Основные команды Linux

Основные команды Linux

Если вы проигнорировали введение, напомним, что команды в статье сгруппированы по исполняемой ими функции. Что может быть более необходимым для новичка, чем функция справки?

Команды для получения справки

man — manual, получение справки

Самая первая команда Linux для начинающих — manual — для получения полной справочной информации по другой команде. Некоторые пользователи настаивают, что искать в интернете информацию по команде проще и эффективнее. Однако, информация в интернете не всегда является верной, статья в интернете может быть устаревшей, не релевантной для вашей версии ОС и т.д. Использовать man всегда хорошая идея.

Чтобы получить справку по команде, введите перед ней man. Например, **man man** выдаст руководство по команде man. Также можно вывести мануал терминала Linux (**man bash**):

```
BASH(1)                                  General Commands Manual
BASH(1)



NAME

       bash - GNU Bourne-Again SHell



SYNOPSIS

       bash [options] [command_string | file]
```

COPYRIGHT

Bash is Copyright (C) 1989-2018 by the Free Software Foundation, Inc.

DESCRIPTION

Bash is an sh-compatible command language interpreter that executes commands read from the standard input or from a file. Bash also incorporates useful features from the Korn and C shells (ksh and csh).

Bash is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification (IEEE Standard 1003.1). Bash can be configured to be POSIX-conformant by default.

OPTIONS

All of the single-character shell options documented in the description of the set builtin command, including -o, can be used as options when the shell is invoked. In addition, bash interprets the following options when it is invoked:

-c If the -c option is present, then commands are read from the first non-option argument command_string. If there are arguments after the command_string, the first argument is assigned to $0 and any remaining arguments are assigned to the positional

help — когда не работает man

Не у каждой команды имеется свое полноценное руководство и не всегда оно требуется. В таких случаях помогает help, которая выводит краткую справку.

```
history: history [-c] [-d смещение] [n] или history -anrw [файл] или history -ps аргумент
[аргумент...]

   Display or manipulate the history list.

   Display the history list with line numbers, prefixing each modified
   entry with a `*'.  An argument of N lists only the last N entries.

   Options:

     -c        clear the history list by deleting all of the entries
     -d offset delete the history entry at position OFFSET. Negative
               offsets count back from the end of the history list

     -a        append history lines from this session to the history file
     -n        read all history lines not already read from the history file
               and append them to the history list
     -r        read the history file and append the contents to the history
               list
     -w        write the current history to the history file

     -p        perform history expansion on each ARG and display the result
               without storing it in the history list
     -s        append the ARGs to the history list as a single entry

   If FILENAME is given, it is used as the history file.  Otherwise,
   if HISTFILE has a value, that is used, else ~/.bash_history.

   If the HISTTIMEFORMAT variable is set and not null, its value is used
   as a format string for strftime(3) to print the time stamp associated
```

Похожий вариант — вывод справки через специальные ключи — **<команда> -h** или **<команда> —help**.

Tab — автозавершение команды

Командная строка Linux может предложить вам доступные варианты завершения команды. Например, если вы хотите узнать, куда можно перейти из текущей папки, наберите cd, затем дважды нажмите Tab. Если хотите узнать, какие папки начинаются с символа точки, введите cd . и завершите двойным нажатием Tab. Работает с любой командой, но только в современных оболочках — bash и zsh.

```
debtop@DebTop:~$ cd

.aptitude/     .gnupg/        .mozilla/      Видео/         Изображения/   Рабочий стол/

.cache/        .kde/          .pki/          Документы/     Музыка/        Шаблоны/

.config/       .local/        snap/          Загрузки/      Общедоступные/

debtop@DebTop:~$ cd .

./         .aptitude/ .config/   .kde/      .mozilla/
../        .cache/    .gnupg/    .local/    .pki/
```

cat /etc/*-release — какой дистрибутив установлен на моей машине

Если вы задумывались об установке Linux или уже использовали эту ОС, вы должны были знать, что существует несколько дистрибутивов. В зависимости от дистрибутива команды могут отличаться, поэтому полезно иметь возможность узнать, какой именно дистрибутив на этой машине. Это можно сделать и через информацию о системе, из графического интерфейса окружения пользователя, но наша статья о терминале. Команда **cat /etc/*-release (без пробелов)** покажет вам основную информацию о дистрибутиве — имя, версия, и т.д. Аналогами в данном случае будут являться **lsb_release -a**, которая выведет почти ту же информацию, а **lsb_release -i** напишет ID дистрибутива.

```
debtop@DebTop:~$ cat /etc/*-release

PRETTY_NAME="Debian GNU/Linux 10 (buster)"
```

```
NAME="Debian GNU/Linux"

VERSION_ID="10"

VERSION="10 (buster)"

VERSION_CODENAME=buster

ID=debian

HOME_URL="https://www.debian.org/"

SUPPORT_URL="https://www.debian.org/support"

BUG_REPORT_URL="https://bugs.debian.org/"

debtop@DebTop:~$ lsb_release -a

No LSB modules are available.

Distributor ID: Debian

Description:    Debian GNU/Linux 10 (buster)

Release:        10

Codename:       buster

debtop@DebTop:~$ lsb_release -i

Distributor ID: Debian
```

whoami — какой пользователь сейчас используется

Терминал Linux позволяет работать от имени любого пользователя, но не всегда удается удержать в памяти текущего пользователя. Чтобы вспомнить текущего пользователя, существует простая команда whoami.

```
debtop@DebTop:~$ whoami

desktop
```

whatis — что за программа?

Команда Linux терминала whatis дает краткое описание любой установленной программы.

```
debtop@DebTop:~$ whatis nano
nano (1)             - Nano's ANOther editor, an enhanced free Pico clone

debtop@DebTop:~$ whatis apt
apt (8)              - command-line interface



debtop@DebTop:~$ whatis krita
krita: ничего подходящего не найдено.
```

Очевидно, что, если программа не установлена, то получить ее описание не получится.

whereis — полный путь к программе

Допустим, вы нашли нужную программу, но хотите попасть в ее директорию. Узнать директорию программы поможет whereis, показывающая полный путь к исполняемому файлу программы.

```
debtop@DebTop:~$ whereis bash

bash: /usr/bin/bash /etc/bash.bashrc /usr/share/man/man1/bash.1.gz
```

Команды управления сетью

hostname, ifconfig — управление сетью

Эти команды покажут DNS-домен и IP-адрес вашего компьютера.

```
debtop@DebTop:~$ hostname
DebTop

debtop@DebTop:~$ hostname -i
127.0.1.1

debtop@DebTop:~$ ifconfig
bash: ifconfig: команда не найдена
```

В некоторых версиях дистрибутивов Linux поддерживается команда ifconfig, которая также выводит текущий IP, но она работает не всегда. Взамен устаревшей ifconfig современные дистрибутивы отзываются на ***ip a[ddress]***, которая выведет на экран настройки сети и позволяет их редактировать. Команда является частью пакета утилит для настройки параметров сетевых устройств — iproute2. Команды из набора iproute2 пригодятся системным администраторам или тем, кто хочет создать доменную сеть дома.

```
debtop@DebTop:~$ ip a

1: lo:  mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever

2: enp4s0:  mtu 1500 qdisc pfifo_fast state DOWN group default qlen 1
000
   link/ether b4:b5:2f:7f:fe:eb brd ff:ff:ff:ff:ff:ff

3: wlo1:  mtu 1500 qdisc mq state UP group default qlen 1000
   link/ether a4:17:31:2a:bc:59 brd ff:ff:ff:ff:ff:ff
   inet 192.168.1.195/24 brd 192.168.1.255 scope global dynamic noprefixroute wlo1
      valid_lft 85273sec preferred_lft 85273sec
   inet6 fe80::3f0f:a837:e4c0:e52d/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
```

```
debtop@DebTop:~$ ip

Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename

where  OBJECT := { link | address | addrlabel | route | rule | neigh | ntable |
                   tunnel | tuntap | maddress | mroute | mrule | monitor | xfrm |
                   netns | l2tp | fou | macsec | tcp_metrics | token | netconf | ila |
                   vrf | sr }

       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                    -h[uman-readable] | -iec | -j[son] | -p[retty] |
                    -f[amily] { inet | inet6 | ipx | dnet | mpls | bridge | link } |
                    -4 | -6 | -I | -D | -M | -B | -0 |
                    -l[oops] { maximum-addr-flush-attempts } | -br[ief] |
                    -o[neline] | -t[imestamp] | -ts[hort] | -b[atch] [filename] |
                    -rc[vbuf] [size] | -n[etns] name | -a[ll] | -c[olor]}
```

ping — диагностика неисправностей сети

Ping используют для поиска неисправностей, тестирования и диагностики сетевых соединений. Команда отправляет сообщение на указанный IP адрес и ждет ответ. Когда пакет адресат получает пакет, он отправляет ответ. Таким образом можно определить задержку сообщений или потерю пакетов.

Достаточно ввести, например, ***ping google.com***, компьютер сам определит IP адрес домена и продолжит выполняться, пока ее не отменить сочетанием Ctrl+C.

Команды для управления файлами и директориями

Теперь, когда мы выяснили, где находится терминал, мы можем перейти в его директорию. Однако, этого делать не стоит, иначе можно что-нибудь сломать. Давайте попробуем поработать с директорией песочницы.

mkdir — создание директории

Для начала песочницу необходимо создать. Сделать это очень просто, находясь в домашней директории пользователя введем в bash:

mkdir playpen

Отлично, и что дальше? Точно ли директория создалась?

ls — list, отобразить директории и файлы

Чтобы убедиться в наличии директории нужно набрать ls, после выполнения команда покажет все каталоги и файлы.

```
debtop@DebTop:~$ mkdir playpen

debtop@DebTop:~$ ls

playpen   Видео       Загрузки      Музыка         'Рабочий стол'

snap      Документы   Изображения   Общедоступные   Шаблоны
```

cd — change directory, сменить директорию

Теперь перейдем в созданную директорию. Для навигации в терминале Linux нужно ввести ***cd имя_директории***. В нашем случае ***cd playpen***.

```
debtop@DebTop:~$ cd playpen

debtop@DebTop:~/playpen$
```

Если работа в данной директории закончена и необходимо подняться на уровень вверх из текущей директории, используйте «**cd** пробел .. (две точки)»:

cd ..

А если вы работаете одновременно в двух директориях и периодически перемещаетесь из одной в другую, не нужно постоянно держать в голове, какой же была предыдущая. Достаточно использовать быстрый переход к предыдущей директории «**cd** пробел — (дефис)»:

cd —

```
debtop@DebTop:~/playpen/dir00$ cd /usr/lib/x86_64-linux-gnu/audacious

debtop@DebTop:/usr/lib/x86_64-linux-gnu/audacious$ cd -

/home/debtop/playpen/dir00

debtop@DebTop:~/playpen/dir00$ cd -

/usr/lib/x86_64-linux-gnu/audacious

debtop@DebTop:/usr/lib/x86_64-linux-gnu/audacious$ cd -

/home/debtop/playpen/dir00

debtop@DebTop:~/playpen/dir00$ cd ..
```

pwd — где я сейчас?

Обратите внимание, как только мы сменили директорию, поменялся и префикс до символа $, так терминал показывает нам, где мы находимся в данный момент. Конечно, терминал не станет отображать полный путь, если он слишком длинный, поэтому не стоит всегда полагаться на префикс. Быстро понять текущую рабочую директорию поможет команда ***pwd***. Расшифровка этой команды простая — три английских слова — Print Working Directory. На самом деле расшифровка команд Linux всегда простая, они все произошли от английских слов, но были сокращены для лучшего запоминания.

Обратно к песочнице. Пока что директория (также каталог, папка) пуста и не содержит ни одного файла. Давайте создадим в ней 43 директории, в каждой по 43 файла. Можно создавать вручную, каждый раз прописывая имя, а можно поступить проще:

mkdir dir{00..42}

Синтаксис команды простой: mkdir создает директорию, dir задает начальное имя для каждой новой директории, числа в фигурных скобках {00..42} указывают переменную часть имени создаваемых папок. То есть, команда ***mkdir test_folder_{0..4}*** создала бы нам 5 папок с начальным именем test_folder_0.

Давайте проверим, создались ли наши тестовые папки — ***ls.***

```
debtop@DebTop:~/playpen$ ls

dir00  dir03  dir06  dir09  dir12  dir15  dir18  dir21  dir24  dir27  dir30  dir33  dir36
dir39  dir42

dir01  dir04  dir07  dir10  dir13  dir16  dir19  dir22  dir25  dir28  dir31  dir34  dir37
dir40

dir02  dir05  dir08  dir11  dir14  dir17  dir20  dir23  dir26  dir29  dir32  dir35  dir38
dir41
```

Как создать файл?

Замечательно, теперь создадим по 43 файла в каждой. Для создания файла существует более 10 разных способов, самый простой:

```
&gt; имя_файла
```

На самом деле >, как и обратный знак < не совсем являются командами, это символы перенаправления или связи. Однако, если перед именем несуществующего файла поставить знак > вместе с указанием расширения файла, в текущей директории появится пустой файл.

Но нам требуется поместить по 42 файла в 42 директории, а терминал Linux откажется воспринимать команду со знаком перенаправления, поэтому используем команду touch. Эта команда позволяет задать время последнего изменения файла или создать новые файлы:

touch dir{00..42}/text{00..42}.txt

Проверим:

```
debtop@DebTop:~/playpen$ touch dir{00..42}/text{00..42}.txt

debtop@DebTop:~/playpen$ ls dir00

text00.txt  text06.txt  text12.txt  text18.txt  text24.txt  text30.txt  text36.txt  text42.txt
```

```
text01.txt  text07.txt  text13.txt  text19.txt  text25.txt  text31.txt  text37.txt

text02.txt  text08.txt  text14.txt  text20.txt  text26.txt  text32.txt  text38.txt

text03.txt  text09.txt  text15.txt  text21.txt  text27.txt  text33.txt  text39.txt

text04.txt  text10.txt  text16.txt  text22.txt  text28.txt  text34.txt  text40.txt

text05.txt  text11.txt  text17.txt  text23.txt  text29.txt  text35.txt  text41.txt
```

rm — remove, удаление

Чтобы удалить какой-либо файл, достаточно ввести команду rm с именем файла:

rm text00.txt

Можно удалить несколько:

rm text{01..12}.txt

Если попытаться удалить целый каталог по аналогии, то ничего не получится. Все потому, что для удаления каталога нужна другая команда: ***rmdir***:

rmdir delete_me

Команда rmdir тоже не всесильна, она не позволит удалить каталог, в котором есть что-то еще — файл или другой каталог (даже пустой).

Для удаления каталога с содержимым можно использовать команду **rm** с опцией рекурсивного удаления ***-r***. Обратите внимание на картинку ниже, все описанные переходы по директориям, упомянутые в описании команды **cd** действуют и для **rm**. Поэтому, при нахождении в подкаталоге **dir01**, команда будет выглядеть вот так:

rm -r ../dir03

```
debtop@DebTop:~/playpen/dir01$ rm text00.txt

debtop@DebTop:~/playpen/dir01$ rm text{01..12}.txt

debtop@DebTop:~/playpen/dir01$ mkdir delete_me
```

```
debtop@DebTop:~/playpen/dir01$ ls

delete_me   text16.txt  text20.txt  text24.txt  text28.txt  text32.txt  text36.txt  text40.txt

text13.txt  text17.txt  text21.txt  text25.txt  text29.txt  text33.txt  text37.txt  text41.txt

text14.txt  text18.txt  text22.txt  text26.txt  text30.txt  text34.txt  text38.txt  text42.txt

text15.txt  text19.txt  text23.txt  text27.txt  text31.txt  text35.txt  text39.txt

debtop@DebTop:~/playpen/dir01$ rm delete_me

rm: невозможно удалить 'delete_me': Это каталог

debtop@DebTop:~/playpen/dir01$ rmdir delete_me

debtop@DebTop:~/playpen/dir01$ ls

text13.txt  text17.txt  text21.txt  text25.txt  text29.txt  text33.txt  text37.txt  text41.txt

text14.txt  text18.txt  text22.txt  text26.txt  text30.txt  text34.txt  text38.txt  text42.txt

text15.txt  text19.txt  text23.txt  text27.txt  text31.txt  text35.txt  text39.txt

text16.txt  text20.txt  text24.txt  text28.txt  text32.txt  text36.txt  text40.txt

debtop@DebTop:~/playpen/dir01$ rmdir ../dir03

rmdir: не удалось удалить '../dir03': Каталог не пуст

debtop@DebTop:~/playpen/dir01$ rm -r ../dir03

debtop@DebTop:~/playpen/dir01$ ls ..

dir00  dir04  dir07  dir10  dir13  dir16  dir19  dir22  dir25  dir28  dir31  dir34  dir37
dir40

dir01  dir05  dir08  dir11  dir14  dir17  dir20  dir23  dir26  dir29  dir32  dir35  dir38
dir41
```

```
dir02  dir06  dir09  dir12  dir15  dir18  dir21  dir24  dir27  dir30  dir33  dir36  dir39
dir42
```

cp — copy, копирование

Кроме создания и удаления файлов и директорий терминал позволяет их копирование.

Снова перейдем в директорию dir02 и скопируем файл text00.txt в директорию dir01:

cp text00.txt /home/debtop/playpen/dir01

Поскольку **text00.txt** находится в текущей директории, до него можно не писать полный путь. Такое указание пути называется относительным. Путь до **dir01** мы указали полностью, от домашней директории, такой путь называется абсолютный. Можно было указать просто **../dir01**, тогда это был бы снова относительный путь.

Если попробовать повторить копирование теперь, когда в **dir01** опять появился файл **text00.txt**, внешне не произойдет ничего, но на самом деле файл будет заменен на файл с тем же именем без каких-либо вопросов.

```
debtop@DebTop:~/playpen/dir01$ cd ../dir02

debtop@DebTop:~/playpen/dir02$ cp text00.txt home/debtop/playpen/dir01

cp: невозможно создать обычный файл 'home/debtop/playpen/dir01': Нет такого файла или каталога

debtop@DebTop:~/playpen/dir02$ cp text00.txt /home/debtop/playpen/dir01

debtop@DebTop:~/playpen/dir02$ ls ../dir01

text00.txt  text16.txt  text20.txt  text24.txt  text28.txt  text32.txt  text36.txt  text40.txt

text13.txt  text17.txt  text21.txt  text25.txt  text29.txt  text33.txt  text37.txt  text41.txt

text14.txt  text18.txt  text22.txt  text26.txt  text30.txt  text34.txt  text38.txt  text42.txt

text15.txt  text19.txt  text23.txt  text27.txt  text31.txt  text35.txt  text39.txt
```

Каталоги копируются по тому же принципу, только нужно не забывать про добавление опции рекурсивного копирования:

cp -r dir00 dir03

```
debtop@DebTop:~/playpen$ ls

dir00  dir04  dir07  dir10  dir13  dir16  dir19  dir22  dir25  dir28  dir31  dir34  dir37
dir40

dir01  dir05  dir08  dir11  dir14  dir17  dir20  dir23  dir26  dir29  dir32  dir35  dir38
dir41

dir02  dir06  dir09  dir12  dir15  dir18  dir21  dir24  dir27  dir30  dir33  dir36  dir39
dir42

debtop@DebTop:~/playpen$ cp dir00 dir03

cp: не указан -r; пропускается каталог 'dir00'

debtop@DebTop:~/playpen$ cp -r dir00 dir03

debtop@DebTop:~/playpen$ cp -r dir00 dir03/

debtop@DebTop:~/playpen$ cd dir03

debtop@DebTop:~/playpen/dir03$ ls

dir00       text05.txt  text11.txt  text17.txt  text23.txt  text29.txt  text35.txt  text41.txt
text00.txt  text06.txt  text12.txt  text18.txt  text24.txt  text30.txt  text36.txt  text42.txt
text01.txt  text07.txt  text13.txt  text19.txt  text25.txt  text31.txt  text37.txt
text02.txt  text08.txt  text14.txt  text20.txt  text26.txt  text32.txt  text38.txt
text03.txt  text09.txt  text15.txt  text21.txt  text27.txt  text33.txt  text39.txt
text04.txt  text10.txt  text16.txt  text22.txt  text28.txt  text34.txt  text40.txt
```

Если при копировании каталога или файла в качестве адреса копирования указать имя несуществующей директории (или файла), то система автоматически воспримет это как команду создать файл или каталог с таким именем.

Когда вы хотите скопировать не всю директорию целиком, а только ее содержимое, укажите слэш после места назначения, а после него поставьте звездочку (*):

cp -r dir00/* dir03/dir

Принцип действия команд rm и cp очень простой и очевидный, но не стоит забывать, что новичку многое кажется сложнее, чем есть на самом деле.

mv, move — перемещение

Третья команда для управления файлами — **mv** — перемещает файлы и директории.

Принцип действия полностью аналогичен предыдущим двум командам:

mv что_перемещаем куда_перемещаем

```
debtop@DebTop:~/playpen$ mv dir16 dir19/dir

debtop@DebTop:~/playpen$ ls dir19

dir         text05.txt  text11.txt  text17.txt  text23.txt  text29.txt  text35.txt  text41.txt

text00.txt  text06.txt  text12.txt  text18.txt  text24.txt  text30.txt  text36.txt  text42.txt

text01.txt  text07.txt  text13.txt  text19.txt  text25.txt  text31.txt  text37.txt

text02.txt  text08.txt  text14.txt  text20.txt  text26.txt  text32.txt  text38.txt

text03.txt  text09.txt  text15.txt  text21.txt  text27.txt  text33.txt  text39.txt

text04.txt  text10.txt  text16.txt  text22.txt  text28.txt  text34.txt  text40.txt
```

При перемещении каталогов мы перемещаем их целиком так, что они перестают существовать в исходном местоположении. То есть, каталог **dir16** из предыдущего примера больше не существует в каталоге **~/playpen**. Но это не всегда удобно, бывает необходимо

переместить не сам каталог, а только содержимое — все файлы, каталоги и файлы, вложенные в них. Для этого используется опция —**v[erbose]**, покажем на примере. Переместим все содержимое папки **dir19** в папку **dir17** командой mv, оставив папку **dir19** пустой:

mv -v dir19/* dir17/

```
debtop@DebTop:~/playpen$ ls

dir00  dir04  dir08  dir12  dir17  dir21  dir25  dir29  dir33  dir37  dir41

dir01  dir05  dir09  dir13  dir18  dir22  dir26  dir30  dir34  dir38  dir42

dir02  dir06  dir10  dir14  dir19  dir23  dir27  dir31  dir35  dir39

dir03  dir07  dir11  dir15  dir20  dir24  dir28  dir32  dir36  dir40

debtop@DebTop:~/playpen$ man mv

debtop@DebTop:~/playpen$ ls dir19

dir         text05.txt  text11.txt  text17.txt  text23.txt  text29.txt  text35.txt  text41.txt

text00.txt  text06.txt  text12.txt  text18.txt  text24.txt  text30.txt  text36.txt  text42.txt

text01.txt  text07.txt  text13.txt  text19.txt  text25.txt  text31.txt  text37.txt

text02.txt  text08.txt  text14.txt  text20.txt  text26.txt  text32.txt  text38.txt

text03.txt  text09.txt  text15.txt  text21.txt  text27.txt  text33.txt  text39.txt

text04.txt  text10.txt  text16.txt  text22.txt  text28.txt  text34.txt  text40.txt

debtop@DebTop:~/playpen$ mv -v dir19/* dir17/

переименован 'dir19/dir' -&gt; 'dir17/dir'

переименован 'dir19/text00.txt' -&gt; 'dir17/text00.txt'
```

```
переименован 'dir19/text01.txt' -&gt; 'dir17/text01.txt'

переименован 'dir19/text02.txt' -&gt; 'dir17/text02.txt'

переименован 'dir19/text03.txt' -&gt; 'dir17/text03.txt'

переименован 'dir19/text40.txt' -&gt; 'dir17/text40.txt'

переименован 'dir19/text41.txt' -&gt; 'dir17/text41.txt'

переименован 'dir19/text42.txt' -&gt; 'dir17/text42.txt'

debtop@DebTop:~/playpen$ ls

dir00  dir04  dir08  dir12  dir17  dir21  dir25  dir29  dir33  dir37  dir41

dir01  dir05  dir09  dir13  dir18  dir22  dir26  dir30  dir34  dir38  dir42

dir02  dir06  dir10  dir14  dir19  dir23  dir27  dir31  dir35  dir39

dir03  dir07  dir11  dir15  dir20  dir24  dir28  dir32  dir36  dir40

debtop@DebTop:~/playpen$ ls dir19

debtop@DebTop:~/playpen$ ls dir17

dir         text05.txt  text11.txt  text17.txt  text23.txt  text29.txt  text35.txt  text41.txt

text00.txt  text06.txt  text12.txt  text18.txt  text24.txt  text30.txt  text36.txt  text42.txt

text01.txt  text07.txt  text13.txt  text19.txt  text25.txt  text31.txt  text37.txt

text02.txt  text08.txt  text14.txt  text20.txt  text26.txt  text32.txt  text38.txt

text03.txt  text09.txt  text15.txt  text21.txt  text27.txt  text33.txt  text39.txt

text04.txt  text10.txt  text16.txt  text22.txt  text28.txt  text34.txt  text40.txt
```

Глубина вложенности

Можно пойти дальше — создать папку с большой вложенностью **dir18** и переместить только файлы первого уровня в папку **dir19**:

find dir18/ -maxdepth 1 -type f -print0 | xargs -0 mv -t dir19

Эта сложная команда найдет (**find**) все файлы (**-type f**) с глубиной вложенности 1 (**-maxdepth 1**) внутри dir18 и переместит их в **dir19** как обычные файлы (**mv -t**). Часть «**xargs -0**» служит в роли команды, связующей две других и позволяет им работать вместе. С глубиной вложенности **-maxdepth** можно экспериментировать, задавать ей разные значения (1, 2, 3 и т.д.) получая новый результат каждый раз.

Команды для архивирования

Важный аспект работы с файлами — возможность создавать архивы. В Linuxe это реализовано с помощью утилит терминала. Рассмотрим несколько базовых команд архивирования:

- **gzip** — создать архив. Работает следующим образом: ***gzip файл.расширение***. Можно сразу несколько, например, **gzip text12.txt text13.txt**.

```
debtop@DebTop:~/playpen$ cd dir00

debtop@DebTop:~/playpen/dir00$ gzip text14.txt

debtop@DebTop:~/playpen/dir00$ ls

text00.txt  text07.txt  text14.txt.gz  text21.txt  text28.txt  text35.txt  text42.txt

text01.txt  text08.txt  text15.txt     text22.txt  text29.txt  text36.txt

text02.txt  text09.txt  text16.txt     text23.txt  text30.txt  text37.txt

text03.txt  text10.txt  text17.txt     text24.txt  text31.txt  text38.txt

text04.txt  text11.txt  text18.txt     text25.txt  text32.txt  text39.txt

text05.txt  text12.txt  text19.txt     text26.txt  text33.txt  text40.txt

text06.txt  text13.txt  text20.txt     text27.txt  text34.txt  text41.txt
```

- **gunzip** — распаковать архив — ***gunzip файл.расширение.gz***.

```
debtop@DebTop:~/playpen/dir00$ ls

text00.txt  text07.txt  text14.txt.gz  text21.txt  text28.txt  text35.txt  text42.txt

text01.txt  text08.txt  text15.txt     text22.txt  text29.txt  text36.txt

text02.txt  text09.txt  text16.txt     text23.txt  text30.txt  text37.txt

text03.txt  text10.txt  text17.txt     text24.txt  text31.txt  text38.txt

text04.txt  text11.txt  text18.txt     text25.txt  text32.txt  text39.txt

text05.txt  text12.txt  text19.txt     text26.txt  text33.txt  text40.txt

text06.txt  text13.txt  text20.txt     text27.txt  text34.txt  text41.txt

debtop@DebTop:~/playpen/dir00$ gunzip text14.txt.gz

debtop@DebTop:~/playpen/dir00$ ls

text00.txt  text06.txt  text12.txt  text18.txt  text24.txt  text30.txt  text36.txt  text42.txt

text01.txt  text07.txt  text13.txt  text19.txt  text25.txt  text31.txt  text37.txt

text02.txt  text08.txt  text14.txt  text20.txt  text26.txt  text32.txt  text38.txt

text03.txt  text09.txt  text15.txt  text21.txt  text27.txt  text33.txt  text39.txt

text04.txt  text10.txt  text16.txt  text22.txt  text28.txt  text34.txt  text40.txt

text05.txt  text11.txt  text17.txt  text23.txt  text29.txt  text35.txt  text41.txt
```

- **zip** — более привычный формат архива. Чтобы передать файл на другую операционную систему, и чтобы адресат мог его распаковать лучше использовать утилиту **zip**.

Утилита zip терминала Linux позволяет упаковать сразу несколько файлов в один архив (gzip также позволяет поместить несколько файлов в архив, но с ней сложнее разобраться) с помощью рекурсивной опции **-r**. Текст команды Linux:

zip -r texts.zip text12.txt text13.txt

В данном случае **texts.zip** будет названием архива, а **text12.txt** и **text13.txt** — файлы, которые нужно поместить в архив.

```
debtop@DebTop:~/playpen/dir00$ zip -r texts.zip text12.txt text13.txt
 adding: text12.txt (stored 0%)
 adding: text13.txt (stored 0%)
debtop@DebTop:~/playpen/dir00$ ls
text00.txt  text06.txt  text12.txt  text18.txt  text24.txt  text30.txt  text36.txt  text42.txt
text01.txt  text07.txt  text13.txt  text19.txt  text25.txt  text31.txt  text37.txt  texts.zip
text02.txt  text08.txt  text14.txt  text20.txt  text26.txt  text32.txt  text38.txt
text03.txt  text09.txt  text15.txt  text21.txt  text27.txt  text33.txt  text39.txt
text04.txt  text10.txt  text16.txt  text22.txt  text28.txt  text34.txt  text40.txt
text05.txt  text11.txt  text17.txt  text23.txt  text29.txt  text35.txt  text41.txt
```

- **tar** — создание резервных копий. Утилита **tar** создает архивы с расширением **.tar**, утилиту обычно используют для резервного копирования так как она не сжимает файлы, а только собирает в архив. Чтобы создать архив .tar, выполните команду с опциями —**cvf**. Например:

tar -cvf newark.tar text07.txt ../dir22

Создаст архив с именем **newark.tar**, в который будут помещены файл **text.07.txt** и все файлы из каталога **dir22**, располагающегося внутри родительского каталога.

Для просмотра содержимого **.tar** архива используются опции **-tvf**:

tar -tvf newark.tar

```
debtop@DebTop:~/playpen/dir00$ tar -cvf newark.tar text07.txt ../dir22

text07.txt

tar: Удаляется начальный `../' из имен объектов

../dir22/

../dir22/text04.txt

tar: Удаляются начальные `../' из целей жестких ссылок

../dir22/text07.txt

../dir22/text40.txt

../dir22/text08.txt

../dir22/text32.txt

../dir22/text28.txt

../dir22/text26.txt

../dir22/text11.txt

../dir22/text25.txt

debtop@DebTop:~/playpen/dir00$ tar -tvf newark.tar

-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 text07.txt

drwxr-xr-x debtop/debtop     0 2020-11-25 13:44 dir22/

-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text04.txt
```

```
-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text07.txt

-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text40.txt

-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text08.txt

-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text32.txt

-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text28.txt

-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text26.txt

-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text11.txt

-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text25.txt
```

Опции для распаковки **.tar** архива **-xvf**:

tar -xvf newark.tar

```
-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text27.txt

-rw-r--r-- debtop/debtop     0 2020-11-25 13:44 dir22/text39.txt

hrw-r--r-- debtop/debtop     0 2020-11-25 13:44 text07.txt ссылка на text07.txt

debtop@DebTop:~/playpen/dir00$ ls

arch.tar    text05.txt  text11.txt  text17.txt  text23.txt  newark.tar  text35.txt  text41.txt

dir00       text06.txt  text12.txt  text18.txt  text24.txt  scrolls.tar text36.txt  text42.txt

text01.txt  text07.txt  text13.txt  text19.txt  text25.txt  text31.txt  text37.txt

text02.txt  text08.txt  text14.txt  text20.txt  text26.txt  text32.txt  text38.txt

text03.txt  text09.txt  text15.txt  text21.txt  text27.txt  text33.txt  text39.txt
```

```
text04.txt  text10.txt  text16.txt  text22.txt  text28.txt  text34.txt  text40.txt  texts.zip
```

После выполнения в текущий каталог будут скопированы все файлы из архива, но архив не будет удален. Удалить его можно, выполнив **rm**.

Команды для работы с текстом

Nano, Vim — редактирование текстовых файлов

Как посмотреть содержимое файла в Linux терминале? Через утилиты терминала Linux часто требуется открыть или отредактировать текстовые файлы. Для этого перейдем в директорию **dir42** и откроем **text42.txt** с помощью **Nano** — простого текстового редактора.

nano text42.txt

Запишем текст в файле, затем сохраним его сочетанием Ctrl+O (^ в терминале Linux всегда значит Ctrl) и выйдем сочетанием Ctrl+X. Nano позволяет редактировать файлы прямо из окна терминала, обладает базовым, но не очень богатым набором функций. Кроме **Nano** есть еще более продвинутый **Vim**, который открывает гораздо большие возможности редактирования файлов. Только Vim не всегда установлен в систему и из него сложнее выйти, потому что он не отображает подсказки. Для выхода из Vim используйте **:q**, для выхода без сохранения изменений — **:q!** и **:w** для сохранения всех изменений. Обязательно сначала использовать двоеточие, так как оно активирует командный режим.

grep — поиск

Чтобы найти текст в файле используйте **grep**, эта команда выведет соответствующие запросу строки текста. Можно искать словом, строкой или регулярным выражением, а вывод может быть файлом или папкой, совпадающим или наоборот — несовпадающим. Полезно использовать **grep** при поиске по большим логам.

Кроме непосредственно grep есть варианты **pgrep, fgrep, egrep**, которые выполняют ту же функцию, но для других целей. Часто **grep** используют в сочетании с другими командами, чтобы упростить работу с большими объемами данных, ниже мы еще увидим примеры такого использования.

head, tail — начало и конец текста

Команды head и tail используются для вывода первых и последних строк текста в одном или нескольких документах.

Синтаксис команды:

head/tail опции файл

По умолчанию команды выводят 10 строк текста, но количество можно изменить с помощью опции **-n**. Специально для эксперимента мы создали большой текстовый файл, из которого хотим получить ровно 1 первую или последнюю большую строку:

head -n 1 text00.txt

tail -n 1 text00.txt

```
debtop@DebTop:~/playpen/dir00$ head -n 1 text00.txt

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ut lacus ut justo blandit
mattis

. Sed magna elit, pharetra ac nulla eu, vulputate elementum ex. Donec porta sapien at dui
males

uada, quis pretium neque volutpat. Aliquam a tincidunt mi. Nunc posuere vehicula velit, sed
int

erdum lorem ullamcorper at. Aliquam finibus ultrices metus pharetra elementum. Aliquam erat
vol

utpat. In sit amet sapien commodo dolor feugiat sodales at vitae turpis. Donec ullamcorper
dolo

r quis lacus tempor, ac aliquet sem consectetur. Duis nec dignissim ligula. Sed gravida sed
arc

u quis commodo. Aliquam vestibulum nunc sed viverra imperdiet. Suspendisse accumsan
sollicitudi

n ex, interdum facilisis eros efficitur non.

debtop@DebTop:~/playpen/dir00$ tail -n 1 text00.txt

Aenean non ullamcorper diam. Ut iaculis vehicula libero sit amet elementum. Nullam ipsum
metus,

molestie vehicula dui in, tempus finibus nisl. In hac habitasse platea dictumst. In vehicula v
```

```
elit nec massa porttitor finibus. Nulla libero justo, egestas vitae fringilla ut, gravida eu
ri

sus. Ut eget risus erat. Aenean interdum pretium semper. Aliquam vehicula dolor non augue
lacin

ia sodales. Morbi molestie massa sed enim placerat, id euismod odio condimentum. Nunc aliquet
l

acinia lacus, quis mattis tortor viverra sed. Vestibulum ullamcorper aliquet nibh, nec lacinia

nunc tincidunt fringilla. Mauris sollicitudin nec lectus eget mollis. Aliquam eu accumsan
nulla.
```

С помощью опции **-c** можно запросить уже не строки, а байты.

Например, команды для получения 100 байт текста:

head -c 100 text00.txt

```
debtop@DebTop:~/playpen/dir00$ tail -c 100 text00.txt

ia nunc tincidunt fringilla. Mauris sollicitudin nec lectus eget mollis. Aliquam eu accumsan
nulla.

debtop@DebTop:~/playpen/dir00$ head -c 100 text00.txt

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ut lacus ut justo blandit
mattis. Sed
```

tail -c 100 text00.txt

Другие опции можно посмотреть через **man head**.

NAME

head - output the first part of files

SYNOPSIS

head [OPTION]... [FILE]...

DESCRIPTION

Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name.

With no FILE, or when FILE is -, read standard input.

Mandatory arguments to long options are mandatory for short options too.

-c, --bytes=[-]NUM
print the first NUM bytes of each file; with the leading '-', print all but the last NUM bytes of each file

-n, --lines=[-]NUM
print the first NUM lines instead of the first 10; with the leading '-', print all but the last NUM lines of each file

-q, --quiet, --silent
never print headers giving file names

-v, --verbose

Команды для управления процессами

Операционная система Linux не имеет встроенного графического диспетчера задач в отличие от Windows, вместо этого всеми процессами можно управлять из терминала. Говорят, что Linux стабильнее винды, но иногда все равно приходится гуглить «Как завершить процесс Linux», потому что что-то зависло.

kill / pkill / killall — завершить процесс

Первый вопрос — почему так много, какую команду использовать? Давайте по порядку:

- **kill** — убить процесс по его идентификатору (Process ID);
- **pkill** — убить процесс по его имени;
- **killall** — убить все процессы с указанным именем.

ps / pgrep — узнать ID процесса

Если вы не компьютер, то вряд ли вам известен ID нужного процесса. PS (Process Status) выведет на экран информацию о запущенных процессах. Чтобы показать список всех активных процессов, используется **ps axu**.

```
debtop@DebTop:~/playpen$ top

top - 14:21:22 up 8 days, 21:57,  3 users,  load average: 0,49, 0,66, 0,65

Tasks: 181 total,   1 running, 180 sleeping,   0 stopped,   0 zombie

%Cpu(s):  0,8 us,  0,4 sy,  0,0 ni, 98,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st

MiB Mem :   7856,4 total,   4009,2 free,   1944,6 used,   1902,6 buff/cache

MiB Swap:   8068,0 total,   7913,4 free,    154,6 used.   5390,8 avail Mem
```

```
PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND</b>

 690 root      20   0  309888  34416  16856 S   2,0   0,4   5:46.67 Xorg
```

```
11426 debtop    20   0 1195064  78272  62036 S   1,7   1,0   0:15.55
konsole

 936 debtop    20   0 2994244  59840  36732 S   1,3   0,7   6:10.33
kwin_x11

11586 debtop    20   0 9188248 513296 127392 S   1,0   6,4  14:44.20
chrome

11068 debtop    20   0  973096 310508 142424 S   0,3   3,9   2:27.42
chrome

11107 debtop    20   0  390416 110176  62368 S   0,3   1,4   0:44.15
chrome

11516 debtop    20   0 9177080 384808 114828 S   0,3   4,8   0:58.68
chrome

   1 root      20   0  169620   6872   5272 S   0,0   0,1   0:02.15
systemd

   2 root      20   0       0      0      0 S   0,0   0,0   0:00.02
kthreadd

   3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00
rcu_gp

   4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00
rcu_par_gp

   6 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/0:0H-
kblockd

   8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00
mm_percpu_wq

   9 root      20   0       0      0      0 S   0,0   0,0   0:00.07
ksoftirqd/0

  10 root      20   0       0      0      0 I   0,0   0,0   0:09.29
```

```
rcu_sched

  11 root      20   0       0      0      0 I   0,0   0,0   0:00.00
rcu_bh

  12 root      rt   0       0      0      0 S   0,0   0,0   0:00.17
migration/0

  14 root      20   0       0      0      0 S   0,0   0,0   0:00.00
cpuhp/0

  15 root      20   0       0      0      0 S   0,0   0,0   0:00.00
cpuhp/1

  16 root      rt   0       0      0      0 S   0,0   0,0   0:00.37
migration/1

  17 root      20   0       0      0      0 S   0,0   0,0   0:00.15
ksoftirqd/1

  19 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/1:0H-kblockd
```

Проблема в том, что найти нужный процесс среди этого полотна очень сложно. Решений этой проблемы несколько.

1. **top** (Table Of Processes) — каждые 2 секунды обновляет активные процессы. Колонка PID указывает на ID процесса, USER на пользователя и т.д. Удобно. Если процесс активен постоянно, но уследить за постоянно обновляющимся списком не всегда получается.

2. Более продвинутые утилиты командной строки Linux — **atop, htop**. Отображают не только активные процессы с обновлением в две секунды, но и контроль над загрузкой и т.д. Утилиты полезны для системных администраторов, но для обычного пользователя все-таки перебор.

3. **ps axu | grep имя_процесса**. С помощью утилиты **grep** можно вычленить нужный процесс, если известно его имя.

```
debtop@DebTop:~/playpen$ ps axu | grep bash
```

```
debtop   11468  0.0  0.0   8316  5192 pts/1    Ss   13:20   0:00 /bin/bash

debtop   14232  0.0  0.0   6224   888 pts/1    S+   14:22   0:00 grep bash
```

4. **pidof** — самый короткий способ выяснить ID процесса, если известно его имя.

```
debtop@DebTop:~/playpen$ pidof bash

11468
```

Пакетные менеджеры или как установить файл

По сравнению с Windows, в Linux установка программы требует чуть больше усилий. Даже для установки браузера Chrome придется залезть в терминал.

dpkg — установить файл с расширением .deb (только для систем на Debian).

Многие программы, скачанные из интернета, которые поддерживаются ОС Linux, имеют расширение **.deb**. Это файлы, оптимизированные под дистрибутивы на основе Debian (Debian, Ubuntu и Ubuntu-based, Mint и т.д.). Когда вы скачаете и попытаетесь открыть файл (точнее пакет) **.deb** кликом указателя скорей всего ничего не произойдет, потому что это необходимо делать через **dpkg**:

dpkg -i package.deb — установит пакет .deb;

dpkg —remove package — удалит пакет. Да, при удалении не требуется указывать расширение, так как пакет уже установлен в системе и идентифицируется по имени.

Примечание: Если вы работали в Windows, вы помните, что приложения удаляются не полностью, остаются пустые папки и записи в реестре. У Linux тоже есть такая особенность, остаточные файлы и папки называются dependencies (зависимости). Избавиться от них в Linux гораздо проще, чем в ОС от Майкрософт. Достаточно ввести:

dpkg —purge package.

apt — менеджер установки пакетов Debian-based

Интересная функция Linux — пакетные менеджеры. В каждом дистрибутиве есть свой пакетный менеджер, который может запрашивать и устанавливать пакеты из удаленных репозиториев. Это очень удобно — требуется знать только название пакета, который желаете установить и команду. Пакетный менеджер по умолчанию в Debian и дистрибутивах на основе Debian — **apt** (Advanced Packaging Tool).

Для установки пакета с помощью **apt** необходимо ввести команду:

apt install имя_пакета

Часто для этой операции потребуются права суперпользователя, но об этом см. раздел ниже.

Нередко можно встретить **apt-get** вместо **apt** — это тот же самый пакетный менеджер, только более старая его версия. Тем не менее оба варианта существуют и работают. Пакетный менеджер **apt** объединяет сразу несколько прежних команд (**apt-get** и **apt-cache** в **apt install** и **apt search**), а также оптимизирует процесс установки пакетов.

Команда для удаления установленного пакета с помощью **apt**:

apt remove имя_пакета

Для удаления зависимостей вместо **remove** используется **purge**:

apt purge имя_пакета

Для автоматического удаления всего лишнего из системы используется **apt autoremove**.

Автоматическое удаление лишнего — это как встроенный в систему менеджер очистки, команда удаляет все файлы и зависимости, которые больше не требуются в системе, например, потому что соответствующего им приложения больше нет в системе.

yum — менеджер установки пакетов Red Hat

Red Hat — один из наиболее популярных дистрибутивов Linux. Утилита **yum** написана на python. Это пакетный менеджер Red Hat и дистрибутивов на основе Red Hat, например CentOS. Менеджер **yum** (Yellowdog Updater, Modified) использует формат пакетов **rpm** вместо **deb**, но команды остались прежними:

- ***yum install*** для установки пакета;
- ***yum remove*** для удаления;
- ***yum autoremove*** для очистки.

При удалении пакета менеджер yum по умолчанию всегда сохраняет файлы конфигурации, которые отличаются от изначальных по умолчанию и не имеет команды аналогичной **apt purge**. Впрочем, способы достижения аналогичного результата существуют всегда.

pacman — менеджер пакетов ArchLinux

Еще один дистрибутив, имеющий свой пакетный менеджер — ArchLinux. Утилита **pacman** написана на языке C (Си). ArchLinux имеет только одну версию — текущую, которая постоянно обновляется, и для этой системы pacman подходит идеально. Пакеты здесь представляют собой архивы формата **.tar.xz**.

Пакетный менеджер pacman работает иначе чем **yum**, **apt** или **dpkg**. Чтобы установить или удалить пакет с помощью pacman используются опции:

- **-S** устанавливает пакет ArchLinux;
- **-Sw** скачивает, но не устанавливает пакет;
- **-U** для установки пакетов с локального компьютера;
- **-s** для поиска пакетов;
- **-u** для обновления установленных пакетов;
- **-R** для удаления;
- **-Rn** для удаления резервных копий файлов конфигурации;
- **-Rs** для удаления зависимостей.

После опции всегда идет имя пакета, т.е.:

pacman -S имя_пакета

Команды управления пользователями

sudo — запуск команд и приложений через терминал в Linux от имени администратора (суперпользователя)

Все важные действия в Linuxe требуют подтверждения от имени администратора, в Linux администратор называется суперпользователем или root. Чтобы выполнить команду от имени суперпользователя, нужно ввести перед ней **sudo**. Это позволит выполнять команды от имени суперпользователя до окончания текущей сессии в терминале.

Если вы не хотите вводить **sudo** каждый раз, когда терминал просит это сделать, вы можете переключиться в режим суперпользователя. Для этого выполните **su**, введите пароль, тогда вы сможете выполнять любые команды по умолчанию от имени суперпользователя. Однако, лучше поступать так только, если вы точно знаете, что делаете. Сломать что-то важное в системе получится вряд ли, но вероятность возрастает лучше использовать **sudo**.

useradd (adduser) / userdel (deluser) / usermod — добавить, удалить, изменить пользователя

С помощью этих команд вы сможете управлять учетными записями пользователей. Полезно иметь под рукой такую возможность в сетевом домене или домашней сети, выдавать и ограничивать права.

Для добавления пользователя используйте ***useradd имя_пользователя.***

Пароль для нового пользователя задается командой ***passwd имя_пользователя***.

Аналогичным образом, командой ***userdel имя_пользователя***, можно удалить созданную учетную запись.

Третья команда — ***usermod*** позволяет изменять учетные записи пользователей: изменять домашнюю директорию, имя учетной записи, оболочку входа в систему, «срок годности» пароля и т.д.

У каждой из трех команд есть свои опции, знать которые новичкам нет необходимости, но знать об их существовании не возбраняется.

Команды выключения и перезагрузки

Linux можно выключить или перезагрузить из командной строки. Зачем это делать? Во-первых, команды очень полезны, если отсутствует графическая оболочка системы — окружение пользователя, и вся работа происходит из терминала. Во-вторых, различные команды или опции помогут произвести перезагрузку или выключение с разными параметрами.

shutdown — выключение

shutdown опции время сообщение

Время — отсрочка выключения. Можно указывать:

- в 24-часовом формате когда нужно указать точное время выключения с точностью до минуты **shutdown 10:10**;
- в формате **+мин**, например, **shutdown +10** выключит компьютер через 10 минут после выполнения. Команды **shutdown +0** и shutdown now выключат компьютер немедленно.

Сообщение (wall) — текстовое сообщение, отправляемое пользователям компьютеров в сети. Если требуется добавить комментарий с причиной выключения. При добавлении сообщения обязательно указывать время.

Некоторые **опции**:

- **-H (Halt)** — останавливает все функции процессора, но не выключает компьютер. Halt можно использовать как самостоятельную команду, отдельно от **shutdown**;
- **-P** — полное выключение компьютера. Можно использовать как команду **poweroff**;
- **-r (reboot)** — перезагрузка. В качестве команды — **reboot**;
- **-k** — не производить никаких действий, отправить сообщение;
- **-c** — отменить запланированное выключение. Можно отменить выключение, только если не было задано время **+0** или **now**.

Заключение, несколько советов

Мы рассмотрели основные команды Linux с примерами, надеемся, что они окажутся вам полезными. Напоследок приведем несколько советов по использованию терминала одной строкой.

Двойной символ & (амперсанд)

Предназначен для выполнения нескольких команд последовательно:

команда1 && команда2 && команда3

команда3 выполнится только после успешного исполнения команды2, а команда2 — после команды1.

Вертикальная черта | (pipe)

Вводит результат первой команды в последующую. Например, следующая команда добавит таблицу процессов к команде поиска:

ps axu | grep имя_процесса

&& и || можно комбинировать. Например вот так: команда1 && команда2 || команда3

Стрелки вверх и вниз на клавиатуре

Помогают осуществлять навигацию по последним командам. Стрелка вверх (Ctrl+P) — предыдущая выполненная команда, стрелка вниз (Ctrl+N) — следующая.

history — история

Если вы забыли, как вы выполнили действие пять дней назад, а вот оно снова потребовалось, выполните **history**, и терминал выведет на экран последнюю тысячу команд.

Новая вкладка bash

Не всегда удобно иметь несколько окон. Некоторые терминалы, как и браузеры, дают возможность открыть несколько вкладок сочетанием клавиш **Ctrl+Shift+T.**

Копирование и вставка, прерывание команды

Пробовали ли вы копировать текст и вставлять в терминал? Пробовали **Ctrl+C**? **Ctrl+C** прервет выполнение текущей команды, например таблицы процессов, сбросит текст, введенный в строку. Скопировать текст из bash — **Ctrl+Shift+C**, вставить — **Ctrl+Shift+V** (для MacOS — **Command+C** и **Command+V** соответственно).

Revision #1
Created 2023-10-25 19:58:17 UTC by admin
Updated 2023-10-25 19:59:03 UTC by admin