



УРОК 2. ПЕРВЫЕ КОМАНДЫ

НАЧАЛО РАБОТЫ С ТЕРМИНАЛОМ	2
КОМАНДЫ LINUX	5
ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ	10
РАБОТА С ТЕРМИНАЛОМ	11
ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ	13
ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ	17



НАЧАЛО РАБОТЫ С ТЕРМИНАЛОМ



Терминал (командная оболочка) служит для обработки команд и общения пользователя с операционной системой.

Все вводимые команды будут обрабатываться и результат будет виден в терминале.

Терминал, оболочка для ввода команд есть в любой ОС под управлением ядра Linux. Оболочка для ввода команд присутствуют в MacOS -базовые команды идентичны и выполняют такие же функции.

Терминалы служат для общения пользователя с системой. Вводим команды в терминале в виде текста и получаем результат выполнения в виде текста как результат работы программ.

Вход на сервер [JSLinux](https://bellard.org/jslinux/) - “песочницу”, это - Линукс в браузере.

CPU	OS	User Interface	VFsync access	Startup Link	TEMU Config	Comment
x86	Alpine Linux 3.12.0	Console	Yes	click here	url	
x86	Alpine Linux 3.12.0	X Window	Yes	click here	url	Right mouse button for the m
x86	Windows 2000	Graphical	No	click here	url	Disclaimer.
x86	FreeDOS	VGA Text	No	click here	url	
riscv64	Buildroot (Linux)	Console	Yes	click here	url	
riscv64	Buildroot (Linux)	X Window	Yes	click here	url	Right mouse button for the m



Терминал для ввода команд открывается для каждой браузерной сессии. При работе в этих терминалах нельзя обновлять страницу браузера или двигаться вперед и назад командами браузера, чтобы не потерять историю введенных команд и результаты работы.

Нет необходимости знать все команды наизусть. Можно воспользоваться подсказками или ввести в терминале:

КОМАНДА `-help` - `-help`(иногда `-h`)

Самое первое что мы видим, открыв терминальную сессию это `localhost:~#`

```
> < bellard.org/jslinux/vm.html?url=alpine-x86.cfg&mem=192
Loading...
Welcome to JS/Linux (i586)

Use 'vflogin username' to connect to your account
You can create a new account at https://vfsync.org
Use 'export_file filename' to export a file to you
Imported files are written to the home directory.

localhost:~#
```

Разберем, что же это такое:



localhost - это название сервера, его внутреннее имя. Вместо него может быть IP адрес.

- `~` говорит нам о том, что мы работаем из домашней директории.
 - По умолчанию все пользователи начинают работу из домашней директории
- `#` - символ суперадминистратора или суперпользователя.
 - Это тот, кто является самым главным на сервере и может создавать и удалять все файлы и папки, запускать процессы и администрировать все ресурсы и модули, подключенные на удаленном устройстве.



- Для привилегированного пользователя вместо # будет %. Этот пользователь может поднять уровень привилегий до суперпользователя, но по умолчанию им не является.



Корневой раздел - начало файловой системы.

Он обозначается символом / и в нем находятся все папки, с которыми происходит взаимодействие в процессе работы.

Содержимое корневой директории типичного linux дистрибутива:

```
461 Jan 15 18:50 .
461 Jan 15 18:50 ..
905 Jun 24 2020 .ash_history
0 Jan 15 18:42 .fscmd
297 Jul 5 2020 .preload2
2089 Jan 9 2021 bin
2240 Jan 15 18:42 dev
1979 Jan 9 2021 etc
37 Jul 5 2020 home
901 Aug 17 2020 lib
102 May 29 2020 media
37 May 29 2020 mnt
58 Jan 9 2021 opt
0 Jan 15 18:42 proc
237 Jan 9 2021 root
140 Jan 15 18:42 run
2312 Nov 21 2020 sbin
37 May 29 2020 srv
0 Jan 15 18:42 sys
37 Jan 9 2021 tmp
229 Jun 24 2020 usr
348 Aug 27 2020 var
```

Все файлы, папки, картинки, музыка и даже устройства, система видит как обычные текстовые файлы.

По цветам можно определить файлы, директории, исполняемый файлы, архивы и многое другое. Файлы белого цвета, директории фиолетовые или синие, исполняемый файлы зеленые, архивы красного.

Основным рабочим инструментом в системе будет текстовый редактор.



КОМАНДЫ LINUX

Отсутствие ответа от ОС после ввода команды - стандартное поведение. Например, при создании каталога система ничего не возвращает, а просто создает нужную директорию.

Все команды отделяются друг от друга. Несколько команд можно ввести в одну строку, но это не популярная мера. Команды необходимо отделять от аргументов, ключей и прочей вводимой информации пробелами.

Команда	Полное название	Пояснение
cd	(change directory)	Переход по определенному пути, смена текущего расположения
pwd	(print working directory)	Показать текущее расположение
ls	(list)	Показать список файлов, просмотр содержимого каталога

```
localhost:~# cd
localhost:~# cd /
localhost:/# pwd
/
localhost:/# ls
bin    etc    lib    mnt    proc  run    srv    tmp    var
dev    home  media  opt    root  sbin   sys    usr
```



ls - выдача информации о файлах или каталогах. Флаги или ключи идут со знаком минус "-" и отделяются от команды и аргумента(ов) пробелами

Командой ls обрабатываются следующие флаги:

- -t : Имена файлов сортируются не по алфавиту, а по времени (сначала идут самые свежие файлы). По умолчанию используется время последнего изменения.
- -l : Вывод в длинном формате: перед именами файлов выдается режим доступа, количество ссылок на файл, имена владельца и группы, размер в

байтах и время последней модификации

Режим доступа к файлу при указании флага -l выводится в виде 10 символов.
Первый символ означает:

d	Файл является каталогом.
-	Обычный файл.

- -a: Вывести список всех файлов (обычно не выводятся файлы, имена которых начинаются с точки, то есть скрытые файлы).
- -h (human readable) вывести размер файла в "человекочитаемых" единицах - вместо тысяч килобайтов - мегабайты, а вместо тысяч гигабайт - терабайты и т.д.

Можно также «свернуть» флаги в один флаг ls -la

Пример использования ls, ls -a, ls -la :

```
localhost:/# ls
bin    etc    lib    mnt    proc   run    srv    tmp    var
dev    home  media  opt    root   sbin   sys    usr
localhost:/# ls -a
.      bin    media  run    usr
..     dev    mnt    sbin   var
.ash_history  etc    opt    srv
.fscmd        home   proc   sys
.preload2     lib    root   tmp
localhost:/# ls -la
total 68
drwxrwxrwx 21 root   root           461 Jan 16 01:28 .
drwxrwxrwx 21 root   root           461 Jan 16 01:28 ..
-rw-r----- 1 root   root           905 Jun 24 2020 .ash_history
-rw-rw-rw-  1 root   root            0 Jan 15 18:42 .fscmd
drwxr-xr-x  2 root   root          297 Jul  5 2020 .preload2
drwxr-xr-x  2 root   root         2089 Jan  9 2021 bin
drwxr-xr-x  4 root   root         2240 Jan 15 18:42 dev
drwxr-xr-x 39 root   root         1979 Jan  9 2021 etc
drwxr-xr-x  2 root   root            37 Jul  5 2020 home
drwxr-xr-x  8 root   root           901 Aug 17 2020 lib
drwxr-xr-x  5 root   root          102 May 29 2020 media
drwxr-xr-x  2 root   root            37 May 29 2020 mnt
drwxr-xr-x  3 root   root            58 Jan  9 2021 opt
dr-xr-xr-x 40 root   root            0 Jan 15 18:42 proc
drwxr-xr-x  5 root   root           237 Jan  9 2021 root
drwxr-xr-x  4 root   root          140 Jan 15 18:42 run
drwxr-xr-x  2 root   root          2312 Nov 21 2020 sbin
```



С помощью команды **cd** (change directories) мы можем перейти по определенному пути или зайти в какую-то папку.

cd - эта команда без дополнительной информации перейдет в домашний каталог. Если пользователь уже находится в домашнем каталоге, команда не выполнит никаких действий.

Чаще команда используется следующим образом: cd **/путь/к/каталогу**

В данном случае фразу **/путь/к/каталогу** нужно заменить реальным путем к каталогу, в который нужно перейти. **/путь/к/каталогу** в данном случае будет являться аргументом и отделяется от команды пробелом

```
localhost:/# cd
localhost:~# pwd
/root
localhost:~# cd /home
localhost:/home# pwd
/home
localhost:/home#
```

Команда	Пояснение
mkdir (make directory)	Создание папки -p очень важный параметр
touch	Создание файла



cp (copy)	Копирование
mv (move)	Перемещение и переименование
rm (remove)	Удаление файла/папки *Иногда может удалять целые каталоги с файлами, но для этого нужен флаг -rf

```
localhost:/home# mkdir user1
localhost:/home# mkdir -p user2/task2
localhost:/home# touch user2/task2/file
localhost:/home# cp user2/task2/file /tmp/newfile
localhost:/home# mv /tmp/newfile /opt
localhost:/home# rm /opt/newfile
localhost:/home# rm -rf user2/
localhost:/home#
```

Команда	Пояснение
whoami	Узнать, под каким именем мы работаем в системе
clear	Очистить экран
tree	Просмотр дерева каталогов
history	Вывод истории
export_file	Скачать файл из песочницы
date	Вернуть текущую дату



```
localhost:/home# whoami
root
localhost:/home# tree
.
└─ user1

1 directory, 0 files
localhost:/home# date
Tue Jan 16 01:41:40 UTC 2024
localhost:/home# history
 0 ls /
 1 ls -la /
 2 ls -la /var
 3 ls /var/log
 4 cd
 5 cd /
 6 pwd
 7 ls
```



ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Соотнесите название команды и то, что она делает:

Команда	Пояснение
1. mkdir	A. Создание файла
2. touch	B. Удаление файла/папки
3. cp	C. Перемещение
4. mv	D. Копирование
5. rm	F. Создание папки

2. Как вы поняли, что такое терминал и командная оболочка?
3. Какая логика построения команд в Linux?



РАБОТА С ТЕРМИНАЛОМ

clear , tree:

```

 8 ls -l /var
 9 ls -l /tmp
10 ls -l /home
11 ls -l /etc
12 ls
13 ls -l
14 ls
15 ls -a
16 ls -la
17 cd
18 pwd
19 cd /home
20 pwd
21 mkdir user1
22 mkdir -p user2/task2
23 touch user2/task2/file
24 cp user2/task2/file /tmp/newfile
25 mv /tmp/newfile /opt
26 rm /opt/newfile
27 rm -rf user2/
28 whoami
29 tree
30 date
31 history
32 clear
33 history
localhost:/home#
localhost:/home#
localhost:/home# clear

```

Выполняем clear :

```
localhost:/home#
```

Если экран забит лишней информацией, то можно сделать команду **clear** и терминал



станет чистым. **Ctrl + L** - очистить экран или команда **clear**.

Показать содержимое директории и субдиректорий можно с помощью команды **tree**.

Пример: `tree /home/ :`

```
localhost:/home# tree
.
├── user1
├── user2
│   └── task2
│       └── file
3 directories, 1 file
```

Не стоит выполнять команду **tree** в корневом каталоге, так как терминал начнет показывать абсолютно все файлы.

Горячие клавиши:

Arrows up(down) - посмотреть историю

<Ctrl a> - к началу строки

<Tab> - для автозаполнения

<Ctrl r> - поиск по истории

<Ctrl c> - завершить процесс



ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

- Создадим несколько папок и файлов.
- В конечном итоге мы хотим получить следующую структуру файлов и папок :
 - `/home/user1/task1/file1.txt`
 - `/home/user2/task2/file2.txt`
- В папке `home` должно получиться несколько папок (одна в одной).
- `txt` - необязательное расширение файла для `linux` , все файлы текстовые, `file2.txt` - это имя файла целиком



Упрощение - `mkdir -p`:

Можно создавать файлы и папки одной строкой, не заходя каждый раз в новосозданную папку и подпапки.

- `mkdir -p /home/user2/task2/`
- `touch /home/user2/task2/file2.txt`
- `mkdir -p /home/user3/task3/`

Флаг **-p** позволяет создать путь сокращение от path, если нужно создать папку и путь.

Если нужно создать одну папку то `mkdir`, при необходимости - можно создать весь путь с субдиректориями, просто добавив флаг **-p**

- **mkdir -p** используется для избежания ошибок при повторных запусках команда создания папок
- **mkdir -p** используют для скриптов, чтобы не возвращать ошибку, а убеждаться в наличии путей.

Полный, абсолютный путь linux от корня файловой системы - этот путь вы уже видели в примере выше, он начинается от корня "/" и описывает весь путь к файлу;

Относительный путь linux - это путь к файлу относительно текущей папки, такие пути часто вызывают путаницу.

Путь относительно домашней папки текущего пользователя - путь в файловой системе, только не от корня, а от папки текущего пользователя.

Если имя объекта начинается на « / » — это полный путь, в любом другом случае — относительный.

Закрепление материала по созданию папок и файлов:

- Создадим еще несколько папок и файлов, используя относительный путь `user3/task3` при помощи **mkdir -p user3/task3**



- Копируем `/home/user2/task2/file2.txt` с переименованием как `user3/task3/file3.txt` при помощи **cp** `/home/user2/task2/file2.txt` **user3/task3/file3.txt**
- Создадим папку `/home/user4/task4/` используя полный путь при помощи **mkdir** **-p** `/home/user4/task4`
- В ней создадим новый файл `file4.txt` при помощи **touch** `/home/user4/task4/file4.txt`
- Проверяем: `ls -a /`
`ls /home/`
`ls /home/user1/task1/`
`ls /home/user2/task2/`
`ls /home/user3/task3/`
`ls /home/user4/task4/`
- Скопировать `file2.txt` в папку `/tmp` при помощи **cp** `/home/user2/task2/file2.txt` **/tmp**
- Скопировать `file3.txt` с переименованием `/tmp/newfile3.txt` при помощи **cp** `/home/user3/task3/file3.txt` `/tmp/newfile3.txt`
- Скопировать папку `/home/user2` в `/tmp` с переименованием `mydirectory` при помощи **cp -r** `/home/user2` `/tmp/mydirectory`
- Проверяем

Перемещение и переименование происходит при помощи команды `mv`, используя 2 аргумента: что переносим и куда переносим. Переносить и копировать можно с переименованием:

старое название → новое название

При переименовании файл переносится в ту же локацию, только с другим именем.

mv /opt/newfile /opt/newfile.txt

mv /opt/newfile.txt /tmp/new_text



```
rm /tmp/file2.txt
```

```
ls /tmp
```

Удаление файлов и директорий:

- Удаление пустых папок командой `rmdir`
- Удаление файлов и папок командой `rm`
- Удаление иерархии с вложениями `rm -r`
- Удаление файлов без подтверждения удаления каждого `rm -rf`

Посмотреть историю можно при помощи команды **history**. Вывод истории в файл скачать ее из браузера себе на свой компьютер:

```
history > /tmp/history.txt && export_file /tmp/history.txt
```




ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Как вы поняли разницу между полным и относительным путем?
2. Для чего нужен ключ -r в mkdir?
3. Какой ключ мы используем для копирования папки целиком?