LINUX NOTES

By,

Shantanu Patil

CONTENT

Chapter 1: Introduction to Linux

Before learning Linux, we should know what operating system is. An Operating System or OS is the program which provides an environment to run other applications. OS interacts with hardware which means it acts as mediator between hardware and user interface. Based on their utilization, OS can be classified as Desktop OS and Server OS. Following are some differences between them,

Desktop Operating System	Server Operating System
• Basically, used in Desktop System. • It runs client applications like Photoshop, Office, Games, etc. faster. • Easy to access web services, like email, browsing, searching. • Rich media services. • Easy to use for non-expert users. • Can work on a domain as a member (professional edition only - home edition cannot join a domain). • Rich connectivity support (LAN, wireless, Bluetooth, etc.) • Cheaper than the server version. • Ex. Windows7, LinuxMint, Ubuntu Desktop, etc.	• Basically, used in server systems. • Unlimited user connections. • Use of large amounts of memory. • Can act as web server, database server, email server and other server-like roles. • Optimized for network, instead of local application execution extended management. • Extended fault tolerance to avoid downtimes. • Can hold a domain. • Expensive. • Ex. Windows Server 2008, RHEL, Centos, Debian, etc.

History of Linux:

Evolution of Computer

In earlier days, computers were as big as houses or parks. So, you can imagine how difficult it was to operate them. Moreover, every computer has a different operating system which made it completely worse to operate on them. Every software was designed for a specific purpose and was unable to operate on other computer. It was extremely costly and normal people neither can afford it nor can understand it.

Evolution of UNIX

In 1969, a team of developers of Bell Labs started a project to make a common software for all the computers and named it as 'UNIX'. It was simple and elegant, used 'C' language

instead of assembly language and its code was recyclable. As it was recyclable, a part of its code now commonly called 'kernel' was used to develop the operating system and other functions and could be used on different systems. Also, its source code was open source. Initially, UNIX was only found in large organizations like government, university, or larger financial corporations with mainframes and minicomputers (PC is a microcomputer).

UNIX Expansion

In eighties, many organizations like IBM, HP and dozen other companies started creating their own UNIX. It results in a mess of UNIX dialects. Then in 1983, Richard Stallman developed GNU project with the goal to make it freely available UNIX like operating system and to be used by everyone. But his project failed in gaining popularity. Many other UNIX like operating system came into existence but none of them was able to gain popularity.

Evolution of Linux

In 1991, Linus Torvalds a student at the University of Helsinki, Finland, thought to have a freely available academic version of UNIX started writing its own code. Later this project became the Linux kernel. He wrote this program especially for his own PC as he wanted to use UNIX 386 Intel computer but couldn't afford it. He did it on MINIX using GNU C compiler. GNU C compiler is still the main choice to compile Linux code, but other compilers are also used like Intel C compiler. He started it just for fun but ended up with such a large project. Firstly, he wanted to name it as 'Freax' but later it became 'Linux'. He published the Linux kernel under his own license and was restricted to use as commercially. Linux uses most of its tools from GNU software and are under GNU copyright. In 1992, he released the kernel under GNU General Public License.

Linux Today

Today, supercomputers, smart phones, desktop, web servers, tablet, laptops and home appliances like washing machines, DVD players, routers, modems, cars, refrigerators, etc. use Linux OS.

Linux	UNIX
The Source Code of Linux is freely available to its Users.	The Source Code of UNIX is not available for the public.
Linux primarily uses Graphical User Interface with an optional Command Line Interface.	UNIX primarily uses Command Line Interface.
Linux OS is portable and can be executed in different Hard Drives.	UNIX is not portable.
Linux is very flexible and can be installed on most of the Home-Based and Servers.	UNIX has a rigid requirement of the Hardware. Hence, cannot be installed on every other machine.

Different Versions of Linux are Ubuntu, Debian, OpenSuse, RedHat, Solaris, etc.	Different Versions of UNIX are AIS, HP-UX, BSD, Iris, etc.
Linux Installation is economical and doesn't require much specific and high-end hardware.	UNIX Installation is comparatively costlier as it requires more specific hardware circuitry.
The File systems supported by Linux are as follows: xfs, ramfs, nfs, vfat, cramfsm ext3, ext4, ext2, ext1, ufs, autofs, devpts, ntfs Linux is development by an active Linux Community worldwide.	The File systems supported by UNIX are as follows: zfs, js, hfx, gps, xfs, gps, xfs, vxfs. UNIX is developed by AT&T Developers

Architecture of Linux:

Linux architecture consist of inner most Hardware layer, kernel, shell, and outer application layer, as shown in fig 1.1. Each layer has different functionalities and different uses.

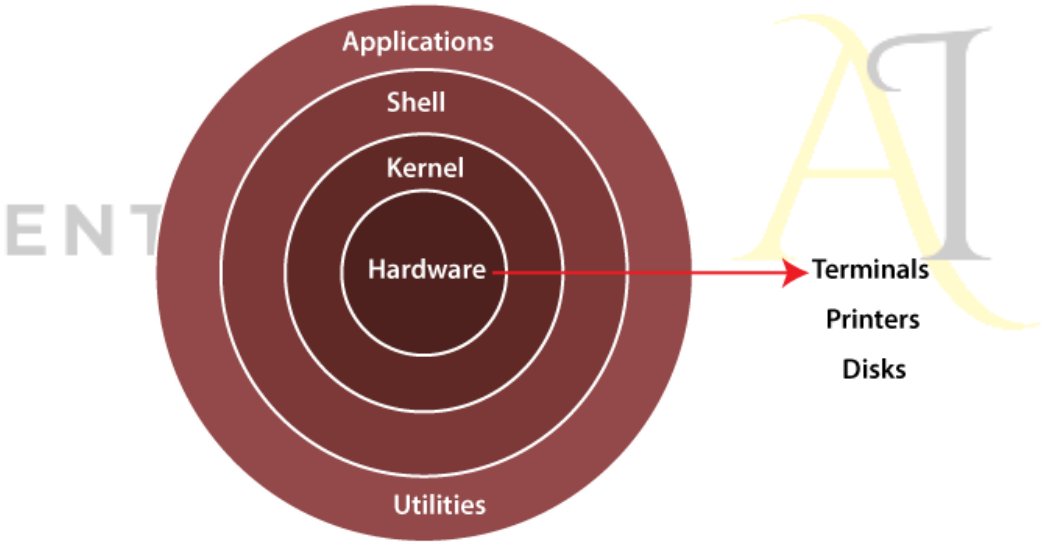

Fig 1.1 Architecture of Linux System

- **Application:** Users interact with the system through varies applications such as office, games, etc. These applications run in outer layer of architecture.
- **Shell:** Shell provides environment to run any application. It provides interface to the user to interact with hardware. We can say that it converts higher level language to lower-level language.
- **Kernel:** Kernel is the program which communicates with the hardware. Combining shell and kernel forms Operating System.
- **Hardware:** All the hardware components such as motherboard, CPU, hard disk, etc. come under this layer.

Chapter 2: Basic Commands

Commands are nothing but executable programs which perform specific task written in it. These executable programs can be called using their name as per provided syntax.

General Syntax

 # <command> -<options> -<arguments> -<multiple arguments>

Commands: To run the command.

Options: To adjust behaviour of the commands.

Arguments: The behaviour, file folder name.

Basic Commands

#tty	This command displays current user terminal number.
#who	Display current logged in all users with their details.
#who am i	Display current user details.
#whoami	Display current user username.
#w	Display all current user details with their time schedule.
#wh <tab> <tab>	To see similar commands or autocomplete cmd.
#clear or ctrl+l	It clears screen but not background data.
#cal	It displays current month of calendar.
#cal <year>	It displays specified year calendar.
#cal <month> <year>	It displays specific months calendar in particular year.
#cal -3	It displays previous, current and next month's calendar of current year.
#cal -j <year>	It displays Julian days of particular year.
#date	It shows current date and time.
#date -s "DD MM YYYY HH:MM:SS"	It set date and time.
#reboot OR #init 6	To restart the system.
#power off OR #init 0	To power off system.
#shutdown	Its shutdown system after 1 min.
#shutdown -h 10	Its shutdown system after 10 min and broadcast message to all users.
#shutdown -c	It cancels shutdown timer and broadcast message to all users.
#shutdown now OR shutdown -h	It immediately shutdown system.
#logout OR exit OR ctrl + d	Logout from currently logged in user.
#hostname OR hostname -s	Shows machine name.

COMMAND TO VIEW SYSTEM INFORMATION

#hostnamectl	Show detailed information about system.
#uname -a	It displays operating system kernel name.
#uname -r	It displays all information regarding system.
#free -h	It displays RAM information in human readable form.
#lsusb	It lists all available USB devices.
#lspci	It lists all PCI devices.
#lscpu	It lists processor information.
#dmidecode	It displays all hardware information.(root)

GETTING HELP FROM COMMANDS

#man <command>	Show manual of mentioned command.
#info <command>	Same as that of man command.
#whatis <command>	Display one line description of manual page. (#mandb command should use to update database manuals.)
#<command> --help	Show short description of manual page. –help is option thus, some command may not support this option.
#which <command>	To know path of command file.

Chapter 3: Editors in Linux

Editors are used to create new files and edit or modify the content inside it. Simply editors are used to read and write data in existing or newly created file. Editors can be classified based on interfaces that they use, i.e. Graphical Editors and Command Line Editors.

Graphical editor:

Graphical editor uses graphical user interface. It is easy to use but consumes more memory than command line editors. In Linux, following are some well-known graphical editors,

1. **gedit**: - It is same as that of notepad in windows. You can open gedit graphically in application menu and using command as **$gedit**.
2. **kedit**: - It is like gedit but contain some advance features. Generally, we must install kedit separately to use it.
3. **Open office**: - Open Office is same as that of MS Office. This Open Office is specially developed for Linux based operating systems.

Command line editor:

Command line editor uses command line interface. These editors are much master and consumes less memory than any graphical editors. Following are some well-known command line editors,

1. **nano**: Nano editor is easy to use since it provides simple features to edit data from files.

   Syntax - **#nano <filename>.<extension>**

2. **pico:**

   Syntax - **#pico <filename>**

3. **vi and vim:** vi (virtual interface) and vim (virtual interface modified) are most used editors. vi and vim both editors are same whereas vim is the advance version of vi editor. Thus, it contains some additional features. These editors work in four different modes,
   a. Insert mode
   b. Ex-mode
   c. Command mode
   d. Visual mode

   Syntax - **#vim <filename>**

Cursor Movement

J	For upward navigation.
K	For downward navigation.
H	For leftward navigation.
L	For rightward navigation.

Command Mode

This is default mode. Press esc to exit from any mode and enter into command mode.

dd	Delete current line
<n>dd	Delete n no. of lines from current line
dw	Delete current word
<n>dw	Delete n no. of words from current word
yy	Copy current line
<n>yy	Copy n no. of lines from current line
yw	Copy current word
<n>yw	Copy n no. of words from current word
cc	Cut current line and enter in insert mode
<n> cc	Cut n no. of lines and enter in insert mode
cw	Cut current word and enter in insert mode
<n> cw	Cut n no. of words from current word and enter in insert mode
p	Paste
s	Remove current character and enter in insert mode
S	Remove current line and enter in insert mode
u	Undo
Ctrl+r	Redo
H	Move cursor to the top of screen
M	Move cursor to the middle of screen
L	Move cursor to the bottom of screen
G	Move cursor at the end of file
gg	Move cursor at the beginning
<n>gg	Move cursor at nth line
/<word>	Search word/string/character
n	Show next search result
N	Show previous search result

Visual Mode

This mode is used for selection

v	Select character by character
V	Select line by line
Ctrl+v	Select block
y,d,c	For copy, delete, and cut selected area

Insert Mode

Following are commands to enter in insert mode.

i	Insert text at current cursor position
I	Insert text at start of the current line
a	Insert text just right of the current character
A	Insert text at end of the current line
o	Insert new line below the current line
O	Insert new line above the current line
r	It replaces single character
R	Replace multiple characters

Ex-mode

Special executing mode

:q	Quit without saving
:q!	Quit without saving forcefully
:w	Save and stay in file
:wq or :x	Save and quit
:wq!	Save and quit forcefully
:set nu	Set line numbers
:<n>	Jump to nth line
:set nonu	Remove line numbers
:/<word>	Highlight word/string/character
:nohl	Remove highlight
:%s/<old>/<new>/g	Find and replace old word with new word
:!<command>	Execute any command on terminal without leaving editor

Chapter 4: Working with Text Files

File Creation:

#touch – touch command is used to create files. Using touch command, multiple files can be created.

Syntax: #touch <filename>

Example: Creating file in current directory

```
[root@server0 /]# touch file1.txt
```

Creating file in desired directory

```
[root@server0 /]# touch /root/file1.txt
```

Creating multiple files at different locations (below example will create two files, one at /root/Desktop and 2nd at /etc directory. You can add more file names along with their path and separate them by space.)

```
[root@server0 /]# touch /root/Desktop/file1.txt /etc/data.mp3
```


Creating multiple files at same location but with different file names (below example will create 3 files. You can add more file names and separate them by coma)

```
[root@server0 /]# touch /root/{data.txt,file.txt,demo.mp3}
```

Creating multiple files having continuous number in their names (below example will create hundred files with name starting from file1 to file100.)

```
[root@server0 /]# touch /root/file{1..100}.txt
```

Directory Creation:

#mkdir – mkdir command creates directories. Creating multiple directories are also possible using mkdir command.

Syntax: #mkdir <option>/<path /directory name>

Example: Creating directory in / directory

```
[root@server0 /]# mkdir /dir1
```

Creating multiple directories

```
[root@server0 /]# mkdir /dir1 /root/Desktop/dir2 /etc/demo
[root@server0 /]# mkdir /root/{demo,data,practice}
[root@server0 /]# mkdir /practical{1..10}
```

Creating parent directory if it does not exist

```
[root@server0 /]# mkdir –p /demo/data/practice
```

File Operations:

READ: - Read operation can be performed to view content of the file. There are five commands which we can use for read operation.

- **cat**: cat command is used to get data of file as output on the terminal. Reading out large file leads to navigate in terminal, which require separate scrolling device (mouse). So, cat command is very useful in reading smaller files with few lines of data in command line.

  ```
  [root@server0 /]# cat /root/anaconda-ks.cfg
  ```

- **more**: more command provides line by line navigation and page by page navigation in downward direction but, upward scrolling not possible.

  ```
  [root@server0 /]# more /root/anaconda-ks.cfg
  ```

- **less**: less command allows navigation keys for scrolling up and down. Thus, it is more useful command than any other four command.

  ```
  [root@server0 /]# less /root/anaconda-ks.cfg
  ```

- **head**: head command show few lines from top of the file. If head command is used without any option, it will show top ten lines by default. –n is used to give count of lines to be shown.

  ```
  [root@server0 /]# head /root/anaconda-ks.cfg
  [root@server0 /]# head –n 5 /root/anaconda-ks.cfg
  ```

- **tail**: tail command shows few lines from bottom of file. If tail command is used without any option, it will show bottom ten lines by default. –n is used to give count of lines to be shown.

  ```
  [root@server0 /]# tail /root/anaconda-ks.cfg
  [root@server0 /]# tail –n 4 /root/anaconda-ks.cfg
  ```

SORT: - Sort command will display result in ascending or descending order. Without option, data will be shown in ascending order.

r	To show output in reverse order
k<n>	To show output of arranged by sorting nth column

```
[root@server0 /]# sort file1.txt
[root@server0 /]# sort –r file1.txt
```

COPY: - Copy operation use to copy files and directories in Linux, from one location to another. It will copy contents of one file to another. If destination file does not exist in given location, then automatically new file will be generated.

f	Forcefully
v	Verbose/View
r	Recursive (To copy directory)
a	Preserve permissions while copying

Copy one file content to another file

```
[root@server0 /]# cp /root/anaconda-ks.cfg ~/Desktop/kickstart.txt
```


Copy file from one location to another

```
[root@server0 /]# cp /root/anaconda-ks.cfg /mnt/  → copy single file
```

Copy directory and multiple files

```
[root@server0 /]# cp -r /etc /root                  → copy directory etc
[root@server0 /]# cp –r /root/* /media              → copy all files from root
[root@server0 /]# cp –rv /abc.txt /xyz.mp3 /media   → copy multiple files
```

MOVE AND RENAME: - Move and rename both operations can be performed using ‘mv’ command. It moves files and directories from one location to another. It is possible to move and rename at the same time.

Move files/directories from one location to another

```
[root@server0 /]# mv /root/anaconda-ks.cfg /mnt/    → move single file
[root@server0 /]# mv /media ~/Desktop/              → move directory
[root@server0 /]# mv /root/* /mnt/                  → move all files
```

Move and rename together

```
[root@server0 /]# mv /root/anaconda-ks.cfg ~/Desktop/kickstart.txt
```

Rename file or directory

```
[root@server0 /]# mv fower flower.txt                    → rename file
[root@server0 /]# mv /root/anaconda-ks.cfg /root/kickstart.cfg
[root@server0 /]# mv /opt /demo                          → rename directory
```

REMOVE: - 'rmdir' command is used to remove empty directory whereas 'rm' command is used to remove files. Directories also can be removed using 'rm' command. (Note: Removing files using rm or rmdir command will delete files permanently and doesn't move in recycle bin or any other place.)

#rmdir – rmdir command only deletes empty directories.
Trying to remove non-empty directory using rmdir. It gives error.

```
[root@server0 ~]# rmdir /Demo
rmdir: failed to remove '/Demo': Directory not empty
```

Removing empty directory using rmdir

```
[root@server0 ~]# rmdir /Demo/data
```


#rm – rm command is used to delete files and directories.

```
[root@server0 ~]# rm /Demo/file10.txt
rm: remove regular file '/Demo/file10.txt'? y
```

Remove files without interaction, (-**f** : forcefully, -**v** : verbosely)

```
[root@server0 ~]# rm -vf /Demo/file1.txt removed '/Demo/file1.txt'
```

Remove all files

```
[root@server0 ~]# rm –f /Demo/*
```

Remove directories, (-r : recursive)

```
[root@server0 ~]# rm –r /Demo
```

REDICTORS: Redirectors are used to write terminal output into file. Output, generated from any command, on terminal can be transferred into existing file. If file does not exist, automatically new file will be created. Following are some redirectors.

Single Redirector (>): Single redirector replace existing data in the file with newly redirected data. It overwrites the contents of existing file.

Double Redirector (>>): Double redirector keeps existing data, and newly redirected data will be added at the end of the file. It appends redirected data in existing file.

Syntax,

<command_to_generate_output> [> or >>] <new/existing_file>

```
[root@server0 ~]# echo “Hello World” > ~/abc.txt
[root@server0 ~]# cat ~/abc.txt Hello World
Hello World

[root@server0 ~]# echo “I am Shantanu” >> ~/abc.txt
[root@server0 ~]# cat ~/abc.txt
Hello World
I am Shantanu
```

Redirectors with cat command:-

Display content of existing file

```
[root@server0 ~]# cat /root/anaconda-ks.cfg
```

Create a new file and provide writing into it

```
[root@server0 ~]# cat > /root/Desktop/flower.txt
Lotus, rose, tulip, lily.          → Press [ctrl+d] to save file

[root@server0 ~]# cat /root/Desktop/flower.txt
Lotus, rose, tulip, lily.
```


Overwrite content of existing file

```
[root@server0 ~]# cat > /root/Desktop/flower.txt
Hello World!
[root@server0 ~]# cat /root/Desktop/flower.txt
Hello World!
```

Add content in existing file, (Append)

```
[root@server0 ~]# cat >> /root/Desktop/flower.txt
I am Shantanu

[root@server0 ~]# cat /root/Desktop/flower.txt
Hello World!
I am Shantanu
```

View content of files

```
[root@server0 ~]# cat /etc/redhat-release        → display OS name and Version
[root@server0 ~]# cat /proc/meminfo              → display ram information
[root@server0 ~]# cat /proc/cpuinfo              → display processor information
```

Copy content of one file to another file

```
[root@server0 ~]# cat /etc/redhat-release  >  /osname.txt
```

Append content of one file to another file

```
[root@server0 ~]# cat /proc/meminfo  >>  /osname.txt
```

File content merging

```
[root@server0 ~]# cat /flower.txt /number.txt /alpha.txt  >  /newfile.txt
```

Execute Multiple Commands:-

Semicolon (;) – Executes two or more commands in same command line argument.

Example,

```
[root@server0 ~]# touch /abc.txt ; mkdir /mydir ; cp –r /mnt /media
```


Pipe (|) – It matches first command output to the second command and execute it.

Example,

```
[root@server0 ~]# dmidecode | less [root@server0 ~]# ls / | wc
```