# Linux Crash Course

## Essential Commands

### ls

ls # Lists the contents of the current working directory.  
  
ls <path> # Lists the contents of the specified path.  
  
ls -l # Lists all the contents along with its owner, permissions and timestamp.  
  
ls -a # Lists all the contents including hidden items.  
  
ls --author # Lists all the contents along with its owner.  
  
ls -S # Lists all the contents sorted by size.  
  
ls \*.php # Lists only the items with the specified format.

### cd

cd # Changes the directory to the home directory ( ~/ ).  
  
cd ~ # Changes the directory to the home directory ( ~/ ) too.  
  
cd / # Changes the directory to the root directory ( / ).  
  
cd .. # Changes the directory to its parent directory.  
  
cd "new folder" # Use quotes if the directory name contains spaces.

### cat

cat <file> # Displays the contents of text files and concatenates several files into one.  
  
cat -b <file> # Adds line numbers to non-blank lines.  
  
cat -n <file> # Adds line numbers to all lines.  
  
cat -s <file> # Squeezes blank lines into one line.  
  
cat -E <file> # Shows $ at the end of the line.  
  
cat > file.txt # Create a new file and enter text contents to it from the terminal. Press ctrl+d to exit insert mode.  
  
cat >> file.txt # Appends text content to an existing file from the terminal. Press ctrl+d to exit insert mode.  
  
cat file1.txt >> file2.txt # Appends the contents of the first file to the second.

### grep

grep # Searches for a specific string in text. Grep stands for "Global Regular Expression Print".  
  
grep "target" file.txt # Returns all the matches for the string "target".  
  
grep -i "target" file.text # Returns all the matches (case insensitive).  
  
grep -n "target" file.txt # Returns all the matches along with their line number.  
  
grep -v "target" file.txt # Returns the lines which doesn't include the search string.  
  
grep -c "target" file.txt # Returns how many lines/matches was found.

### sort

sort # Sorts the results of a search either alphabetically or numerically. Files. file contents and directories can be sorted.  
  
sort file.txt # Sorts the contents of file.txt and returns them in alphabetical order (does not modify the original file contents only the output).  
  
sort file1.txt file2.txt # Sorts the contents of both file1.txt and file2.txt.  
  
sort -r file.txt # Returns the results in reverse order.  
  
sort -f file.txt # Case insensitive sorting.  
  
sort -n file.txt # Returns the results as per numerical order.

### Pipeline |

The Pipe command | is used to output the result of one command as input to another command.

# Example:  
grep target file1.txt file2.txt | sort # Searches for the string "target" from both files and sorts the results. You can use multiple pipes to perform multiple operations in one line.

### cp

cp <options> <source> <destination> # Copies files and directories  
  
cp -i # Enters interactive mode; CLI asks before overwriting files.  
  
cp -n # Doesn't overwrite files.  
  
cp -u # Updates the destination file only when sources file is different from the destination file; Skip if the file are the exact same.  
  
cp -R # Recursive mode for copying directories; Copies even hidden files.  
  
cp -v # Verbose; Prints source and destination files.

### mv

mv <options> <source> <destination> # Moves files and directories.  
  
mv -i # Enters interactive mode; CLI asks before overwriting files.  
  
mv -u # Updates the destination file only when sources file is different from the destination file; Skip if the file are the exact same.  
  
mv -v # Verbose; Prints source and destination files.  
  
mv Hello.txt HelloWorld.txt # Renames the file `Hello.txt` to `HelloWorld.txt`.

### mkdir

mkdir # Creates a new directory.  
  
mkdir -p # Creates both a new parent directory and a sub-directory.  
  
mkdir --parent # Creates both a new parent directory and a sub-directory.  
  
mkdir {d1,d2,d3} # Creates multiple directories inside the parent directory.

### rm & rmdir

rmdir # Removes the specified directory only if it's empty.  
  
rmdir -p # Removes both the parent and child directories.  
  
rmdir -v # Verbose.  
  
rm # Removes files.  
  
rm -r # Recursive mode; Removes directories (empty and non-empty directories).  
  
rm -rp # Removes non-empty directories including parent and child directories.

### File Content Commands:

head # Displays the first 10 lines of the file.  
  
tail # Displays the last 10 lines of the file. Useful for reading error messages.  
  
tac # Displays the file content in reverse order; It's the reverse of cat.  
  
more # Similar to the cat command, it's used to display content. The only difference is in case of larger files where a screenful output is displayed.  
  
less # Similar to the more command, It automatically adjusts the size of the terminal window.

### User Commands:

su # Used to switch to root user (super user privileges).  
  
su <username> # Switches to a different user.  
  
sudo <command> # Executes a command with super user privileges.  
  
id # Displays the user ID (UID) and group ID (GID).  
  
id -g, --group <username> # Print only the effective group ID  
  
id -G, --groups <username> # Print all group IDs  
  
id -n, --name <username> # Print a name instead of a number, for -ugG  
  
id -r, --real <username> # Print the real ID instead of the effective ID, with -ugG  
  
id -u, --user <username> # Print only the effective user ID  
  
whoami # Displays the current user.

### Filter Commands:

cut # Selects a specific column of a file. With no FILE, or when FILE is -, read standard input.  
  
cut -d<separator> -f<ColumnNumber> <filename> # The `-d` option is used as a separator at which the column ends.  
  
comm # Compares two files or streams. By default it displays 3 columns. the first displays non-matching columns of the first file, second for second file and third for third file.  
  
sed # Edit files using a regular expression. It doesn't modify the files; instead the edited content remains only on display. Known as a stream editor.  
  
<command> | sed 's/<oldWord>/<newWord>/'  
  
tee # Copy standard input to each FILE, and also to standard output.  
  
tr # Translates the file content like from lowercase to uppercase.  
  
uniq # Forms a sorted list in which every word will occur only once.  
  
wc # Prints newline, word, and byte (character) counts for each FILE, and a total line if more than one FILE is specified.

### Miscellaneous

pwd # Displays the current working directory.  
  
echo # Writes its arguments to standard output.  
  
clear # Clears the terminal's content.  
  
exit # Exits the current terminal session.  
  
rename # Renames files. Useful for renaming a large group of files.  
  
rename <target-pattern> <input-pattern> # Renames all the files matched by the input pattern with the target pattern.  
  
man <command> # Shows the manual page of the tool/command.  
  
watch # Runs a specific command periodically and displays the output. Helpful for monitoring changes.  
  
watch -n 1 <command> # Displays the output every 1 second.  
  
watch -d 1 <command> # Displays the output every 1 second.

## User Permissions

**File Types:**

* Normal File => -
* Directory => d
* Character Special File => c
* Binary Special File => b

**File Permissions:**

* Read => r
* Write => w
* Execute => x

**Format:**

[file/dir type] [user permissions] [group permissions] [other's permissions]

**Example:**

* -rwx-rw-r-- => File with read/write/execute permissions for the user, read/write for the group and read-only for other’s.

**Numeric Format:**

* Read (r) = 4
* Write (w) = 2
* Execute (x) = 1

| Permission | Numeric Value |
| --- | --- |
| — | 0 |
| –x | 1 |
| -w- | 2 |
| -wx | 3 |
| r– | 4 |
| r-x | 5 |
| rw- | 6 |
| rwx | 7 |

**Other Segments:**

[File/Dir Type] [Permissions] [Symbolic Links] [Owner Name] [Group Name] [File Size in Bytes] [Timestamp]

### Changing Permissions

chmod # Changes the access permissions of files or directories.  
  
chown # Change the owner of files or directories.  
  
chgrp # Change the group ownership of files or directories.  
  
chmod g+wx file.txt # Gives write and execute permissions to group members.  
  
chmod u=rwx,o-wx file.txt # Gives read, write and execute permission to owners, and removes the write and execute ownership from other members.  
  
chown username file.txt # Changes the owner of the specified file.  
  
chown username:groupName file.txt # Changes both the owner and group ownership of the specified file.  
  
chgrp groupName file.txt # Changes the group ownership of the specified file.

## Tar Files

When you download a package from the internet the downloaded file comes in compressed form.

The following commands can be used to compress and decompress files in Linux:

* gzip: To compress files with .gz format.
* gunzip: To decompress .gz files.
* tar: To compress and decompress files with .tar format.

**Example:**

tar -cvf output\_file input\_file # Compresses files to tar format.  
  
tar -xvf input\_file # Decompresses files in tar format.  
  
tar -c, --create # Create a new archive.  
tar -x, --extract, --get # Extract files from an archive.  
tar -f # Use archive file.

## Environment Variables

Environment variables control the behavior of the software packages installed in Linux.

The path where the packages have been installed will be specified in environment variables.

**Related Commands:**

printenv # Prints the list of all environment variables.  
  
# Basic Variables:  
$HOME # The path of the home directory of the user.  
$PATH # A colon-separated list of directories in which shell looks for commands (executables).  
$HOSTNAME # The hostname.  
$USERNAME # The username.  
$LANG # The device's language.  
$BASH\_VERSION # The version of the bash shell being used.

## Regular Expression

Regular Expressions (RegEx) are used to match/search patterns in strings.

It can be piped along with grep command to find patterns of text in the file.

| Symbol | Explanation |
| --- | --- |
| . | Replaces any character |
| ^ | Matches the start of the string |
| $ | Matches the end of the string |
| \* | Matches the preceding character zero or more times |
| ? | Matches the preceding character one or more times |
| () | Groups regular expressions |
| \ | Escapes special characters |

## Processes, Adding Users and SSH

### Processes

Process: An instance of a program is called a process.

Any command given to Linux kernel starts a new process.

The same program can have multiple processes (instances).

Processes is divided into two types:

* Foreground Processes
* Background Processes

You can run top or htop commands to see all the running processes.

**Process Form:**

[PID] [USER] [PR] [NI] [VRT] [RES] [SHR] [S] [%CPU] [%MEM] [TIME+] [COMMAND]

Here’s a breakdown of each part:

* PID: Process ID
* USER: Username
* PR: Priority; Ranges from -20 to 20
* NI: Niceness Value; Ranges from -20 to 20
* VRT: Virtual Memory
* RES: Physical Memory
* SHR: Shared Memory
* S: Status
* %CPU: CPU Time
* %MEM: Physical Memory Used
* TIME+: Total CPU Time
* COMMAND: The Command that’s running the process

**Related Commands:**

bg <command> # Runs the command as a background process.  
  
ps ux # Lists all processes started by the user.  
  
ps # Lists all processes started by all users.  
  
pidof <process> # Returns the PID of the process.  
  
kill <PID> # Kills the selected process.

### Adding Users

**Users:**

sudo useradd <username> # Add a new user.  
  
sudo passwd <username> # Set the password for the selected user.  
  
sudo userdel <username> # Delete a user

**Groups:**

sudo groupadd <groupName> # Add a new group.  
  
sudo groupdel <groupName> # Delete a group.  
  
sudo usermod -g <groupName> <username> # Add a user to a primary group.

### SSH

SSH stands for Secure Shell, it’s used for remote machine access.

**Steps To Setup SSH:**

# Stop the firewall  
sudo service iptables stop # Drops the IP table temporarily.  
sudo chkconfig iptables off # Permanently disables the IP table.  
  
sudo nano /etc/hosts # Add the below IP addresses in both the master and slave's hosts file.  
# master 192.168.56.102  
# slave 192.168.56.103  
  
ip addr show # Show the IP address.  
  
sudo ip addr del <IP-address-from-prevoius-command> dev eth1 # Deleting the existing IP.  
sudo ip addr add 192.168.56.102/24 dev eth1 # Adding IP at the master's node.  
sudo ip addr add 192.168.56.103/24 dev eth1 # Adding IP at the slave's node.  
  
ssh master # Running this command at the slave node will give remote access to master.  
ssh slave # Running this command at the master node will give remote access to slave.

## The Kernel & The Shell

**What’s The Kernel?**

The computer programs that allocate the system resources and coordinate all the details of the computer’s internals is called the operating system or the Kernel. Users communicate with the OS through a program called the Shell.

**What’s The Shell?**

The Shell is the command line interpreter. it translates commands entered by the user and converts them into a language that’s understood by the Kernel.

**What’s a Shell Script?**

It’s a set of commands written in the order of the execution to perform a specific task.

## Evolution of The Shell

### Bourne Shell (SH)

Created by Stephen Bourne in 1977, the Bourne shell is a useful shell even today and in some cases, as the default root shell. Its grammar is very similar to the Algorithmic Language (ALGOL).

The Bourne shell had two primary goals: serve as a command interpreter and for scripting.

### C Shell (CSH)

The C Shell was developed by Bill Joy in 1978 with the objective of achieving a scripting language similar to C Programming Language. This was useful given that C was a primary language in use back then which also made it easier and faster to use.

### Korn Shell (KSH)

Developed by David Korn in 1983, the Korn Shell combined features of both Bourne Shell and C Shell. It’s Backward compatible with the former.

It includes features from the C Shell such as job control, command aliasing and command history.

### Bourne-Again Shell (BASH)

One of the most widely used shells, today, it was written by Brian Fox in 1989 for the GNU project as a pre software replacement for the Bourne Shell. Shows all features from the Bourne Shell but is much more efficient and easy to use.

It supports filename globbing, piping, command substitution, and control structures for conditional testing and iteration.

### Shell vs Bash

**Shell Script:** The Shell Command Language is a programming language according to the POSIX Standards. It is a specification.

**BASH:** Bash is one of the many implementations of the Shell Command Language.

### Which Shell Is For You

**C Shell:** Ideal for learners comfortable with the C programming language. Mostly network admins use this shell.

**Korn Shell:** This highly interactive shell is generally used to automate a wide variety of operating system and system administration tasks.

**Bourne-Again Shell:** Versatile and easy to learn, most widely taught and learnt shell for scripting and interfacing purposes.