Useful Commands

`sudo`	Run the given command as the root user
`ls`	List the contents of a directory
`cd`	Change directory
`cat`	Concatenate files together and print them to screen (`std out`), if only one file is given just print the file to the screen
`touch`	Create a file
`w`	Print what other users are running
`man`	Manual page viewer
`apropos`	Manual page searcher
`history`	Print the command history to the terminal
`echo`	Print to the terminal
`env`	Print the current environment variables
`whoami`	Display the user running the commands
`lastlog`	Shows the last time any user on the system has logged in
`df`	Shows the size of all the filesystems

Bang (!) commands

`!!`	Run the previous command. `sudo !!` is very helpful
`!:0`	The first word of the last command
`!$`	The last word of the previous command. Example, `mkdir /tmp/foo`, then run `cd !$` to change directory to the one you've just created
`!foo`	Run the previous command starting with foo
`!foo:p`	Print the previous command starting with foo
`!!:s/f oo/ba r`	Substitute foo with bar within the previous command
`^foo^bar`	Same as above, runs the previous command replacing foo with bar

Bang (!) commands (cont)

`!comm: s/f oo/b ar`	This will search for commands starting with comm, replace foo with bar and execute the command. Can be appended with `:p` to print out the command before running
`!n`	Will run command n in the history, replace with whatever number you want from the output of the history command. `!1` will run the first command. `!-1` will run the previous command, same as `!!`
`$_`	Variable for the last argument in the previous command.
`$()`	Will run whatever is in the `()` as a subshell and return the result before running the rest of the command

Hint in some shells you can hit space after the bang command and the command in full will populate

Remote Network Connections

`ssh`	Use the Secure Shell application to log into a given remote host
`ftp`	Use the File Transfer Protocol to download/upload a file to a given host
`telnet`	Interact with network sockets

Web Builtins

`curl`	Interact with files / webpages / API endpoints
`wget`	Download files over the network

Processes

`top`	Output top running processes
`kill`	Send signals to the given process or stop a process

Keyboard Shortcuts

`[up arrow]`	Move up in the command history
`[down arrow]`	Move down in the command history
`[left arrow]`	Move the cursor left
`[right arrow]`	Move the cursor right
`[TAB]`	Autocompletion
`[SPACE]cmd`	In some shells will not save the command to the history file
`Ctrl+a`	Move the cursor to the front of a command
`Ctrl+e`	Move to the cursor to the end of the command
`Ctrl+r`	Reverse search history
`Ctrl+c`	Terminate the current running command or clear the terminal of the command about to run
`Ctrl+S hift+c`	Copy the selected text out of the terminal window
`Ctrl+S hift+v`	Paste into the terminal window
`Ctrl+q`	Send the current command to the queue for one command before it re-populates the terminal
`Ctrl+l`	Clear the terminal

Shell Redirection Operators

`cmd > file`	Send the output of the command to a file
`cmd >> file`	Append the output of the command to a file
`cmd 2> file`	Send Standard Error (`stderr`) to a file
`cmd 2> /dev/null`	Send `stderr` to null. In essence throw away all outputted errors
`cmd1 | cmd2`	Send the output of `cmd1` to `cmd2`
`cmd1 ; cmd2`	Run `cmd1` then `cmd2`
`cmd1 && cmd2`	Run `cmd2` if `cmd1` is successful
`cmd1 || cmd2`	Run `cmd2` if `cmd1` is not successful

Networking

`ifconfig`	Lists the current networking interfaces and information about them like the assigned IP addresses
`ip a`	Like ifconfig, lists the current interfaces and associated IP addresses
`ss`	Lists active networking sockets
`nc`	Opens a network socket

Text

`wc`	Text count
`grep`	Text searcher
`nano`	Text editor
`vim`	Text editor
`emacs`	Text editor
`less`	Text viewer
`more`	Text viewer
`awk`	Text manipulator
`sed`	Text manipulator
`head`	Text reader
`tail`	Text reader

Jobs

`cmd &`	Run the command in the background as a job
`jobs`	List all running jobs
`Ctrl+z`	Send the current running process to the background and suspend it
`bg %n`	Background job ID n
`fg %n`	Foreground job ID n
`kill %n`	Kill job ID n
`kill -SIGNAL %n`	Send a given SIGNAL to job ID n
`diswon %n`	Disown the job ID n so it will run even if the terminal exits