<https://drive.google.com/drive/u/0/folders/1-kCl4jXjtJHmwSkkmaYBGFrUuPcXO3pI> - Prasanth Drive Link

Linux Basics

Linux is an opensource operating system and community developed Operating system for computers, servers, mainframes, mobile devices and embedded devices.

**Difference between Unix and Linux:**

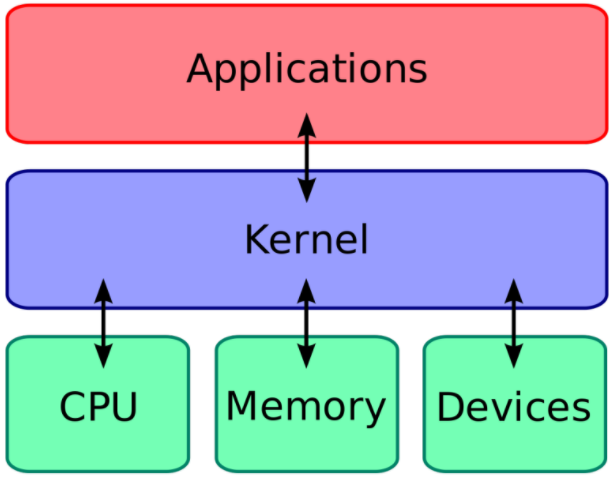
Unix is installed only on specific Hardware. Linux is mostly free and Opensource. Unix was firstly developed for multi user and multitasking purposes in mid 1970s. Unix mostly used by SUN as Solaris. Unix supports only fewer file systems.

**What is a kernel in Linux?**

The Linux kernel is the main component of a Linux operating system (OS) and is the core interface between a computer's hardware and its processes. It communicates between the 2, managing resources as efficiently as possible

Important parts of the kernel

* Process Management
* Memory Management
* Hardware device drivers
* Filesystem drivers
* Network drivers
* Various other bits and pieces



**What is a shell?**

The shell is a command line interpreter. It takes each command and passes it to the OS kernel to be acted upon. It displays results of this operation on your screen. There are various types of shells like Bourne Shell, C Shell, Korn Shell, TC shell, bash …

|  |  |
| --- | --- |
| sudo cat /etc/shells | Lists out no.of shells in your OS |

**Difference between commands, terminal and the shell**

Commands are plain text interpreted by the shell. Different shells can interpret the same text in different ways. Commands are case sensitive. Terminal is window to the shell. Commands added to the PATH are directly accessed.

**Different Linux distributions**

* Redhat
* Centos
* Fedora
* OpenSuse
* Debian
* Ubuntu etc.,

**Linux Vs. Windows**

|  |  |  |
| --- | --- | --- |
|  | Linux | Windows |
| Price | Free | $ |
| Ease | Not user friendly | User Friendly |
| Reliability | Runs for years | Often requires reboot |
| Software | Mostly Enterprise level SW | Much larger selection of SW |
| Multi-tasking | Best for multi tasking | Available with high CPU/Memory |
| Security | Very Secure | Somewhat secure |
| Opensource | Open to public | Not an opensource |

**Where is Linux used?**

US Government, Bullet Trains in Japan, NASA, Traffic Control, NYSE, Amazon, Ebay, Healthcare, fortune 500 companies, startups etc.,

**What is SSH?**

Secure Shell, sometimes referred to as Secure Socket Shell, is a protocol which allows you to connect securely to a remote computer or a server by using a text-based interface.

Graphical user interface, text

Description automatically generated

To connect to your linux vm or to an EC2 instance, we should have a ssh client on our Local machine. So we install either putty or gitbash.

To connect to the VM from host, we run the command on gitbash, putty or mobaXterm

|  |  |
| --- | --- |
| ssh user@hostname  ssh user@ipaddress | Port 24 to connect to the server |

**Few things about root user?**

Linux has super admin root. Root is the most powerful account that can create modify delete accounts and make changes to system configuration files. /root is the root user’s home directory.



**Linux File system:**

An operating system stores data on disk drives using structure called a file system consisting of files, directories and information needed to access and locate them. Many different types of files systems are present like EXT2, EXT3, XFS, . Linux file system store information in hierarchy of directories and files

A picture containing text, electronics, keyboard, several

Description automatically generated

* /boot : contains file that is used by the boot loader (grub.cfg)
* /root : root user home directory. It’s not the same as /
* /dev : System Devices
* /etc : Configuration files
* /bin: Everyday user commands
* /sbin: System or file system commands
* /opt: Optional add on apps
* /proc: Running processes (only exist in memory)
* /lib: C program library files needed by commands
* /tmp: directory for temporary files
* /home: directory for user • /var: system logs
* /run: System daemons that run/ start very early to store tmp runtime files like PID files
* /mnt : to mount external file systems (Ex. NFS)
* /media: for CDROM Mounts

**Filesystem paths**

2 paths to navigate file system

Absolute path : begins with a “/” . from the root directory

Relative Path: to navigate to folders in the existing folder

**Directory listing attributes:**

* Type of file
* Permissions
* Number of Links
* Owner
* Group
* Size
* Month
* Day
* Time
* Name of file

**Linux File Types:**

|  |  |
| --- | --- |
| File symbol | Meaning |
| - | Regular file |
| d | Directory |
| l | Link |
| c | Special file or device file |
| s | Socket |
| p | Named pipe |
| b | Block device |

**Commands**

*Few points to remember:*

* Commands are case sensitive
* In Commands we can include options and inputs
* Options change the behavior of a command (ex. ls and ls -a)
* Commands operate on inputs (ex. echo and echo “Hello World”)
* Command names need to be on shell’s search path (echo $PATH)
* Command options you can either issue with long form or short form. For short form, you can merge multiple options. Not possible with long forms.
* Tab Completion – autocompletes the commands.
* To find commands that begin with letter t, press t and tab twice

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | = | CommandName | -Options | Inputs |
| Ex: |  | ls | -l | file1 |
|  |  | echo | -e | “horses\nDonkeys” |

Navigating file System

cd – change directory

|  |  |
| --- | --- |
| cd /home/ubuntu/dir1 | Takes to dir1 in /home/ubuntu |
| cd | Takes to user’s home directory |
| cd ~ | Takes to current user’s home directory |
| cd .. | Takes a step back to parent directory |
| cd . | Within the directory itself |
| cd / | Takes to the absolute beginning root directory |

pwd – print working directory

|  |  |
| --- | --- |
| ubuntu@ubuntu1804:~$pwd | Prints the directory where control is currently in.  ~ represents current user’s working directory so it prints  /home/ubuntu |
| root@ubuntu1804:~#pwd | Prints root user’s home directory which is /root |

ls – lists out the files and directories present in current location

|  |  |
| --- | --- |
| ubuntu@ubuntu1804:~$ls | Lists all files and directories in current user’s home directory. Just names |
| *ubuntu@ubuntu1804:~$dir* | *Does the same but difficult to identify directories and files* |
| ubuntu@ubuntu1804:~$ls -l | Long listing with metadata. |
| ubuntu@ubuntu1804:~$ls -l -a | Lists all hidden files |
| ubuntu@ubuntu1804:~$ls -l – h | Human readable format |
| ubuntu@ubuntu1804:~$ls -lt | Based on timestamp, sorts, oldest first |
| ubuntu@ubuntu1804:~$ls -ltr | Based on timestamp, sorts, newest first |
| ubuntu@ubuntu1804:~$ls -li | Prints the inode numbers of files and directories |
| ubuntu@ubuntu1804:~$ls -l file1 | Only the metadata of file1 gets printed |
| ubuntu@ubuntu1804:~$ls -F | Lists all directories with /at the end |
| ubuntu@ubuntu1804:~$ls ~ | Lists current user’s working dir |

Hidden files

Hidden files are the files that contain dot/period at the beginning of the filename. To create a hidden file in Linux, preceed the file name with a period .

Every inode in the Linux structure has a unique number identified with it. It is also called the index number and has the following attributes:

* Size
* Owner
* Date/time
* Permissions and access control
* Location on the disk
* File types
* Number of links
* Additional metadata about the file

cal – calendar

|  |  |
| --- | --- |
| cal 2020 | Displays entire 2020 calendar |
| cal -A 1 12 2020 | Displays 1 month after Dec 2020 |
| cal -B 1 12 2020 | Displays 1 month Before Dec 2020 |
| cal 12 1988 | Displays December month of year 1988 |
| cal -y | Gives current year |
| man cal | Manual for cal command |

date – displays day, month, date, time, zone, year

|  |  |
| --- | --- |
| date | Displays date and time |
| date -u | UTC format |
| date –universal | Longform of previous command |

history – History command. Contains all the history of command executed so far

|  |  |
| --- | --- |
| History | A list of commands executed so far. |
| !45 | The command at 45th location gets executed |
| !! | Most recent command in list gets executed |
| history -c -w | Clears the history and writes changes. |

which - To see where a command is located

|  |  |
| --- | --- |
| which man | Shows the location of command |

man - How to use each command?

|  |  |
| --- | --- |
| man ls | Manual page to command ls |
| man echo | Manual page to echo command |

How manual is structured? man -k commandname

1. User Commands
2. System Calls
3. C Library functions
4. Devices and Special Files
5. File formats and conventions
6. Games
7. Miscellaneous
8. System Administration (root privileges)

Help Commands

|  |  |
| --- | --- |
| Whatis | Single line description of a command |
| Man | Manual page |
| Help | documentation |
| command --help | Quick help on any command |

Creating Directories and files

|  |  |
| --- | --- |
| Mkdir | To create directory |
| mkdir -p | To create directories within directories |
| Touch | To create empty files / to update timestamps on existing files |
| cat | To view files |
| cat > | To create files with content  To overwrite existing files |
| tac filename | Displays last lines first |
| rev filename | Displays lines in right to left style |
| cat >> | To append data to a file |
| vim | Text editor |
| Rm | To remove files |
| Rmdir | To delete empty directories |
| rm -r | To delete non empty directories |
| rm -ri | To remove files with our permission |

Copying files, moving files and directories:

|  |  |
| --- | --- |
| cp file1 file3 | Copies file1 as file3 |
| cp file1 /home/ubuntu/file4 | Copies file1 as file4 in Ubuntu user’s home directory. If file4 is already present there, this new file gets overwritten on existing file |
| mv file1 file7 | Renames file1 to file7 |
| mv file1 /home/ubuntu/file1 | Movies file1 from current location to ubuntu user’s home directory |

Command Input and Output

* Standard Input 0
  + Input from file to stdin

cat 0< input.txt

* Standard Output 1
* Standard Error 2
* Command Line arguments

Piping

You can redirect the standard output of one command to the standard input of another in a process known as piping.

Ex. cat /etc/passwd | grep ubuntu

date | cut --delimiter “ “ --fields 1

date | cut --delimiter “ “ --fields 1 > today.txt

Redirection:

Accepts standard input, prints on screen passes to standard output

Ex. echo “Hello World” > new.txt

|  |  |
| --- | --- |
| cat 0< input.txt | Reads input.txt file. Similar to  cat input.txt |
| cat 1> output.txt | write the output to a text file |
| cat 1>> output.txt | Writing to the file using redirection without truncating |
| (cal;date) 1> caldate.txt | Date and cal output stores in caldate.txt |
| (cal;data) 1> caldate.txt | only cal output gets stored in caldate.txt |
| (cal;data) 2> caldate.txt | only data error gets stored in caldate.txt |
| (cal;data) &> caldate.txt | error redirection. Both cal and data error gets stored |

*Exercise : Redirect text from one terminal to another terminal.*

Tee Command

To Save a data snapshot without breaking pipelines, use the tee command.

date | tee date.txt | cut --delimiter “ “ --fields 1

Xargs

If a command doesnot accept stdin, but you want to pipe it, use xargs. Echo does not accept standard inputs. Instead it accepts only Command Line arguments

Ex: date | echo (doesn’t give any output)

date | xargs echo (gets output. Converts date into CLA)

Ex: filestodelete is a file that contains filenames that are supposed to be deleted. To delete those files, if we give

cat filestodelete | rm (it doesn’t work)

cat filestodelete | xargs rm

(it works since xargs converts standard inputs to command line arguments)

Find and Locate Commands

Locate uses a prebuilt database which should be regularly updated.

To find all files with .conf extension with locate command,

locate \*.conf (\* is a wildcard character)

Locate command is much faster than find but it can be inaccurate if the database is not updated. To update the database, run updatedb

|  |  |
| --- | --- |
| sudo updatedb | Updates the database |

Few commands over locate

|  |  |
| --- | --- |
| locate \*.conf | Locates all files with .conf extension |
| locate -i \*.loG | Locates all log files. Ignores case |
| locate -i --limit 3 \*.log | Limits only to 3 search results |

Find iterates over a filesystem to locate files.

find . -name “file you are searching for”

|  |  |
| --- | --- |
| Find |  |
| find / |  |
| find . -maxdepth 1 |  |
| find . -type f | Finds all files in current dir |
| find . -type d | Finds all directories |
| find . -type d -maxdepth 5 | Digs 5 directory structures deeper |
| find . -name “hello.txt” | Finds hello.txt from current dir |
| find . -maxdepth 1 -iname “heLLo.tXt” | Ignores case. Finds files by crawling into 1 directory deeper |
| find / -type f -size 10k | Find files with size 10Kilobytes |

WildCards:

Wildcards are used to build patterns called “Regular Expressions”

Anything that matches the pattern will be passed to the command line arguments.

|  |  |
| --- | --- |
| \* | Anything irrespective of length.  0 or more |
| ? | Anything but just one place |
| [ ] | One place but allows to specify options |

|  |  |
| --- | --- |
| ls \* | Lists all files and directories |
| ls D\* | Lists all folders and files beginning with letter D |
| ls ?.txt | Lists all files that have only one character before .txt |
| ls ??le.\* |  |
| ls file[0-9][a-z][A-Z].txt | |

Brace Expansion:

To create multiple files in one single command,

mkdir dir\_{1..10}

This command creates 10 directories with naming convention of dir\_1, dir\_2, dir\_3…. dir\_10

If it’s not working on your distro, enable bracket expansion feature

To enable : set -B

To disable: set +B

**Quick Summary:**

touch to create files

mkdir to create directories

mkdir -p to create entire folder paths

No spaces in filenames instead use underscores

Brace expansion can be used to do very complicated things

Rm command to remove items from your system

Wildcards can be used to make commands even more powerful

**Viewing long files:**

Less : command to page through large amounts of data

Ex: less /etc/passwd (or) cat /etc/passwd | less

Head : Allows you to see the top lines

Ex: cat /etc/passwd | head -5

Tail : To see the bottom of the file

Ex: cat /etc/passwd | tail

cat /etc/passwd | tail -n 5 (you can say tail -5 as well)

To find number of lines, characters,words in a file

|  |  |
| --- | --- |
| wc -c | # of characters |
| wc -w | # of words |
| wc -l | # of lines |

Sorting: To sort a file with random words in an order

|  |  |
| --- | --- |
| sort words.txt | tac | Sorts in reverse order |
| sort -r words.txt | Same as above command |
| sort -n numbers.txt | Sorts numbers in ascending order |
| sort -nr numbers.txt | Sorts numbers in descending order |
| ls -l /etc |head -n 20 | sort -k 5 | |
| ls -l /etc | head -n 20 | sort -k 5 nr | |

Searching file Content (grep)

cat /etc/passwd | grep UbUntu (no output)

cat /etc/passwd | grep -i UbUntu (ignores case and gives output)

cat /etc/passwd | grep -v ubuntu (all info except ubuntu)

Finding files through grep command

ls -l | grep hello.txt

ls -F /etc | grep -v /

**File archiving and compression:**

How to create zip files

zip <thenameyougive> <files you want to add>

To Unzip

unzip <thenameofzipfileyoucreated/existingzipfile>

Tar & gzip

Step1: Create Tar file - tar -cvf <name.tar> <files to add>

Step2: Gzip the tar – gzip <name.tar>

To extract gzip

tar -xvzf <name.tar.gz>

To Tar and gzip in one go:

tar -cvzf <name.tar.gz> <files>

*To untar and extract*

tar -cvzf <name.tar.gz>

Another Archiving and compression technique bzip2

tar and bzip2 in a go

tar -cvjf <name.tar.bz2> <files>

*To untar and extract*

tar -xvjf name.tar.bz1

**Soft Links and Hardlinks**

SoftLink: Link will be removed if a file is removed/renamed. Softlink points to the original file. Softlink is created with ln -s

Hardlink: Deleting/moving the original file will not affect the hardlink. Hadlink points to the inode. Hardlink created with ln command

|  |
| --- |
| To create a soft link to a file in /home/ubuntu/dir1/dir2/file1.txt on /home/ubuntu directory,  Navigate to /home/ubuntu  ln -s <name> /home/ubuntu/dir1/dir2/file1.txt  A softlink gets created  If you delete/rename the original file, your softlink file becomes a dangling softlink |
| To create a hard link to a file in /home/ubuntu/dir1/dir2/file1.txt on /home/ubuntu directory,  Navigate to /home/ubuntu  ln <name> /home/ubuntu/dir1/dir2/file1.txt  Deleting or renaming the original file doesn’t affect the hardlink as hardlink is pointing to the inode number of the original file. |

**Combining files :**

Combines the output of file1,file2 and file3 to onefile.txt

cat file1 file2 file3 > onefile.txt

**Splitting files:**

Let’s say file.txt contains 500 lines and I want to split every 100 lines into a new file. I can do that using the below command. I get 5 files with file\_aa, file\_ab, file\_ac,file\_ad,file\_ae

split -l 100 file.txt file\_

**Truncate file Size:**

The linux truncate command is often used to shrink or extend the size of a file to the specified file

truncate -s <sizeyouwanttoreduceindigits> <filename>

If the file size is reduced, some of the characters in the file gets vanished. If the file size is greater, the extra size is replaced by ‘0’ characters.

**File Permissions:**

We can assign 3 types of permissions on files and directories. These permissions can be restricted to users, groups and other users. File permissions can be assigned in numerical or alphabetical notation.

|  |  |  |
| --- | --- | --- |
|  | Numerical | Alphabetical |
| Read | 4 | r |
| Write | 2 | w |
| Execute | 1 | x |

The permissions work on a link, or a directory or on a file. Since directories have executable permissions, we are allowed to CD into those directories. Only root user has privileges to access all the directories.

To change permissions on a file, we use the command chmod

|  |  |
| --- | --- |
| chmod a+rwx <filename> | Assigns all permissions to the file |
| chmod u+x <filename> | Assigns executable permission to owner/user |
| chmod 755 filename | Assigns read,write,execute to owner, read and execute to group and other users |
| chmod u-x,o-rwx filename | Removes execute permission to owner, and all permissions to other users |

Cut Command

whatis cut – removes sections from each line of files.

Ex:

ls -l | cut -c1-2 (displays the first 2 characters in the list command)

AWK Command

whatis awk – pattern scanning and text processing language

Ex:

ls -l | awk ‘{print $1}’ (displays the first column in list output)

uniq command:

uniq command identifies unique occurrences in the file. No duplicates. A sort has to be performed before uniq command is run.

Awk ‘{print $2}’ | sort |uniq

***Comparing files***

Diff (line by line)

Cmp (byte by byte)

**SED (stream Editor)**

Substitutes text within another text

Replace a string in a file with a new string

Find and delete a line

Remove empty lines

Remove first or n lines in a file

Replace tabs with spaces

Show defined lines from a file

Substitute within vi editor and much more

Ex: To replace all occurrences of nani to Prashanth

sed ‘s/nani/prashanth/g’ <filename>

SED doesn’t affect the changes in the original file. It only outputs the changes on the screen. If you want to commit changes on the file,

sed -i ‘s/nani/prashanth/g’ <filename>

To remove a name from file,

sed -i ‘s/docker//g’ <filename>

To delete any character in the file

sed ‘/docker/d’ <filename>

To remove file 1st line

sed ‘1d’ <filename>

sed ‘1,2d’ <filename>

To remove all the tabs in file

sed ‘s/\t//g’ <filename>