Урок 3

Файловые системы

Принципы организации файловых систем. Разновидности файловых систем. Интересные особенности: жёсткие и символические ссылки. Файловые системы в GNU/Linux.

- Введение
- Принципы работы с жёсткими дисками
  - CHS
    - Large
  - LBA
  - Кластер
- Файловая система на примере FAT
  - Загрузочный сектор
  - FAT таблица
  - Директории
  - Имя файла и расширение
  - Корневой каталог
  - Развитие (VFAT, exFAT)
- Разделы, форматирование
  - MBR
  - GPT
  - Форматирование
    - Низкоуровневое форматирование
    - Высокоуровневое форматирование

Восстановление информации

Удаление без возможности восстановления

Файловые системы

Нежурналируемые файловые системы

Журналируемые файловые системы

Атрибуты файлов в файловых системах

Интересные возможности файловых систем и ОС в работе с ФС

Директория как диск

Монтирование

Ярлыки

Символические ссылки

Символические ссылки в Debian

Жёсткие ссылки

Файловые потоки (альтернативные данные файлов)

Снимки (снапшоты)

Управление томами

Работа с файловыми системами в Linux

Устройство файловой системы на примере ext2/ext4

LVM

Практическое задание

Дополнительные материалы

Используемая литература

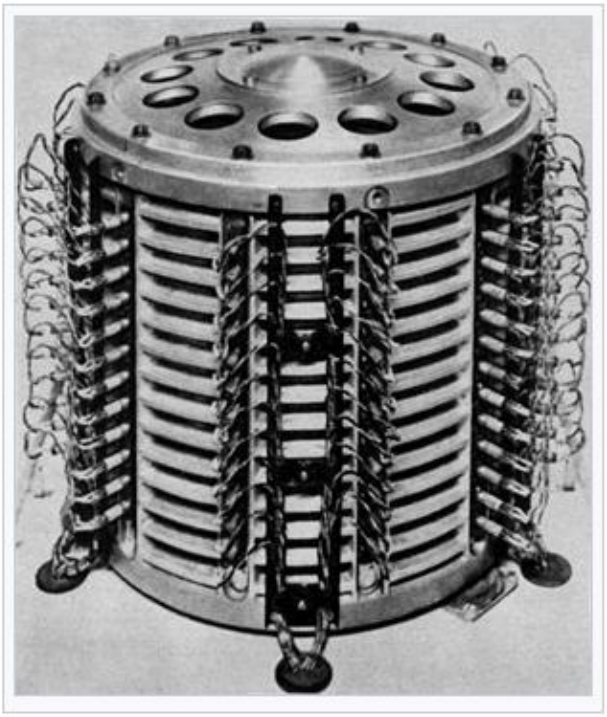
Введение

Задача хранения информации появилась достаточно давно. Перфоленты и перфокарты возникли раньше вычислительных машин и использовались для хранения музыкальных произведений для автоматических инструментов, рисунков для ткацких станков и так далее.

Хранение цифровой записи на бумаге не было надежным и удобным (хотя похожий тип записи все равно продолжает использоваться для коротких сообщений, а также в виде штрих-кодов и QR-кодов).

Накопители на магнитных лентах.

Дальнейшее развитие привело к использованию магнитных лент, а затем и магнитных барабанов.



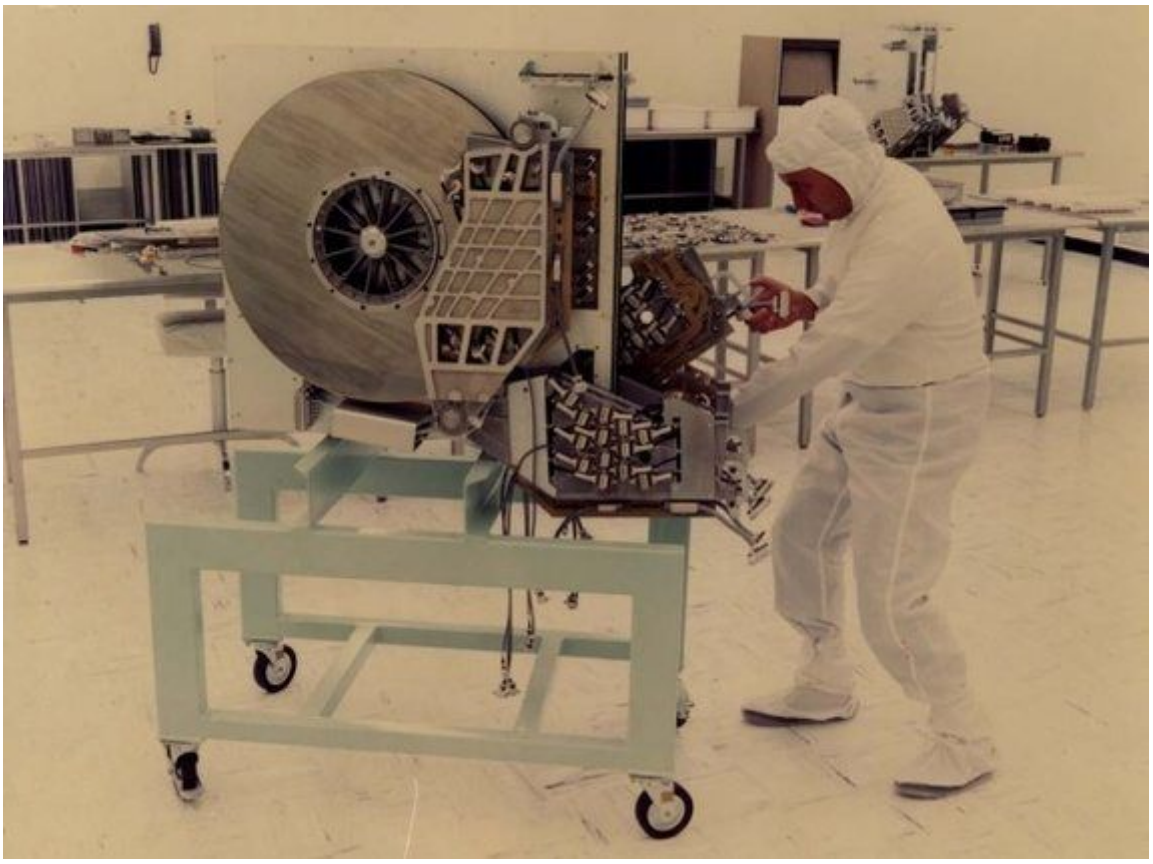
Магнитные барабаны:
Слева: для польского компьютера ZAM-41.
Справа: советского производства.

Источники:
https://commons.wikimedia.org/w/index.php?curid=239809
https://commons.wikimedia.org/w/index.php?curid=6358185

Магнитный барабан представлял собой вращающийся цилиндр, внешняя поверхность которого была покрыта ферромагнитным слоем – он образовывал магнитные дорожки, опоясывавшие цилиндр. Каждая дорожка считывалась отдельной головкой.

В отличие от современных жёстких дисков, информация записывалась на внешней стороне барабана, а не на плоской поверхности диска.

Магнитные барабаны были довольно громоздкими, и через некоторое время их вытеснили жёсткие диски.



250 MB HDD 1979 года для ibm 1956.

Жёсткие диски (HDD – Hard Disk Drive) включали в себя несколько дисков, головки и контроллер и предназначались для постоянного хранения информации на компьютере.

Похожим образом устроены были гибкие диски (дискеты), но, в отличие от жестких дисков, в них не было ни головок, ни контроллера – таким образом, дисковод с контроллером и головками работал со сменными носителями.

В дальнейшем дискеты были вытеснены CD-ROM (CD-R, CD-RW), которые применялись сначала в качестве носителя дистрибутивов, а затем и для архивирования и частично обмена информацией (CD-RW). CD-RW не позволяли быстро копировать файлы и поэтому обычно использовались для обмена большими объёмами информации.

В качестве механизма для обмена информацией дискеты были окончательно вытеснены флеш-накопителями. Флеш-накопитель не содержит движущихся частей и включает в себя матрицу, в

известной степени подобную матрице оперативной памяти. При этом, в отличие от RAM, во флеш-памяти каждая ячейка содержит затвор, который не требует постоянного питания. Ограничением флеш-памяти является количество циклов перезаписи. По сравнению с HDD флеш-накопители более быстрые, но пока ещё более дорогие. Со временем ёмкости флеш-накопителей растут, а стоимость падает. Если изначально флеш-технология использовалась в ПЗУ, потом для внешних накопителей, то сейчас флеш-накопители всё чаще используются и в качестве постоянных дисков как альтернатива HDD и SSD (solid state disk).

Когда компьютер включен, оперативная память и жёсткий диск могут применяться для хранения информации и частично разделяют эту функцию: жёсткий диск служит для выполнения задач оперативной памяти (файл или раздел подкачки), оперативная память может эмулировать жесткий диск (что часто применяется в Live-CD комплектах ОС): для системы диск выглядит как действительно существующий, но при выключении питания данные не сохраняются. Одним из вариантов загрузки операционных систем (с дискеты, Live-CD; он даже применяется в Linux: initrd) является распаковка из архива образа диска и работа с таким RAM-диском.

Итак, недостатки RAM-диска – это недолговременная память, потребность в питании, а достоинство – очень высокая скорость работы.

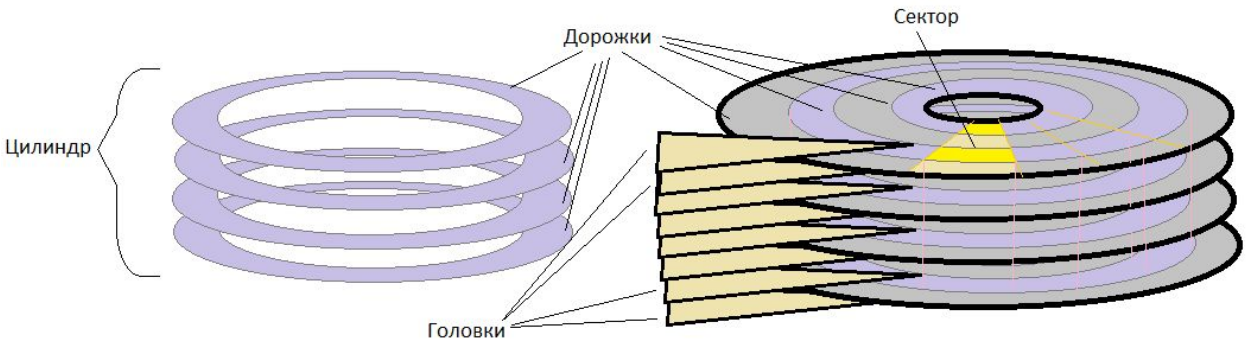
Существуют варианты использования RAM-памяти как долговременной. Если снабдить микросхемы памяти аккумулятором и контроллером, который позволит ЭВМ воспринимать данную память как диск, получится реализация быстрого устройства, сохраняющего данные условно без питания. Такая технология носит название i-RAM.

i-RAM. Обратите внимание на слоты для планок оперативной памяти и аккумулятор.

В настоящее время широко применяются HDD- и SSD-накопители.

Принципы работы с жёсткими дисками

Жёсткий диск – HDD (Hard Disk Drive) состоит из набора дисков и головок, причём на каждом диске присутствуют кольцевые дорожки. Для доступа к дорожкам используется понятие «сектор».



CHS – цилиндр, головки, сектора

На диск с обеих сторон нанесено ферромагнитное покрытие, поэтому один диск обслуживают две головки – таким образом, головок в два раза больше, чем дисков. На каждой из сторон диска имеются замкнутые (кольцевые) дорожки. Благодаря шаговому двигателю привод головок может перемещаться от внешних дорожек-колец к внутренним.

Шаговый двигатель перемещает головки на нужный радиус.

CHS

Итак, головки на диске перемещаются между дорожками. Привод головок перемещает головки только от диска к диску (точнее, все головки перемещаются от цилиндра к цилиндру). В выключенном состоянии головки находятся за пределами диска в парковочной зоне, чтобы поверхность диска не портилась. Более того, во время работы головки также не касаются диска – они парят в потоке воздуха.

Головки движутся синхронно – таким образом, речь идет не о чтении конкретной дорожки, а о чтении дорожек всех дисков одного радиуса разом. Дорожки одного радиуса называются цилиндром.

Для позиционирования информации на дорожках/цилиндрах используется понятие «сектор»: чтобы позиционировать сектор, нужно указать цилиндр, номер головки и собственно сектор. Такая система получила название CHS (Cylinder, Head, Sector).

Логично, что такая система не может быть применена, например, к флеш-дискам.

Large

Рост объема дисков не мог позволить BIOS работать с физическими параметрами диска. Тогда контроллеры стали сообщать системе «виртуальные» данные о дисках, головках, секторах. В результате число головок оказывалось больше, чем могло быть на самом деле.

Одна из таких систем получила именование Large. В Large логических головок было в два раза больше, чем физических, а цилиндров, напротив, в два раза меньше.

Контроллер преобразовывал данные из логического представления в физическое, однако это приводило к проблемам в работе низкоуровневых средств для работы с дисками. Так возникла новая схема работы – LBA (Logical Block Addressing).

LBA

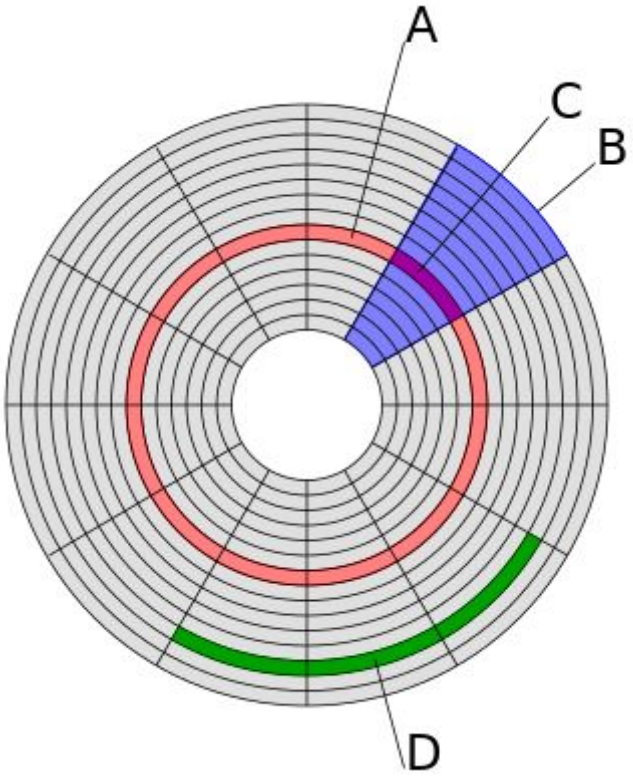
LBA позволял пронумеровать все блоки: теперь контроллер не сообщает физическую геометрию диска.

Номера блоков рассчитываются по формуле

```
Адрес_блока_LBA = номер_цилиндра * Кол-во_головок * Количество_секторов_в_дорожке + номер_головки * Количество_секторов_в_дорожке + Номер_сектора - 1
```

Таким образом, первый блок (Цилиндр 0, Головка 0, Сектор 1) получает номер 0.

Кластер



A – дорожка, B – геометрический сектор, C – сектор дорожки, D – кластер.

Кластер, как правило, состоит из нескольких соседних секторов дорожки, хотя может состоять и из одного сектора. Часто файловые системы оперируют кластерами, и кластер является минимальным размером, выделяемым для файла. Например, в DOS для FAT16, если размер кластера составлял 2 Кб, а требовалось записать на диск файл размером в 4 байта, для записи фактически отводился весь кластер (несмотря на то, что в оглавлении файловой системы было указано, что размер файла – 4 байта, физически файл занимал все 2 Кб).

Файловая система на примере FAT

FAT (File Allocation Table, «таблица размещения файлов») – известная и простая файловая система, созданная для MS DOS Биллом Гейтсом и Марком МакДональдом в 1976-1977 годах. Она по сей день используется для флеш-накопителей.

Рассмотрим устройство FAT на примере FAT12.

Нулевой сектор содержит Boot Record, содержащий информацию о файловой системе. Затем следует Таблица размещения файлов (FAT), которая дублируется с целью увеличения надежности. Первая и

вторая таблицы занимают 1-9 и 10-18 секторы соответственно. Далее несколько секторов отводится под корневой каталог (единственный каталог, который обязательно должен присутствовать и отличается от других некоторыми особенностями). Что интересно, первые версии FAT не позволяли создавать подкаталоги. После корневого каталога размещаются секторы с содержимым файлов.

Загрузочный сектор

Загрузочный сектор содержит информацию о файловой системе. Его формат может варьироваться в разных версиях FAT.

Значения байтов с порядковыми номерами:

- 0-3 – JMP BootStart;
- 4-10 – игнорируются;
- 11-12 – количество байтов в каждом секторе;
- 13 – количество секторов в каждом кластере;
- 14-15 – количество зарезервированных секторов;
- 16 – количество FAT-таблиц;
- 17-18 – максимальное число файлов и папок в корневой директории;
- 19-20 – общее число секторов;
- 21 – игнорируется;
- 22-23 – количество секторов, которое требуется для каждой FAT таблицы;
- 24-25 – количество секторов на дорожке;
- 26-27 – количество головок;
- 28-31 – игнорируется;
- 32-35 – игнорируется в FAT12;
- 36-37 – игнорируется;
- 38 – boot signature – загрузочная сигнатура;
- 39-42 – id тома;
- 43-53 – метка тома;
- 54-61 – тип файловой системы;
- 62 – далее следует начало программы BootStart (которая передаст управление в IO.SYS, что интересно, не опираясь на оглавление, а физически считав следующий кластер после таблиц FAT).

Здесь и далее многие поля заполняются до границы поля пробелами. Это касается метки, названия файловой системы; более того, такой же подход используется и в именах файлов, где под имя и расширение выделено по 8 и 3 байта соответственно.

FAT таблица

FAT – это таблица, которая содержит массив индексных указателей («ячеек»), соответствующих кластерам области данных. Каждый элемент содержит порядковый номер следующего кластера файла либо установлен в специальное значение, которое может меняться в зависимости от версии FAT.

Вот более подробный список значений и того, что под ними подразумевается:

- 0x000 – не используется;
- 0xFF0-0xFF6 – зарезервирован;
- 0xFF7 – плохой кластер (поврежден и не используется);
- 0xFF8-0xFFF – последний кластер в файле.

Директории

Директории, включая корневую, хранятся на диске как файлы (фактически являясь особого вида файлами) и занимают один или несколько кластеров. Каждый кластер (512 байт) содержит указатели на 16 файлов, то есть по 32 байта на каждый указатель. Указатель описывает файл на диске или поддиректорию, которая тоже является особого вида файлом, а также указывает на кластер, где находится файл.

Номерам байтов соответствовало следующее содержимое:

- 0-7 – имя файла (8 байт);
- 8-10 – расширение (3 байта);
- 11 – атрибуты;
- 12-13 – зарезервировано, не используется;
- 14-15 – время создания;
- 16-17 – дата создания;
- 18-19 – дата последнего использования;
- 20-21 – игнорируется в FAT12;
- 22-23 – дата последнего изменения;
- 24-25 – время последнего изменения;
- 26-27 – указание на первый кластер файла или директории;
- 28-31 – размер файла в байтах.

Особым образом записывались даты. Два байта содержали следующие значения:

- 0-4 – день месяца, допускаются значения 1-31;
- 5-8 – месяц года, допускаются значения 1-12;
- биты 9-15 – год считая от 1980 («эпоха MS-DOS»); возможны значения от 0 до 127 включительно, то есть 1980-2107 годы.

В FAT не используется специальный признак существования файла: чтобы указать, что файл удалён, используется первый байт имени. При удалении он устанавливается в значение 0x29, а кластеры помечаются как свободные. Так как указание на первый кластер остаётся, файл возможно восстановить в том случае, если он занимал несколько соседних секторов. Такие программы, как Undelete, Unerase и встроенное восстановление файлов в Dos Navigator, запрашивали первую, затертую букву и выводили информацию о восстановлении: например, если указанный кластер был уже задействован другим файлом, восстанавливать уже было нечего.

Если следующая запись при чтении каталога начиналась с 0x00 (условно – первый байт в позиции имени), это означало, что больше указателей в каталоге нет.

Значения байта атрибутов:

- 0x01 – только чтение (Read Only);
- 0x02 – скрытый (Hidden. io.sys, msdos.sys и некоторые другие файлы в корневом каталоге имели атрибут, по умолчанию не отображались при выводе dir);
- 0x04 – системный (System, io.sys, msdos.sys в корневом каталоге имели этот атрибут);
- 0x08 – Volume Label (метка тома, хранится так же, как и файл. В отличие от остальных, имя не делится на имя и расширение, потому в его значении может быть до 11 символов. Кроме того, в комбинации с предыдущими атрибутами метка указывает, что это не запись отдельного файла, а часть длинного имени LFN);
- 0x10 – указание на то, что файл является папкой;
- 0x20 – архивный (Archive); использовался нечасто, но предполагалось, что его будут применять программы архивации для указания, что файл был заархивирован или подлежит архивации;

- 0x40 – не используется (зарезервировано для устройств);
- 0x80 – не используется.

Значения могут быть комбинированы: например, если нужно, чтобы файл представлял собой скрытую системную папку, соответствующий байт устанавливается в значение 0x16, потому что 0x16 = 0x02 + 0x04 + 0x10.

Для расширения LFN используется признак 0x0F=0x01+0x02+0x04+0x08. Каждая такая запись является не файлом, а частью имени длинного файла.

Утилита attrib позволяла устанавливать четыре из указанных атрибутов: R – только для чтения, A – архивный, S – системеный, H – скрытый.

Имя файла и расширение

В FAT12 выделяется 8 символов под имя файла и 3 символа под расширение – так называемый формат 8.3. Если имя файла занимает меньше 8 символов, оставшиеся заполняются пробелами (ASCII 32 или 0x20), то же касается расширений. Директории также могут иметь расширение. Точка никак не хранится, но при работе в DOS необходима. Расширение может быть пустым, в то время как имя файла должно содержать хотя бы один символ.

Все имена и расширения хранятся в ВЕРХНЕМ РЕГИСТРЕ, при этом для работы регистр значения не имеет: COMMAND.COM, Command.Com и command.com – одно и то же имя файла. В одной директории не может быть одновременно двух файлов (обычных или директорий) с одинаковым именем и расширением: если есть файл config.sys, директорию config.sys уже создать нельзя.

Примеры:

Имя файла	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
command.com	C	O	M	M	A	N	D		C	O	M
io.sys	I	O							S	Y	S
msdos.sys	M	S	D	O	S				S	Y	S
hello.c	H	E	L	L	O				C		
DOS	D	O	S								
WINDOWS	W	I	N	D	O	W	S				
descript.ion	D	E	S	C	R	I	P	T	I	O	N

Имена файлов даны как при выводе dir или в файловом менеджере: файлы – в нижнем регистре, директории – в верхнем. На самом деле это условность, так как файловая система здесь регистронезависима. Иногда с файлами поставлялся файл descript.ion: фактически его имя было descript, нестандартное разрешение – ion. Обычно он содержал описание файлов и/или ASCII-арт.

Также в директории присутствуют специальные файлы «.» и «..».

«.» является ссылкой на текущую директорию (cd . не меняет текущую директорию). «..» является ссылкой на родительскую директорию (cd .. переходит на уровень выше).

Корневой каталог

По структуре корневой каталог аналогичен другим каталогам (директориям, папкам), но на него не бывает ссылки. Он не имеет ни имени (в системе к нему можно обратиться по имени «\», cd \ – перейти в корень диска), ни атрибутов. По той же причине корневой каталог имеет фиксированный размер. Это накладывает ограничение на количество размещаемых в корне файлов и при этом позволяет передать управление в io.sys, не опираясь на сам корневой каталог, учитывая только физическое расположение io.sys и msdos.sys после корневого каталога.

Развитие (VFAT, exFAT)

VFAT – совместимое с FAT расширение, добавляющее записи в формате LFN (Long File Name) в кодировке Unicode-16. На одну SFN (Short File Name – традиционную для FAT) запись добавляется одна или несколько LFN-записей. С появлением такого расширения имя файла могло быть до 255 символов длиной; кроме того, теперь точка включалась в состав имени. Таким образом, файл с длинным именем в Windows 95 в MS DOS или Windows 3.11 выглядел как РАБОЧИ~1 – это SFN-запись.

exFAT (extended FAT) – это расширенная версия FAT, оптимизированная под флеш-накопители, которые имеют ограниченное число циклов перезаписи. FAT как нежурналируемая система более щадяще относится к перезаписям, не тратя дополнительные операции на журнал. В exFAT были созданы дополнительные механизмы для уменьшения числа перезаписи (исключение перезаписи одного и того же сектора, улучшение распределения записанных кластеров благодаря бит-карте свободного места и т.д.)

Разделы, форматирование

Для установки нескольких операционных систем на один жёсткий диск или для разделения хранимой информации (системная, файлы, архивы) применяется разметка диска на разделы. Каждый раздел выглядит для системы как отдельный диск и имеет в DOS и Windows отдельную букву диска (например, диски C: и D:).

Для управления разделами используется таблица разделов (partition table) – служебная область в начале жёсткого диска, которая описывает дисковую разметку. Каждый раздел может содержать свою файловую систему и использоваться для загрузки размещённой в ней операционной системы, если в этом есть потребность.

MBR

MBR (Master Boot Record – главная загрузочная запись) содержит исполняемый код, необходимый для передачи управления загрузчик и таблицу разделов (partition table). Она может содержать только 4 первичных (primary) раздела, а при необходимости увеличения их количества вместо одного из первичных разделов можно создавать расширенный раздел (extended partition). Такой раздел будет содержать внутри себя ещё несколько (до 16) разделов, называемых логическими (logical). С созданием разделов и установленных на нем ОС связаны определённые ограничения: например, Windows необходимо обязательно устанавливать в первичный раздел, помеченный как активный (загрузочный). Для разделов Linux таких ограничений сейчас нет.

GPT

GPT (GUID Partition Table) в настоящее время уже не содержит загружаемый код: эти функции отнесены к UEFI. Более того, GPT является частью стандарта UEFI. Тем не менее блок MBR все равно присутствует в начале для совместимости и защиты от повреждения утилитами, не умеющими работать с GPT, но понимающими MBR. В совместимом MBR указан один раздел, охватывающий весь диск.

GPT не накладывает ограничений на разделы, поэтому такие понятия, как расширенные и первичные разделы, в работе с ним не используются.

Fdisk не работает с GPT, вместо него следует использовать gparted.

Форматирование

Различается низкоуровневое и высокоуровневое форматирование.

Низкоуровневое форматирование

Низкоуровневое форматирование создает разметку дорожек и секторов на диске. В настоящее время, как правило, эта часть выполняется в заводских условиях и пользователям недоступна.

Высокоуровневое форматирование

Высокоуровневое форматирование записывает на диск (в раздел) файловую систему. После этого диск доступен для использования в операционных системах, поддерживающих файловую систему, в которой раздел был отформатирован.

Создание загрузочной записи может выполняться как при форматировании, так и отдельно специальными утилитами.

Восстановление информации

Так как высокоуровневое форматирование фактически изменяет только оглавление диска, потерянные данные в результате ошибочного форматирования теоретически можно восстановить. При этом возможны частичные потери иерархии/структуры/данных. Информацию позволяют восстановить такие программы, как GetDataBack для Windows (в данном случае для NTFS). В случае, когда вышел из строя контроллер диска, восстановление заключается в технической операции с использованием устройства-донора.

Удаление без возможности восстановления

С учётом особенностей системы для удаления конфиденциальных данных недостаточно удалить файл или отформатировать диск. Более того, специальные технические средства позволяют восстановить данные даже после двукратной перезаписи секторов. Потому существуют утилиты, которые не просто удаляют файлы и разделы, но и принудительно перезаписывают нулями или случайной информацией трижды использованные ранее секторы.

Файловые системы

Файловые системы можно разделить на две группы в зависимости от того, контролируется в них факт успешного завершения операций с диском или нет.

Нежурналируемые файловые системы

По происхождению нежурналируемые системы являются более ранними, чем журналируемые. Примеры таких файловых систем – FAT и EXT2.

Если в процессе записи действие будет остановлено, возникнут ошибки файловой системы, повреждённые файлы и директории. Именно поэтому при выключении Windows без специальной опции при следующем старте запускалась проверка системы. В FAT использовался специальный бит, который указывал, нужно проверять систему или нет.

Наверняка многие, кто переходил от использования FAT к NTFS, обращали внимание, что проблем с проверкой больше не возникало независимо от корректности завершения системы. Обусловлено это тем, что NTFS относится к журналируемым файловым системам. Они более устойчивы к непредвиденным событиям (таким, как неожиданное выключение питания компьютера), но при этом требуют больше ресурсов. Потому EXT2 считается эталоном производительности файловой системы. В настоящее время использование её ввиду нежурналируемости оправданно в качестве разделов для временных файлов, аналогичных программных кешей (не путать со SWAP – это отдельная, особого образа файловая система, доступная только для ядра Linux: в Windows, напротив, используется файл). Также EXT2 используется там, где требуется быстродействие и совместимость – например, для раздела загрузки /boot при использовании LVM. Так как GRUB не умеет работать с LVM, для /boot оставляют особый раздел в EXT2.

Кроме того, нежурналируемые файловые системы используются на флеш-накопителях (exFAT, ext2) благодаря меньшему использованию операций перезаписи из-за отсутствия журнала.

Журналируемые файловые системы

Журналируемые файловые системы работают по транзакционному принципу. В структуре раздела имеется журнал (Journal), в котором записываются действия. Транзакция – это набор действий, выполняемых как одно целое, то есть оно или полностью завершено, или нет. Если транзакция будет прервана, то в соответствии с записью в журнале эти изменения откатятся, и транзакция будет считаться невыполненной. Такой подход привносит в работу определённые удобства и надежность.

Журналируемые файловые системы работают медленнее, чем если бы они были отформатированы в аналогичные нежурналируемые. Затраты на чтение и запись делают журналируемые файловые системы малопригодными для использования во флэш-накопителях.

К журналируемым файловым системам относятся NTFS, EXT4 и многие другие современные файловые системы.

Атрибуты файлов в файловых системах

Набор атрибутов файлов полностью определяется форматом файловой системы. Например, в DOS имя состояло строго из имени и расширения. Является ли файл системным или скрытым, определялось атрибутами, а возможность его выполнения – расширением (.com, .exe, .bat)

В Linux расширений как таковых нет, возможность выполнения файла определяется его атрибутом, а понятия скрытого или системного файла отсутствуют. Системность файла определяется правами

доступа и владельцем файла, а также расположением файла в иерархии файловой системы Linux. О том, что файл является скрытым, говорит точка в начале его названия (.rc, .rc.local, .htaccess – примеры скрытых файлов). При этом в Linux могут использоваться расширения на уровне приложений – как дань традиции, так и для опознавания файлов приложениями: это расширения .c, .s, .cpp, .ko и т.д.

Интересные возможности файловых систем и ОС в работе с ФС

Рассмотрим возможности, существующие для разных файловых систем.

Директория как диск

Команда subst в DOS и Windows позволяла использовать директорию как отдельный диск.

Пример:

```
C:\> subst z: c:\Users
C:\> z:
C:\> dir
C:\> subst /D z:
```

Монтирование

В Linux используется только виртуальная файловая система. Остальные файловые системы монтируются. Точка монтирования – пустая директория, в которую после монтирования будет отображаться корень примонтированной файловой системы.

```
# [ -d /mnt/cdrom ]||mkdir /mnt/cdrom
# cd /mnt/cdrom
# mount /dev/cdrom /mnt/cdrom
# cd /mnt/cdrom
# cd /
# umount /mnt/cdrom
```

В NTFS также поддерживаются точки монтирования. В среде Windows существует два вида точек монтирования: точка монтирования каталога (англ. junction point) и точка монтирования тома (англ. volume mount point). Создание точек монтирования первого типа осуществляется через консольную команду mklink /J, создание точек монтирования второго типа – через команду mountvol.

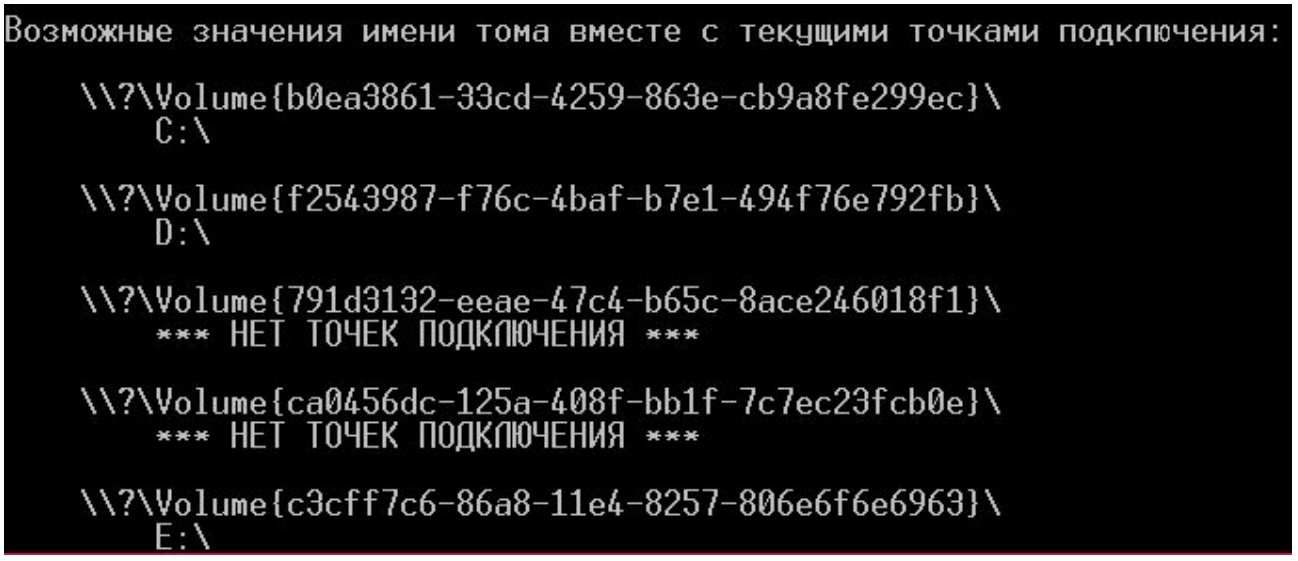
```
C:\> mklink /J c:\tmp\disk_d d:\
C:\> dir c:\tmp\disk_d
```

```
C:\>mklink /J c:\tmp\disk_d d:\
соединение создано для c:\tmp\disk_d <<===>> d:\
```

В этом случае, в отличие от монтирования, в Linux создавать пустую директорию не нужно.

Для монтирования точек монтирования в Windows используется команда mountvol. По умолчанию она выводит список опций и перечень точек монтирования:

```
c:\> mountvol
```



Ярлыки

Ярлыки не относятся к возможностям операционной системы, и возможность работы с ними осуществляется на уровне приложения (как правило, менеджера рабочего стола).

Изначально в Windows присутствовали файлы PIF для настроек параметров запуска DOS приложений, к которым добавились файлы .LNK, указывающие на местоположение программы или документа. В окружении рабочего стола клик на ярлык приводил к тем же последствиям, что и клик на значок самой программы, но при удалении файла ярлык переставал работать. При перемещении ярлыка перемещался сам ярлык, а не программа, и при попытке открыть ярлык, например, в строке командного интерпретатора, он не срабатывал. Вывод содержимого ярлыка в консоль выводил в консоль не документ, на который ярлык ссылался (даже если это текстовый документ), а содержание самого ярлыка.

Символические ссылки

Символическая ссылка – особого рода файл, ссылающийся на местоположение другого файла. Операционная система представляет символическую ссылку приложению так, как если бы он открыл сам файл. В этом состоит отличие символической ссылки от ярлыка.

При попытке прочитать содержимое символической ссылки на экран будет выведено содержимое файла. В работе использование символической ссылки почти никак не отличается от указания настоящего имени файла. Символические ссылки уместно сравнить с ярлыками, обрабатываемыми на уровне ядра операционной системы.

В Linux символические ссылки создаются с помощью команды ln -s файл имя_ссылки.

Вывод оглавления директории с атрибутами ls -la также указывает и место назначения ссылок.

```
$ echo Hello >test
$ ln -s test test1
$ ls -la
$ cat test1
$ rm test
$ la -la
$ cat test1
```

При удалении файла, на который ведет ссылка, она становится «битой». По своей природе символическая ссылка – это особого рода файл с указанием местоположения, поэтому такие ссылки могут вести на директории и на файлы, размещённые в других (смонтированных) разделах. Однако при размонтировании отдела такая ссылка не будет работать.

В остальном работа с символической ссылкой не отличается от работы с самим файлом. Так, копирование ссылки приведёт к копированию файла: в отличие от копирования ярлыка, при копировании ссылки будет создан сам файл.

В Windows символические ссылки создаются с помощью mklink, который по умолчанию создаёт символическую ссылку на файл, с опцией /D – на директорию, с опцией /H – жесткую ссылку. Об опции /J мы уже говорили.

Символические ссылки в Debian

Символические ссылки активно используются в Debian и Ubuntu. Например, при работе с веб-сервером Apache2 в директории /etc/apache2/conf-available находятся доступные конфигурационные файлы, а в /etc/apache2/conf-enabled – символические ссылки на соответствующие файлы из conf-available с аналогичными именами. Утилиты a2enconf и a2disconf и аналогичные фактически создают ссылки (ln -s) и удаляют их (rm). Подобный подход применяется и в nginx при использовании в Debian или Ubuntu, с той разницей, что ссылки создавать и удалять приходится через ln и rm.

Жёсткие ссылки

Фактически жёсткая ссылка - это ещё одно имя файла. В linux жёсткая ссылка создается командой ln имяфайла имяссылки.

Пример:

```
$ echo Hello >test
$ ln test test1
$ ls -ila
$ cat test1
$ rm test
$ la -ila
$ cat test1
```

Даже при удалении исходного файла такая ссылка продолжает работать. Более того, выяснить, какой файл оригинальный, а какой – ссылка, невозможно без номера inode. Очевидно, что жёсткие ссылки невозможно создавать на другие разделы. Также невозможно создать жёсткую ссылку на каталог. В Windows жёсткие ссылки создаются командой mklink с параметром /H.

Файловые потоки (альтернативные данные файлов)

Файловые потоки были созданы в NTFS для совместимости с ФС HFS в MAC OS и использовались для хранения метаданных – таких, как иконки приложений. В противовес жёстким ссылкам файловый поток может содержать, кроме основного содержимого, альтернативное. Большинство программ не используют эту возможность, игнорируя её. При попытке копировать файл штатными средствами Windows альтернативные данные не могут быть скопированы.

Возможности файловых потоков могут использоваться вирусами.

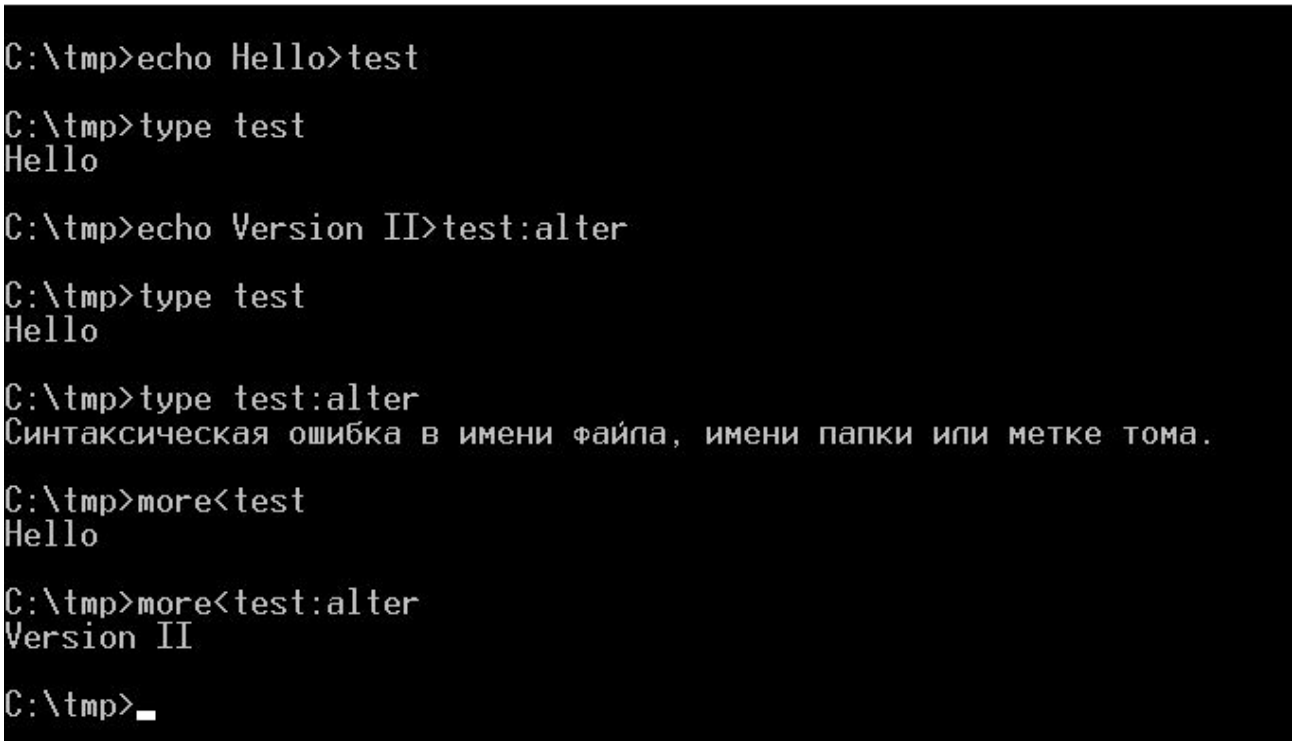
Примеры команд:

Создать файл и альтернативный файловый поток:

```
C:\TMP\> echo Hello >test
C:\TMP\> type test
C:\TMP\>echo VersionII > test:alter
```

Вывести на экран (type и некоторые другие привычные способы работы для потока не сработают):

```
C:\TMP\> type test
C:\TMP\> more <test
C:\TMP\> type test:alter
C:\TMP\> more <test:alter
```



Снимки (снапшоты)

Снапшоты используются для создания копий и восстановления системы. Их идея заключается в том, что систему без остановки работы с диском можно «заставить» не перезаписывать старую информацию, а только добавлять данные. Если «законсервировать» всю информацию таким образом, мы получим снимок (снэпшот), который можно смонтировать.

Возможность получения снимков есть в файловой системе ZFS и в системе управления томами LVM.

Управление томами

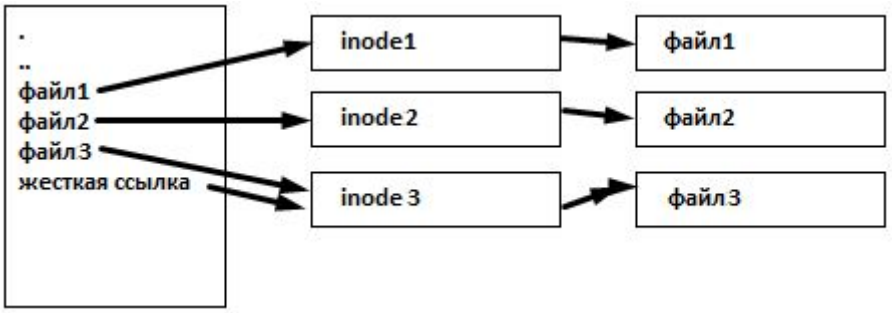
Управление томами подразумевает возможность объединить несколько физических устройств в группу томов, в которой можно выделить логические тома и использовать как диски/разделы. Такой подход позволяет безболезненно увеличивать и перераспределять пространство между устройствами, а также перемещать данные с одних дисков на другие без остановки работы.

Управление томами обеспечивается системой LVM и также встроено в файловую систему ZFS.

Работа с файловыми системами в Linux

Устройство файловой системы на примере ext2/ext4

В отличие от системы FAT, в Linux принята организация файловых системы с отдельным хранением атрибутов. Таким образом, каталог ссылается не сразу на файл, а на айнод (i-node), содержащий атрибуты (метаинформацию о файле), который уже, в свою очередь, ссылается на файлы. Этот подход определяет логику работы с ФС в Linux.



ID айндов можно посмотреть при помощи команды:

```
$ ls -ila
```

Традиционно в Linux для каждого файла присутствуют атрибуты r (чтение), w (запись), x (выполнение), которые устанавливаются для трех групп: владелец, группа и остальные. На самом деле атрибутов больше: есть дополнительные атрибуты (SUID, SGID, sticky bit), а также директории, символические ссылки, файловые потоки, которые тоже являются особого вида атрибутами.

Восьмеричная система счисления позволяет описать разный уровень прав просмотра и работы с файлами в виде одного из восьми значений (от 0 до 7).

	r	w	x		r	w	x		
–	–	–	–		0	0	0		0
x	–	–	x		0	0	1		1
w	–	w	–		0	1	0		2
wx	–	w	x		0	1	1		3
r	r	–	–		1	0	0		4
rx	r	–	x		1	0	1		5
rw	r	w	–		1	1	0		6
rwx	r	w	x		1	1	1		7

Таким образом, права 777 означают, что файл могут читать, изменять и исполнять все.

Права изменяются с помощью chmod:

```
$ chmod 777 myfile
```

Особым образом интерпретируется атрибут x для директорий. По умолчанию он установлен. Если атрибут x снят с директории, ее невозможно сделать активной (cd).

Кроме того, атрибут x дает доступ к inode. Если атрибут x снят, файл невозможно ни изменить, ни прочитать из-за невозможности получить доступ к атрибутам.

Существует атрибут, задаваемый по умолчанию. Для файлов это 644, для директорий – 755. Чтобы задать атрибут по умолчанию сразу, используется следующий принцип: берется некая маска, которая вычитается из 666 для файлов и 777 для директорий. По умолчанию это 022 – таким образом, 666 – 022 = 744, 777 – 022 = 755.

Такая операция задается командой umask:

```
$ umask 022
```

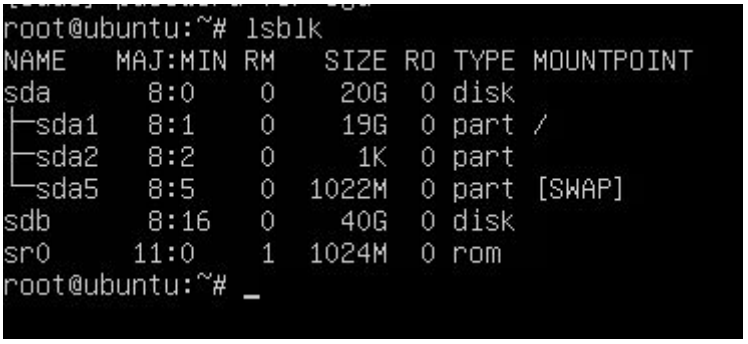
LVM

LVM – Logical Volume Manager.

Термины, используемые LVM:

- Физический том – PV (Physical Volume), диск или раздел на диске.
- Группа томов – VG (Volume Group): состоит из физических томов, но при этом может делиться на логические тома.
- Логический том – LV (Logical Volume): выделяется из группы томов, но не зависит от размера того или иного физического тома, то есть размер логического тома ограничен только суммарным размером группы томов и другими логическими томами. Логический том является блочным устройством, соответственно может содержать файловую систему.

Исходная система:



В системе присутствуют два диска: на одном установлена Ubuntu, второй пуст.

Запустив fdisk, узнаем перечень команд, посмотрим таблицу разделов.

```
root@ubuntu:~# fdisk /dev/sdb

Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help): p

Disk /dev/sdb: 42.9 GB, 42949672960 bytes
255 heads, 63 sectors/track, 5221 cylinders, total 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x9f83936e

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): _
```

Создадим новый первичный раздел размером 250 Мб и запишем его.

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): +250M
Value out of range.
Partition number (1-4, default 1): q
Partition number (1-4, default 1): ^C
root@ubuntu:~# fdisk /dev/sdb

Command (m for help): p

Disk /dev/sdb: 42.9 GB, 42949672960 bytes
255 heads, 63 sectors/track, 5221 cylinders, total 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x9f83936e

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-83886079, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-83886079, default 83886079): +250M

Command (m for help): _
```

```
Command (m for help): p

Disk /dev/sdb: 42.9 GB, 42949672960 bytes
255 heads, 63 sectors/track, 5221 cylinders, total 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x9f83936e

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048      514047      256000   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
root@ubuntu:~# _
```

Действия записываются так:

```
# fdisk /dev/sdb
```

```
n
p
1
2048
+250M
w
```

Аналогично создаем и второй раздел:

```
root@ubuntu:~# fdisk /dev/sdb

Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (1-4, default 2):
Using default value 2
First sector (514048-83886079, default 514048):
Using default value 514048
Last sector, +sectors or +size{K,M,G} (514048-83886079, default 83886079):
Using default value 83886079

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

```
root@ubuntu:~# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0    20G  0 disk
├─sda1   8:1    0    19G  0 part /
├─sda2   8:2    0     1K  0 part
└─sda5   8:5    0  1022M  0 part [SWAP]
sdb      8:16   0    40G  0 disk
├─sdb1   8:17   0   250M  0 part
└─sdb2   8:18   0  39.8G  0 part
sr0     11:0    1  1024M  0 rom
```

Установим lvm2;

```
# apt-get install lvm2
```

Инициализируем /dev/sdb2 в качестве физического тома:

```
# pvcreate /dev/sdb2
```

Создаём группу томов:

```
# vgcreate vg0 /dev/sdb2
```

```
root@ubuntu:~# vgcreate vg0 /dev/sdb2
  Volume group "vg0" successfully created
root@ubuntu:~# _
```

Посмотрим физические диски:

```
# pvdisplay
```

```
root@ubuntu:~# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sdb2
  VG Name               vg0
  PV Size               39.75 GiB / not usable 0
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              10177
  Free PE               10177
  Allocated PE          0
  PV UUID               lOUysd-DPIO-EW60-KNZr-fdrS-rzlA-3jVzma
```

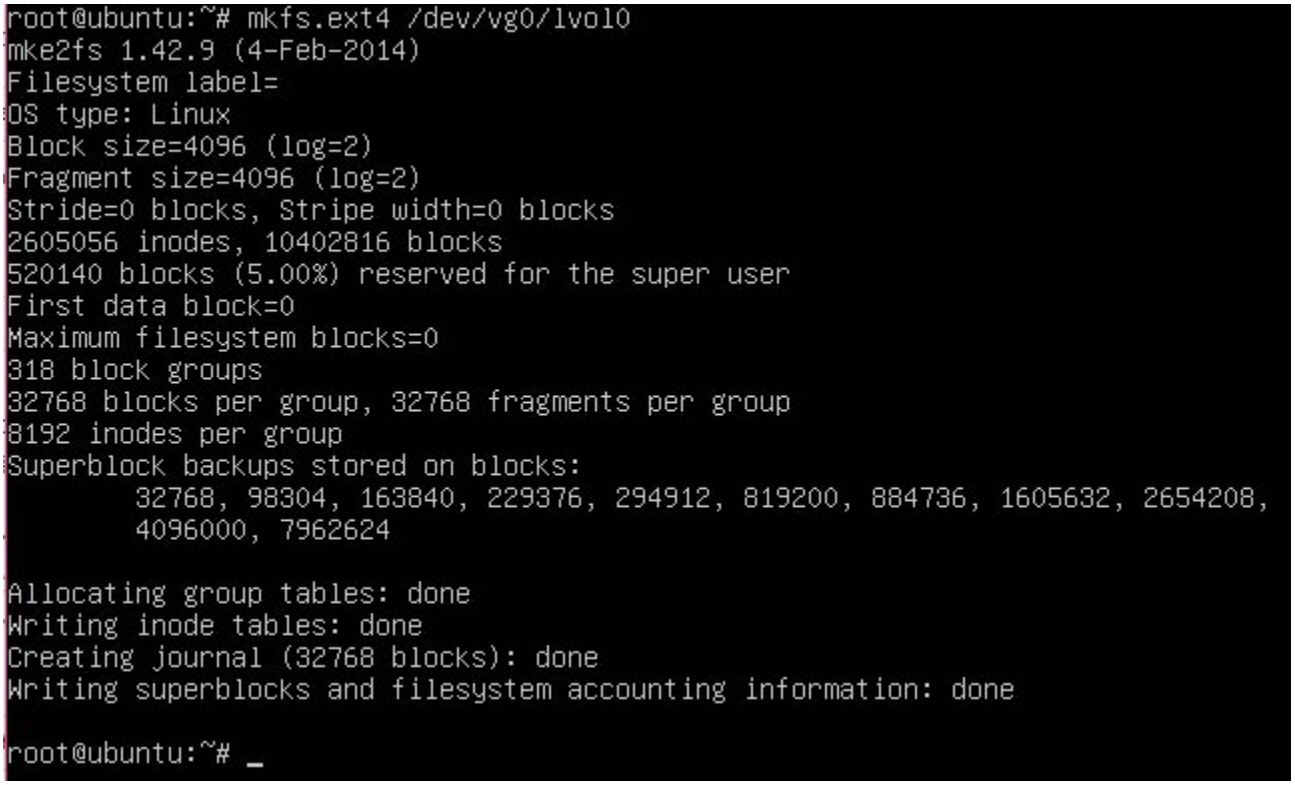
Создаём логический том. Если есть ошибка, её можно удалить и создать заново.

```
# lvcreate -L41036M vg0
# lvremove /dev/vg0/lvol0
#lvcreate -L46036M vg0
```

```
root@ubuntu:~# lvcreate -L40136M vg0
  Logical volume "lvol0" created
root@ubuntu:~# lsblk
NAME                  MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                     8:0    0    20G  0 disk
├─sda1                  8:1    0    19G  0 part /
├─sda2                  8:2    0     1K  0 part
└─sda5                  8:5    0  1022M  0 part [SWAP]
sdb                     8:16   0    40G  0 disk
├─sdb1                  8:17   0   250M  0 part
└─sdb2                  8:18   0  39.8G  0 part
  └─vg0-lvol0 (dm-0) 252:0    0  39.2G  0 lvm
sr0                    11:0    1  1024M  0 rom
root@ubuntu:~# lvremove /dev/vg0/lvol0
Do you really want to remove and DISCARD active logical volume lvol0? [y/n]: y
  Logical volume "lvol0" successfully removed
root@ubuntu:~# lvcreate -L40736M vg0
  Volume group "vg0" has insufficient free space (10177 extents): 10184 required.
root@ubuntu:~# lvcreate -L40636M vg0
  Logical volume "lvol0" created
root@ubuntu:~# lsblk
NAME                  MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                     8:0    0    20G  0 disk
├─sda1                  8:1    0    19G  0 part /
├─sda2                  8:2    0     1K  0 part
└─sda5                  8:5    0  1022M  0 part [SWAP]
sdb                     8:16   0    40G  0 disk
├─sdb1                  8:17   0   250M  0 part
└─sdb2                  8:18   0  39.8G  0 part
  └─vg0-lvol0 (dm-0) 252:0    0  39.7G  0 lvm
sr0                    11:0    1  1024M  0 rom
root@ubuntu:~# _
```

Создадим журналируемую файловую систему ext4 на логическом томе:

```
# mkfs.ext4 /dev/vg0/lvol0
```



Монтируем:

```
# mkdir /mnt/new
# mount /dev/vg0/lvol0 /mnt/new
# rsync -avPHAX --exclude /proc --exclude /sys --exclude /boot / /mnt/new
```

Копироваться будет долго… очень долго.

Создадим разделы для монтирования:

```
# mkdir /mnt/new/proc
# mkdir /mnt/new/sys
# mkdir /mnt/new/boot
```

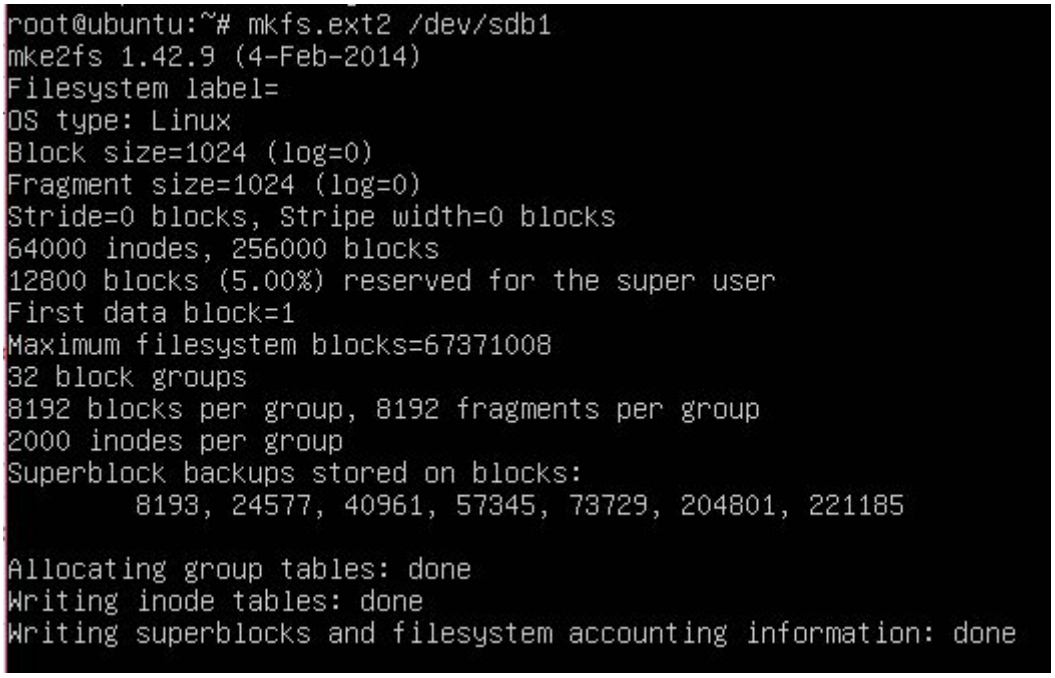
Или, что то же самое:

```
#mkdir /mnt/new/{proc,sys,boot}
```

Создадим файловую систему ext2 на разделе /dev/sda1. Там будет boot. GRUB2 не умеет работать с LVM, потому загружаться будем с него.

```
# mkfs.ext2 /dev/sdb1
```

Создание файловой системы на /dev/sdb1:



Смонтируем:

```
# mkdir /mnt/newboot
# mount /dev/sdb1 /mnt/newboot
```

Скопируем:

```
# rsync -avPHAX /boot /mnt/newboot
```

Монтируем старый корень:

```
# mkdir /mnt/old
# mount --bind / /mnt/old
```

Переходим в консоль tty1 (Alt-F1), логинимся рутом (или sudo -i) и переходим в однопользовательский режим:

```
# init 1
```

Снова копируем:

```
# rsync -avPHAX --exclude /proc --exclude /sys --exclude /boot /mnt/old /mnt/new
```

С помощью blkid смотрим UUID нашего /dev/sda5, с которого грузится система.

```
root@ubuntu:/mnt/new/mnt/new# blkid
/dev/sda1: UUID="6fbf9805-96c9-4ccd-a606-d1f6a000c40f" TYPE="ext4"
/dev/sda5: UUID="43831ac4-9688-4742-b08d-4bdf5bdc5bdc" TYPE="swap"
/dev/sdb1: UUID="43831ac4-9688-4742-b08d-4bdf5bdc5bdc" TYPE="ext2"
/dev/sdb2: UUID="10Uysd-DPIO-EW60-KNZr-fdrS-rz1A-3jVzma" TYPE="LVM2_member"
/dev/mapper/vg0-lvol0: UUID="9bb13e94-a272-47bf-85da-c69b17624df8" TYPE="ext4"
root@ubuntu:/mnt/new/mnt/new# blkid|grep /dev/sda1|grep -P "(?<=UUID=.)[0-9a-f-]*"
/dev/sda1: UUID="6fbf9805-96c9-4ccd-a606-d1f6a000c40f" TYPE="ext4"
root@ubuntu:/mnt/new/mnt/new# blkid|grep /dev/sda1|grep -oP "(?<=UUID=.)[0-9a-f-]*"
6fbf9805-96c9-4ccd-a606-d1f6a000c40f
```

Пишем регулярное выражение (grep с ключом -P позволит использовать PCRE, а -o оставляет только совпадение). В результате получаем:

```
# blkid /dev/sda5|grep /dev/sda1|grep -oP "(?<=UUID=.)[0-9a-f-]*"
```

Найденное значение пропишем для /dev/sdb1 с помощью tunefs.

```
root@ubuntu:/mnt/new/mnt/new# tune2fs -U $(blkid|grep /dev/sda1|grep -oP "(?<=UUID=.)[0-9a-f-]*") \
> /dev/sdb1
tune2fs 1.42.9 (4-Feb-2014)
root@ubuntu:/mnt/new/mnt/new# blkid|grep -P "(?<=UUID=.)[0-9a-f-]*"
/dev/sda1: UUID="6fbf9805-96c9-4ccd-a606-d1f6a000c40f" TYPE="ext4"
/dev/sda5: UUID="43831ac4-9688-4742-b08d-4bdf5bdc5bdc" TYPE="swap"
/dev/sdb1: UUID="6fbf9805-96c9-4ccd-a606-d1f6a000c40f" TYPE="ext2"
/dev/sdb2: UUID="10Uysd-DPIO-EW60-KNZr-fdrS-rz1A-3jVzma" TYPE="LVM2_member"
/dev/mapper/vg0-lvol0: UUID="9bb13e94-a272-47bf-85da-c69b17624df8" TYPE="ext4"
root@ubuntu:/mnt/new/mnt/new# _
```

Устанавливаем grub на /dev/sdb

```
# grub-install /dev/sdb
```

```
root@ubuntu:/mnt/new/mnt/new# grub-install /dev/sdb
Installing for i386-pc platform.
Installation finished. No error reported.
root@ubuntu:/mnt/new/mnt/new# _
```

Забыли про swap. Уменьшаем размер ФС и логического тома:

```
# umount /mnt/new
# fsadm -l resize /dev/vg0/lvol0 -2G
```

```
root@ubuntu:/# fsadm -l resize /dev/vg0/lvol0 -2G
fsck from util-linux 2.20.1
/dev/mapper/vg0-lvol0: 11/2605056 files (0.0% non-contiguous), 208526/10402816 blocks
resize2fs 1.42.9 (4-Feb-2014)
Resizing the filesystem on /dev/mapper/vg0-lvol0 to 9878528 (4k) blocks.
The filesystem on /dev/mapper/vg0-lvol0 is now 9878528 blocks long.

  Reducing logical volume lvol0 to 37.68 GiB
  Logical volume lvol0 successfully resized
root@ubuntu:/# _
```

Создаём ещё один логический том размером 2 Гб:

```
# lvcreate -L2G vg0
```

```
root@ubuntu:/# lvcreate -L2G vg0
  Logical volume "lvol1" created
root@ubuntu:/# lsblk
NAME                  MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                     8:0    0    20G  0 disk
|-sda1                  8:1    0    19G  0 part /
|-sda2                  8:2    0     1K  0 part
`-sda5                  8:5    0  1022M  0 part [SWAP]
sdb                     8:16   0    40G  0 disk
|-sdb1                  8:17   0   250M  0 part
`-sdb2                  8:18   0  39.8G  0 part
  |-vg0-lvol0 (dm-0) 252:0    0  37.7G  0 lvm
  `-vg0-lvol1 (dm-1) 252:1    0     2G  0 lvm
sr0                    11:0    1  1024M  0 rom
root@ubuntu:/#
```

Сделаем swap:

```
# mkswap /dev/vg0/lvol1
```

```
root@ubuntu:/# mkswap /dev/vg0/lvol1
mkswap: /dev/vg0/lvol1: warning: don't erase bootbits sectors
        on whole disk. Use -f to force.
Setting up swapspace version 1, size = 2097148 KiB
no label, UUID=b4b1a639-51cc-452a-a341-d19678351b6a
root@ubuntu:/# _
```

Монтируем виртуальные файловые системы к новой фс:

```
# mount --bind /dev /mnt/new/dev
# mount --bind /proc /mnt/new/proc
```

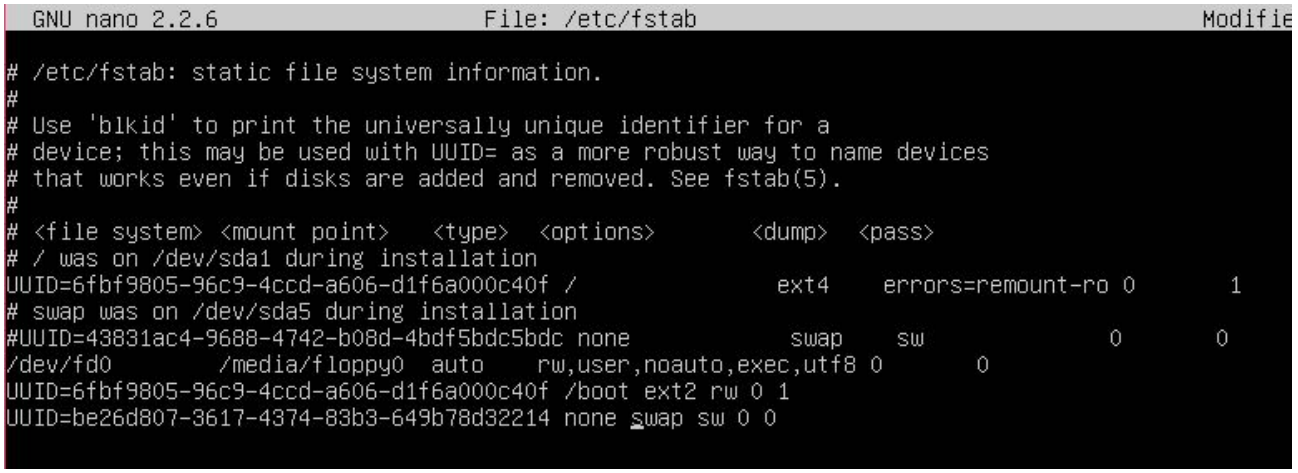
И раздел boot:

```
# umount /mnt/newboot
# mount /dev/sdb1  /mnt/new/boot
```

Изменяем корень:

```
# chroot  /mnt/new
```

Пропишите UUID для swap и boot в разделе fstab самостоятельно (blkid и nano /etc/fstab).



reboot

Настроим grub:

```
# grub-mkdevicemap
# update-grub2
```

С помощью fdisk удалите разделы диска /dev/sda

Перезагрузитесь.

Если произошла ошибка, загрузитесь вручную с помощью grub-restore.

```
ls
set root=(hd1,msdos1)
ls /
set prefix=(hd1,msdos1)/grub
insmod normal
normal
```

Вместо hd1 может быть hd0 (в зависимости от порядка загрузки в BIOS/UEFI). Также может понадобиться изменить порядок загрузки в BIOS, заново переконфигурировать grub и добавить /dev/sda1 в группу:

```
# vgextend vg0 /dev/sda
# lsblk
```

Увеличьте размер файловой системы самостоятельно.

Практическое задание

Для работы понадобится установленная в VMWare или VirtualBox система GNU/Linux.

1. Создать текстовый файл. Создать на него несколько символических и жёстких ссылок. Посмотреть, что происходит при удалении самого файла, жёстких или символических ссылок. Попробовать создать жёсткую ссылку на файл, находящийся на примонтированном разделе. Создать символическую ссылку на файл, находящийся на примонтированном разделе. Посмотреть, что произойдёт, если размонтировать раздел.

2. Разобрать, как сохранить файлы со всеми атрибутами на флеш-накопитель.

3. Подключить через VirtualBox или VMWare жёсткий диск. Сделать разметку на разделы и отформатировать. Примонтировать его в файловую систему Linux.

4. * Сделать то же самое с использованием LVM.

Примечание. Задания со звёздочкой предназначены для тех, кому недостаточно заданий 1-3 и требуются более сложные задачи.

Дополнительные материалы

1. http://www.rdm.kiev.ua/pages/lessons/urok4/
2. http://ab57.ru/cmdlist/mountvol.html
3. http://hex.pp.ua/using-alternate-data-streams.php
4. http://xgu.ru/wiki/ZFS
5. http://xgu.ru/wiki/LVM
6. http://xgu.ru/wiki/Сравнение_ZFS_и_Linux_LVM_+_RAID
7. http://freecoder.ru/content/podklyuchaem-novyi-zhestkii-disk-v-debian
8. http://help.ubuntu.ru/wiki/swap
9. http://it-kmm.com/articles/main_web.php?p=445
10. http://www.antmix.ru/2011/08/13/perenos-sistemy-na-shifrovannyi-lvm-bez-poteri-dannykh
11. http://freecoder.ru/content/perenos-lvm-razdela-na-novyi-zhestkii-disk
12. http://www.lexpr.ru/node/413
13. Пишем загрузочный сектор: http://www.compdoc.ru/prog/asm/boot_sector/

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. http://sd-company.su/article/computers/magnetic_tape
2. https://ru.wikipedia.org/wiki/Магнитный_барабан
3. i-RAM http://www.thg.ru/storage/20050910/onepage.html
4. http://olimp-service.ru/2015/07/30/external-hdd-inside/#.WK2jzzvyjIU
5. Физические основы восстановления информации на жестких дисках: http://www.vevivi.ru/best/Fizicheskie-osnovy-vosstanovleniya-informatsii-zhestkikh-magnitnykh-diskov-ref21380.html
6. https://ru.wikipedia.org/wiki/CHS
7. https://support.microsoft.com/ru-ru/help/140365/default-cluster-size-for-ntfs,-fat,-and-exfat
8. http://www.pvsm.ru/fajlovaya-sistema/14110
9. https://ru.wikipedia.org/wiki/FAT
10. https://ru.wikipedia.org/wiki/: Главная_загрузочная_запись
11. https://geektimes.ru/post/46935/