Пользователи и группы

Типы пользователей

- **root** - суперпользователь UID=0
  - имеет право на выполнение всех операций. Присутствует в системе по умолчанию. UID=0
- **Системные (фиктивные) пользователи** UID=1-999
  - системные процессы у которых есть учетные записи для управления привилегиями и правами доступа к файлам.
  - Создаются системой автоматически
- **Обычные пользователи**
  - учетные записи пользователей, допущенных к управлению системой. Создаются суперпользователем. UID >=1000

Учетные записи пользователей

- Системное имя (user name)
- Пароль
- Идентификатор пользователя (UID)
- Идентификатор группы (GID)
- Полное имя (full name)
- Домашний каталог (home directory)
- Начальная оболочка (login shell)

Учетные записи пользователей

- Информация о пользователях хранится в файлах
  - /etc/passwd пользователи
  - /etc/group группы пользователей
  - /etc/shadow зашифрованные пароли пользователей
  - /etc/gshadow зашифрованные пароли групп

- Дополнительно
  - /etc/default/useradd - свойства по умолчанию для новых пользователей
  - /etc/login.defs - настройки новых пользователей
  - /etc/skel/ - каталог, файлы из которого копируются в домашний каталог нового пользователя

Информация о пользователях
cat /etc/passwd

```
aktios@ubuntu:/etc$ cat passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
saned:x:108:115::/home/saned:/bin/false
whoopsie:x:109:116::/nonexistent:/bin/false
speech-dispatcher:x:110:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/
sh
avahi:x:111:117:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
lightdm:x:112:118:Light Display Manager:/var/lib/lightdm:/bin/false
colord:x:113:121:colord colour management daemon,,,:/var/lib/colord:/bin/false
hplip:x:114:7:HPLIP system user,,,:/var/run/hplip:/bin/false
pulse:x:115:122:PulseAudio daemon,,,:/var/run/pulse:/bin/false
aktios:x:1000:1000:ubuntu VM,,,:/home/aktios:/bin/bash
```

Информация о группах

```
root@ubuntu:~# cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,aktios
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:aktios
floppy:x:25:
tape:x:26:
sudo:x:27:aktios,poit5
audio:x:29:pulse
dip:x:30:aktios
```

Команды управления учетными данными

- **Команды для пользователя**
  - useradd - добавить нового пользователя
  - **adduser** – интерактивное добавление пользователя (ответить на вопросы)
  - passwd - установить пароль пользователя
  - usermod - измененить параметры учетной записи пользователя
  - userdel - удалить учетную запись пользователя

- **Команды для группы**
  - groupadd - создать новую группу
  - gpasswd - установить пароль группы
  - groupmod - измененить параметры группы
  - groupdel - удалить группу

Файловая система

Типы файловых систем используемых в Linux

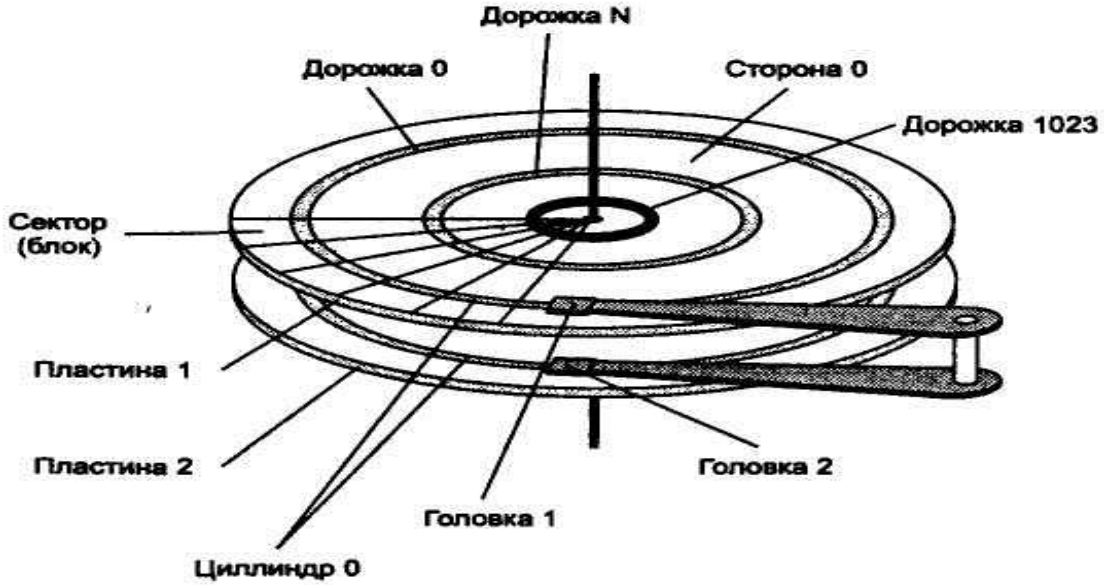
- Ext2;
- Ext3;
- Ext4;
- JFS;
- ReiserFS;
- XFS;
- Btrfs;
- ZFS.

Файловая система

- *Файловая система* – множество файлов организованных по определенным правилам на внешнем носителе (FAT, NTFS).
- *Система управления файлами* – часть операционной системы, позволяющая пользователю получить доступ к файлам конкретной файловой системы.

- Оба понятия часто объединяют под одним названием – файловая система.

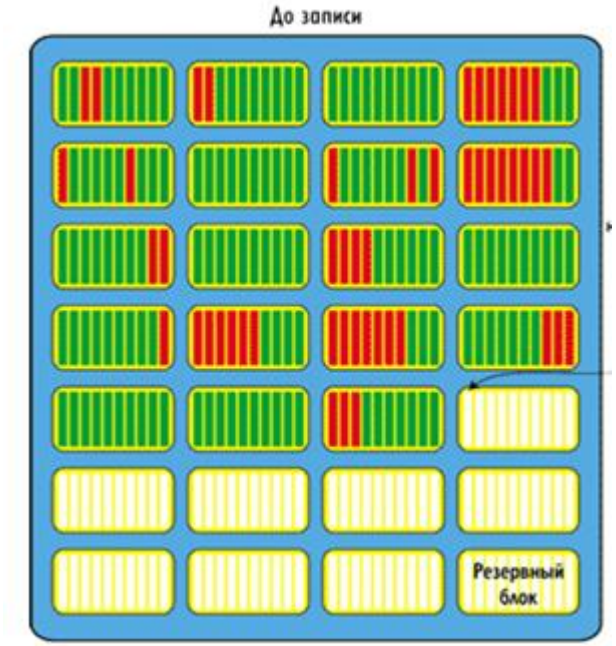
HDD

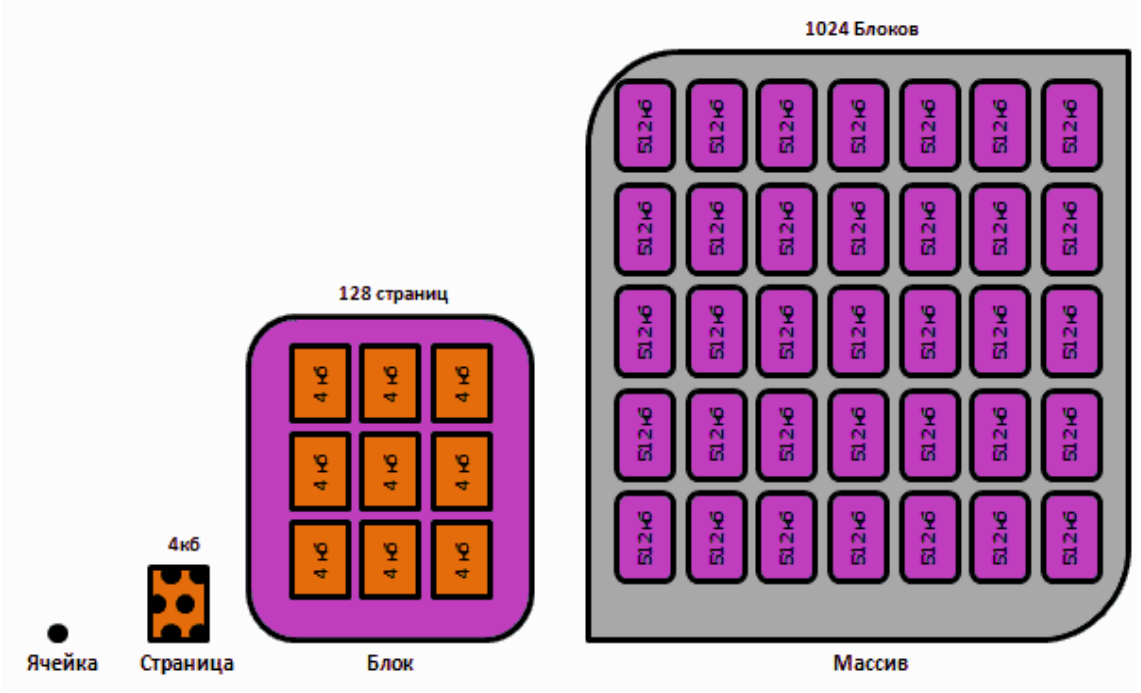


Заголовок сектора	Данные	ECC

- Вся поверхность разбивается кольцевые дорожки
- Каждая дорожка разбивается на сектора, размером 512байт
- **Адрес сектора** состоит из :
  - номера цилиндра (**Cylinder**) + номера поверхности (головки **Head**) + номера сектора (**Sector**) – **CHS -адресация**.
- Для удобства обращения **CHS** заменяется на **LBA** (*Logical block addressing*
- При **LBA** каждый сектор имеет свой логический номер
- Контроллер диска преобразует **LBA** в **CHS**

SSD-диск





- Флзш-память разбивается на страницы размером, кратным 512 байт. Каждая страница имеет свой физический адрес PBA.
- Страницы группируются в блоки, размером, кратным размеру страницы
- Информация записывается и считывается страницами, а стирается блоками
- При этом логический адрес записываемой страницы сопоставляется с физическим адресом,
- Таблица сопоставления (LBA-PBA mapping) размещается в оперативной памяти SSD-диска.

Сектор, Кластер, Форматирование диска

- **Низкоуровневое форматирование**
  - Процесс разбивки диска на **дорожки и сектора (страницы SSD),** каждому присваивается номер
  - Не зависит от типа операционной системы
  - Для всей поверхности диска размер сектора/страницы одинаков

- Кластер/блок – единица данных используемая ОС при обращении к диску.
- Один кластер включает определенное число секторов, кратное размеру сектора 1024, 2048, 4096, 8192 . . . байт
- **Высокоуровневое форматирование**
  - Процесс назначения группам физических секторов диска номеров кластеров и создания таблиц кластеров и структур данных под конкретный тип файловой системы называется высокоуровневым форматированием

Разделы

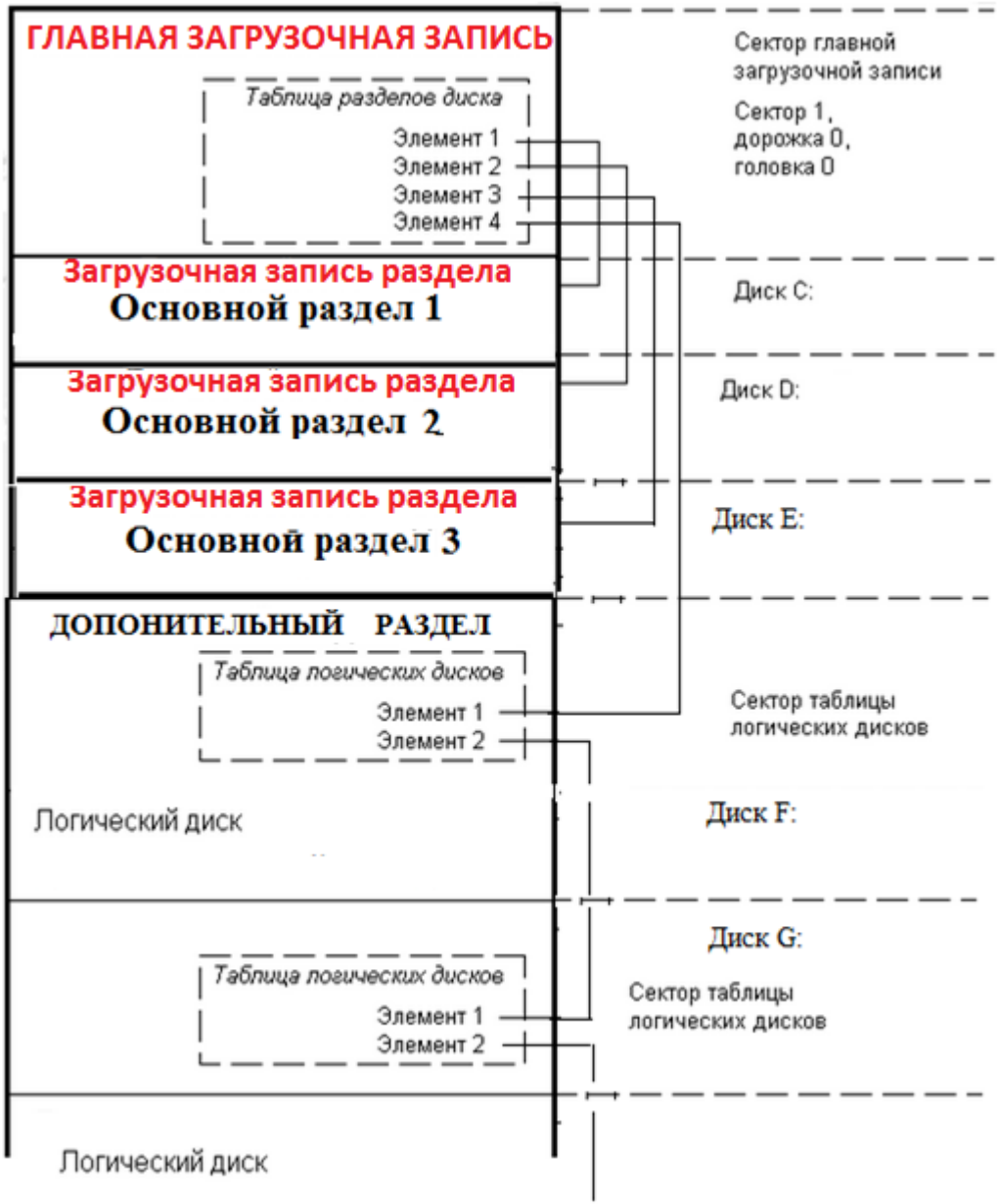
- **Раздел** — это непрерывная часть одного физического диска, которую ОС предоставляет пользователю, как отдельный диск.
  - Раздел имеет начальный и конечный номера секторов.

- Каждый раздел может быть отформатирован только под одну файловую систему.
  - Каждый раздел может иметь различный размер кластера
  - **Том** – диск или раздел, отформатированный под конкретную файловую систему

Разбивка диска на разделы

MBR – главная загрузочная запись (первичный загрузчик)
Создается при разбивке диска на разделы и модифицируется при форматировании и установке ОС.

PBR (**P**artition **B**oot **R**ecord) – **загрузочная запись раздела**
Создается при разбиении и заполняется при установке в данном разделе ОС.
Сюда записывается **вторичный загрузчик ОС.**



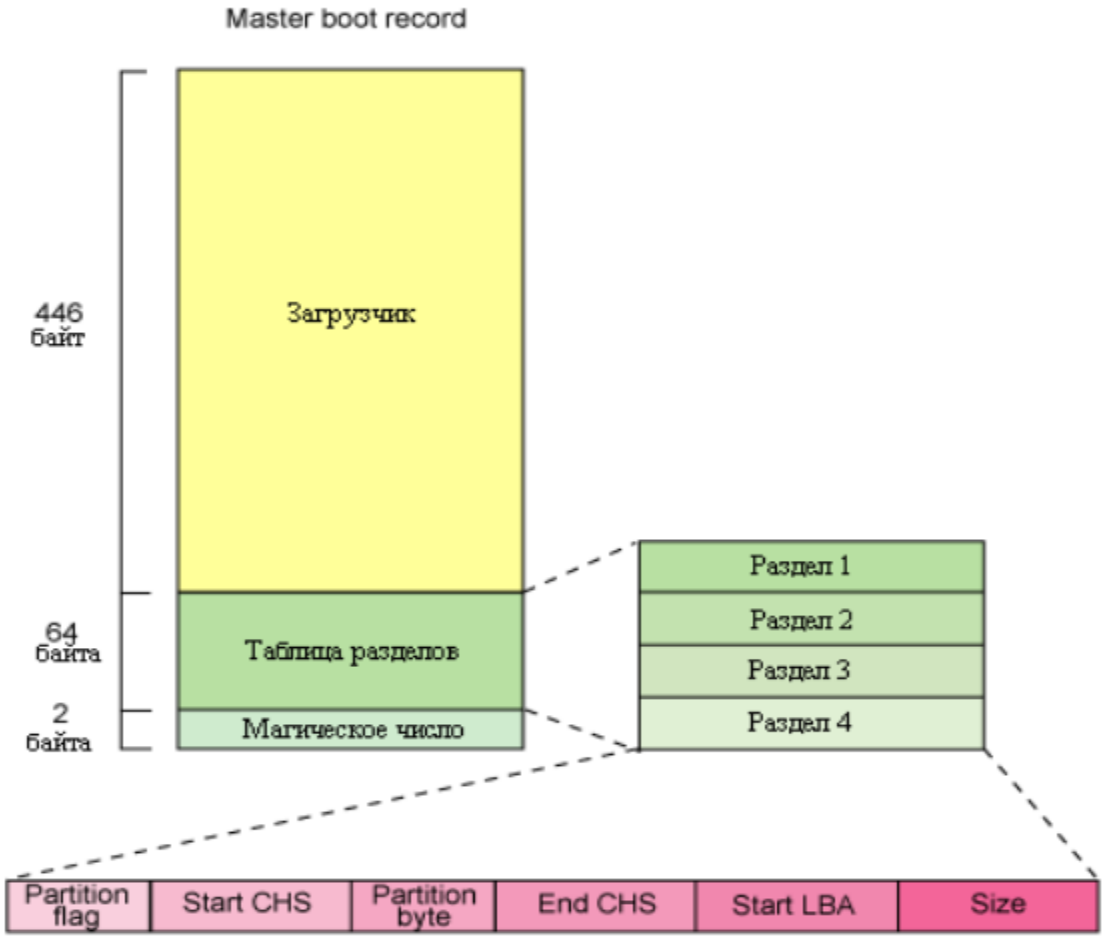
Разделы

- Раздел, на который может быть установлена ОС называется **основным** (первичным).
- Основной раздел, на который установлена ОС называется **активным**.
- Раздел, который не является основным, называется **дополнительным (расширенный).**

- При использовании механизма разбивки MBR (главной загрузочной записи) диск может быть разбит на
  - **четыре основных (первичных)** раздела
  - или на **три основных и один дополнительный** (расширенный) раздел.

-

Формат MBR



- **Главный Загрузчик** – программа начальной загрузки . Проверяет таблицу разделов, находит активный раздел и вызывает вторичный загрузчик этого раздела
- Магическое число (сигнатура диска) — 55AAh (последовательность 01)

для проверки линий передачи данных контроллера диска.

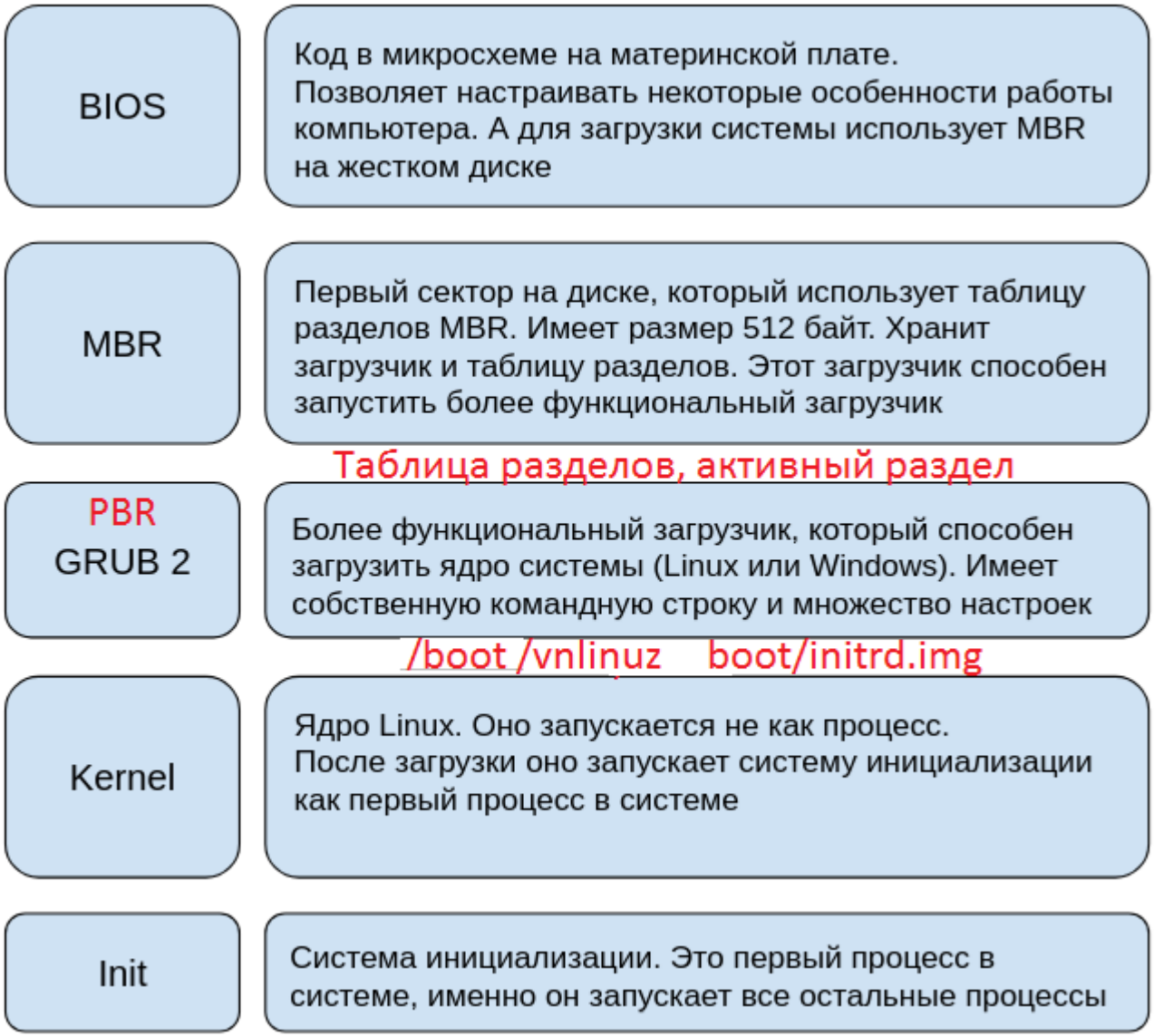
Таблица разделов (Partition)

смещение					разм.	назначение
000	1BE	1CE	1DE	1EE	BYTE	флаг активного загрузочного раздела. (Boot Indicator) 80h – загрузочный раздел, 00h – не загрузочный
001	1BF	1CF	1DF	1EF	BYTE	стартовая головка раздела
002	1C0	1D0	1E0	1F0	BYTE	стартовый сектор раздела (биты 0 – 5) старшие биты стартового цилиндра (биты 6-7)
003	1C1	1D1	1E1	1F1	BYTE	младшие биты стартового цилиндра (биты 0-7)
004	1C2	1D2	1E2	1F2	BYTE	идентификатор системы (Boot ID),
005	1C3	1D3	1E3	1F3	BYTE	конечная головка раздела
006	1C4	1D4	1E4	1F4	BYTE	конечный сектор раздела (биты 0 – 5) старшие биты конечного цилиндра (биты 6-7)
007	1C5	1D5	1E5	1F5	BYTE	младшие биты конечного цилиндра (биты 0-7)
008	1C6	1D6	1E6	1F6	DWORD	смещение раздела относительно начала таблицы разделов в секторах
00C	1CA	1DA	1EA	1FA	DWORD	кол-во секторов раздела

- Адреса секторов разделов задаются:
  - либо в формате CHS (Cylinder-Head-Sector — Цилиндр-Головка-Сектор), например 6 - 1 – 32.
  - либо LBA формате(Logical Block Address – Логический Адрес Блока, в виде последовательности номеров 01 2 ….).

- Конкретный формат определяется типом раздела (Boot ID), записанным в 04h байте

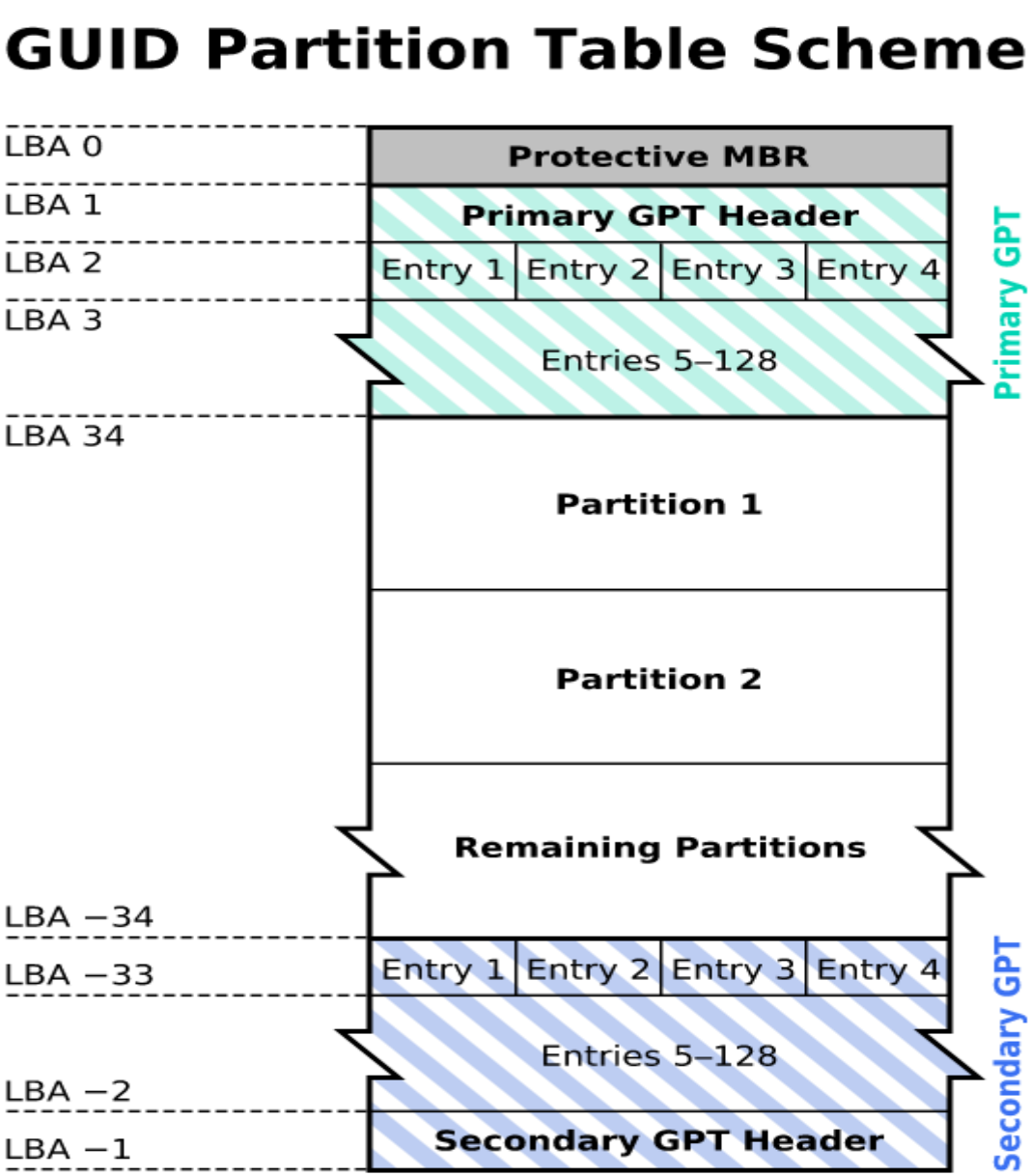
Последовательность загрузки Linux MBR



UEFI (*Unified Extensible Firmware Interface* *Унифицированный расширяемый интерфейс встроенных микропрограмм*)

- Наличие графического интерфейса для настройки.
- Поддержка загрузки операционных систем (ОС) с GPT-дисков.
- Хранение дампов после сбоя в NVRAM (при наличие возможности со стороны ОС).
- Запуск исполняемых файлов
- Добавление драйверов (*например, добавить драйвер сетевой карты и получить доступ к сети до загрузки ОС*)

Формат GPT

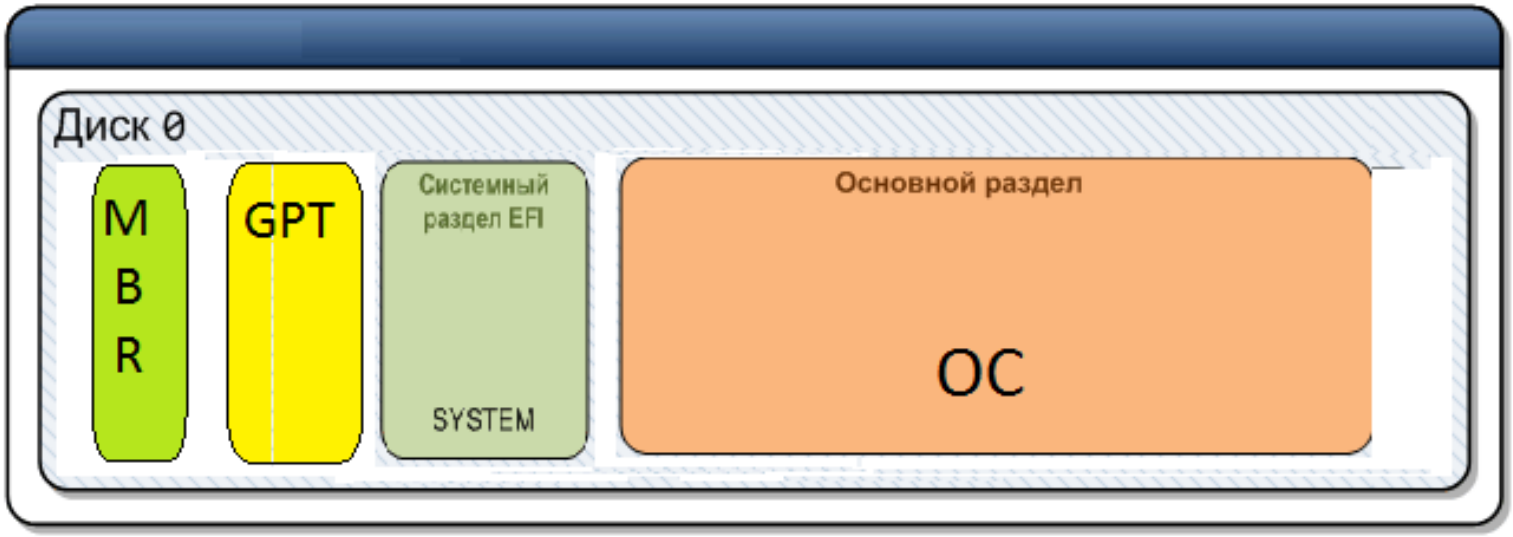


Допускает неограниченное количество разделов.

Лимит устанавливает ОС - для Windows 128 разделов. GNU/Linux-256

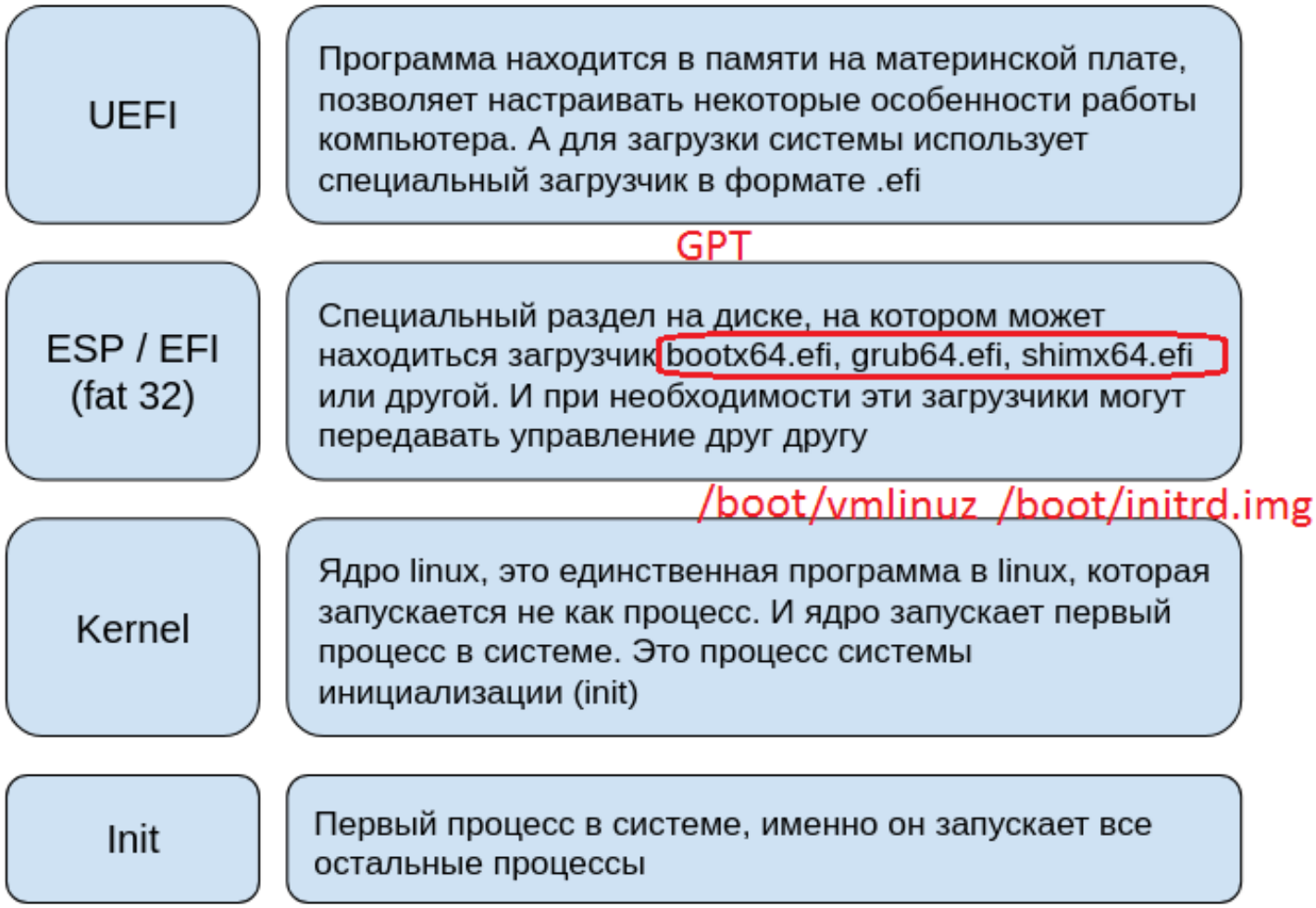
Размер одной записи о разделе в GPT 128 байт

Структура носителя UEFI



При UEFI на диске создается специальный системный раздел EFI (ESP)
После включения компьютера UEFI сначала выполняет функции системной конфигурации, также как и BIOS.
Затем UEFI считывает GPT — таблицу разделов GUID.
GPT определяет таблицу разделов на диске, на которой загрузчик EFI распознает системный раздел EFI.
GPT располагается в первых секторах диска, сразу после сектора 0, где по-прежнему хранится главная MBR загрузочная запись для Legacy BIOS.
В каталогах EFI\ должны лежать загрузчики ОС указанного поставщика для всех разделов диска.
Если у UEFI нет явного указания какой загрузчик загружать, то он загружает файл EFI\BOOT\BOOTx64.EFI

Загрузка Linux UEFI



Два способа организации каталогов

Запись о файле в каталоге

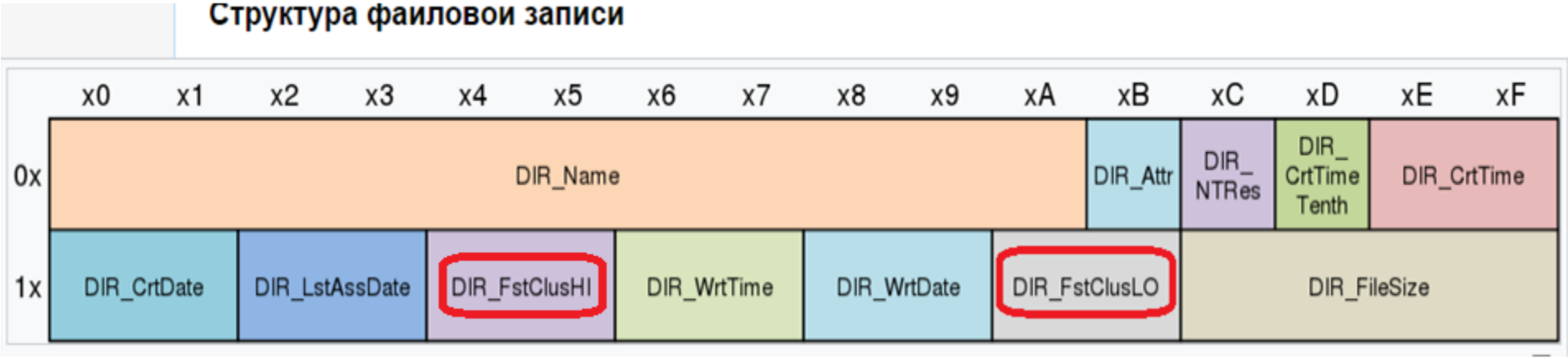
Запись (атрибуты) о файле может храниться :

- Полностью в каталоге;
- В каталоге хранится только имя файла и ссылка на блок в котором хранятся все остальные атрибуты файла.

Структура каталогов: *а* — структура записи каталога MS-DOS FAT 32 (32 байта), *б* — структура записи каталога ОС UNIX

R-только для чтения A- архивный H- скрытый S-системный

Уточненная структура файловой записи для FAT

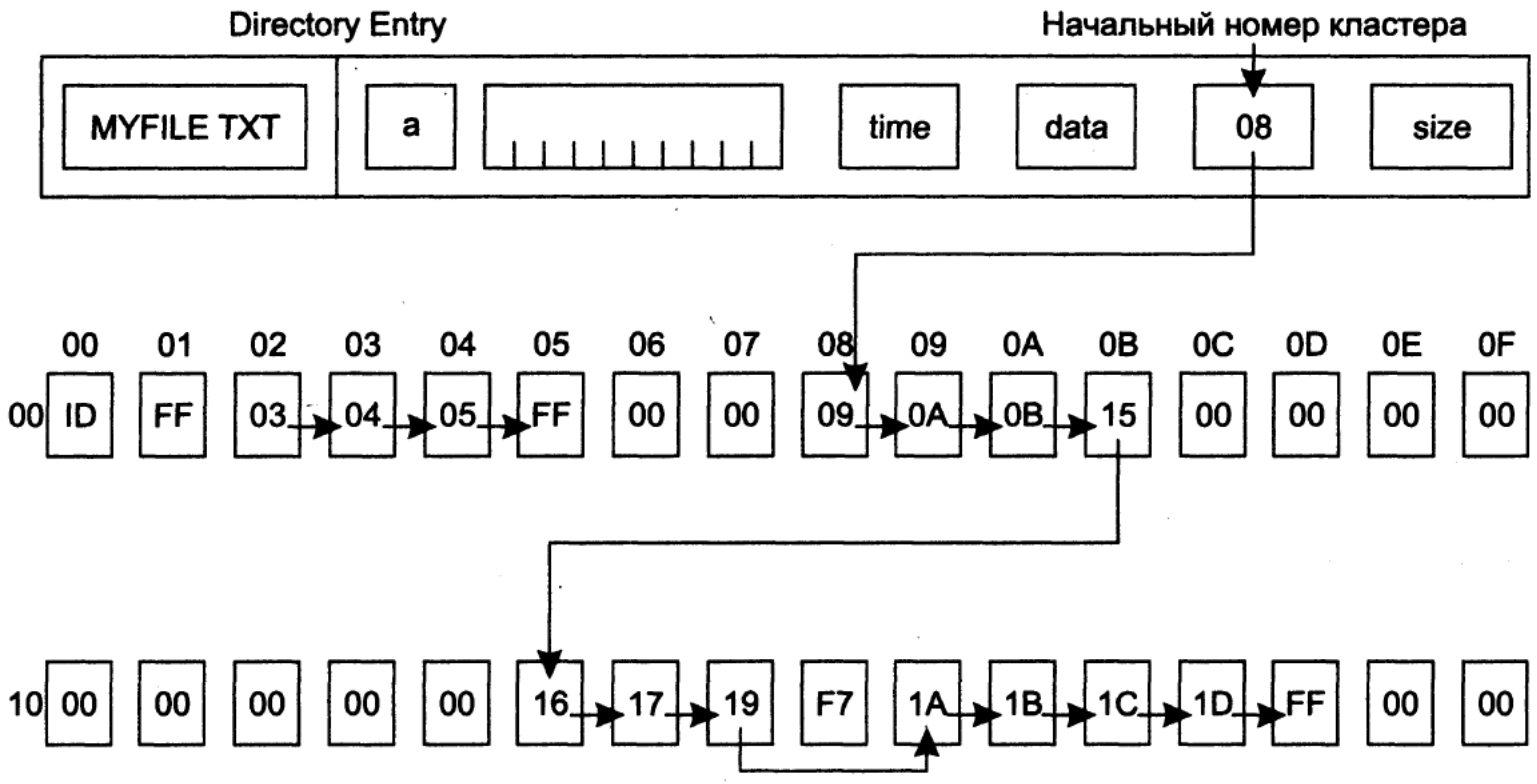


DIR_FstClusHI. 2 байта по адресу 0x14. Номер первого кластера файла (старшее слово) в FAT12/FAT16 равен нулю)

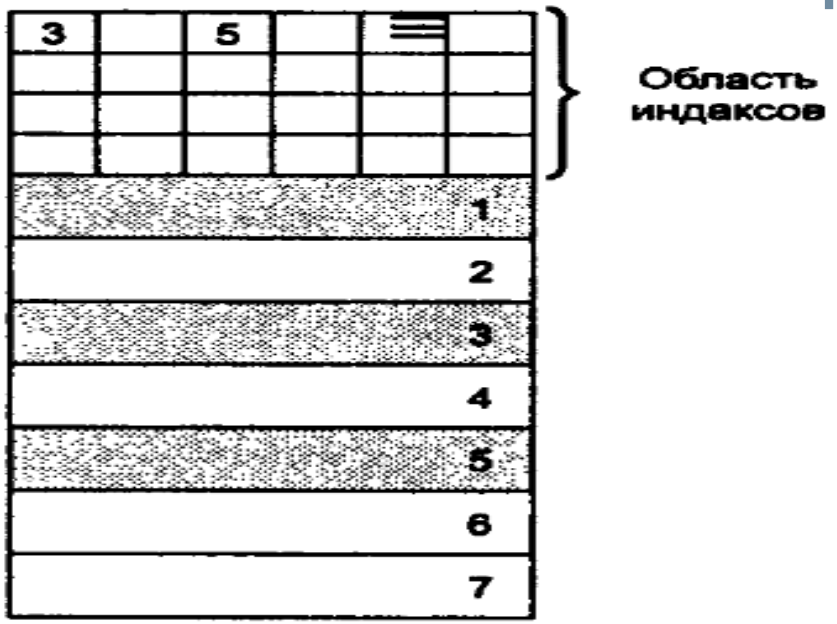
DIR_FstClusLO. 2 байта по адресу 0x1A. Номер первого кластера файла (младшее слово).

DIR_FileSize. размера файла в байтах.
Ограничение FAT32 — максимальный размер файла (4 байта) составляет 0xFFFFFFFF = 4Гб

Размещение связанными списками кластеров

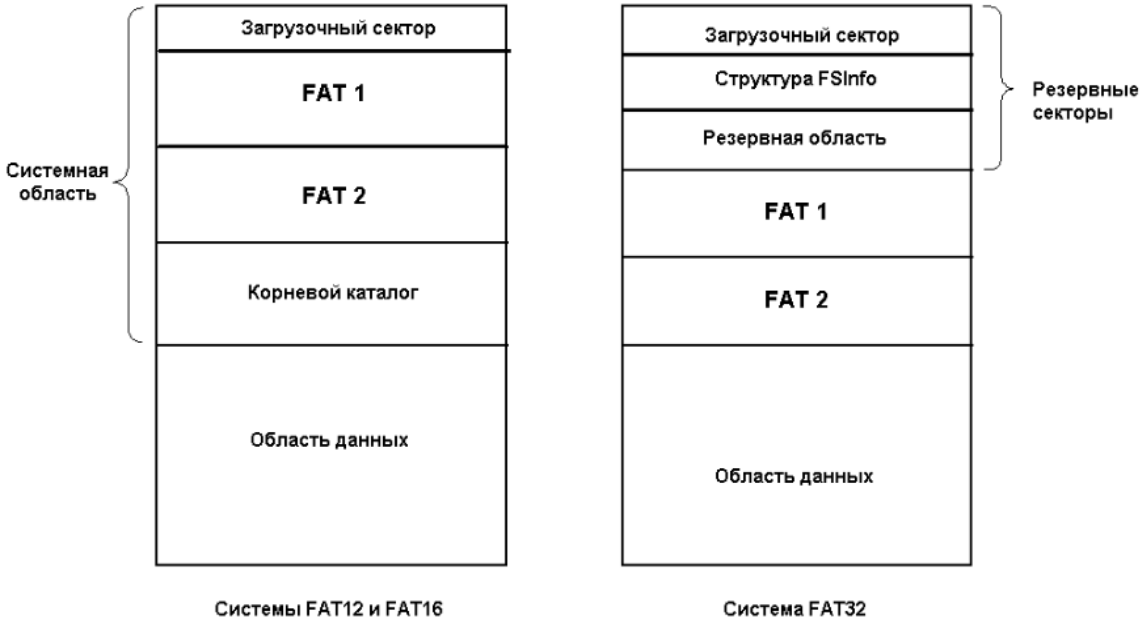


Размещение связанным списком индексов



- Каждому кластеру на физическом носителе соответствует ячейка (индек
  в таблице размещения файлов FAT (*File Allocation Table*).
- FAT создается при форматировании тома, при этом ячейки обнуляются
- При записи файла каждая ячейка содержит указатель на следующий
  кластер, занимаемый файлом.
- Содержимое ячйки:
  - 0 кластер свободен
  - Номер следующего кластера
  - 0x0FFFFFFF (FAT 32) последний кластер файла
  - 0x0FFFFFF7 (для FAT32) кластер поврежден
- Для ускорения поиска таблица загружается в ОП.

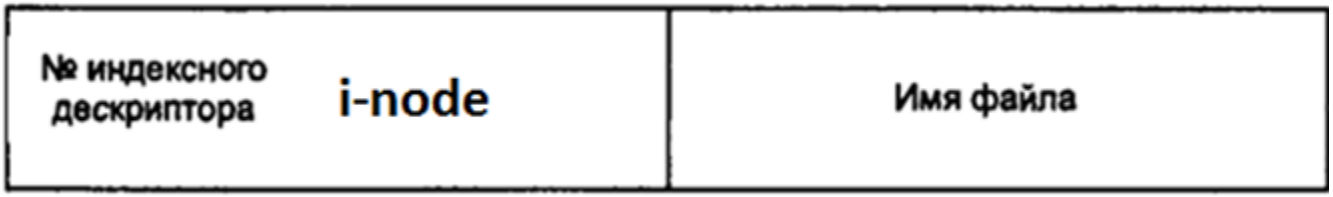
FAT структура раздела



- **Загрузочный сектор** – хранит тип файловой системы, размер кластера и количество кластеров в томе, адрес и размер таблицы FAT и области данных, *адрес первого кластера корневого каталога*
- **FSInfo -** содержит текущую информацию о свободных кластерах
- **Резервная область** – может содержать загрузочную запись раздела PBR, если с раздела загружается ОС.
- Области FAT. Хранят основную и резервную таблицу FAT.

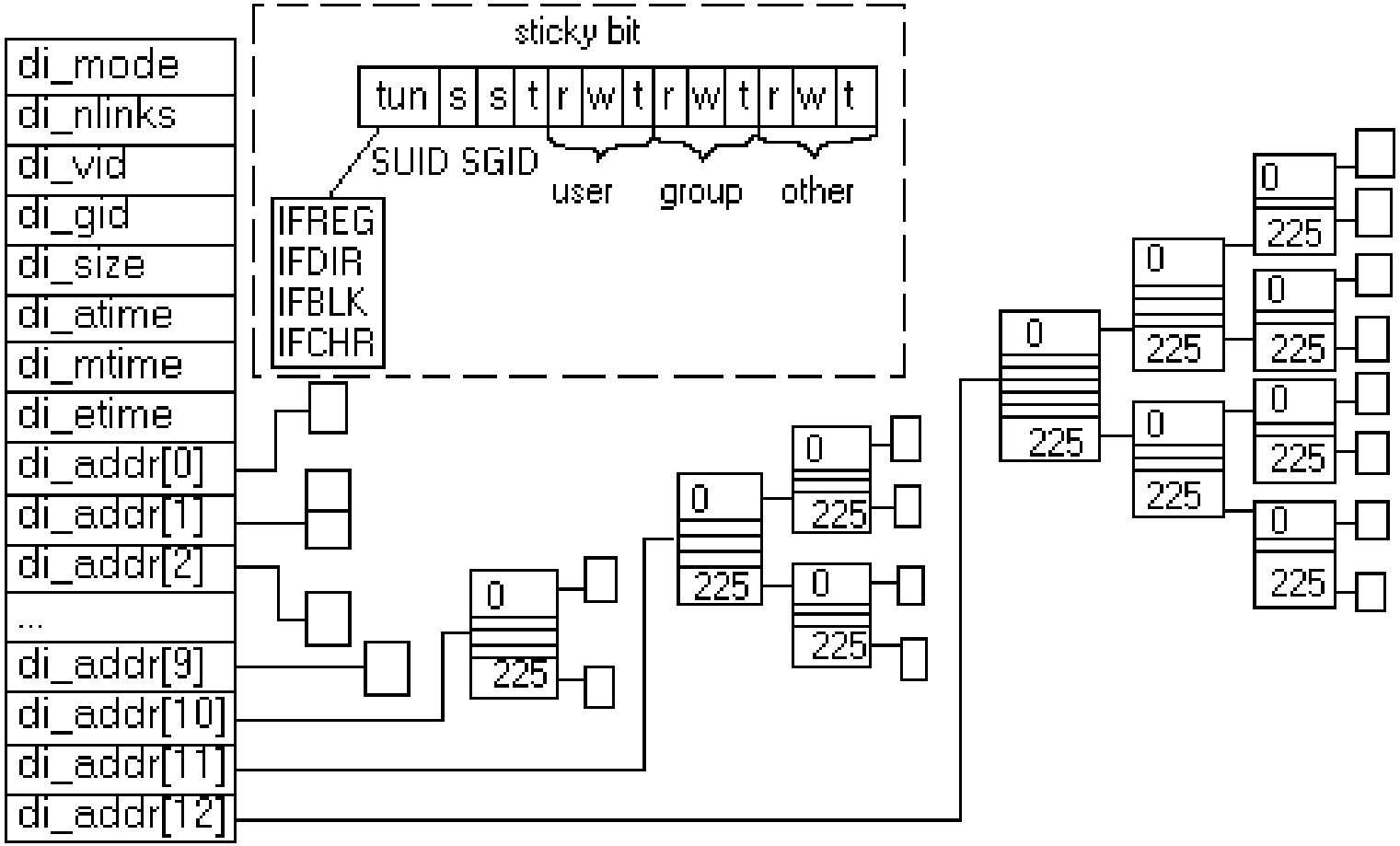
i-node

Файловая запись каталога Linux



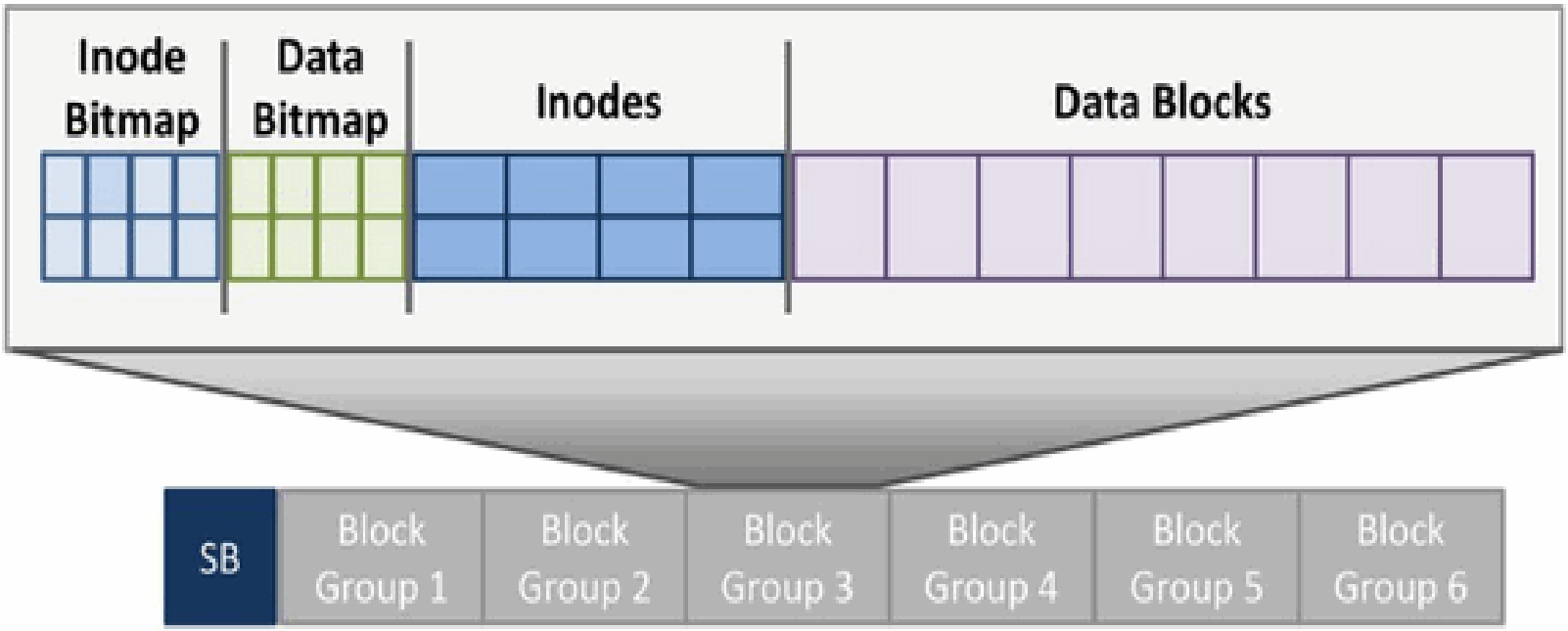
- ***struct dirent*** {
  long d_ino; // номер i-node
  off_t d_off;// смещение данного элемента в реальном каталоге
  unsigned short d_reclen; // длина структуры
  char d_name [1];// // начало массива символов, задающего имя элемента каталога
  };

Структура I-node



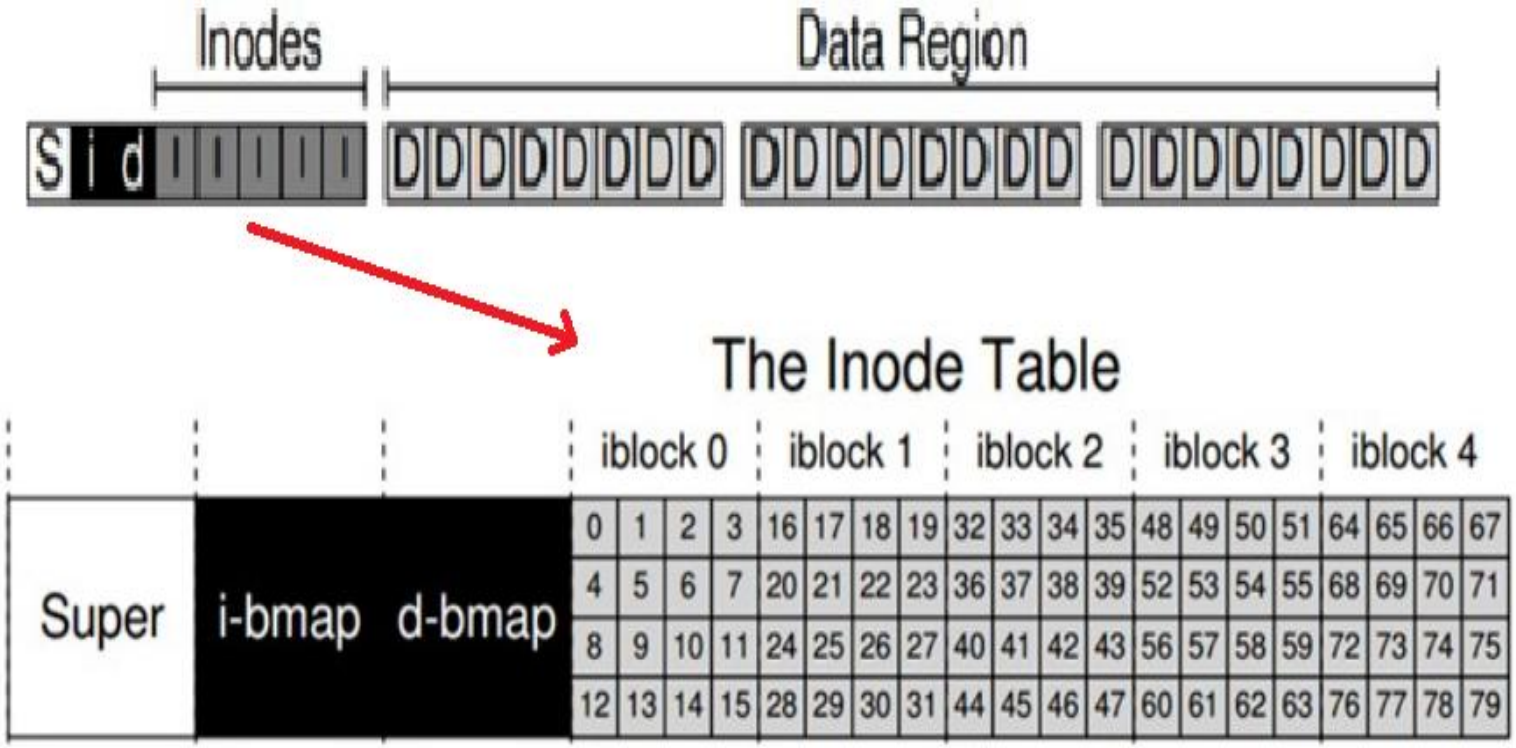
Для ext2 I-node занимает 128 байт

Структура раздела

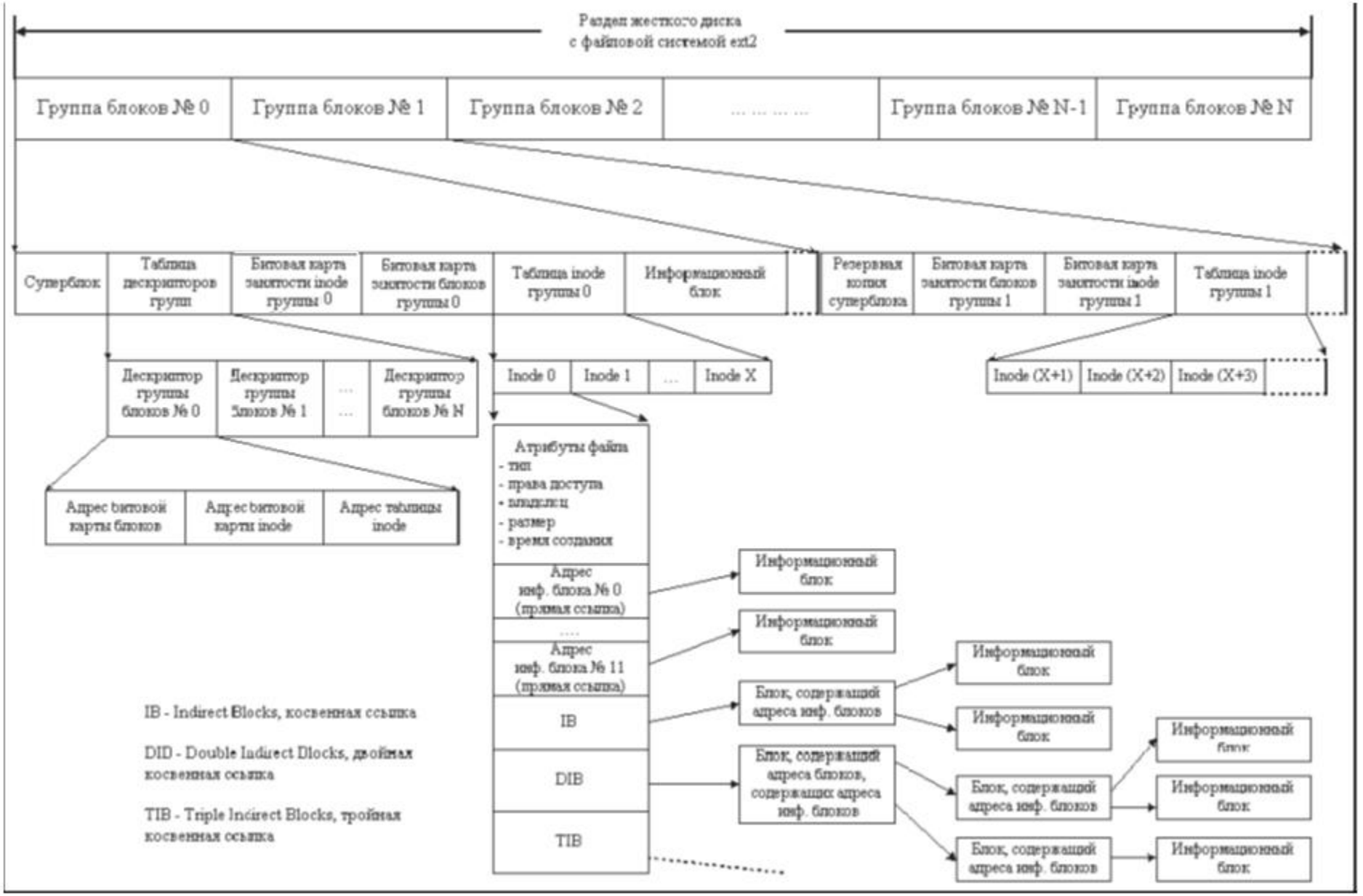


Суперблок содержит общую информацию

Структура группы блоков



Общая уточненная схема структуры раздела



Суперблок

Содержит общую информацию о файловой системе:

- общее число блоков и индексных дескрипторов в файловой системе,
- число свободных блоков и индексных дескрипторов в файловой системе,
- размер блока файловой системы,
- количество блоков и индексных дескрипторов в группе блоков,
- размер индексного дескриптора,
- идентификатор файловой системы (магическое число 0xEF53 для семейства файловых систем ext),
- дата последней проверки файловой системы,
- количество произведённых монтирований.

Этапы чтения файла

Рис. 4. Порядок выполнения процедуры чтения файла в файловой системе ext2 (на примере файла /test.file)

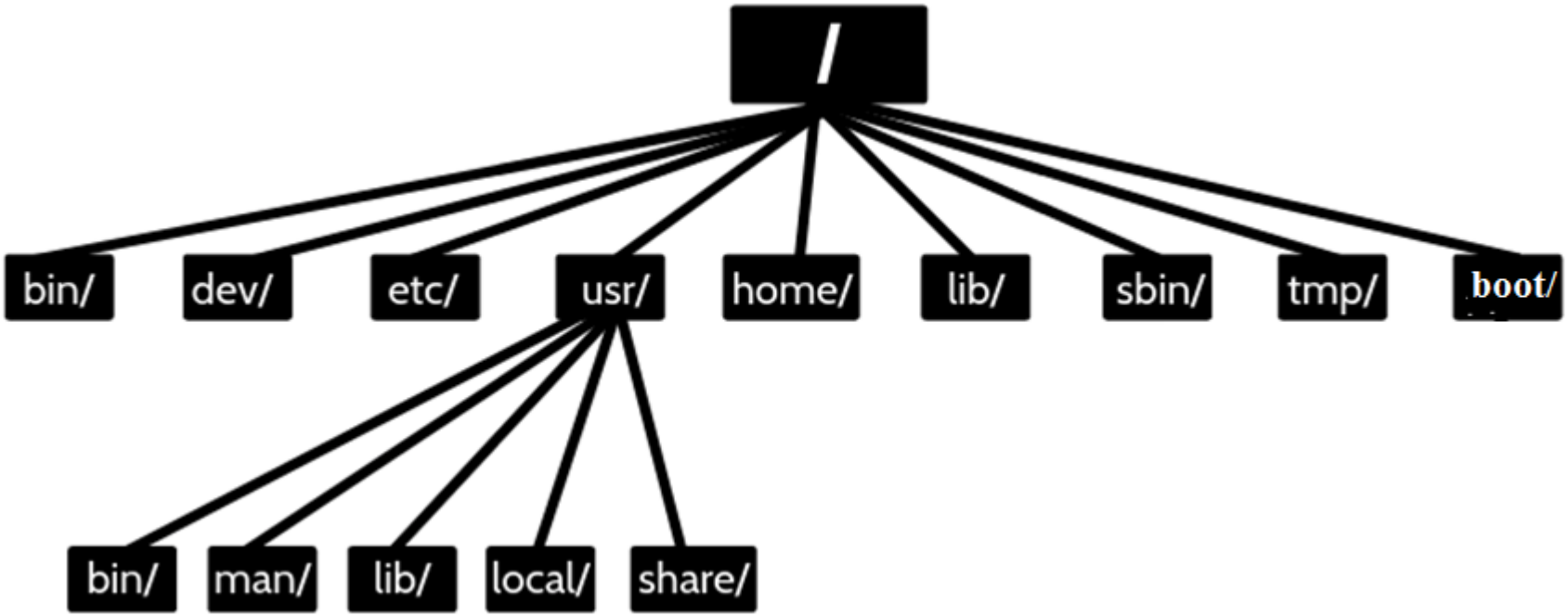
За корневым каталогом закреплен i-node 2 блока 0

stat /

ext4

- максимальный объём одного раздела диска **1 эксбибайт** (2^{60} байт) при размере блока **4 кибибайт**;
- размер одного файла до 16 тебибайт (2^{44} байт);
- механизм протяжённой **(extent)** записи файлов, уменьшает фрагментацию и повышающий производительность
- Максимальное число вложенных каталогов **65 535**
- Размер i-node – **256 байт**
- Журналирование

Структура каталогов



/bin	системные программы общего назначения (ls, dir …)
/sbin	программы системного администрирования (iptables, fdisk)
/boot	файлы загрузчика системы
/home	каталоги и файлы пользователей
/dev	файлы устройств
/etc	файлы настроек
/lib	системные библиотеки и модули ядра
/media	точка монтирования компакт-дисков и flash-карт
/mnt	временные точки монтирования
/proc	виртуальная файловая система, содержащая файлы с информацией о выполняющихся в данный момент процессах и системная информация
/root	домашний каталог суперпользователя
/tmp	временные файлы
/var	каталоги данных, файлы журналов, почтовые ящики, очереди печати и т.д.
/usr	Прикладные программы, исходные коды, документация

Структура каталогов

- **/bin** содержит исполняемые бинарные файлы основных служб, доступных **любому пользователю**, например исполняемые файлы встроенных команд оболочки: ps, ls, ping, grep, cp и саму оболочку bash
- **/sbin** содержит исполняемые бинарные файлы различных служб, доступных **пользователею root** для системного администрирования, например, команда fdisk и др.
- **/boot** содержит файлы, необходимые для загрузки системы - загрузчик **GRUB** и сжатые ядра Linux. Однако файлы конфигурации загрузчика не находятся здесь - они находятся в /etc вместе с другими файлами конфигурации.
- **/dev** содержит специальные файлы, представляющие устройства в виде файлов. Это файлы точки подключения к драйверам. Например, /dev/sda представляет собой первый диск **SATA** в системе.

Структура каталогов

- **/etc** содержит общесистемные файлы конфигурации (например имя хоста), которые обычно можно редактировать вручную в текстовом редакторе.
- - пользовательские файлы конфигурации находятся в домашнем каталоге каждого пользователя.
- **/home** содержит домашнюю папку для каждого пользователя. Каждый пользователь имеет право записи только в свою домашнюю папку и должен получить повышенные права (стать пользователем **root**) для изменения других файлов в системе.
- **/lib** содержит системные библиотеки, необходимые для основных двоичных файлов в папке /bin и /sbin и подгружаемые модули ядра. Библиотеки, необходимые для двоичных файлов в папке /usr/bin, находятся в /usr/lib. Имена файлов библиотеки: ld* или lib*.so.*.
- **lib64** содержит библиотеки, код которых специфичен для версии(разрядности) процессора.

Структура каталогов

- **/proc** содержит информацию о выполняющихся в данный момент процессах и системе. Этого каталога физически на диске нет, является виртуальной файловой системой, информация в нем генерируются ядром автоматически и обновляются на лету .
- **/run** предоставляет приложениям стандартное место для хранения необходимых им временных файлов, таких как сокеты и идентификаторы процессов. Эти файлы нельзя хранить в /tmp, потому что файлы в /tmp могут быть удалены.
- **/usr** (unix system resources) содержит дополнительные исполняемые файлы, библиотеки заголовочные файлы и файлы документации (man) для внутренних служб, а также данные программ, установленные пользователями.
- **sys** - виртуальная файловая система, содержит данные ядра операционной системы и его модулей. Этот каталог перезаписывается после каждой перезагрузки операционной системы.

Структура каталогов

- **/media** содержит подкаталоги, в которые **ОС** автоматически монтирует съемные носители.
- **/mnt** – аналогична /media это место, куда **пользователи** монтируют временные файловые системы во время их использования.
- **/opt** содержит подкаталоги для дополнительных пакетов программного обеспечения. Он обычно используется проприетарным программным обеспечением, которое не подчиняется стандартной иерархии файловой системы - например, проприетарная программа может выгружать свои файлы в /opt/application при ее установке.

/proc

- /proc/cpuinfo - информация о процессоре (модель, семейство, размер кэша и т.д.)
- /proc/meminfo - информация о RAM, размере свопа и т.д.
- /proc/mounts - список подмонтированных файловых систем.
- /proc/devices - список устройств.
- /proc/filesystems - поддерживаемые файловые системы.
- /proc/modules - список загружаемых модулей.
- /proc/version - версия ядра.

Файлы устройств

- В Linux при подключении тома (раздела) для него в каталоге **/dev** создается **файл с именем устройства**, который является точкой подключения к драйверу устройства
- Основной характеристикой файлов устройств является пара чисел **major, minor** (иногда называемых характеристическими числами), привязывающая их к конкретному драйверу и управляемому им устройству.
- **major – номер драйвера**
- **minor – номер устройства**

```
root@ubuntu:/# cd /dev
root@ubuntu:/dev# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0    20G  0 disk
├─sda1   8:1    0    18G  0 part /
├─sda2   8:2    0     1K  0 part
└─sda5   8:5    0     2G  0 part [SWAP]
sdc      8:32   1   974M  0 disk
└─sdc1   8:33   1 973,9M  0 part /media/myflash
sr0     11:0    1  1024M  0 rom
```

Устройства в Linux

- Имя устройства имеет формат:
  /dev/xxYN, где:
  - **/dev** —каталог, в котором расположены файлы, связанные с устройствами;
  - **xx** - тип устройства (две буквы), на котором размещается раздел.
    - **sd** – SATA/SCSI – устройства;
    - **hd** – IDE устройства;
  - **Y** - номер устройства
    - a – первое устройство, b – второе и.т.д
  - **N** - обозначает раздел.
    - Первичные (основные) разделы нумеруются числами с 1 по 4. Расширенные (логических) разделы начинается с 5, даже если первичных разделов меньше четырех.
    - Например, /dev/sda2 второй основной раздел на первом диске

Монтирование файловой системы

- Идея монтирования : представить имя раздела в виде каталога монтирования в общем дереве каталогов Linux и получит доступ к разделу.
- Уточнить имя подключаемого раздела: ***fdisk –l, lsblk. blkid***
- Создать каталог монтирования

```
# mkdir /dev/media/myflash
```

- Подключить (смонтировать) раздел к каталогу монтирования:

```
# mount /dev/sdc1 /media/myflash -o utf8
```

  - -o utf8 позволяет русифицировать имена файлов

- Проверить список смонтированных разделов:

```
# mount
```

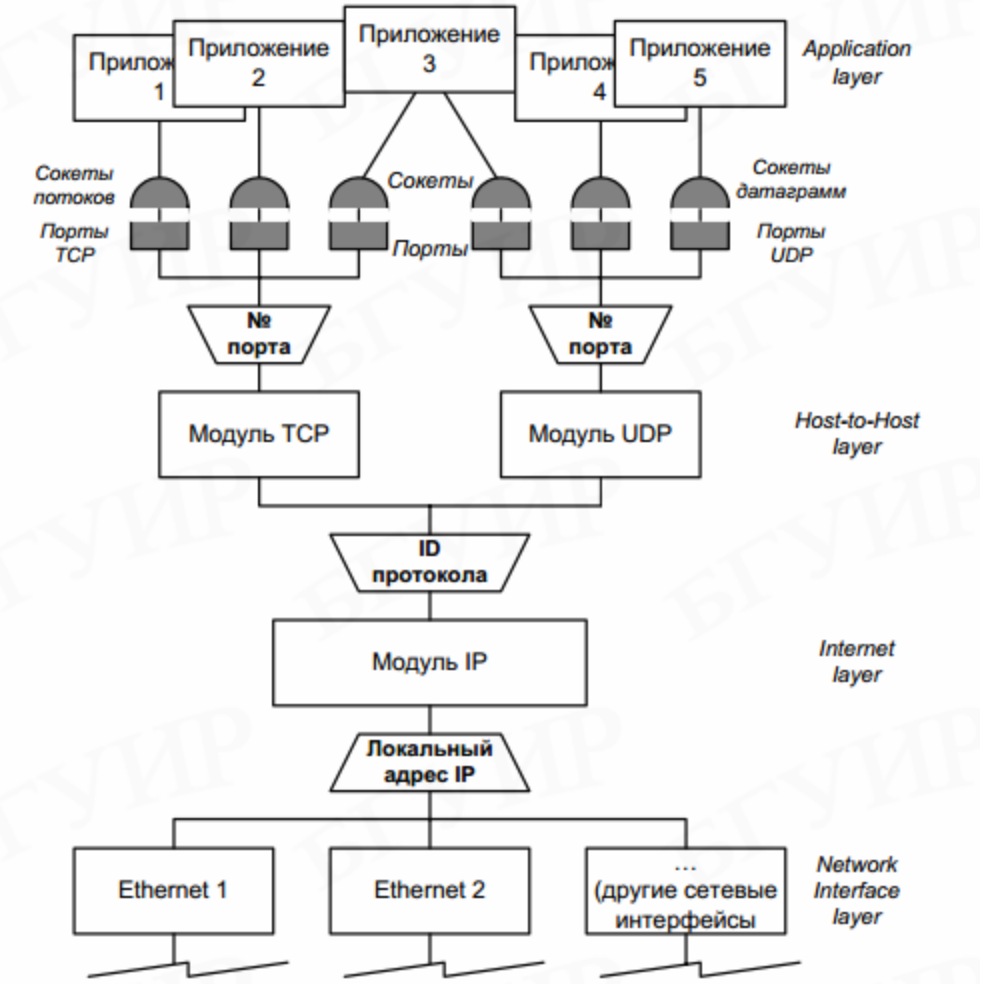
- При необходимости размонтировать:

```
# umount /media/myflash
```

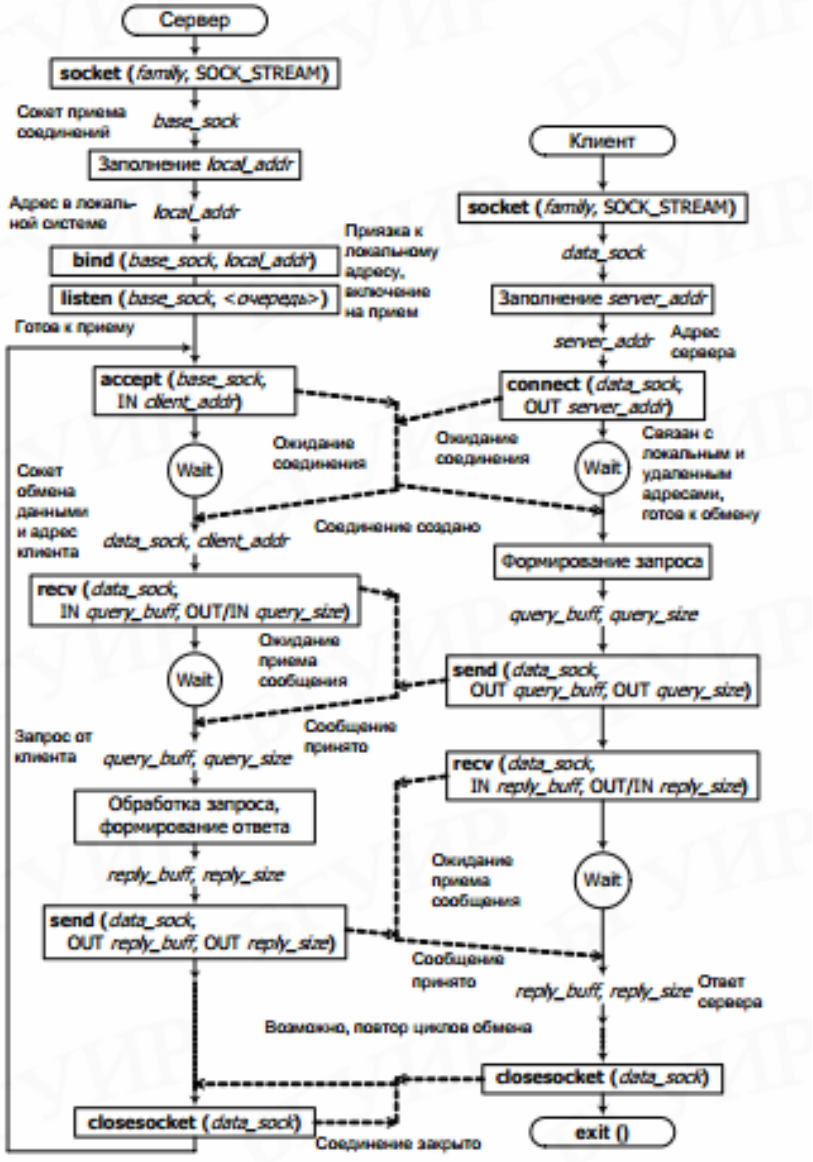
Типы файлов в Linux

Типы файлов		Назначение
Обычные файлы	-	Хранение символьных и двоичных данных
Каталоги	d	Организация доступа к файлам
Символьные ссылки	l	Предоставление доступа к файлам, расположенных на любых носителях
Блочные устройства	b	Предоставление интерфейса для взаимодействия с аппаратным обеспечением компьютера
Символьные устройства	c	
Каналы	p	Организация взаимодействия процессов в операционной системе
Сокеты	s	

Socket


Приложение 1
Приложение 2
Приложение 3
Приложение 4
Приложение 5
Application layer
Сокеты потоков
Порты TCP
Сокеты
Порты
Сокеты датаграмм
Порты UDP
№ порта
№ порта
Модуль TCP
Модуль UDP
Host-to-Host layer
ID протокола
Модуль IP
Internet layer
Локальный адрес IP
Ethernet 1
Ethernet 2
… (другие сетевые интерфейсы
Network Interface layer

UDP


Сервер
socket (family, SOCK_STREAM)
Сокет приема соединений
base_sock
Заполнение local_addr
Адрес в локальной системе
local_addr
bind (base_sock, local_addr)
Привязка к локальному адресу, включение на прием
listen (base_sock, <очередь>)
Готов к приему
accept (base_sock, IN client_addr)
Wait
Ожидание соединения
Сокет обмена данными и адрес клиента
data_sock, client_addr
Соединение создано
recv (data_sock, IN query_buff, OUT/IN query_size)
Wait
Ожидание приема сообщения
Запрос от клиента
query_buff, query_size
Сообщение принято
Обработка запроса, формирование ответа
reply_buff, reply_size
send (data_sock, OUT reply_buff, OUT reply_size)
Сообщение принято
Возможно, повтор циклов обмена
closesocket (data_sock)
Соединение закрыто
Клиент
socket (family, SOCK_STREAM)
data_sock
Заполнение server_addr
server_addr
Адрес сервера
connect (data_sock, OUT server_addr)
Ожидание соединения
Wait
Связан с локальным и удаленным адресами, готов к обмену
Формирование запроса
query_buff, query_size
send (data_sock, OUT query_buff, OUT query_size)
recv (data_sock, IN reply_buff, OUT/IN reply_size)
Ожидание приема сообщения
Wait
reply_buff, reply_size
Ответ сервера
closesocket (data_sock)
exit ()

Атрибуты доступа

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Тип файла				SUID	SGID	T-бит	права владельца			права группы			права всех остальных		

chmod [uga] [+-=] [rwx] имя_файла
chmod 775 file_name

Атрибуты доступа

Двоичный код	Восьмеричный код	Права доступа к файлу
100	4	чтение (r)
010	2	запись (w)
001	1	выполнение (x)
110	6	чтение и запись (rw)
101	5	чтение и выполнение (rx)
011	3	запись и выполнение (wx)
111	7	чтение, запись и выполнение (rwx)
000	0	нет доступа

chmod 775 file_name

chown владелец имя_файла

Дополнительные биты доступа

t	Sticky bit
s	Set UID, SUID
s	Set GID, SGID
l	Блокирование

- Sticky bit - может быть назначен на файл или каталог
- Запрещает удаление файла или каталога всем, кроме владельца
- -T - права на выполнение отключены.
- - t - права на выполнение включены.
- l - установить обязательное блокирование файлов при выполнении
  - используется для устранения конфликтов, когда одновременно несколько задач работают с одним и тем же файлом
- **S** - разрешает запускать исполняемые файлы от имени другого пользователя (группы), не владельца файла.

Права на файлы и каталоги

Права	Применимо к файлам	Применимо к каталогам
Read	Прочитать содержимое файла	Отобразить список содержимого каталога
Write	Изменить содержимое файла	Разрешить создавать, удалять или задавать права на файлы
Execute	Запустить файл как программу	Переход в каталог

Маски для проверки прав доступа

S_IFMT	0170000	битовая маска для битовых полей типа файла
S_IFSOCK	0140000	socket
S_IFLNK	0120000	символическая ссылка
S_IFREG	0100000	обычный файл
S_IFBLK	0060000	блокировать устройство
S_IFDIR	0040000	каталог
S_IFCHR	0020000	устройство персонажа
S_IFIFO	0010000	FIFO
S_ISUID	0004000	установить бит UID
S_ISGID	0002000	бит set-group-ID (см. ниже)
S_ISVTX	0001000	sticky бит
S_IRWXU	00700	маска для разрешений владельца файла
S_IRUSR	00400	владелец имеет разрешение на чтение
S_IWUSR	00200	владелец имеет разрешение на запись
S_IXUSR	00100	владелец имеет разрешение на выполнение
S_IRWXG	00070	маска для разрешений группы
S_IRGRP	00040	группа имеет разрешение на чтение
S_IWGRP	00020	у группы есть разрешение на запись
S_IXGRP	00010	группа имеет разрешение на выполнение
S_IRWXO	00007	маска разрешений для других (не в группе)
S_IROTH	00004	у других есть разрешение на чтение
S_IWOTH	00002	у других есть разрешение на запись
S_IXOTH	00001	у других есть разрешение на выполнение

Позволяют получить
для проверки
определенные биты
прав доступа

Макросы определяющие тип файла

S_ISREG(m)
is it a regular file?

S_ISDIR(m)
directory?

S_ISCHR(m)
character device?

S_ISBLK(m)
block device?

S_ISFIFO(m)
FIFO (named pipe)?

S_ISLNK(m)
symbolic link

S_ISSOCK(m)
socket?

Макроконстанты определяющие тип файла

DT_BLK
This is a block device.

DT_CHR
This is a character device.

DT_DIR
This is a directory.

DT_FIFO
This is a named pipe (FIFO).

DT_LNK
This is a symbolic link.

DT_REG
This is a regular file.

DT_SOCK
This is a Unix domain socket.

DT_UNKNOWN
The file type is unknown.

DIR – хранит информацию о каталоге

```
struct __dirstream  {
    void *__fd;    // файловый дескриптор каталога
    char *__data;
    int __entry_data;
    char *__ptr;
    int __entry_ptr;
    size_t __allocation;
    size_t __size;
    __libc_lock_define (, __lock)
   };
  typedef struct __dirstream  DIR;
```

- Функция ***DIR * opendir (const char * pathname)***
  - возвращает указатель на структуру DIR, для заданного каталога который используется в функциях:
  - ***readdir*** (чтение записей из директории),
  - ***closedir*** (закрытие директории)
  - ***rewinddir*** (перемещение позиции считывания к началу директории)

Dirent – хранит информацию о файле в каталоге

```
struct dirent {
    ino_t          d_ino;       /* inode number */
    off_t          d_off;       /* offset to the next dirent */
    unsigned short d_reclen;    /* length of this record */
    unsigned char  d_type;      /* type of file; not supported
                                   by all file system types */
    char           d_name[256]; /* filename */
};
```

STAT – содержит текущее состояние файла

- ***struct stat {***
- ***dev_t st_dev;*** ***/*** логическое устройство, где находится файл ****/***
- ***ino_t st_ino;*** ***/*** номер индексного дескриптора ****/***
- ***mode_t st_mode;*** ***/*** права доступа к файлу ****/***
- ***nlink_t st_nlink;*** ***/*** количество жестких ссылок на файл ****/***
- ***uid_t st_uid;*** ***/*** ID пользователя-владельца ****/***
- ***gid_t st_gid;*** ***/*** ID группы-владельца ****/***
- ***dev_t st_rdev;*** ***/*** тип устройства ****/***
- ***off_t st_size;*** ***/*** общий размер в байтах ****/***
- ***unsigned long st_blksize;*** ***/*** размер блока ввода-вывода ****/***
- ***unsigned long st_blocks;*** ***/*** число блоков, занимаемых файлом ****/***
- ***time_t st_atime;*** ***/*** время последнего доступа ****/***
- ***time_t st_mtime;*** ***/*** время последней модификации ****/***
- ***time_t st_ctime;*** ***/*** время последнего изменения ****/***
- ***};***