Файловая система

Часть 1. Хранение и доступ к данным

Общие сведения

Термин «файловая система» имеет два значения:

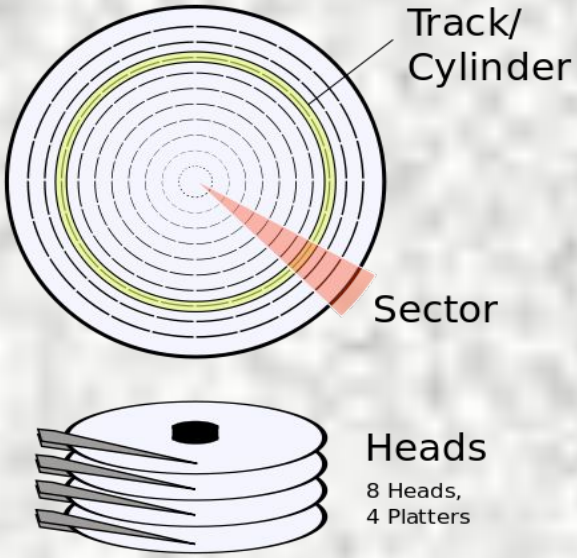
- способ хранения и организации доступа к информации, хранящей-ся на внешнем запоминающем устройстве (ВЗУ);
- компонент операционной системы, реализующий весь комплекс действий по работе с информацией, хранящейся на ВЗУ (распределение внешней памяти, контроль прав доступа, запись, удаление и т.д.)

В этой части курса будем рассматривать файловую систему как способ хранения и организации доступа.

Файл – поименованная совокупность данных на ВЗУ. При записи для каждого файла совместно с основным потоком данных создается и сохраняется набор метаданных (имя, атрибуты, размер, дата и время создания или последнего изменения, адрес и т.д.).

Файлы могут быть исполняемыми и неисполняемыми. Исполняемый файл содержит команды для центрального процессора (программа) или для ОС (командный файл, сценарий). Неисполняемые файлы содержат данные, обрабатываемые различными программами.

Физическая модель внешней памяти



Дорожка- концентрическая окружность на магнитной *поверхности*.

Сектор - участок дорожки МД, хранящий мини-мальную порцию информации, которая может быть считана или записана за одно обращение к диску.

Цилиндр – совокупность *д*орожек с одинаковы-ми номерами на различных *поверхностях* диска

Дорожки нумеруются в пределах поверхности, а сектора – в пределах дорожки. Стандартный размер сектора - 512 байт (0,5 Кбайт).

Для получения доступа к информации, хранящейся на диске, необхо-димо указать *физический адрес*, который включает номер цилиндра, но-мер головки и номер сектора *(CHS).*

Физическая модель используется контроллером диска.

Конструкция жесткого диска

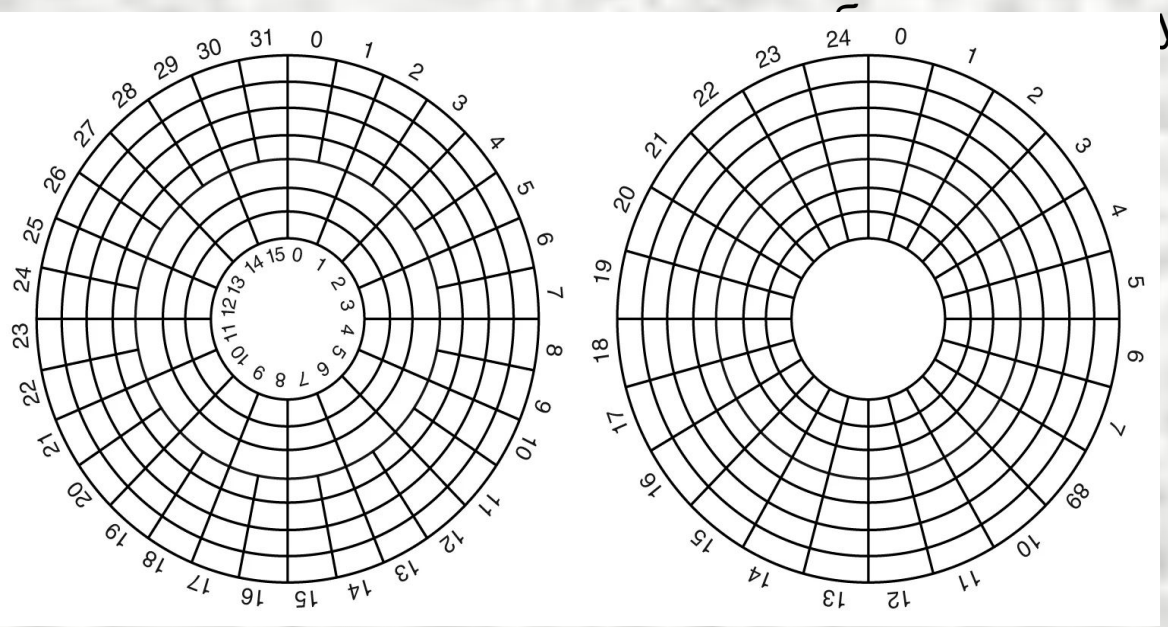
Схема жесткого диска

Фото жесткого диска Samsung

Разбиение дорожек на сектора

Возможны два способа разбивки дорожек на сектора – с постоянным количеством сек-торов на дорожке и с переменным количеством секторов на дорожке. В первом способе каждая дорожка диска имеет одинаковое число секторов, что обеспечивается **изменени-ем плотности записи** при переходе с одной дорожки на другую. Максимальная плотность записи используется на дорожках с минимальным радиусом. Для получения доступа к произвольному сектору диска необходимо указать его *физический адрес (CHS).* Такой способ адресации использовался только для дисков небольшого объема (до 500 Мбайт).

Современные диски большого объема используют второй способ, при котором дорожки на каждой поверхности диска группируются в зоны. Все дорожки в одной зоне имеют одинаковое количество секторов, дорожки в зонах с меньшим радиусом



том и плотность записи, т.е. без изменения технологии производства увеличивается общее число секторов на диске и его объем. При этом используется линейная адресация всех секторов диска (LBA). В настоящее время для задания LBA-номера сектора используется 6 байтов, что даёт возможность адресовать на диске 2^{48} секторов

Логическая модель внешней памяти

С логической точки зрения все адресное пространство диска представляет собой набор последовательно пронумерованных **секторов**. Небольшая часть секторов выделяется для хранения слу-жебной информации (**системная область**), а остальные сектора предназначены для хранения файлов и каталогов и образуют **область данных**.

Минимальной единицей дисковой памяти является **блок** (клас-тер), содержащий несколько секторов. Размер блока, а также раз-мещение системной области (в смежных или несмежных блоках), определяется файловой системой. Например, в Linux размер блока обычно равен 1 Кбайт, а в Windows – 4 Кбайт.

С логической моделью диска работает операционная система.

Разбиение диска на разделы

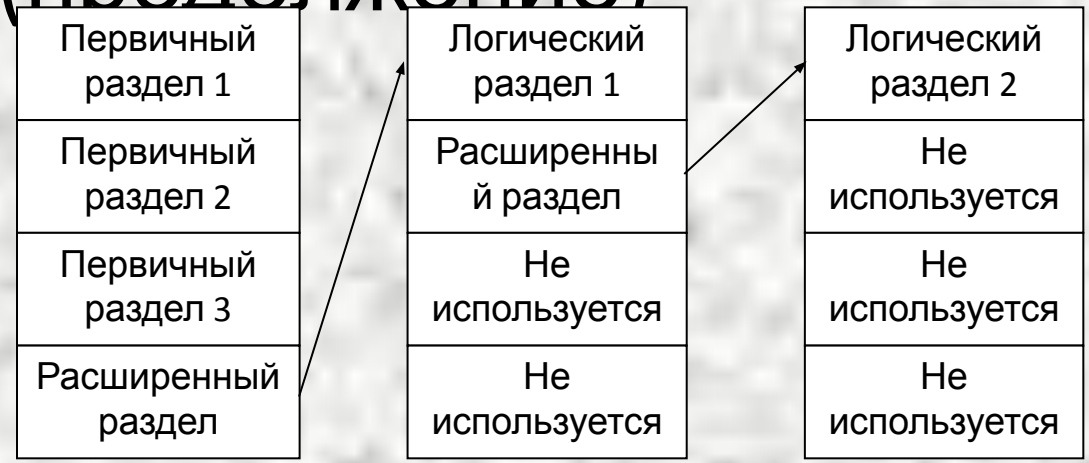
Все линейное дисковое пространство обычно делится на несколько разделов. Раздел – это часть диска, имеющая собственную файловую систему, в один раздел объединяется группа смежных блоков. Для каждого раздела на диске необходимо хранить инфор-мацию о его начале и конце.

Достоинства:

- увеличивается скорость выполнения операций чтения и записи;
- появляется возможность структурирования данных (например, можно отделить файлы пользователя от файлов ОС);
- на одном диске можно установить несколько ОС

Информация о разбиении диска хранится в главной загрузочной записи диска (MBR) в виде **таблицы разделов**, содержащей четыре записи по 16 байтов. Каждая запись может

Разбиение диска на разделы (продолжение)



Для увеличения числа разделов диска один из первичных раз-делов объявляется расширен-ным и в нем создаются логичес-кие разделы. Расширенный раздел в таблице может быть только один.

Расширенные разделы не используются для хранения данных, они могут лишь хранить информацию о логических разделах. Каждый расширенный раздел име-ет свою таблицу разделов, в которой используются только две записи, задающие один логический и один расширенный, то есть получается цепочка из таблиц разделов.

Каждый раздел имеет собственное имя, например, в Windows – c:, d:, e:, в Linux – hda1, hda2 (диски с интерфейсом IDE или Parallel ATA, PATA) или sda1, sda2 (диски с интерфейсом Serial ATA,SATA).

В последнее время начинает все чаще использоваться таблица GPT (GUID Partiti-on Table, таблица с глобальными идентификаторами). Если диск имеет таблицу разбиения GPT, в на нем по умолчанию зарезервировано место под 128 разделов, каждый из которых является первичным.

Пример разбиения на разделы

Раздел	Начало	Размер	Тип
C	0	200	первичный
D	200	200	первичный
E	400	200	первичный
	600	400	расширенный

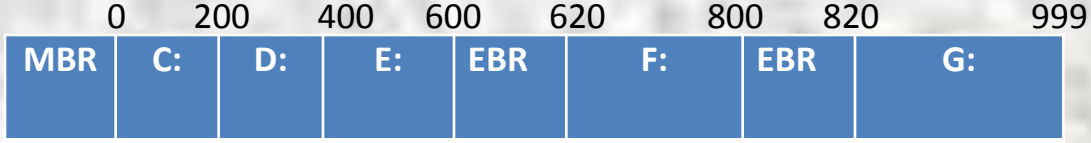
Расширенная загрузочная запись раздела F:

Раздел	Начало	Размер	Тип
F	620	180	логический
	800	200	расширенный

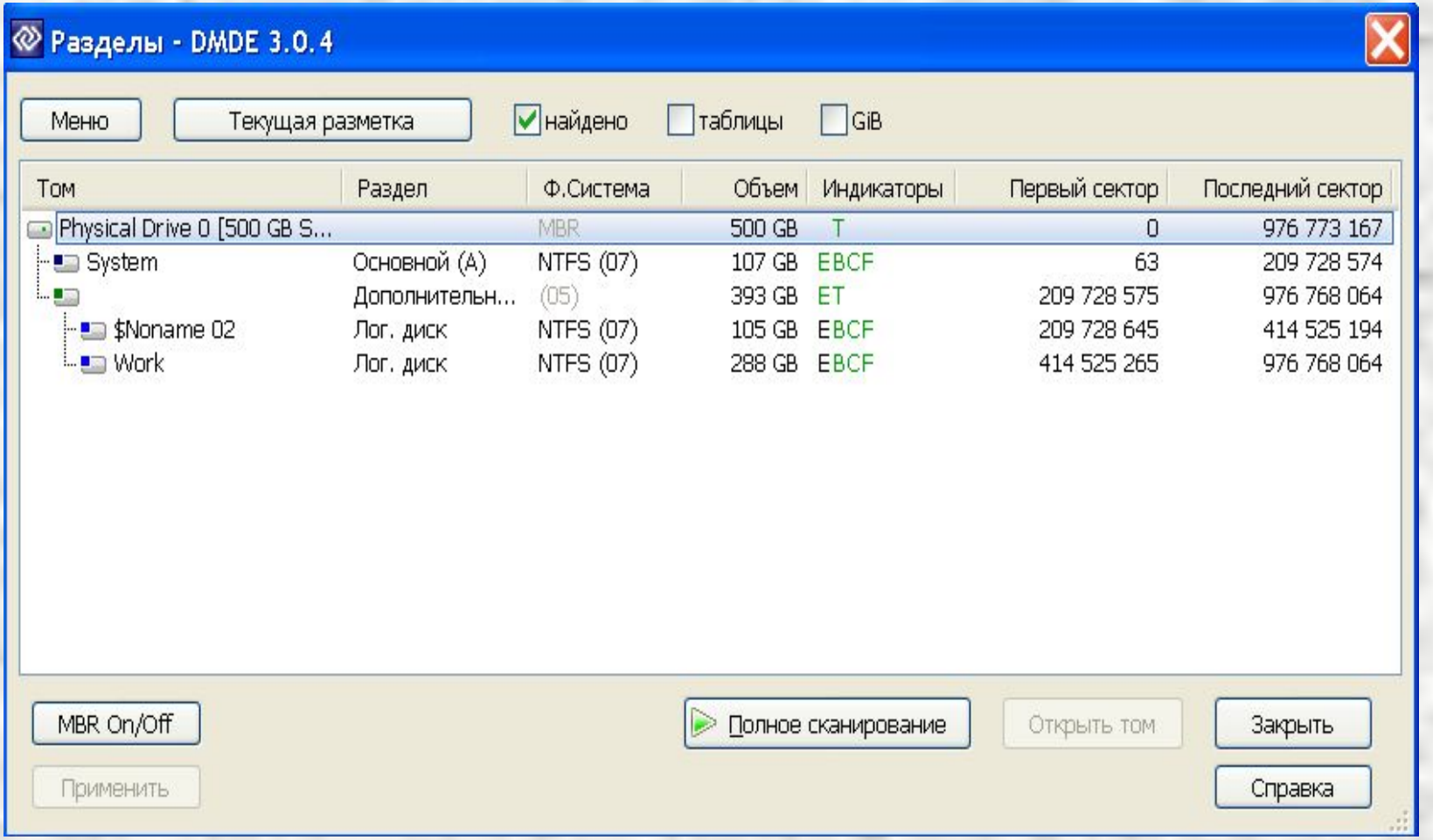
Расширенная загрузочная запись раздела G:

Раздел	Начало	Размер	Тип
G	820	180	логический

Общая структура физического диска



Пример разбиения диска на разделы



Здесь один первичный, один расширенный и два логических раздела

Объединение дисков

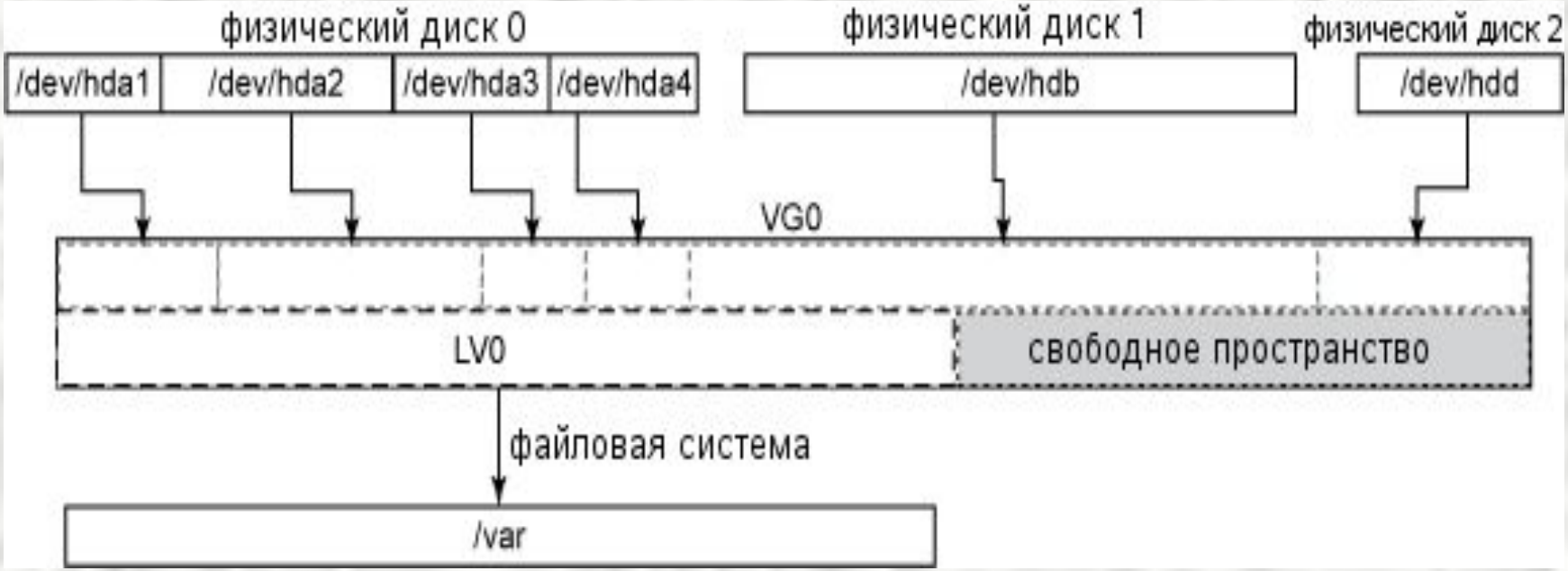
Дисковые разделы могут создаваться не только для разбиения, но и для объединения дисков. При этом на общем непрерывном дисковом пространстве может быть создана единая файловая система. Способы объединения : а) организация RAID массивов (redundant array of inexpensive disks, избыточный (резервный) массив недорогих дисков), б) применение менеджера логических томов.

Технология RAID используется для избыточности и повышения произво-дительности систем хранения данных. Например, RAID уровня 1 предна-значен для увеличения надежности хранения данных и реализуется путем зеркалирования (дублирования) дисков.

Менеджер логических томов (Logic Volume Manager, LVM) – компонент ОС, позволяющий:

- использовать разные области одного жёсткого диска и области различных жёстких дисков как один логический том.
- иметь файловую систему, которая превышает размер наибольшего диска;
- добавлять диски или разделы в дисковую группу и расширять существующие файловые системы «на лету»;

Пример использования LVM



Здесь 4 раздела физического диска hda и два физических диска (hdb и hdd) отображаются в группу томов VG0, в которой создан логический том LV0 и оставлено свободное место для других логических томов или для последующего роста LV0.

LVM может использоваться для разбиения на разделы одного физического диска как альтернатива классического метода разбие-ния с помощью расширенных разделов.

Структура дисковой памяти сервера students.ami.nstu.ru

Диск	Раздел	Тип	Размер (Гб)	Тип ФС	Точка монтир.	Имя устр-ва
sda 127 Гб	sda1	первичный	0.5	xfs	/boot	sda1
	sda2	первичный под LVM	97,7	LVM2 member		sda2
	dm-0	логический том	9,8	swap	[SWAP]	centos-swap
	dm-1	логический том	39,1	ext4	/	centos-root
	dm-2	логический том	9,8	ext4	/tmp	centos-tmp
	dm-3	логический том	39,1	ext4	/home	centos-home
	свободен		30			

Методы размещения файлов

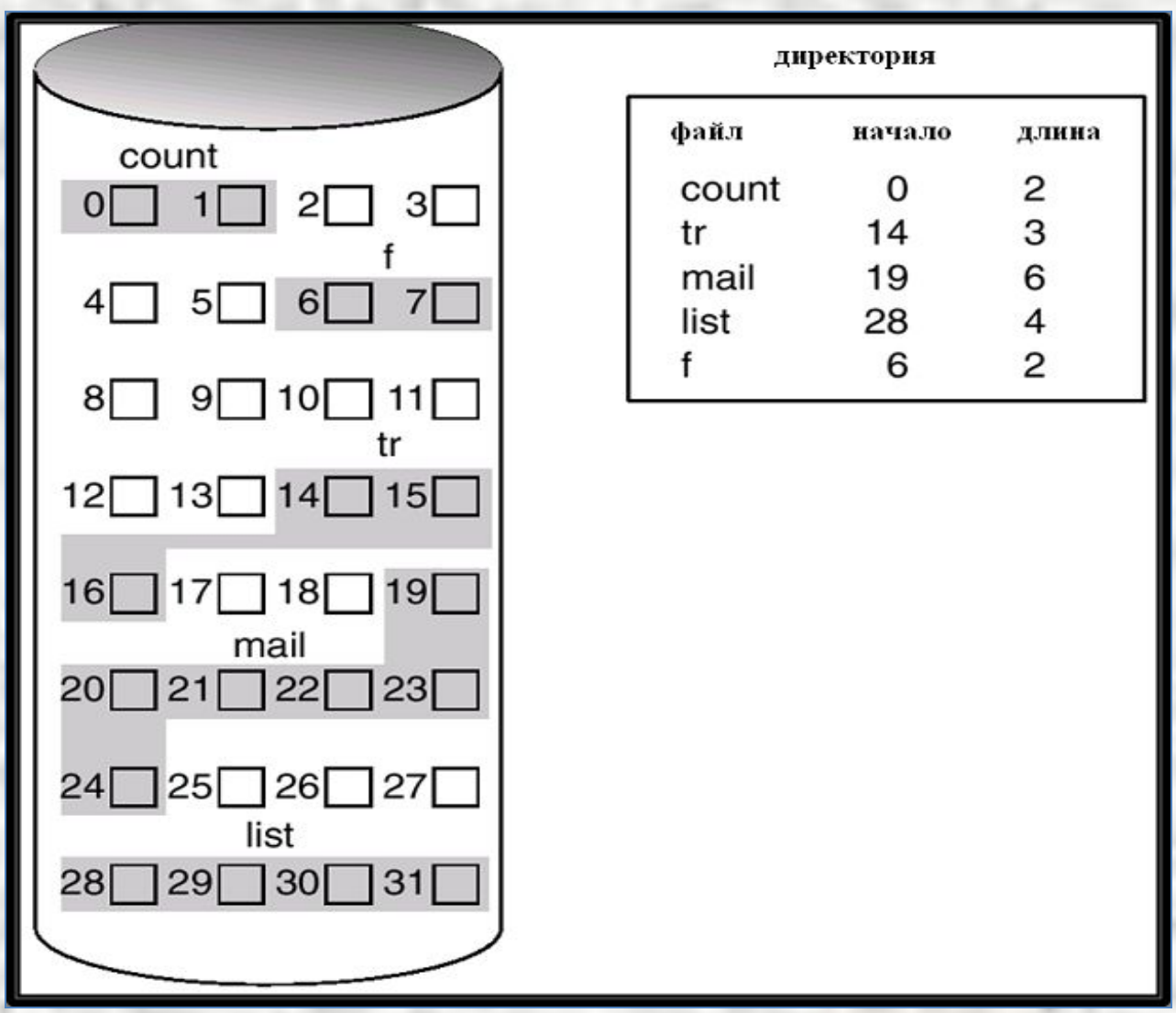
Возможны три метода размещения блоков файла на диске:

- смежное размещение;
- ссылочное размещение;
- индексированное размещение.

Смежное размещение.

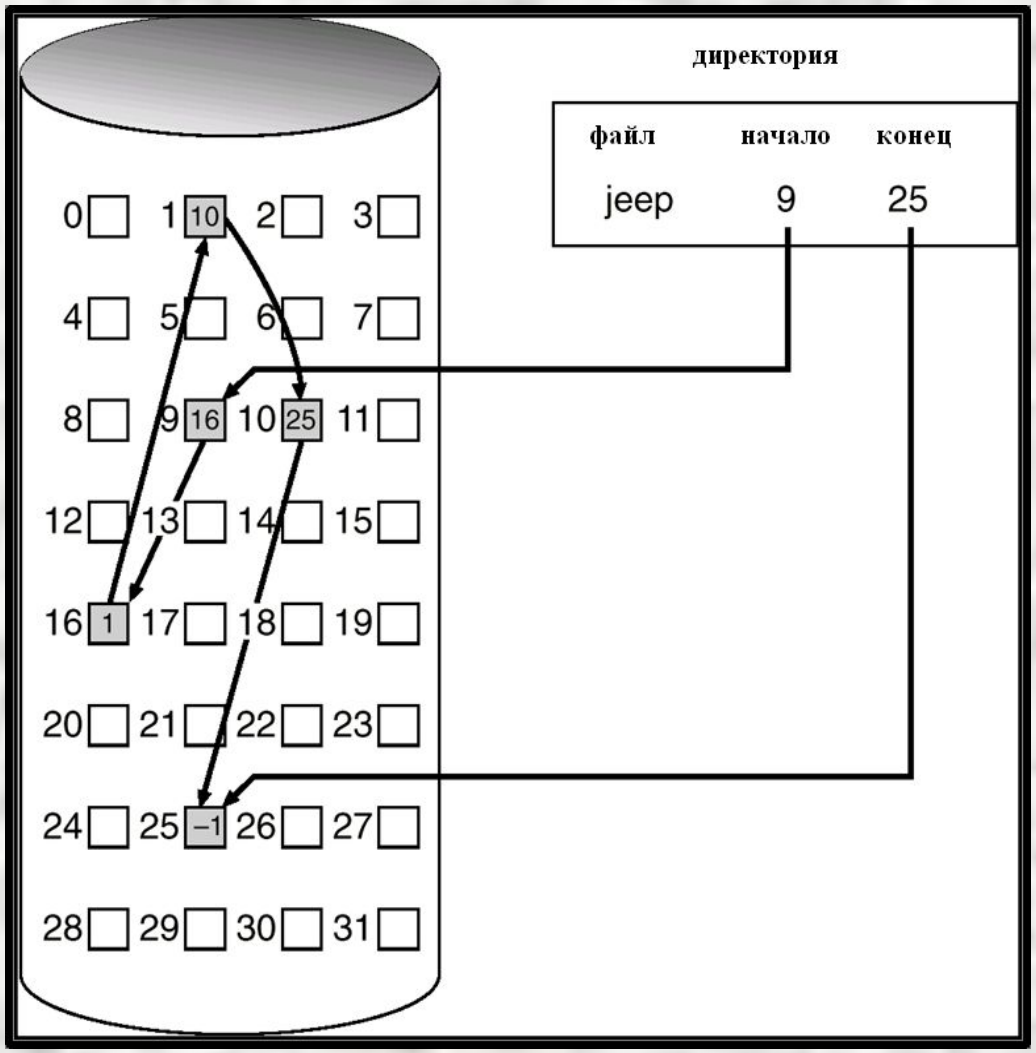
1. Каждый файл занимает набор смежных блоков на диске.
2. Достоинство – простота, т.к. в каталоге требуется хранить только одну ссылку (номер блока) и длину (число блоков).
3. Недостатки - невозможность увеличить размер файла и проблема фрагментации.
4. Возможна модификация – файлы с применением расшире-ний: дисковые блоки размещаются в расширениях (extents). Рас-ширение – это набор смежных блоков на диске, файл состоит из одного или нескольких расширений.
5. Способ использовался в старых ОС.

Смежное размещение файлов

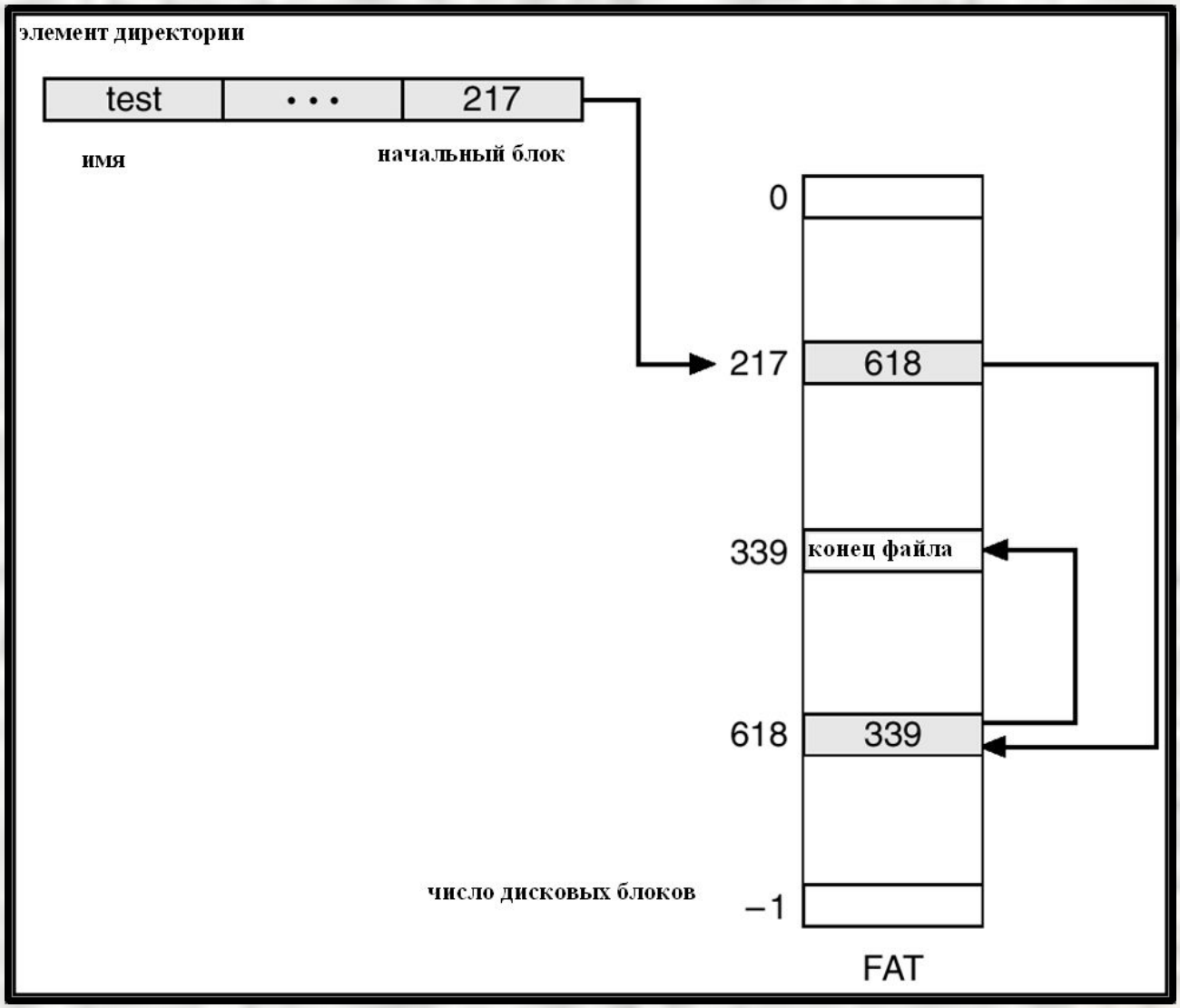


Ссылочное размещение файла

1. Каждый файл представлен в виде связанного списка дисковых блоков, которые могут быть разбросаны по диску.
2. Элементы списка могут размещаться в самих дисковых блоках или в отдельной специальной таблице размещения файлов.
3. Используется в MS DOS, MS Windows, OS/2 (файловая система FAT).
4. Достоинство – простота, т.к. необходимо хранить только начальный адрес и длину файла.



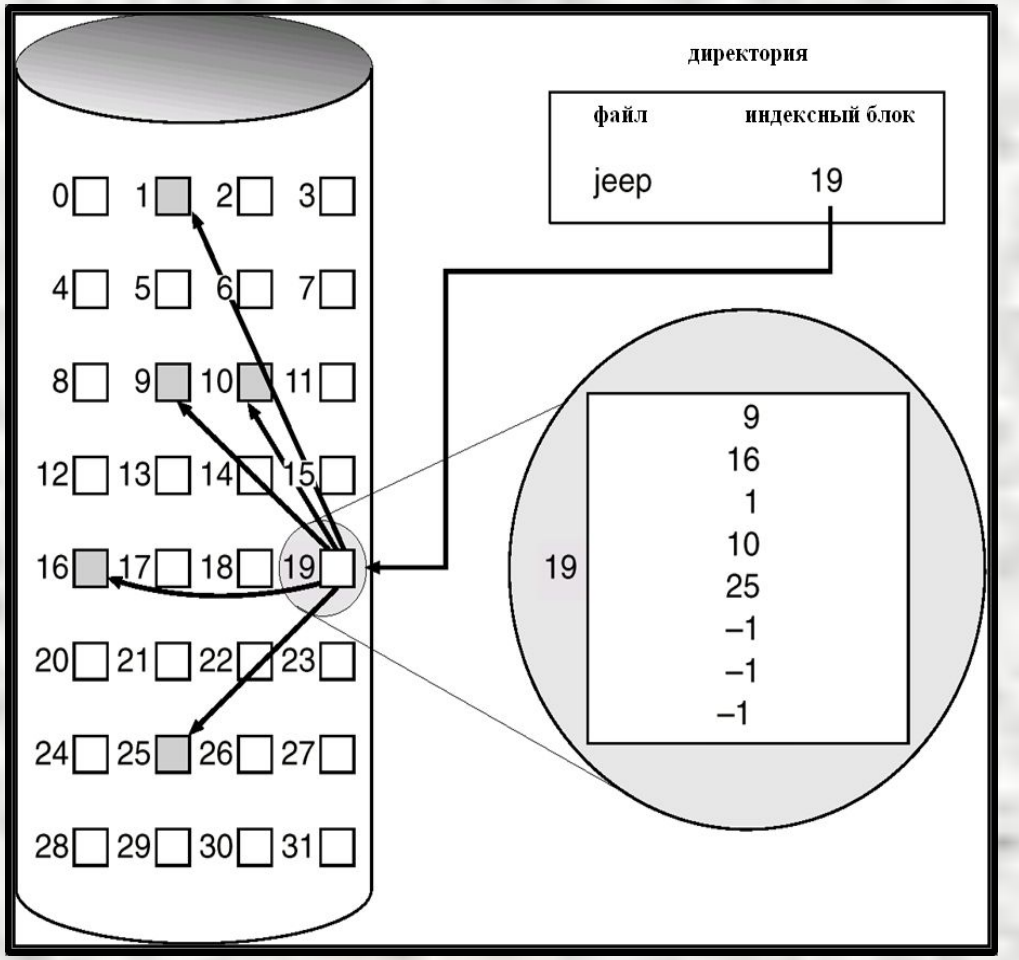
Использование таблицы размещения файлов



Здесь каждому блоку диска соответствует один элемент в таблице.

Индексированное размещение

1. Все указатели собраны вместе в индексный блок.
2. Используется индексная таблица, ссылающаяся на блоки данных файла.
3. Динамический доступ без внешней фрагментации, но ухудшается использование дисковой памяти, т. к. для каждого файла необходимо выделять индексный блок.
5. Используется в семействе ОС UNIX и в ОС Windows (NTFS).
6. Достоинство - уменьшение времени доступа.



Реализация файловых систем

1. Файловые системы ОС Linux

Особенности файловых систем Linux:

- отсутствует понятие логического диска;
- все файлы и каталоги физического диска образуют единое иерархическое дерево;
- под файлом понимается не только поименованная совокупность информации на ВЗУ, но и любое **устройство**, которое может хранить, поставлять или потреблять информацию. В этом случае устройство подключается (монтируется) к существующему дереву файловой системы в указанной пользователем точке.
- типы файлов: *обычные, каталоги, специальные (блочные и символьные), ссылки, каналы.*

Специальные блочные файлы соответствуют устройствам, на которые запись и считывание информации проводится блоками (например, ВЗУ). Символьные файлы соответствуют устройствам, взаимодействие с которыми производится посимвольно в режиме потока байтов (например, терминал).

Права доступа в ОС Linux

Каждый файл или каталог имеет права доступа. Права доступа определяют, **КТО** и **ЧТО** может делать с содержимым файла. Права доступа отображаются командой **ls -l** в виде строки, состоящей из 10 символов, например: -rwxr-xr- -

Права доступа задаются командой **chmod режим имя_файла**

Например, chmod g+x, o+x myfile1

Право	**Обозначение**	**Файл**	**Каталог**
Чтение	r	Файл можно посмотреть и скопировать	Можно посмотреть список входящих файлов
Запись	w	Файл можно изменить и переименовать	Можно создавать и удалять файлы
Выполнение	x	Файл можно запустить на выполнение (скрипты и программы)	Можно входить, делать текущим

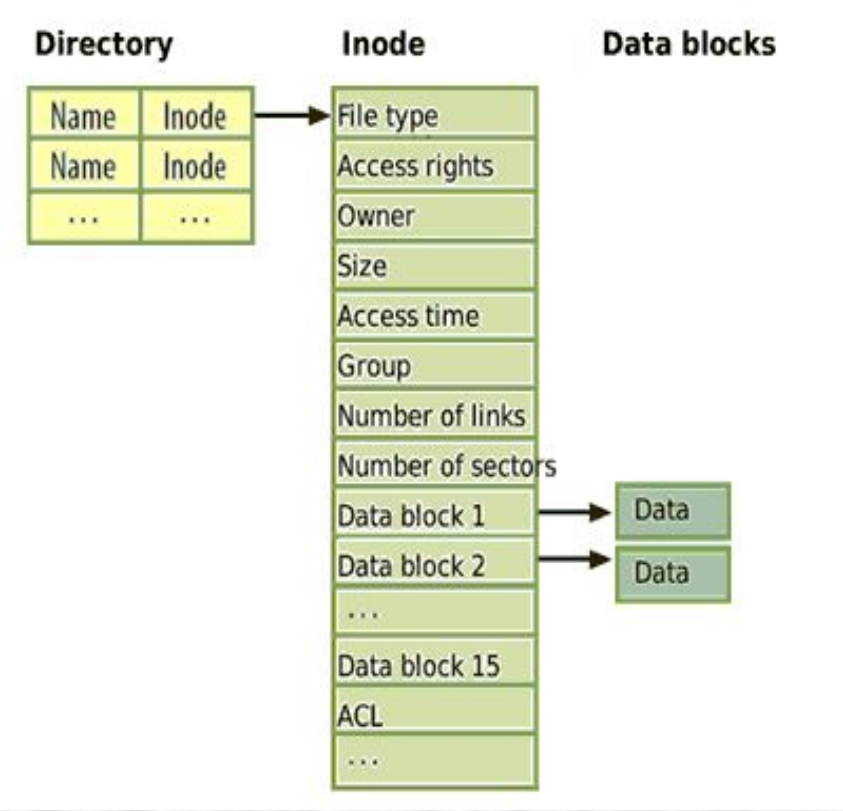
Типы файловых систем Linux

Файловые системы ОС Linux делятся на два типа – локальные и рас-пределённые (сетевые). Локальные файловые системы могут распола-гаться во внешней памяти (ext2, ext3, ext4 и др.) или в оперативной памяти (псевдо-файловые системы). К последней группе относятся:

- /proc – используется в качестве интерфейса к структурам данных в ядре; большинство расположенных в ней файлов доступны только для чтения;
- /tmpfs – позволяет не записывать на физические диски временные файлы, которые формируются в оперативной памяти, а затем удаляют-ся; поддерживает работу с виртуальной памятью;
- /devfs – предназначена для управления устройствами;
- /sysfs – используется для получения информации о всех устройствах и драйверах.

Распределенные файловые системы предназначены для объединения на логическом уровне файловых систем отдельных компьютеров в еди-ное целое. В Linux такой системой является nfs

Организация хранения данных в Linux



Файловые системы Linux используют три главных структуры данных: катало-ги, индексные дескрипторы (**inode, i-узлы**) и блоки данных (**data blocks**).

Каталоги содержат только записи имен файлов и соответствующих им номеров **inode**. При этом на один и тот же **inode** могут указывать несколько записей каталогов. Такие записи называются жесткими ссылками.

Мягкой или символической ссылкой (симлинк) называется файл, *содер-жимое* которого указывает на *имя* другого файла, а не на индексный дескриптор.

Каталоги хранятся на жестком диске как обычные файлы, отличающиеся от последних только типом файла и тем, что их содержимое представляет собой обязательную структуру.

Файловая система ext2

В первом блоке ext2 распо-лагается **загрузчик**, все остальное пространство делится на блоки равного размера (обычно 1 Кбайт). Блоки объединяются в группы.

В каждую группу входит суперблок, описатель группы, два битовых массива (для блоков и для i-узлов), i-узлы и блоки для хранения данных. **Суперблок** хранит информацию о размере группы, о количестве i-узлов и блоков данных в группе, и т.д.

Описатель группы содержит информацию о расположении битовых масси-вов, количестве свободных блоков и i-узлов в группе, а также о количестве каталогов в группе. **Битовые массивы** предназначены для учета свободных блоков и i-узлов, для хранения каждого массива выделяется один блок. При размере блока 1Кбайт размер группы равен 8192 блока и количество i-узлов равно 8192.

Файловая система ext2 (продолжение)

Ниже показана структура i-узлов, размер каждого узла составляет 128 байт

Ре-жим	Счетчик связей	UID	GID	Размер	Метки времени	Адреса блоков	Однократный косвенный блок	Двукратный косвенный блок	Трехкратный косвенный блок

Здесь обозначено:

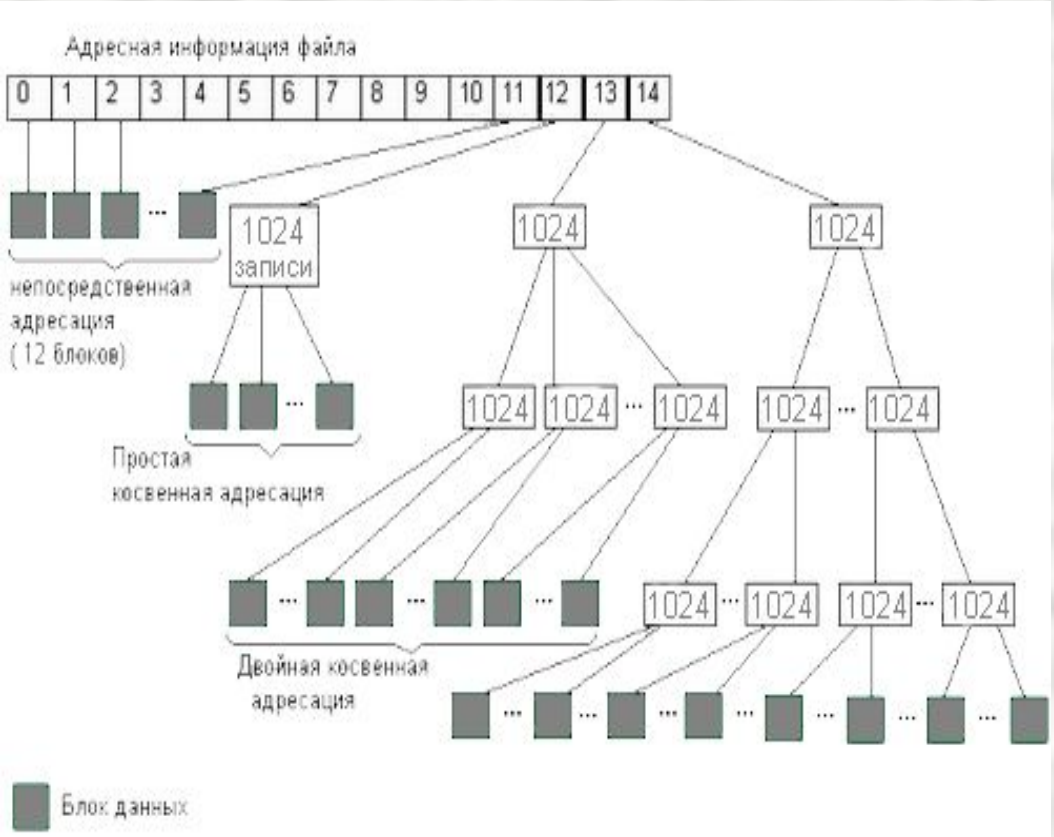
- режим – тип файла, биты защиты;
- счетчик связей – число записей каталогов, указывающих на этот i-узел;
- UID – идентификатор владельца файла;
- GID – идентификатор группы владельца;
- размер – размер файла в байтах;
- метки времени – времена последнего доступа и последнего изменения файла, а также время последнего изменения i-узла;
- адреса блоков – адреса первых 12 блоков файла, размер адреса – 4 байта;
- однократный косвенный блок – адрес одинарного косвенного блока, который содержит адреса дополнительных блоков файла;
- двукратный косвенный блок – содержит адреса 256 одинарных косвенных блоков, каждый из которых содержит адреса 256 блоков данных;
- трехкратный косвенный блок – содержит адреса 256 двукратных косвенных блоков.

Часть информации из i-узла можно посмотреть командой **stat *имя_файла***

Система адресации в ext2

Для хранения адреса файла выделено 15 полей размером 4 байта. Если файл умещается в 12 блоков, то номера этих блоков перечисляются в первых 12-ти полях адреса.

Если размер файла превышает 12 блоков, то в 13-е поле записывается адрес блока, в котором расположены номера следующих блоков файла. При размере блока 4096 байт он может содержать адреса 1024 блоков. Т.е. 13-е поле используется для косвенной адресации.



Если размер файла превышает (12 + 1024) блоков, то используется 14-е по-ле, в котором находится адрес блока, содержащего 1024 номеров блоков, каждый из которых ссылается на 1024 блока файла. Т.е. это поле двойной косвенной адресации.

15-е поле используется для тройной косвенной адресации (если файл включает более (12+1024+104857) блоков).

Данная система адресации позволя-ет для блока 4 Кбайт

Особенности файловой системы ext4

1. Максимальный размер файла – до 16 Тбайт (2^{44} байт).

2. Используются *48-битные* номера блоков. При размере блока 4 Кб это позволяет адресовать до одного экзобайта (2^{60} байтов), т.е. размер одного раздела диска может достигать 2^{60} байтов.

3. Используется смежно-индексированное выделение памяти с применением экстентов. Экстенты позволяют адресовать большое количество последовательно идущих блоков (до 128 Мб) одним дескриптором. До четырёх указателей на экстенты может разме-щаться непосредственно в i-узле.

4. Размер индексного дескриптора увеличен до 256 байтов.

2. Файловая система FAT

Системная область магнитного диска содержит:

- *загрузочная запись* (начальный загрузчик), содержится в нулевом секторе диска;
- *таблица размещения файлов* (File Allocation Table, FAT) – карта диска, содержит информацию о размещении файлов в области данных. Размер таблицы зависит от объема диска и размера блока (кластера). На любом диске всегда хранится два экземпляра FAT для обеспечения надежного доступа к данным. Элементы FAT могут быть 12-ти, 16-ти и 32-х разрядными, в зависимости от объема диска, соответственно файловые системы называются FAT12, FAT16 или FAT32 .
- *корневой каталог* – главный каталог диска, который занимает сектора, следующие за FAT.

FAT.Организация каталогов

Корневой каталог – совокупность записей размером 32 байта, струк-тура которых показана в таблице. Номер начального кластера определяет точку входа в FAT для данного файла и одновременно дис-ковый адрес собственно файла. Все атрибуты хранятся в одном байте. Корневой каталог обычно имеет размер 512 записей.

Каталоги нижнего уровня (подкаталоги) имеют структуру, анало-гичную корневому каталогу, только в отличие от него они не имеют фиксированного размера и фиксированного дискового адреса, т.е. хранятся на диске в области данных как обычные файлы.

Номер поля	Длина (байт)	Назначение поля
1	8	имя файла
2	3	расширение имени
3	1	атрибуты
4	10	резерв
5	2	время создания/модификации
6	2	дата создания/модификации
7	2	номер начального кластера
8	4	размер файла

FAT. Хранение длинных имен в Windows

Файловая система VFATдля каждого файла и подкаталога хранит два имени – длинное и короткое. Хранение длинных имен организуется в специальных запи-сях каталога, у которых байт атрибутов равен 0Fh. Такие записи невидимы для 16-разрядных программ и могут хранить до 13 символов в кодировке UNICODE.

Для регистрации файла с длинным именем в каталоге выделяется необходи-мое количество *специальных* записей, а также одна *стандартная* запись для хранения короткого имени. Блок специальных записей всегда располагается в ка-талоге перед стандартной записью, поэтому если к каталогу обращается 16-раз-рядная программа, то она будет видеть только короткое имя файла, а 32-разряд-ные Windows-приложения могут работать с длинными именами.

Короткое имя образуется из длинного следующим образом: оставляется 6 сим-волов длинного имени и дописываются знак “~” (тильда) и порядковый номер в пределах каталога для файлов, у которых первые 6 символов имени совпадают.

Например для файла «Отчет по лабораторной работе» (28 символов) в каталоге будет выделено 3 специальных записи и одна стандартная , хранящая короткое имя «Отчет ~1» и другие метаданные файла.

FAT. Таблица размещения файлов

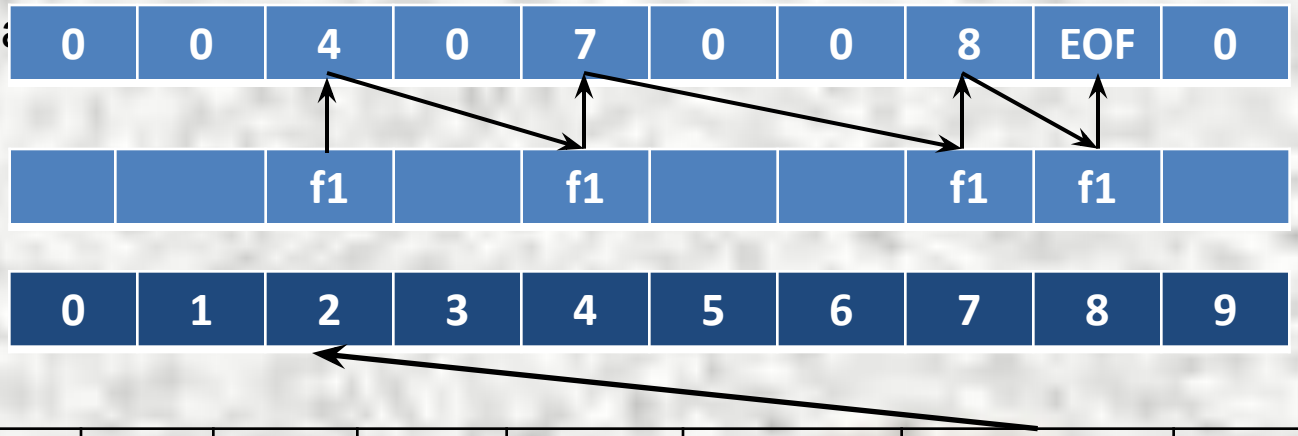
Таблица FAT содержит информацию о номерах кластеров, выделенных для хранения каждого файла. Она представляет собой карту области данных. Каждый элемент табли-цы соответствует одному кластеру в области данных. Возможные значения элементов:

- «0» - если кластер свободен;
- целое число (номер следующего кластера) – если кластер занят, но не является последним для файла;
- «EOF» (признак конца файла) - если кластер занят и является последним для файла.

Минимальный размер кластера на диске с файловой системой FAT зависит об объема диска ($V_{диска}$) и разрядности элемента FAT (r):

$$V_{кл\_min} = V_{диска} / 2^{r}$$

Пример доступа



Запись каталога

Имя	Тип	Атри-бут	Резерв	Время созд.	Дата созд.	Номер нач. кластера	Размер файла
F1		r		12-50	01.05.14	2	14652

FAT. Удаление файлов

При удалении файла выполняются следующие действия:

- в FAT обнуляются все элементы, выделенные для этого файла;

- в соответствующем элементе каталога изменяется имя файла – вместо первого символа в поле имени записывается символ «x».

Остальные характеристики файла в элементе каталога, а также содер-жимое файла в кластерах диска, не изменяются, поэтому всегда есть возможность полностью или частично восстановить удаленный файл. Полное восстановление возможно, если:

- не перезаписан соответствующий элемент каталога;

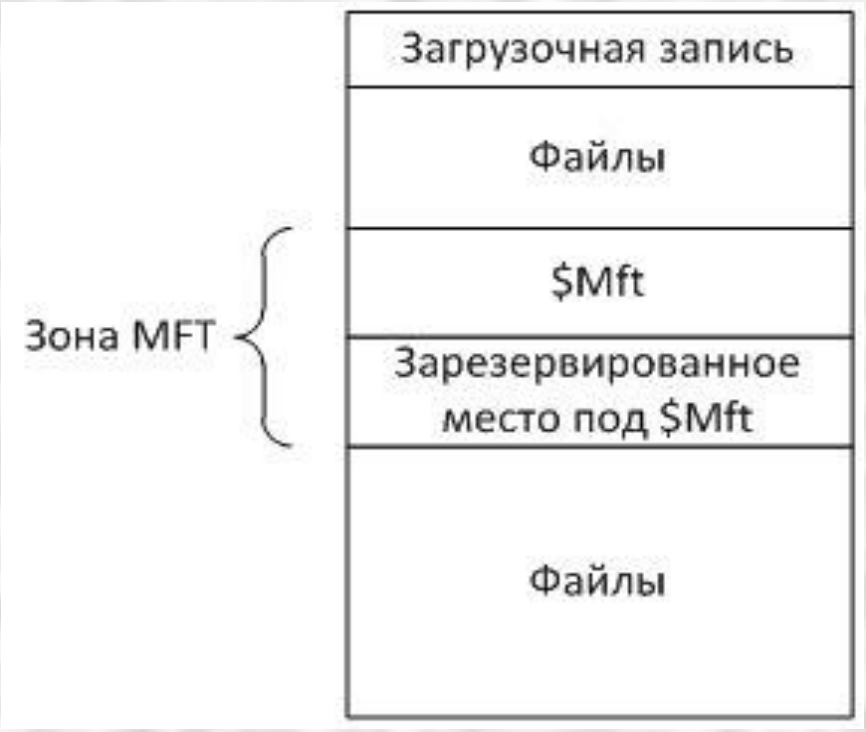
- имеется доступ к каталогу;

- кластеры, ранее занимаемые файлом, не выделены другим файлам или каталогам;

- удаленный файл был нефрагментированным.

Для восстановления удаленных файлов используются специальные программы (утилиты **Undelete**, **Recuva** , дисковые редакторы **Acronis**, **DiskExplorer**, **DMDE** и т.д.).

3. Файловая система NTFS



Основным отличием NTFS от файловой системы FAT является хранение основных системных структур данных в виде обыч-ных файлов.

В первом блоке раздела находится загру-зочная запись ($Boot), содержащая про-грамму загрузки и информацию о разделе (тип файловой системы и адреса основных системных файлов). Загрузочная запись обычно занимает 8 Кбайт (16 секторов).

NTFS использует два вида каталогов – главный и обычный. Каждый раздел диска имеет один главный каталог, в котором регистрируются все файлы раздела - MFT (Master File Table), который хранится в файле $MFT. Размер одной записи MFT – 1 Кбайт. Для хранения MFT на диске выделяется участок размером 12% объема диска (зона MFT). Адрес $MFT хранится в загрузочной записи.

NTFS. Таблица Master File Table

1. В записи MFT хранится вся информация о файле (имя, дата и время создания, размер, положение на диске отдельных фрагментов, и т.д). Если не хватает одной записи MFT, то используются несколько, причем не обязательно подряд. При этом первая запись называется базовой.

2. Каждая запись MFT имеет уникальный номер – индекс, общее количество записей – до 2^{48}.

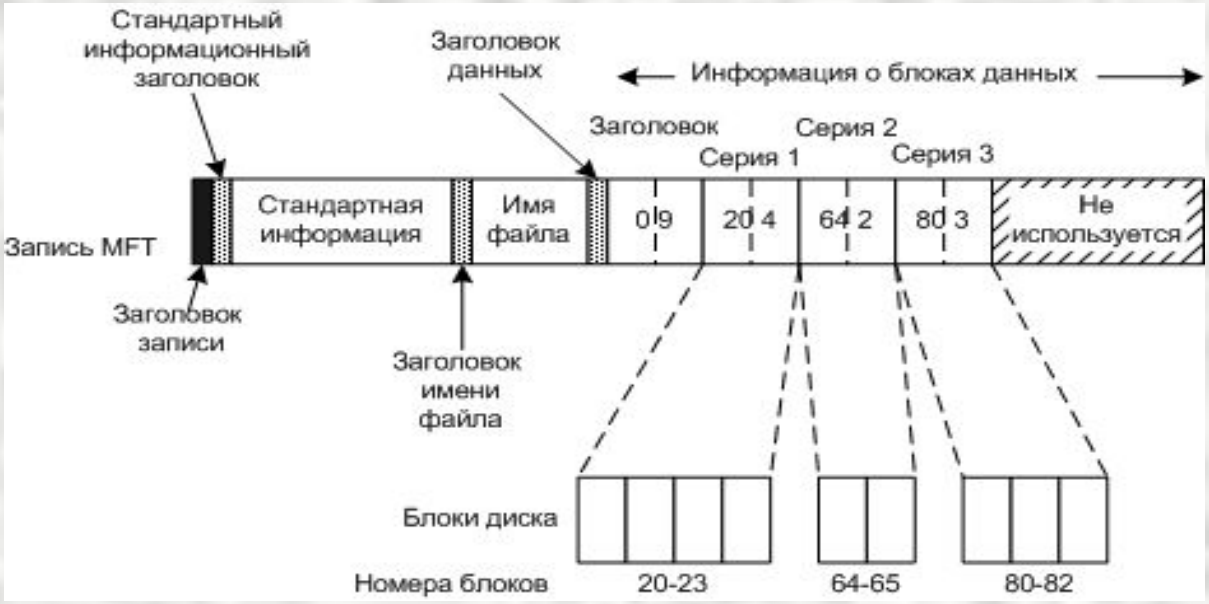
3. Первые 16 записей выделены для описания системных метафайлов, причем самая первая запись описывает структуру самого MFT.

4. В записи MFT можно размещать содержимое небольших по размеру файлов (несколько сотен байтов) без выделения места в области данных, что позволяет существенно экономить дисковое пространство. В этом случае файл называется *непосредственным*.

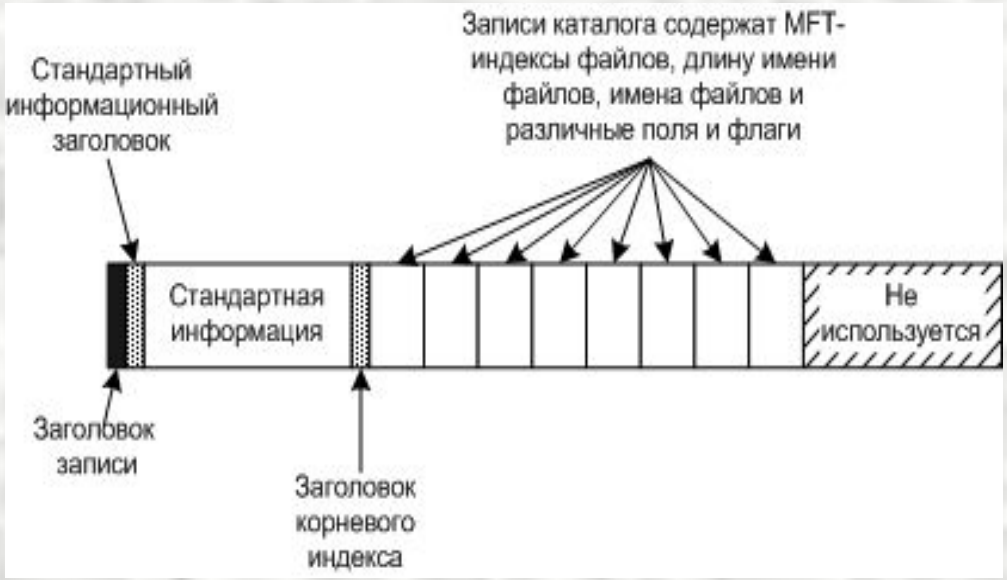
NTFS. Основные системные файлы

Имя файла	Назначение
$MFT	централизованный каталог всех файлов раздела
$MFTmirr	копия первых 4-х записей MFT
$LogFile	файл поддержки журналирования
$Volume	служебная информация - метка тома, версия файловой системы и т.д
$AttrDef	список стандартных атрибутов файлов на томе
$.	корневой каталог
$Bitmap	карта свободного места тома
$Boot	загрузочная запись
$Quota	файл, в котором записаны права пользователей на использование дискового пространства
$Upcase	файл - таблица соответствия строчных и прописных букв в именах файлов на текущем томе; нужен потому, что в NTFS имена файлов записываются в кодировке Unicode, что составляет более 65 тысяч различных символов, искать большие и малые эквиваленты которых достаточно сложно.

NTFS. Логическая структура записи MFT

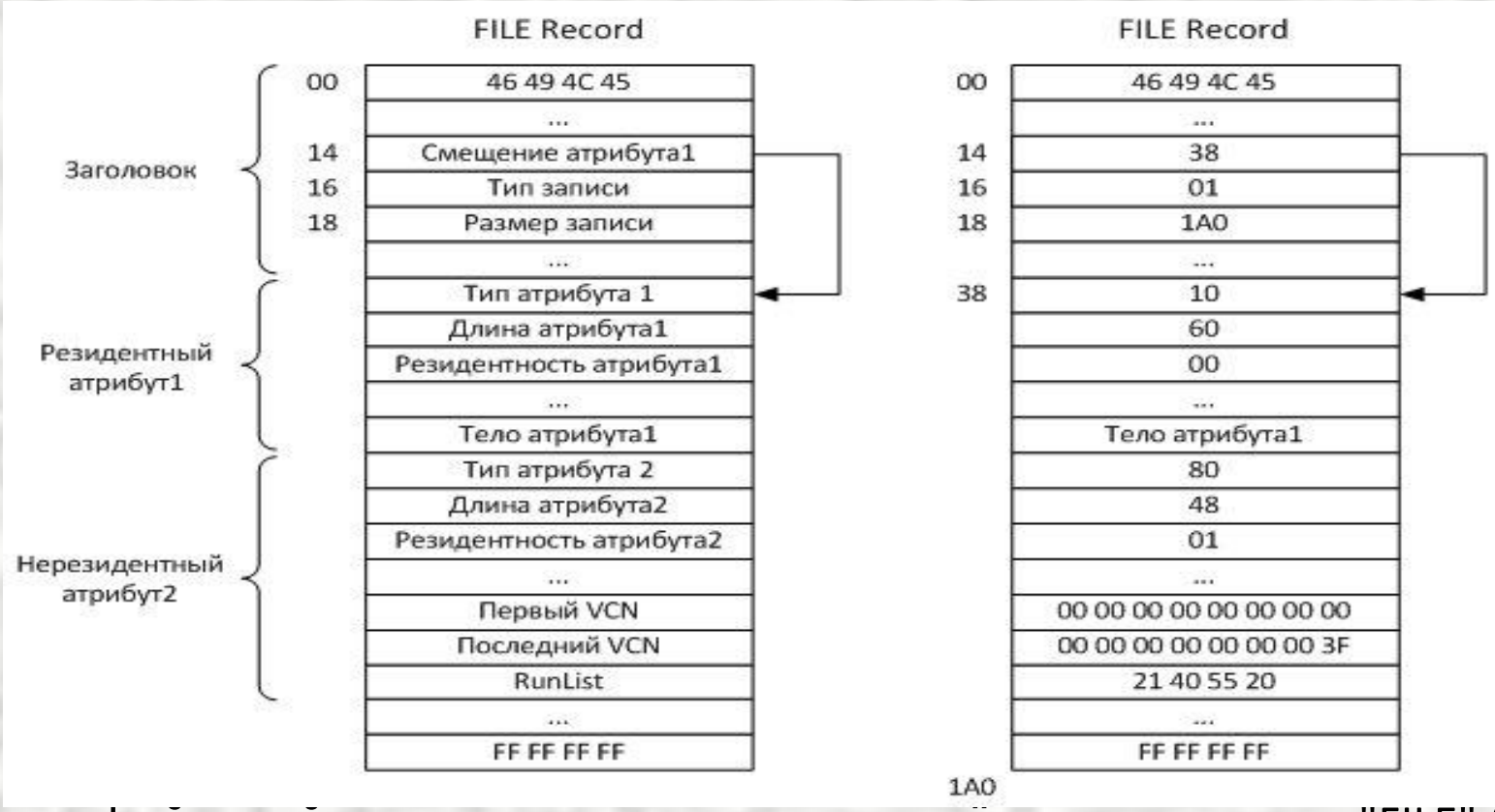


а) для слабо фрагментиро-ванного файла



б) для небольшого каталога

NTFS. Физическая структура записи MFT



В начале файловой записи находится признак её начала – слово "FILE" (46 49 4C 45). Со смещением 16 байт хранится тип записи (01-файл,02-каталог) и размер записи. Далее хранятся атрибуты.

Каждый атрибут имеет поля, указывающие тип, длину и резидентность. Тело рези-дентного атрибута хранится в самой записи, для нерезидентного в записи хранятся номера блоков диска, в которых записано тело атрибута. Типы атрибутов имеют свои численные значения, например, атрибуту $FILE_NAME соответствует значение 0x30, атрибуту $STANDARD_ INFORMATION –

Особенности NTFS

Для **больших каталогов** вместо простого списка файлов используется бинарное дерево, обеспечивающее быстрый поиск файла:

а) имена файлов упорядочиваются;

б) применяется метод половинного деления для определения место-положения искомого файла.

Например, если в каталоге зарегистрированы 1000 файлов, то для последовательного поиска (FAT) придется осуществить в среднем 500 сравнений (наиболее вероятно, что файл будет найден на середине поиска), а системе на основе дерева (NTFS) – не более 12-ти

($2^{10} = 1024$).

Для обеспечения **повышенной надежности** в NTFS используются:

а) журналирование - ведение журнала транзакций, под которыми понимается последовательность действий, совершаемых целиком и корректно или не совершаемых вообще;

б) возможность перемещения или фрагментации по диску всех своих служебных областей при возникновении любых неисправностей поверхностей диска (кроме первых 16 элементов MFT)

Работа с именованными потоками

1. Пусть имеется файл ***primer.txt*** *размером 15 байтов, содержащий строку:*

   Hello,students!

2. Создаем в файле именованный поток:

   Echo This is alternate stream> primer.txt:potok1

3. Просмотрим текущий каталог и определим размер файла :

   Dir (размер файла не изменился, команда не видит второй поток)

4. Просмотрим содержимое файла и потока:

   type primer.txt (вывод основного потока)

   type primer.txt:potok1 (ошибка, команда не видит второй поток !)

   more< primer.txt:potok1 (вывод на экран строки «This is alternate stream»)

5. Создадим в файле новый поток, содержащий графический файл 3630 Кбайт:

   type foto.jpg > primer.txt:potok.jpg

6. Просмотрим текущий каталог и определим размер файла :

   Dir (размер файла не изменился)

7. Извлечем графические данные из потока:

   mspaint primer.txt:*potok.jpg*

Работа с именованными потоками (продолжение)

1. Именованные потоки не распознаются большинством команд ОС и многими программами (например, антивирус-ными) .
2. Для получения информации по всем потокам файла необходимо применять специальные утилиты (например, **nfi.exe, sdir.exe, lads.exe**).
3. В ОС Windows именованные потоки используются только в разделах NTFS и создаются с помощью Проводника в свойствах файла.
4. При удалении небольшого файла на диске NTFS реально может освободиться несколько Гигабайтов памяти.

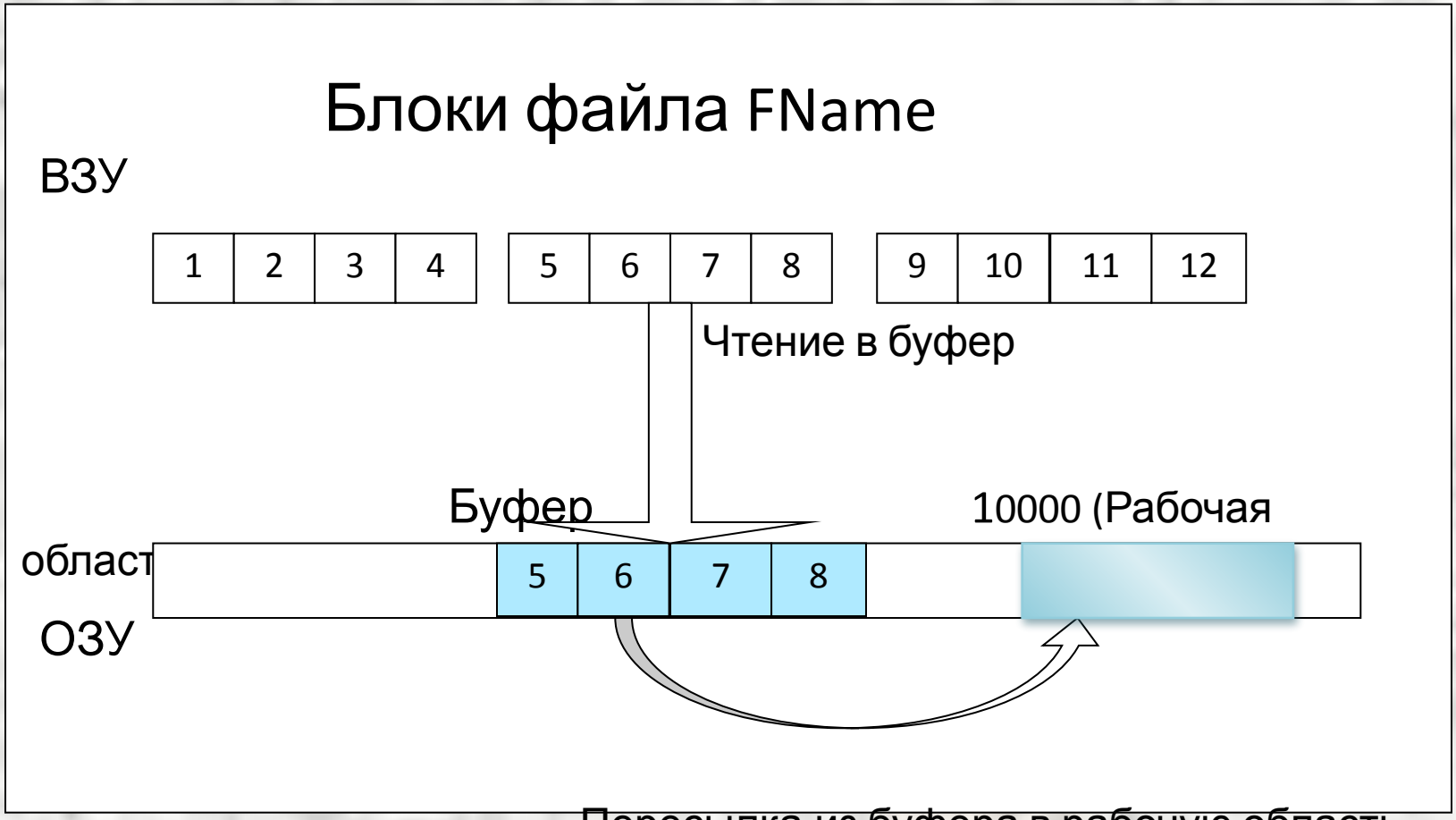
Часть 2

Основные принципы функционирования файловой системы

Структура и порядок выполнения запроса к файловой системе

Структуру файловой системы будем изучать на примере многопользо-вательской ОС семейства UNIX.

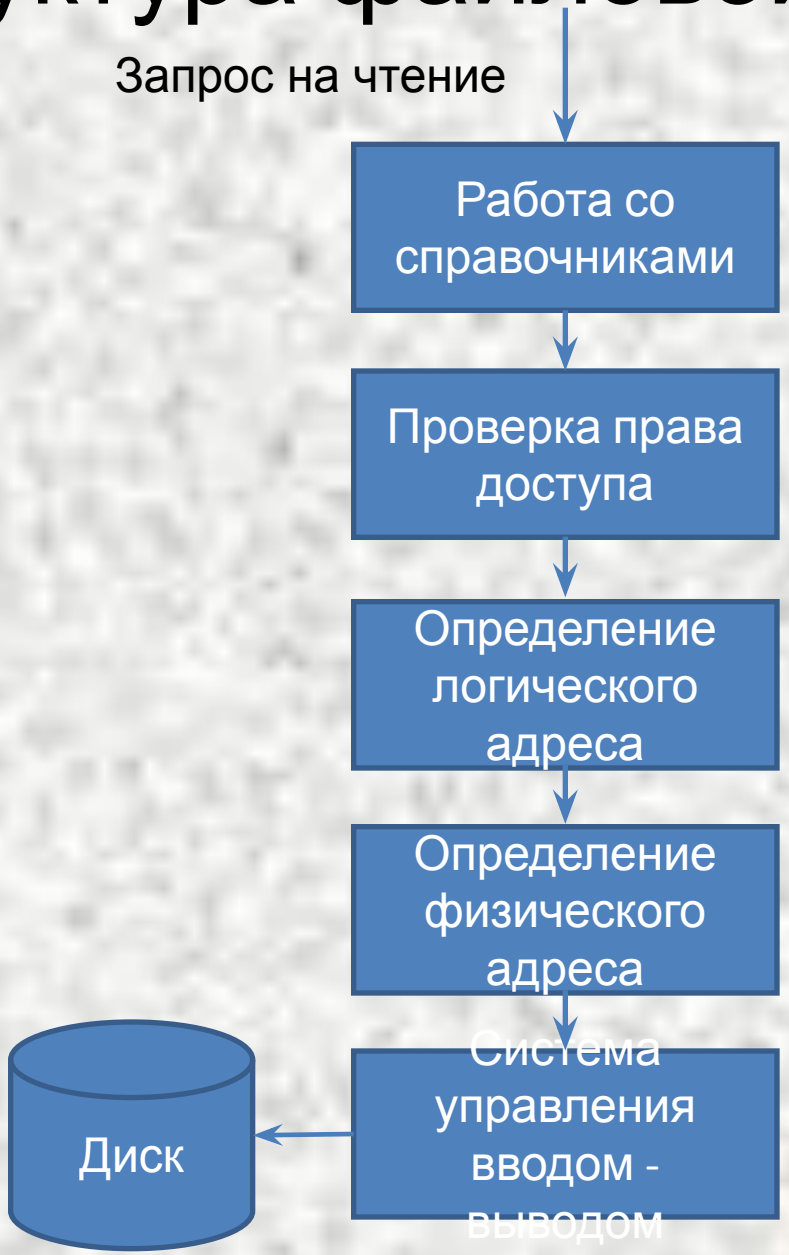
Типовой запрос: *считать из файла **Fname** запись **№ 6** в область памяти по адресу **10000***



Алгоритм выполнения запроса

1. Проверить наличие файла в указанном каталоге. При отсутствии вывести сообщение об ошибке.
2. Выбрать из индексного дескриптора указанного файла все его характеристики и скопировать их в ОЗУ.
3. Проверить право доступа к файлу с указанной операцией (чтение, запись, выполнение). При отсутствии прав вывести сообщение об ошибке.
4. Найти логический номер блока внутри файла.
5. Определить номер физического блока диска, в котором находится требуемая запись.
6. Считать блок с найденным номером с диска в буфер.
7. Выделить из буфера требуемую запись и переслать ее в рабочую область программы (в сегмент данных).

Структура файловой системы



Работа со справочниками (каталогами)

Базовый справочник – единый справочник всех файлов в одном разделе диска. Содержит все метаданные по каждому файлу за исключением имени, вместо которого используется уникальный идентификатор файла (ИДФ). В ОС Linux базовый справочник представлен массивом i-узлов.

Символьный справочник (каталог) – связывает ИДФ и имя файла.

Главный символьный справочник содержит список зарегистриро-ванных пользователей и идентификаторы файлов, содержащих их личные символьные справочники (домашние каталоги).

Функции уровня «Работа со справочниками»:

- ведение символьных справочников;
- ведение базового справочника;
- возможность присваивать различные имена одному файлу;
- возможность присваивать одно имя различным файлам;

Работа со справочниками (пример)

Базовый справочник

ИДФ	Адрес , UID, GID и т.д
1	0
2	100
3	105
4	110
5	114
6	120
7	136
8	141
9	145
10	152

ПС- признак символьного справочника;
ИДФ=1 описывает базовый справочник, ИДФ=2 – главный символьный справочник

Иванов.Alfa
Петров.Zetta

Иванов. Beta
Петров.Alfa
Gamma

Главный символьный справочник

Имя	ИДФ	ПС
Иванов	4	1
Петров	5	1
Сидоров	6	1

Символьный справочник Иванова

Имя	ИДФ	ПС
Alfa	3	0
Beta	7	0
Gamma	9	0

Символьный справочник Петрова

Имя	ИДФ	ПС
Alfa	8	0
Zetta	3	0
July	7	0

Проверка права доступа к файлам

Права доступа к файлам:

- полный доступ (write);
- чтение (read);
- исполнение (execute);
- добавление (add);
- отсутствуют.

Функции:

- хранение и изменение прав доступа;
- проверка прав доступа

Категории пользователей:

владелец;
группа;
остальные

Методы проверки:

- матрица управления доступом;
- список пользователей в базовом справочнике;
- пароли.

Пример матрицы управления доступом

ИДФ	3	4	5	6	7	8	9	10	11
Иванов	W	W	N	N	W	R	W	R	N
Петров	R	N	W	N	E	W	E	W	E
Сидоров	N	N	N	W	E	N	E	A	E
Попов	N	N	N	N	E	N	E	N	W

Определение адресов

Логический адрес записи – адрес записи относительно начала файла. Если к записи обращаемся по ключу, то значение ключа предварительно преобразу-ется в номер записи.

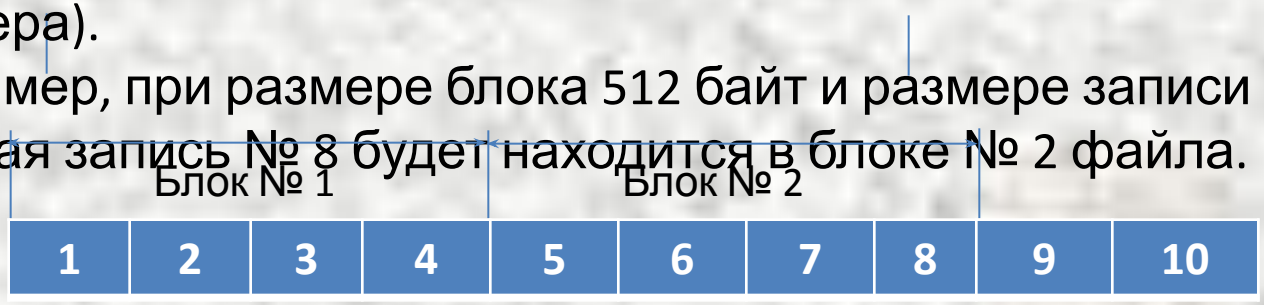
Логический адрес определяется расчетным путем с учетом формата и раз-мера записей. Например, для файла с записями фиксированного размера 128 байтов логический адрес записи № 8 будет равен: 7* 128 = 896 байтов.

1	2	3	4	5	6	7	8	9	10

Логический адрес записи преобразуется в *физический адрес* блока диска. Алгоритм преобразования:

1. По размеру блока и логическому адресу записи определяется логический номер блока внутри файла;
2. Из схемы размещения блоков файла на диске (FAT или i-узлы) определяется физический номер блока диска (номер дискового кластера).

Например, при размере блока 512 байт и размере записи 128 байт искомая запись № 8 будет находится в блоке № 2 файла.

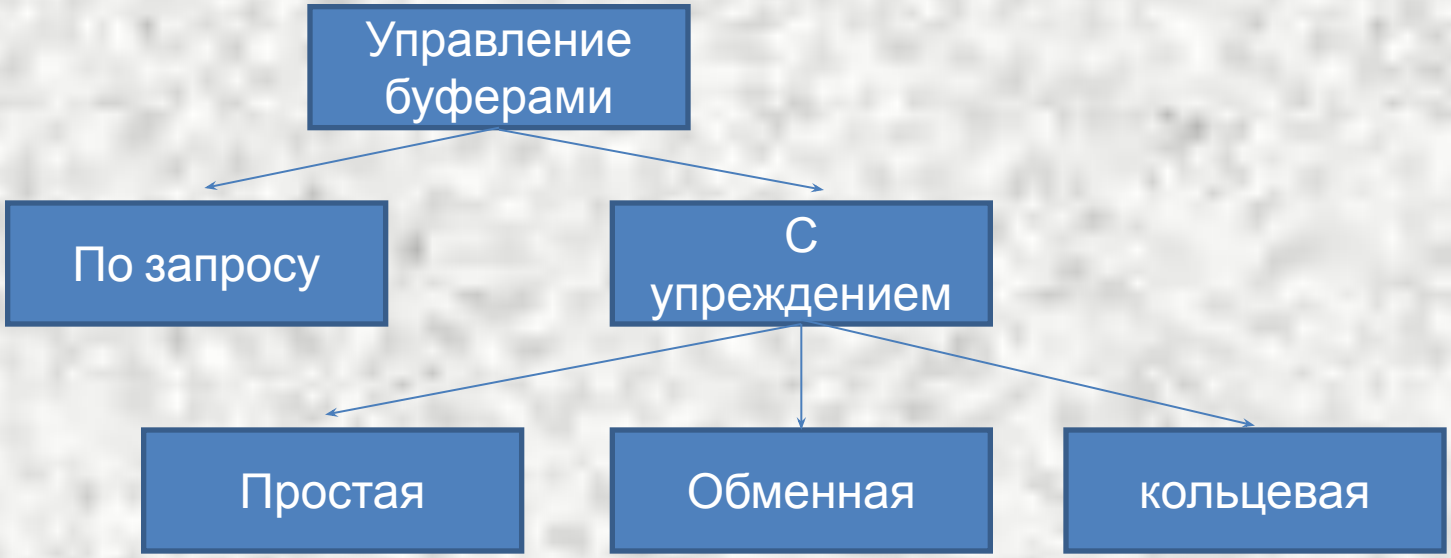


Функции системы ввода-вывода

- обеспечение общего интерфейса к драйверам устройств,
- именование устройств,
- обеспечение независимого размера блока,
- буферизация,
- распределение памяти на блок-ориентирован-ных устройствах,
- распределение и освобождение выделенных устройств,
- уведомление об ошибках.

Буферизация. Способы управления буферами

Буфер – участок оперативной памяти, выделенный для времен-ного хранения блоков данных.



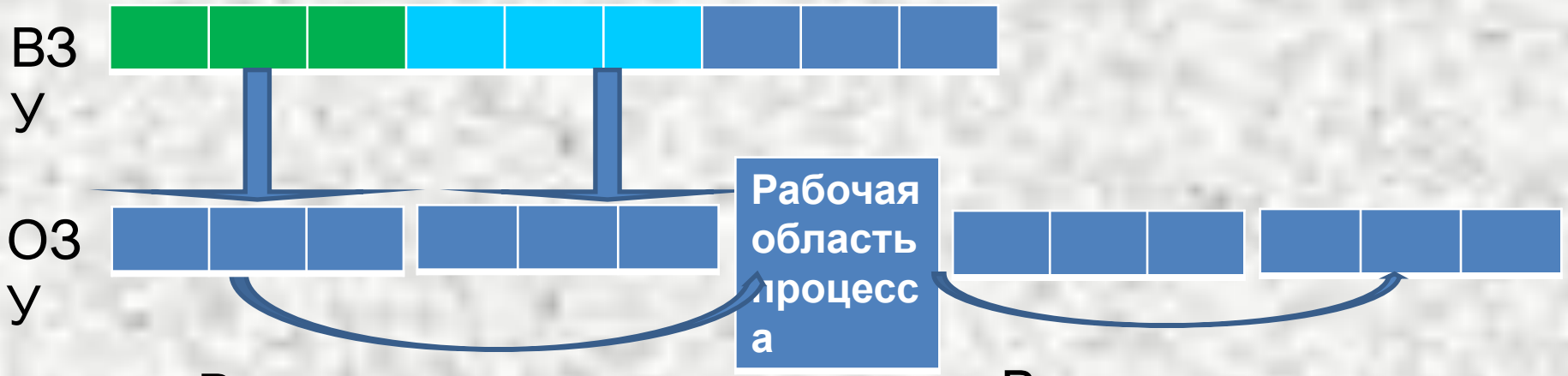
Буферизация по запросу :

1. Запрашивается буфер из буферного пула (**getbuf**);
2. Проводится загрузка данных в буфер (**read**);
3. Проводится синхронизация процессов исполнения и ввода-вывода (**wait**);
4. По окончании работы с файлом буфер освобождается (**freebuf**)

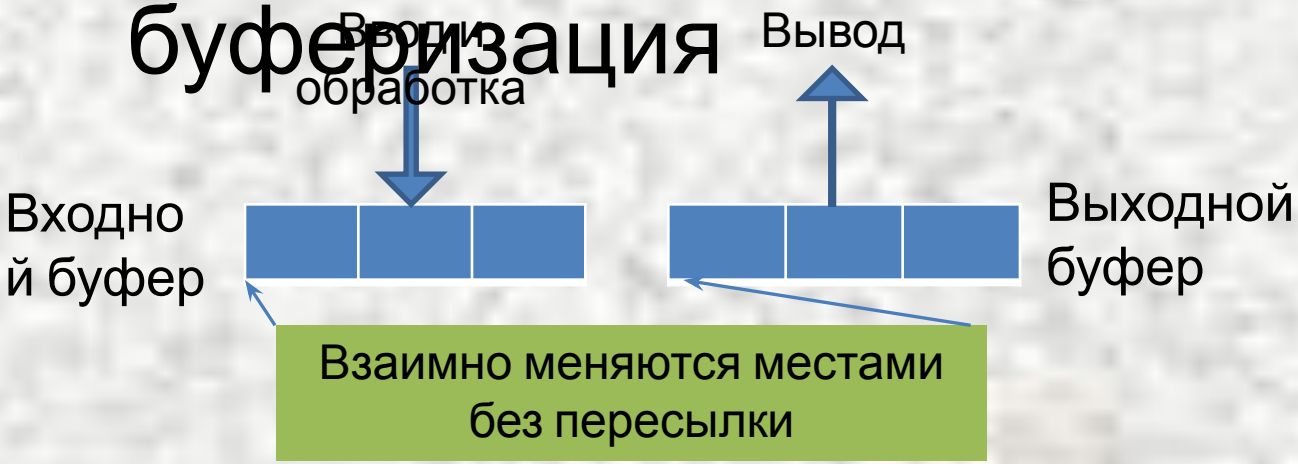
Буферизация с упреждением

1. Используется для обработки последовательных, индексно-последовательных и библиотечных файлов, т.е. когда известна последовательность обработки блоков.
2. Для операций обмена выделяется не менее 2-х буферов.
3. Загрузка буферов проводится до момента выдачи запроса на ввод записей.
4. Синхронизация процессов исполнения программы и обмена проводится файловой системой.

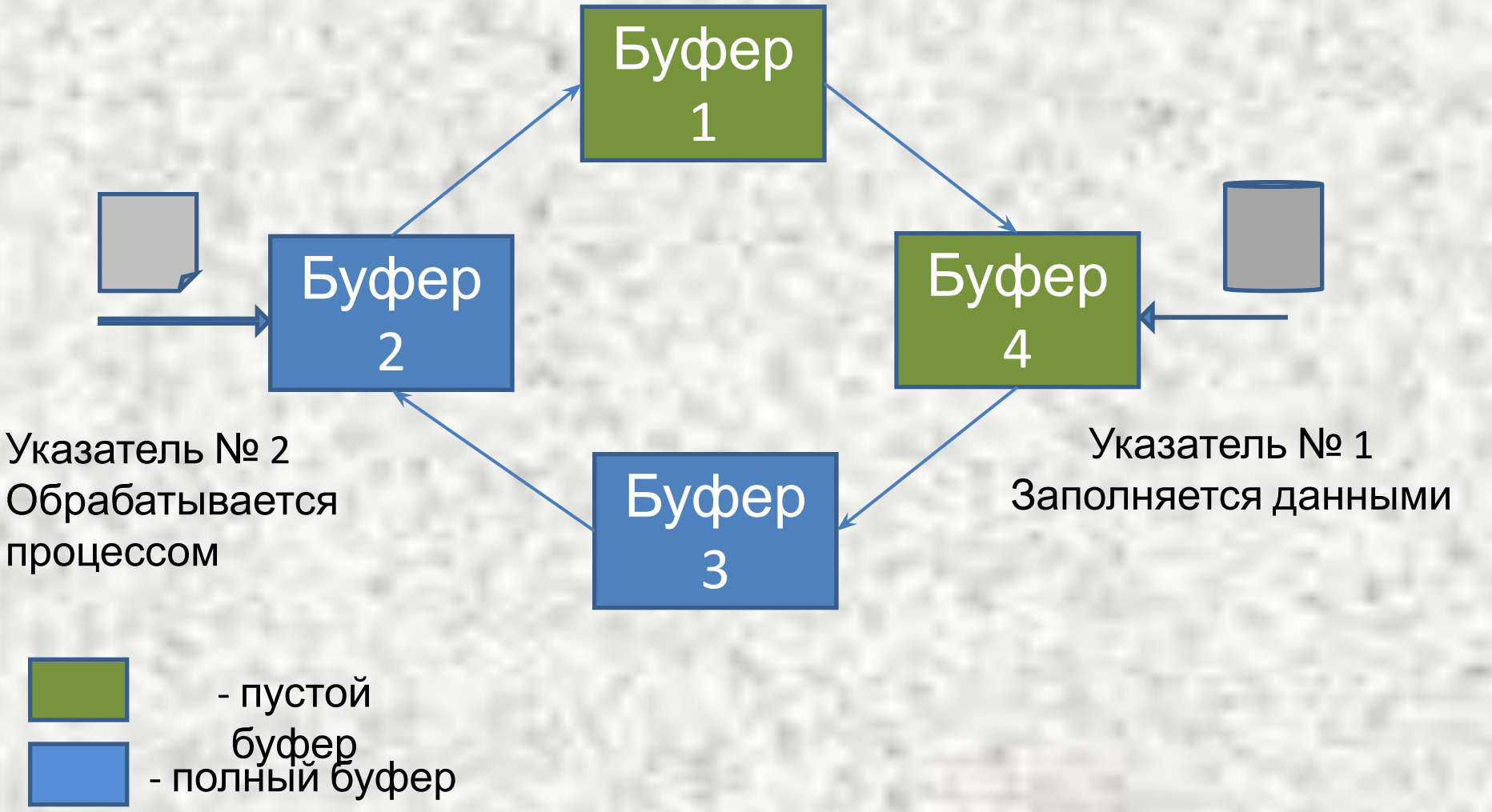
Простая буферизация



Обменная буферизация



Кольцевая буферизация (входные буферы)



Отображаемые файлы

Отображаемый файл – объект в адресном пространстве процесса, значение которого представляет содержимое заданного файла, расположенного на диске. В основе механизма отображения лежит использование виртуальной памяти.

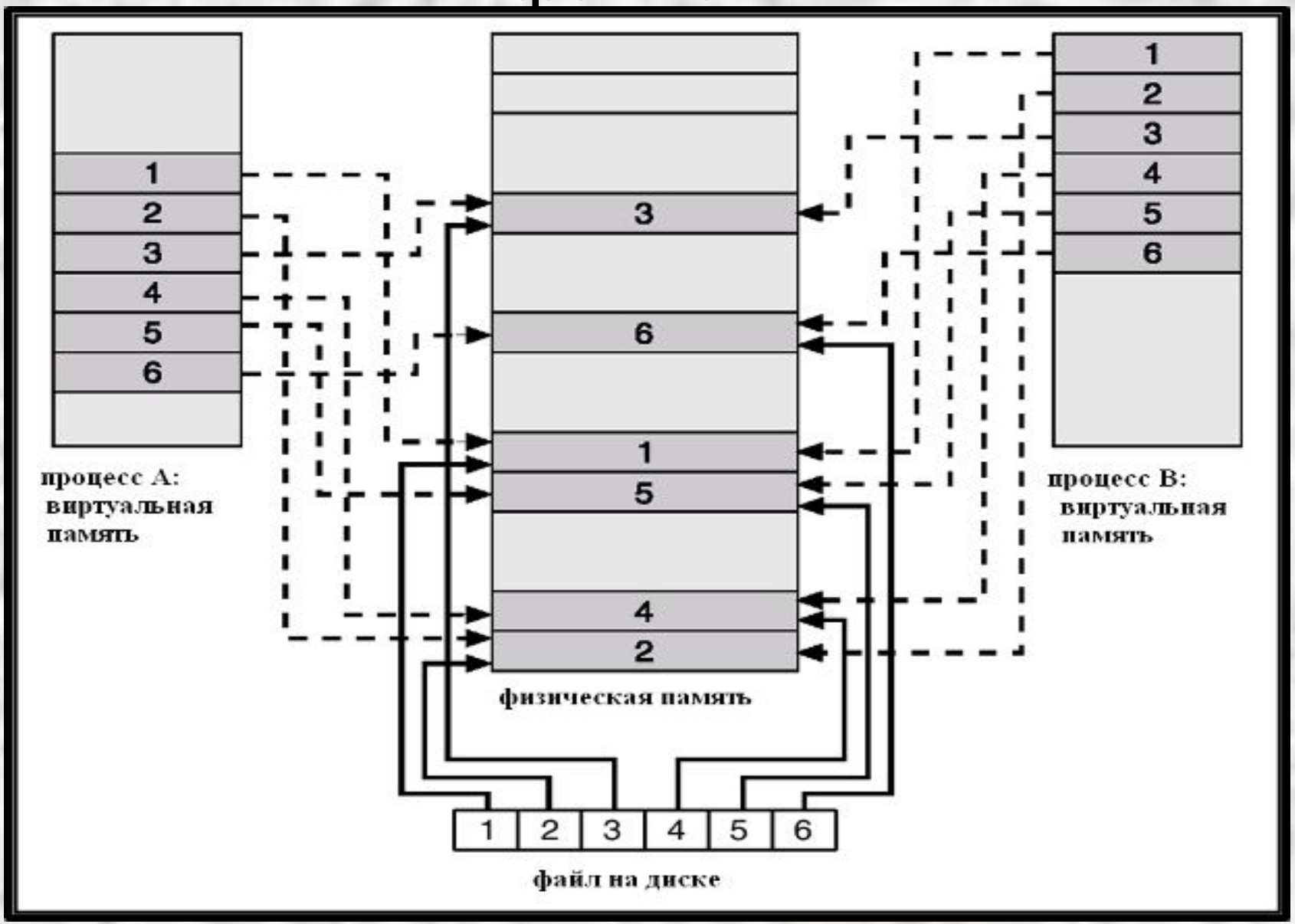
Каждый файл отображается в собственном сегменте адресного пространства процесса.

С одним отображаемым файлом могут одновременно работать несколько процессов.

Достоинства:

1. Отсутствует необходимость использования буферов, т.к. данные сразу попадают в страницы пользовательской памяти.
2. Уменьшается число системных вызовов.
3. Упрощается программирование за счет использования адресных указателей.
4. Могут использоваться для организации обмена данными между процессами.

Пример использования отображаемого файла



Отображаемые файлы (продолжение)

Недостатки:

1. Нельзя увеличить размер отображаемого файла.
2. Если один процесс работает с отображением файла, а другой – с реальным фай-лом на диске, то возникают проблемы с их согласованием, т. к. изменения, вне-сенные в отображаемый файл, будут сохранены только при вытеснении соответ-ствующей страницы на диск.
3. Размер реального файла может превышать максимальный размер сегмента.

Реализация в Object Pascal:

Функции Object Pascal для работы с отображаемыми файлами: *CreateFileMapping()* –создание отображаемого файла; *MapViewOfFile()* – проеци-рование данных файла в адресное пространство процесса; *UnMapViewOfFile()* – прекращение отображения файла в адресное пространство процесса;

Реализация в C#:

Метод *MemoryMappedFile.CreateFromFile* - создание отображаемого файла;
Метод *MemoryMappedFile. OpenExisting* – подключение к существующему отображаемому файлу;
Метод *MemoryMappedFile.CreateViewStream* – получение последовательного доступа к отображаемому файлу;