


Глава 4. Управление устройствами.

4.1. Установка нового оборудования.



Установка нового оборудования

- Подключение оборудования может быть
- произведено двумя основными способами:
 - в слоты расширения
 - к внешним шинам или портам
- Список прерываний - `/proc/interrupts`
- Адреса ввода-вывода - `/proc/ioports`
- Каналы DMA - `/proc/dma`

Подключение оборудования в компьютерах может быть произведено двумя основными способами:

1. установкой плат дополнительного оборудования в слоты расширения;
2. с помощью подключения к внешним шинам или портам.

При установке оборудования в слоты расширения на материнской плате это оборудование непосредственно подключается к локальным шинам компьютера.

Оборудование, подключаемое к внешним шинам, часто подключается к системе посредством плат расширения, включенных в локальную шину.

Пример: устройства SCSI, подключаемые к SCSI шине, соединенной со SCSI хост адаптером, который чаще всего реализован, как плата расширения, включаемая в локальную шину.

Как локальные, так и внешние шины компьютера с помощью согласующего оборудования и контроллеров управления подключаются непосредственно к центральному процессору.

Примечание: Разграничение на локальные и внешние шины условно тем более, например, что контроллеры EIDE и часто даже SCSI хост адаптеры размещаются на материнской плате.

Для того, чтобы дополнительное оборудование могло осуществлять операции ввода-вывода ему должны быть выделены следующие ресурсы:

1. IRQ – канал прерывания;
2. IO/Base – базовый адрес ввода-вывода;

Глава 4. Управление устройствами.

3. DMA – канал прямого доступа к ОЗУ.

Прерывание (IRQ) идентифицируется его номером, и позволяет сигнализировать о завершении операции ввода-вывода, осуществляемой данным устройством.

При получении прерывания процессор должен приостановить выполнение текущего задания и переключиться на обработку прерывания с помощью специальной программы (interrupt handler).

Получить список прерываний, использованных системой, можно изучив содержимое файла `/proc/interrupts`.

Пример:

```

          CPU0
0:      1255430      IO-APIC-edge  timer
1:        8232      IO-APIC-edge  keyboard
2:         0             XT-PIC    cascade
8:         2      IO-APIC-edge  rtc
9:         0      IO-APIC-level  acpi
12:     18877      IO-APIC-edge  PS/2 Mouse
14:     23786      IO-APIC-edge  ide0
15:    1091581      IO-APIC-edge  ide1
16:         0      IO-APIC-level  usb-uhci
17:     22748      IO-APIC-level  Intel ICH4
18:         0      IO-APIC-level  usb-uhci
19:         0      IO-APIC-level  usb-uhci
20:         9      IO-APIC-level  eth0
21:         7      IO-APIC-level  O2 Micro, Inc. OZ6912 Cardbus Controller
22:         2      IO-APIC-level  ohci1394
23:    640464      IO-APIC-level  ehci_hcd
NMI:         0
LOC:    1255407
ERR:         0
MIS:         0
```

Для большинства старых устройств должны устанавливаться уникальные номера прерываний, число которых ограничено.

Многие современные устройства позволяют разделять прерывания, то есть два различных устройства могут использовать одно и то же прерывание.

Не все возможные в данной системе прерывания следует выделять для устройств расширения.

Пример: 2-е и 9-е прерывание представляют собой одно и то же, так как 2-е прерывание каскадировано с 9-м. Некоторые прерывания традиционно зарезервированы для определенных устройств. Следующие прерывания используют:

1. 0 – системный таймер;
2. 1 – клавиатура;
3. 3 и 4 – второй и первый последовательные порты соответственно;
4. 5 и 7 – второй и первый параллельные порты соответственно;
5. 14 и 15 – первичный и вторичный IDE контроллеры.

Регистры памяти, имеющиеся в устройствах расширения, отображаются в специальную

Глава 4. Управление устройствами.

область ОЗУ, доступную ядру операционной системы и называемую памятью устройств.

Эта память предназначена для реализации операций ввода-вывода, то есть в нее записывается передаваемая при этих операциях информация.

Базовый адрес ввода-вывода IO/Base указывает начало области памяти для операций ввода-вывода данного устройства.

Принято записывать адреса ввода-вывода в шестнадцатеричном виде.

Увидеть занятые устройствами адреса ввода-вывода можно в файле `/proc/ioports`.

Пример:

```
$ cat /proc/ioports
0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0070-007f : rtc
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
0376-0376 : ide1
03c0-03df : vesafb
03f6-03f6 : ide0
0cf8-0cff : PCI conf1
4000-40ff : PCI CardBus #02
4400-44ff : PCI CardBus #02
cc00-cc7f : VIA Technologies, Inc. IEEE 1394 Host Controller
cf00-cf3f : Intel Corp. 82801BD PRO/100 VE (CNR) Ethernet Controller
    cf00-cf3f : eepr0100
d400-d4ff : Intel Corp. 82801DB AC'97 Audio Controller
    d400-d4ff : Intel ICH4
d800-d8ff : Intel Corp. 82801DB AC'97 Modem Controller
dc00-dc7f : Intel Corp. 82801DB AC'97 Modem Controller
dea0-debf : Intel Corp. 82801DB USB (Hub #1)
    dea0-debf : usb-uhci
dec0-dedf : Intel Corp. 82801DB USB (Hub #2)
    dec0-dedf : usb-uhci
df00-df3f : Intel Corp. 82801DB AC'97 Audio Controller
    df00-df3f : Intel ICH4
df40-df5f : Intel Corp. 82801DB USB (Hub #3)
    df40-df5f : usb-uhci
dff0-dff7 : Intel Corp. 82852/855GM Integrated Graphics Device
e800-e81f : Intel Corp. 82801DB/DBM SMBus Controller
ffa0-ffaf : Intel Corp. 82801DBM Ultra ATA Storage Controller
    ffa0-ffa7 : ide0
    ffa8-ffaf : ide1
```

Каналы DMA позволяют устройствам производить операции обмена с памятью самостоятельно без обращения к центральному процессору, что существенно снижает нагрузку на процессор.

Каждое устройство, использующее DMA, должно иметь собственный канал DMA, не разделяя его с другими устройствами.

Глава 4. Управление устройствами.

Для просмотра используемых каналов DMA следует изучить файл `/proc/dma`:

Пример:

```
$ cat /proc/dma  
4: cascade
```

4.2. Работа с модулями ядра.



Работа с модулями ядра

- Модули ядра представляют собой объектные файлы с кодом, который можно подключать к работающему ядру Linux
- Модули располагаются в каталоге `/lib/modules/имя_ядра`
- Команда `uname -r` позволяет узнать имя ядра, запущенного в данный момент

Современные ядра Linux построены в соответствии с модульной архитектурой, что позволяет изменять функциональность ядра в работающей системе, а также включать в ядро и удалять драйверы устройств “на лету”.

Модули ядра представляют собой объектные файлы с кодом, который можно подключать к работающему ядру Linux.

Они образуются при сборке ядра (она будет рассмотрена в следующей главе).

1. В ядрах стабильной ветви версии до 2.6.0 файлы модулей ядра имеют суффикс `.o`
2. В ядрах серии 2.6 и выше - `.ko` (kernel object).

Модули располагаются в каталоге `/lib/modules` :

Пример:

```
$ ls -F /lib/modules/  
2.4.22-gentoo-r5/
```

Примечание: Обратите внимание, что в каталоге `/lib/modules` имеется подкаталог (или подкаталоги) с именем ядра. Это связано с тем, что может иметься несколько различных ядер для различных целей. В данном случае имеется единственный каталог с модулями ядра `2.4.22-gentoo-r5`.

Имена подкаталогов `/lib/modules` всегда точно соответствуют именам ядер, имеющихся в системе.

Команда `uname -r` позволяет узнать имя ядра, запущенного в данный момент.

Пример:

```
$ uname -r  
2.4.22-gentoo-r5
```

Глава 4. Управление устройствами.

В каталоге `/lib/modules/<kernelname>` могут находиться не только модули ядра, но и драйверы для устройств, разработанные независимыми от команды разработчиков ядра Linux поставщиками.

Пример:

```
$ ls -F /lib/modules/`uname -r`
build@ modules.dep      modules.isapnpmap      modules.pnpbiosmap
kernel/ modules.generic_string modules.parportmap     modules.usbmap
misc/   modules.ieee1394map modules.pciomap        pcmcia/
nvidia/
```

Примечание: В этом примере показано содержимое подкаталога `/lib/modules`, имя которого задается с помощью командной подстановки и соответствует имени текущего ядра. Каталог `nvidia` содержит драйвер для видеокарты NVidia, полученный от производителя видеокарты.

В зависимости от ветви используемого ядра, а также от обновлений (patches), наложенных на ядро, раскладка каталогов, приведенных ниже может существенно изменяться.

Основная часть модулей ядра, среди которых, в частности, драйверы устройств, находится в подкаталоге `kernel`.

Пример:

```
$ ls -F /lib/modules/`uname -r`/kernel
arch/  crypto/  drivers/  fs/  lib/  net/
```

В подкаталоге `fs` находятся драйверы, модули расширения функциональности и иное программное обеспечение для различных файловых систем.

Пример:

```
$ ls -F /lib/modules/`uname -r`/kernel/fs
affs/      coda/      freevxfs/  intermezzo/  lockd/      nfsd/      smbfs/
befs/      cramfs/    hfs/       jbd/         ntfs/       supermount/  vfat/
bfs/       efs/       hfsplus/   jffs/        msdos/      sysv/       xfs/
binfmt_aout.o  ext3/      hpfs/      jffs2/       ncpfs/      quota_v2.o  udf/
binfmt_misc.o  fat/       ibu/       jfs/         nfs/        romfs/      ufs/
```

Подкаталог `net` содержит программное обеспечение для поддержки сетевой инфраструктуры.

Пример:

```
$ ls -F /lib/modules/`uname -r`/kernel/net
8021q/      ax25/      core/      ipsec/      ipx/      lapb/      sched/      x25/
appletalk/  bluetooth/  decnet/    ipv4/      irda/      netrom/    sunrpc/
atm/        bridge/    econet/    ipv6/      khttpd/    rose/      wanrouter/
```

В подкаталоге `drivers` находятся драйверы для различных устройств.

Пример:

```
$ ls -F /lib/modules/`uname -r`/kernel/drivers
acpi/      bluetooth/  i2c/       input/      message/    parport/    sensors/      usb/
atm/       char/       ide/       isdn/       mtd/        pcmcia/     sound/        video/
block/     hotplug/    ieee1394/  media/      net/        scsi/       telephony/
```

Работа с модулями ядра

- Команды:
 - `lsmod insmod rmmod`
 - `modinfo`
 - `modprobe`
- Файлы:
 - `/proc/modules` — загруженные модули
 - `/etc/modprobe.d` — параметры загрузки модулей
 - `/etc/modules-load.d` — загрузка модулей при старте системы

Для получения списка модулей, установленных в настоящее время в ядро, следует выполнить команду `/sbin/lsmod`.

Пример:

```
$ /sbin/lsmod
Module                Size  Used by    Not tainted
sd_mod                12556   0 (autoclean) (unused)
sr_mod                15128   0 (autoclean) (unused)
i830                  70112  13
agpgart               42168  12 (autoclean)
usb-storage           118192   0 (unused)
hw_random              2968    0 (unused)
i810_rng              2784    0 (unused)
i810_audio            26184    0
ac97_codec            13224    0 [i810_audio]
soundcore              3972    2 [i810_audio]
ohci1394              32328    0 (unused)
ieee1394             187300    0 [ohci1394]
eepro100              20468    1
mii                    2528    0 [eepro100]
ide-scsi              10128    0
scsi_mod              105876   4 [sd_mod sr_mod usb-storage ide-scsi]
ds                     7240    1
yenta_socket          10752    1
pcmcia_core           45536    0 [ds yenta_socket]
keybdev               1920    0 (unused)
mousedev              4212    1
hid                   20996    0 (unused)
input                  3552    0 [keybdev mousedev hid]
uhci                   27152    0 (unused)
ehci-hcd              17992    0 (unused)
usbcore               63904    1 [usb-storage hid uhci ehci-hcd]
```

Глава 4. Управление устройствами.

Примечание: По сути программа `/sbin/lsmmod` просто выводит в форматированном виде информацию из файла `/proc/modules`.

Пример:

```
$ cat /proc/modules
ext3                97416      1 (autoclean)
jbd                 43312      1 (autoclean) [ext3]
sd_mod              12556      2 (autoclean)
sr_mod              15128      0 (autoclean) (unused)
i830                 70112     13
agpgart             42168     12 (autoclean)
usb-storage         118192      1
hw_random            2968      0 (unused)
i810_rng             2784      0 (unused)
i810_audio           26184      0
ac97_codec           13224      0 [i810_audio]
soundcore             3972      2 [i810_audio]
ohci1394             32328      0 (unused)
ieee1394            187300      0 [ohci1394]
eepro100             20468      1
mii                  2528      0 [eepro100]
ide-scsi             10128      0
scsi_mod            105876      4 [sd_mod sr_mod usb-storage ide-scsi]
ds                   7240      1
yenta_socket         10752      1
pcmcia_core           45536      0 [ds yenta_socket]
keybdev              1920      0 (unused)
mousedev             4212      1
hid                  20996      0 (unused)
input                3552      0 [keybdev mousedev hid]
uhci                 27152      0 (unused)
ehci-hcd             17992      0 (unused)
usbcore              63904      1 [usb-storage hid uhci ehci-hcd]
```

Для отображения доступной информации о модуле используйте команду `/sbin/modinfo`.

Пример:

```
$ /sbin/modinfo uhci
filename:      /lib/modules/2.4.22-gentoo-r5/kernel/drivers/usb/host/uhci.o
description:   "USB Universal Host Controller Interface driver"
author:        "Linus 'Frodo Rabbit' Torvalds, Johannes Erdfelt, Randy Dunlap,
Georg Acher, Deti Fliegl, Thomas Sailer, Roman Weissgaerber"
license:       "GPL"
parm:          debug int, description "Debug level"
```

Примечание: В этом примере команда `modinfo` выдала информацию о модуле `uhci`, являющимся универсальным драйвером для USB хост контроллера.

Информация о модулях включается в сами модули, хотя она при необходимости может быть скрыта.

Эта информация стандартизирована и представлена следующими полями:

1. `author` - автор модуля;
2. `description` – описание модуля;

Глава 4. Управление устройствами.

3. `license` – лицензия;
4. `param` – дополнительные параметры;
5. `depend` – зависимости модуля от других модулей;
6. `alias` – псевдоним для модуля.

При использовании ядер серии 2.6 для отображения информации лишь по одному или нескольким полям следует использовать опцию `-F` команды `modinfo` с указанием требуемых полей.

При использовании более старой версии `modinfo` можно указывать опции:

1. `-a` – автор;
2. `-d` – описание;
3. `-l` – лицензия;
4. `-n` – имя файла модуля;
5. `-p` – параметры.

Новые модули могут быть загружены в работающее ядро Linux с помощью команды `/sbin/insmod`.

Установить модуль в ядро может только суперпользователь. В качестве аргумента для команды `insmod` нужно указывать имя файла модуля.

Для удаления модуля из ядра предназначена команда `/sbin/rmmod`.

Пример:

```
# rmmod -v parport
Checking parport for persistent data

# lsmod
Module                Size  Used by    Not tainted
ext3                   97416   1  (autoclean)
jbd                    43312   1  (autoclean) [ext3]
```

Примечание: В этом примере команда `rmmod` удалила драйвер параллельного порта, что демонстрирует сокращенный листинг, выведенный командой `lsmod`. Опция `-v` команды `rmmod` была использована для переключения ее в режим работы с повышенной информативностью вывода.

Часто модули ядра для работоспособности требуют наличия других модулей ядра.

В каталоге, содержащем модули ядра (`/lib/modules/`uname -r``), можно обнаружить текстовый файл `modules.dep`. Он содержит имена файлов модулей и через двоеточие имена файлов модулей, от которых первые зависят.

Файл `modules.dep` генерируется с помощью утилиты `/sbin/depmod`.

Вместо использования команд `insmod` и `rmmod` предпочтительнее использовать команду `modprobe`, обеспечивающую более интеллектуальное поведение.

Эта команда использует файл `modules.dep` для автоматического учета зависимостей модулей, что позволяет загружать модули ядра не задумываясь об их зависимостях. Они будут учтены и требуемые модули будут загружены.

Глава 4. Управление устройствами.

По умолчанию команда `modprobe` устанавливает модуль в ядро, но будучи вызванной с опцией `-r` она способна удалять модули.

Пример:

```
ifconfig eth0 down
modprobe -r eeepro100
modprobe el100
ifconfig eth0 up
```

Примечание: В этом примере первая команда отключила сетевой интерфейс `eth0`. Далее был выгружен модуль ядра – драйвер сетевого адаптера `Intel EtherExpress eeepro100`. При этом был автоматически выгружен модуль ядра, требуемый для работы модуля `eeepro100`. После этого был загружен другой модуль с альтернативным драйвером для этого же сетевого адаптера. Последняя команда вновь активизирует сетевой интерфейс.

Использование опции `-a` позволяет установить сразу несколько модулей, имена которых заданы в качестве аргумента.

Если модуль требует передачи ядру некоторой информации, то ее можно указывать в качестве аргументов команды `modprobe`.

Пример: для установки драйвера сетевой платы `ISA NE2000` требуется указать ее `IRQ` и `IO/Base`:

```
# modprobe ne irq=5 io=0x320
```

Опция `-c` позволяет получить полный список настроек команды `modprobe`.

Он определяется специальными конфигурационными файлами в каталоге `/etc/modprobe.d`

Примечание: В разных дистрибутивах `GNU/Linux` местоположение файлов конфигурации может отличаться.

Одним из главных предназначений файла конфигурации в `/etc/modprobe.d` является обеспечение передачи параметров, требуемых для модулей.

Помимо этого в данном файле можно задавать команды, которые надо выполнить до или после загрузки или удаления модулей.

Также в этих файлах могут задаваться псевдонимы для модулей, делающие работу с ними более удобной.

В каталоге `/etc/modules-load.d/` можно создать файлы со списком модулей, которые должны быть загружены при старте системы.

Обычно в этих файлах прописываются модули драйверов для тех устройств, которые не поддерживают `PnP`.

4.3. UDEV и каталоги /dev и /sys.



UDEV и каталоги /dev и /sys

- /dev — содержит файлы устройств
- /sys — структуры ядра в виде файлов для управления устройствами
- UDEV — работающая в пространстве пользователя система, с помощью которой системный администратор может создавать обработчики событий

Для работы с устройствами в Linux имеются два каталога:

1. /dev – каталог, в котором по умолчанию создаются файлы устройств. Эти файлы устройств позволяют нам «общаться» с устройствами посредством стандартных системных вызовов для работы с файлами: открыть файл `open()`, считать `read()` или записать `write()` данные.

2. /sys – виртуальная файловая системы, которая представляет структуры ядра для работы с устройствами в виде файлов.

В первых ОС на основе ядер Linux каталога /sys не было, а в каталоге /dev файлы устройств создавались вручную или во время инсталляции системы. В результате имелись файлы устройств, а самих устройств не было. Другой проблемой было отсутствие поддержки горячего подключения/отключения (hotplug) устройств.

Для решения проблемы с горячим подключением и «лишними» файлами устройств изначально использовались такие утилиты как `devfs`, `hotplug` и `HAL`

Начиная с ядер версии 2.5 был представлен новый механизм — `udev`.

В апреле 2012 исходный код `udev` слился с исходным кодом `systemd`

UDEV — работающая в пространстве пользователя система, с помощью которой системный администратор может создавать обработчики событий.

События, получаемые UDEV, обычно генерируются ядром Linux в ответ на физические события, происходящие с периферийными устройствами. Например, при обнаружении периферийных устройств или "горячем" подключении UDEV может выполнить определённые действия, в том числе и вернуть управление ядру, если необходима загрузка модулей или прошивок.

Подобно предшественникам, утилитам `devfsd` и `hotplug`, `udev` управляет файлами

Глава 4. Управление устройствами.

устройств в каталоге `/dev`, добавляя их, переименовывая и создавая символические ссылки. `udev` полностью замещает функционал `hotplug` и `hwdetect`.

Благодаря UDEV в каталоге `/dev` находятся файлы только тех устройств, которые в настоящий момент подключены к системе. Каждое устройство имеет свой соответствующий файл. Если устройство отключается от системы, то данный файл удаляется.

Содержимое каталога `/dev` хранится на виртуальной файловой системе, и все файлы, находящиеся в нём, создаются при каждом запуске системы.

Обработка событий в UDEV происходит параллельно, что теоретически улучшает производительность старых систем. С другой стороны, это может усложнить администрирование. Так, при перезапуске системы порядок загрузки модулей ядра может измениться, а при наличии в машине нескольких блочных устройств могут поменяться названия их файлов. Например, для системы с двумя жёсткими дисками файл `/dev/sda` после перезагрузки может превратиться в `/dev/sdb`

UDEV входит в состав `systemd` и установлен по умолчанию. Подробнее см. `systemd-udev.service(8)`.

Существует также отдельный от `systemd` форк, который можно установить с пакетом `eudev` или `eudev-git`

Для настройки `udev` создаются специальные правила. Подробности см. `Writing udev rules` стр. 288

Правила UDEV

- С помощью правил вы можете настроить как реагировать системе на подключение или отключение устройств
 - «Заводские» правила находятся в каталоге `/lib/udev/rules.d/`
 - Пользовательские правила UDEV следует располагать в каталоге `/etc/udev/rules.d`
 - Файлы с правилами должны иметь названия с суффиксом `.rules`

Правила должны располагаться в каталоге `/etc/udev/rules.d/` и иметь названия с суффиксом `.rules`

```
$ cat /etc/udev/rules.d/70-persistent-ipoib.rules
# This is a sample udev rules file that demonstrates how to get udev to
# set the name of IPoIB interfaces to whatever you wish.  There is a
# 16 character limit on network device names.
#
# Important items to note: ATTR{type}=="32" is IPoIB interfaces, and the
# ATTR{address} match must start with ?* and only reference the last 8
# bytes of the address or else the address might not match the variable QPN
# portion.
#
# Modern udev is case sensitive and all addresses need to be in lower case.
#
# ACTION=="add", SUBSYSTEM=="net", DRIVERS=="?*", ATTR{type}=="32",
ATTR{address}=="?*00:02:c9:03:00:31:78:f2", NAME="mlx4_ib3"
```

Мониторинг устройств

- udevadm предназначена для мониторинга и управления UDEV
- Для получения списка устройств PCI и USB вы можете воспользоваться утилитами `lspci` и `lsusb`

Утилита `udevadm` предназначена для мониторинга и управления UDEV.

Пример: создадим специальные правила для подключения USB Flash накопителей определенной модели. Во-первых понаблюдаем за процессом подключения USB Flash:

```
# udevadm monitor
monitor will print the received events for:
UDEV - the event which udev sends out after rule processing
KERNEL - the kernel uevent

KERNEL[158.634005] add          /devices/pci0000:00/0000:00:0b.0/usb1/1-1 (usb)
KERNEL[158.637827] add          /devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0
(usb)
...
UDEV [161.309815] add
/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/
block/sdb (block)
UDEV [162.368973] add
/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/
block/sdb/sdb3 (block)
UDEV [162.519992] add
/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/
block/sdb/sdb2 (block)
UDEV [162.574735] add
/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/
block/sdb/sdb1 (block)
```

Посмотрим характеристики устройства. Обратите внимание, что к тем путям, что выводит `udevadm` нужно просто в начале дописать `/sys`:

```
# ls
/sys/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/
block/sdb
alignment_offset  discard_alignment  hidden            power            sdb1            stat
bdi               events             holders           queue            sdb2            subsystem
```

Глава 4. Управление устройствами.

capability	events_async	inflight	range	sdb3	trace
dev	events_poll_msecs	integrity	removable	size	uevent
device	ext_range	mq	ro	slaves	

```
# ls /sys/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0
authorized          bInterfaceProtocol ep_01      power
bAlternateSetting    bInterfaceSubClass ep_81      subsystem
bInterfaceClass      bNumEndpoints      host3      supports_autosuspend
bInterfaceNumber     driver              modalias   uevent

# ls /sys/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/driver -l
lrwxrwxrwx. 1 root root 0 Feb  6 12:28
/sys/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/driver
-> ../../../../../../../bus/usb/drivers/usb-storage
```

Или более простой путь к устройству:

# ls /sys/block/sdb/					
alignment_offset	discard_alignment	hidden	power	sdb1	stat
bdi	events	holders	queue	sdb2	subsystem
capability	events_async	inflight	range	sdb3	trace
dev	events_poll_msecs	integrity	removable	size	uevent
device	ext_range	mq	ro	slaves	

```
# cat /sys/block/sdb/size
15769600
```

```
# cat /sys/block/sdb/uevent
MAJOR=8
MINOR=16
DEVNAME=sdb
DEVTYPE=disk
```

```
# cat /sys/block/sdb/removable
1
```

```
# cat /sys/block/sda/removable
0
```

Теперь мы с помощью udevadm определим производителя устройства:

```
# # udevadm info /dev/sdb | egrep 'VENDOR|SUBSYS'
E: ID_VENDOR=Generic
E: ID_VENDOR_ENC=Generic\x20
E: ID_VENDOR_ID=cd12
E: SCSI_VENDOR=Generic
E: SCSI_VENDOR_ENC=Generic\x20
E: SUBSYSTEM=block
```

Или

```
# udevadm info -a /dev/sdb | egrep '/devices/|Vendor|SUBSYSTEM'
looking at device
'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/
block/sdb':
    SUBSYSTEM=="block"
looking at parent device
'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0':
    SUBSYSTEMS=="scsi"
looking at parent device
'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0':
```

Глава 4. Управление устройствами.

```
SUBSYSTEMS=="scsi"
looking at parent device
'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3':
SUBSYSTEMS=="scsi"
looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0':
SUBSYSTEMS=="usb"
looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1/1-1':
SUBSYSTEMS=="usb"
ATTRS{idVendor}=="cd12"
looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1':
SUBSYSTEMS=="usb"
ATTRS{idVendor}=="1d6b"
looking at parent device '/devices/pci0000:00/0000:00:0b.0':
SUBSYSTEMS=="pci"
looking at parent device '/devices/pci0000:00':
SUBSYSTEMS==""
```

Создаем правило и проверяем его:

```
# cat /etc/udev/rules.d/10-myflash.rules
ACTION=="add", SUBSYSTEM=="block", ATTRS{idVendor}=="cd12", RUN+="/usr/bin/chgrp
myflash /dev/%k", SYMLINK+="myflsh%n"
```

Не забудьте создать группу:

```
# groupadd myflash
# gpasswd -a admuser myflash

# ls -l /dev/sdb*
brw-rw----. 1 root myflash 8, 16 Feb  6 14:56 /dev/sdb
brw-rw----. 1 root myflash 8, 17 Feb  6 14:56 /dev/sdb1
brw-rw----. 1 root myflash 8, 18 Feb  6 14:56 /dev/sdb2
brw-rw----. 1 root myflash 8, 19 Feb  6 14:56 /dev/sdb3

# ls -l /dev/myflsh*
lrwxrwxrwx. 1 root root 3 Feb  6 14:56 /dev/myflsh -> sdb
lrwxrwxrwx. 1 root root 4 Feb  6 14:56 /dev/myflsh1 -> sdb1
lrwxrwxrwx. 1 root root 4 Feb  6 14:56 /dev/myflsh2 -> sdb2
lrwxrwxrwx. 1 root root 4 Feb  6 14:56 /dev/myflsh3 -> sdb3

$ dd if=/dev/sdb1 of=/dev/null count=10
10+0 records in
10+0 records out
5120 bytes (5.1 kB, 5.0 KiB) copied, 0.0251715 s, 203 kB/s

$ dd if=/dev/sda1 of=/dev/null count=10
dd: failed to open '/dev/sda1': Permission denied
```

Для получения списка устройств PCI и USB вы можете воспользоваться утилитами `lspci` и `lsusb`:

```
$ lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:02.0 VGA compatible controller: VMware SVGA II Adapter
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet
Controller (rev 02)
00:04.0 System peripheral: InnoTek Systemberatung GmbH VirtualBox Guest Service
```


Глава 4. Управление устройствами.

```
00:05.0 Multimedia audio controller: Intel Corporation 82801AA AC'97 Audio
Controller (rev 01)
00:06.0 USB controller: Apple Inc. KeyLargo/Intrepid USB
00:07.0 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:0b.0 USB controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family)
USB2 EHCI Controller
00:0d.0 SATA controller: Intel Corporation 82801HM/HEM (ICH8M/ICH8M-E) SATA
Controller [AHCI mode] (rev 02)
00:0e.0 Non-Volatile memory controller: InnoTek Systemberatung GmbH Device 4e56
```

```
$ lsusb
Bus 001 Device 003: ID cd12:ef18 SMART TECHNOLOGY INDUSTRIAL LTD.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

```
$ lsusb -v | head -5
```


```
Bus 001 Device 003: ID cd12:ef18 SMART TECHNOLOGY INDUSTRIAL LTD.
Device Descriptor:
  bLength                18
  bDescriptorType         1
```

Альтернативно для просмотра информации об устройствах вы можете использовать следующие утилиты:

- `lshw`
- `hwinfo`
- `inxi`
- `dmidecode`

4.4. Работа с устройствами.

4.4.1. Диски и другие накопители.



Диски и другие накопители

- SCSI диски - /dev/sda, sdb, ...
- SD/MMC - /dev/mmcblk0, mmcblk1, ...
- Virtio - /dev/vda, vdb, ...
- Контроллеры NVMe - /dev/nvme0, nvme1, ...
- Накопители NVMe - /dev/nvme0n1, nvme0n2, ...

Правила именования устройств описаны в документации к ядру:

<https://www.kernel.org/doc/html/latest/admin-guide/devices.html>

В современных Linux почти все дисковые накопители не зависимо от реального интерфейса подключения: SAS, SATA, USB и др., работают как SCSI устройства.

Для обозначения SCSI дисков используются файлы устройств вида /dev/sda, b, c, ... Где a — первый диск, b – второй и т. д.

Карты памяти SD/MMC, которые обозначаются как /dev/mmcblk0, 1, 2, ...

Синтетические дисковые устройства в виртуальных машинах kvm типа virtio. Такие устройства обозначаются как /dev/vda, b, c, ...

Для NVMe контроллеров используется файл устройства вида: /dev/nvme0, 1, ...
Накопители NVMe получают файлы вида /dev/nvme0n1, 2, ...

Для взаимодействия с дисками на диски и разделы на них создаются символичные ссылки в подкаталогах каталога /dev/disk/

```
$ ls /dev/disk/
by-id  by-partlabel  by-partuuid  by-path  by-uuid

# ls /dev/disk/by-id/ -l | head -3
total 0
lrwxrwxrwx 1 root root 9 Feb 7 16:42 ata-VBOX_CD-ROM_VB2-01700376 -> ../../sr0
lrwxrwxrwx 1 root root 9 Feb 7 16:42 ata-VBOX_HARDDISK_VB25e0ca90-16d06b2e
-> ../../sda
```

SMART

- В составе пакета smartmontools имеются утилита smartctl и демон smartd
- smartctl используется для управления и получения данных от SMART
- smartd предназначена для мониторинга и выполнения действий в случае возникновения проблем

При работе с дисками важно знать состояние дисков. В этом может помочь SMART.

Для работы с информацией SMART в Linux применяется специальный пакет smartmontools.

В составе пакета имеются утилита smartctl и демон smartd.

Утилита smartctl используется для управления и получения данных от SMART устройств.

Пример: просканируем диски, которые поддерживают SMART и получим о них сведения:

```
# smartctl --scan
/dev/sda -d scsi # /dev/sda, SCSI device
/dev/nvme0 -d nvme # /dev/nvme0, NVMe device

# smartctl --scan | awk '{print "smartctl -a \"$1}" | sh | egrep 'Model|
Capacity|Serial|SECTION'
=== START OF INFORMATION SECTION ===
Device Model:          VBOX HARDDISK
Serial Number:         VB25e0ca90-16d06b2e
User Capacity:         42,949,672,960 bytes [42.9 GB]
=== START OF INFORMATION SECTION ===
Model Number:          ORCL-VBOX-NVME-VER12
Serial Number:         VB1234-56789
Namespace 1 Size/Capacity:      8,589,934,592 [8.58 GB]
=== START OF SMART DATA SECTION ===
```

То же самое но на реальном компьютере:

```
# smartctl --scan
/dev/sda -d scsi # /dev/sda, SCSI device
/dev/sdb -d scsi # /dev/sdb, SCSI device

# smartctl --scan | awk '{print "smartctl -a \"$1}" | sh | egrep 'Model|
```

Глава 4. Управление устройствами.

```
Capacity|Serial|SECTION|overall-health'
=== START OF INFORMATION SECTION ===
Device Model:      WDC WD10SPCX-60KHST0
Serial Number:     WD-WX81A943YYZP
User Capacity:     1 000 204 886 016 bytes [1,00 TB]
=== START OF READ SMART DATA SECTION ===
SMART overall-health self-assessment test result: PASSED
=== START OF INFORMATION SECTION ===
Model Family:      SandForce Driven SSDs
Device Model:      KINGSTON SKC300S37A180G
Serial Number:     50026B723404FDD5
User Capacity:     180 045 766 656 bytes [180 GB]
=== START OF READ SMART DATA SECTION ===
SMART overall-health self-assessment test result: PASSED
```

Служба `smartd` предназначена для мониторинга и выполнения действий в случае возникновения проблем с дисками.

Конфигурационный файл службы `smartd` - `/etc/smartmontools/smartd.conf`

4.4.2. Сетевые интерфейсы.



Сетевые интерфейсы

- Для сетевых интерфейсов не создаются файлы устройств
- PNIDN - Predictable Network Interface Device Names
 - 1) Встроенные сетевые устройства: enoX
 - 2) Устройства PCI Express с горячим подключением: ensX
 - 3) Имена по расположению оборудования: enpXsY
 - 4) Именованное по MAC адресу: enx112233445566
 - 5) Классический способ именования: ethX

Для сетевых интерфейсов не создаются файлы устройств.

В современных системах может использоваться система предсказуемых имен сетевых устройств (PNIDN - Predictable Network Interface Device Names).

В соответствии с PNIDN интерфейсы именуются так:

1. Встроенные сетевые устройства: enoX (X — номер устройства)
2. Устройства PCI Express с горячим подключением: ensX (X — номер слота)
3. Имена по расположению оборудования: enpXsY (X — номер слота PCI, Y — номер устройства)
4. Именованное по MAC адресу: enx112233445566
5. Классический способ именования: ethX

Если вы хотите использовать классическое именование сетевых интерфейсов, то вам необходимо задать опцию загрузки ядра: `net.ifnames=0`

Служба `udev` позволяет давать любые удобные вам названия для сетевых интерфейсов.

Пример: Назначение имени сетевой карте в соответствии с MAC адресом.

```
# ifconfig myownnetname
myownnetname: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet6 fe80::4dbf:94b7:62c6:1b2b prefixlen 64 scopeid 0x20<link>
ether 52:54:00:e3:15:ae txqueuelen 1000 (Ethernet)
RX packets 25 bytes 5288 (5.1 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 11 bytes 1650 (1.6 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Глава 4. Управление устройствами.

```
# cat /etc/udev/rules.d/70-persistent-net.rules SUBSYSTEM=="net", ACTION=="add",  
DRIVERS=="?*", ATTR{address}=="52:54:00:e3:15:ae", NAME="myownnetname"
```

Для активизации сетевого интерфейса предназначена команда `/sbin/ip`

Пример:

```
# ip link  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT  
group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP  
mode DEFAULT group default qlen 1000  
    link/ether 08:00:27:82:36:83 brd ff:ff:ff:ff:ff:ff  
  
# lspci | grep Ether  
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet  
Controller (rev 02)  
  
# ip address show  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group  
default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP  
group default qlen 1000  
    link/ether 08:00:27:82:36:83 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.101.180/24 brd 192.168.101.255 scope global dynamic  
noprofixroute enp0s3  
    valid_lft 75411sec preferred_lft 75411sec  
    inet6 fe80::b80e:e6d7:98ae:a40b/64 scope link noprefixroute  
    valid_lft forever preferred_lft forever
```

Команда `ip` позволяет так же и настроить IP адрес, но этот адрес будет не постоянным.

Для постоянной настройки адресов нужно либо настраивать соответствующие файлы (`/etc/network/interfaces`, `/etc/sysconfig/network-scripts/ifcfg-*`, `/etc/netplan/*.yaml`, ...), либо запускать специальные утилиты (`nmcli`, `nmtui`, `gnome-control-center`, ...).

4.4.3. Поддержка USB.



Поддержка USB

- Как правило поддержка контроллеров включена непосредственно в ядро, а не в виде модулей
- Работает через подсистему SCSI
- Интерфейс имеет несколько версий
- 1.1 драйверы OHCI и UHCI
- 2.0 драйвер EHCI
- 3.0 драйвер XHCI
- Wireless WHCI
- Virtual VHCI
- `grep HCI_HCD /boot/config-$(uname -r)`
- `lsusb`

Ядра Linux серий 2.4 и более поздних включают поддержку USB устройств без необходимости наложения на ядро каких-либо пакетов обновлений. Причем поддерживается возможность горячего подключения устройств (Hot Plug).

Примечание: Для включения поддержки в ядра 2.4 следует выбрать соответствующие опции конфигурации ядра в разделе *USB* и выбрать поддержку *Hot Plug* (более подробно конфигурирование ядра обсуждается в следующей главе).

USB устройства работают через эмуляцию SCSI.

Поддерживаются следующие стандарты:

1. 1.1 драйверы OHCI и UHCI
2. 2.0 драйвер EHCI
3. 3.0 драйвер XHCI
4. Wireless WHCI
5. Virtual VHCI

В некоторых дистрибутивах драйверы USB включены в ядро статически.

Пример: В Centos 8 Stream

```
# grep USB_ .HCI_HCD /boot/config-$(uname -r)
CONFIG_USB_XHCI_HCD=y
CONFIG_USB_EHCI_HCD=y
# CONFIG_USB_EHCI_HCD_PLATFORM is not set
CONFIG_USB_OHCI_HCD=y
CONFIG_USB_OHCI_HCD_PCI=y
# CONFIG_USB_OHCI_HCD_PLATFORM is not set
CONFIG_USB_UHCI_HCD=y
# CONFIG_USB_WHCI_HCD is not set
```

Глава 4. Управление устройствами.

В последней строке видно, что поддержка беспроводного USB в ядро не включено.

```
# grep USBIP_.HCI_HCD /boot/config-$(uname -r)
```

А поддержки Virtual USB даже в коде нет.

В Debian 10:

```
# grep USBIP_.HCI_HCD /boot/config-$(uname -r)
CONFIG_USBIP_VHCI_HCD=m
```

Считывание и изменение настроек USB устройств можно производить с помощью файловой системы `/sys` в подкаталоге `bus/usb`.

Проверить подключенные USB устройства можно командой `lsusb`.

4.4.4. Виртуальные устройства.



Виртуальные устройства

- /dev/null
- /dev/zero
- /dev/random и /dev/urandom
- /dev/loop*
- /dev/tty* и /dev/pts/*

В Linux (Unix) часто используются виртуальные устройства.

В файловой системе /sys для них имеется специальный подкаталог — /sys/devices/virtual

Наиболее часто используются следующие устройства:

1. /dev/null — Черная дыра. Устройство, которое уничтожает любые данные в него записанные.
2. /dev/zero — Генератор нулей.
3. /dev/random и /dev/urandom — Генераторы случайных и псевдослучайных чисел.

Примечание: При чтении данных из устройства /dev/random выводятся только случайные байты, полностью состоящие из битов шума «хаотичного» пула ОС. Если «хаотичный» пул опустел, /dev/random ничего не выдаст, пока необходимое количество битов в пуле не будет создано, читающая /dev/random программа будет ждать появления очередного случайного байта.

В ядре Linux «хаотичный» пул получает энтропию из нескольких источников, в том числе из аппаратного генератора случайных чисел современных процессоров Intel.

Устройство /dev/random может быть необходимо пользователям, которые требуют очень высокого коэффициента случайности, например, при создании ключа шифрования, предполагающего длительное использование.

Чтение данных устройства /dev/urandom возвратит столько байтов, сколько было запрошено. В результате, если в пуле было недостаточно битов, теоретически возможно найти уязвимость алгоритма, использующего это устройство (на настоящее время нет опубликованных работ о такой атаке). Если это важно, следует использовать /dev/random.

```
# dd if=/dev/random of=/dev/null count=1000 iflag=fullblock  
1000+0 records in
```

Глава 4. Управление устройствами.

```
1000+0 records out
512000 bytes (512 kB, 500 KiB) copied, 6.84952 s, 74.7 kB/s

# dd if=/dev/urandom of=/dev/null count=1000 iflag=fullblock
1000+0 records in
1000+0 records out
512000 bytes (512 kB, 500 KiB) copied, 0.00643111 s, 79.6 MB/s
```

4. /dev/loop* - Закольцованные устройства — это блочные устройства, которые отображают блоки данных обычного файла в файловой системе или другое блочное устройство. Начиная с Linux 3.1, ядро предоставляет устройство /dev/loop-control, которое позволяет приложению динамически находить свободное устройство, добавлять и удалять закольцованные устройства из системы. Управление loop устройствами осуществляется командой losetup.

Пример: создадим файл образ и подключим его.

```
# dd if=/dev/zero of=~/file.img bs=1024k count=10

# mkfs.ext2 file.img

# mount file.img /mnt/

# losetup -l
NAME          SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE          DIO LOG-SEC
/dev/loop0    0          0          1  0 /root/file.img    0        512
```

Канонический пример того же самого:

```
# dd if=/dev/zero of=~/fileloop.img bs=1024k count=10

# losetup --find --show ~/fileloop.img
/dev/loop1

# mkfs.ext2 /dev/loop1

# mount /dev/loop1 /mnt/

# umount /mnt

# losetup -l
NAME          SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE          DIO LOG-SEC
/dev/loop1    0          0          0  0 /root/fileloop.img 0        512
/dev/loop0    0          0          1  0 /root/file.img    0        512

# losetup --detach /dev/loop1

# losetup -l
NAME          SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE          DIO LOG-SEC
/dev/loop0    0          0          1  0 /root/file.img    0        512

# umount /mnt
# losetup -l
```

5. /dev/tty* и /dev/pts/* - виртуальные терминалы. Первые /dev/tty* создаются заранее и используются для работы непосредственно с терминалом компьютера (клавиатура, монитор и мышь). Устройства /dev/pts/* создаются по мере необходимости, например, когда подключаются в удаленную сессию (telnet или ssh).