

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ

Раздел #1

АВТОСБОРКА ПРИЛОЖЕНИЙ

Консоль, терминал и командная строка UNIX

Лабораторная работа #11

Основы работы с терминалом и командная строка в UNIX



РАЗРАБОТАЛ

СЕРГЕЙ СТАНКЕВИЧ (SERHEY STANKEWICH)

ЛАБОРАТОРНАЯ РАБОТА #11

Основы работы с терминалом и командная строка в UNIX

Цель работы

Изучить командную строку UNIX и получить основы работы с терминалом в дистрибутивах Linux (UNIX).

Краткие теоретические сведения.

Основные понятия и определения

Командная строка – это специальный интерфейс, через который пользователи получают персональную рабочую среду. Командная строка в UNIX это самый непосредственный способ выполнения множества небольших задач администрирования.

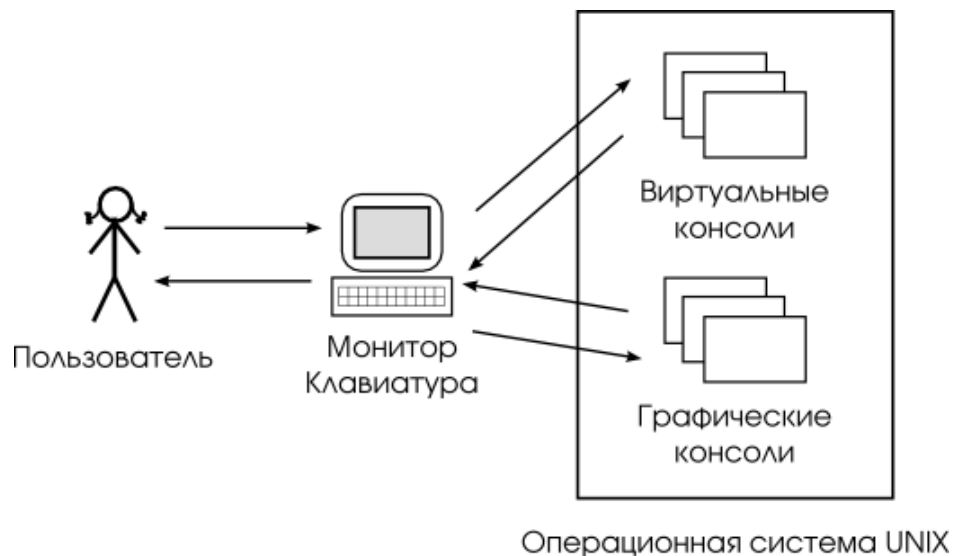
Командная строка имеет строго определённый формат:



Есть два вида интерфейса командной строки:

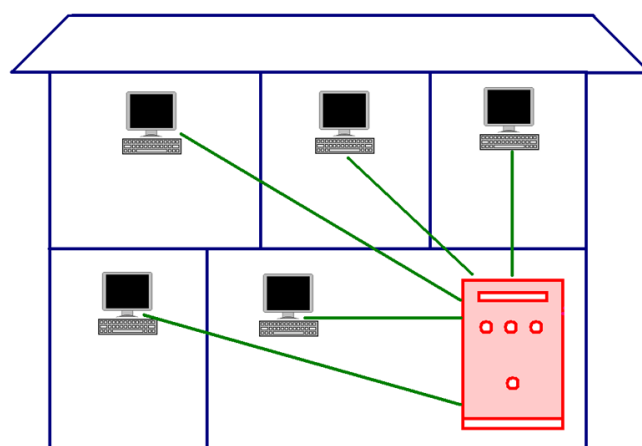
- Консоль
- Терминал

То есть добраться до командной строки можно этими двумя способами.



Command Line Interface (CLI) – это управление программами с помощью команд. Такой интерфейс называется консолью. Он встроен в ядро системы, и доступен, даже если графический интерфейс не запустится. При загрузке консоли пользователь видит только интерфейс командной строки, и больше ничего. Консоль может использоваться в случаях, когда происходит сбой в работе операционной системы, то есть в аварийных случаях, однако при этом возникают неудобства, при работе необходимо читать печатную документацию.

Операционная система UNIX изначально задумывалась как многопользовательская. Пользователи, находясь в разных местах могли подключиться к одному компьютеру с помощью специальных устройств, которые назывались *терминалами*. Терминал состоял из монитора и клавиатуры.



Терминал и *командная оболочка*, это разные понятия. В современных Unix-подобных системах под терминалом обычно понимают программу эмуляции терминала.

В настоящий момент терминалы в дистрибутивах пост-UNIX представлены в виде пользовательского графического интерфейса (**Graphical user interface**,

сокращенно **GUI**). По сути, терминалом сейчас называется специальная графическая программа, программа-эмулятор, которая непосредственно связана и общается с ядром ОС.

Эмулятор терминала (далее – терминал) можно запустить разными способами, но наиболее удобный использовать горячие клавиши.

Ctrl + Alt + t

Сочетание горячих клавиш **Ctrl + Alt + F2** (или любая из 6 функциональных клавиш) откроет виртуальную консоль, иначе **tty**.

Как правило в дистрибутивах есть семь полноэкранных консолей. У каждой свой независимый сеанс.

Переключиться на одну из виртуальных консолей можно нажав сочетание клавиш:

- Ctrl+Alt+F1 - первая виртуальная консоль;
- Ctrl+Alt+F2 - вторая виртуальная консоль;
- Ctrl+Alt+F3 - третья виртуальная консоль;
- Ctrl+Alt+F4 - четвертая виртуальная консоль;
- Ctrl+Alt+F5 - пятая виртуальная консоль;
- Ctrl+Alt+F6 - шестая виртуальная консоль;
- Ctrl+Alt+F7 - седьмая виртуальная консоль, возврат в графический режим.

С первой по шестую с интерфейсом командной строки.

Для получения информации о том какой эмулятор терминала установлен в вашей системе почитайте раздел «Дополнительная информация».

Как и многие другие вещи в Linux-системах, эмулятор терминала можно установить по своему вкусу.

Командная оболочка

Командная оболочка – это отдельная программа в экосистеме Linux (UNIX).

Оболочка предоставляет основной интерфейс между пользователем, сидящим за своим терминалом, и операционной системой, если, конечно, пользователь не использует графический интерфейс. Оболочка принимает команды, введенные с клавиатуры, и передает их операционной системе для выполнения.

Существуют целый ряд оболочек из проекта GNU, включая `csh`, `ksh`, `zsh` и `bash`. Все они поддерживают описываемую здесь функциональность исходной оболочки (`sh`).

Команды оболочки можно разделить на определенные типы по назначению:

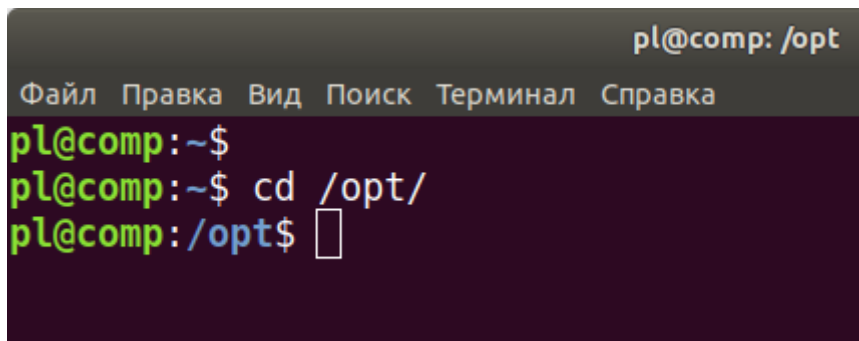
- Утилиты для работы с файлами
- Утилиты для управления процессами

Приглашение командной строки

Особое внимание уделим *приглашению командной строки* в терминале.

Вначале **приглашения** стоят имя пользователя и имя компьютера, разделённые символом @. Между символами : и \$ указывается имя текущего каталога (директории).

Символ тильды ~ в Linux-системах – это сокращенное обозначение домашнего каталога пользователя (/**home/user**). Именно эту директорию Bash делает текущей по умолчанию. Знак доллара \$ обозначает, что мы работаем под непривилегированным пользователем. Если находиться в системе под администратором (с правами суперпользователя), вместо доллара будет знак решетки #. Это знаки привилегии режима работы.



```
pl@comp: /opt
Файл Правка Вид Поиск Терминал Справка
pl@comp:~$
pl@comp:~$ cd /opt/
pl@comp:/opt$
```

Регулярные выражения

Регулярные выражения (Regular expression), это набор символов и/или метасимволов, которые определяют шаблон поиска. Регулярные выражения используют символическую форму записи для идентификации шаблонов в тексте.

Шаблон поиска, это метод описания поискового запроса с использованием метасимволов. Типы шаблонов поиска состоят из *символов* и *операций*. Символы в свою очередь состоят из *лексем* (простых символов) и *метасимволов*.

Метасимвол, это зарезервированный символ, который имеет особое назначение в регулярном выражении. Они используются для определения более сложных критериев сопоставления.

Вот пример метасимволов:

`. * [] ^ $ { } \ + ? | ()`

Часто возникает необходимость интерпретировать метасимволы как литералы, а не как метасимволы. Для этого используется механизм *экранирования*.

Экранирование, это замена в тексте управляющих символов на соответствующие текстовые подстановки. Экранировать можно одиночные символы или группы символов.

Типы регулярных выражений. Существует два основных типа регулярных выражений. Это регулярные выражения, определяемые стандартом POSIX и определяемые языками программирования. В экосистеме Linux (UNIX) используются выражения, определенные стандартом POSIX, которые поддерживаются большинством инструментов командной строки. Стандарт POSIX делит реализации регулярных выражений на два вида: традиционные, основные, базовые (Basic Regular Expressions, BRE) и расширенные (Extended Regular Expressions, ERE).

Эти виды выражений вносят некоторую путаницу с метасимволами. Метасимволами в BRE являются только «`. * [] ^ $`», остальное литералы. Однако в ERE уже 14 метасимволов: «`. * [] ^ $? { } \ + | ()`», остальное литералы.

Утилиты для работы с регулярными выражениями. Некоторые консольные команды были специально созданы для работы с регулярными выражениями, и наоборот, регулярные выражения активно используются с некоторыми утилитами. Такими командами являются:

- grep** – находит и выводит строки, соответствующие шаблону;
- find** – позволяет найти файлы по их именам;
- locate** – поддерживает простые (параметр `--regexr`) и расширенные (параметр `--regex`) регулярные выражения;
- less** – осуществляет поиск текста в файле.

Команда grep. Название `grep` в действительности произошло от фразы «global regular expression print» (глобальный поиск с помощью регулярного выражения и вывод). Можно перефразировать: «*Искать везде соответствующие регулярному выражению строки и выводить их*». Как видите, `grep` имеет некоторое отношение к регулярным выражениям. В сущности, `grep` просматривает текстовые файлы в поисках совпадений с указанным регулярным выражением и выводит в стандартный вывод все строки с такими совпадениями.

Программа `grep` имеет следующий синтаксис:

`grep [параметры] регулярное_выражение [файл...]`

Ниже показано, как выполнить простой поиск в списке файлов:

```
[me@linuxbox ~]$ grep bzip dirlist*.txt
dirlist-bin.txt:bzip2
dirlist-bin.txt:bzip2recover
```

Команда `find`. Программа `find` поддерживает проверку, основанную на регулярном выражении. Существует одно важное обстоятельство, которое следует помнить, используя регулярные выражения в командах `find` и `grep`. Если `grep` выводит строку, содержащую совпадение с регулярным выражением, то `find` требует точного совпадения пути с регулярным выражением.

Команда `locate`. Программа `locate` поддерживает простые (параметр `--regex`) и расширенные (параметр `--regextype`) регулярные выражения. Благодаря этому можно выполнять те же операции, что производились предыдущими командами.

Редакторы `less` и `vim`. Эти программы поддерживают одинаковые способы поиска в тексте. Редактор `vim` поддерживает только простые регулярные выражения.

Хорошее понимание регулярных выражений
позволит вам творить настоящие чудеса.



Best of LUCK with it, and remember to HAVE FUN while you're learning :)
Sergey Stankevich



УПРАЖНЕНИЯ

Упражнение 1

Управление терминалом

Терминал в Linux – это программа первой необходимости. Она позволяет выполнить практически любое действие, такие как: операции с файлами, управление программным обеспечением, настройку системы и многое другое. Все эти действия производятся с помощью ввода специальных команд. Набирать такие команды с клавиатуры посимвольно неудобно и долго. Нам в помощь предусмотрены различные вспомогательные приемы. Для выполнения ряда действий в терминале предусмотрен список горячих клавиш, делающих работу в нем еще удобнее.

Горячие клавиши. Большинство комбинаций клавиш стандартны для «командной строки Linux», но часть из этих комбинаций специфичны для разных командных интерпретаторов (bash, dach, zsh, fish и т.п.), в них могут быть небольшие отличия в работе. Эта часть комбинаций прописана в «настройках по умолчанию» (например, в файле `/etc/inputrc` или в `/etc/bashrc`), которые тоже могут различаться в разных дистрибутивах.

Также некоторые клавиши могут быть настроены и перехватываться графической оболочкой (терминалом), в которой запущен командный интерпретатор.

Таким образом часть комбинаций относятся к «настройкам терминала». А другая часть к командному интерпретатору (*sh).

Клавиши из настроек терминала можно посмотреть, выполнив команду:

```
$ stty -a
speed 38400 baud; rows 7; columns 64; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D;
eol = <undef>; eol2 = <undef>; swtch = <undef>; start = ^Q;
stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W; lnext = ^V;
discard = ^O; min = 1; time = 0;
-parenb -parodd -cmspar cs8 -hupcl -cstopb cread -clocal -crtsct
```

Подробнее значения действия редактирования командной строки можно посмотреть в мануале (man).

```
$ man bash
```

Чтобы узнать какие клавиши привязаны к каким действиям редактирования командной строки— для этого можно воспользоваться командой «bind -P».


```
stank@DESKTOP-61Q8VTL:~$ bind -P
abort can be found on "\C-g", "\C-x\C-g", "\e\C-g".
accept-line can be found on "\C-j", "\C-m".
alias-expand-line is not bound to any keys
arrow-key-prefix is not bound to any keys
backward-byte is not bound to any keys
backward-char can be found on "\C-b", "\eOD", "\e[D".
backward-delete-char can be found on "\C-h", "\C-?".
backward-kill-line can be found on "\C-x\C-?".
backward-kill-word can be found on "\e\C-h", "\e\C-?".
backward-word can be found on "\e\e[D", "\e[1;3D", "\e[1;5D", "\e[5D", "\eb".
beginning-of-history can be found on "\e<".
beginning-of-line can be found on "\C-a", "\eOH", "\e[1~", "\e[H".
bracketed-paste-begin can be found on "\e[200~".
call-last-kbd-macro can be found on "\C-xe".
capitalize-word can be found on "\ec".
```

В целом по своему функционалу все сочетания горячих клавиш можно разделить на следующие группы:

- Запуск терминала
- Управление окнами и вкладками
- Управление отображением
- Управление курсором
- Удаление текста и исправление опечаток
- Работа с буфером обмена
- Операции форматирования
- Функция автодополнения
- Управление историей введенных команд
- Контроль над процессами (работающими программами)

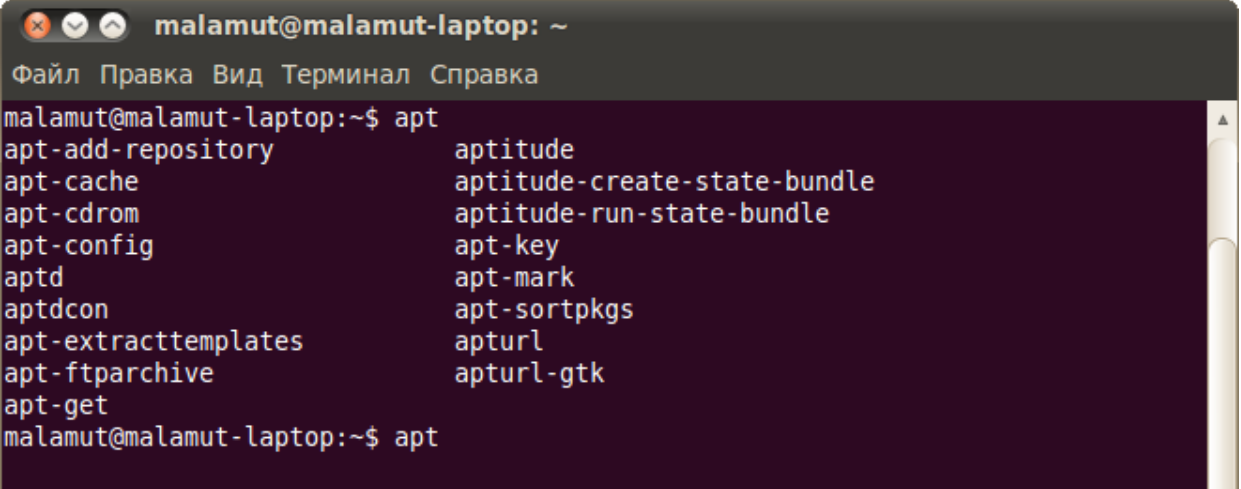
Изучим некоторые из них.

Привычные в Windows сочетания клавиш **Ctrl+C** и **Ctrl+V** в терминале работают по-другому. Они могут остановить работу терминала. Для копирования и вставки добрая пара **Ctrl+Insert** с **Shift+Insert** или же сочетания с **Shift**: **Ctrl+Shift+C** и **Ctrl+Shift+V** соответственно.

Однако профессионалы не копируют команды, а набирают их вручную. Для ускорения на помощь приходит великолепное свойство терминала – *автодополнение*.

Автодополнение. Наберите в терминале первые символы нужной вам команды, а затем клавишу **Tab**. Терминал автоматически дополнит за вас команду. Если клавишу **Tab** нажать два раза подряд, терминал представит список похожие команды на выбор.

Автодополнение в терминале работает практически везде, и не только для команд, но также для их аргументов и имён файлов. Используйте ее постоянно.



```
malamut@malamut-laptop: ~  
Файл Правка Вид Терминал Справка  
malamut@malamut-laptop:~$ apt  
apt-add-repository      aptitude  
apt-cache               aptitude-create-state-bundle  
apt-cdrom               aptitude-run-state-bundle  
apt-config              apt-key  
aptd                    apt-mark  
aptdcon                 apt-sortpkgs  
apt-extracttemplates    apturl  
apt-ftparchive           apturl-gtk  
apt-get  
malamut@malamut-laptop:~$ apt
```

Если вы хотите прервать работу уже запущенной в терминале программы, для этого можно использовать сочетание горячих клавиш **Ctrl+C**.

Другое управляющие сочетания, например **Ctrl+D** (окончание ввода данных) посылает сигнал конца файла запущенному приложению, а без запущенных утилит делает тоже, что и терминальная команда `exit`. Таким образом вы можете быстро закрыть терминал.

Также дополнительную информацию о горячих клавишах можно легко получить в Интернете.

История введенных команд. Терминал хранит историю введенных пользователем команд, которую вы можете листать в реальном режиме стрелками **↑** «вверх» и **↓** «вниз» на клавиатуре. Это очень удобно для повторного исполнения введенных ранее команд. А посмотреть всю историю можно командой

```
history
```

У каждой команды в истории есть номер, выполнить снова команду с определенным номером можно, набрав в терминале восклицательный знак **«!»** и номер нужной команды. А повторить предыдущую набранную команду можно просто написав два восклицательных знака **«!!»**.

```

malamut@malamut-laptop: ~
Файл Правка Вид Терминал Справка
malamut@malamut-laptop:~$ history
 24 clear
 25 history
 26 ls -l
 27 history
malamut@malamut-laptop:~$ !26
ls -l
итого 36
-rw-r--r-- 1 malamut malamut 179 2010-05-06 01:29 examples.desktop
drwxr-xr-x 2 malamut malamut 4096 2010-05-06 01:38 Видео
drwxr-xr-x 2 malamut malamut 4096 2010-05-06 01:38 Документы
drwxr-xr-x 2 malamut malamut 4096 2010-05-06 12:28 Загрузки
drwxr-xr-x 2 malamut malamut 4096 2010-05-06 12:49 Картинки
drwxr-xr-x 2 malamut malamut 4096 2010-05-06 01:38 Музыка
drwxr-xr-x 2 malamut malamut 4096 2010-05-06 01:38 Öffentlich
drwxr-xr-x 5 malamut malamut 4096 2010-05-07 13:12 Рабочий стол
drwxr-xr-x 2 malamut malamut 4096 2010-05-06 01:38 Шаблоны
malamut@malamut-laptop:~$

```

Навигация по системе

Сперва научимся двигаться по каталогам или директориям, (папок в UNIX-е нет), создавать и переименовывать каталоги. Linux всегда создает домашний каталог каждому пользователю. Кроме того, в Linux имеется корневой каталог Root, где хранятся системные файлы и каталоги. С каталогом Root мы ничего делать не будем, это не безопасно для работы операционной системы. Для выдачи команд используем консольное окно (терминал).

Для навигации в файловой системе Linux используются следующие команды:

- pwd** – выводит название текущего рабочего каталога;
- cd** – выполняет переход в другой каталог;
- ls** – выводит список содержимого каталога.

В любой момент времени работы в терминале вы находитесь в некотором каталоге. При запуске терминала текущей директорией является домашний каталог пользователя (/home).

```

stank@DESKTOP-61Q8VTL:~$ pwd
/home/stank
stank@DESKTOP-61Q8VTL:~$

```

Команда **pwd** отображает текущий рабочий каталог и обозначает: *print working directory* – вывести рабочий каталог.

\$ pwd

Поскольку мы находимся в домашнем каталоге пользователь, создадим рабочую директорию с именем `work`:

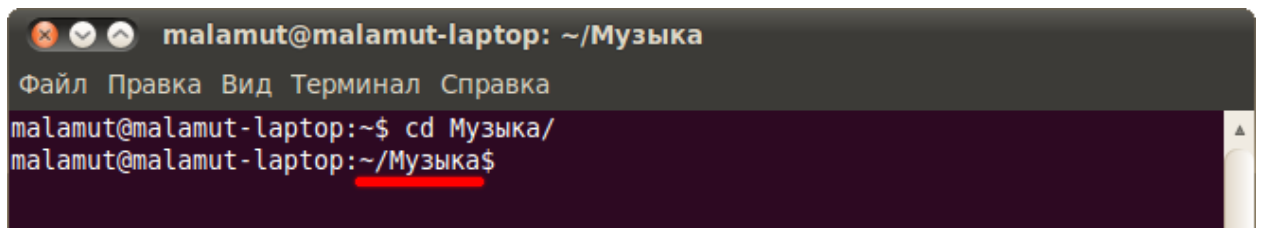
\$ mkdir ИмяКаталога

Команда `mkdir` обозначает — *make a directory*. Аргумент команд задает имя каталога. В нашем случае это будет `work`.

Теперь надо перейти в этот каталог. Для этого воспользуемся командой `cd` (*change directory*):

```
$ cd work    # например
```

Также можно перемещаться по всем директориям.



После команды `cd` можно указывать как полные пути относительно корня (*абсолютный путь*), так и *относительные* от текущего каталога.

Обратим внимание на несколько важных особенностей.

Во-первых, при наборе путей так же работает **автодополнение** по `Tab`, это очень удобно.

Во-вторых, использовать различные *небуквенные символы и пробелы* напрямую при наборе путей нельзя. Например, для того, чтобы перейти в каталог, содержащий в имени символ пробела, надо при наборе пути к такому каталогу перед пробелом поставить символ обратного слеша `\`. Вот так:

```
$ cd Каталог\ с\ плохими\ символами\ в\ имени\<\>
```

Установка обратного слеша перед некоторыми символами называется **экранированием**. Кстати, при использовании автодополнения все слеша расставляются автоматически. Кроме того, можно просто заключить путь в двойные кавычки:

```
$ cd "Каталог с плохими символами в имени<>"
```

Но в этом случае автодополнение работать не будет.

Заменитель адреса домашнего каталога `~` можно использовать и при наборе путей, например:

```
$ cd ~/Музыка
```

Имена файлов и каталогов в Linux чувствительны к регистру символов, то есть Музыка и музыка - это два совершенно разных имени.

А для перемещения непосредственно в домашний каталог достаточно просто набрать `cd` без аргументов.

Для перемещения на каталог выше можно использовать команду

```
$ cd ..
```

две точки обозначают родительский каталог, поэтому можно делать так:

```
$ cd ../..
```

Переместиться в предыдущий посещённый каталог можно командой

```
$ cd -
```

Содержимое текущего каталога можно просмотреть с помощью команды

```
$ ls
```

Это самая часто
используемая команда!



Показать все файлы, даже скрытые, можно с помощью флагов команды

```
$ ls -a
```

Показать файлы с подробным их описанием (дата создания, владелец, права доступа) можно по команде

```
$ ls -l
```

 (это буква эль-малая)

Команда **which command** – указывает директорию, в которой находится команда (**which** – который).

Упражнение 2

Получение справки

Независимо от того, являетесь ли вы «ветераном консоли» или неопытным пользователем терминала, иногда, а скорее очень часто, вы не будете знать, какие команды и как их правильно вводить в терминал. Постоянно нужно будет обращаться к справочной документации. В терминал встроено немало инструментов, которые помогут вам в этом.

Команда `man` (сокращение от “manual” -- руководство, справочник) это традиционная форма интерактивной документации в операционных системах Unix и Linux. Будучи представленными в виде файлов в специальном формате, “man pages” написаны для огромного числа команд и распространяются вместе с программным обеспечением. Справочник `man` – это система справки о командах для терминала.

Синтаксис команды:

```
man команда
```

Например:

```
$ man ls
```

Появится собственно текст справки, разбитый на разделы. Перемещаться по нему можно с помощью стрелок и клавиш `PgUp` и `PgDown`, а для выхода просто нажмите `Q`.

Справочник `man` разбит на разделы страниц руководства. Например, Вы можете увидеть ссылку на `man (1)`. Эта цифра означает, что документация по команде “`man`” находится в 1-м разделе (команды пользователя). Вы можете указать, что вам нужна страница по “`man`” именно из первого раздела с помощью команды `man 1 man`. Указание раздела полезно в том случае, если существует несколько пунктов с одинаковым именем.

Описание самой страницы руководства можно посмотреть с помощью аргументов команды:

```
stank@DESKTOP-61Q8VTL:~$ man 1 intro
stank@DESKTOP-61Q8VTL:~$ man 2 intro
stank@DESKTOP-61Q8VTL:~$ man 3 intro
```

Получение справки на русском языке:

```
root@DESKTOP-61Q8VTL:~# man -L ru man
```

```
root@DESKTOP-61Q8VTL:~# man -L ru 5 passwd
```

В дополнение к `man` также ещё существуют команды `whatis` (1) и `apropos` (1), целью которых является упрощение поиска информации в системе `man`.

Команда `whatis` даёт очень кратное описание системных команд, что-то в духе карманного справочника по командам. Например:

```
%whatis whatis whatis (1) - search the whatis database for complete words
```

Команда `apropos` используется для поиска страницы, содержащей указанное ключевое слово. Например:

```
% apropos wav
cdda2wav (1) - a sampling utility that dumps CD audio data into wav sound files
netwave_cs (4) - Xircom Creditcard Netwave device driver
oggdec (1) - simple decoder, Ogg Vorbis file to PCM audio file (WAV or RAW)
wavelan (4) - AT&T GIS WaveLAN ISA device driver
wavelan_cs (4) - AT&T GIS WaveLAN PCMCIA device driver
wvlan_cs (4) - Lucent WaveLAN/IEEE 802.11 device driver
```

Кроме `man`-страниц у многих утилит есть **встроенная справка**, которую обычно можно посмотреть, запустив программу с ключом `--help`:

```
утилита --help
```

Например:

```
$ ls --help
```

Есть и другие способы получения помощи, например похожая на `man` утилита `info`. Но чаще всего наиболее полную информацию о программе можно получить именно из `man`-страниц, а краткую справку, указав ключ `--help` при вызове.

Упражнение 3

Файлы, каталоги и регулярные выражения

Работа с файлами и каталогами

Linux поддерживает концепцию «*Все есть файл*». Это означает, что любая сущность в операционной системе представлена в виде файла.

Для исследования системы имеются следующие основные команды:

file –определяет тип файла;
less –выводит содержимое файла.

Для *управления файлами и каталогами* пять, наиболее часто используемых команд в Linux:

cp – копирует файлы и каталоги;
mv – перемещает/переименовывает файлы и каталоги;
mkdir – создает каталоги;
rm – удаляет файлы и каталоги;
ln – создает жесткие и символические ссылки.

Можно создать текстовый файл в текущей директории. Для создания файла (пустого) используем команду **touch** (*трогать*):

```
$ touch имя_файла
```

Чтобы занести в файл текстовые строки (в режиме добавления) воспользуемся командой:

```
$ echo "hie, once again" >> file1  
$ echo "hie, once again" > file1 – перезаписывает содержимое file1.
```

Содержимое файла можно выдать с помощью команды:

```
$ cat file1
```

Наконец перенести файл в другое место можно по команде:

```
$ mv имя1 имя2
```

Копирование файла осуществляется командой:

```
$ cp имя1 имя2
```

Если нужно скопировать файл в другой директорий, то имя2 в команде копирования задает имя директория.

Можно записать несколько строк в текстовый файл таким образом:

```
$ cat> file1  
Hello students,
```

What do you want?

В конце ввода следует нажать комбинацию **ctrl-d**.

Очистить содержимое файла проще всего

```
$ > file1
```

Для удаления файла используем

```
$ rm имя
```

Более полный список команд и способов их применения смотрите в разделе «Дополнительная информация» под названием «Шпаргалка команд».

Практическое применение регулярных выражений

а) Создадим «песочницу» с множеством файлов и каталогов

```
[me@linuxbox ~]$ mkdir -p playground/dir-{00{1..9},0{10..99},100}
[me@linuxbox ~]$ touch playground/dir-{00{1..9},0{10..99},100}/file-{A..Z}
```



Какая мощь командной строки! Эти две строки создают каталог **playground**, содержащий 100 подкаталогов и 26 пустых файлов в каждом за несколько секунд.

Попробуйте-ка то же самое сделать в графическом интерфейсе!

Однако, если вам по какой-то причине понадобится удалить эти каталоги, можно применить команду **rm -r**. Флаг «r» необходим для *рекурсивного* обхода каталогов и удаления их вместе с содержимыми файлами. При использовании данной команды **будьте осторожны**, вся информация удаляется, минуя корзину. Теоретически информацию можно восстановить по горячему, но это уже высший пилотаж.

б) Проверка списка телефонов

На практике же часто приходится проверять списки телефонов, поэтому давайте создадим такой список. Для этого воспользуемся волшебной магией командной строки. Магией, потому что вы еще не знакомы с большинством команд, привлеченных для решения поставленной задачи.

Вот это волшебство:

```
$ for i in {1..10}; do echo "(${RANDOM:0:3}) ${RANDOM:0:3}-${RANDOM:0:4}" >> phonelist.txt; done
```

Эта команда создаст файл с именем **phonelist.txt**, содержащий 10 телефонных номеров. Если повторить команду, она добавит в список еще 10 номеров.

Однако если заглянуть в файл, можно заметить проблему:

```
$ cat phonelist.txt
```

Некоторые номера оформлены неправильно. Просканируем файл в поисках недопустимых номеров и вывести их с помощью `grep`:

```
$ grep -Ev '^([0-9]{3}) [0-9]{3}-[0-9]{4}$' phonelist.txt
```

Здесь мы использовали параметр `-v`, чтобы обратить сопоставление и вывести только строки, не соответствующие указанному выражению.

Само выражение содержит якорные метасимволы на обоих концах и тем самым гарантирует отсутствие дополнительных символов слева и справа от номера. Кроме того, в отличие от примера, приведенного выше, это выражение также требует обязательного наличия круглых скобок в номере.

с) Поиск необычных имен файлов с помощью `find`

Команда `find` использует регулярное выражение для поиска путей к файлам, содержащим любые символы, не входящие в следующее множество:

```
[-_./0-9a-zA-Z]
```

В результате такого поиска можно выявить имена файлов и каталогов, содержащие пробелы и другие, потенциально вредные символы:

```
$ find . -regex '.*[^\_./0-9a-zA-Z].*'
```

Из-за требования точного совпадения всего пути мы добавили элемент `*` с обоих концов выражения, замещающий любое количество любых символов (в том числе и отсутствие символов). В середине выражения находится инвертированное выражение в квадратных скобках, содержащее множество символов, допустимых в именах файлов и каталогов.

Поиск файлов с помощью `locate`:

```
$ locate --regex ,bin/(bz|gz|zip)'
```

Используя чередование, мы нашли пути, содержащие `bin/bz`, `bin/gz` или `/bin/zip`.



Полученные знания и умения будут
в дальнейшем будут нещадно про-
веряться на протяжении всего
курса, пока у вас не сформируются
стойкий навыки!

We hope you enjoy working with Linux!



ЗАДАНИЯ

Задание 1

Используя терминал

1. Зайдите в корневую директорию `root` и получите все доступные каталоги. Выведите все файлы и директории в данном каталоге (`root`).
2. Получите данные о вашей системе. Найдите **исполняемый** файл **ядра Linux**. В какой директории он находится?
3. Вернитесь в домашний каталог пользователя (`home`). Выведите сообщение «I'm like Linux!».
4. Получите историю введенных команд.
5. Создайте директорию на рабочем столе. Внутри этой директории создайте 3 текстовых файла одним действием.
6. Удалите один из созданных файлов с помощью мыши (в графическом интерфейсе), а другой файл с помощью консольной команды. Далее с помощью консольной команды попробуйте найти удаленные файлы. Посмотрите атрибуты найденного файла. Объясните в чем разница этих способов удаления файлов.
7. В случае если при удалении файла с помощью консольной команды файл пропал безвозвратно, попробуйте удалить подобный файл с помощью команды (утилиты) `trash-cli`, при необходимости утилиту скачайте и установите. Объясните, как работает утилита.
8. Записать текст «I'm like Linux!» в оставшийся файл.
9. Допишите в этот файл историю команд.
10. Вывести содержимое файла на консоль.
11. Откройте содержимое файла с помощью графического редактора (например, `gedit`, `nano`).

Задание 2

1. Получите справку о справке. Укажите все разделы руководства.
2. Получите справку о *первом* и *пятом* разделе справочника.
3. Получите краткую справку о любой команде, ранее использованной вами.
4. Получите список страниц руководства, в которых содержится ключевое слово команды получения данных о вашей системе.
5. Получите справки о команде **`passwd`** и конфигурационном файле **`passwd`**. Найдите их месторасположение в директориях. Объясните в чем разница.

Задание 3

Для выполнения всего курса лабораторных работ вам необходимо правильно организовать работу. При этом используем всю мощь командной строки. В пользовательской директории `home` создайте каталоги для выполнения и хранения лабораторных работ. Курс можно назвать `LinuxLabs`, он состоит из двух семестров `Sem1` и `Sem2`. В каждом семестре примерно 5 тем (`Lab1 ... Lab5`). В каждой лабораторной примерно три задания (например, `Task31`, `Task32`, `Task33`). Это все каталоги, а в каждом каталоге должен быть текстовый файл, например, `file31`. Дерево каталогов может выглядеть примерно так, `LinuxLabsSem1/LinuxLab3/Task31/file31`. Образец необходимых регулярных выражений представлен в «песочнице». Напишите скрипт, возможно в дальнейшем он вам пригодится.

Пока не сделаете это задание, к выполнению других заданий можете не приступать!

Вообще не приступать!



Также проделайте упражнения с регулярными выражениями, и их результаты подтвердите скриншотами. Объясните полученный результат.

*«Easy things should be easy and hard things should be possible»
«Простые вещи должны быть простыми, а сложные вещи должны быть
возможными»*



Контрольные вопросы:

Управление терминалом

1. В каких технологиях программирования используется умение работать с терминалом UNIX?
2. Что такое GUI и CLI и в чем их отличие?
3. Что такое *терминал*, и чем он отличается от *консоли*?
4. Укажите преимущества терминала перед оконным интерфейсом, и наоборот?
5. Что такое командный интерпретатор?
6. Какой командный интерпретатор используется в вашей экосистеме?
7. В какой директории находится командный интерпретатор вашей экосистемы?
8. Что такое *интерфейс командной строки*?
9. Как запустить терминал с помощью горячих клавиш?
10. Как запустить консоль и затем вернуться в терминал (GUI)?
11. Что такое *команды терминала* и в какой директории системы они находятся?
12. Что такое свойство терминала – **автодополнение**, и как оно вызывается (горячие клавиши)?
13. Чем отличаются понятия: «папка» от «директория (каталог)», и есть ли в терминале «папки»?
14. Как запустить терминал с помощью мыши из текущей папки?
15. Объясните **приглашение командной строки** (*prompt*), расшифруйте его специальные символы.

Навигация по файловой системе

16. Что такое *абсолютный* и *относительный путь* к файлу?
17. Что такое *корневой каталог*, как он называется и обозначается в файловой системе?
18. Что такое *домашний каталог пользователя*?
19. Укажите специальные символы *текущей* и *родительской директории*, и как можно сменить текущую директорию?
20. Почему в именах файлов нельзя использовать *небуквенные символы* и *пробелы*?
21. Что такое **экранирование** символов и имен файлов, и как это делается?

22. Что такое *история введенных команд*?
23. Какими средствами UNIX перехватываются и интерпретируются сочетания горячих клавиш?
24. В чем заключается мощь командной строки?
25. Что такое регулярные выражения? Опишите интерфейс регулярных выражений.
26. Какие типы регулярных выражений вы знаете?

Получение справки

27. Что такое справочник `man`? Как вызвать справку о справке?
28. Назовите основные разделы справочника `man`.
29. Как проводится навигация по справочнику `man`, укажите основные опции (флаги) навигации?
30. Что такое *встроенная справка утилиты*, как ее получить?
31. Как получить справку о сочетании горячих клавиш, перехватываемых терминалом и интерпретируемых командной оболочкой?

Дополнительная информация

Используемая литература

1. Подробно песочница представлена к книге: Шоттс У. «Командная строка Linux. Полное руководство.» — СПб.: Питер, 2017. — 480 с.: ил. — (Серия «Для профессионалов»). на страницах 225-226.
2. Робачевский А. М. Операционная система UNIX®. - СПб.: 2002. - 528 ил.

Шпаргалка команд

cd ../..	перейти в директорию двумя уровнями выше
cd	перейти в домашнюю директорию
cd ~user	перейти в домашнюю директорию пользователя user
cd -	перейти в директорию, в которой находились до перехода в текущую директорию
pwd	показать текущую директорию
mkdir dir	создать каталог dir
mkdir dir1	создать директорию с именем 'dir1'
mkdir dir1 dir2	создать две директории одновременно
mkdir -p /tmp/dir1/dir2	создать дерево директорий
rm file	удалить file
rm -r dir	удалить каталог dir
rm -f file	удалить форсированно file
rm -rf dir	удалить форсированно каталог dir
rm -f file1	удалить файл с именем 'file1'
rmdir dir1	удалить директорию с именем 'dir1'
rm -rf dir1	удалить директорию с именем 'dir1' и рекурсивно всё её содержимое
rm -rf dir1 dir2	удалить две директории и рекурсивно их содержимое
cp file1 file2	скопировать file1 в file2
cp -r dir1 dir2	скопировать dir1 в dir2; создаст каталог dir2, если он не существует
cp dir/	копировать все файлы директории dir в текущую директорию
cp -a /tmp/dir1	копировать директорию dir1 со всем содержимым в текущую директорию
cp -a dir1 dir2	копировать директорию dir1 в директорию dir2
mv dir1 new_dir	переименовать или переместить файл или директорию
mv file1 file2	переименовать или переместить file1 в file2. если file2 существующий каталог - переместить file1 в каталог file2
ln -s file1 lnk1	создать символическую ссылку на файл или директорию

ln file1 lnk1	создать «жёсткую» (физическую) ссылку на файл или директорию
touch file	создать file
touch -t 0712250000 fileditest	модифицировать дату и время создания файла, при его отсутствии, создать файл с указанными датой и временем (YYMMDDhhmm)
cat > file	направить стандартный ввод в file
more file	вывести содержимое file
head file	вывести первые 10 строк file
tail file	вывести последние 10 строк file
tail -f file	вывести содержимое file по мере роста, начинается с последних 10 строк
cat file1	вывести содержимое файла file1 на стандартное устройство вывода
tac file1	вывести содержимое файла file1 на стандартное устройство вывода в обратном порядке (последняя строка становится первой и т.д.)
more file1	постраничный вывод содержимого файла file1 на стандартное устройство вывода
less file1	постраничный вывод содержимого файла file1 на стандартное устройство вывода, но с возможностью пролистывания в обе стороны (вверх-вниз), поиска по содержимому и т.п.
head -2 file1	вывести первые две строки файла file1 на стандартное устройство вывода. По-умолчанию выводится десять строк
tail -2 file1	вывести последние две строки файла file1 на стандартное устройство вывода. По-умолчанию выводится десять строк
cat file_originale [operation: sed, grep, awk, grep и т.п.] > result.txt	общий синтаксис выполнения действий по обработке содержимого файла и вывода результата в новый
cat file_originale [operazione: sed, grep, awk, grep и т.п.] » result.txt	общий синтаксис выполнения действий по обработке содержимого файла и вывода результата в существующий файл. Если файл не существует, он будет создан
grep Aug /var/log/messages из файла '/var/log/messages'	отобрать и вывести на стандартное устройство вывода строки, содержащие «Aug»
grep ^Aug /var/log/messages из файла '/var/log/messages'	отобрать и вывести на стандартное устройство вывода строки, начинающиеся на «Aug»
grep [0-9] /var/log/messages из файла '/var/log/messages'	отобрать и вывести на стандартное устройство вывода строки, содержащие цифры
grep Aug -R /var/log/*	отобрать и вывести на стандартное устройство вывода строки, содержащие «Aug», во всех файлах, находящихся в директории /var/log и ниже
sed 's/stringa1/stringa2/g' example.txt	в файле example.txt заменить «string1» на «string2», результат вывести на стандартное устройство вывода
sed '/^\$/d' example.txt	удалить пустые строки из файла example.txt
sed '/ *#/d; /^\$/d' example.txt	удалить пустые строки и комментарии из файла example.txt
echo 'esempio' tr '[:lower:]' '[:upper:]'	преобразовать символы из нижнего регистра в верхний

sed -e '1d' result.txt	удалить первую строку из файла example.txt
sed -n '/string1/p'	отобразить только строки содержащие «string1»
sed -e 's/ *\$' example.txt /удалить пустые символы в в конце каждой строки / sed -e 's/string1g' example.txt	удалить строку «string1» из текста не изменяя всего остального
sed -n '1,8p;5q' example.txt	взять из файла с первой по восьмую строки и из них выве- сти первые пять
sed -n '5p;5q' example.txt	вывести пятую строку
sed -e 's/0*/0/g' exam- ple.txt	заменить последовательность из любого количества нулей одним нулём
cat -n file1	пронумеровать строки при выводе содержимого файла
cat example.txt awk 'NR%2==1'	при выводе содержимого файла, не выводить чётные строки файла
echo a b c awk '{print \$1}'	вывести первую колонку. Разделение, по умолчанию, по проблелу/пробелам или символу/символам табуляции
echo a b c awk '{print \$1,\$3}'	вывести первую и третью колонки. Разделение, по умолча- нию, по проблелу/пробелам или символу/символам табуля- ции
paste file1 file2	объединить содержимое file1 и file2 в виде таблицы: строка 1 из file1 = строка 1 колонка 1-n, строка 1 из file2 = строка 1 колонка n+1-m
paste -d '+' file1 file2	объединить содержимое file1 и file2 в виде таблицы с раз- делителем «+»
sort file1 file2	отсортировать содержимое двух файлов
sort file1 file2 uniq	отсортировать содержимое двух файлов, не отображая по- второв
sort file1 file2 uniq -u	отсортировать содержимое двух файлов, отображая только уникальные строки (строки, встречающиеся в обоих фай- лах, не выводятся на стандартное устройство вывода)
sort file1 file2 uniq -d	отсортировать содержимое двух файлов, отображая только повторяющиеся строки
comm -1 file1 file2	сравнить содержимое двух файлов, не отображая строки принадлежащие файлу 'file1'
comm -2 file1 file2	сравнить содержимое двух файлов, не отображая строки принадлежащие файлу 'file2'
comm -3 file1 file2	сравнить содержимое двух файлов, удаляя строки встреча- ющиеся в обоих файлах

Горячие клавиши

<https://habr.com/ru/company/lanit/blog/537596/>

<https://eternalhost.net/base/vps-vds/terminal-ubuntu-gorjachie-klavishi>

<https://www.opennet.ru/man.shtml?topic=bash&russian=0>

Как проверить, какой терминал я использую

терминал является интерфейсом оболочки и часто запускает саму оболочку. Поэтому мы можем выяснить, какой процесс является родительским процессом нашей оболочки:

```
$ ps -p $$ -o args,ppid
COMMAND          PPID
bash              1234
$ ps -p 1234 -o args
COMMAND
/usr/lib/gnome-terminal/gnome-terminal-server
```

попробуем с другим терминальным приложением `sakura`:

```
$ ps -p $$ -o args,ppid
COMMAND          PPID
/bin/bash        16950
$ ps -p 16950 -o args
COMMAND
sakura
```

если мы хотим конкретно работать с терминалом с графическим интерфейсом, мы можем запустить его `xprop`, щелкнуть по нужному окну, `grep` его `pid` и узнать, как называется этот `pid`, соответствующий процессу:

```
$ ps aux | grep $(xprop | awk -F=' ' '/PID/ {print $2}')
xieerqi  2124  0.6  1.7 208068 34604 ?        S1   18:47   1:49 gnome-terminal
```

Кроме того, согласно спецификации, оконные менеджеры должны установить `WM_CLASS` свойство. Таким образом, мы можем получить это и от `xprop`:

```
$ xprop WM_CLASS
WM_CLASS(STRING) = "sakura", "Sakura"
```

[Как проверить, какой терминал я использую? \(qastack.ru\)](http://qastack.ru)