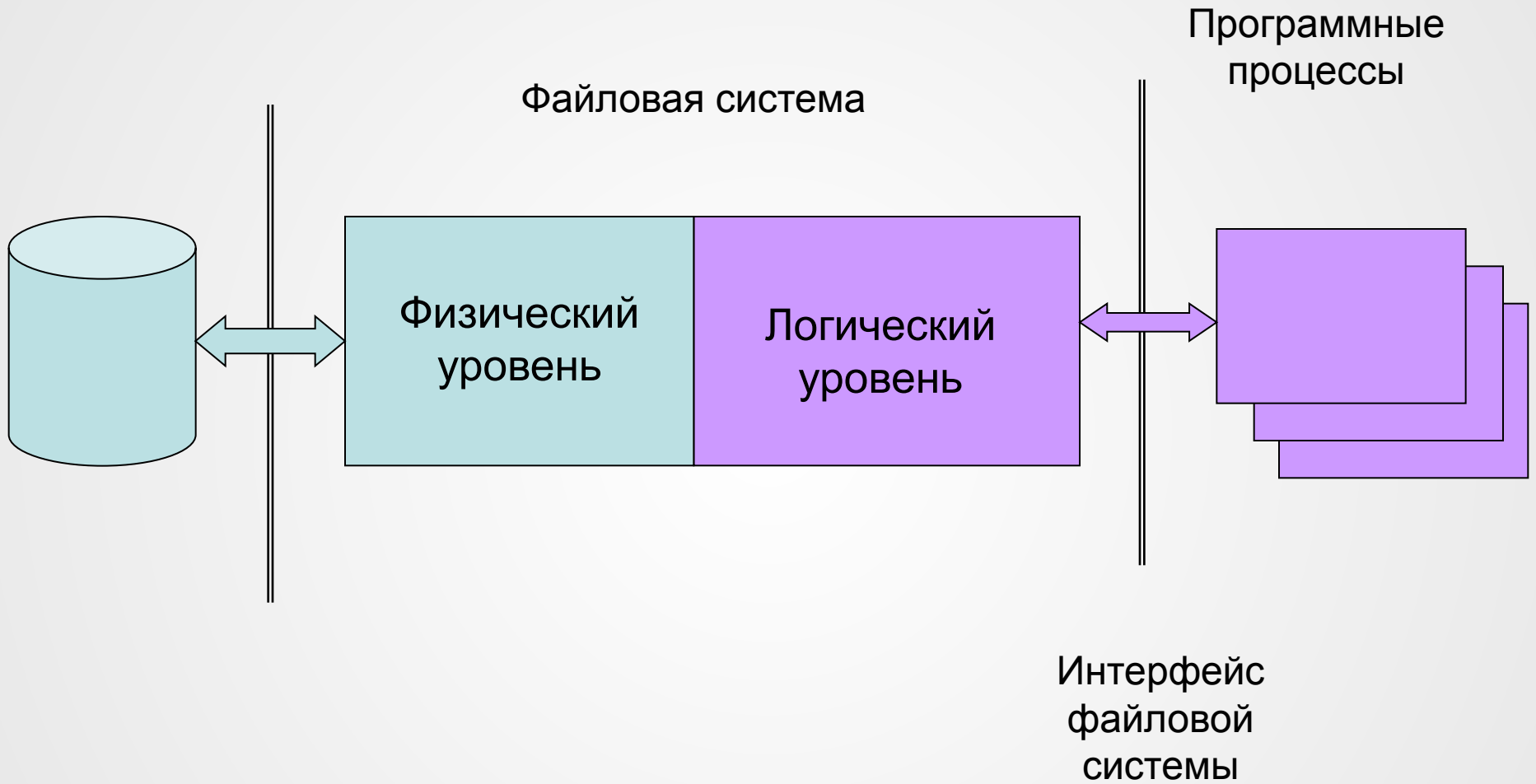


# Система управления данными

- Понятие файла
- Организация файлов
  - Логическая организация
  - Распределение цепочками блоков
  - Распределение с цепочками индексных блоков
  - Распределение с табличными поблочного отображения
- Структура магнитного диска
- Файловые системы ОС UNIX
  - Файловый интерфейс
  - Файловая система System V
  - Файловая система BSD UNIX
  - Виртуальная файловая система
  - Буферный кэш
- Файловая система NTFS

# Понятие файла

- Файл – поименованная совокупность данных
- Файл как набор данных на носителе - физический файл
- Файл как логический объект для работы с данными из программы пользователя – логический файл
- Система управления файлами обеспечивает интерфейс между физическими файлами и программными процессами в ней следует различать физический и логический уровни (подсистемы)



## Операции над файлами

- Открытие файла
- Закрытие файла
- Создание файла
- Удаление файла
- Копирование файла

## Операции над данными

- Чтение
- Запись
- Обновление
- Вставка
- Исключение

# Функции файловой системы

- Создание, модификация и уничтожение файлов.
- Разделение файлов. При этом обеспечивается возможность одновременной работы с файлом нескольких процессов.
- Защита данных от несанкционированного доступа.
- Защита данных от разрушения и обеспечение средств восстановления файла после ошибок.
- Обеспечение пользователя интерфейсом на логическом уровне.

# Дескриптор файла

- Имя файла (в операционной системе UNIX имя файла не входит в дескриптор).
- Данные, необходимые для указания на размещение файла.
- Способ организации файла.
- Тип устройства.
- Данные для управления доступом к файлу.
- Тип файла (текст, объектный модуль и др.).
- Время создания, последнего изменения и последнего доступа к файлу.

В различных операционных системах в дескриптор файла может входить и другая информация.

# Организация файлов

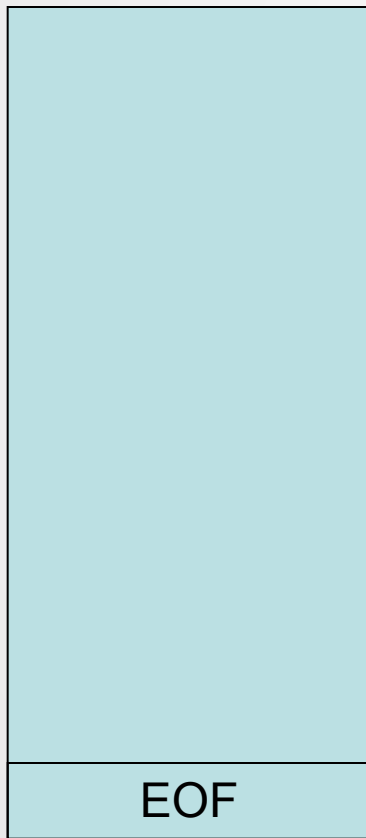


# Логическая организация

- **Физическая запись** – минимальный блок данных при организации обмена с накопителем.
- **Логические записи** – структуры данных, которая рассматривается как единое целое с точки зрения пользователя.
- **Несблокированные записи** – каждая логическая запись содержит одну физическую запись.
- **Сблокированные записи** – логическая запись включает в себя несколько физических записей.
- **Записи фиксированной длины.** Все записи имеют одинаковую длину.
- **Записи переменной длины.** В этом случае необходимо явно указывать фактическую длину каждой записи. Для сблокированных записей кроме длины записи указывается длина каждого блока.
- **Записи неопределенной длины.** Записи имеют различную длину, но длина записи явно не задается, при этом для разделения записей используются специальные маркеры конца записи.

# Логическая организация файла

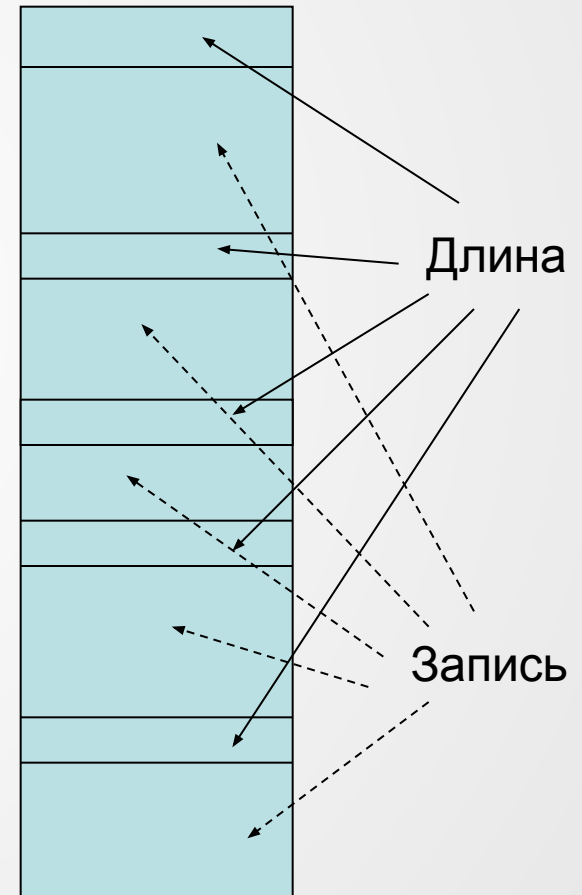
Неструктурированный файл



Записи фиксированной длины

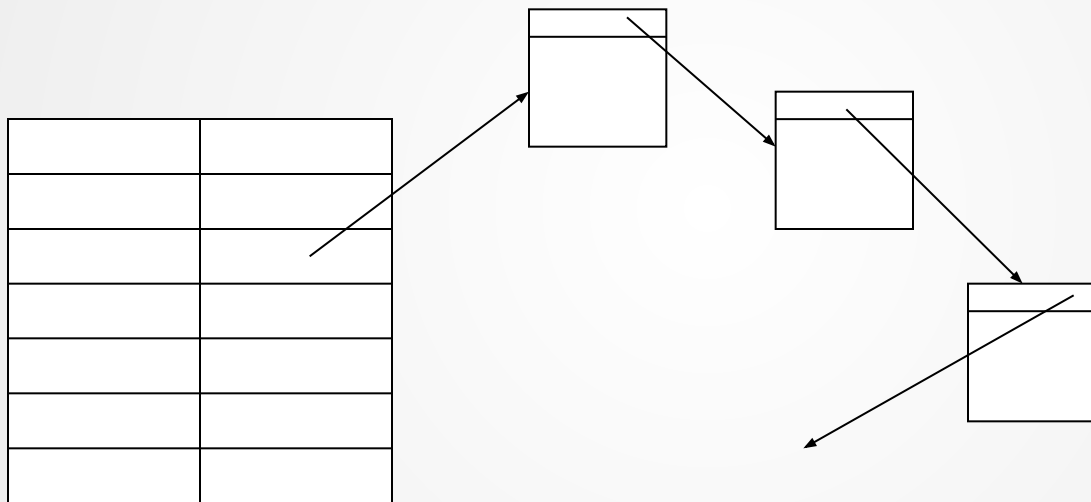


Записи переменной длины

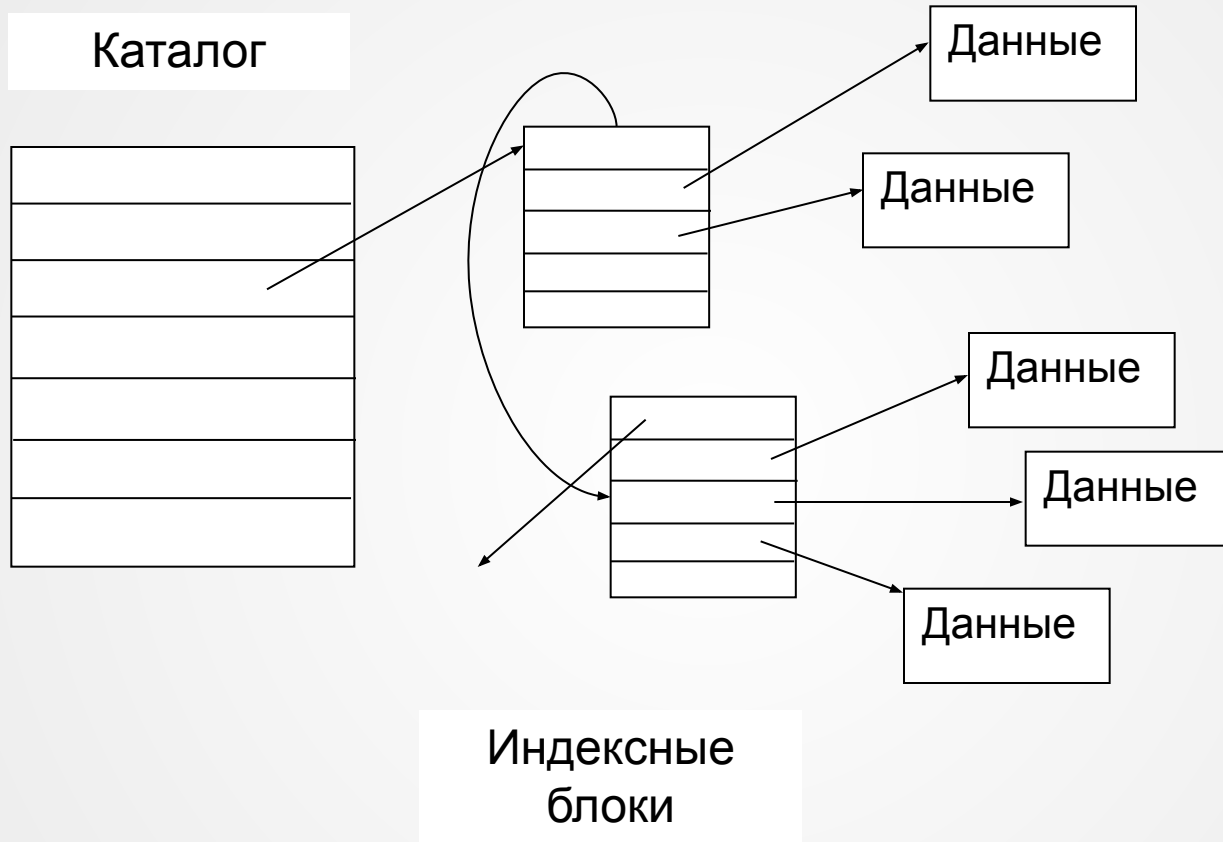


# Распределение цепочками блоков

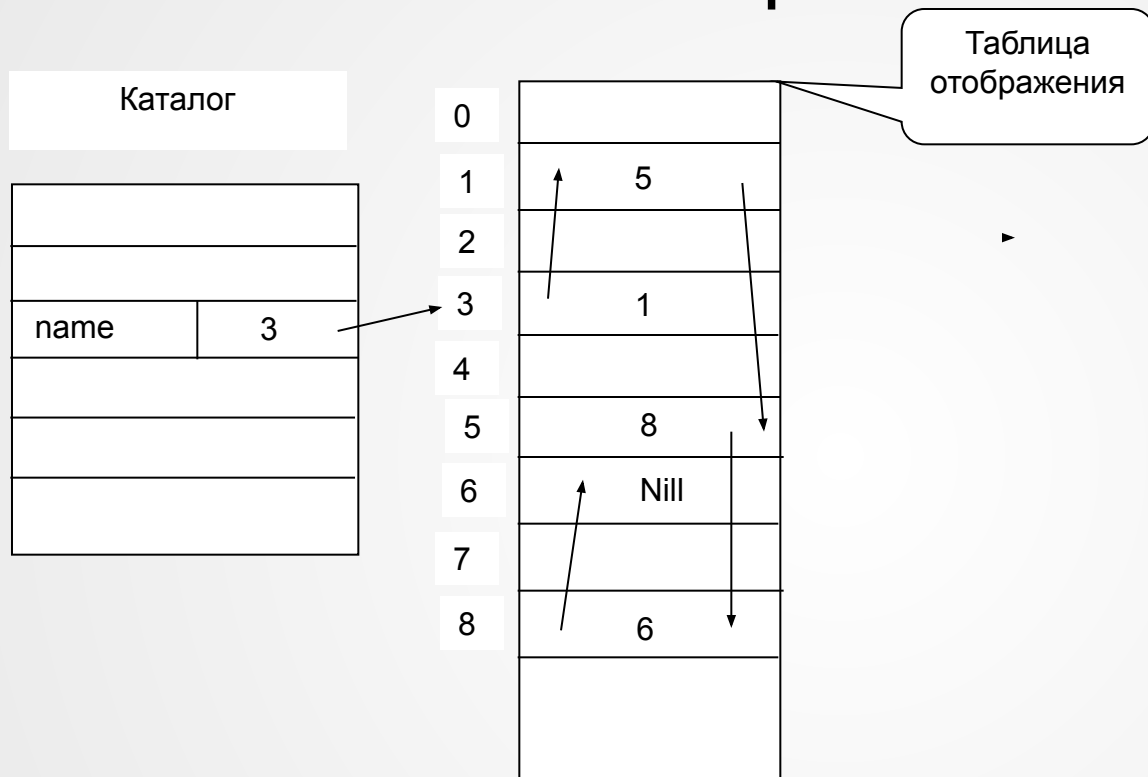
Каталог



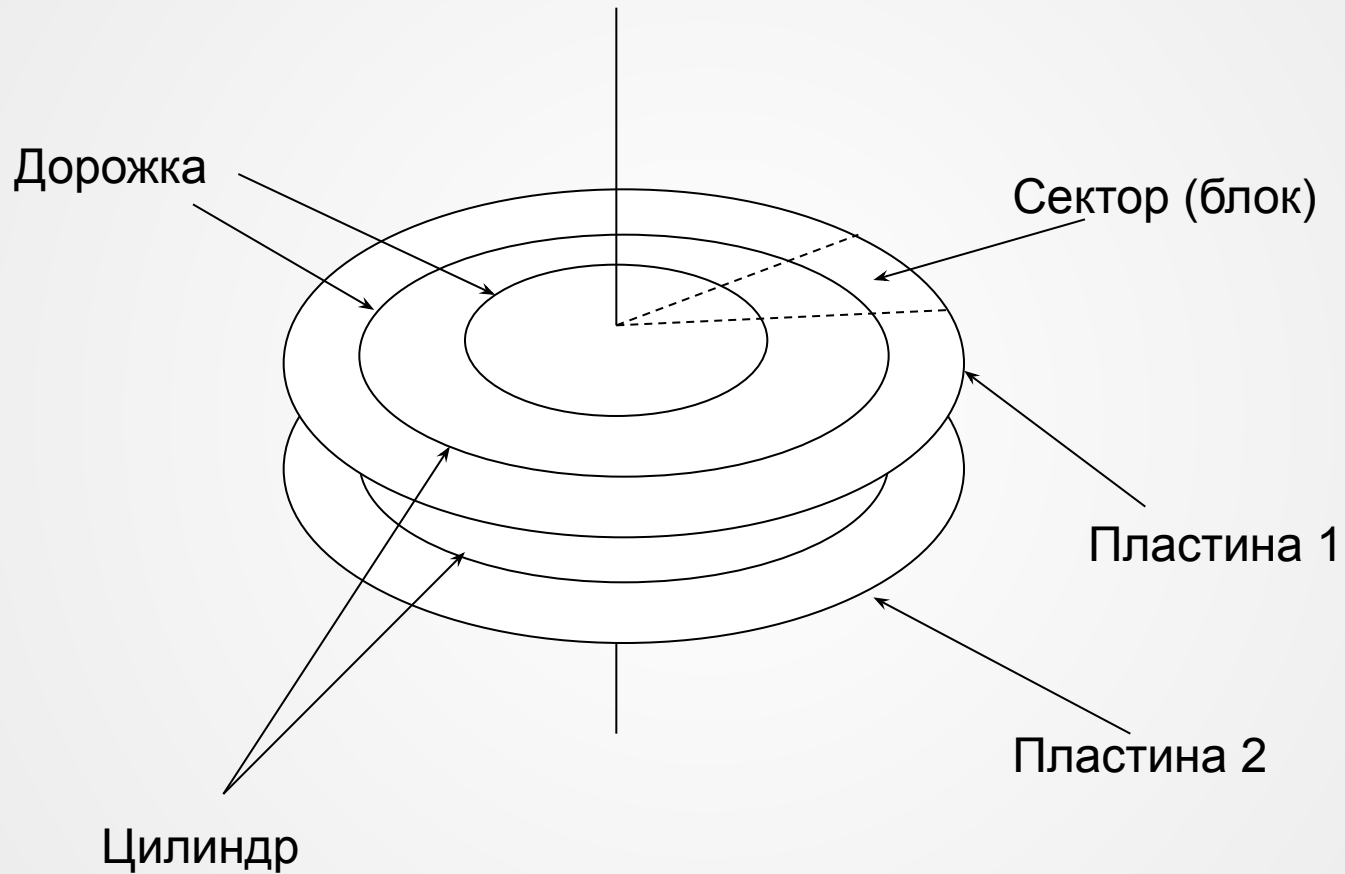
# Распределение с цепочками индексных блоков



# Распределение с таблицами поблочного отображения



# Структура магнитного диска



# Разбиение диска на разделы

- Физический адрес [c-h-s]
- Номер цилиндра - c
- Номер головки (рабочей поверхности) – h
- Номер сектора – s
- Типы разделов – primary и extended
- Главная загрузочная запись – MBR (master boot record) – [0-0-1]

Размер	Содержимое
446	Non-system bootstrap
16	Partition 1 entry
16	Partition 2 entry
16	Partition 3 entry
16	Partition 4 entry

# Формат элемента таблицы разделов

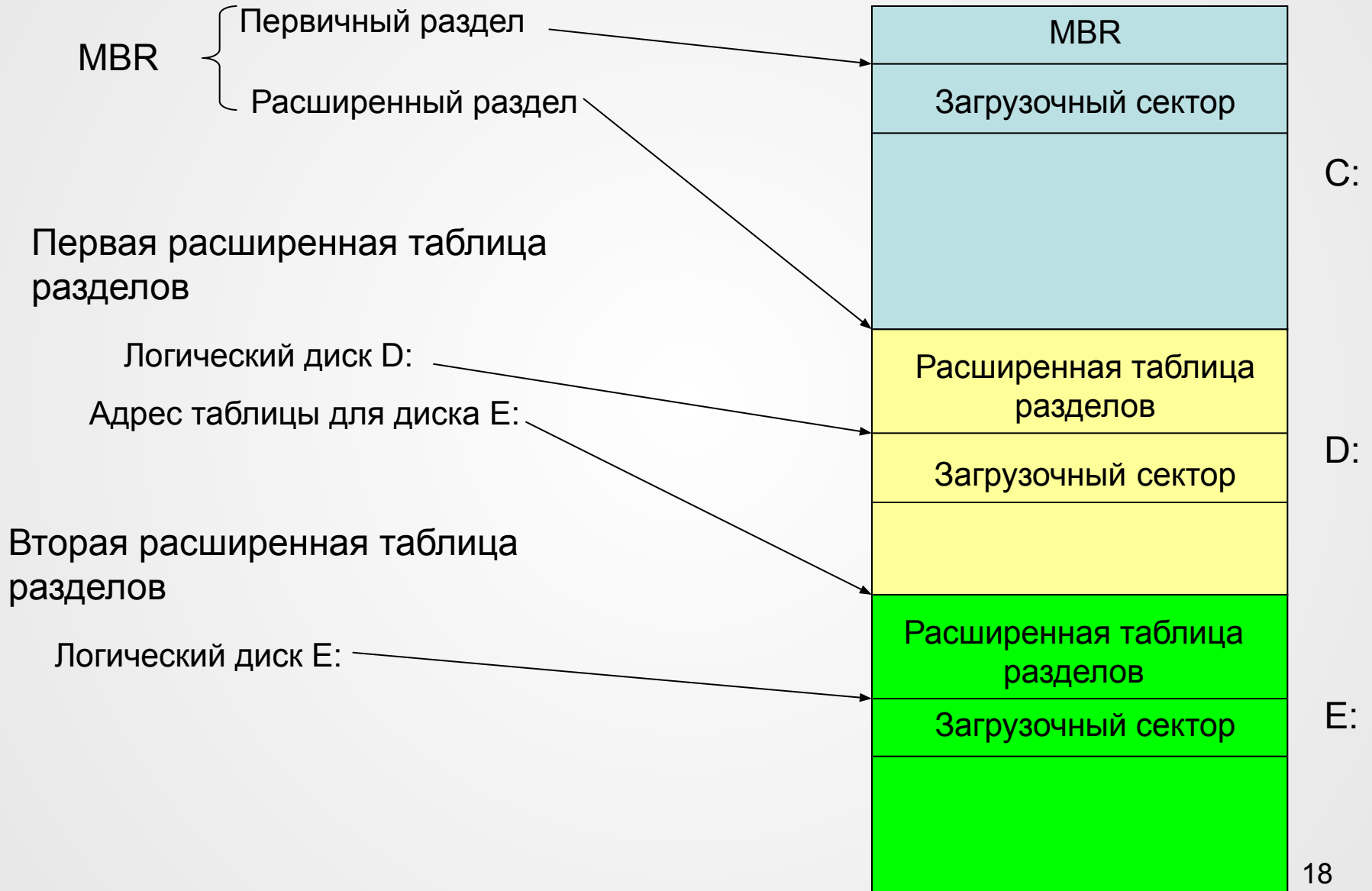
Название записи элемента	Длина
Флаг активности раздела	1
Номер головки начала раздела	1
Номер сектора и цилиндра загрузочного раздела	2
Кодовый идентификатор операционной системы	1
Номер головки конца раздела	2
Номер сектора и цилиндра последнего сектора раздела	2
Младшее и старшее двухбайтовое слово относительного номера начального сектора	4
Младшее и старшее двухбайтовое слово размера раздела в секторах	4



# Типы разделов

- 00 – Empty
- 01 – FAT12
- 04 – FAT16 (<32MB)
- 05 – Extended
- 06 – FAT16
- 07 – NTFS
- 82 – Linux swap
- 83 – Linux native
- 85 – Linux extended
- 86 – NTFS volume set

# Разбиение диска на разделы



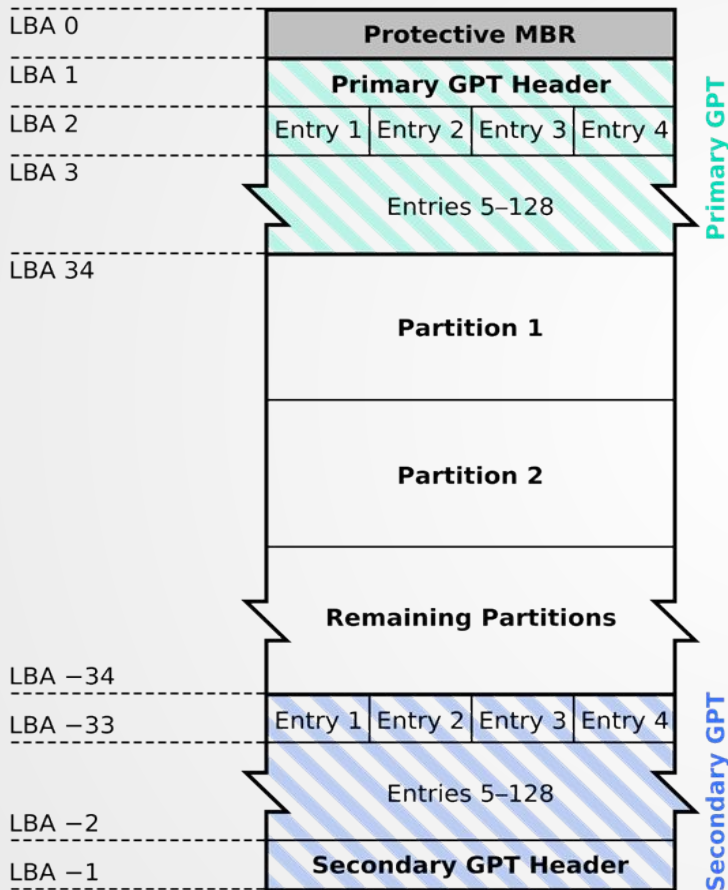
# Ограничения MBR

- Диск в формате MBR может иметь **только четыре основных раздела** и может управлять данными **только до 2 ТБ** .
- Диски MBR резервируют первый сектор диска для хранения информации о разделах диска и расположении файлов операционной системы.
- В случае MBR данные о разделе диска и загрузочные команды хранятся в одном месте.
- MBR плохо справляется с ошибками и плохо поддается восстановлению.

# Таблица разделов с глобально уникальными идентификаторами (GUID Partition Table)

- **GPT** — стандарт формата размещения таблиц разделов на физическом жестком диске.
- Он является частью Расширяемого микропрограммного интерфейса (*Extensible Firmware Interface*, EFI) — стандарта, предложенного Intel на смену BIOS.
- EFI использует GPT там, где BIOS использует Главную загрузочную запись (MBR).

## GUID Partition Table Scheme



- Каждый логический блок (LBA) имеет размер 512 байт
- Каждая запись (entry) — 128 байт.
- Отрицательные адреса логических блоков обозначают нумерацию с конца диска (-1 — последний блок, -2 — предпоследний и т.д.)

- MBR присутствует в самом начале диска (блок LBA 0) как для защиты, так и в целях совместимости.
- GPT использует современную систему адресации логических блоков (LBA).

- **Оглавление таблицы разделов (LBA 1)** указывает те логические блоки на диске, которые могут быть задействованы пользователем . Оно также указывает число и размер записей данных о разделах, составляющих таблицу разделов. Стандартно в Microsoft Windows резервируется 128 записей данных о разделах. Таким образом, возможно создание 128 разделов на диске.
- Оглавление содержит GUID (глобально уникальный идентификатор) диска. В оглавлении также содержится его собственный размер и местоположение (всегда блок LBA 1), а также размер и местоположение вторичного (запасного) оглавления и таблицы разделов, которые всегда размещаются в последних секторах диска.

- Записи данных о разделах (Partition entries) (**LBA 2-33**) расположены с равным приращением адресов. Первые 16 байт определяют GUID типа раздела. Например, GUID системного EFI-раздела имеет вид «C12A7328-F81F-11D2-BA4B-00A0C93EC93B».
- Следующие 16 байт содержат GUID, уникальный для данного конкретного раздела. Далее записываются данные о начале и конце 64-битных LBA, если они имеются. Остальное место отводится информации об именах и атрибутах разделов.



# Различия между MBR и GPT

- 1. MBR-диск может содержать только до 4 основных разделов, в то время как GPT-диски могут иметь до 128 основных разделов.
- 2. Если необходимо создать более четырех разделов, надо создать расширенный раздел на дисках MBR, а затем создать логические разделы, тогда как на дисках GPT такого принуждения нет.
- 3. Первый сектор и только первый сектор MBR-дисков содержат информацию о жестком диске, в то время как в GPT-дисках информация о жестком диске и его разделах реплицируется более одного раза, поэтому он работает, даже если первый сектор поврежден

4. MBR-диск не сможет управлять дисками емкостью более 2 ТБ, в то время как для дисков GPT такого ограничения нет

5. Все операционные системы поддерживают диски MBR, в то время как для GPT совместимы только 64-разрядные Windows XP и более поздние версии Windows.

6. Для поддержки загрузки только Windows 8 и 10 поддерживают 32-разрядную загрузку, в противном случае все предыдущие версии, такие как Windows 7, Windows Vista, 32-разрядные версии Windows XP, не могут загружаться с GPT-дисков.

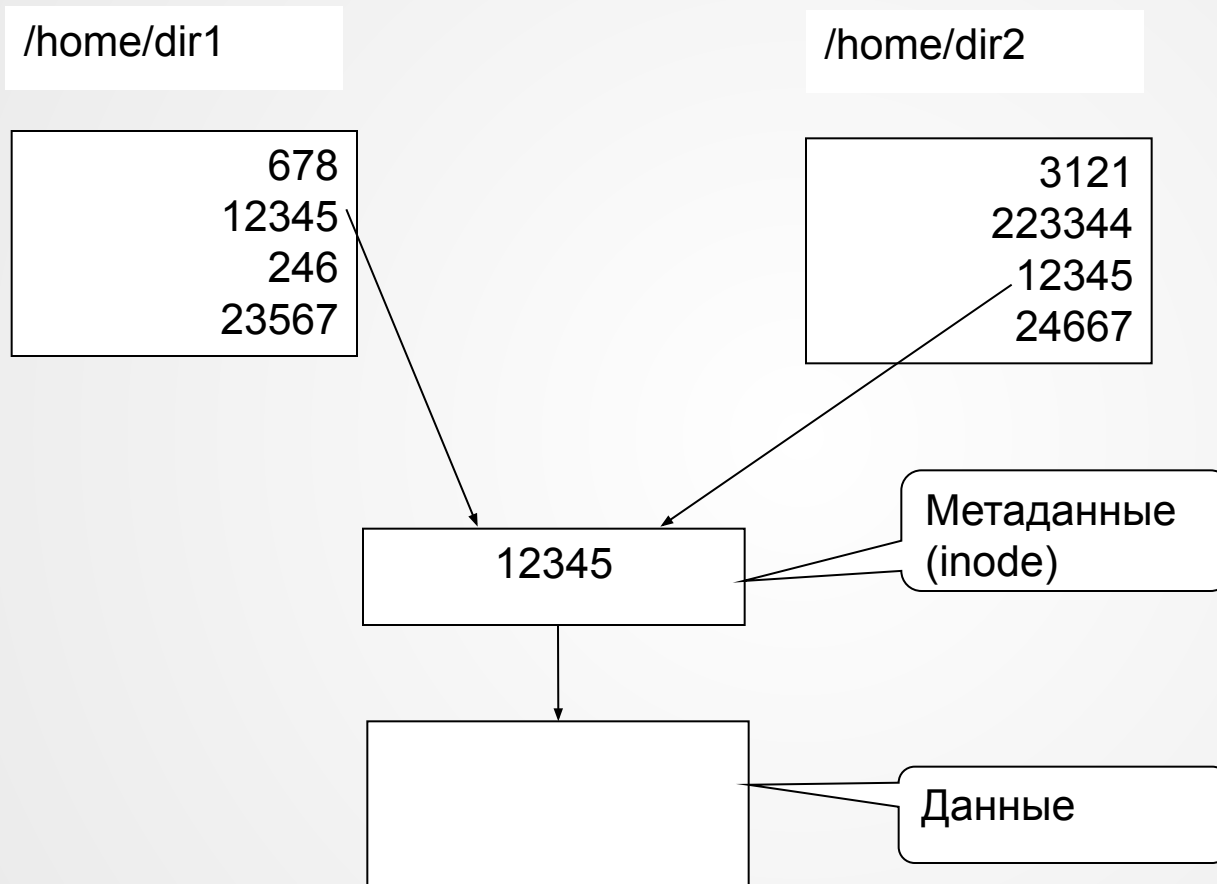
# Файловые системы ОС UNIX

- Типы файлов:
  - Обычный файл
  - Каталог
  - Специальный файл устройства
  - Именованный канал
  - Связь
  - Сокет

# Структура каталога

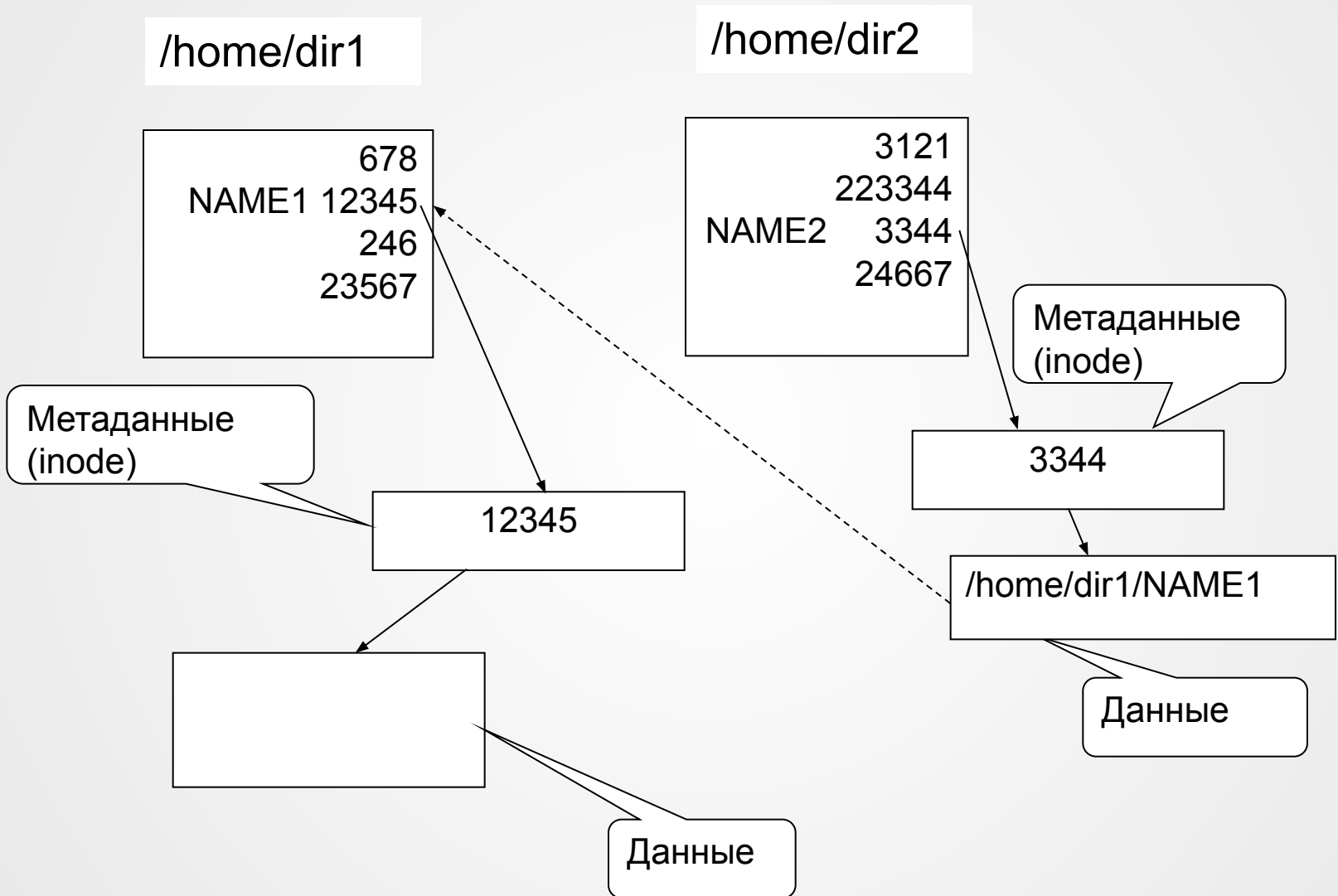
Номер inode	Имя файла

# Организация жесткой связи



In <имя файла>

# Символическая связь



# Файловый интерфейс

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
```

```
#include <fcntl.h>
```

```
int open(const char *path, int flags, mode_t mode);
```

O\_RDONLY - открытие файла только для чтения;

O\_WRONLY - открытие файла только для записи;

O\_RDWR - открытие файла для чтения и записи.

Значение параметра может логически складываться с модификаторами:

O\_APPEND - данные добавляются в конец файла;

O\_CREAT - создается файл, если он не существует;

O\_TRUNC - если файл существует, то его содержимое теряется, а размер устанавливается равным 0;

O\_EXCL - используется совместно с флагом O\_CREAT, в этом случае попытка создать файл, если он уже существует оканчивается неудачей.

- Чтение из файла:

```
int read(int fdes, char *buf, size_t count);
```

- Запись в файл может выполняться по функции:

```
int write(int fdes, char *buf, size_t count);
```

- Первый параметр используется дескриптор файла.
- Вторым параметром указывается на буфер обмена.
- Третьим параметром - длина буфера.
- Закрывается файл функцией  

```
int close(int fdes);
```



# Работа с каталогами

```
#include <sys/types.h>
```

```
#include <unistd.h>
```

```
int mkdir(const char *path_name, mode_t mode);
```

```
int rmdir(const char *path_name);
```

```
int chdir(const char *path_name);
```

```
char *getcwd(char *name, size_t size);
```

```
int chmod(const char *path_name, mode_t flag);
```

```
int fchmod(int fdesc, mode_t flag);
```

# POSIX.1, SVR4

```
#include <sys/types.h>
```

```
#include <dirent.h>
```

```
DIR *opendir(const char *dirname);
```

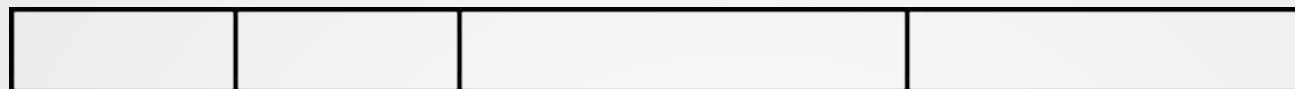
```
int closedir(DIR *dirptr);
```

```
struct dirent *readdir(DIR *dirptr);
```

```
void rewinddir(DIR *dirptr);
```

```
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>
main(int argc, char *argv[])
{
    struct dirent *mydir;
    DIR *dir_ds;
    if((dir_ds = opendir(argv[1])) == NULL) {
        perror("Ошибка открытия каталога");
        return 1;
    }
    while((mydir = readdir(dir_ds)) != NULL)
        printf("Файл - %s, inode = %d\n", mydir->d_name, mydir->d_ino);
    puts("Конец каталога");
    closedir(dir_ds);
    return 0;
}
```

# Файловая система System V (s5fs)



блок  
загрузки

супер-  
блок

список индексов

информационные  
блоки

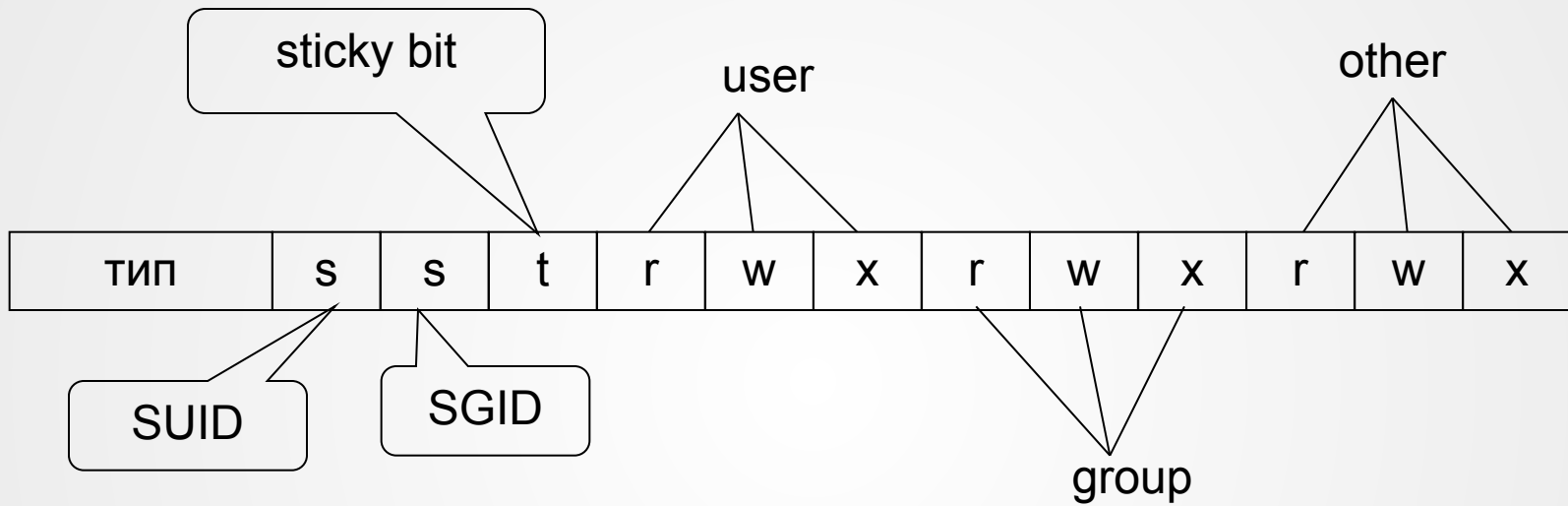
## • Суперблок:

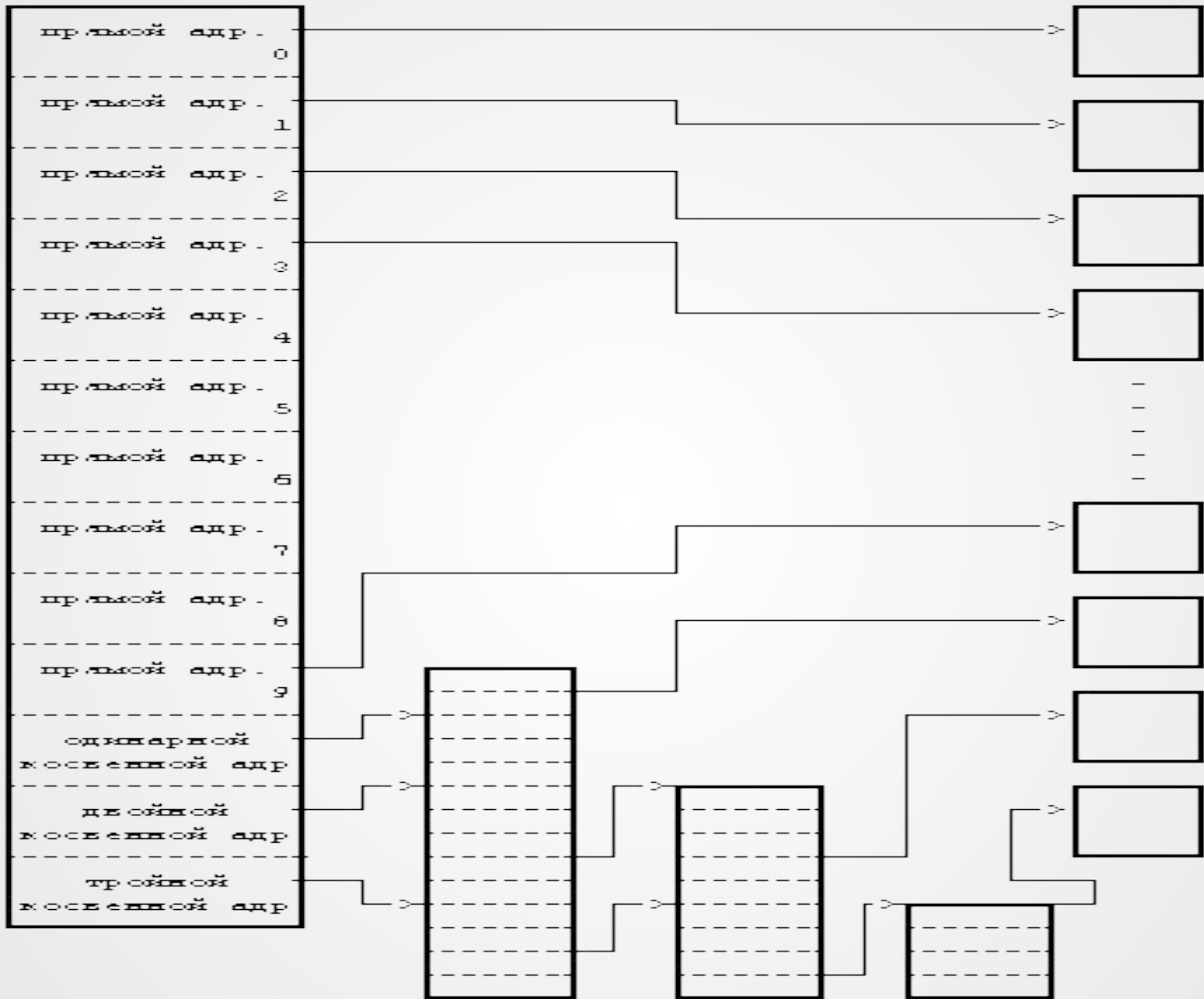
- Тип файловой системы (`s_type`)
- Размер файловой системы в логических блоках, включая сам суперблок, `ilist` и блоки хранения данных (`s_fsize`)
- Размер массива индексных дескрипторов (`s_istize`)
- Число свободных блоков, доступных для размещения (`s_tfree`)
- Число свободных `inode`, доступных для размещения (`s_tinode`)
- Флаги (флаг модификации `s_fmod`, флаг режима монтирования `s_fronly`)
- Размер физического блока (512, 1024, 2048)
- Список номеров свободных `inode`
- Список адресов свободных блоков

# Структура индексного дескриптора

```
struct dinode {  
    unsigned short di_mode; //режим доступа и тип файла  
    short di_nlink;        //счетчик жестких ссылок  
    short di_uid;         //идентификатор владельца  
    short di_gid;         //идентификатор группы  
    off_t di_size;        //размер файла в байтах  
    char di_addr[40];     //указатели на блоки диска  
    time_t di_atime;      //время последнего доступа к файлу  
    time_t di_mtime;      //время последней модификации данных  
    time_t di_ctime;      //время последней модификации inode  
};
```

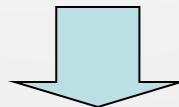
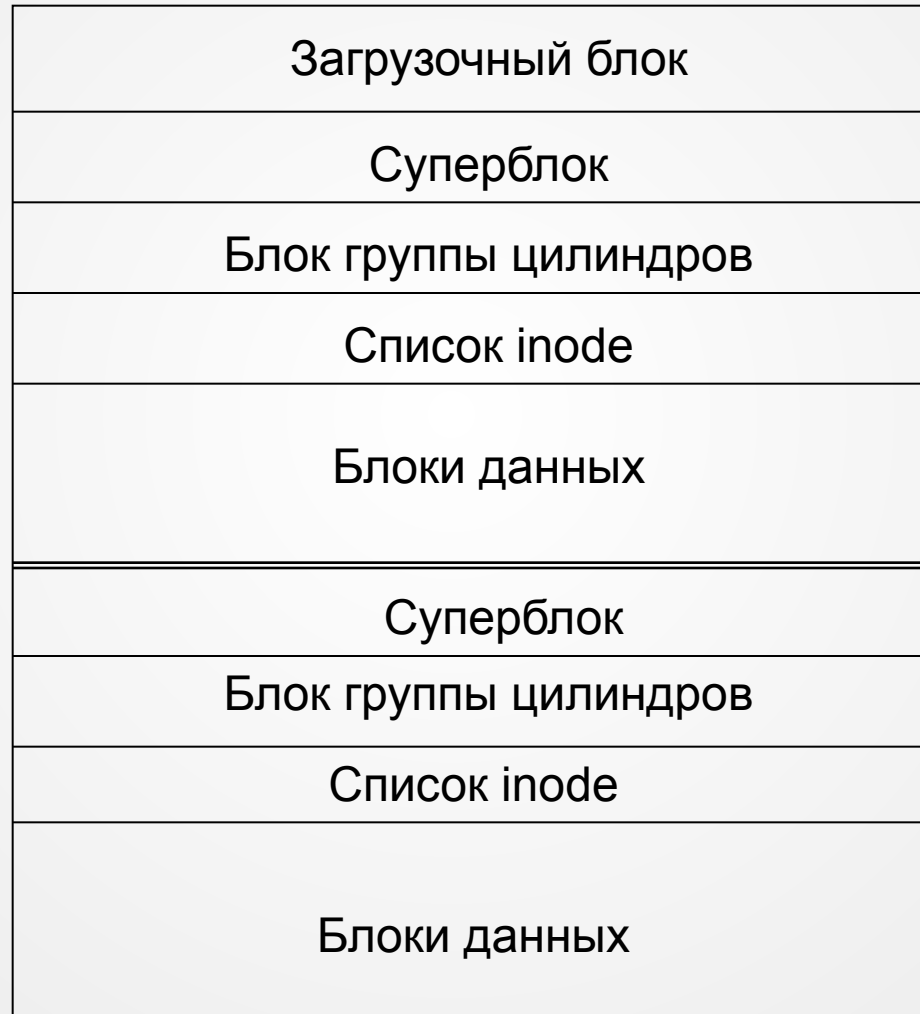
# di\_mode



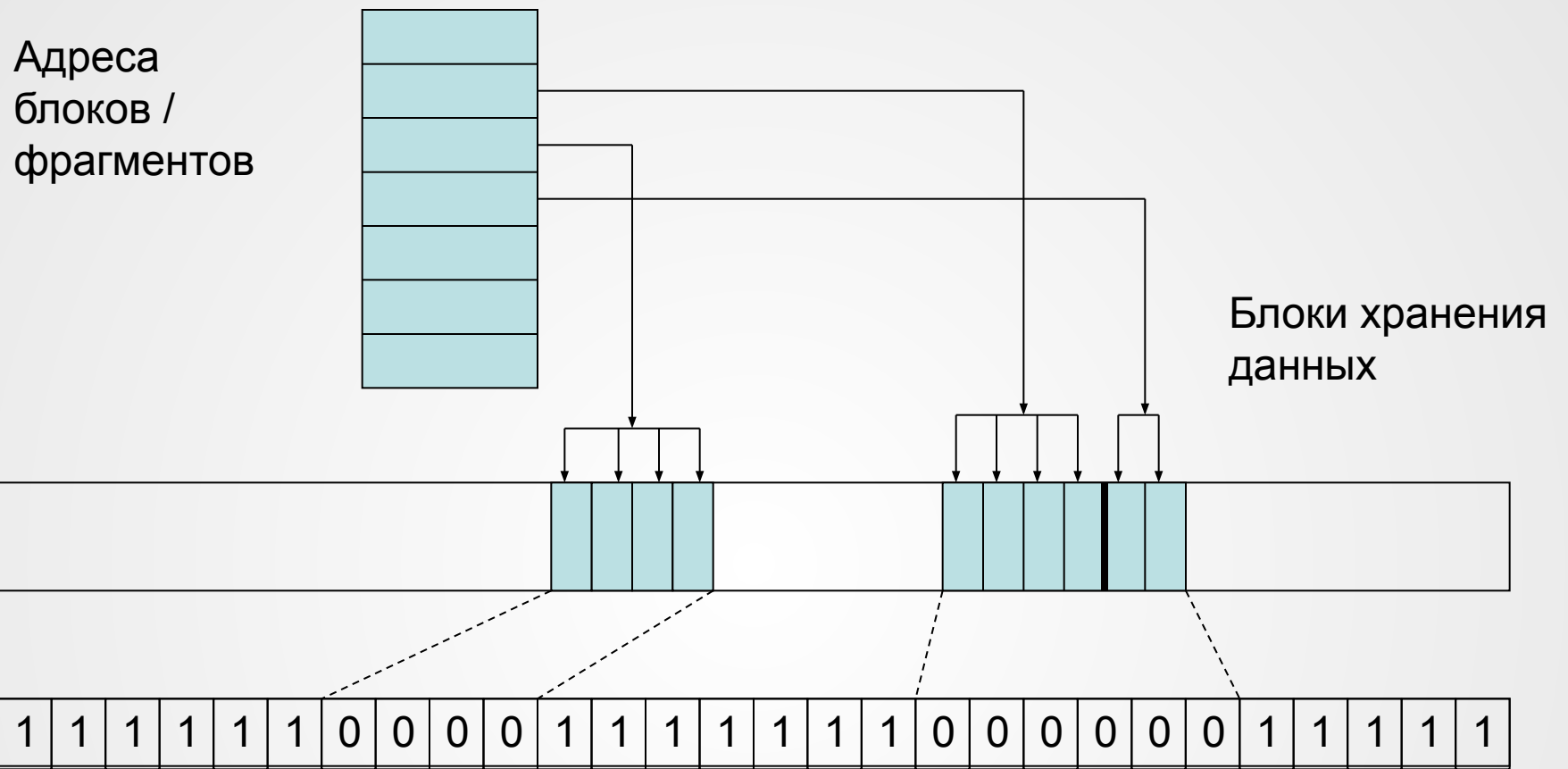


# Файловая система BSD UNIX (FFS, ufs)

## Физическая организация файловой системы







Карта свободных блоков/фрагментов

# Файловые системы ОС Linux

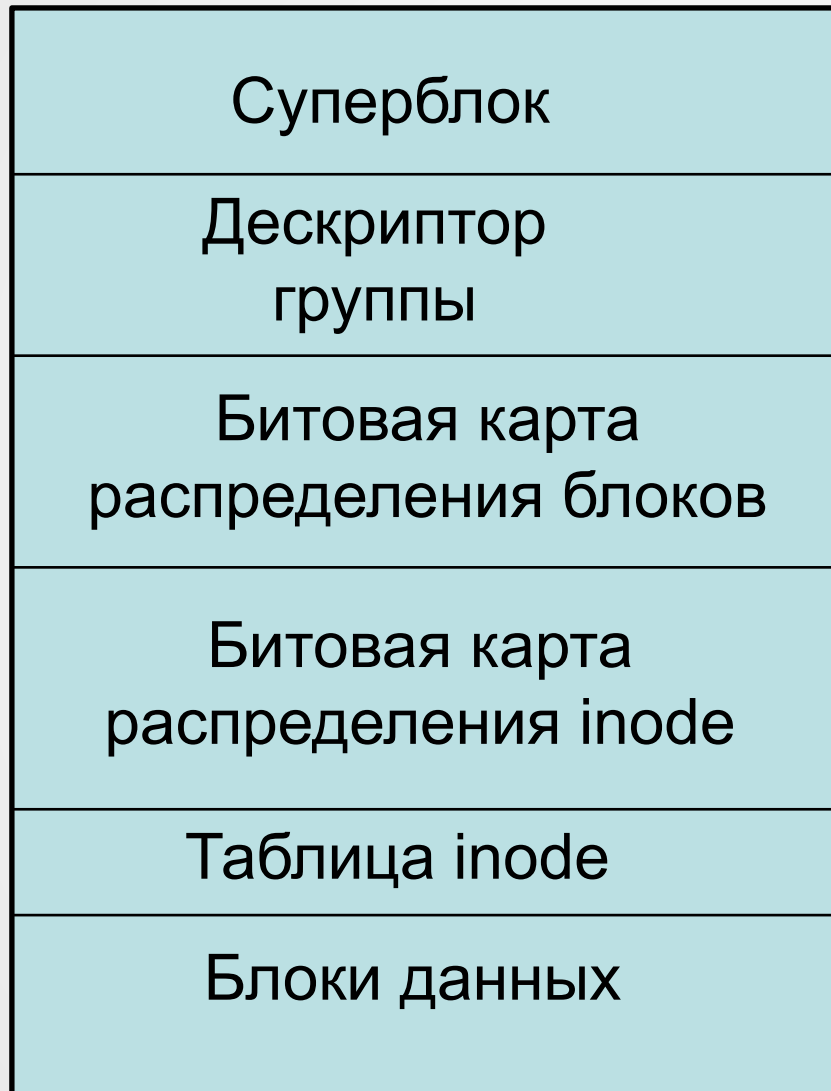
Second Extended File System – ext2

ext – 1992 год

ext2 – 1993 год

## Физическая организация

- Блоки 1024, 2048, 4096, 8192 байта



Структура блочной группы

# Суперблок

- Общее число блоков inode в ФС
- Размер блока ФС
- Количество блоков и inode в группе блоков
- Размер inode
- Идентификатор ФС

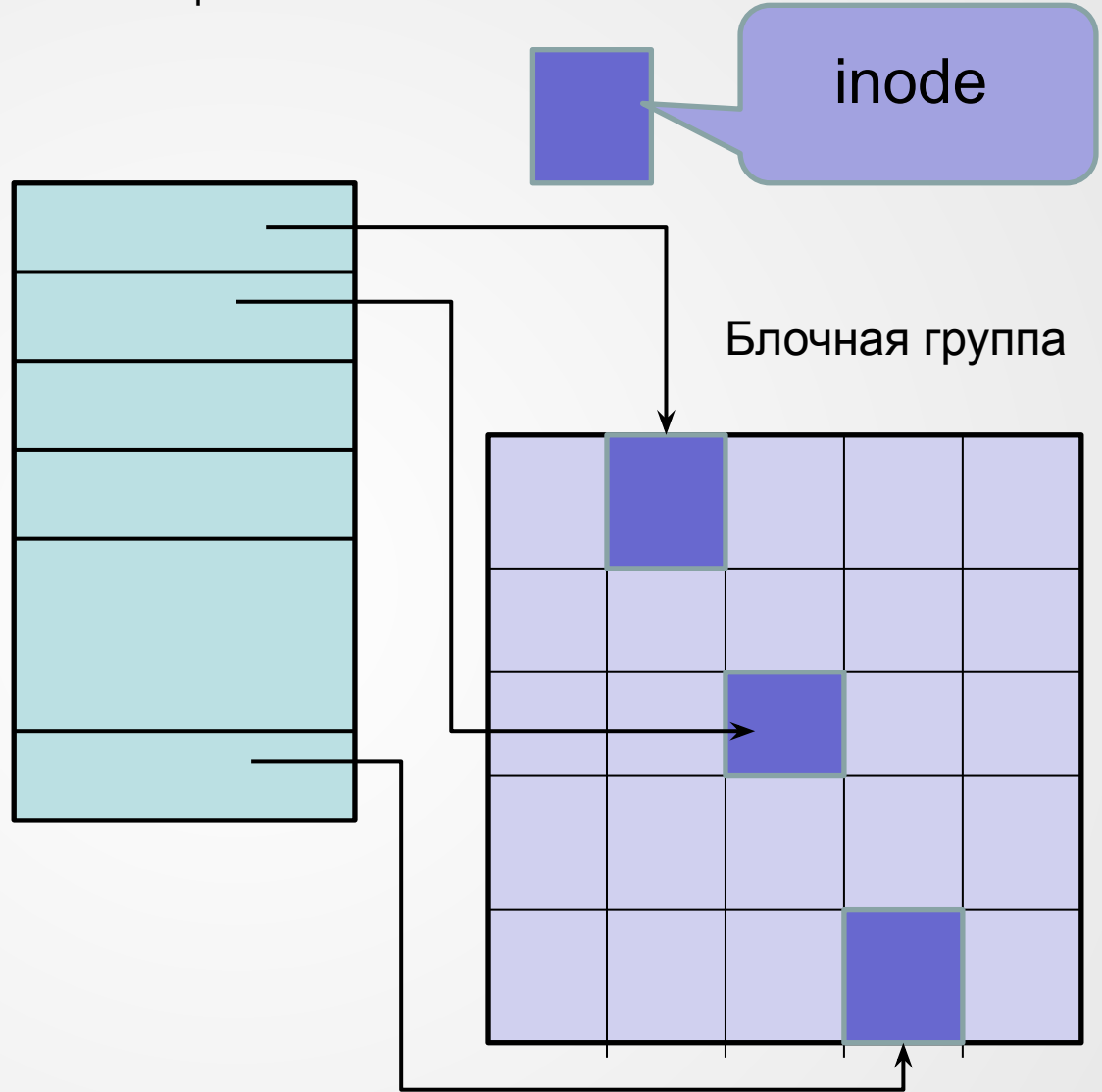
# Запись в глобальной дескрипторной таблице

- Номера блоков, соответствующих местоположению битовой карты распределения блоков
- Номер блока для битовой карты расположения inode
- Номер блока в таблице inode
- Число свободных блоков в группе
- Число inode, содержащих каталоги

# Дескриптор группы

Битовая карта распределения inode
Битовая карта распределения блоков
Таблица inode
Количество свободных блоков
Количество свободных inode

# Таблица inode



# Элемент каталога

- Номер inode
- Длина элемента каталога
- Длина имени файла
- Тип файла
- Имя файла

# Ограничения ext2

- Мах размер файла: 16 Гб – 2 ТБ (в зависимости от размера блока)
- Мах число файлов:  $10^{18}$
- Мах длина имени файла: 255 байт
- Мах размер тома: 2 – 32 ТБ
- Точность хранения даты: 1 секунда



# Файловая система ext3

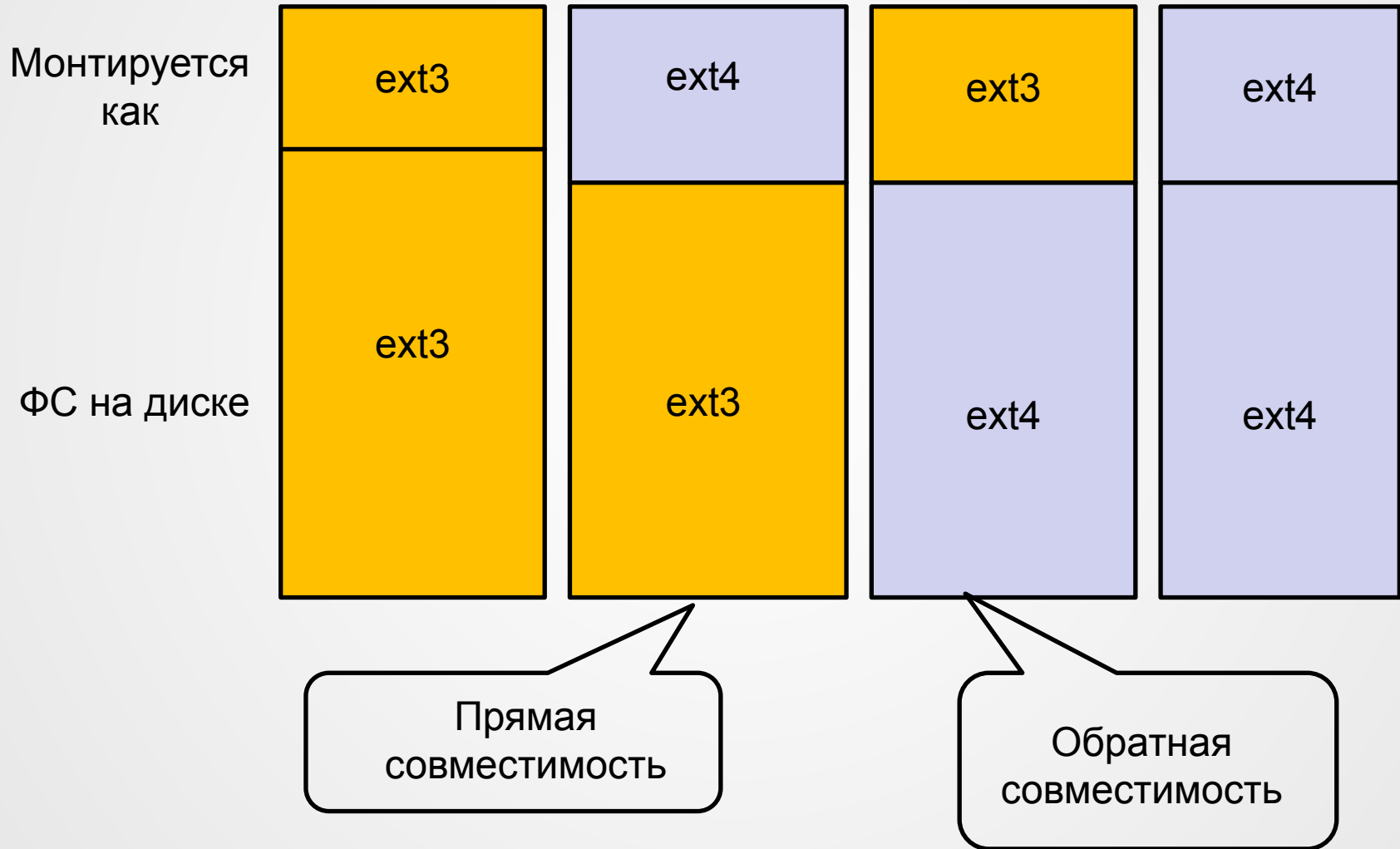
- Режимы журналирования:
  - `writeback` – в журнал записываются только метаданные файловой системы;
  - `ordered` – запись в файл производится до записи об изменении этого файла;
  - `journal` – полное журналирование, как метаданных, так и пользовательских данных.

```
mount /dev/hda6 /mnt/disc_C -t ext3 -o data <режим>
```

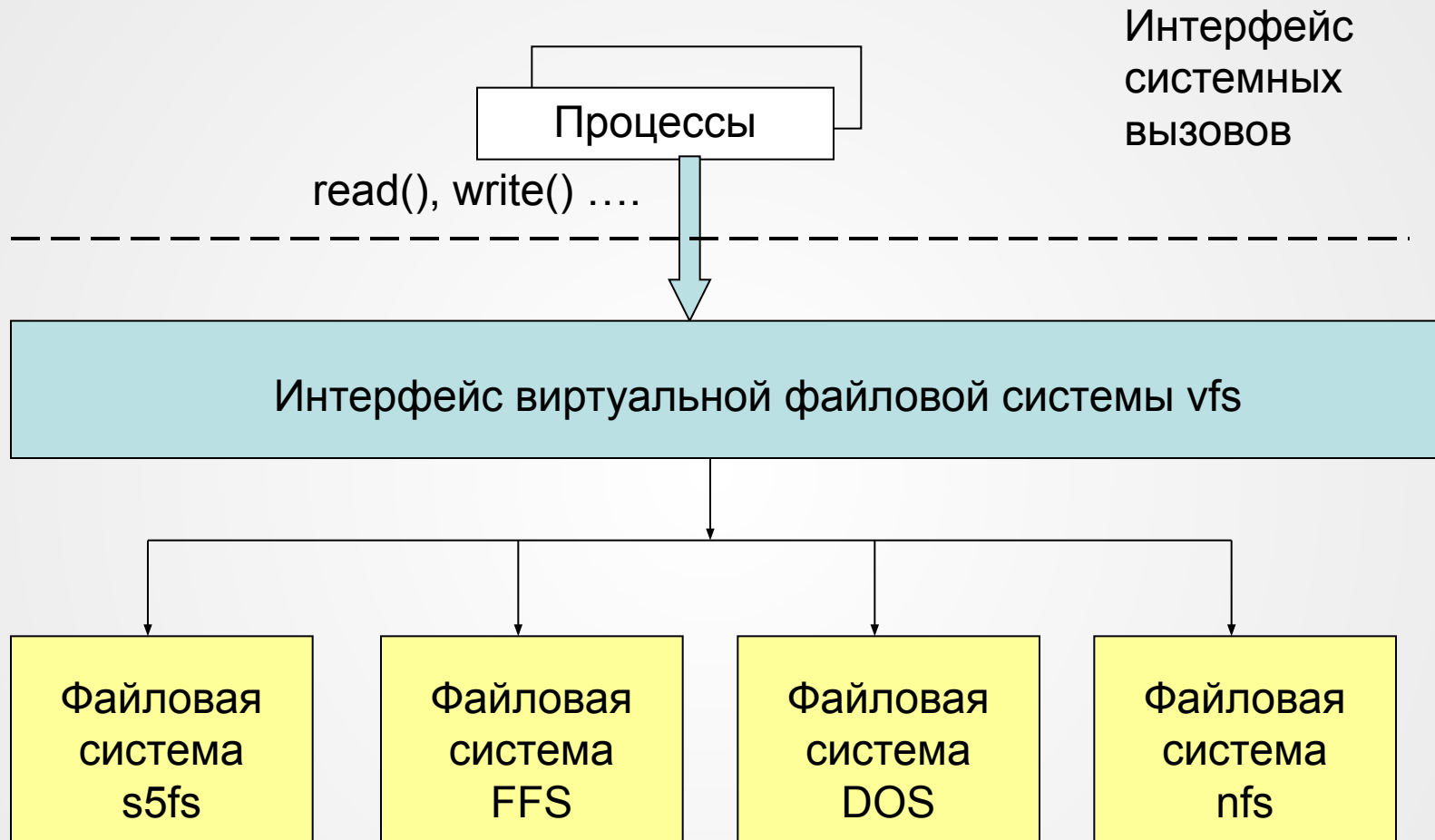
# Ограничения размеров

<b>Размер блока</b>	<b>Максимальный размер файла</b>	<b>Максимальный размер файловой системы</b>
1024	16 ГБ	До 2 ТБ
2048	256 ГБ	До 4 ТБ
4096	2 ТБ	До 8 ТБ
8192	2 ТБ	До 16 ТБ

# Файловая система ext4



# Виртуальная файловая система (vfs)

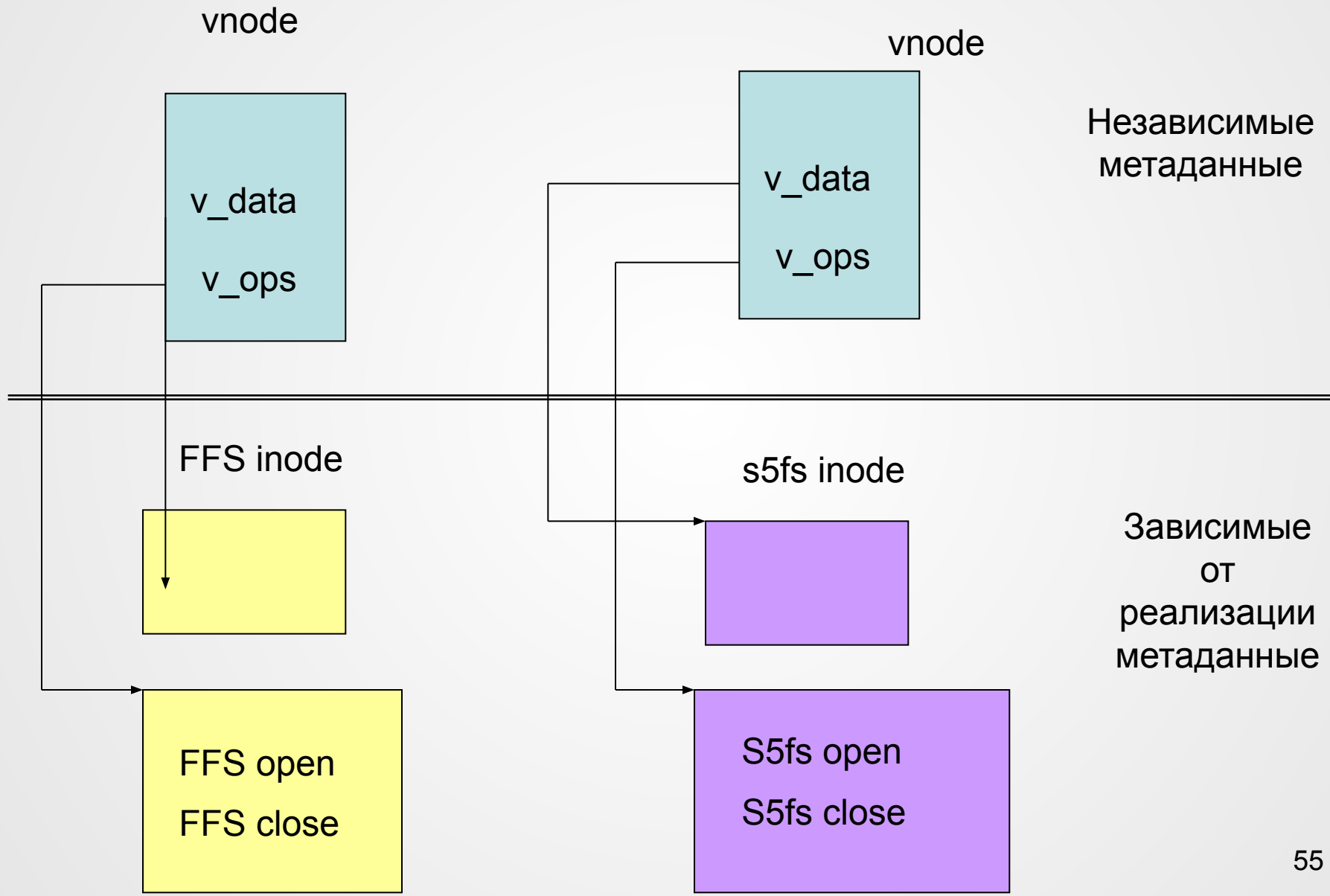


# Основные поля vnode

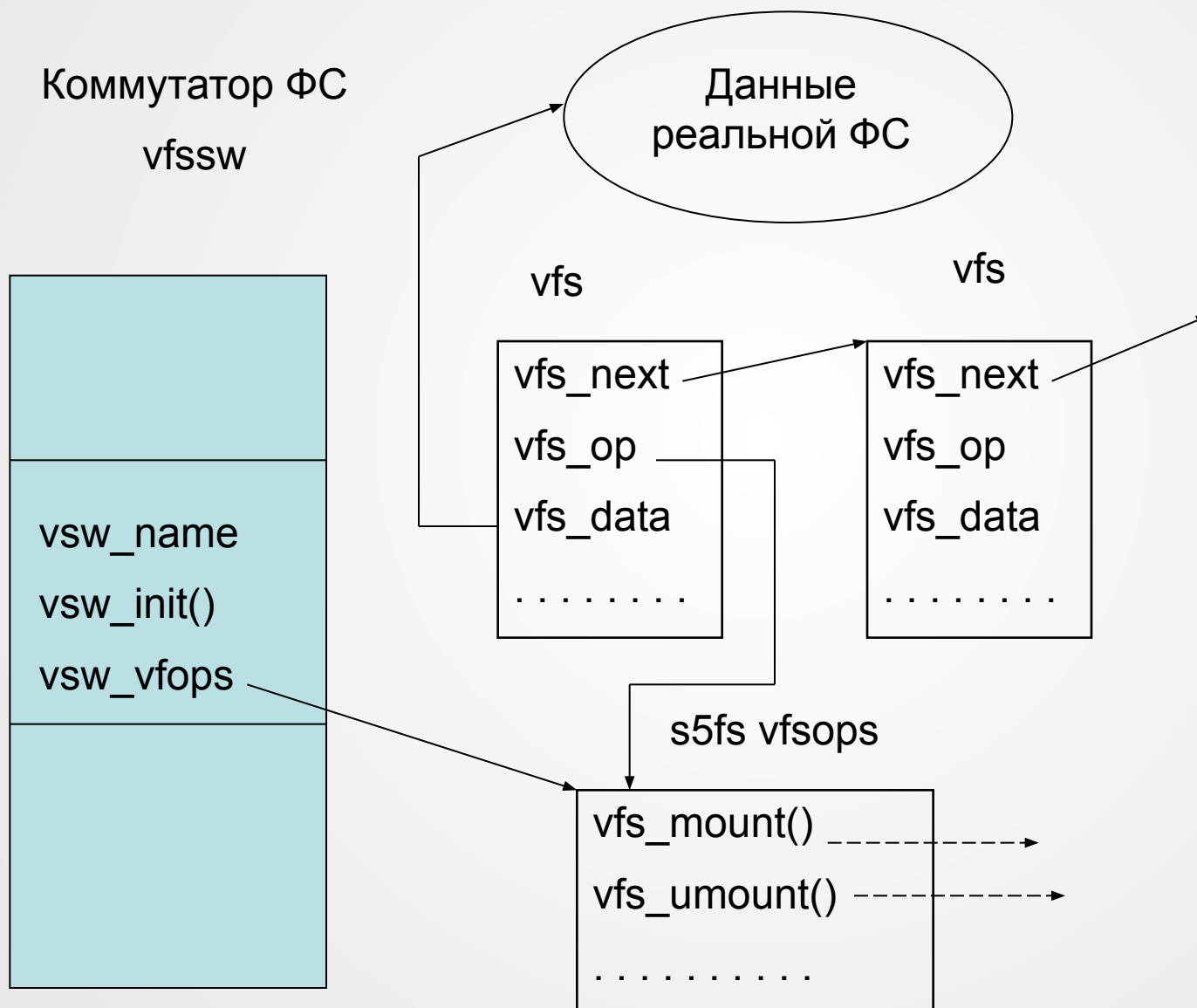
Поле	Описание
<code>u_short vflag</code>	Флаги vnode
<code>u_short v_count</code>	Число ссылок на vnode
<code>struct filock *v_filocks</code>	Блокировки файла
<code>struct vfs *v_fsmountedhere</code>	Указатель на подключаемую файловую систему, если это точка монтирования
<code>struct vfs *v_vfsp</code>	Указатель на файловую систему, в которой находится файл
<code>enum vtype v_type</code>	Тип vnode: обычный файл, каталог, файл устройства, символическая связь, сокет
<code>caddr_t v_data</code>	Указатель на данные, относящиеся к реальной файловой системе
<code>struct vnodeops *v_op</code>	Операции vnode

# Операции vnode

(*vn_open)()	Открыть vnode
(*vn_close)()	Закрыть vnode
(*vn_read)()	Чтение данных из файла
(*vn_write)()	Запись данных в файл
(*vn_ioctl)()	Задание управляющей команды
(*vn_getaddr)()	Получить атрибуты vnode
(*vn_setaddr)()	Установить атрибуты vnode
(*vn_access)()	Проверить права доступа к файлу
(*vn_lookup)()	Произвести трансляцию имени файла в соответствующий ему vnode
(*vn_create)()	Создать новый файл
(*vn_remove)()	Удалить имя файла в указанном vnode каталоге



# Монтирование файловых систем





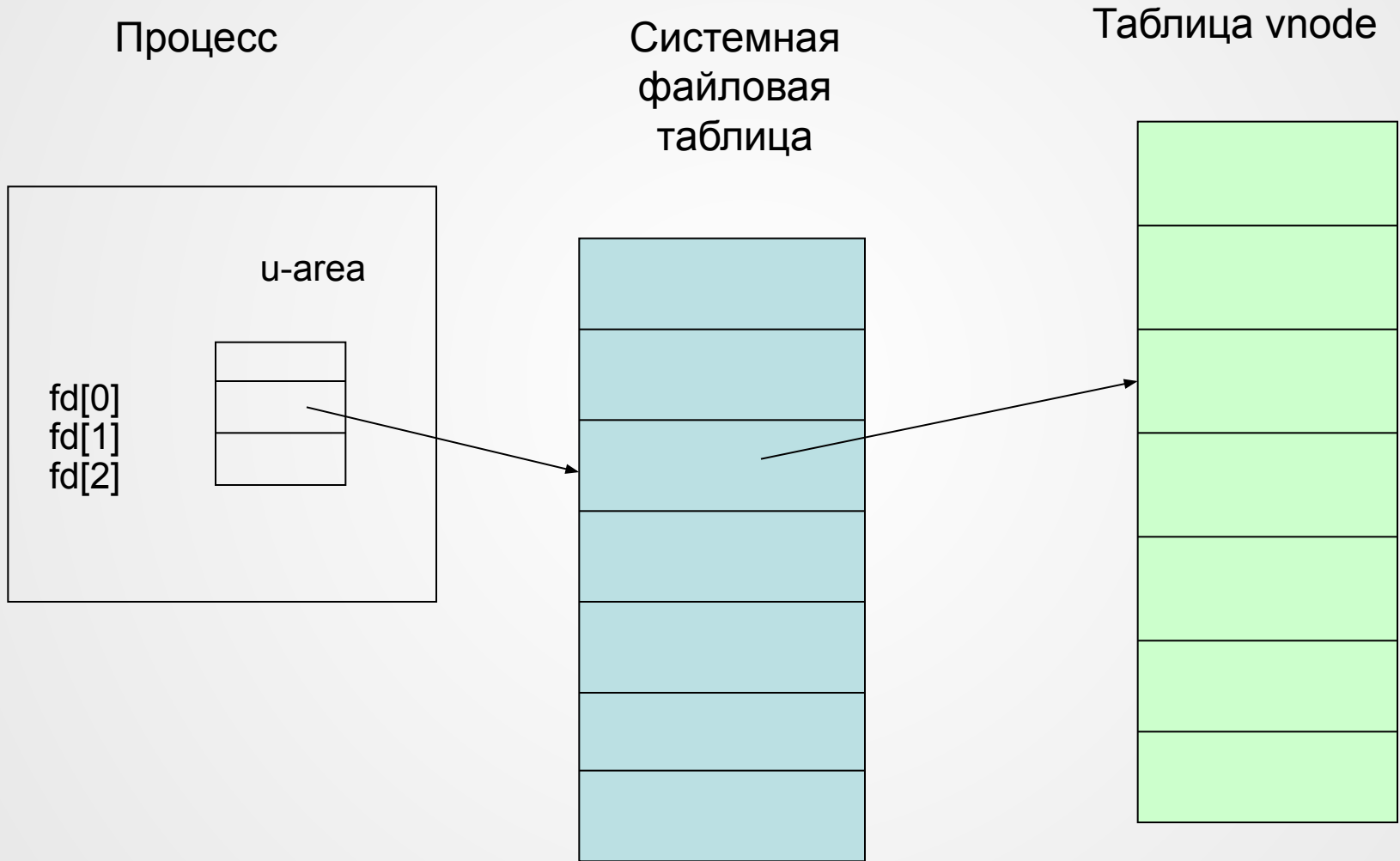
# Структура vfs

<code>struct vfs *vfs_next</code>	Следующая файловая система в списке
<code>struct vfsops *vfs_op</code>	Операции файловой системы
<code>struct vnode *vfs_vnjdecovered</code>	vnode, перекрываемой файловой системы
<code>int vfs_flag</code>	Флаги доступа
<code>int vfs_bsize</code>	Размер блока файловой системы
<code>caddr_t vfs_data</code>	Указатель на специфические данные, относящиеся к реальной файловой системе

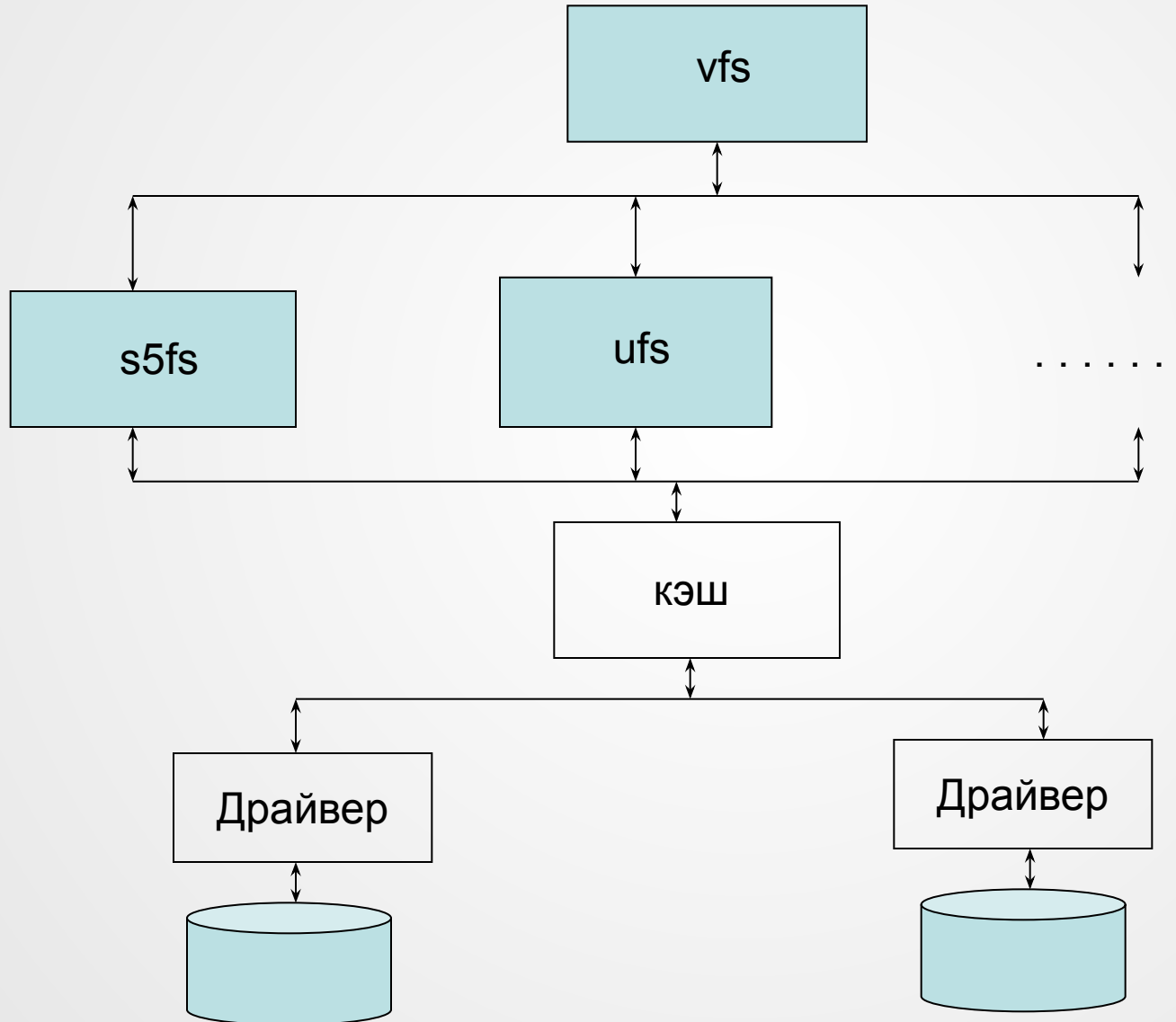
# Операции файловой системы

<code>(*vfs_mount)()</code>	Подключает файловую систему
<code>(*vfs_umount)()</code>	Отключает файловую систему
<code>(*vfs_root)()</code>	Возвращает корневой vnode файловой системы
<code>(*vfs_statfs)()</code>	Возвращает общую информацию о файловой системе
<code>(*vfs_sync)()</code>	Актуализирует все кэшированные данные файловой системы
<code>(*vfs_fid)()</code>	Возвращает файловый идентификатор в данной файловой системе
<code>(*vfs_vget)()</code>	Возвращает указатель на vnode для файла данной файловой системы

# Доступ к файлу



# Буферный кэш



# Заголовок буфера

Номер устройства
Номер блока
Поле состояния
Указатель на область данных
Указатель на следующий буфер в очереди
Указатель на предыдущий буфер в очереди
Указатель на следующий буфер в списке свободных
Указатель на предыдущий буфер в списке свободных

# Файловая система NTFS

- Поддержка больших файлов и больших дисков до  $2^{64}$ Кб
- Восстанавливаемость после сбоев и отказов программ и аппаратуры
- Высокая скорость операций
- Низкий уровень фрагментации
- Гибкая структура, допускающая развитие за счет добавления новых типов записей и атрибутов файлов с сохранением совместимости с предыдущими версиями
- Устойчивость к отказам дисковых накопителей
- Поддержка длинных символьных имен
- Контроль доступа к каталогам и отдельным файлам

# Основные единицы хранения

- **Сектор.** Физическая запись (512 байт).
- **Кластер.** Один или несколько последовательных секторов на одной дорожке. Количество секторов в кластере равно степени двойки. Возможный диапазон размера кластера от 512 байт до 64Кбайт.
- **Том.** Логический раздел диска. Том может занимать весь диск, его часть или включать в себя несколько дисков.

# Структура тома





# Метафайлы NTFS

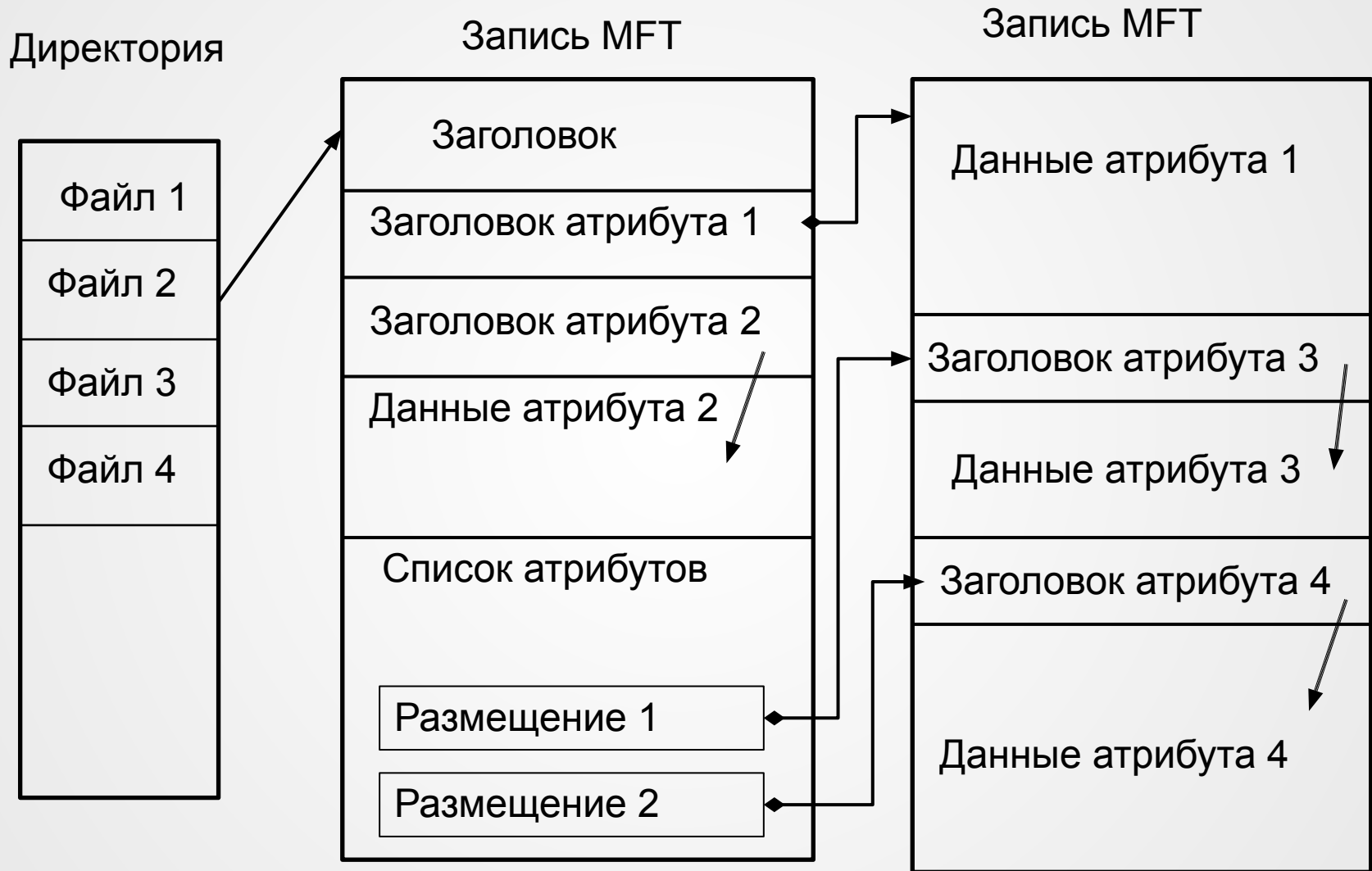
Системный файл	Имя файла	Назначение файла
Главная таблица файлов	\$Mft	Содержит полный список файлов
Копия главной таблицы файлов	\$MftMirr	Зеркальная копия первых трех записей MFT
Файл журнала	\$LogFile	Список транзакций, используемых для восстановления файловой системы
Том	\$Volume	Имя тома, версия NTFS
Таблица определения атрибутов	\$AttrDef	Таблица имен, номеров и описаний атрибутов
Индекс корневого каталога	\$.	Корневой каталог
Битовая карта кластера	\$Bitmap	Разметка использованных кластеров тома
Загрузочный сектор раздела	\$Boot	Адрес загрузочного сектора раздела
Файл плохих кластеров	&BadClus	Файл, содержащий список плохих кластеров
Таблица квот	\$Quota	Квоты используемого пространства на диске для каждого пользователя
Таблица преобразований регистра символов	\$UpCase	Преобразование регистра для кодировки Unicode

# Системный набор атрибутов

- Attribute List (список атрибутов) – ссылки на номер записи MFT, где расположен каждый атрибут, используется, если атрибуты файла не умецаются в одной записи MFT.
- File Name (имя файла) – имя файла в Unicode, а также номер входа в таблице MFT
- MS-DOS Name (имя MS-DOS) – имя файла в формате MS DOS
- Version (версия) – номер последней версии файла
- Security Descriptor (дескриптор безопасности) – информация о защите файла
- Volume Version (версия тома) – используется только в системных файлах тома
- Volume Name (имя тома) – имя тома
- Data (данные) – содержит обычные данные файла
- MFT bitmap (битовая карта MFT) – карта использования блоков на томе

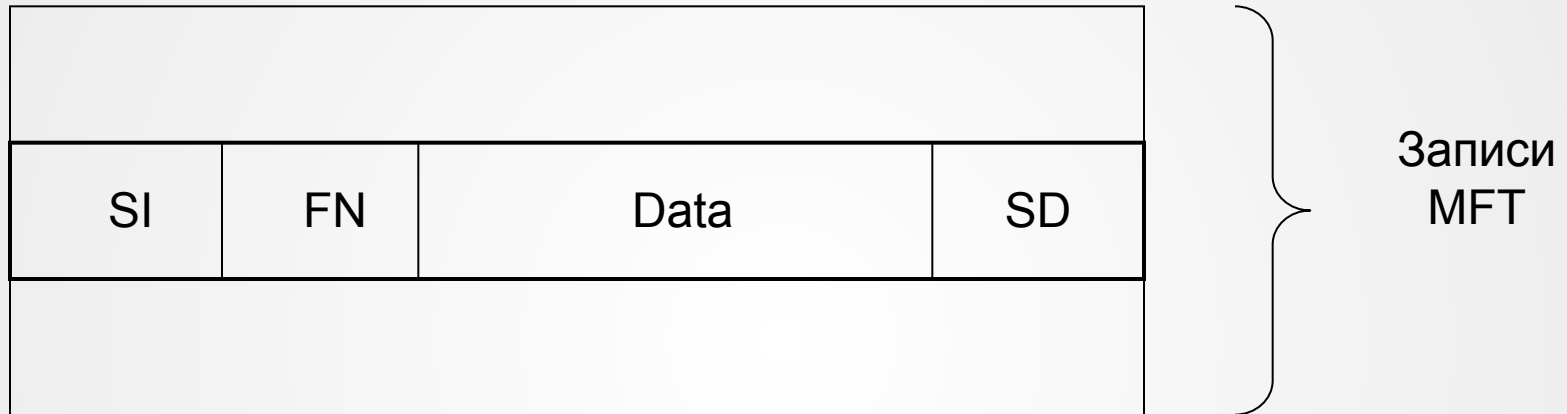
- Index Root (корень индекса) – корень B-дерева, используемого для поиска файлов в каталоге
- Index Allocation (размещение индекса) – нерезидентные части индексного списка B-дерева
- Standard Information (стандартная информация) – время создания, время обновления и т.д.

# Пример записи MFT



# Типы файлов

Небольшой файл (small) размер менее 1500 байт



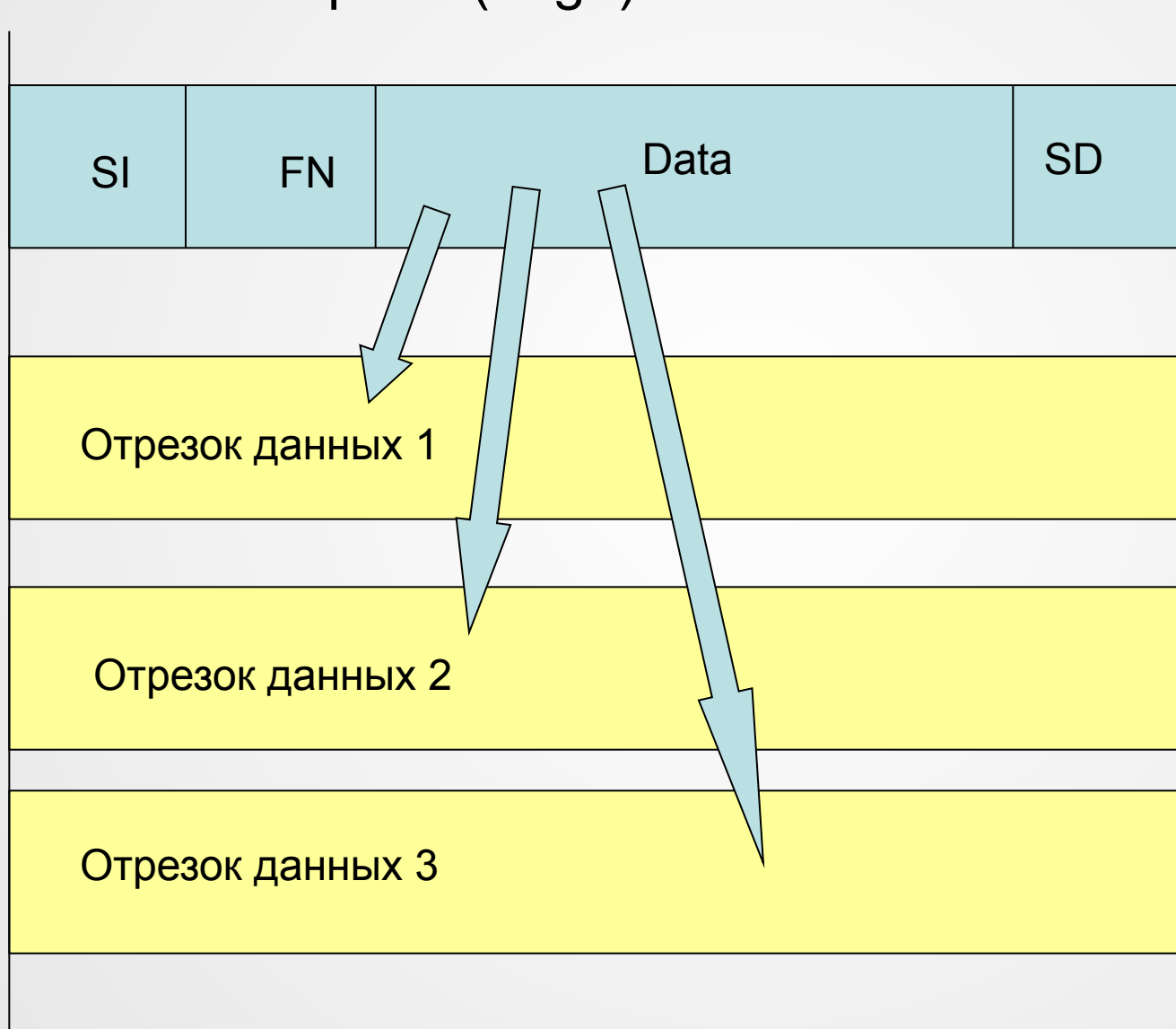
SI – стандартная информация

FN – имя файла

Data – данные

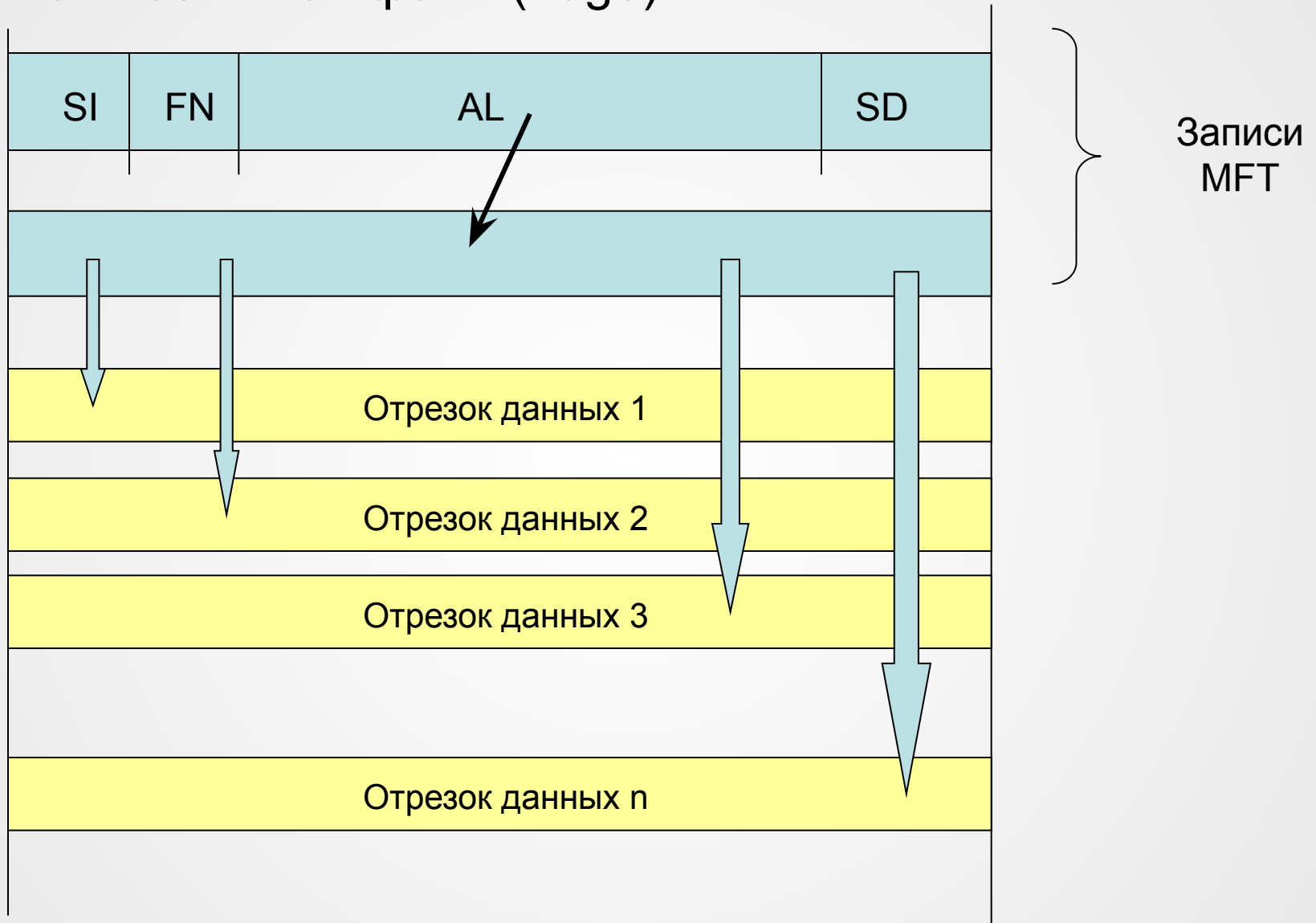
SD – дескриптор безопасности

# Большой файл (large)

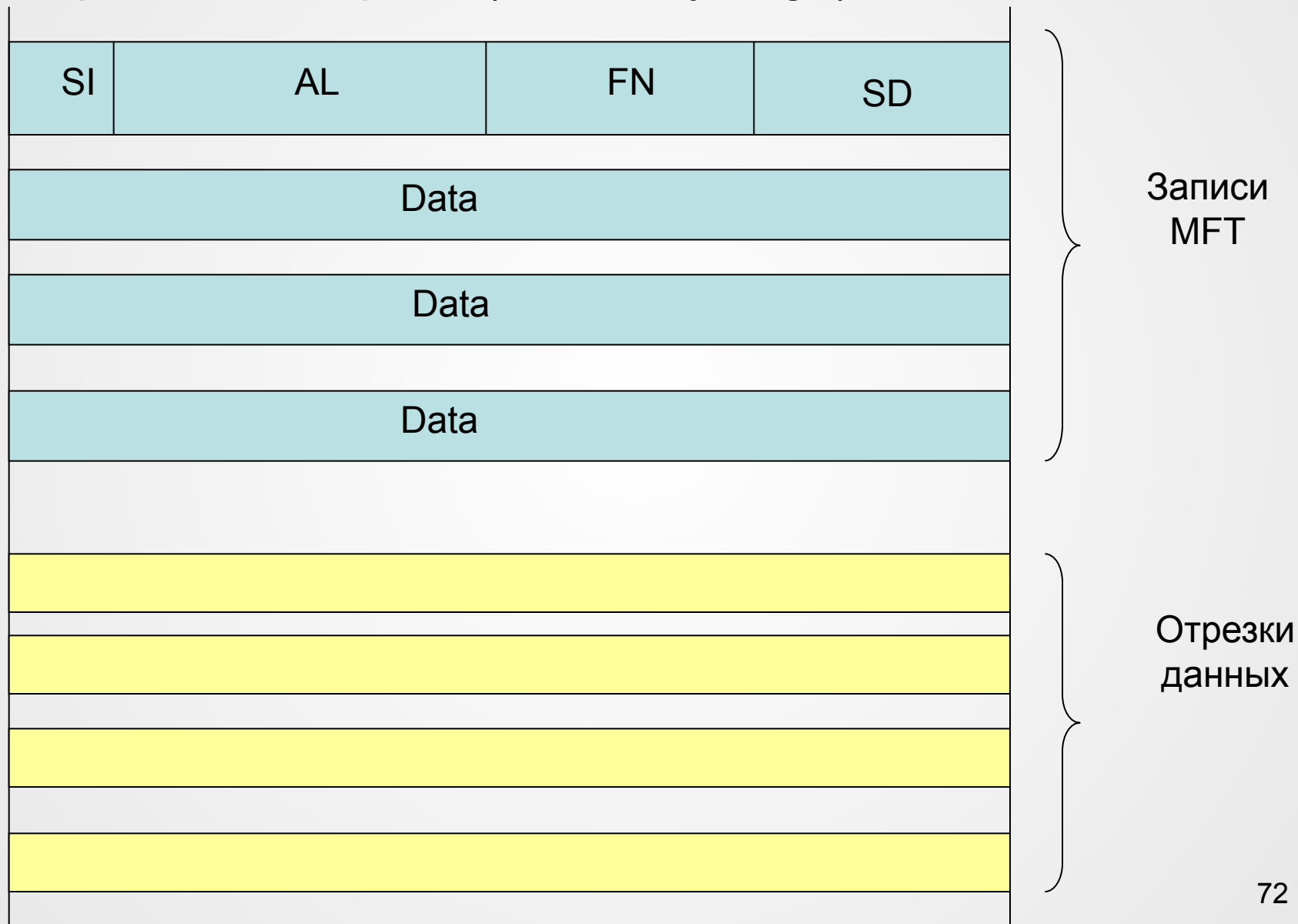


Запись  
MFT

# Очень большой файл (huge)



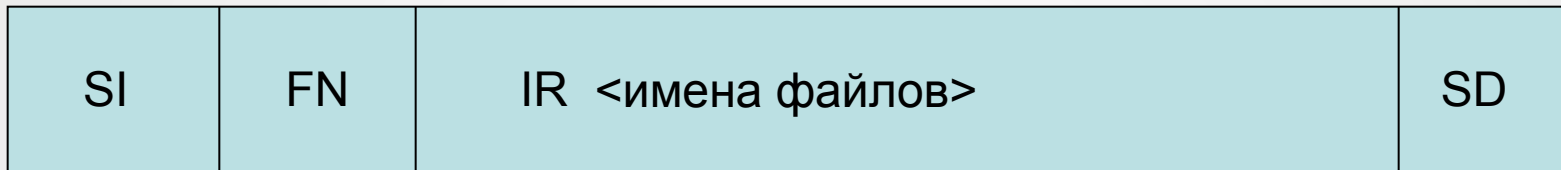
# Сверхбольшой файл (extremely huge)



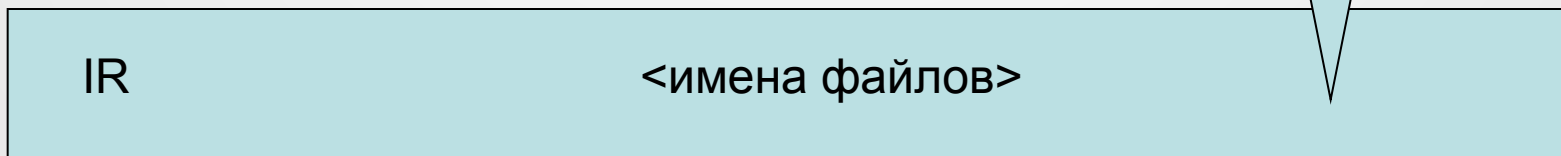
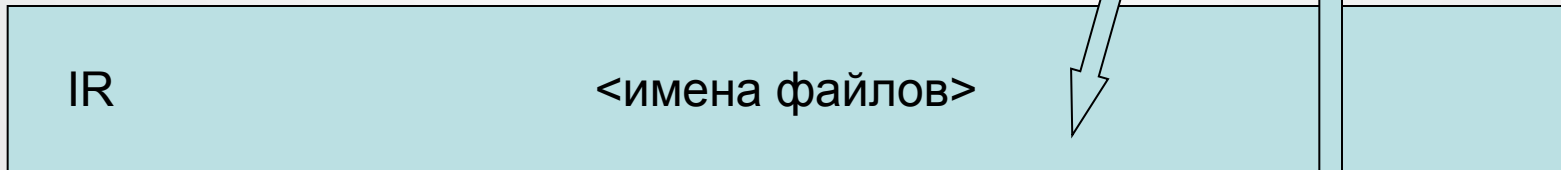
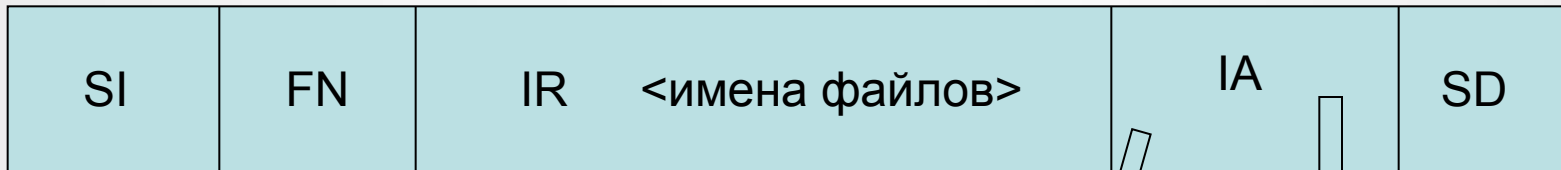


# Каталоги

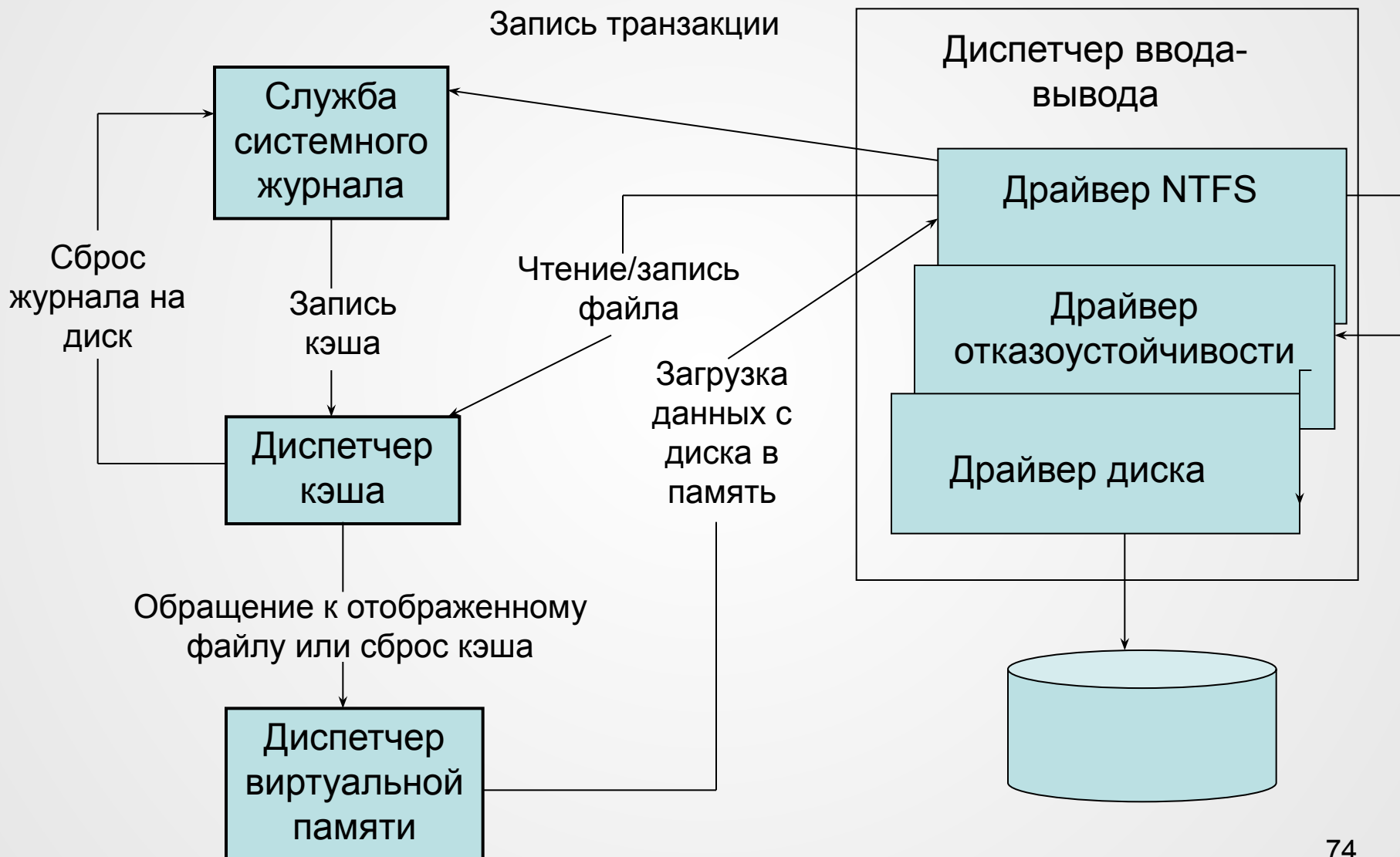
Небольшой каталог



Большой каталог



# Система восстановления данных



# Процедура восстановления данных

1. NTFS вызывает системный журнал для записи в него в кэш-памяти всех транзакций модифицирующих структуру тома
2. NTFS модифицирует том в кэш-памяти
3. Диспетчер кэш-памяти сбрасывает системный журнал на диск
4. После сброса системного журнала диспетчер кэш-памяти перемещает изменения тома на диск