Содержание

Глава 3: Устройства

3.1 Файлы устройств

linux использует тот же дизайн для файлов устройств, что и другие системы Unix: помещает спеицальные файлы устрйоств в /dev. Такие файлы называются **узлами устройств**. Хотя не все возможности устройств доступны через файловый ввод-вывод.

Первый символ в выводе `ls -l /dev` - символ типа файла. Если это символы b, c, p или s, то файл является устройством. Эти буквы обозначают **block** (блочное устройство), **character** (символьное устройство), **pipe** (конвейер) и **socket** (сокет) соответственно.

- Блочное устройство. Программы получают доступ к данным с блочного устройства фиксированными частями. Устройство sda1 в приведенном ранее примере - это дисковое устройство, тип блочного устройства. Диски можно легко разбить на блоки данных. Поскольку общий размер блочного устройства фиксирован и его легко индексировать, процессы имеют быстрый произвольный доступ к любому блоку в устройстве с помощью ядра.
- Символьное устройство. Символьные устройства работают с потоками данных. Вы можете считывать символы с символьных устройств или записывать их на символьные устройства, как было показано ранее на устройстве /dev/null. Символьные устройства не имеют размера: когда вы читаете из одного из них или записываете в него, ядро обычно выполняет на нем операцию чтения или записи. Принтеры, непосредственно подключенные к компьютеру, представляются символьными устройствами. Важно отметить, что во время взаимодействия с символьным устройством ядро не может создавать резервные копии и повторно выполнять проверку после передачи данных устройству или процессу.
- Конвейер. Именованный конвейер по структуре такой же, как символьные устройства, но с другим процессом на другом конце потока ввода-вывода вместо драйвера ядра.
- Сокет. Сокеты - это специальные интерфейсы, которые используются для межпроцессной связи. Они часто находятся за пределами каталога /dev. Файлы сокетов представляют собой доменные сокеты Unix (подробнее о них вы узнаете в главе 10).

Не у всех устройств есть файлы, поскольку интерфейсы ввода-вывода блочных и символьных устройств подходят не во всех случаях. Например, сетевые интерфейсы не содержат файлов устройств. Теоретически возможно взаимодействовать с сетевым интерфейсом с помощью одного символьного устройства, но поскольку это проблематично, ядро предлагает другие интерфейсы ввода-вывода.

3.2 Путь к устройству sysfs

Традиционный каталог Unix /dev удобен для того, чтобы пользовательские процессы ссылались на устройства, поддерживаемые ядром, и взаимодействовали с ними, но это очень упрощенная схема. Имя устройства в /dev немногое говорит об устройстве. Другая проблема заключается в

том, что ядро назначает устройства в том порядке, в котором они найдены, поэтому между перезагрузками устройство может получить другое имя.

Чтобы обеспечить единообразное представление подключенных устройств на основе их фактических аппаратных атрибутов, ядро Linux предлагает интерфейс sysfs для обозначения файлов и каталогов. Базовый путь для устройств - /sys/devices.

Например, жесткий диск SATA в `/dev/sda` может иметь следующий путь в интерфейсе sysfs: `/sys/devices/pci0000:00/0000:00:17.0/ata3/host0/target0:0:0/0:0:0:0/block/sda`

Файл /dev позволяет пользовательским процессам применять устройство, в то время как путь /sys/devices задействуется для просмотра информации и управления устройством.

В каталоге /sys есть несколько ярлыков. Например, /sys/block должен содержать все блочные устройства, доступные в системе. Однако это всего лишь символические ссылки: лучше запустить команду ls -l /sys/block, чтобы выявить истинные sysfs пути.

3.3 Утилита dd и устройства

Программа dd чрезвычайно полезна в работе с блочными и символьными устройствами. Ее единственная функция заключается в чтении из входного файла или потока и записи в выходной файл или поток с выполнением некоторых преобразований кодирования по мере необходимости.

Параметр	**Описание**
if=file	Файл ввода (по умолч. stdin)
of=file	Файл вывода (по умолч. stdout)
bs=size	Размер блока для чтения/записи. Суффиксы: b (512 байт), k (1024 байта)
ibs=size, obs=size	Размеры блоков ввода и вывода (если разные)
count=num	Количество блоков для копирования
skip=num	Пропустить первые num блоков входного потока

3.4 Имена устройств

Найти имя устройства можно несколькими способами:

- Использовать **udevadm** (см. раздел 3.5)
- Найти в каталоге **/sys**
- Проверить **journalctl -k** или журнал ядра
- Для дисков: **mount**
- Выполнить **cat /proc/devices** для просмотра драйверов

Первый метод наиболее надежен, но требует udev. В следующих разделах рассмотрены основные типы устройств Linux.

3.4.1 Жесткие диски: /dev/sd*

Большинство дисков в Linux именуются с префиксом **sd** (/dev/sda, /dev/sdb и т.д.). «sd» расшифровывается как **SCSI-диск** (Small Computer System Interface). Ядро создает отдельные файлы для разделов (/dev/sda1, /dev/sda2).

Список SCSI-устройств:

```
$ lsscsi
[0:0:0:0] disk ATA WDC WD3200AAJS-2 01.0 /dev/sda
[2:0:0:0] disk FLASH Drive UT_USB20 0.00 /dev/sdb
```

Система назначает устройства в порядке обнаружения драйверами. **Проблема**: при удалении диска остальные переименовываются (/dev/sdc становится /dev/sdb). Решение: **UUID** или **LVM** (Logical Volume Manager).

3.4.2 Виртуальные диски: /dev/xvd*, /dev/vd*

Дисковые устройства для виртуальных машин (AWS, VirtualBox, Xen).

3.4.3 Устройства долговременной памяти: /dev/nvme*

Интерфейс **NVMe** для SSD. Просмотр:

```
nvme list
```

3.4.4 Виртуальные блочные устройства: /dev/dm-*, /dev/mapper/*

Используется менеджер **LVM** со сопоставителем устройств (device mapper). Подробнее в главе 4.

3.4.5 CD- и DVD-приводы: /dev/sr*

Linux распознает оптические накопители как SCSI-устройства (/dev/sr0, /dev/sr1). Для **чтения** используйте /dev/sr*, для **записи** - /dev/sg0.

3.4.6 Жесткие диски PATA: /dev/hd*

PATA - устаревший интерфейс хранения. Устройства /dev/hda, /dev/hdb, /dev/hdc, /dev/hdd используются в старых ядрах Linux и оборудовании. Если диск SATA распознается как PATA-устройство, это означает, что диск работает в режиме совместимости с пониженной производительностью. Проверьте BIOS для переключения контроллера SATA в собственный режим.

3.4.7 Терминалы: /dev/tty*, /dev/pts/* и /dev/tty

Терминалы - устройства для передачи символов между процессом пользователя и устройством ввода-вывода.

Большинство современных терминалов - это **псевдотерминальные устройства**, эмулирующие реальные терминалы через интерфейс ввода-вывода ядра (окно оболочки).

Основные устройства:

- **/dev/tty1** - первая виртуальная консоль
- **/dev/pts/0** - первое псевдотерминальное устройство (/dev/pts - выделенная файловая система)
- **/dev/tty** - управляющий терминал текущего процесса

Режимы отображения и виртуальные консоли

Linux поддерживает два режима: **текстовый** и **графический**. Каждая виртуальная консоль может работать в любом режиме.

В текстовом режиме переключение между консолями:

- **Alt+F1** → /dev/tty1
- **Alt+F2** → /dev/tty2 и т. д.

В графическом режиме требуется **Ctrl+Alt+Fn** для переключения, или **Ctrl+Alt+F1** для перехода в текстовую консоль.

Если переключение не работает, используйте команду:

```
# chvt 1
```

3.4.8 Последовательные порты: /dev/ttyS*, /dev/ttyUSB*, /dev/ttyACM*

Устройство Windows	Устройство Linux	Назначение
COM1	/dev/ttyS0	Последовательный порт RS-232
COM2	/dev/ttyS1	Последовательный порт RS-232
USB-адаптер	/dev/ttyUSB0, /dev/ttyUSB1…	USB последовательный адаптер
USB-адаптер ACM	/dev/ttyACM0, /dev/ttyACM1…	USB ACM-устройство

Для подключения используйте команду screen:

```
screen /dev/ttyS0
```

Может потребоваться добавить пользователя в группу dialout для доступа к портам. Полезно для платформ на микроконтроллерах (Arduino, CircuitPython и т. д.).

3.4.9 Параллельные порты: /dev/lp0 и /dev/lp1

Устаревший интерфейс, заменен USB и сетями. Однонаправленные параллельные портовые устройства:

Устройство Linux	Аналог Windows
/dev/lp0	LPT1
/dev/lp1	LPT2

Можно отправлять файлы непосредственно на параллельный порт с помощью команды **cat**, но может потребоваться подать страницу или перезагрузить принтер. Сервер печати, такой как **CUPS**, эффективнее справляется при взаимодействии с принтером.

Двунаправленные параллельные порты: **/dev/parport0** и **/dev/parport1**.

3.4.10 Аудиоустройства: /dev/snd/*, /dev/dsp, /dev/audio и другие

Linux имеет два набора аудиоустройств:

Система	Описание
ALSA (Advanced Linux Sound Architecture)	Устройства в /dev/snd, сложны для прямого использования
OSS (Open Sound System)	Более старая открытая звуковая система

Системы Linux с ALSA поддерживают обратную совместимость OSS (при загрузке поддержки в ядре).

С драйвером **dsp** OSS возможны элементарные операции. Например, воспроизведение WAV-файла:

```
cat file.wav > /dev/dsp
```

Однако могут возникнуть проблемы с несоответствием частоты, и устройство часто уже занято.

> **Примечание:** Звуки в Linux — это запутанная история множества слоев. Помимо устройств уровня ядра, в пользовательском пространстве работают серверы (pulse-audio), управляющие звуком из разных источников и выступающие посредниками между звуковыми устройствами и процессами пользовательского пространства.

3.4.11 Создание файла устройства

В новых системах Linux файлы устройств создаются автоматически утилитами **devtmpfs** и **udev** (см. раздел 3.5). Однако полезно знать, как создать устройство вручную (редко требуется для именованных конвейеров или файлов сокетов).

Команда **mknod** создает одно устройство. Необходимо знать его имя, основной и второстепенный номера:

```
# mknod /dev/sda1 b 8 1
```

Параметр `b 8 1` указывает на **блочное устройство** с основным номером 8 и второстепенным номером 1. Для символьных устройств используйте параметр `c` вместо `b` (для именованных конвейеров опустите номера).

История управления устройствами

В старых версиях Unix и Linux каталог /dev требовал ручного обслуживания. С каждым обновлением ядра возникали новые номера устройств, решаемые программой **MAKEDEV**. Эта статическая система оказалась неэффективной:

- **devfs** — реализация /dev в пространстве ядра (имела ограничения)
- **udev и devtmpfs** — современная динамическая система управления устройствами

3.5 Менеджер устройств udev

Ядро Linux может отправлять уведомления процессу пользовательского пространства, называемому **udevd**, при обнаружении нового устройства в системе (например, когда кто-то подключает USB-накопитель). Процесс udevd может изучить характеристики нового устройства, создать его файл, а затем выполнить любую инициализацию устройства.

Файлы устройств необходимы еще на этапе загрузки, так что udevd не должен зависеть от каких-либо устройств. Udevd должна инициализировать устройства очень быстро, что бы остальная система не ждала её во время загрузки.

3.5.1 Файловая система devtmpfs

devtmpfs разработана для решения проблемы доступности устройства во время загрузки. Ядро создает файлы устройств по мере необходимости и уведомляет **udevd** о наличии нового устройства.

Получив сигнал, udevd:

- Не создает файлы устройств
- Выполняет инициализацию устройства
- Настраивает права доступа
- Уведомляет другие процессы о новых устройствах
- Создает символические ссылки в /dev для идентификации устройств

Символические ссылки в /dev/disk/by-id

Пример ссылок для типичного диска (подключен в /dev/sda):

```
$ ls -l /dev/disk/by-id
lrwxrwxrwx 1 root root 9 Jul 26 10:23 scsi-SATA_WDC_WD3200AAJS-_WD-
```

```
WMAV2FU80671 -> ../../sda
lrwxrwxrwx 1 root root 10 Jul 26 10:23 scsi-SATA_WDC_WD3200AAJS-_WD-
WMAV2FU80671-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Jul 26 10:23 scsi-SATA_WDC_WD3200AAJS-_WD-
WMAV2FU80671-part2 -> ../../sda2
lrwxrwxrwx 1 root root 10 Jul 26 10:23 scsi-SATA_WDC_WD3200AAJS-_WD-
WMAV2FU80671-part5 -> ../../sda5
```

Процесс udevd назывет ссылки по типу интерфейса, информации о производителе, модели, серийному номеру и разделу.

> **Примечание:** Часть **tmp** в devtmpfs указывает на то, что файловая система находится в оперативной памяти с возможностью чтения/записи процессами пользовательского пространства. Это позволяет udevd создавать символические ссылки.

3.5.2 Работа и настройка менеджера udevd

Процесс работы udevd

1. Ядро отправляет `udevd` уведомление о событии `(uevent)` по внутренней сетевой ссылке
2. Демон `udevd` загружает все атрибуты события `uevent`
3. Демон `udevd` анализирует правила, фильтрует и обновляет событие, выполняет действия и устанавливает атрибуты

Пример события uevent

```
ACTION=change
DEVNAME=sde
DEVPATH=/devices/pci0000:00/0000:00:1a.0/usb1/1-1/1-1.2/1-1.2:1.0/host4/targ
et4:0:0/4:0:0:3/block/sde
DEVTYPE=disk
DISK_MEDIA_CHANGE=1
MAJOR=8
MINOR=64
SEQNUM=2752
SUBSYSTEM=block
UDEV_LOG=3
```

После получения события udevd узнает имя устройства, путь sysfs и атрибуты, затем готов к обработке правил.

Расположение файлов правил

Каталог	Назначение
/lib/udev/rules.d	Правила по умолчанию
/etc/udev/rules.d	Переопределения пользователя

Алгоритм чтения правил udevd

1. `udevd` считывает правила от начала до конца файла
2. После чтения и выполнения правила `udevd` продолжает чтение текущего файла для получения следующих применимых правил
3. Директивы (например, `GOTO`) позволяют пропускать части правил (обычно в верхней части файла)

Пример правил для ATA

```
# ATA
KERNEL=="sd*[!0-9]|sr*", ENV{ID_SERIAL}!="?*", SUBSYSTEMS=="scsi",
ATTRS{vendor}=="ATA", IMPORT{program}="ata_id --export $devnode"

# ATAPI devices (SPC-3 or later)
KERNEL=="sd*[!0-9]|sr*", ENV{ID_SERIAL}!="?*", SUBSYSTEMS=="scsi",
ATTRS{type}=="5",ATTRS{scsi_level}=="[6-9]*", IMPORT{program}="ata_id --
export $devnode"
```

Эти правила соответствуют дискам ATA и оптическим носителям через подсистему SCSI. Правила пытаются подобрать устройство, начинающееся с **sd** или **sr** без номера, с подсистемой SCSI и другими атрибутами. При истинности всех условий выполняется директива:

```
IMPORT{program}="ata_id --export $devnode"
```

Команда ata_id

```
# /lib/udev/ata_id --export /dev/sda
ID_ATA=1
ID_TYPE=disk
ID_BUS=ata
ID_MODEL=WDC_WD3200AAJS-22L7A0
ID_MODEL_ENC=WDC\x20WD3200AAJS22L7A0\x20...\x20\x20
ID_REVISION=01.03E10
ID_SERIAL=WDC_WD3200AAJS-22L7A0_WD-WMAV2FU80671
```

Импорт устанавливает переменные окружения из вывода команды. Любое следующее правило теперь распознает **ENV{ID_TYPE}** как диск.

Условные выражения в правилах

Важное условие для переменной **ID_SERIAL**:

```
ENV{ID_SERIAL}!="?*"
```

Это выражение истинно, если **ID_SERIAL** не задана. Если значение уже установлено, условие ложно и udevd пропускает текущее правило.

Цель: запустить процесс ata_id для поиска серийного номера, затем добавить атрибуты в рабочую копию uevent.

Создание символических ссылок

Когда **ENV{ID_SERIAL}** установлена, используется правило для создания ссылок:

```
KERNEL=="sd*|sr*|cciss*", ENV{DEVTYPE}=="disk", ENV{ID_SERIAL}=="?*",
SYMLINK+="disk/by-id/$env{ID_BUS}-$env{ID_SERIAL}"
```

Директива **SYMLINK+=** добавляет символическую ссылку для обнаруженного устройства.

Отличие условных выражений от директив

Тип	Синтаксис	Описание
Условные выражения	== или !=	Проверка условий
Директивы	= или += или :=	Выполнение действий

3.5.3 Команда udevadm

udevadm - инструмент администрирования udevd для поиска устройств, изучения их атрибутов и мониторинга uevent-событий из ядра.

Просмотр всех атрибутов udev для устройства:

```
$ udevadm info --query=all --name=/dev/sda
```

Вывод содержит атрибуты с префиксами:

Префикс	Значение
P:	путь в sysfs
N:	имя узла устройства (/dev)
S:	символические ссылки
E:	дополнительная информация

3.5.4 Отслеживание устройств

Мониторинг uevent-событий:

```
$ udevadm monitor
```

Вывод показывает события ядра (KERNEL) и обработанные udevd события (UDEV).

Основные параметры:

Параметр	Описание
–kernel	только события ядра
–udev	только события udevd
–property	показать все атрибуты события
–subsystem-match=SUBSYS	фильтр по подсистеме

Пример фильтрации по SCSI:

```
$ udevadm monitor --kernel --subsystem-match=scsi
```

Примечание: Демон **udisksd** автоматически присоединяет диски и уведомляет процессы о новых устройствах.

3.6 Подробнее об интерфейсе SCSI и ядре Linux

Архитектура SCSI

Традиционная SCSI конфигурация состоит из хост-адаптера, связанного с цепочкой устройств на шине SCSI. Каждое устройство имеет **SCSI ID** (8 или 16 идентификаторов в зависимости от версии).

Современные варианты:

- Serial Attached SCSI (SAS) — высокая производительность
- USB-накопители с SCSI командами
- ATAPI (CD/DVD-приводы) — версия SCSI команд
- SATA диски — транслируются через libata

Пример: просмотр устройств SCSI

```
$ lsscsi
[0:0:0:0] disk ATA WDC WD3200AAJS-2 01.0 /dev/sda
[1:0:0:0] cd/dvd Slimtype DVD A DS8A5SH XA15 /dev/sr0
[2:0:0:0] disk USB2.0 CardReader CF 0100 /dev/sdb
[3:0:0:0] disk FLASH Drive UT_USB20 0.00 /dev/sdf
```

Формат [H:B:T:L]:

Позиция	Значение
H	номер хост-адаптера SCSI
B	номер шины SCSI
T	SCSI-ID устройства (целевой номер)
L	LUN (номер логического блока)

Трёхуровневая архитектура подсистемы SCSI

Верхний уровень — обработка операций класса устройств (драйвер **sd** для дисков, преобразует запросы ядра в SCSI команды).

Средний уровень — модерирование и маршрутизация сообщений SCSI между верхним и нижним уровнями, отслеживание шин и устройств.

Нижний уровень — обработка действий, зависящих от оборудования (отправка/приём SCSI сообщений адаптерам хоста).

Ключевое правило: для каждого файла устройства ядро использует один драйвер верхнего и один драйвер нижнего уровня. Пример: для /dev/sda используются драйвер **sd** (верхний) и драйвер **ATA bridge** (нижний).

3.6.1 USB-накопитель и SCSI

USB-накопитель, хотя и понимает команды SCSI, требует для взаимодействия с ядром трёхуровневую подсистему USB (подобную подсистеме SCSI): класс устройств, управление шиной, хост-контроллеры.

Драйвер USB-накопителя выступает переводчиком между SCSI и USB протоколами, переупаковывая команды SCSI в USB-сообщения.

Для связи подсистем используется **простой SCSI-мост** (нижний уровень SCSI подключается к драйверу хранения USB-подсистемы).

3.6.2 Интерфейсы SCSI и ATA

Диски SATA и ATAPI-приводы подключаются к подсистеме SCSI через драйвер-мост **libata**.

Устройство	Протокол	Сложность
ATAPI привод (CD/DVD)	SCSI команды в ATA	простая упаковка/распаковка
Диск SATA	нет SCSI команд	полный перевод команд

Аналогия: ATAPI — как переписывание английского текста (синтаксическая задача), SATA диск — как перевод с немецкого (требует понимания смысла).

3.6.3 Обобщенные устройства SCSI

Обобщённые устройства SCSI (/dev/sg*) позволяют пользовательским процессам напрямую отправлять команды SCSI, минуя драйверы классов устройств.

Просмотр обобщённых устройств:

```
$ lsscsi -g
[0:0:0:0] disk ATA WDC WD3200AAJS-2 01.0 /dev/sda /dev/sg0
[1:0:0:0] cd/dvd Slimtype DVD A DS8A5SH XA15 /dev/sr0 /dev/sg1
```

Причина использования: сложные операции (например, запись на CD/DVD) выполняются в пользовательском пространстве через sg-устройства, а не в ядре. Это упрощает разработку и поддержку.

3.6.4 Множественный доступ к одному устройству

Одно физическое устройство может быть доступно через несколько точек входа.

Пример: оптический привод:

- **sr** (/dev/sr0) — чтение через блочный драйвер
- **sg** (/dev/sg1) — запись через обобщённое SCSI устройство

Такие процессы обычно **не запускаются одновременно** для одного устройства.

Примечание: жёсткие диски имеют ещё больше слоёв доступа поверх блочных устройств (описано далее).