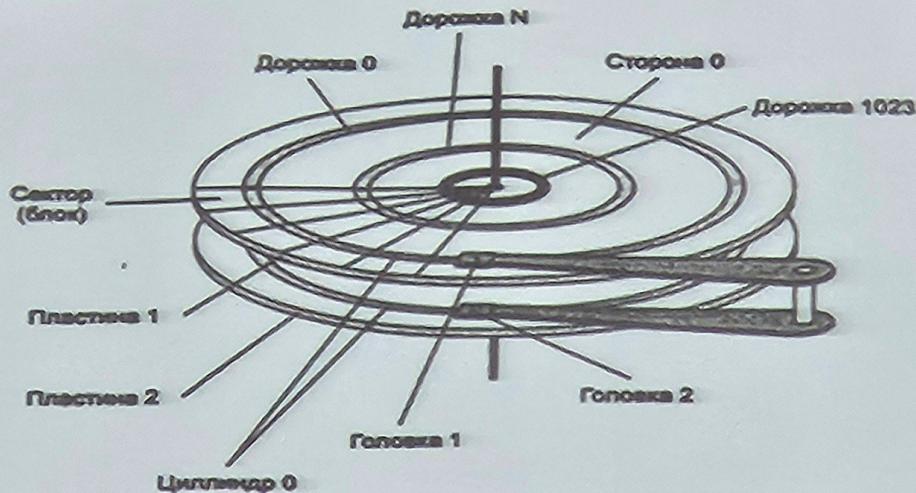


Файловая система

- *Файловая система* – множество файлов организованных по определенным правилам на внешнем носителе (FAT, ext, 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*.

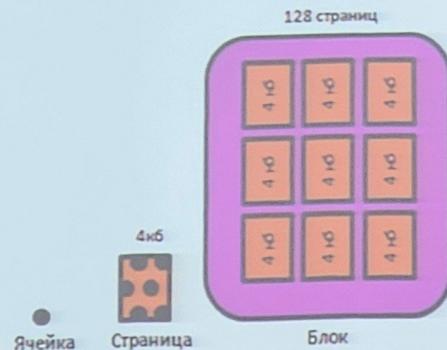
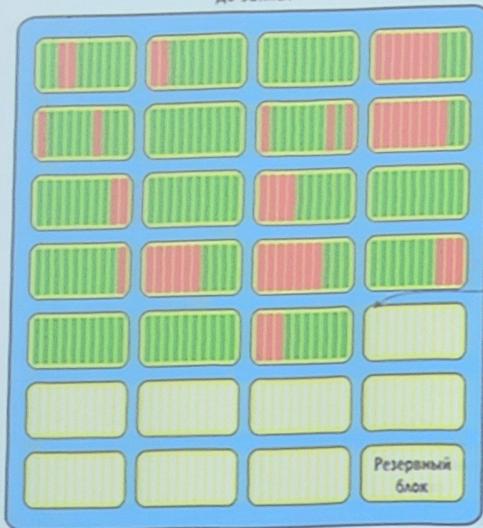


Заголовок сектора	Данные	ECC
-------------------	--------	-----

- 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

SSD-диск

До записи



- Ячейки группируются в **физические страницы** размером, кратным 512 байт.
- Каждая страница имеет свой **физический адрес РВА**.
- 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**,

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 логически блоков (кластеров) и создания таблиц кластеров, а также структур данных под конкретный тип файловой системы называется **высокоуровневым форматированием**

Кластер/сектор

```
aktios@ubuntu:~$ stat hello.c
```

Файл: «hello.c»

Размер: 93

Блоков: 8

Блок В/В: 4096

обычный файл

Устройство: 801h/2049d Inode: 286768

Ссылки: 1

доступ: (0664/-rw-rw-r--)Uid: (1000/ aktios) Gid: (1000/ aktios)

доступ: 2024-11-17 07:44:49.709030445 -0800

Модифицирован: 2023-12-07 00:43:55.219663777 -0800

Изменён: 2023-12-07 00:43:55.219663777 -0800

Создан:

```
aktios@ubuntu:~$ stat --format=%B hello.c
```

512

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

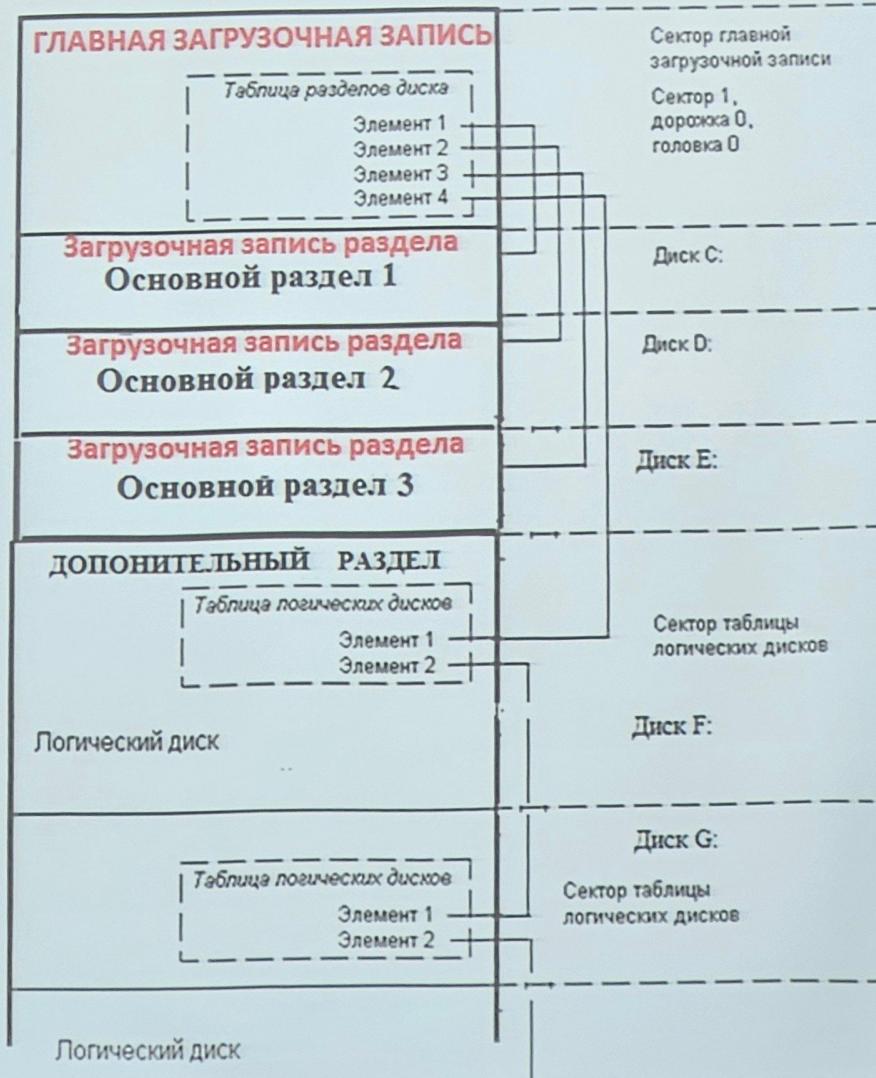
MBR – главная загрузочная запись (первичный загрузчик)

Создается при разбиении диска на разделы и модифицируется при форматировании и установке ОС.

PBR (Partition Boot Record) – загрузочная запись раздела

Создается при разбиении и заполняется при установке в данном разделе ОС.

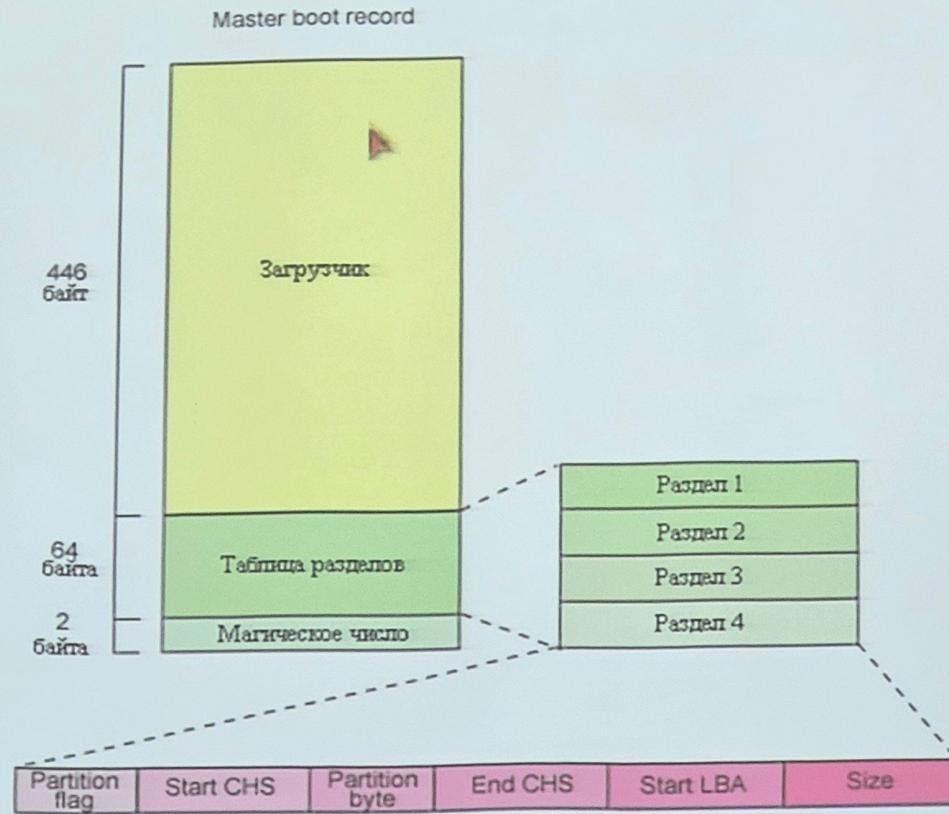
Сюда записывается вторичный загрузчик ОС.



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

MBR

Первый сектор на диске, который использует таблицу разделов MBR. Имеет размер 512 байт. Хранит загрузчик и таблицу разделов. Этот загрузчик способен запустить более функциональный загрузчик

Таблица разделов, активный раздел

PBR
GRUB 2

Более функциональный загрузчик, который способен загрузить ядро системы (Linux или Windows). Имеет собственную командную строку и множество настроек

/boot/vmlinuz boot/initrd.img

Kernel

Ядро Linux. Оно запускается не как процесс. После загрузки оно запускает систему инициализации как первый процесс в системе

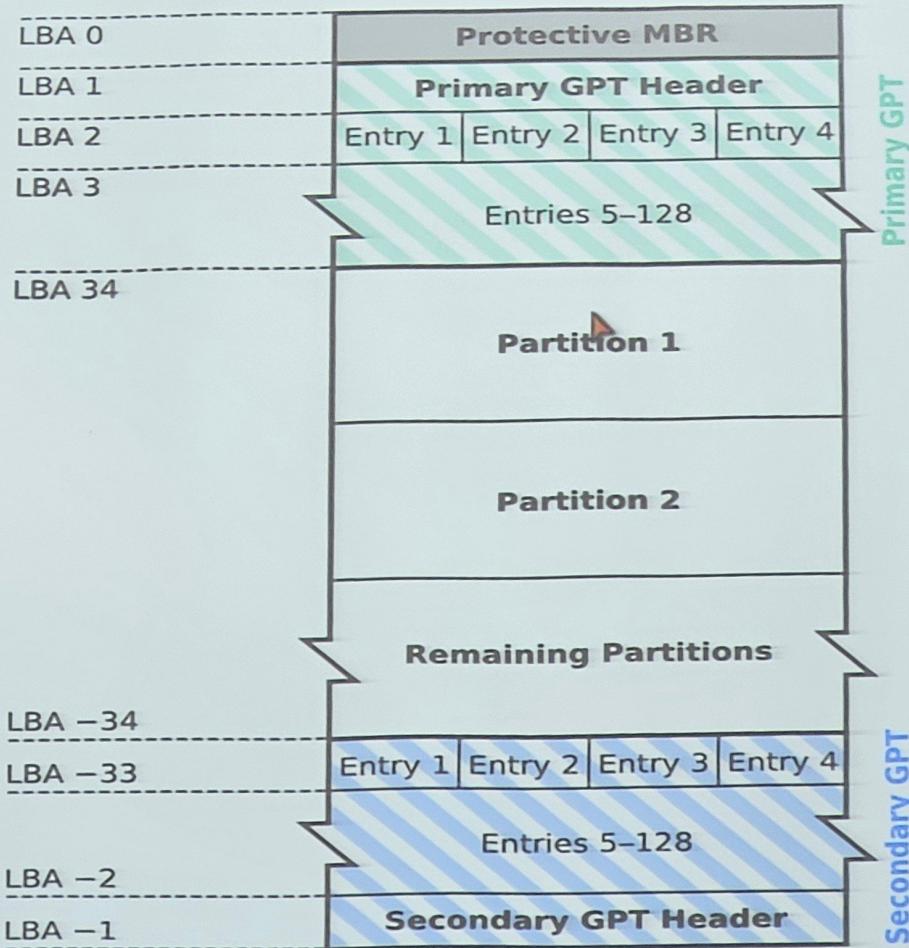
init

Init

Система инициализации. Это первый процесс в системе, именно он запускает все остальные процессы

GUID Partition Table Scheme

Формат GPT

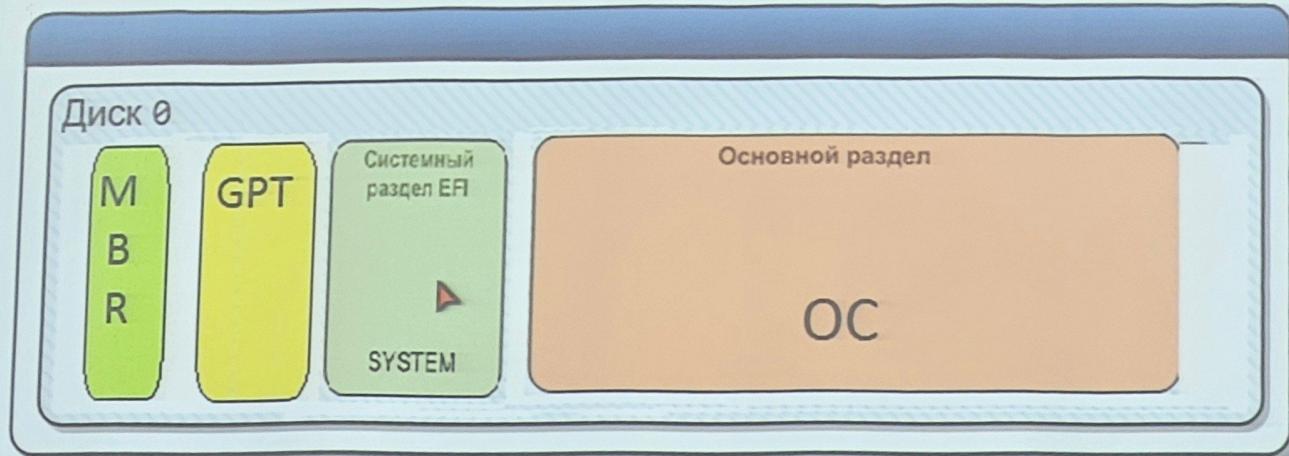


Допускает неограниченное количество разделов.

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.

Загрузка Linux UEFI

UEFI

Программа находится в памяти на материнской плате, позволяет настраивать некоторые особенности работы компьютера. А для загрузки системы использует специальный загрузчик в формате .efi

GPT

ESP / EFI
(fat 32)

Специальный раздел на диске, на котором может находиться загрузчик `bootx64.efi`, `grub64.efi`, `shimx64.efi` или другой. И при необходимости эти загрузчики могут передавать управление друг другу

`/boot/vmlinuz` `/boot/initrd.img`

Kernel

Ядро linux, это единственная программа в linux, которая запускается не как процесс. И ядро запускает первый процесс в системе. Это процесс системы инициализации (`init`)

Init

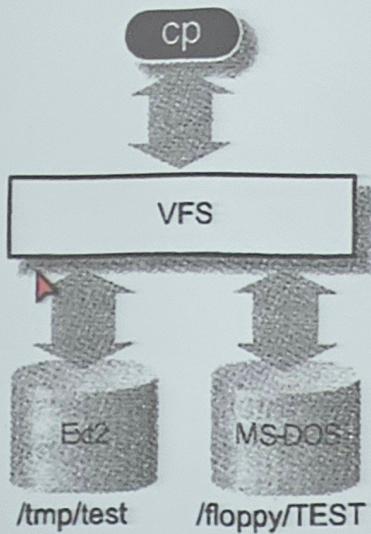
Первый процесс в системе, именно он запускает все остальные процессы

Ядро Linux



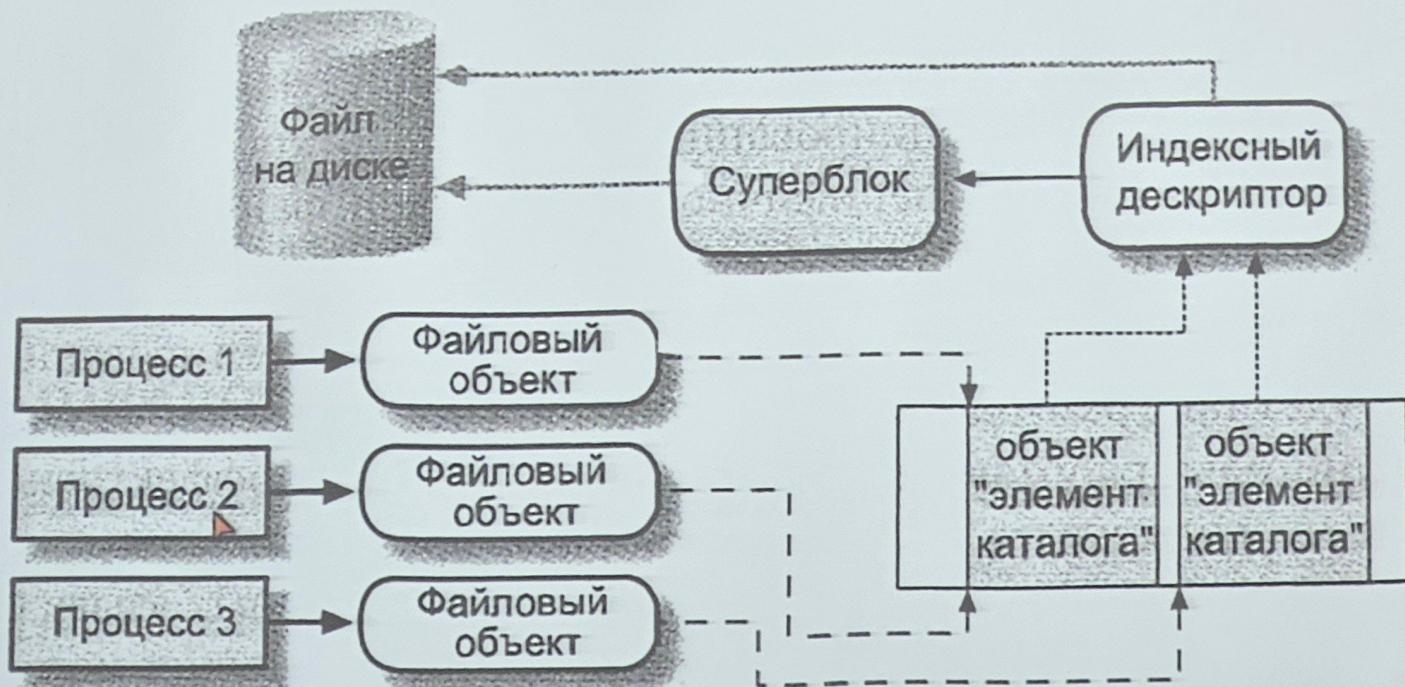
Виртуальный переключатель файловых систем (Virtual Filesystem Switch, VFS) предоставляет общий интерфейса к нескольким видам файловых систем.

Виртуальный переключатель файловых систем (Virtual Filesystem Switch, VFS)

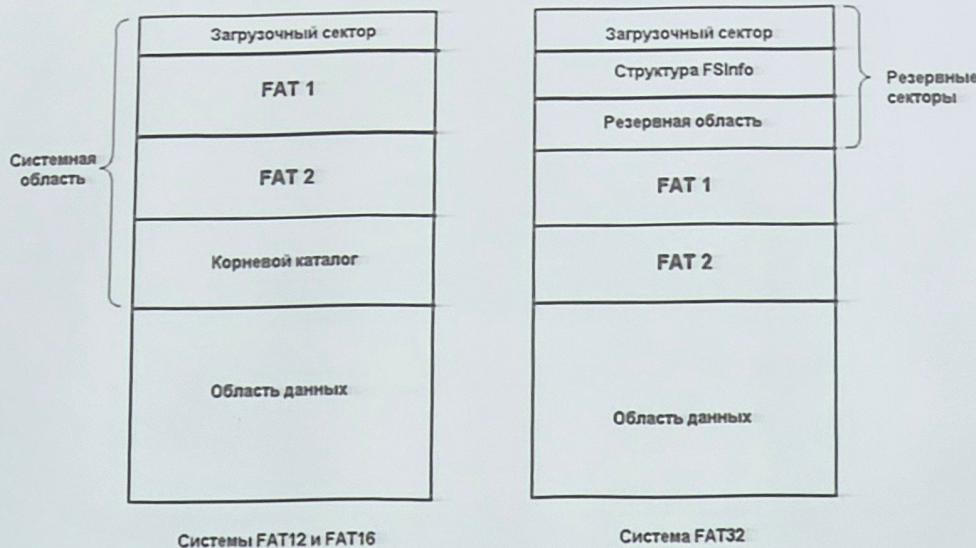


```
inf = open("/floppy/TEST", O_RDONLY, 0);
outf = open("/tmp/test",
            O_WRONLY|O_CREAT|O_TRUNC, 0600);
do {
    i = read(inf, buf, 4096);
    write(outf, buf, i);
} while (i);
close(outf);
close(inf);
```

Общая модель VFS



FAT структура раздела (тома)

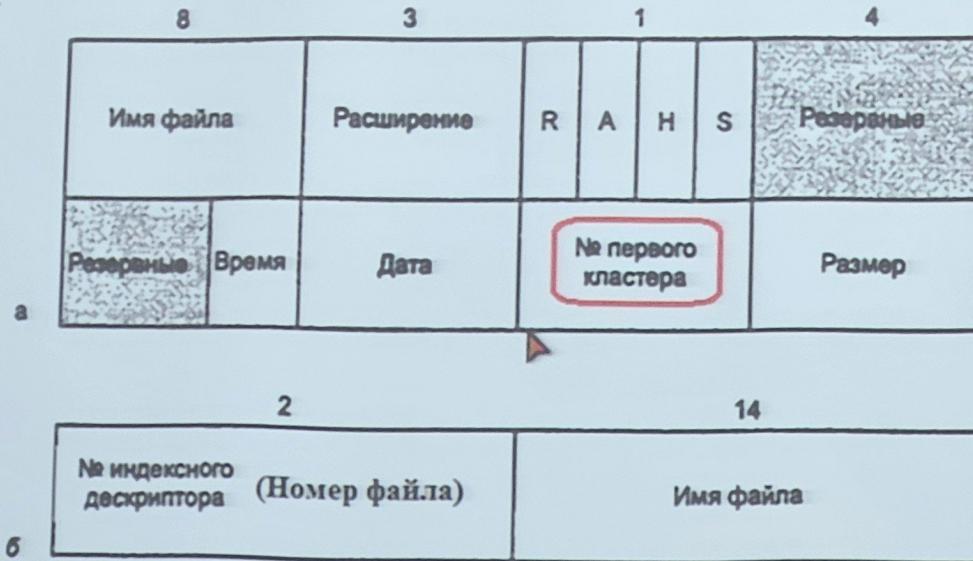


- **Загрузочный сектор** – хранит тип файловой системы, размер кластера и количество кластеров в томе, адрес и размер таблицы FAT и области данных, *адрес первого кластера корневого каталога*
- **FSInfo** - содержит текущую информацию о свободных кластерах
- **Резервная область** – может содержать загрузочную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

Запись о файле в каталоге

Запись (атрибуты) о файле может храниться :

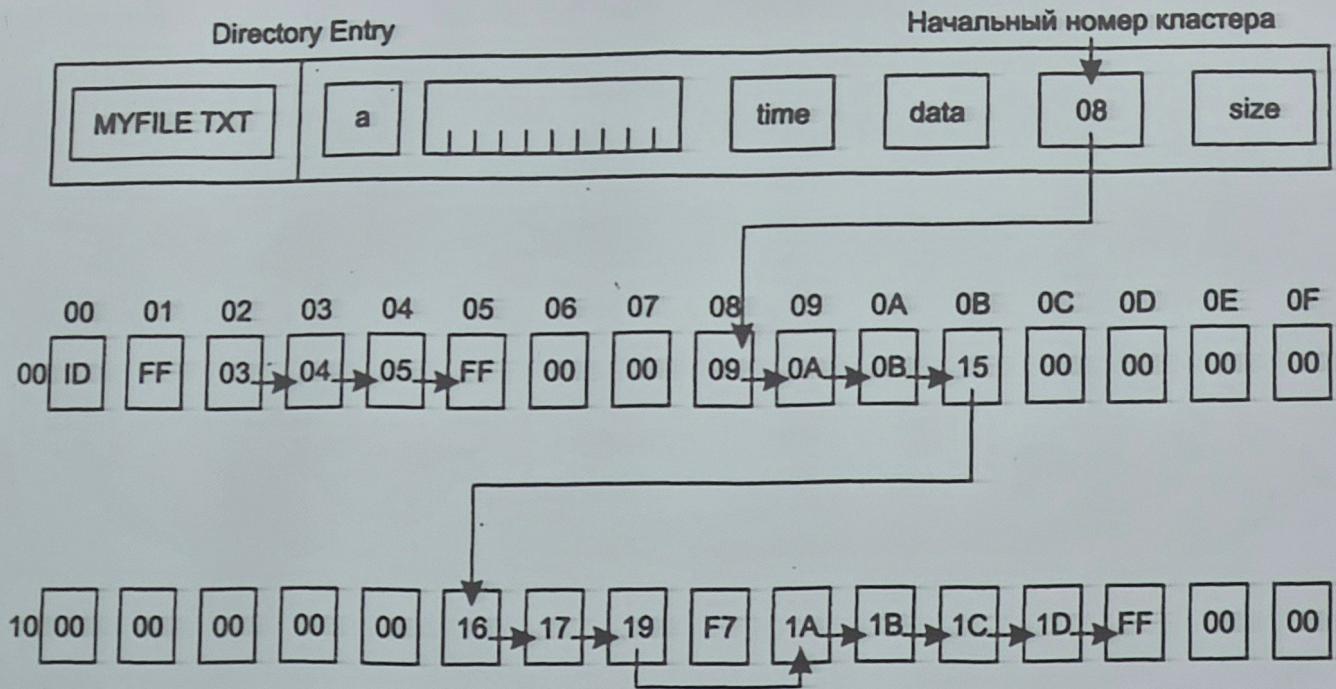
- Полностью в каталоге;
- В каталоге хранится только имя файла и ссылка на блок в котором хранятся все остальные атрибуты файла.



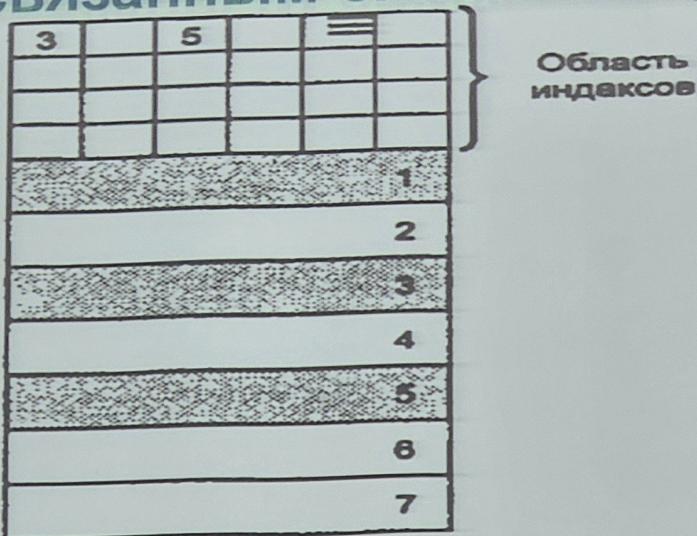
Структура каталогов: а — структура записи каталога MS-DOS FAT 32 (32 байта),
б — структура записи каталога OC UNIX

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

Размещение связанными списками кластеров



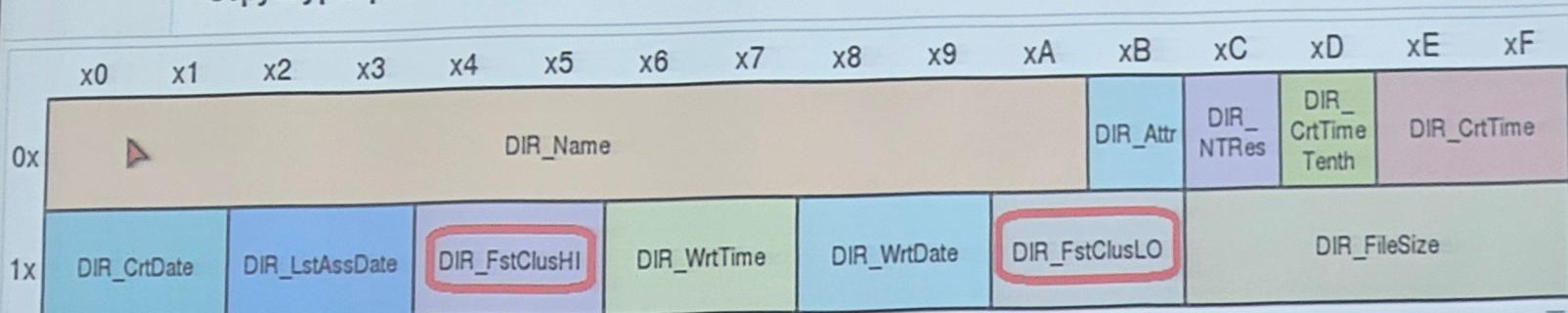
Размещение связанным списком индексов



- Каждому кластеру на физическом носителе соответствует ячейка (индекс) 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 ячейки:
 - 0 кластер свободен
 - Номер следующего кластера
 - 0xFFFFFFFF (FAT 32) последний кластер файла
 - 0xFFFFFFFF7 (для FAT32) кластер поврежден
- Для ускорения поиска таблица загружается в ОП.

Уточненная структура файловой записи для FAT

Структура файловой записи



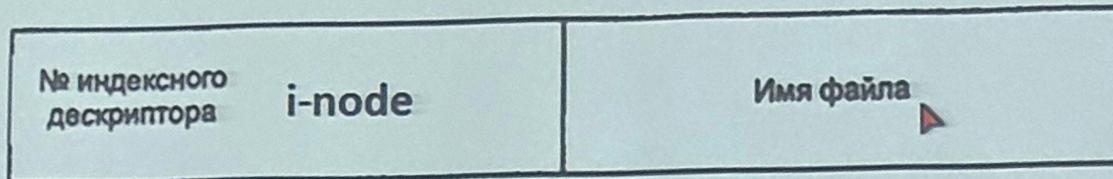
DIR_FstClusHI. 2 байта по адресу 0x14. Номер первого кластера файла (старшее слово) в FAT12/FAT16 равен нулю)

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

DIR_FileSize. размера файла в байтах.

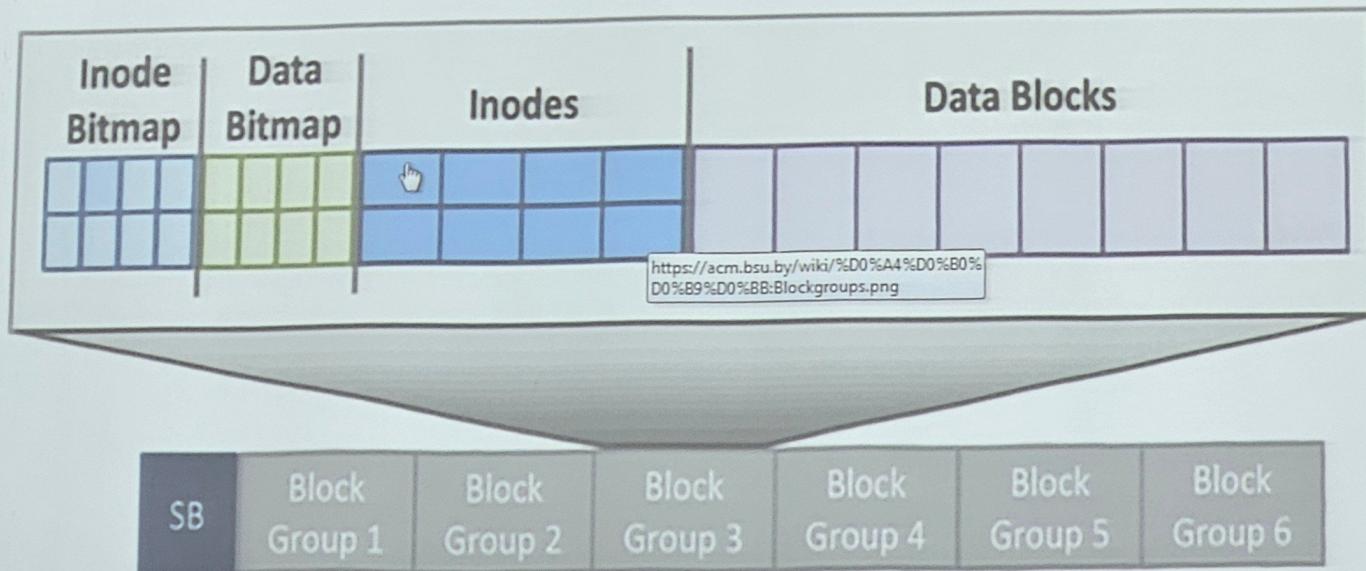
Ограничение FAT32 — максимальный размер файла (4 байта) составляет 0xFFFFFFFF = 4ГБ

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
};`

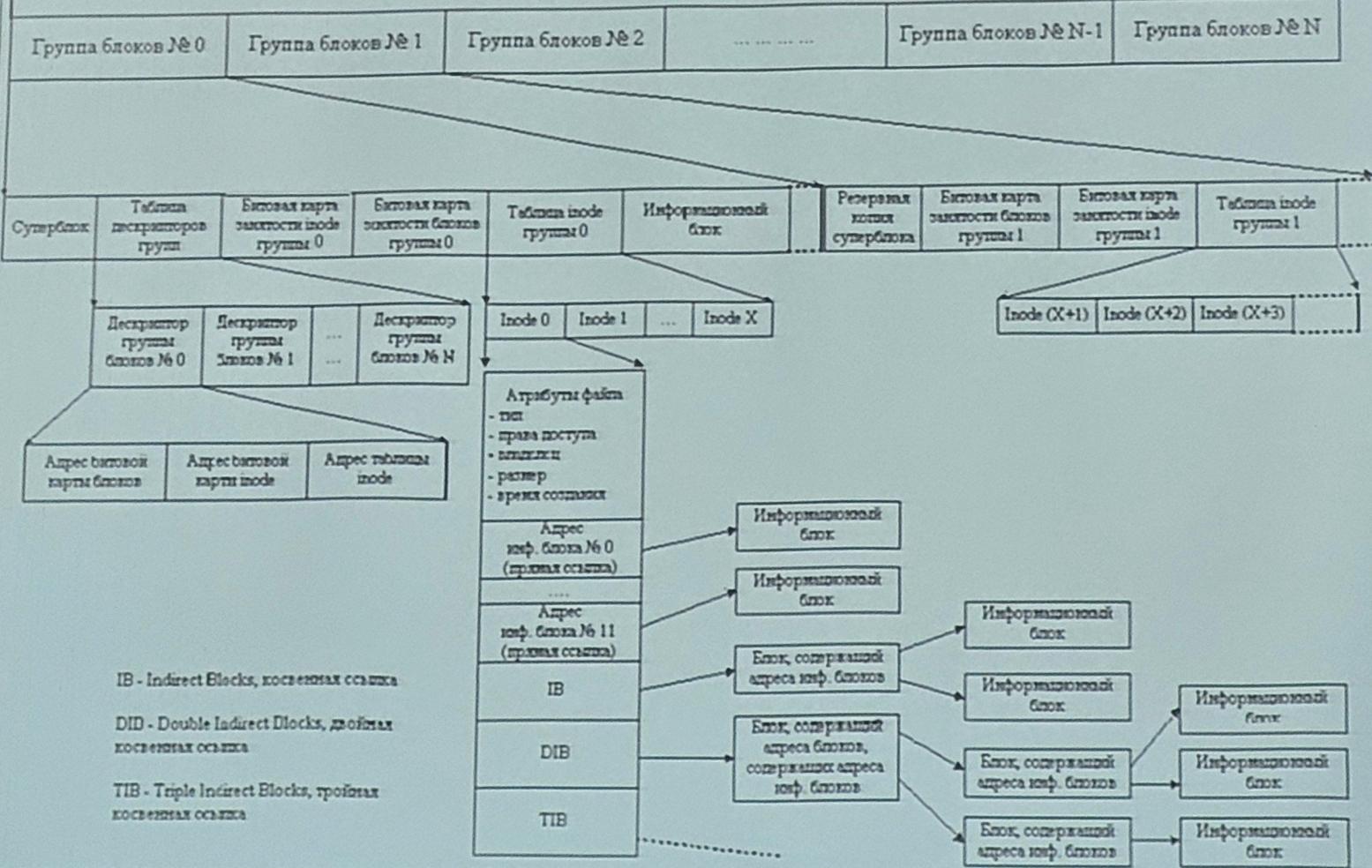
Структура раздела



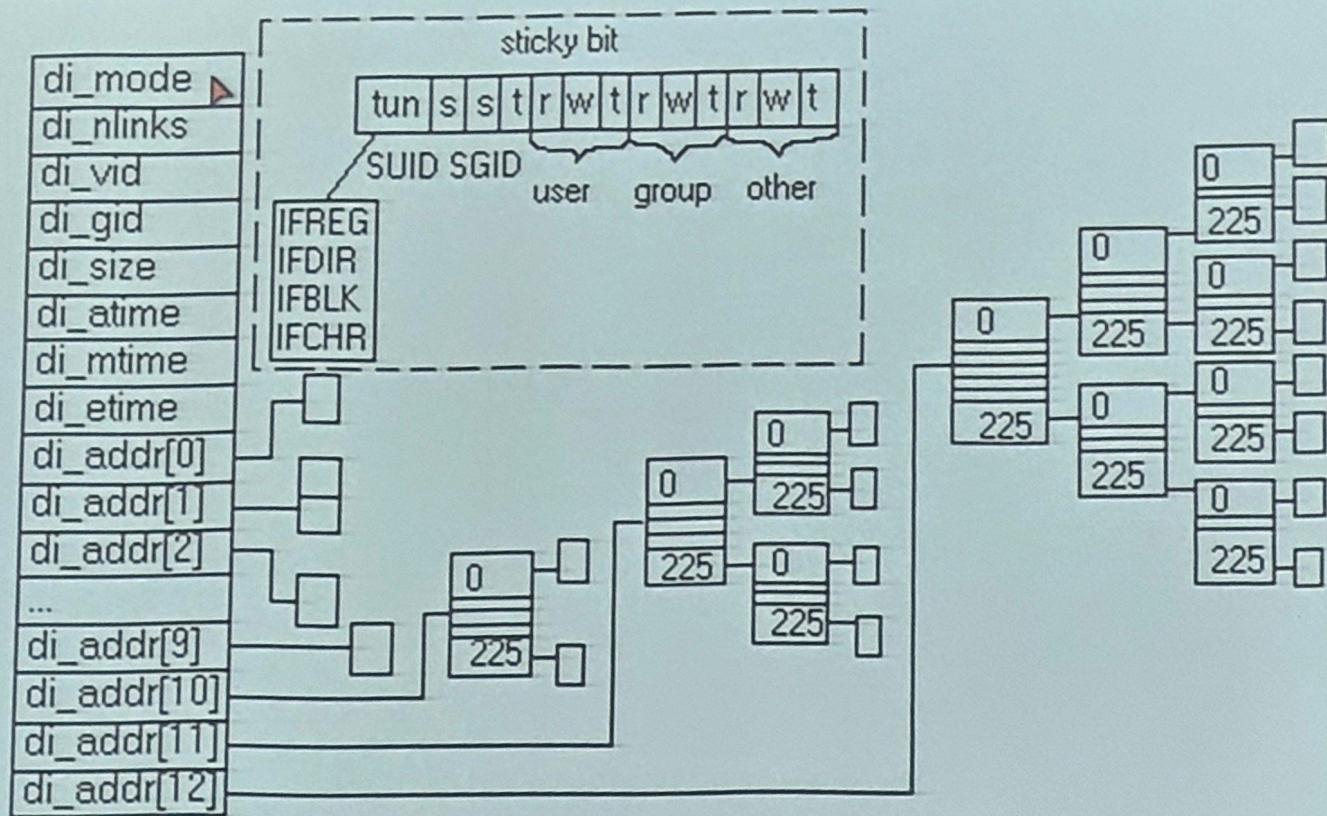
Суперблок содержит общую информацию

Общая уточненная схема структуры раздела

Раздел жесткого диска
с файловой системой ext2



Структура I-node



Для ext2 I-node занимает 128 байт

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

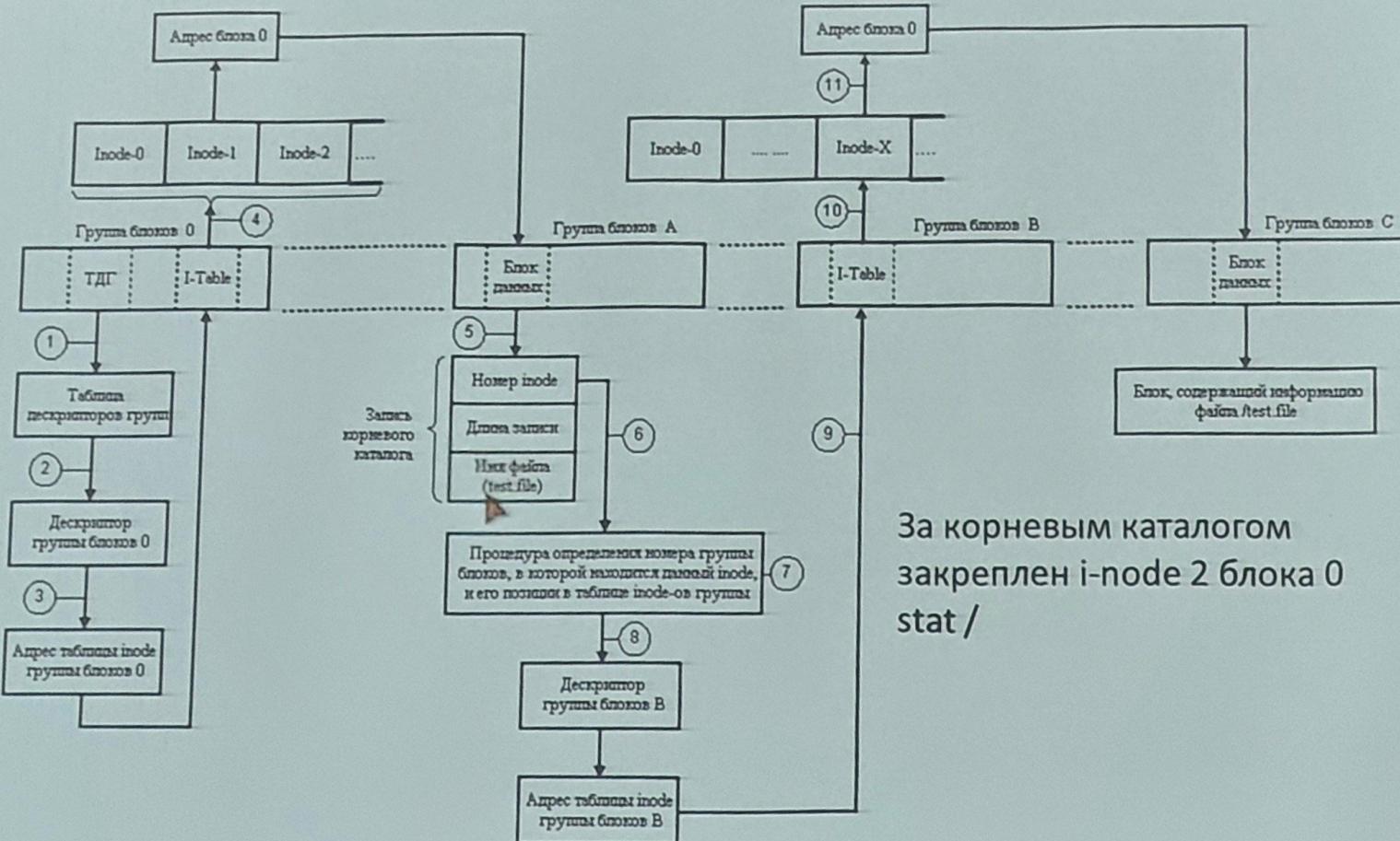


Рис. 4. Порядок выполнения процедуры чтения файла в файловой системе ext2 (на примере файла /test.file)

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 генерируется ядром автоматически и обновляется 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.

ext4

- максимальный объём одного раздела диска **1 эксбибит** (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

/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.



Устройства в Linux

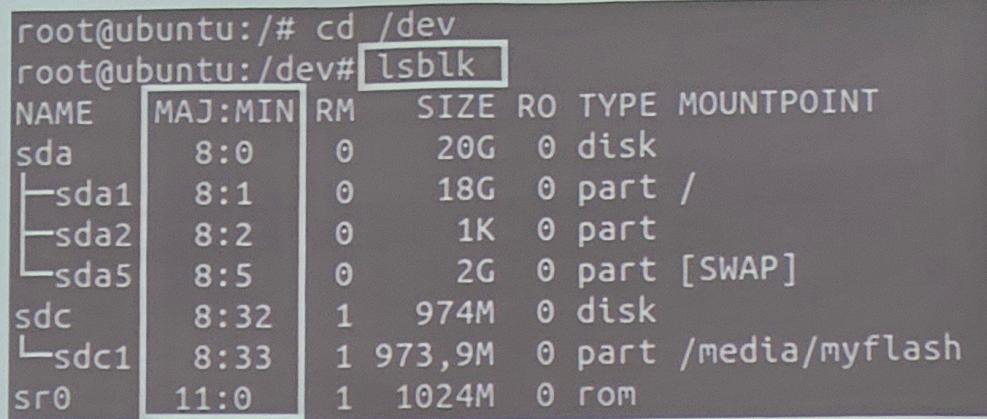
- Имя устройства имеет формат:

`/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
 - а – первое устройство, b – второе и.т.д
- `N` - обозначает раздел.
 - Первичные (основные) разделы нумеруются числами с 1 по 4. Расширенные (логических) разделы начинаются с 5, даже если первичных разделов меньше четырех.
 - Например, `/dev/sda2` второй основной раздел на первом диске

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



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	

Монтирование файловой системы

- Идея монтирования : представить имя раздела в виде каталога монтирования в общем дереве каталогов Linux и получить 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
```

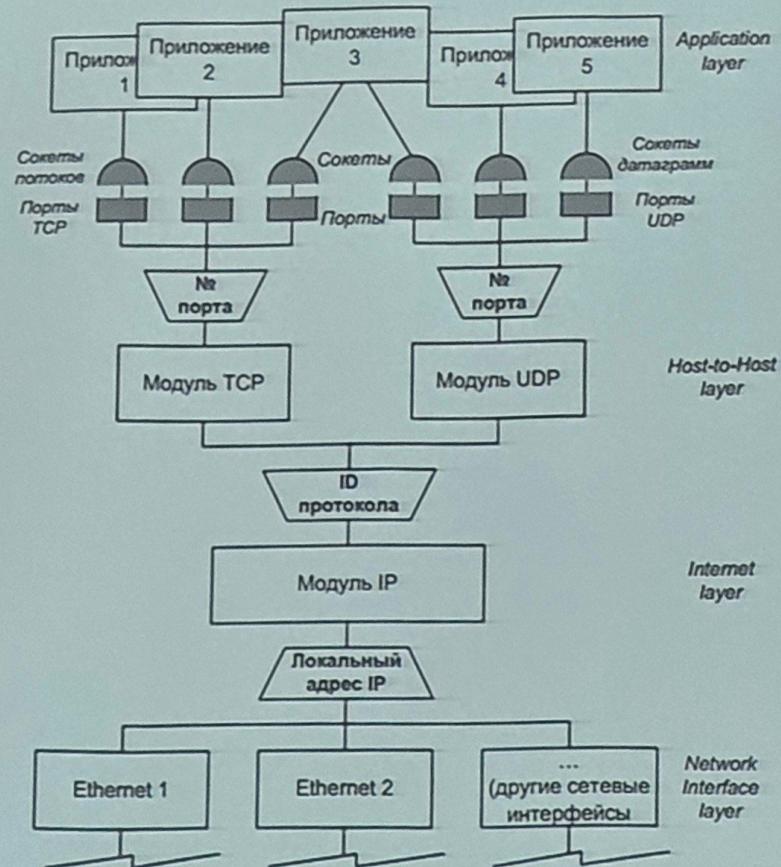
- При необходимости размонтировать:

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

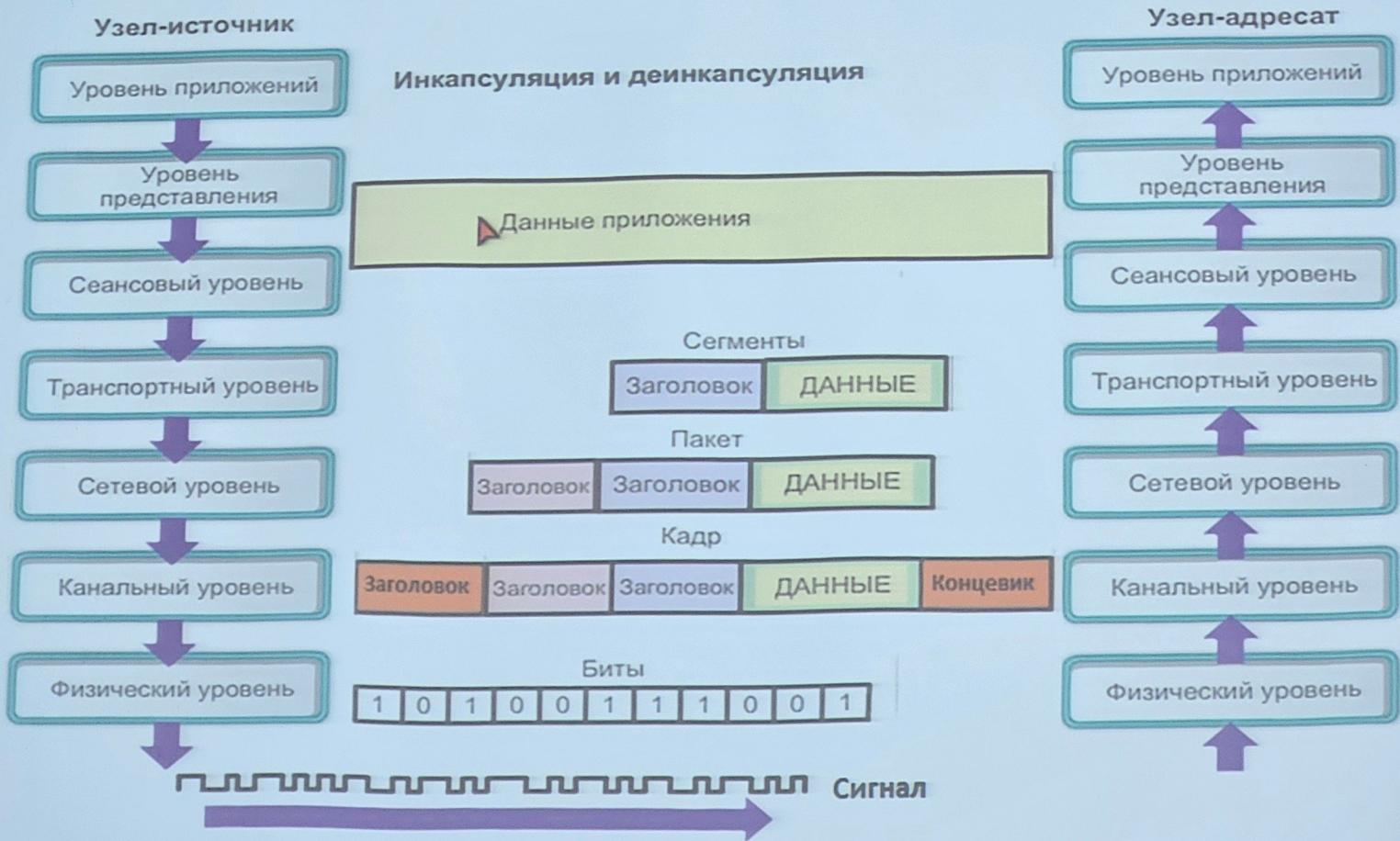
Типы файлов в Linux

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



Модель OSI и блоки данных



Сравнение моделей OSI и TCP/IP

Модель OSI	Набор протоколов TCP/IP	Модель TCP/IP
Уровень приложений		Уровень приложений
Уровень представления	HTTP, DNS, DHCP, FTP	
Сеансовый уровень		
Транспортный уровень	TCP, UDP	Транспортный уровень
Сетевой уровень	IPv4, IPv6, ICMPv4, ICMPv6	Межсетевой уровень
Канальный уровень	PPP, Frame Relay, Ethernet	Уровень сетевого доступа
Физический уровень		

Набор протоколов, обеспечивающих процесс передачи данных по сети называется **стеком протоколов (стек TCP/IP)**

Функции локального управления

- Эти функции используются, для выполнения подготовительных действий, необходимых для организации взаимодействия двух программ-партнеров.

- Создание socket'a**

- Возвращает дескриптор сокета

- int socket (af, type, protocol)*



- int af;*

- int type;*

- int protocol;*

- Аргументы :

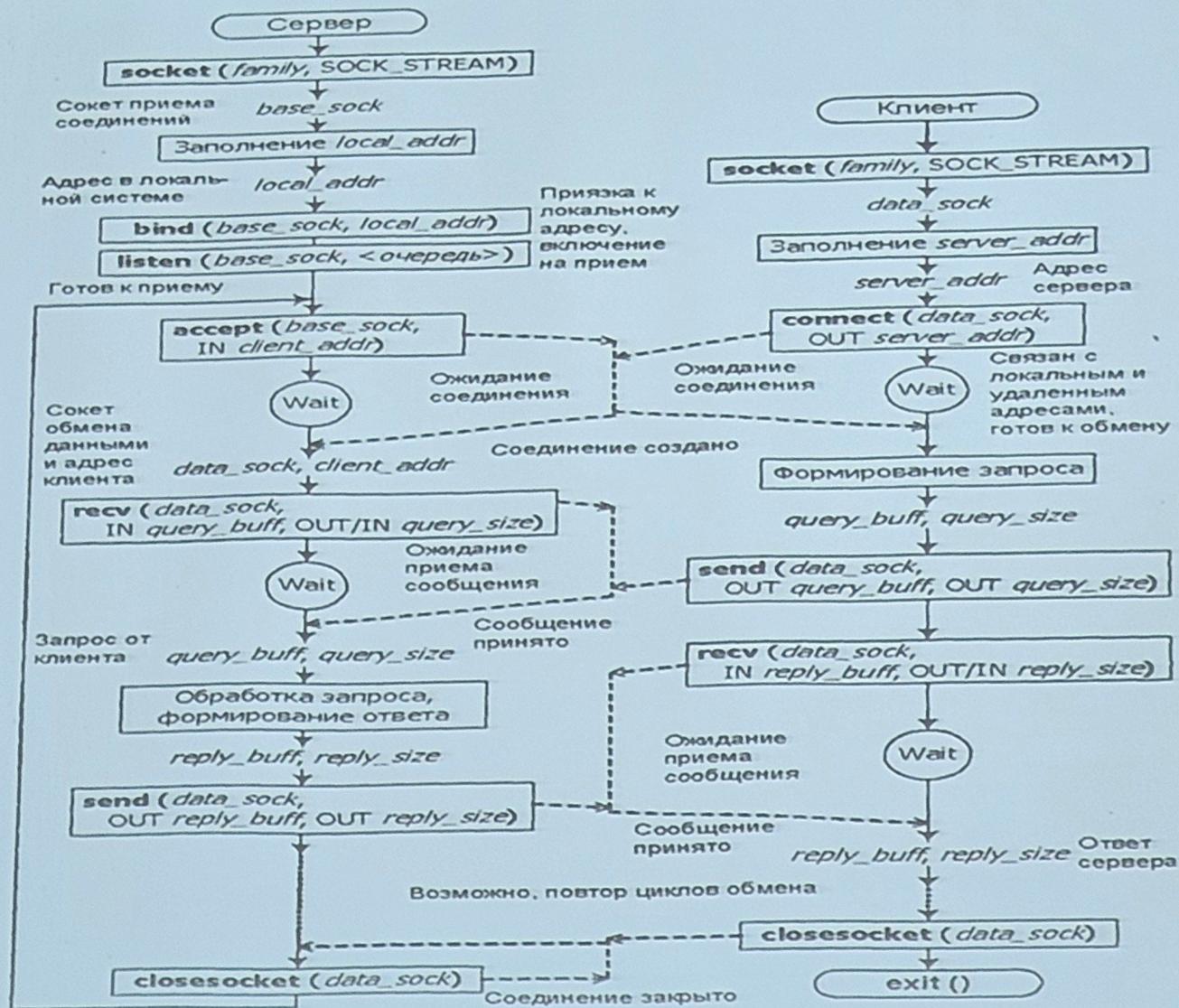
- **address family** задает используемый для взаимодействия тип адресов.
Описывается библиотечными константами.
- AF_INET, PF_INET(семейство Internet-адресов) - , то есть схема «IP-адрес – порт»;
- AF_UNIX, PF_UNIX используется идентификация компьютера с помощью символьного имени;

- **type** задает тип сокета:

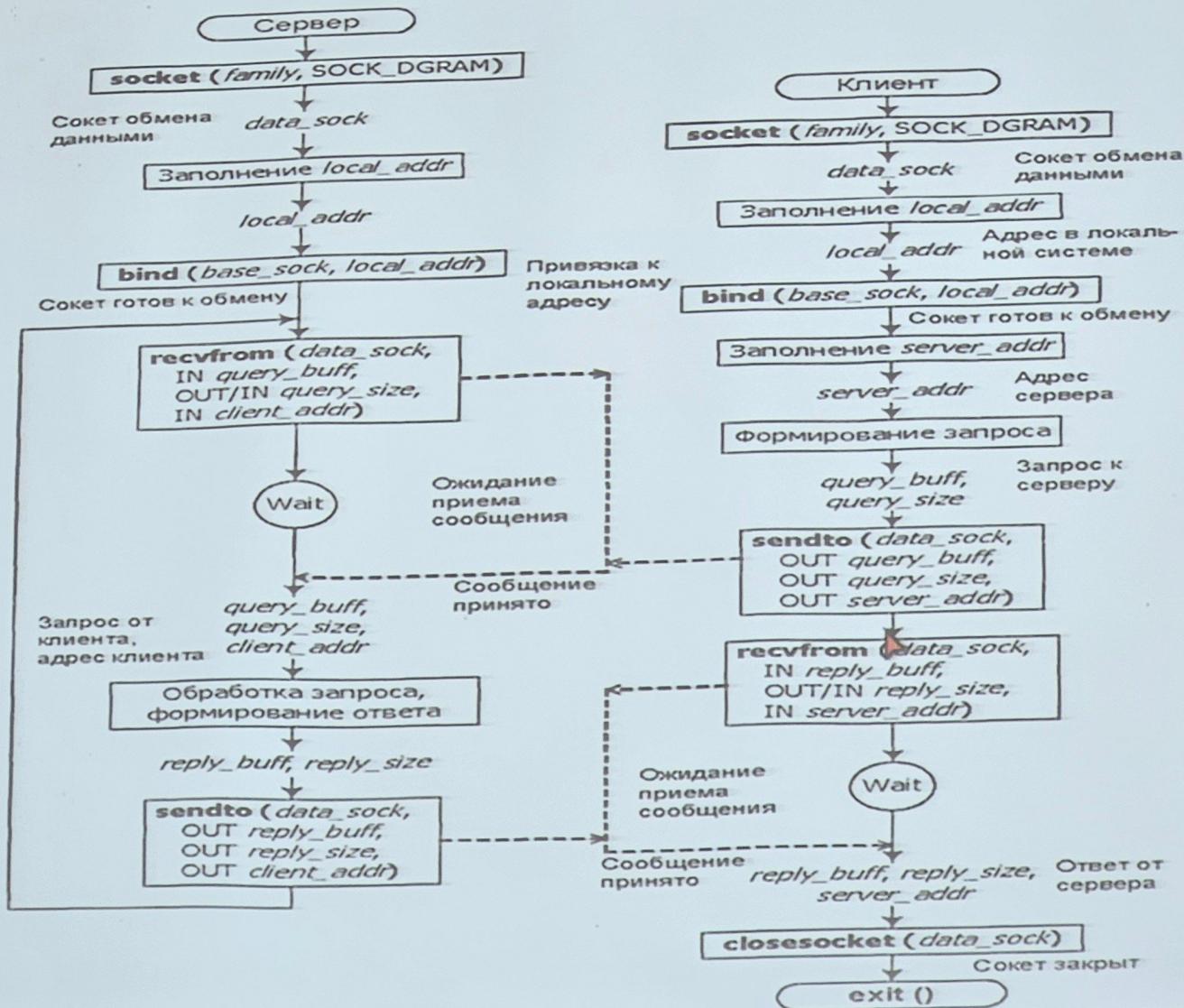
- SOCK_STREAM - с установлением соединения (потоковый).

- SOCK_DGRAM - без установления соединения(датаграммный).

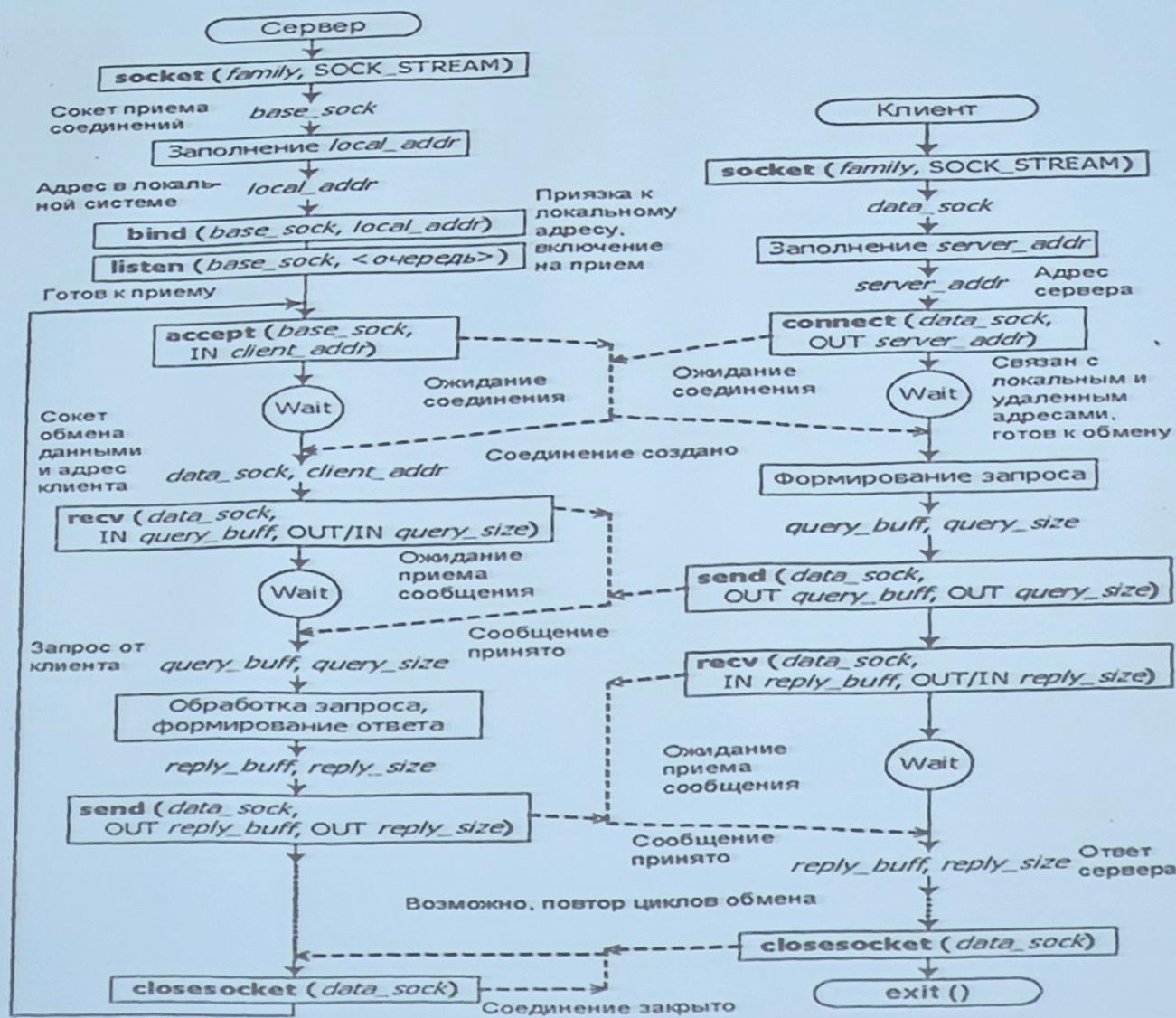
Взаимодействие с установленным соединением (протокол TCP)



Взаимодействие без установления соединения (протокол UDP)



Взаимодействие с установленным соединением (протокол TCP)



Дополнительные биты доступа

t Sticky bit

s Set UID, SUID

s Set GID, SGID

- 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.

- S - разрешает запускать исполняемые файлы от имени другого пользователя (группы), не владельца файла.
- Suid – устанавливается для файлов.
- Sgid – устанавливается для файлов и для папок

Атрибуты доступа

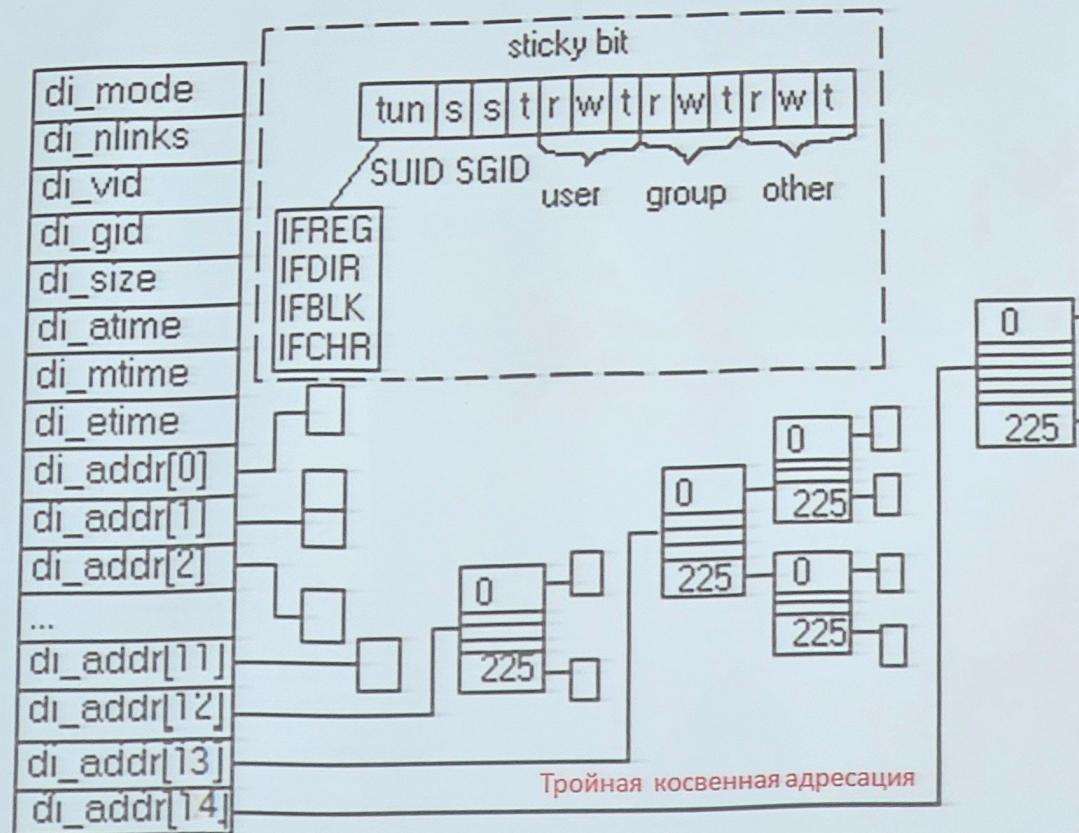
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Тип файла	SUID	SGID	Т-бит	права владельца		права группы		права всех остальных							

Первые четыре бита отвечают за тип файла:

Двоичный код	Десятичный код	Тип файла
1010	10	сокет
1100	12	символическая ссылка
1000	8	обычный файл
0110	6	блочное устройство
0100	4	каталог
0010	2	символьное устройство
0001	1	именованный канал

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

Структура I-node



Для ext2 I-node занимает 128 байт

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. $UID >= 1000$

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?

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 персонаажа
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 бить
прав доступа

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)

Команды управления учетными данными

■ Команды для пользователя

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

■ Команды для группы

- groupadd - создать новую группу
- gpasswd - установить пароль группы
- groupmod - изменить параметры группы
- groupdel - удалить группу