Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)

Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

В.Ю. Мельников

Исследование методов организации внешней памяти.

Электронное учебное издание

Методические указания по выполнению лабораторных работ
по дисциплине "Операционные системы"

2021

Введение

Цель работы - исследование файловых систем, применяющихся в UNIX-подобных системах, а также освоение основных утилит для работы с файлами.

Продолжительность работы - 4 часа.

Порядок выполнения работы:

1. Согласно данному пособию выполнить предложенные действия для освоения работы с файловой системой. Эту часть лабораторной работы в отчёт можно не включать, но сохраняйте снимки экрана. Часть из них можно будет использовать при оформлении отчёта по следующему пункту.

2. Выполнить задания из файла «5 zadanie_Files»

3. Оформить отчёт. Отчёт должен включать:

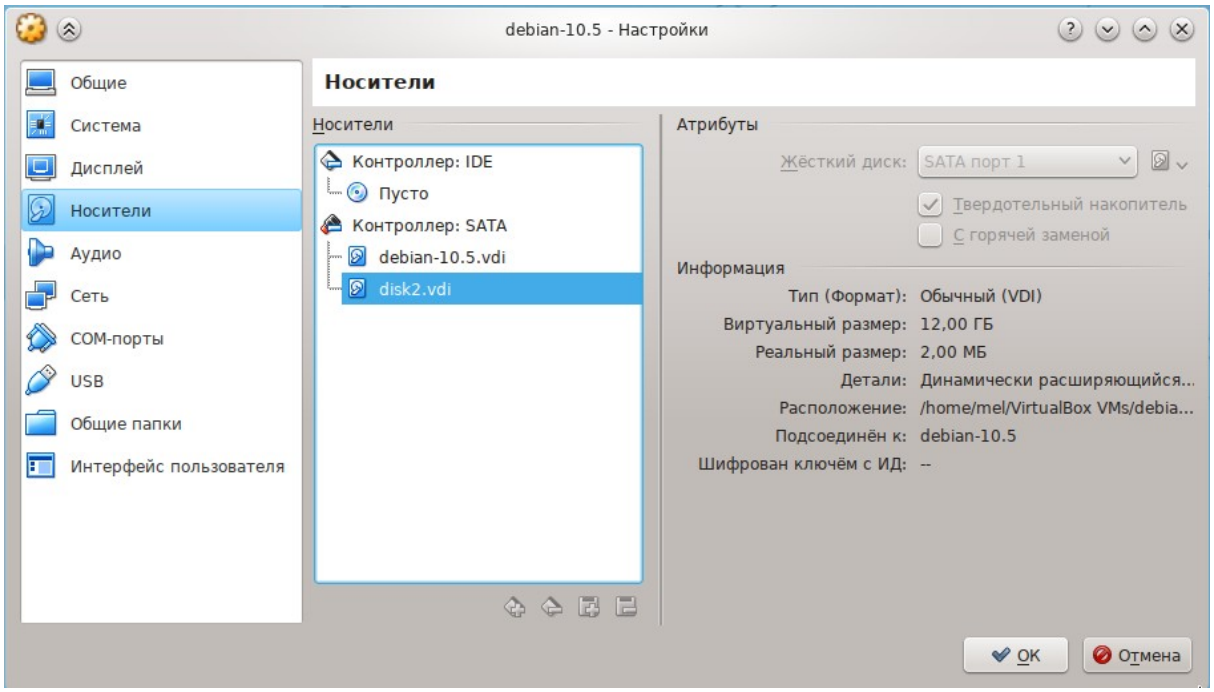
- Титульный лист
- Название работы и ее цель;
- Выполнение заданий из файла «5 zadanie_Files». Для каждого задания привести:
  - Текст задания
  - Команды используемые при выполнении задания (если их не видно на снимке экрана)
  - Результаты выполнения команд (на снимках экрана).

Контрольные вопросы в одноимённом разделе служат для самоконтроля освоения материала и подобны вопросам, задаваемым на защите. Ответы включать в отчёт не надо.

Подключение нового жёсткого диска

Какой бы большой жёсткий диск не стоял на Вашем компьютере, рано или поздно свободное место на нём закончится. Надо будет подключить дополнительный диск или перенести данные на новый диск. Научимся это делать.

Перед запуском виртуальной машины добавьте второй жёсткий диск



Запустите виртуальную машину, введите имя пользователя «root» и его пароль.

Посмотрим текущую конфигурацию жёстких дисков и других устройств внешней памяти командой «`fdisk -l`».

```
root@debian10:~# fdisk -l
Disk /dev/sda: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe6dd9247

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *        2048 14680063 14678016    7G 83 Linux
/dev/sda2       14682110 16775167  2093058 1022M  5 Extended
/dev/sda5       14682112 16775167  2093056 1022M 82 Linux swap / Solaris


Disk /dev/sdb: 12 GiB, 12884901888 bytes, 25165824 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Диск «/dev/sda» мы уже видели при выполнении первой работы. После подключения нового диска к виртуальной машине появился диск «/dev/sdb». Но пока его использовать нельзя. Сначала надо создать на нём разделы.

Разметка жёсткого диска

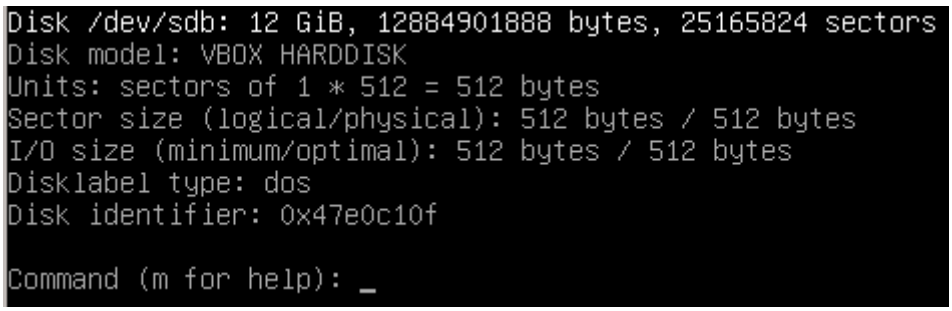
Удобно сделать это в интерактивном режиме команды «fdisk». Дайте команду:

```
fdisk /dev/sdb
```

В этом режиме изменения остаются в памяти, пока Вы не дадите команду «w» (write).

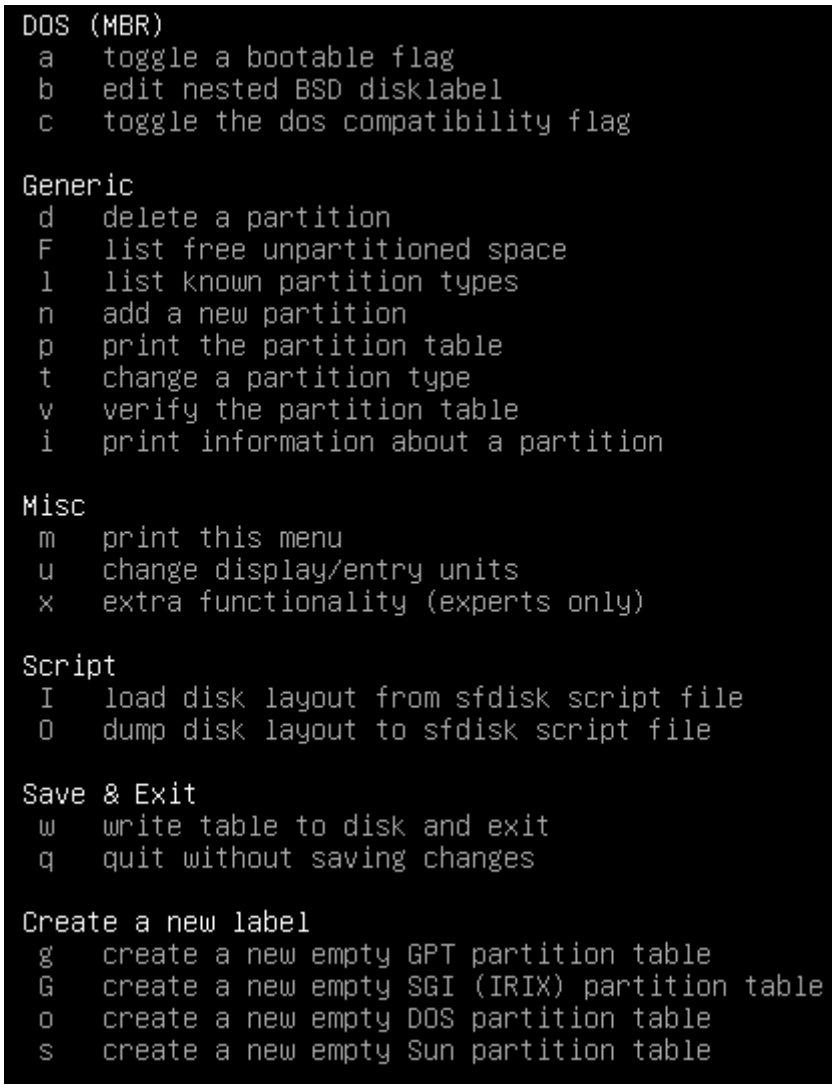
А если Вы ошибётесь, введите команду «q» - выход без сохранения.

Однако, будьте внимательны, если перепутаете диск, вы уничтожите разметку основного диска. Поэтому прежде всего убедимся, что выбранный диск пуст. Введите команду «p» (print) и нажмите «ENTER».



Разделов нет. Значит выбрали правильный диск.

В подсказке команды предлагается ввести команду «m» для получения краткой справки по командам.



Прежде чем создавать разделы надо создать таблицу разделов. Тут нам надо выбрать между уже знакомой нам таблицей в MBR и новым форматом таблицы разделов GPT.

MBR (Master Boot Record) — Главная загрузочная запись. Может хранить не более 4 разделов размером не более 2 Тб. При необходимости большего числа разделов приходится создавать Extended раздел.

GPT (GUID Partition Table) — Таблица разделов с глобальными идентификаторами. Это новый формат таблицы разделов. Он позволяет создавать до 128 разделов размером до 9,4 зеттабайта (10^{21}байт). Кроме того хранится 2 копии таблицы разделов с контрольной суммой. Так что в случае повреждения 0-сектора диска будет использоваться вторая копия.

Но надо учитывать, что старые операционные системы и некоторые программы не поддерживают GPT. А ещё, одновременно с GPT на смену BIOS пришла UEFI (Unified Extensible Firmware Interface). В отличие от зашитой в ПЗУ BIOS большая часть UEFI хранится в отдельном разделе на жёстком диске, что снимает ограничение на размер и

позволяет обновлять UEFI по мере появления новых устройств. Причём, драйвера устройств из UEFI будут доступны операционной системе, когда она загрузится.

Для совместимости со старыми ОС и программами, UEFI может имитировать BIOS. Если Вам важна совместимость со старым ПО, при загрузке компьютера, войдите в программу настройки (setup), выберите режим «Legacy BIOS» и сохраните настройки. В документации на модель Вашего компьютера указано какую кнопку надо нажать при загрузке компьютера, чтобы войти в режим настроек.

По умолчанию, «VirtualBox» создаёт виртуальную машину с BIOS. Если в будущем, Вы захотите создать с помощью «VirtualBox» виртуальную машину с UEFI, выберите в левом списке элемент «Система», и на вкладке «Материнская плата» поставьте отметку «Включить EFI» (второй флажок снизу). Если этого флажка нет, то надо установить «VirtualBox extension pack».

Обычно, совместно с UEFI используется GPT, а совместно с BIOS — MBR. Но не обязательно. Debian 10 поддерживает как MBR так и GPT. В этой лабораторной работе операционная система, загрузившись с помощью BIOS в первого диска будет использовать работать с GPT на втором диске. С MBR уже познакомились на первой лабораторной работе, теперь поработаем с GPT.

Дайте команду «g».

Создадим, для тренировки, два раздела:

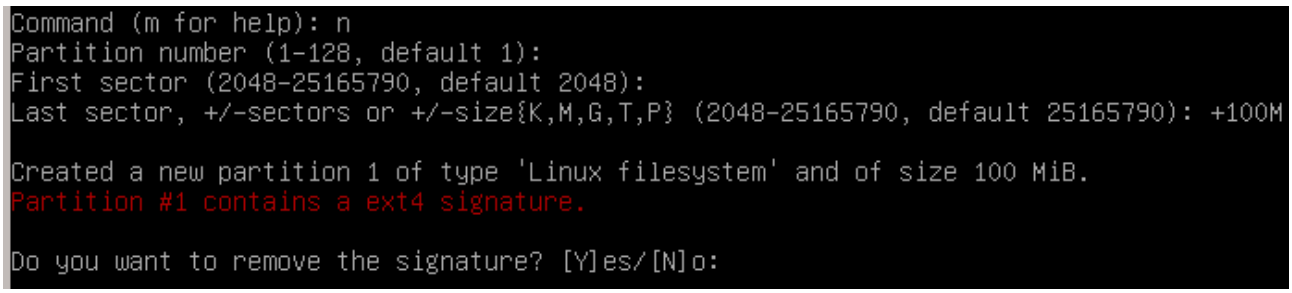
- Первый 100Мб — зарезервируем под UEFI (на будущее)
- Второй — обычный раздел Linux.

Дайте команду «n» (new).

Программа запрашивает номер раздела и предлагает по умолчанию «1». Просто нажмите «Enter».

Программа запрашивает с какого сектора начинать раздел. По умолчанию 2048. Годится, нажмите «Enter». Когда будете размечать реальный диск, не экономьте, вдруг потом придётся устанавливать на диск системный загрузчик.

Программа запрашивает номер последнего сектора раздела. По умолчанию - весь диск. А мы хотели 2 раздела. Поэтому введите «+100M»



По умолчанию создаётся раздел типа «Linux filesystem». А мы хотели его под UEFI зарезервировать. Поэтому на вопрос «Удалить подпись?» отвечаем «Y».

Теперь изменим тип раздела командой «t» (type)

Программа запрашивает тип раздела. Посмотрим, какие бывают. Введите «L». Нам повезло — нужный нам тип «1» (EFI) уже на первой странице. Листать можно «PgUp», «PgDown». Для выхода из режима списка нажмите «q» (как в man, more и большинстве интерактивных команд).

Введите тип раздела «1» (EFI это синоним UEFI, просто в разных источниках их называют по разному).

```
Partition type (type L to list all types): 1
Changed type of partition 'Linux filesystem' to 'EFI System'.
```

Второй раздел — обычный — «Linux filesystem» размером на всё оставшееся место создать просто. Дайте команду «n» и на все вопросы нажимайте «Enter».

```
Command (m for help): n
Partition number (2-128, default 2):
First sector (206848-25165790, default 206848):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (206848-25165790, default 25165790):

Created a new partition 2 of type 'Linux filesystem' and of size 11,9 GiB.
```

Наконец введите команду «w» (write) для записи изменений таблицы разделов (или «q» для выхода без сохранения)

Проверим, таблицу разделов командой «`lsblk -f`» (**`li`**`st of `**`bl`**`oc`**`k`** `devices`)

```
NAME   FSTYPE LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINT
sda
├─sda1 ext4         b2511cd1-a7c8-4d0c-9f57-4bb5ebfc8d87    5,6G    13% /
├─sda2
└─sda5 swap         414ca8f3-67bd-4e6c-ac79-a208fa613732                [SWAP]
sdb
├─sdb1
└─sdb2
```

Созданные нами разделы имеют имена «sdb1» и «sdb2» или полностью «`/dev/sdb1`» и «`/dev/sdb2`».

Создание файловой системы

Пока мы просто выделили место под разделы. Чтобы раздел можно было использовать, надо его создать в нём файловую систему командой «mkfs».

Иногда эту операцию называют «Отформатировать диск». Это устаревшее название осталось со времён, когда когда вместе с созданием файловой системы производилось низкоуровневая разметка физического носителя. В наше время низкоуровневая разметка производится на заводе изготовителе. А команда создания файловой системы мгновенно удаляет старую таблицу размещения файлов и создаёт

новую (кстати, даже не переспрашивает, уверены ли вы). Кажется, что вы удалили все данные раздела. Но так «левые доходы» он налоговой не спрячешь. Есть программы, которые восстанавливают большую часть файлов и каталогов после этой операции.

Дайте команды

```
mkfs -t vfat /dev/sdb1
mkfs -t ext4 /dev/sdb2
```

Примечание: Если первая команда выдала ошибку «mkfs.vfat: нет такого файла или каталога». Установите пакет «dosfstools» и повторите команду.

```
root@debian10:~# mkfs -t vfat /dev/sdb1
mkfs.fat 4.1 (2017-01-24)
root@debian10:~# mkfs -t ext4 /dev/sdb2
mke2fs 1.44.5 (15-Dec-2018)
Creating filesystem with 3119867 4k blocks and 780288 inodes
Filesystem UUID: 526cfc7d-69a5-4e2d-a685-cb6fa2c85e98
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

В первом разделе создаём файловую систему «FAT32» (как положено по спецификации UEFI). Во втором разделе «ext4», такую же как у раздела «sda1», созданного при инсталляции «Debian».

Монтирование разделов

В Linux нет литеры дисков (как в Windows), путь к файлам указывается от корня файловой системы. Разделы (логические диски) перед использованием надо смонтировать в один из имеющихся каталогов файловой системы.

Создадим каталог «storage» и смонтируем в него «sdb2» командами:

```
mkdir /storage
mount /dev/sdb2 /storage
```

«/» в начале имени каталога означает, что мы задаём полный путь от корневого каталога. Таким образом, мы создаём каталог «storage» непосредственно в корневом каталоге. И монтируем в него наш новый раздел.

На экран не выдано никаких сообщений. Всё нормально. В Linux так принято. Если нет сообщений об ошибках, значит всё нормально. Если Вас это смущает, почти во всех командах есть опция «-v» (verbose).

Запишем в «/storage» файл и посмотрим содержимое каталога:

```
echo "xxxx" >/storage/test.txt
ls /storage
lost+found  test.txt
```

Монтирование флешки

Если вы хотите использовать флешку без использования файлового менеджера, через командную строку, её тоже придётся смонтировать в какой нибудь каталог. Вставьте вашу флешку в USB на реальном компьютере и дайте в окне вашей виртуальной машины из меню «Устройства» подменю «USB» и выберите Вашу флешку.

В Linux принято временно монтировать файловые системы в каталоге «`/mnt`». Создадим каталог «`usb1`» в этом каталоге:

```
mkdir /mnt/usb1
```

Самостоятельно посмотрите имя устройства вашей флешки командой «`lsblk`» и смонтируйте её в созданном каталоге.

Посмотрим содержимое Вашей флешки

```
ls /mnt/usb1
```

По окончании работы с флешкой демонтируем её:

```
umount /mnt/usb1
```

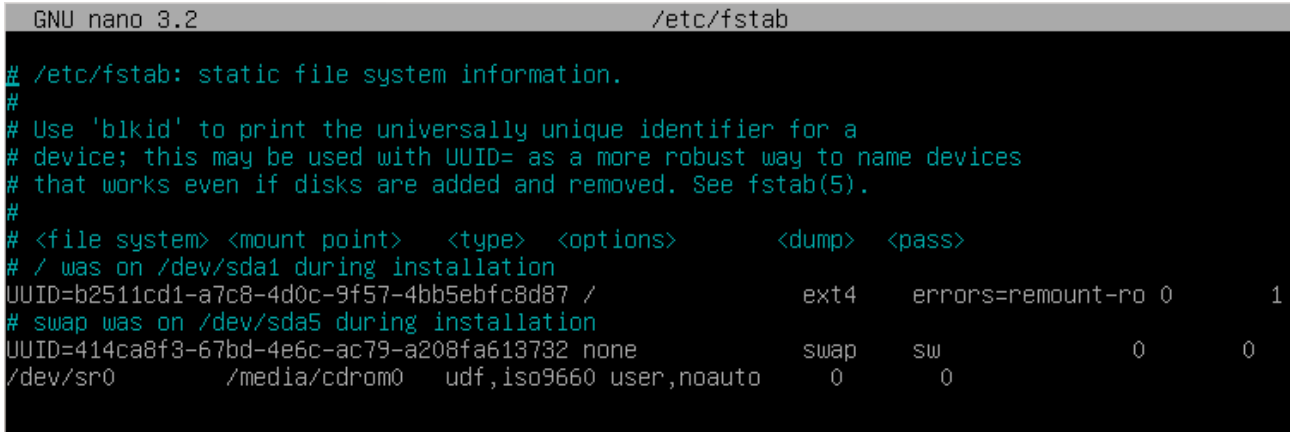
Если теперь посмотреть содержимое каталога «/mnt/usb1» он окажется пустым.

Примечание. Файловые менеджеры, сами автоматически обнаруживают и монтируют флешки. Командой «mount» реально приходится пользоваться очень редко.

Автоматическое монтирование

После перезагрузки таблица монтирования теряется. Чтобы раздел автоматически монтировался при загрузке Linux, надо добавить запись в файл «`/etc/fstab`».

Дайте команду «`nano /etc/fstab`»



Каждая строка описывает раздел, который нужно смонтировать и содержит следующие поля (перечисленные через пробел или табуляцию):

- Устройство — раздел диска. Можно указать просто «/dev/sdb1», но если мы подключим новые жёсткие диски, отключим их или переключим в другие разъёмы на материнской плате, имена дисков могут измениться, поэтому надёжнее задать уникальные UUID раздела как в приведённом снимке экрана. Идентификатор раздела

можно посмотреть с помощью команды «`lsblk -f`»

- Точка монтирования - каталог в который мы хотим смонтировать раздел. На приведённом снимке экрана первый раздел имеет точку монтирования «/» - в корень файловой системы.
- Тип файловой системы. Например, ext4
- Опции — позволяют задать различные параметры в формате параметр1=значение1, пареметр2=значение2,… Например, при монтировании сетевого диска в опциях задается имя пользователя и пароль.
- Резерв - указывает нужно ли делать резервную копию раздела
- Проверка — указывает очередь проверки на ошибки (1 - в первую очередь, 2 — во вторую, 0 - не проверять)

Для подключения нашего нового раздела добавьте строку:

```
/dev/sdb2 /storage ext4 rw 0 2
```

Здесь:

«`/dev/sdb2`» - Созданный нами раздел. Не хочется набирать 36 символов идентификатора созданного нами диска. Для лабораторной сойдёт, но в реальной работе не поленитесь ввести идентификатор.

«`/storage`» - каталог в который монтируем наш раздел

«`ext4`» - тип файловой системы, который мы указывали в команде «`mkfs`»

«`rw`» - Монтирование файловой системы для чтения и записи

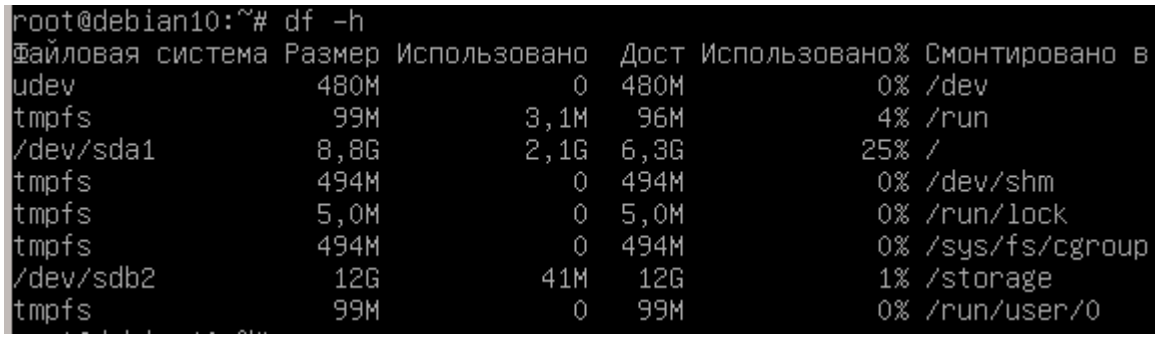
«0» - Резервную копию не создавать (как у первого раздела)

«2» - Проверять во вторую очередь (1 устанавливается для раздела, который монтируется в корень файловой системы, 2 для прочих)

Сохраните изменения «Ctrl+O», выйдите из редактора «Ctrl+X» и перезагрузите виртуальную машину «`reboot`».

Свободное место в разделах

Проверим, смонтировался ли наш новый раздел, и заодно познакомимся с ещё одной полезной командой «`df -h`»



```
root@debian10:~# df -h
Файловая система Размер Использовано  Дост Использовано% Смонтировано в
udev                480M            0  480M            0% /dev
tmpfs                99M          3,1M   96M            4% /run
/dev/sda1           8,8G          2,1G  6,3G           25% /
tmpfs               494M            0  494M            0% /dev/shm
tmpfs               5,0M            0  5,0M            0% /run/lock
tmpfs               494M            0  494M            0% /sys/fs/cgroup
/dev/sdb2            12G           41M   12G            1% /storage
tmpfs                99M            0   99M            0% /run/user/0
```

Эта команда показывает свободное место на разделах диска (Disk Free).

В данном случае, «/dev/sda1», смонтированный в корне файловой системы заполнен на 25%. А наш новый раздел «/dev/sdb2», смонтированный в каталог «/storage» заполнен на 1 %.

Опция «-h» (human readable) как и во многих других командах задаёт вывод размеров в килобайтах, мегабайтах и гигабайтах, что гораздо нагляднее вывода в байтах, который используется по умолчанию.

Обязательно выполняйте эту команду время от времени на сервере, чтобы своевременно обнаружить, что место заканчивается, почистить диск или подключить новый. Свободное место на диске имеет свойство исчезать загадочным образом по разнообразным причинам.

Обнаружив в колонке «Использовано» 99%, возникает закономерный вопрос - куда делось свободное место? Команда «du» (Disk Usage) поможет его найти.

```
du -h /storage
```

```
16K     /storage/lost+found
24K     /storage/
```

Эта команда показывает сколько места занимает заданный каталог и сколько его подкаталоги.

Часто подкаталогов много. Мы просто утонем в обилии вложенных каталогов. (попробуйте выполнить «du -h /root»).

Удобнее использовать эту команду с опцией «-s». При наличии этой опции, команда выводит место занятое заданным каталогом с подкаталогами, но не выводит размеры подкаталогов по отдельности. Например, «du -s -h /storage».

```
24K     /storage/
```

Однобуквенные опции можно объединять, эту же команду можно записать

```
du -sh /storage
```

Однако, так получается другая крайность. По одному размеры каталогов проверять долго, удобнее сделать так:

```
du -sh /storage/*
```

```
16K     /storage/lost+found
4,0K    /storage/test.txt
```

«/storage/*» означает, все файлы и подкаталоги каталога «/storage». Мы увидим размеры всех подкаталогов каталога «storage», но без подкаталогов следующего уровня. Символ «*» можно применять во многих командах. Он означает, что на этом месте в имени файла или каталога может стоять любое число любых символов. Например, «t*», «*.txt», «test.*».

Чтобы найти куда делось место на диске, сначала смотрим размер каталогов

корневого каталога, затем подробно подкаталоги подозрительно больших каталогов.

Теперь взгляните на размер файла «text.txt», помните, мы в него записали 4 символа «xxxx». Откуда «4K»? Если вы посмотрите размеры других файлов в других каталогах можно заметить, что размеры всех файлов и каталогов кратны «4Кб». Дело в том, что место под файлы выделяется блоками (в зависимости от типа файловой системы и размера раздела). В нашем случае это 4 килобайта.

Поэтому, когда пишете программу, избегайте записи мелких файлов. Например, разбивайте на группы по месяцу и сразу добавляйте в архивы. Даже без сжатия будете иметь огромную экономию места на диске. И это не только на Linux. На Windows, в добавок, больше 1000 файлов в каталоге тормозят так, что работать невозможно.

Ходим по каталогам

До сих пор мы во всех командах указывали полный путь от корня файловой системы. Это необязательно. Если файлы с которыми собираетесь работать в лежат в одном каталоге, удобно перейти в него и указывать в командах просто имена файлов без указания каталога.

Перейдём в каталог «/storage»

```
cd /storage
```

```
root@debian10:~# cd /storage/
root@debian10:/storage# _
```

Обратите внимание, как изменилось приглашение командной строки. После «:» отображается текущий каталог. Посмотрим его содержимое. Команда «ls КАТАЛОГ» (**l**i**s**t) выводит список файлов и каталогов заданного каталога.

```
ls
```

```
lost+found  test.txt
```

Если имя каталога в команде мы не указываем, то высвечивается содержимое текущего каталога.

Можно получить больше информации о файлах, задав опцию «-l» (**l**ong listnig format)

```
ls -l
```

```
root@debian10:~# ls -l /storage
итого 20
drwx------ 2 root root 16384 янв 23 23:39 lost+found
-rw-r--r-- 1 root root     4 мар 27 22:34 test.txt
```

Буква «d» (Directory) в начале строки показывает что это каталог

Каталог «lost+found» был создан автоматически. В него записываются «потерянные» файловой системой файлы. Обычно такие образуются при аварийной перезагрузке. Если в этой папке появятся файлы, посмотрите содержимое, если это не ваши файлы — можно их удалить, чтобы место не занимали.

Про «rwx...» и «root» поговорим в следующей лабораторной (права доступа).

Ещё мы видим размеры файлов. В частности, размер файла «test.txt» = 4 байта, и дату и время создания.

Создадим каталог «test» и зайдём в него

```
mkdir test
cd test
```

```
root@debian10:/storage# mkdir test
root@debian10:/storage# cd test
root@debian10:/storage/test#
```

Дайте команду

```
ls -la
```

```
drwxr-xr-x 2 root root 4096 апр 11 20:43 .
drwxr-xr-x 4 root root 4096 апр 11 20:43 ..
```

Опция «-l» выводит список в «широком формате», опция «-a» отображает скрытые файлы и каталоги. В данном случает это каталоги «`.`» и «`..`». Эта пара есть в каждом каталоге.

Каталог «`..`» - ссылка на родительский каталог (уровнем выше текущего). В нашем случае «/storege»

Каталог «`.`» - ссылка на текущий каталог. Она используется в командах, когда надо указать на текущий каталог. Например:

```
cp ../test.txt ./
```

```
root@debian10:/storage/test# cp ../test.txt ./
root@debian10:/storage/test# ls -la
итого 12
drwxr-xr-x 2 root root 4096 апр 11 21:06 .
drwxr-xr-x 4 root root 4096 апр 11 20:43 ..
-rw-r--r-- 1 root root    5 апр 11 21:06 test.txt
```

Эта команда копирует файл «`test.txt`» из родительского каталога (`../`) в текущий каталог (`./`) (можно просто «`.`»).

Если надо указать на каталог двумя уровнями выше, пишем «`../../`».

Возвращаемся в родительский каталог «`..`» и посмотрим, что получилось

```
cd ..
```

```
root@debian10:/storage/test# cd ..
root@debian10:/storage# _
```

Мы снова в «`/storage`». Посмотрим содержимое.

```
root@debian10:/storage# ls -l
итого 24
drwx------ 2 root root 16384 мар 28 16:50 lost+found
drwxr-xr-x 2 root root  4096 апр 11 21:06 test
-rw-r--r-- 1 root root     5 мар 28 17:08 test.txt
```

Видим наш новый каталог «test»

Рассмотрим ещё одну часто используемую форму команды «`cp`»

```
cp -r test test2
```

```
root@debian10:/storage# cp -r test test2
root@debian10:/storage# ls -l
итого 28
drwx------ 2 root root 16384 мар 28 16:50 lost+found
drwxr-xr-x 2 root root  4096 апр 11 21:06 test
drwxr-xr-x 2 root root  4096 апр 11 21:26 test2
-rw-r--r-- 1 root root     5 мар 28 17:08 test.txt
```

Эта команда создаёт копию каталога «test» пои именем «test2».

Опция «-r» (Recursive) — означает копировать каталог рекурсивно (вместе с файлами и подкаталогами). Не забывайте её, когда копируете каталоги. Новым файлам присваивается текущее время создания и владельцем становится текущий пользователь. Если надо сохранить владельца, права доступа и время создания, следует вместо «-r» использовать «-a».

Имеются так же команды «mv» - переместить файл или каталог и «rm» - удалить файл или каталог с похожими параметрами. Кому понадобятся, посмотрите описание в приложении.

Но вернёмся к указателям на каталоги. Дайте команды

```
cd ~
pwd
```

```
root@debian10:/storage# cd ~
root@debian10:~# pwd
/root
```

Команда «pwd» (Print Working Directory) выводит текущий каталог.

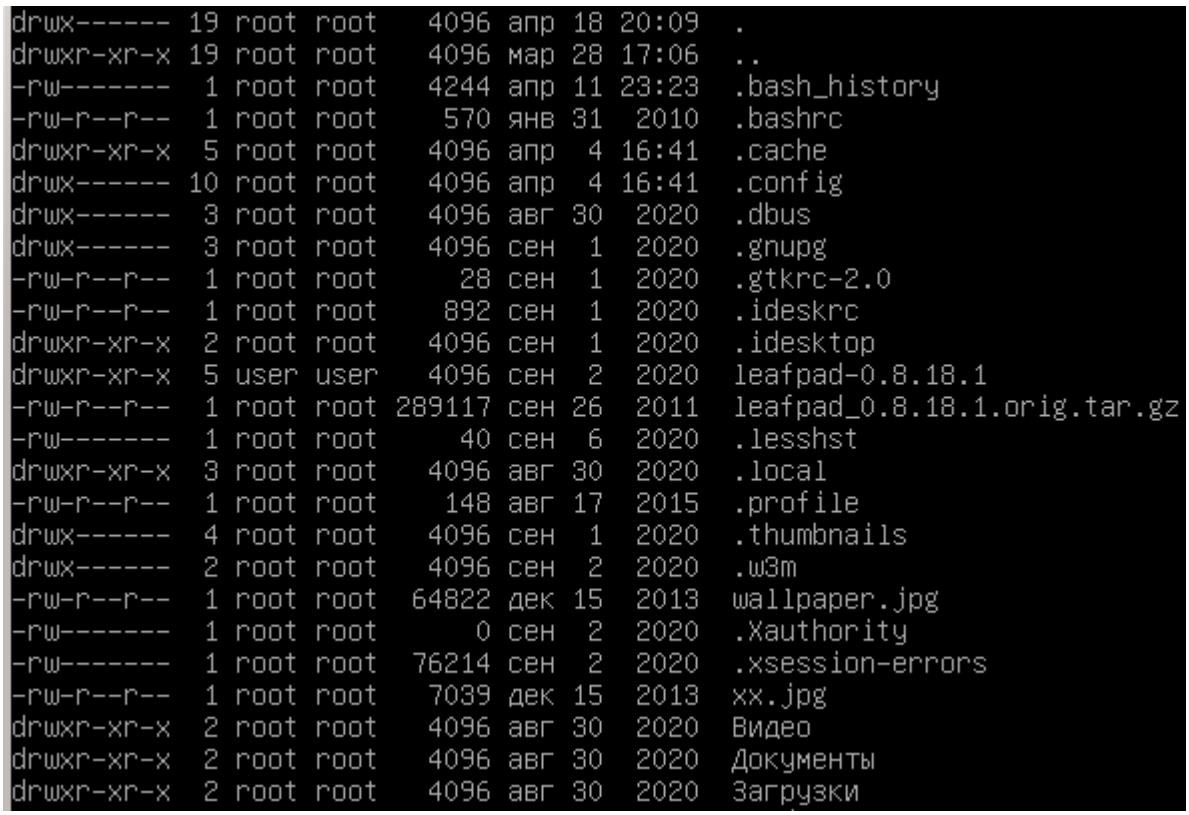
«~» в любой команде обозначает домашний каталог текущего пользователя (не путайте с текущим). Для каждого пользователя имеется свой домашний каталог. Для пользователя «root» это каталог «/root», расположенный в корневом каталоге. Домашние каталоги прочих пользователей, расположены в каталоге «/home» и имеют имена, совпадающие с именем пользователя. Например, «/home/user» - домашний каталог пользователя «user».

Не путайте, «/home» это не домашний каталог, а каталог в котором лежат домашние каталоги пользователей.

При входе в систему текущим становится домашний каталог.

Посмотрим содержимое домашнего каталога, включая скрытые файлы и каталоги:

```
ls -la
```



В отличие от Windows, в Linux у файлов нет специального флага для скрытых файлов. Признаком скрытого файла и каталога является точка в начала имени файла (в том числе знакомые нам «`.`» и «`..`»).

Все программы, сохраняют настройки, связанные с пользователем в домашнем каталоге. И Вы лучше создавайте свои рабочие файлы и каталоги в домашнем каталоге, чтобы потом не искать по всему диску.

Сведём свои знания в одну таблицу:

`/file.txt`	Файл в корневом каталоге
`./file.txt`	Файл в текущем каталоге
`../file.txt`	Файл в родительском каталоге
`~/file.txt`	Файл в домашнем каталоге

Обзор системных каталогов Linux

Выведите список подкаталогов корневого каталога «`ls -l /`»

Следующую таблицу запоминать не надо, но почитать полезно.

/boot	В этом каталоге лежат файлы, необходимые для загрузки ОС (grub, initrd, образ ядра)
/etc	Конфигурационные файлы linux и различных программ
/home	Домашние каталоги простых пользователей
/media	Точки монтирования сменных носителей (флешки, DVD)
/mnt	Временно монтируемые файловые системы (сетевые диски, ...)

/opt	Когда устанавливается несколько версий одной и той же программы, их записывают в этот каталог.
/tmp	Временные файлы. Всё что здесь лежит можно смело удалять, но потом лучше перезагрузиться.
/var	Изменяемые файлы программ
/cache	Кэш приложений. Например, кеш браузера. Всё что здесь лежит можно смело удалять.
/log	Файлы протоколов
/lib	Несмотря на название это не библиотеки, а информация о состоянии программ. В частности, MySQL хранит здесь по умолчанию базу данных.
/spool	Очередь печати и подобное
/dev /proc /run	Виртуальные каталоги и файлы с информацией об устройствах, процессах и подобное
/bin /sbin /lib	Программы и библиотеки одно-пользовательского режима. В Debuan 10 это ссылки на каталоги «/usr/bin», «/usr/sbin»,«/usr/lib»
/usr	Программы и библиотеки много-пользовательского режима.
/bin	Команды и утилиты простых пользователей
/sbin	Команды и утилиты суперпользователя
/lib	Библиотеки
/share	Архитектурно-независимые общие данные для обмена между программами
/src /include	Исходные тексты и заголовочные файлы (для разработки и сборки из исходников)
/local	Программы, библиотеки и данные, специфичные для данного компьютера.

Ссылки

В Windows вы, наверное, встречались со ссылками — это ярлыки программ на рабочем столе. Программы разбросаны по каталогам, в которые их записали инсталляторы, а ссылки на них лежат в одном месте — на рабочем столе. Можно удалить ярлыки программ, которыми Вы редко пользуетесь, программы остаются в свои каталогах и доступны для запуска из меню «Пуск».

В Linux тоже есть ярлыки на рабочем столе, а ещё имеются ссылки, применяемых в разных целях:

1. Если вы часто обращаетесь к какому то файлу или каталогу, указывая в командах длинный путь к нему, можно упростить жизнь, создав ссылку на него. Например:

   `ln /etc/xdg/openbox/autostart`

   В текущем каталоге будет создана ссылка с именем, файла на который ссылаемся.

Если хотите другое имя, или хотите записать ссылку в другой каталог, укажите его вторым параметром.

Вы можете обращаться к файлу и каталогу используя в любой команде ссылку. Например, «`cat autostart`». Причём даже можно создавать ссылку на ссылку (например, на дочерний каталог каталога, к которому обращаемся по ссылке).

2. Для совместимости. Когда в очередной версии Linux или программы разработчик решает переместить некий файл или каталог в другое место, обычно, на старом месте создаётся ссылка (например, «`/bin`» это ссылка на «`/usr/bin`»).
3. Если на один компьютер устанавливается несколько версий программы, обычно создаётся ссылка на одну из них (обычно последнюю версию), но в любой момент можно поменять ссылку. Например,

```
root@debian10:~# ls -l /bin/cpp*
lrwxrwxrwx 1 root root  5 фев 25  2019 /bin/cpp -> cpp-8
lrwxrwxrwx 1 root root 22 апр  6  2019 /bin/cpp-8 -> x86_64-linux-gnu-cpp-8
```

Мы видим, что команда компиляции программ на языке «C++» - «`cpp`» ссылается на компилятор 8 версии, с длиннейшим именем «x86_64-linux-gnu-cpp-8», который лежит в «`/bin/cpp-8`», а точнее «`/usr/bin/cpp-8`» (см п.2).

4. Чтобы переместить большой каталог в другой раздел, где есть свободное место, а менять настройки не хочется. Например, СУБД MySQL, по умолчанию, записывает файлы базы данных в каталог «/var/lib/mysql». Между тем, большинство релизов Linux по умолчанию создаёт небольшой раздел для хранения программ и монтирует его в корень файловой системы, а всё остальное место на диске выделяется второму разделу, который монтируется в каталог «/home». И вот, в какой то момент, в базу данных на сервере внезапно прекращается запись, а ваш телефон разрывается от гневных звонков пользователей, которые не могут выполнить срочную работу. Ведь файлы базы данных оказались в небольшом разделе, куда предполагалось записывать только программы. Поэтому, сразу после установки «MySQL» (и её ответвления «MariaDB») надо дать команды:

```
systemctl stop mysql
mv /var/lib/mysql /home
ln -s /home/mysql /var/lib
systemctl start mysql
```

Команды, «`systemctl stop`» и «`systemctl start`» останавливают и запускают сервис.

Команда «`mv`» перемещает большой каталог данных «`/var/lib/mysql`» в «`/home`»

Команда «`ln`» создаёт ссылку на каталог «/home/mysql» на старом месте (в каталоге «`/var/lib`»).

Обратите внимание на параметр «-s» (Soft link). В Linux имеется 2 типа ссылок:

- soft — «мягкая» ссылка. В каталоге хранится путь к файлу или каталогу. Если Вы удалите файл, на который ссылается мягкая ссылка, то при обращении по ней получите ошибку.
- hard - «жёсткая» ссылка. В каталоге хранится идентификатор файла. В выводе команды «ls -l» выглядит как обычный файл (только во второй колонке вместо 1 будет 2). При удалении файла, на который создана жёсткая ссылка, ссылка продолжает работать. Дело в том, что файлы хранятся не в каталогах, а отдельно. Когда вы создаёте файл, в каталог записывается первая жёсткая ссылка на него. Пока не будут удалены все ссылки файл не удаляется. Мало того, файл не удалится пока он открыт хотя бы в одной программе. Файла нигде нет, а чтение и запись в него продолжается, место на диске не освобождается, пока программу не перезапустите. Чудеса.

По умолчанию команда «`ln`» создаёт жёсткие ссылки. К сожалению, жёсткие ссылки нельзя создать на каталог и на файл из другого раздела. Поэтому не забывайте ставить опцию «-s».

Архивы

Если надо сбросить на флешку, или скопировать по сети или отправить по e-mail сразу несколько файлов, полезно их заархивировать в один файл архива. На запаковку и распаковку архива тратится некоторое время, но обычно это «окупается», поскольку уменьшается объём передаваемых данных, а значит и время. А так же экономится место на флешке.

Запишем в каталог «/storage/test» ещё один файл

```
echo "yyy">/storage/test/file2.txt
```

И заархивируем

```
tar -czvf test.tar.gz /storage/test
```

```
root@debian10:~# tar -czvf test.tar.gz /storage/test
tar: Удаляется начальный `/' из имен объектов
/storage/test/
/storage/test/test.txt
/storage/test/file2.txt
```

Параметр «-czvf» это объединение нескольких однобуквенных опций:

«`-c`» (Create) означает что будет создан новый архив

«`-z`» означает что полученный архив будет сжат программой «g**z**ip». Если не поставить эту опцию, файлы запишутся в архив без сжатия. Архив будет иметь размер даже больше суммарного размера включённых в него файлов. Для чего нужен такой архив? Дело в том, что многие типы файлов практически не сжимаются. Это архивы, изображения (кроме

BMP), видео, аудио файлы они уже сжаты пусть и другими методами. Нет смысл тратить время на попытки их сжать. Архивируйте их без сжатия. Всё равно будет экономия на том, что нет округления размера файлов до размера блока в большую сторону.

«-v» (Verbose) означает что будут выводиться имена архивируемых файлов. Если не поставить эту опцию, команда «молча» отработает без вывода имён файлов. Когда файлы большие, легче ждать когда видно процесс.

«-f» (File) означает что в следующем параметре задано имя архива. Если не задать эту опцию архив будет выводится в «stdout». Эта форма используется в конвейере, когда надо заархивировать каталог и отправить его на другой компьютер (без промежуточной записи архива на диск).

Теперь посмотрите на имя файла «test.tar.gz». Расширение «.tar.gz» принято задавать сжатым архивам. А расширение «.tar» для архивов без сжатия. Однако команда «tar» не контролирует имя файла архива. Если вы забудете поставить опцию «z», получите архив без сжатия, несмотря на расширение.

Далее идёт архивируемый файл или каталог. Можно задать через пробел несколько файлов или каталогов.

Мы задали полный путь к каталогу. В результате, как мы видим, в архив записались полные пути к файлам. При распаковке такого архива, будет создан каталог «storage», в нём каталог «test». Обычно это не неудобно. Запакуем тот же каталог иначе:

```
tar -czvf test.tar.gz -C /storage test
```

```
root@debian10:~# tar -czvf test.tar.gz -C /storage test
test/
test/test.txt
test/file2.txt
```

Следом за опцией «-C» задаётся имя каталога, относительно которого задаются имена архивируемых файлов и каталогов. Если мы архивируем каталог в текущем каталоге (а обычно так и есть) «-C» задавать не потребуется.

Распакуем полученный архив и посмотрим, что получилось

```
tar -xvf test.tar.gz
```

```
root@debian10:~# tar -xvf test.tar.gz
test/
test/test.txt
test/file2.txt
```

```
ls ./test
```

```
file2.txt  test.txt
```

Заархивированный каталог «test» распаковался в текущий каталог.

Если требуется разархивировать в другой каталог, используйте опцию «-C».

Поиск

Поиск файлов

Случается так, что Вы помните, что создавали файл, но не можете его найти.

Предположим, что вы помните имя файла.

```
find ./ -name test.txt
./test/test.txt
```

«`./`» означает что мы ищем в текущем каталоге и его подкаталогах

«`-name` файл» ищем файл по имени

Если помните только часть имени, можно использовать «*»

```
root@debian10:~# find ./ -name 'test*'
./test.tar.gz
./test
./test/test.txt
```

Но часто бывает, что забыли не только куда файл положили, но и его имя. Помнится только, что создали его примерно неделю назад. Тут нам поможет поиск по времени последнего изменения:

```
root@debian10:~# find ./ -mtime -7
./
./.local/share/mc
./.local/share/mc/history
./.bash_history
./.lesshst
./.cache/mc
./.cache/mc/Tree
./.selected_editor
./test.tar.gz
./test
./test/file2.txt
```

«-7» означает, что надо искать файлы, изменённые 7*24 часа назад и менее

«+7» означает, что надо искать файлы, изменённые 7*24 часа назад и более

А ещё, команду «find» с опцией «-exec» удобно использовать в ежедневно исполняемом сценарии для удаления старых файлов протокола или архивных копий базы данных, например, «`find /backup -name "*.sql.gz" -mtime +7 -exec rm -f {} \;`». Для каждого найденного файла выполняется команда, заданная после «-exec». Вместо «{}» в команду подставляется полный путь к файлу. Не забудьте «`\;`» в конце команды.

Поиск исполняемого файла

Все команды «linux» это исполняемые файлы. Мы привыкли просто указывать имя команды. Интерпретатор командной строки ищет выполняемый файл в каталогах, перечисленных в переменной окружения «PATH» и запускает его. Но бывает, требуется узнать полный путь к файлу.

В сценариях или «crontab» рекомендуется указывать полный путь к исполняемому файлу, чтобы не зависеть от переменных окружения.

А ещё, бывает что от имени одного пользователя программа работает, а от другого нет. Чаще всего дело в отсутствии прав доступа к какому то файлу. Но бывает, дело в том, что установлены 2 версии программы, а у одного пользователя в «PATH» оказался задан каталог с одной версией программы, у у другого другой.

Нам поможет команда «`wich`». Дайте команду:

```
which tar
```

```
root@debian10:~# which tar
/usr/bin/tar
```

В отличие от «find» эта команда выполняется почти мгновенно, потому что команде требуется просмотреть всего несколько каталогов, перечисленных в переменной окружения «PATH».

Поиск файла по содержимому

Бывает надо найти все текстовые файлы, содержащие заданную подстроку. Это может быть путь к каталогу или имя пользователя или имя функции (при поиске в исходных текстах программы), и много ещё чего.

Для поиска текстовых файлов, содержащих заданную подстроку можно использовать уже знакомую нам команду «`grep`».

```
root@debian10:~# grep 'xxx' -r --color ./
./.config/user-dirs.dirs:# Format is XDG_xxx_DIR="$HOME/yyy", where yyy is a shell-escaped
./.config/user-dirs.dirs:# homedir-relative path, or XDG_xxx_DIR="/yyy", where /yyy is an
./.bash_history:/etc/init.d/samba-ad-dc xxx
./.bash_history:echo "xxxx" >/storage/test.txt
./leafpad-0.8.18.1/ABOUT-NLS:`LC_xxx' environment variables set, you should unset them before
./test/test.txt:xxxx
```

В апострофах я указал искомую строку.

«`-r`» (Recursive) означает рекурсивный поиск в заданном каталоге и его подкаталогах.

«`--color`» подсветит красным цветом заданную подстроку.

«`./`» - поиск проводим в текущем каталоге и его подкаталогах.

Контрольные вопросы

В этом разделе приведены типичные вопросы, задаваемые на защите лабораторной работы. Ответы в отчёт включать не надо.

- Пусть Вы собираетесь создать на диске раздел размером 3Тб. Какую таблицу разделов надо создать MBR или GPT?
- Напишите команду монтирования раздела «/dev/sdb1» в каталог «/mnt/UEFI»
- Почему в «/etc/fstab» и других конфигурационных файлах рекомендуется использовать идентификаторы разделов?

- Почему файл, в который мы записали 4 символа занимает на диске 4К?
- На флешке 99Кб свободного места. Уместятся ли на ней 200 файлов размером по 100 байт каждый?
- В какой каталог мы перейдём командой «`cd /`»
- В какой каталог мы перейдём командой «cd ~»?
- В каком каталоге находится домашний каталог пользователя «user»?
- В каком каталоге находится домашний каталог пользователя «root»?
- В какой каталог перейдёт пользователь «user» командой «`cd ~/..`»?
- Как отличить скрытые файлы и каталоги?
- Предположим, вы создали жесткую и мягкую ссылку на файл, а потом удалили этот файл. Какая ссылка будет продолжать работать? Почему?
- Жёсткую или мягкую ссылку вы будете использовать для ссылки на каталог?
- Почему даже архив без сжатия может занимать на диске меньше места, чем входящие в него файлы?
- Что означает параметр -c в команде архивирования «tar -cvzf archive.tar.gz file1 file2»?
- Что выведет на экран команда «tar -czf archive.tar.gz file1 file2»?
- Что означает параметр -z в команде архивирования «tar -cvzf archive.tar.gz file1 file2»?
- Куда запишет архив команда «tar -c archive.tar.gz file1 file2»?
- В каком каталоге ищет файл команда «find -name xx.txt»?

Приложения

Текст этого раздела представляет собой перевод фрагментов справки из команды «man», относящихся к материалу данного учебного пособия. Данный материал даёт расширенную информацию по рассмотренным командам, а так же может использоваться для выполнения заданий, приведённых в тексте.

В описании команд в квадратных скобках указываются опции (необязательные параметры.)

Из описания команд исключены общие опции:

`--version` — Вывести и выйти версию и выйти

`--help` — Показать текст справки и выйти

fdisk

```
fdisk [опции] [устройство]
fdisk -L устройство [устройство2] ...
```

Утилита «fdisk» это программа с диалоговым управлением для создания таблиц разделов и управления ими. Он понимает таблицы разделов GPT, MBR, Sun, SGI и BSD.

Блочные устройства можно разделить на один или несколько логических дисков, называемых разделами. Это деление записывается в таблицу разделов, обычно находящуюся в секторе 0 диска.

По умолчанию все разбиения ограничены заданным устройством внешней памяти.

fdisk может оптимизировать структуру диска для размера блока 4K и использовать выравнивание смещения на современных устройствах для MBR и GPT. Рекомендуется следовать настройкам fdisk по умолчанию, поскольку значения по умолчанию (например, первый и последний секторы раздела) и размеры разделов, указанные в нотации +/- <размер> {M, G, ...}, всегда выравниваются в соответствии со свойствами устройства.

Параметры:

Устройство - обычно / dev/sda, /dev/sdb ... Имя устройства относится ко всему диску. Старые системы без libata учитывают различия между дисками IDE и SCSI. В таких случаях имя устройства будет /dev/hd * (IDE) или /dev/sd * (SCSI).

Раздел - это имя устройства, за которым следует номер раздела. Например, /dev/sda1 - это первый раздел на первом жестком диске в системе.

Основные опции:

«`-l, --list`» — Вывести таблицы разделов для заданных устройств и выйти.

Если устройства не указаны, используются те, которые указаны в «/proc/partitions» (если этот файл существует).

«`-b, --sector-size sectorsize`» - Задаёт размер сектора диска. Допустимые значения: 512, 1024, 2048 и 4096. (Последние ядра знают размер сектора. Используйте эту опцию только в старых ядрах или для переопределения идей ядра.)

Команды диалогового режима:

Создание таблицы разделов

g — Создать пустую таблицу разделов типа GPT

G — Создать пустую таблицу разделов типа SGI (IRIX)

o — Создать пустую таблицу разделов типа DOS(MBR)

s — Создать пустую таблицу разделов типа Sun

GPT

M — Переключиться в режим защищённой / гибридной MBR

DOS (MBR)

a — установить / снять признак загрузочного раздела

b — редактировать вложенную метку диска BSD

c — установить / снять признак совместимости с DOS

Общие

d — удалить раздел

F — перечислить фрагменты свободного, неразделенного пространства

l — вывести известные типы разделов

n — добавить новый раздел

p — вывести таблицу разделов

t — изменить тип раздела

v — проверить таблицу разделов

i — вывести информацию о разделе

Разное

m — вывести меню команд диалогового режима

u — изменить единицы при отображении / вводе размеров и позиций разделов

x — дополнительная функциональность (только для экспертов)

Сценарий

I — загрузить макет диска из файла сценария

O — записать макет диска в файл сценария

Сохранение и выход

w — записать таблицу на диск и выйти

q — выйти без сохранения изменений

lsblk

`lsblk [опции] [устройство...]`

lsblk выводит информацию обо всех доступных или указанных блочных устройствах. По умолчанию команда печатает все блочные устройства (кроме RAM-дисков) в древовидном формате.

Формат вывода информации может быть изменён. Поэтому, когда это возможно, вам следует избегать использования выходных данных по умолчанию в ваших скриптах. Всегда явно определяйте ожидаемые столбцы с помощью --output columns-list в средах, где требуется стабильный вывод.

Основные опции:

`-a, --all` — выводит также пустые устройства и RAM-диски (виртуальные диски, расположенные в оперативной памяти)

`-b, --bytes` — выводит столбец SIZE в байтах, а не в удобочитаемом формате.

`-d, --nodeps` — Не выводит сменные носители и вторичные устройства. Например, «`lsblk --nodeps /dev/sda`» выводит информацию только об устройстве sda (без разделов)

`-e, --exclude устройства` — Исключает устройства, перечисленные через запятую. Обратите внимание, что RAM-диски (major = 1) по умолчанию исключаются, если --all не указано. Фильтр применяется только к устройствам верхнего уровня. Это может сбивать с толку формат вывода --list, где иерархия устройств неочевидна.

`-f, --fs` — Вывод информации о файловых системах. Эта опция эквивалентна «`-o NAME,FSTYPE,LABEL,UUID,MOUNTPOINT`».

`-h, --help` — Показать текст справки и выйти.

`-I, --include устройства` — Включает в выводимый список указанные устройства, перечисленные через запятую. Фильтр применяется только к устройствам верхнего уровня. Это может сбивать с толку для формата вывода --list, где иерархия устройств неочевидна.

`-i, --ascii` —Использует символы ASCII для форматирования дерева вместо псевдографики.

`-J, --json` — Выводить информацию в формате JSON.

`-l, --list` — Выводить информацию в форме списка. По умолчанию в форме дерева.

`-m, --perms` — Выводить информации о владельце устройства, группе и режиме. Эта опция эквивалентна «`-o NAME,SIZE,OWNER,GROUP,MODE`».

`-n, --noheadings` — Не выводить строку заголовка таблицы.

`-o, --output колонки` — Задаёт (через запятую) список выводимых колонок. Имена колонок приведены ниже в разделе «Имена колонок».

`-O, --output-all` — Выводить все доступные коллонки. Имена колонок приведены ниже в разделе «Имена колонок».

`-P, --pairs` — Выводить данные в виде пар ключ = "значение". Все потенциально опасные символы экранированы шестнадцатеричным кодом (\ x <code>).

`-p, --paths` — Выводить полные пути устройств.

`-r, --raw` — Производить вывод в необработанном формате. Все потенциально небезопасные символы в столбцах NAME, KNAME, LABEL, PARTLABEL и MOUNTPOINT экранированы шестнадцатеричным кодом (\ x <code>).

`-S, --scsi` — Выводить информацию только об устройствах SCSI. Все разделы, ведомые устройства и сменные носители игнорируются.

`-s, --inverse` — Выводить зависимости в обратном порядке. Если запрашивается вывод --list, то строки по-прежнему упорядочиваются по зависимостям.

`-t, --topology` — Выводить информацию о топологии блочного устройства. Эта

опция эквивалентна «-o NAME, ALIGNMENT, MIN-IO, OPT-IO, PHY-SEC, LOG-SEC, ROTA, SCHED, RQ-SIZE, RA, WSAME».

`-V, --version` — Вывести информацию о версии и выйти.

`-x, --sort column` — Сортировать выходные строки по столбцу. Эта опция по умолчанию включает выходной формат --list. Можно использовать параметр --tree для принудительного вывода в виде дерева, а затем ветви дерева сортируются по столбцу. Имена колонок приведены ниже в разделе «Имена колонок».

Имена колонок:

NAME имя устройства

KNAME внутреннее имя устройства а уровне ядра linux

PATH путь к устройству

MAJ:MIN старший:младший номер устройства

FSAVAIL доступный размер для файловой системы

FSSIZE Размер файловой системы

FSTYPE Тип файловой системы

FSUSED Размер используемой части файловой системы

FSUSE% Процент использования файловой системы

MOUNTPOINT точка монтирования файловой системы

LABEL метка файловой системы

UUID - UUID файловой системы

PTUUID Идентификатор таблицы разделов (обычно UUID)

PTTYPE Тип таблицы разделов

PARTTYPE тип раздела

PARTLABEL метка раздела

PARTUUID UUID раздела

PARTFLAGS флаги раздела

RA с опережающим чтением устройства

RO устройство только для чтения

RM Съемное устройство

HOTPLUG Съемное или горячее подключение (usb, pcmcia, ...)

MODEL идентификатор устройства

SERIAL серийный номер диска

SIZE размер устройства

STATE состояние устройства

OWNER Имя пользователя

GROUP Название группы

MODE права доступа

ALIGNMENT смещение выравнивания

MIN-IO минимальный размер ввода / вывода

OPT-IO Оптимальный размер ввода / вывода

PHY-SEC Размер физического сектора

LOG-SEC Размер логического сектора

ROTA Ротационное устройство

SCHED имя планировщика ввода / вывода

RQ-SIZE Размер очереди запросов

TYPE тип устройства

WWN Уникальный идентификатор хранилища

PKNAME внутреннее имя родительского устройства ядра

TRAN Транспортный тип устройства

REV Версия устройства

VENDOR производитель устройства

mkfs

`mkfs [options] [-t type] [fs-options] device [size]`

mkfs используется для создания файловой системы типа «`type`» на устройстве «`device`». Код выхода, возвращаемый mkfs, равен 0 в случае успеха и 1 в случае неудачи.

Данная утилита устарела. Рекомендуется использовать mkfs.ext4, mkfs.vfat и другие специфичные для разных файловых систем утилиты, которые всё равно вызывает данная утилита.

Параметры:

«`device`» - раздел жесткого диска (например, /dev/sdb2), либо обычный файл, который должен содержать файловую систему.

«`fs-options`» - опции, характерные для файловой системы

«`size`» - это количество блоков, которые будут использоваться для файловой системы.

mount

Все файлы, доступные в системе Unix, организованы в одно большое дерево, файловую иерархию с корнем /. Эти файлы можно разложить на несколько устройств. Команда mount служит для присоединения файловой системы, найденной на каком-либо устройстве, к большому файловому дереву. И наоборот, команда umount отсоединит его

снова. Файловая система используется для управления тем, как данные хранятся на устройстве или предоставляются виртуальным способом сетью или другими службами.

Получение списка монтирования

```
mount [-l] [-t type]
```

Данная форма команды выводит список всех смонтированных файловых систем (типа type)

Монтирование устройства

```
mount [-t type] device dir
```

Говорит ядру прикрепить файловую систему, найденную на устройстве «device» типа «type», в каталог «dir». Опция -t не является обязательной. Команда mount обычно может определить тип файловой системы. По умолчанию для монтирования файловой системы необходимы права root. Предыдущее содержимое каталога (если есть), а также владелец и режим dir становятся невидимыми, и пока эта файловая система остается смонтированной, путь dir соответствует корню файловой системы на устройстве.

Если указан только каталог или только устройство, например, «mount /dir», mount ищет точку монтирования (и, если она не найдена, то устройство) в файле «/etc/fstab». Можно использовать параметры --target или --source, чтобы избежать неоднозначной интерпретации данного аргумента. Например, «mount --target /mountpoint»

Одна и та же файловая система может быть смонтирована более одного раза, а в некоторых случаях (например, сетевые файловые системы) одна и та же файловая система может быть смонтирована на одной и той же точке монтирования несколько раз. Команда mount не реализует никакой политики для управления этим поведением. Все поведение контролируется ядром и обычно зависит от драйвера файловой системы. Исключением является --all, в этом случае уже смонтированные файловые системы игнорируются (подробнее см. --All ниже).

Задание устройства

«device» - Большинство устройств обозначаются именем файла (блочного специального устройства), например «/dev/sda1», но есть и другие возможности. Например, в случае монтирования NFS устройство может выглядеть как «knuth.cwi.nl:/dir». Также возможно указать специальное блочное устройство, используя его метку файловой системы или UUID (см. Параметры -L и -U ниже).

Имя устройства дисковых разделов нестабильно; реконфигурация оборудования, добавление или удаление устройства может вызвать изменение имен. По этой причине настоятельно рекомендуется использовать идентификаторы файловой системы или разделов, такие как UUID или LABEL.

Команда lsblk --fs обеспечивает обзор файловых систем, меток и UUID на доступных

блочных устройствах. Команда blkid -p <device> предоставляет подробную информацию о файловой системе на указанном устройстве.

Не забывайте, что нет никакой гарантии, что UUID и метки действительно уникальны, особенно если вы перемещаете, совместно используете или копируете устройство. Используйте «`lsblk -o +UUID,PARTUUID`», чтобы убедиться, что UUID действительно уникальны в вашей системе.

Рекомендуется использовать теги (например, UUID = uuid), а не /dev/disk/by-{label, uuid, partuuid, partlabel} символические ссылки udev в файле /etc/fstab. Теги более удобочитаемы, надежны и переносимы. Команда mount внутренне использует символические ссылки udev, поэтому использование символических ссылок в /etc/fstab не имеет преимущества перед тегами.

Монтирование от имени обычных пользователей

Обычно монтировать файловые системы может только суперпользователь. Однако, когда fstab содержит параметр «user» в некой строке, любой может смонтировать соответствующую файловую систему. Например, если задана строка:

```
/dev/cdrom  /cd  iso9660  ro,user,noauto,unhide
```

Тогда любой пользователь может смонтировать файловую систему iso9660, найденную на вставленном CDROM, с помощью команды:

```
mount /cd
```

Только пользователь, смонтировавший файловую систему, может размонтировать ее снова. Если другой пользователь должен иметь возможность размонтировать его, тогда используйте users вместо user в строке fstab. Параметр «owner» аналогичен параметру «user» с тем ограничением, что пользователь должен быть владельцем специального файла. Это может быть полезно, например, для «/dev/fd», если сценарий входа делает пользователя консоли владельцем этого устройства. Опция «group» аналогична, с ограничением, что пользователь должен быть членом группы специального файла.

Привязка монтирования

Можно перемонтировать часть файловой иерархии в другое место командой:

```
mount --bind olddir newdir
```

или используя запись fstab:

```
/olddir /newdir none bind
```

После этого вызова одно и то же содержимое доступно в двух местах.

Важно понимать, что «привязка» не предназначена для создания какого-либо второсортного или специального узла в ядре VFS. «Связывание» - это просто еще одна операция по присоединению файловой системы. Нигде не хранится информация о том, что файловая система была прикреплена операцией "привязки". Olddir и newdir независимы, и

olddir может быть размонтирован.

Также можно перемонтировать один файл (в один файл). Также можно использовать привязку монтирования для создания точки монтирования из обычного каталога, например:

```
mount --bind foo foo
```

Вызов bind mount подключает только (часть) единой файловой системы, без возможности субмонтирования. Вся файловая иерархия, включая субмаунты, прикрепляется на втором месте с помощью:

```
mount --rbind olddir newdir
```

Обратите внимание, что параметры монтирования файловой системы, поддерживаемые ядром, останутся такими же, как и в исходной точке монтирования. Параметры монтирования пользовательского пространства (например, _netdev) не будут скопированы с помощью mount, необходимо явно указать параметры в командной строке монтирования.

mount позволяет изменять параметры монтирования, передавая соответствующие параметры вместе с --bind. Например:

```
mount -o bind,ro foo foo
```

Это решение не атомарно.

Альтернативный (классический) способ создания монтирования привязки только для чтения - использовать операцию повторного монтирования, например:

```
mount --bind olddir newdir
mount -o remount,bind,ro olddir newdir
```

Обратите внимание, что привязка только для чтения создаст точку монтирования только для чтения (запись VFS), но исходный суперблок файловой системы по-прежнему будет доступен для записи, что означает, что olddir будет доступен для записи, а newdir будет доступен только для чтения.

Также возможно изменить флаги записи VFS nosuid, nodev, noexec, noatime, nodiratime и relatime с помощью операции «remount, bind». Другой (например, специфические флаги файловой системы) игнорируется. Рекурсивно изменить параметры монтирования (например, с помощью -o rbind, ro) невозможно.

mount, поскольку v2.31 игнорирует флаг привязки из / etc / fstab при операции повторного монтирования (если в командной строке указано «-o remount»). Это необходимо для полного управления параметрами монтирования при повторном подключении из командной строки. В предыдущих версиях всегда применялся флаг привязки, и было невозможно повторно определить параметры монтирования без взаимодействия с семантикой привязки. Такое поведение mount не влияет на ситуации, когда в файле /etc/fstab указано «remount, bind».

Операция перемещения

Можно переместить смонтированное дерево в другое место (атомарно) командой:

```
mount --move olddir newdir
```

Это приведет к тому, что содержимое, которое раньше отображалось в olddir, теперь будет доступно в newdir. Физическое расположение файлов не изменилось. Обратите внимание, что olddir должен быть точкой монтирования.

Также обратите внимание, что перемещение монтирования, находящегося под общим монтированием, недопустимо и не поддерживается. Используйте findmnt -o TARGET, PROPAGATION, чтобы увидеть текущие флаги распространения.

Общее поддерево

Начиная с Linux 2.6.15, можно пометить монтирование и его вспомогательные монтирования как общие, частные, подчиненные или несвязываемые. Совместное крепление дает возможность создавать зеркала этого крепления, при которых операции монтирования и размонтирования внутри любого из зеркал распространяются на другое зеркало. Подчиненное устройство получает распространение от своего мастера, но не наоборот. Частное монтирование не обладает способностями к распространению. Несвязываемое монтирование - это частное монтирование, которое нельзя клонировать с помощью операции привязки. Подробная семантика задокументирована в файле Documentation / filesystems / sharedsubtree.txt в дереве исходного кода ядра.

Поддерживаемые операции:

```
mount --make-shared mountpoint
mount --make-slave mountpoint
mount --make-private mountpoint
mount --make-unbindable mountpoint
```

Следующие команды позволяют рекурсивно изменять тип всех монтирований в данной точке монтирования.

```
mount --make-rshared mountpoint
mount --make-rslave mountpoint
mount --make-rprivate mountpoint
mount --make-runbindable mountpoint
```

mount не читает fstab, когда запрашивается операция --make- *. Вся необходимая информация должна быть указана в командной строке.

Обратите внимание, что ядро Linux не позволяет изменять несколько флагов распространения с помощью одного системного вызова mount, и эти флаги нельзя смешивать с другими параметрами и операциями монтирования.

Пожалуй, этого достаточно для ознакомления с возможностями команды «mount». Опции и параметры этих опций, возвращаемые коды ошибок вы можете посмотреть командой «`man mount`»

df

`df [опции]`

Выводит отчет об использовании дискового пространства файловой системы

Опции:

`-a, --all` — включать псевдо, дубликаты, недоступные файловые системы

`-B, --block-size=SIZE` — выводить размеры в заданной размерности (K,M,G,T,P,E,Z,Y)

`-h, --human-readable` — выводить размеры в человеко-читаемой форме, с суффиксом (K,M,G,T,P,E,Z,Y), в зависимости от размера, по основанию 1024

`-H, --si` — выводить размеры печати по основанию 1000 (а не 1024)

`-i, --inodes` — выводить информацию об индексных дескрипторах вместо размеров в блоках

`-k` — Выводить размеры в килобайтах

`-l, --local` — ограничить листинг локальными файловыми системами

`--output[=FIELD_LIST]` — выводит колонки, перечисленные в FIELD_LIST

`--total` — выводит строку с общей суммой свободного пространства на всех дисках

`-t, --type=TYPE` — вывести только файловые системы типа TYPE

`-T, --print-type` — выводить тип файловой системы

`-x, --exclude-type=TYPE` — вывести файловые системы кроме типа TYPE

du

`du [опции] [ФАЙЛ] [КАТАЛОГ]...`

Выводит использование диска набором файлов и каталогов (рекурсивно).

Опции:

`-B, --block-size=SIZE` — вывести размеры в заданном масштабе. SIZE это единица измерения (K,M,G,T,P,E,Z,Y) или целое число + единица измерения . Например, «K» - размеры выводятся в килобайтах, «10K» размеры выводятся в «блоках» по 10*1024 байт.

`-a, --all` — выводить размеры не только каталогов, но и файлов

`-c, --total` — вывести строку «итого»

`-D, --dereference-args` — выводить каталоги, соответствующие символическим ссылкам, перечисленным в командной строке

`-d, --max-depth=N` — вывести итоговые суммы для каталогов (или файлов с опцией --all), только если она находится на N или менее уровней ниже аргумента командной

строки; --max-depth = 0 то же самое, что --summarize

`--files0-from=FILE` — суммировать использование диска файлами и каталогами, перечисленными в заданном файле (с NUL в конце каждого имени). Если «--», то считывать имена со стандартного потока ввода

`-H` — эквивалент --dereference-args (-D)

`-h, --human-readable` — размеры выводятся в удобочитаемом формате (например, 1K 234M 2G)

`--inodes` — вывести информацию об использовании индексных дескрипторов вместо размера в блоках

`-k` — эквивалент --block-size=K

`-L, --dereference` — раскрывать все символические ссылки

`-m` — эквивалент --block-size=M

`-P, --no-dereference` — не раскрывать по символическим ссылкам (по умолчанию)

`-S, --separate-dirs` — для каталогов не включать размер подкаталогов

`--si` — подобна -h, но использовать степень 1000, а не 1024

`-s, --summarize` — отображать только сумму для каждого файла и каталога, перечисленного в командной строке

`-t, --threshold=SIZE` — исключить записи меньше SIZE, если SIZE положительное, или записи больше SIZE, если оно отрицательное

`--time=WORD` — показывать вместо времени модификации, atime, access, use, ctime или status

`--time-style=STYLE` — показывать время в формате «STYLE» (full-iso, long-iso, iso, или +FORMAT (как в команде «date»)).

`--exclude=PATTERN` — исключить файлы, соответствующие заданному шаблону

`-X, --exclude-from=FILE` — исключить файлы и каталоге, соответствующие шаблонам, перечисленным в файле «`FILE`»

mkdir

`mkdir [опции] DIR [DIR2]...`

Создаёт каталоги DIR, DIR2..., если они не существуют.

Опции:

`-m, --mode=MODE` — устанавливает флаги прав доступа (как в команде chmod)

`-p, --parents` — Создаёт недостающие родительские каталоги для создаваемых каталогов. Если создаваемый каталог уже существует, ошибка не выдаётся. Рекомендуется

использовать в сценариях.

`-v, --verbose` — Выдаёт сообщение для каждого создаваемого каталога. По умолчанию, при отсутствии ошибко сообщений не выдаётся.

--context[=CTX] — устанавливает контекст безопасности SELinux равный «CTX» для создаваемых каталогов. Если «CTX» не задан, устанавливает контекст безопасности «по умолчанию» (как -Z)

-Z — устанавливает контекст безопасности SELinux «по умолчанию» для каждого создаваемого каталога

cd

```
cd [-L|[-P [-e]] [-@]] [DIR]
```

Изменяет текущий каталог на DIR. Если DIR не задано, использует значение переменной оболочки «HOME» (домашний каталог пользователя).

Опции:

`-L` — принудительно использовать символические ссылки: в текущий путь подставляются символические ссылки, а не каталог им соответствующий (после подстановки «`..`»). Этот режим используется по умолчанию.

`-P` — использовать физическую структуру каталогов без следующих символических ссылок: в текущий путь подставляются не символические ссылки, а соответствующий им каталог.

`-e` — если указана опция -P и текущий рабочий каталог не может быть определен успешно, выйти с ненулевым статусом

`-@` — в системах, которые его поддерживают, представить файл с расширенными атрибутами как каталог, содержащий атрибуты файла

«`..`» обрабатывается путем удаления непосредственно предыдущего компонента из текущего пути.

ls

```
ls [опции] [каталог1] [каталог2] [файл]...
```

Отображает информацию о заданных файлах и содержимое заданных каталогов. Если файлы и каталоги не заданы, отображает содержимое текущего каталога.

Опции:

`-a, --all` — не пропускать скрытые файлы и каталоги. Скрытыми считаются файлы, имя которых начинается с точки.

`-A, --almost-all` — не отображать каталоги «`.`» и «`..`»

`--author` — с опцией -l выводит автора каждого файла.

`-b, --escape` — печатать служебные символы в стиле языка «C» (кроме графических символов)

`--block-size=SIZE` — с опцией -l выводит размеры в масштабе SIZE. SIZE это единица измерения (K,M,G,T,P,E,Z,Y) или целое число + единица измерения . Например, «K» - размеры выводятся в килобайтах, «10K» - размеры выводятся в «блоках» по 10*1024 байт.

`-c` — с опцией «`-lt`»: сортировать и показывать ctime (время последнего изменения информации о статусе файла); с опцией «`-l`»: показать ctime и сортировать по имени; в противном случае: сортировать по ctime (сначала самые новые)

`-C` — список имён по столбцам (отменяет «`-l`»)

`--color[=WHEN]` — раскрасить в соответствии с типом записей. «`WHEN`» может иметь значения «always» - (по умолчанию), «auto», «never»

`-d, --directory` — вывести сами каталоги, а не их содержимое.

`-f` — не сортировать, включить -aU, отключить -ls --color

`-F, --classify` — выводить индикатор типа записей: «`*`» - исполняепмый файл, «`/`» - каталог, «=» - сокет, «@» - символьная ссылка.

`--file-type` — аналогично, но не добавлять «*»

`--format=WORD` — формат вывода:

«across» соответствует «`-x`»,

«commas» соответствует «`-m`»,

«horizontal» соответствует «`-x`»,

«long» соответствует «`-l`»,

«single-column» соответствует «`-1`» (единица),

«verbose» соответствует «`-l`»,

«vertical» соответствует «`-C`»

`--full-time` — с опцией -l выводит время в формате «full-iso» - «ГГГГ-ММ-ДД чч:мм:сс.milliseconds»

`-g` — аналогично -l, но не указывать владельца

`--group-directories-first` — группировать каталоги перед файлами. Можно дополнить опцией --sort, но использование --sort = none (-U) отключает группировку

`-G, --no-group` — с опцией -l не выводит группу-владельца.

`-h, --human-readable` — с опциями -l и -s, выводит размеры в человеко-читаемом формате (1K 234M 2G...)

--si — аналогично, но использует множитель 1000, а не 1024

-H, --dereference-command-line — переходит по символическим ссылкам, указанным в командной строке, если они указывают каталоги.

--hide=ШАБЛОН — не выводить записи, соответствующие ШАБЛОНУ (отменяется -a или -A)

--indicator-style=WORD — выводить индикаторы типа записей: «none» (по умолчанию), «slash» (соответствует -p), «file-type» (соответствует --file-type), «classify» (соответствует -F)

-i, --inode — распечатать номер индексного дескриптора (идентификатор) каждого файла.

-I, --ignore=ШАБЛОН — не выводить записи, соответствующие ШАБЛОНУ.

-l — использовать длинный (построчный) формат списка файлов

-L, --dereference — для символьной ссылки показывать информацию о файле, на который ссылается ссылка, а не о самой ссылке

-m — выводить имена файлов построчно, через запятую

-n, --numeric-uid-gid — подобна «-l», но перечисляет числовые идентификаторы пользователей и групп а не их имена

-N, --literal — выводить имена файлов и каталогов без кавычек

-o — подобна «-l», но не выводит информацию о группе

-p, --indicator-style=slash — добавить индикатор «/» к именам каталогов

-q, --hide-control-chars — выводить «?» вместо неграфических символов

--show-control-chars — отображать неграфические символы как есть

-Q, --quote-name — заключает имена записей в двойные кавычки

-r, --reverse — обратный порядок сортировки

-R, --recursive — рекурсивно перечислить подкаталоги

-s, --size — выводить выделенный на диске объём для каждого файла в блоках (по умолчанию килобайтах)

-S — сортировать по размеру файла (по убыванию)

--sort=WORD — сортировать по: «none» (соответствует опции -U), «size» (соответствует опции -S), «time» (соответствует опции -t), «version» (соответствует опции -v), «extension» (соответствует опции -X)

--time=WORD — с опцией «-l», выводит в колонке времени:

«atime» или «access» или «use» - соответствует «-u»;

«ctime» или «status» - соответствует «-c»;

по умолчанию — выводится время последней модификации

Это же время используется для сортировки.

`-t` — сортировать по времени модификации, сначала самые новые

`-u` — с опцией «`-lt`»: сортировать и показывать время доступа; с «`-l`»: показать время доступа и сортировать по имени; в противном случае: сортировать по времени доступа (сначала самые новые)

`-U` — не сортировать записи

`-w, --width=COLS` — при выводе в формате колонок предполагать ширину терминала равную «COLS» символов. По умолчанию 0 (неограничено).

`-x` — выводить имена файлов и каталогов по колонкам

`-X` — сортировать в алфавитном порядке по расширениям файлов

`-Z, --context` — выводить контекст безопасности SELINUX каждого файла

`-1` (единица) — список по одному файлу в строке.

cp

`cp [опции] ИСТОЧНИКИ ЦЕЛЬ`

Копирует один или несколько файлов или каталогов в целевой каталог или целевой файл с заданным именем.

Опции:

`a, --archive` — соответствует -dR --preserve=all

`--attributes-only` — не копирует данные файла, только атрибуты

`--copy-contents` — копировать содержимое специальных файлов при рекурсии

`-d` — то же, что и --no-dereference --preserve=links

`-f, --force` — если существующий целевой файл не может быть открыт, удалит его и повторит попытку (этот параметр игнорируется, если также используется параметр -n)

`-i, --interactive` — запрашивать перед перезаписью (отменяет предыдущую опцию -n)

`-H` — переходить по символическим ссылкам, заданным в параметре ИСТОЧНИК

`-l, --link` — создаются жесткие ссылки вместо копирования файлов

`-L, --dereference` — всегда переходить по символическим ссылкам, заданным в параметре «ИСТОЧНИК»

`-n, --no-clobber` — не перезаписывать существующий файл (отменяет предыдущую опцию -i)

`-P, --no-dereference` — никогда не переходите по символическим ссылкам, заданным в параметре «ИСТОЧНИК»

`-p` — то же, что и --preserve=mode,ownership,timestamps

`--preserve[=ATTR_LIST]` — сохранить указанные атрибуты (по умолчанию: режим, владельцы, временные метки). Дополнительные атрибуты: context, links, xattr, all

`--no-preserve=ATTR_LIST` — не сохранять указанные атрибуты

`-R, -r, --recursive` — копировать каталоги рекурсивно

`--remove-destination` — удалить каждый существующий файл назначения перед попыткой его открытия (в отличие от --force)

`--strip-trailing-slashes` — удалить завершающие косые черты из каждого аргумента ИСТОЧНИК

`-s, --symbolic-link` — создавать символические ссылки вместо копирования

`-T, --no-target-directory` — рассматривать ЦЕЛЬ как обычный файл

`-u, --update` — копировать только в том случае, если файл ИСТОЧНИК новее, чем файл назначения, или когда файл назначения отсутствует

`-v, --verbose` — выводить, что делается

`-Z` — Установить в контекст безопасности SELinux значение по умолчанию для целевых файлов

`--context=CTX` — Установить контекст безопасности SELinux равный CTX для целевых файлов

mv

`mv [опции] ИСТОЧНИКИ ЦЕЛЬ`

Переименовывает ИСТОЧНИК в ЦЕЛЬ или перемещает один или несколько файлов или каталогов в целевой каталог.

Опции:

`-f, --force` — не запрашивать подтверждение перед перезаписью

`-i, --interactive` — запрашивать подтверждение перед перезаписью

`-n, --no-clobber` — не перезаписывать существующий файл

Если вы укажете более одного из -i, -f, -n, вступит в силу только последний.

`--strip-trailing-slashes` — удалить завершающие косые черты из каждого параметра ИСТОЧНИКИ

`-T, --no-target-directory` — рассматривать ЦЕЛЬ как обычный файл

`-u, --update` — копировать только в том случае, если файл ИСТОЧНИК новее, чем файл назначения, или когда файл назначения отсутствует

`-v, --verbose` — выводить, что делается

`-Z` — Установить в контекст безопасности SELinux значение по умолчанию для целевых файлов

`--context=CTX` — Установить контекст безопасности SELinux равный CTX для целевых файлов

rm

`rm [опции] ФАЙЛЫ`

Удаляет заданные файлы или каталоги.

По умолчанию каталоги не удаляются.

Если задана опция -I или --interactive=once, и имеется более трех файлов или заданы -r, -R или --recursive, то rm запрашивает у пользователя, следует ли продолжить всю операцию. Если ответ отрицательный, то команда прерывается.

В противном случае, если файл недоступен для записи, стандартный ввод - это терминал, а параметр -f или --force не задан или задан параметр -i или --interactive = always, rm запрашивает у пользователя, следует ли удалить файл. Если ответ не утвердительный, файл пропускается.

Опции:

`-f, --force` — игнорировать несуществующие файлы, не запрашивать подтверждение

`-i` — запрашивать подтверждение перед каждым удалением

`-I` — запрашивать один раз перед удалением более трех файлов или при удалении рекурсивно; менее навязчив, чем -i, но при этом обеспечивает защиту от большинства ошибок

`--interactive[=WHEN]` — запрашивать подтверждение в соответствии с WHEN: never - никогда, once — однократно (-I), always — каждый раз (-i);

`--one-file-system` — при рекурсивном удалении иерархии пропустить любой каталог, который находится в файловой системе, отличной от каталога заданного параметром командной строки

`--no-preserve-root` — не защищать корневой каталог

`--preserve-root[=all]` — не удалять корневой каталог (по умолчанию); с 'all' отклонить любой аргумент командной строки на устройстве, отличном от его родительского

`-r, -R, --recursive` — рекурсивно удалять каталоги и их содержимое

`-d, --dir` — удалить пустые каталоги

`-v, --verbose` — выводить, что делается

ln

```
ln [опции] [-T] ЦЕЛЬ ИМЯ_ССЫЛКИ
ln [опции] ЦЕЛЬ [КАТАЛОГ]
```

Создаёт ссылку на целевой файл или каталог в заданном каталоге. Если каталог не задан, ссылка создаётся в текущем каталоге.

По умолчанию создаются жесткие ссылки, символические ссылки с опцией --symbolic. По умолчанию каждый пункт назначения (имя новой ссылки) не должен существовать. При создании жестких ссылок каждая ЦЕЛЬ должна существовать. Символические ссылки могут содержать произвольный текст; если позже будет разрешено, относительная ссылка интерпретируется относительно своего родительского каталога.

Опции:

`-d, -F, --directory` — разрешить суперпользователю пытаться создать жёсткую ссылку на каталог (примечание: вероятно, произойдет сбой из-за системных ограничений, даже для суперпользователя)

`-f, --force` — удалять существующие файлы при создании ссылки

`-i, --interactive` — запрашивать подтверждение перед удалением существующего файла на месте ссылки

`-L, --logical` — разыменование ЦЕЛЕЙ, которые являются символическими ссылками

`-n, --no-dereference` — рассматривать ИМЯ_ССЫЛКИ как обычный файл, если это символическая ссылка на каталог

`-P, --physical` — делать жесткие ссылки прямо на символические ссылки

`-r, --relative` — создавать символические ссылки относительно каталога, в котором создаётся ссылка

`-s, --symbolic` — создать символьные ссылки вместо жестких

`-t, --target-directory=КАТАЛОГ` — указывает КАТАЛОГ для создания ссылок

`-T, --no-target-directory` — всегда относится к LINK_NAME как к обычному файлу

`-v, --verbose` — напечатать имя каждого связанного файла

tar

```
tar ОПЕРАЦИЯ [опции] ФАЙЛЫ
```

Операции:

`-A, --catenate, --concatenate` — Добавить архив в конец другого архива. Аргументы рассматриваются как имена добавляемых архивов. Все архивы должны иметь тот

же формат, что и архив, к которому они добавляются, в противном случае полученный архив может быть непригоден для использования с реализациями tar, отличными от GNU. Также обратите внимание, что если указано более одного архива, элементы из архивов, отличных от первого, будут доступны только в итоговом архиве. при использовании параметра -i (--ignore-zeros). Сжатые архивы не могут быть объединены.

`-c, --create` — Создать новый архив. Аргументы предоставляют имена файлов, которые нужно заархивировать. Каталоги архивируются рекурсивно, если не задана опция --no-recursion.

`-d, --diff, --compare` — Найти различия между архивом и файловой системой. Аргументы являются необязательными и указывают элементы архива для сравнения. Если не указан, предполагается текущий рабочий каталог.

`--delete` — Удалить из архива. Аргументы предоставляют имена удаляемых элементов архива. Должен быть приведен хотя бы один аргумент. Эта опция не работает со сжатыми архивами.

`-r, --append` — Добавить файлы в конец архива. Аргументы имеют то же значение, что и для -c (--create).

`-t, --list` — Вывести список содержимого архива. Аргументы необязательны. Когда они даны, они указывают имена участников для списка.

`--test-label` — Протестировать метку тома архива и выйти. При использовании без аргументов печатает метку тома (если есть) и завершает работу со статусом 0. Если заданы один или несколько аргументов командной строки. tar сравнивает метку тома с каждым аргументом. Он завершается с кодом 0, если найдено совпадение, и с кодом 1 в противном случае. Вывод не отображается, если не используется вместе с параметром -v (--verbose).

`-u, --update` — Добавлять в архив файлы новее, чем соответствующая копия. Аргументы имеют то же значение, что и параметры -c и -r. Обратите внимание, что новые файлы не заменяют свои старые архивные копии, а вместо этого добавляются в конец архива. Таким образом, итоговый архив может содержать несколько элементов с одинаковым именем, соответствующих различным версиям одного и того же файла.

`-x, --extract, --get` — Извлечь файлы из архива. Аргументы необязательны. Когда они даны, они указывают имена извлекаемых членов архива.

`--show-defaults` — Показать встроенные значения по умолчанию для различных параметров tar и выйти. Никакие аргументы не допускаются.

`-?, --help` — Отобразить краткую сводку опций и выйти.

`--usage` — Отобразить список доступных опций и выйти.

`--version` — Распечатать версию программы и информацию об авторских правах и выйти.

Опции:

Управление перезаписью файлов

`-k, --keep-old-files` — Не заменять существующие файлы при извлечении.

`--keep-newer-files` — Не заменять существующие файлы, если они новее, чем их архивные копии.

`--keep-directory-symlink` — Не заменять существующие символические ссылки на каталоги при распаковке.

`--no-overwrite-dir` — Сохранить метаданные существующих каталогов.

`--one-top-level[=DIR]` — Распаковать все файлы в DIR или, если используется без аргументов, в подкаталог, названный базовым именем архива (за вычетом стандартных суффиксов сжатия, заданных параметром --auto-compress).

`--overwrite` — При извлечении перезаписывать существующие файлы.

`--overwrite-dir` — Заменять метаданные существующих каталогов при извлечении (используется по умолчанию).

`--recursive-unlink` — Рекурсивно удалить все файлы в каталоге перед его извлечением.

`--remove-files` — Удалить файлы с диска после добавления их в архив.

`--skip-old-files` — Не заменять существующие файлы при извлечении, молча пропускать их.

`-U, --unlink-first` — Удалить каждый файл перед распаковкой поверх него.

`-W, --verify` — Проверить архив после его записи.

Выбор выходного потока

`--ignore-command-error` — Игнорировать коды завершения подпроцесса.

`-no-ignore-command-error` — Обрабатывать ненулевые коды выхода дочерних элементов как ошибку (используется по умолчанию).

`-O, --to-stdout` — Распаковать файлы в стандартный поток вывода.

`--to-command=COMMAND` — Отправлять по конвейеру извлеченные файлы в команду COMMAND. Аргумент - это путь к внешней программе. Можно задать аргументы командной строки. Программа будет вызвана, и ей будет передано содержимое извлекаемого файла в стандартный поток ввода. Дополнительные данные будут предоставлены через следующие переменные среды:

TAR_FILETYPE — Тип файла. Это отдельная буква со следующим значением: f - обычный файл, d — каталог, l - символическая ссылка, h - жесткая ссылка, b - блочное устройство, c - Символьное устройство. В настоящее время поддерживаются только обычные файлы.

TAR_MODE — Файловый режим, восьмеричное число.

TAR_FILENAME — Имя файла.

TAR_REALNAME — Имя файла, хранящегося в архиве.

TAR_UNAME — Имя владельца файла.

TAR_GNAME — Имя группы - владельца файла.

TAR_ATIME — Время последнего доступа. Это десятичное число, обозначающее секунды с начала эпохи. Если архив предоставляет время с точностью до наносекунды, наносекунды добавляются к метке времени после десятичной точки.

TAR_MTIME — Время последней модификации.

TAR_CTIME — Время последнего изменения статуса.

TAR_SIZE — Размер файла.

TAR_UID — UID владельца файла.

TAR_GID — GID владельца файла.

TAR_VERSION — Номер версии GNU tar.\

TAR_ARCHIVE — Имя обрабатываемого архива tar.

TAR_VOLUME — Порядковый номер обрабатываемого тома tar (устанавливается при чтении многотомного архива).

TAR_FORMAT — Формат обрабатываемого архива. Один из: gnu, oldgnu, posix, ustar, v7. TAR_SUBCOMMAND Короткая опция (с дефисом в начале), описывающая операцию, которую выполняет tar.

Обработка атрибутов файла

--atime-preserve[=METHOD] — Сохранять время доступа к файлам, либо восстанавливая время после чтения (METHOD = replace, это значение по умолчанию), либо не устанавливая время (METHOD = system)

--delay-directory-restore — Отложить время изменения настроек и разрешений извлеченных каталогов до конца извлечения. Используйте эту опцию при извлечении из архива с необычным порядком элементов.

--group=NAME[:GID] — Использовать NAME как группу-владельца для добавленных файлов. Если GID не указан, NAME может быть либо именем пользователя, либо числовым GID. В этом случае недостающая часть (GID или имя) будет считана из базы

данных группы текущего хоста. При использовании с --group-map = FILE влияет только на те файлы, группа владельцев которых не указана в FILE.

`--group-map=FILE` — Прочитать карту перевода групп-владельцев из файла FILE. Пустые строки игнорируются. Комментарии обозначаются знаком # и продолжаются до конца строки. Каждая непустая строка в ФАЙЛЕ определяет перевод для отдельной группы. Она должна содержать два поля, разделенных любым количеством пробелов:

OLDGRP NEWGRP[:NEWGID]

В результате каждый входной файл с группой владельцев OLDGRP будет сохранен в архиве с группой владельцев NEWGRP и GID NEWGID.

`--mode=CHANGES` — Принудительное изменение прав доступа для добавленных файлов.

`--mtime=DATE-OR-FILE` — Установить время для добавленных файлов. `DATE-OR-FILE` - это либо дата / время в почти произвольном формате (как в команде «date»), либо имя существующего файла. В последнем случае будет использоваться время mtime этого файла.

`-m, --touch` — Не извлекать время изменения файла.

`--no-delay-directory-restore` — Отменяет действие предыдущей опции --delay-directory-restore.

`--no-same-owner` — Извлекать файлы от имени текущего пользователя (по умолчанию для обычных пользователей).

`--no-same-permissions` — Применять umask пользователя при извлечении разрешений из архива (по умолчанию для обычных пользователей).

`--numeric-owner` — Всегда использовать числа вместо имен пользователей / групп.

`--owner=NAME[:UID]` — Сделать NAME владельцем добавленных файлов. Если UID не указан, NAME может быть либо именем пользователя, либо числовым UID. В этом случае недостающая часть (UID или имя) будет выведена из пользовательской базы данных текущего хоста. При использовании с --owner-map = FILE влияет только на те файлы, владелец которых не указан в FILE.

`--owner-map=FILE` — Прочитать карту перевода владельца из файла FILE. Пустые строки игнорируются. Комментарии обозначаются знаком # и продолжаются до конца строки. Каждая непустая строка в FILE определяет перевод для одного UID. Она должна содержать два поля, разделенных любым количеством пробелов:

OLDUSR NEWUSR [: NEWUID]

OLDUSR - это действительное имя пользователя или UID с префиксом +. Если не указан NEWUID, NEWUSR также должен быть действительным именем пользователя или + UID. В противном случае и NEWUSR, и NEWUID не нужно указывать в базе данных пользователей системы. В результате каждый входной файл, принадлежащий OLDUSR, будет сохранен в архиве с именем владельца NEWUSR и UID NEWUID.

`-p, --preserve-permissions, --same-permissions` — извлекать информацию о правах доступа к файлам (по умолчанию для суперпользователя)

`--preserve` — То же, что и -p и -s

`--same-owner` — Постараться извлечь файлы с тем же владельцем, что и в архиве (по умолчанию для суперпользователя).

`-s, --preserve-order, --same-order` — Сортировать имена в каталоге извлечения в соответствии с архивом.

`--sort=ORDER` — При создании архива отсортирует записи каталога в соответствии с ORDER, который может быть одним из значений none, name или inode. По умолчанию --sort = none, при котором элементы архива хранятся в том же порядке, что и операционная система. Использование --sort = name обеспечивает единообразие и воспроизводимость порядка элементов в созданном архиве. Использование --sort = inode уменьшает количество обращений к диску при создании архива и, таким образом, может значительно ускорить архивирование. Этот порядок сортировки поддерживается только в том случае, если базовая система предоставляет необходимую информацию.

Выбор устройства

`-f, --file=ARCHIVE` — Использовать архивный файл или устройство. Если эта опция не указана, tar сначала проверит переменную окружения «TAPE». Если она установлена, её значение будет использоваться как имя архива. В противном случае tar примет встроенное значение по умолчанию. Значение по умолчанию можно проверить с помощью параметра --show-defaults или в конце вывода команды tar --help. Имя архива с двоеточием указывает файл или устройство на удаленном компьютере. Часть перед двоеточием используется как имя компьютера или IP-адрес, а часть после него - как путь к файлу или устройству, например:

--file=remotehost:/dev/sr0

Необязательное имя пользователя может быть добавлено к имени хоста, поместив между ними знак @. По умолчанию доступ к удаленному хосту осуществляется с помощью команды rsh (1). В настоящее время вместо этого обычно используется ssh (1). Вы можете сделать это, указав следующий параметр командной строки: --rsh-command=/usr/bin/ssh На удаленной машине должна быть установлена команда rmt (8). Если его путь не совпадает с

именем tar по умолчанию, вы можете сообщить tar о правильном имени пути, используя параметр --rmt-command.

`--force-local` — Файл архива является локальным, даже если в нем есть двоеточие.

`-F, --info-script=COMMAND, --new-volume-script=COMMAND` — Выполняет COMMAND в конце каждой ленты (подразумевается использование опции -M). Команда может включать аргументы. При запуске он унаследует окружение tar плюс следующие переменные:

`TAR_VERSION` — Номер версии GNU tar.

`TAR_ARCHIVE` — Имя архива tar обрабатывается.

`TAR_BLOCKING_FACTOR` — количество 512-байтовых блоков в записи.

`TAR_VOLUME` — Порядковый номер обрабатываемого тома tar (устанавливается при чтении многотомного архива).

`TAR_FORMAT` — Формат обрабатываемого архива. Один из: gnu, oldgnu, posix, ustar, v7.

`TAR_SUBCOMMAND` — Короткая опция (с дефисом в начале), описывающая операцию, выполняемую tar.

`TAR_FD` — Файловый дескриптор, который можно использовать для передачи имени нового тома tar.

В случае сбоя информационного скрипта tar завершает работу; в противном случае он начинает писать следующий том.

`-L, --tape-length=N` — Сменить ленту после записи Nx1024 байт. Если за N следует суффикс размера, суффикс указывает множитель, который будет использоваться вместо 1024. Эта опция подразумевает -M.

`-M, --multi-volume` — Создать / перечислить / извлечь многотомный архив.

`--rmt-command=COMMAND` — Использовать COMMAND вместо rmt при доступе к удаленным архивам. См. Описание опции -f выше.

`--rsh-command=COMMAND` — Использовать КОМАНДУ вместо rsh при доступе к удаленным архивам. См. Описание опции -f выше.

`--volno-file=FILE` — Когда эта опция используется вместе с --multi-volume, tar будет отслеживать, с каким томом многотомного архива он работает в ФАЙЛЕ.

Опции сжатия:

`-a, --auto-compress` — Использовать суффикс архива для определения программы сжатия.

`-I, --use-compress-program=COMMAND` — Фильтрует данные через команду `COMMAND`. Команда должна принимать параметр -d для распаковки. Аргумент может содержать параметры командной строки.

`-j, --bzip2` — Сжимает получающийся архив с помощью bzip2 (1).

`-J, --xz` — Сжимает получающийся архив с помощью xz (1).

`--lzip` — Сжимает получающийся архив с помощью lzip(1).

`--lzma` — Сжимает получающийся архив с помощью lzma (1).

`--lzop` — Сжимает получающийся архив с помощью lzop (1).

`--no-auto-compress` — Не использовать суффикс архива для определения программы сжатия.

`-z, --gzip, --gunzip, --ungzip` — Сжимает получающийся архив с помощью gzip (1).

`-Z, --compress, --uncompress` — Сжимает получающийся архив с помощью compress (1).

`--zstd` — Сжимает получающийся архив с помощью zstd (1).

Выбор локального файла

`--add-file=FILE` — Добавить ФАЙЛ в архив (полезно, если его имя начинается с тире).

`--backup[=CONTROL]` — Сохраняет существующий файл перед удалением. Аргумент CONTROL, если он указан, управляет политикой резервного копирования. Его допустимые значения: none, off - не делать резервные копии, t, numbered - Сделать пронумерованные резервные копии, nil, existing - Создавать нумерованные резервные копии, если существуют нумерованные резервные копии, в противном случае - простые резервные копии, never, simple - Всегда создавать простые резервные копии. Если CONTROL не указан, значение берется из переменной среды VERSION_CONTROL. Если он не установлен, предполагается, что существует.

`-C, --directory=DIR` — Изменить текущий каталог на DIR перед выполнением каких-либо операций. Этот параметр чувствителен к порядку, т. Е. Влияет на все последующие параметры.

`--exclude=PATTERN` — Исключить файлы, соответствующие шаблону (с использованием подстановочных знаков) в стиле glob (3).

`--exclude-backups` — Исключить резервные копии и файлы блокировок.

`--exclude-caches` — Исключить содержимое каталогов, содержащих файл CACHEDIR.TAG, за исключением самого файла тегов.

`--exclude-caches-all` — Исключить каталоги, содержащие файл CACHEDIR.TAG и сам файл.

`--exclude-caches-under` — Исключить все в каталогах, содержащих CACHEDIR.TAG

`--exclude-ignore=FILE` — Перед сбросом каталога проверьте, содержит ли он ФАЙЛ. Если да, прочтите шаблоны исключения из этого файла. Шаблоны влияют только на сам каталог.

`--exclude-ignore-recursive=FILE` — То же, что и --exclude-ignore, за исключением того, что шаблоны из FILE влияют как на каталог, так и на все его подкаталоги.

`--exclude-tag=FILE` — Исключить содержимое каталогов, содержащих ФАЙЛ, кроме самого ФАЙЛА.

`--exclude-tag-all=FILE` — Исключить каталоги, содержащие ФАЙЛ.

`--exclude-tag-under=FILE` — Исключить все, что находится в каталогах, содержащих ФАЙЛ.

`--exclude-vcs` — Исключить каталоги системы контроля версий.

`--exclude-vcs-ignores` — Исключить файлы, соответствующие шаблонам, считанным из файлов игнорирования, относящихся к VCS. Поддерживаемые файлы: .cvsignore, .gitignore, .bzrignore и .hgignore.

`-h, --dereference` — Следовать символическим ссылкам; архивировать и выгружать файлы, на которые они указывают.

`--hard-dereference` — Переходить по жестким ссылкам; заархивировать и сделайте дамп файлов, на которые они ссылаются.

`-K, --starting-file=MEMBER` — Начать с данного члена в архиве.

`--newer-mtime=DATE` — Работать с файлами, данные которых изменились после DATE. Если ДАТА начинается с / или. предполагается, что это имя файла; в качестве даты используется время mtime этого файла.

`--no-null` — Отключить действие предыдущей опции --null.

`--no-recursion` — Избегать автоматического спуска в каталогах.

`--no-unquote` — Не удаляйте кавычки из имен входных файлов или членов.

`-N, --newer=DATE, --after-date=DATE` — Сохранять только файлы новее DATE. Если ДАТА начинается с / или. предполагается, что это имя файла; ctime этого файла используется как дата.

`--one-file-system` — При создании архива оставатьмя в локальной файловой

системе.

`-P, --absolute-names` — При создании архивов не удаляйте начальные косые черты из имен файлов.

`--recursion` — Рекурсия в каталоги (по умолчанию).

`--suffix=STRING` — Резервное копирование перед удалением, переопределите обычный суффикс. Суффикс по умолчанию ~, если он не переопределен переменной среды SIMPLE_BACKUP_SUFFIX.

`-T, --files-from=FILE` — Задаёт имена для извлечения или создания из файла `FILE`. Если не указано иное, ФАЙЛ должен содержать список имен, разделенных ASCII LF (т.е. по одному имени в строке). Считанные имена обрабатываются так же, как аргументы командной строки. Они проходят удаление кавычек и разделение слов. Любая строка, начинающаяся с -, обрабатывается как параметр командной строки tar. Если такое поведение нежелательно, его можно отключить с помощью параметра --verbatim-files-from. Этот параметр влияет на все параметры --files-from, следующие после него в командной строке. Его действие отменяется параметром --no-verbatim-files-from. Этот параметр подразумевается параметром --null. См. Также --add-file.

`-X, --exclude-from=FILE` — Исключить файлы, соответствующие шаблонам, указанным в FILE.

Параметры сопоставления имен файлов

`--ignore-case` — Игнорировать регистр.

`--no-anchored` — Шаблоны совпадают после любого / (по умолчанию для исключения).

`--no-ignore-case` — Учитывать регистр (по умолчанию).

`--no-wildcards` — Дословное сопоставление строк, без подстановочных знаков.

`--no-wildcards-match-slash` — Не учитывать в качестве подстановочного знака /.

`--wildcards` — Использовать подстановочные знаки (по умолчанию для исключения).

`--wildcards-match-slash` — учитывать в качестве подстановочного знака / (по умолчанию для исключения).

Управление выводом

`--checkpoint[=N]` — Отображать сообщения о ходе выполнения каждую N-ю запись (по умолчанию 10).

`--checkpoint-action=ACTION` — Выполните ДЕЙСТВИЕ на каждой

контрольной точке.

`--clamp-mtime` — Устанавливть время только в том случае, если файл более свежий, чем указано в параметре --mtime.

`--full-time` — Распечатать полное время файла.

`--index-file=FILE` — Отправить подробный вывод в ФАЙЛ.

`--no-quote-chars=STRING` — Отключить заключение в кавычки символов из STRING.

`--quote-chars=STRING` — Дополнительно заключать в кавычки символов из STRING.

`--quoting-style=STYLE` — Установите стиль цитирования для имен файлов и членов. Допустимые значения для СТИЛЯ: literal, shell, shell-always, c, c-might, escape, locale, clocale.

`-R, --block-number` — Показывать номер блока в архиве с каждым сообщением.

`--show-omitted-dirs` — При перечислении или извлечении перечислять все каталоги, не соответствующие критериям поиска.

`--show-transformed-names, --show-stored-names` — Показывать имена файлов или архивов после преобразования с помощью параметров --strip и --transform.

`--utc` — Вывести время модификации файла в формате UTC.

`-v, --verbose` — Выводить список обработанных файлов. Если не задать этот параметр, команда отработает «молча», если не будет ошибок.

`-w, --interactive, --confirmation` — Просить подтверждения для каждого действия.

find

`find [опции] [СТАРТОВАЯ_ТОЧКА] [ВЫРАЖЕНИЕ]`

Ищет иерархически файлы в каталогах, перечисленных через запятую в параметре СТАРТОВАЯ_ТОЧКА, оценивая ВЫРАЖЕНИЕ слева направо в соответствии с правилами приоритета, пока не станет известен результат (левая часть равна false для операций AND, true для операций OR), после чего find перейдет к следующему имени файла. Если начальная СТАРТОВАЯ_ТОЧКА не указана, то предполагается текущий каталог.

Опции

`-P` — Никогда не переходить по символическим ссылкам. Это поведение по умолчанию. Когда find проверяет или печатает информацию о файле, а файл является символической ссылкой, используемая информация должна быть взята из свойств самой

символической ссылки.

`-L` — Переходить по символическим ссылкам. Когда find проверяет или печатает информацию о файлах, используемая информация должна быть взята из свойств файла, на который указывает ссылка, а не из самой ссылки (если это не является неработающей символической ссылкой или find не может проверить файл, на который точки ссылки). Использование этой опции подразумевает -noleaf. Если вы позже используете опцию -P, -noleaf все равно будет действовать. Если действует -L и find обнаруживает символическую ссылку на подкаталог во время своего поиска, будет выполнен поиск подкаталога, на который указывает символическая ссылка. Когда действует опция -L, предикат -type всегда будет соответствовать типу файла, на который указывает символическая ссылка, а не самой ссылке (если символическая ссылка не разорвана). Действия, которые могут привести к нарушению символьных ссылок во время выполнения find (например, -delete), могут привести к сбивающему с толку поведению. Использование -L приводит к тому, что предикаты -lname и -ilname всегда возвращают false.

`-H` — Не переходить по символическим ссылкам, кроме как при обработке аргументов командной строки. Когда find проверяет или печатает информацию о файлах, используемая информация берется из свойств самой символической ссылки. Единственное исключение из этого поведения - когда файл, указанный в командной строке, является символической ссылкой, и ссылка может быть разрешена. В этой ситуации используемая информация берется из того, на что указывает ссылка (то есть переход по ссылке). Информация о самой ссылке используется в качестве запасного варианта, если файл, на который указывает символическая ссылка, не может быть исследован. Если действует -H и один из путей, указанных в командной строке, является символической ссылкой на каталог, содержимое этого каталога будет проверено (хотя, конечно, -maxdepth 0 предотвратит это).

Выражение

Это своего рода спецификация запроса, описывающая, как мы сопоставляем файлы и что мы делаем с файлами, которые были сопоставлены. Выражение состоит из последовательности следующих элементов

Тесты возвращают истинное или ложное значение, обычно на основе некоторого свойства файла, который мы рассматриваем. Например, проверка -empty верна только тогда, когда текущий файл пуст.

Действия имеют побочные эффекты (например, печать чего-либо на стандартном выводе) и возвращают либо истину, либо ложь, обычно в зависимости от того, успешны они или нет. Действие -print, например, выводит имя текущего файла на стандартный вывод.

Глобальные опции влияют на работу тестов и действий, указанных в любой части

командной строки. Глобальные параметры всегда возвращают истину. Например, опция -depth заставляет find перемещаться по файловой системе в порядке глубины.

Позиционные опции влияют только на тесты или действия, которые следуют за ними. Позиционные параметры всегда возвращают значение true. Например, параметр -regextype является позиционным, определяя диалект регулярных выражений для регулярных выражений, которые появляются позже в командной строке.

Операторы объединяют другие элементы в выражении. К ним относятся, например, -o (что означает логическое ИЛИ) и -a (что означает логическое И). Если оператор отсутствует, предполагается -a.

Если все выражение не содержит никаких действий, кроме -prune или -print, -print выполняется для всех файлов, для которых все выражение истинно.

Действие -delete также действует как опция (поскольку подразумевает -depth).

Позиционные опции

Позиционные опции всегда возвращают значение true. Они влияют только на тесты, выполняемые позже в командной строке.

`-daystart` — Измерять время (для -amin, -atime, -cmin, -ctime, -mmin и -mtime) с начала сегодняшнего дня, а не с 24 часов назад. Эта опция влияет только на тесты, которые появляются позже в командной строке.

`-regextype type` — Изменяет синтаксис регулярных выражений, понимаемый тестами -regex и -iregex, которые выполняются позже в командной строке. Чтобы узнать, какие типы регулярных выражений известны, используйте -regextype help.

`-warn, -nowarn` — Включить или отключить предупреждающие сообщения. Эти предупреждения относятся только к использованию командной строки, а не к любым условиям, которые могут возникнуть при поиске в каталогах. Поведение по умолчанию соответствует -warn, если стандартный ввод - это tty, и -nowarn в противном случае. Если выводится предупреждающее сообщение, относящееся к использованию командной строки, статус выхода find не изменяется. Если установлена переменная среды POSIXLY_CORRECT и также используется -warn, не указывается, какие предупреждения будут активными, если таковые имеются.

Глобальные опции

Глобальные опции всегда возвращают true. Глобальные опции действуют даже для тестов, которые выполняются ранее в командной строке. Во избежание путаницы глобальные опции следует указывать в командной строке после списка начальных точек, непосредственно перед первым тестом, позиционной опцией или действием. Если вы укажете глобальную опцию в другом месте, find выдаст предупреждающее сообщение,

поясняющее, что это может сбивать с толку.

`-d` — Синоним -depth для совместимости с FreeBSD, NetBSD, MacOS X и OpenBSD.

`-depth` — Обрабатывать содержимое каждого каталога перед самим каталогом. Действие -delete также подразумевает -depth.

`-help, --help` — Распечатать сводку использования команды и выйти

`-ignore_readdir_race` — Обычно find выдает сообщение об ошибке, если не удается обработать файл. Если вы дадите эту опцию и файл будет удален между моментом, когда find считывает имя файла из каталога и когда он пытается зафиксировать файл, сообщение об ошибке не будет выдано. Это также относится к файлам или каталогам, имена которых указаны в командной строке. Эта опция вступает в силу во время чтения командной строки, что означает, что вы не можете искать в одной части файловой системы, если эта опция включена, и в части, если эта опция выключена (если вам нужно это сделать, вам нужно будет выполнить два вместо этого найдите команды, одна с опцией, а другая без нее). Кроме того, find с параметром -ignore_readdir_race будет игнорировать ошибки действия -delete в случае, если файл исчез с момента чтения родительского каталога: он не будет выводить диагностику ошибки, и код возврата действия -delete будет истинным. .

`-maxdepth levels` — Спуститься на не более чем `levels` уровней (неотрицательное целое число) уровней каталогов ниже начальных точек. -maxdepth 0 означает, что тесты и действия применяются только к самим начальным точкам.

`-mindepth levels` — Не применять никаких тестов или действий на уровнях меньше `levels`. -mindepth 1 означает обработку всех файлов, кроме стартовых точек.

`-mount` — Не переходить по каталогам в других файловых системах. Альтернативное имя -xdev для совместимости с некоторыми другими версиями find.

`-noignore_readdir_race` — Отключает эффект -ignore_readdir_race.

`-noleaf` — Не оптимизировать, предполагая, что каталоги содержат на 2 подкаталога меньше, чем их количество жестких ссылок. Эта опция необходима при поиске файловых систем, которые не соответствуют соглашению о ссылках на каталоги Unix, например файловых систем CD-ROM или MS-DOS или точек монтирования томов AFS. Каждый каталог в нормальной файловой системе Unix имеет как минимум две жесткие ссылки: свое имя и свой `. ' Вход. Кроме того, каждый его подкаталог (если он есть) имеет запись `.. ', связанную с этим каталогом. Когда find исследует каталог, после того, как он определил на 2 подкаталога меньше, чем счетчик ссылок каталога, он знает, что остальные записи в каталоге не являются каталогами ("листовые" файлы в дереве каталогов). Если нужно исследовать только имена файлов, их не нужно регистрировать; это дает значительное

увеличение скорости поиска.

`-version, --version` — Вывести номер версии и выйти.

`-xdev` — Не переходить по каталогам в других файловых системах.

Тесты

Некоторые тесты, например -newerXY и -samefile, позволяют сравнивать проверяемый в данный момент файл и некоторый справочный файл, указанный в командной строке. Когда используются эти тесты, интерпретация эталонного файла определяется параметрами -H, -L и -P и любыми предыдущими -follow, но эталонный файл проверяется только один раз во время анализа командной строки. Если справочный файл не может быть исследован (например, системный вызов stat (2) для него не работает), выдается сообщение об ошибке, и поиск завершается с ненулевым статусом.

Числовые аргументы могут быть указаны следующим образом

`+n` — для большего, чем n,

`-n` — меньше, чем n,

`n` — ровно на n.

`-amin n` — Последний раз доступ к файлу осуществлялся n минут назад.

`-anewer file` — К файлу последний раз обращались позже, чем он был изменен. Если файл является символической ссылкой и действует опция -H или -L, всегда используется время доступа к файлу, на который он указывает.

`-atime n` — Последний раз доступ к файлу осуществлялся n * 24 часа назад. Когда find выясняет, сколько 24-часовых периодов назад к файлу последний раз обращались, любая дробная часть игнорируется, поэтому для соответствия -atime +1 файл должен быть доступен как минимум два дня назад.

`-cmin n` — Последний раз статус файла изменялся n минут назад.

`-cnewer file` — Последний раз статус файла был изменен позже, чем файл был изменен. Если файл является символической ссылкой и действует опция -H или -L, всегда используется время изменения статуса файла, на который он указывает.

`-ctime n` — Последний раз статус файла был изменен n * 24 часа назад.

`-empty` — Файл пуст и является либо обычным файлом, либо каталогом.

`-executable` — Соответствует исполняемым файлам и каталогам, в которых текущий пользователь может выполнять поиск (в смысле разрешения имен файлов). При этом учитываются списки управления доступом и другие артефакты разрешений, которые игнорируются тестом -perm. В этом тесте используется системный вызов access (2), поэтому серверы NFS могут обмануть его, выполняя сопоставление UID (или подавление корневого

идентификатора), поскольку многие системы реализуют доступ (2) в ядре клиента и поэтому не могут использовать информация об отображении UID, хранящаяся на сервере. Поскольку этот тест основан только на результате системного вызова access (2), нет никакой гарантии, что файл, для которого этот тест прошел успешно, действительно может быть выполнен.

`-false` — Всегда ложно.

`-fstype type` — Файл находится в файловой системе типа type. Допустимые типы файловых систем различаются в зависимости от версии Unix; Неполный список типов файловых систем, которые принимаются в той или иной версии Unix: ufs, 4.2, 4.3, nfs, tmp, mfs, S51K, S52K. Вы можете использовать -printf с директивой %F, чтобы увидеть типы ваших файловых систем.

`-gid n` — файл принадлежит группе с идентификатором n

`-group gname` — файл принадлежит группе `gname`

`-ilname pattern` — Подобно -lname, но совпадение нечувствительно к регистру. Если действует опция -L или опция -follow, этот тест возвращает false, если символическая ссылка не разорвана.

`-iname pattern` — Подобно -name, но при совпадении регистр не учитывается. Например, шаблоны `fo * 'и` F ??' сопоставить имена файлов `Foo ',` FOO', `foo ',` fOo' и т. д. Шаблон `* foo *` также будет соответствовать файлу с именем '.foobar'.

`-inum n` — Файл имеет номер inode n. Обычно проще использовать тест -samefile.

`-ipath pattern` — Подобно -path. но совпадение нечувствительно к регистру.

`-iregex pattern` — Подобно -regex, но совпадение нечувствительно к регистру.

`-iwholename pattern` — См. -Ipath. Эта альтернатива менее переносима, чем -ipath.

`-links n` — Файл имеет n жестких ссылок.

`-lname pattern` — Файл - это символическая ссылка, содержимое которой соответствует шаблону оболочки. Метасимволы не обрабатывают символы "/" или "". специально. Если действует опция -L или опция -follow, этот тест возвращает false, если символическая ссылка не разорвана.

`-mmin n` — Последний раз данные файла были изменены n минут назад.

`-mtime n` — Последний раз данные файла были изменены n * 24 часа назад.

`-name pattern` — База имени файла (путь с удаленными ведущими каталогами) соответствует шаблону оболочки. Поскольку ведущие каталоги удаляются, имена файлов, рассматриваемые для сопоставления с -name, никогда не будут содержать косую черту, поэтому «-name a / b» никогда не будет соответствовать чему-либо (вам, вероятно,

потребуется вместо этого использовать -path). Если вы попытаетесь это сделать, выдается предупреждение, если не установлена переменная среды POSIXLY_CORRECT. Метасимволы (`* ',`?' И `[] ') соответствуют`.' в начале имени базы. Чтобы игнорировать каталог и файлы в нем, используйте -prune; см. пример в описании -path. Фигурные скобки не считаются особенными, несмотря на то, что некоторые оболочки, включая Bash, наделяют фигурные скобки особым значением в шаблонах оболочки. Сопоставление имени файла выполняется с использованием библиотечной функции fnmatch (3). Не забудьте заключить шаблон в кавычки, чтобы защитить его от расширения оболочкой.

`-newer file` — Файл был изменен позже, чем файл `file`. Если файл является символической ссылкой и действует опция -H или -L, всегда используется время модификации файла, на который он указывает.

`-nogroup` — Ни одна группа не соответствует числовому идентификатору группы файла.

`-nouser` — Ни один пользователь не соответствует числовому идентификатору пользователя файла.

`-path pattern` — Имя файла соответствует шаблону оболочки. Метасимволы не обрабатывают символы "/" или "". специально; так, например, найди. -path "./sr*sc" напечатает запись для каталога с именем './src/misc' (если он существует). Чтобы игнорировать все дерево каталогов, используйте -prune вместо проверки каждого файла в дереве. Например, чтобы пропустить каталог src/emacs и все файлы и каталоги в нем и распечатать имена других найденных файлов, сделайте что-то вроде этого: найти . -path ./src/emacs -prune -o -print Обратите внимание, что проверка соответствия шаблону применяется ко всему имени файла, начиная с одной из начальных точек, указанных в командной строке. Здесь имеет смысл использовать абсолютный путь только в том случае, если соответствующая начальная точка также является абсолютным путем. Это означает, что эта команда никогда ничего не найдет: найти bar -path /foo/bar/myfile -print Поиск сравнивает аргумент -path с объединением имени каталога и базового имени файла, который он исследует. Поскольку конкатенация никогда не заканчивается косой чертой, аргументы -path, заканчивающиеся косой чертой, не будут соответствовать ничему (кроме, возможно, начальной точки, указанной в командной строке). Предикат -path также поддерживается функцией поиска HP-UX и является частью стандарта POSIX 2008.

`-perm mode` — Биты прав доступа к файлу точно соответствуют режиму (восьмеричному или символьному). Поскольку требуется точное совпадение, если вы хотите использовать эту форму для символьных режимов, вам, возможно, придется указать довольно сложную строку режима. Например, «-perm g = w» будет соответствовать только

файлам с режимом 0020 (то есть тем, для которых разрешение на запись группы является единственным набором разрешений). Более вероятно, что вы захотите использовать формы «/» или «-», например, «-perm -g = w», что соответствует любому файлу с правом записи группы.

`-perm -mode` — Для файла установлены все биты прав доступа. В этой форме принимаются символьные режимы, и обычно вы хотите их использовать именно так. Вы должны указать `u ',` g' или `o ', если вы используете символьный режим.

`-perm /mode` — Для файла установлен любой из битов разрешения режима. В этой форме принимаются символьные режимы. Вы должны указать `u ',` g' или `o ', если вы используете символьный режим. Если в режиме биты разрешений не установлены, этот тест соответствует любому файлу (идея заключается в согласовании с поведением -perm -000).

`-perm +mode` — больше не поддерживается (устарело с 2005 года). Вместо этого используйте -perm / mode.

`-readable` — Соответствует файлам, доступным для чтения текущему пользователю. При этом учитываются списки управления доступом и другие артефакты разрешений, которые игнорируются тестом -perm. В этом тесте используется системный вызов access (2), поэтому серверы NFS могут обмануть его, выполняя сопоставление UID (или подавление корневого идентификатора), поскольку многие системы реализуют доступ (2) в ядре клиента и поэтому не могут использовать информация об отображении UID, хранящаяся на сервере.

`-regex pattern` — Имя файла соответствует шаблону регулярного выражения. Это совпадение на всем пути, а не поиск. Например, чтобы найти файл с именем `./fubar3', вы можете использовать регулярное выражение «.*Bar.» или «.*b.*3», но не «f.*r3». Регулярные выражения, понимаемые с помощью find, по умолчанию являются регулярными выражениями Emacs (за исключением того, что «.» Соответствует новой строке), но это можно изменить с помощью параметра -regextype.

`-samefile name` — Файл ссылается на тот же индексный дескриптор, что и имя. Когда действует -L, это может включать символические ссылки.

`-size n[cwbkMG]` — Файл занимает n единиц пространства с округлением в большую сторону. Могут использоваться следующие суффиксы:

`b` — для блоков размером 512 байт (это значение по умолчанию, если не используется суффикс)

`c` — для байтов

`w` — двухбайтовых слов

`k` — для кибибайтов (KiB, единицы по 1024 байта)

`M` — для мебибайт (MiB, единицы 1024 * 1024 = 1048576 байт)

`G` — для гибибайтов (ГиБ, единицы 1024 * 1024 * 1024 = 1073741824 байта)

Префиксы + и - обозначают больше и меньше, как обычно; т.е. точный размер n единиц не совпадает. Имейте в виду, что размер округляется до следующей единицы. Следовательно, -size -1M не эквивалентен -size -1048576c. Первый соответствует только пустым файлам, второй соответствует файлам от 0 до 1 048 575 байтов.

`-true` — Всегда true.

`-type c` — Файл имеет тип c: b блок, c символ, d каталог, p именованный канал (FIFO), f обычный файл, l символическая ссылка, s сокет. Для одновременного поиска более чем одного типа вы можете предоставить объединенный список типовых букв, разделенных запятой.

`-used n` — Последний раз доступ к файлу осуществлялся через n дней после последнего изменения его статуса.

`-user uname` — Файл принадлежит пользователю uname (разрешен числовой идентификатор пользователя).

`-wholename pattern` — Смотрите -path. Эта альтернатива менее переносима, чем -path.

`-writable` — Соответствует файлам, которые доступны для записи текущему пользователю.

`-xtype c` — То же, что и -type, если файл не является символической ссылкой. Для символических ссылок: если была указана опция -H или -P, true, если файл является ссылкой на файл типа c; если была указана опция -L, истина, если c равно `l '. Другими словами, для символических ссылок -xtype проверяет тип файла, который -type не проверяет.

`-context pattern` — Контекст безопасности SELinux файла соответствует шаблону.

Действия

`-delete` — Удалить файлы, Возвращает «истина», если удаление прошло успешно. Если удаление не удалось, выдается сообщение об ошибке. Если -delete завершается неудачно, статус выхода find будет отличным от нуля (когда он в конечном итоге завершится). Использование -delete автоматически включает параметр -depth.

`-exec command ;` — Выполнить команду; Возвращает «истина», если команда завершилась с 0 статусом. Все следующие аргументы для поиска считаются аргументами

команды до тех пор, пока не будет аргумент, состоящий из `; ' встречается. Строка «{}» заменяется текущим именем файла, обрабатываемым везде, где она встречается в аргументах команды, а не только в аргументах, где она одна, как в некоторых версиях find. Обе эти конструкции, возможно, потребуется экранировать (с помощью «\») или заключить в кавычки, чтобы защитить их от расширения оболочкой. Указанная команда запускается один раз для каждого совпадающего файла. Команда выполняется в стартовом каталоге. Есть неизбежные проблемы безопасности, связанные с использованием действия -exec; вместо этого вам следует использовать параметр -execdir.

`-exec command {} +` — Этот вариант действия -exec запускает указанную команду для выбранных файлов, но командная строка строится путем добавления каждого выбранного имени файла в конце; общее количество вызовов команды будет намного меньше количества совпадающих файлов. Командная строка строится почти так же, как xargs строит свои командные строки. В команде разрешен только один экземпляр `{} ', и (когда find вызывается из оболочки) он должен быть заключен в кавычки (например, '{}'), чтобы защитить его от интерпретации оболочками. Команда выполняется в стартовом каталоге. Если какой-либо вызов с формой `+ 'возвращает ненулевое значение в качестве статуса выхода, тогда find возвращает ненулевой статус выхода. Если find обнаруживает ошибку, это иногда может вызвать немедленный выход, поэтому некоторые ожидающие команды могут вообще не выполняться. Этот вариант -exec всегда возвращает истину.

`-execdir command ;`

`-execdir command {} +` — Подобно -exec, но указанная команда запускается из подкаталога, содержащего соответствующий файл, который обычно не является каталогом, в котором вы начали поиск.

`-fls file` — Возвращает true. Подобна -ls, но записывает в файл как -fprint. Выходной файл создается всегда, даже если предикат никогда не сопоставляется.

`-fprint file` —Возвращает true. Распечатать полное имя файла в файл. Если файл не существует при запуске find, он создается; если он существует, он усекается. Выходной файл создается всегда, даже если предикат никогда не сопоставляется.

`-fprint0 file` — Возвращает true. Подобна -print0, но записывает в файл как -fprint. Выходной файл создается всегда, даже если предикат никогда не сопоставляется.

`-fprintf file format` — Возвращает true. Подобна -printf, но записывать в файл как -fprint. Выходной файл создается всегда, даже если предикат никогда не сопоставляется

`-ls` — Возвращает true. Вывести текущий каталог в формате ls -dils в стандартный поток вывода

-print — Возвращает true. Вывести полное имя файла в стандартный поток вывода с последующим переводом на новую строку. Если вы передаете вывод find в другую программу и существует вероятность того, что файлы, которые вы ищете, могут содержать новую строку, вам следует серьезно подумать об использовании опции -print0 вместо -print.

-print0 — Возвращает true. Вывести полное имя файла в стандартный поток вывода, за которым следует нулевой символ (вместо символа новой строки, который использует -print). Это позволяет программам, обрабатывающим вывод find, правильно интерпретировать имена файлов, содержащие символы новой строки или другие типы пробелов. Этот параметр соответствует параметру -0 в xargs.

-printf format — Возвращает true. Вывести имя файла в заданном формате в стандартный поток вывода, интерпретируя экранирующие символы «\» и директивы «%». Ширину полей и точность можно указать, как в функции «C» printf(). Обратите внимание, что многие поля печатаются как %s, а не как %d, и это может означать, что флаги работают не так, как вы могли ожидать. Это также означает, что флаг «-» работает (заставляя поля выравниваться по левому краю). В отличие от -print, -printf не добавляет новую строку в конец строки. Можно применять следующие экраны и директивы:

\f — новый лист

\n — Новая строка.

\r — Возврат каретки.

\t — горизонтальная табуляция

\v — Вертикальная табуляция.

\0 — ASCII NUL.

\\ — Символ «\»

\NNN — Символ с восьмеричным кодом NNN

Символ «\», за которым следует любой другой символ, рассматривается как обычный символ, поэтому печатаются оба символа.

%% — Символ «%»

%a — Время последнего доступа к файлу в формате, возвращаемом функцией «C» ctime().

%Ak — Время последнего доступа к файлу в формате, заданном параметром k:

H — час (00..23)

I — час (01..12)

k — час (0..23)

l — час (1..12)

M — минут (00..59)

p — AM или PM

r — время 12-hour (hh:mm:ss [AP]M)

S — Секунда (00.00 .. 61.00). Есть дробная часть.

T — Время, 24 часа (чч: мм: сс.xxxxxxxxx)

+ — Дата и время, разделенные знаком «+», например «2004-04-28 + 22: 22: 05.0»

X — Представление времени локали X (H:M:S). Поле секунд включает дробную часть.

Z — Часовой пояс или ничего, если часовой пояс не определяется

a — сокращенное название дня недели (Вс..Сб) (для локали)

A — Полное название дня недели (воскресенье… суббота) (для локали)

b — сокращенное название месяца

B — Полное название месяца

c — Дата и время в формате локали

d — день месяца (01..31)

D — Дата (мм/дд/гг)

h — то же, что и b

j — день года (001..366)

m — месяц (01..12)

U — номер недели года с воскресеньем в качестве первого дня недели (00..53)

w — день недели (0..6)

W — номер недели года с понедельником в качестве первого дня недели (00..53)

x — представление даты в формате текущей локали

y — две последние цифры года (00..99)

Y — год (1970 ...)

%b — Объем дискового пространства, используемого для этого файла в блоках по 512 байт

%c — Время последнего изменения статуса файла в формате, возвращаемом функцией «C» ctime()

%Ck — Время последнего изменения статуса файла в формате, заданном

параметром k, таком же, как для %Ak.

`%d` — Глубина файла в дереве каталогов; 0 означает, что файл является стартовой точкой.

`%D` — Номер устройства, на котором расположен файл.

`%f` — Имя файла без родительских каталогов (только последний элемент).

`%F` — Тип файловой системы, в которой находится файл

`%g` — Имя группы-владельца файла или числовой идентификатор группы, если у группы нет имени.

`%G` — числовой идентификатор группы-владельца файла.

`%h` — Начальные каталоги имени файла (все, кроме последнего элемента). Если имя файла не содержит косой черты (поскольку оно находится в текущем каталоге), выводится «.».

`%H` — Начальная точка, под которой был найден файл.

`%i` — Номер inode файла (в десятичном формате).

`%k` — Объем дискового пространства, используемого для этого файла в блоках по 1 КБ.

`%l` — Объект на который ссылается символьная ссылка (пустая строка, если файл не является символьной ссылкой).

`%m` — Биты прав доступа к файлу (в восьмеричном формате).

`%M` — Права доступа к файлу (в символической форме, как в ls)

`%n` — Количество жестких ссылок на файл.

`%p` — Имя файла.

`%P` — Имя файла с именем начальной точки, под которой он был найден.

`%s` — Размер файла в байтах.

`%S` — Разреженность файла. Рассчитывается как (BLOCKSIZE * st_blocks / st_size).

`%t` — Время последней модификации файла в формате, возвращаемом функцией «C» ctime().

`%Tk` — Время последней модификации файла в формате, заданном параметром k, таком же, как в %Ak.

`%u` — Имя пользователя-владельца файла или числовой идентификатор пользователя, если у пользователя нет имени.

`%U` — Числовой идентификатор пользователя-владельца файла.

`%y` — Тип файла (как в команде ls -l)

`%Y` — Тип файла (как в %y) плюс символ: «L» - цикл, «N» - несуществующий, «?» для любой другой ошибки при определении типа цели символической ссылки.

`%Z` — контекст безопасности SELinux для файла.

`-quit` — Немедленно завершить работу.

Операторы

Перечислены в порядке убывания приоритета:

`( expr )` — Приоритетная последовательность. Поскольку круглые скобки являются специальными для оболочки, вам обычно нужно заключать их в кавычки или использовать экранирование «\(...\)» .

`! expr` — Истинно, если выражение ложно. Этот символ также обычно нуждается в защите от интерпретации оболочкой.

`-not expr` — То же что «! expr», но не совместим с POSIX.

`expr1 expr2` — Истинно, если оба выражения истинны. expr2 не оценивается, если expr1 ложно.

`expr1 -a expr2` — То же, что и expr1 expr2.

`expr1 -and expr2` — То же, что expr1 expr2, но не соответствует POSIX.

`expr1 -o expr2` — Оператор ИЛИ. expr2 не оценивается, если expr1 истинно.

`expr1 -or expr2` — То же, что expr1 -o expr2, но не соответствует POSIX.

`expr1 , expr2` — Список. Оба выражения expr1 и expr2 всегда оцениваются. Оператор запятая может быть полезен для поиска нескольких разных типов вещей, но при обходе иерархии файловой системы только один раз.