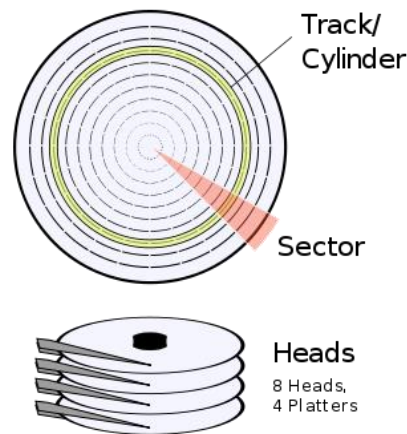


## Операции с файлами. Права доступа. Файловые системы ufs, ext3.

### Архитектура диска



С целью адресации пространства поверхности пластин диска делятся на *дорожки* — концентрические кольцевые области. Каждая дорожка делится на равные отрезки — *сектора*. Адресация CHS предполагает, что все дорожки в заданной зоне диска имеют одинаковое число секторов.

*Цилиндр* — совокупность дорожек, равноотстоящих от центра, на всех рабочих поверхностях пластин жесткого диска. *Номер головки* задает используемую рабочую поверхность (то есть конкретную дорожку из цилиндра), а *номер сектора* — конкретный сектор на дорожке.

Чтобы использовать адресацию CHS, необходимо знать *геометрию* используемого диска: общее к-во цилиндров, головок и секторов в нем. Первоначально эту информацию требовалось задавать вручную; в стандарте ATA-1 была введена функция автоопределения геометрии (команда Identify Drive)

### Адресация данных

Минимальной адресуемой областью данных на жестком диске является *сектор*. Размер сектора традиционно равен 512 байт. В 2006 г. IDEMA объявила о переходе на размер сектора 4096 байт, который планируется завершить к 2010 году. В окончательной версии Windows Vista, вышедшей в 2007 г., присутствует ограниченная поддержка дисков с таким размером сектора.

Существует 2 основных способа адресации секторов на диске: *цилиндр-головка-сектор* (*cylinder-head-sector*, CHS) и *линейная адресация блоков* (*linear block addressing*, LBA).

### CHS

При этом способе сектор адресуется по его физическому положению на диске 3 координатами — *номером цилиндра*, *номером головки* и *номером сектора*. В современных дисках со встроенными контроллерами эти координаты уже не соответствуют физическому положению сектора на диске и являются «логическими координатами»

## LBA

При этом способе сектор задается единственным числом — своим абсолютным номером на диске. Стандарты ATA-1 требуют однозначного соответствия между режимами CHS и LBA:

### Сравнение интерфейсов

	Проп-ная способ-ть, Мбит/с	Максимальная длина кабеля, м	Требуется ли кабель питания	Количество накопителей на канал	Число проводников в кабеле	Другие особенности
UltraATA/133	1064	0,46	Да (3,5") / Нет (2,5")	2	40/80	Controller+2Slave, горячая замена невозможна
SATA/300	2400	1	Да	1	4	Host/Slave, возможна горячая замена на некоторых контроллерах
FireWire/400	400	4,5 (при последовательном соединении до 72 м)	Да/Нет (зависит от типа интерфейса и накопителя)	63	4/6	устройства равноправны, горячая замена возможна
FireWire/800	800	4,5 (при последовательном соединении до 72 м)	Нет	63	4/6	устройства равноправны, горячая замена возможна
USB 2.0	480	5 (при последовательном соединении, через хабы, до 72 м)	Да/Нет (зависит от типа накопителя)	127	4	Host/Slave, горячая замена возможна
Ultra-320 SCSI	2560	12	Да	16	50/68	устройства равноправны, горячая замена возможна
SAS	3000	8	Да	Свыше 16384		горячая замена; возможно подключение SATA-устройств в SAS-контроллеры
eSATA	2400	2	Да	1 (с умножителем портов до 15)	4	Host/Slave, горячая замена возможна

Концепция разделения диска на разделы

Принципы раздела диска на разделы:

- несколько операц-ых систем

-безопасность данных

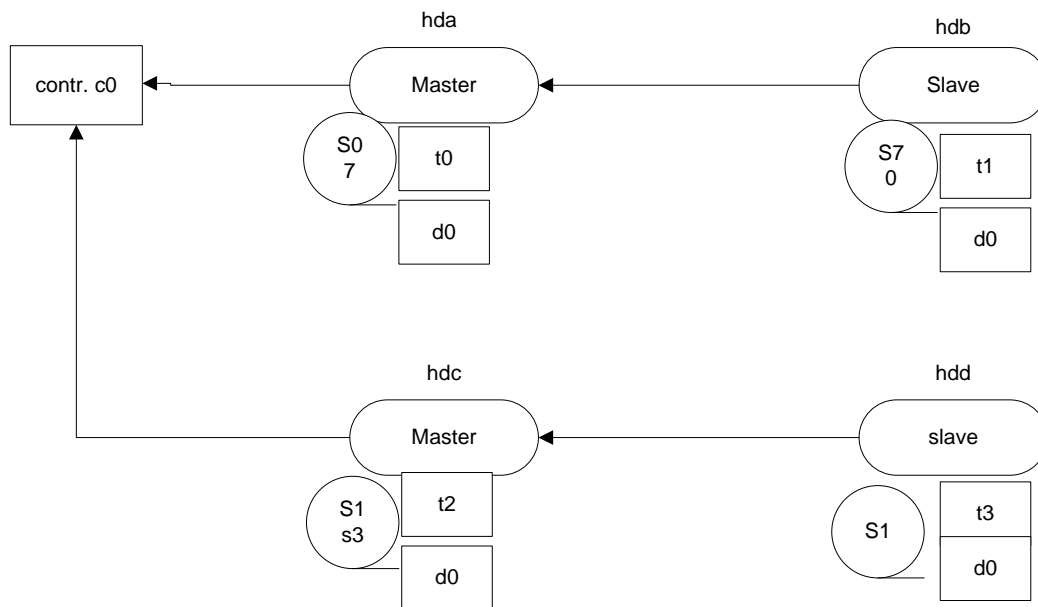
Квоты на диске, файл

1-й сектор на диске – **master boot record** (512 байт)

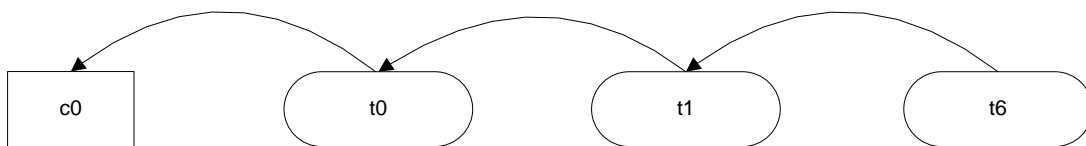
Цель MBR — ещё не загрузка ОС, а всего лишь выбор, «с какого раздела жёсткого диска следует загрузить ОС». На стадии MBR происходит выбор раздела диска и ничего более. Загрузка самой ОС происходит на более поздних этапах.

Первичных раздела 4-е (может быть 4 на диске)

IDE – integrated device electronic



SCSI

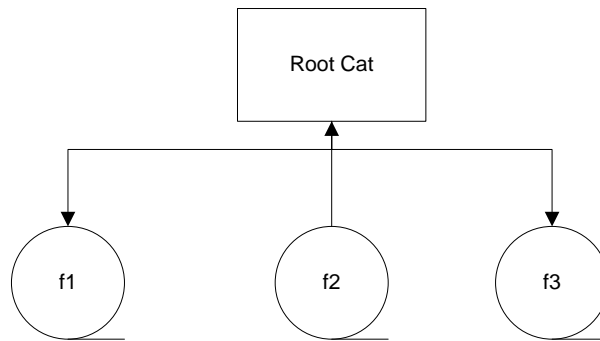


## Файлы и файловые системы

- В файлах хранятся данные пользователей
- В файлах диктуются права пользователя
- Файлы обеспечивают доступ к периферийным устройствам

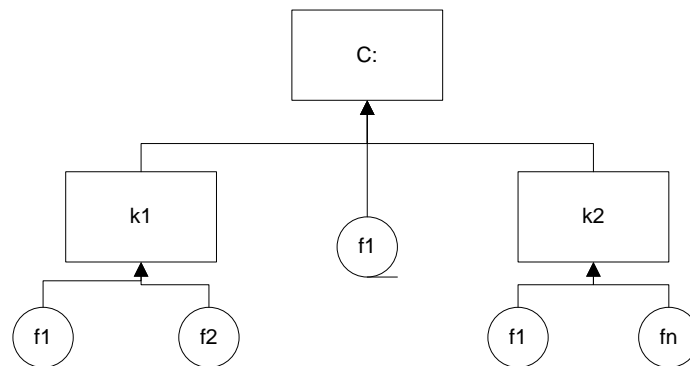
## Модели файловых систем

### А) RT11 одноуровневая система

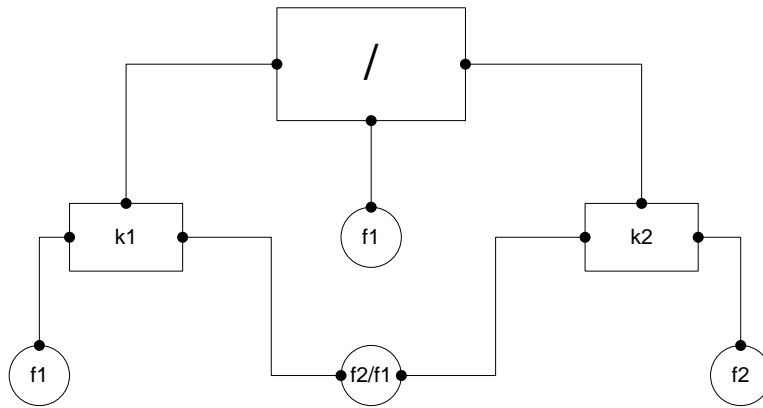


F1-файлы

### Б) Иерархическая W9k, w2k



### В) сетевая модель создания файлов систем UNIX/LINUX



## Атрибуты файлов

Понятие «файл» включает не только хранимые им данные и имя, но и *атрибуты*. Атрибуты — это информация, описывающая свойства файла. Примеры возможных атрибутов файла:

- тип файла (обычный файл, каталог, специальный файл и т. п.);
- владелец файла;
- создатель файла;
- пароль для доступа к файлу;
- информация о разрешенных операциях доступа к файлу;
- времена создания, последнего доступа и последнего изменения;
- текущий размер файла;
- максимальный размер файла;
- признак «только для чтения»;
- признак «скрытый файл»;
- признак «системный файл»;
- признак «архивный файл»;
- признак «двоичный/символьный»;
- признак «временный» (удалить после завершения процесса);
- признак блокировки;
- длина записи в файле;
- указатель на ключевое поле в записи;
- длина ключа.

Набор атрибутов файла определяется спецификой файловой системы: в файловых системах разного типа для характеристики файлов могут использоваться разные наборы атрибутов. Например, в файловых системах, поддерживающих неструктурированные файлы, нет необходимости использовать три последних атрибута в приведенном списке, связанных со структуризацией файла. В однопользовательской ОС в наборе атрибутов будут отсутствовать характеристики, имеющие отношение к пользователям и защите, такие как владелец файла, создатель файла, пароль для доступа к файлу, информация о разрешенном доступе к файлу.

## Файловая система

*Файловая система* - это часть операционной системы, назначение которой состоит в том, чтобы обеспечить пользователю удобный интерфейс при работе с данными, хранящимися на диске, и обеспечить совместное использование файлов несколькими пользователями и процессами.

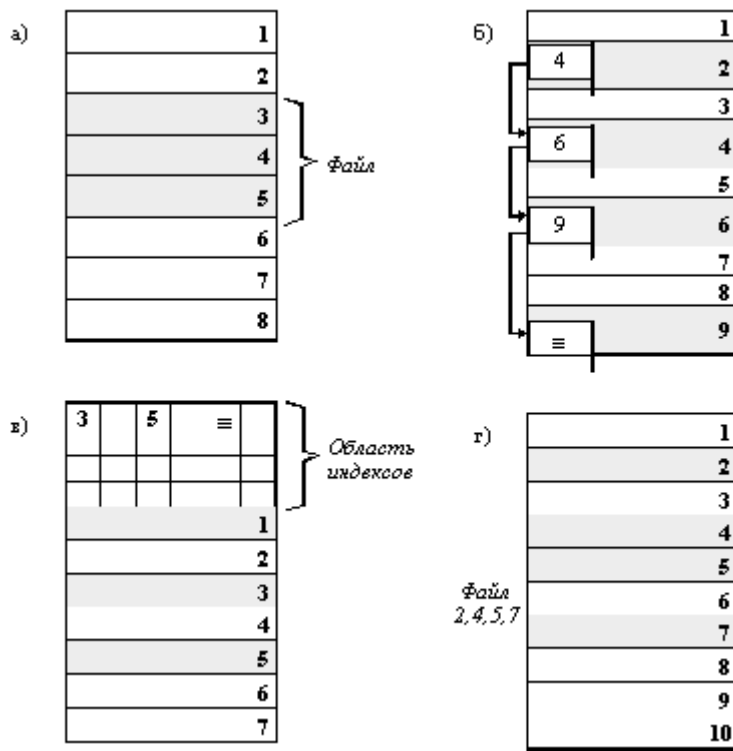
В широком смысле понятие "файловая система" включает:

- совокупность всех файлов на диске,
- наборы структур данных, используемых для управления файлами, такие, например, как каталоги файлов, дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске,
- комплекс системных программных средств, реализующих управление файлами, в частности: создание, уничтожение, чтение, запись, именование, поиск и другие операции над файлами.

### Физическая организация и адрес файла

Физическая организация файла описывает правила расположения файла на устройстве внешней памяти, в частности на диске. Файл состоит из физических записей - блоков. Блок - наименьшая единица данных, которой внешнее устройство обменивается с оперативной памятью. Непрерывное размещение - простейший вариант физической организации (рисунок 1, а), при котором файлу предоставляется последовательность блоков диска, образующих единый сплошной участок дисковой памяти. Для задания адреса файла в этом случае достаточно указать только номер начального блока. Другое достоинство этого метода - простота. Но имеются и два существенных недостатка. Во-первых, во время создания файла заранее не известна его длина, а значит не известно, сколько памяти надо зарезервировать для этого файла, во-вторых, при таком порядке размещения неизбежно возникает фрагментация, и пространство на диске используется не эффективно, так как отдельные участки маленького размера (минимально 1 блок) могут остаться не используемыми.

Следующий способ физической организации - размещение в виде связанного списка блоков дисковой памяти (рисунок 1, б). При таком способе в начале каждого блока содержится указатель на следующий блок. В этом случае адрес файла также может быть задан одним числом - номером первого блока. В отличие от предыдущего способа, каждый блок может быть присоединен в цепочку какого-либо файла, следовательно фрагментация отсутствует. Файл может изменяться во время своего существования, наращивая число блоков. Недостатком является сложность реализации доступа к произвольно заданному месту файла: для того, чтобы прочитать пятый по порядку блок файла, необходимо последовательно прочитать четыре первых блока, проследив цепочку номеров блоков. Кроме того, при этом способе количество данных файла, содержащихся в одном блоке, не равно степени двойки (одно слово израсходовано на номер следующего блока), а многие программы читают данные блоками, размер которых равен степени двойки.



*Рис. 1. Физическая организация файла  
а - непрерывное размещение; б - связанный список блоков;  
в - связанный список индексов; г - перечень номеров блоков*

Популярным способом, используемым, например, в файловой системе FAT операционной системы MS-DOS, является использование связанного списка индексов (рисунок 1,в). С каждым блоком связывается некоторый элемент - индекс. Индексы располагаются в отдельной области диска (в MS-DOS это таблица FAT). Если некоторый блок распределен некоторому файлу, то индекс этого блока содержит номер следующего блока данного файла. При такой физической организации сохраняются все достоинства предыдущего способа, но снимаются оба отмеченных недостатка: во-первых, для доступа к произвольному месту файла достаточно прочитать только блок индексов, отсчитать нужное количество блоков файла по цепочке и определить номер нужного блока, и, во-вторых, данные файла занимают блок целиком, а значит имеют объем, равный степени двойки.

Допускается задание физического расположения файла путем простого перечисления номеров блоков, занимаемых этим файлом. ОС UNIX использует вариант данного способа, позволяющий обеспечить фиксированную длину адреса, независимо от размера файла. Для хранения адреса файла выделено 13 полей. Если размер файла меньше или равен 10 блокам, то номера этих блоков непосредственно перечислены в первых десяти полях адреса. Если размер файла больше 10 блоков, то следующее 11-е поле содержит адрес блока, в котором могут быть расположены еще 128 номеров следующих блоков файла. Если файл больше, чем 10+128 блоков, то используется 12-е поле, в котором находится номер блока, содержащего 128 номеров блоков, которые содержат по 128 номеров блоков данного файла. И, наконец, если файл больше 10+128+128(128), то используется последнее 13-е поле для тройной косвенной адресации, что позволяет задать адрес файла, имеющего размер максимум  $10 + 128 + 128(128 + 128(128(128)))$ .

### **Общие понятия файловой системы. Общая модель файловой системы**

Функционирование любой файловой системы можно представить многоуровневой моделью (рисунок 2), в которой каждый уровень предоставляет некоторый интерфейс (набор функций) вышележащему уровню, а сам, в свою очередь, для выполнения своей работы использует интерфейс (обращается с набором запросов) нижележащего уровня.

Задачей символьного уровня является определение по символьному имени файла его уникального имени. В файловых системах, в которых каждый файл может иметь только одно символьное имя (например, MS-DOS), этот уровень отсутствует, так как символьное имя, присвоенное файлу пользователем, является одновременно уникальным и может быть использовано операционной системой.



Рис. 2. Общая модель файловой системы

В других файловых системах, в которых один и тот же файл может иметь несколько символьных имен, на данном уровне просматривается цепочка каталогов для определения уникального имени файла. В файловой системе UNIX, например, уникальным именем является номер индексного дескриптора файла (i-node).

На следующем, базовом уровне по уникальному имени файла определяются его характеристики: права доступа, адрес, размер и другие. Характеристики файла могут входить в состав каталога или храниться в отдельных таблицах. При открытии файла его характеристики перемещаются с диска в оперативную память, чтобы уменьшить среднее время доступа к файлу. В некоторых файловых системах (например, HPFS) при открытии файла вместе с его характеристиками в оперативную память перемещаются несколько первых блоков файла, содержащих данные.

Следующим этапом реализации запроса к файлу является проверка прав доступа к нему. Для этого сравниваются полномочия пользователя или процесса, выдавших запрос, со списком разрешенных видов доступа к данному файлу. Если запрашиваемый вид доступа разрешен, то выполнение запроса продолжается, если нет, то выдается сообщение о нарушении прав доступа.

На логическом уровне определяются координаты запрашиваемой логической записи в файле, то есть требуется определить, на каком расстоянии (в байтах) от начала файла находится требуемая логическая запись. При этом абстрагируются от физического расположения файла, он представляется в виде непрерывной последовательности байт. Алгоритм работы данного уровня зависит от логической организации файла. Например, если файл организован как последовательность логических записей фиксированной длины  $l$ , то  $n$ -ая логическая запись имеет смещение  $l(n-1)$  байт. Для определения координат логической записи в файле с индексно-последовательной организацией выполняется чтение таблицы индексов (ключей), в которой непосредственно указывается адрес логической записи.

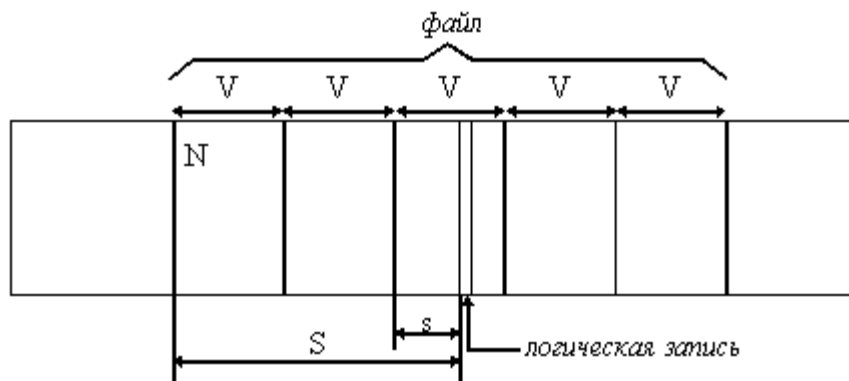


Рис. 3. Функции физического уровня файловой системы

Исходные данные:

V - размер блока

N - номер первого блока файла

S - смещение логической записи в файле

Требуется определить на физическом уровне:

n - номер блока, содержащего требуемую логическую запись

s - смещение логической записи в пределах блока

$n = N + [S/V]$ , где  $[S/V]$  - целая часть числа  $S/V$

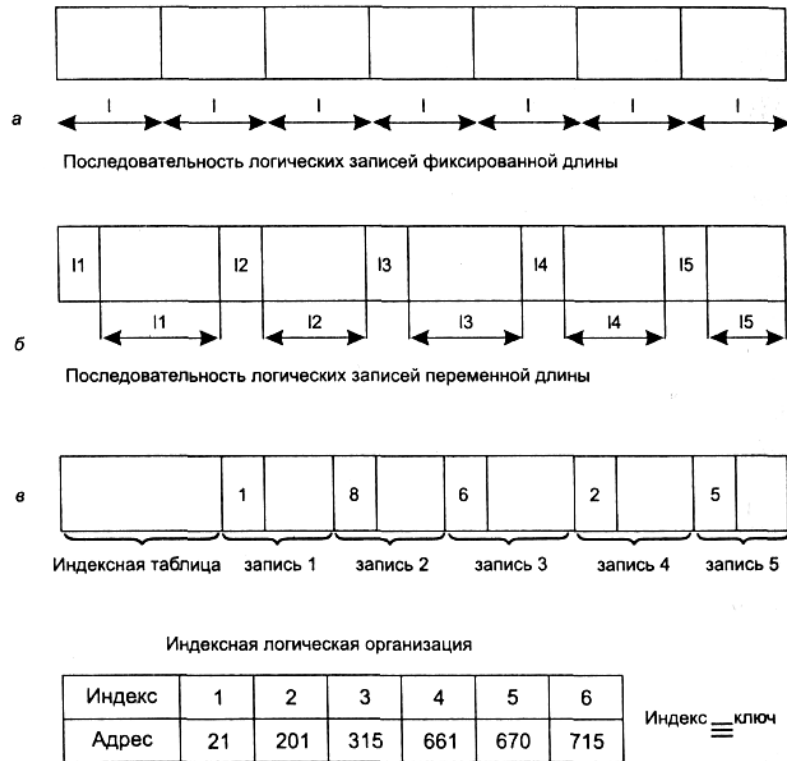
$s = R [S/V]$  - дробная часть числа  $S/V$

На физическом уровне файловая система определяет номер физического блока, который содержит требуемую логическую запись, и смещение логической записи в физическом блоке. Для решения этой задачи используются результаты работы логического уровня - смещение логической записи в файле, адрес файла на внешнем устройстве, а также сведения о физической организации файла, включая размер блока. Рисунок 3 иллюстрирует работу физического уровня для простейшей физической организации файла в виде непрерывной последовательности блоков. Задача физического уровня решается независимо от того, как был логически организован файл.

После определения номера физического блока, файловая система обращается к системе ввода-вывода для выполнения операции обмена с внешним устройством. В ответ на этот запрос в буфер файловой системы будет передан нужный блок, в котором на основании полученного при работе физического уровня смещения выбирается требуемая логическая запись.

8		3		1				4	
Имя файла			Расширение		R	A	H	S	Резервные
Резервные	Время	Дата		№ первого кластера		Размер			
2					14				
№ индексного дескриптора					Имя файла				

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

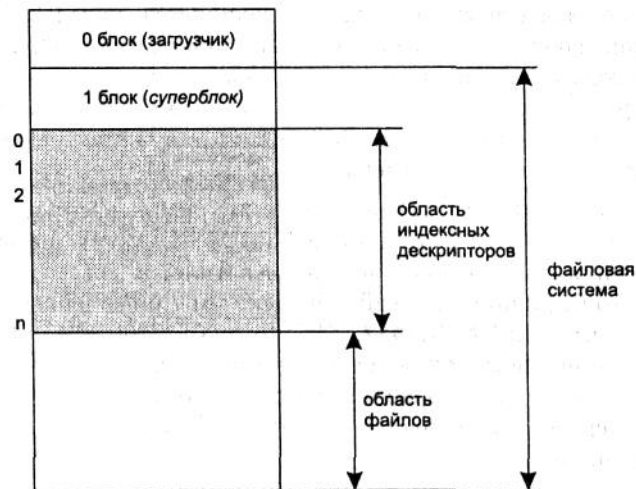


Способы логической организации файлов

## S5FS

Расположение файловой системы s5 на диске иллюстрирует рис. 7.15. Раздел диска, где размещается файловая система, делится на четыре области:

- загрузочный блок;
- суперблок (*superblock*) содержит самую общую информацию о файловой системе: размер файловой системы, размер области индексных дескрипторов, число индексных дескрипторов, список свободных блоков и список свободных индексных дескрипторов, а также другую административную информацию;
- область индексных дескрипторов (*inode list*), порядок расположения индексных дескрипторов в которой соответствует их номерам,
- область данных, в которой расположены как обычные файлы, так и файлы-каталоги, в том числе и корневой каталог; специальные файлы представлены в файловой системе только записями в соответствующих каталогах и индексными дескрипторами специального формата, но места в области данных не занимают.



**Рис. 7.15. Расположение файловой системы s5 на диске**

Основной особенностью физической организации файловой системы s5 является отделение имени файла от его характеристик, хранящихся в отдельной структуре, называемой *индексным дескриптором (mode)*. Индексный дескриптор в s5

имеет размер 64 байта и содержит данные о типе файла, адресную информацию, привилегии доступа к файлу и некоторую другую информацию:

- идентификатор владельца файла;
- тип файла; файл может быть файлом обычного типа, каталогом, специальным файлом, а также конвейером или символьной связью;
- права доступа к файлу;
- временные характеристики: время последней модификации файла, время последнего обращения к файлу, время последней модификации индексного дескриптора;
- число ссылок на данный индексный дескриптор, равный количеству псевдонимов файла;
- адресная информация (структура адреса рассмотрена выше в разделе «Физическая организация и адресация файла»);
- размер файла в байтах.

Каждый индексный дескриптор имеет номер, который одновременно является уникальным именем файла. Индексные дескрипторы расположены в особой области диска в строгом соответствии со своими номерами. Соответствие между полными символьными именами файлов и их уникальными именами устанавливается с помощью иерархии каталогов. Система ведет список номеров свободных индексных дескрипторов. При создании файла ему выделяется номер из этого списка, а при уничтожении файла номер его индексного дескриптора возвращается в список.

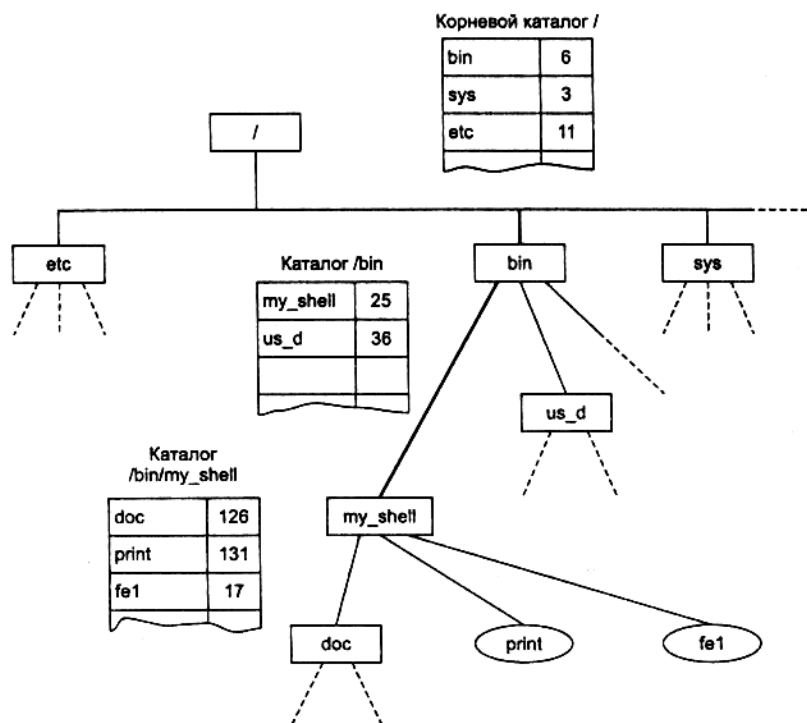
Запись о файле в каталоге состоит всего из двух полей: символьного имени файла и номера индексного дескриптора. Например, на рис. 7.16 показана информация, содержащаяся в каталоге /user.

Имя	№ индексного дескриптора
Prog 1	23
firelights	126
doc_23.txt	51
glazing.txt	17
lambda_good	

**Рис. 7.16. Структура каталога в файловой системе s5**

Файловая система не накладывает особых ограничений на размер корневого каталога, так как он расположен в области данных и может увеличиваться как обычный файл.

Доступ к файлу осуществляется путем последовательного просмотра всей цепочки каталогов, входящих в полное имя файла, и соответствующих им индексных дескрипторов. Поиск завершается после получения всех характеристик из индексного дескриптора заданного файла.



**Рис. 7.17. Поиск адреса файла по его символьному имени**

Рассмотрим эту процедуру на примере файла `/bin/my_shell/print`, входящего в состав файловой системы, изображенной на рис. 7.17. Определение физического адреса этого файла включает следующие этапы.

- Прежде всего просматривается корневой каталог с целью поиска первой составляющей символьного имени — `bin`. Определяется номер (в данном примере — 6) индексного дескриптора каталога, входящего в корневой каталог. Адрес корневого каталога известен системе.
- Из области индексных дескрипторов считывается дескриптор с номером 6. Начальный адрес дескриптора определяется на основании известных системе номера

начального сектора области индексных дескрипторов и размера индексного дескриптора. Из индексного дескриптора 6 определяется физический адрес каталога /bin.

- □ Просматривается каталог /bin с целью поиска второй составляющей символического имени my\_shell. Определяется номер индексного дескриптора каталога /bin/my\_shell (в данном случае — 25).
- □ Считывается индексный дескриптор 25, определяется физический адрес /bin/my\_shell.
- □ Просматривается каталог /bin/my\_shell, определяется номер индексного дескриптора файла print (в данном случае - 131).
- □ Из индексного дескриптора 131 определяются номера блоков данных, а также другие характеристики файла /bin/my\_shell/print.

Эта процедура требует в общем случае нескольких обращений к диску, пропорционально числу составляющих в полном имени файла. Для уменьшения среднего времени доступа к файлу его дескриптор копируется в специальную системную область оперативной памяти. Копирование индексного дескриптора входит в процедуру открытия файла.

**Инодом** (или **индексным дескриптором**) (произносится *айнод* или *инод*) называют структуру данных в традиционных файловых системах Unix.

При создании файловой системы создаются также и структуры данных, содержащие информацию о файлах. Каждый файл имеет свой инод, идентифицируемый по номеру инода (часто называемый 'i-номером' или 'инодом'), в файловой системе, в которой располагается сам файл.

Иноды хранят информацию о файлах, такую как принадлежность владельцу (пользователю и группе), режим доступа (чтение, запись, запуск на выполнение) и тип файла. Существует определенное число инодов, которое указывает максимальное количество файлов, допускаемое определенной файловой системой. Обычно, при создании файловой системы примерно 1% ее выделяется под иноды.

Термин *инод* обычно указывает на иноды блочных устройств, управляющие постоянными файлами, каталогами и, по возможности, символическими ссылками. Подобная концепция играет важную роль при восстановлении поврежденных файловых систем.

- Номер инода заносится в таблицу инодов в определенном месте устройства; по номеру инода ядро системы может считать содержимое инода, включая указатели данных и прочий контент файла.
- Номер инода файла можно посмотреть используя команду `ls -i`, а команда `ls -l` покажет информацию, хранящуюся в иноде.

Имена файлов и содержимое каталогов

- Иноды не хранят имена файлов, только информацию об их содержимом.
- Каталоги в Unix являются списками 'ссылочных' структур, каждая из которых содержит одно имя файла и один номер инода.
- Ядро должно просматривать каталог в поисках имени файла, затем конвертировать это имя в соответствующий номер инода, в случае успеха.

Постоянные файлы должны иметь следующие атрибуты:

- Длина файла в байтах.
- ID устройства (это идентифицирует устройство, содержащее файл).
- ID пользователя, являющегося владельцем файла.
- ID группы файла.
- *Режим* файла, определяющий какие пользователи могут считывать, записывать и запускать файл.
- Timestamp указывает дату последнего изменения инода (*ctime*, *change time*), последней модификации содержимого файла (*mtime*, *modification time*), и последнего доступа (*atime*, *access time*).
- Счетчик ссылок указывают количество жестких ссылок, указывающих на инод.
- Указатели на блоки диска, хранящие содержимое файла (смотри структура указателя в иноде).

Системный вызов `stat` считывает номер инода файла и некоторую информацию из инода.

Если узел `inode` имеет ACL, то это поле указывает на теневого `inode`. В файловой системе теневые `inode` используются как обычные файлы. Каждый теневой `inode` хранит ACL в своем блоке данных. Несколько файлов с одним и тем же ACL могут ссылаться на один и тот же теневой `inode`.

## EXT3/EXT3

**ext2** или **2я расширенная файловая система** — файловая система для ядра Linux. Она не является журналируемой файловой системой и это её главный недостаток. Развитием ext2 стала журналируемая файловая система ext3, полностью совместимая с ext2.

## История

На заре развития Linux использовала файловую систему ОС Minix. Эта файловая система была довольно стабильна, но была 16-разрядной и как следствие имела жёсткое ограничение в 64 Мегабайта на раздел. Также присутствовало ограничение имени файла: оно составляло 14 символов. Эти и не только ограничения повлекли появление в апреле 1992 года «расширенной файловой системы» (extended file system), решавшей 2 главные проблемы Minix. Новая файловая система расширила ограничения на размер файла до 2 терабайт и установила предельную длину имени файла в 255 символов. Но она всё равно имела проблемы: не было поддержки отдельного доступа, временных меток модификации данных.

Решением всех проблем стала новая файловая система, разработанная в январе 1993 года. В ext2 были сразу реализованы соответствующие стандарту POSIX списки контроля доступа ACL и расширенные атрибуты файлов.

## Логическая организация файловой системы ext2

Граф, описывающий иерархию каталогов файловой системы ext2, представляет собой сеть, это достигается тем, что один файл может входить сразу в несколько каталогов.

Все типы файлов имеют символьные имена. В иерархически организованных файловых системах обычно используются три типа имен — файлов: простые, составные и относительные. Не является исключением и «вторая расширенная файловая система». Ограничения на простое имя состоят в том что, его длина не должна превышать 255 символов, а также в имени не должны присутствовать символ NUL и '/'. Ограничения на символ NUL связаны с представлением строк на языке Си, а на символ '/' с тем, что он используется как разделительный символ между каталогами. Полное имя представляет

собой цепочку простых символьных имен всех каталогов, через которые проходит путь от корня до данного файла. В файловой системе ext2 файл может входить в несколько каталогов, а значит, иметь несколько полных имен; здесь справедливо соответствие «один файл — много полных имен». В любом случае полное имя однозначно определяет файл.

Атрибутами файловой системы ext2 являются:

- Тип и права доступа к файлу;
- Владелец, группа;
- Информация о разрешённых операциях доступа к файлу;
- Времена создания, последнего доступа, последнего изменения и время последнего удаления;
- текущий размер файла;
- тип файла;
  - обычный файл;
  - каталог;
  - файл байт-ориентированного устройства;
  - файл блочно-ориентированного устройства;
  - Сокет;
  - именованный канал;
  - символическая ссылка;
- число блоков, занимаемых файлом;
- ACL
- другие

Атрибуты файлов хранятся не в каталогах, как это сделано в ряде простых файловых систем, а в специальных таблицах. В результате каталог имеет очень простую структуру, состоящую всего из двух частей: номера индексного дескриптора и имени файла.

## Физическая организация файловой системы ext2

### Структура дискового раздела

Как и в любой файловой системе UNIX, в составе ext2 можно выделить следующие составляющие:

- блоки и группы блоков;
- индексный дескриптор;
- суперблок;

Всё пространство раздела диска разбивается на блоки фиксированного размера, кратные размеру сектора — 1024, 2048 и 4096 байт. Размер блока указывается при создании файловой системы на разделе диска. Меньший размер блока позволяет экономить место на жёстком диске, но также ограничивает максимальный размер файловой системы. Все блоки имеют порядковые номера. С целью уменьшения фрагментации и количества перемещений головок жёсткого диска при чтении больших массивов данных блоки объединяются в группы блоков.

Базовым понятием файловой системы является индексный дескриптор (информационный узел), *information node*, или *inode*. Это специальная структура, которая содержит информацию об атрибутах и физическом расположении файла.

Суперблок (Superblock)
Описани группы блоков (Group Descriptors)
Битовая карта блоков (Block Bitmap)
Битовая карта индексных дескрипторов (Inode Bitmap)
Таблица индексных дескрипторов (Inode Table)
Данные (Data)

### Обобщенная структурная схема ФС ext2

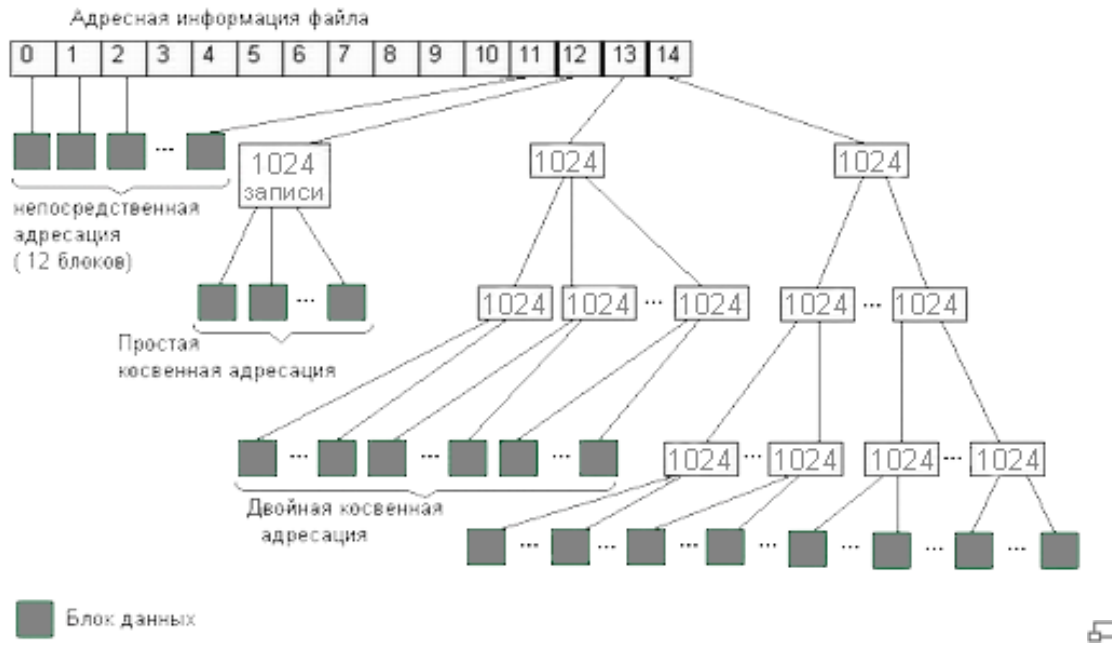
Каждая группа блоков имеет одинаковое строение. Суперблок — основной элемент файловой системы ext2. Он содержит общую информацию о файловой системе:

- 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;
- размер индексного дескриптора;
- идентификатор файловой системы.

От целостности суперблока напрямую зависит работоспособность файловой системы. Операционная система создаёт несколько резервных копий суперблока для возможности его восстановления в случае повреждения. Описание группы блоков, представляет собой массив, содержащий общую информацию обо всех блоках раздела. Битовая карта блоков — это структура, каждый бит которой показывает, отведён ли соответствующий ему блок какому-либо файлу. Если бит равен 1, то блок занят. Аналогичную функцию выполняет битовая карта индексных дескрипторов, показывая какие именно индексные дескрипторы заняты, а какие нет.

Все оставшееся место, обозначенное в таблице, как данные, отводится для хранения файлов.

## Система адресации данных



## Система адресации ФС ext2

Система адресации данных — это одна из самых существенных составных частей файловой системы. Именно система адресации позволяет находить нужный файл среди множества как пустых, так и занятых блоков на диске. Файловая система ext2 использует следующую схему адресации блоков файла. Для хранения адреса файла выделено 15 полей, каждое из которых состоит из 4 байт. Если размер файла меньше или равен 12 блокам, то номера этих кластеров непосредственно перечисляются в первых двенадцати полях адреса. Если размер файла превышает 12 блоков, то следующее 13-е поле содержит адрес кластера, в котором могут быть расположены номера следующих блоков файла. Таким образом, 13-й элемент адреса используется для косвенной адресации. При максимальном размере блока равном 4096 байт, 13-й элемент, может содержать до 1024 номеров следующих кластеров данных файла. Если размер файла превышает 12+1024 блоков, то используется 14-е поле, в котором находится номер блока, содержащего 1024 номеров блоков, каждый из которых хранят 1024 номеров блоков данных файла. Здесь применяется уже двойная косвенная адресация. И наконец, если файл включает более  $12+1024+1048576 = 1049612$  блоков, то используется последнее 15-е поле для тройной косвенной адресации.

Таким образом, описанная выше система адресации, позволяет при максимальном размере блока 4 Кб иметь файлы размера до 2 терабайт. или больше

## Назначение основных системных каталогов

`/bin` - Этот каталог содержит в основном готовые к исполнению программы, большинство из которых необходимо во время старта системы (или в однопользовательском системном режиме, используемом для отладки). Здесь хранится значительное количество общеупотребительных команд Linux

`/boot` - Содержит основные постоянные файлы для загрузки системы, в частности загружаемое ядро. Файлы из этого каталога нужны только во время загрузки системы

`/dev` Каталог специальных файлов или файлов устройств.

`/etc` Этот каталог и его подкаталоги содержат большинство данных, необходимых для начальной загрузки системы, и основные конфигурационные файлы. В `/etc` находятся, например, файл `inittab`, определяющий загружаемую конфигурацию и файл паролей пользователей `passwd`. Часть конфигурационных файлов может находиться и в `/usr/etc`. Каталог `/etc` не должен содержать двоичных файлов (их следует перенести в `/bin` или `/sbin`).

`/home` Обычно в этом каталоге находятся каталоги пользователей

`/lib` Этот каталог содержит разделяемые библиотеки функций, необходимых компилятору языка C, и модули (драйверы устройств). Даже если в системе не установлен компилятор языка C, разделяемые библиотеки необходимы, поскольку они используются многими прикладными программами. Они загружаются в память по мере необходимости выполнения каких-то функций, что позволяет уменьшить объем кода программ — в противном случае один и тот же код многократно повторялся бы в различных программах

`/lost+found` - Этот каталог используется при восстановлении файловой системы командой `fsck`. Если `fsck` обнаруживает файл, родительский каталог которого определить невозможно, она помещает такой файл в каталог `/lost+found`. Поскольку родительский каталог потерян, то таким файлам присваиваются имена, совпадающие с номерами их индексных дескрипторов

`/mnt` - Это точка монтирования для временно монтируемых файловых систем. Если на компьютере запускается поочередно Linux и MS-DOS, то этот каталог обычно используется, чтобы монтировать файловую систему MS-DOS. Если вы имеете привычку монтировать несколько дополнительных носителей, например, дискеты, CD-ROM, дополнительный жесткий диск и т. д., то можно создать в нем соответственно дополнительные подкаталоги для каждого носителя

`/proc` - Это точка монтирования для файловой системы `proc`, которая обеспечивает информацию о выполняющихся процессах, ядре, оборудовании вычислительной установки и т. д. Специальные файлы из этого каталога используются для получения и передачи данных ядру

`/root` - Это домашний каталог суперпользователя. Обратите внимание на то, что он расположен не там, где располагаются личные каталоги остальных пользователей (`/home`)

`/sbin` - Подобно каталогу `/bin`, содержит в основном исполняемые файлы — программы и утилиты ОС, используемые в процессе загрузки и запускаемые системным администратором. В стандарте FHS говорится, что в этот каталог надо помещать те исполняемые файлы, которые используются после успешного подключения файловой системы `/usr`. Минимальное содержимое этого каталога включает программы `clock`, `getty`, `init`, `update`, `mkswap`, `swapon`, `swapoff`, `halt`, `reboot`, `shutdown`, `fdisk`, `fsck.*`, `mkfs.*`, `lilo`, `arp`, `ifconfig`, `route`

`/tmp` - Каталог для временных файлов. В любой момент суперпользователь может удалить файлы из этого каталога без большого ущерба для остальных пользователей. Однако не стоит удалять файлы из этого каталога, если вам не стало ясно, что конкретный файл или группа файлов мешают продолжению продуктивной работы на машине. Система сама периодически очищает этот каталог, поэтому не следует хранить тут файлы, которые вам могут понадобиться в дальнейшем

`/usr` - Этот каталог огромен и его структура в основном повторяет структуру корневого каталога. В его подкаталогах находятся все основные приложения. В соответствии со стандартом FHS рекомендуется выделять для этого каталога отдельный раздел диска или вообще располагать его на сетевом диске, общем для всех компьютеров в сети. Такой раздел или диск монтируют только для чтения и располагают в нем общие конфигурационные и исполняемые файлы, документацию, системные утилиты и библиотеки, а также включаемые файлы (файлы типа `include`) `/usr/bin` Готовые к исполнению программы — утилиты и приложе-

ния, которые часто вызывают обычные пользователи.

`/var` Этот каталог содержит файлы, в которых сохраняются различные переменные данные, определяющие конфигурацию некоторых программ при следующем запуске или временно охраняемую информацию, которая будет использоваться позже в ходе текущего сеанса. Объем данных в этом каталоге может сильно изменяться, поскольку он содержит, например, файлы

## Символические ссылки

*In -s имя\_файла\_или\_каталога имя\_ссылки*

## Команды для работы с файлами и каталогами

### Команды *chown* и *chgrp*

Эти команды служат для смены владельца файла и группы файла. Выполнять смену владельца может только суперпользователь, смену группы может выполнить сам владелец файла или суперпользователь. Для того чтобы иметь право сменить группу, владелец должен дополнительно быть членом той группы, которой он хочет дать права на данный файл. Формат этих двух команд аналогичен:

```
[root]! chown vasja имя-файла
```

```
[root]! chgrp usersgrp имя-файла
```

### Команда *mkdir*

Команда `mkdir` позволяет создать подкаталог в текущем каталоге. В качестве аргумента этой команде надо дать имя создаваемого каталога. Во вновь созданном каталоге автоматически создаются две записи: `.` (ссылка на этот самый каталог) и `..` (ссылка на родительский каталог). Чтобы создать подкаталог, вы должны иметь в текущем каталоге право записи. Можно создать подкаталог не в текущем, а в каком-то другом каталоге, но тогда необходимо указать путь к создаваемому каталогу:

```
[user]$ mkdir /home/kos/book/glava5/part1
```

### Команда *cat*

Команда `cat` часто используется для создания файлов (хотя можно воспользоваться и командой `touch`). По команде `cat` на стандартный вывод (то есть на экран) выводится содержимое указанного файла (или нескольких файлов, если их имена последовательно задать в качестве аргументов команды). Если вывод команды `cat` перенаправить в файл, то можно получить копию какого-то файла:

```
[user]$ cat file1 > file2
```

Собственно, первоначальное предназначение команды `cat` как раз и предполагало перенаправление вывода, т. е. эта команда создана для конкатенации, т. е. объединения нескольких файлов в один:

```
[user]$ cat file1 file2 ... fileN > new-file
```

Именно возможности перенаправления ввода и вывода этой команды и используются для создания новых файлов. Для этого на вход команды `cat` направляют данные со стандартного ввода (то есть с клавиатуры), а вывод команды — в новый файл:

```
[user]$ cat > newfile
```

После того как вы напечатаете все, что хотите, нажмите комбинацию клавиш `<Ctrl>+<D>` или `<Ctrl>+<C>`, и все, что вы ввели, будет записано в `newfile`. Конечно, таким образом создаются, в основном, короткие текстовые файлы.

### Команда *cp*

Хотя для копирования файлов иногда пользуются командой `cat`, но в Linux существует для этого специальная команда `cp`. Ее можно применять в одной из двух форм:

```
[user]$ cp [options] source destination
```

```
[user]$ cp [options] source_directory new_directory
```

В первом случае файл или каталог `source` копируется, соответственно, в файл или каталог `destination`, а во втором случае файлы, содержащиеся в каталоге `source_directory`, копируются в каталог `new_directory`.

### Команда *mv*

Если вам необходимо не скопировать, а переместить файл из одного каталога в другой, вы можете воспользоваться командой `mv`. Синтаксис этой команды аналогичен синтаксису команды `cp`. Более того, она сначала копирует файл (или каталог), а только потом удаляет исходный файл (каталог).

```
[user]$ mv oldname newname
```

Но учтите, что команда `mv` не позволяет переименовать сразу несколько файлов (используя шаблон имени), так что команда `mv *.xxx *.uuu` не будет работать. При использовании команды `mv`, также как и при использовании `cp`, не забывайте применять опцию `-i` для того, чтобы получить предупреждение, когда файл будет перезаписываться.

### Команды *rm* и *rmdir*

Для удаления ненужных файлов и каталогов в Linux служат команды `rm` (удаляет файлы) и `rmdir` (удаляет пустой каталог). Для того чтобы воспользоваться этими командами, вы должны иметь право записи в каталоге, в котором расположены удаляемые файлы или каталоги. При этом полномочия на изменение самих файлов необязательны.

#### Команды *more* и *less*

Для просмотра содержимого файлов (конечно, текстовых) используются команды `more` и `less` (или текстовые редакторы). Команда-фильтр `more` выводит содержимое файла на экран отдельными страницами размером как раз в целый экран. Для того чтобы увидеть следующую страницу, надо нажать на клавишу пробела. Нажатие на клавишу `<Enter>` приводит к смещению на одну строку.

### Команда *find* и символы шаблонов для имен файлов

Еще одной часто используемой командой для работы с файлами в Linux является команда поиска нужного файла `find`. Команда `find` может искать файлы по имени, размеру, дате создания или модификации и некоторым другим критериям.

Общий синтаксис команды `find` имеет следующий вид:

```
find [список_каталогов] критерий_поиска
```

Можно сократить объем поиска, если задать вместо одного корневого каталога список из нескольких каталогов (естественно, тех, в которых может находиться искомый файл):

```
[user]$ find /usr/share/doc /usr/doc /usr/locale/doc -name instr.txt
```

### Команда *split*— разбиваем файл на несколько частей

Иногда бывает необходимо разбить один большой файл на несколько файлов меньшего объема.

## Программа tar

tar расшифровывается как Tape ARchiver, он не сжимает данные, а лишь объединяет их в единый файл с последовательным доступом для последующей записи на ленту. По умолчанию этот архивный файл создается на ленточном накопителе, точнее на устройстве /dev/rmt0.

### 4.7.2. Программа gzip

Хотя программа tar создает архивы, она, как было сказано, не сжимает архивы, а просто соединяет отдельные файлы в единый архивный файл. Для сжатия этого файла часто применяют команду gzip. В простейшем случае она вызывается в следующем формате:

```
[user]$ gzip файл
```

В командной строке можно указать сразу несколько имен файлов или шаблон имени файла. Но в этом случае каждый из указанных файлов будет заархивирован отдельно (общий архив не создается).

Для того чтобы распаковать архив, используйте команду

```
[user]$ gzip -d файл_архива
```

ИЛИ

```
[user]$ gunzip файл_архива
```

Исходные файлы после сжатия удаляются, остается только архивный файл (файлы перемещаются в архив), а при разархивации удаляется архив.