Работа с дисками

fdisk, gdisk, sysfs, LVM, mdadm, smartctl, sgdisk, partprobe

[Виртуальная файловая система, устройства](#_4q9m4zmg02qx)

[root](#_mx3luv6xqjlj)

[initrd](#_t3sumwqyxrla)

[procfs](#_3vosllrxnbe8)

[devfs](#_4wue0th8ejiq)

[sysfs](#_v2m2a8ofblu3)

[Управление дисками](#_evr0xh9fgm8w)

[Разделы, форматирование](https://docs.google.com/document/d/1RRCuoLBZvJM1r7ZZQG0_-lDsUmRG1PQfdn2n_TackW8/edit#heading=h.koznk6eyrf07)

[MBR](#_bq1grt4pfz60)

[GPT](#_ru6ube8zwkha)

[Файловые системы](#_jz5zvu2tebqk)

[Нежурналируемые файловые системы](#_fhjsjb3xtwc4)

[Журналируемые файловые системы](#_z02r9r2g3lvk)

[Некоторые известные ФС](#_2eghtnguiqrd)

[Подключение внешних устройств](#_ct93bbvkbu2v)

[Монтирование файловой системы и /etc/fstab](#_1zpf6yid68g9)

[LVM](#_k1rg4okp30vd)

[Некоторые примеры](#_t54xkmjd5xmr)

[Снимки (снэпшоты)](#_o91v98eiw9fi)

Практическое [задание](#_qvdlncp7jbiy)

[Дополнительные материалы](#_jeeoh8oz8lfe)

[Используемая литература](#_jqak36cpzk0l)

# 

# Виртуальная файловая система, устройства

Существенным различием Linux(да и других UNIX-подобных систем) от таких, как DOS и Windows, является тот факт, что работа ведется только с одной файловой системой – виртуальной. Фактически с одним деревом, в иерархии которого находится тот или иной файл, независимо от того, где он находится на самом деле (на каком разделе, на диске, по сети или создан из какой-то структуры ядра). Более того, Linux следует концепции «всё есть файл», потому устройства, драйвера, процессы – всё отображается в структуру файловой системы, благодаря чему достигается существенная гибкость в работе(скрипты могут получить доступ к любой необходимой информации, впрочем, как и пользователь).

Рассмотрим некоторые важные точки монтирования и монтируемые файловые системы.

## root

Корень – / – является точкой монтирования всей системы. Корень (/) в Linux не аналогичен, например, указанию вида C:\ в MS DOS и WINDOWS. То есть / не есть оглавление некоего системного раздела, это именно корень. Другой факт, что в / может быть смонтировано содержимое, например, /dev/sda1, но именно смонтировано. Первоначально при загрузке монтируется виртуальный образ initrd, после размонтируется и монтируется, скажем, /dev/sda1.

## initrd

Init RAM disk – образ файловой системы, который распаковывается в RAM-диск и монтируется в корень. Необходим для загрузки модулей ядра и выполнения простейших скриптов. В настоящее время образ initrd представляет собой архив cpio сжатый gzip.

Распаковать и изучить образ можно следующим способом.

|  |
| --- |
| mkdir /tmp/initrd && cd /tmp/initrd  cp /boot/initrd.img-4.4.0-109-generic /tmp/initrd  (cpio -id; zcat | cpio -id) < initrd.img-4.4.0-109-generic  ls  mc |

В некоторых случаях, в init ram disk находится микрокод для различных поддерживаемых видов процессоров (в данном случае AMD и Intel):

|  |
| --- |
| # binwalk initrd.img-4.4.0-131-generic  DECIMAL HEXADECIMAL DESCRIPTION  --------------------------------------------------------------------------------  0 0x0 ASCII cpio archive (SVR4 with no CRC), file name: ".", file name length: "0x00000002", file size: "0x00000000"  112 0x70 ASCII cpio archive (SVR4 with no CRC), file name: "kernel", file name length: "0x00000007", file size: "0x00000000"  232 0xE8 ASCII cpio archive (SVR4 with no CRC), file name: "kernel/x86", file name length: "0x0000000B", file size: "0x00000000"  356 0x164 ASCII cpio archive (SVR4 with no CRC), file name: "kernel/x86/microcode", file name length: "0x00000015", file size: "0x00000000"  488 0x1E8 ASCII cpio archive (SVR4 with no CRC), file name: "**kernel/x86/microcode/AuthenticAMD.bin**", file name length: "0x00000026", file size: "0x00006B2A"  28072 0x6DA8 ASCII cpio archive (SVR4 with no CRC), file name: "TRAILER!!!", file name length: "0x0000000B", file size: "0x00000000"  28672 0x7000 ASCII cpio archive (SVR4 with no CRC), file name: "kernel", file name length: "0x00000007", file size: "0x00000000"  28792 0x7078 ASCII cpio archive (SVR4 with no CRC), file name: "kernel/x86", file name length: "0x0000000B", file size: "0x00000000"  28916 0x70F4 ASCII cpio archive (SVR4 with no CRC), file name: "kernel/x86/microcode", file name length: "0x00000015", file size: "0x00000000"  29048 0x7178 ASCII cpio archive (SVR4 with no CRC), file name: "**kernel/x86/microcode/GenuineIntel.bin**", file name length: "0x0000002A", file size: "0x00170C00"  1539600 0x177E10 ASCII cpio archive (SVR4 with no CRC), file name: "TRAILER!!!", file name length: "0x0000000B", file size: "0x00000000"  **1540096 0x178000 gzip compressed data**, from Unix, last modified: 2019-02-21 10:22:56 |

В такой ситуации, приведенный выше скрипт ( (cpio -id; zcat | cpio -id) < initrd.img) работать не будет, так как, в начале, cpio разархивирует первый архив, с kernel/x86/microcode/AuthenticAMD.bin , а затем zcat попытается разархивировать следующий cpio с kernel/x86/microcode/GenuineIntel.bin в нем. Что приведет к ошибке.

Соответственно, изначальный скрипт, мы должны модифицировать с учетом этой особенности:

|  |
| --- |
| mkdir /tmp/initrd && cd /tmp/initrd  cp /boot/initrd.img-4.4.0-131-generic /tmp/initrd  (cpio -id; cpio -id; zcat | cpio -id) < initrd.img-4.4.0-131-generic  ls  mc |

## procfs

Виртуальная файловая система, отображающая информацию о процессах и системе. В директории /proc находится некоторая информация о системе, а в директориях с PID процесса – о соответствующих процессах.

Примеры.

|  |
| --- |
| # cat /proc/interrupts  # cat /proc/cpuinfo  # cat /proc/2/stat  # cat /proc/2267/environ |

Содержит также ряд полезных подразделов с информацией о прерываниях, файловых системах, системных параметрах.

Интерфейс позволяет не только получать информацию о процессах, но и изменять.

Например.

|  |
| --- |
| echo "1" > /proc/sys/net/ipv4/ip\_forward |

Разрешить IP- Forwarding, чтобы машина могла работать как маршрутизатор. Обратите внимание, что маршрутизация выполняется на уровне ядра, без привлечения пользовательского пространства. Сделано это в интересах быстродействия. Механизмы для маршрутизации, сетевого фильтра – реализованы внутри ядра. (Также стоит отметить, что после перезагрузки это состояние не сохранится. Чтобы опция была включена постоянно, следует отредактировать файл /etc/sysctl.conf.)

## dev

Очень важный элемент системы, реализующий концепцию “всё есть файл”. Ведь если мы хотим отправить сообщение в терминал 1, то пишем.

|  |
| --- |
| # echo Test >/dev/ttу1 |

Существующие в директории /dev файлы – особого рода имена устройств, с которыми можно работать. Более точно, это именованные каналы (FIFO), позволяющие взаимодействовать с драйверами устройств.

Ряд устройств создается автоматически операционной системой. К таким относятся терминалы, устройства дисков (hda – IDE, sda – SATA). Также стоит отметить, что такое именование устройств характерно именно для Linux. В других UNIX-подобных системах нумерация может отличаться. К примеру, во FreeBSD можно увидеть такие наименования: ad0, ad0s1a, ad0s1b. В загрузчике GRUB раздел /dev/sda1 будет выглядеть как (hd0, msdos1), а /dev/sdb5 как (hd1,msdos5), при этом msdos не значит, что используется FAT, это может быть и etx2 и ext4.

Устройства делятся на блочные и символьные. С символьными можно работать как с обычными текстовыми файлами. Блочные работают блоками, пример – устройства дисков.

У данных файлов нет размера. Но они обладают такими атрибутами, как старший и младший номер устройства. Старший определяет тип устройства, младший – порядковый номер среди однотипных устройств.

Посмотрите, какие старшие и младшие номера присутствуют у дисковых устройств (/dev/sda, /dev/sda1, /dev/sdb).

Но не всегда файл нужного устройства будет присутствовать. В этом случае помогает команда mknod.

Она позволяет с ключом b создавать блочные устройства, с ключом u – символьные и с ключом p – именованные каналы.

Примеры.

|  |
| --- |
| # mknod /dev/test1 b 200 1  # mknod /dev/test2 u 201 1  # mknod /tmp/test3 p |

Фактически это ещё один механизм межпроцессного взаимодействия.

Также обратите внимание, что после перезагрузки созданные файлы устройств в директории /dev не сохранятся. Потому они должны создаваться с помощью mknod скриптами при загрузке либо с помощью аналогичного системного вызова соответствующего модуля ядра.

## sysfs

Так же как и profs предоставляет для пользовательского пространства информацию, доступную ядру. В /sys содержатся структурированные данные в виде текстовых файлов о найденном ядром оборудовании. Здесь есть блочные устройства, шины(в том числе можно посмотреть информацию по PCI, закрепленным аппаратным прерываниям и т.д.), драйвера.

sysfs позволяет добавлять и удалять устройства (с точки зрения ядра операционной системы).

Удаление устройства.

|  |
| --- |
| # echo 1 >/sys/block/sdX/device/delete |

Где sdX — sda, sdb, sdc (для /dev/sda и т.д., соответственно).

Команда низкоуровневая и не проверяет, было ли устройство смонтировано или нет.

Добавление устройства.

|  |
| --- |
| # echo "- - -" >/sys/class/scsi\_host/hostX/scan |

Где X — номер шины, совпадает с номером SATA порта на материнской плате. Если Вы не знаете номер, можно по очереди перебирать номера (нумерация идет с 0) до достижения успеха. Фактически команда заставляет контроллер перечитать список подключенных устройств, например.

|  |
| --- |
| # echo "- - -" >/sys/class/scsi\_host/host0/scan  # echo "- - -" >/sys/class/scsi\_host/host1/scan  # echo "- - -" >/sys/class/scsi\_host/host2/scan |

Вышеописанные команды с sudo.

Удаление устройства.

|  |
| --- |
| $ echo 1 |sudo tee /sys/block/sdX/device/delete |

Добавление устройства.

|  |
| --- |
| $ echo "- - -" |sudo tee /sys/class/scsi\_host/hostX/scan |

# Таблицы разделов

Для установки нескольких операционных систем на один жесткий диск или для разделения хранимой информации (системная, файлы, архивы) применяется разметка диска на разделы. Каждый раздел выглядит для системы как отдельный диск, в DOS и Windows имеет отдельную букву диска (например, диски C: и D:).  
Для управления разделами используется таблица разделов (partition table). Это служебная область в начале жесткого диска, которая описывает дисковую разметку. Каждый раздел может содержать свою файловую систему и использоваться для загрузки размещенной в ней операционной системы (если есть такая потребность).

## MBR

MBR – Main Boot Record (главная загрузочная запись). Содержит исполняемый код, необходимый для передачи управления загрузчик и таблицу разделов (partition table). Может содержать только 4 первичных (primary) раздела, а при необходимости большего количества разделов 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). Есть определённые ограничения, связанные с созданием разделов и установленных на нем ОС. Например, Windows необходимо обязательно устанавливать в первичный раздел, помеченный как активный (загрузочный). Для разделов Linux таких ограничений сейчас нет.

Кроме того, MBR позволяет использовать диски размером не более 2Tb. В случае необходимости использовать диски более 2Tb требуется использовать GPT.

## GPT

GUID Partition Table. Больше 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 понятия как расширенные и первичные разделы не используются.  
fdisk не будет работать с gpt, вместо него следует использовать gdisk.

gdisk — интерактивная консольная утилита, похожая на fdisk.

sgdisk — то же самое, что и gdisk, но в командном, не интерактивном режиме.

Пример: скопировать GPT с /dev/sdd в /dev/sdc.

Обратите внимание, что в начале идет — куда мы копируем, а потом - откуда (что может показаться непривычным по сравнению с командами cp и тому подобными).

|  |
| --- |
| sgdisk -R /dev/sdc /dev/sdd |

# Файловые системы

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

## Нежурналируемые файловые системы

Изначально более ранние, чем журналируемые.

Примеры таких файловых систем FAT и EXT2.

Если в процессе записи действие будет остановлено, возникнут ошибки файловой системы. Повреждённые файлы, 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 переходили от использования FAT к NTFS, обращали внимание, что больше проблемы с проверкой не возникало. Независимо от корректности завершения системы.

Обусловлено это тем, что NTFS относится к журналируемым файловым системам. Они более устойчивы к непредвиденным событиям (как неожиданное выключение питания компьютера), но с другой стороны, требуют больше ресурсов. Потому, например, EXT2 считается эталоном производительности файловой системы. Но 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.

Также нежурналируемые файловые системы используются на флеш-накопителях (exFAT, ext2) из-за меньшего использования 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 - набор действий, выполняемых как одно целое. То есть оно либо полностью завершено, либо нет. Если транзакция будет прервана, то в соответствии с записью в журнале эти изменения будут откачены и транзакция будет считаться невыполненной. Такой подход привносит определенные удобства и надежность.

При этом журналируемые файловые системы будут работать медленнее, чем если бы были отформатированы в аналогичные нежурналируемые. Затраты на чтение/запись делают журналируемые файловые системы мало пригодными для использования во флэш-накопителях.

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

## Некоторые известные ФС

FAT (File Allocation Table – «таблица размещения файлов») – известная и простая неужрналируемая файловая система, созданная для MS DOS Биллом Гейтсом и Марком МакДональдом (англ.) в 1976-1977 годах. Используется и сейчас для флеш-накопителей.

VFAT – расширение, совместимое с FAT, 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. Теперь имя файла могло быть до 255 символов длиной, и 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 – расширенная версия FAT, оптимизированная под флеш-накопители. Флеш-накопители 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.).

# Подключение внешних устройств

Создадим виртуальный диск объемом 8 Gb. Для этого потребуется остановить нашу виртуальную машину и зайти в ее раздел настроек. В VirtualBox выбираем секцию «Носители», контроллер SATA, добавить новый диск. Все параметры можно составить по умолчанию: тип VDI, динамический. В VMWare выбираем Add и тип Hard disk. Параметры также можно оставить по умолчанию.

Далее надо указать имя и размер диска и кликнуть «Создать»/«Finish».

Запускаем нашу виртуальную машину с новым диском и стартуем терминальную сессию через приложение «Терминал» или с использованием виртуального терминала (Ctrl-Alt-F1).

Дальнейшие действия потребуют полномочий суперпользователя, поэтому все команды будут предваряться префиксом sudo. Первое, что надо сделать, - выяснить имя нового устройства. Команда fdisk предназначена для создания логических разделов на диске, но с параметром -l показывает разметку существующих дисков. Эта утилита не самая удобная из существующих в linux, но она есть почти в любой Linux-системе, поэтому умение с ней работать будет полезным.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| user@vlamp:~$ sudo fdisk -l  [sudo] password for user:  Диск /dev/sda: 16.1 Гб, 16106127360 байт  255 головок, 63 секторов/треков, 1958 цилиндров, всего 31457280 секторов Units = секторы of 1 \* 512 = 512 bytes  Размер сектора (логического/физического): 512 байт / 512 байт I/O size (minimum/optimal): 512 bytes / 512 bytes Идентификатор диска: 0x0000d0e2   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Устр-во Загр |  | Начало | Конец | Блоки | Id | Система | | /dev/sda1 | \* | 2048 | 27341891 | 13669922 | 83 | Linux | | /dev/sda2 |  | 27344894 | 31455231 | 2055169 | 5 | Расширенный | | /dev/sda5 |  | 27344896 | 31455231 | 2055168 | 82 | Linux своп / Solaris |   Диск /dev/sdb: 8589 МБ, 8589934592 байт  255 головок, 63 секторов/треков, 1044 цилиндров, всего 16777216 секторов Units = секторы of 1 \* 512 = 512 bytes  Размер сектора (логического/физического): 512 байт / 512 байт I/O size (minimum/optimal): 512 bytes / 512 bytes Идентификатор диска: 0x00000000  На диске /dev/sdb отсутствует верная таблица разделов |

Из вывода программы понятно, что наш диск объемом 8 Gb называется /dev/sdb и логическая разметка на нем отсутствует. Создадим на диске один первичный раздел на весь диск.

|  |
| --- |
| user@vlamp:~$ sudo fdisk /dev/sdb  Устройство не содержит ни верной таблицы разделов DOS, ни метки диска Sun, SGI или OSF Building a new DOS disklabel with disk identifier 0x9ef3eaca.  Changes will remain in memory only, until you decide to write them. After that, of course, the previous content won't be recoverable.  Предупреждение: неверный флаг 0x0000 таблицы разделов 4 будет исправлен записью |

Сначала покажем список команд.

|  |
| --- |
| Команда (m для справки): m  Действие команды  a переключение флага загрузки  b редактирование метки диска bsd  c переключение флага dos-совместимости  d удаление раздела  l список известных типов файловых систем  m вывод этого меню  n добавление нового раздела  o создание новой пустой таблицы разделов DOS  p вывод таблицы разделов  q выход без сохранения изменений  s создание новой чистой метки диска Sun  t изменение id системы раздела  u изменение единиц измерения экрана/содержимого  v проверка таблицы разделов  w запись таблицы разделов на диск и выход  x дополнительная функциональность (только для экспертов) |

Убедимся, что разделы на диске отсутствуют.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Команда (m для справки): p   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Диск /dev/sdb: 8589 МБ, 8589934592 байт | | | |  |  | | 255 головок, 63 секторов/треков, | | | 1044 цилиндров, всего 16777216 секторов | | | | Units = секторы of 1 \* 512 = 512 | | | bytes |  |  | | Размер сектора (логического/физического): | | | | 512 байт / 512 байт | | | I/O size (minimum/optimal): 512 | | | bytes / 512 bytes | |  | | Идентификатор диска: 0x9ef3eaca | | |  |  |  | | Устр-во Загр | Начало | Конец | | Блоки | Id Система | |  |  |  |  |  |  | |

Создадим первичный раздел на весь диск (значения секторов по умолчанию).

|  |
| --- |
| Команда (m для справки): n Partition type:  p primary (0 primary, 0 extended, 4 free) e расширенный  Select (default p): Using default response p  Номер раздела (1-4, по умолчанию 1): Используется значение по умолчанию 1 Первый сектор (2048-16777215, по умолчанию 2048): Используется значение по умолчанию 2048  Last сектор, +секторы or +size{K,M,G} (2048-16777215, по умолчанию 16777215): Используется значение по умолчанию 16777215 |

Проверим, что раздел был создан. Обратите внимание на имя раздела /dev/sdb1, оно нам еще понадобится.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Команда (m для справки): p  Диск /dev/sdb: 8589 МБ, 8589934592 байт   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 255 головок, 63 секторов/треков, 1044 цилиндров, всего 16777216 секторов  Units = секторы of 1 \* 512 = 512 bytes | | |  |  |  | | Размер сектора (логического/физического): | | | 512 байт / 512 байт | | | | I/O size (minimum/optimal): 512 bytes / 512 bytes | | | |  |  | | Идентификатор диска: 0x9ef3eaca | | |  |  |  | | Устр-во Загр | Начало | Конец | Блоки | Id | Система | | /dev/sdb1 | 2048 | 16777215 | 8387584 | 83 | Linux | |

Сохраним таблицу разделов диска.

|  |
| --- |
| Команда (m для справки): w  Таблица разделов была изменена!  Вызывается ioctl() для перечитывания таблицы разделов. Синхронизируются диски. |

Теперь создадим файловую систему с типом ext4 на разделе /dev/sdb1.

Процесс создания ФС (файловой системы) в UNIX заключается в записи на секторах диска определенных для данной файловой системы структур данных: это суперблок и его копии, айноды (i-nodes), блоки данных, журнал файловой системы.

|  |
| --- |
| user@vlamp:~$ sudo mkfs -t ext4 /dev/sdb1 mke2fs 1.42.9 (4-Feb-2014)  Filesystem label= OS type: Linux  Block size=4096 (log=2) Fragment size=4096 (log=2)  Stride=0 blocks, Stripe width=0 blocks 524288 inodes, 2096896 blocks |

Обратите внимание на строчку.

|  |
| --- |
| 104844 blocks (5.00%) reserved for the super user |

Она означает, что 5% диска зарезервировано.

|  |
| --- |
| First data block=0  Maximum filesystem blocks=2147483648 64 block groups  32768 blocks per group, 32768 fragments per group  8192 inod'ов в группе  Superblock backups stored on blocks:  32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632  Allocating group tables: done  Сохранение таблицы inod'ов: done Creating journal (32768 blocks): done  Writing superblocks and filesystem accounting information: done |

Как было замечено выше, 5% диска зарезервировано под запись для процессов с правами суперпользователя. По идее, это делается по умолчанию с целью предотвратить фрагментацию, но это не всегда нужно. Чтобы вернуть себе 5%, используем команду tune2fs.

|  |
| --- |
| root@vlamp:~# tune2fs -m 0 /dev/sdb1 tune2fs 1.42.9 (4-Feb-2014)  Setting reserved blocks percentage to 0% (0 blocks) |

## Монтирование файловой системы и /etc/fstab

Затем нам требуется смонтировать ФС. Монтирование - это процесс подключения иерархической ФС на носителе (или удаленной сетевой ФС) в один из каталогов общей ФС. Мы создадим каталог /arch и смонтируем в него ФС командой mount.

|  |
| --- |
| root@vlamp:~# sudo mkdir /arch/  root@vlamp:~# sudo mount /dev/sdb1 /arch/ |

Посмотрим, что в каталоге /arch.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| root@vlamp:~# ls -al /arch   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | итого 24 |  |  |  |  |  |  |  | | drwxr-xr-x | 3 | root | root | 4096 | июня | 11 | 07:09 . | | drwxr-xr-x 24 | | root | root | 4096 | июня | 11 | 07:10 .. | | drwx------ | 2 | root | root 16384 | | июня | 11 | 07:09 lost+found | |

Обратите внимание на каталог /arch/lost+found. Это стандартный каталог, который создаётся mkfs при создании ФС. Он необходим, если вдруг в ФС что-то сломалось, например, при аварийном выключении. При старте система поймет, что нормально размонтировать файловую систему не удалось и будет запущен процесс проверки ФС. Возможно после этого в lost+found появятся файлы с числовыми номерами, в которых могут быть ваши данные. Так что не удаляйте lost+found.

Еще один интересный момент заключается в том, что если в каталоге монтирования уже были какие-то файлы, они заместятся файлами смонтированной ФС. Но физически они никуда не деваются, их вновь можно будет обнаружить при размонтировании ФС. Возможен и обратный эффект. Например, вы пытались смонтировать флэш-диск, но так и не смонтировали и скопировали файлы не на флэш, а в саму директорию монтирования. Будьте внимательны!

После перезагрузки нам придётся опять подключать ФС на новом диске командой mount. Для автоматического монтирования ФС при запуске системы служит файл /etc/fstab. Чтобы /arch подключался автоматически, в него надо добавить строчку.

|  |
| --- |
| /dev/sdb1 /arch ext4 defaults 0 2 |

Где поля имеют следующие значения:

/dev/sdb1 — имя раздела, на котором находится ФС. Для других файловых систем здесь обычно указывают значения вроде.

|  |
| --- |
| UUID=261490e7-230f-4a74-bc20-a34c30f831d4 или  LABEL=arch |

Это более современный вариант. Имя раздела как в нашем примере также можно использовать, но при определённых обстоятельствах имя устройства может поменяться, а метка (LABEL) или UUID останутся прежними.

Метку LABEL можно задать утилитой e2label, а UUID посмотреть командой.

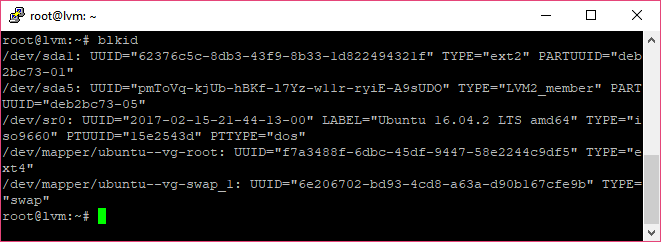
|  |
| --- |
| user@vlamp:~$ sudo tune2fs -l /dev/sdb1 | grep UUID  Filesystem UUID: 6637da8b-6c2c-45e8-a9b0-8189e9f2af15 |

* /arch — точка монтирования.
* ext4 — тип ФС.
* defaults — опции монтирования (команды mount).
* 0 — опция резервного копирования, здесь не имеет значения.
* 2 — порядок проверки ФС при старте системы. Обычно для корневой ФС значение 1, для остальных 2.

Более того, с помощью tune2fs можно и изменить UUID у раздела.

Кроме того, UUID можно узнать с помощью команды blkid.

|  |
| --- |
| # blkid |



## 

## LVM

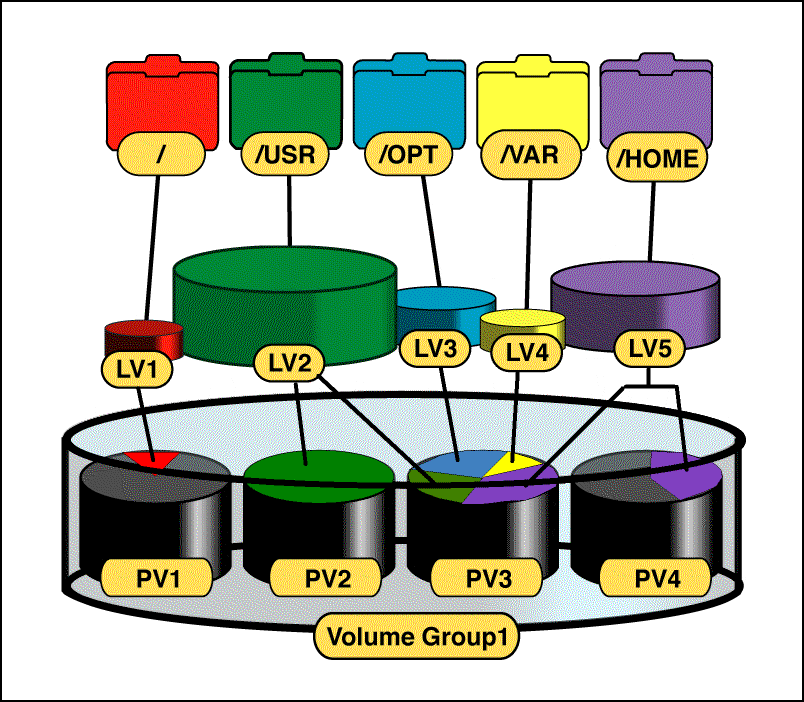
Технология LVM (Logical Volume Manager) добавляет большую гибкость в работе с дисками и разделами, добавляя несколько промежуточных слоев абстракции, позволяя объединять физические диски или разделы (физические тома) в группы физических томов, которые можно распределять на логические тома, динамически используя дисковое пространство всей группы физических томов как одно целое. В свою очередь, логические тома вы можете использовать так, как если бы это был физический диск или раздел.

Разберем подробнее.

Физический том (physical volume, pv) — физическая основа работы LVM. Как правило, это диск (/dev/sda, /dev/sdb) или раздел на диске (/dev/sda1, /dev/sda5). Последнее возможно в том случае, когда требуется использование файловых систем не в LVM (например, для раздела boot).

Группа томов (volume group, vg) — раздел, в который объединяются физические тома, воспринимается как одно пространство, которое в дальнейшем делится на логические тома. Это основа гибкости системы.

Логический том (logical volume) — раздел в группе томов аналогично тому, как на физическом диске выделяется раздел. Преимущество в том, что логический том может превышать емкость отдельно взятых физических томов, а также гибкость в работе с ними.



Структура LVM. Группа физических томов — Volume Group 1, включает Physical Volumes — физические тома, далее пространством группы томов используется для создания Logical Volumes — логических томов, которые уже форматируются в ext2 или ext4 и монтируются в соответствующие точки монтирования виртуальной файловой системы Linux.

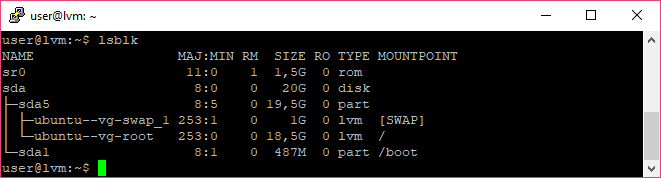
### Некоторые примеры

Для работы нам понадобится виртуальная машина, установленная с опцией использовать LVM, Либо, как вариант, можно использовать LVM только для вновь подключенных дисков, либо попытаться перенести систему с не-LVM раздела на LVM-раздел, хотя это может быть и трудоемко.

Предположим, что разделы уже установлены.

Выполним команду lsblk для того, чтобы посмотреть информацию о блочных устройствах.

|  |
| --- |
| lsblk |

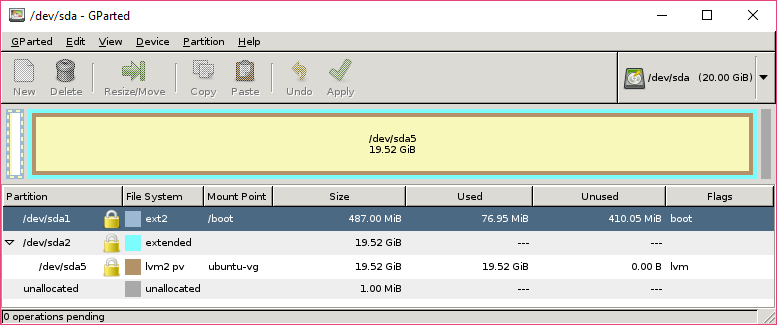


Здесь мы видим следующие блочные устройства.

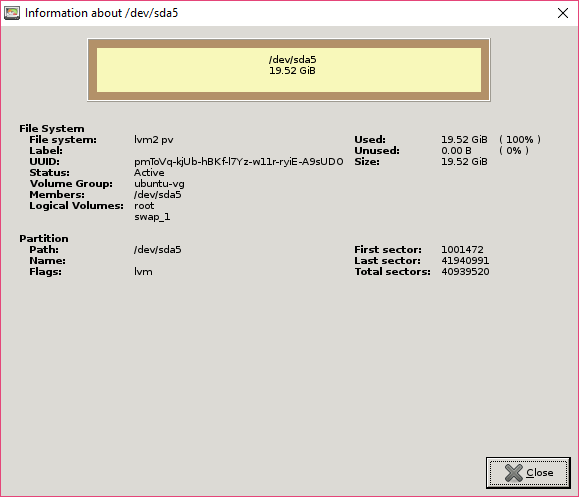
1. sr0 — CD-ROM.
2. sda — физический диск, на котором выделено два раздела.
3. sda5 — расширенный раздел, который добавлен в группу физических томов, в котором, в свою очередь, выделены логические тома.
   1. ubuntu--vg-swap\_1 — раздел для свопа.
   2. ubuntu-vg-root — корневой раздел Linux с файловой системой ext4.
4. sda1 — раздел с файловой системой ext2, предназначенный для загрузчика. Загрузчик GRUB не умеет работать с LVM, поэтому обычно оставляют один раздел, например, /dev/sda1, не включенным в группу физических томов. Этот раздел содержит загрузчик GRUB, образ ядра Linux и образ initrd, который распаковывается в RAM, монтируется в корень, загружаются драйвера, в т.ч. для поддержки lVM, после чего RAM-диск отмонтируется от корня, и вместо него монтируется ubuntu-vg-root. Сам же sda1 монтируется в директорию /boot.

Для сравнения, как мы видим через gparted.

|  |
| --- |
| sudo gparted |



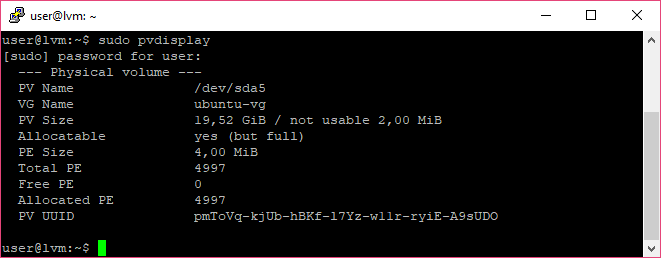
Кликнем по /dev/sda5.



Теперь поработаем с утилитами lvm.

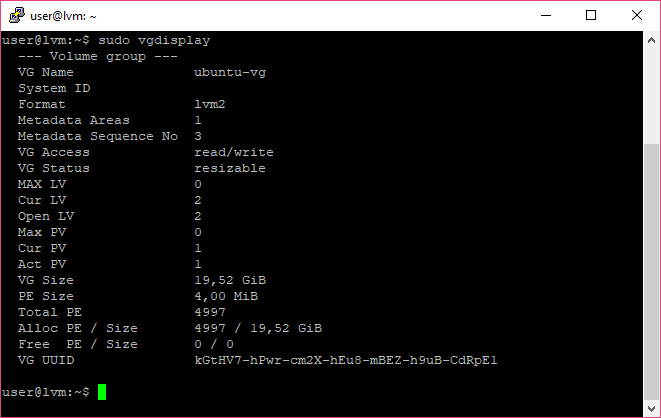
Посмотрим физические тома.

|  |
| --- |
| sudo pvdisplay |



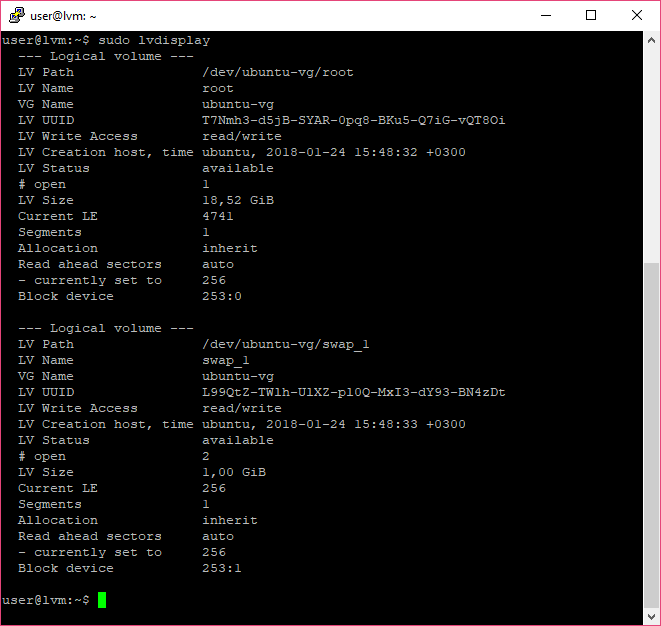
Посмотрим группу физических томов.

|  |
| --- |
| sudo vgdisplay |



Посмотрим логические тома.

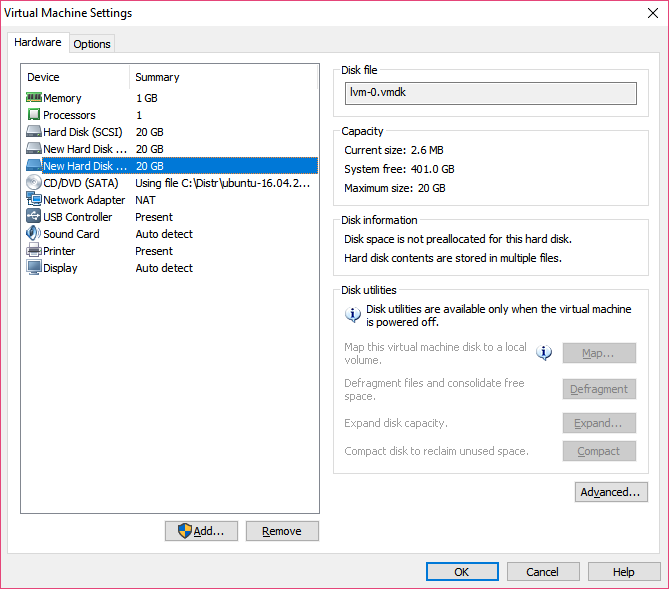
|  |
| --- |
| sudo lvdisplay |



Попрактикуемся. Сначала создадим два новых диска и создадим тома с нуля.

Кликаем Manage>Virtual Machine Settings>Hard Disk.

Добавляем два новых диска.



Жмем Ok.

Если мы посмотрим, какие диски доступны с помощью

|  |
| --- |
| fdisk -l |

или

|  |
| --- |
| #lshw -C disk |

Мы увидим, что новые диски в системе не видны.



Чтобы система обнаружила новые диски, вспомним про sysfs.

Далее создадим новые группы томов и логические тома. Данное упражнение можно сделать даже если еще LVM не установлен, для этого надо установить.

|  |
| --- |
| # apt-get install lvm2 |

Если же LVM была выбрана при установке, этого делать не надо.

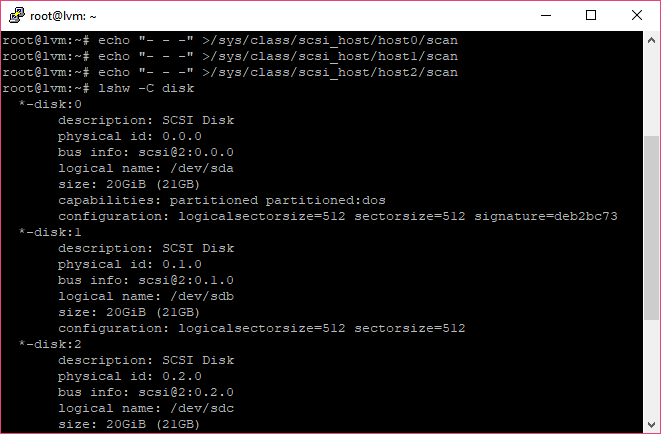
|  |
| --- |
| # echo "- - -" >/sys/class/scsi\_host/host0/scan  # echo "- - -" >/sys/class/scsi\_host/host1/scan  # echo "- - -" >/sys/class/scsi\_host/host2/scan |

После чего.

|  |
| --- |
| # lshw -C disk |

(Нужно подождать, команда работает не мгновенно.)

Прекрасно, все диски обнаружились.

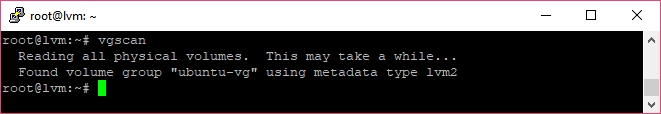


Начинаем работать с LVM.

Просканируем диски на наличие групп томов.

|  |
| --- |
| # vgscan |

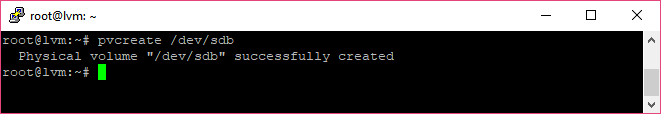
В нашем случае найден группа томов, созданная при установке.



Перед использованием дисков их надо активировать.

Сделаем это для диска /dev/sdb.

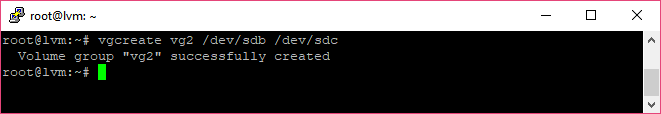
|  |
| --- |
| # pvcreate /dev/sdb |



Тоже самое для /dev/sdc проделайте самостоятельно.

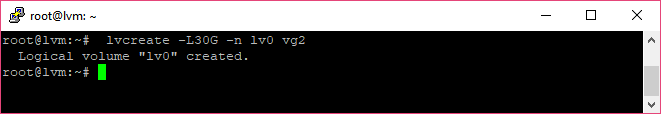
Теперь объединим /dev/sdb и /dev/sdc в группу томов. Назовем нашу группу vg2.

|  |
| --- |
| # vgcreate vg2 /dev/sdb /dev/sdc |



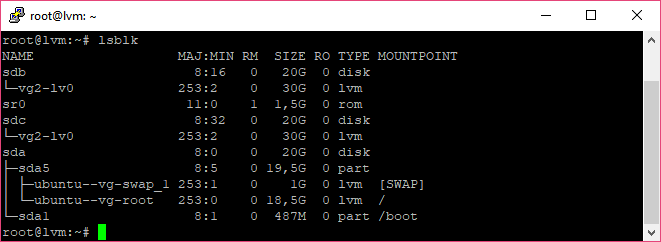
Теперь можно создать и логический том. Мы создали диски размером в 20Gb. Следовательно можно попробовать создать логический том размером в 30Gb.

|  |
| --- |
| # lvcreate -L30G -n lv0 vg2 |



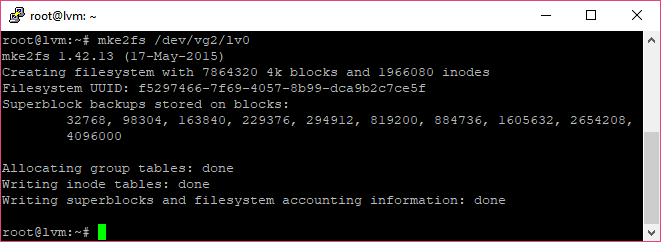
-L указывает размер и по умолчанию использует мегабайт (при этом в метрической системе, как и производителями дисков понимаемый в десятичной системе, то есть содержащий 1000 килобайте, а не 1024). Мы указали G для использования в метрических гигабайтах, T — в метрических терабатйах. Если нужно использовать привычные мегабайты-гигабайты (точнее мибибайты и гибибайты, содержащие по 1024 кибибайта и мибибайта, соответственно), необходимо использовать суффиксы в нижнем регистре (m,g и т.д.).

Посмотрим, что получилось.



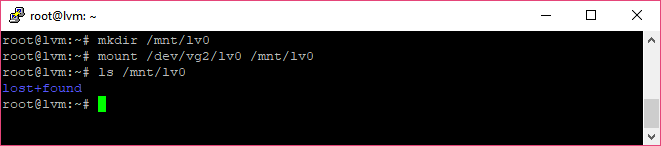
Но чтобы работать с новым разделом, его нужно отформатировать и смонтировать.

|  |
| --- |
| # mke2fs /dev/vg2/lv0 |



Смонтируем диск.

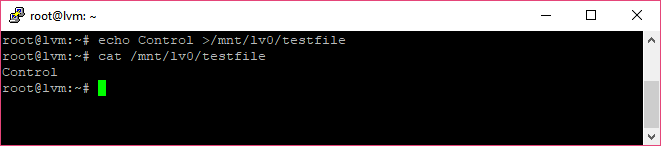
|  |
| --- |
| # mkdir /mnt/lv0  # mount /dev/vg2/lv0 /mnt/lv0  # ls /mnt/lv0 |



## Снимки (снэпшоты)

Создадим в смонтированном разделе директорию для неких данных и новый файл.

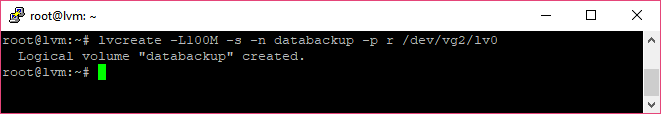
|  |
| --- |
| mkdir /mnt/lv0/data  echo control > /mnt/lv0/data/testfile |

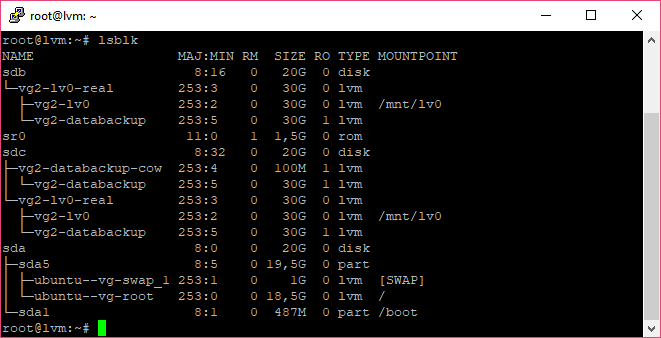


Создадим снимок логического тома.

|  |
| --- |
| lvcreate -L100M -s -n databackup -p r /dev/vg2/lv0 |

Для снимка должно быть достаточно место в группе физических томов, снимок похож на логический том, но при этом доступен только для чтения.





Удалим контрольный файл.

|  |
| --- |
| rm /mnt/lv0/testfile |

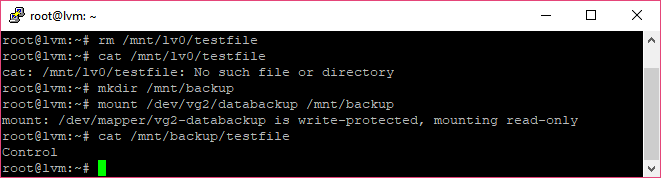
Создадим точку монтирования бэкапа.

|  |
| --- |
| mkdir /mnt/backup |

Смонтируем бэкап.

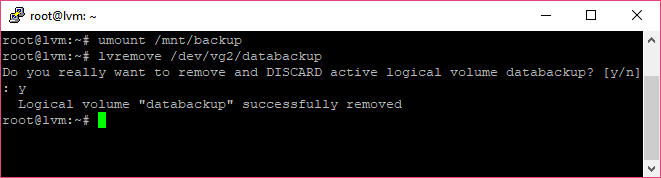
|  |
| --- |
| mount /dev/vg2/databackup /mnt/backup |

Посмотрим содержимое файла.



Если нам backup больше не нужен, его можно удалить (предварительно размонтировав).

|  |
| --- |
| umount /mnt/backup  lvremove /dev/vg2/databackup |

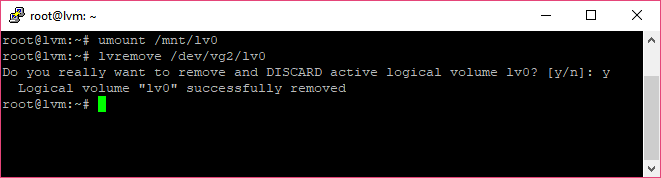


Удаление логических томов.

Удалять логические тома мы уже умеем.

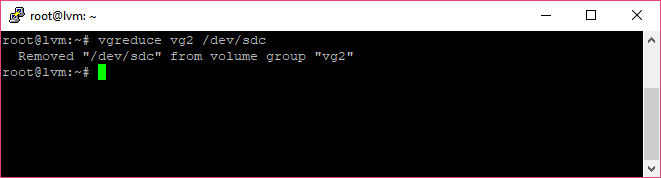
Удалим теперь все, что наследили (подразумевается, что данные нам не нужны, на практике будьте внимательны).

Удалить логический том (только сначала размонтируйте): lvremove /dev/vg2/lv0.

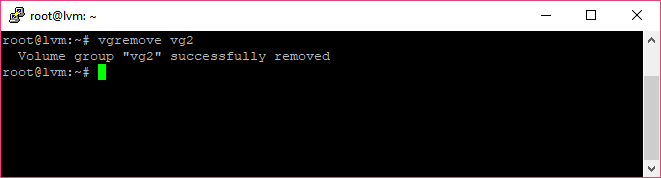


Удалить диски из группы томов (если есть логические тома, нужно перенести данные на оставшийся физический том, см. доп .литературу).

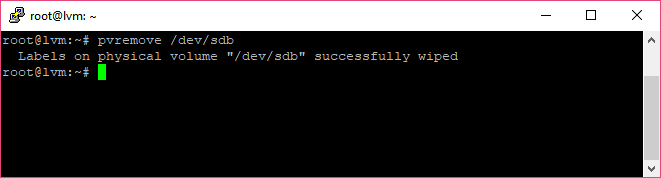
|  |
| --- |
| vgreduce vg2 /dev/sdc |



Удаление группы томов vgremove vg2.



Удалим с диска метку физического тома.



Тоже самое для sdc самостоятельно.

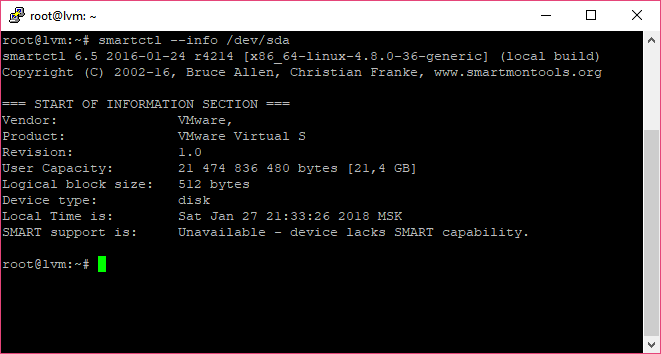
# smartctl

Жесткие диски имеют свойство со временем портиться. И лучше о возможных проблемах узнать заранее, чем однажды получить нерабочую систему. Благо, современные контроллеры дисков уже содержат встроенную систему мониторинга Self-Monitoring, Analysis and Reporting Technology systems (SMART) Эти данные можно получить с помощью smartctl.

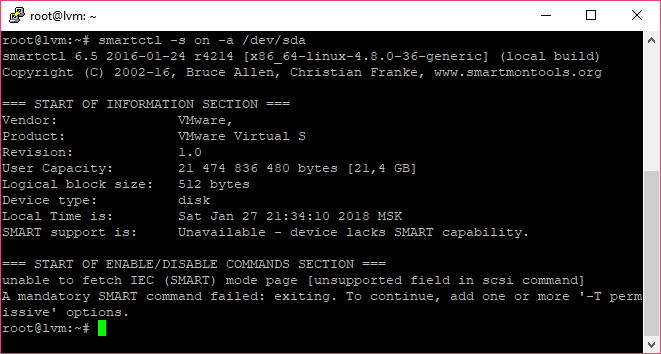
Установка.

|  |
| --- |
| apt install smartmontools |

Отображение информации о жестком диске smartctl --info /dev/sda.



Запустить проверку smartctl -s on -a /dev/sda.



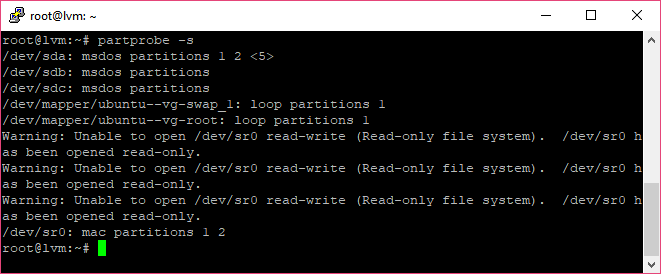
Как мы видим, для виртуального диска не удается воспользоваться этой утилитой, но для настоящих дисков использование этой утилиты обязательно.

# partprobe

В том случае, если мы изменили таблицу разделов, об этом следует уведомить операционную систему.

Для этого следует использовать partprobe.  
Посмотреть информацию о разделах.

|  |
| --- |
| partprobe -s |



Уведомить об изменении таблицы /dev/sdc.

|  |
| --- |
| partprobe /dev/sdc |

Если не указать раздел, информировать будет обо всех разделах.

# RAID

RAID (Redundant Array of Independent Disks — избыточный массив независимых дисков) — технология, которая главным образом известна использованием для повышения отказоустойчивости за счет резервирования. Но RAID это не просто “копирование с одного диска” на другой, как можно было подумать, это сложная технология. Более того, существует несколько разновидностей RAID, одна из которых не только не повышает надежность, а даже наоборот понижает, но за счет этого повышается скорость.

Существуют несколько уровней RAID.

* RAID-0 — тот самый случай, когда надежность не увеличивается, а уменьшается. Зато повышается быстродействие. Достигается это таким методом, что блоки поочередно пишутся на первый и второй диск. Соответственно каждый диск несет только фрагментарную часть информации. И если второй жесткий диск “полетит”, информация будет потеряна.
* RAID-1 — зеркальный дисковой массив. Интересно, что скорость чтения с такого массива выше, чем с использованием одинарного диска, а вот скорость записи несколько снижена (за счет избыточности). Это решение подходит как раз для надежности. Если один из дисков “полетит”, она задублирована на втором диске.
* RAID-10 — объединение достоинств RAID-1 и RAID-0. Сначала диски объединяются попарно для зеркалирования, а потом эти пары используются для последовательного распределения информации, как в RAID-0. Имеет и надежность, и скорость.
* RAID-2 — зеркальный массив с использованием самокорректирующихся кодов Хэмминга. Одна группа дисков используется для данных, вторая для коррекции. Повышается скорость работы, возможна коррекция ошибок на лету. Но для эффективности использования данного вида RAID по сравнению с другими в массиве требуется не менее 7 дисков.
* RAID-3 — зеркальный массив из n дисков, когда n-1 дисков несут данные, и 1 используется для коррекции (хранения блоков четности). Невозможна коррекция данных на лету, зато возможно восстановление.
* RAID-4 — похож на RAID-3, но работа ведется не блоками, а байтами. Соответственно один диск содержит байты четности. Также недостатком RAID-3 и RAID-4 является использование отдельного диска для хранения блоков или байт четности, и соответственно растет нагрузка на этот диск.
* RAID-5 — широко распространенный и экономичный вариант. Используется n дисков, при этом для каждых n-1 блоков, размещенных на n дисков, используется 1 блок коррекции на оставшемся диске, блоки данных и блоки коррекции размещаются равномерно по дискам. Это решает недостатки RAD-3 и 5, добавляет надежность и скорость работы.
* RAID-6 — похож на RAID 5, но имеет более высокую степень надёжности, так как используются три диска данных и два диска контроля чётности. Основан на кодах Рида — Соломона и обеспечивает работоспособность после одновременного выхода из строя любых двух дисков.

Следует отметить, что реализация RAID тоже может осуществляться разными способами. А именно: аппаратным и программным. Аппаратный — в случае использование контроллера, поддерживающего RAID. В случае выхода из строя контроллера восстановление данных может оказаться проблематичным, если не найдется такого же. При этом для большинства уровней RAID отдельный диск не является эквивалентом не-RAID-диска и не будет читаться, как обычный диск.

Программный вариант реализуется с помощью соответствующих драйверов и утилит. Операционная система “видит” обычные диски, а драйвер реализует “виртуальное дисковое устройство”, используя тот или иной алгоритм реализации RAID.

Один из вариантов реализации программного RAID-массива — mdadm.

## 

## mdadm

Название mdadm объясняется, если понять, что такое md. md — multiple devices. Каждое устройство в RAID-массиве md может содержать (а может и не содержать) метаданные, информация, которая позволяет заново собирать массив после остановки. При этом использование метаданных не нужно для RAID-0, а также в некоторых других случаях.

Для работы необходимо установить утилиту mdadm и включить модуль ядра md\_mod (если он еще не загружен).

|  |
| --- |
| # modprobe md\_mod  # apt-get install mdadm |

Итак, создать массив.

|  |
| --- |
| # mdadm --create --raid-devices=<n> --level=<level> <md-device> <block-device-1> ... <block-device-n> |

1. n — количество устройств массиве.
2. level — уровень RAID, поддерживаются.
   1. linear — это не RAID-массив, а объединение нескольких разделов или дисков в один “диск” большой емкости.
   2. raid0.
   3. raid1.
   4. raid4.
   5. raid5.
   6. raid6.
   7. raid10.
   8. multipath.
3. md-device — путь к блочному устройству RAID-массива.
4. block-device-1 — первый диск RAID-массива.
5. block-device-n — последний диск RAID-массива.

Пример создания.

|  |
| --- |
| # mdadm --verbose --create --raid-devices=2 --level=raid1 /dev/md0 /dev/sdb /dev/sdc |

Также имеется возможность указать запасные устройства, которые нужно использовать, если одно из указанных устройств выйдет из строя. Для этого используется опция --spare-devices.

Пример.

|  |
| --- |
| # mdadm --verbose --create --raid-devices=2 --level=raid1 /dev/md0 /dev/sdb /dev/sdc --spare-devices=1 /dev/sdd |

Чтобы узнать состояние массива и входящих в него дисков, достаточно посмотреть содержимое файла в procfs /proc/mdstat.

|  |
| --- |
| # cat /proc/mdstat |

Объединить диски в массив мало. Нужно полученный “диск” разбить и создать файловую систему, и смонтировать.

|  |
| --- |
| # fdisk /dev/md0  # mkfs.ext4 /dev/md0p1  # mount /dev/md0p1 /mnt/raid0 |

Все хорошо, но после перезагрузки или команд.

|  |
| --- |
| # umount /mnt/raid0  # mdadm --stop /dev/md0 |

Чтобы снова работать с диском, его нужно собрать.

Делается это так.

|  |
| --- |
| # mdadm --assemble /dev/md0 /dev/sdb /dev/sdc /dev/sdd |

Уровень RAID-массива мы не указываем, она берется из метаданных, только перечисляем диски, входящие в массив. Но можно еще проще.

|  |
| --- |
| # mdadm --assemble --scan |

# Практическое задание

1. Распаковать initrd. Какие скрипты, программы там содержатся? (выполнение задания необходимо для понимания в будущем процесса загрузки Linux).
2. Добавить в систему новый диск, смонтировать, настроить, чтобы автоматически монтировался при запуске (используя файл /etc/fstab).
3. (Подразумевается, что используется LVM). Добавить новый диск, добавить его в существующую группу томов, расширить корневой раздел ОС на новое пространство.
4. \* Добавить на диск произвольный файл, сделать снэпшот с помощью LVM, удалить или изменить файл, восстановить файл используя возможности LVM.
5. \*Создать софтовый RAID1 с помощью mdadm, проверить его работоспособность, имитировав, что 1 из дисков вышел из строя.

Примечание. *Задания 4 и 5 даны для тех, кому упражнений 1-3 показалось недостаточно.*

# Дополнительные материалы

1. LVM: http://xgu.ru/wiki/Lvm
2. Mdadm: http://xgu.ru/wiki/mdadm
3. smartctl: <https://losst.ru/diagnostika-hdd-v-linux>
4. RAID <http://www.timcompany.ru/article4.html>
5. RAID https://ru.wikipedia.org/wiki/RAID

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы.

1. <https://habrahabr.ru/post/102387/>
2. sysfs http://rus-linux.net/MyLDP/BOOKS/Moduli-yadra-Linux/05/kern-mod-05-10.html
3. http://mydebianblog.blogspot.ru/2013/02/sysfs-linux.html
4. http://src-code.net/fajlovaya-sistema-sysfs/
5. procfs https://mnorin.com/nemnogo-o-direktorii-proc-v-linux.html
6. http://linuxdoc.ru/mknod.html
7. http://rus-linux.net/kos.php?name=/papers/boot/boot-03initrd.html
8. http://www.k-max.name/linux/yadro-linux-poluchenie-informacii-i-upravlenie/
9. http://blog.getid.org/2013/10/enlarge-your-space.html
10. <http://shurshun.ru/linux-dobavlyaem-diski-na-letu/>
11. <http://www.lexpr.ru/node/336>
12. <http://xgu.ru/wiki/GPT>
13. <http://www.it-world.ru/tech/science/119997.html>
14. RAID <http://www.timcompany.ru/article4.html>
15. RAID <https://ru.wikipedia.org/wiki/RAID>
16. mdadm <http://ashep.org/2012/programmnyj-raid-v-linux-metadannye-mdadm-i-sozdanie-massiva/#.Wm4ZKKhl_IV>