

# MAN PAGES

## 4 . Специальные файлы

### CONSOLE

**НАЗВАНИЕ**

`console` - текстовый терминал и виртуальные консоли

**ОПИСАНИЕ**

В Linux может быть до шестидесяти трех **виртуальных консолей** (символьных устройств со старшим числом, равным 4-м, и младшим от 1-ого до 63-х), обычно указываемых как `/dev/ttypn` с  $1 \leq n \leq 63$ . К консоли можно обратиться с помощью указания `/dev/console` (или `/dev/tty0`), символьного устройства со старшим числом, равным 4-м, и младшим, равным нулю. Файлы устройств `/dev/*` обычно создаются скриптом `MAKEDEV` или при помощи `mknod(1)`, который имеет права, равные 0622, и владельца `root.tty`.

Пока не появилась версия ядра под номером 1.1.54, число виртуальных консолей было встроенным в ядро (а именно, в `tty.h: #define NR_CONSOLES 8`) и изменялось редактированием и перекомпиляцией. Начиная с версии 1.1.54, виртуальные консоли создаются по мере надобности.

Обычными способами запуска процесса на консоли являются:

- (а) задание команды `init(8)`, в `inittab(5)` для запуска `getty(8)`;
- (б) запрос `open(1)` для запуска процесса на консоли;
- (в) запуск X (она найдет первую неиспользуемую консоль и отобразит на ней данные). (Также в вашем распоряжении имеется старый `doshell(8)`.)

Переключение с одной консоли на другую осуществляется:

- (а) нажатием `Alt+Fn` (или `Ctrl+Alt+Fn`) для переключения на консоль  $n$  или `AltGr+Fn`: это сочетание переключит Вас на консоль  $n+12$  [в данном случае `Alt` и `AltGr` - левая и правая клавиши `Alt` соответственно];
- (б) нажатием `Alt+RightArrow` или `Alt+LeftArrow` для поочередного переключения с одной консоли на другую;
- (в) при помощи программы `chvt(1)` (комбинации клавиш можно определить вновь, см. `loadkeys(1)`; описанные выше сочетания заданы по умолчанию).

Команда `deallocvt(1)` (в прошлом `disalloc`) освободит память, занимаемую буферами экрана тех консолей, которые уже не имеют ассоциированных с ними процессов.

**СВОЙСТВА**

Консоль может работать в различных режимах. Возможно, они будут описаны в следующей версии этого документа. Одним из свойств консоли является возможность создавать терминалы `vt100`. Консоль приводится в первоначальное состояние с помощью написания двух символов: `ESC` и `s`. Все последовательности описываются в `console_codes(4)`.

**ФАЙЛЫ**

`/dev/console`  
`/dev/tty*`

**СМ. ТАКЖЕ** `chvt(1)`, `deallocvt(1)`, `loadkeys(1)`, `mknod(1)`, `openvt(1)`, `console_codes(4)`, `console_ioctl(4)`, `tty(4)`, `ttys(4)`, `charsets(7)`, `getty(8)`, `init(8)`, `mapscrn(8)`, `resizecons(8)`, `setfont(8)`

Linux

1994-10-31

CONSOLE(4)

### CONSOLE\_CODES

**НАЗВАНИЕ**

`console_codes` - управляющие и `ESC`-последовательности консоли Linux

**ОПИСАНИЕ**

Консоль Linux имеет большой набор средств управления терминалами VT102 и ECMA-48/ISO 6429/ANSI X3.64, а также некоторым количеством частных последовательностей для изменения цветовой палитры, распределений символьного

набора и т.п. В таблицах, приведенных ниже (во второй колонке), указывается мнемонический код ECMA-48 или DEC (если последнее приписано к DEC) данной функции. Последовательности без мнемонического кода не принадлежат ни к ECMA-48, ни к VT102.

Первым процессом, который выполняется после работы обычного процесса по выводу информации и отправки к драйверу потока символов консоли (для их реального вывода), является перевод кодов, используемых при обычной работе, в коды, применяемые для печати.

Если консоль работает в режиме UTF-8, то входящие байты сначала организуются в 16-битовом уникоде. В ином случае каждый байт преобразуется согласно текущей таблице распределения, которая переводит данные в уникод. (Смотри раздел НАБОРЫ СИМВОЛОВ ниже).

Обычно значения уникода преобразуются в индекс шрифта, который сохраняется в видеопамяти таким образом, что соответствующий глиф (обнаруженный в видео-ПЗУ) появляется на экране. Заметим, что работа с уникодом (и текущими аппаратными средствами компьютеров) позволяет одновременно использовать только 512 глифов.

Если текущим значением уникода является управляющий символ или если в данный момент обрабатывается ESC-последовательность, значение будет обрабатываться особым образом. Вместо того, чтобы значение преобразовало себя в шрифт и отобразилось как глиф, оно будет выполнять определенные действия (например, передвигать курсор) или выполнять какие-нибудь другие функции управления. Смотрите ниже раздел УПРАВЛЕНИЕ КОНСОЛЬЮ LINUX.

Некорректно строго задавать программам управление терминалом. Linux поддерживает работу с базой возможностей терминала. **terminfo(5)**. Предпочитая не вводить управляющие ESC-последовательности вручную, Вы, наверняка, захотите использовать для этой работы библиотеку информации об экране terminfo-aware или другие утилиты, такие, как: **ncurses(3)**, **tput(1)** или **reset(1)**.

## УПРАВЛЕНИЕ КОНСОЛЬЮ LINUX

Этот раздел описывает все управляющие символы и ESC-последовательности, которые выполняют специальные команды (т.е., все, что отличается от простого вывода символа в текущую позицию курсора) консоли Linux.

### Управляющие символы

Символ считается управляющим, если (до преобразования согласно таблице перекодировки) он содержит один из 14-и кодов: 00 (NUL), 07 (BEL), 08 (BS), 09 (HT), 0a (LF), 0b (VT), 0c (FF), 0d (CR), 0e (SO), 0f (SI), 18 (CAN), 1a (SUB), 1b (ESC), 7f (DEL). Можно установить режим `показывать управляющие символы' (см. ниже), при этом символы 07, 09, 0b, 18, 1a, 7f будут выводиться на экран как глифы. С другой стороны, в режиме UTF-8 все коды с 00 по 1f воспринимаются как управляющие символы (независимо от режима `показывать управляющие символы'). Если встречается управляющий символ, то он немедленно указывается, а в дальнейшем никак не учитывается (даже если он стоял в середине ESC-последовательности), и управляющая последовательность продолжается со следующего символа. Однако, символ ESC, начинаящий новую управляющую последовательность, возможно, отменит незаконченную предыдущую последовательность, а команды CAN и SUB точно закончат управляющую последовательность. Распознаваемыми управляющими символами являются: BEL, BS, HT, LF, VT, FF, CR, SO, SI, CAN, SUB, ESC, DEL, CSI. Они выполняют стандартные для них действия:

BEL (0x07, ^G) издает звуковой сигнал;

BS (0x08, ^H) удаляет предыдущий символ (смещается влево на одну колонку, но не далее, чем за начало строки);  
 HT (0x09, ^I) перемещается на следующую позицию табуляции или в конец строки, если до этого не было установлено позиций табуляции;  
 LF (0x0A, ^J), VT (0x0B, ^K) и FF (0x0C, ^L) задают перевод строки;  
 CR (0x0D, ^M) задает перевод каретки;  
 SO (0x0E, ^N) приводит в действие набор символов G1, и если LF/NL (режим новой строки) включен, то задает перевод каретки;  
 SI (0x0F, ^O) приводит в действие набор символов G0;  
 CAN (0x18, ^X) и SUB (0x1A, ^Z) прерывают ESC-последовательность;  
 ESC (0x1B, ^[) начинает ESC-последовательность;  
 DEL (0x7F) игнорируется;  
 CSI (0x9B) эквивалентно ESC [ .

#### **ESC-, но не CSI-последовательности**

ESC C	RIS	Сброс.
ESC D	IND	Пропуск строки.
ESC E	NEL	Новая строка.
ESC H	HTS	Установить позицию табуляции в текущей колонке.
ESC M	RI	Возврат строки (обратный пропуск).
ESC Z	DECID	Частное определение DEC. Система возвращает строки ESC [ ? 6 с, претендуя на VT102.
ESC 7	DECSC	Сохранить текущую состояние (координаты курсора, атрибуты, наборы символов, указанные G0, G1).
ESC 8	DECRC	Восстановить последнее сохраненное ESC 7 состояние.
ESC [	CSI	Ввод управляющей последовательности
ESC %		Начать последовательность с выбором набора символов
ESC % @		выбрать по умолчанию (ISO 646 / ISO 8859-1)
ESC % G		выбрать UTF-8
ESC % 8		выбрать UTF-8 (устаревшее)
ESC # 8	DECALN	экранный тест на выравнивание DEC: заполняет экран
символами Е.		
ESC (		Начать последовательность с выбором набора символов
G0		
ESC ( B		выбрать по умолчанию (распределение ISO 8859-1)
ESC ( 0		выбрать графическое распределение vt100
ESC ( U		выбрать null-распределение: сразу в символьное
ПЗУ		
ESC ( K		выбрать распределение пользователя: загружаемое утилитой <b>mapscrn(8)</b> .
ESC )		Начать последовательность с выбором набора символов
G1		(включая символы: B, 0, U, K,- как в примере выше).
ESC >	DECPNM	Использовать дополнительную клавиатуру как цифровую
ESC =	DECPAM	Использовать дополнительную клавиатуру как
управляющую		
ESC ]	OSC	(Сокращение "команд операционной системы")
последнейР :- (.		ESC ] P nrrgbb: установить палитру с параметрами, заданными в 7-и шестнадцатеричных разрядах после
(0-255).		Здесь p является цветом 0-16, а rrggbb указывает на использование красного/зеленого/голубого цвета
		ESC ] R: сбросить палитру

#### **ECMA-48 CSI-последовательности**

CSI (или ESC [) сопровождаются последовательностью параметров, являющихся десятичными номерами, разделенными точкой с запятой (самый больший из них - NPAR (16)). "Пустой" параметр (или отсутствие его) приравнивается нулю. Последовательность параметров может начинаться

одним знаком вопроса. Тем не менее, после CSI [ (или ESC [ [ ]) считывается единственный символ и оставшаяся часть последовательности игнорируется. (Смысл этого в том, чтобы игнорировать отображаемую функциональную клавишу). Результат задания CSI-последовательности определяется ее конечным символом.

@	ICH	Вставить N пустых символов.
A	CUU	Переместить курсор вверх на N рядов.
B	CUD	Переместить курсор вниз на N рядов.
C	CUF	Переместить курсор вправо на N столбцов.
D	CUB	Переместить курсор влево на N столбцов.
E	CNL	Переместить курсор вниз на N рядов, в столбец #1.
F	CPL	Переместить курсор вверх на N рядов, в столбец #1.
G	CHA	Переместить курсор в указанный столбец текущего ряда.
H	CUP	Переместить курсор в указанный ряд и столбец (начало в 1,1).
J	ED	"Очистить" экран (по умолчанию от курсора до конца экрана).
		ESC [ 1 J: "очистить" от начала столбца до курсора.
		ESC [ 2 J: "очистить" весь экран.
K	EL	"Очистить" строку (по умолчанию от курсора до ее конца).
		ESC [ 1 K: "очистить" от начала строки до курсора.
		ESC [ 2 K: "очистить" всю строку.
L	IL	Вставить N пустых строк.
M	DL	Удалить N строк.
P	DCH	Удалить (со смещением в строке) N символов в текущей строке.
X	ECH	"Очистить" (без смещения в строке) N символов в текущей строке.
a	HPR	Переместить курсор вправо на N столбцов.
c	DA	Ответить ESC [ ? 6 c: `Я являюсь VT102'.
d	VPA	Переместить курсор в указанный ряд текущего столбца.
e	VPR	Переместить курсор вниз на N рядов.
f	HVP	Переместить курсор в указанный ряд и столбец.
g	TBC	Без параметров: "очистить" текущую позицию табуляции.
		ESC [ 3 g: удалить все позиции табуляции.
h	SM	Режим установки (см. ниже).
l	RM	Режим сброса (см. ниже).
m	SGR	Установка атрибутов (см. ниже).
n	DSR	Отчет о статусе (см. ниже).
q	DECLL	Установить режимы работы индикаторов на клавиатуре.
		ESC [ 0 q: выключить все индикаторы
		ESC [ 1 q: включить индикатор "Scroll Lock"
		ESC [ 2 q: включить индикатор "Num Lock"
		ESC [ 3 q: включить индикатор "Caps Lock"
r	DECSTBM	Установить область прокрутки; параметрами будут верхний и нижний ряды.
s	?	Сохранить местоположение курсора.
u	?	Восстановить местоположение курсора.
'	HPA	Переместить курсор в указанный столбец текущего ряда.

#### ECMA-48 Установка параметров графики

Последовательность ECMA-48 SGR управляющих символов ESC [ <parameters> m устанавливает атрибуты экрана. Несколько атрибутов могут быть заданы в одной последовательности.

парам.	результат
0	сбросить все атрибуты в их значения по умолчанию
1	установить жирный шрифт
2	установить более яркий (имитированное цветом на цветном дисплее)
4	установить подчеркивание (имитированное цветом на цветном дисплее);
	цвета, используемые для имитации затемнения или подчеркивания, устанавливаются
	при помощи ESC ] ...

5                    включить мерцание  
 7                    включить режим инвертированного видео  
 10                  сбросить выбранное распределение, флаги управления экраном  
 и переключить метафлаг  
 11                  выбрать null-распределение, установить флаг управления  
 экраном,  
 сбросить переключатель метафлага.  
 12                  выбрать null-распределение, установить флаг управления  
 экраном,  
 включить переключатель метафлага. Переключение метафлага  
 задает переключение старшего бита в байте  
 до его трансформации согласно таблице распределения.  
 21                  включить режим нормальной интенсивности (несовместимо с ECMA-  
 48)  
 22                  выключить режим нормальной интенсивности  
 24                  выключить подчеркивание  
 25                  выключить мерцание  
 27                  выключить инвертированное видео  
 30                  установить черный цвет символов  
 31                  установить красный цвет символов  
 32                  установить зеленый цвет символов  
 33                  установить коричневый цвет символов  
 34                  установить синий цвет символов  
 35                  установить сиреневый цвет символов  
 36                  установить голубой цвет символов  
 37                  установить белый цвет символов  
 38                  включить подчеркивание, установить цвет символов по умолчанию  
 39                  выключить подчеркивание, установить цвет символов по  
 умолчанию  
 40                  установить черный цвет фона  
 41                  установить красный цвет фона  
 42                  установить зеленый цвет фона  
 43                  установить коричневый цвет фона  
 44                  установить синий цвет фона  
 45                  установить сиреневый цвет фона  
 46                  установить голубой цвет фона  
 47                  установить белый цвет фона  
 49                  установить цвет фона по умолчанию

#### **ECMA-48 Переключение режимов**

ESC [ 3 h                    DECCRM (по умолчанию выключено) : Показывать  
 управляющие символы.  
 ESC [ 4 h                    DECIM (по умолчанию выключено) : Установить режим  
 вставки/замены.  
 ESC [ 20 h                  LF/NL (по умолчанию выключено) : Автоматически  
 выводить код CR после кодов: LF, VT или FF.

#### **ECMA-48 Команды сообщения статуса**

ESC [ 5 n                  Сообщение о статусе устройства (DSR) : Ответом  
 является ESC [ 0 n (Терминал в порядке).  
 ESC [ 6 n                  Сообщение о позиции курсора (CPR) : Ответом является  
 ESC [ y ; x R, где x,y являются координатами  
 курсора.

#### **DEC-последовательности частного режима (DECSET/DECRST)**

Не описаны в ECMA-48. Далее будут перечислены последовательности установки режимов; в последовательности для сброса режимов последний знак 'h' заменяется на 'l'.

ESC [ ? 1 h                DECCKM (по умолчанию выключено) : Если включено, то  
 клавиши курсора выдают сигнал, начинающийся с ESC  
 O, а не с ESC [.

ESC [ ? 3 h  
     DEC COLM (по умолчанию выключено = 80 столбцов):  
     режим переключения количества столбцов на 80 или  
     132. В исходных версиях драйвера задано, что одной  
     этой команды недостаточно; некоторые  
     пользовательские утилиты, использующие  
     нестандартные режимы, такие, как **resizecons(8)**,  
     должны менять регистры настройки консольной  
     видеокарты.  
 ESC [ ? 5 h  
     DEC SCNM (по умолчанию выключено): Включить  
     инвертированный видеорежим.  
 ESC [ ? 6 h  
     DECOM (по умолчанию выключено): Если включено, то  
     координаты курсора рассматриваются относительно  
     верхнего левого угла в области прокрутки.  
 ESC [ ? 7 h  
     DEC AWM (по умолчанию включено): Включить режим  
     автопереноса. В этом режиме графический символ,  
     вводящийся после столбца #80 (или #132, если DEC-  
     COLM включено), переносится в начало следующей  
     строки.  
 ESC [ ? 8 h  
     DEC ARM (по умолчанию включено): Включить режим  
     автоповтора символов при нажатии клавиши  
     клавиатуры.  
 ESC [ ? 9 h  
     X10 Отчет о состоянии мыши (по умолчанию  
         выключено): Установить режим отчета о состоянии  
         мыши, равным единице или нулю (сброс), см. ниже.  
 ESC [ ? 25 h  
     DEC CM (по умолчанию включено): Сделать курсор  
         видимым.  
 ESC [ ? 1000 h  
     X11 Отчет о состоянии мыши (по умолчанию  
         выключено): Установить режим отчета о состоянии  
         мыши, равным 2-м или нулю (сброс), см. ниже.

#### **Частные последовательности CSI консоли Linux**

Следующие последовательности не являются ни ECMA-48-, ни VT102-последовательностями. Они являются "родными" для драйвера консоли Linux. Цвета в параметрах SGR: 0 = черный, 1 = красный, 2 = зеленый, 3 = коричневый, 4 = синий, 5 = сиреневый, 6 = голубой, 7 = белый.

ESC [ 1 ; n ]                 Установить цвет n как цвет подчеркивания  
 ESC [ 2 ; n ]                 Установить цвет n как цвет затемнения  
 ESC [ 8 ]                     Установить текущую пару цветов атрибутами по  
     умолчанию.  
 ESC [ 9 ; n ]                 Установить начало "очистки" экрана через n минут.  
 ESC [ 10 ; n ]                Установить частоту звукового сигнала (в герцах).  
 ESC [ 11 ; n ]                Установить длительность звукового сигнала в  
     ( миллисекундах).  
 ESC [ 12 ; n ]                Переместить указанную консоль впереди остальных.  
 ESC [ 13 ]                    Восстановить экран после "очистки".  
 ESC [ 14 ; n ]                Установить VESA-интервал отключения (в минутах).

#### **НАБОРЫ СИМВОЛОВ**

Ядро располагает информацией о 4-х типах перевода байтов в символы консоли экрана. Эти четыре таблицы являются трансформирующими: a) Latin1 -> PC, b) графика VT100 -> PC, c) PC -> PC, d) определяется пользователем. В системе вообще существует два набора символов, называемых G0 и G1, и один из них является текущим набором системы (изначально это G0). Ввод ^N заставляет набор G1 стать текущим, ^O делает текущим набор G0. Переменные G0 и G1 указывают на таблицы символов и могут меняться пользователем.

Изначально они указывают на таблицы а) и б) соответственно. Последовательности ESC ( В и ESC ( 0 и ESC ( U и ESC ( К заставляют G0 ссылаться на таблицы перевода а), б), с) и д) соответственно. Последовательности ESC ) В и ESC ) 0 и ESC ) U и ESC ) К заставляют G1 ссылаться на таблицы перевода а), б), с) и д) соответственно. Последовательность ESC с приводит к сбросу терминала, то есть к тому, что Вам необходимо при наличии "мусора" на экране. Рекомендация, указанная после "echo ^V^O", только сделает G0 текущим набором, но нет никакой гарантии, что G0 будет указывать на таблицу а). В некоторых версиях есть программа **reset(1)**, которая всего лишь выполняет команду "echo ^[c]". Если элемент terminfo является правильным для консоли (и содержит элемент rs1=\Ec), то "tput reset" также будет выполнять операции. Определенная пользователем таблица распределения может быть установлена с помощью **mapscrn(8)**. Результатом распределения будет то, что при указании символа с в видеопамять будет отправлен символ s = map[c]. Побитное изображение, соответствующее s, имеется в ПЗУ символов и может меняться при помощи **setfont(8)**.

#### РАБОТА С МЫШЬЮ

Система для отслеживания работы мыши предполагает выдачу xterm-совместимых сигналов о статусе мыши. Так как драйвер консоли не распознает такого устройства, как мышь (или ее типа), то эти сигналы отправляются во входной поток консоли, только когда драйвер виртуального терминала получает сигнал ioctl об обновлении статуса мыши. Эти сигналы ioctls должны генерироваться пользовательскими приложениями, поддерживающими работу с мышью (такими, как демон **gpm(8)**). Параметром для всех созданных xterm ESC-последовательностей (сигналов о перемещении мыши) будет один символ, код которого равен value+040. Например, '!' соответствует единице. Система координат экрана при отслеживании работы мыши базирована на знаке единицы. В режиме совместимости с X10 при нажатии на кнопки посыпаются ESC-последовательности, в которых кодируется и состояние мыши, и информация о нажатой кнопке. Режим запускается при выдаче ESC [ ? 9 h и выключается при выдаче ESC [ ? 9 l. При нажатии на кнопку xterm посылает ESC [ M bxy (6 символов), где b - это кнопка-1, а x и у равны координатам x и y при нажатии на эту кнопку. Это такие же коды, какие генерируют и выдают ядро системы. В обычном режиме отслеживания мыши (который не был реализован в Linux 2.0.24) ESC-последовательности посыпаются при нажатии и при отпускании кнопки мыши. Также посыпается информация о модификаторе. Запускается режим при выдаче ESC [ ? 1000 h и выключается при выдаче ESC [ 1000 l. При нажатии и отпускании кнопки xterm выдает ESC [ M bxy. Два нижних бита b содержат информацию о кнопках: 0=MB1 нажата, 1=MB2 нажата, 2=MB3 нажата, 3=отпущена. Старшие биты содержат информацию о том, какие модификаторы были выключены когда кнопка была нажата. Эта информация складывается при нажатии: 4=Shift, 8=Meta, 16=Control. Еще раз: x и y являются координатами x и y мыши при обработке события. Координаты верхнего левого угла рассматриваются как (1,1).

#### СРАВНЕНИЕ С ДРУГИМИ ТЕРМИНАЛАМИ

Множество других типов терминалов описаны наподобие консоли Linux (как 'VT100-совместимые'). Далее мы обсудим различия между Linux-консолью и двумя другими важнейшими типами, DEC VT102 и **xterm(1)**.

#### Обработка управляемых символов

Тип vt102 также распознает следующие управляемые символы:

MAN PAGES 4.Специальные файлы

NUL (0x00) был проигнорирован;  
ENQ (0x05) запущено обратное ответное сообщение;  
DC1 (0x11, ^Q, XON) продолженная передача;  
DC3 (0x13, ^S, XOFF) указывает, что vt100 необходимо  
игнорировать и останавливать передачу всех кодов за  
исключением XOFF и XON  
VT100-подобная DC1/DC3 обработка может быть запущена  
драйвером tty.  
Программа xterm (в режиме vt100) распознает управляющие  
следующие символы: BEL, BS, HT, LF, VT, FF, CR, SO, SI,  
ESC.

#### **ESC-последовательности**

VT100-последовательности консоли, не реализованные в  
консоли Linux:

ESC N	SS2	Сдвиг на 2. (Выбрать набор символов G2 только для следующего символа.)
ESC O	SS3	Сдвиг на 3. (Выбрать набор символов G3 только для следующего символа.)
ESC P	DCS	Строка управления устройством (заканчивается на ESC \)
ESC X	SOS	Начало строки.
ESC ^	PM	Частное сообщение (заканчивается на ESC \\)
ESC \	ST	Завершающий 0 строки
ESC * ...		Определить набор символов G2
ESC + ...		Определить набор символов G3
Программа xterm (в режиме vt100) распознает: ESC c, ESC # 8, ESC >, ESC =, ESC D, ESC E, ESC H, ESC M, ESC N, ESC O, ESC P ... ESC Z (оно выдает последовательности ESC [ ? 1 ; 2 с ответ: `Я являюсь vt100 с улучшенными видеопараметрами' и ESC ^ ... ESC с теми же значениями, какие указаны выше. Принимаются: ESC (, ESC ), ESC *, ESC +,- сопровождаемые 0, A, B для специальных символов DEC и режима рисования линий, UK и USASCII соответственно. Принимаются ESC ] для установки работы таких ресурсов, как:		
ESC ] 0 ; txt BEL		Установить имя иконки и заголовок окна в режиме txt.
ESC ] 1 ; txt BEL		Установить имя иконки в режиме txt.
ESC ] 2 ; txt BEL		Установить заголовок окна в режиме txt.
ESC ] 4 6 ; name BEL		Сменить файл логов на name (обычно
отключено		
параметрами компиляции)		
ESC ] 5 0 ; fn BEL		Установить шрифт fn.
Распознаются следующие параметры с несколько измененным значением:		
ESC 7 DECSC		Сохранить курсор
ESC 8 DECRC		Восстановить курсор
Также распознается		
ESC F		Курсор в нижний левый угол
экрана (если разрешено ресурсом hpLowerleftBugCompat)		
ESC 1		Блокирование памяти (для HP- терминалов).
курсора.		Блокирует память выше
ESC m		Разблокирование памяти (для HP-терминалов).
ESC n	LS2	Ввод набора символов G2.
ESC o	LS3	Ввод набора символов G3.
ESC	LS3R	Ввод набора символов G3 в качестве GR.

xterm.		Не имеет видимого эффекта в
ESC }	LS2R	Ввод набора символов G2 в
качестве GR.		Не имеет видимого эффекта в
xterm.		
ESC ~	LS1R	Ввод набора символов G1 в
качестве GR.		Не имеет видимого эффекта в
xterm.		
He распознается ESC % ...		

#### **CSI-последовательности**

Программа *xterm* (по данным XFree86 3.1.2G) не распознает мерцание или режим невидимости SGRs. Версии X11R6 не распознают цветоустановки SGR. Все другие ECMA-48 CSI-последовательности, распознаваемые Linux, также распознаются *xterm* и наоборот. Программа *xterm* будет распознавать все последовательности частного режима DEC, перечисленные выше, но не будут распознавать ни одну из последовательностей частного режима в Linux. Для обсуждения последовательностей частного режима для *xterm* обратитесь к документу *XtermControlSequences*, созданного Edward Moy и Stephen Gildea и сопровождаемого дистрибутивом X Window System.

#### **НАЙДЕННЫЕ ОШИБКИ**

В версии 2.0.23 набор CSI является неправильным и недейственным, NUL не игнорируется внутри ESC-последовательностей.

**СМ. ТАКЖЕ** **console(4)**, **console\_ioctl(4)**, **charsets(7)**

Linux

October 31, 1996

CONSOLE\_CODES(4)

## **CONSOLE\_IOCTL**

**НАЗВАНИЕ** **console ioctl** - ioctl для консольных терминалов и виртуальных консолей

**ОПИСАНИЕ ВНИМАНИЕ:** Вы используете эту информацию на свой страх и риск. **ВНИМАНИЕ:** ioctl - неописанные ранее возможности Linux, изменяются без предупреждения. Пользуйтесь функциями POSIX. Поддерживаются Linux-ориентированные запросы **ioctl()**. Каждый запрос определяется третьим аргументом, далее именуемым *argp*.

#### **KDGETLED**

Получить состояние индикаторов. *argp* указывает на длинное целое (long int). Младшие три бита \**argp* устанавливают состояние индикаторов следующим образом:

LED_CAP	0x04	индикатор caps lock
LEC_NUM	0x02	индикатор num lock
LED_SCR	0x01	индикатор scroll lock

#### **KDSETLED**

Устанавливает состояние индикаторов. Состояние индикаторов устанавливается в соответствии с тремя младшими битами *argp*. Однако, если установлен старший бит, индикаторы возвращаются в исходное состояние: отображают статус функций клавиатуры caps lock, num lock и scroll lock.

До появления ядра 1.1.54 индикаторы состояния отражали только состояние флагов клавиатуры, которые также могли быть изменены KDGETLED/KDSETLED. После появления 1.1.54 они могли отображать уже произвольную информацию, но по умолчанию показывали флаги клавиатуры. Описанные ниже два запроса ioctl используются для доступа к флагам клавиатуры.

#### **KDGKBLED**

Считывает флаги клавиатуры: CapsLock, NumLock, ScrollLock (не индикаторы). *argp* указывает на символ (*char*), который получает состояния флагов. Младшие три бита (маска 0x7) получают текущее состояние флагов, следующие три бита (маска 0x70) присваивают устанавливаемое состояние флагам по умолчанию. (Со времени появления ядра 1.1.54).

#### KDSKBLED

Устанавливает флаги клавиатуры: CapsLock, NumLock, ScrollLock (не индикаторы). *argp* содержит требуемые состояния флагов. Три младших бита (маска 0x7) содержит состояния флагов, следующие три бита (маска 0x70) содержат состояния флагов по умолчанию. (Со времени появления ядра 1.1.54).

#### KDGKBVTYPE

Считывает тип клавиатуры. Возвращается значение, равное KB\_101, определено как 0x02.

#### KDADDIO

Добавляет порт ввода/вывода (I/O) как допустимый. Эквивалентно вызову ioperm(arg,1,1).

#### KDDELIO

Удаляет порт ввода/вывода (I/O) как недопустимый. Эквивалентно вызову ioperm(arg,1,0).

#### KDENABIO

Разрешает ввод/вывод на видеокарту. Эквивалентно вызову ioperm(0x3b4, 0x3df-0x3b4+1, 1).

#### KDDISABIO

Запрещает ввод/вывод на видеокарту. Эквивалентно вызову ioperm(0x3b4, 0x3df-0x3b4+1, 0).

#### KDSETMODE

Установить текстовый/графический режим. *argp* содержит одно из двух значений:

KD\_TEXT 0x00  
KD\_GRAPHICS 0x01

#### KDGETMODE

Устанавливает режим (текстовый/графический). *argp* указывает на длинное целое, которому присваивается одно из вышеперечисленных устанавливаемых значений.

#### KDMKTONE

Генерирует тон заданной длительности. Младшие 16 битов *argp* задают частоту тона (период в тактах), старшие 16 битов устанавливают его длительность в мсек. Если длительность равна нулю, звук выключается. Управление возвращается немедленно. Например, *argp* = (125<<16) + 0x637 определяет звук, обычно связанный с ctrl-G. (Работает с 0.99pl1; не работает в 2.1.49-50).

#### KIOCSOUND

Начинает и останавливает генерацию звука. Младшие 16 битов *argp* определяют период в тактах (то есть *argp* = 1193180/частота). Значение *argp* = 0 выключает звук. В обоих случаях управление возвращается немедленно.

#### GIO\_CMAP

Считывает текущую цветовую карту по умолчанию из ядра. *argp* указывает на массив из 48-и байтов. (Со времени появления ядра 1.3.3).

#### PIO\_CMAP

Изменяет цветовую карту по умолчанию для текстового режима. *argp* указывает на массив из 48-и байтов, который содержит, по порядку, значения красного, зеленого и синего для 16-и доступных экранных цветов: 0 - составляющая выключена, 255 - полная

интенсивность. Цвета по умолчанию, по порядку: black, dark red, dark green, brown, dark blue, dark purple, dark cyan, light grey, dark grey, bright red, bright green, yellow, bright blue, bright purple, bright cyan и white. (Со времени появления ядра 1.3.3).

#### **GIO\_FONT**

Получить экранный шрифт (256 символов) в расширенной форме. *argp* указывает на массив из 8192 байтов. Возвращает код ошибки **EINVAL**, если текущий загруженный шрифт содержит 512 символов или консоль находится не в текстовом режиме.

#### **GIO\_FONTX**

Считывает экранный шрифт и связанную с ним информацию. *argp* указывает на структуру *consolefontdesc* (смотри **PIO\_FONTX**). При вызове поле *charcount* должно иметь максимальное число символов в качестве установленного значения, которое помещается в буфер, указываемый как *chardata*. При возврате *charcount* и *charheight* содержат полученную информацию о текущем загруженном шрифте, массив *chardata* содержит данные шрифта, если начальное значение *charcount* показывало доступное пространство; в противном случае буфер остается неизменным и установленное значение *errno* равно **ENOMEM**. (Со времени появления ядра 1.3.1).

#### **PIO\_FONT**

Устанавливает экранный шрифт в объеме 256 символов. Шрифт загружается в знакогенератор EGA/VGA. *argp* указывает на карту размером 8192 байта (32 байта на символ). Только первые *N* из них используются для шрифта 8x*N* ( $0 < N \leq 32$ ). Этот вызов также отменяет установленные соответствия символов Unicode шрифту.

#### **PIO\_FONTX**

Устанавливает экранный шрифт и соответствующую информацию для изображения. *argp* указывает на структуру:

```
struct consolefontdesc {  
    u_short charcount; /* число символов в шрифте (256 или 512)  
 */  
    u_short charheight; /* число линий сканирования символа (1-  
 32) */  
    char *chardata; /* данные шрифта в расширенной форме */  
};
```

Если требуется, соответственно изменяются размеры экрана и **SIGWINCH** посыпается соответствующим процессам. Этот вызов также отменяет установленные соответствия символов Unicode и шрифта. (Со времени появления ядра 1.3.1).

#### **PIO\_FONTRESET**

Пересыпает экранный шрифт, размер и соответствие символов Unicode шрифту в установки по умолчанию при загрузке. *argp* не используется, но установленное значение ее должно быть **NULL**, чтобы эта версия была совместима с будущими версиями Linux. (Со времени появления ядра 1.3.28).

#### **GIO\_SCRNMAP**

Считывает разметку экрана из ядра. *argp* указывает на область размером *E\_TABSZ*, которая заполняется позициями символов шрифта, используемыми при отображении. Вызов возвращает бесполезную информацию, если текущий загруженный шрифт содержит

более 256 символов.

#### **GIO\_UNISCRNMAP**

Считывает полное экранное соответствие символов Unicode шрифту из ядра. *argp* указывает на область размером `E_TABSZ*sizeof(unsigned short)`, которая заполняется представлением в Unicode каждого символа. Специальный набор Unicode, начинающийся с `U+F000`, используется для представления перевод символов ``напрямую в шрифт''. (Со временем появления ядра 1.3.1).

#### **PIO\_SCRNMAP**

Загружает ``определенную пользователем'' (четвертую) таблицу, которая отображает байты в символах на экране консоли из ядра. *argp* указывает на область размером `E_TABSZ`.

#### **PIO\_UNISCRNMAP**

Загружает ``определенную пользователем'' (четвертую) таблицу из ядра, которая переводит байты в символы Unicode, преобразующиеся затем в экранные символы согласно текущей загруженной карте соответствия символов Unicode и шрифта. Специальные коды Unicode, начинающиеся с `U+F000`, могут использоваться для прямого перевода байтов в символы шрифта. (Со временем появления ядра 1.3.1).

#### **GIO\_UNIMAP**

Проверяет соответствие символов Unicode шрифту из ядра. *argp* указывает на структуру

```
struct unimapdesc {
    u_short entry_ct;
    struct unipair *entries;
};
```

где *entries* указывает на массив структур

```
struct unipair {
    u_short unicode;
    u_short fontpos;
};
```

(Со временем появления ядра 1.1.92).

#### **PIO\_UNIMAP**

Помещает соответствие символов Unicode и экранного шрифта в ядро. *argp* указывает на структуру *unimapdesc*. (Со временем появления ядра 1.1.92).

#### **PIO\_UNIMAPCLR**

Сбрасывает таблицу знакогенератора (возможно использовать алгоритм хэширования). *argp* указывает на структуру

```
struct unimapinit {
    u_short advised_hashsize; /* 0, если не использовать */
    u_short advised_hashstep; /* 0, если не использовать */
    u_short advised_hashlevel; /* 0, если не использовать */
};
```

(Со временем появления ядра 1.1.92).

#### **KDGKBMODE**

Считывает текущий режим клавиатуры. *argp* указывает на длинное целое (`long`), устанавливаемое значение которого равно одному из:

K_RAW	0x00
K_XLATE	0x01
K_MEDIUMRAW	0x02
K_UNICODE	0x03

#### **KDSKBMODE**

Устанавливает текущий режим клавиатуры. *argp* - длинное целое (`long`), равное одному из значений, указанных выше.

**KDGKBMETA**

Считывает режим обработки метаклавиш. *argp* указывает на длинное целое (long), устанавливаемое значение которого равно одному из:  
K\_METABIT 0x03 установить старший бит  
K\_ESCPREFIX 0x04 префикс escape

**KDSKBMETA**

Устанавливает режим обработки метаклавиш. *argp* - длинное целое (long), равное одному из значений, указанных выше.

**KDGKBENT**

Считывает одну запись из таблицы трансляции клавиш (код клавиши для кода действия). *argp* указывает на структуру

```
struct kbentry {  
    u_char kb_table;  
    u_char kb_index;  
    u_short kb_value;  
};
```

с двумя первыми полями, установленные значения которых представляют собой: *kb\_table* - выбранную таблицу клавиш ( $0 \leq kb\_table < MAX\_NR\_KEYMAPS$ ), *kb\_index* - код клавиши ( $0 \leq kb\_index < NR\_KEYS$ ). *kb\_value*, которой присваивается соответствующий код действия или K\_HOLE, если нет такой клавиши, или K\_NOSUCHMAP, если значение *kb\_table* неверно.

**KDSKBENT**

Делает запись в таблице трансляции клавиш. *argp* указывает на структуру *kbentry*.

**KDGKBSENT**

Считывает запись строки одной функциональной клавиши. *argp* указывает на структуру

```
struct kbsentry {  
    u_char kb_func;  
    u_char kb_string[512];  
};
```

*kb\_string* равна строке, заканчивающейся нулем и соответствующей коду действия функциональной клавиши *kb\_func*.

**KDSKBSENT**

Делает запись строки одной функциональной клавиши. *argp* указывает на структуру *kbsentry*.

**KDGKBDIACR**

Считывает таблицу акцентов ядра. *argp* указывает на структуру

```
struct kbdiacrs {  
    unsigned int kb_cnt;  
    struct kbdiacr kbdiacr[256];  
};
```

где *kb\_cnt* - это число записей в массиве, каждая из которых является структурой struct kbdiacr { u\_char diacr, base, result; };

**KDGETKEYCODE**

Считывает запись таблицы кодов клавиш ядра (сканкоды - коды клавиш). *argp* указывает на структуру

```
struct kb keycode { unsigned int scancode, keycode; };
```

*keycode* устанавливается в соответствии с заданным *scancode*. (Допускается  $89 \leq scancode \leq 255$ . Для сканкодов  $1 \leq scancode \leq 88$ , *keycode==scancode*). (Со времени появления ядра 1.1.63).

**KDSETKEYCODE**

Производит запись в таблицу кодов клавиш ядра. *argp*

MAN PAGES 4.Специальные файлы

указывает на структуру kb keycode. (Со времени появления ядра 1.1.63).

#### KDSIGACCEPT

Вызываемый процесс показывает свою готовность к приему сигнала *argp*, если он генерируется нажатием соответствующей комбинации клавиш. ( $1 \leq argp \leq NSIG$ ). (Смотри функцию spawn\_console() в linux/drivers/char/keyboard.c).

#### VT\_OPENQRY

Возвращает первую доступную (неоткрытую) консоль. *argp* указывает на целое (int), устанавливаемое значение которого равно номеру виртуального терминала ( $1 \leq *argp \leq MAX_NR_CONSOLES$ ).

#### VT\_GETMODE

Считывает режим активного виртуального терминала.

*argp* указывает на структуру

```
struct vt_mode {
    char mode;      /* режим терминала */
    char waitv;     /* при установленном; остановка при записи,
если терминал неактивен */
    short relsig;   /* сигнал для восстановления при освобождении */
*/           short acqsig;   /* сигнал для восстановления при захвате */
    short frsig;    /* не используется (устанавливается равным
нулю) */
};
```

*mode* приобретает одно из устанавливаемых значений:

VT_AUTO	автоматическое переключение виртуального терминала
VT_PROCESS	переключение контролирует процесс
VT_ACKACQ	подтверждение переключения

#### VT\_SETMODE

Устанавливает режим активного виртуального терминала. *argp* указывает на структуру *vt\_mode*.

#### VT\_GETSTATE

Считывает общую информацию о состоянии виртуального терминала. *argp* указывает на структуру

```
struct vt_stat {
    ushort v_active; /* активный виртуальный терминал */
    ushort v_signal; /* посыпаемый сигнал */
    ushort v_state;  /* битовая маска виртуального терминала */
*/
```

Для каждого используемого виртуального терминала устанавливается соответствующий бит в поле *v\_state*. (В версиях с 1.0 до 1.1.92).

#### VT\_RELISP

Освобождает дисплей.

#### VT\_ACTIVATE

Переключается на виртуальный терминал *argp* ( $1 \leq argp \leq MAX_NR_CONSOLES$ ).

#### VT\_WAITACTIVE

Ожидает, пока виртуальный терминал *argp* не станет активным.

#### VT\_DISALLOCATE

Освобождает выделенную виртуальному терминалу *argp* память. (Со времени появления ядра 1.1.54).

#### VT\_RESIZE

Устанавливает представление размера экрана в ядре.

*argp* указывает на структуру

```
struct vt_sizes {
    ushort v_rows;      /* число строк */
    ushort v_cols;      /* число колонок */
};
```

MAN PAGES 4.Специальные файлы

```

        ushort v_scrollsize; /* больше не используется */
};

Этот вызов не изменяет видеорежим. Смотрите руководство resizecons(8). (Со времени появления ядра 1.1.54).

VT_RESIZEX
Устанавливает значение различных параметров экрана в ядре. argp указывает на структуру
struct vt_consizer {
    ushort v_rows; /* число строк */
    ushort v_cols; /* число колонок */
    ushort v_vlin; /* число строк экрана в пикселях */
    ushort v_clin; /* число строк в пикселях на символ */
    ushort v_vcol; /* число колонок экрана в пикселях */
    ushort v_ccol; /* число колонок в пикселях на символ */
};

Любому параметру может быть присвоено установленное нулевое значение, то есть он может быть оставлен ``без изменений'', но, если установлено несколько параметров, они должны быть согласованы. Этот вызов не изменяет видеорежим. Смотрите руководство resizecons(8). (Со времени появления ядра 1.3.3).

Действие следующих ioctl зависит от первого байта структуры, указанной argp, далее называемого subcode. Доступны только суперпользователю или владельцу текущего tty.

TIOCLINUX, subcode=0
Выгружает дамп экрана. Исчезло в 1.1.92. (В ядре 1.1.92 и более поздних версий используется чтение из /dev/vcsN или /dev/vcsaN).

TIOCLINUX, subcode=1
Считывает информацию о задании. Исчезло в 1.1.92.

TIOCLINUX, subcode=2
Производит выделение блока. argp указывает на структуру
struct {char subcode;
         short xs, ys, xe, ye;
         short sel_mode;
     } xs и ys - начальные колонка и строка. xe и ye - конечные колонка и строка. (Левый верхний угол - строка = колонка = 1). sel_mode равен нулю для выделения "символ за символом", единице для выделения "слово за словом" или двум для выделения "строки за строкой". Выделенные символы подсвечиваются и сохраняются в статическом массиве sel_buffer, определенном в devices/char/console.c.

TIOCLINUX, subcode=3
Вставляет выделенный блок. Символы в буфере выделения записываются в fd.

TIOCLINUX, subcode=4
Выводит экран из режима энергосбережения.

TIOCLINUX, subcode=5
Устанавливает содержимое 256-битной таблицы определения символов в "word" для выделения "слово за словом". (Со времени появления ядра 1.1.32).

TIOCLINUX, subcode=6
argp указывает на символ (char), который устанавливает значение переменной ядра shift_state. (Со времени появления ядра 1.1.32).

TIOCLINUX, subcode=7
argp указывает на символ (char), который устанавливает значение переменной ядра report_mouse. (Со времени появления ядра 1.1.33).

```

**TIOCLINUX, subcode=8**

Выгружает ширину и высоту экрана, позицию курсора и все пары символ-атрибут. (Только в версиях с 1.1.67 по 1.1.91. В ядре 1.1.92 и более поздних версий используется чтение из /dev/vcsa\*).

**TIOCLINUX, subcode=9**

Восстанавливает ширину и высоту экрана, позицию курсора и все пары символ-атрибут. (Только в версиях с 1.1.67 по 1.1.91. В ядре 1.1.92 и более поздних версий используется запись в /dev/vcsa\*).

**TIOCLINUX, subcode=10**

Обработчик функций энергосбережения для нового поколения мониторов. Режим погашения экрана VESA устанавливается равным *argp[1]*, который определяет тип гашения:

0: гашение экрана запрещено.

1: текущие установки регистров видеоадаптера сохраняются, когда контроллер программируется на выключение вертикальной синхронизации. Происходит перевод монитора в режим "резерв". Если на мониторе есть таймер Off\_Mode, то он может в итоге сам выключить питание.

2: текущие установки сохраняются, когда и вертикальная, и горизонтальная развертки отключаются. Происходит перевод в режим "выключено". Если на мониторе нет таймера Off\_Mode или Вы хотите отключить питание сразу же по истечении времени blank\_timer, Вы можете выбрать эту опцию. (Внимание: частое выключение питания может повредить монитор.) (Со времени появления ядра 1.1.76).

**ВОЗВАЩАЕМЫЕ ЗНАЧЕНИЯ**

При ошибке возвращаемое значение -1, а переменной *errno* присваивается значение соответствующих кодов ошибок.

**КОДЫ ОШИБОК**

*errno* может принимать следующие значения:

**EBAADF** (неверный описатель файла);

**ENOTTY** (описатель файла не связан со специальным символьным устройством, или заданный запрос не может быть выполнен);

**EINVAL** (описатель файла или *argp* неверен);

**EPERM** (ошибка доступа).

**ПРЕДУПРЕЖДЕНИЕ**

Не рассматривайте эту страницу руководства как документацию об ioctl консоли Linux. Она предназначена только для любознательных как альтернатива исходных текстов. Ioctl - неописанные ранее возможности Linux, которые могут изменяться без предупреждения. На самом деле, это руководство более или менее полно описывает параметры ядра версии 1.1.94; они несколько отличаются от параметров ядра более ранних версий. Очень часто ioctl вводится для совместной работы ядра с какой-либо конкретной программой (fdisk, hdparm, setserial, tunelp, loadkeys, selection, setfont и т.д.), и его поведение изменяется по требованию этой программы. Программы, использующие такие ioctl, не могут быть перенесены в другие версии Unix, не будут работать в старых версиях Linux и могут не работать в будущих версиях Linux. Рекомендуется работать с функциями POSIX.

**СМ. ТАКЖЕ** **kbd\_mode(1)**, **loadkeys(1)**, **dumpkeys(1)**, **mknod(1)**, **setleds(1)**, **setmetamode(1)**, **ioperm(2)**, **execve(2)**, **fcntl(2)**, **termios(3)**, **console(4)**, **console\_codes(4)**, **mt(4)**, **sd(4)**, **tty(4)**, **ttys(4)**, **vcs(4)**, **vcsa(4)**, **charsets(7)**, **map-**

```
scrn(8),           setfont(8),           resizecons(8),
/usr/include/linux/kd.h, /usr/include/linux/vt.h
Linux          September 18, 1995      CONSOLE_IOCTL(4)
```

## DSP56K

**НАЗВАНИЕ**      `dsp56k` - устройство интерфейса DSP56001

**СИНТАКСИС** `ssize_t read(int fd, void *data, size_t length);`  
`ssize_t write(int fd, void *data, size_t length);`  
`int ioctl(int fd, DSP56K_UPLOAD, struct dsp56k_upload *program);`  
`int ioctl(int fd, DSP56K_SET_TX_WSIZE, int wsize);`  
`int ioctl(int fd, DSP56K_SET_RX_WSIZE, int wsize);`  
`int ioctl(int fd, DSP56K_HOST_FLAGS, struct dsp56k_host_flags *flags);`  
`int ioctl(int fd, DSP56K_HOST_CMD, int cmd);`

### КОНФИГУРАЦИЯ

Устройство `dsp56k` является символьным устройством с главным числом 55 и второстепенным числом 0.

**ОПИСАНИЕ** Motorola DSP56001 - это полностью программируемый 24-битный цифровой процессор сигналов используемый в компьютерах Atari Falcon030 и совместимых с ними. Специальный файл `dsp56k` используется для управления DSP56001 и для посылки и получения данных, используя двунаправленный порт.

Для того чтобы послать данные процессору сигналов, используйте `write()`, а для получения обработанных данных `read()`. Данные могут быть посланы и получены машиной в 8, 16, 24, или 32-битной форме, а процессором сигналов только в 24-битной форме.

Следующие вызовы `ioctl(2)` используются для управления устройством `dsp56k`:

#### DSP56K\_UPLOAD

обнуляет DSP56001 и загружает программу. Третьим аргументом `ioctl()` должен быть указатель на **struct** `dsp56k_binary` с членами `bin` указывающими на двоичную программу DSP56001, и `len` установленную на длину программы сосчитанную кратно 24 битам.

#### DSP56K\_SET\_TX\_WSIZE

устанавливает размер передаваемых слов. Допустимые значения находятся в пределах от 1 до 4, и будет являться числом байт посылаемых за раз для DSP56001. Эти частички информации будут заполнены нулевыми байтами или преобразованы в формат 24-битных данных.

#### DSP56K\_SET\_RX\_WSIZE

устанавливает размер принимаемых слов. Допустимые значения находятся в диапазоне от 1 до 4, и являются числом байт принимаемых за один раз от процессора DSP56001. Эти частички информации также будут обработаны, чтобы соответствовать 24-битному формату DSP56001.

#### DSP56K\_HOST\_FLAGS

Считывает и записывает флаги машины. Флаги машины это четыре бита так сказать общего назначения, которые могут быть считаны как машиной, так и процессором DSP56001. Биты первый и нулевой пишутся машиной, а второй и третий пишутся устройством DSP56001. Для получения доступа к флагам машины третий аргумент `ioctl()` должен быть указателем на **struct** `dsp56k_host_flags`. Если первый или нулевой бит переменной `dir` установлены в единичное значение, то соответствующий бит в переменной `out` будет записан в флаг машины. Значение всех флагов машины будет возвращено в

младшие четыре бита переменной **status**.

**DSP56K\_HOST\_CMD**

посыпает команды. Допустимые величины находятся в диапазоне от 0 до 31, они могут быть определены пользователем.

**ФАЙЛЫ** /dev/dsp56k

**АВТОРЫ** Fredrik Noring <noring@nocrew.org>, lars brinkhoff <lars@nocrew.org>, Tomas Berndtsson <tomas@nocrew.org>.

**СМ. ТАКЖЕ** linux/include/asm-m68k/dsp56k.h,  
linux/drivers/char/dsp56k.c, http://dsp56k.nocrew.org/,  
DSP56000/DSP56001 Digital Signal Processor User's Manual

Linux March 1, 2000

DSP56K(4)

---

**FD**

**НАЗВАНИЕ** fd - накопитель на гибких дисках

**КОНФИГУРАЦИЯ**

Накопители на гибких дисках являются блочными устройствами с главным числом 2. Обычно владельцем является root.floppy (т.е. пользователь root, группа floppy) с правами 0660 (доступ проверяется через проверку группы пользователей) или с правами 0666 (доступ имеют все). Второстепенные числа представляют тип устройства, номер устройства и номер контроллера. Для каждого типа устройства (это плотность и число дорожек) есть основной второстепенный номер. К этому базовому числу добавьте номер накопителя и 128 если накопитель находится на втором контроллере. В нижеследующих таблицах, *n* FP - это номер накопителя.

**Внимание:** Если вы используете форматы с большим количеством дорожек чем поддерживает ваш накопитель, то вы можете механически его повредить.

Независимые от накопителя файлы автоматически определяют формат и емкость:

Имя Осн. второстепенное число

---

**fdn** 0

Файлы для 5.25 дюймовых дисков двойной плотности:

Имя	Объем	Цил.	Сект.	Голов.	Осн. второстепенное число
-----	-------	------	-------	--------	---------------------------

**fdnd360** 360K 40 9 2 4

Файлы для 5.25 дюймовых дисков повышенной плотности:

Имя	Объем	Цил.	Сект.	Голов.	Осн. второстепенное число
-----	-------	------	-------	--------	---------------------------

**fdnh360** 360K 40 9 2 20

**fdnh410** 410K 41 10 2 48

**fdnh420** 420K 42 10 2 64

**fdnh720** 720K 80 9 2 24

**fdnh880** 880K 80 11 2 80

**fdnh1200** 1200K 80 15 2 8

**fdnh1440** 1440K 80 18 2 40

**fdnh1476** 1476K 82 18 2 56

**fdnh1494** 1494K 83 18 2 72

**fdnh1600** 1600K 80 20 2 92

Файлы для 3.5 дюймовых дисков двойной плотности:

Имя	Объем	Цил.	Сект.	Голов.	Осн. второстепенное число
-----	-------	------	-------	--------	---------------------------

**fdnD360** 360K 80 9 1 12

**fdnD720** 720K 80 9 2 16

**fdnD800** 800K 80 10 2 120

**fdnD1040** 1040K 80 13 2 84

**fdnD1120** 1120K 80 14 2 88

Файлы для 3.5 дюймовых дисков повышенной плотности:

Имя	Объем	Цил.	Сект.	Голов.	Осн. второстепенное число
-----	-------	------	-------	--------	---------------------------

**fdnh360** 360K 40 9 2 12

<b>fdnH720</b>	720K	80	9	2	16
<b>fdnH820</b>	820K	82	10	2	52
<b>fdnH830</b>	830K	83	10	2	68
<b>fdnH1440</b>	1440K	80	18	2	28
<b>fdnH1600</b>	1600K	80	20	2	124
<b>fdnH1680</b>	1680K	80	21	2	44
<b>fdnH1722</b>	1722K	82	21	2	60
<b>fdnH1743</b>	1743K	83	21	2	76
<b>fdnH1760</b>	1760K	80	22	2	96
<b>fdnH1840</b>	1840K	80	23	2	116
<b>fdnH1920</b>	1920K	80	24	2	100

Файлы для 3.5 дюймовых дисков экстра-плотности:

Имя	Объем	Цил.	Сект.	Голов.	Осн. второстепенное число
<b>fdnE2880</b>	2880K	80	36	2	32
<b>fdnCompaQ</b>	2880K	80	36	2	36
<b>fdnE3200</b>	3200K	80	40	2	104
<b>fdnE3520</b>	3520K	80	44	2	108
<b>fdnE3840</b>	3840K	80	48	2	112

**ОПИСАНИЕ** **fd** специфайлы получают доступ к накопителю напрямую.

Следующие вызовы **ioctl(2)** поддерживают устройства **fd**:

#### **FDCLRPRM**

стирает информацию о диске (о геометрии диска).

#### **FDSETPRM**

устанавливает информацию о геометрии диска. Эта информация будет утеряна, когда будет сменена дискета.

#### **FDDEFPRM**

устанавливает информацию о геометрии диска. Эта информация сохранится при смене дискеты и этим выключит автораспознавание. Для того чтобы обратно вернуть автораспознавание напишите **FDCLRPRM**.

#### **FDGETDRVTYPE**

возвращает тип накопителя (название). Для форматов работающих с несколькими типами накопителей, вызов **FDGETDRVTYPE** возвращает название соответствующее самому старому типу поддерживающему этот формат.

#### **FDFLUSH**

очищает кэш данного накопителя.

#### **FDSETMAXERRS**

устанавливает пороговое число ошибок, после которых идет сообщение об ошибке, завершение операции, сброс и чтение сектора за сектором.

#### **FDSETMAXERRS**

считывает текущее пороговое число ошибок.

#### **FDGETDRVTYPE**

считывает внутреннее название дисковода.

#### **FDERRORCLR**

очищает записи об ошибках.

#### **FDERRORGET**

считывает записи об ошибках. Это включает в себя полное количество ошибок, место и дискета первой и последней ошибок. Дискеты определяются по их номеру, который каждый раз увеличивается (при их смене).

#### **FDTWADDLE**

Выключает двигатель дисковода на несколько микросекунд. Это может потребоваться в случае, когда секторы диска расположены очень близко друг к другу.

#### **FDSETDRVPRM**

устанавливает различные параметры диска.

**FDGETDRVPRM**

считывает эти параметры.

**FDGETDRVSTAT**

считывает состояние диска (диск сменен, защищен от записи и т.д.)

**FDPOLLDRVSTAT**

запрашивает дисковод и возвращает его состояние.

**FDGETFDCSTAT**

считывает состояние контроллера.

**FDRESET**

перезапускает контроллер при определенных условиях.

**FDRAWCMD**

посыпает команду напрямую контроллеру.

Для получения более подробной информации прочтите файлы входящие в <linux/fd.h> и <linux/fdreg.h> и страницу руководства по управлению дисководом.

**ЗАМЕЧАНИЯ**

Различные форматы позволяют считывать и записывать на разные типы дисков. Если диск отформатирован с маленьким внутренним межсекторным расстоянием, то считывание может задержано на несколько секунд до того момента, когда будет считана следующая дорожка. Для того чтобы избежать этого, используйте форматы со смещенной нумерацией секторов. Нет возможности чтения дисков отформатированных GCR, использующейся компьютерами Apple II и Macintosh (800Кб). Чтение дисков которые механически разбиты на сектора (отверстие помечает сектор) не поддерживается. Это использовалось для старых 8-ми дюймовых дисков.

**ФАЙЛЫ** /dev/fd\*

**АВТОРЫ** Alain Knaff (Alain.Knaff@imag.fr), David Niemi (niemidc@clark.net), Bill Broadhurst (bbroad@netcom.com).

**СМ. ТАКЖЕ** floppycontrol(1), mknod(1), chown(1), getfdprm(1), superformat(1), mount(8), setfdprm(8)

Linux January 29, 1995

FD(4)

---

## FIFO

**НАЗВАНИЕ** fifo - спецфайл, организующий очередь (first-in first-out) и называемый каналом

**ОПИСАНИЕ** Спецфайл FIFO (называемый каналом) похож на канал, кроме того, что к нему можно осуществить доступ, как к части файловой системы. Он может быть открыт многими процессами для записи и чтения. Когда процессы обмениваются данными через FIFO, ядро передает их без записи в файловую систему. Поэтому спецфайл FIFO ничего не содержит в файловой системе. Запись в файловой системе служит процессу при организации доступа к каналу через файловую систему ссылкой.

Ядро поддерживает один канал для каждого спецфайла FIFO, который открыт хотя бы одним процессом. Для того, чтобы пропускать данные FIFO должен быть открыт как для чтения, так и для записи. Обычно при открытии FIFO он блокируется до тех пор, пока вышеописанные условия не будут выполнены. Процесс может открыть FIFO в неблокирующем режиме. В этом случае спецфайл открывается только для чтения, даже если никто не открывал его для записи; если это произойдет, то вернется сообщение об ошибке ENXIO (такого устройства или адреса не существует), хотя файл открыт для чтения.

В Linux открытие FIFO для чтения и записи может быть осуществлено в блокирующем и неблокирующем режимах. Так как POSIX не описывает эти положения, это может быть использовано для открытия FIFO для чтения в отсутствие считающих процессов. Процесс, использующий этот файл для

чтения и записи (для связи с самим собой), не должен допустить возникновения безвыходных ситуаций.

#### ЗАМЕЧАНИЯ

Если процесс попытается записать данные в файл FIFO, недоступный для чтения, то вернется сообщение об ошибке SIGPIPE. Специфайл FIFO создается **mkfifo(3)**, а отображается в **ls -l**.

**СМ. ТАКЖЕ** **mkfifo(3)**, **mkfifo(1)**, **pipe(2)**, **socketpair(2)**, **open(2)**, **signal(2)**, **sigaction(2)**

Linux Man Page

20 Jun 1999

FIFO(4)

---

## FULL

**НАЗВАНИЕ** full - всегда полное устройство

**ОПИСАНИЕ** Файл **/dev/full** имеет основной номер устройства, равный единице, и второстепенный, равный семи.

При записи в файл **/dev/full** будет возвращено сообщение об ошибке ENOSPC. При попытке чтения файла **/dev/full** будут возвращаться нули. Команда seek всегда выполнима при работе с файлом **/dev/full**.

#### НАСТРОЙКА

Если в Вашей системе нет файла **/dev/full**, то его можно создать при помощи следующих команд:

```
mknod -m 666 /dev/full c 1 7  
chown root:root /dev/full
```

**ФАЙЛЫ** /dev/full

**СМ. ТАКЖЕ** **mknod(1)**, **null(4)**, **zero(4)**

Linux

August 2, 1997

FULL(4)

---

## HD

**НАЗВАНИЕ** hd - MFM/IDE-жесткие диски

**ОПИСАНИЕ** Устройства **Hd\*** - это блочные устройства для прямого доступа к винчестерам MFM/IDE. Главным (master) диском на первичном контроллере IDE (основной его номер - 3) является **hda**; зависимый (slave) диск - **hdb**. Главный диск на вторичном контроллере (основной номер 22) - **hdc**, а зависимый - **hdd**.

Названия блочных IDE-устройств имеют форму **hdX**или **hdXP**где буква X указывает на физический диск, а P - номер, указывающий на раздел этого диска. Первая форма **hdX** используется для обращения ко всему диску. Номера разделов указаны в порядке их обнаружения, и номера присваиваются только "непустым" нерасширенным разделам. Так или иначе, номера с первого по четвертый присваиваются разделам, описанным в MBR ('основным' (primary) разделам), независимо от того, используются ли они расширенными (extended) или нет. Поэтому первым логическим разделом будет **hdX5**. Поддерживаются как разделы DOS, так и BSD. На IDE-диске может быть не более 63-х разделов.

Например, **/dev/hda** ссылается на все первые IDE-накопители системы, а **/dev/hdb3** ссылается на третий, 'основной', раздел DOS вторичного накопителя.

Файлы устройств создаются следующим образом:

```
mknod -m 660 /dev/hda b 3 0  
mknod -m 660 /dev/hdal b 3 1  
mknod -m 660 /dev/hda2 b 3 2  
...  
mknod -m 660 /dev/hda8 b 3 8  
mknod -m 660 /dev/hdb b 3 64  
mknod -m 660 /dev/hdb1 b 3 65  
mknod -m 660 /dev/hdb2 b 3 66  
...
```

```
mknod -m 660 /dev/hdb8 b 3 72
chown root:disk /dev/hd*
```

**ФАЙЛЫ** /dev/hd\*
**СМ. ТАКЖЕ** mknod(1), chown(1), mount(8), sd(4)
Linux 17 December 1992

HD(4)

## INITRD

**НАЗВАНИЕ** initrd - электронный диск, устанавливаемый во время загрузки системы

**ОПИСАНИЕ** Специальный файл **/dev/initrd** является блочным устройством только для чтения. Устройство **/dev/initrd** - это электронный диск, устанавливается во время загрузки системы до того момента, как запускается ядро. Поэтому ядро может использовать содержимое файлов блочных устройств **/dev/initrd** для двухступенчатой загрузки системы.

На первой ступени загрузки ядро запускается и устанавливает корневую файловую систему, содержащую **/dev/initrd**. На второй ступени загружаются дополнительные драйверы и модули из начального корневого раздела. После загрузки дополнительных модулей устанавливается новая корневая файловая система.

### ОПЕРАЦИЯ ЗАГРУЗКИ

С помощью **initrd** система загружается в следующем порядке:

1. Начальный загрузчик загружает программу ядра и содержимое **/dev/initrd** в память компьютера.
2. При запуске ядро "распаковывает" и копирует содержимое устройства **/dev/initrd** в устройство **/dev/ram0**, а затем освобождает память, используемую устройством **/dev/initrd**.
3. Ядро устанавливает (для чтения и записи) устройство **/dev/ram0** как начальную корневую файловую систему.
4. Если обычная корневая система является и начальной (например, **/dev/ram0**), то ядро переходит к последнему шагу в загрузке.
5. Если запускаемый файл **/linuxrc** присутствует в начальной корневой системе, то **/linuxrc** запускается с идентификатором пользователя, равным нулю. (Файл **/linuxrc** должен иметь права доступа к системе, чтобы осуществить соответствующий запуск. Файл **/linuxrc** может быть любым запускаемым файлом, включая скрипт shell.)
6. Если **/linuxrc** не запущен или **/linuxrc** прерван, то устанавливается нормальная корневая файловая система. (Если работа **/linuxrc** с файловой системой, установленной над начальной корневой файловой системой, прерывается, то поведение ядра становится **НЕАДЕКВАТНЫМ**. Прочтите раздел **ЗАМЕЧАНИЯ** для получения более подробной информации).
7. Если обычная корневая файловая система имеет каталог **/initrd**, то устройство **/dev/ram0** переносится из **/** в **/initrd**. Иначе, если каталога **/initrd** не существует, то устройство **/dev/ram0** удаляется. При переносе из **/** в **/initrd** **/dev/ram0** не удаляется и процесс в дальнейшем работает с **/dev/ram0**. Если каталога **/initrd** не существует в обычной корневой системе и остаются процессы, работающие с **/dev/ram0** (при существовании **/linuxrc**), поведение ядра становится **НЕПРЕДСКАЗУЕМЫМ**. Прочтите раздел **ЗАМЕЧАНИЯ** для получения более подробной информации об этом.
8. Обычная процедура загрузки (например, вызов **/sbin/init**) производится обычной корневой файловой системой.

**ОПЦИИ** Нижеследующие опции начального загрузчика с использованием **initrd** влияют на загрузку ядра:

**initrd=имяфайла**

Указывает файл, который загружается как содержимое **/dev/initrd**. Для **LOADLIN** это опция командной строки. Для **LILO** Вам следует использовать эту команду в файле настроек **LILO** под названием **/etc/lilo.config**. Название файла, указанное в этой опции, будет образом файловой системы, заархивированным программой **gzip**.

**noinitrd**

Эта опция отключает двухэтапную загрузку. Ядро выполняет обычную процедуру загрузки, как при наличии неинициализированного **/dev/initrd**. В этом режиме любое содержимое **/dev/initrd**, загруженное в память при помощи начального загрузчика, не используется. Это позволяет **/dev/initrd** содержать какую угодно информацию. Так или иначе, устройство **/dev/initrd** предназначено только для чтения и может быть прочитано только один раз после загрузки системы.

**root=название-устройства**

указывает на устройство, используемое как нормальная корневая файловая система. Для **LOADLIN** это опция командной строки. Для **LILO** это опция, используемая во время загрузки; она может быть опцией в файле настроек **LILO** под названием **/etc/lilo.config**. Устройство, указанное в этой строке, должно устанавливаться при наличии подходящей корневой файловой системы.

## СМЕНА ОБЫЧНОЙ КОРНЕВОЙ СИСТЕМЫ

По умолчанию установки ядра (с параметром **rdev** или сборка ядра с этим параметром) или установки в опциях начального загрузчика используются для обычной корневой файловой системы. При сборке файловой системы NFS как корневой надо использовать опции загрузки **nfs\_root\_name** и **nfs\_root\_addrs** для настройки NFS. Для получения более подробной информации по системе NFS, собранной как корневая, прочтите файл документации о ядре **nfsroot.txt**. Дополнительную информацию об установке корневой файловой системы можно также прочесть в документации о **LILO** и **LOADLIN**.

При помощи **/linuxrc** можно изменить устройство корневой системы. Для этого необходимо собранное **/proc**. После установки **/proc** **/linuxrc** меняет корневую файловую систему, делая записи в файлах: **/proc/sys/kernel/real-root-dev**, **/proc/sys/kernel/nfs-root-name** и **/proc/sys/kernel/nfs-root-addrs**. Для смены физического устройства корневой системы **/linuxrc** записывает номер нового устройства в **/proc/sys/kernel/real-root-dev**. Для смены корневой системы NFS **/linuxrc** записывает настройки NFS в файлы: **/proc/sys/kernel/nfs-root-name** и **/proc/sys/kernel/nfs-root-addrs**, - и указывает 0xff (например, псевдономер устройства NFS) в файле **/proc/sys/kernel/real-root-dev**. Например, следующая командная строка shell сменит устройство корневой системы на **/dev/hdb1**:

```
echo 0x365 >/proc/sys/kernel/real-root-dev
```

В случае с NFS нижеследующая строка сменит устройство корневой системы на каталог NFS **/var/nfsroot** локального сервера NFS с IP-адресом 193.8.232.7 (для машины с IP-адресом 193.8.232.7 и именем 'idefix'):

```
echo /var/nfsroot >/proc/sys/kernel/nfs-root-name  
echo 193.8.232.2:193.8.232.7::255.255.0:idefix \
```

MAN PAGES 4.Специальные файлы

```
>/proc/sys/kernel/nfs-root-addrs  
echo 255 >/proc/sys/kernel/real-root-dev
```

## ИСПОЛЬЗОВАНИЕ

Главной причиной использования **initrd** была возможность настраивать модули ядра при установке системы.

Далее приводятся этапы установки:

1. Программа-загрузчик устанавливается с дискеты или с любого другого носителя вместе с минимальным ядром (например, поддержка для **/dev/ram**, **/dev/initrd** и файловая система ext2) загружает **/dev/initrd** со сжатой начальной файловой системой.
2. Скрипт **/linuxrc** определяет: что необходимо для сборки корневой файловой системы (какой тип устройства, драйверы устройства и т.д.), носители (например, CD-ROM, сеть и т.д.). Это может быть сделано путем опроса пользователя, методом автопробации или используя оба подхода.
3. Скрипт **/linuxrc** загружает необходимые модули из начальной файловой системы.
4. Далее **/linuxrc** создает корневую файловую систему. (На этой ступени она еще не является рабочей).
5. Скрипт **/linuxrc** устанавливает **/proc/sys/kernel/real-root-dev**, разбирает **/proc**, загрузочную корневую файловую систему и любую файловую систему, которую он собрал, и завершает свою работу.
6. Ядро устанавливает рабочую корневую файловую систему.
7. Когда файловая система установлена и функционирует, можно установить начальный загрузчик.
8. Начальный загрузчик настроен на установку в **/dev/initrd**, файловую систему с набором модулей для сборки системы. (Устройство **/dev/ram0** может быть изменено, затем отсоединено, а образ может быть скопирован из устройства в файл).
9. Теперь система может быть загружена.

Основной ролью **/dev/initrd** в вышеописанном процессе является использование данных настройки во время работы загрузочной корневой файловой системы без требования полного ядра или его сборки.

Второй вариант используется, главным образом, если Linux работает в системах с различной конфигурацией и в сети, администрируемой одним человеком. В таких случаях удобно использовать одно ядро и минимальное количество специфического программного обеспечения. Создайте общий файл со всеми необходимыми модулями, тогда будут различаться только скрипт **/linuxrc** или скрипт, запускаемый **/linuxrc**.

Такая информация, как местонахождение раздела с корневой файловой системой, не нужна во время загрузки; система, загружаемая с **/dev/initrd**, может провести опрос пользователя и/или делать определения автоматически.

Также дистрибутивы Linux CD-ROM могут использовать **initrd** для упрощенной установки системы с помощью CD-ROM. Дистрибутив может использовать **LOADLIN** для прямой загрузки **/dev/initrd** с CD-ROM без каких-либо дискет. Дистрибутив также может использовать загрузочную дискету **LILO** и затем загрузить больший электронный диск при помощи **/dev/initrd** с CD-ROM.

## КОНФИГУРАЦИЯ

**/dev/initrd** – это блочное устройство только для чтения с основным номером 1 и второстепенным номером 250. Обычно права на **/dev/initrd** принадлежат **root.disk** с режимом 0400 (только для чтения пользователем root). Если система

Linux не имеет готового **/dev/initrd**, он может быть создан следующими командами:

```
mknod -m 400 /dev/initrd b 1 250
chown root:disk /dev/initrd
```

Поддержка для электронного диска и загрузочного электронного диска (например, **CONFIG\_BLK\_DEV\_RAM=y** и **CONFIG\_BLK\_DEV\_INITRD=y**) должна быть собрана вместе с ядром для того, чтобы использовать **/dev/initrd**. При использовании **/dev/initrd** драйвер электронного диска не может быть загружен как модуль.

**ФАЙЛЫ**

```
/dev/initrd
/dev/ram0
/linuxrc
/initrd
```

**СМ. ТАКЖЕ**

1. С существующим ядром любая файловая система, остающаяся установленной при переносе **/dev/ram0** из **/** в **/initrd**, доступна. Записи в **/proc/mounts** не обновляются.
2. В случае с текущей версией ядра: если каталога **/initrd** не существует и если **/dev/ram0** используется каким-либо процессом (или в нем установлена какая-либо файловая система), то "**/dev/ram0**" НЕ будет полностью удален. Если **/dev/ram0** НЕполностью отсоединен, то **/dev/ram0** останется в памяти системы.
3. Пользователи **/dev/initrd** не должны зависеть от поведения описанного в предыдущих замечаниях. скорее всего это будет исправлено в последующих версиях ядра Linux.

**ЗАМЕЧАНИЯ** Исходный текст в ядре для **initrd** был написан Werner Almesberger <almesber@lrc.epfl.ch> и Hans Lermen <lermen@elserv.fmm.fgan.de>. Исходный текст **initrd** добавляется в ядро, начиная с версии 1.3.73.

**СМ. ТАКЖЕ** **chown(1)**, **mknod(1)**, **/dev/ram(4)**, **freeramdisk(8)**, **rdev(8)**, файл документации в исходных файлах ядра **initrd.txt**, документацию по LILO, документацию по LOADLIN и документацию по SYSLINUX.

Linux 2.0

6 November 1997

**INITRD(4)**

---

## INTRO

**НАЗВАНИЕ** intro - введение в специальные файлы

**ОПИСАНИЕ** Эта глава описывает специальные файлы.

**ФАЙЛЫ** **/dev/\*** -- файлы устройств

Linux 24 July 1993 INTRO(4)

---

## LP

**НАЗВАНИЕ** lp - принтеры

**СИНТАКСИС** #include <linux/lp.h>

**КОНФИГУРАЦИЯ**

**lp[0-2]** - это символьные устройства для работы параллельного порта; оно имеет основное число 6 и второстепенное 0-2. Второстепенные числа соответствуют следующим адресам порта принтера: 0x03bc, 0x0378 и 0x0278. Обычно они работают в режиме 220 с пользователем **root** группы lp. Управлять принтерами можно при помощи опроса и при помощи сигнала прерывания. Прерывания рекомендуются при высоких скоростях печати, например, для лазерных принтеров. Для обычных матричных принтеров будет достаточно опроса портов. По умолчанию устанавливается

режим опроса.

**ОПИСАНИЕ** Поддерживаются следующие вызовы: **ioctl(2)**:

**int ioctl(int fd, LPETIME, int arg)**

Устанавливает время, в течение которого драйвер ожидает проверку принтера, когда его буфер заполнен *arg*. Если принтер быстрый, то лучше уменьшить это число, а если медленный, то лучше увеличить. Это чило представлено в сотых секунды; по умолчанию указано число 2, что означает 0,02 секунды. Это отражается только на драйвере опроса.

**int ioctl(int fd, LPCHAR, int arg)**

Устанавливает максимальное количество циклов, в течение которых драйвер опроса ожидает готовность принтера к переводу символа в переменную *arg*. Если печать медленная, то рекомендуется увеличить это число, а если медленная система, то увеличить. По умолчанию это число равно 1000 и оно влияет только на драйвер опросов.

**int ioctl(int fd, LPABORT, int arg)**

Если *arg* равен нулю, то драйвер принтера будет повторять попытки при ошибках, в противном случае он прекратит их повторение. По умолчанию устанавливаемое значение равно нулю.

**int ioctl(int fd, LPABORTOPEN, int arg)**

Если *arg* равен нулю, то **open(2)** будет прерван при ошибке, иначе ошибка будет проигнорирована. Значение по умолчанию не равно нулю.

**int ioctl(int fd, LPCAREFUL, int arg)**

Если *arg* равен нулю, то сигналы о том, что закончилась бумага, принтер не готов или ошибка, должны быть с отрицательным значением: при другом значении они игнорируются. По умолчанию значение не равно нулю.

**int ioctl(int fd, LPWAIT, int arg)**

Устанавливает число циклов ожидания до стробирования принтера и для получения только что напечатанного символа, а также устанавливает число циклов ожидания до остановки стробирования в зависимости от значения *arg*. В руководстве сказано, что это время должно равняться 0.5-м микросекундам, но опыт показал, что в данном случае хватает и времени задержки программы. Поэтому устанавливаемое значение по умолчанию равно нулю. Это используется как для опрашивающего, так и прерывающего принтера.

**int ioctl(int fd, LPSETIRQ, int arg)**

Этот ioctl() требует прав суперпользователя. Он создает int с новым IRQ в значении аргумента. Побочный эффект - повторный запуск принтера. Когда *arg* равен нулю, будет использован драйвер опроса, установленный по умолчанию.

**int ioctl(int fd, LPGETIRQ, int \*arg)**

Сохраняет используемый IRQ в *arg*.

**int ioctl(int fd, LPGETSTATUS, int \*arg)**

Сохраняет значение статуса порта, равное *arg*. Биты имеют следующие значения:

LP_PBUSY	принтер занят, значение по умолчанию - 1
LP_PACK	подтверждение, значение по умолчанию - 0
LP_POUTPA	закончилась бумага, значение по умолчанию - 1
LP_PSELECD	принтер выбран, значение по умолчанию - 1
LP_PERRORP	ошибка, значение по умолчанию - 0

Обратитесь к руководству по Вашему принтеру для

уяснения значения сигналов. Замечание: неописанные биты тоже могут иметь значение (в зависимости от Вашего принтера).

**int ioctl(int fd, LPRESET)**

вновь запускает принтер, аргументы для этого не используются.

**ФАЙЛЫ** /dev/lp\*

**АВТОРЫ** Изначально драйвер принтера был описан Jim Weigand и Linus Torvalds. Далее он был доработан Michael K. Johnson. Программа прерывания написана Nigel Gamble. Alan Cox сделал ее модульной. LPCAREFUL, LPABORT, LPGETSTATUS были добавлены Chris Metcalf.

**СМ. ТАКЖЕ** mknod(1), chown(1), chmod(1), tunelp(8), lpcntl(8)

January 15, 1995

LP(4)

---

## MEM

**НАЗВАНИЕ** mem, kmem, port - системная память, память ядра и порты системы

**ОПИСАНИЕ** Mem - это файл символьного устройства, являющийся образом физической памяти компьютера. Этот файл может быть использован для проверки системы (и даже для внесения в нее исправлений). Адреса байтов в памяти интерпретируются как адреса физической памяти. Ссылки на несуществующие адреса вызывают возвращаемые ошибки. Проверка системы или внесение в нее исправлений иногда приводят к непредвиденным результатам в том случае, если есть биты, которые разрешается только читать и только записывать. Файл создается: mknod -m 660 /dev/mem с 1 1 chown root:mem /dev/mem Файл kmem идентичен файлу mem, исключая то, что первый обеспечивает Вас доступом только к памяти ядра. Он создается следующим образом: mknod -m 640 /dev/kmem с 1 2 chown root:kmem /dev/kmem Port идентичен mem, но он предоставляет Вам доступ к портам ввода-вывода. Он создается следующим образом: mknod -m 660 /dev/port с 1 4 chown root:mem /dev/port **ФАЙЛЫ** /dev/mem /dev/kmem /dev/port **СМ. ТАКЖЕ** chown(1), mknod(1), ioperm(2) Linux 21 November 1992 MEM(4)

---

## MOUSE

**НАЗВАНИЕ** mouse - последовательный интерфейс мыши

**НАСТРОЙКА** Мышь подсоединяется к последовательному порту RS232/V24.

Для получения более подробной информации прочтите **tty(4)**

**ОПИСАНИЕ**

**Введение**

ниже приведена описание разъема с девятью штырями, используемого для подключения мыши:

штырь	название	используется для
2	RX	Данные
3	TX	-12 V, I <sub>max</sub> = 10 mA
4	DTR	+12 V, I <sub>max</sub> = 10 mA
7	RTS	+12 V, I <sub>max</sub> = 10 mA
5	GND	Заземление

Это описание приводится в документации, хотя 9-и V хватит почти любой мыши.

Драйвер мыши может распознать мышь, посылая слабый сигнал RTS и затем увеличивая его. Примерно через 14 миллисекунд мышь вернет 0x4D ('M'). Еще через 63 миллисекунды трехкнопочная мышь Microsoft-compatible вернет сигнал 0x33 ('3').

Относительное движение мыши посыпается как dx (положительное значение означает движение направо) и dy (положительное значение означает движение вниз). Различные мыши могут работать с разными скоростями. Для определения

скорости устанавливаются следующие ее значения (по очереди): 9600, 4800, 2400 и 1200 бит/с; каждый раз при установке значения скорости пишутся 2 символа из таблицы, приведенной ниже, и после этого идет время ожидания, равное 0,1 секунды. В таблице указаны скорости и символы, соответствующие им:

бит/с	символы
9600	*q
4800	*p
2400	*o
1200	*n

первый байт из пакета может использоваться для синхронизации.

#### Протокол Microsoft

Протокол **Microsoft** использует один начальный бит, 7 битов данных (без проверки четности), один стоп-бит; протокол работает со скоростью 1200 бит/с. Данные пересыпаются в RxD трехбайтовыми пакетами. Движения по dx и dy пересыпаются как дополнения друг друга. Значение lb (rb) устанавливается при нажатии левой (правой) кнопки:

байт	d6	d5	d4	d3	d2	d1	d0
1	1	lb	rb	dy7	dy6	dx7	dx6
2	0	dx5	dx4	dx3	dx2	dx1	dx0
3	0	dy5	dy4	dy3	dy2	dy1	dy0

#### Протокол Microsoft для трехкнопочной мыши

Обычно мышь Microsoft имеет две кнопки. Но есть несколько трехкнопочных мышей, также использующих протокол Microsoft. О нажатии средней кнопки сообщается отсыпанием пакета с нулевым движением и ненажатыми клавишами. (Таким образом, в отличие от двух других кнопок, статус средней не отображается в каждом пакете).

#### Протокол Logitech

Трехкнопочная мышь Logitech использует варианты протокола Microsoft: когда средняя кнопка отпущена, посыпается вышеописанный трехбайтовый пакет, а когда она нажата, посыпается четырехбайтовый пакет, где четвертый байт имеет значение 0x20. В частности, о нажатии средней кнопки сообщается с помощью пакета 0,0,0,0x20 при том, что остальные кнопки остаются ненажатыми.

#### Протокол Mousesystems

Протокол **Mousesystems** использует один начальный бит, 8 битов данных, не использует проверку по четности и два стоп-бита при скорости 1200 бит/с. Данные посыпаются на RxD пятибайтовыми пакетами. dx посыпается как совокупность двух величин, состоящих из двух частей, dy посыпается как совокупность двух отрицательных величин, состоящих из двух частей. lb (mb, rb) обнуляются при нажатии левой (средней, правой) кнопки:

байт	d7	d6	d5	d4	d3	d2	d1	d0
1	1	0	0	0	0	lb	mb	rb
2	0	dx <sub>a</sub> 6	dx <sub>a</sub> 5	dx <sub>a</sub> 4	dx <sub>a</sub> 3	dx <sub>a</sub> 2	dx <sub>a</sub> 1	dx <sub>a</sub> 0
3	0	dy <sub>a</sub> 6	dy <sub>a</sub> 5	dy <sub>a</sub> 4	dy <sub>a</sub> 3	dy <sub>a</sub> 2	dy <sub>a</sub> 1	dy <sub>a</sub> 0
4	0	dx <sub>b</sub> 6	dx <sub>b</sub> 5	dx <sub>b</sub> 4	dx <sub>b</sub> 3	dx <sub>b</sub> 2	dx <sub>b</sub> 1	dx <sub>b</sub> 0
5	0	dy <sub>b</sub> 6	dy <sub>b</sub> 5	dy <sub>b</sub> 4	dy <sub>b</sub> 3	dy <sub>b</sub> 2	dy <sub>b</sub> 1	dy <sub>b</sub> 0

Четвертый и пятый байты описывают изменения, произошедшие за время передачи второго и третьего байта.

#### Протокол Sun

Протокол **Sun** – это трехбайтовая версия протокола Mousesystems (описанного выше), только два последних байта не отсыпаются.

#### Протокол MM

Протокол **MM** производит проверку на нечетность, использует один начальный бит, восемь битов данных и один стоп-бит.

при скорости 1200 бит/с. Данные посылаются на RxD трехбайтовыми пакетами. dx и dy отсылаются как отдельные значения со своими знаками, бит знака имеет отрицательное значение. lb (mb, rb) отсылаются при нажатии левой (средней, правой) кнопки:

байт	d7	d6	d5	d4	d3	d2	d1	d0
1	1	0	0	dxs	dys	lb	mb	rb
2	0	dx6	dx5	dx4	dx3	dx2	dx1	dx0
3	0	dy6	dy5	dy4	dy3	dy2	dy1	dy0

**ФАЙЛЫ** /dev/mouse  
- это обычно используемая символьная ссылка на устройство "мышь".

**СМ. ТАКЖЕ** ttys(4), gpm(8)

Linux February 10, 1996

MOUSE(4)

---

## NULL.

---

**НАЗВАНИЕ** null, zero - пустое устройство

**ОПИСАНИЕ** Любые данные, записанные в специальные файлы **null** или **zero**, стираются.

Считывание информации из специального файла **null** возвращает конец файла, а чтение файла **zero** символы \0.

Файлы **null** и **zero** создаются так:

```
mknod -m 666 /dev/null c 1 3
mknod -m 666 /dev/zero c 1 5
chown root:mem /dev/null /dev/zero
```

**ЗАМЕЧАНИЯ**

Если эти файлы закрыть для чтения и записи для всех пользователей, то многие программы будут работать непонятным образом.

**ФАЙЛЫ** /dev/null  
/dev/zero

**СМ. ТАКЖЕ** chown(1), mknod(1)

Linux 21 November 1992

NULL(4)

---

## RAM

---

**НАЗВАНИЕ** ram - устройство для работы с электронным диском

**ОПИСАНИЕ** Ram - это блочное устройство для доступа к электронному диску.

Оно создается следующим образом:

```
mknod -m 660 /dev/ram b 1 1
chown root:disk /dev/ram
```

**ФАЙЛЫ** /dev/ram

**СМ. ТАКЖЕ** mknod(1), chown(1), mount(8)

Linux 21 November 1992

RAM(4)

---

## RANDOM

---

**НАЗВАНИЕ** random, urandom - устройство ядра для создания случайных чисел

**ОПИСАНИЕ** Специальные посимвольные файлы **/dev/random** и **/dev/urandom** (появились в Linux версии 1.3.30) предоставляют интерфейс генератору случайных чисел в ядре. Файл **/dev/random** обозначает единицей главное число устройства и цифрой 8 второстепенное. Файл **/dev/urandom** обозначает единицей главное число устройства и цифрой 9 второстепенное.

Генератор случайных чисел выводит шумы из драйверов устройств и других источников в "хаотичный" пул (entropy pool). Генератор также сохраняет необходимое количество битов шума в этом пуле и формирует из него случайные числа.

При чтении данных в устройстве **/dev/random** создаются

только случайные байты, состоящие из битов шума "хаотичного" пула. Устройство **/dev/random** может быть необходимо пользователям, которые требуют очень высокого коэффициента случайности, например, при создании ключа доступа и т.п. Если "хаотичный" пул опустел, чтение **/dev/random** блокируется, пока необходимое количество битов в пуле не будет создано.

Чтение данных устройства **/dev/urandom** возвратит столько байтов, сколько было запрошено. В результате, если в пуле было недостаточно битов, теоретически возможно будет найти уязвимость алгоритма, использующего это устройство. Если это важно, используйте **/dev/random**.

#### НАСТРОЙКА

Если в Вашей системе нет файлов **/dev/random** и **/dev/urandom**, они могут быть созданы заданием команд:

```
mknod -m 644 /dev/random c 1 8  
mknod -m 644 /dev/urandom c 1 9  
chown root:root /dev/random /dev/urandom
```

При запуске системы входной "хаотичный" пул может быть слишком мал для нормальной работы. Это уменьшает коэффициент случайности. Чтобы избежать этого эффекта, включите следующие строки в загрузочный скрипт:

```
echo "Initializing kernel random number generator..."  
if [ -f /var/random-seed ]; then  
    cat /var/random-seed >/dev/urandom  
fi  
dd if=/dev/urandom of=/var/random-seed count=1
```

Также добавьте следующие строки в скрипт, запускающийся при завершении работ:

```
echo "Saving random seed..."  
dd if=/dev/urandom of=/var/random-seed count=1
```

**ФАЙЛЫ** **/dev/random**,  
**/dev/urandom**

**АВТОР** Генератор случайных чисел ядра написан  
Theodore Ts'o (tytso@athena.mit.edu).

**СМ. ТАКЖЕ** mknod (1)  
RFC 1750, "Randomness Recommendations for Security"

Linux August 1, 1997 RANDOM(4)

---

## SD

**НАЗВАНИЕ** sd - драйвер для дисковых накопителей SCSI

**СИНТАКСИС** RPART \*/

**НАСТРОЙКА** Это блочное устройство. Имя устройства имеет следующую форму: **sdlp**, где **l** - буква, обозначающая физический накопитель, а **p** - номер раздела в этом физическом накопителе. Часто номер раздела **p** бывает пропущен, когда устройство соответствует полностью всему накопителю. Диски SCSI содержат цифру 8 (это основной номер устройства), а второстепенные числа устройства составляются по форме (16 \* номер\_накопителя) + номер\_раздела, где **номер\_накопителя** - это номер физического накопителя в порядке обнаружения, а **номер\_раздела**:

раздел 0 - накопитель полностью

разделы #1-#4 являются основными ("primary") DOS-разделами  
разделы #5-#8 - расширенные или логические ("extended" или "logical") DOS-разделы. Например, основной номер **/dev/sda** - 8, второстепенный - 0; устройство присвоит их всем первым накопителям SCSI в системе; основной номер **/dev/sdb3** - 8, второстепенный - 19, устройство присвоит их третьему основному разделу DOS на втором накопителе SCSI. В настоящий момент поддерживаются только блочные устройства.

**ОПИСАНИЕ** Доступны следующие вызовы IR ioctl :

```
HDIO_GETGEO
Возвращает параметры диска из BIOS в виде следующей структуры:
struct hd_geometry {
    unsigned char heads;
    unsigned char sectors;
    unsigned short cylinders;
    unsigned long start;
}
```

Указатель на эту структуру является параметром **ioctl(2)**. Информация, возвращаемая в параметр геометрии дискового накопителя, понимается DOS! Эта геометрия не является физической геометрией накопителя. Применяется, когда составляется таблица разделов накопителя и есть необходимость для совместимости ее с операциями **fdisk(1)**, **efdisk(1)** и **lilo(1)**. Если информация о геометрии диска недоступна, всем параметрам возвращается нулевое значение.

**BLKGETSIZE**  
Возвращает размер устройства в секторах. Параметр **ioctl(2)** будет указывать на длину этого устройства.

**BLKRRPART**  
Заставляет перечитать таблицы разделов на диске SCSI. Указывается без параметра. ioctl также поддерживает **scsi(4)**. Если параметр **ioctl(2)** оказался затребован, но его не существует, тогда **ioctl(2)** возвратит -EINVAL.

**ФАЙЛЫ** /dev/sd[a-h]: целое устройство  
/dev/sd[a-h][0-8]: отдельные блоки разделов

**СМ. ТАКЖЕ** **scsi(4)**

Thu Dec 17 10:15:53 1992

**SD(4)**

---

## ST

**НАЗВАНИЕ** st - ленточный накопитель SCSI  
**СИНТАКСИС**

```
#include <sys/mtio.h>
int ioctl(int fd, int request [, (void *)arg3]);
int ioctl(int fd, MTIOCSTOP, (struct mtstop *)mt_cmd);
int ioctl(int fd, MTIOCGET, (struct mtget *)mt_status);
int ioctl(int fd, MTIOCPOS, (struct mtpos *)mt_pos);
```

**ОПИСАНИЕ**  
Драйвер **st** участвует в установке связи с различными ленточными накопителями SCSI. В настоящее время драйвер позволяет управлять любыми устройствами "последовательного доступа". В драйвере **st** цифра 9 используется как основное число устройства.  
Каждое устройство использует восемь второстепенных номеров устройства. Первые пять битов во второстепенных номерах определяют последовательность обнаружения. Второстепенные номера могут быть сгруппированы в два набора из четырех чисел: главные (автоперемотка) младшие числа устройства *n*, и числа устройства "неперемотки", (*n*+ 128). Открытые устройства используют главный номер устройства, посылая команду REWIND, когда закрываются. (Заметьте, что использование устройства автоперемотки для установки ленты в определенное положение, для примера, *mt*, не даст желаемого результата: лента перемотается после команды *mt*, и следующая команда будет выполняться с начала ленты).  
В каждой группе четыре второстепенных номера доступны для определения устройств с особыми характеристиками (такими,

как: длина блока, сжатие, плотность и другое). Когда система запускается, только первое устройство доступно. Другие три приводятся в действие, когда определены некоторые их характеристики (смотрите ниже). Путем изменения константы при компиляции возможно изменение баланса между максимальным числом ленточных накопителей и числом из второстепенных номеров каждого накопителя. Начальное размещение позволяет контролировать 32 ленточных устройства. Для примера, возможно контролировать до 64-х ленточных устройств с двумя второстепенными номерами.

Устройства обычно создаются так:

```
mknod -m 666 /dev/st0 c 9 0
mknod -m 666 /dev/st01 c 9 32
mknod -m 666 /dev/st0m c 9 64
mknod -m 666 /dev/st0a c 9 96
mknod -m 666 /dev/nst0 c 9 128
mknod -m 666 /dev/nst01 c 9 160
mknod -m 666 /dev/nst0m c 9 192
mknod -m 666 /dev/nst0a c 9 224
```

Нет соответствующего блочного устройства.

Драйвер использует внутренний буфер, что достаточно для сохранения в нем одного блока ленты. В ядре до появления версии 2.1.121 буфер был размещен как один прилегающий блок. Ограничение размера блока обеспечивается распределителем памяти ядра может. Ограничение в настоящее время составляет: 128 kB для 32-битной и 256 kB для 64-битной архитектуры. В новейших ядрах накопитель располагает буфер в нескольких частях, если это необходимо. По умолчанию максимальное число частей - 16. Это способ сделать максимальный размер блока очень большим (2MB, если выделяется 16 блоков по 128 kB).

Размер внутреннего буфера драйвера определяется константой при сборке; эта константа может быть изменена при загрузке ядра. Вдобавок к этому, драйвер пытается разместить больший временный буфер во время запуска, если это необходимо. Тем не менее, размещение во время запуска больших блоков памяти может быть неудачным, и целесообразно не слишком полагаться на размещение активного буфера в ядре версии 2.1.121 и более ранних (это применяется также при загрузке драйвера по требованию kernald или kmod).

После старта системы параметры ленточного устройства определены программно-аппаратными средствами накопителя. Для примера, если производитель выбирает фиксированный блочный режим, то ленточное устройство использует этот режим. Параметры могут быть изменены вызовом функции **ioctl()** и действовать, когда устройство было закрыто и потом открыто вновь. Установка параметров влияет на оба устройства (автоперемотки и неперемотки).

Особые параметры могут быть определены для остальных устройств. Параметры начинают работать, когда устройство открыто. Для примера, системный администратор может определить одно устройство, которое записывает данные в режиме фиксированных блоков (с определенным размером блока), и другое, которое записывает данные в режиме переменных блоков (если накопитель поддерживает оба режима).

Драйвер поддерживает **ленточные** разделы, но только если их поддерживает сам накопитель (заметьте, что ленточные разделы не влияют на разделы диска. Разделенная лента может быть отображена как несколько логических лент). С ioctl возможна поддержка разделов. Местоположение ленты зафиксировано внутри каждого раздела. Раздел используется для запуска последующих ленточных операций, выбранных с

помощью ioctl. Максимальное число разделов на ленте определяется константой при компиляции (изначально оно равно четырем). Драйвер ограничивает ioctl, что позволяет форматировать ленту с одним или двумя разделами.

Устройство **/dev/tape** обычно создает по умолчанию как "жесткую", так и символьную ссылку ленточного устройства в системе.

#### ПЕРЕДАЧА ДАННЫХ

Драйвер поддерживает операции в обоих режимах: фиксированных и переменных блоков (только если оба режима поддерживают сам накопитель). В режиме фиксированных блоков накопитель записывает блоки определенного размера, и размер блока не зависит от количества требуемых байтов, которые вызываются для записи системой. В режиме переменных блоков один блок ленты записывается при каждом вызове записи и требует определения размера соответствующего блока ленты.

В режиме переменных блоков при чтении требуемых байтов размер блока ленты точно не подбирается. Если требуемых байтов больше, чем в следующем блоке ленты, драйвер возвращает данные, а функция возвращает текущий размер блока. Если размер блока больше, чем количество требуемых байтов, читается только количество данных с начала блока, а остаток блока отбрасывается.

В режиме фиксированных блоков чтение количества байтов может быть произвольным, если буферизация дает такую возможность, или пропорциональным размеру блока ленты, если такой возможности нет. Ядро версии, появившихся до 2.1.121, позволяет записывать блок с произвольным количеством байтов. Во всех других случаях количество байтов должно быть пропорциональным размеру блока ленты. Метка файла автоматически записывается на ленту, если последней операцией до закрытия была запись.

Если данные остаются в буфере, то метка файла будет найдена и буферизация данных продолжится. Следующее чтение вернет ноль байтов, затем вернет данные из следующего файла. В конце записи данные сообщают о возвращении нуля байтов для второго упорядоченного вызова записи. И, наконец, третье чтение вернет ошибку.

#### ВЫЗОВЫ IOCTL

Драйвер поддерживает три запроса ioctl. Запросы, не опознанные драйвером **st**, пропускаются в драйвер SCSI. См. **/usr/include/linux/mtio.h**:

#### MTIOCTOP - выполнение ленточных операций

Это требует использования типа аргумента (**struct mtop \***). Не любой накопитель поддерживает все операции. Драйвер возвращает ошибку EIO, если накопитель не выполнил операцию.

Операции с магнитной лентой для нормального использования:

MTBSF           Перемотка ленты на **mt\_count** файловых маркеров назад.

MTBSFM          Перемотка ленты на **mt\_count** файловых маркеров назад. Перемотка ленты на сторону ЕОТ последней метки файла.

MTBSR          Перемотка ленты на **mt\_count** записей (блоков ленты) назад.

MTBSS          Перемотка ленты на **mt\_count** установленных маркеров назад.

MTCOMPRESSION Включение режима сжатия данных на ленте накопителя, если **mt\_count** не нулевое, и отключение компрессии, если **mt\_count** равно нулю. Эта команда использует страницу 15 MODE, поддерживаемую многими устройствами

	DAT.
MTEOM	Перемотка ленты к концу данных (для добавляемых файлов).
MTERASE	Стирает информацию с ленты в кассете.
MTFSF	Перемотка ленты на <b>mt_count</b> файловых маркеров вперед.
MTFSFM	Перемотка ленты на <b>mt_count</b> файловых маркеров вперед. Перемотка ленты на сторону ВОТ последней метки файла.
MTFSR	Перемотка ленты на <b>mt_count</b> записей (блоков).
MTFSS	Перемотка ленты на <b>mt_count</b> установленных маркеров вперед.
MTLOAD	Осуществляет загрузку команд SCSI. Специальные возможности доступны некоторым автозагрузчикам HP. Если <b>mt_count</b> - константа, то MT_ST_HPOUNDER_OFFSET прибавит к ней число, которое передается накопителю для управления автозагрузчиком.
MTLOCK	Блокирует дверцу ленточного накопителя.
MTMKPART	Форматирует ленту на один или два раздела. Если <b>mt_count</b> - не нулевое, этот параметр выдает размер первого раздела, а второй раздел содержит остаток ленты. Если <b>mt_count</b> является нулевым, лента форматируется на один раздел. Эта команда запрещена для накопителя, исключая тот случай, когда накопитель поддерживает создание разделов (см. ниже MT_ST_CAN_PARTITIONS).
MTNOP	Сброс буфера накопителя. Операция должна выполняться до чтения статуса с MTIOCGET.
MTOFFL	Перематывает ленту и отключает устройство.
MTRESET	Возвращает устройству исходное состояние.
MTRETN	Подтягивает ленту в касете.
MTREW	Перематывает ленту назад.
MTSEEK	Переход к блоку ленты с номером <b>mt_count</b> . Эта операция требует одного из двух накопителей: SCSI-2, поддерживающего команду LOCATE (адрес определенного устройства), или Tandberg - совместимого накопителя SCSI-1 (Tandberg, Archive Viper, Wangtek, ...). Номер блока должен быть равен единице, что заранее возвращает MTIOCPOS, если используется адрес определенного устройства. Присваивает длине блока накопителя значение <b>mt_count</b> . Нулевое значение устанавливает режим переменных блоков накопителя.
MTSETBLK	Устанавливает плотность данных на ленте в значение <b>mt_count</b> . Плотность данных, поддерживаемая накопителем, может быть найдена в документации самого устройства.
MTSETDENSITY	Изменение значения активного раздела на значение <b>mt_count</b> . Разделы нумеруются, начиная с нуля. Эта команда только для накопителей, которые имеют поддержку разделов (см. ниже MT_ST_CAN_PARTITIONS).
MTSETPART	Осуществляет выгрузку команд SCSI (но не выталкивает кассету).
MTUNLOAD	Разблокирует дверцу ленточного накопителя.
MTUNLOCK	Устанавливает <b>mt_count</b> маркера файла в текущую позицию на ленте.
MTWSM	Записывает <b>mt_count</b> установленных меток.

Операции с магнитной лентой для установки параметров устройства (для суперпользователя):  
MTSETDRVBUFFER

Установка значения **mt\_count** различных накопителей и параметров драйвера соответственно количеству битов. Это: режим буферизации накопителя (13 битовых параметров драйвера), буфер записи по умолчанию для размера блока, плотность и время ожидания (только для ядра версии 2.1 или более поздних). Одна операция может присваивать устройству значение только одного пункта из списка выше (13 битовых параметров считаются одним пунктом).

Нулевое значение старших 4-х битов будет использовано для установки режима буферизации накопителя. Режимы буферизации:

- 0 Накопитель не сообщит, что запись прошла успешно до тех пор, пока блоки данных фактически не записаны на носитель.
- 1 Накопитель может сообщить, что запись прошла успешно после того, как все данные будут перемещены во внутренний буфер накопителя.
- 2 Накопитель может сообщить о том, что запись прошла успешно после того, как: (а) все данные будут перемещены во внутренний буфер накопителя и (б) все буферизированные данные будут успешно записаны на носитель.

Для контроля записи пороговое значение **mt\_count** должно включать в себя константу **MT\_ST\_WRITE\_THRESHOLD** с количеством блоков в первых 28-и битах. Она содержит количество блоков по 1024 байта (это не физические блоки на диске). Пороговое значение не может превышать размер внутреннего буфера накопителя. (см. выше **DESCRIPTION**)

Для установки и "очистки" битовых параметров **mt\_count** должна включать в себя одну из констант: **MT\_ST\_BOOLEANS**, **MT\_ST\_SETBOOLEANS**, **MT\_ST\_CLEARBOOLEANS**, **MT\_ST\_DEFBOOLEANS** или любую их комбинацию. Использование опций **MT\_ST\_BOOLEANS** может быть определено в соответствующих битах. В работе с опциями **MT\_ST\_SETBOOLEANS** могут применяться выборочные установки, а с опциями **MT\_ST\_DEFBOOLEANS** – выборочная очистка.

Изначально параметры для ленточного устройства установлены равными **MT\_ST\_DEFBOOLEANS**. Неактивное ленточное устройство (например, устройство со второстепенными номерами 64 или 160) активируется, когда изначальные параметры для него определены в первый раз. Активированное устройство получает при загрузке параметры, не установленные явно.

Битовые параметры:

**MT\_ST\_BUFFER\_WRITES** (по умолчанию **true**)

Резервирует все операции записи в режиме фиксированных блоков. Если установленное значение этой опции равно **false** и накопитель использует режим фиксированных блоков, то все операции записи должны выполняться для различных размеров блока. Установленное значение этого параметра должно быть равным **false** для записи надежных многотомных архивов.

**MT\_ST\_ASYNC\_WRITES** (по умолчанию true)

Когда значение true этого параметра установлено, операции записи возвращают данные перемещения на накопитель, если данные заполнили буфер драйвера. Начальная запись определяет, как должен быть заполнен буфер до новых SCSI-команд записи. Любые ошибки, возвращенные накопителем, хранятся до следующей операции. Установленное значение данного параметра должно быть false; это дает возможность записывать многотомные архивы.

**MT\_ST\_READ\_AHEAD** (По умолчанию: true)

Этот параметр приказывает драйверу использовать операцию чтения в режиме фиксированных блоков. Если установленное значение этого параметра равно false и накопитель использует режим фиксированных блоков, то все операции чтения используются для блоков переменной длины.

**MT\_ST\_TWO\_FM** (По умолчанию: false)

Этот параметр изменяет поведение драйвера после закрытия файла. По умолчанию драйвер записывает один маркер файла. Если установленное значение параметра равно true, то драйвер запишет два маркера файла. Замечание: этому параметру не может быть присвоено значение true для ленточных накопителей QIC. Многие современные накопители также находят конец записанных данных, и использование двух меток файла обычно необходимо только при замене ленты в некоторых других системах.

**MT\_ST\_DEBUGGING** (По умолчанию: false)

Этот параметр включает в себя возможность вывода сообщений драйвера при отладке, она эффективна только в том случае, если драйвер был собран с ненулевым значением DEBUG.

**MT\_ST\_FAST\_EOM** (По умолчанию: false)

Этот параметр позволяет посыпать операцию MTEOM накопителю непосредственно, потенциально увеличивая скорость операций; но возникает опасность потерять "дорожку" текущего файла. Если установленное значение параметра MT\_ST\_FAST\_EOM равно false, то драйвер отвечает на запрос MTEOM, перематывая ленту к концу записанных файлов.

**MT\_ST\_AUTO\_LOCK** (по умолчанию false)

Когда этому параметру присвоено значение true и устройство открыто, дверца накопителя заблокирована, когда устройство закрыто, дверца разблокирована.

**MT\_ST\_DEF\_WRITES** (По умолчанию: false)

Параметры чтения и записи на ленту (размер блока, режим, компрессия и т.д.) могут изменяться в случае с различными устройствами, связанными с накопителем, в зависимости от того, как настроено устройство. Этот параметр определяет, будет ли использоваться драйвер при помощи команд SCSI, или накопитель сам автоматически будет выполнять операции,

используя свои возможности. Если установленное значение этой опции равно *false*, то драйвер будет посыпать команды непосредственно устройству. Если установленное значение опции равно *true*, команды SCSI не отправляются до запроса записи. В этом случае программно-аппаратные средства накопителя позволяют выполнять обнаружение структуры ленты во время чтения, а команды SCSI используются только для проверки правильности записи на ленту в соответствии со спецификацией.

**MT\_ST\_CAN\_BSR** (по умолчанию *false*)

Если используется упреждающее чтение, лента должна быть иногда перемотана назад для правильного ее размещения при закрытом устройстве. Для этих целей используется команда SCSI, которая выполняет перемотку ленты назад на одну запись. Некоторые старые накопители не могут правильно обрабатывать эту команду, и этот параметр может использоваться для того, чтобы драйвер данную команду не применял. В конечном итоге, с упреждающим чтением и в режиме фиксированных блоков файл может неверно быть записан на ленту при закрытом устройстве.

**MT\_ST\_NO\_BLKLIMS** (по умолчанию *false*)

Некоторые накопители не принимают SCSI-команду READ BLOCK LIMITS (ограничения чтения блока). Если она используется, то драйвер не исполняет команду. Недостатком этого является невозможность проверки драйвером до отправки команд, поддерживается ли выбранный размер блока накопителем.

**MT\_ST\_CAN\_PARTITIONS** (по умолчанию *false*)

Этот параметр дает возможность поддерживать несколько разделов на ленте. Он применяется ко всем устройствам, связанным с накопителем.

**MT\_ST\_SCSI2LOGICAL** (по умолчанию *false*)

Этот параметр заставляет драйвер использовать адрес логического блока, определенного в стандарте SCSI-2 при выполнении операций поиска и пересчета (с обеими командами MTSEEK и MTIOCPOS и в том случае, когда изменяются разделы ленты). Желательно установить этот параметр, если накопитель поддерживает логические адреса, потому что они рассчитывают также и метки файла.

**MT\_ST\_SYSV** (по умолчанию *false*)

Когда этот параметр установлен, ленточные устройства используют семантику SystemV. В других случаях используется семантика BSD. Основная разница между этими семантиками (когда устройство, открытное для чтения, закрыто) состоит в следующем: по семантике SYSV лента перематывается вперед за следующий маркер файла, если этого не произошло во время использования устройства. В семантике BSD местоположение ленты не меняется.

#### ПРИМЕР

```
struct mttop mt_cmd;
mt_cmd.mt_op = MTSETDRVBUFFER;
mt_cmd.mt_count = MT_ST_BOOLEANS |
                  MT_ST_BUFFER_WRITES |
                  MT_ST_ASYNC_WRITES;
ioctl(fd, MTIOCSTOP, &mt_cmd);
```

Размер блока для устройства может быть установлен с помощью **MT\_ST\_DEF\_BLKSIZE**, а плотность данных может быть установлена с помощью **MT\_ST\_DEFDENSITY**. Значения параметров ORed - с помощью кода операции.

В ядре версии 2.1.x и более поздних время ожидания может быть установлено подкомандой **MT\_ST\_SET\_TIMEOUT** с секундным значением. Долгое время ожидания (используется для перемотки и выполнения других команд, занимающего длительное время) может быть установлено равным **MT\_ST\_SET\_LONG\_TIMEOUT**. Значения, используемые в ядре по умолчанию, слишком велики, поэтому нельзя быть уверенными в том, время ожидания не истечет. Действие драйвера в этом случае может быть приостановлено. Эти команды могут быть использованы для установки оптимальных значений различных накопителей. Установка времени ожидания для одного устройства применяется для всех устройств, связанных с этим накопителями.

#### MTIOCGET - получение статуса

Этот запрос использует аргумент типа (**struct mtget \***).  
/\* структура для MTIOCGET \*/  
struct mtget {  
 long mt\_type;  
 long mt\_resid;  
 /\* следующие регистры являются устройствозависимым \*/  
 long mt\_dsreg;  
 long mt\_gstat;  
 long mt\_erreg;  
 /\* следующие два поля используются не всегда \*/  
 daddr\_t mt\_fileno;  
 daddr\_t mt\_blkno;  
};

**mt\_type** Заголовок файла определяет множество значений **mt\_type**, но текущий драйвер сообщает только об общих типах **MT\_ISSCSI1** (SCSI-1) и **MT\_ISSCSI2** (SCSI-2).  
**mt\_resid** включает в себя номер текущего раздела диска.  
**mt\_dsreg** сообщает о текущих настройках размера блока драйвера (в младших 24-х битах) и плотности (в старших 8-х битах). Эти поля определены в: **MT\_ST\_BLKSIZE\_SHIFT**, **MT\_ST\_BLKSIZE\_MASK**, **MT\_ST\_DENSITY\_SHIFT** и **MT\_ST\_DENSITY\_MASK**.  
**mt\_gstat** сообщает об общем (устройствонезависимом) статусе. Заголовок файла определяет макрос для тестирования:  
    **GMT\_EOF(x)**: текущая позиция сразу после маркера файла (значение всегда становится ложным после выполнения операции **MTSEEK**)  
    **GMT\_BOT(x)**: текущая позиция в начале первого файла (всегда ложно после операции **MTSEEK**)  
    **GMT\_EOT(x)**: текущая позиция в конце ленты.  
    **GMT\_SM(x)**:  
    **GMT\_EOD(x)**: текущая позиция в конце записанных

данных.

**GMT\_WR\_PROT(x)**: накопитель защищен от записи.  
Для некоторых накопителей это может означать, что накопитель не поддерживает запись на носитель этого типа.

**GMT\_ONLINE(x)**:

**GMT\_D\_6250(x), GMT\_D\_1600(x), GMT\_D\_800(x)**: Сообщает о настройках текущей плотности для 9-и дорожек.

**GMT\_DR\_OPEN(x)**: нет ленты в накопителе.

**GMT\_IM REP\_EN(x)**: сообщает о режиме. Этот бит устанавливается, если нет гарантии, что данные были правильно записаны на ленту. Этот бит приобретает нулевое значение, когда драйвер не записывает данные в буфер и накопитель не настроен на буферизацию данных.

**mt\_errreg** Поле, определенное в **mt\_errreg**, возвращает номер ошибки в младших 16-и битах (ошибка определена в **MT\_ST\_SOFTERR\_SHIFT** и **MT\_ST\_SOFTERR\_MASK**).

**mt\_fileno** сообщает текущий номер файла. Значение установлено равным -1, когда номер файла неизвестен (например, после MTBSS или MTSEEK).

**mt\_blkno** сообщает о номере блока в пределах текущего файла. Значение установлено равным -1, когда номер блока неизвестен (например, после MTBSF, MTBSS или MTSEEK).

#### **MTIOCPOS - получить текущую позицию**

Этот запрос использует аргумент типа (**struct mtpos \***) и сообщает текущий номер блока ленты, который необязательно будет равен значению **mt\_blkno**, возвращаемому MTIOCGET. Накопитель должен быть накопителем SCSI-2, чтобы поддерживать команды READ POSITION, или Tandberg-совместимым накопителем SCSI-1 (Tandberg, Archive Viper, Wangtek, ...).

```
/* структура для MTIOCPOS */
struct mtpos {
    long mt_blkno; /* текущий номер блока */
};
```

#### **ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ**

EIO	Запрошеные операции могут быть завершены.
ENOSPC	Операции записи не могут завершиться, потому что лента закончилась.
EACCES	Попытка записать данные на ленту или "очистить" ленту, защищеннную от записи. Эта ошибка не появляется во время использования функции <b>open()</b> .
EFAULT	Параметры команды указывают на память, не принадлежащую вызывающему процессу.
ENXIO	Во время открытия ленточного устройства обнаруживается, что его не существует.
EBUSY	Устройство уже используется, или драйвер не способен выделить буфер.
EOVERFLOW	Попытка чтения или записи блоков переменной длины, больших, чем внутренний буфер драйвера.

**EINVAL**            **ioctl()** был передан неверный аргумент или запрошен неправильный размер блока.

**ENOSYS**          Неизвестный **ioctl()**.

**EROFS**            Попытка открыть накопитель с помощью O\_WRONLY или O\_RDWR, если лента в нем защищена от записи.

**ФАЙЛЫ** /dev/st\* : ленточные устройства SCSI с автоперемоткой  
/dev/nst\* : ленточные устройства SCSI без автоперемотки

**АВТОР** Драйвер был создан Kai Makisara (Kai.Makisara@metla.fi) на основе драйвера, описанного Dwayne Forsyth. В этой работе принимали участие и другие авторы.

**СМ. ТАКЖЕ**        **mt(1)**

Файл README.st в исходных текстах ядра содержит самую свежую информацию о драйвере и его возможных конфигурациях.

#### **ЗАМЕЧАНИЯ**

1. При обмене данными между системами обе системы согласовывают физический размер блока ленты. Параметры накопителя после загрузки часто не являются теми, которые использует большинство операционных систем, работающих с этими устройствами. Большинство систем используют накопители в режиме переменных блоков, если этот режим поддерживается накопителем. Это применимо к большинству современных накопителей, включающих в себя DAT, DLT и т.д. Возможно, целесообразно использовать эти накопители в режиме переменных блоков также и в Linux (т.е., используйте MTSETBLK или MTSETDEFBLK при запуске системы для установки необходимого режима);

2. Многие программы (например, tar) позволяют пользователю определить блокировку в командной строке. Заметьте, что это помогает определить размер физического блока на ленте, но только в режиме переменных блоков;

3. Для использования ленточных накопителей SCSI и базового драйвера SCSI в ядре созданы (или загружены в него как модули): один из драйверов SCSI-адаптера и драйвер ленты SCSI. Если драйвера SCSI-ленты не существует, накопитель становится привилегированным, но поддержка ленты, описанная здесь, вряд ли будет Вами применяться;

4. Драйвер записывает сообщения об ошибках на консоль/в журнал.

Linux 2.0 - 2.2

January 18, 1999

ST(4)

---

## **TTY**

**НАЗВАНИЕ**        tty - управляющий терминал

**ОПИСАНИЕ** Файл **/dev/tty** - это символьный файл с главным числом 5 и второстепенным числом 0, имеющий режим доступа 0666 и владельца-группу root.tty. Это копия устройства, управляющего процессами, если они есть.

В дополнение к вызовам **ioctl()**, поддерживаемым устройством, на которое ссылается **tty**, также возможны следующие запросы **ioctl()**:

**TIOCNOTTY**

Отсоединяет текущий процесс от его управляющего терминала и исключает его из текущей группы процессов, не связывая его с новой группой процессов (т.е., устанавливает в качестве идентификатора группы 0). Этот вызов **ioctl()** работает только с описателями файла, присоединенного к **/dev/tty**; это используется процессами-демонами, когда они вызываются с терминала. Данный процесс пытается открыть

**/dev/tty;** если его открыть удалось, он отсоединяется от терминала с помощью **TIOCNOTTY**, если же операция не удалась, то, очевидно, процесс не присоединен к терминалу и не требует отсоединения.

**ФАЙЛЫ** /dev/tty

**СМ. ТАКЖЕ** **mknod(1)**, **chown(1)**, **getty(1)**, **termios(3)**, **console(4)**, **ttys(4)**

Linux January 21, 1992

TTY (4)

## TTYS

**НАЗВАНИЕ** ttys - последовательные терминальные линии

**ОПИСАНИЕ** **ttys[0-3]** - это символьные устройства для последовательных терминальных линий.

Обычно они создаются так:

```
mknod -m 660 /dev/ttys0 c 4 64 # адрес 0x03f8  
mknod -m 660 /dev/ttys1 c 4 65 # адрес 0x02f8  
mknod -m 660 /dev/ttys2 c 4 66 # адрес 0x03e8  
mknod -m 660 /dev/ttys3 c 4 67 # адрес 0x02e8  
chown root:tty /dev/ttys[0-3]
```

**ФАЙЛЫ** /dev/ttys[0-3]

**СМ. ТАКЖЕ** **mknod(1)**, **chown(1)**, **getty(1)**, **tty(4)**

Linux 19 December 1992

TTYS (4)

## VCS

**НАЗВАНИЕ** vcs, vcса - память виртуальной консоли

**ОПИСАНИЕ** **/dev/vcs0** - это устройство, с основным числом 7 и второстепенным - 0; обычно права его доступа - 0644, и владелец - root.tty. Устройство ссылается на память открытого в этот момент виртуального терминала.

**/dev/vcs[1-63]** - это символьные устройства для виртуальных терминалов, основное число которых - 7 и второстепенный - от 1-ого до 63-х; права доступа этого устройства - 0644, и владелец - root.tty. **/dev/vcsa[0-63]** является таким же устройством, но с атрибутами и приставкой из четырех байтов, задающих размеры экрана и одну из следующих позиций курсора: *lines*, *columns*, *x*, *y*. *x = y = 0* означает верхний левый угол экрана.

Это заменяет I. ioctl образов экрана **console(4)**, таким образом, системный администратор может контролировать доступ к используемой файловой системе.

Устройства для первых восьми консолов могут быть созданы с помощью следующих команд:

```
for x in 0 1 2 3 4 5 6 7 8; do  
    mknod -m 644 /dev/vcs$x c 7 $x;  
    mknod -m 644 /dev/vcsa$x c 7 ${$x+128};  
done  
chown root:tty /dev/vcs*
```

Запросы I. ioctl не поддерживаются.

## ПРИМЕРЫ

Вы можете создать образ vt3, переключившись на vt1 и написав *cat /dev/vcs3 >foo*. Обратите внимание, что вывод не будет содержать символов перевода строки. Поэтому можно использовать команду *fold -w 81 /dev/vcs3 | lpr* или *setterm -dump 3 -file /proc/self/fd/1*.

Устройство */dev/vcsa0* используется для поддержки Braille. Эта программа выводит символ и атрибуты экрана, находящиеся под курсором, со второй виртуальной консоли и затем меняет цвет фона в том же месте.

```
int main() {  
    int fd;
```

```

        struct {char lines, cols, x, y;} scrn;
        char ch, attrib;
        fd = open("/dev/vcsa2", O_RDWR);
        (void)read(fd, &scrn, 4);
        (void)lseek(fd, 4 + 2*(scrn.y*scrn.cols + scrn.x), 0);
        (void)read(fd, &ch, 1);
        (void)read(fd, &attrib, 1);
        printf("ch='%c' attrib=0x%02x\n", ch, attrib);
        attrib ^= 0x10;
        (void)lseek(fd, -1, 1);
        (void)write(fd, &attrib, 1);
        return 0;
    }
}

```

**ФАЙЛЫ** /dev/vcs[0-63]  
           /dev/vcsa[0-63]

**АВТОР** Andries Brouwer <aeb@cwi.nl>  
**ПРЕДЫСТОРИЯ** Впервые появилось в версии ядра 1.1.92.  
**СМ. ТАКЖЕ** **console(4)**, **tty(4)**, **ttys(4)**, **selection(1)**  
Linux February 19, 1995 VCS (4)

---

## WAVELAN

**НАЗВАНИЕ** wavelan - драйвер устройства AT&T GIS WaveLAN ISA.  
**СИНТАКСИС** **insmod wavelan\_cs.o [io=B,B..] [irq=I,I..] [name=N,N..]**  
**ОПИСАНИЕ** wavelan - это низкоуровневый драйвер для беспроводных ethernet-устройств NCR / AT&T / Lucent **WaveLAN ISA** и Digital (DEC) **RoamAboutDS**. Этот драйвер доступен как модуль или может быть включен в ядро при сборке. Драйвер поддерживает несколько карт (до 4-х) в любых сочетаниях. Каждой карте выделяется доступное ethernet-устройство (eth0..eth#), кроме случаев, когда название устройства не указано конкретно (см. выше). Название устройства будет указано в журнале ядра вместе с MAC-адресом, NWID и частотой работы карты.

**ПАРАМЕТРЫ**

Эта часть применима только к модулю ядра (параметры передаются с помощью **insmod(8)** из командной строки). Если драйвер находится в ядре, используйте синтаксис **ether=IRQ,IO,NAME** в командной строке ядра.

**io** Определяет список адресов, по которым производится поиск устройств (они установлены с помощью переключателей на карте). Если список адресов не задан, драйвер просканирует адреса 0x390 и 0x3E0, которые могут конфликтовать с другими устройствами.

**irq** Определяет список irq, который должна использовать каждая карта (данные об irq сохраняются для дальнейшего использования).

**name** Определяет список с именами устройств, которые будут использоваться для каждой карты (эти имена использует также **ifconfig(8)**).

**БЕСПРОВОДНОЕ СОЕДИНЕНИЕ**

Используйте **iwconfig(8)** для управления беспроводными соединениями.

**NWID (или домен)**

Определяет сетевой ID [0-FFFF] или отключает его [off]. Поскольку NWID сохраняется в постоянной памяти карты, он может быть многократно использован в дальнейшем.

**Частота и каналы**

Для карт типа 2.4GHz 2.00 Вы должны установить частоту, определяемую одним из десяти следующих каналов: 2.412, 2.422, 2.425, 2.4305, 2.432, 2.442, 2.452, 2.460, 2.462 и 2.484, - или напрямую указать ее величину. Выбранная

частота будет установлена, и это значение записано в память. Доступность частоты зависит от радионастроек...

#### **Статистика**

Определяет список сетевых MAC-адресов для драйвера (до 8-и) и выводит сведения о качестве последнего соединения по каждому из адресов (см. **iwspy(8)**).

#### **/proc/net/wireless**

*status* - это статус состояния, сообщаемый модемом. *Link quality* сообщает качество модуляции (direct sequence spread spectrum) [максимум = 16]. *Level* и *Noise* сообщает об уровне сигнала и шумах в линии [максимум = 64]. Счетчики *crypt discarded packet* и *misc discarded packet* еще не действуют.

#### **ВЫЗОВЫ IOCTL**

Вы можете использовать **iwpriv(8)** для управления вызовами ioctl's.

#### **Качество и уровень порога**

Запуская эту опцию, Вы задаете качество и уровень порога, с которыми работает модем.

#### **Гистограмма**

Позволяет установить интервалы между уровнями сигнала и количеством пакетов, посланных в каждый из этих интервалов. Эта часть может быть использована для вычисления среднего значения и стандартного отклонения уровня сигнала от нормы.

#### **СПЕЦИФИЧЕСКИЕ ЗАМЕЧАНИЯ**

Этот драйвер не может распознать некоторые **ne-NCR/ATT&T/Lucent** Wavelan-карты. Если Вы выбрали именно их, взгляните в исходный код и добавьте Вашу карту в процедуру распознавания.

**СМ. ТАКЖЕ** **wavelan\_cs(4)**, **ifconfig(8)**, **insmod(8)**, **iwconfig(8)**,  
**iwspy(8)**, **iwpriv(8)**

Univ. of Sydney (Basser Dep. 10/22/96 Sci.)

WAVELAN(4)