количеству и *currency_symbol* или *int_curr_symbol.*

4 Знаковая строка следует за *currency_symbol* или *int_curr_symbol.*

n_sign_posn

0 Круглые скобки включают в себя количество и *currency_symbol* или *int_curr_symbol.*

1 Знаковая строка предшествует количеству и *currency_symbol* или *int_curr_symbol.*

2 Знаковая строка следует за количеством и *currency_symbol* или *int_curr_symbol.*

3 Знаковая строка предшествует *currency_symbol* или *int_curr_symbol.*

4 Знаковая строка следует за *currency_symbol* или *int_curr_symbol.*

Определение для категории **LC_MONETARY** заканчивается строкой *END LC_MONETARY.*

LC_NUMERIC

Определение начинается с команды **LC_NUMERIC** в первой колонке. Существуют следующие разрешенные ключевые слова:

decimal_point
сопровождается строкой, которая будет использована в качестве разделителя (десятичной точки) при форматировании числовых количеств.

thousands_sep
сопровождается строкой, которая будет использована в качестве разделителя тысяч при форматировании числовых количеств.

grouping
сопровождается строкой, которая описывает форматирование числовых количеств.

Определение для категории **LC_NUMERIC** заканчивается строкой *END LC_NUMERIC.*

LC_TIME

Определение начинается с команды **LC_TIME** в первой колонке. Существуют следующие разрешенные ключевые слова:

abday сопровождается списком кратких названий дней недели. Список начинается с Sunday или его перевода (например, воскресенье).

day сопровождается списком названий дней недели. Список начинается с воскресенья.

abmon сопровождается списком кратких названий месяцев.

mon сопровождается списком названий месяцев.

am_pm Соответственное представление строк **am** и **pm**.

d_t_fmt Дата и формат времени соответственно.

d_fmt Формат даты.

t_fmt Формат времени.

t_fmt_ampm 12-часовой формат времени.

Определение для категории **LC_TIME** заканчивается строкой *END LC_TIME.*

LC_MESSAGES

Определение начинается с команды **LC_MESSAGES** в первой колонке. Существуют следующие разрешенные ключевые слова:

yesexpr
(сопровождается регулярным выражением, которое описывает возможные ответы "да");

noexpr (сопровождается регулярным выражением, которое описывает возможные ответы "нет");

Определение для категории **LC_MESSAGES** заканчивается строкой *END LC_MESSAGES.* Смотрите стандарт POSIX.2 для более подробной информации.

ФАЙЛЫ

/usr/lib/locale/ - база данных для текущих настроек локали в этой категории /usr/lib/nls/charmap/* - charmap-файлы */

НАЙДЕННЫЕ ОШИБКИ

Эта страница руководства находится в стадии разработки и не содержит всей необходимой информации.

СООТВЕТСТВИЕ POSIX.2

СМ. ТАКЖЕ setlocale(3), localeconv(3), charmap(5), locale(1), localedef(1)

National Language Support 09 Nov 1994 **locale(5)**

MOTD

НАЗВАНИЕ motd - сообщение дня

ОПИСАНИЕ

Содержимое файла **/etc/motd** отображается программой **login(1)** после того, как осуществлен успешный вход в систему и перед запуском командного интерпретатора.

Сокращение "motd" расшифровывается как "message of the day" (сообщение дня), и данный файл традиционно использовался именно для этой цели (он требует гораздо меньше дискового пространства, чем почтовое сообщение, посылаемое каждому пользователю).

ФАЙЛЫ /etc/motd

СМ. ТАКЖЕ **login(1), issue(5)**

Linux December 29 1992 MOTD**(5)**

NOLOGIN

НАЗВАНИЕ nologin - не позволяет войти в систему пользователю без прав root

ОПИСАНИЕ

Если файл **/etc/nologin** существует, то вход в систему (**login(1)**) может произвести только пользователь с правами "root". Любым другим пользователям будет отказано в доступе, а на экран выведено содержимое этого файла.

ФАЙЛЫ /etc/nologin

СМ. ТАКЖЕ **login(1), shutdown(8)**

Linux December 29 1992 NOLOGIN**(5)**

---

nscd.conf

НАЗВАНИЕ /etc/nscd.conf - онфигурационный файл демона сервиса кэширования имен

ОПИСАНИЕ

Файл **/etc/nscd.conf** считывается **nscd(8)** в начале работы. Каждая строка определяет либо атрибут и значение, либо атрибут, сервис и значение. Поля разделяются либо ПРОБЕЛАМИ либо символами табуляции (TAB). Символ `#' (знак решетки) определяет строку комментариев; все, что располагается за этим символом, не обрабатывается

программой nscd.

Корректными сервисами будут passwd, group или hosts.

logfile *имя_файла_отчета*
Определяет имя файла, в который будет выдаваться информация об отладке.

debug-level *значение*
Определяет уровень отладки.

threads *число*
Определяет количество запускаемых трэдов, ожидающих запросов. 5 трэдов могут быть созданы всегда.

server-user *пользователь*
Если указан этот параметр, то nscd попытается запуститься как указанный пользователь, а не как root. Если используется параметр раздельного кэширования для разных пользователей (параметр -S), то данный параметр игнорируется.

enable-cache *сервис <yes|no>*
Включает или отключает кэширование указанного *сервиса*.

positive-time-to-live *сервис значение*
Устанавливает время жизни (TTL - time-to-live) для положительных элементов (успешных запросов) в указанном кэше для *сервиса*. *Значение* указывается в секундах. Большие значения увеличивают частоту использования кэша и уменьшают время ответа, но при этом увеличивают проблемы с соответствием содержания кэша.

negative-time-to-live *сервис значение*
Устанавливает время жизни (TTL - time-to-live) для отрицательных элементов (безуспешных запросов) в указанном кэше для *сервиса*. *Значение* указывается в секундах. Может привести к значительному увеличению производительности, если существуют несколько файлов, идентификаторы владельцев которых (ID) не находятся в системной базе данных (,например выполнение "untar" для исходников ядра linux под пользователем root); значение должно быть небольшим для уменьшения проблем соответствия содержания кэша.

suggested-size *сервис значение*
Это размер внутренней таблицы кэша, *значение* должно оставаться простым числом для оптимальной эффективности.

check-files *сервис <yes|no>*
Включает или отключает проверку на принадлежность файла к указанному *сервису* для изменений. Тут файлами могут быть */etc/passwd*, */etc/group* и */etc/hosts*.

СМ. ТАКЖЕ **nscd(8)**

АВТОРЫ **nscd** было создано Thorsten Kukuk и Ulrich Drepper.

GNU C Library 1999-10 **nscd.conf(5)**

NSSWITCH.CONF

НАЗВАНИЕ nsswitch.conf - конфигурационный файл системных баз данных и переключателя сервисов имен

ОПИСАНИЕ

Многие функции в бибилиотеках C должны быть настроены так, чтобы без сбоев работать в локальной среде. Раньше это делалось с помощью соответствующих файлов (например, `/etc/passwd'), но потом появились другие сервисы имен (такие, как Network Information Service (NIS) и Domain

Name Service (DNS)), ставшие популярными и включенные в библиотеки C.
В Linux libc5 есть поддержка NYS, а в glibc 2.x (libc.so.6) содержится более простое и эффективное решение этой проблемы. Оно было реализовано на основе метода, использованного Sun Microsystems в библиотеке C Solaris 2. Мы последовали этому примеру и назвали эту схему "Name Service Switch" (NSS). Источники "баз данных" и порядок их просмотра задаются в файле **/etc/nsswitch.conf**.
В NSS поддерживаются следующие базы данных:

aliases
(почтовые алиасы (псевдонимы), используемые **sendmail(8),которые** в настоящее время игнорируются);

ethers (ethernet-адреса);

group (группы пользователей, с которыми работает функция **getgrent(3)**);

hosts (имена и адреса машин, используемые функцией **gethostbyname(3)** и ей подобными);

netgroup
(общесетевой список машин и пользователей, используемый для определения прав доступа к системе. Библиотеки C до glibc 2.1 поддерживали эти группы только для NIS);

network
(имена и адреса сетей, используемые функцией **getnetent(3)**);

passwd (пароли пользователей, обрабатываемые функцией **getpwent(3)**);

protocols
(сетевые протоколы, используемые функцией **getprotoent(3)**);

publickey
(открытые и закрытые ключи для Secure_RPC, используемые NFS и NIS+);

rpc (имена и номера удаленных вызовов процедур RPC, используемые функцией **getrpcbyname(3)** и ей подобными);

services
(сетевые службы, работающие с функцией **getservent(3)**);

shadow (скрытые (shadow) пароли пользователей, обрабатываемые функцией **getspnam(3)**).

Файл **/etc/nsswitch.conf** может выглядеть примерно так (такое содержимое используется и по умолчанию, если файл **/etc/nsswitch.conf** отсутствует):

```
passwd:         compat
group:          compat
shadow:         compat

hosts:          dns [!UNAVAIL=return] files
networks:       nis [NOTFOUND=return] files
ethers:         nis [NOTFOUND=return] files
protocols:      nis [NOTFOUND=return] files
rpc:            nis [NOTFOUND=return] files
services:       nis [NOTFOUND=return] files
```

Первая колонка - это название базы данных. Остальная часть строки описывает процесс поиска данных. Этот процесс можно задать для каждой базы данных в отдельности.
Описание конфигурации каждой базы данных может содержать два типа элементов:

* Название сервиса, например, `files', `db' или `nis'.
* Ответ на результат работы сервиса, например, `[NOTFOUND=return]'.

В libc5 с NYS разрешены следующие названия сервисов: `files', `nis' и `nisplus'. Для имен машин можно использовать сервис `dns', для passwd и group - `compat' (но не для shadow).

В glibc, у Вас должен быть файл **/lib/libnss_СЕРВИС.so.***X* для каждого СЕРВИСА, с которым Вы работаете. Вы увидите `files', `db', `nis' и `nisplus'. Для имен машин можно использовать сервис `dns', для passwd, group и shadow - `compat'. Эти сервисы не будут использоваться в libc5 с NYS. Номер версии *X* равен 1 в glibc 2.0 и 2 в glibc 2.1.

Второй вариант элемента в описании конфигурации позволяет более гибко контролировать процесс поиска данных. Элементы действий помещаются между двумя названиями сервисов и заключаются в квадратные скобки. Общая форма элемента действий:

`[' ( `!'? СТАТУС `=' ДЕЙСТВИЕ )+ `]'

where

СТАТУС => success | notfound | unavail | tryagain
ДЕЙСТВИЕ => return | continue

Регистр написания ключевых слов неважен. СТАТУС - это результат вызова соответствующей функции заданного сервиса:

success
: Ошибки не произошло, и возвращено искомое значение. Стандартное действие для этого статуса - `return'.

notfound
: Поиск проведен успешно, но искомое значение не найдено. Стандартное действие - `continue'.

unavail
: Сервис недоступен. Это означает, что файла не существует или, в случае DNS, сервер недоступен или не позволяет отправлять запросы. Стандартное действие - `continue'.

tryagain
: Сервис временно недоступен. Это означает, что файл заблокирован или сервер не может создать соединение. Стандартное действие - `continue'.

Работа с синтаксисом +/- (режим compat)

libc5 в Linux без NYS не имеет NSS, но имеет простую встроенную форму контроля пользовательской работы. В файле **/etc/passwd** могут находиться строки в форме +пользователь или +@сетевая_группа (включить заданного пользователя в карту NIS passwd), -user или -@netgroup (исключить заданного пользователя) и + (включить всех пользователей, кроме исключаемых из карты NIS passwd). Многие помещают один + в конце файла **/etc/passwd**; в этом случае NSS предлагает более быструю альтернативу (`passwd: files nis'), которой не нужен + в файлах **/etc/passwd**, **/etc/group** и **/etc/shadow**. Если этого недостаточно, то сервис `compat' NSS полностью поддерживает семантику +/-. По умолчанию источник равен `nis', но это можно изменить, задав `nisplus' в качестве сервиса псевдобаз данных **passwd_compat**, **group_compat** и **shadow_compat**. Псевдобазы доступны только в GNU-библиотеке C.

ФАЙЛЫ Сервис под именем СЕРВИС включен в разделяемую библиотеку под названием **libnss_СЕРВИС.so.***X;* эта библиотека находится в каталоге */lib*.

/etc/nsswitch.conf	файл конфигурации
/lib/libnss_compat.so.*X*	сервис `compat' в glibc2
/lib/libnss_db.so.*X*	сервис `db' в glibc2
/lib/libnss_dns.so.*X*	сервис `dns' в glibc2
/lib/libnss_files.so.*X*	сервис `files' в glibc2
/lib/libnss_hesiod.so.*X*	сервис `hesiod' в glibc2
/lib/libnss_nis.so.*X*	сервис `nis' в glibc2
/lib/libnss_nisplus.so.2	сервис `nisplus' в glibc 2.1

ЗАМЕЧАНИЯ Каждому процессу, использующему **nsswitch.conf**, весь файл читается один раз; если файл после этого изменить, то процесс будет работать со старой конфигурацией.

В Solaris программы, использующие NSS, нельзя собирать статически. В Linux этой проблемы нет.

Linux 17 January 1999 NSSWITCH.CONF**(5)**

PASSWD

НАЗВАНИЕ passwd - файл паролей

ОПИСАНИЕ Passwd -- это текстовый файл, содержащий список учетных записей пользователей системы, каждая из которых содержит такую информацию, как: идентификатор пользователя, идентификатор группы, домашний каталог, командную оболочку и т.д. Часто в этом файле содержатся зашифрованные пароли каждого пользователя. Этот файл должен быть доступен всем для чтения (многие утилиты, например, **ls(1)** используют этот файл для сопоставления идентификаторов и имен пользователей), но запись информации в него разрешена только суперпользователю.

Раньше не было никаких проблем с доступностью этого файла для чтения. Каждый мог прочитать зашифрованный пароль, но электроника была слишком медленной, чтобы раскрыть его, и, более того, основным принципом было дружелюбное отношение пользователей друг к другу. Сейчас многие используют ту или иную версию комплекта теневых паролей, где */etc/passwd* обозначает звездочками зашифрованный пароль, а сами эти пароли хранятся в */etc/shadow*, который доступен для чтения только суперпользователю.

Независимо от того, используются ли теневые пароли, многие системные администраторы используют звездочку в качестве зашифрованного пароля, чтобы убедиться, что какой-либо пользователь не сможет войти в систему под своим паролем (см. ниже главу ЗАМЕЧАНИЯ).

Если Вы создаете новую учетную запись, сначала поместите в поле пароля знак звездочки, затем используйте **passwd(1)**, чтобы задать настоящий пароль.

Каждая учетная запись находится в одной строке, а каждая строка имеет формат

account:password:UID:GID:GECOS:directory:shell

Вот описание полей:

account	имя пользователя в системе. Оно не должно содержать заглавных букв;
password	зашифрованный пароль пользователя или "звездочка";
UID	цифровой идентификатор пользователя;
GID	цифровой идентификатор основной группы пользователя;
GECOS	Это поле не является обязательным и используется только в информационных целях. Обычно здесь указывается полное

имя пользователя. GECOS означает General Electric Comprehensive Operating System, которая была переименована в GCOS, когда подразделение больших систем компании General Electric было продано компании Honeywell. Денис Ричи говорил: "Иногда мы посылали файл для распечатки или набор задач для пакетной обработки в машину с GCOS. В поле GCOS, в файле паролей, пряталась информация для $IDENTcard. Не очень умно."

directory - домашний каталог пользователя ($HOME).

shell - программа, которая выполняется при входе в систему (если здесь ничего не указано, используется **/bin/sh**). Если здесь указан несуществующий исполняемый файл, пользователь не сможет войти в систему с помощью **login(1)**.

ЗАМЕЧАНИЕ

Если вы хотите создавать группы пользователей, то их GID должны быть одинаковы, а в файле */etc/group* должна быть запись об этой группе, или же эта группа не будет существовать.

Если в поле с зашифрованным паролем находится "звездочка", пользователь не сможет войти в систему с помощью **login(1)**, но сможет сделать это с помощью **rlogin(1)**, а также выполнять существующие процессы и создавать новые с помощью **rsh(1)**, **cron(1)**, **at(1)** или почтовых фильтров (и т.п). Попытка заблокировать учетную запись, изменяя поле shell, приведет к тем же результатам и дополнительно позволит использовать **su(1)**.

ФАЙЛЫ */etc/passwd*

СМ. ТАКЖЕ **passwd(1)**, **login(1)**, **su(1)**, **group(5)**, **shadow(5)**

January 5, 1998 PASSWD**(5)**

PROC

НАЗВАНИЕ proc - псевдофайловая система с информацией о процессах

ОПИСАНИЕ /proc - это псевдофайловая система, представляющая собой интерфейс структур данных ядра, более простой по своей сути, чем процесс чтения и интерпретации /dev/kmem. Многие из этих "файлов" доступны только для чтения, но некоторые файлы позволяют изменять значения некоторых переменных ядра.

Ниже приведено краткое описание псевдофайловой системы /proc.

[номер]

Для каждого процесса существует каталог, имя которого равно номеру процесса. Каждый из них содержит следующие файлы и каталоги:

cmdline

Этот файл содержит полную командную строку запуска процесса до тех пор, пока процесс не будет "выгружен" или не станет "зомби". В последних двух случаях любое чтение информации из этого файла вернет 0 байтов информации. Содержимое файла оканчивается нулем, но не символом новой строки.

cwd Это символьная ссылка на текущий рабочий каталог процесса. Например, для того, чтобы узнать рабочий каталог процесса под номером 20, Вы можете ввести следующее:

```
cd /proc/20/cwd; /bin/pwd
```

Заметьте, что команда pwd часто бывает встроена в shell и может работать неправильно. В bash можно использовать pwd -P.

environ

Этот файл содержит окружение процесса. Записи отделяются друг от друга символами с кодом 0; 0 может также стоять в конце окружения. Таким образом, чтобы узнать содержимое окружения процесса 1, надо выполнить следующее:

```
(cat /proc/1/environ; echo) | tr "\000" "\n"
```

Причины, по которым это может быть необходимо, см. в **lilo(8)**.

exe В Linux 2.2 и 2.4 *exe* является символьной ссылкой, содержащий действительный имя пути исполняемой команды. Символьная ссылка *exe* может быть нормально обработана - попытка открыть *exe* приведет к открытию исполняемого файла. Можно даже для запуска копии процесса с определенным номером ввести команду */proc/[номер]/exe*. В Linux 2.0 и более ранних версиях *exe* является символьной ссылкой, указывающей на двоичный файл, который был запущен. Вызов **readlink(2)** для этого специального файла I. exe в Linux 2.0 возвращает строку следующего формата:

```
[device]:inode
```

Например, [0301]:1502 означало inode 1502 устройства с основным числом 03 (дисководы IDE, MFM и т.п.) и второстепенным 01 (первый раздел в первом устройстве).
Для поиска файла можно использовать команду **find(1)** с опцией -inum.

fd - это подкаталог, содержащий одну запись для каждого открытого процессом файла. Имя записи соответствует описателю файла, а сама запись является символьной ссылкой на открытый процессом файл (аналогично записи exe). Таким образом, 0 - это стандартный ввод, 1 - стандартный вывод, 2 - стандартный вывод ошибок и т.п. Программы, которые в качестве входного параметра рассматривают имя файла (но не понимают стандартный ввод) или записывают результат работы в файл, но не на стандартный вывод, могут быть "обмануты" этой системой. Предполагая, что флаг -i описывает входной файл, а -o - результирующий, можно выполнить следующее:
foobar -i /proc/self/fd/0 -o /proc/self/fd/1 ...
,- и у Вас появится работающий фильтр. Заметьте, что это не сработает в том случае, если программа ищет входные файлы, потому что поиск в каталоге fd не производится. /proc/self/fd/N - это примерно то же самое, что и /dev/fd/N в некоторых UNIX и UNIX-подобных системах. На самом деле, многие скрипты MAKEDEV в Linux создают символьные ссылки /dev/fd на /proc/self/fd.

maps Этот файл содержит данные о текущих сегментах памяти процесса и права доступа к ним. Его формат следующий:

```
address           perms offset  dev   inode      pathname
08048000-08056000 r-xp 00000000 03:0c 64593      /usr/sbin/gpm
08056000-08058000 rw-p 0000d000 03:0c 64593      /usr/sbin/gpm
08058000-0805b000 rwxp 00000000 00:00 0
40000000-40013000 r-xp 00000000 03:0c 4165       /lib/ld-2.2.4.so
40013000-40015000 rw-p 00012000 03:0c 4165       /lib/ld-2.2.4.so
4001f000-40135000 r-xp 00000000 03:0c 45494      /lib/libc-2.2.4.so
40135000-4013e000 rw-p 00115000 03:0c 45494      /lib/libc-2.2.4.so
4013e000-40142000 rw-p 00000000 00:00 0
bffff000-c0000000 rwxp 00000000 00:00 0
```

(где address - это адресное пространство, занятое процессом, perms - набор следующих прав:

r = чтение
w = запись
x = исполнение
s = разделяемый
p = частный ("копирование-при-записи")

,- offset -это смещение в файле (или где-то еще), dev - устройство (старшее:младшее числа), а inode - это inode этого устройства). 0 означает, что этой области памяти не соответствует ни один файл, как будет в случае с bss.

В Linux 2.0 поле, содержащее путь к файлу, отсутствует.

mem Через файл *mem* можно получить доступ к страницам памяти процесса через **open(2)**, **read(2)** и **fseek(3)**.

root Unix и, в частности, Linux поддерживают идею попроцессных корневых файловых систем, установленных при помощи системного вызова **chroot(2)**. Root указывает на корень файловой системы и работает так же, как exe, fd/* и т.п.

stat Информация о процессе. Обычно она используется командой **ps(1)** и определена в */usr/src/linux/fs/proc/array.c*.
Ниже приведен список полей с указанием их форматов в соответствии с **scanf(3)**:

pid %d Идентификатор процесса;

comm %s
Имя исполняемого файла процесса в скобках. Это имя доступно независимо от того, загружен ли исполняемый файл в область подкачки или нет;

state %c
Один символ из строки "RSDZT"; где R означает, что процесс запущен, S означает, что он находится в прерываемом ожидании, D - в непрерываемом ожидании или загружается, Z - зомби, и T означает, что процесс отлаживается или остановлен сигналом, а W - что процесс ожидает подгрузки страницы.

ppid %d
Идентификатор родительского

процесса.

pgrp %d
Идентификатор группы процессов процесса.

session %d
Идентификатор сессии процесса

tty_nr %d
Терминал (tty), используемый процессом.

tpgid %d
Идентификатор группы процессов процесса, владеющего tty, к которому подключен данный процесс.

flags %lu
Флаги процесса. Математический бит - это десятичное 4, а бит трассировки - десятичное 10.

minflt %lu
Количество некритических страничных ошибок процесса, не требующих подгрузки страницы с диска.

cminflt %lu
Количество некритических страничных ошибок процесса и его дочерних процессов.

majflt %lu
Количество критических ошибок процесса, приведших к чтению страницы памяти с диска.

cmajflt %lu
Количество критических ошибок процесса и его дочерних процессов.

utime %lu
Количество тиков (jiffies) времени, отведенных процессу в пользовательском режиме (user mode).

stime %lu
Количество тиков (jiffies) времени, отведенных процессу в режиме ядра (kernel mode).

cutime %ld
Количество тиков (jiffies) времени, отведенных процессу и его дочерним процессам в пользовательском режиме.

cstime %ld
Количество тиков (jiffies) времени, отведенных процессу и его дочерним процессам в режиме ядра.

priority %ld
Стандартное значение nice, увеличенное на 15. Это значение не может быть отрицательным в ядре.
nice %ld Значение nice от 19 (наименьший) до -19 (наибольший приоритет).
0 %ld Это значение является жестко прошитым 0 вместо убранного поля.

itrealvalue %ld
Время (в тиках) до отправки процессу следующего сигнала SIGALRM, связанного с таймером.

starttime %lu
Время запуска процесса в тиках с

момента загрузки системы.

vsize %lu
Размер виртаульной памяти в байтах.

rss %ld
Resident Set Size: количество страниц процесса, находящихся в физической памяти (минус 3 для административных целей). Это лишь те страницы, которые связаны с областями программы, данными или стеком. К ним не относятся страницы в режиме "загрузка при чтении" или выгруженные в область подкачки.

rlim %lu
Текущее ограничение процесса rss в байтах (обычно 4 294 967 295).

startcode %lu
Начальный адрес, с которого начинается исполняемая программа.

endcode %lu
Конечный адрес, которым заканчивается исполняемая программа.

startstack %lu
Адрес начала стека.

kstkesp %lu
Текущее значение esp (указателя стека) в соответствие со страницей процессного стека в ядре.

kstkeip %lu
Текущее значение EIP (указателя выполняемых инструкций).

signal %lu
Маска отложенных сигналов (обычно 0).

blocked %lu
Маска блокируемых сигналов (обычно 0, 2 для shell).

sigignore %lu
Маска игнорируемых сигналов.

sigcatch %lu
Маска "перехватываемых" сигналов.

wchan %lu
"Канал", в котором ожидается процесс. На самом деле, это адрес системного вызова, и его можно найти в списке вызовов, если Вам необходимо текстовое значение. Если у Вас есть обновленный файл /etc/psdatabase, то возможно использовать *ps -l* для того, чтобы увидеть поле WCHAN в действии).

nswap %lu
Количество страниц записанных на устройство подкачки - не поддерживается.

cnswap %lu
Сумма *nswap* для всех дочерних процессов.

exit_signal %d
Сигнал, который будет послан родительскому процессу при завершении.

Номер процессора, на котором эта программа выполнялась в последний раз.

statm Информация о состоянии памяти в страницах. Список колонок:

size	общий размер программы
resident	размер резидентного набора
share	разделенные (shared) страницы
trs	текст (код)
drs	данные/стек
lrs	библиотека
dt	"грязные" страницы

status Значительная часть информации из *stat* и *statm* в формате, удобном для обработки человеком.

bus Содержит подкаталоги для установленных шин.

pci Содержит различные подкаталоги шины и псевдо-файлы, с информацией о шинах pci, установленных устройствах и драйверах устройств. Некоторые из этих файлов имеют двоичный формат.

devices
Информация об устройствах pci. Получить к ней доступ можно с помощью **lspci(8)** и **setpci(8)**.

cmdline
Аргументы, переданные ядру Linux при загрузке. Часто это делается с помощью менеджера загрузки, такого как **lilo(1)**.

cpuinfo
Содержит набор параметров CPU и системной архитектуры (отдельный список для каждой из них). Общими являются записи: *processor*, указывающая на номер процессора, и *bogomips* - системная константа, подсчитываемая при инициализации ядра. Мультипроцессорные машины содержат информацию для каждого процессора.

devices
Список главных чисел и групп устройств. Его можно использовать при работе со скриптами MAKEDEV для того, чтобы каталог /dev подходил ядру.

dma Список используемых зарегистрированных *ISA* DMA (прямой доступ к памяти) каналов.

driver Пустой подкаталог.

execdomains
Список execution domains (ABI personalities).

fb Информация о фрейм-буфере, доступная, если при компиляции ядра был определен CONFIG_FB.

filesystems
Список файловых систем, поддерживаемых ядром. Используется **mount(1)** для сортировки типов файловых систем, если параметр не задан.

ide Существует на системах, имеющих шину ide. Для каждого канала ide и подключенного устройства имеются каталоги. Файлы включают в себя:

cache	размер буфера в Кб
capacity	количество секторов
driver	версия драйвера
geometry	физическая и логическая геометрия
identify	в шестнадцатиричном
media	тип носителя
model	номер модели производителя
settings	настройки дисководанастройки дисковода

smart_thresholds в шестнадцатиричном
smart_values в шестнадцатиричном

Доступ к этой информации в удобном виде предоставляет утилита **hdparm(8)**.

interrupts
Счетчики количества прерываний IRQ в архитектуре i386. Осуществляют очень простое форматирование, соответствующее ASCII.

iomem Карта памяти ввода-вывода в Linux 2.4.

ioports
Список текущих зарегистрированных областей портов ввода-вывода.

kcore Этот файл представляет собой физическую память системы и записан в формате файла ELF core. При работе с этим псевдофайлом и неоптимизированным кодом ядра (/usr/src/linux/tools/zSystem) можно использовать GDB для того, чтобы изучать текущее состояние структур данных ядра. Общий размер этого файла равен размеру физической памяти машины плюс 4 Кб.

kmsg Этот файл можно использовать вместо **syslog(2)** для записи сообщений ядра в журнал. Процесс должен иметь права root, и только один процесс может читать этот файл. Этот файл нельзя читать, если запущен процесс syslog, использующий системный вызов **syslog(2)** для работы с сообщениями ядра. Информация из этого файла может быть получена при помощи программы **dmesg(8)**.

ksyms Содержит произведенные ядром экспортированные символьные определения, используемые утилитами **modules**(X) для динамической сборки и запуска загружаемых модулей.

loadavg
Средняя загрузка системы, то есть среднее количество заданий в очереди (статус R) или среднее количество заданий, ожидающих обмена данными с дисками (статус D), ожидающих исполнения за последние 1, 5 и 15 минут. Это числа, совпадающие со средней загрузкой системы, выдаваемой командой **uptime(1)** и ей подобными.

locks Список заблокированных в текущий момент файлов.

malloc Этот файл существует лишь в том случае, если в процессе компиляции ядра был задан параметр CONFIGDEBUGMALLOC.

meminfo
Этот файл используется командой **free(1)** для вывода информации о количестве свободной и использованной памяти (физическая память+память в области подкачки), а также данных о разделяемой памяти и буферах ядра. Формат данных - это результат работы **free(1)**, только объемы представлены в байтах, а не в килобайтах.

modules
Список модулей, загруженных системой. См. также **lsmod(8)**.

mtrr Memory Type Range Registers. Подробности приведены в */usr/src/linux/Documentation/mtrr.txt*.

net Каталог с различными псевдофайлами, касающимися сети, каждый из которых содержит информацию о каком-либо ресурсе сети. Эти файлы содержат различные ASCII-структуры, поэтому их содержимое

можно просматривать при помощи команды cat. Однако, стандартный набор утилит **netstat(8)** предоставляет Вам более простой интерфейс для работы с этими файлами.

arp Этот файл содержит ASCII-совместимую версию таблицы ARP ядра, используемую для распознавания сетевых адресов. Здесь находятся как динамически сформированные, так и предварительно сделанные записи. Эта таблица выглядит так:

```
IP address     HW type  Flags  HW address         Mask  Device
192.168.0.50   0x1      0x2    00:50:BF:25:68:F3  *     eth0
192.168.0.250  0x1      0xc    00:00:00:00:00:00  *     eth0
```

, где 'IP address' - это IPv4-адрес машины, 'HW type' - это тип аппаратного адреса в соответствии с RFC 826. 'Flags' - это внутренние флаги ARP-структуры (определенные в /usr/include/linux/if_arp.h), а 'HW address' - это физическое отражение заданного IP-адреса, если оно известно.

dev Псевдофайл dev содержит информацию о состоянии сетевых устройств. В данном абзаце приведено количество посланных и принятых пакетов, количество ошибок при "столкновении" пакетов и другие статистические данные. Эта информация используется командой **ifconfig(8)** для вывода информации об устройстве. Данные выглядят примерно так:

```
 Inter-|   Receive                                                |  Transmit
  face |bytes    packets errs drop fifo frame compressed multicast|bytes
packets errs drop fifo colls carrier compressed
     lo: 2776770   11307    0    0    0     0          0         0  2776770
11307    0    0    0     0       0          0
   eth0: 1215645    2751    0    0    0     0          0         0  1782404
4324    0    0    0   427       0          0
   ppp0: 1622270    5552    1    0    0     0          0         0   354130
5669    0    0    0     0       0          0
   tap0:    7714      81    0    0    0     0          0         0     7714
81    0    0    0     0       0          0
```

rarp Информация в этом файле того же формата, что и в файле **arp**. Она представляет собой работающую в данный момент базу данных обратного ARP, используемую **rarp(8)** для обратного преобразования адресов. Если RARP не собран в ядре, то этот файл не будет создан.

raw Содержит таблицу RAW-сокетов ядра. Большая часть этой информации необходима только при отладке. Значение 'sl' - это хэш-слот сокета в ядре, это внутренний статус сокета. "tx_queue" и "rx_queue" - это исходящий и входящий поток данных, сообщающий об использовании памяти ядром. Поля "tr", "tm->when" и "rexmits" не используются RAW. Поле uid содержит идентификатор создателя сокета.

snmp Этот файл содержит текстовые данные, необходимые для управления IP, ICMP, TCP и

UDP при помощи snmp-агента.

tcp Содержит таблицу TCP-сокетов ядра. Большая часть этой информации необходима только при отладке. Значение 'sl' - это хэш-слот сокета в ядре, 'local address' - это локальный адрес и номер протокола. "remote address" - это удаленный адрес и номер протокола. "St" - это внутренний статус сокета. "tx_queue" и "rx_queue" - это исходящий и входящий потоки данных, сообщающие об использовании памяти ядром. Поля "tr", "tm->when" и "rexmits" содержат информацию о внутреннем состоянии сокета, необходимую при отладке. Поле uid содержит идентификатор создателя сокета.

udp Содержит таблицу UDP-сокетов ядра. Большая часть этой информации необходима только при отладке. Значение 'sl' - это хэш-слот сокета в ядре, 'local address' - это локальный адрес и номер протокола. "remote address" - это удаленный адрес и номер протокола. "St" - это внутренний статус сокета. "tx_queue" и "rx_queue" - это исходящий и входящий потоки данных, сообщающие об использовании памяти ядром. Поля "tr", "tm->when" и "rexmits" не используются UDP. Поле uid содержит идентификатор создателя сокета. Выглядит эта таблица примерно так:

```
 sl  local_address rem_address   st tx_queue rx_queue tr rexmits  tm->when
uid
  1: 01642C89:0201 0C642C89:03FF 01 00000000:00000001 01:000071BA 00000000 0
  1: 00000000:0801 00000000:0000 0A 00000000:00000000 00:00000000 6F000100 0
  1: 00000000:0201 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0
```

unix Список доменных UNIX-сокетов, существующих в системе, а также их статус. Выглядит эта таблица примерно так:

```
Num RefCount Protocol Flags    Type St Path
 0: 00000002 00000000 00000000 0001 03
 1: 00000001 00000000 00010000 0001 01 /dev/printer
```

где 'Num' - это номер слота в таблице ядра, 'RefCount' - количество пользователей сокета, 'Protocol' - всегда 0, 'Flags' - внутренние флаги ядра, отражающие состояние сокета. Тип всегда равен '1' (доменные datagram-сокеты Unix пока не поддерживаются ядром). 'St' - это внутренний статус сокета, а Path - это установленный путь сокета (если таковой существует).

partitions
Содержит главный и дополнительный номера для каждого раздела, а также количество блоков и имя раздела.

pci Здесь содержится список всех PCI-устройств, обнаруженных при инициализации ядра, а также их названия и параметры.

scsi Каталог с псевдофайлом scsi среднего уровня, а также с каталогом (или каталогами), соответствующим драйверам SCSI низкого уровня. Внутри этих каталогов есть файлы (по одному на каждый SCSI-хост в этой системе), в которых описывается состояние одной из подсистем SCSI. Эти файлы содержат различные ASCII-структуры, и поэтому их содержимое можно просматривать при помощи команды cat. Более того, в некоторые из этих файлов разрешено

записывать информацию, что позволяет изменять настройки соответствующих подсистем или запускать и отключать некоторые функции SCSI.

scsi Этот файл содержит список всех SCSI-устройств, известных ядру. Этот список похож на тот, что выдается на экран при загрузке системы. Файл scsi поддерживает только одну команду, *add-single-device*, позволяющую включить одно из известных Plug&Play SCSI-устройств. Команда **echo 'scsi add-single-device 1 0 5 0' > /proc/scsi/scsi** заставит хост scsi1 просканировать канал SCSI под номером 0 на наличие в нем устройства ID 5 LUN 0. Если уже есть устройство с этим адресом или данный адрес неправилен, то будет возвращено сообщение об ошибке.

имя_драйвера

имя_драйвера может быть NCR53c7xx, aha152x, aha1542, aha1740, aic7xxx, buslogic, eata_dma, eata_pio, fdomain, in2000, pas16, qlogic, scsi_debug, seagate, t128, u15-24f, ultrastore или wd7000. Эти каталоги появляются для всех драйверов, за которыми зарегистрировано хотя бы одно SCSI-устройство. Каждый каталог содержит файлы (по одному на каждый зарегистрированный хост). Каждый файл назван в соответствии с номером хоста, который был выделен им при инициализации. При чтении этих файлов обычно выдаются сообщения о настройках драйвера и хоста, статистика и т.п. Запись информации в эти файлы позволяет производить различные действия над хостами. Например, команды *latency* и *nola-tency* могут запускать и отключать код измерения задержки в драйвере eata_dma. Команды *lockup* и *unlock* контролируют блокировки шины, эмулируемые драйвером scsi_debug.

self Этот каталог соответствует процессу, читающему каталог /proc. Это то же самое, что и подкаталог каталога /proc, названный в соответствии с номером текущего процесса.

slabinfo

Информация о кэшах ядра. Список колонок:

cache-name
num-active-objs
total-objs
object-size
num-active-slabs
total-slabs
num-pages-per-slab

Дополнительная информация приведена в **slabinfo(5)**.

stat Статистика ядра/системы. Зависит от архитектуры. Общие записи включают в себя:

cpu 3357 0 4313 1362393

Количество тиков (1/100 секунды), произведенных системой в обычном пользовательском режиме, в пользовательском режиме с низким приоритетом, в системном режиме и в режиме ожидания соответственно. Последнее значение должно быть равно умноженному на 100 значению 2-ого поля в

псевдофайле uptime.

page 5741 1808
Количество созданных и удаленных страниц памяти.

swap 1 0
Количество страниц памяти, выгруженных в область подкачки и загруженных из нее.

intr 1462898
Количество прерываний, полученных системой с начала загрузки.

disk_io: (2,0):(31,30,5764,1,2) (3,0):...
(major,minor):(noinfo, read_io_ops, blks_read, write_io_ops, blks_written)

ctxt 115315
Количество смен контекста исполнения с начала работы системы.

btime 769041601
Время загрузки системы (в секундах с начала данной эпохи: 1 января 1970 года).

processes 86031
Количество ветвлений процессов с момента загрузки.

swaps Используемые области подкачки. См. также **swapon(8)**.

sys Этот каталог (созданный в ядре версии 1.3.57) содержит файлы и каталоги, соответствующие переменным ядра. Эти переменные могут быть считаны и иногда изменены с помощью файловой системы *proc*, а также при помощи системного вызова **sysctl(2)**. В настоящее время существуют подкаталоги *abi*, *debug*, *dev*, *fs*, *kernel*, *net*, *proc* и *vm*, содержащие дополнительные файлы и подкаталоги.

abi Этот каталог может быть пустым.

debug Этот каталог может быть пустым.

dev Этот каталог может быть пустым.

fs Содержит файлы *dentry-state*, *dir-notify-enable*, *dquot-nr*, *file-max*, *file-nr*, *inode-nr*, *inode-state*, *lease-break-time*, *leases-enable*, *overflowgid* и *overflowuid* назначение которых понятно из их имен.

Файл *file-nr* содержит количество открытых в данный момент файлов (только для чтения). Файл *file-max* содержит максимальное количество файлов, которые могут быть открыты ядром. Если количества 1024 недостаточно, попытайтесь задать следующую команду:
echo 4096 > /proc/sys/kernel/file-max
Файлы *inode-nr* и *inode-max* также содержат текущее и максимальное количество inode.

kernel Содержит файлы *cad_pid*,*cap-bound*,*core_uses_pid*,*ctrl-alt-del* *domain-name*,*hostname*, *modprobe*, *msgmax*, *msgmnb*, *msgmni*, *osrelease*, *ostype*, *overflowgid*, *overflowuid*, *panic*, *printk*, *random*, *rtsig-max*, *rtsig-nr*, *sem*, *shmall*, *shmmax*, *shmmni*, *sysrq*, *tainted*, *threads-max* и *version* назначение которых понятно из их имен. Файлы *ostype*, *osrelease* и *version* содержат

подстроки файла */proc/version*. Файл *panic* осуществляет доступ к переменной *panic_time-out* ядра. Если значение переменной равно 0, то ядро "зависнет" после выдачи сообщения класса panic; в противном случае произойдет перегрузка ядра по истечении данного количества секунд.

sysvipc
Подкаталог, содержащий псевдо-файлы *msg*, *sem* и *shm*. Эти файлы для облегчения их прочтения имеют заголовки и форматирование.

tty Подкаталог, содержащий псевдо-файлы и подкаталоги для драйверов tty и дисциплин линий.

uptime В этом файле содержатся два числа, обозначающие: время работы системы с момента загрузки (в секундах) и время, проведенное системой в ожидании (в секундах).

version
Эта строка идентифицирует версию текущего ядра, например:

Linux version 1.0.9 (quinlan@phaze) #1 Sat May 14 01:51:54 EDT 1994

СМ. ТАКЖЕ

cat(1), find(1), free(1), mount(1), ps(1), tr(1), uptime(1), readlink(2), mmap(2), chroot(2), syslog(2), hier(7), arp(8), dmesg(8), route(8), hdparm(8), ifconfig(8), lsmod(8), lspci(8), netstat(8), netstat(8), procinfo(8) /usr/src/linux/Documentation/filesystems/proc.txt

СООТВЕТСТВИЕ

Данные этой страницы приблизительно соответствуют данным о Linux версии 2.4.17, поэтому страница должна быть обновлена! Последний раз эта страница обновлялась при описании Linux версии 2.4.17.

ПРЕДОСТЕРЕЖЕНИЯ

Заметьте, что многие строки (например, "окружение" и "командная строка") представлены во внутреннем формате, где поля разделяются символами NUL; эти строки можно сделать легко читаемыми, если использовать *od -c* или *tr "\000" "\n"* для их обработки.

НАЙДЕННЫЕ ОШИБКИ

Файловая система */proc* может быть не совсем безопасной для процессов, работающих в среде **chroot(2)**. Например, если файловая система */proc* подключена к иерархии **chroot**, то задание команды **chdir(2)** в каталоге */proc/1/root* приведет к переходу к исходному, корневому каталогу системы. Вообще-то, это больше свойство, чем ошибка, так как Linux до сих пор не поддерживает системный вызов **fchroot(2)**.

2001-12-16 PROC**(5)**

PROTOCOLS

НАЗВАНИЕ protocols - файл, содержащий определения протоколов

ОПИСАНИЕ

В этом текстовом файле описываются различные inter-net-протоколы DARPA, доступные подсистеме TCP/IP. Желательно заново проверить номера протоколов в файлах заголовков ARPA, так как эти номера используются в заголовках ip-пакетов.

Не изменяйте содержимое этого файла, так как это может привести к непредсказуемым последствиям. Конкретные номера протоколов и их имена определяются информационным центром DDN.

Каждая строка этого файла имеет следующий формат:
protocol number aliases ... ,

где поля разделены пробелами или табуляциями. Пустые строки и строки,
начинающиеся с символа (#), игнорируются. Остаток строки после символа (#) также игнорируется.

Описание полей приведено ниже:
protocol (имя протокола, например, ip, tcp или udp); *number* (официальный номер протокола, использующийся в заголовке ip-пакета); *aliases* (необязательный псевдоним (алиас) протокола).
Этот файл может распространяться по сети с помощью служб типа Yellow
Pages/NIS или BIND/Hesiod.

ФАЙЛЫ */etc/protocols* Файл определения протоколов.

СМ. ТАКЖЕ **getprotoent(3)**, руководство по Yellow Pages Service, руководство по BIND/Hesiod Service.

Linux 18 October 1995 PROTOCOLS**(5)**

RESOLVER

НАЗВАНИЕ resolver - файл конфигурации резольвера

СИНТАКСИС /etc/resolv.conf

ОПИСАНИЕ

resolver является комплектом программ в библиотеке C, которые обеспечивают доступ к системе имени доменов в интернет. Файл конфигурации резольвера содержит информацию, которая считывается программами резольвера при первом его запуске процессом. Файл конфигурации разработан таким образом, чтобы его можно было просто читать. Он содержит некоторый список ключевых слов со значениями, которые упрощают восприятие резольвером разных типов информации.
В нормально сконфигурированной системе данный файл вообще не нужен. Единственный сервер имени, которому направляются запросы, расположен в локальной машине; сервер имени определяется по имени хоста, а путь поиска домена создается на основе имени домена.
Изменяемыми параметрами конфигурации являются:

nameserver
(интернет-адрес (в записи с точками) сервера имени, которому должен отправлять запросы резольвер). Всего может быть указано до MAXNS (сейчас это число равно 3-м) серверов имени, по одному на ключевое слово. Если существует несколько серверов, то библиотека резольвера передает им запросы в порядке их перечисления. Если не указано ни одного элемента **nameserver**, то по умолчанию используется сервер имени локальной машины. (Используется такой алгоритм: попытаться запросить сервер имени, а если время запроса истекло, то запросить следующий сервер; и если все перечисленные серверы закончились, то повторить всю эту процедуру столько раз, сколько разрешено системой);

domain Локальное имя домена. Большинство запросов может использовать короткие имена локального домена. Если не указаны элементы **domain**, то домен определяется по имени локального хоста,

возвращаемого *gethostname*(); где часть, относящаяся к домену, рассматривается как все, что расположено после первого символа `.'. Наконец, если имя хоста не содержит доменную часть, то в качестве этой части указывается root-домен.

search Список поиска имени хоста. Список поиска обычно определяется по имени локального домена; по умолчанию он содержит только имя локального домена. Его можно изменить, перечислив необходимые пути поиска доменов после ключевого слова *search*. Пробелы или символы табуляции являются разделителями имен доменов. Большинство запросов резольвера будут проходить по всем перечисленным путям до тех пор, пока не будет найдено нужное совпадение. Заметим, что этот процесс может быть крайне медленным и увеличивать сетевой трафик, если серверы перечисленных доменов не являются локальными; время запроса может истечь, если не будет найдено доступного сервера для одного из доменов.

Список поиска на текущий момент ограничен шестью доменами (256-б символами).

sortlist

Список сортировки позволяет сортировать адреса, возвращенные gethostbyname. Список сортировки определяется парой сетевых масок IP-адреса. Сетевая маска является дополнительной, и устанавливаемое значение ее по умолчанию равно простой маске сети. IP-адреса и дополнительные сетевые пары разделяются косой чертой. Всего может быть указано до 10-и пар.

Например: sortlist 130.155.160.0/255.255.240.0 130.155.0.0

options

Параметр 'options' позволяет менять определенные внутренние переменные резольвера. Синтаксис:

options *option ...* ,

где *option* является одним из следующих:

debug -- устанавливает RES_DEBUG равным _res.options.

ndots:*n* -- устанавливает порог количества точек, которые должны появляться в имени, данном **res_query** (смотрите **resolver(3)**), перед тем, как будет сделан *начальный абсолютный запрос*. По умолчанию *n* равно ``1''; это означает, что если в имени есть точки, то сначала произойдет попытка восприятия этого имени как абсолютного, а потом к нему будут добавлены элементы *списка поиска*.

Команды *domain* и *search* являются взаимно исключающими. Если встречается несколько этих слов, то учитываться будет только самое последнее.

Действие команды *search* системного файла *resolv.conf* может быть отменено на уровне процессов при помощи установки переменной окружения ``LOCALDOMAIN'' в разделенный пробелами список доменов для поиска.

Действие команды *options* системного файла *resolv.conf* может быть скорректировано на уровне процессов с помощью установки переменной окружения ``RES_OPTIONS'' в разделенный пробелами список параметров резольвера, как описано выше в разделе **options**.

Команды и значения должны появляться в конфигурационном файле в виде одной строки, причем команда (например, **name-server**) должна начинать строку. Значения должны следовать

за командами через пробел.

ФАЙЛЫ */etc/resolv.conf*

СМ. ТАКЖЕ **gethostbyname(3), hostname(7), named(8),**
Руководство по операциям сервера имени для BIND.

1993-11-11 RESOLVER**(5)**

RPC

НАЗВАНИЕ rpc - база данных номеров программ rpc

СИНТАКСИС **/etc/rpc**

ОПИСАНИЕ

Файл *rpc* содержит имена rpc-программ, которые можно использовать вместо их номеров. В каждой строке содержится следующая информация:

название сервера rpc-программы
номер rpc-программы
алиасы (псевдонимы)

Элементы могут отделяться друг от друга любым количеством пробелов и/или символов табуляции. Символ ``#'' означает начало комментария; все символы, находящиеся правее него в строке, игнорируются.

Ниже приведен пример файла */etc/rpc* из дистрибутива исходных текстов Sun RPC:

```
portmapper          100000    portmap sunrpc
rstatd              100001    rstat rstat_svc rup perfmeter
rusersd             100002    rusers
nfs                 100003    nfsprog
ypserv              100004    ypprog
mountd              100005    mount showmount
ypbind              100007
walld               100008    rwall shutdown
yppasswdd           100009    yppasswd
etherstatd          100010    etherstat
rquotad             100011    rquotaprog quota rquota
sprayd              100012    spray
3270_mapper         100013
rje_mapper          100014
selection_svc       100015    selnsvc
database_svc        100016
rexd                100017    rex
alis                100018
sched               100019
llockmgr            100020
nlockmgr            100021
x25.inr             100022
statmon             100023
status              100024
bootparam           100026
ypupdated           100028    ypupdate
keyserv             100029    keyserver
tfsd                100037
nsed                100038
nsemntd             100039
```

ФАЙЛЫ */etc/rpc* база данных номеров программ rpc

СМ. ТАКЖЕ **getrpcent(3)**

26 September 1985 RPC**(5)**

SECURETTY

НАЗВАНИЕ securetty - выводит список устройств, с помощью которых может заходить в программу пользователь root.

ОПИСАНИЕ

/etc/securetty используется программой **login(1);** в этом файле находятся имена устройств текстового вывода (tty) (по одному названию устройства в строке без */dev/*), с помощью которых суперпользователь может заходить в систему.

ФАЙЛЫ /etc/securetty

СМ. ТАКЖЕ **login(1)**

Linux December 29 1992 SECURETTY**(5)**

SERVICES

НАЗВАНИЕ services - список служб сети интернет

ОПИСАНИЕ

services является простым ASCII-файлом, обеспечивающим распределение соответствующих текстовых имен между службами интернет, связанных с ними назначений портов и типов протоколов. Любая сетевая программа должна сначала обратится к этому файлу для получения номера порта (и протокола) для своих служб. Программы библиотеки C **getservent(3)**, **getservbyname(3)**, **getservbyport(3)**, **setservent(3)**, и **endservent(3)** обеспечивают запросы программ к этому файлу. Номера портов назначаются IANA (Internet Assigned Numbers Authority), и их текущей политикой является указание TCP- и UDP-протоколов при назначении номера порта. Следовательно, большинство элементов будет иметь двойные значения даже для служб, которые используют только TCP. Номера портов меньше 1024-х (так называемый 'низкий уровень') могут быть назначены только суперпользователем (смотрите **bind(2)**, **tcp(7)** и **udp(7)**). Это делается для того, чтобы клиенты, подключающиеся к портам низкого уровня, могли не сомневаться в правильной работе сервисов этого порта, являющихся обычными сервисами, а не "подставными" сервисами какого-нибудь пользователя в машине. Известные номера портов, определенные IANA, обычно предназначены в этом пространстве только суперпользователю. Присутствие элемента службы в файле **services** не обязательно означает, что эта служба сейчас в машине действует. Смотрите **inetd.conf(5)** для знакомства с предложенной конфигурацией служб интернета. Заметим, что не все сетевые службы запускаются с помощью **inetd(8)**, поэтому не появятся в **inetd.conf(5)**. В частности, серверы новостей (NNTP) и почтовые серверы (SMTP) часто инициализируются с помощью системных загрузочных скриптов. Расположение файла **services** определяется параметром **_PATH_SERVICES** в */usr/include/netdb.h*. Обычно оно установлено в */etc/services*. Каждая строка определяет одну службу и имеет такую форму:

service-name port/*protocol* [*aliases ...*] ,

где:

service-name
: является известным именем службы, по которому она в дальнейшем будет определяться. Оно зависит от регистра. Часто программы пользователя называются *service-name*.

port
: является номером порта (в десятичном формате), используемым данной службой.

protocol
: является типом протокола, который будет использоваться. Это поле должно совпадать с элементом в файле **protocols(5) tcp** и **udp**.

aliases
: является дополнительным списком имен этой службы, разделенных символами табуляции или пробелами (смотрите секцию СООБЩЕНИЯ ОБ ОШИБКАХ

ниже). Снова заметим, что имена зависят от регистра.

Для разделения полей могут использоваться пробелы или символы табуляции. Комментарии начинаются с символа решетки (#) и продолжаются до конца строки. Пустые строки пропускаются. *service-name* должно начинаться с первого столбца файла, так как начальные пробелы не удаляются. *service-names* могут быть любыми печатаемыми символами (исключая пробелы и табуляцию), однако, лучше использовать стандартный набор символов для уменьшения вероятности ошибок. Например, a-z, 0-9 и дефис (-)
входят в стандартный набор. Строки не в этом формате не должны присутствовать в файле. В данный момент они просто пропускаются программами **getservent(3)**, **getservbyname(3)** и **getservbyport(3)**. Однако, на это в дальнейшем не стоит полагаться. Для совместимости со старыми стандартами косая черта (/) между номером *port* и именем *protocol* может быть заменена запятой (,). В современных версиях запятая не используется. Данный файл может распространяться по сети с помощью сетевой службы имен, таких, как: Yellow Pages/NIS или BIND/Hesiod. Эталонный файл **services** может выглядеть таким образом:

```
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp          # протокол отправки сообщения
msp             18/udp          # протокол отправки сообщения
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp             21/tcp
telnet          23/tcp
```

НАЙДЕННЫЕ ОШИБКИ

Максимальное число существующих псевдонимов - 35 из-за особенностей написания **getservent(3)**. Строки, длиннее **BUFSIZ** символов (в данный момент 1024), будут игнорироваться программами **getservent(3)**, **getservbyname(3)** и **getservbyport(3)**. Однако, это, возможно, приведет к тому, что последующая строка будет ошибочно разделена.

ФАЙЛЫ */etc/services* - список служб интернет.
/usr/include/netdb.h - определение **_PATH_SERVICES**

СМ. ТАКЖЕ

getservent(3), **getservbyname(3)**, **getservbyport(3)**, **setservent(3)**, **endservent(3)**, **protocols(5)**, **listen(2)**, **inetd.conf(5)**, **inetd(8)** Assigned Numbers RFC, most recently RFC 1700, (AKA STD0002) Guide to Yellow Pages Service Guide to BIND/Hesiod Service

Linux 11 Jan 1996 SERVICES**(5)**

SHELLS

НАЗВАНИЕ shells - файл, содержащий пути к командным оболочкам пользователя, которые можно загружать после входа в систему

ОПИСАНИЕ

/etc/shells -- это текстовый файл, содержащий пути к командным оболочкам пользователя, которые можно загружать после входа в систему (прим. переводчика: необходимо указать только одну из систем). К этому файлу обращается **chsh(1)**, а также некоторые другие программы.

Имейте в виду, что некоторые программы обращаются к этому файлу, чтобы выяснить, является ли пользователь обычным пользователем. Например, ftp-демоны традиционно запрещают доступ к программе пользователям, чьи командные оболочки не перечислены в этом файле.

ПРИМЕРЫ **/etc/shells** может содержать такие строки:

/bin/sh
/bin/csh

ФАЙЛЫ */etc/shells*

СМ. ТАКЖЕ **chsh(1), getusershell(3)**

November 21, 1993 SHELLS**(5)**

TERMCAP

НАЗВАНИЕ termcap - база данных терминальных параметров

ОПИСАНИЕ База данных termcap - это устаревший метод описания возможностей алфавитно-цифровых терминалов и принтеров. Она оставлена лишь для обеспечения совместимости со старыми программами; новые программы должны использовать базу данных **terminfo(5)** и соответствующие ей библиотеки.

/etc/termcap - это ASCII-файл (исходный текст базы данных), содержащий список параметров, принадлежащих различным типам терминала. Программы могут читать содержимое termcap для того, чтобы распознавать управляющие последовательности, необходимые для контроля за визуальными атрибутами терминала. Другие свойства терминалов контролируются stty. База данных termcap проиндексирована в соответствии с переменной окружения TERM.

Записи termcap должны быть расположены в одной логической строке, в которой `:'. Первое поле каждой записи начинается с крайней левой колонки и содержит список названий терминала, разделенных '|'.

Первое "подполе" может содержать краткое название терминала, состоящее из двух символов (это касается записей termcap в BSD версий 4.3 и более ранних). Это имя может состоять из прописных и строчных букв. В записях termcap BSD версии 4.4 это поле не учитывается.

Второе "подполе" (на самом деле, первое, только в формате BSD 4.4) содержит название терминала, используемое в переменной окружения TERM. Оно должно быть написано строчными буквами. Различающиеся аппаратные возможности должны отмечаться при помощи суффикса, добавляемого к названию терминала через дефис. Смотрите пример, приведенный ниже. Обычные суффиксы: - w (ширина терминала больше 80-и символов), am (автоматические границы), nam (нет автоматических границ) и rv (инверсный видеодисплей).

Третье "подполе" содержит длинное описательное название записи termcap.

Последующие поля содержат параметры терминала; любые строки, являющиеся продолжением записи, должны начинаться с одного символа табуляции.

Несмотря на то, что порядок задания параметров не определен, рекомендуется сначала задавать переключатели, затем числовые и только после них строковые параметры терминала. Каждая группа должна быть отсортирована в алфавитном порядке, без учета регистра. Параметры похожих свойств терминала должны быть написаны в одной строке.

Пример:

```
Заголовок: vt|vt101|Терминал DEC VT 101 в 80-символьном режиме:\
Заголовок: Vt|vt101-w|Терминал DEC VT 101 в 132-символьном режиме:\
Переключатели: :bs:\
Числовые: :co#80:\
Строковые: :sr=\E[H:\
```

Параметры-переключатели

```
5i   Принтер не отражает "эхо" на экране
am   Автоматические границы (автоматический перенос строки)
bs   Control-H (десят. 8) - забой
```

bw Забой на левой границе возвращает Вас к правой границе предыдущей строки
da Дисплей удерживается над экраном
db Дисплей удерживается под экраном
eo Пробел удаляет все символы с позиции курсора
es Управляющие последовательности и специальные символы действуют в строке состояния
gn Стандартное устройство
hc Этот терминал является печатным
HC Курсор плохо виден не в нижней строке
hs Терминал имеет строку состояния
hz Ошибка Hazeltine: терминал не печатает символы тильды
in Терминал использует символы null вместо пробелов для заполнения пустых мест
km Терминал имеет клавишу Meta
mi Курсор передвигается в режиме вставки
ms Курсор передвигается в режиме выделения/подчеркивания
NP Нет символа pad
NR ti не реверсирует состояние te
nx Заполнения нет, необходимо использовать XON/XOFF
os Терминал может печатать символы поверх существующих
ul Терминал может подчеркивать, но не может печатать расположенные поверх символы
xb Ошибка Beehive: f1 посылает ESCAPE, f2 посылает ^C
xn Ошибка, связанная с переводом строки
xo Терминал использует протокол xon/xoff
xs Текст, выведенный поверх выделенного, также будет выделен
xt Ошибка Teleray: "разрушающая" табуляция и неправильный режим выделения

Числовые параметры

co Количество колонок
dB Задержка в миллисекундах при печати забоя на печатных терминалах
dC Задержка в миллисекундах при печати возврата каретки на печатных терминалах
dF Задержка в миллисекундах при печати перевода формата на печатных терминалах
dN Задержка в миллисекундах при печати перевода строки на печатных терминалах
dT Задержка в миллисекундах при печати символа остановки табулятора на печатных терминалах
dV Задержка в миллисекундах при печати символа остановки вертикального табулятора на печатных терминалах
it Расстояние между позициями табуляции
lh Высота меток
lm Количество строк в памяти
lw Ширина меток
li Количество строк
Nl Количество меток
pb Минимальная скорость при заполнении
sg Проблема с выделением
ug Проблема с подчеркиванием
vt Виртуальный номер терминала
ws Ширина строки состояния, если она отличается от ширины экрана

Строковые параметры

!1 клавиша shift+save (рег+запись)
!2 клавиша shift+suspend (рег+приостановить)
!3 клавиша shift+undo (рег+отменить)
%0 клавиша redo (вернуть)
%1 клавиша help (помощь)
%2 клавиша mark (отметка)
%3 клавиша message (сообщение)
%4 клавиша move (перемещение)
%5 клавиша next-object (следующий объект)
%6 клавиша open (открыть)

```
        %7   клавиша options (опции)
        %8   клавиша previous-object (предыдущий объект)
        %9   клавиша print (печать)
        %a   клавиша shift+message (рег+сообщение)
        %b   клавиша shift+move (рег+перемещение)
        %c   клавиша shift+next (рег+следующий)
        %d   клавиша shift+options (рег+опции)
        %e   клавиша shift+previous (рег+предыдущий)
        %f   клавиша shift+print (рег+печать)
        %g   клавиша shift+redo (рег+вернуть)
        %h   клавиша shift+replace (рег+заменить)
        %i   клавиша shift+курсор вправо
        %j   клавиша shift+resume (рег+возобновить)
        &0   клавиша shift+cancel (рег+отменить)
        &1   клавиша reference (ссылка)
        &2   клавиша refresh (обновить)
        &3   клавиша replace (заменить)
        &4   клавиша restart (перезапустить)
        &5   клавиша resume (возобновить)
        &6   клавиша save (сохранить)
        &7   клавиша suspend (приостановить)
        &8   клавиша undo (отменить)
        &9   клавиша shift+begin (рег+начало)
        *0   клавиша shift+find (рег+поиск)
        *1   клавиша shift+command (рег+команда)
        *2   клавиша shift+copy (рег+копировать)
        *3   клавиша shift+create (рег+создать)
        *4   клавиша shift+delete character (рег+удалить символ)
        *5   клавиша shift+delete line (рег+удалить строку)
        *6   клавиша select (выбор)
        *7   клавиша shift+end (рег+конец)
        *8   клавиша shift+clear line (рег+"очистить" строку)
        *9   клавиша shift+exit (рег+выход)
        @0   клавиша find (искать)
        @1   клавиша begin (начать)
        @2   клавиша cancel (отменить)
        @3   клавиша close (закрыть)
        @4   клавиша command (задать команду)
        @5   клавиша copy (копировать)
        @6   клавиша create (создать)
        @7   клавиша end (конец)
        @8   клавиша enter/send (ввод/послать)
        @9   клавиша exit (выход)
        al   Вставить одну строку
        AL   Сдвинуть %1 строк
        ac   Пары символов для преобразования псевдографики
        ae   Конец альтернативного набора символов
        as   Начало альтернативного набора символов, содержащего символы
псевдографики
        bc   Забой, если он не равен ^H
        bl   Звуковой сигнал
        bt   Возврат к предыдущей позиции табуляции
        cb   "Очистка" от начала строки до курсора
        cc   Символ простой команды
        cd   "Очистка" до конца экрана
        ce   "Очистка" до конца строки
        ch   Переместить курсор на колонку %1
        cl   "Очистка" экрана и перевод курсора в его начало
        cm   Переместить курсор в ряд %1 и колонку %2 (на экране)
        CM   Переместить курсор в ряд %1 и колонку %2 (в памяти)
        cr   Возврат каретки
        cs   Прокрутить область со строки %1 до строки %2
        ct   "Очистить" табуляцию
        cv   Переместить курсор вертикально в строку %1
```

```
        dc   Удалить один символ
        DC   Удалить %1 символов
        dl   Удалить одну строку
        DL   Удалить %1 строк
        dm   Начало режима удаления
        do   Курсор вниз на одну строку
        DO   Курсор вниз на #1 строк
        ds   Запретить строку состояния
        eA   Разрешить альтернативный набор символов
        ec   Очистить %1 символов, начиная с курсора
        ed   Завершение режима удаления
        ei   Завершение режима вставки
        ff   Символ перевода формата на печатных терминалах
        fs   Возвратить курсор на прежнюю позицию после перехода к строке
состояния
        F1   Строка, посылаемая функциональной клавишей f11
        F2   Строка, посылаемая функциональной клавишей f12
        F3   Строка, посылаемая функциональной клавишей f13
        ...  ...
        F9   Строка, посылаемая функциональной клавишей f19
        FA   Строка, посылаемая функциональной клавишей f20
        FB   Строка, посылаемая функциональной клавишей f21
        ...  ...
        FZ   Строка, посылаемая функциональной клавишей f45
        Fa   Строка, посылаемая функциональной клавишей f46
        Fb   Строка, посылаемая функциональной клавишей f47
        ...  ...
        Fr   Строка, посылаемая функциональной клавишей f63
        hd   Переместить курсор на полстроки вниз
        ho   Переместить курсор в начало строки
        hu   Переместить курсор на полстроки вверх
        i1   Инициализационная строка номер 1 при входе в систему
        i3   Инициализационная строка номер 3 при входе в систему
        is   Инициализационная строка номер 2 при входе в систему
        ic   Вставить один символ
        IC   Вставить %1 символов
        if   Файл инициализации
        im   Начало режима вставки
        ip   Добавлять время вставки и специальные символы после вставки
        iP   Программа инициализации
        K1   Левая верхняя клавиша на дополнительной клавиатуре
        K2   Центральная клавиша на дополнительной клавиатуре
        K3   Верхняя правая клавиша на дополнительной клавиатуре
        K4   Нижняя левая клавиша на дополнительной клавиатуре
        K5   Нижняя правая клавиша на дополнительной клавиатуре
        k0   Функциональная клавиша 0
        k1   Функциональная клавиша 1
        k2   Функциональная клавиша 2
        k3   Функциональная клавиша 3
        k4   Функциональная клавиша 4
        k5   Функциональная клавиша 5
        k6   Функциональная клавиша 6
        k7   Функциональная клавиша 7
        k8   Функциональная клавиша 8
        k9   Функциональная клавиша 9
        k;   Функциональная клавиша 10
        ka   Клавиша "очистки" всех табуляций
        kA   Клавиша вставки строки
        kb   Клавиша забоя
        kB   Клавиша обратной табуляция
        kC   Клавиша "очистки" экрана
        kd   Клавиша 'курсор вниз'
        kD   Клавиша удаления одного символа в позиции курсора
        ke   Отключить дополнительную клавиатуру
```

```
        kE   Клавиша "очистки" до конца строки
        kF   Клавиша прокрутки вперед/вниз
        kh   Клавиша "курсор в начало строки"
        kH   Клавиша "Cursor hown down"
        kI   Клавиша вставки символа/включения режима вставки
        kl   Клавиша "курсор влево"
        kL   Клавиша удаления строки
        kM   Клавиша выхода из режима вставки
        kN   Клавиша "следующая страница"
        kP   Клавиша "предыдущая страница"
        kr   Клавиша "курсор вправо"
        kR   Клавиша прокрутки назад/вверх
        ks   Включить дополнительную клавиатуру
        kS   Клавиша "очистки" до конца экрана
        kt   Клавиша "очистки" этой табуляции
        kT   Клавиша установки табуляции
        ku   Клавиша "курсор вверх"
        l0   Название нулевой функциональной клавиши, если оно не равно f0
        l1   Название первой функциональной клавиши, если оно не равно f1
        l2   Название второй функциональной клавиши, если оно не равно f2
        ...  ...
        la   Название десятой функциональной клавиши, если оно не равно f10
        le   Курсор влево на один символ
        ll   Переместить курсор в левый нижний угол экрана
        LE   Курсор влево на %1 символов
        LF   Выключить метки
        LO   Включить метки
        mb   Мигающий шрифт
        MC   "Очистить" необязательные границы
        md   Жирный шрифт
        me   Отключить все режимы типов: so, us, mb, md и mr
        mh   Начало режима половинной яркости
        mk   "Темный" режим (символы невидимы)
        ML   Задать левую необязательную границу
        mm   Включить meta-режим терминала
        mo   Отключить meta-режим терминала
        mp   Включить защищенные атрибуты
        mr   Инверсный шрифт
        MR   Задать правую необязательную границу
        nd   Курсор вправо на один символ
        nw   Команда возврата каретки
        pc   Символ заполнения
        pf   Выключить принтер
        pk   Запрограммировать клавишу %1 на отправку строки %2, как будто она
набрана пользователем
        pl   Запрограммировать клавишу %1 на исполнение строки %2 в локальном
режиме
        pn   Запрограммировать метку %1 на вывод строки %2
        po   Включить принтер
        pO   Включить принтер на %1 (<256) байтов
        ps   Распечатать содержимое экрана на принтере
        px   Запрограммировать клавишу %1 на отправку строки %2 на компьютер
        r1   Первая строка сброса, возвращающая нормальный режим терминала
        r2   Вторая строка сброса, возвращающая нормальный режим терминала
        r3   Третья строка сброса, возвращающая нормальный режим терминала
        RA   Запретить автоматические границы
        rc   Восстановить сохраненное положение курсора
        rf   Имя файла со строкой сброса
        RF   Запрос ввода с терминала
        RI   Курсор вправо на %1 символов
        rp   Повторить символ %1 %2 раз
        rP   Заполнение после отправки символа в режиме замены
        rs   Строка сброса
        RX   Отключить контроль потока XON/XOFF
```

```
sa   Установить атрибуты %1 %2 %3 %4 %5 %6 %7 %8 %9
SA   Разрешить автоматические границы
sc   Сохранить позицию курсора
se   Конец режима выделения
sf   Нормальная прокрутка на одну строку
SF   Нормальная прокрутка на %1 строк
so   Начало режима выделения
sr   Обратная прокрутка
SR   Обратная прокрутка на %1 строк
st   Установить остановку табулятора во всех рядах этой колонки
SX   Включить контроль потока XON/XOFF
ta   Переместиться на следующую аппаратную позицию табуляции
tc   Прочитать описание терминала из другой записи
te   Конец программы, использующей перемещение курсора
ti   Начало программы, использующей перемещение курсора
ts   Переместить курсор в колонку %1 строки состояния
uc   Подчеркнуть символ под курсором и переместить курсор вправо
ue   Завершить подчеркивание
up   Переместить курсор на строку вверх
UP   Переместить курсор на %1 строк вверх
us   Начать подчеркивание
vb   Визуальный сигнал
ve   Обычный курсор
vi   Курсор невидим
vs   Выделенный курсор
wi   Задать размер окна со строки %1 по %2 и с колонки %3 по %4
XF   Символ XOFF, если он не равен ^S
```

Существует несколько способов определения управляющих кодов в строковых параметрах:

обычные символы, кроме '^','\' и '%', говорят сами за себя;

\x соответствует специальному символу. x может быть следующим:

```
       E Escape (27)
       n Перевод строки (10)
       r Возврат каретки (13)
       t Табуляция (9)
       b Забой (8)
       f Перевод формата (12)
       0 Null. \xxx означает символ с с восьмеричным
       кодом xxx.

i      Увеличить параметр на 1
r      Возможность одного параметра
+      Добавить значение следующего символа к этому
       параметру и произвести двоичный вывод
2      Произвести ASCII-вывод этого параметра с полем из
       2-х символов
d      Произвести ASCII-вывод этого параметра с полем из
       3-х символов
%      Напечатать символ'%'
```

Если Вы используете двоичный вывод, то Вы должны избегать использования символа null, так как он означает конец строки. Вам также надо "очистить" расширение табулятора, если в двоичном выводе будет присутствовать символ табуляции.

Внимание:

описанные выше метасимволы могут быть неправильными. Они описывают termcap Minix, который

может быть несовместимым с Linux termcap.

Символы псведографики определяются при помощи трех строковых параметров:

as начало альтернативного набора символов;
ae окончание альтернативного набора символов;
ac пары символов. Первый символ - это имя символа псевдографики, а второй - его определение.

Доступны следующие имена:

+	стрелка вправо (>)
,	стрелка влево (<)
.	стрелка вниз (v)
0	полный квадрат (#)
I	решетка (#)
-	стрелка вверх (^)
'	ромб (+)
a	шахматная доска (:)
f	градус (')
g	плюс-минус (#)
h	квадрат (#)
j	правый нижний угол (+)
k	правый верхний угол (+)
l	левый верхний угол (+)
m	левый нижний угол (+)
n	крест (+)
o	верхняя горизонтальная линия (-)
q	средняя горизонтальная линия (-)
s	нижняя горизонтальная линия (_)
t	ответвление влево (+)
u	ответвление вправо (+)
v	ответвление вниз (+)
w	ответвление вверх (+)
x	вертикальная линия (|)
~	параграф (???)

Значения, указанные в скобках, используются curses по умолчанию, если эти параметры отсутствуют.

СМ. ТАКЖЕ **termcap(3), curses(3), terminfo(5)**

Linux TERMCAP**(5)**

TTYTYPE

НАЗВАНИЕ ttytype - соответствие устройств типам терминалов

ОПИСАНИЕ Файл */etc/ttytype* задает соответствие типов терминалов, указанных в termcap/terminfo, tty-линиям. Каждая строка содержит тип терминала, за которым после пустого пространства стоит название tty (имя устройства без префикса */dev/*). Этот файл используется программой **tset(1)** для установки значения переменной окружения TERM, задающей стандартный тип терминала текущего tty пользователя. Описание типов терминалов в этом файле было разработано для стандартных систем, разделяющих время работы с алфавитно-цифровыми терминалами, подключенными к микрокомпьютеру с системой Unix. Она редко используется на современных рабочих станциях и в персональных компьютерах.

ПРИМЕР Обычно файл */etc/ttytype* выглядит примерно так:

```
con80x25 tty1
vt320 ttys0
```

ФАЙЛЫ */etc/ttytype* - файл определения типов tty.

СМ. ТАКЖЕ **getty(1), terminfo(5), termcap(5)**

Linux 24 July 1993 TTYTYPE**(5)**

TZFILE

НАЗВАНИЕ tzfile - информация о часовом поясе

СИНТАКСИС

#include <tzfile.h>

ОПИСАНИЕ

Файлы с информацией о часовых поясах, используемые функцией **tzset(3)**, начинаются со специальных символов "TZif", указывающих, что это - файл с информацией о часовом поясе. За ними следуют 16 байтов, зарезервированных для их последующего использования. После них находятся шесть четырехбайтных значений типа **long**, записанных в "стандартном" порядке байтов (старшие байты записываются первыми). Это следующие значения:

tzh_ttisgmtcnt
(количество UTC/местных индикаторов в файле);

tzh_ttisstdcnt
(количество стандартных/местных индикаторов в файле);

tzh_leapcnt
(количество високосных секунд в файле);

tzh_timecnt
(количество "переходных периодов" в файле);

tzh_typecnt
(количество "типов местного времени" в файле. Оно не должно быть равно 0);

tzh_charcnt
(количество символов в "аббревиатурах часовых поясов" файла).

За этим заголовком следуют четырехбайтные значения *tzh_timecnt* типа **long**, расположенные в порядке возрастания. Эти значения записаны в "стандартном" порядке байтов. Каждое из этих значений является временем "переходного периода" (в формате, возвращаемом функцией **time(2)**), при задании которого изменяются правила вычисления местного времени. После этого следуют однобайтные значения *tzh_timecnt* типа **unsigned char**; каждое из этих значений указывает, какому типу "местного времени", описанному в файле, соответствует "переходный период", порядковый номер которого в предыдущем списке совпадает с номером этого значения. Эти значения выступают в роли индексов массива структур *ttinfo*; массив находится в файле после этих значений. Эти структуры состоят из следующих полей:

```
struct ttinfo {
     long          tt_gmtoff;
     int           tt_isdst;
     unsigned int  tt_abbrind;
};
```

Каждая структура состоит из четырехбайтного значения *tt_gmtoff* типа **long**, записанного в "стандартном" порядке байтов; за этим значением следует однобайтное значение *tt_isdst* и однобайтное значение *tt_abbrind*. В каждой структуре *tt_gmtoff* означает, сколько секунда надо добавить к UTC; *tt_isdst* означает, что *tm_isdst* должно быть установлено при помощи **localtime(3)**, и *tt_abbrind* является индексом массива аббревиатур часовых поясов; массив следует за структурами *ttinfo*.

После этого массива стоят пары четырехбайтных значений *tzh_leapcnt*, записанные в "стандартном" порядке байтов; первое значение каждой пары задает время (в формате **time(2))** високосной секунды; вторая пара определяет *общее* количество високосных секунд, добавляемых к указанному

моменту времени. Пары значений располагаются в порядке возрастания.
Затем в файле располагаются индикаторы стандартного/местного времени *tzh_ttisstdcnt*, каждый из которых имеет однобайтное значение; эти индикаторы определяют, как заданы "переходные периоды" местного времени: согласно стандартному или местному времени. Эти индикаторы используются, когда файл с информацией о часовых поясах работает с POSIX-совместимыми переменными окружения часовых поясов.
И, наконец, в этом файле находятся UTC/местные индикаторы *tzh_ttisgmtcnt*, каждый из которых представлен в виде однобайтного значения; эти индикаторы определяют, как заданы "переходные периоды" местного времени: согласно UTC или местному времени. Эти индикаторы используются, когда файл с информацией о часовых поясах работает с POSIX-совместимыми переменными окружения часовых поясов.
Localtime использует первую структуру стандартного местного времени *ttinfo* из этого файла (или просто первую структуру *ttinfo* при отсутствии структур стандартного местного времени), если *tzh_timecnt* равен 0, или аргумент времени меньше временного аргумента первого "переходного периода", записанного в файле.

СМ. ТАКЖЕ **newctime(3)**

UTMP

НАЗВАНИЕ utmp, wtmp - записи регистрационного имени

СИНТАКСИС

ОПИСАНИЕ Файл **utmp** позволяет получать информацию о том, кто в данный момент работает в системе. Пользователей, в данное время использующих
систему, может быть большое количество, поскольку не все программы используют регистрацию посредством utmp.
ВНИМАНИЕ: utmp не должен быть записываемым, так как многие системные программы (по каким-то необъяснимым причинам) зависят от его целостности. Вы рискуете спутать системные файлы статистики и внести изменения в системные файлы, если Вы предоставите любому пользователю возможность написать файл **utmp** .
Этот файл представляет собой последовательность элементов со следующей структурой (заметим, что в данном случае указано только одно из нескольких определений; детали зависят от версии libc):

```
        struct exit_status {
          short int e_termination;      /* статус завершения процесса.
*/
          short int e_exit;     /* статус выхода из процесса.  */
        };
        struct utmp {
          short ut_type;        /* тип входа */
          pid_t ut_pid; /* идентификатор pid входного процесса */
          char ut_line[UT_LINESIZE];    /* имя устройства tty - "/dev/"
*/
          char ut_id[4];        /* начальный id или сокращенное ttyname
*/
          char ut_user[UT_NAMESIZE];    /* имя пользователя */
          char ut_host[UT_HOSTSIZE];    /* имя хоста для удаленного
доступа */
          struct exit_status ut_exit;   /* статус выхода процесса,
                  отмеченного как DEAD_PROCESS. */
```

```
                long ut_session;      /* ID сессии, используемый для
управления окнами */
                struct timeval ut_tv; /* был создан элемент времени.  */
                int32_t ut_addr_v6[4];        /* IP-адрес удаленного хоста.
*/
                char pad[20]; /* Зарезервировано для будущего использования.
*/
              };
              /* Оставлено для совместимости со старыми версиями.  */
```

Эта структура дает имя специальному файлу, связанному с терминалом пользователя, именем входа пользователя и временем входа, обозначенным как **time(2)**. Поля строк заканчиваются **'\0'**, если они короче, чем размер целого поля.

Первые элементы, когда-либо созданные, заставят **init(8)** запустить **inittab(5)**. Хотя перед тем, как элемент обрабатывается, **init(8)** "очищает" utmp, устанавливая **ut_type** равным **DEAD_PROCESS** и заполняя **ut_user**, **ut_host** и **ut_time** нулевыми байтами в записях, в которых **ut_type** не является **DEAD_PROCESS** или **RUN_LVL**, и где нет существующих процессов с PID, равным **ut_pid**. Если не найдено ни одной пустой записи с нужным **ut_id**, то init создает новый. Устанавливаемые значения **ut_id** из inittab, **ut_pid** и **ut_time** равны текущим значениям, и **ut_type** равно **INIT_PROCESS**.

getty(8) находит элементы по их идентификатору pid, меняет **ut_type** на **LOGIN_PROCESS**, меняет **ut_time**, устанавливает **ut_line** и ожидает установки соединения. **login(8)** после того, как пользователь был идентифицирован, меняет **ut_type** на **USER_PROCESS**, меняет **ut_time** и устанавливает **ut_host** и **ut_addr**. В зависимости от **getty(8)** и **login(8)**, записи могут быть расположены согласно **ut_line** вместо предпочитаемого расположения согласно **ut_pid**.

Когда **init(8)** находит, что процесс завершился, он определяет, осуществлен ли вход процесса в utmp по **ut_pid**; меняет **ut_type** на **DEAD_PROCESS** и заполняет **ut_user**, **ut_host** и **ut_time** нулевыми байтами.

xterm(1) и другиме эмуляторы терминала непосредственно создают запись **USER_PROCESS** и генерируют **ut_id**, используя последние две буквы **/dev/ttyp***%c* или используя **p***%d* для **/dev/pts/***%d*. Если они обнаруживают флаг **DEAD_PROCESS** для этого идентификатора, то они удаляют его или создают новый элемент. Если допускается, то они пометят его как **DEAD_PROCESS** при выходе ; предполагается, что они также обнуляют **ut_line**, **ut_time**, **ut_user** и **ut_host**.

xdm(8) не должно создавать записи в utmp, так как нет назначенного ему терминала. Если позволить ему создавать их, то это приведет примерно к такому результату: finger: can not stat /dev/machine.dom. Данный аргумент должен создавать элементы в wtmp, так, как это делает **ftpd(8)**.

telnetd(8) устанавливает элемент **LOGIN_PROCESS** и оставляет остальное, как это обычно бывает, аргументу **login(8)**. После того, как закончится telnet-сессия, **telnetd(8)** "очищает" utmp вышеописанным путем.

Файл **wtmp** записывает все входы и выходы в систему. Его формат в точности похож на формат **utmp** (за исключением того, что "пустое" имя пользователя означает выход из системы через связанный терминал). Кроме того, название терминала **"~"** с именем пользователя **"shutdown"** или **"reboot"** означает отключение системы или ее перезагрузку, а пара названий терминала **"|"/"}"** означает старое/новое

системное время в случае, когда **date(1)** меняет их. **wtmp** поддерживается **login(1)**, **init(1)** и некоторыми версиями **getty(1)**. Ни одна из этих программ не создает файл, поэтому если он удален, то ведение записей заканчивается.

ФАЙЛЫ

/var/run/utmp
/var/log/wtmp

СООТВЕТСТВИЕ

Элементы Linux utmp не соответствуют ни v7/BSD, ни SYSV: они являются комбинацией их обеих. v7/BSD имеет меньшее количество полей; важнее всего то, что в нем нет **ut_type**, который заставляет v7/BSD-совместимые программы выводить, к примеру, записи "зависания" системы или входа в нее. Также в этой версии отсутствует конфигурационный файл, который определяет места сессий. Все это делается в BSD из-за отстутствия поля **ut_id**. В Linux (как и в SYSV) поле **ut_id** записи никогда не меняется после того, как один раз установлено, что оно резервирует для себя место в конфигурационном файле без необходимости. "Очищение" **ut_id** может привести к повреждению элементов utmp и нарушению безопасности системы. Заполнение вышеупомянутых полей нулевыми байтами не требуется согласно семантике SYSV, но позволяет запускать многие программы, которые понимают семантику BSD и которые не изменяют utmp. Linux использует преобразования BSD для работы с содержимым строк, как описано выше.

SYSV использует только типовые поля для отметок и регистрирует информационные сообщения (такие, как "**new time**") в поле строк. **UT_UNKNOWN**, похоже, является нововведением Linux. SYSV не имеет полей **ut_host** и **ut_addr_v6**.

В отличие от других систем, где регистрация сообщений utmp может быть снята при помощи удаления файла, utmp всегда должен находиться в Linux. Если Вы хотите отключить **who(1)**, то просто сделайте чтение utmp недоступным всем.

Заметим, что структура utmp, описанная в libc5, изменилась в libc6. Из-за этого бинарные файлы, использующие старую структуру версии libc5, будут повреждать */var/run/utmp* и/или */var/log/wtmp*. Система Debian включает в себя исправленные libc5, которые используют новый формат utmp. Но проблема с wtmp все еще существует, так как обращения к нему идут непосредственно из libc5.

ОГРАНИЧЕНИЯ

Формат файла зависит от машины, поэтому рекомендуется, чтобы он обрабатывался только с помощью той машинной архитектуре, в которой он создавался.

НАЙДЕННЫЕ ОШИБКИ

Эти страницы основаны на материалах libc5, в новых же версиях может быть указано иное.

СМ. ТАКЖЕ ac(1), date(1), getutent(3), init(8), last(1), login(1), updwtmp(3), who(1)

July 2, 1997 UTMP**(5)**