Администрирование операционной системы Linux
(6 семестр)

Лекция 7. Работа с жесткими дисками и файловыми системами

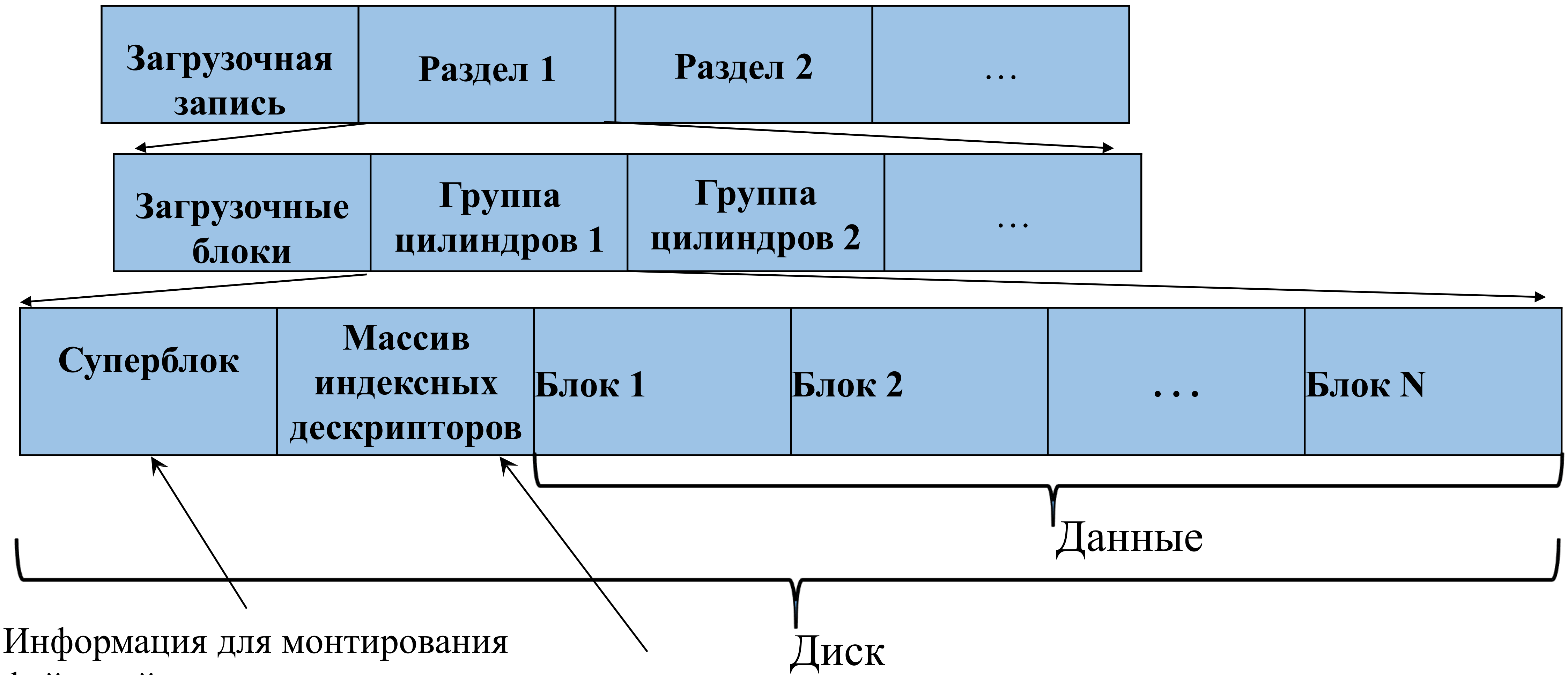
Старший преподаватель, к.т.н. Митричев Иван Игоревич

Москва 2020

План лекции

- Устройство файловой системы.
- Использование жестких связей и символических ссылок.
- Работа с жесткими дисками и файловыми системами.
- Физическая структура накопителя.
- Имена жестких магнитных дисков.
- Создание разделов при помощи fdisk. Создание файловой системы. Проверка целостности файловой системы.
- Монтирование файловых систем.
- Работа с разделом подкачки.
- Мониторинг дисковых ресурсов.
- Оптимизация производительности диска IDE.
- Резервное копирование.
- Команда dd.
- Архивирование файлов.
- Производительное копирование файлов при помощи утилиты rsync.

Устройство файловой системы ext4



Информация для монтирования файловой системы:
тип файловой системы;
размер блока и количество блоков;
и т.п.

Данные о файлах

В загрузочных блоках находится загрузчик системы стадии 2.

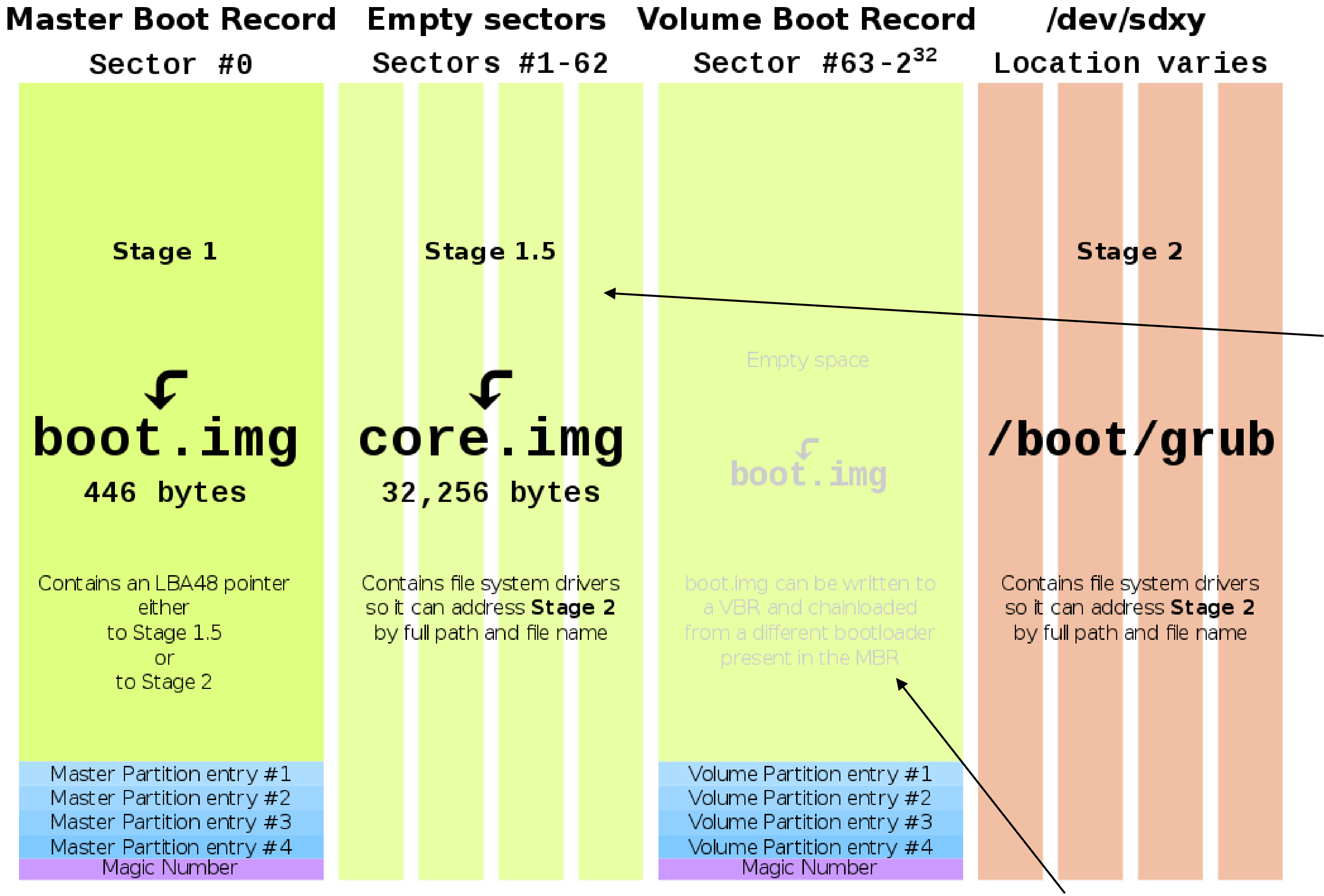
Процесс загрузки Linux

BIOS	Базова система ввода/вывода Загружает MBR
MBR	Начальная загрузочная запись Загружает GRUB
GRUB	GRand Unified Bootloader Загружает ядро
Kernel	Ядро Загружает /sbin/init
Init	Init Загружает программы уровней выполнения
Runlevel	Программы уровней выполнения загружаются из /etc/rc.d/rc*.d/

EFI

GPT

Загрузка Linux с Grub и MBR



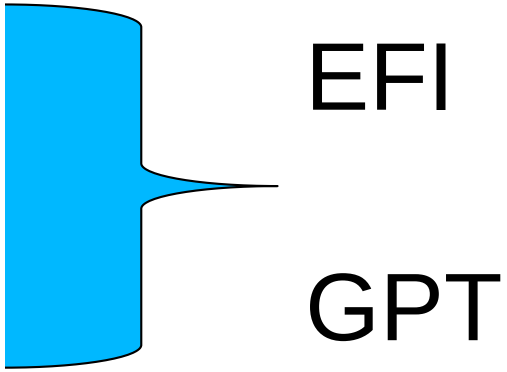
Каждая запись раздела состоит из 16 байт (октетов)

Загрузчик стадии 1.5 содержит драйвера файловой системы, чтобы вызвать загрузчик стадии 2 по полному пути /boot… Загрузчик стадии 1.5 Grub показывает меню, и позволяет вызвать загрузчик стадии 2 из нужного раздела. Если на стадии 1 использован загрузчик Windows, он сразу передает управление загрузчику стадии 2 в раздел, минуя загрузчик стадии 1.5.

boot.img может быть записан сюда и загружен по цепочке (chainloaded)

К чтению https://ru.wikipedia.org/wiki/Главная_загрузочная_запись

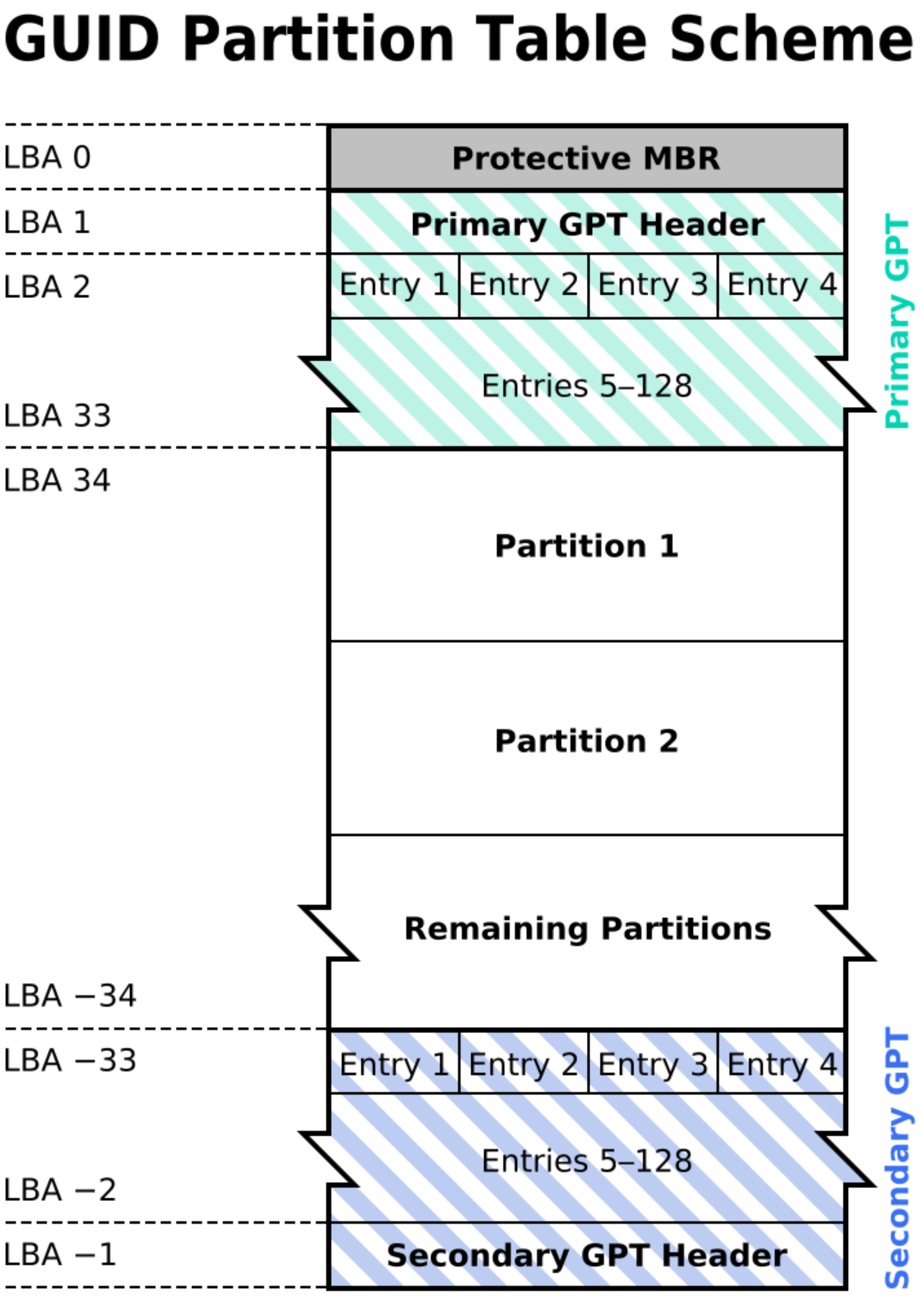
Процесс загрузки Linux с EFI



С GPT-раздела с идентификатором EF00 и файловой системой FAT32 загружается файл \efi\boot\boot[архитектура].efi, к примеру \efi\boot\bootx64.efi
Загрузчик Linux при этом не нужен.

Режим совместимости - возможна загрузка с MBR.

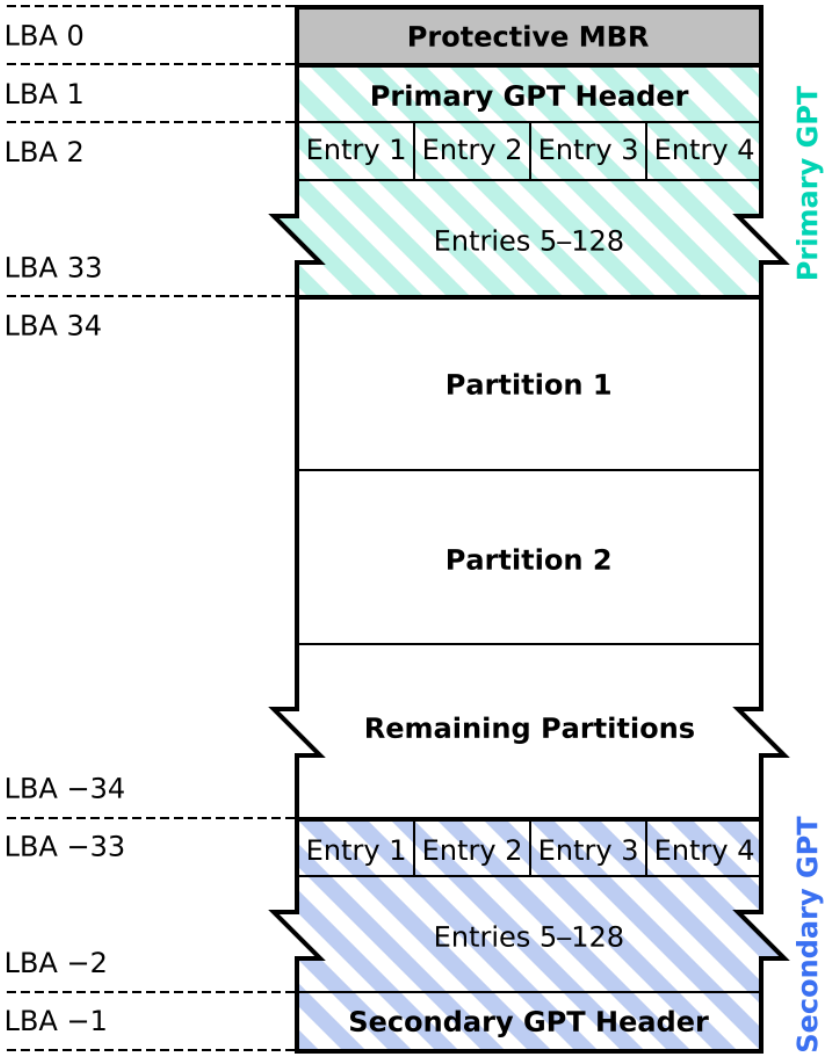
Таблица разделов GPT



- Пришли на смену MBR.
- Логическая адресация блоков (LBA) по 512 байт вместо «цилиндр-головка-сектор» (CHS). Универсальнее. Можно использовать для флэш-дисков.
- LBA0 = Защитная MBR - защита от ПО, не понимающего GPT + совместимость
- LBA1 - первичный заголовок GPT. Содержит указатель на место таблицы разделов (обычно - LBA2), контрольные суммы таблицы разделов + своя.
- Дублирование таблицы разделов в конце диска.

К чтению https://ru.wikipedia.org/wiki/Таблица_разделов_GUID

Таблица разделов GPT (продолжение)


GUID Partition Table Scheme

- Записи данных о разделах содержат GUID типа раздела (16 байт) и GUID раздела.
- Всего одна запись - 128 байт, 4 записи в одном LBA.
- Место зарезервировано по умолчанию под 128 записей разделов.

Диск делится на разделы (partitions). При использовании MBR - максимум - 4 первичных, один из разделов может быть расширенным (extended) и содержать внутри себя в первом секторе расширенную таблицу разделов с двумя записями - логический раздел (logical partition) и ссылку на следующий логический раздел.

К чтению: https://habr.com/ru/post/347002/

Индексные дескрипторы

В файловой системе ext4: 2 – inode корневой файловой системы (/);

```
2 drwxr-xr-x 25 root root 4096 апр  3 11:25 /
```

`stat <filename>` ← получить информацию об индексном дескрипторе файла.

`ls -li` ← получить номер индексного дескриптора файла.

Использование жестких и мягких связей

Жесткая связь – разные имена для файла, имеющего один заданный inode.

```
ln <filename> <newname>
```

- создать жесткую связь;

```
cp -l
```

- создавать жесткие связи вместо копирования.

Символическая ссылка (мягкая связь) – указатель на другой файл.

```
ln -s <filename> <linkname>
```

- создать символическую ссылку;

```
cp -s
```

- вместо копирования создавать символические ссылки.

При копировании файлов с использованием cp, автоматически копируются файлы по ссылкам.

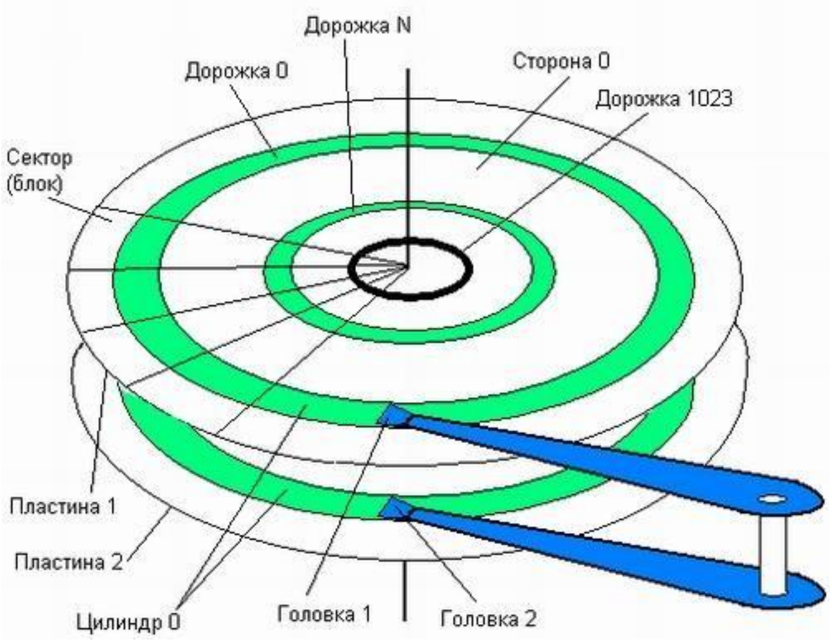
```
cp -d
```

копировать сами ссылки, а не файлы

Устройство жесткого диска

Накопители на жестких магнитных дисках состоят из одного или нескольких магнитных дисков и магнитных головок (head). Магнитная головка перемещается над поверхностью одного магнитного диска.

Магнитная поверхность диска разбита на дорожки (track). Дорожки одного диаметра на всех магнитных поверхностях образуют цилиндр (их число - cyl).



track = cyl × head.

Каждая дорожка разбита на секторы (sect), стандартный размер которых 512 байт.
Объем диска в байтах:

capacity = cyl × head × sect × 512.

Стандартное количество секторов в дорожке – 63.

Основные параметры диска (геометрия) – количества цилиндров cyl и головок head.

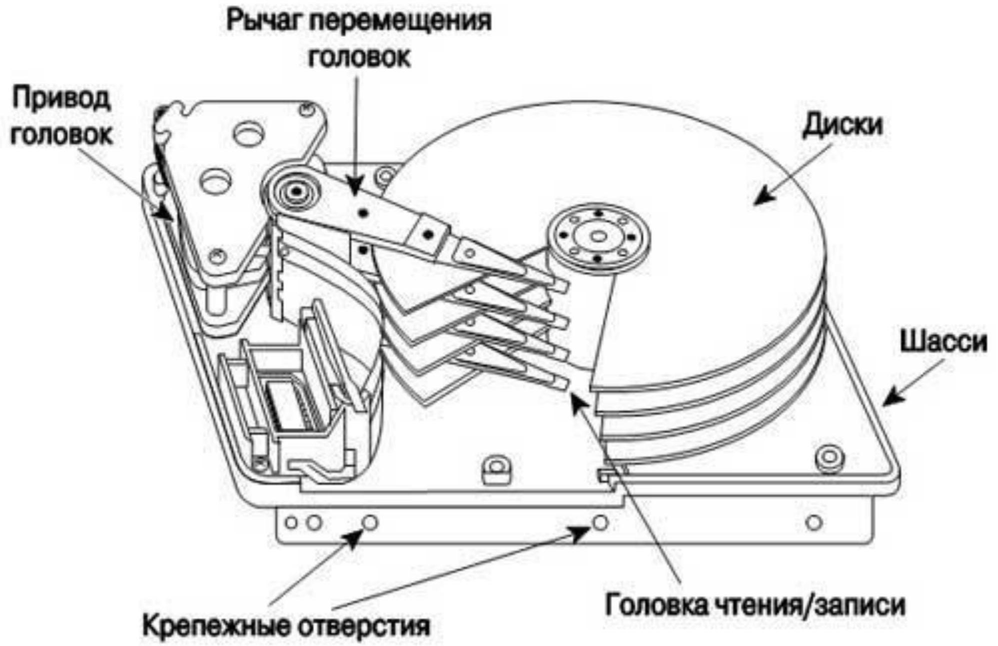
Сведения о геометрии жесткого диска

```
/sbin/fdisk -l
```

- получить сведения о геометрии диска.

Сведения о геометрии диска:

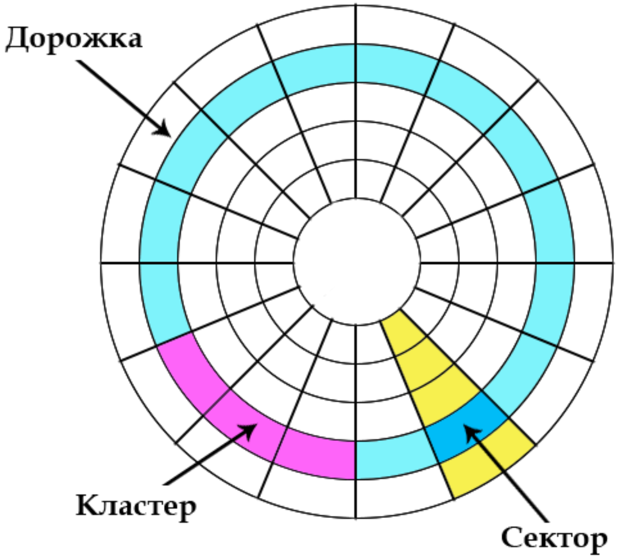
```
Disk /dev/sda: 160.0 GB, 160041885696 bytes
255 heads, 63 sectors/track, 19457 cylinders, total
312581808 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: …..
```



Твердотельные накопители (SSD) имеют более высокую стоимость за ГБ объема, меньшее число циклов перезаписи, но 1) намного более быстрые (скорость чтения), 2) бесшумные, нет механических частей, 3) легкие, 4) потребляют меньше энергии.

Имена жестких дисков SATA/SCSI

/dev/sda – Primary Master
/dev/sdb – Primary Slave
/dev/sdc – Secondary Master
/dev/sdd – Secondary Slave



Команда ls -l для файлов устройств вместо размера выводит главный ID (major number) и вспомогательный (minor number).

Главный ID– номер драйвера для данного типа устройств в Linux (8- SATA жесткий диск, 4 – терминал, 1 – псевдоустройство - /dev/null, /dev/random, dev/zero), вспомогательный – порядковый номер устройства данного типа (0 – для всего диска, 1,2,3,4 – для первичных разделов, 5-15 – для логических разделов).

Изменение разделов при помощи fdisk

```
fdisk <имя устройства>
```

← редактировать таблицу разделов.

Команды интерактивной утилиты fdisk:

m – список доступных команд;

q – завершение работы без сохранения изменений;

l – список доступных типов разделов;

d – удалить раздел;

n – создать раздел;

t – установка типа созданного раздела;

w – записать таблицу разделов;

p – напечатать таблицу разделов;

x – дополнительные функции (для экспертов).

Пример:

```
Command (m for help): n
Partition type:
p     primary (3 primary, 0 extended, 1 free)
e     extended
Select (default e): e
Select partition 5
First sector (102326202-485063864): 102326202
Last sector, +sectors or +size{K,M,G} (102326202-
485063864, default 485063864):
Using default value 485063864
Partition 5 of type Linux and of size 195 GiB is set
```

Создание файловой системы

```
mkfs <раздел>
```

- создание файловой системы;

Опции команды:

- t – тип ФС;
- c – проверка на bad blocks;
- b – установить размер блока (1024/2048/4096) в байтах.

Пример:

```
# mkfs -t ext4 /dev/sda6
```

Проверка целостности файловой системы

При сбое питания или другом сбое файловая система (ФС) может потерять целостность. При этом может произойти

- частичная потеря данных, над которыми производились операции;
- появление inode, которые указывают на уже освобожденные блоки, и, наоборот, появление блоков данных, на которые не ссылается ни один inode;
- другие искажения системы метаданных.

При запуске системы Ubuntu на экране можно видеть запись /dev/sda5 – clean или ошибки, как например «orphaned block…». Первая запись свидетельствует о флаге clean файловой системы (целостное состояние), вторая запись – о нарушении целостности ФС.

```
fsck <опции> <раздел>
```

← • проверить целостность ФС;

Проверять целостность можно только для отмонтированной ФС! Иначе, возможна полная потеря данных.

Файловая система ext4, используемая часто для Ubuntu, поддерживает журналирование.
Журналирование – инструмент поддержания целостности ФС. Восстановление по журналу позволяет избежать в большинстве случаев запуска fdisk после сбоя работы Ubuntu.

Команда fsck

Утилиты восстановления целостности файловой системы не восстанавливают данные, а лишь возвращают ее к неповрежденному, непротиворечивому состоянию (consistency).

```
# fsck -t ext3 /dev/sdb2
```

- проверить целостность файловой системы ext3;

```
ls -l /sbin/fsck.*
```

- список утилит для проверки ФС.

<u>Как работает журналирование</u> – на диске выделяется место для хранения информации об операциях с блоками. Все операции с диском происходят как транзакции, где в журнале записывается полная информация о транзакции. При внезапном сбое по журналу можно понять, какие этапы были сделаны, а какие – нет. Например, при перенесение файла на другой раздел, 1) сначала копируется содержимое, 2) устанавливается новый inode, 3) затем удаляется старый inode. При сбое после шага 2) файл останется в двух местах, если не ведется журнал.
Ext2 – без поддержки журналирования (между тем, данные нетрудно восстановить утилитами восстановления данных). Ext3, Ext4 – с журналированием. **<u>Доступные режимы журналирования (ext3):</u>**
Writeback – журналирование только метаданных. При сбоях данные могут теряться.
Ordered – в журнале сохраняются только метаданные. Метаданные изменяются после записи данных на диск. Это - значение по умолчанию. Таким образом, если есть изменения метаданных, то данные точно попали в кеш жесткого диска (не обязательно успели записаться до сбоя).
Journal – как данные, так и метаданные, сохраняются в журнале. Любое изменение метаданных гарантировано предваряется изменением данных на диске.
<u>В Ext4 журналирование можно отключить</u> (режим Off). Ext3 поддерживает файлы до 2 TB, файловая система может быть до 32 TB. Ext4 поддерживает файлы до 16 TB, файловая система может быть до 1 EB (Exabyte = 1024 Petabyte =

Монтирование файловых систем

Монтирование – процесс подключения файловой системы, существующей на блочном устройстве, к корневой файловой системе.

Точка монтирования – директория, в которую монтируется устройство.

/mnt – директория для точек монтирования ФС внутренних накопителей;

/media – директория для точек монтирования ФС на съемных носителях.

```
mount
```

- вывод таблицы примонтированных устройств;

```
mount <опции> <устройство> <точка монтирования>
```

- монтирование;

```
umount <устройство>
umount <точка монтирования>
```

- отмонтировать

Примеры:

```
mkdir /mnt/new_disk
mount /dev/sda5 /mnt/new_disk
```

Отмонтировать корневую ФС можно

1) в однопользовательском режиме (уровень init = 1)
2) загрузившись с другой ОС Linux (например, установочный диск)

```
# init 1
# umount /home
# umount /dev/sda
```

Работа с разделом подкачки

Своппинг (подкачка) – перемещение неактивных страниц памяти из оперативной памяти на диск для экономии места в оперативной памяти (в Linux – исторически, чаще, это раздел диска, в windows – файл pagefile.sys в корне диска). В Linux своп-раздел (или файл) также используется для гибернации (в Windows – файл hyberfil.sys в корне диска).

```
/sbin/swapon -s
```

- вывести информацию об использовании подкачки.

Создание файла подкачки объемом 2 Гб:

```
dd if=/dev/zero of=swap.file bs=1k count=2097152
```

```
/sbin/mkswap swap.file
```

Создание раздела подкачки:

```
/sbin/mkswap -c <раздел>
```

```
swapon <файл> или <раздел>
```

- включить подкачку;

```
swapoff <файл> или <раздел>
```

- выключить подкачку.

Подкачка может нарушить безопасность при использовании зашифрованных данных, так как на диске могут оказаться незашифрованные данные из оперативной памяти!

Файл /etc/fstab

Файл /etc/fstab содержит список файловых систем, монтируемых автоматически при загрузке, или монтируемых по требованию пользователя.

Поля fstab: имя файла монтируемого устройства; точка монтирования; тип файловой системы; опции монтирования; <dump>: надо ли для этой ФС производить автоматическое резервное

копирование командой dump: 1 – разрешено, 0 – запрещено. Порядок проверки раздела (0- не проверяется, 1 - для корневого раздела, 2 - для остальных разделов).

```
mount <точка монтирования> или <файл устройства>
```

- при наличии записи в fstab.

Опция монтирования	Цель использования
defaults	опции rw, suid, dev, exec, auto, nouser, asynch
asynch	асинхронный режим ввода-вывода.
auto/noauto	включить/отключить автомонтирование во время загрузки
exec/noexec	разрешение/запрет на выполнение файлов
suid/nosuid	можно/нельзя устанавливать бит SUID
user/nouser	можно/нельзя монтировать обычному пользователю
ro	монтировать только для чтения
rw	Монтировать для чтение и записи

После изменения fstab применить изменения без перезагрузки можно так:

```
mount -o remount раздел
```

Оптимизация производительности диска

```
hdparm <имя файла устройства>
```

- вывести текущие параметры жесткого диска;

```
hdparm <опции> <имя файла устройства>
```

- установить параметры жесткого диска.

Пример: вывод полной информации о диске

```
$sudo hdparm -i /dev/sda
/dev/sda:
 Model=WDC WD10EZEX-08WN4A0, FwRev=01.01A01,
SerialNo=......
```

```
hdparm -t /dev/sda
```

– тест скорости чтения с диска

```
hdparm -I /dev/sda
```

– список опций для ускорения диска (использовать часть из них без опыта может быть опасно для диска!)

```
hdparm -M 128 /dev/sda
```

– «тихий режим» (управление скоростью диска – число от 128 до 254)

Команда dd

Низкоуровневое поблочное копирование:

dd if=<входящий файл> of=<исходящий файл>

Опции:

count=<n> – количество блоков;
skip=<n> – сколько блоков пропустить во входном файле;
seek=<n> – сколько блоков пропустить в выходном файле до начала записи.

```
dd if=/dev/sdX of=/dev/sdY bs=64K conv=noerror,sync,notrunc status=progress
```

bs - размер блока. По умолчанию 512 байт (осталось с 80-х годов). Лучше использовать значения около 1M.

noerror - продолжать работу, игнорируя все ошибки чтения. Поведение по умолчанию - dd должно останавливаться при любой ошибке.

sync - заполняет входные блоки с нулями, если есть ошибки чтения, поэтому данные в источнике и приемнике остаются синхронными, несмещенными.

status=progress - показывает статистику.

Команда tar

```
tar <опции> <имя архива> <файлы и каталоги>
```

Извлечь файлы

```
tar -xvzf archive.tar.gz
```

Пояснение к команде:

x - извлечь файлы из архива
v - печатать имена распаковываемых файлов
z - файл является gzip-сжатым файлом
f - имя файла с архивом

j - bzip-сжатие

```
tar -xvjf archive.tar.bz2
```

Извлечь в заданную директорию

```
tar -xvzf archive.tar.gz -C /opt/folder/
```

Извлечь только заданные файлы

```
tar -xvz -f archive.tar.gz "./1.txt" "./2/3.txt"
```

-t - просмотреть содержимое tar-архива

Архивировать директорию (или файл)

```
tar -cvf my_archive.tar dir/
tar -czvf my_archive.tar.gz dir/
tar -cjvf my_archive.tar.bz dir/
```

Добавить файл в существующий несжатый архив (сжатый - надо распаковть и запаковать заново)

```
tar -rv -f my_archive.tar 555.txt
```

Резервное копирование папки ~/main_documents на примонтированный удаленный сервер

```
tar -cvz -f
/mnt/installed_server/daily_docs_$(дата+%Y%m%d)
.tar.gz ~/main_documents
```

Использование rsync

```
rsync <опции> <откуда> <куда>
```

← копирование файлов с синхронизацией

Опции:

-v – сообщать о проводимых операциях;

-r – копировать данные рекурсивно (но не сохранять метки времени права);

-a – рекурсивно копировать файлы и сохранять символические ссылки, права и временные метки;

-z – сжать данные файла;

-h – удобный формат чисел в выводе размер файлов;

-e протокол - использовать указанный протокол;

--partial - продолжить прерванную передачу файлов, а не начинать сначала;

--progress - показывать процент выполнения операций;

-P = --progress и --partial;

-u – пропуск файлов, существующих в месте назначения, с более поздней датой модификации.

Копирование директории на сервер и с сервера

```
rsync -avz dir/ user@10.0.1.214:
rsync -avz user@10.0.1.214:dir/.
```

Копирование через защищенное соединение

```
rsync -avzhe ssh --progress user@8.8.8.8:dir/.
```

Продолжить прерванную передачу файлов

```
rsync -avzhe ssh -P user@8.8.8.8:dir/.
```

Резервное копирование (по умолчанию, старые файлы в destination получают тильду перед названием):

```
rsync -a -b source/ destination/
```

Залог сохранности данных – регулярное обязательное резервное копирование!

Места хранения резервных копий:

- Съемные носители;
- Удаленные машины / облачные сервисы.

Создание файла с ФС

```
$ dd if=/dev/zero of=file.img bs=1k count=10000
10000+0 records in
10000+0 records out
# создадим устройство-заглушку
$ losetup /dev/loop0 file.img
# Создание файловой системы ext2
$ mke2fs -c /dev/loop0 10000
mke2fs 1.35 (28-Feb-2004)
max_blocks 1024000, rsv_groups = 1250, rsv_gdb = 39
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
2512 inodes, 10000 blocks
500 blocks (5.00%) reserved for the super user
```

```
$ mkdir /mnt/point1
$ mount -t ext2 /dev/loop0 /mnt/point1
$ ls /mnt/point1
lost+found
```

Файловая система создана и примонтирована