

ИУ-10

Системное

Программное

Обеспечение

Системы виртуализации

**Работа с дисковыми
пространствами. Разделы,
LVM, точки монтирования.**

На этом уроке

1. Разберем структуру файловой системы
2. Узнаем, что такое точка монтирования
3. Познакомимся с понятием раздела, научимся их создавать
4. Познакомимся с LVM
5. Выясним, для чего используются ссылки и inode

Содержание

На этом уроке	1
Структура файловой системы	3
Точка монтирования	4
Организация хранения	6
Структура диска	6
MBR vs GPT	7
MBR	8
GPT	8
Создание и подключение разделов	9
Создание разделов с помощью parted	9
Создание файловой системы	11
XFS	11
Ext4	11
Монтирование файловой системы	13
Автоматическое монтирование	14
Расширенные возможности хранения данных	16
Работа с LVM	17
Создание логических томов	17
Увеличение логических томов	20
Мониторинг доступного места	22
Ссылки и файловые дескрипторы	23
Практическое задание	25
Глоссарий	25

Дополнительные материалы	26
Используемые источники	26

Структура файловой системы

В Linux, так же как и в Windows, физическое хранение данных на дисках организуется системой при помощи файловой системы. А за то, как логически располагаются директории и за их количество отвечает стандарт иерархии файловой системы (Filesystem Hierarchy Standard). Такую же конструкцию вы могли видеть в Windows, например, если мы зайдём в директорию диска C: то мы, как минимум увидим там папки Windows, Program Files, Users и так далее.

Дерево каталогов можно посмотреть, используя утилиту **tree**. Она не идёт в стандартной поставке, поэтому её необходимо установить, используя команду `sudo apt install tree -y`. Утилита показывает содержимое каталогов в виде дерева. Корневой каталог обозначается точкой. Для удобства вывода используется параметр **-d** — показывать только каталоги.

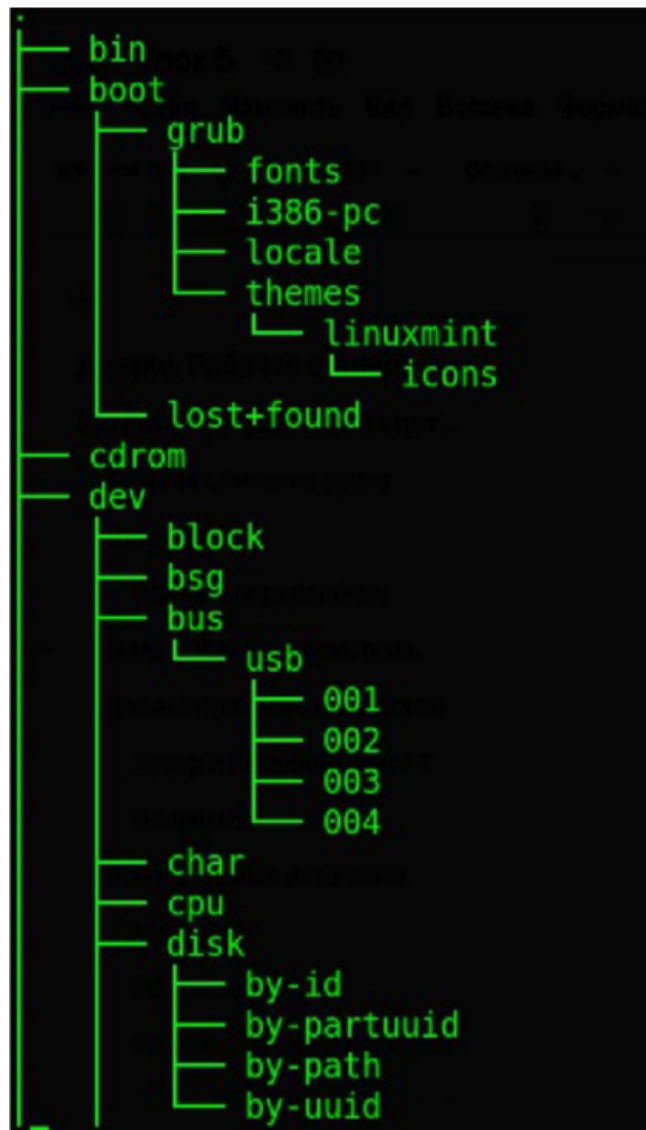


Рис.1 Структура файловой системы

Вложенные каталоги могут быть как самостоятельными единицами файловой системы, хранящими в себе файлы и другие каталоги, так и точками монтирования для разделов. Разделы либо организуются на части жёсткого диска, либо занимают всё пространство жесткого диска. На каждом разделе создаётся собственная файловая система.

Точка монтирования

В Windows при подключении к ПК правильно отформатированного flash-накопителя или диска автоматически появляется директория (диск D/E/F), внутри которой будет находиться все содержимое подключенного устройства. Во всех ОС подобная операция называется монтированием, но в Linux у нее есть свои особенности:

- Директория автоматически создана не будет. Необходимо вручную создать папку, в которой впоследствии будет находиться все содержимое диска или раздела диска. Эта созданная папка называется точкой монтирования.
- Для доступа к физически подключенному устройству понадобится выполнить команду `mount`. В Windows, к примеру, вручную мы только отключаем флэшку для дальнейшего извлечения из ПК.

В результате выполненных действий для конечного пользователя всё будет представляться как единое пространство данных.

```
root@server:~# lsblk | grep -v loop
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                                  8:0      0   10G  0 disk
├─sda1                              8:1      0    1M  0 part
├─sda2                              8:2      0    1G  0 part /boot
└─sda3                              8:3      0    9G  0 part
   └─ubuntu--vg-ubuntu--lv 253:0      0    9G  0 lvm  /
```

Рис.2 Блочные устройства в Linux

В файловой системе Linux обычно используется несколько точек монтирования (MOUNTPOINT). С помощью команды `lsblk` можно отобразить все блочные устройства (например, жесткие диски), используемые в системе. Из рисунка 2 следует, что корневая файловая система находится на одном устройстве и подключена в точке монтирования `/`, а в точке `/boot` подключено совсем другое устройство. Схожим образом может осуществляться монтирование и других устройств, в последствии интегрированных в файловую систему.

Чаще всего по разным дискам распределены разные типы данных. Это происходит по следующим причинам:

- **Безопасность**

Для подключаемого устройства по необходимости могут быть применены более или менее строгие правила безопасности в зависимости от того, что находится в данном разделе.

- **Управление**

К точке монтирования могут применяться определенные параметры монтирования, которые необходимы в определенных случаях и не нужны в других.

Рассмотрим назначение основных каталогов корневой файловой системы. Обратим внимание, что многие из них на самом деле являются точками монтирования:

1. **/boot** — каталог, который хранит в себе конфигурационные файлы загрузчика ОС, образы ядра и файлы `initrd`.
2. **/bin** и **/sbin** — в данных каталогах хранятся основные исполняемые файлы и утилиты, необходимые для работы и администрирования операционной системы.
3. **/etc** — каталог предназначен для хранения конфигурационных файлов операционной системы и всех служб.
4. **/home** — стандартный каталог для хранения личных файлов и каталогов пользователей.
5. **/dev** — каталог, в котором хранятся файлы устройств (оборудование ПК, которое подключается при старте ядра ОС).
6. **/proc** — точка монтирования для виртуальной файловой системы `procfs`, которая используется для хранения и предоставления информации о процессах.
7. **/sys** — точка монтирования для виртуальной файловой системы `sysfs`, которая используется для хранения и предоставления информации об инициализированных устройствах, сгруппированных по разным критериям: типам устройств, шинам, диагностическим протоколам доступа и т. п. Одни и те же устройства могут быть показаны в разных каталогах через особый тип файлов, который называется символические ссылки, о них поговорим ниже.

8. **/usr** — предназначен для хранения пользовательских программ, документации, исходных кодов программ и ядра.
9. **/var** — каталог содержит в себе постоянно изменяемые файлы, например журналы работы операционной системы.
10. **/lib** — каталог, который хранит в себе библиотеки и модули ядра, необходимые для функционирования ОС.

Команда `findmnt` отобразит все точки монтирования, существующие в системе.

TARGET	SOURCE	FSTYPE	OPTIONS
/	/dev/mapper/ubuntu--vg-ubuntu--lv	ext4	rw,relatime
├─/sys	sysfs	sysfs	rw,nosuid,nodev,noexec,relatime
│ └─/sys/kernel/security	securityfs	securityfs	rw,nosuid,nodev,noexec,relatime
│ └─/sys/kernel/debug	debugfs	debugfs	rw,nosuid,nodev,noexec,relatime
│ └─/sys/kernel/tracing	tracefs	tracefs	rw,nosuid,nodev,noexec,relatime
│ └─/sys/kernel/config	configfs	configfs	rw,nosuid,nodev,noexec,relatime
├─/proc	proc	proc	rw,nosuid,nodev,noexec,relatime
├─/dev	udev	devtmpfs	rw,nosuid,noexec,relatime,size=9
│ └─/dev/pts	devpts	devpts	rw,nosuid,noexec,relatime,gid=5,
│ └─/dev/shm	tmpfs	tmpfs	rw,nosuid,nodev
│ └─/dev/hugepages	hugetlbfs	hugetlbfs	rw,relatime,pagesize=2M
│ └─/dev/mqueue	mqueue	mqueue	rw,nosuid,nodev,noexec,relatime
├─/run	tmpfs	tmpfs	rw,nosuid,nodev,noexec,relatime,
│ └─/run/lock	tmpfs	tmpfs	rw,nosuid,nodev,noexec,relatime,
│ └─/run/snapd/ns	tmpfs[/snapd/ns]	tmpfs	rw,nosuid,nodev,noexec,relatime,
│ └─└─/run/snapd/ns/lxd.mnt	nsfs[mnt:[4026532252]]	nsfs	rw
└─/run/user/1000	tmpfs	tmpfs	rw,nosuid,nodev,relatime,size=20

Рис. 3 Точки монтирования

Организация хранения

Структура диска

В большинстве случаев к файловой системе подключается не целый диск, а лишь его часть, называемая разделом или партицией (partition). Основной целью разбиения диска на разделы является изоляции данных.

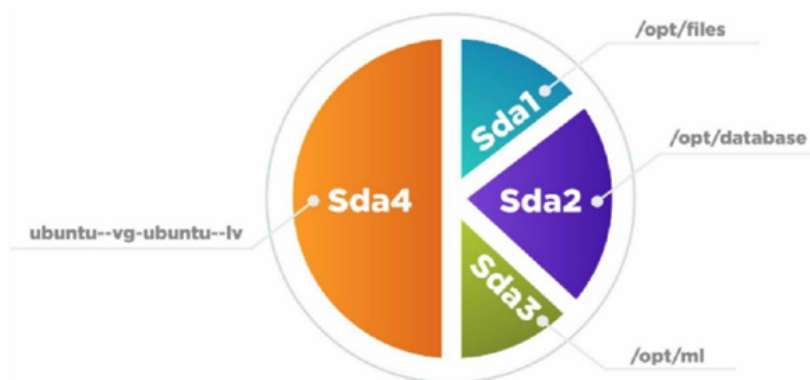
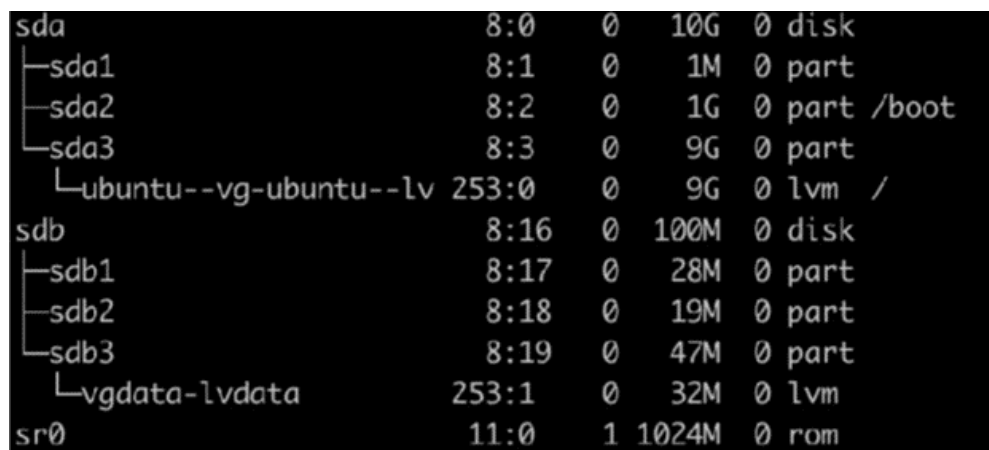


Рис. 4 Разделы диска

На рисунке 4 изображено разделение диска на 4 партии определенного размера. Для разделов с первого по третий указанные каталоги будут использоваться в качестве точек монтирования. Таким образом количество данных внутри этих каталогов не сможет превысить размера партии, а значит и влиять на другие данные в системе.



```

sda                8:0    0   10G    0 disk
├─sda1             8:1    0    1M    0 part
├─sda2             8:2    0    1G    0 part /boot
├─sda3             8:3    0    9G    0 part
└─┬ubuntu--vg-ubuntu--lv 253:0    0    9G    0 lvm /
sdb                8:16   0  100M    0 disk
├─sdb1             8:17   0   28M    0 part
├─sdb2             8:18   0   19M    0 part
├─sdb3             8:19   0   47M    0 part
└─┬vgdata-lvdata    253:1    0   32M    0 lvm
sr0               11:0    1 1024M    0 rom

```

Рис.5 Имена разделов

Возвращаясь к выводу `lsblk`, обратим внимание на тот факт, что диск `sda` разделен на несколько разделов (`sda1,sda2,sda3`). У каждого диска есть имя, которое зависит от типа используемого устройства. `sda` - это имя, которое применяется для первого SCSI-диска в системе, `sdb` - для второго и так далее. В зависимости от имени диска разделы будут названы соответственно.

MBR vs GPT

При разбиении диска на разделы возникает два основных вопроса:

- Какое количество партий может существовать?
- Какого размера может достигать раздел?

В ответе на оба вопроса важную роль играет то, как операционная система загружается. Первым этапом старта любой ОС, не только Linux, является диагностика оборудования и запуск загрузчика системы. Загрузчиком может выступать либо BIOS (Basic Input/Output System), либо UEFI (The Unified Extensible Firmware Interface). BIOS основан на оригинальной спецификации ПК 1981 года и был изобретен для работы с системами того времени. В те дни, если у компьютера был жесткий диск на пять мегабайт, он считался большим компьютером. BIOS не был рассчитан на размеры современных дисков, которые продолжают расти. Вследствие чего в 2010 году появился UEFI. В зависимости от того, какая система используется

для загрузки, используются и разные способы адресации дискового пространства. Системы BIOS обычно работают с MBR - Master Boot Record, UEFI обычно поставляются с GPT - GUID Partition Table.

MBR

- Общий размер в 512 байт, что очень мало
- 64 байта используется для хранения информации о разделах
- Ограничение в 4 раздела размером не более 2 Терабайт
- Логические партии с ограниченной функциональностью могут быть использованы для дополнительного разбиения, если не хватает 4

GPT

- Больше места для хранения информации о разделах
- Используется для преодоления ограничений MBR
- 128 разделов, нет необходимости в логических партициях

Создание и подключение разделов

Для создания разделов существуют несколько утилит.

- `fdisk` создаст разделы с помощью MBR
- `gdisk, parted` создадут разделы с помощью GPT

Мы будем пользоваться `parted` как наиболее современной утилитой для создания GUID разделов.

Создание разделов с помощью parted

Первым шагом необходимо выбрать, на каком диске будут создаваться разделы.

```
root@server:~# parted /dev/sdb
GNU Parted 3.3
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
```

Просмотрим и по необходимости создадим таблицу разделов. Следует использовать GPT.

```
(parted) print
Error: /dev/sdb: unrecognised disk label
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
(parted) mklabel gpt
(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
--------	-------	-----	------	-------------	------	-------

При наличии таблицы можно переходить к созданию самих разделов с помощью команды `mkpart`.

```
(parted) mkpart
Partition name? []? database
File system type? [ext2]?
Start? 0
End? 100

Warning: The resulting partition is not properly aligned for best
performance: 34s % 2048s != 0s

Ignore/Cancel?
Ignore/Cancel? Ignore

(parted)
(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      Flag
  1      17.4kB    100MB     100MB     ext2          databas
```

Важно! *Start/End* - начало/окончание раздела отсчитываются от начала диска и по умолчанию считается в Мегабайтах. При создании второго раздела сразу после приведенного в пример *Start* необходимо указывать - 100, *End* - 100 + необходимый размер партиции.

После создания всех необходимых разделов выходим из утилиты и выполняем `udevadm settle`, чтобы убедиться, что устройство было создано.

```
(parted)quit

root@server:~# udevadm settle

root@server:~# lsblk

sdb                8:16    0      1G    0    disk
└─sdb              8:17    0   95.4M    0    part
```

При создании раздела вы не создаете файловую систему, как это часто делают другие операционные системы. В `parted` есть атрибут файловой системы, но этот атрибут можно игнорировать, так как он записывает только некоторые незначительные метаданные файловой системы.

Создание файловой системы

После создания раздела нужно установить файловую систему поверх него. Файловая система определяет метод, который используется для записи данных на диск, организации этих данных в блоки, а также выполнения всех остальных действий, связанных с хранением информации на диске. Существуют два основных типа файловых систем: XFS и ext4.

XFS

- Быстрая и масштабируемая
- Использует технологию записи CoW (Copy on Write) для гарантии целостности файлов. Перед записью изменения на диск исходный файл копируется в другое место для возможности возврата к предыдущему состоянию файла.
- Существует возможность увеличения размеров файловой системы, но не уменьшения

Ext4

- Для гарантии целостности файлов используется журналирование. При любой записи в файл журнал отслеживает изменения, и, если что-то пойдет не так, на основе информации в журнале легко вернуться к предыдущему состоянию файла
- Существует возможность как увеличения так и уменьшения размеров файловой системы
- Обратно совместима с устаревшей ext2

- Архитектура файловой системы основана на подходе к хранению данных, который более не используется

Существуют и другие файловые системы (btrfs,zfs), но они менее распространены.

Примечание! Так как для каждого раздела создается своя файловая система, то для нескольких разделов можно использовать разные файловые системы. На одном может xfs, на втором - ext4, на третьем - zfs и тд. Все эти файловые системы будут нормально сосуществовать внутри одной ОС.

Создание файловых систем в других операционных системах известно как форматирование раздела или диска. В Linux предпочтительна терминология именно создания, так как для этого используется утилита mkfs. Для xfs используется команда mkfs.xfs и mkfs.ext4 - для ext4 соответственно.

```
root@server:~# mkfs.xfs /dev/sdb1
meta-data=/dev/sdb1              agcount=4, agsize=6103 blks
        =                        sectsz=512   attr=2,
        =                        crc=          finobt=1, sparse=1,
        =                        reflink=
data      =                        bsize=4096   blocks=24409,
        =                        sunit=        swidth=0 blks
naming    =version 2              bsize=4096   ascii-ci=0,
log        =internal log          bsize=4096   blocks=1368,
        =                        sectsz=512   sunit=0 blks, lazy-
realtime   =                      extsz=4096   blocks=0,
```

Важно! Не используйте mkfs без каких-либо экстеншенов (xfs,ext4), так как в этом случае будет создана устаревшая файловая система Ext2.

Монтирование файловой системы

Последним шагом является непосредственно монтирование созданной файловой системы в точке монтирования. Эта операция выполняется с помощью команды `mount <файловая система> <точка монтирования>`.

```
root@server:~# mkdir /opt/database

root@server:~# du -sh /opt/database/

4.0K /opt/database/

root@server:~# mount /dev/sdb1
/opt/database/

└─/opt/database                               /dev/sdb                               xfs
rw,relatime,attr2,inode64,logbufs=8,logbsize=32k,noquota

root@server:~# du -sh /opt/database/

0 /opt/database/
```

Теперь на на данный раздел диска с файловой системой можно записывать данные, а с помощью команды `umount <точка монтирования>` выполнить операцию отключения раздела от точки монтирования.

```
root@server:~# mount /dev/sdb1 /opt/database/

root@server:~# cat /var/log/syslog > /opt/database/some_data

root@server:~# du -sh /opt/database/

16K /opt/database/

root@server:~# ls -l

/opt/database/ total 4

-rw-r--r-- 1 root root 20 Jan 7 17:21 some_data

root@server:~# umount /opt/database

root@server:~# du -sh /opt/database/

4.0K /opt/database/

root@server:~# ls -l

/opt/database/ total 0
```

При помощи команды `mount` без каких-либо ключей можно увидеть примонтированные файловые системы.

```

root@server:~# mount

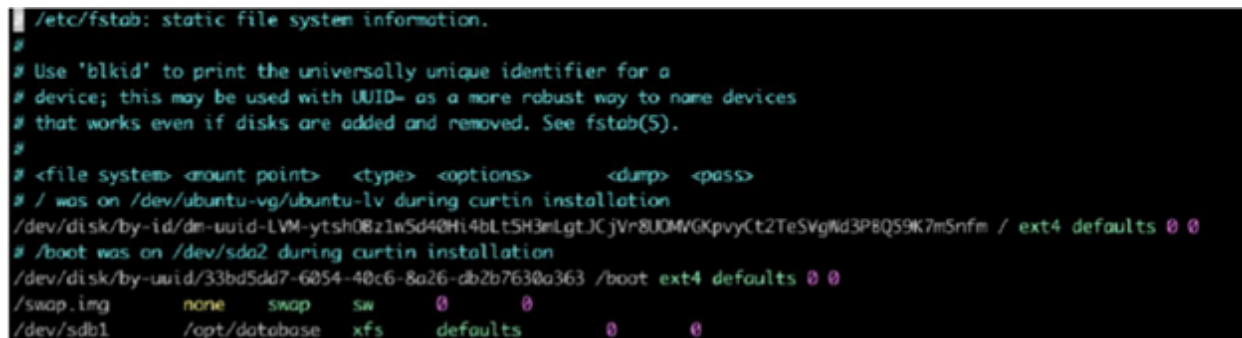
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
/dev/mapper/ubuntu--vg-ubuntu--lv on / type ext4 (rw,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
...

```

Важно! При перезагрузке системы все файловые системы будут автоматически размонтированы.

Автоматическое монтирование

Монтирование разделов в точки монтирования при старте операционной системы организуется через специальный файл **/etc/fstab**, в котором прописываются имя устройства, тип файловой системы, точка монтирования и дополнительные опции монтирования.



```

# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/ubuntu-vg/ubuntu-lv during curtin installation
/dev/disk/by-id/dm-uuid-LVM-ytsh0Bz1nSd40H14bLtSH3mLgtJCjVr8UOMVgKpvyCt2TeSVgNd3PBQ59K7mSnfm / ext4 defaults 0 0
# /boot was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/33bd5dd7-6054-40c6-8a26-db2b7630a363 /boot ext4 defaults 0 0
/swap.img none swap sw 0 0
/dev/sdb1 /opt/database xfs defaults 0 0

```

Рис 6. Содержание **/etc/fstab**

На рисунке 6 в последней строке описаны инструкции для монтирования созданной ранее файловой системы. Разберем ее по порядку:

- **/dev/sdb1** - ранее созданный раздел диска с файловой системой
- **/opt/database** - точка монтирования
- **xfs** - тип используемой файловой системы
- **defaults** - опции, связанные с файловой системой. К примеру файловую систему можно примонтировать в режиме read only

- Последние два нуля связаны с отключением использования утилит `dump` и `fsck`.

Файловые системы будут автоматически примонтированы при старте, либо можно воспользоваться командой `mount -a`. Более подробно о утилитах `mount`, `umount`, а также файле `/etc/fstab` можно прочитать на соответствующих страницах встроенного справочного руководства `man`.

Важно! Необходимо быть крайне внимательным при заполнении `/etc/fstab`, так как в случае ошибок и опечаток операционная система загрузится в `rescue mode`, то есть с ограниченным функционалом и будет доступна только через терминал сервера или виртуальной машины.

Расширенные возможности хранения данных

Если использовать традиционные разделы, которые обсуждались ранее в уроке, мы столкнемся с набором ограничений:

- без остановки системы разделы невозможно переразбить или заменить.
- отсутствует возможность начать работать с двумя дисками как с одним разделом. Можно смонтировать разные разделы и диски в одну виртуальную файловую систему. У вас могут быть /home, /bin, /usr/bin и т.д. — все на разных дисках (когда-то так и было). Но это не позволит, например, увеличить в два раза дисковое пространство под /home без остановки системы и замены диска/раздела.

Для решения описанных проблем можно использовать LVM (Logical Volume Manager), Stratis, или VDO (Virtual Data Optimizer). LVM существует уже довольно давно, широко распространен и используется по умолчанию при установке Ubuntu, поэтому мы подробнее остановимся именно на нем.

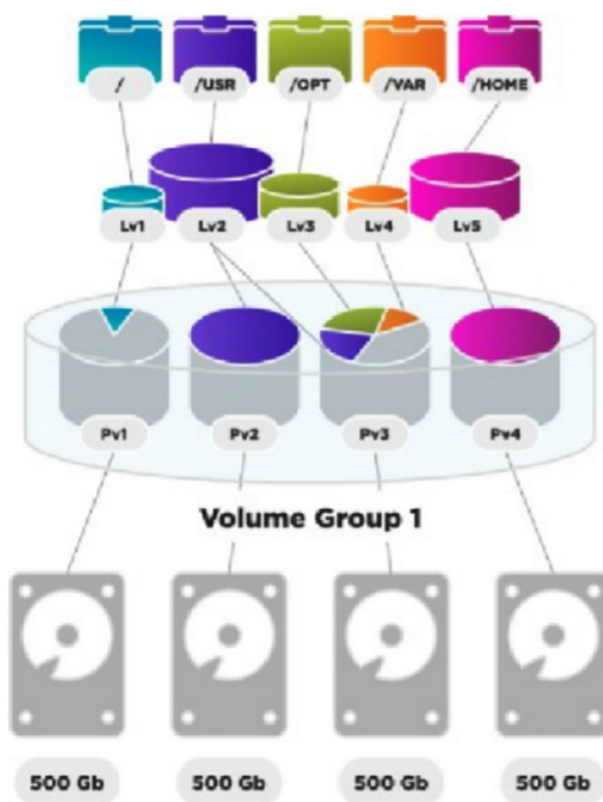


Рис.7 Структура LVM

На рисунке 7 изображена структура LVM. Центральным элементом является группа томов или Volume Group. Volume Group - это объединение устройств хранения информации, доступных в системе, в единое целое. Важно отметить, что эти устройства хранения могут быть чем угодно:

дисками, разделами, LUN в SAN - все они представляются в виде абстракции, называемой физическим томом (Physical Volume - PV). Таким образом, если у нас есть 4 диска по 500Гб, мы можем преобразовать каждый в физический том (на рисунке PV1-4) и объединить в группу, получив 2Тб доступного места.

Полученные 2Тб можно разделить на логические тома (Logical Volume - LV). Эти логические тома используются аналогично стандартным разделам диска, то есть поверх них создаются свои файловые системы, которые впоследствии будут смонтированы в точках монтирования (на рисунке `//usr/home/opt/var/home`).

В частном случае можно все свободное место в группе отдать под один логический том, тем самым получив виртуальный диск на 2Тб.

LVM хорош по нескольким причинам:

- LVM предлагает гибкое решение для управления хранилищем. Объемы больше не привязаны к ограничениям физических жестких дисков (LV2 на рисунке 7). Если требуется дополнительное пространство для хранения, группу томов можно легко расширить, добавив новый физический том, чтобы можно было добавить дисковое пространство к логическим томам.
- Поддержка snapshot. snapshot сохраняет текущее состояние логического тома и может использоваться для отката изменений или создания резервной копии файловой системы.
- возможность простой замены вышедшего из строя оборудования. Если жесткий диск выходит из строя, данные могут быть перемещены внутри группы томов с помощью команды `pvmove`, отказавший диск затем может быть удален из группы томов, а новый жесткий диск может быть добавлен динамически, без простоя.

Работа с LVM

Создание логических томов

Давайте пошагово разберем, как можно взаимодействовать с Logical Volume Manager.

LVM может работать с целыми устройствами, но рекомендуется использовать разделы, потому что это упрощает распознавание того, что проис-

ходит на устройствах в настройке LVM, так что сначала нужно создать раздел. В parted потребуется установить лейбл `lvm on`.

```
(parted) mkpart
Partition name? []? lvm_part
File system type? [ext2]?
Start? 100
End? 200

Warning: The resulting partition is not properly aligned for best
performance: 195313s % 2048s != 0s

Ignore/Cancel? Ignore

(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       17.4kB   100MB   100MB     xfs          database
 2       100MB   200MB   100MB     ext2         lvm_part

(parted) set 2 lvm on
(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       17.4kB   100MB   100MB     xfs          database
 2       100MB   200MB   100MB     ext2         lvm_part   lvm
```

Создадим из раздела Physical Volume или физический том.

```
root@server:~# pvcreate /dev/sdb2

Physical volume "/dev/sdb2" successfully created.

root@server:~# pvs
```

PV	VG	Fmt	Attr	
/dev/sda3	ubuntu-vg	lvm2	a--	0
/dev/sdb		lvm2	---	<95.37m <95.37m

Создадим новую Volume Group и добавим в нее созданный ранее физический том. Все это можно сделать с помощью одной команды. Последующее расширение группы выполняется с помощью `vgextend`.

```
root@server:~# vgcreate example_vg

/dev/sdb2 Volume group "example_vg"

successfully created
```

VG	#PV	#LV	#SN	Attr	VSize	VFree
exampl	1	0	0	wz--n-	92.00m	
ubuntu-vg	1	1	0	wz--n-		0

Теперь можем выделить часть группы под логический том.

```
root@server:~# lvcreate -n example_lv -L 20M

root@server:~# lvcreate -n example_lv -L 20M

example_vg

Logical volume "example lv" created.
```

LV	VG	Attr	LSize	Pool	Origin	Data%	Meta%	Move	Log
Cpy%	Sync	Convert							
example_lv	example_vg								
		-wi-a-----							

С помощью опции `-n` можно задать имя, а `-L` - размер тома.

Примечание! Если имя не задано, то ОС сгенерирует сама и оно будет не очевидно.

Создаем и монтируем файловую систему по аналогии с обычным раз-делом.

```
root@server:~# mkfs.xfs /dev/example_vg/example_lv
meta-data=/dev/example_vg/example_lv          isize=512  agcount=1,
          =                                   sectsz=512   attr=2,
          =                                   crc=          finobt=1, sparse=1,
          =                                   reflink=
data      =                                   bsize=4096   blocks=5120,
          =                                   sunit=        swidth=0 blks
naming    =version 2                       bsize=4096   ascii-ci=0, ftype=1
log        =internal log                   bsize=4096   blocks=1368,
          =                                   sectsz=512   sunit=0 blks, lazy-
realtime                        extsz=4096   blocks=0,
root@server:~# mkdir /opt/example_lvm
root@server:~# mount
/dev/example_vg/example_lv /opt/ database/
example_lvm/
```

Увеличение логических томов

Для создания логического тома было использовано не все пространство, доступное в созданной Volume Group.

```
root@server:~# vgs
VG          #PV #LV #SN Attr   VSize  VFree
exempl      1   1   0 wz--n- 92.00m 72.00m

root@server:~# lvs
LV          VG          Attr      LSize  Pool  Origin  Data%  Meta%  Move  Log
Cpy%Sync
example_lv   example_vg  -wi-ao---- 20.00m
```

Для увеличения логического тома следует воспользоваться утилитой `lvextend`.

```

root@server:~# lvextend -r -L +72M /dev/
/dev/example_vg/example_lv /dev/ubuntu-vg/ubuntu-lv
root@server:~# lvextend -r -L +72M
/dev/example_vg/example_lv

Size of logical volume example_vg/example_lv changed from 20.00 MiB
(5 extents) to 92.00 MiB (23 extents).

Logical volume example_vg/example_lv successfully resized.

meta-data=/dev/mapper/example_vg-example_lv isize=512 agcount=1, agsize
=5120 blks
        =                               sectsz=512   attr=2, projid32bit=1
        =                               crc=1         finobt=1, sparse=1, rmapbt=0
        =                               reflink=1
data      =                               bsize=4096   blocks=5120, imaxpct=25
        =                               sunit=0       swidth=0 blks
naming    =version 2                       bsize=4096   ascii-ci=0, ftype=1
log        =internal log                   bsize=4096   blocks=1368, version=2
        =                               sectsz=512   sunit=0 blks, lazy-count=1
realtime  =none                             extsz=4096   blocks=0, rtextents=0

data blocks changed from 5120 to 23552

root@server:~# vgs

VG          #PV #LV #SN Attr          VSize VFree
example_vg          1  1          0 wz--n-
92.00m          0
ubuntu-vg          1  1  0  wz--n- <9.00g  0

root@server:~# lvs

LV          VG          Attr          LSize Pool Origin Data% Meta% Move
Log
Cpy%Sync Convert
example_lv          example_vg  -

```

Помимо изменения размеров логического тома, требуется изменить и

размер файловой системы, установленной на него. Для этого следуют воспользоваться утилитами `e2resize`, `xfs_growfs` или ключом `-r` в утилите `lvextend`. С помощью ключа `-L` указываем, на сколько нужно увеличить размер логического тома.

Исходя из выводов, все свободное пространство в группе было использовано. Для дальнейшего расширения логического тома или создания нового понадобится создать новый физический том и добавить его в группу `example_vg` с помощью команды `vgextend`.

Мониторинг доступного места

Для нахождения количества свободного места существуют две утилиты: `df`, `du`. Отличие этих утилиты заключается в том, откуда они берут информацию о доступном пространстве. `df` запрашивает информацию из установленной файловой системы, `du` в свою очередь опирается на данные непосредственно с диска. Бывают случаи, когда данные от утилит разнятся.

В обеих утилитах полезным ключом является `-h` или `-human-readable`. Без него информация будет отражена в блоках.

```
root@server:~# df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	951M	0	951M	0%	/dev
tmpfs	199M	1M	198M	1%	/run
/dev/mapper/ubuntu--vg-ubuntu--lv	8.8G	4.6G	3.8G	55%	/
tmpfs	994M	0	994M	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	994M	0	994M	0%	/sys/fs/cgroup
/dev/sda2	976M	264M	712M	22%	/boot
/dev/loop0 /1932	56M	56M	0	100%	/snap/core18
/dev/loop1 /1944	56M	56M	0	100%	/snap/core18
/dev/loop2	72M	72M	0	100%	/snap/lxd/16099
/dev/loop3	68M	68M	0	100%	/snap/lxd/18150

/dev/loop4 /10492	32M	32M	0	100%	/snap/snapd
tmpfs	199M	0	199M	0%	/run/user/1000
/dev/loop6 /10707	32M	32M	0	100%	/snap/snapd
/dev/mapper/example	87M	2.0M	85M	3%	

Утилита `du` требует указания пути до каталога.

```
root@server:~# du -h
/opt/database/ 16K
```

Ссылки и файловые дескрипторы

Для сохранения свободного места можно воспользоваться концепцией ссылок. Ссылки — это особенность файловой системы, которая позволяет размещать один и тот же файл в разных каталогах.

Перед тем, как воспользоваться ссылками, следует познакомиться с понятием индексного дескриптора. С каждым файлом в операционной системе Linux связана особая структура данных — индексный дескриптор (**inode**). Эти данные хранят метаданные о файле: владелец, права доступа, время последнего изменения и т. д. Inode также содержит информацию о физическом расположении данных.

Inode уникальны на уровне разделов. Вы можете иметь два файла с одинаковым номером inode, если они находятся в разных разделах. Информация inode хранится в табличном формате в виде структуры в начале каждого раздела. Просмотреть inode можно, используя команду **ls -li**.

```
# ls -li
434234 drw-r--r-- 2 root      examples 4096 Dec 1 14:27 dir
434231 drw-rwSr-t 2 root      developers 4096 Dec 1 14:52 dir1
434232 drw-rw-rwT 2 root      root      4096 Dec 1 14:53 dir2
422147 -rwsr-xr-x 1 root root      0 Dec 1 14:04 myfile
```

Каталог содержит таблицу соответствия **имя_файла** → **inode**. В этой таблице требуется уникальность имён, но не уникальность номеров inode.

Благодаря этому каждый объект файловой системы может иметь несколько имён.

Вернемся к ссылкам. **Жёсткая ссылка** — это запись в каталоге, указывающая на inode. Жёсткая ссылка создаётся только для файлов, за исключением специальных записей, указывающих на саму директорию (.) и родительскую директорию (..). Жёсткие ссылки используются только в пределах одного раздела. На практике жёсткие ссылки уже почти не применяются в работе. Создать жёсткую ссылку можно, используя команду **ln** без параметров, например `ln file link`, где `file` — имя файла-источника, `link` — имя ссылки на файл. Родительский файл можно перемещать или даже удалять без вреда для ссылки, то есть фактически файл, на который есть хотя бы одна жёсткая ссылка, не перестанет существовать. Изменение содержимого файла и разрешений также повлияет и на ссылку.

Символическая ссылка — это запись в каталоге, указывающая на имя объекта с другим inode. Символическая ссылка наиболее близка к ярлыку в Windows. Символическая ссылка может ссылаться на файл и на каталог. Символические ссылки могут существовать на разных разделах. Права доступа и inode у символической ссылки отличаются от родительского файла. При перемещении или удалении файла-родителя символическая ссылка «ломается», так как ссылка привязывается именно к имени файла. Создать символическую ссылку можно командой `ln -s file link`, где параметр **-s** говорит, что мы создаём символическую ссылку, `file` — имя источника файла, `link` — имя ссылки на файл.

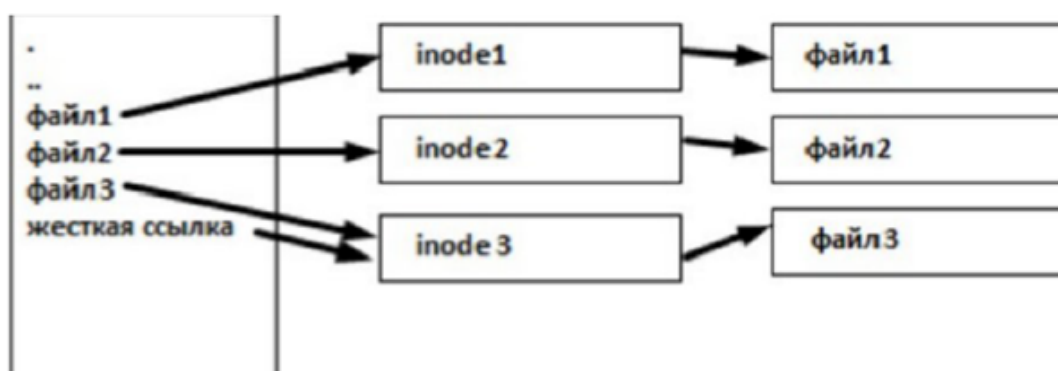


Рис. 9 Ссылки и дескрипторы

Практическое задание

Внимание! Перед началом выполнения работы рекомендуется сделать snapshot виртуальной машины, чтобы можно было откатить изменения в случае ошибок. Размер добавляемого диска и создаваемых разделов допускается варьировать, в зависимости от возможностей ПК.

1. Добавить к серверу второй жесткий диск размером 300 Мегабайт
 - a. Разбить диск на 2 раздела по 150 Мб
 - b. На первом разделе создать файловую систему Ext4 и примонтировать к папке `/mnt/ext4`
 - c. Добавить точку монтирования `/mnt/ext4` в `/etc/fstab`
2. На втором разделе создать локальный том (LVM) размером 100 Мб. Создать файловую систему XFS и примонтировать к папке `/mnt/xfs`.
3. Создать файл `/mnt/ext4/file1` и наполнить его произвольным содержанием. Скопировать его в `file2`. Создать символическую ссылку `file3` на `file1`. Создать жёсткую ссылку `file4` на `file1`. Посмотреть, какие inode у файлов. Удалить `file1`. Что стало с остальными созданными файлами? Попробовать вывести их на экран.
4. Создать ссылку вида `/mnt/xfs/link`, по которой будет доступен `/mnt/ext4/file4`.

Глоссарий

Раздел — часть долговременной памяти жёсткого диска или флеш-накопителя, выделенная для удобства работы и состоящая из смежных блоков. На одном устройстве хранения может быть несколько разделов.

Точка монтирования — это каталог, с помощью которого обеспечивается доступ к новой файловой системе, каталогу или файлу. Точка монтирования используется для реализации возможности динамически присоединять разделы диска к файловой системе и отсоединять их во время работы операционной системы.

Файловая система — часть операционной системы, которая обеспечивает чтение и запись файлов на дисковых носителях информации. Файловая система устанавливает физическую и логическую структуру файлов, правила их создания и управления ими, а также сопутствующие данные файла

и идентификацию. Конкретная файловая система определяет размер имени файла и максимально возможный размер файла.

Logical Volume Manager (LVM) — система управления томами с данными для Linux. Она позволяет создавать поверх физических разделов (или даже неразбитых жёстких дисков) логические тома, которые в самой системе будут видны как обычные блочные устройства с данными, то есть как обычные разделы.

Inode — индексный дескриптор, структура данных в файловых системах Linux. Предназначена для хранения метаданных о стандартных файлах, каталогах или других объектах файловой системы, кроме непосредственно данных и имени.

Дополнительные материалы

Типы файловых систем Linux

inode и каталоги

Используемые источники

Робачевский Андрей М. Операционная система Unix

В.Костромин, "Linux для пользователя", изд. "БХВ-Петербург", 2002 г., серия "Самоучитель"

LVM - это просто!