Quintesential Unix Shell commands

- `ls` **- list files**
    - `ls -al` - list all files with extra information
- `mv` - move file/folder
- `cp` **- copy file/folder**
    - `cp -r` - copy folder recursively
- `pwd` - gives u the current directory, like which, full path
- `rm` - delete file
- `rm -rf` - remove folder recursively and force, IE, ignore all warnings. yes it will delete the whole drive if run as sudo on /
- `mkdir` -makes an empty directory
- `touch` -makes an empty file
- `rmdir` -remove empty directory only(safety feature)
- `less` -read a file that is longer than the screen. scroll by hitting enter, space, arrows, pgdown, search with `/` use q to quit
- `su` - setuser/superuser its supposed to stand for, su bob will make your user bob, you need his password. su makes u root(if you have a root password
- `sudo` - run a command as root, became the normal way to do things in administration after a while. before it was just get a root shell with su. it will ask for a password and if you are an admin user yours will work. you have to be in the group sudoers.
- `more` - basically the same as less but slightly different in an unmemorable and barely perceivable way
- `top` - like task manager, list everything. full featured interface, can kill things, sort everything, etc
- `ps` **-list processes defaults to ones in your shell**
    - `ps aux`- lists processes from all users with more information
- `grep` **- search files for string or regular expression, print whole line**
    - `grep -v` - exclude files
    - `grep -A n -B m` - print lines n after matching line and m before matching line
- `kill` **-end process with signal 15, smooth exit**
    - `kill -s 9` - end it right now, no shutdown sequence
- `cat` -spit entire file to stout
- `curl` -send http request and spit output to stdout
- `nc` **-netcat, same as cat but uses raw tcp socket. can work on udp too**
    - `nc -l n` - liston on port n, add -u for udp
- `sed` -more advanced regular expression oriented grep with inplace editig focus
- `awk` -similar to sed, complex grep type thing regepts inplace editing etc
- `perl` -a whole language like python, partially specialized for the tasks sed and awk do, can write one liners in shell. regexps

- `chmod` **-modify permissions, uses a number code of 3 digits or letter**
    - `cmod +x file` - set file to be executable
    - `chmod 777` - let all users read write and execute. dont do it
    - `chmod 666` - all users read and write,
    - `chmod 770` owner user and group for owner user cab w r ex
- `man` -manual page, man <command> shows the page, it is the help files, it is the best reference for arguments of commands. YOU SHOULD REFERENCE THE MAN PAGE COMMANDS. it is the only source you need for these base commands u see here, and old software. it is not necessarily the best wy to learn how to use vim.
- `screen` -make a new screen. `ctrl-a (release) d` detaches/exits from it, ctrol-a c closes. tis is the way you run things in the background
- `nohup` -preceeds command/proram and prevents hangup signals from hitting it so it will run until killed or closed from internal logic. alternative to screen for backgrond process tat will persist on logout
- `md5sum` -jsut called md5 on mac/bsd jsut does an md5 checksum hash of a file. for comparison of files of any size
- `sha256sum` -same as above woth sha256 algorithm. also exists others.
- `who` -lists out the current logins/screens. shows u who is logged in(whci hsuers and where)
- `whoami` -tells u which user u are. used to check if you've successfully hacked things and became root. or in innocent shell scripts
- `lsof` -spit out data about various things going on with processes and devices and filesystem. example lsof -i:8000 gives u info about proc using port 8000
- `lsusb` -list the usb devs. good to check if it can see a device
- `lspci` -same but for pci devices
- `sort` -sorts text file line by line
- `find` -for searching the file system. most stupid way can be done like find .|grep filenameiwant. recursive list of full dir tree uis the default behavior
- `uniq` -deletes duplicate lines that appear next to eachother in text.
- `echo` -prints whatever is in its args to stdout
- `which` -gives total path to an executable in the shell path
- `strings` -spits strings out from binary file
- `hexdump` -spits out hex of a file
- `diff` -gives u the difference of 2(text) files line by line. yes this is where the term diff comes from in git repos etc
- `tar` -deals with tar archives. to untar a tar.gz tar xvzf file.tar.gz, for tar.bz2, tar xvjf
- `gzip` -compression. works on one file, takes inut from file or stdout(!) good on text, fast
- `bzip2` -slower more intense compression
- `gunzip` -un-gzipps file
- `bunzip2` -unbz2 a file
- `bzgrep` -grep a bzfile, handy, exists also bzless bzcat bzexe...
- `lsblk` -list block devices. handy to se drives that are not mounted
- `df -h` - lsits mounted drives with size ad free space in human readable format

- `du -h` - check file size. it is recursive by default so it is good to set the max view deth with -d 0. du -h -d0 file
- `lsmod` -list kernel modules(generally are drivers), whcih are code that can be hotplugged into the kernel. this is used when trubleshooting hardware and driver issues
- `modprobe` -load up a module, they ahve a path thing built in so you can tab tab to see wahts abailable
- `time` -TIMES A COMMAND in human readable down to ms
- `date` -the timestamp in a human readable format, can spit out other formats cehck man apge
- `ln` -typically invoked as `ln -s`, which creates a symbolic link
- `fsck` -checks hard drives
- `fdisk` -partition hard drves
- `mkfs` -makes the default fs, ext4 or whatever your system thinks is the default, for other fs do `mkfs.<x>` or `mkfs -t <x>` - make other kinds of fs <x>, IE format partitions
- `yes` -endless loop of 'y'... for dealign with annoying menus with the y/n? prompts using pipe
- `wipefs` -removed disk label
- `shred` - destroy files by writing random data to the location they were stored on disk(doesnt work on some filesystems) or write random data to a whole disk
- `cryptsetup` - setup luks volumes
- `cron` -service for running periodic tasks.
- `ranger` - file explorer command line tool. vim bindings, written in python. navigate filesystem in ncurses text interface
- `lfm` - shitty version of ranger seems really old
- `lf` - newer unfinished version of ranger lighter and focused on the use of external tools to open things, not in repos https://github.com/gokcehan/lf
- `head`- get top 10 lines of the file, use -n to specify numlines
- `tail`- some as above, last 10 lines as default
- `cut`- more general than the 2 above, check the manpage, cuts chars bytes lines....

editors:

- `vi` -the old version of vim. it sucks. if u have a new install and type vi this is what is usually there. it makes people hate vim. dont use it. install vim and it will clobber the path to this
- `vim` -the new version of vi, if installed will alias as vi overriding above command, for serious people only. perfect for people that hate their mouse. extensible to the point of absurdity. it is a modal editor, meaning it has modes of interaction with the file. hit escape to dissasociate from a mode, hit a letter to change to that mode. in this case the letter `i` is insert (normal edit mode), `v` is visual(select and delete copy and stuff large blocks to text). in the default mode and in visual `d` is delete, hit it twice to delete a line. visual mode `d` deletes selection. `u` is undo. the `:` char (yes use shift) lets u type in commands for user defined things and interactions with filesystem. `:w` is write. `:wq` is write and quit. `:q` is quit. `q!` is quick rtfn with no confirmation. `:r <file>` is read(a file and output it at current cursor position). `:read !<commands>` does the same for a shell command `! <cmd>` opens the shell and hides theeditor, returning when ytou exit
- `elvis` - this is another editor, a better version of vi, lighter than vim(if i remember correctly)
- `neovim` - a new and cooler vim that people who think theyre cool use. also has qt graphical neovim-qt, aparently feature-rich and more efficient cleaner codebase as it was written more recently

- `pico` -simple old editor not sure its ever used anymore.
- `nano` -a fork/copy/something of pico, newer, good for noobs, often used and well respected. commands are on the screen when using it and ctrl-X based.
- `emacs` -a complex and extensible editor, bulky for a command line utility. generally serious editor nerds that use stuff in this section use either emacs or vim, and have strong convictions about it.
- `ed` -the simplest editor from extremely long time ago, only used in extreme emergencies. the kind of editor a eunich would use.
- `gedit` - simple grpahical editor, good, basically notepad with syntax highlighting.

system things(debian based mint/ubuntu):

- `sudo` -run following command as root (admin)
- `su` -set user, defaults to root. can specify shell with -s
- `service` -control a service. service <name of it> <start, stop, restart, reload> ex: sudo service posrtgresl restart
- `hostname` -prints hostname, if given arg it will set the hostname to the arg. if u do this, should also manually change /etc/hostname and make sure /etc/hosts refects that change if necessary
- `adduser` -`adduser <newusername>` maeks a new user. many options. none are really required, even a password.
- `usermod` -mod shell and stuff of a givemn user usermod -aG common for adding group
- `passwd` -password change, `passwd <user>` does it for user when u are admin
- `dd` -writes raw data. dd if=indevice of=outdevice bs=1M. if is a filesyste objet to be read, of is the filesystem object to be written and bs is the block size which can be written human readablel ike 1M 2M 4M and in bytes like 1024(the old way). you use this when wipeing disks with random data. you use it when 'burning' a flash drive with a disk image like dd if=linux.iso of=/dev/sdc bs=4M. If you mess up with this as root you can easily overwrite your hard drive. do not do it to mounted filesystem
- `chsh`- change the shell for a user
- `chgroup`- change group of file... group ownership
- `chmod`- change permissions of file chmod 777 file makes everyone read write ex it, chmod 666 is read write for all.... chmod 600 is antoeh common one ls -al will show the perms
- `mount` - attaches a block device to a folder, allowing you to browse the filesystem
- `umount`- unmounts somethign takes mountpoint or /dev /device as target
- `dmesg`- prints messages generated at boot
- `env`- show ur environment vars, set them then run command(too)
- `uptime`- time up
- `wipefs`- removed disk label
- `cryptsetup` - setup luks volumes
- `cron`- service for running periodic tasks.

shells:

- `bash` -common, youre prob on it. "bourne again shell" wahteve that means
- `csh` -differentm advanced too - C shell

- `tcsh` -mac uses it? freebsd? its good
- `sh` - the most simple barebones one used when there is nothing else in some broke-ass embedded system or something

env vars:

the shell and other software uses many environment vars

these give background information about your system and things to software that needs it

this information is stored here because it doesntn eed to be chagned often, but always needs ot be sepcified

type env to see them all. echo $VAR to see VAR. export VAR=sgfsgs to set VAR to sgfsgs for your session. setting `VAR=5 someprogram`, will modify VAR for that single line running someprogram.

shell vars in general have a $ infront of them when yolu access them. but not when you set them

- `$PATH` - path to binarys, default is /bin /usr/bin /usr/local/bin etc
- `$DISPLAY` - x11/xorg display, typically :0. machines can have multiple displays, like all unix things, its multiuser
- `$PYTHONPATH` - where python looks for modules
- `$USER, $HOME`, - username and home directory path
- `$_` - arguments of last program ran?
- `alias` - it is a command that tells the shell to make a macro for other commands
- `env` shows your env
- `export` -declare env var for remainder of session until u clsoe this shell
- `jobs` - lists the jobs in shell(if you have pauzed iwth ctrl z) with jobid
- `bg <jobid>` and `fg <jobid>` - background a paused job or foreground a paused job respectively.

strange obscure barely useful:

- `motd` - message of the day, displayed on login
- `links` - text only browser
- `lynx` - older more useless text only browser
- `irssi` - irc client ncurses flavor. leet af
- `rexima` - command line sound volume control mixer thingy
- `beep` - makes a console beep

graphical

- `xterm` -old school bare bones terminal emulator for x11
- xorg/x11 - always started by scripts, but it is the name of the service that runs the GUI in linux generally. x1 was the old name xorg is the new one. there are forks...
- `xv` -old and simple image viewer
- `mplayer` -old simple and great media player. no GUI, just do mplayer file.mp4 or whatnot
- `mpv` - like mplayer but better
- `gimp` - powerful image editing, old schoool MIT project, shit interface, opens any format basically

- `ibus` - this is a package for controlling advanced input methods that are a lot more than a change of layout; like Chinese, Korean,
- `xviewer` -seems to be the version of xv/xview available in modern ubuntu? stupid name

network & hax

- `nmap` -port scanner highly advanced, many modes and options
- `masscan` -speed optimized port scanner for large volume scanning, target acquisition. usually preceeds the use of nmap whcih yields more detailed information
- `nc` -previously merntioned, netcat, raw conns
- `ettercap` -manipulation of ARP, DNS, other protocols, generally for the purpose of man in the middle attack
- `wireshark` -watch network packets go by. need to change group to work properly. can run as root and always works that way, but not recomended. used to be called ethereal - the new name sucks. still hate them for it. the new name reads like it should be the name of a chinese electrician tool or a korean children's cartoon
- `ngrep` -network grep, just reads packets going by your box and spits that out to stdout if it matches what ur looking for
- `tcpdump` -captures and dumps packets, dump files can be reloaded, minor dissection available with some calssification, can load the dumps up with anything
- `ifconfig` -old network interface config command line utility. windows ipconfig is the ripoff version with a weird name
- `ip` -the newer, 'better' network interface and routing table configuration tool
- `route` -orouting table edit and explore
- `httping` -sends a http packet to a server on default prot of 80, gives response time
- `ping` -normal old school icmp ping. not waht it used to be
- `telnet` -old school shell/terminal over the wire. completely unencrypted, not much more complex than netcat. helpful for testing connections, manual single prot probing like tenet <host> 80 to connect to port 80 on <host>
- `nslookup` -look up an ip or hostname in DNS
- `john` -old school powerful password hash cracker. supports extensions and a lot of hash algorithms. parallelism exists too, not sure about GPU kernels. likely better things these days. called john the ripper(after the famous amteur serial hooker-disection enthusiast)
- `whois` -information on domain ownership, reverse look up of IP addresses. just an entry from a database about the owner and registrar stuff for IPs and domains.
- `traceroute` -old school packet routing trace, not sure if it really works the same anymore, but shows you the path packets take to a server. seems like maye routers out in the widl drop the packets it uses now often? not sure. dont use it much and its not what it used to be is the word
- `arping` -executes a ping-analogous function using the arp protocol. v nice.
- `tsocks` -wrap any protocol through socks
- `httping`- ping a http server. IE, give the response time to a http service
- `aircrack-ng` - a suite of utilities for security analysis of wifi networks
- `iwconfig`-ike ifconfig but with specific features for wifi adapters/driver interfaces. it is old school
- `iw` - same as above but not as old school

- `bluetoothctl`- shell style interface to bluetooth hardware. quite good
- `yersinia`- a powerful security analysis too that i am not too familiar with, but worth a mention. some kid in vegas looked at me like i was insane for not using it. appears very powerful.
- `netstat`- usually i invoke as netstat -n, lists the connections in and out of the machine. godo stuff is by the top so try netstat -n|head

SSH STUFF

- `ssh` -secure shell, replaced telnet when people realizsed u could ngrep peoples files out off the network
- `ssh-keygen` - generates keypairs for ssh auth
- `scp` -copies files over ssh, wil ldefault to copy locally for composibility and uses same args generally. typical use scp user@host:/home/user/stuff stuff. username is often needed. tab to complete works if you have passwordless ssh set up. USE IT PASSWORDLESS AND USE TAB to complete. tab is slow though. remember you can copy to /tmp always, too.
- `ssh -X` - this arg will forward x11, IE, let u run graphicalprograms over ssh(if u have x11 on both sides)
- `ssh -D 8888` - runs a socks5 proxy on prot 8888 that tunnels connections from localhsot through the remote host
- `ssh -L8888:host:8888` - tunnel localhost 8888 to remove host's view of host:8888
- `ssh -R8888:host:8888` - reverse tunnel, goes from remote host to localhsots view of host:8888
- `sftp` - ftp liek client thingy for scp. never use it, might be the original client and actual protocol name for the machinery that does scp
- `sshfs` - smount - use the above sftp facilities to emualted a mounted filesystem

operators in shell(bash)

- `|` pipe, puts stdout into stdin like `cat bob|grep <word>`
- & - runs concurrently with following command.
- && - run next program sequentially
- > - stdout into a file cat bob > bobfile. OVERWRITES THE FILE
- >> - APPENDS TO THE FILE like ls >> listfile will append to the botom of nugget list the folder contents
- 2> - same as > but does stderr,
- < file on right into stdin of command on left
- <<< - string on the right into stdin on the left
- `ctrl-z` - pause - immediate effect always
- `ctrl-c` exit, doest leave shell(thats logout) clears the line though. sends a `kill -s 15` to the thread in foreground
- `ctrl-d` logout
- `[TAB]` - tab - hit this key a lot, it works to complete MANY things. used to just be files, now almsot anything. `git add [TAB] [TAB]` lists your changed files, for instance

- back quotes - *kill `pgrep firefox ` * - inserts stdout from the command in backquotes into the shell as if you had typed it. pgrep outputs a list of pids that match the string you give it, here that is being picked up by kill so that it kills anything that matches firefox
- `*` - wildcard, `ls *.py` gives list of python scripts in current directory
- `[0-9]` - matches digits in shell, `ls [0-9]*` liss eveyrhing that starts with a digit. can use comma separated singletons, works with letters too [a-z]...

root filesystem synopsis

Int the past many of these were separate partitions, hence some of the seemingly redundant things. Now this is not as important with solid state drives and (i supposed) more modern file systems

- `/tmp` - temp folder, anyone can write in it. it is there on every system and great place to copy things to if you are not sure where to do it
- `/etc` - pronounced et-SEE. all the configuration files and global settings are in here by default. in the past administration could be done exclusively by modificaion of files here, more or less. programs like passwd are tools to automatically edit files here
- `/var` - various data here, var/log is a default global spot for logs. often home to global data storage, such as the root of a webserver with static content, or database disk footprint.
- `/usr` - user installed things generally.... comes with a lot in it these days. it is like an alternative root where u generally would modify things for system wide access. has the same diretory structure as /
- `/proc` - process information emulatd as block storage devices and stuff liek this. can get info abotu some hardware from drivers, and access some other weird low level things, dynamic emulated files that are read from live executing daemons
- `/dev` - devices, filesystem emulation of actual hardware. all disks are here, your sound devices, usb devices, all accessed from here if you want to do it directly. it isa like proc, not actual files, but dynamic emulated files that make access to devices liek accessing a file
- `/opt` - not sure what it is suppsoed to be but it is often used to store globally accessed proprietary software that doesnt have facility to install in the typical global directory structure(where thigns are in /bin and /lib and sstuff
- `/bin` - binarys, these are where the commands are stored for the base system. most of the higher level suff is in /usr/bin and /usr/local/bin
- `/home` - home diretorys for each user here. all user settings and information and data are in their home folder. copy it to an ew system an it will all be there
- `/root` - home direxdtory for admin/root user
- `/boot` - ccontains the kernel and initial root disk, boot loader stuff IE GRUB. is more commonly a separate partition still
- `/cdrom` - vestigal artifact of a time when people used cdrom
- `/mnt` - this was originalyl where you would mount drives, IE, any drive that was not hosting system critical contents, like removeable media, was mounted here. you added these to be automounted using /etc/fstab, and mounting had to be done by root
- `/media` - this is where thigns are mounted now, un a path like /media/<username>/<uuid serial thing>

notable filesystem objects, global

- `/proc/cpuinfo` - cpu core info, pretty great
- `/dev/random` - random data from hardware. cat this and u get a dump of real physical entropy

- `/dev/urandom` - output of a psrng using above as seed. cat this and get infinite 'random' data generated from finite entropy harvested from ahrdware
- `/etc/passwd` - old school place where some user info is stored, originally included encrypted passwords
- `/etc/shadow` - where they moved the encrypted passwords from passwd to hide them from users when ti as realzied they could be cracked
- `/etc/hosts` - lsit of hosts that are basically added to DNS, can put some of your servers here so u dont type ip
- `/etc/hostname` - yur hostname, for some reason i feel i usualy msut edit this and use the hostname command at the same time/session
- `/etc/rc.local` - old school palce to put commands to have them run on boot, on many linux systems.
- `/etc/resolv.conf` - old way of keeping global nameservers. depends on the system now....
- `/etc/motd` - text displayed at login. put stuff here if you have users, info about the system, advertisements, cuss them out, etc

notable filesystem objects, local

- ~ - alias to your homefolder `/home/username`
- `~/.ssh/authorized_keys` - pt in a ocpy of someones id_rs.pub file as a line, and it allows the guy wti hteh private key to get in via passwordless ssh
- `~/.ssh/config` - lts u preconfig defults for various servers and things, pivotal wehn using scp and git reguarly. man ssh_config exists and shows syntax
- `~/.ssh/id_rsa.pub` - dfault place for public ssh key, without the .pub its default for private
- `~/.bashrc` - i u use bash, this is a place you can add commadns that run on login. such as adding things to ur $PATH
- `~/.bash_history` - hitory of commands in bash, some cap length, grep this to find stuff you did and need th command for
- `.profile` - tis is like .bashrc but not specific to bash. on many systems. efintiely check if you are not using bash
- `~/.local` - hs a root filesystem mirror structure that user installed things (like pip packages) can sit in. like a personal /usr/local. pip user installed stuff gos here
- `~/.config` - it is now considered bst practice for packages to put their user config files in here rather than randomly as a hidden file or folder in ~

host a git, barebones

simple amd dirty instructions always use passwordless SSH or this make git user on server. no password on it. NO PASSWORD ON IT. no way to log in with password

```
>>>
which git-shell #find path to git shell - comes with git, set this as the shell for the git user on the server. this prevents users from logging in with ssh but they can do the git operations
adduser # set git-shell full path to the shell as you go through the menu and set no password. SET NO PASSWORD
sudo su -s /bin/bash git# makes u git user and override shell so u can have an interactive session
#make folders as you need them in /home/git. cd into the folter. do:
mkdir package # to make git called package
git init
git config receive.denyCurrentBranch ignore # over rides some annoying check that maks the fist commit a pain
```

put public keys in /home/git/.ssh/authorized_keys as a line, on the host n

on cients: git clone ssh://git@server:/home/git/package

then make an initial commit to master to make sure it works

pull requests seem like a thing you dont want to do without a web interface like github

git client side

process of creating branch and merge:

```
>>>
git checkout master
git pull# - make sure its up to date
git branch mybranchname #- make a branch
git cheeckout mybranchname #- now you are on it, it is forekd off main
#do stuff
git add stuff
git commit -m"new stuff"
git push #- upload it to the remove server
#keep doing stuff, eventually ready to merge
git checkout master
git pull #-make sure its up todate
git merge mybranchname
#now if theres conflicts, you make sure it works, correct them.
#you can checkout a file from master by git checkout file, and add that one, to
#blidnly tke the master verion of file
git push
```

#there are other commands in betbween sometimes, but it will tell you waht they are git is very user friendly for a command line interface but remember to push after you merge, push and pull and clone are remote commands. rest are local

docker

docker is super helpful, especially if youre a noob. It allows you to do things as root but not destroy your baremetal system.

It was originally to make back end services scaleable, reproducible, and sandboxed while avoiding the use of a VM

docker has a built in management system for images shared by project teams and the community

stuff in docker runs on your kernel but network and disk is sandboxed and communicates through whatever avenues you specify(shared folders and port forwards)

you can run things in docker like any other program

if you dont use it youre basically failing at life

also a good way to give people root-like power on servers, without allowing them to trash the system and spy on people through unfettered hardware access

- `docker-compose` - utility for launching a few differentd ocker containers of different services, allowig you to easily config them to be interconnected in one file. simply put `docker-compose.yml` in an empty folder and edit/generate/write it to your specs. editing yaml can be kind of annoying due to autistic standards with whitespace and stuff. so work off of a copypaste
- `docker` - the normal interface to docker to run one container
- `docker stats` shows current running containers wioth resource use