Содержание

Глава 4: Диски и файловые системы

Разделы являются более мелкими частями всего диска. В Linux они обозначаются цифрой после блочного устройства, поэтому у них есть такие имена, как /dev/sda1 и /dev/sdb3. Ядро представляет каждый раздел как блочное устройство, как представляло бы весь диск. Разделы определяются на небольшой области диска, называемой **таблицей разделов** (также **метка диска**).

Следующий уровень разбиения на разделы - это **файловая система**, база данных файлов и каталогов, с которыми вы привыкли взаимодействовать в пользовательском пространстве. Файловая система находиться в разделе. Таким образом есть следующая иерархия: диск → раздел → ФС

Для доступа к данным на диске ядро Linux и применяет систему слоев, показанную на рис.4.2 Подсистема SCSI и все остальное, описанное в разделе 3.6, представлены одним блоком. Обратите внимание: вы можете работать с диском как через 104 Глава 4. Диски и файловые системы файловую систему, так и непосредственно через дисковые устройства.

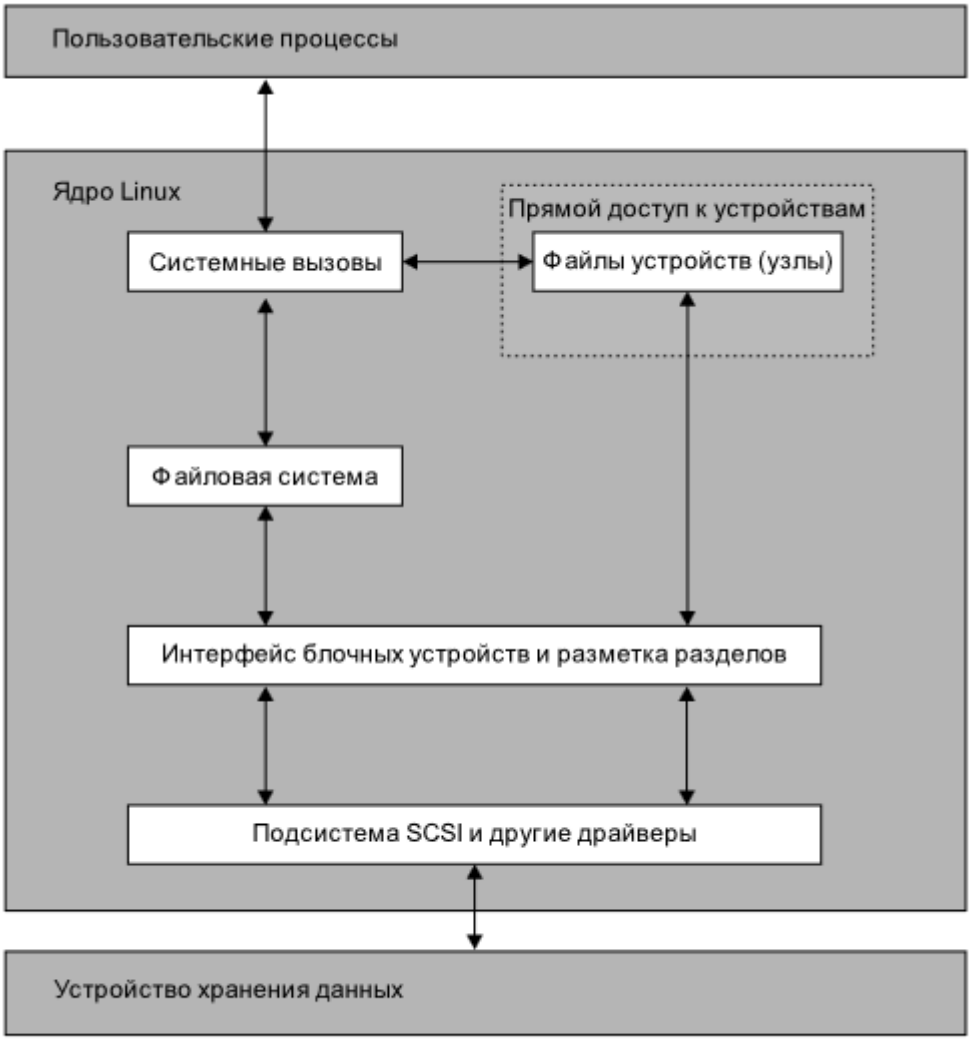


Рис. 4.2. Схема ядра с доступом к диску

4.1 Разбиение дисков на разделы

Таблица разделов - это набор данных о разделении блоков на диске.

Основные типы таблиц:

Тип	Описание
MBR	традиционная, с ограничением на 4 основных раздела
GPT	современная, с уникальными идентификаторами (GUID)

Инструменты:

- **parted** - текстовый, поддерживает MBR и GPT (используется для просмотра)
- **fdisk** - традиционный, интерактивный (используется для создания/изменения)

4.1.1 Просмотр таблицы разделов

```
# parted -l
```

Пример вывода:

```
# parted -l
Model: ATA KINGSTON SM2280S (scsi)
Disk /dev/sda: 240GB 
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
Number Start End Size Type File system Flags
1 1049kB 223GB 223GB primary ext4 boot
2 223GB 240GB 17.0GB extended
5 223GB 240GB 17.0GB logical linux-swap(v1)
106 Глава 4. Диски и файловые системы
Model: Generic Flash Disk (scsi)
Disk /dev/sdf: 4284MB 
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
Number Start End Size File system Name Flags
1 1049kB 1050MB 1049MB myfirst
2 1050MB 4284MB 3235MB mysecond
```

Различия MBR и GPT:

- MBR: нет столбца Name, традиционная нумерация разделов
- GPT: поддержка произвольных имён разделов (myfirst, mysecond)

Основы MBR

Структура разделов:

- **Основной** - обычный раздел (макс. 4 штуки в MBR)
- **Расширенный** - контейнер для логических разделов
- **Логический** - раздел внутри расширённого

Пример: раздел 2 (расширённый) содержит раздел 5 (логический).

Примечание: тип ФС в parted может отличаться от MBR-идентификатора (83 - Linux, 82 - swap). Для идентификаторов используйте:

```
$ fdisk -l
```

Разделы LVM

Если видны разделы с флагом **lvm** (код 8e) или устройства `/dev/dm-*` - система использует менеджер LVM. Вместо прямого разделения диска используется слой абстракции (подробнее в разделе 4.4).

Пример вывода в системе с LVM:

```
# parted -l
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 10.7GB
Partition Table: msdos
Number Start End Size Type File system Flags
1 1049kB 10.7GB 10.7GB primary boot, lvm

Model: Linux device-mapper (linear) (dm)
Disk /dev/mapper/ubuntu--vg-root: 9672MB
Partition Table: loop
Number Start End Size File system Flags
1 0.00B 9672MB 9672MB ext4
```

Логические разделы LVM отделены от таблицы разделов и отображаются как отдельные «диски» (/dev/mapper/*).

Первичное чтение ядром

При загрузке ядро читает таблицу MBR и выводит отладку (просмотр: `journalctl -k`):

```
sda: sda1 sda2 < sda5 >
```

Обозначение `sda2 < sda5 >` указывает, что /dev/sda2 - расширённый раздел, содержащий логический /dev/sda5. Сам расширённый раздел игнорируется (важны логические разделы).

4.1.2 Редактирование таблиц разделов

Основные риски и предосторожности

Изменение таблицы разделов затрудняет восстановление данных. **Требует резервной копии** важных данных. Разделы не должны быть примонтированы во время изменений.

Выбор инструмента

Утилита	Особенности
fdisk	Создает таблицу в памяти, применяет изменения только при выходе. Один системный вызов ядру.
parted	Применяет изменения сразу при выполнении команд. Сигнализирует ядру для каждого раздела отдельно.
gparted	Графический интерфейс для parted.

Отслеживание изменений (parted)

- **udevadm monitor –kernel** - показывает удаление/добавление устройств
- **/proc/partitions** - полная информация о разделах
- **/sys/block/device/** или **/dev** - измененные интерфейсы разделов

Принудительная перезагрузка таблицы разделов

Используйте команду для повторного чтения ядром таблицы разделов:

```
# blockdev --rereadpt /dev/sdf
```

4.1.3 Создание таблицы разделов

Условия примера

- Диск: 4 ГБ (USB-флеш)
- Таблица: MBR
- Разделы: ext4 (200 МБ + 3.8 ГБ)
- Устройство: /dev/sdd

Процесс с fdisk

```
# fdisk /dev/sdd
```

Просмотр таблицы:

```
Command (m for help): p
```

Удаление существующего раздела:

```
Command (m for help): d
Selected partition 1
```

Создание первого раздела (200 МБ):

```
Command (m for help): n
Partition type: p (primary)
Partition number: 1
First sector: 2048 (default)
Last sector: +200M
```

Создание второго раздела (остаток): Используйте все значения по умолчанию (команда `n`, затем Enter для всех параметров).

Просмотр результата:

```
Command (m for help): p
Device Boot Start End Sectors Size Id Type
/dev/sdd1 2048 411647 409600 200M 83 Linux
/dev/sdd2 411648 8368127 7956480 3.8G 83 Linux
```

Сохранение таблицы:

```
Command (m for help): w
```

Диагностика

Просмотрите логи ядра (только для fdisk):

```
journalctl -k
```

4.1.4. Геометрия дисков и разделов

Физическая структура жесткого диска

Компонент	Описание
Пластина	Вращающийся магнитный диск на шпинделе
Головка	Считывает данные с пластины, прикреплена к коромыслу
Коромысло	Движущийся рычаг для позиционирования головки
Цилиндр	Фиксированный круг данных при одном положении головки
Сектор	Доля цилиндра
Дорожка	Часть цилиндра для одной головки

Адресация типа **CHS** (cylinder-head-sector) - старая схема.

CHS vs LBA

CHS - **Устаревшая**, значения на современных дисках неправдивы

LBA - **Логическая блочная адресация**, номер блока (современный стандарт)

Таблица разделов **MBR содержит оба формата**, но LBA игнорирует границы цилиндров.

Границы цилиндров больше не важны

Старые программы разбиения предупреждали о невыравнивании цилиндров. **Игнорируйте это предупреждение** - модернизированная логика LBA и новые утилиты гарантируют эффективное расположение разделов независимо от CHS границ.

4.1.5 Чтение твердотельных дисков (SSD)

Выравнивание разделов - критический фактор

SSD читают данные **страницами** (4096–8192 байт), начиная с границы, кратной размеру страницы. Невыравненный раздел требует **двух операций чтения вместо одной**.

Современные утилиты выравнивают разделы по границе **1 МБ** (2048 блоков × 512 байт), что совместимо со всеми размерами страниц.

Проверка выравнивания раздела

```
$ cat /sys/block/sdf/sdf2/start
1953126
```

Результат - смещение в единицах (512 байт). Для проверки выравнивания раздела на 4096-байтовые страницы (8 секторов):

- **Разделите порядковый номер раздела на 8**
- Если делится **нацело** - раздел выравнен оптимально
- Если **нацело не делится** - производительность снижена

Пример: 1953126 ÷ 8 = 244140.75 → не выравнен

4.2 Файловые системы

ФС - связующее звено между ядром и userspace'ом, позволяющее преобразования простого

блочного устройства в сложную иерархию файлов и подкаталогов, понятную пользователям.

Раньше ФС распологались только на дисках, однако древовидная структура каталогов и интерфейс ввода-вывода файловых систем довольно универсальны, поэтому файловые системы теперь выполняют множество задач, к примеру, роль системных интерфейсов, которые отображаются в каталогах в /sys и /proc.

Уровень абстракции виртуальной файловой системы (Virtual File System, VFS) завершает реализацию файловой системы. Подобно тому как подсистема SCSI стандартизирует связь между различными типами устройств и командами управления ядром, VFS гарантирует, что все реализации файловой системы поддерживают стандартный интерфейс, чтобы приложения пользовательского пространства могли одинаково обращаться к файлам и каталогам. Поддержка VFS позволила Linux поддерживать чрезвычайно большое количество файловых систем.

4.2.1 Типы файловых систем

- ext2 - традиционная FS Linux, вдохновленная Unix (UFS, FFS)
- ext3 - добавила журналирование для целостности данных и ускорения загрузки
- ext4 - текущая версия; поддерживает большие файлы и большое количество подкаталогов
- Btrfs - новейшая FS Linux, расширяет возможности ext4
- XFS - высокопроизводительная FS; стандарт Red Hat Enterprise Linux 7.0+
- FAT/msdos - примитивная MS-DOS FS
- VFAT - Microsoft FS для съемных носителей (до 4 ГБайт); стандарт для SD-карт и USB-накопителей
- exFAT - Microsoft FS для больших файлов (4 ГБайт и более)
- NTFS - продвинутая Windows FS
- HFS+ - стандарт Apple для Macintosh
- ISO9660 - стандарт CD-ROM

Обратная совместимость ext

ext2 и ext3 монтируются как ext4, но ext4 не монтируется как ext2 или ext3.

4.2.2 Создание файловой системы

Создание файловой системы - операция в пользовательском пространстве, прямой доступ к блочному устройству.

Утилита mkfs

mkfs - интерфейс для программ mkfs.fs (где fs - тип ФС):

- Команда `mkfs -t ext4` запускает `mkfs.ext4`
- **mkfs.ext4, mkfs.ext3, mkfs.ext2** - символические ссылки на **mke2fs**
- **mkfs.vfat, mkfs.msdos** - ссылки на **mkdosfs**

- **mkfs.ntfs** - ссылка на **mkntfs**

Смотрите man-страницы конкретных утилит (например, **mke2fs(8)** для ext4).

подробнее

```
$ ls -l /sbin/mkfs.*
-rwxr-xr-x 1 root root 17896 Mar 29 21:49 /sbin/mkfs.bfs
-rwxr-xr-x 1 root root 30280 Mar 29 21:49 /sbin/mkfs.cramfs
lrwxrwxrwx 1 root root 6 Mar 30 13:25 /sbin/mkfs.ext2 -> mke2fs
lrwxrwxrwx 1 root root 6 Mar 30 13:25 /sbin/mkfs.ext3 -> mke2fs
lrwxrwxrwx 1 root root 6 Mar 30 13:25 /sbin/mkfs.ext4 -> mke2fs
lrwxrwxrwx 1 root root 6 Mar 30 13:25 /sbin/mkfs.ext4dev -> mke2fs
-rwxr-xr-x 1 root root 26200 Mar 29 21:49 /sbin/mkfs.minix
lrwxrwxrwx 1 root root 7 Dec 19 2011 /sbin/mkfs.msdos -> mkdosfs
lrwxrwxrwx 1 root root 6 Mar 5 2012 /sbin/mkfs.ntfs -> mkntfs
lrwxrwxrwx 1 root root 7 Dec 19 2011 /sbin/mkfs.vfat -> mkdosfs
```

Создание ФС

```
# mkfs -t ext4 /dev/sdf2
```

mkfs автоматически определяет количество блоков и устанавливает значения по умолчанию. **Не меняйте их без необходимости.**

Суперблок - критический компонент ФС. mkfs создает несколько резервных копий. **Запишите номера резервных копий** - они понадобятся для восстановления при сбое диска.

ВНИМАНИЕ: Создание ФС поверх существующей уничтожает все старые данные. Выполняйте только один раз на новый раздел.

4.2.3 Монтирование файловой системы

Присоединение ФС к работающей системе. Ядро монтирует корневой каталог (/) при загрузке.

Требуемые параметры монтирования

- **Устройство** - раздел диска (/dev/sda1) или идентификатор ФС. Специальные ФС (proc, sysfs) не требуют
- **Тип ФС** - ext4, vfat, ntfs и т.д.
- **Точка монтирования** - обычный каталог в иерархии (/music, /home/extra и т.д.). Может быть в любом месте системы

Просмотр смонтированных ФС

```
$ mount
```

```
/dev/sda1 on / type ext4 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
```

Элемент	Пример	Описание
Устройство	/dev/sda1	Может быть нереальным (proc, sysfs)
on	on	Разделитель
Точка монтирования	/	Путь в системе
type	type	Разделитель
Тип ФС	ext4	Идентификатор
Параметры	(rw,errors=remount-ro)	Опции монтирования (см. 4.2.6)

Команды монтирования

Монтировать:

```
# mount -t ext4 /dev/sdf2 /home/extra
```

Параметр `-t` обычно опционален, но требуется для различения похожих типов (например, FAT-варианты).

Демонтировать:

```
# umount /home/extra
```

Можно демонтировать по устройству или точке монтирования.

4.2.4 Идентификатор UUID файловой системы

Имена устройств могут изменяться в зависимости от порядка их обнаружения ядром. Решение - использование UUID (Universally Unique Identifier) для идентификации и монтирования файловых систем. UUID генерируют программы создания файловых систем, такие как mke2fs.

Для просмотра список устройств, их файловых систем и UUID используйте **blkid**:

```
# blkid
/dev/sdf2: UUID="b600fe63-d2e9-461c-a5cd-d3b373a5e1d2" TYPE="ext4"
/dev/sda1: UUID="17f12d53-c3d7-4ab3-943e-a0a72366c9fa" TYPE="ext4"
PARTUUID="c9a5ebb0-01"
/dev/sda5: UUID="b600fe63-d2e9-461c-a5cd-d3b373a5e1d2" TYPE="swap"
PARTUUID="c9a5ebb0-05"
/dev/sde1: UUID="4859-EFEA" TYPE="vfat"
```

Раздел FAT имеет серийный номер тома вместо UUID.

Монтирование по UUID:

```
# mount UUID=b600fe63-d2e9-461c-a5cd-d3b373a5e1d2 /home/extra
```

UUID предпочтителен для автоматического монтирования в /etc/fstab при загрузке и используется дистрибутивами для съемных носителей. Ubuntu монтирует FAT-раздел в /media/user/4859-EFEA.

Изменить UUID можно через **tune2fs(8)** (для ext2/ext3/ext4).

4.2.5 Буферизация диска, кэширование и файловые системы

Linux буферизует запись на диск: ядро сохраняет изменения в RAM и записывает их на диск в подходящий момент. Это невидимо пользователю и повышает производительность.

Команда **umount** автоматически синхронизирует буфер с диском. Команда **sync** принудительно синхронизирует все диски. Выполните sync перед выключением, если не можете размонтировать файловую систему.

Ядро кэширует прочитанные блоки в RAM, предотвращая повторные обращения к диску при повторном доступе к файлам.

Параметры монтирования файловой системы

Параметры монтирования изменяют поведение команды mount. Делятся на две категории: общие (работают со всеми ФС) и специфичные для конкретной ФС.

Основные короткие параметры

Параметр	Описание
-r	Монтировать в режиме только для чтения
-n	Не обновлять /etc/mtab (критично при загрузке)
-t	Указать тип файловой системы

Длинные параметры

Активируются через **-o** с ключевыми словами, разделенными запятыми:

```
# mount -t vfat /dev/sde1 /dos -o ro,uid=1000
```

Параметр	Описание
exec/noexec	Разрешить/запретить выполнение программ
suid/nosuid	Разрешить/запретить setuid
ro	Режим только для чтения
rw	Режим чтения-записи

Примечание: Unix использует \n для конца строки, DOS - \r\n. Текстовые редакторы (vim) автоматически определяют стиль.

4.2.7 Повторное монтирование файловой системы

Чтобы изменить параметры монтирования смонтированной ФС, используйте опцию **remount**. Пример - переключение корневого раздела в режим чтения-записи:

```
# mount -n -o remount /
```

Параметр **-n** необходим, чтобы не обновлять /etc/mtab, когда / доступен только для чтения. Команда предполагает наличие устройства в /etc/fstab.

4.2.8 Таблица файловой системы /etc/fstab

Постоянный список ФС для автоматического монтирования при загрузке хранится в **/etc/fstab**:

```
UUID=70ccd6e7-6ae6-44f6-812c-51aab8036d29 / ext4 errors=remount-ro 0 1
UUID=592dcfd1-58da-4769-9ea8-5f412a896980 none swap sw 0 0
/dev/sr0 /cdrom iso9660 ro,user,nosuid,noauto 0 0
```

Поле	Описание
Устройство/UUID	Идентификатор ФС (UUID предпочтителен)
Точка монтирования	Куда присоединить ФС
Тип ФС	ext4, swap, iso9660 и т.д.
Параметры	Опции через запятую
Dump	Всегда 0 (dump устаревшая)
fsck порядок	1 для /, 2 для других локальных ФС, 0 для остальных

Специальные параметры

Параметр	Назначение
defaults	Стандартные опции (rw, dev, exec, suid)
errors	Поведение при ошибках ext2/3/4: continue, remount-ro, panic
noauto	Исключить из mount -a (для съемных носителей)
user	Позволить обычным пользователям монтировать (автоматически добавляет nosuid, noexec, nodev)

Монтирование через ярлык: **mount /cdrom** (если ФС в fstab).

Монтирование всех ФС: **mount -a** (кроме помеченных noauto).

4.2.9 Альтернативы файлу /etc/fstab

Вместо единого файла /etc/fstab возможны:

- **/etc/fstab.d** - каталог с отдельными конфиг-файлами ФС
- **systemd units** - конфигурация через systemd (часто генерируется из /etc/fstab)

4.2.10 Емкость файловой системы

Используйте **df** для просмотра размера и использования смонтированных ФС:

```
$ df
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/sda1 214234312 127989560 75339204 63% /
/dev/sdd2 3043836 4632 2864872 1% /media/user/uuid
```

Поле	Описание
Filesystem	Устройство
1K-blocks	Общая емкость в 1024-байтных блоках
Used	Занято
Available	Свободно
Use%	Процент использования
Mounted on	Точка монтирования

Команда **df dir** ограничивает вывод ФС конкретного каталога.

Расхождение в расчетах

Разница между используемым и свободным пространством объясняется **зарезервированными блоками** (~5% на ext*). Только суперпользователь может использовать их - это защита от паники при переполнении диска.

Опции df

- **-h** - читаемый формат (K, M, G, T)
- **-m** - в мегабайтах
- **-k** - в килобайтах (по умолчанию)

Поиск крупных файлов

Команда **du** показывает использование диска по каталогам:

```
du -s *              # Итог для каждого элемента текущего каталога
du -s                # Итог для текущего каталога
cd /; du             # Весь диск (длинный список)
```

4.2.11 Проверка и восстановление файловых систем

Файловые системы Unix требуют периодической проверки на ошибки, возникающие из-за физических проблем оборудования и некорректного выключения системы (отключение питания). При неправильном отключении кэш ФС в памяти может не совпадать с данными на диске. Несмотря на поддержку журналирования, правильное завершение системы критично.

fsck - инструмент для проверки ФС. Для каждого типа ФС существует своя версия (e2fsck для ext2/ext3/ext4 распознается автоматически).

```
# fsck /dev/sdb1
```

Никогда не используйте fsck на смонтированной ФС - ядро может модифицировать данные, вызывая несоответствия и сбои. Исключение: корневой раздел в режиме read-only в однопользовательском режиме.

Проход	Описание
Pass 1	Проверка inodes, блоков и размеров
Pass 2	Проверка структуры каталогов
Pass 3	Проверка связанности каталогов
Pass 4	Проверка счетчиков ссылок
Pass 5	Проверка сводной информации групп

При ошибках fsck останавливается и запрашивает разрешение на исправление. Потерянные файлы размещаются в *lost+found* с номером в качестве имени.

Параметр **-p** (или **-a**) автоматически исправляет типичные ошибки без запроса. Дистрибутивы Linux используют fsck -p при загрузке.

При подозрении на серьезные ошибки используйте **fsck -n** для проверки без изменений. Если проблема с суперблоком: **fsck -b num** - восстановление из резервной копии. Для просмотра номеров резервных суперблоков: **mkfs -n /dev/device**.

Проверка ext3 и ext4

Журнал обеспечивает целостность данных, поэтому ручная проверка обычно не требуется. Для сброса журнала:

```
# e2fsck -fy /dev/disk_device
```

Повреждённую ФС можно смонтировать в режиме ext2.

Наихудший случай

При серьезных проблемах:

- **dd** - экстракция образа ФС на другой диск
- Монтирование в read-only режиме и сохранение данных
- **debugfs** - просмотр и копирование файлов (режим read-only по умолчанию)

Команда **fsck -y** автоматически ответит на все вопросы (крайний случай).

Если резервных копий нет - обратитесь к профессиональным сервисам восстановления данных.

4.2.12 Файловые системы специального назначения

Не все ФС хранят данные на физических носителях. Многие служат системными интерфейсами, предоставляя информацию о процессах, ядре и оборудовании. Эта идея восходит к /dev и /proc механизмам Unix.

ФС	Точка монтирования	Назначение
proc	/proc	Информация о процессах и ядре. Нумерованные каталоги соответствуют PID, /proc/self - текущему процессу. Содержит /proc/cpuinfo и прочие данные оборудования (передаются в /sys)
sysfs	/sys	Системная информация устройств (см. глава 3)
tmpfs	/run и др.	Временное хранилище в памяти и swap. Параметры: size, nr_blocks. Осторожно: переполнение памяти вызовет сбои
squashfs	/snap и др.	Read-only ФС со сжатым содержимым, извлекаемым по требованию. Используется snap-пакетами
overlay	контейнеры	Объединяет каталоги в составную систему. Применяется в контейнеризации (глава 17)

4.3 Область подкачки swap

Раздел диска используется не только для ФС, но и для расширения оперативной памяти. Система виртуальной памяти Linux автоматически перемещает неактивные страницы памяти на диск (подкачка). Область диска для этого называется **swap** (подкачка).

Просмотр использования swap:

```
$ free
total used free
Swap: 514072 189804 324268
```

Раздел диска как область подкачки

Шаг	Команда	Описание
1	-	Убедиться, что раздел пуст
2	mkswap /dev/xxx	Поместить сигнатуру swap на раздел
3	swapon /dev/xxx	Активировать область в ядре

Для автоматической активации при загрузке добавьте в /etc/fstab:

```
/dev/sda5 none swap sw 0 0
```

Или с UUID:

```
UUID=xxxx-xxxx none swap sw 0 0
```

Файл как область подкачки

При необходимости используйте обычный файл вместо раздела:

```
# dd if=/dev/zero of=swap_file bs=1024k count=num_mb
# mkswap swap_file
# swapon swap_file
```

Для деактивации используйте **swapoff** (требуется свободная память в других областях).

Определение размера области подкачки

Классическое правило: swap = 2 × RAM. Однако это зависит от использования:

- **Многопользовательские системы**: swap = 2 × RAM - перемещение памяти неактивных пользователей
- **Однопользовательские системы**: зависит от количества одновременно активных процессов
- **Серверы с высокой нагрузкой**: swap минимален или отсутствует (избежать дисковых операций)

Частое обращение к swap вызывает критическое падение производительности (диск медленнее памяти).

На компьютере общего назначения swap необходима. Без неё при исчерпании памяти ядро запускает OOM Killer, убивая процессы. На серверах с мониторингом и балансировкой нагрузки это предотвращается архитектурно.

Подробнее о виртуальной памяти - глава 8.

4.4 Менеджер логических томов LVM

Гайд по пользованию LVM

Прямое управление разделами затруднено при изменении конфигурации после установки. Добавление диска требует переразбиения, переноса данных и перезагрузок. **LVM** (Logical Volume Manager) решает эту проблему добавлением слоя абстракции между физическими устройствами и файловыми системами.

Компоненты LVM

Компонент	Описание
Physical Volume (PV)	Физический блочный диск или раздел (/dev/sda1, /dev/sdb и т. д.)
Volume Group (VG)	Общий пул, созданный из одного или нескольких PV
Logical Volume (LV)	Виртуальный блочный диск, выделяемый из VG. На LV размещаются ФС или swap

Логические тома действуют аналогично разделам на физическом диске, но размещение управляет LVM, а не пользователь.

4.6 Что находится внутри традиционной файловой системы

Традиционная ФС Unix состоит из двух компонентов:

- **Пул блоков данных** - хранилище информации
- **Система БД на основе inode** - управление пулом данных

inode - структура данных, описывающая файл: тип, права доступа, расположение в пуле. Идентифицируются номерами в таблице inode.

Каталоги - тоже inode, содержащие список имен файлов и ссылки на другие inode.

Пример структуры:

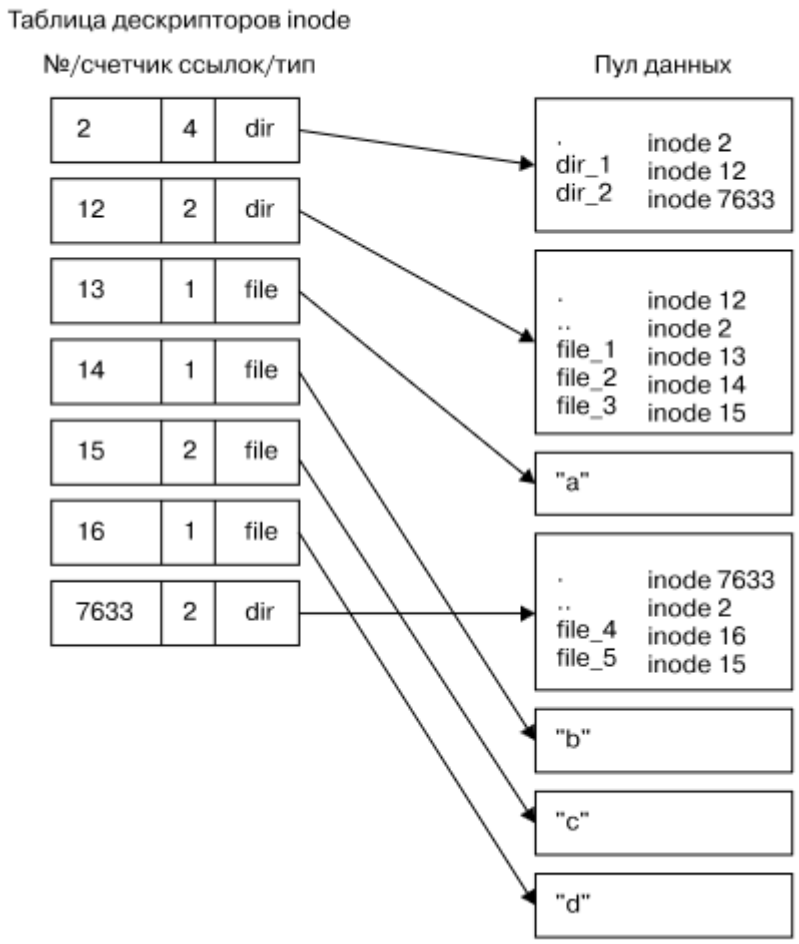
```
mkdir dir_1 dir_2
echo a > dir_1/file_1
echo b > dir_1/file_2
echo c > dir_1/file_3
echo d > dir_2/file_4
ln dir_1/file_3 dir_2/file_5   # жесткая ссылка
```

Разрешение пути (например `dir_1/file_2`):

Шаг	Действие
1	Парсинг пути: компоненты `dir_1` и `file_2`
2	Переход по корневому inode (2) к данным каталога
3	Поиск `dir_1` в inode 2 → указывает на inode 12
4	Проверка inode 12: это каталог
5	Переход по inode 12 к его данным каталога
6	Поиск `file_2` в inode 12 → указывает на inode 14
7	Проверка inode 14: это файл, доступ разрешен

Специальные записи каталогов:

- `.` - текущий каталог (ссылка на себя)
- `..` - родительский каталог (кроме корневого)



4.6.1 Сведения о дескрипторе и количество ссылок

Для просмотра дескриптора используйте `ls -i` или `stat`.

Link count - общее количество записей каталога, указывающих на дескриптор.

Операция	Действие
Создание файла	Новая запись каталога + новый дескриптор
Жесткая ссылка	Вручную созданная запись на существующий дескриптор (`ln` без `-s`)
Удаление файла	Удаление ссылки: счетчик −1, при 0 дескриптор удаляется

При удалении `rm file`: ядро находит запись каталога, уменьшает счетчик. Если `rm dir_1/file_3`, счетчик inode 15 = 2→1 (т.к. `dir_2/file_5` еще указывает на него) - дескриптор остается.

Для каталогов: счетчик включает саму папку (.), родительскую (..) и все подкаталоги. Корневой inode имеет +1 ссылку в суперблоке.

4.6.2 Распределение блоков

Block bitmap (битовая карта) - каждый бит = один блок:

- 0 = свободен

- 1 = используется

Проблемы возникают при рассинхронизации дескрипторов и распределения блоков (некорректное выключение). Программа `fsck`:

```
# Проверяет и восстанавливает
fsck /dev/device
```

Процесс: сканирование таблицы дескрипторов → генерация новых счетчиков и битовой карты → сравнение с диском → исправление → «осиротевшие» файлы в `lost+found`.

4.6.3 Работа с файловыми системами в пользовательском пространстве

Процессы доступ получают через системные вызовы ядра (`stat()` возвращает inode и счетчики).

Данные доступны в основном для обратной совместимости. Не все ФС имеют inode:

Файловая система	Жесткие ссылки	Примечание
ext4	Да	Традиционная UNIX ФС
VFAT	Нет	`ln` не работает
VFS (абстракция)	Условно	Возвращает значения, не всегда значимые

Системные вызовы Linux обеспечивают достаточную абстракцию - знать о дескрипторах не обязательно. Поддержка ФС может быть в ядре или пользовательском пространстве (FUSE).