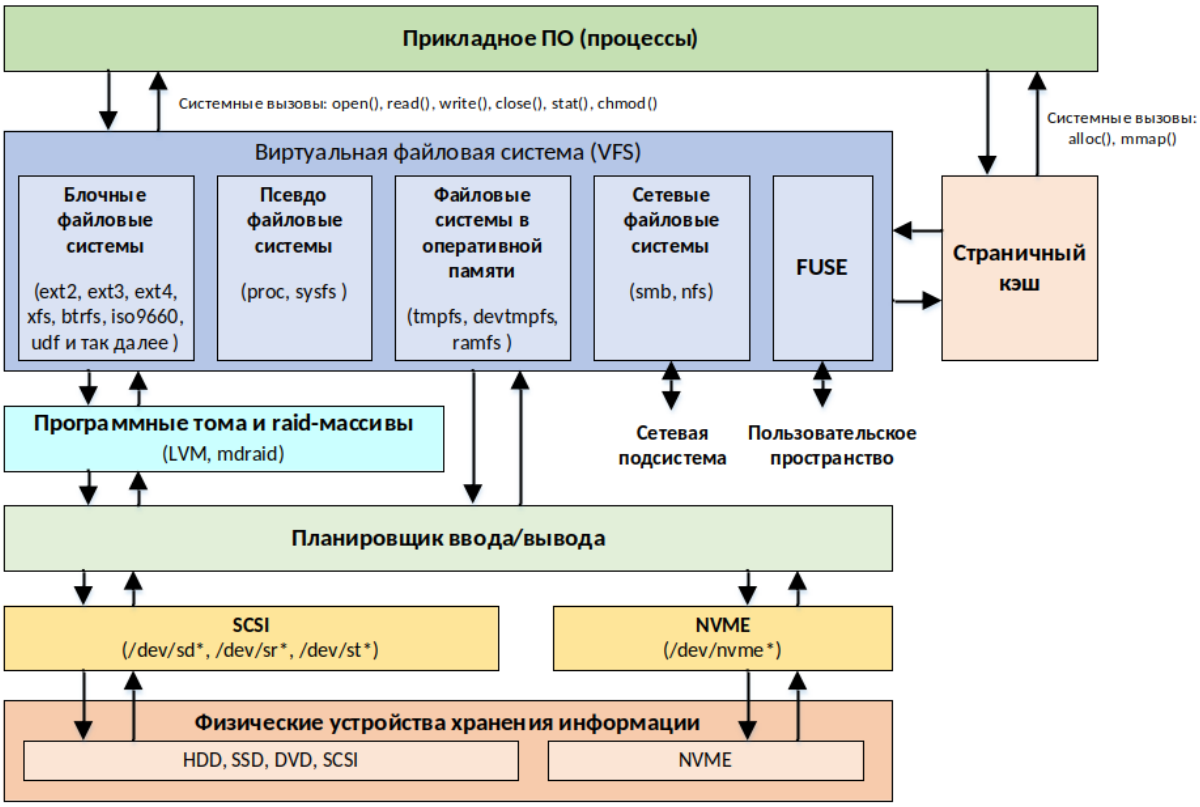
Лабораторная работа №13. Файловая система в Linux. Работа с блочными устройствами и LVM

Введение

В этой лабораторной мы значительно расширим наши представления о работе с блочными устройствами и файловыми системами, а также познакомимся с менеджером логических томов LVM, который значительно расширяет возможности по управлению системами хранения.

Подсистема хранения данных в Linux

Мы уже обсуждали различные типы файловых систем: блочные (ext4, btrfs), псевдофайловые (proc, sys), в оперативной памяти (tmpfs, ramfs), сетевые (smb, nfs). Кроме того, мы затрагивали также инструмент FUSE (*англ. Filesystem in USEr space*), с помощью которого можно монтировать диски без привилегий суперпользователя. Однако, подсистема хранения в Linux намного обширнее, и файловые системы – это только ее часть, (см. рисунок).



Архитектура подсистемы хранения данных

Прикладное ПО

На верхнем уровне расположены прикладные программы. Они позволяют как управлять подсистемой хранения (создавать, изменять и удалять разделы дисков, форматировать разделы и так далее) так и работать с данными, хранящимися в файловых системах (создавать файлы, изменять их, менять права, просматривать атрибуты и так далее).

Через системные вызовы прикладное ПО взаимодействует с виртуальной файловой системой (*англ. Virtual File System, VFS*) и со страничным кэшем (*англ. Page Cache*).

Виртуальная файловая система и страничный кэш

Виртуальная файловая система (еще называемая виртуальным коммутатором файловой системы) — это прослойка, уровень абстракции, между системными вызовами и конкретной реализацией файловой системы. Она предназначена для единообразия доступа прикладного ПО к различным типам файловых систем (блочной, в оперативной памяти, сетевой и так далее). VFS декларирует программный интерфейс, позволяющий без внесения изменений в ядро ОС обеспечивать поддержку новых файловых систем.

Страничный кэш – это набор страниц в оперативной памяти, в которые была записана информация с физического устройства хранения для обеспечения быстрого доступа к данным. При каждой операции страничного ввода-вывода, ядро ОС проверяет наличие страницы в кэше, и, по возможности, не использует затратную процедуру обращения к физическому устройству.

Программные тома и RAID-массивы

Виртуальная файловая система может взаимодействовать с планировщиком ввода/вывода непосредственно или через подсистему программных томов или RAID-массивов (таких как LVM или MDRAID), позволяющих гибко управлять разделами, расширять их, размещать их на нескольких физических устройствах и так далее, без необходимости прерывания работы (без простоя) и без необходимости переконфигурирования физических устройств хранения. Платой за это служит некоторое (небольшое) количество накладных расходов, за еще один уровень абстракции.

Планировщик ввода/вывода

Планировщик ввода/вывода – это компонент ядра, который выполняет оптимизацию запросов к диску для повышения производительности и пропускной способности, что

особенно критично при работе с жесткими дисками, у которых перемещение считывающей головки является крайне затратной операцией.

Алгоритмов планирования существует несколько, и с помощью команды `sudo dmesg | grep "io scheduler"` вы можете посмотреть, какие из них были зарегистрированы в системе. Команда `cat /sys/block/sda/queue/scheduler` покажет, какой из них используется для конкретного диска.

По умолчанию ALSE использует планировщик mq-deadline, который хорошо подходит для операций асинхронной записи на HDD. Если же у вас SSD, то вам лучше использовать алгоритм kyber, который повышает приоритет операций чтения над записью, или отключить планировщик вовсе.

Физические устройства хранения информации и их файлы

На нижнем уровне находятся физические устройства хранения информации, а в качестве интерфейсов доступа к ним выступают файлы каталога `/dev`. Файлы блочных устройств, к примеру, именуются следующим образом:

- Файл `/dev/sd<N>` — дисковые устройства хранения информации с интерфейсами, поддерживающими последовательную передачу данных (SATA, SCSI).
- Файл `/dev/hd<N>` — дисковые устройства хранения с интерфейсами, поддерживающими параллельную передачу данных (PATA).
- Файл `/dev/st<N>` — ленточные накопители SCSI.
- Файл `/dev/nvme<N>` — NVMe накопители.

Для дисковых устройств sd и hd имя задается по маске sd/hd<номер_устройства>[<раздел>], где в качестве номера устройства выступают строчные буквы латинского алфавита (a,b,c,…), а номера раздела может быть как целым положительным числом (порядковый номер раздела на диске), так и отсутствовать (весь диск), например sdb1 – это первый раздел на втором накопителе SCSI.

Диски SSD могут именоваться как sd, если для подключения используют интерфейс SATA, и как nvme, если для подключения используется интерфейс PCI Express. Формат имен nvme довольно сильно отличается от sd/hd устройств, т.к. это не просто быстрые диски, а целая технология, которая была разработана специально для твердотельных накопителей. Имена NVMe устройств задаются в следующем формате:

```
nvme<номер_контроллера>n<пространство_имён>p<номер_раздела>
```

Где:

• **nvme<номер_контроллера>** — номер контроллера NVMe, начиная с нуля. Контроллер является символьным устройством.

• **n<пространство_имён>** — пространство имён (*англ. namespace*), нумерация начинается с единицы. Пространства имен напоминают RAID-массивы, так как могут выходить за пределы одного устройства и создавать логические тома, распределенные по нескольким контроллерам сразу. Это блочное устройство.

• **p<номер_раздела>** — номер раздела в пределах пространства имен, нумерация начинается с единицы. Эти разделы являются полной аналогией с разделами жестких дисков. Это блочное устройство.

Например, nvme0n1p1 – это первый раздел в первом пространстве имен на первом nvme-контроллере компьютера.

Примечание

Как вы знаете, в системе Windows во избежание проблем с именами дисков для логических томов, находящихся на дополнительных устройствах, рекомендуется использовать буквы с конца алфавита (X:\, Y:\, Z:\). Это связано с тем, что автоматическая нумерация устройств зависит не только от того, к каким портам они подключены, но и от внешних факторов, например, как быстро они включаются.

В части автоматической нумерации дисков в Linux буквы присваиваются дискам автоматически, и вы не можете полагаться на них, т.к. устройства вполне могут устроить день перемены имен.

Опираясь на информацию предыдущей лабораторной, вы конечно сможете закрепить конкретные имена за конкретными устройствами с помощью udev-правил. Но если вспомнить, что в Linux вместо букв логических дисков используется единое пространство имен файловой системы, то станет очевидно, что будет вполне достаточно в файле `/etc/fstab` примонтировать диски сразу по идентификаторам UUID. Как в ALSE, собственно, и сделано.

Виртуальная файловая система

Из лабораторной по работе с файлами в Linux мы уже знаем, что файловая система Unix-подобных систем представлена единым пространством имен, что достигается через механизм монтирования различных файловых систем в единую структуру каталогов. Учитывая, что в единой структуре каталогов могут присутствовать совершенно разные файловые системы, потребовался унифицированный интерфейс, который бы обеспечил единообразное взаимодействие со всеми файловыми системами, и таким инструментом стала виртуальная файловая система (*от англ. Virtual File System, VFS*).

Система VFS позволяет с помощью одних и тех же системных вызовов взаимодействовать с любыми типами файловых систем, в том числе и сетевыми, как будто это обычная файловая система ext4 на локальном блочном устройстве. В состав VFS входят следующие типы объектов, которыми она оперирует:

- **суперблок** – метаданные файловой системы;
- **индексные дескрипторы** (*англ. inode*) – метаданные файла;
- **записи каталогов** (*англ. directory entry, dentry*) – специальные файлы, в которых хранятся жесткие ссылки, связывающие имена файлов с их индексными дескрипторами;
- **открытые файлы** – структуры, содержащие информацию об открытых файлах.

Для хранения информации о поддерживаемых типах файловых систем используется таблица, которая создается во время компиляции ядра. Каждая запись этой таблицы включает наименование типа и указатель на функцию, вызываемую во время монтирования этой файловой системы (если значение не указано, используется функция монтирования по умолчанию).

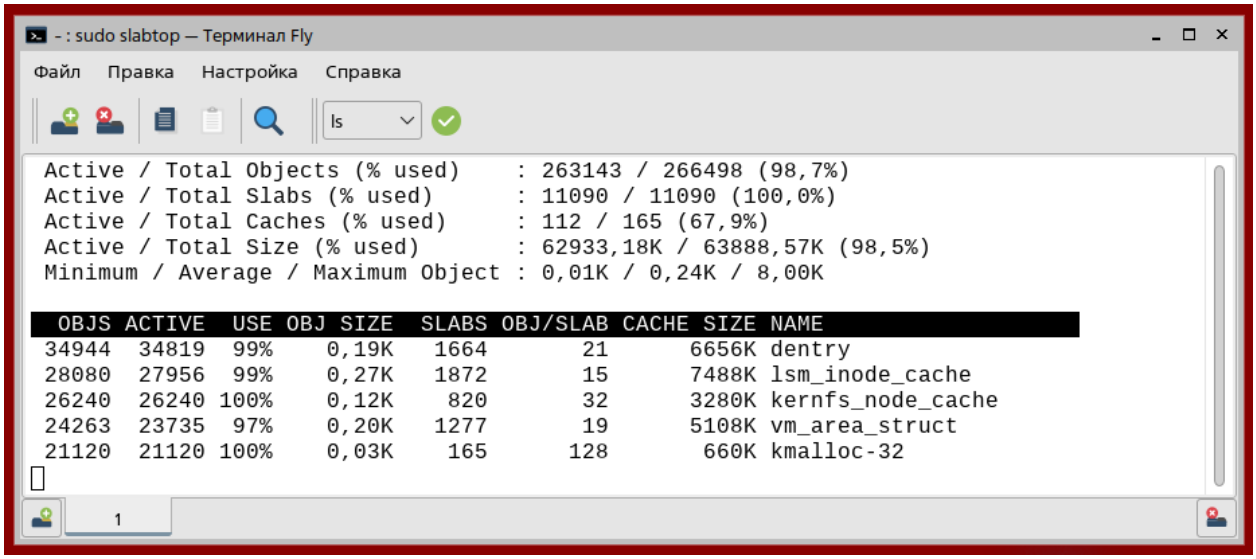
Функция монтирования используется для считывания суперблока, установки внутренних переменных и возврата дескриптора смонтированной системы обратно в VFS. После того, как система смонтирована, функции VFS используют открытый дескриптор для доступа к процедурам чтения/записи этой файловой системы.

Для более эффективного управления памятью и устранения значительной фрагментации данных система VFS кэширует метаданные, используемые при монтировании (суперблок, индексные дескрипторы, записи каталогов и др.), с помощью специального механизма управления памятью, который называется «распределение slab» (*англ. slab allocation*). Секрет быстродействия этого алгоритма состоит в повторном использовании памяти, освобожденной одним объектом, для размещения другого объекта того же типа.

Управление кэшированием метаданных осуществляется с помощью переменной ядра ОС `vm.vfs_cache_pressure` в файле `/proc/sys/vm/vfs_cache_pressure`, значение которой представляет из себя целое число:

- `0` – ядро сохраняет (не освобождает) страничный кэш, может привести к нехватке оперативной памяти.
- `100` – ядро пытается сохранить справедливое распределение памяти между кэшем страниц (pagecache) и кэшем подкачки (swapcache). Используется по умолчанию.
- `>100` – ядро активно выгружает метаданные в VFS. При значении существенно выше 100 это может негативно сказаться на производительности VFS.

Текущую статистику по использованию кэшей slab можно посмотреть утилитой `sudo slabtop`



Статистика по использованию кэшей slabtop

Для принудительного сохранения «грязных» страниц кэша (*англ. dirty pages*), которые были изменены после их помещения в кэш, можно воспользоваться утилитой `sync`. Она гарантирует, что все изменения в памяти будут записаны на диск, предотвращая потерю изменений в кэше при аварийном завершении работы системы.

Если же вам потребуется освободить память, зарезервированную под кэш, вы можете из-под суперпользователя выполнить команду:

```
echo 3 | sudo tee /proc/sys/vm/drop_caches
```

Где:

- `1` – освобождает pagecache,
- `2` – dentrie и inode кэши,
- `3` – все сразу.

```
sa@astra:~$ echo 3 | sudo tee /proc/sys/vm/drop_caches
3
```

Файловая система ext4

Операционная система Linux способна работать с разными типами файловых систем, как поддерживающими концепцию индексных дескрипторов (ext4, xfs, BtrFS), так и совершенно чуждые им (exFAT, NTFS). Список поддерживаемых файловых систем

определяется драйверами, то есть модулями, которые были включены в состав ядра и загружены в данный момент.

Список всех модулей ядра с драйверами файловых систем можно получить с помощью команды `ls /lib/modules/$(uname -r)/kernel/fs`.

```
sa@astra:~$ ls /lib/modules/$(uname -r)/kernel/fs
9p      bfs             cramfs  freevxfs  isofs  nfs         ocfs2      quota         udf
adfs    binfmt_misc.ko  dlm     fscache   jffs2  nfs_common  omfs       reiserfs      ufs
affs    btrfs           efs     fuse      jfs    nfsd        orangefs   romfs         vboxsf
afs     cachefiles      erofs   gfs2      ksmbd  nilfs2      overlayfs  shiftfs.ko    xfs
aufs    ceph            exfat   hfs       lockd  nls         pstore     smbfs_common  zonefs
autofs  cifs            f2fs    hfsplus   minix  ntfs        qnx4       sysv
befs    coda            fat     hpfs      netfs  ntfs3       qnx6       ubifs
```

Список файловых систем, поддерживаемых ядром в данный момент, можно получить командой `cat /proc/filesystems`. Словом, «nodev» (*англ. no device*) обозначаются файловые системы, которые не привязаны к устройствам, но это не означает, что данная система не используется, т.к. в эту категорию попадают псевдофайловые системы и системы, размещаемые в памяти.

```
sa@astra:~$ cat /proc/filesystems
nodev   sysfs
nodev   tmpfs
nodev   bdev
nodev   proc
nodev   cgroup
nodev   cgroup2
nodev   cpuset
nodev   devtmpfs
nodev   configfs
nodev   debugfs
nodev   tracefs
nodev   securityfs
nodev   sockfs
nodev   bpf
nodev   pipefs
nodev   ramfs
nodev   hugetlbfs
nodev   devpts
        ext3
        ext2
        ext4
        squashfs
        vfat
nodev   ecryptfs
        fuseblk
nodev   fuse
nodev   fusectl
nodev   mqueue
nodev   pstore
nodev   parsecfs
nodev   autofs
```

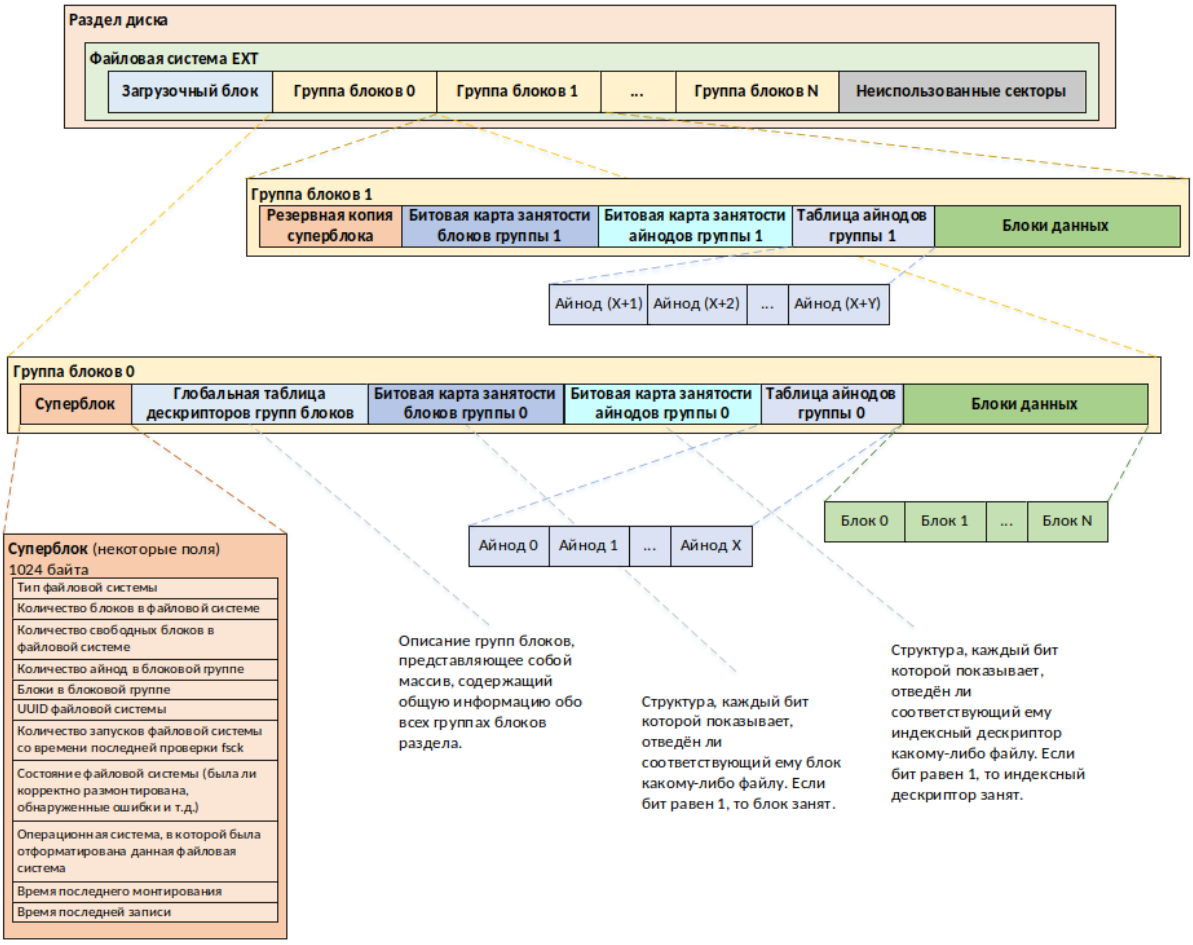
Для Linux родными считаются файловые системы из семейства Extended File System, EXT. Их разработка началась в начале 90-х для преодоления ограничений предшествующей им Minix File System как по длине имен файлов, так и по размеру поддерживаемых дисков.

К настоящему времени разработано несколько редакций, каждая из которых становилась значительно лучше предыдущей. В 2008 году была представлена четвертая и крайняя на текущий момент версия ext4, отличительными особенностями которой являются:

- Максимальный размер файла до 16 Тебибайт, 2^{40} байт (против 2 ТБ для ext3).
- Максимальный размер тома: до 1 Эксбибайта, 2^{60} байт (против 32 ТБ для ext3).
- Максимальное число подкаталогов в каждом каталоге 64 000 (против 32 000 для ext3).
- Максимальная длина имени файла 255 байт (или 127 русских символов в utf8).
- Журналируемая ФС, что помогает сохранить целостность файловой системы при сбоях.
- Использует контрольные суммы для автоматического обнаружения повреждения данных из-за аппаратных сбоев. КС добавлены для суперблока, inode, битовых карт блоков, блоков дерева экстентов, htree-узлов, журналов и др.
- Обратно-совместима к ext2 и ext3, что позволяет монтировать старые файловые системы через драйвер ext4.
- Поддерживает **пространственную запись файлов** – резервирование на диске всего требуемого пространства перед началом записи файла на диск.
- Поддерживает **экстенты** – хранение в дескрипторе адресов только первого и последнего блока из последовательного массива блоков данных. Размер экстента может достигать 128 МБ, что позволяет существенно повысить производительность ФС.
- Уменьшает фрагментацию данных за счет более рационального выделения блоков.
- Поддерживает отложенное выделение блоков памяти (непосредственно перед их записью на диск), что позволяет снизить нагрузку на кэш и увеличить производительность.
- Для работы с каталогами использует разновидность B-дерева, что позволяет без снижения производительности работать с большим количеством каталогов.
- Поддерживает быструю проверку занятых блоков, исключая из проверки индексные дескрипторы и свободные блоки данных.

К недостаткам ext4 можно отнести только отсутствие поддержки таких функций как дедупликация данных и снижение производительности при работе с разделами более 100 ТБ.

Хотя файловые системы семейства ext и отличаются друг от друга в деталях, но их структура практически идентична.



Общая структура файловой системы семейства ext

В системе Linux процедура разметки раздела для работы с конкретной файловой системой называется созданием файловой системы, но более привычный для Windows-администраторов термин «форматирование диска» тоже используется.

Как мы помним, в начале раздела часть секторов выделяется под размещение загрузочного блока, а после него размещаются данные. Для уменьшения фрагментации и повышения производительности файловая система ext4 нарезает все доступное пространство на фрагменты, которые называются группами блоков.

В файловой системе NTFS нет аналога для групп блоков, так как каждая группа в какой-то степени является еще одной файловой системой, у которой есть собственные блоки данных, таблица файловых дескрипторов и дополнительная метаинформация. Маленькие файлы, конечно, не должны выходить за пределы одной группы, но если нам потребуется сохранить большой архив, то он без проблем оккупирует несколько групп сразу.

Состав группы блоков 0

Первая группа блоков, которая идет под номером ноль, немного отличается от всех остальных и содержит дополнительные метаданные. В нулевую группу блоков входят следующие разделы, см. рисунок.

• **Суперблок** – это структура данных размером 1024 байта, в которой содержится информация о файловой системе. Для обеспечения надежности файловая система создаёт еще несколько резервных копий суперблока, и хранит их внутри других групп блоков в разных частях диска. Суперблок содержит следующую информацию:

- Тип файловой системы (s_type).
- Размер файловой системы в логических блоках, включая сам суперблок, ilist и блоки хранения данных (s_fsize).
- Размер массива индексных дескрипторов (s_isize).
- Число свободных блоков, доступных для размещения (s_tfree).
- Число свободных индексных дескрипторов, доступных для размещения (s_t inode).
- Состояние файловой системы (флаг модификации s_fmod, флаг режима монтирования s_fronly).
- Размер логического блока в виде числа, обозначающего степень 2 (9 – 512 байт, 10 – 1КБ).
- Количество логических блоков в группе блоков.
- Количество индексных дескрипторов в группе блоков.
- Идентификатор файловой системы (UUID).
- Время последней записи.
- Время последнего монтирования.

• **Глобальная таблица дескрипторов групп блоков** — это массив, в котором содержится информация о расположении всех групп блоков раздела.

• **Битовые карты занятости блоков и индексных дескрипторов** — помогают быстро определять, какие из блоков и индексных дескрипторов заняты, а какие свободны. Они представляют из себя массив двоичных данных, где каждый бит соответствует блоку данных или индексному дескриптору, а значение показывает занят он (1) или свободен (0).

• **Таблица индексных дескрипторов** — содержит массив индексных дескрипторов. Структура индексного дескриптора подробно рассматривалась в модуле про работу с файлами. Стоит отметить, что индексные дескрипторы создаются для всего раздела в целом и их нумерация уникальна для всех групп блоков.

• **Блоки данных** — объединяют группу секторов диска для хранения содержимого файлов, каталогов, косвенной адресации. Размер блоков данных кратен размеру сектора, который в большинстве случаев составляет 512 байт, но есть диски с секторами и по 4 Кб. Блоки данных по своей сути ближе всего к кластерам NTFS.

Состав групп блоков 1-N

Остальные группы блоков содержат следующие разделы:

- Битовая карта занятости блоков группы 0.
- Битовая карта занятости индексных дескрипторов группы 0.
- Таблица индексных дескрипторов группы 0.
- Блоки данных.

Некоторые группы блоков могут содержать резервную копию суперблока.

Посмотреть информацию суперблока можно, например, командой `sudo tune2fs -l /dev/sda1`.

```
sa@astra:~$ sudo tune2fs -l /dev/sda1
tune2fs 1.44.5 (15-Dec-2018)
Filesystem volume name:   <none>
Last mounted on:          /
Filesystem UUID:          eba34003-adda-47f3-a50d-8b9513eb81dc
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype needs_recovery
exten
t 64bit flex_bg sparse_super large_file huge_file dir_nlink extra_isize metadata_csum
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              1248480
Block count:              4992512
Reserved block count:     249625
Free blocks:              2847582
Free inodes:              1008564
First block:              0
Block size:               4096
Fragment size:            4096
Group descriptor size:    64
Reserved GDT blocks:      1024
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8160
Inode blocks per group:   510
Flex block group size:    16
Filesystem created:       Wed Mar 22 11:29:51 2023
Last mount time:          Wed Apr 23 10:08:39 2025
Last write time:          Wed Apr 23 10:08:37 2025
Mount count:              41
Maximum mount count:      -1
Last checked:             Wed Mar 22 11:29:51 2023
Check interval:           0 (<none>)
Lifetime writes:          68 GB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               256
Required extra isize:     32
Desired extra isize:      32
Journal inode:            8
Default directory hash:   half_md4
```

```
Directory Hash Seed:      c3561263-efd0-4edb-b8fb-7ec7a5c9538c
Journal backup:           inode blocks
Checksum type:            crc32c
Checksum:                 0x687dbb64
```

Чтобы получить информацию о файловых системах ext, можно воспользоваться утилитой `dumpe2fs`. Например, для вывода информации обо всех резервных копиях суперблока можно отфильтровать её вывод через grep:

```
sa@astra:~$ sudo dumpe2fs /dev/sda1 | grep superblock
dumpe2fs 1.44.5 (15-Dec-2018)
 Primary superblock at 0, Group descriptors at 1-3
 Backup superblock at 32768, Group descriptors at 32769-32771
 Backup superblock at 98304, Group descriptors at 98305-98307
 Backup superblock at 163840, Group descriptors at 163841-163843
 Backup superblock at 229376, Group descriptors at 229377-229379
 Backup superblock at 294912, Group descriptors at 294913-294915
 Backup superblock at 819200, Group descriptors at 819201-819203
 Backup superblock at 884736, Group descriptors at 884737-884739
 Backup superblock at 1605632, Group descriptors at 1605633-1605635
 Backup superblock at 2654208, Group descriptors at 2654209-2654211
 Backup superblock at 4096000, Group descriptors at 4096001-4096003
```

Рекомендуемая схема разбиения

В Linux у дисков есть разделы, но мы с ними не работаем как с логическими дисками Windows. Файловая система раздела может быть смонтирована в любую точку единой файловой системы, которая соответствует системному диску, с которого была выполнена загрузка. Точно также монтируются внешние носители информации, такие как USB-флешки, DVD-диски.

Например, для работы сервера можно использовать один раздел диска, но в высоконагруженных системах некоторые каталоги рекомендуется выносить отдельно по следующим причинам:

Каталог	Для чего используется	Почему может быть выгодно перенести в другой раздел	Рекомендуемый размер
/home	Пользовательские файлы	Домашние каталоги пользователей, аналог `C:\Users` в Windows. Перенос на другой раздел диска помогает исключить переполнение системного диска и гарантирует сохранность файлов при переустановке.	Не менее 100 МБ
/tmp	Временные файлы	Каталог хранит временные файлы. Перенос на другой раздел диска помогает исключить переполнение системного диска.	Не менее 50 МБ

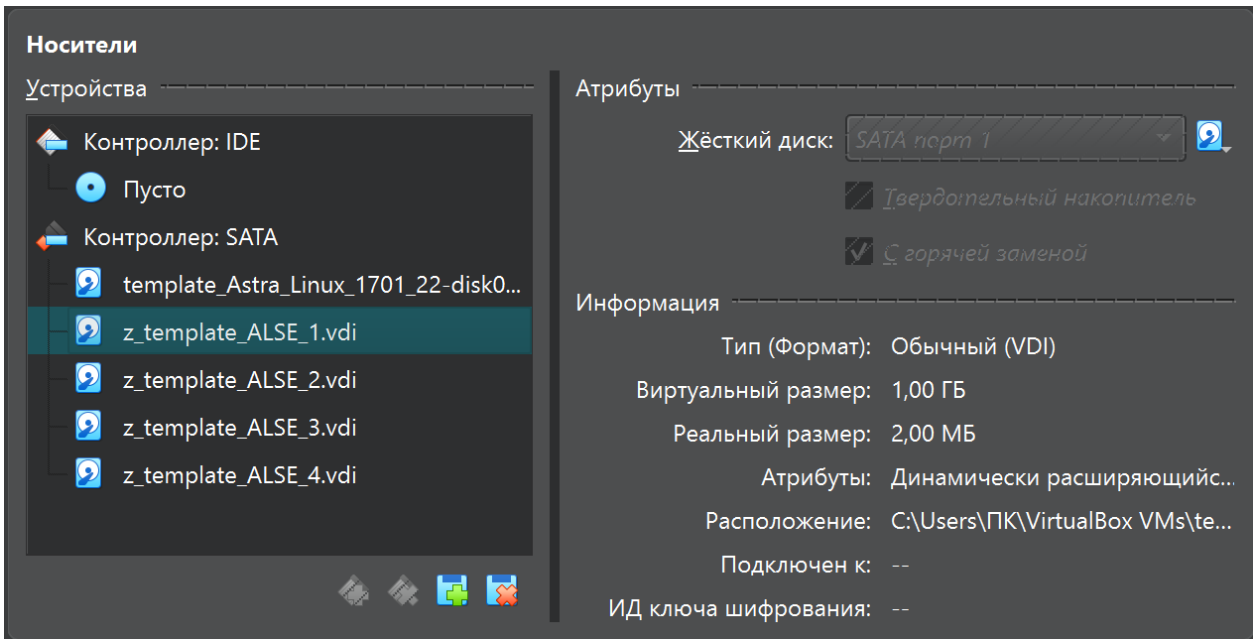
Каталог	Для чего используется	Почему может быть выгодно перенести в другой раздел	Рекомендуемый размер
/usr,/opt	Файлы программ	В этих каталогах хранятся файлы программ, аналог `Program Files` Если система работает на быстром SSD, то программы можно вынести на HDD большого объема.	Не менее 8 ГБ
/var	Динамические данные	Этот каталог используется для динамических данных, таких как файлы журнала, кэшированные данные и т.д. Перенос на другой раздел диска помогает исключить переполнение системного диска	Не менее 400 МБ
swap	Раздел подкачки	Раздел подкачки необходим для увеличения объема доступной памяти, аналог файла подкачки в Windows. Перенос на другой раздел диска позволяет повысить производительность, если это SSSD диск, и уменьшить фрагментацию файла подкачки.	Выбирается исходя из объема оперативной памяти. Если необходима гибернация, то не менее объема оперативной памяти + небольшой запас в 1-2 ГБ.

Однако, следует учитывать, что при разбиении диска на разделы мы можем ошибиться в своих предположениях о том, сколько места потребуется на те или иные задачи, и поменять это решение «на лету» уже не получится.

Решением этой проблемы является использование дополнительного уровня абстракции в виде системы логических томов, работающих поверх дисковых разделов. Универсальным решением являются менеджер логических томов LVM, который будет рассмотрен далее.

Управление дисковыми разделами и файловыми системами

Перед практическим изучением работы утилит, нам необходимо добавить несколько виртуальных дисков в нашу виртуальную машину, см. рисунок. Добавим 3 диска по 1ГБ и один диск, объемом 2ГБ.



Добавление дисков в VM в VirtualBox

Управление дисковыми разделами

Для управления дисковыми разделами, в Astra Linux доступно несколько утилит:

- Утилита `fdisk` – консольная утилита для разметки диска (создания и удаления разделов);
- Утилита `parted` – консольная утилита для разметки диска (создания, удаления и изменения разделов);
- Утилита `sfdisk` – утилита для использования в скриптах;
- Утилита `cfdisk` – псевдографическая утилита работы с разделами;
- Утилита `gparted` – графическая утилита разметки диска.

Утилита fdisk

Получение информации о дисках и разделах

После добавления дисков посмотрим список устройств хранения и их разделы с помощью команды `sudo fdisk -l` (L строчная).

```
sa@astra:~$ sudo fdisk -l
[sudo] пароль для sa:
Диск /dev/sda: 20 GiB, 21474836480 байт, 41943040 секторов
Disk model: VBOX HARDDISK
Единицы: секторов по 1 * 512 = 512 байт
Размер сектора (логический/физический): 512 байт / 512 байт
Размер I/O (минимальный/оптимальный): 512 байт / 512 байт
Тип метки диска: dos
Идентификатор диска: 0x4f437c6e

Устр-во    Загрузочный    начало    Конец  Секторы Размер Идентификатор Тип
```

```
/dev/sda1  *              2048 39942143 39940096    19G            83 Linux
/dev/sda2             39944190 41940991  1996802   975M             5 Расширенный
/dev/sda5             39944192 41940991  1996800   975M            82 Linux своп / Solaris


Диск /dev/sdb: 1 GiB, 1073741824 байт, 2097152 секторов
Disk model: VBOX HARDDISK
Единицы: секторов по 1 * 512 = 512 байт
Размер сектора (логический/физический): 512 байт / 512 байт
Размер I/O (минимальный/оптимальный): 512 байт / 512 байт


Диск /dev/sdc: 1 GiB, 1073741824 байт, 2097152 секторов
Disk model: VBOX HARDDISK
Единицы: секторов по 1 * 512 = 512 байт
Размер сектора (логический/физический): 512 байт / 512 байт
Размер I/O (минимальный/оптимальный): 512 байт / 512 байт


Диск /dev/sdd: 1 GiB, 1073741824 байт, 2097152 секторов
Disk model: VBOX HARDDISK
Единицы: секторов по 1 * 512 = 512 байт
Размер сектора (логический/физический): 512 байт / 512 байт
Размер I/O (минимальный/оптимальный): 512 байт / 512 байт


Диск /dev/sde: 2 GiB, 2147483648 байт, 4194304 секторов
Disk model: VBOX HARDDISK
Единицы: секторов по 1 * 512 = 512 байт
Размер сектора (логический/физический): 512 байт / 512 байт
Размер I/O (минимальный/оптимальный): 512 байт / 512 байт
```

Как мы видим, утилита вывела информацию о системном диске sda. Первые 4 раздела на нем являются основными, поэтому у третьего, расширенного раздела имя sda5. На новом созданном диске sdb разделов пока нет.

Для вывода информации только об одном диске необходимо к команде добавить имя диска, например, `sudo fdisk -l /dev/sdb`.

```
sa@astra:~$ sudo fdisk -l /dev/sdb
Диск /dev/sdb: 1 GiB, 1073741824 байт, 2097152 секторов
Disk model: VBOX HARDDISK
Единицы: секторов по 1 * 512 = 512 байт
Размер сектора (логический/физический): 512 байт / 512 байт
Размер I/O (минимальный/оптимальный): 512 байт / 512 байт
```

Утилита fdisk может работать также в интерактивном режиме. Для этого необходимо просто указать имя диска `sudo fdisk /dev/sdb`.

```
sa@astra:~$ sudo fdisk /dev/sdb

Добро пожаловать в fdisk (util-linux
2.33.1).
Изменения останутся только в памяти до тех пор, пока вы не решите записать
их.
Будьте внимательны, используя команду write.

Устройство не содержит стандартной таблицы разделов.
Создана новая метка DOS с идентификатором 0xa68f5e30.

Команда (m для справки):
```

Введите `m` и нажмите `Enter`, чтобы утилита вывела справку по работе с ней:

```
Команда (m для справки): m

Справка:

DOS (MBR)
a    переключение флага загрузки
b    редактирование вложенной метки диска BSD
c    переключение флага dos-совместимости

Общие
d удалить раздел
F показать свободное неразмеченное пространство
l список известных типов разделов
n добавление нового раздела
p вывести таблицу разделов
t изменение типа раздела
v проверка таблицы разделов
i вывести информацию о разделе

Разное
m вывод этого меню
u изменение единиц измерения экрана/содержимого
x дополнительная функциональность (только для экспертов)

Сценарий
I загрузить разметку из файла сценария sfdisk
O записать разметку в файл сценария sfdisk

Записать и выйти
w запись таблицы разделов на диск и выход
q выход без сохранения изменений

Создать новую метку
g создание новой пустой таблицы разделов GPT
G создание новой пустой таблицы разделов SGI (IRIX)
o создание новой пустой таблицы разделов DOS
s создание новой пустой таблицы разделов Sun

Команда (m для справки):
```

Введем список известных типов разделов командой `l` и нажмите `Enter`:

```
...
Команда (m для справки): l

0 Пустой           24  NEC DOS         81  Minix / старый  bf  Solaris
1 FAT12            27  Скрытый NTFS Wi 82  Linux своп / So c1  DRDOS/sec (FAT-
2 XENIX root       39  Plan 9          83  Linux           c4  DRDOS/sec (FAT-
3 XENIX usr        3c  PartitionMagic  84  OS/2 hidden or  c6  DRDOS/sec (FAT-
4 FAT16 <32M       40  Venix 80286     85  Linux расширен  c7  Syrinx
5 Расширенный      41  PPC PReP Boot   86  NTFS набор томо da  Данные не ФС
6 FAT16            42  SFS             87  NTFS набор томо db  CP/M / CTOS / .
7 HPFS/NTFS/exFAT  4d  QNX4.x          88  Linux plaintext de  Dell Utility
8 AIX              4e  QNX4.x 2-я част 8e  Linux LVM       df  BootIt
9 AIX загрузочный  4f  QNX4.x 3-я част 93  Amoeba          e1  DOS access
a OS/2 Boot-менед  50  OnTrack DM      94  Amoeba BBT      e3  DOS R/O
b W95 FAT32        51  OnTrack DM6 Aux 9f  BSD/OS          e4  SpeedStor
c W95 FAT32 (LBA)  52  CP/M            a0  IBM Thinkpad hi ea  Rufus alignment
e W95 FAT16 (LBA)  53  OnTrack DM6 Aux a5  FreeBSD         eb  BeOS фс
f W95 расшир. (LB  54  OnTrackDM6      a6  OpenBSD         ee  GPT
10 OPUS            55  EZ-Drive        a7  NeXTSTEP        ef  EFI (FAT-12/16/
11 Скрытый FAT12   56  Golden Bow      a8  Darwin UFS      f0  Linux/PA-RISC з
12 Compaq диагност 5c  Priam Edisk     a9  NetBSD          f1  SpeedStor
14 Скрытый FAT16 < 61  SpeedStor       ab  Darwin загрузоч f4  SpeedStor
16 Скрытый FAT16   63  GNU HURD или Sy af  HFS / HFS+      f2  DOS вторичный
17 Скрытый HPFS/NT 64  Novell Netware  b7  BSDI фс         fb  VMware VMFS
18 AST SmartSleep  65  Novell Netware  b8  BSDI своп       fc  VMware VMKCORE
1b Скрытый W95 FAT 70  DiskSecure Mult bb  Boot Wizard скр fd  Автоопределение
```

```
1c  Скрытый W95 FAT 75  PC/IX           bc  Acronis FAT32 L fe  LANstep
1e  Скрытый W95 FAT 80  Old Minix       be  Solaris загр.   ff  BBT

Команда (m для справки):
```

Как видите, список не маленький, тут есть и FAT и даже NTFS. Эта таблица в дальнейшем нам еще пригодится.

Создание разделов

Создадим новый раздел, используя команду `n` (при этом на первые 3 вопроса ответы можно опустить – нас устроит значение по умолчанию):

n (создать раздел) -> [**p**] (первичный) -> [**1**] (номер раздела) -> [**2048**] (первый сектор раздела, традиционно место в начале диска зарезервировано под код первичного загрузчика и «пустые» сектора).

В примере ниже `Enter` указан как кнопка, т.е. будет введены значения по умолчанию, вычисленные под размер вашего диска:

```
...
Команда (m для справки): n

Тип раздела
  p   основной    (0 первичный, 0 расширенный, 4 свободно)
  e   расширенный (контейнер для логических разделов)
Выберите (по умолчанию - p): <Enter>

Используется ответ по умолчанию p
Номер раздела (1-4, default 1): 1
Первый сектор (2048-2097151, default 2048): <Enter>
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2097151, default 2097151):
```

Затем утилита попросит ввести последний сектор раздела (значение по умолчанию – самый последний сектор, весь диск). Кроме номера сектора, можно использовать выражения **+/-<сектора>** или **+/-<размер>{K,M,G,T,P}**, где K, M, G, T и P – это килобайты, мегабайты, и так далее.

А мы создадим раздел размером в 200 МБ. Для это введем **+200M**.

```
...
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2097151, default 2097151): +200M

Создан новый раздел 1 с типом 'Linux' и размером 200 MiB

Команда (m для справки):
```

Введем команду `p` и выведем информацию о разделах на диске:

```
...
Команда (m для справки): p
Диск /dev/sdb: 1 GiB, 1073741824 байт, 2097152 секторов
Disk model: VBOX HARDDISK
Единицы: секторов по 1 * 512 = 512 байт
Размер сектора (логический/физический): 512 байт / 512 байт
```

```
Размер I/O (минимальный/оптимальный): 512 байт / 512 байт
Тип метки диска: dos
Идентификатор диска: 0xa68f5e30

Устр-во    Загрузочный начало  Конец Секторы Размер Идентификатор Тип
/dev/sdb1                2048 411647  409600   200M            83 Linux
```

Как видите, по умолчанию разделу был присвоен тип Linux (ему соответствует идентификатор 83).

Создадим еще пару разделов по 100МБ.

```
Устр-во   Загрузочный начало  Конец Секторы Размер Идентификатор Тип
/dev/sdb1                2048 411647 409600 200M              83 Linux
/dev/sdb2              411648 616447 204800 100M              83 Linux
/dev/sdb3              616448 821247 204800 100M              83 Linux
```

Изменение типа раздела

Изменим тип у второго раздела (/dev/sdb2) с Linux (83) на NTFS (87). Для этого введем команду `t`, выберем раздел и тип:

```
Команда (m для справки): t
Номер раздела (1-3, default 3): 2
Шестнадцатеричный код (введите L для получения списка кодов): 87

Тип раздела 'Linux' изменен на 'NTFS volume set'.
```

Выведем информацию о разделах ещё раз (команда `p`):

```
Команда (m для справки): p
Устр-во    Загрузочный начало Конец Секторы Размер Идентификатор Тип
/dev/sdb1                2048 411647 409600 200M              83 Linux
/dev/sdb2              411648 616447 204800 100M              87 NTFS набор томов
/dev/sdb3              616448 821247 204800 100M              83 Linux
```

Удаление раздела

Удалим второй раздел командой `d`.

```
...
Команда (m для справки): d
Номер раздела (1-3, default 3): 2

Раздел 2 был удален.

...
Команда (m для справки): p
Устр-во Загрузочный начало Конец Секторы Размер Идентификатор Тип
/dev/sdb1              2048 411647 409600 200M              83 Linux
/dev/sdb3            616448 821247 204800 100M              83 Linux
```

Завершение работы с утилитой fdisk

Завершим работу с утилитой fdisk командой `q` (без сохранения изменений).

Важно

Для того что бы сохранить изменения и завершить работу с утилитой, необходимо воспользоваться командой `w`, в противном случае, выполненные вами изменения с таблицей разделов не будут физически записаны на диск!

Создание таблицы разделов на диске

Для создания таблицы разделов на диске, запустим интерактивный режим работы утилиты с диском sdb воспользуемся командой `g`.

```
sa@astra:~$ sudo fdisk /dev/sdb

Добро пожаловать в fdisk (util-linux 2.33.1).
Изменения останутся только в памяти до тех пор, пока вы не решите записать их.
Будьте внимательны, используя команду write.

Устройство не содержит стандартной таблицы разделов.
Создана новая метка DOS с идентификатором 0x96ad1d22.

Команда (m для справки): g
Created a new GPT disklabel (GUID: 448A4E91-2482-B04E-A506-87C4BE8D7501).
```

Создадим один раздел размером 100МБ, выведем информацию о созданном разделе командой `<i>` и выйдем, сохранив изменения.

```
...
Команда (m для справки): n
Номер раздела (1-128, default 1):
Первый сектор (2048-2097118, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2097118, default 2097118): +100M

Создан новый раздел 1 с типом 'Linux filesystem' и размером 100 MiB.

Команда (m для справки): i
Выбранный раздел 1
        Device: /dev/sdb1
         Start: 2048
           End: 206847
       Sectors: 204800
          Size: 100M
          Type: Файловая система Linux
     Type-UUID: 0FC63DAF-8483-4772-8E79-3D69D8477DE4
          UUID: 07ADD7EC-E46D-D841-8683-F7D70C560196

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

После создания разделов с помощью `fdisk` стоит проверить, дошла ли информация о новых разделах до ядра ОС. Для этого воспользуемся командой `lsblk`.

```
sa@astra:~$ 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  975M  0 part [SWAP]
sdb      8:16   0    1G  0 disk
└─sdb1   8:17   0  100M  0 part
sdc      8:32   0    1G  0 disk
```

```
sdd      8:48   0    1G  0 disk
sde      8:64   0    2G  0 disk
sr0     11:0    1 1024M  0 rom
```

Если информация не обновилась, необходимо воспользоваться командой `sudo partprobe`.

Утилита parted

Знакомство с утилитой `parted` стоит начать со справки. Запустим утилиту `sudo parted` и выведем список доступных команд используя команду (parted) `help`.

```
sa@astra:~$ sudo parted
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) help
 align-check TYPE N                        check partition N for TYPE(min  opt) alignment
 help [COMMAND]                           print general help, or help on COMMAND
 mklabel,mktable LABEL-TYPE               create a new disklabel (partition table)
 mkpart PART-TYPE [FS-TYPE] START END     make a partition
 name NUMBER NAME                         name partition NUMBER as NAME
 print [devices  free  list,all|NUMBER]   display the partition table, available devices,
       free space, all found partitions, or a particular partition
quit                                      exit program
rescue START END                          rescue a lost partition near START and END
resizepart NUMBER END                     resize partition NUMBER
rm NUMBER                                 delete partition NUMBER
select DEVICE                             choose the device to edit
disk_set FLAG STATE                       change the FLAG on selected device
disk_toggle [FLAG]                        toggle the state of FLAG on selected device
set NUMBER FLAG STATE                     change the FLAG on partition NUMBER
toggle [NUMBER [FLAG]]                    toggle the state of FLAG on partition NUMBER
unit UNIT                                 set the default unit to UNIT
version                                   display the version number and copyright
      information of GNU Parted
(parted)
```

Как видим, в отличии от утилиты `fdisk`, которая не умеет сама изменять размеры созданных разделов, утилита `parted`, напротив, позволяет это делать.

Важно

В отличии от `fdisk`, `gparted` вносит изменения сразу, перед некоторыми опасными операциями она просит подтвердить выполнение действия, например, при уменьшении размера диска. Но при выполнении некоторых операций, она такого подтверждения не потребует, например, при удалении раздела.

Обратите внимание, что по умолчанию утилита будет использовать первое устройство (3 строка – Using /dev/sda). Для выбора определенного устройства при запуске утилиты необходимо указать его имя, например, `sudo parted /dev/sdb`.

И в том и в другом случае, утилита parted будет работать в интерактивном режиме, ожидая от пользователя ввода её внутренних команд (parted). Для запуска утилиты в не

интерактивном режиме достаточно вызвать её с необходимыми параметрами, например, `sudo parted /dev/sdc mklabel gpt`.

```
sa@astra:~$ sudo parted /dev/sdc mklabel gpt
Information: You may need to update /etc/fstab.
```

Получение информации о дисках и разделах

Команда (parted) `print` выведет информацию о текущем диске.

```
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 21,5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system     Flags
1      1049kB  20,5GB  20,4GB  primary   ext4            boot
2      20,5GB  21,5GB  1022MB  extended
5      20,5GB  21,5GB  1022MB  logical   linux-swap(v1)
```

Выведем список устройств командой (parted) `print devices`.

```
(parted) print devices
/dev/sda (21,5GB)
/dev/sdb (1074MB)
/dev/sdc (1074MB)
/dev/sdd (1074MB)
/dev/sde (2147MB)
```

Кроме `print devices` есть и другие команды для вывода информации о дисках.

Команда (parted) `print free` выводит информацию о свободном месте разделах текущего диска.

```
(parted) print free
Model: ATA VBOX HARDDISK (scsi)

Disk /dev/sda: 21,5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system     Flags
        32,3kB  1049kB  1016kB            Free Space
1       1049kB  20,5GB  20,4GB  primary   ext4            boot
        20,5GB  20,5GB  1048kB            Free Space
2       20,5GB  21,5GB  1022MB  extended
5       20,5GB  21,5GB  1022MB  logical   linux-swap(v1)
        21,5GB  21,5GB  1049kB            Free Space
```

Команда (parted) `print list` выводит информацию о дисках списком.

```
(parted) print list
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 21,5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
```

```
Number  Start   End     Size    Type      File system     Flags
1      1049kB  20,5GB  20,4GB  primary   ext4            boot
2      20,5GB  21,5GB  1022MB  extended
5      20,5GB  21,5GB  1022MB  logical   linux-swap(v1)


Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End    Size   File system  Name  Flags
1      1049kB  106MB  105MB

. . .
```

Выбор раздела

Для выбора раздела введем команду (parted):

```
select <имя_устройства>
```

Например, выберем второй диск командой `select /dev/sdb`.

Создание разделов

Для создания разделов воспользуемся командой (parted):

```
mkpart <PART-TYPE> [<FS-TYPE>] <START> <END>
```

Где:

• **<PART-TYPE>** — для MBR указывает тип раздела (primary, extended или logical), а для GPT устанавливает метку раздела,

• **<FS-TYPE>** – определяет тип файловой системы,

• **<START>** и **<END>** – начало и конец раздела в процентном выражении от объема диска, в секторах (s), мегабайтах (MB), гигабайтах (GB) и других единицах.

Создадим раздел размером в 20% объёма диска /dev/sdb командой

```
(parted) mkpart Data ext4 106MiB 20%
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number Start  End    Size File system Name Flags
1      1049kB 106MB 105MB
2      111MB  215MB 104MB            Data
```

Изменение размера раздела

Для изменения размера раздела, воспользуемся командой (parted):

```
resizepart <номер_раздела> <новый_конец_раздела>
```

Например, увеличим второй раздел на 100МиБ `resizepart 2 315MiB`.

```
(parted) resizepart 2 315MiB
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number Start   End   Size   File system  Name  Flags
1      1049kB 106MB 105MB
2      111MB  330MB 219MB   ext4        Data
```

А теперь уменьшим размер второго раздела на 19 Мб `resizepart 2 297MiB`.

```
(parted) resizepart 2 297MiB
Warning: Shrinking a partition can cause data loss, are you sure you want to continue?
Yes/No? Y
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start  End    Size    File system    Name    Flags
1      1049kB  106MB 105MB
2      111MB   311MB 200MB    ext4           Data
```

При этом, утилита попросит нас подтвердить операцию, так как при ней возможно потеря данных.

Удаление раздела

Для удаления раздела служит команда (parted) `rm <номер_раздела>`. Удалим первый раздел на диске /dev/sdb командой `rm 1`.

```
(parted) select /dev/sdb
Using /dev/sdb
(parted) rm 1
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start  End   Size    File system   Name   Flags
2      111MB  311MB  200MB                Data
```

Завершим работу с утилитой командой (parted) `quit`.

```
(parted) quit
sa@astra:~$
```

После создания разделов утилитой `parted` стоит проверить, дошла ли информация о новых разделах до ядра ОС. Для этого воспользуемся командой `lsblk`.

```
sa@astra:~$ 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 975M  0 part [SWAP]
sdb    8:16      0   1G  0 disk
└─sdb2 8:18      0 191M  0 part
sdc    8:32      0   1G  0 disk
sdd    8:48      0   1G  0 disk
sde    8:64      0   2G  0 disk
sr0    11:0      1 1024M 0 rom
```

Если информация не обновилась, необходимо воспользоваться командой `sudo partprobe`.

Управление файловыми системами

Создание и изменение файловой системы

Для управления файловыми системами есть несколько команд:

- `mkfs.<тип_файловой_системы> [<параметры>] <устройство>` – создание файловой системы
- `mkswap <устройство>` – создание раздела подкачки
- `resize2fs <устройство> <размер>` – изменение размера раздела файловой системы

Команда `mkfs` использует настройки по умолчанию, которые заданы в файле `/etc/mke2fs.conf`. Обратите внимание, что некоторые команды `mkfs` являются ссылками на другие утилиты:

```
sa@astra:~$ ls -l /usr/sbin | grep mkfs
lrwxrwxrwx 1 root root         8 мая 13  2018 mkdosfs -> mkfs.fat
-rwxr-xr-x 1 root root     14848 сен 30  2022 mkfs
-rwxr-xr-x 1 root root     35336 сен 30  2022 mkfs.bfs
-rwxr-xr-x 1 root root     39440 сен 30  2022 mkfs.cramfs
lrwxrwxrwx 1 root root         9 ноя  1  2018 mkfs.exfat -> mkexfatfs
lrwxrwxrwx 1 root root         6 янв 10  2020 mkfs.ext2 -> mke2fs
lrwxrwxrwx 1 root root         6 янв 10  2020 mkfs.ext3 -> mke2fs
lrwxrwxrwx 1 root root         6 янв 10  2020 mkfs.ext4 -> mke2fs
-rwxr-xr-x 1 root root     39968 мая 13  2018 mkfs.fat
-rwxr-xr-x 1 root root    105048 сен 30  2022 mkfs.minix
lrwxrwxrwx 1 root root         8 мая 13  2018 mkfs.msdos -> mkfs.fat
lrwxrwxrwx 1 root root         6 июн  9  2022 mkfs.ntfs -> mkntfs
lrwxrwxrwx 1 root root         8 мая 13  2018 mkfs.vfat -> mkfs.fat
-rwxr-xr-x 1 root root    355768 фев 22  2019 mkfs.xfs
```

С параметрами утилиты можно ознакомиться в справке, но стоит выделить параметр `-i`, который задает количество байт информации на один индексный дескриптор, что в свою очередь определяет, сколько места на диске будет отдано под индексные дескрипторы. Так как для файловых систем ext количество индексных дескрипторов жестко задается при создании ФС, стоит сразу на этапе её создания определиться с их числом.

Создадим файловую систему ext4 на разделе диска `/dev/sdb` с параметрами по умолчанию:

```
sa@astra:~$ sudo mkfs.ext4 /dev/sdb2
mke2fs 1.44.5 (15-Dec-2018)
Creating filesystem with 195584 1k blocks and 48960 inodes
Filesystem UUID: 746ad73d-a27f-4c32-bcf3-1431e06dc15d
Superblock backups stored on blocks:
       8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

Теперь немного расширим этот раздел (~100МБ) и расширим файловую систему.

```
sa@astra:~$ sudo parted /dev/sdb resizepart 2 400MiB
Information: You may need to update /etc/fstab.
sa@astra:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   20G  0 disk
├─sda1   8:1    0   195G  0 part /
├─sda2   8:2    0    1K  0 part
└─sda5   8:5    0  975M  0 part [SWAP]
sdb      8:16   0    1G  0 disk
└─sdb2   8:18   0  294M  0 part
sdc      8:32   0    1G  0 disk
sdd      8:48   0    1G  0 disk
sde      8:64   0    2G  0 disk
sr0      11:0   1 1024M  0 rom
sa@astra:~$ sudo resize2fs /dev/sdb2
resize2fs 1.44.5 (15-Dec-2018)
Resizing the filesystem on /dev/sdb2 to 301056 (1k) blocks.
The filesystem on /dev/sdb2 is now 301056 (1k) blocks long.
```

Проверка целостности файловой системы

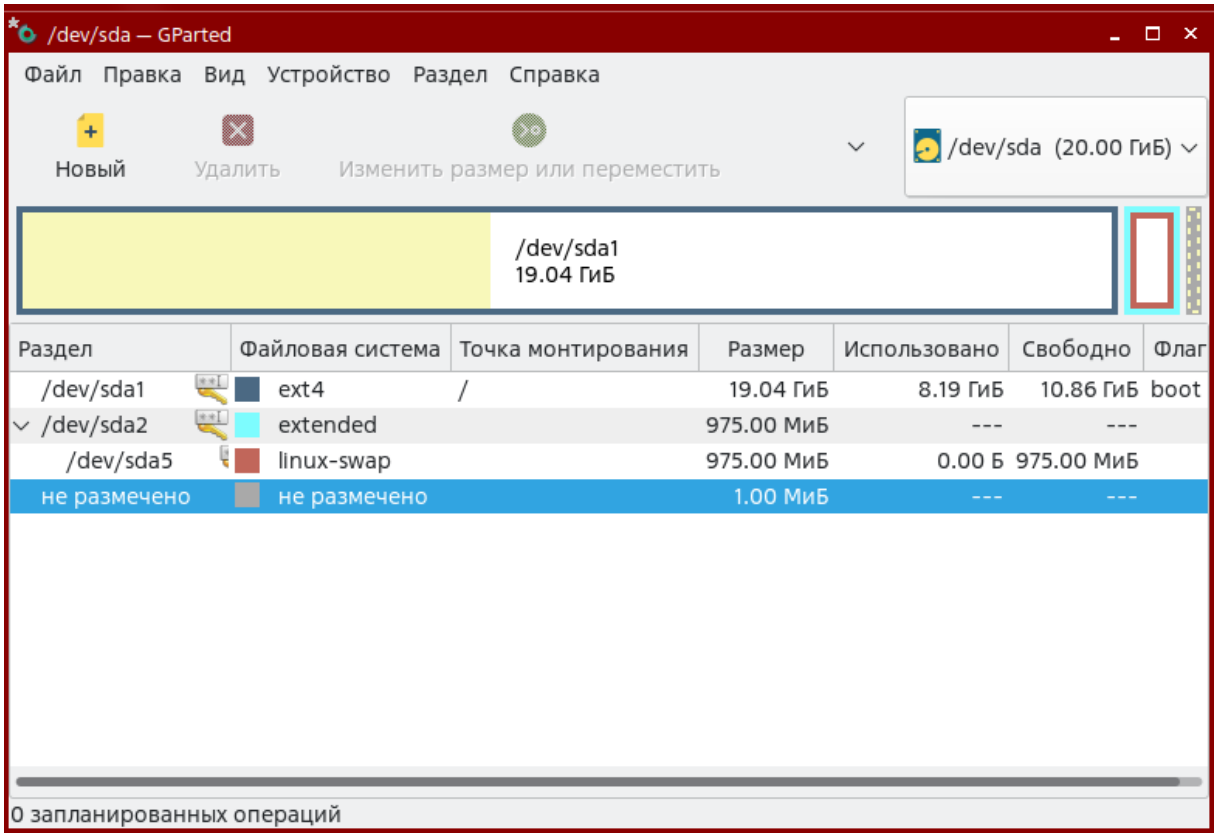
Для проверки целостности файловой системы служит команда `fsck` (*англ. file system check*). Для её успешной работы раздел не должен быть смонтирован. Запустим её для раздела `/dev/sdb2`:

```
sa@astra:~$ sudo fsck /dev/sdb2
fsck из util-linux 2.33.1
e2fsck 1.44.5 (15-Dec-2018)
/dev/sdb2: clean, 11/75480 files, 15692/301056 blocks
```

Графическая утилита GParted

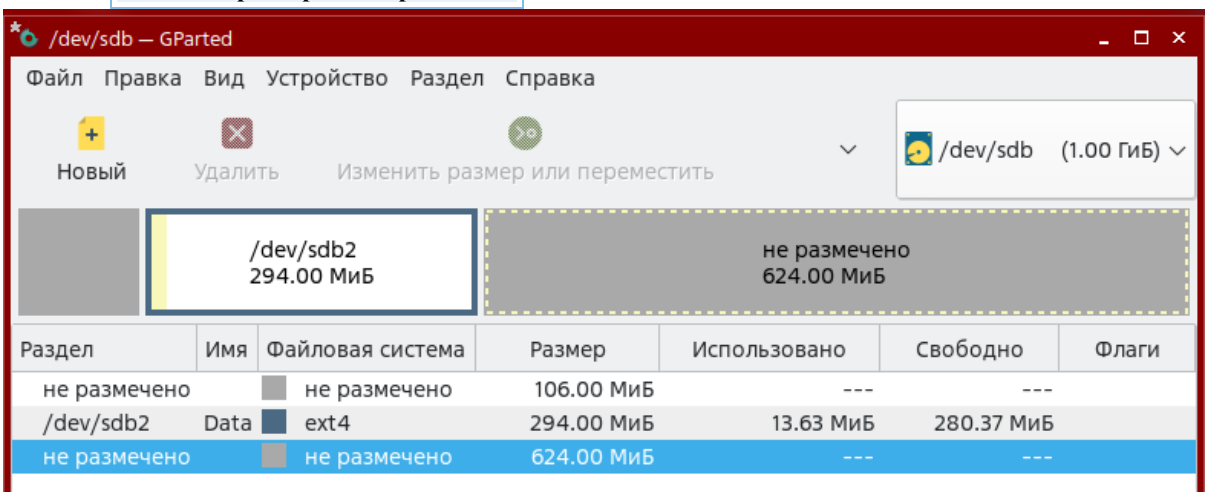
С помощью графической утилиты `gparted`, имеющей интуитивно понятный интерфейс, можно как разметить диск, так и создать на его разделах файловые системы, а при необходимости изменить их размеры и атрибуты.

Утилита Gparted может быть запущена из меню **Пуск ▸ Системные ▸ Редактор разделов Gparted** или из командной строки (`sudo gparted`).

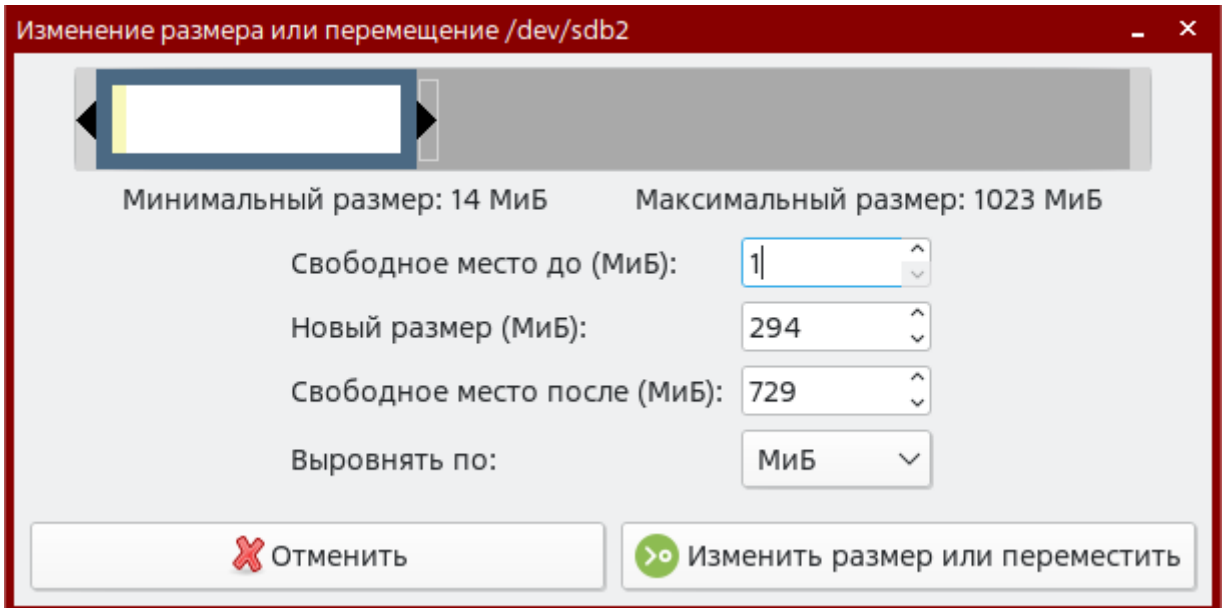


Окно утилиты Gparted

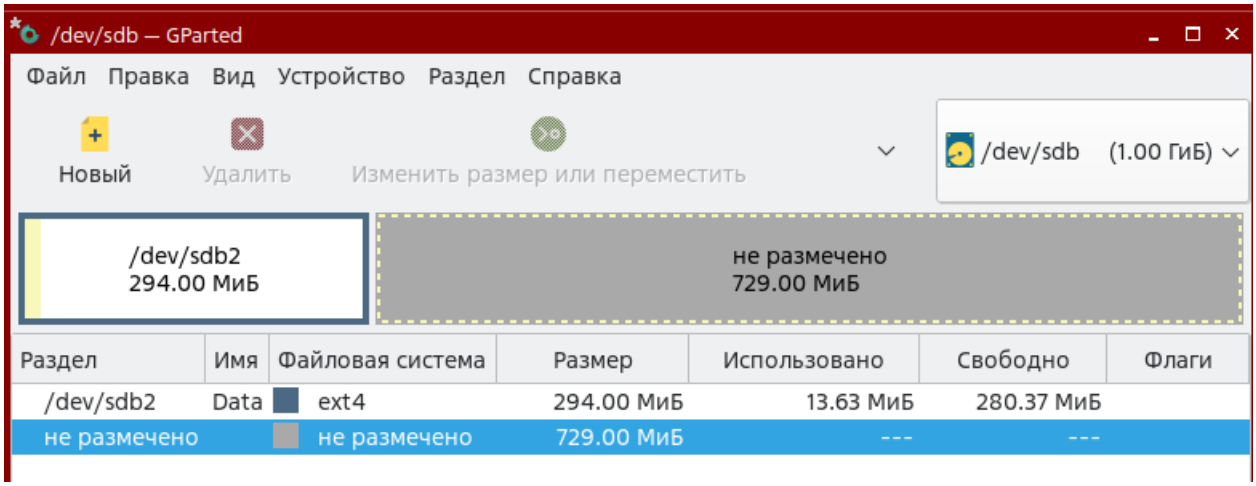
Выберем в правом верхнем углу диск `/dev/sdb`, в нем раздел sdb2, и применим действие **Изменить размер или переместить**



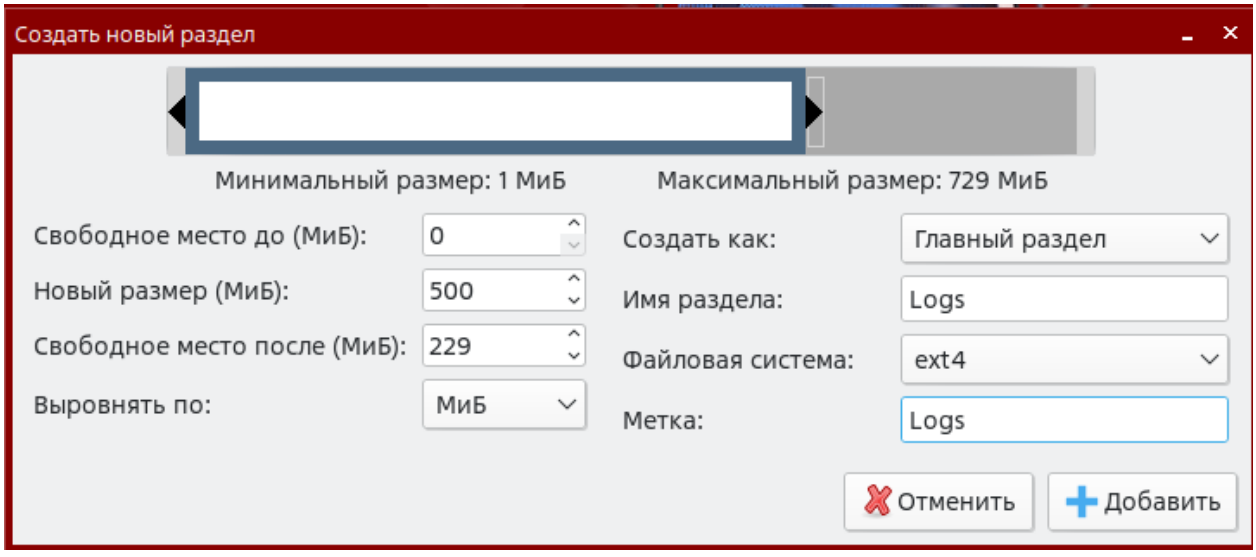
Переместим этот раздел в начало диска.



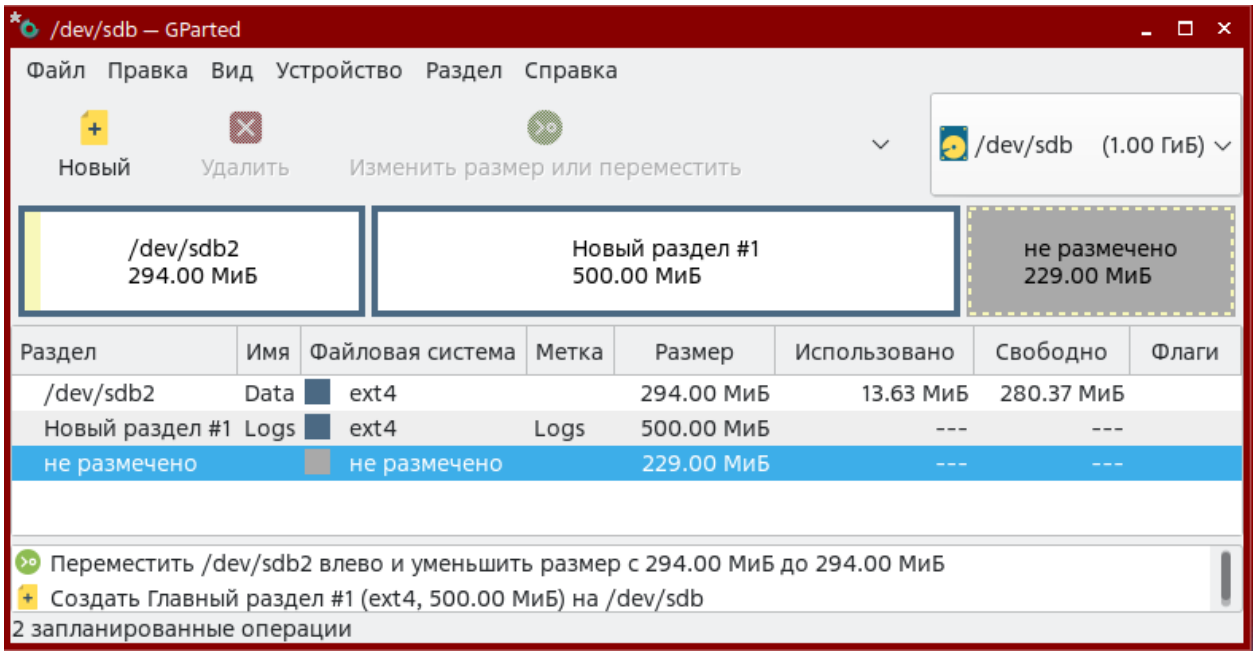
Согласимся с предупреждением утилиты.



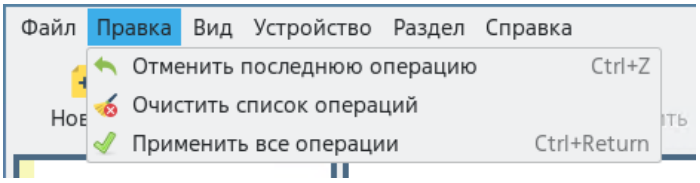
Теперь создадим новый раздел в свободной области диска размером в 500МБ.



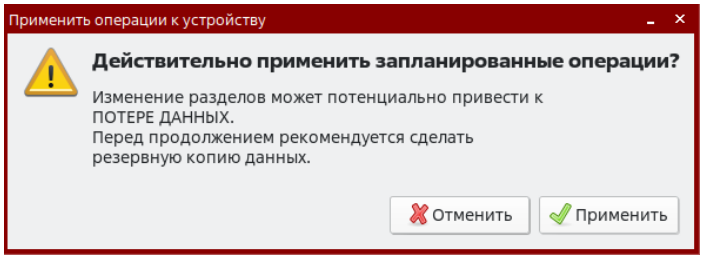
Раздел был создан и осталось свободное место.



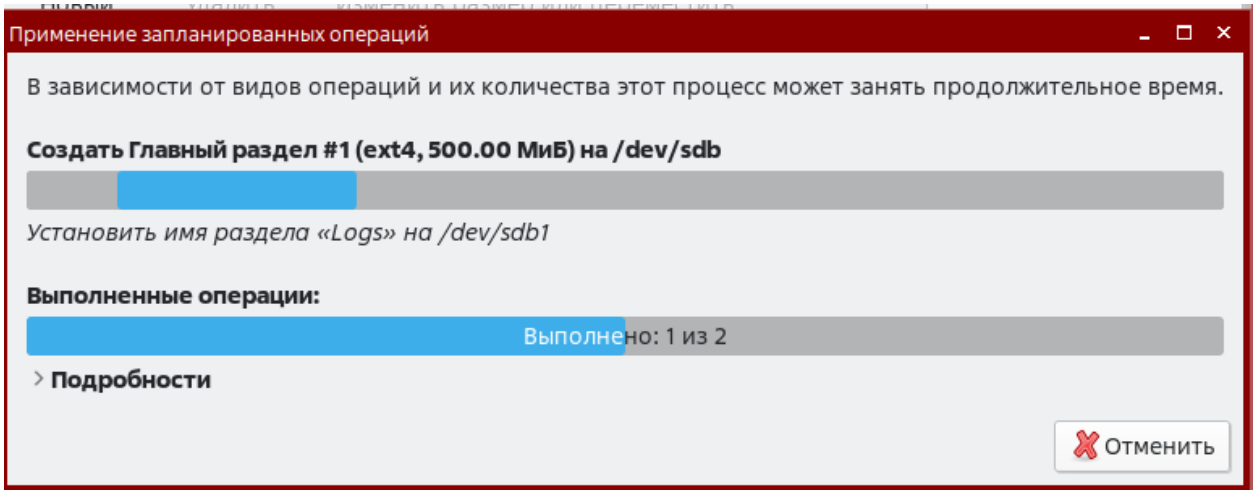
Заметьте, что фактических операций с диском еще не проводилось, они были только запланированы. Для применения изменений, необходимо выбрать пункт **Применить все операции** в меню **Правка**.

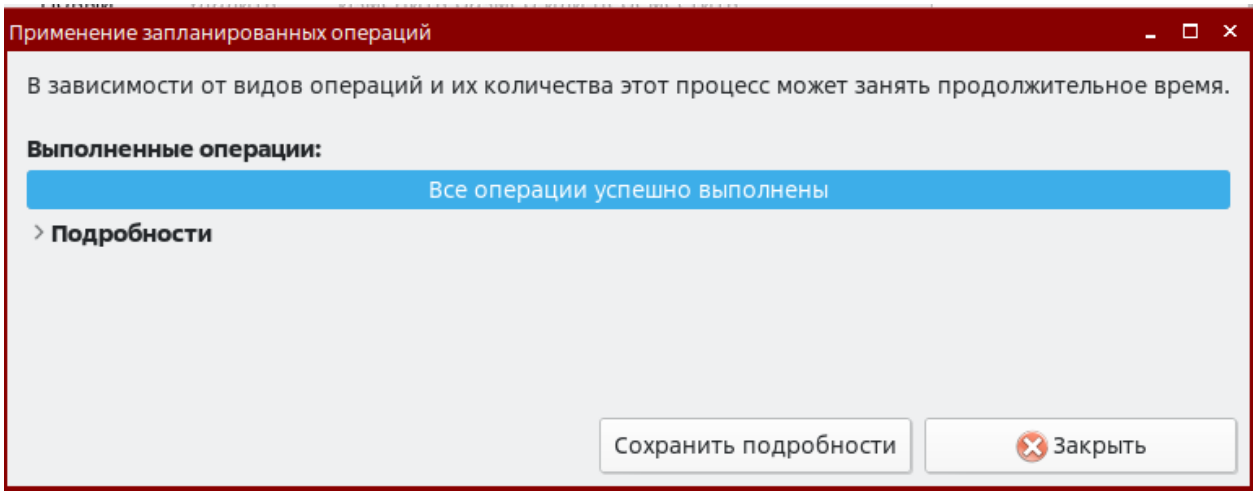


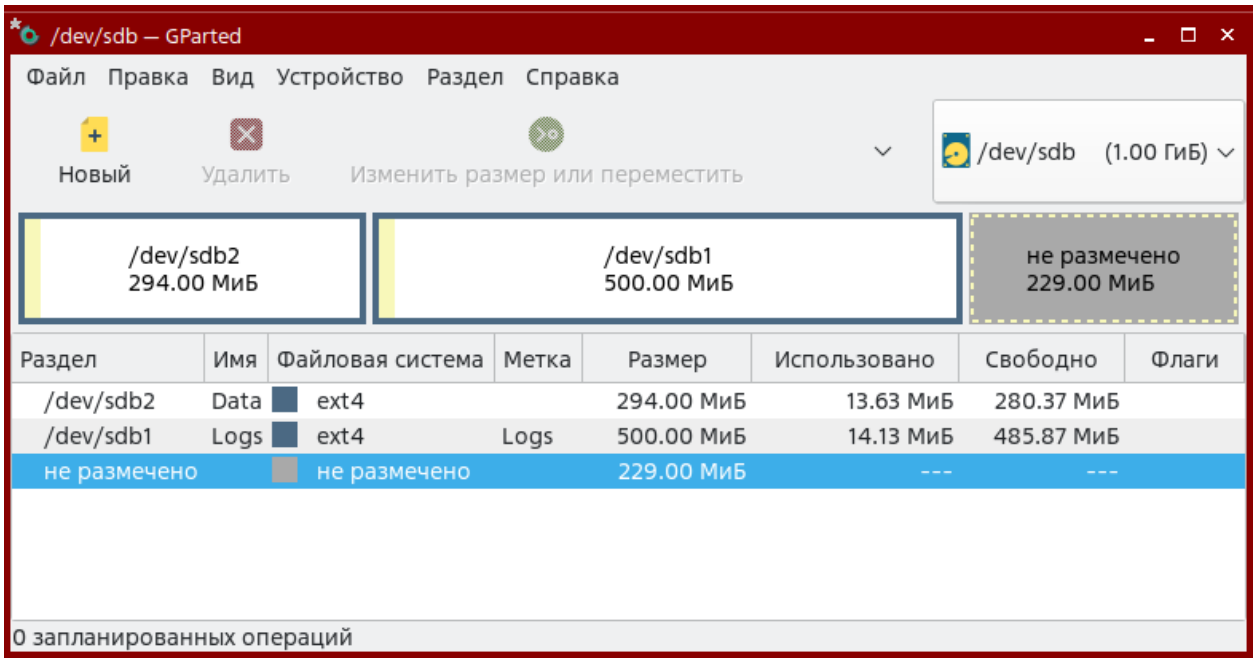
Применим операции, подтвердив их.



В процессе применения изменений вы будете видеть статус операции, а после их завершения отчет о выполнении.







Мы познакомились с тем, как в Linux можно разметить диски и создать на них файловые системы. Однако, для доступа к файловым системам этого недостаточно. Как в Windows мы не можем стандартными средствами обратиться к данным на диске, если ему не назначена буква, так и в Linux мы не можем обратиться к файловой системе, если она не

была смонтирована. Поэтому, процессу монтирования ФС будет посвящен следующий раздел.

Монтирование файловых систем

Чтобы данные раздела диска стали доступны, этот раздел должен быть смонтирован в корневую файловую систему. Процесс монтирования файловой системы с точки зрения оператора ПК не очень сложен.

Существуют следующие виды монтирования:

- Временное через консольную команду `mount`.
- Постоянное через файл `/etc/fstab`.
- Монтирование с помощью службы systemd.

Опции монтирования:

- **auto** – файловая система монтируется автоматически.
- **ro** – монтируется в режиме «только чтение».
- **rw** – монтируется в режиме «чтение и запись».
- **dev** – файловая система может содержать файлы блочных и символьных устройств.
- **exec** – файловая система может содержать исполняемые файлы.
- **suid** – разрешено использование битов SUID и SGID.
- **user** – разрешено обычному пользователю размонтировать данную файловую систему и пр этом используются значения по умолчанию (noexec, nosuid, nodev).
- **defaults** – установки по умолчанию (rw, suid, dev, exec, nouser, async).
- **codepage=кодовая_страница** – применять указанную кодировку к именам файлов.
- **iocharset=набор_символов** – отображать имена файлов в соответствии с указанным набором символов.
- **noauto** – файловая система не может быть автоматически смонтирована.
- **nodev** – файловая система не может содержать файлы блочных и символьных устройств.
- **noexec** – файловая система не позволяет запускать исполняемые файлы.
- **nosuid** – запрещено использование битов SUID и SGID.
- **nouser** – обычному пользователю запрещено размонтировать данную файловую систему.

Временное монтирование

Временное монтирование выполняется утилитой `mount` и действует до очередной перезагрузки системы. Программа mount способна самостоятельно определить тип

файловой системы, либо тип можно указать явно с помощью опции `-t`. Для определения типа файловой системы утилита mount использует библиотеку *blkid*.

Вывод информации о примонтированных ФС

Для вывода информации обо всех примонтированных в данный момент ФС и их параметров монтирования можно вызвать утилиту `mount` с ключом `-l` (строчная L) или без параметров вовсе.

```
sa@astra:~$ mount -l
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs
(rw,nosuid,relatime,size=1960700k,nr_inodes=490175,mode=755,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=402048k,mode=755,inode64)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
parsecfs on /parsecfs type parsecfs (rw,relatime,sync)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755,inode64)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup
(rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup
(rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/misc type cgroup (rw,nosuid,nodev,noexec,relatime,misc)
mqueue on /dev/mqueue type mqueue (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs
(rw,relatime,fd=33,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=17567)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
tmpfs on /run/user/107 type tmpfs
(rw,nosuid,nodev,relatime,size=402044k,mode=700,uid=107,gid=118,inode64)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=402044k,mode=700,uid=1000,gid=1000,inode64)
```

Для отображения информации только о реальных файловых системах можно воспользоваться фильтрацией вывода через утилиту `grep mount | grep "^/dev"`.

```
sa@astra:~$ mount | grep "^/dev"
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
```

Более наглядный вывод дает утилита `findmnt`. Чтобы с ее помощью получить список реальных файловых систем, можно использовать ее с ключом `findmnt --real`.

```
sa@astra:~$ findmnt --real
TARGET        SOURCE    FSTYPE   OPTIONS
/             /dev/sda1 ext4     rw,relatime,errors=remount-ro
├─/sys/fs/bpf bpf       bpf      rw,nosuid,nodev,noexec,relatime,mode=700
└─/parsecfs   parsecfs  parsecfs rw,relatime,sync

sa@astra:~$ findmnt
TARGET                          SOURCE     FSTYPE        OPTIONS
/                               /dev/sda1  ext4          rw,relatime,errors=remount-ro
├─/sys                          sysfs      sysfs         rw,nosuid,nodev,noexec,relatime
│ ├─/sys/kernel/security        securityfs security      rw,nosuid,nodev,noexec,relatime
│ ├─/sys/fs/cgroup              cgroup2    cgroup2
rw,nosuid,nodev,noexec,relatime,nsdelegat
│ ├─/sys/fs/pstore              pstore     pstore        rw,nosuid,nodev,noexec,relatime
│ ├─/sys/fs/bpf                 bpf        bpf
rw,nosuid,nodev,noexec,relatime,mode=700
│ ├─/sys/kernel/debug           debugfs    debugfs       rw,relatime
│ ├─/sys/kernel/config          configfs   configfs      rw,relatime
│ └─/sys/fs/fuse/connections    fusectl    fusectl       rw,relatime
├─/proc                         proc       proc          rw,nosuid,nodev,noexec,relatime
│ └─/proc/sys/fs/binfmt_misc    systemd-1  autofs
rw,relatime,fd=25,pgrp=1,timeout=0,minpro
├─/dev                          udev       devtmpfs
rw,nosuid,relatime,size=962268k,nr_inodes
│ ├─/dev/pts                    devpts     devpts
rw,nosuid,noexec,relatime,gid=5,mode=620,
│ ├─/dev/shm                    tmpfs      tmpfs         rw,nosuid,nodev,inode64
│ ├─/dev/mqueue                 mqueue     mqueue        rw,relatime
│ └─/dev/hugepages              hugetlbfs  hugetlbf      rw,relatime,pagesize=2M
├─/run                          tmpfs      tmpfs
rw,nosuid,noexec,relatime,size=202552k,mo
│ ├─/run/lock                   tmpfs      tmpfs
rw,nosuid,nodev,noexec,relatime,size=5120
│ ├─/run/user/1000              tmpfs      tmpfs
rw,nosuid,nodev,relatime,size=202548k,mod
│ │ └─/run/user/1000            tmpfs[/user/private/1000/l0i0c0x0t0x0]
│ │                             tmpfs
rw,nosuid,noexec,relatime,size=202552k,mo
│ └─/run/user/107               tmpfs      tmpfs
rw,nosuid,nodev,relatime,size=202548k,mod
│ └─/run/user/107               tmpfs[/user/private/107/l0i0c0x0t0x0]
│                               tmpfs
rw,nosuid,noexec,relatime,size=202552k,mo
└─/parsecfs                     parsecfs   parsecfs      rw,relatime,sync
```

Вызов команды `findmnt` с опцией -p позволит следить за процессом монтирования и размонтирования в реальном времени. Откроем новую вкладку в терминале и запустим там эту команду `findmnt -p`.

Создание точки монтирования

Создадим в каталоге `/mnt` подкаталог data и примонтируем в него устройство `/dev/sdb2` с опциями по умолчанию. Синтаксис команды `mount` при монтировании:

`sudo mount [<опции>] <устройство> <точка_монтирования>`.

```
sa@astra:~$ sudo mkdir /mnt/data
sa@astra:~$ sudo mount /dev/sdb2 /mnt/data
sa@astra:~$ ls /mnt/data
lost+found
```

Теперь создадим файл test.txt в /mnt/data и наполним его содержимым.

```
sa@astra:~$ sudo -i
root@astra:~# echo "Привет,мир!" > /mnt/data/test.txt
root@astra:~# exit
выход
sa@astra:~$ cat /mnt/data/test.txt
Привет,мир!
```

Перемонтированные устройства

Теперь перемонтируем файловую систему с опцией только на чтение. Для этого выполним команду:

```
sa@astra:~$ sudo mount -o remount,ro /dev/sdb2 /mnt/data
```

Попробуем прочитать файл `/mnt/data/test.txt` и внести в него изменения: `sudo nano /mnt/data/test.txt`. Как мы можем убедиться, теперь даже из-под суперпользователя мы не можем вносить изменения в эту файловую систему. Однако, журнал для ФС ext3 и ext4 может все еще работать. Для отключения возможности записи и в журнал, необходимо добавить при монтировании еще одну опцию **noload**.

Размонтирование устройства

Для размонтирования устройства служит команда `sudo umount <точка_монтирования>` или `sudo umount <устройство>`. Размонтируем устройство /dev/sdb2 `sudo umount /dev/sdb2`.

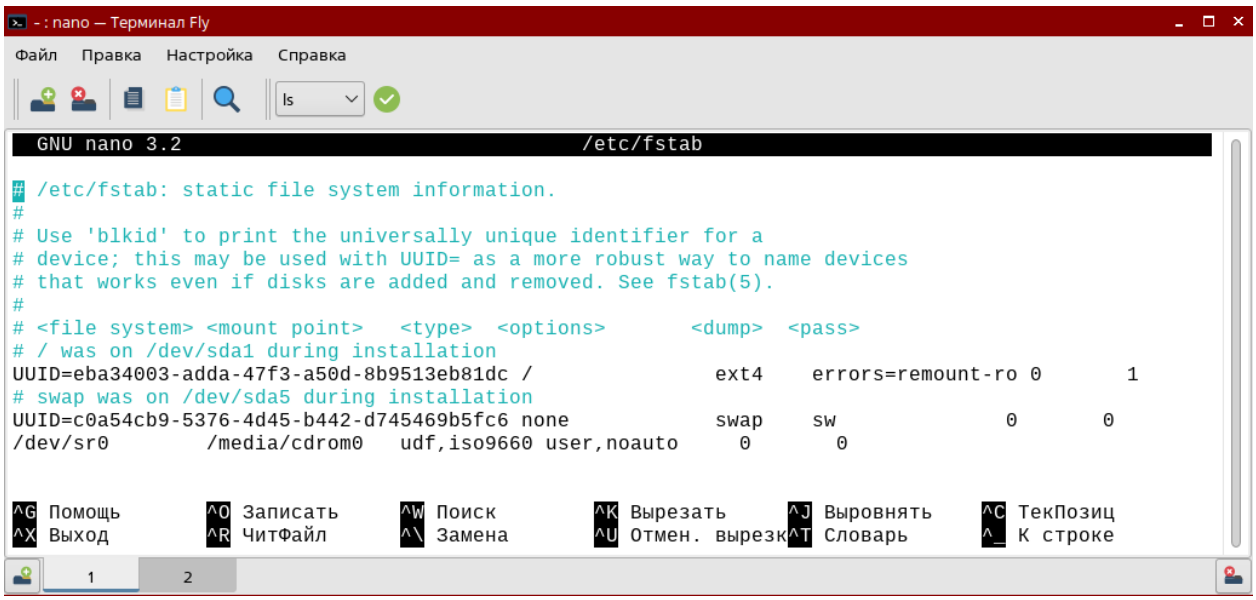
```
sa@astra:~$ sudo umount /dev/sdb2
```

Откроем вкладку, на которой был запущен процесс `findmnt -p`, и ознакомимся с выполненными операциями.

```
sa@astra:~$ findmnt -p
ACTION      TARGET    SOURCE    FSTYPE OPTIONS
монтировать /mnt/data /dev/sdb2 ext4   rw,relatime
перемонтировать /mnt/data /dev/sdb2 ext4   ro,relatime
размонтировать /mnt/data /dev/sdb2 ext4   ro,relatime
```

Постоянное монтирование

Для постоянного монтирования (или долговременного) необходимо вносить изменения в конфигурационный файл `/etc/fstab`. Посмотрим его содержимое:



Формат файла представляет из себя таблицу с полями, разделенными пробелами. Строки, начинающиеся с символа решетки, являются комментариями, а пустые строки – игнорируются.

Всего используется шесть полей:

- **File system (fs_spec)** -– имя, метка (LABEL) или идентификатор (UUID) устройства. Указание метки или идентификатора предпочтительнее, так как порядок обнаружения оборудования может быть изменен, что приведет к изменению имени устройства.
- **Mount point (fs_file)** –- точка монтирования (если нет, то значение none, как например у раздела подкачки). Если точка монтирования содержит пробелы или табуляции, их можно использовать как «\040» и «\011» соответственно.
- **Type (fs_vfstype)** – предполагаемый тип файловой системы. Можно указать несколько значений через запятую.
- **Options (fs_mntops)** -– параметры монтирования.
- **Dump (fs_freq)** –- используется утилитой dump. Определяет, какие файлы системы нужно выгружать при дампе. По умолчанию 0 (не выгружать).
- **Pass (fs_passno)** –- используется утилитой fsck. Определяет порядок проверки файловых систем при вызове утилиты fsck. Для корневой ФС этот параметр должен быть 1. Для остальных ФС равен 2 или 0 (не проверять). Системы на разных дисках будут проверяться параллельно, а на одном диске – последовательно.

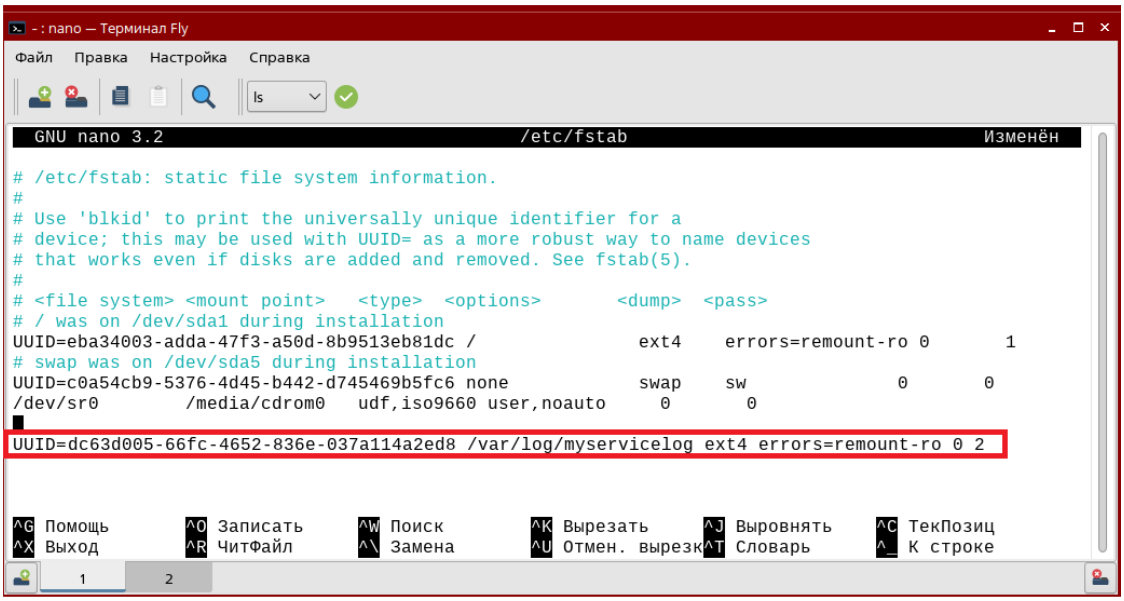
Создадим постоянное монтирование для устройства `/dev/sdb1`. Для этого:

1. Создадим каталог `/var/log/myservicelog`, в который мы будем монтировать это устройство `sudo mkdir /var/log/myservicelog`.

2. Затем нам необходимо узнать UUID блочного устройства `/dev/sdb`. Это можно сделать, используя команду `lsblk -f | grep sdb1`.

```
sa@astra:~$ lsblk -f | grep sdb1
├─sdb1   ext4   Logs  dc63d005-66fc-4652-836e-037a114a2ed8
```

3. Добавим в файл `/etc/fstab` новую строку «*UUID=<UUID устройства> /var/log/myservicelog ext4 errors=remount-ro 0 2*»



4. Сохраним файл `/etc/fstab`.

5. Проверим, может ли система быть смонтирована, с помощью команды `sudo mount -a`, которая монтирует все файловые системы, перечисленные в файле `/etc/fstab`, за исключением тех, у которых задана опция noauto. Проверим, было ли примонтировано устройство `/dev/sdb1` `mount | grep "^/dev"`.

```
sa@astra:~$
sa@astra:~$ mount | grep "^/dev"
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
/dev/sdb1 on /var/log/myservicelog type ext4 (rw,relatime,errors=remount-ro)
```

6. Как видим, устройство успешно смонтировалось. Теперь перезагрузим систему, и убедимся, что и после перезагрузки оно доступно.

Менеджер логических томов LVM

Менеджер логических томов LVM (*англ. Logical Volume Manager*) – это подсистема управления логическими томами, позволяющая использовать разные области одного жесткого диска и/или области с разных жестких дисков как один логический том.

С большой натяжкой аналогом LVM в Windows можно назвать динамические диски, которые тоже позволяют создавать программные RAID массивы и объединять несколько разделов на разных физических носителях в один логический раздел.

Примечание

LVM основан на Device mapper, которая используется также в реализации программных RAID-массивов, системы шифрования дисков dm-crypt и создания снимков файловой системы.

Основные преимущества LVM:

- Одну группу логических томов можно создавать поверх любого количества физических разделов.
- Размер логических томов можно легко менять прямо во время работы.
- LVM поддерживает механизм снимков состояний (*англ. snapshot*), копирование разделов «на лету» и зеркалирование, подобное RAID-1 и чередующуюся запись, подобно RAID-0.

К сожалению, получая преимущества приходиться чем-то жертвовать, и LVM не исключение. Удобство и гибкость при работе с LVM мы разменяваем на следующие недостатки:

- Работа дополнительно «прослойки» и уровня абстракции требует дополнительных накладных расходов, что может снизить производительность дисковой подсистемы (в среднем потери можно оценить в 5-10%)
- При использовании LVM на нескольких дисках, при потере одного из них, все данные хранимые на LVM будут утеряны.
- Начальная настройка LVM более сложна, чем просто разбиение диска. Необходимо понимать терминологию и модель LVM (логические тома, физические тома, группы томов), прежде чем вы сможете начать его использовать.

Структура и состав LVM

На рисунке ниже представлена схема структуры LVM и её взаимодействие с физическими устройствами, разделами и файловыми системами.

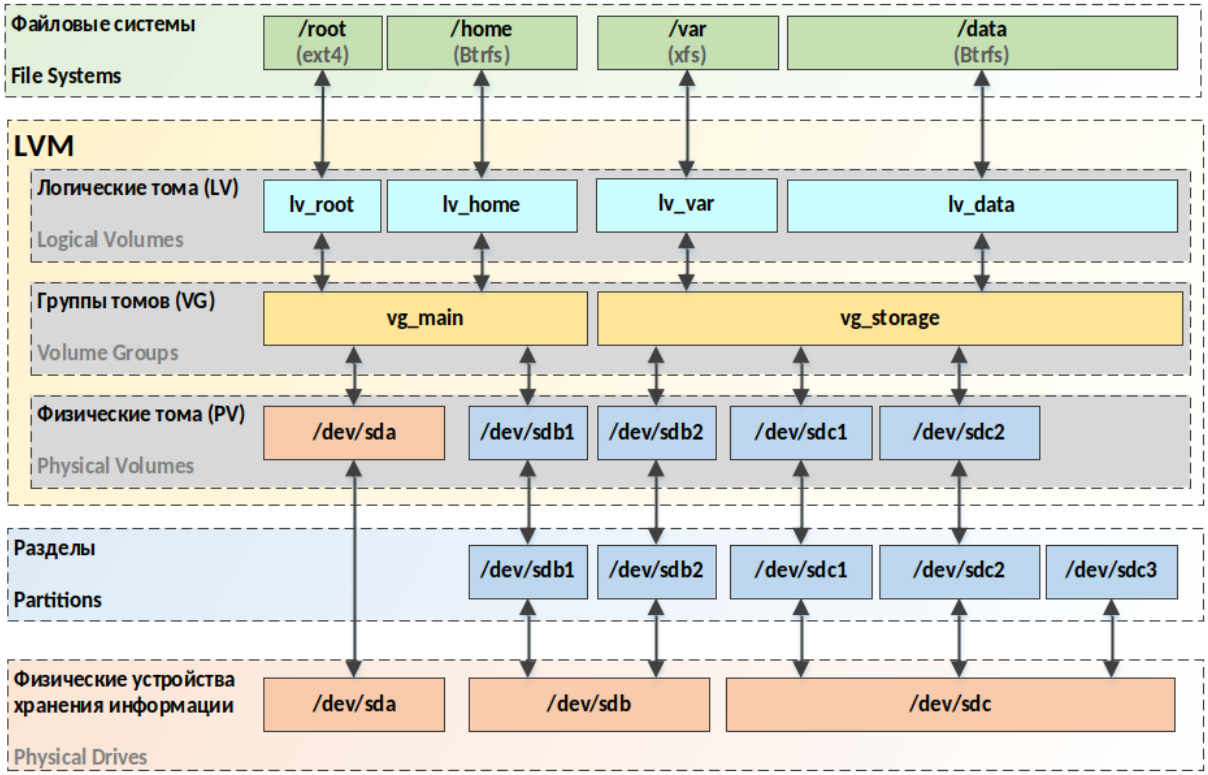


Схема структуры LVM

В состав LVM входят следующие компоненты:

• **Физический том** (*англ. Physical Volume*, PV) –- раздел на диске или весь диск. В том числе, устройства программного и аппаратного RAID (которые уже могут включать в себя несколько физических дисков). Физические тома входят в группу томов.

• **Группа томов** (*англ. Volume Group, VG*) –- это самый верхний уровень абстрактной модели, используемой системой LVM. С одной стороны, группа томов состоит из физических томов, с другой – из логических и представляет собой единую административную единицу.

• **Логический том** (*англ. Logical Volume, LV*) –- раздел группы томов, эквивалентен разделу диска в не-LVM системе. Представляет собой блочное устройство и, как следствие, может содержать файловую систему.

Кроме указанных понятий существуют также логический (LE, Logical extent) и физический (PE, Physical extent) экстенты (порции данных). Физические тома делятся на порции данных, равные по размеру физическим экстентам, а логические тома на порции данных равные по размеру логическим экстентам, размер логического экстента не меняется в пределах группы томов и равен нескольким мегабайтам. При этом существует два алгоритма отображения физических экстентов на логическом:

• **Линейное отображение**, когда физические экстенты последовательно назначаются логическим экстентам.

• **Расслоенное отображение** (*англ. stripped*), когда порции данных логических экстентов разделяют на определенное количество физических томов (похожая схема используется в RAID 0). Это может увеличить производительность работы логического тома, однако логический том с таким отображением не может быть расширен за пределы физических томов, на которых он изначально был создан.

Управление LVM

Для работы с LVM необходимо установить пакет lvm2, который по умолчанию не установлен: `sudo apt install lvm2 -y`. Как вы можете заметить, при этом создаётся новые сервисы и образ Initrd включающий необходимые модули для работы с LVM. В своих экспериментах мы воспользуемся ранее добавленными дисками `/dev/sdc`, `/dev/sdd` и `dev/sde`.

Полный процесс создания LVM и подготовки её к использованию можно представить шагами:

- Создание физических томов.
- Создание групп томов.
- Создание логических томов.
- Создание ФС на логических томах.
- Монтирование ФС.

Мы уже рассмотрели создание ФС и монтирование, сосредоточимся на первых 3 шагах.

Внимание

Стоит отметить, что все изменения с LVM, на логических томах которых созданы файловыми системами должны выполняться с размонтированными ФС.

Управление физическими томами

Создание физических томов

Для создания физического тома PV (инициализации целого диска или его раздела) используем команду:

```
sudo pvcreate <устройство1> [<устройство2> <…> <устройствоN>]
```

Где:

• **устройство1** .. **устройство<N>** — это имя устройства целиком (например, `/dev/sdb`) или его раздела (например, `/dev/sdb1`).

Инициализируем `/dev/sdc` командой `sudo pvcreate /dev/sdc`.

```
sa@astra:~$ sudo pvcreate /dev/sdc
Cannot use /dev/sdc: device is partitioned
```

Однако, мы получим ошибку, так как ранее разметили этот диск под GPT. Мы можем удалить таблицу разделов выполнив следующие команды:

```
sa@astra:~$ sudo dd if=/dev/zero of=/dev/sdc bs=1k count=1
1+0 записей получено
1+0 записей отправлено
1024 байт (1,0 kB, 1,0 KiB) скопирован, 0,020706 s, 49,5 kB/s
sa@astra:~$ sudo blockdev --rereadpt /dev/sdc
```

Повторим попытку `sudo pvcreate /dev/sdc`.

```
sa@astra:~$ sudo pvcreate /dev/sdc
 Physical volume "/dev/sdc" successfully created.
```

Теперь создадим два раздела на устройстве `/dev/sdd` по 500MB каждый и инициализируем их.

```
sa@astra:~$ sudo parted -s /dev/sdd mklabel gpt mkpart lvm1 ext4 1MiB 500MiB mkpart lvm2 ext4
501MiB 1000MiB
sa@astra:~$ sudo pvcreate /dev/sdd1 /dev/sdd2
Physical volume "/dev/sdd1" successfully created.
Physical volume "/dev/sdd2" successfully created.
```

Вывод информации о физических томах

Для вывода информации о физических томах существуют следующие команды:

• Команда `pvs` –- позволяет настроить формат вывода, показывая по одному тому в каждой строке.

• Команда `pvdisplay` –- формирует подробный отчет для каждого физического тома, включая информацию о размере, экстентах, группе томов и пр. Формат вывода фиксирован.

• Команда `pvscan` –- проверяет все поддерживаемые блочные устройства в системе на предмет наличия физических томов.

```
sa@astra:~$ sudo pvs
PV         VG Fmt  Attr PSize   PFree
/dev/sdc      lvm2 ---    1,00g   1,00g
/dev/sdd1     lvm2 ---  499,00m 499,00m
/dev/sdd2     lvm2 ---  499,00m 499,00m

sa@astra:~$ sudo pvscan
PV /dev/sdc                      lvm2 [1,00 GiB]
PV /dev/sdd1                     lvm2 [499,00 MiB]
PV /dev/sdd2                     lvm2 [499,00 MiB]
Total: 3 [1,97 GiB] / in use: 0 [0 ] / in no VG: 3 [1,97 GiB]

sa@astra:~$ sudo pvdisplay
```

```
"/dev/sdc" is a new physical volume of "1,00 GiB"
 --- NEW Physical volume ---
 PV Name               /dev/sdc
 VG Name
 PV Size               1,00 GiB
 Allocatable           NO
 PE Size               0
 Total PE              0
 Free PE               0
 Allocated PE          0
 PV UUID               3grEfU-7HGT-vrlc-2GJe-TsbU-W2xj-AY897m

 "/dev/sdd1" is a new physical volume of "499,00 MiB"
 --- NEW Physical volume ---
 PV Name               /dev/sdd1
 VG Name
 PV Size               499,00 MiB
 Allocatable           NO
 PE Size               0
 Total PE              0
 Free PE               0
 Allocated PE          0
 PV UUID               b4YhpL-W8Pc-0IZH-zIAf-ZER0-XY0R-8KsNtj

 "/dev/sdd2" is a new physical volume of "499,00 MiB"
 --- NEW Physical volume ---
 PV Name               /dev/sdd2
 VG Name
 PV Size               499,00 MiB
 Allocatable           NO
 PE Size               0
 Total PE              0
 Free PE               0
 Allocated PE          0
 PV UUID               0IjxeV-EUZR-cYeF-yjjW-SYaB-wqxQ-drKKF2
```

Удаление физических томов

Для удаления физических томов используется команда `pvremove <имя_PV>`. Удалим PV `/dev/sdd2` `sudo pvremove /dev/sdd2`.

```
sa@astra:~$ sudo pvremove /dev/sdd2
 Labels on physical volume "/dev/sdd2" successfully wiped.
```

Управление группами томов

После создания физических томов мы можем создать группу томов. Соглашение об именовании предписывает начинать имя группы с префикса «vg_» и использовать либо порядковый номер, либо (что лучше) значащее имя, отражающее назначение этой группы.

Создание группы томов

Для создания группы томов служит команда:

```
sudo vgcreate <имя_группы> <имя_физического_тома1> [<имя_физического_тома2> <имя_физического\_
...> <имя_физического_N>]
```

Создадим группу томов «myservice» из томов `/dev/sdc` и `/dev/sdd1`.

```
sa@astra:~$ sudo vgcreate vg_myservice /dev/sdc /dev/sdd1
 Volume group "vg_myservice" successfully created
```

Вывод информации о группах томов

Для вывода информации о группах томов служат команды, аналогичные тем, что используются для вывода информации о физических томах:

• Команда `vgs` –- позволяет настроить формат вывода, показывая по одной группе в каждой строке.

• Команда `vgdisplay` -– формирует подробный отчет для каждой группы томов, включая информацию о размере, количестве физических и логических томов и пр. Формат вывода фиксирован.

• Команда `vgscan` –- проверяет все поддерживаемые дисковые устройства в системе на предмет наличия физических томов и групп томов.

```
sa@astra:~$ sudo vgs
VG            #PV #LV #SN Attr     VSize  VFree
vg_myservice    2   0   0 wz--n-   1,48g  1,48g

sa@astra:~$ sudo vgdisplay
--- Volume group ---
VG Name               vg_myservice
System ID
Format                lvm2
Metadata Areas        2
Metadata Sequence No  1
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                0
Open LV               0
Max PV                0
Cur PV                2
Act PV                2
VG Size               1,48 GiB
PE Size               4,00 MiB
Total PE              379
Alloc PE / Size       0 / 0
Free PE / Size        379 / 1,48 GiB
VG UUID               EqrAT9-lMpW-99qd-LkUJ-7NfP-CkHP-OdbKdR

sa@astra:~$ sudo vgscan
Found volume group "vg_myservice" using metadata type lvm2
```

Изменение группы томов

Добавление томов в группу

Для добавления нового тома в группу томов используется команда:

```
sudo vgextend <имя_группы> <имя_тома>
```

Создадим раздел на устройстве `/dev/sde` и добавим его в группу томов «vg_myservice». При этом нам необязательно создавать физический том на устройстве, он будет создан автоматически.

```
sa@astra:~$ sudo parted -s /dev/sde mklabel gpt mkpart lvm1 ext4 1MiB 500MiB
sa@astra:~$ sudo vgextend vg_myservice /dev/sde1
 Physical volume "/dev/sde1" successfully created.
 Volume group "vg_myservice" successfully extended
```

Создадим еще один раздел и добавим его в группу.

```
sa@astra:~$ sudo parted /dev/sde mkpart lvm2 ext4 501MiB 1000MiB
Information: You may need to update /etc/fstab.

sa@astra:~$ sudo vgextend vg_myservice /dev/sde2
 Physical volume "/dev/sde2" successfully created.
  Volume group "vg_myservice" successfully extended
```

Удаление томов из группы

Для удаления тома из группы воспользуемся командой:

```
sudo vgreduce <имя_группы> <имя_тома>
```

Удалим из группы том `/dev/sde2` командой:

```
sa@astra:~$ sudo vgreduce vg_myservice /dev/sde2
 Removed "/dev/sde2" from volume group "vg_myservice
```

Переименование группы

Для переименования группы воспользуемся командой:

```
vgrename <текущее_имя> <новое_имя>
```

Переименуем группу в «vg_test»:

```
sa@astra:~$ sudo vgrename vg_myservice vg_test
 Volume group "vg_myservice" successfully renamed to "vg_test"
```

Удаление группы томов

Для удаления группы томов служит команда `sudo vgremove <имя_группы>`. Не будем удалять группу, она нам еще потребуется для создания логических томов.

Управление логическими томами

Создание логического тома

Для создания логического тома используется команда:

```
sudo lvcreate [<опции>] <размер_тома> -n <имя_тома> <имя_группы>
```

Где размер тома может быть задан несколькими способами:

- `-L<число>` - размер в мегабайтах;
- `-L<число><единицы_объема>` - размер в указанных единицах объема;
- `-l <число>` - размер в логических экстентах;
- `-l100%FREE` – весь объем группы.

Команда `sudo lvcreate -i2 -I4 -l100 -n <имя_тома> <имя_группы>` создаст логический том размером в 100 логических экстентов с расслоением по двум физическим томам и размером блока 4 KB.

Выведем информацию о количестве физических экстентов в созданной нами группе томов:

```
sa@astra:~$ sudo vgdisplay vg_test | grep "Total PE"
 Total PE 503
```

И создадим логический том «vl_test» размером в 300 экстентов:

```
sa@astra:~$ sudo lvcreate -l 300 -n vl_test vg_test
 Logical volume "vl_test" created.
```

Вывод информации о логических томах

И в этом случае команды аналогичны тем, что использовались для физических томов и групп:

- Команда `lvs` – позволяет настроить формат вывода, показывая по одному тому в каждой строке.
- Команда `lvdisplay` – формирует подробный отчет для каждой группы томов, включая информацию о размере, количестве физических и логических томов и пр. Формат вывода фиксирован.
- Команда `lvscan` – проверяет все поддерживаемые дисковые устройства в системе на предмет наличия физических томов, групп томов и логических томов.

```
sa@astra:~$ sudo lvs
 LV      VG      Attr       LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert
 vl_test vg_test -wi-a----- 1,17g
```

```
sa@astra:~$ sudo lvscan
ACTIVE             '/dev/vg_test/vl_test' [1,17 GiB] inherit

sa@astra:~$ sudo lvdisplay
--- Logical volume ---
LV Path                /dev/vg_test/vl_test
LV Name                vl_test
VG Name                vg_test
LV UUID                rphX2q-bhxW-du2T-RExH-cAqV-CUdf-iaWs4U
LV Write Access        read/write
LV Creation host, time  astra, 2025-04-27 13:53:14 +0300
LV Status              available
# open                 0
LV Size                1,17 GiB
Current LE             300
Segments               2
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           253:0
```

Изменение логического тома

Переименование тома

Допустим, мы заметили, что допустили ошибку в имени тома. Переименуем том «vl_test» в «lv_test» с помощью команды:

```
sa@astra:~$ sudo lvrename /dev/vg_test/vl_test lv_test
 Renamed "vl_test" to "lv_test" in volume group "vg_test"
```

Изменение размера тома

Для увеличения логического тома служит команда:

```
sudo lvextend <размер>|<на_сколько_увеличить_размер> <полное_имя_логического_тома>
```

Размер задается как при создании тома (опции -l и -L), а для указания того, насколько требуется увеличить размер, используется выражение типа -L+1G или -l +50.

Увеличим размер логического тома на 25 экстентов:

```
sa@astra:~$ sudo lvextend -l +25 /dev/vg_test/lv_test
 Size of logical volume vg_test/lv_test changed from 1,17 GiB (300 extents) to <1,27 GiB
(325 extents).
 Logical volume vg_test/lv_test successfully resized.
```

Если использовать опцию `-r`, то наряду с увеличением размера тома его файловая система тоже будет автоматически расширена:

```
sudo lvextend –r -l +25 /dev/vg_test/lv_test.
```

Для уменьшения размера тома служит команда `lvreduce`, синтаксис которой подобен lvextend:

```
sudo lvreduce <размер>|<на_сколько_уменьшить_размер> <полное_имя_логического_тома>
```

Кроме lvextend и lvreduce есть команда `lvresize`, которая может использоваться как для увеличения, так и для уменьшения размера логического тома.

Удаление логического тома

Для удаления логического тома служит команда `lvremove`:

```
sudo lvremove <полное_имя_логического_тома>
```.

Типовой сценарий переноса данных на другой носитель

Описание сценария

При правильном планировании на этапе установки ОС риск столкнуться с потребностью переноса данных на другой носитель информации уменьшаются. Однако в процессе эксплуатации машины сценарий её использования может измениться, как может измениться и потребление ресурсов, в частности дискового пространства.

Рассмотрим типовой сценарий, когда на этапе установки ОС корневая ФС была установлена в обычный раздел, а не LVM, а место на диске заканчивается. В ходе анализа мы выяснили, что основным потребителем места является каталог `/home`. Мы решили перенести его на другой носитель информации. При разметке диска будем использовать LVM для повышения гибкости в управлении, справедливо полагая, что в будущем нам возможно придётся еще расширять доступное пространство для точки монтирования `/home`.

Примечание

В сценарии, для простоты, опустим некоторые детали переноса данных в продуктивной среде. В частности, не будем проверять активные сессии или выставлять запрет на изменение данных в каталоге `/home` после их копирования на новый раздел и его монтирования.

Перенос данных на внешнюю ФС

На нашей машине осталось свободное место на дисках sdd и sde, воспользуемся ими. Перенесем данные на внешнюю ФС.

1. Создадим логический том lv_home на разделах `/dev/sdd2` и `/dev/sde2` и создадим на нем файловую систему ext4:

• Используем раздел `/dev/sdd2` в качестве физического тома LVM:

```
sa@astra:~$ sudo pvcreate /dev/sdd2
Physical volume "/dev/sdd2" successfully created.
```

• Создадим на физических томах `/dev/sdd2` и `/dev/sde2` группу томов LVM vg_home:

```
sa@astra:~$ sudo vgcreate vg_home /dev/sdd2 /dev/sde2
Volume group "vg_home" successfully created
```

• Создадим логический том lv_home в группе vg_home используя весь объем:

```
sa@astra:~$ sudo lvcreate --extents 100%FREE -n lv_home vg_home
Logical volume "vl_home" created.
```

• Создадим файловую систему ext4 на логическом томе lv_home:

```
sa@astra:~$ sudo mkfs.ext4 /dev/vg_home/lv_home
mke2fs 1.44.5 (15-Dec-2018)
Creating filesystem with 253952 4k blocks and 63488 inodes
Filesystem UUID: 4b458903-5c41-4a9b-a157-baa1db3a452a
Superblock backups stored on blocks:
   32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

2. Удалим содержимое каталога `/home/sa/Загрузки` и скопируем содержимое каталога `/home` на логический том lv_home:

• Удалим содержимое каталога `/home/sa/Загрузки`, чтобы объем данных в `/home` не превышал 1 ГБ:

```
sa@astra:~$ sudo rm -rf /home/sa/Загрузки/*``
```

• Выполним временное монтирование ФС с устройства `/dev/mapper/vg_home-lv_home` в каталог `/mnt/home`:

```
sa@astra:~$ sudo mkdir /mnt/home
sa@astra:~$ sudo mount /dev/mapper/vg_home-lv_home /mnt/home
```

• Скопируем содержимое каталога `/home` в `/mnt/home` с сохранением прав доступа. Для этого воспользуемся утилитой `cp` с ключом `-a`, предварительно установив её:

```
sa@astra:~$ sudo cp -a /home /mnt/home
sa@astra:~$ ls -l /mnt/home/
итого 20
drwxr-xr-x 5 root root  4096 мар 22  2023 home
drwx------ 2 root root 16384 апр 27 14:00 lost+found
```

3. Выполним временное монтирование ФС `/dev/mapper/vg_home-lv_home` в `/home`, а затем создадим постоянное монтирование:

• Выполним временное монтирование `/dev/mapper/vg_home-lv_home` в `/home`:

```
sa@astra:~$ sudo mount /dev/mapper/vg_home-lv_home /home
sa@astra:~$ ls -l /home
итого 20
drwxr-xr-x 5 root root  4096 мар 22  2023 home
drwx------ 2 root root 16384 апр 27 14:00 lost+found
```

• Создадим постоянное монтирование используя файл `/etc/fstab`, добавив в него строку: `/dev/mapper/vg_home-lv_home /home ext4 defaults 0 2`

```
sa@astra:~$ echo "/dev/mapper/vg_home-lv_home /home ext4 defaults 0 2" | sudo tee -a
/etc/fstab
/dev/mapper/vg_home-lv_home /home ext4 defaults 0 2
sa@astra:~$ tail -n 1 /etc/fstab
/dev/mapper/vg_home-lv_home /home ext4 defaults 0 2
```

• Удостоверимся, что все работает:

```
sa@astra:~$ ls -l /home
итого 20
drwxr-xr-x 5 root root  4096 мар 22  2023 home
drwx------ 2 root root 16384 апр 27 18:32 lost+found
```

Примечание

После монтирования внешней ФС в корневую ФС все новые данные записанные в эту точку монтирования будут фактически храниться на внешней ФС. Однако, такой операцией мы не освободим место на диске корневой ФС. После успешного переноса данных на внешнюю ФС и проверки корректности монтирования, исходные данные могут быть удалены. Чтобы удалить данные, например, из каталога `/home` на устройстве `/dev/sda1`, необходимо предварительно примонтировать это устройство в корневую ФС.

Добавление места логическому тому и расширение ФС

Если возникает необходимость расширить пространство у логического тома lv_home, то для этого нам необходимо вначале добавить новый физический том LVM и расширить группу томов vg_home:

1. Добавим нашей виртуальной машине еще один диск размером 1 ГБ:

```
sa@astra:~$ sudo lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                   8:0    0   20G  0 disk
├─sda1                8:1    0   19G  0 part /
. . .
└─sde2                8:66   0  499M  0 part
└─vg_home-lv_home 253:1    0  992M  0 lvm  /home
sdf                   8:80   0    1G  0 disk
sr0                  11:0    1 1024M  0 rom
```

2. Добавим новый физический том sdf в группу томов LVM vg_home:

```
sa@astra:~$ sudo vgextend vg_home /dev/sdf
```

```
Physical volume "/dev/sdf" successfully created.
Volume group "vg_home" successfully extended
```

3. Проверим текущий размер логического тома lv_home:

```
sa@astra:~$ sudo lvscan | grep lv_home
ACTIVE            '/dev/vg_home/lv_home' [992,00 MiB] inherit
```

4. Расширим логический том lv_home на все доступное пространство группы томов lv_home:

```
sa@astra:~$ sudo lvextend -l +100%FREE /dev/vg_home/lv_home
Size of logical volume vg_home/lv_home changed from 992,00 MiB (248 extents) to 1,96 GiB (503 extents).
Logical volume vg_home/lv_home successfully resized.
```

Как видим, логический том был расширен, и пользователи могут продолжать работать и сохранять свои данные в домашних директориях.

Практические задания

Задание 1.

- Добавьте к существующей виртуальной машине новый жесткий диск размером 500 Мб;
- Установите метку диска msdos утилитой parted в режиме командной строки;
- Создайте на подключенном жестком диске 2 раздела по 100 Мб утилитой parted в режиме командной строки;
- Создайте на новых разделах файловые системы EXT3 и XFS и смонтируйте их в каталоги /mnt/part1 и /mnt/part2;
- Создайте постоянное монтирование для указанных разделов со следующими условиями:

- /mnt/part1 монтируется в режиме только для чтения;
- /mnt/part2 монтируется с запретом на запуск исполняемых файлов;
- Перезагрузите ОС для проверки созданных настроек;
- После проверки настроек удалите созданные точки монтирования и разделы;

Задание 2.

- На диске /dev/sdb создайте раздел размером 150Мб;
- Создайте LVM physical volume (PV) на созданном разделе;
- Создайте LVM volume group (VG) с именем TEST_VG на созданном PV;
- Создайте LVM logical volume (LV) с именем TEST_LV на созданном VG и занимающий весь доступный объем;
- Создайте файловую систему EXT4 на созданном LV;
- На диске /dev/sdb создайте еще один раздел размером 150Мб;

• Выполните процедуру расширения имеющегося LVM раздел на размер /dev/sdb2 с расширением файловой системы;

• Проверьте что LV и файловая система были расширены и удалите созданные разделы.

Вопросы

1. Какая файловая система используется при подключении usb-накопителей для монтирования их без привилегий?

2. Как получить список файловых систем, поддерживаемых ядром в данный момент?

3. В каком каталоге хранятся программные файлы, доступные только для чтения, такие как исполняемые файлы, библиотеки и прочее?

4. Какой командой можно новый создать раздел диска в утилите fdisk?

5. За что отвечает опция noexec в файле fstab?

6. Расположите в правильном порядке шаги создания дисковых разделов LVM.

a. Создание логических томов
b. Создание физических томов
c. Монтирование ФС
d. Создание групп томов
e. Создание ФС на логических томах

7. При расширении раздела LVM, после выполнения расширения логического раздела, необходимо выполнить расширение файловой системы на этом разделе отдельной командой. Можно ли одной командой расширить и раздел и ФС?