**Notes on Linux commands, Bash, SSH**

* ~ home dir
* $ standard user
* # root user
* Linux is case sensitive

**Commands**

| **cmd** | **info** | **examples** |
| --- | --- | --- |
| dir | lists directories and files in current dir | dir / dir yourDirName |
| ls | list storage / lists files or folders in current dir | ls / ls new-dir or ls /bin / ls -a (all hidden files) / ls -d lists dir / ls -s lists file size / ls -S sorts by file size / ls -t sorts by time and date / ls -x sorts by extension size / `ls $HOME |
| pwd | print working directory - full path/absolute path |  |
| cd | change directory | cd (goes to home folder) cd /dir cd .. (up one lvl) cd ~ (goes to home dir) cd - (goes back to last folder used) |
| mkdir | create dir | mkdir newdir / mkdir newdir1 newdir2 newdir3 (creates 3 dirs) mkdir newDir && cd newDir (creates dir and cd to it) mkdir -p newDir/{subDir1,subDir2} (creates a dir and sub dirs inside dir) |
| touch | create file | touch newfile.txt |
| rm | removes/deletes dir or file | rm dir / rm file.txt / rm -r newdir (dels dir) / rm file?.txt (dels file1, file2, file3, etc) |
| rmdir | removes empty dir | rmdir dir1 / rmdir \* (all empty dir del) |
| cp | copy file/dir from 1 loc to another | cp oldfile.txt /dir2/newfile.txt (copy, rename, move to another dir) / cp ~/.bashrc bashrc (cp from home to current dir) |
| mv | renames file | mv newcopy.txt newcopy2.txt / mv \*.txt dir (moves all txt files to dir) mv dir/\* . (moves files in dir to current folder) |
| echo | prints message | echo Hello World |
| cat | prints content in file | cat newcopy2.txt / cat > outputs to new file created or replaces old text if file already exists - cat > file2.txt / ls -al / > file.txt (full filesystem and hidden files) / cat >> - appends to bottom - cat >> newfile.txt // ctrl + d - done - after typing text (cat > newfile.txt) |
| 2> | redirects error msgs | ls -l video.mpg blah.foo 2> errors.txt / see 'STDERR' / echo > file.txt 2>&1 - sends standard output and errors to file |
| head | lists 1st 10 lines in file | head file.txt / head -20 file.txt - view 20 lines |
| tail | lists last 10 lines in file | tail file.txt |

| - piping - output one cmd/prog and inputs to another

* ls -al | cat > lsout.txt (list files in current dir inside lsout.txt)
* echo Hello World | cat >> newfile.txt - adds message to bottom of newfile.txt

global regular expression print. grep searches files for keywords | grep keyword file.txt

* grep options:
  + grep keyword filename - prints all lines containing the keyword
  + grep '\!$' filename - using regular expression to match lines ending in a !
  + --color=always - highlight matched text in color
  + -c - how many lines matched
  + -n - matches and line number
  + -i - case insensitive grep -ni "keyword" file.txt
  + -v - invert match, find all lines that don't match reg exp
  + -l - only show the filenames of the files that matched
  + -L - shows names of files that don't contain matching lines
  + -r - recursive - search all files in dirs and subdirs
  + -R - ls -R folder-name lists files and folders inside folder-name
  + -o - only print matching part of the line, not whole line
  + -a - search binaries - treat binary data like its text instead of ignoring it
  + -F - don't treat the match string as a regex
  + -E - extended support of meta characs, escapes these special characters by default ([use if you want regexps like '.+' to work, otherwise you need to use '.+'](https://wizardzines.com/comics/grep/)
  + grep -A 3 keyword shows 3 lines of context after a match
  + cat auth.log | grep "keyword" | awk '{print $4}' | sort -u > file.txt prints sorted list from column 4 from the auth.log file to file.txt,
  + ([avoid egrep and fgrep](https://linuxhandbook.com/grep-egrep-fgrep/))
  + grep alternatives - ack, ag, ripgrep - better for searching code

**Extras**

| **cmd** | **info** | **examples** |
| --- | --- | --- |
| -h or --help | short desc / long desc with options of a cmd | pushd -h / mkdir --help |
| man | manual/man pages - about the command (desc, name, options) | man appname |
| more | prints content in file / spacebar to go page to page | more newfile.txt |
| less | use arrows to read content page by page, 'q' to exit | less file.txt |
| sed | stream editor - find and replace keyword | sed 's/mysql/MySQL/g file.txt > newfile.txt' / s - substitute, g - everywhere/replace globally, '>' - to save results to newfile / replace 'g' with a number and will replace that occurence of the word (sed s/mysql/MySQL/2 = 2nd occurence replaced) |
| pushd | jump from current dir to another | pushd /dir |
| popd | back to last dir used after pushd cmd |  |
| wc | counts lines (-l), words (-w), characters (-c) in file | wc - lw newcopy2.txt |
| uniq | removes dupes in file | uniq file.txt |
| history | lists all cmds typed |  |
| locate | locate file in dir with this keyword or locate app | locate keyword |
| sudo updatedb | use to quickly update db if file not found with 'locate' |  |
| apt-get | install/update/upgrade software packages | apt-get install pkgname / apt-get remove pkgname / apt-get purge pkgname - removes pkg and config files / apt-get update - update out-of-date pkgs / apt-get upgrade - upgrade out-of-date pkgs |
| git clone | clone software | git clone (github URL) |
| which | find out if app/cmd is installed | which appname |
| whois | info about domain name's owner | whois oreilly.com less |
| whatis | short info about a cmd | whatis pushd |
| whereis | locate binary file, get source, man page, location of file |  |
| whoami | displays username |  |
| wget | d/l file from a website to current dir | wget http://linuxpocketguide.com/sample.pdf |
| ps | lists processes running on terminal | ps aux |
| top | processes running the most resources |  |
| kill | used to end process | kill -15 (PID#) / kill (PID#) - PID - process ID #, listed under ps or top |
| jobs | displays list of current jobs running in bg |  |
| fg | moves a background process into the foreground | fg (job #) |
| ctrl + z | pauses process and moves to bg |  |
| passwd | changes password |  |
| df | total storage in sys | df -h |
| ln | link files together, [content in 1st file is updated in 2nd file](https://www.freecodecamp.org/news/the-linux-commands-handbook/#the-linux-ln-command) | ln -s file.txt newfile.txt |
| info | simple info page | info cmd |
| apropos | searches and lists cmds related to that cmd | apropos cmd |
| uname -a | info about kernel version and name, hostname, OS, date and time |  |
| uptime | time, # of users, load, up for # of hours |  |
| awk | prints selected line | awk '{print}' file.txt / awk '{print $1,$2,$NF}' file.txt (prints 1st, 2nd, last line) |
| tar | tar caf myfile.tar.gz myDirectory myDirectory is compressed to tar.gz folder |  |
| zip / unzip | compress or decompress |  |
| gzip | compresses file with GNU Zip, [more info](https://www.freecodecamp.org/news/the-linux-commands-handbook/#the-linux-gzip-command) | gzip -c filename > filename.gz gzip -d filename.gz or gunzip (decompress) |
| last | all logins and reboots in the system |  |
| lpr | line printer - sends file to printer, '# 5' prints 5 pgs lpr -P myprinter myfile # 5 |  |
| mail | create/send an email from [Linux mail prog](https://www.interserver.net/tips/kb/linux-mail-command-usage-examples/) | `echo "body of email" |

advanced

* [regular expressions](https://ryanstutorials.net/linuxtutorial/grep.php#reoverview)
* [aliases](https://www.freecodecamp.org/news/the-linux-commands-handbook/#the-linux-alias-command)

**vim**

Enhanced version of vi. 2 modes - insert (enter text) and command (del text, copy/paste and other ops)

| **cmd** | **task** | **example** |
| --- | --- | --- |
| gvim | open editor in new window |  |
| vim | run in existing shell window | vim or vim script.sh |
| i | switch to insert mode then type any text |  |
| ESC | switch to command mode / end command in progress |  |
| : | switch to command line mode |  |
| :wq | save and quit |  |
| :q! | quit w/o saving |  |
| :w | save / save as | :w filename |
| l or -> | move right |  |
| h or <- | move left |  |
| k or up arrow | move up |  |
| j or down arrow | move down |  |
| w | move to next word |  |
| b | move to prev word |  |
| 0 | move to beg of line |  |
| $ | move to end of line |  |
| ^f | move down one screen |  |
| ^b | move up one screen |  |
| gg | move to beg of doc |  |
| G | move to end of doc |  |
| x | del next charac |  |
| X | del prev charac |  |
| de | delete next word |  |
| db | del prev word |  |
| dd | del current line |  |
| D | del end of line |  |
| :help | get help / view manual |  |
| :snytax on | turns on syntax highlighting |  |
| vim ~/.vimrc | make certain settings permanent (type w/o colon) | syntax on |

**Shortcuts**

| **shortcut** | **what it does** |
| --- | --- |
| Tab | tab completion - autocompletes file or dir name |
| ctrl + shift + c | copies from clipboard |
| ctrl + shift + v | pastes from clipboard |
| ctrl + c | stops/cancels program running |
| ctrl + d | terms program / ex. after typing text (cat > newfile.txt) |
| ctrl + - / ctrl + | larger/smaller text |
| ctrl + u | cuts all text before cursor, ctrl + y to paste cropped text |
| ctrl + k | cuts all text after cursor, ctrl + y to paste cropped text |
| ctrl + a | moves cursor to beginning of line |
| ctrl + e | moves cursor to end of line |
| ctrl + r | and then type 1st few letters from previous cmds - searches for cmds used, press enter to run or Esc to end search |
| ctrl + left / right arrow | moves left / right 1 word |
| up / down arrow | scroll through previous commands |
| clear or ctrl + l | clears screen |
| exit | exits terminal |

**File permissions**

View files and perms with 'ls -l'. File perms are in 1st col.

* Example: drwxr-xr-x. 1st charac is the file type. d = dir or '-' for file
  + r = readable means can view file, w = writable means can edit/del file, x = executable means can execute and access file/dir.
  + rwx means full access and a dash means no access given.
* There are 3 sets - user, group, others (UGO).
  + 1st set/1st 4 chars - 'drwx' in this dir, user/owner has full rwx perms.
  + 2nd set/5-7th chars - 'r-x' - this group can read (r) and execute (x) but not write (w).
  + 3rd set/8-10th chars - 'r-x' - all other users can read (r) and execute (x) but not write (w).
  + r = 4 bits, w = 2 bits, x = 1 bit = 4+2+1 = 7 bits / rwx = 777 = full access / 774 = all perms to user and group, read perms to others / 766 = user has rwx, all others and group rw only (4+2=6)

For more info - [chmod Command in Linux](https://linuxize.com/post/chmod-command-in-linux/) and [commands with examples](https://tecadmin.net/chmod-command-in-linux-with-examples/)

* chmod - change perms/mode, root user or file owner only. | chmod 774 file.txt
* UGO (text) method instead of numeric (777)
* References: (u) user, (g) group, (o) others
* Operators: - means removes, + means adds, = means sets/assigns
* Modes: (r) read, (w) write, (x) execute, (a) all

Examples:

1. chmod u-w file.txt (removes w perm for user)
2. chmod u+x, o+x file.txt (user and others can execute to file.txt)
3. chmod ugo=rx file.txt (everyone has rx perms)
4. chmod a=rx file.txt (same as ugo, all users have rx perms)
5. chmod ugo+rwx file.txt (all perms to all users)

Tables from LinkedIn Learning course References | Read (4) | Write (2) | Execute (1) | Result --- | --- | --- | --- | --- | Users | r | w | x | 7 Groups | r | - | x | 5 Others | r | - | - | 4

| **References** | **Read** | **Write** | **Execute** | **Result** |
| --- | --- | --- | --- | --- |
| Users | + | + | + | u+rwx |
| Groups | = | - | - | g=r |
| Others | - | - | - | o-rwx |
| All |  |  |  | a+rwx |

**Bash scripting**

* Begin with #!/bin/bash (means it's executable, saved as .sh file by default). / #!/bin/bash -x - debugging (remove later) / place in script, then run file in terminal:

set -x

(what you want to debug)

+x

* add scripts to your bin dir - /home/username/bin (exe in the bin only available to you, can't run with sudo) or /usr/local/bin (avail to all users, can run sudo)
* New scripts are NOT executable. chmod 755 bashfile.sh or chmod +x bashfile.sh to make it executable.
* Commands work exactly the same on the command line as they do within a script.
* To run file bash script.sh or ./bashfile in the terminal. (./ = to run script)
* Comments -- single line # your comment -- multi-line

:'your comment in between single quotes

next line

another line'

= - to set variable, $ - call var | myvar="yo world" echo $myvar - outputs 'yo world' when your run script -i = removes case sensitivity / -n = line number

*Conditionals*

if statements

if [ statement ]

then

echo 'your message'

exit

fi

else

echo 'another message'

fi

if (( ... ))

then

echo "..."

else

echo "..."

fi

*operators*

* && = and, -a
* || = or, -o

myVar=(var)

if ["$myVar" -gt 18] && ["$age" -lt 40]

then

echo "true"

else

echo "not true"

fi

Can also use:

* ["$myVar" -gt 18 -a "$age" -lt 40]
* [ ... -o ...]
* [..] OR [..]
* [ ... || ... ]
* [ ... ] || [ ... ]

**user input**

echo "Enter your name:"

read name

echo "Enter your age:"

read age

echo "Hello" $name, "you are" $age "years old"

Read - for user input. Run script, and user inputs name, age. [Notes from](https://devqa.io/bash-script-read-user-input/)

read -p "Enter your username: " username

read -sp "Enter your password: " password

echo -e "\nYour username is $username and Password is $password"

* -p prompt a message, displays on screen
* -sp - doesn't display on screen

**Resources**

* [Bash scripting tut](https://ryanstutorials.net/bash-scripting-tutorial)

**SSH**

RSA 2048-bit encryption, which is comparative to a 617 digit long password. More secure than passwords.

SSH keys are made up of a private key and a public key. Never share private key. If key is exposed, generate a new ssh key pair.

Can choose to either lock your private key with a passphrase. creating a key pair without a passphrase is more convenient and potentially essential for certain scripts and automation tasks. But it's also less secure. Generate a separate key pair for each service or connection you want to use, adding a passphrase only for critical services.

By default you private key is stored at /Users/myname/.ssh/id\_rsa and your public key The private key - id\_rsa - should never be shared

Public keys (id\_rsa.pub) are meant to be shared or placed on a remote server. It's stored at /Users/myname/.ssh/id\_rsa.pub. The private key is stored at /Users/myname/.ssh/id\_rsa.

To generate a key - ssh-keygen

~/.ssh/id\_rsa ~/.sshid\_rsa.pub - public key goes into server "authorized\_keys" file

To connect to server - ssh name@ip.address.here or ssh name@server.domainname.com, enter the servers password to authenticate (not local machine)

To add public key and enable pw-less login ssh-copy-id user@host

**Create a config file and alias** cd .ssh ls to view files, and touch config vi config

Host dev

HostName <domain or ip>

Port 22

User <username>

IdentityFile ~/.ssh/id\_rsa

You can now SSH into the systems with this command ssh dev

SSH Tunneling Access resources on remote server or allow acccess to your local resources to someone else. (Creating a tunnel to ssh to a machine and forward data from one port to another). Need a ssh client (work) and ssh server (home)

Setup public SSH Server **Local Port Forwarding** - access remote resources that you can't access, internal remote database, RDP ssh -L local\_port:remote\_address:remote\_port username@sshserver Ex. ssh -L 8888:192.168.1.3:8080 44.11.22.33

**Remote Port Forwarding** - other ppl to have access to local resources they don't have access to, eg. local web server ssh -R remote\_port:local\_address:local\_port username@sshserver Ex. ssh -R 8888:10.0.0.3:8080 44.11.22.33 The ssh server config /etc/ssh/sshd\_config should have a property GatewayPorts yes

**Github** Go to Settings, then ssh and gpg keys, add new, add your public key (ls to view files, cat id\_rsa.pub), copy full key and paste in gh. Cloning with ssh - git clone <git@github.com link>