The Linux Terminal
https://www.digitalocean.com/community/tutorials/an-introduction-to-the-linux-terminal

Terminal

allows using a terminal in a GUI environment

Shell

a command line interface that interprets a user's commands and script files

we use the Bourne-Again shell (bash)

Command prompt

```
user_name@host_name:current_directory$
prof@latech:~$
```

we run commands at the prompt

Root

the superuser account

Process

an instance of a running command

a process must finish before we can run another one

Frequently used commands

```
ls
cd
man
```

note that everything in Linux is case sensitive!

Arguments

parameters that can affect the behavior of a command

```
ls /etc
cd /
```

Options

flags or switches that modify the behavior of a command

```
ls -lh /etc
```

Environment variables

named values that are used to change how commands and processes are executed

```
env
echo $HOME
echo $PATH
x=5
y="Hello World"
echo $x
echo $y
```

new variables are created; existing variables are overwritten

we can export variables so that child processes can use them (e.g., in scripts)

```
export PATH=$PATH:~
```

Useful keys

`tab` (and `tab tab`)

`!` and `!!` (more later)

`.` and `..`

`Shift+PgUp` and `Shift+PgDn`

Basic Linux Navigation
https://www.digitalocean.com/community/tutorials/basic-linux-navigation-and-file-management

Where am I?

```
pwd
```

How do I go to the root directory?

```
cd /
```

How do I go to my home directory?

```
cd ~
```

How do I list files in the current directory?

```
ls
```

What about a more detailed listing?

```
ls -l
```

But the file sizes are not easy to read...

```
ls -lh
```

Can I see hidden files?

```
ls -alh
```

Where is my history (i.e., when I type history)?

```
~/.bash_history
```

When I start up a terminal, is there a script that auto runs?

```
~/.bashrc
```

`rc` means run commands

How do I "see" a file?

```
cat ~/.bashrc
more ~/.bashrc
head -n 20 ~/.bashrc
tail -n 20 ~/.bashrc
vim ~/.bashrc
nano ~/.bashrc
```

How do I create directories?

```
cd ~
ls -lh
mkdir tmp
ls -lh
mkdir "tmp/another directory"
ls -lh tmp
mkdir -p "tmp/yet another directory/and another one inside"
ls -lh tmp
rm tmp
```

How do I copy files?

```
cd ~
touch 1 2 3 4 5
mkdir tmp
cp 1 tmp
ls -lh
```

```
ls -lh tmp
cp 2 3 4 tmp
ls -lh
ls -lh tmp
rm 1 2 3 4 5 tmp
```

What about copying recursively?

```
cd ~
mkdir -p tmp/a tmp/b
touch tmp/file1 tmp/a/file2 tmp/b/file3
ls -lh tmp/*
cp -r tmp tmp2
ls -lh tmp2/*
rm tmp tmp2
```

How do I move files?

```
cd ~
touch 1 2 3 4 5
ls -lh
mkdir tmp
mv 1 tmp
ls -lh
ls -lh tmp
mv 2 3 4 tmp
ls -lh
ls -lh tmp
rm 5 tmp
```

A few final useful things

to clear the screen: `clear`

but Ctrl+L also works (without clearing the command line!)

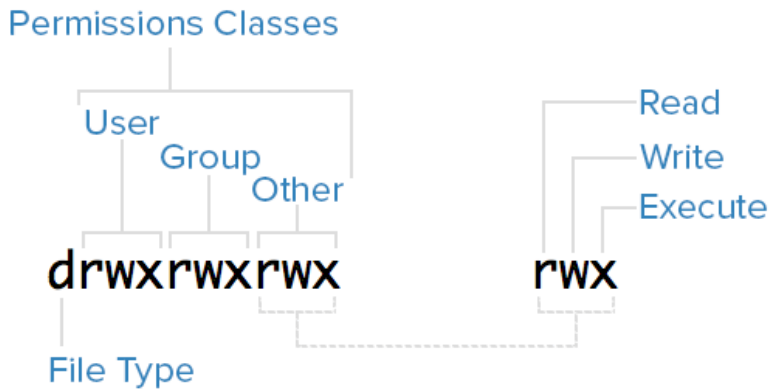
to clear the command line before the cursor: Ctrl+U (also copies what it cleared to the clipboard)

Linux File Permissions
https://www.digitalocean.com/community/tutorials/an-introduction-to-linux-permissions

```
cd ~
ls -alh
```

mode		owner	group	size	last_mod	filename
drwxr-xr-x	20	prof	prof	4.0K	Sep 13 14:02	.
drwxr-xr-x	4	root	root	4.0K	Sep 12 08:45	..
-rw-------	1	prof	prof	2.6K	Sep 13 14:20	.bash_history
-rw-r--r--	1	prof	prof	220	Sep 12 08:45	.bash_logout
drwx------	11	prof	prof	4.0K	Sep 13 14:12	.cache
drwxr-xr-x	15	prof	prof	4.0K	Sep 12 09:04	.config
drwx------	3	root	root	4.0K	Sep 13 14:02	.dbus
drwxr-xr-x	2	prof	prof	4.0K	Sep 12 08:47	Desktop
-rw-------	1	prof	prof	25	Sep 13 14:01	.dmrc
drwxr-xr-x	2	prof	prof	4.0K	Sep 12 08:47	Documents
drwxr-xr-x	2	prof	prof	4.0K	Sep 12 09:06	Downloads

Mode/permissions



File types

`-`: a normal file
`d`: a directory
`l`: a link

Permission classes

user: the owner of a file belongs to this class
group: the members of a file's group belong to this class
other: any users that are not part of the user or group classes belong to this class

Permissions

`r`: read
`w`: write
`x`: execute (must be enabled for directory entry)
`-`: permission is not available

Read

can view the contents of a file
can view the names of files in a directory

Write

can modify a file and delete it
can delete a directory, modify its contents (create, delete, and rename files in it)

can modify the contents of files in a directory (that have the read permission)

Execute

can execute a file (must also have the read permission)
can access (traverse into) a directory
can access metadata about files in the directory

Common modes

```
-rw-------
-rw-r--r--
-rwxr-xr-x
drwxr-xr-x
drwx------
```

Modifying permissions

`chmod`

```
cd ~
touch test
ls -l test
chmod u+x-w test
ls -l test
chmod g-r+wx,o-r+w test
ls -l test
chmod 777 test
ls -l test
chmod 644 test
ls -l test
rm test
```

Recursive: `chmod -R`

Numeric modes (binary!)

0: `---`
1: `--x`
2: `-w-`
3: `-wx`
4: `r--`
5: `r-x`
6: `rw-`
7: `rwx`

Changing ownership

`chown`

```
chown prof:prof file
```

Linux I/O Redirection

https://www.digitalocean.com/community/tutorials/an-introduction-to-linux-i-o-redirection

Streams

standard input: `stdin` (also numbered 0)

usually means the user's keyboard

standard output: `stdout` (also numbered 1)

usually means the user's monitor

standard error: `stderr` (also numbered 2)

Stdin

```
cat
```

type:

```
1
2
3
Ctrl+D
```

Stdout

```
echo "Sent to the terminal through stdout"
```

Redirection

```
>: redirect stdout
<: redirect stdin
2>: redirect stderr
```

Can also append

```
>>: redirect stdout
<<: redirect stdin
2>>: redirect stderr
```

e.g.,

```
cd ~
cat > file.txt
```

type stuff, then Ctrl+D

```
cat file.txt
cat > file.txt
```

type different stuff, then Ctrl+D

```
cat file.txt
cat >> file.txt
```

type different stuff, then Ctrl+D

```
cat file.txt

rm file.txt
```

Pipes

used to redirect one stream to another; e.g.,

```
ls -lh /usr/bin
ls -lh /usr/bin | more
```

/dev/null

a special file that is used to trash anything that's redirected to it

```
ls -lh /usr/bin > /dev/null
```

Redirecting stderr

```
cd ~
mkdir '' (this is two single quotes, side by side)
mkdir '' 2>> errors.txt
find / -maxdepth 2 2>> errors.txt
cat errors.txt
rm errors.txt
find / -maxdepth 2 > files.txt 2>> errors.txt
cat files.txt
cat errors.txt
rm files.txt errors.txt
```

Filters

grep (print lines matching a pattern)

```
find /usr/bin
find /usr/bin | grep wc
```

wc (print newline, word, and byte counts)

```
find /usr/bin | wc
find /usr/bin | wc -l
```

tee (read from stdin and write to stdout/files)

```
cd ~
wc /etc/magic
wc /etc/magic | tee magic_file.txt
cat magic_file.txt
rm magic_file.txt
```

tr (translate or delete characters)

```
tail -n +10 ~/.bashrc | tr a A | tr -d s
```

Multiple pipes

```
find /etc | grep conf
find /etc | grep conf | tr l 1 | tr e 3 | tr a 4 | tr s 5 | tr b 8 | tr o 0
```

Regular Expressions

https://www.digitalocean.com/community/tutorials/an-introduction-to-regular-expressions

All about pattern matching

usually because we want to find things in a lot of things

Regular expression == regex

Consists of **literal** characters and **meta** characters

meta == power

e.g., with a large dictionary file

```
cat dictionary.txt | wc -l
tail -n 20 dictionary.txt
```

Anchor meta characters

`^`: matches the **start** of a pattern

`$`: matches the **end** of a pattern

```
grep -E '^a' dictionary.txt
grep -E 'a$' dictionary.txt
```

The dot (.) meta character

matches a single character

```
grep -E '^a.....b$' dictionary.txt
```

Character groups

uses square brackets

```
grep -E '^[aeiou]......[aeiou]$' dictionary.txt
```

Groupings

uses parentheses

```
grep -E '^[a](si|fric)a$' dictionary.txt
```

Quantifiers

instead of

```
grep -E '^a.....b$' dictionary.txt
```

we can

```
grep -E '^a.{5}b$' dictionary.txt
```

we can also provide a range

```
grep -E '^a.{4,5}b$' dictionary.txt
```

the quantifier {0,1} can be shortened to ?

```
grep -E '^a.{0,1}b$' dictionary.txt
grep -E '^a.?b$' dictionary.txt
grep -E '^af?r' dictionary.txt
```

the quantifier {0,} can be shortened to *

```
grep -E '^a.{0,}b$' dictionary.txt
grep -E '^a.*b$' dictionary.txt
grep -E '^af*r' dictionary.txt
```

the quantifier {1,} can be shortened to +

```
grep -E '^a.{1,}b$' dictionary.txt
```

```
grep -E '^a.+b$' dictionary.txt
grep -E '^af+r' dictionary.txt
```

e.g.,

matching a US ZIP code

```
grep -E '^[0-9]{5}(-[0-9]{4})?$'
```

matching all valid times in a 24 hour clock

```
grep -E '^([01][0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9]'
```

Word boundaries

```
grep -E '\bpour' dictionary.txt
grep -E 'pour\b' dictionary.txt
grep -E '\bpour\b' dictionary.txt
```

Back references

```
grep -E '(au).*\1' dictionary.txt
grep -E '^([a-zA-Z]+)([a-zA-Z]+)\1\2\1' dictionary.txt
```

To list files in /usr/bin that begin with >=1 digit followed by >=0 lowercase letters, followed by 1 digit

```
ls /usr/bin | grep -E '^[0-9]+[a-z]*[0-9]'
```

We can use sed to replace!

```
grep tio dictionary.txt
grep tio dictionary.txt | sed 's/tio/sh/'
grep tio dictionary.txt | sed 's/tio/sh/i'
grep tio dictionary.txt | sed 's/tio/sh/g'
grep tio dictionary.txt | sed 's/tio/sh/ig'
```

Working with Bash History

https://www.digitalocean.com/community/tutorials/how-to-use-bash-history-commands-and-expansions-on-a-linux-vps

Also see `man history`

History defaults can be set in ~/.bashrc

`HISTSIZE=5000`

load the last 5,000 lines in memory

`HISTFILESIZE=10000`

save the last 10,000 lines to disk

`HISTCONTROL=ignoredups:ignorespace`

don't store duplicate commands

don't store commands that begin with a space

By default, history is saved for the last opened terminal

this sucks if you have multiple terminals open

we can save history for all opened terminals via

```
shopt -s histappend
```

How do we see our history?

```
history
history 10
history | grep sudo
```

Executing commands from history

`!n`: execute the command associated with history #n

`!-n`: execute n commands ago; e.g.,

`!-2`: execute two commands ago (i.e., the one before the most recent)

!!: execute the last command

useful when you forget to use sudo; e.g.,

```
touch /root/file.txt
sudo !!
ls -Alh /root
sudo !!
rm /root/file.txt
sudo !!
```

`!blah`: execute the last command that **began** with **blah**

```
!grep
```

`!?blah?`: execute the last command that **contained blah**

```
!?tio?
```

`!?blah`: execute the last command that **ended** with **blah**

```
!?txt
```

made a mistake typing something?

```
cat /etc/hosst
^hosst^hosts^
```

From the command line, use the **up arrow key** to scroll backward through history

and use the **down arrow key** to scroll forward through history

Useful Linux Commands

see http://ss64.com/bash/

```
alias
apt-get
apt-cache
bc
cat
cd
chmod
chown
clear
cp
cut
curl
date
df
diff
du
echo
exit
export
fg
find
grep
head
history
ifconfig
kill
killall
ln
locate
logout
ls
man
mkdir
more
mv
netstat
passwd
ping
ps
pwd
rm
rmdir
rsync
screen
scp
sed
sort
split
ssh
su
sudo
tail
tar
tee
time
timeout
touch
top
traceroute
tr
unalias
uname
uniq
uptime
useradd
userdel
usermod
vim
wc
whereis
who
wget
```