Unit 1: Introduction to Open Source 3
Open Source Software 3
Principle of Open Source 3
History of Open Source Initiative (OSI) 4
Need and Advantages of Open Source 4
Unit 2: OPEN SOURCE OPERATING SYSTEM: LINUX 5
UNIX Vs LINUX 5
LINUX VS Windows 5
Features and Advantages of Linux 5
Variants of Linux 6
Overview of Linux Architecture 7
Unit 3: FILES, DIRECTORIES AND BASIC COMMANDS IN LINUX 9
Linux Standard Directories 9
Commands for Files and Directory Handling in Linux 10
File Structure and Hierarchy 19
File permission, changing permission and ownership 21
Unit 4: System Administration Basics 25
Boot Process in Linux 25
BIOS 25
MBR 25
GRUB 25
Kernel 25
Init 26
Runlevel Programs 26
Linux Kernel Fundamentals 27
Introduction of Kernel 27
Operations of Kernel 27
Types of Kernel 28
Kernel Space Vs User Space 28
Kernel Space 28
User Space 28
Background Processing 29
Introduction of Process 29
Types of Processes 29
Running a Background Process 30

How to start Linux Process in Background? 30
Unit 8: Open Source Web Programming 31
Basics of web programming using PHP (HTML post and get) 31
PHP File 31
Setting Up PHP 31
PHP Syntax 31
PHP Comments 32
PHP Variables 32
PHP echo and print Statements 32
PHP Data Types 32
String 32
Integer 33
Float 33
Boolean 33
Array 33
Object 33
Null 33
PHP Operators 33
PHP Flow Control 34
PHP Loop 34

Unit 1: Introduction to Open Source

If there is something that can be modiefied and is shareable, then it can be termed as opens source. Open source is designed in such a way that it can be accessed publicly.

Open Source Software

- If there is software with source code and the code can be easily modified, enhanced and inspected, then the software can be called open-source software.
- Source code can't be easily seen by the users who use the software but the programmer can manipulate and change it.
- Programmers with access to source code can enhance and improve the program or software by adding the features in it. Also, bugs can be fixed easily.

Principle of Open Source

- Its open source way which is set of principles. This principle is driven from open source software development.

Most common and important principles of open source

- Transparency
  - Have access to information and necessary materials
  - Anyone can watch
  - Ability to see what's going on
- Collaboration
  - One can freely participate
  - This helps to enhance work of eacj other
  - New possibilities can be seen
- Early and Often Release
  - More collaboration, more perspective
  - Leads to early and often release
- Community / Participation
  - Shared value with common purpose
  - Community goal
- Openness
  - Easy reporting of defects in software
  - Allowing to see those defects and feedbacks to public
  - Public forum for criticism

History of Open Source Initiative (OSI)

- 1983: Richard Stallman launched GNU project [to write OS with free use if source code]
- 1985: Stallman published GNU Manifesto
- 1989: First vesion of GNU General Public License published
- 1991:
- OSI formed in 1998 as educational, advocacy organization
- Term "open source" was coined at session on 3rd February, 1998 in California after Netscape Source code was announced to be released
- OSI was founded on 1998 February, founders were Eric Raymond and Bruce Perence with mission of education and advocacy.
- OCT 1999: OSI published list of approved license

To be continued…

Need and Advantages of Open Source

- Cost of hardware is less as linux and os are portable with higher compressible using less hardware power than other operationg systems
- Quality of software is high and it can be efficiently used
- Dependency in vendor is almost zero
- Management of license is very simple
- Cost Effective: Cost of software is comparatively very less. Licensing fees and other reduces your cost
- Community Support
- Easy to scale
- Flexibility and Agility is high along with higher speed
- Future is open source

▪

Unit 2: OPEN SOURCE OPERATING SYSTEM: LINUX

UNIX Vs LINUX

UNIX	LINUX
Source code is not available to public	Source code is available to general public
Used for servers, workstations and high end computers	Used for personal computer, game development etc
Expensive	Free
High end hardware is required	No high end hardware is required
Uses command line interface	Uses command line interface and GUI
e.g Solaries, HP UNIX etc.	e.g. Ubuntu, Fedora
Can only be used by copyrighters	Can be used publicly
1970	1991
Developed by AT&T	Developed by open source development

LINUX VS Windows

LINUX	WINDOWS
Users have access to source code	Access of source code is not available to users
Highly customizable	Less customization option
GPL License – free to modify and redistribute	Microsoft's License, can't be modified and redistribute by users
More command lines for daily uses	More GUI than command lines
Community Support	Microsoft Support
Users have control on upate, no reboot required	No control for update time, reboot frequently required
More secured	Less secure
Open Source	Closed Source
Free	Cost is needed
No Virus, malware	Vire, malware exists
Low hardware cost	High Hardware cost
Linux Kernel released on 199	Windows first released on 1985
Developed by Linus Torvald	Developed by Microsoft
More stable	Can be easily attacked without user's knowledge

Features and Advantages of Linux

1. Open Source
    - Code easily available
    - Can customize OS
    - Contribute, moify, distribute, enhance
2. Security
    - Less vulnerable
    - Applications needs to be admin authorized

- No antivirus required

3. Free
   - Free to use
   - No license required
   - GNU GPL License
4. Lightweight
   - Requirement to run linux are low end
   - 128 MB RAM / disk space device can also have linux
5. Stable
   - More stable than others
   - No reboot required ti maintain performance level
   - Rare hangup
   - Larger uptime
6. Performance
   - High level of performance
   - Can handle large number of user simultaneously
7. Flexibility
   - Very flexible
   - Used for desktop, embedded system, servers
8. Updates
   - Are in user control
   - Faster than others
9. Distribution
   - Large number of distros are available
   - E.g Ubuntu, Fedora
10. GUI
    - Is command line
    - But GUI is also interactive
11. Community Support
    - Large community
    - Various forums for support
12. Privacy

Variants of Linux

- There are hundres if variants of linux
- **For you, which is best linux?**
- To answer this , it depends on factors like
  - Skills
  - Area of focus
  - Support Required
  - Hardware you are using
  - Softwares you need to use
- Here are top Linux distributions
  - Ubuntu
    - Most popular distro
    - Announced in 2004
    - Uses Debian

- Have app like firefox, open office
- Have remixes like Kubuntu, Xubuntu, Lubuntu etc
- For beginners

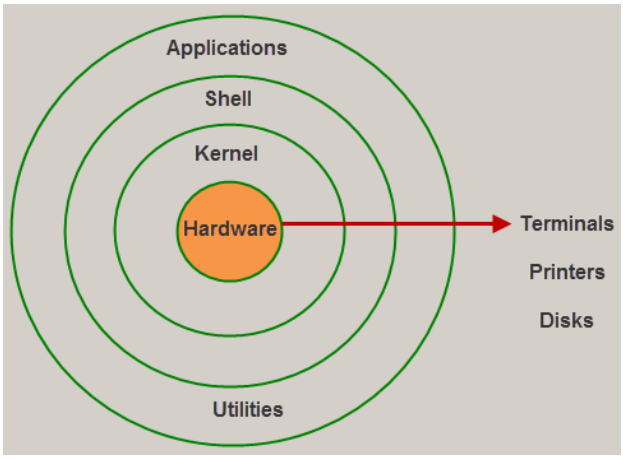
- Fedora
  - Free version of Red Hat
  - Strong on enterprise feature
  - Security is excellent
- Linux Mint
  - Ubuntu based
  - Released on 2006
  - Variety of graphical tools
  - Good for beginning users
- openSUSE
  - medium on difficulty level
  - targeted for open source developers and system admins
- PCLinuxOS
  - Uses KDE environment
  - Light weight version of mandriva
  - Supports graphic driver, browser plugins
- Debian
  - 1993
  - One of the well tested
  - Bug free distro
  - Suitable for linux experienced
  - Slow release cycle
- CentOS
  - Based on RHEL
- Kali Linux
  - Debian based
  - Designed fr penetration testing
  - Used for digital forensics
  - Meant for nmap, metasploit framework, maltego, aircrack ng etc
  - For cybersecurity experts
- Mandriva
- Gentoo
- Arch Linux
  - Lightweight
  - Flexible
  - For advanced users

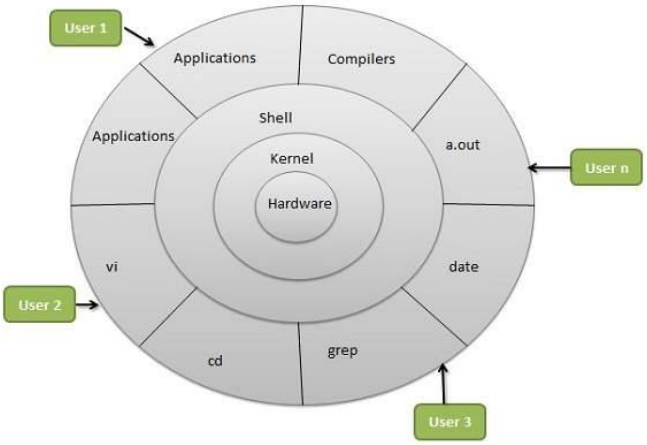
Overview of Linux Architecture

- Hardware
  - Consists all peripheral devices
  - Physical devices
  - E.g RAM, HDD, Processer, motherboard etc
- Kernel

- Core Component
- Directly interacts with hardware

- Shell
  - Interface to kernel
  - Hides complexity of Kernel's function
  - Takes command / input from users
  - Executes function of kernels
- Utilities
  - Provides functionalities of OS to users
  - Utilities programs runs on shells
  - Ex. Browser, media palyer etc





Unit 3: FILES, DIRECTORIES AND BASIC COMMANDS IN LINUX

Linux Standard Directories

- All files and directories are under root directory (/)
- Some standard directories are

  - /root
    - Every file and directory starts from root
    - Root user can write on this directory
    - /root is home directory of root user
    - It is not same as /
  - /bin
    - All executable binary files
    - Files required for booting, repair are here
    - Common binaries
    - Common linux command in single user mode
    - Eg cat, ls, cp, tar, history etc
  - /boot
    - Static file of boot loader
    - Holds file required during boot process
    - Linux kernel
  - /dev
    - Contains device files
    - Hardware device oriented
    - Usb, cdrom etc attached with system
    - Eg dev
  - /etc
    - Contains configuration files of applications
    - All configuration files that run in the system
    - Also have start abd shutdown shell scripts to start/stop program
  - /home
    - User's home directory
    - Contains directory like desktop, downloads, documents
  - /lib
    - Contains shared library image to boot system
    - Also contains kernel modules
  - /lost /found
    - Directory installed during linux installation
    - For recovery purpose
    - Recovery of file due to unexpected shut down
  - /media
    - Directory for removable devices
    - Temporary mount directory
    - E.g /media/cdrom , /media/floppy
  - /mnt
    - Temporary mount directory

- System admins can mount file systems
- /opt
  - Optional apps
  - Third party apps
- /proc
  - Virtual file
  - Information about running process with process id i.e pid
- /run
- /sbin
  - Binary executable programs
- /srv
  - Server specific files
  - Service related files
- /sys
  - Virtual file system
  - Devices connected to system
- /tmp
  - Stores temporary files
- /usr
- /var

Commands for Files and Directory Handling in Linux

- Cd
  - To change directory
  - E.g cd / [we are moving to root directory]

```
┌──(root㉿ 9107f15960be)-[/]
└─# mkdir abhash
┌──(root㉿ 9107f15960be)-[/]
└─# cd abhash
┌──(root㉿ 9107f15960be)-[/abhash]
└─# 
```

- Cd..
  - Takes one level up of directory
  - E.g cd..

```
┌──(root㉿ 9107f15960be)-[/abhash]
└─# cd ..
┌──(root㉿ 9107f15960be)-[/]
└─# 
```

- Ls
  - To list the items of directory

```
└─# ls
avs_file.txt
┌──(root㉿ 9107f15960be)-[/avs]
└─# 
```

- Ls –la
  - List content of directory including hidden files
  - E.g ls-la

```
└─# ls -la
total 8
drwxr-xr-x 2 root root 4096 May 11 12:56 .
drwxr-xr-x 1 root root 4096 May 11 12:51 ..
-rw-r--r-- 1 root root    0 May 11 12:56 avs_file.txt
┌──(root㉿ 9107f15960be)-[/avs]
└─#
```

- Cp
  - Copies the content from file a to file b
  - If file b is not there, file b is created
  - It it exists, b is overwritten
  - E.g cp ict.txt byte.txt [we are copying content from ict.txt to byte.txt]
- Cp –r
  - Copies content of directory
  - If second directory is not found, second directory is created
  - If it exists, content is overwritten
- Mv
  - To rename files and directories
  - To move files and directories

```
┌──(root㉿ 9107f15960be)-[/]
└─# mv abhash avs
┌──(root㉿ 9107f15960be)-[/]
└─# cd avs
┌──(root㉿ 9107f15960be)-[/avs]
└─#
```

- Cat
  - Prints content of file
  - E.g cat filename.txt
- Head
  - Prints first ten line of file
  - E.g head ict.txt
- Rm
  - To delete file
- Rm –f
  - Forecefully delets file
- Mkdir
  - Create a new directory if no exists

```
┌──(root㉿ 9107f15960be)-[/]
└─# mkdir abhash
┌──(root㉿ 9107f15960be)-[/]
└─# cd abhash
┌──(root㉿ 9107f15960be)-[/abhash]
└─# 
```

- Mkdir –p
  - Creates nested directory
  - E.g mkdir /p example/ict/byte
- Rmdir
  - Remove/delete directory

```
┌──(root㉿ 9107f15960be)-[/avs]
└─# mkdir abhash
┌──(root㉿ 9107f15960be)-[/avs]
└─# cd abhash
┌──(root㉿ 9107f15960be)-[/avs/abhash]
└─# cd..
bash: cd..: command not found
┌──(root㉿ 9107f15960be)-[/avs/abhash]
└─# cd ..
┌──(root㉿ 9107f15960be)-[/avs]
└─# rmdir abhash
┌──(root㉿ 9107f15960be)-[/avs]
└─# cd abhash
bash: cd: abhash: No such file or directory
┌──(root㉿ 9107f15960be)-[/avs]
└─#
```

- Pwd
  - Displays currently working directory

```
┌──(root㉿ 9107f15960be)-[/abhash]
└─# pwd
/abhash
┌──(root㉿ 9107f15960be)-[/abhash]
└─#
```

- File
  - To determine type of file
  - E.g file filename
- More
  - View text file in command prompt one screen at a time if file is large

```
└─# more avs_file.txt
text in abhash ko file
text in abhash ko file file of abhash 1 2 3 4 5 5 7 8 9 6 56 54 4 f erdgdfhg
┌──(root㉿ 9107f15960be)-[/avs]
└─#
```

- Less

- Read content of text file one page per time

❖ Help

- Shows help menu

```
┌──(root㉿ 9107f15960be)-[/avs]
└─# help
GNU bash, version 5.1.4(1)-release (x86_64-pc-linux-gnu)
These shell commands are defined internally.  Type `help' to see this list.
Type `help name' to find out more about the function `name'.
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

 job_spec [&]                                    history [-c] [-d offset] [n] or history -anrw [f>
 (( expression ))                                if COMMANDS; then COMMANDS; [ elif COMMANDS; the>
 . filename [arguments]                          jobs [-lnprs] [jobspec ...] or jobs -x command [>
 :                                               kill [-s sigspec | -n signum | -sigspec] pid | j>
 [ arg... ]                                      let arg [arg ...]
 [[ expression ]]                                local [option] name[=value] ...
 alias [-p] [name[=value] ... ]                  logout [n]
 bg [job_spec ...]                               mapfile [-d delim] [-n count] [-O origin] [-s co>
 bind [-lpsvPSVX] [-m keymap] [-f filename] [-q na>  popd [-n] [+N | -N]
 break [n]                                       printf [-v var] format [arguments]
 builtin [shell-builtin [arg ...]]               pushd [-n] [+N | -N | dir]
 caller [expr]                                   pwd [-LP]
 case WORD in [PATTERN [| PATTERN]...) COMMANDS ;;>  read [-ers] [-a array] [-d delim] [-i text] [-n >
 cd [-L|[-P [-e]] [-@]] [dir]                    readarray [-d delim] [-n count] [-O origin] [-s >
 command [-pVv] command [arg ...]                readonly [-aAf] [name[=value] ...] or readonly ->
 compgen [-abcdefgjksuv] [-o option] [-A action] [>  return [n]
```

-

❖ Touch

- To create a file
- Touch abhash.txt

```
┌──(root㉿ 9107f15960be)-[/avs]
└─# touch avs_file.txt
┌──(root㉿ 9107f15960be)-[/avs]
└─# 
```

❖ echo

To move data (text) into a file

```
┌──(root㉿ 9107f15960be)-[/avs]
└─# echo this is maile lekheko text file_abhash.txt
this is maile lekheko text file_abhash.txt
┌──(root㉿ 9107f15960be)-[/avs]
└─# 
```

❖ sudo

Run as superusers

❖ df

- see available disk space

```
┌──(root㉿ 9107f15960be)-[/avs]
└─# df
Filesystem     1K-blocks   Used Available Use% Mounted on
overlay        263174212 961428 248774628   1% /
tmpfs              65536      0     65536   0% /dev
tmpfs            6480980      0   6480980   0% /sys/fs/cgroup
shm                65536      0     65536   0% /dev/shm
/dev/sdc       263174212 961428 248774628   1% /etc/hosts
tmpfs            6480980      0   6480980   0% /proc/acpi
tmpfs            6480980      0   6480980   0% /sys/firmware
```

- ❖ Hostname
  - ➢ To know name in host or networ

```
└─# hostname
9107f15960be
┌──(root㉿ 9107f15960be)-[/avs]
└─#
```

- ❖ cat
  - ➢ see content of the file

```
└─# echo text in abhash ko file >> avs_file.txt
┌──(root㉿ 9107f15960be)-[/avs]
└─# cat avs_file.txt
text in abhash ko file
┌──(root㉿ 9107f15960be)-[/avs]
└─#
```

- ❖ head
  - ➢ view first line of any text

```
└─# head avs_file.txt
text in abhash ko file
text in abhash ko file file of abhash 1 2 3 4 5 5 7 8 9 6 56 54 4 f erdgdfhg
┌──(root㉿ 9107f15960be)-[/avs]
└─#
```

- ❖ tail
  - ➢ shows 10 last line of text

```
┌──(root㉿ 9107f15960be)-[/avs]
└─# tail avs_file.txt
text in abhash ko file
text in abhash ko file file of abhash 1 2 3 4 5 5 7 8 9 6 56 54 4 f erdgdfhg
┌──(root㉿ 9107f15960be)-[/avs]
└─#
```

- ❖ cat
  - ➢ see content of the file
- ❖ history

- shows the previously run command

```
┌──(root㉿ 9107f15960be)-[/avs]
└─# history
    1  cd..
    2  mkdir abhash
    3  cd abhash
    4  cd..
    5  ls
    6  cd..
    7  ls
    8  pwd
    9  cd..
   10  cd ..
   11  mv abhash avs
   12  cd avs
   13  ls
   14  pwd
   15  ls
   16  cd abhash
   17  mkdir abhash
   18  cd abhash
   19  cd..
   20  cd ..
   21  rmdir abhash
   22  cd abhash
   23  touch avs_file.txt
   24  man
   25  help
```

-

❖ useradd

- add new user

```
└─# useradd abhash
┌──(root㉿ 9107f15960be)-[/avs]
└─# 
```

-

❖ userdel

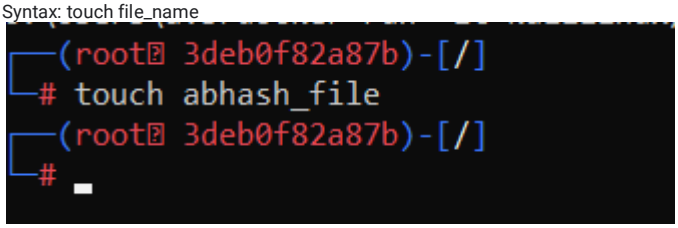
- to delete user

```
┌──(root㉿ 9107f15960be)-[/avs]
└─# useradd abhash
┌──(root㉿ 9107f15960be)-[/avs]
└─# userdel abhash
┌──(root㉿ 9107f15960be)-[/avs]
└─# userdel test
userdel: user 'test' does not exist
┌──(root㉿ 9107f15960be)-[/avs]
└─# 
```

-

❖ **Touch**

- Creates, changes and modify timestamp of file
- Creates file with no content
- Syntax: touch file_name



- The file can be seen using ls command

- To create multiple file syntax: touch filename1 filename2 filename3
- Options in touch command
  - Touch –a
    - Changes the access time
    - touch-a command is used to change or update the last access/modification time
    - Syntax: touch –a filename
  - touch – c
    - checks if file is created or not
    - Syntax: touch – c filename
  - Touch –c –d
    - Used to update access / modification time
    - Syntax: touch –c-d filename
  - Touch –m
    - Changes the modification time only
    - Updates last modification time
    - Syntax : touch –m filename
  - Touch r
    - Used to use timestamp of another file
    - Syntax: touch –r secondfile firstfile
  - Touch –t
    - Creates file with specified timestamp
    - Syntax: touch – t YYMMDDHHMM filename

```
┌──(root㉿ 3deb0f82a87b)-[/]
└─# touch -a abhash_file
┌──(root㉿ 3deb0f82a87b)-[/]
└─# touch -c
touch: missing file operand
Try 'touch --help' for more information.
┌──(root㉿ 3deb0f82a87b)-[/]
└─# touch -c abhash_filename
┌──(root㉿ 3deb0f82a87b)-[/]
└─# touch -m abhash_file
┌──(root㉿ 3deb0f82a87b)-[/]
└─# touch -t 2108061843
touch: missing file operand
Try 'touch --help' for more information.
┌──(root㉿ 3deb0f82a87b)-[/]
└─# touch -t 2108061843 abhash_file
┌──(root㉿ 3deb0f82a87b)-[/]
└─#
```

- **Update**
  - Updates OS
  - Syntax: apt-get update

```
┌──(root㉿ 3deb0f82a87b)-[/]
└─# apt-get update
Get:1 http://kali.cs.nctu.edu.tw/kali kali-rolling InRelease [30.5 kB]
Get:2 http://kali.cs.nctu.edu.tw/kali kali-rolling/contrib amd64 Packages [108 kB]
Get:3 http://kali.cs.nctu.edu.tw/kali kali-rolling/non-free amd64 Packages [199 kB]
Get:4 http://kali.cs.nctu.edu.tw/kali kali-rolling/main amd64 Packages [17.7 MB]
21% [4 Packages 561 kB/17.7 MB 3%]
```

- **Whoami**
  - Shows your username
- **Finger**

```
┌──(root㉿ 3deb0f82a87b)-[/]
└─# apt-get install finger
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  finger
0 upgraded, 1 newly installed, 0 to remove and 12 not upgraded.
Need to get 19.8 kB of archives.
After this operation, 50.2 kB of additional disk space will be used.
Get:1 http://kali.cs.nctu.edu.tw/kali kali-rolling/main amd64 finger amd64 0.17-17 [19.8 kB]
Fetched 19.8 kB in 2s (11.7 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package finger.
(Reading database ... 6756 files and directories currently installed.)
Preparing to unpack .../finger_0.17-17_amd64.deb ...
Unpacking finger (0.17-17) ...
Setting up finger (0.17-17) ...
┌──(root㉿ 3deb0f82a87b)-[/]
└─#
```

  - Gives details of all users logged in
  - Commonly used by system admin
  - It provides details like
    - Login name

- User name
- Idle time
- Login time
- Email address etc
- Similar to pinky tool
- **Syntax: finger**
- To install finger
  - Debian / Ubuntu: $sudo apt-get install finger
  - CentOS / RedHat : $sudo yum install finger
  - Fedora: $sudo dnf install finger
  -
  -

```
┌──(root㉿ 3deb0f82a87b)-[/]
└─# finger
No one logged on.
┌──(root㉿ 3deb0f82a87b)-[/]
└─#
```

-
-
- Example
  - $finger abhash
  - Here, abhash is username

```
┌──(root㉿ 3deb0f82a87b)-[/]
└─# finger root
Login: root                             Name: root
Directory: /root                        Shell: /bin/bash
Never logged in.
No mail.
No Plan.
┌──(root㉿ 3deb0f82a87b)-[/]
└─#
```

❖ **Passwd**
- Changes the user account password
- Root users can change password of any
- Normal user can change password of themselves
- Syntax: passwd

❖

```
┌──(root㉿ 3deb0f82a87b)-[/]
└─# passwd
New password:
Retype new password:
passwd: password updated successfully
┌──(root㉿ 3deb0f82a87b)-[/]
└─#
```

- **Passwd –d :** deletes user password
- **Passwd – e:** expires the password, force to change the password in next login
- **Passwd – h :** password help

❖ **Date**
- displays system date and time
- -u : displays time in GMT
- Date –date=”5 years ago”
- Date- - date=”5 sec ago”

```
└─# date
Tue Jun  8 13:29:12 UTC 2021
┌──(root㉿ 3deb0f82a87b)-[/]
└─# date -u
Tue Jun  8 13:29:15 UTC 2021
┌──(root㉿ 3deb0f82a87b)-[/]
└─# date --date="02/02/2000"
Wed Feb  2 00:00:00 UTC 2000
```

- ➢
- ❖ **Cut**
- ❖ **Sort**
- ❖ **grep**

- ❖ docker –version
- ❖ docker run -it kalilinux/kali-rolling

File Structure and Hierarchy

- File system is a collection of files in a structure on disk or partition
- When memory is segmented and it contains data, it is a partion
- In normal, every partition has file system.
- To store the data systematically, we need a location. Folowing are reasons why we should have file system
    - RAM is volatile, where computer stores data primarily
    - Storage of data in hard drive is preferred, which is non volatile memory.

Linux File System

- It is built in layer of Linux OS, which is used to handle data, and manage the storage
- Arranges file in disk storage.
- File name, size, creation date etc are managed here

Linux File system contains

- Root directory (/)
- Specific storage format (EXT, XFS etc)
- Partition (Logical Volume) with particular file system

Structure of Linux File System

- Has hierarchical file system, containing root and its subdirectories
- From root director, all the directories can be accessed easily.
- File system is designed to manage and provide space for non volatile storage.
- All file system has namespace – naming and organizational methodology. It gives process for naming, file name length etc
- Logical structure of file on memory is also defined here.

- After defining namespace, meta data is also described.

Types of Linux File System

After linux OS is installed, there are file systems such as

- Ext
- Ext2
- Ext 3
- Ext 4
- JFS
- ReiserFS
- XFS
- Btrfs
- Swap

Ext File System

- Extended File System.
- Primarily developed for MINIX OS
- Older one, no useful at this time

Ext2

- First file systm in linux which allows management of terabytes of data

Ext 3

- Developed and upgraded through ext 2 which contains backward compatibility
- Disadvantage: Doesn't support servers and file recovery

Ext 4

- Fastest of all Ext
- Default file system in Linux
- Compatible for SSD

JFS

- Journal Filed System
- For AIX Unix, IBM developed JFS
- Can be used as replacement for Ext 4
- If power of CPU is limited, it is handy file system
- More stable with less resource

ReiserFS File System

- Alternative of Ext 3
- Has advanced features and improved performance
- Used to be default file system in SUSE Linux but due to some policy changes, SUSE used Ext 3
- Dynamically supports file extension

XFS File System

- Higher speed JFS
- Parallel I/O processing is supported
- NASA uses XFS

Btrfs File System

- B tree Fi;e System
- For system repair, fault tolerance, fun administration, extensive storage configuration, Btrfs file system is used.
- Not suitable for production system
- Copy – on – write file system
- Announced by Oracle in 2007
- Published under GNU GPL

Swap File System

- Memory paging in Linux is done by swap file system during system hibernation
- If system never goes hibernation, swap space should be equal to its RAM size
-

File permission, changing permission and ownership

For security purpose, LINUX divides authorization in

- Ownership
- Permission

Ownwership

- In linux, every file and directory is assigned with 3 tyoes of owner
    - User
        - Owner of the file
        - Who creates a file is owner by default
        - Its owner of the file
    - Group
        - Has multiple users
        - All users in a group has same permission to access the file
        - If you assigned permission to group, no one else can read except group members
    - Other
        - Other users other than users and group
        - Everybody else
        - This user hasn't created file and doesn't belong to any group

Permission

- In linux, every file can be assigned with 3 permissions
    - Read
        - Has authority to open and read file
        - Content of directory can be listed
    - Write
        - Access to modify content
        - Adding, removal of content

- File can be renamed

- Execute
  - .exe in windows, which can run
  - To run program, execute permission is needed

Examples:

```
┌──(root㉿ 5c12b5793026)-[/]
└─# ls -l
total 48
lrwxrwxrwx   1 root root    7 Apr 25 04:09 bin -> usr/bin
drwxr-xr-x   2 root root 4096 Feb 16 10:16 boot
drwxr-xr-x   5 root root  360 Jun  1 09:51 dev
drwxr-xr-x   1 root root 4096 Jun  1 09:51 etc
drwxr-xr-x   2 root root 4096 Feb 16 10:16 home
lrwxrwxrwx   1 root root    7 Apr 25 04:09 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Apr 25 04:09 lib32 -> usr/lib32
lrwxrwxrwx   1 root root    9 Apr 25 04:09 lib64 -> usr/lib64
lrwxrwxrwx   1 root root   10 Apr 25 04:09 libx32 -> usr/libx32
drwxr-xr-x   2 root root 4096 Apr 25 04:09 media
drwxr-xr-x   2 root root 4096 Apr 25 04:09 mnt
drwxr-xr-x   2 root root 4096 Apr 25 04:09 opt
dr-xr-xr-x 214 root root    0 Jun  1 09:51 proc
drwx------   2 root root 4096 Apr 25 04:10 root
drwxr-xr-x   3 root root 4096 Apr 25 04:09 run
lrwxrwxrwx   1 root root    8 Apr 25 04:09 sbin -> usr/sbin
drwxr-xr-x   2 root root 4096 Apr 25 04:09 srv
dr-xr-xr-x  11 root root    0 Jun  1 09:51 sys
drwxrwxrwt   2 root root 4096 Apr 25 04:10 tmp
drwxr-xr-x  14 root root 4096 Apr 25 04:09 usr
drwxr-xr-x  11 root root 4096 Apr 25 04:09 var
```

Is –l on linux terminal gives

Here, rwxrwxrwx code tells about the permission that are given owner, group and world

Character means,

r = read

w = write

x = execute

-= no permission

Let us see

rw-

which means for the user in file / folder can

- Read file
- Write or edit the file
- Can't execute as permission is set –

First part is for user, second for group and third for world.

Changing file/directory permission

- We can use chmod command
- Chmod stands for changemode
- Permission (read, write, execute) can be set with this command
- Syntax: chmod permission filename

Chmod command can be used in two ways

1. **Absolute mode**

- Here, permission are represented as three digit octal number

Number	Permission	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx

- Example : chmod 777 filename
    - 777 represents read write and execute permission for user, groud and others all
- Chmod 707
    - 707 represents user have read write and execute permission, group has no permission and others have read write and execute permisso.

2. **Symbolic Mode**

- In absolute, we change permission for all 3 owners
- In symbolic, permission can be modified for specific owner
- Mathematical symbol is used

Operator	Description

+	Adds a permission to a file or directory
-	Removes the permission
=	Sets the permission and overrides the permissions set earlier.

Users are represented as

u=user

g=group

o=other

a=all

Example

Chmod o=rwx examplefile

This is setting permission tp others

Chmod g-r test

This is removing read permission for group from file test

Changing ownership and group

- To changeownership, chown command is used
- **Chown user**

If you want to change the user and group for a file

Chown user:group filename

If you want to change group owner only

Chgroup group-name filename

Chgrp stands for change group

Unit 4: System Administration Basics

Boot Process in Linux

- We can distinct boot process in linux in 6 different stages
- These includes
    - BIOS
    - MBR
    - GRUB
    - Kernel
    - Init
    - Run Level Programs

BIOS

a. Acronym for Basic Input Output System
b. BIOS loads and executes MBR boot loader
c. When computer is started, integrity test is carried put by BIOS. It does integrity test of HDD or SSD
d. Then, searches, loads and executes boot loader program. Boot loader programs are on MBR
e. MBR sometimes can be CD ROM, pen drive etc which have live installation of linux.
f. BIOS provides control of system to boot loader program after boot loader program is detected.
g. BIOS loads and executes MBR boot loader.

MBR

- Acronym of Master Boot Record
- Located in /dev/sda or /dev/hda, according to hardware. This means MBR is in 1st sector of bootable disk
- Has information about GRUB. (In old systems – LILO)
- Has size less than 512 bytes, with components
    - Primary boot loader info (1st 446 bytes)
    - Partition table info (64 bytes)
    - MBR validation check (last 2 bytes)
- MBR loads and executes GRUB boot loader

GRUB

- GRUB is Grand Unified Bootloader
- You can choose one if you have multiple kernel images installed in your system, else default will load
- GRUB displays in screen and waits few seconds where you can choose the kernel, else default kernel image is loaded.
- GRUB configuration file can be found in /boot/grub/grub.conf
- It contains kernel and initrd image
- GRUB loads kernel and initrd image
- Initrd is Initial RAM Disk

Kernel

- Kernel is core of OS
- Complete control over anything is there with Kernel

- Executes /sbin/init program
- Init was the first program to be executed by Kernel which contains which has process id (PID)
- Kernel uses initrd as temporary root file system until the time when Kernel is booted and real root file is mounted. Necessary drivers are also there with initrd. These drivers help to access hardwares and harddrive partitions.

Init

- System executes run level programme
- Available run level are
    - 0 : Halt
    - 1 : Single User Mode
    - 2 : Multiuser with no NFS
    - 3 : Full Multiuser Mode
    - 4 : Unused
    - 5 : X11
    - 6 : Reboot
- Identifies default initlevel and loads required programs [initlevel is obtained from /etc/inittab)
- Identifies default run level (By executing "grep initdefault/etc/inittab" on system)
- Most of the time 3 or 5 is set as default run level

Runlevel Programs

- Above mentioned runlevel have their own directories
    - Run level 0 – /etc/rc0.d/
    - Run level 1 – /etc/rc1.d/
    - Run level 2 – /etc/rc2.d/
    - Run level 3 – /etc/rc3.d/
    - Run level 4 – /etc/rc4.d/
    - Run level 5 – /etc/rc5.d/
    - Run level 6 – /etc/rc6.d/

- In the /etc/rc.d/rc*.d/ directories, we can see program starting wuth S and K

- S : Programmes start with s are needed while system startup

- K : Programmes start with K are needed while system shutdown

- Startup programs are executed while system start ups amd while shutdown, there is kill program

BIOS	**Basic Input / Output System | Executes MBR**
MBR	Master Boot Record | Executes GRUB
GRUB	Grand Unified Boot Loader | Executes Kernel
Kernel	Executes /sbin/init
Init	Executes Run Level PRograms
Run Level	Executed From /etc/rc/rc*.d/

Linux Kernel Fundamentals

Introduction of Kernel

- Kernel is core of OS representing core aspect of Linux distribution for computers and servers
- Architecture of Kernel is monolithic and OS operates in Kernel space.
- Complete control over anything is there with Kernel
- Linux Kernel is a layer between hardware of device and software.
- Kernel is responsible for
    - For application execution, process is managed by Kernel
    - Memory Management
    - I/O Management
    - System call control
    - With help of device drivers, Kernel has role in device management.

Operations of Kernel

- As Kernel controls all the programs in system, sometimes Kernel is considered is heart of the operating system.
- When the device starts, initialization process starts where functions such as checking of memory is performed.
- Memory allocation is performed and environment is created where application runs with no disturbances.
- As Kernel works as service provider, for accomplishing multiple tasks like requesting the usage of disk, network card etc, program can request Kernel.
- For CPU to enable multitasking, Kernel can set interrupts.
- By not letting faulty programs to enter operational functions of others, computational environment is protected by Kernel.

- Unauthorized programs are stopped by Kernel at entrance itself by not allowing memory space.
- Kernel can also limits the CPU time consumed by unauthorized programmes.

Types of Kernel

Monolithic Kernel

- Contains many drivers.
- Creates communication interface between h/w and s/w if device
- Consist various modules which can be dynamically loaded and unloaded.
- This architecture expands the capacities of OS and allows easy extension to Kernel
- Maintenance of Kernel is easy as it allows concern module to load and unload when there is need of bug fixation
-

Microkernel

- Executes basic functionality without interruptions
- Addresses the issue of ever growing size of Kernel code (monolithic failed in this)
- To run in user space, some basic services are allowed. Services are protocol stack, device driver management, file system etc.
- Capability of OS can be enhanced with minimum code, improved security and ensuring stability.
- All the basic OS services are available to program via IPC – Interprocess Communication
- Allows direct interaction between device drivers and hardware

Hybrid Kernel

- Combines monolithic and microkernel
- Can decide what it wants to run in user and supervisor mode.
- Device drivers, filesystens I/O etc run in user mode
- In supervisor mode, server calls and IPC are placed.
-

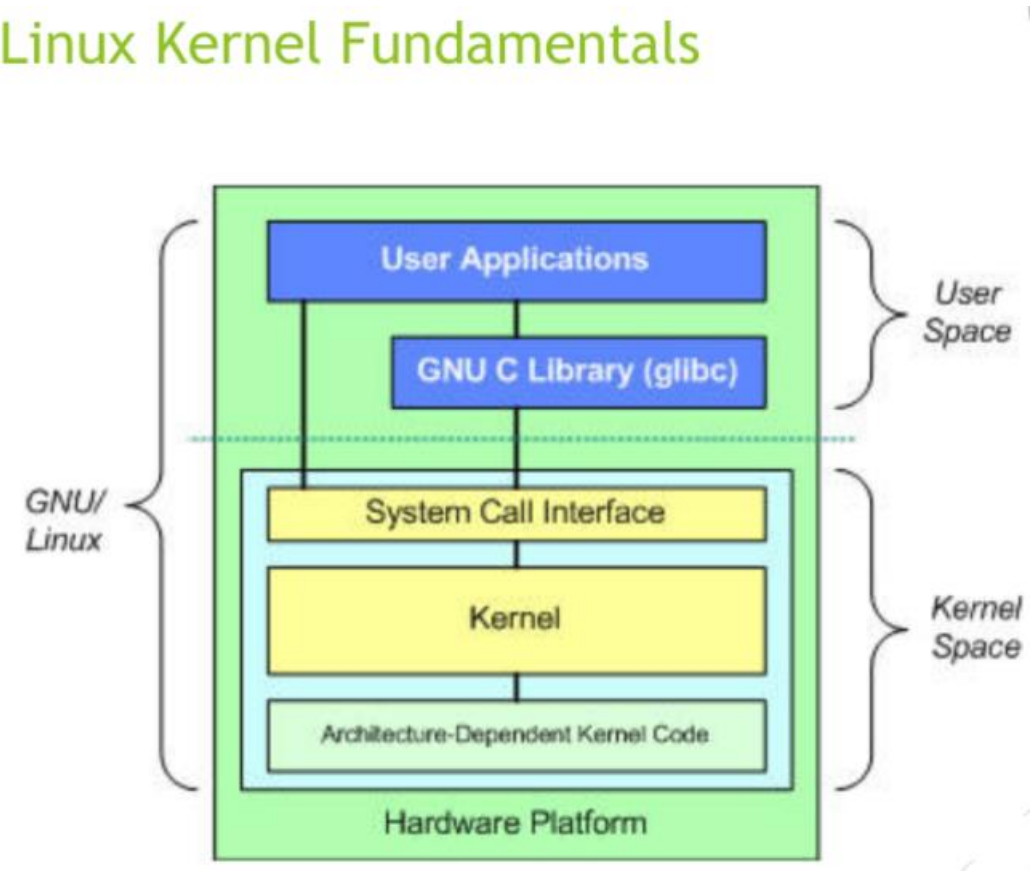
Kernel Space Vs User Space

Kernel Space

- Found in elevated state. Kernel space provides full access of hardware devices
- It protects memory space
- Memory space and user space is togetherly called Kernel space.
- In environment of Kernel space, core access to system service and hardware are maintained.
- This is provided as sercice to rest of the system.

User Space

- It's a code that run outside OS Kernel Environment
- Userspace is various applications/programs/libraries which are used by OS to connect with Kernel.



Background Processing

Introduction of Process

- Process is an instance of program. A new process is started when you gibe any command to Linux machine.
- For a single program, there can be multiple processes.

Types of Processes

Foreground Processes

- Run on the screen
- Input from user is needed
- Example: Office Programs

Background Processes

- Run on background
- Usually, input from user is not needed
- Example: Antivirus

Running a Background Process

- If a program / process is started from terminal, we can't work on terminal up to the time when program is up and running.
- Data intensive tasks can have lot of processing time which needs a lot of time to be completed. And holding terminal for such long time is not desirable
- Also, if terminal is closed, process along with its child can also be terminated.
- For avoiding this, you need to send the program to background and then only, terminal gets available to you.

How to start Linux Process in Background?

- If process is in executin, press **ctrl+z** and enter **bg** command. This will execute process in background as a job.
- All background jobs can be seen by typing **jobs** command
- To tun the process on foreground, we use command "fg"
- Syntax: fg jobname

Some Commands:

- "bg" to esecute process in background
- "fg" to run process on foreground
- "top" tells user about all the running processes
- "ps" command is use to see all the processes running under a user. E.g ps abhash [This is similar to task manager in windows]
- "nice"
    - Nice commands prioritize the process in linux as per your requirements
    - This is niceness in linux , with value in range -20 to 19. Lower the nicensess value, higher the priority of the task.
    - Default value is 0
    - Syntax: nice-n "nice value" process-name
    - Ex: $ nice -9 19 bann
    - If there is process already running, the value can be renice. Syntax: renice "nice value" –p "PID"
- "df" reports free disk space of file sysem
- "free" command shows free and used memory (RAM)

Unit 8: Open Source Web Programming

Basics of web programming using PHP (HTML post and get)

- Hypertext PreProcessor: PHP is open source scripting language executing on server side
- It is free to download and use
- It is the core of WordPress – blogging system

PHP File

- PHP file can have HTML, CSS, JS, PHP cpde, texts
- PHP is server side and result is executed on browser
- .php is the extension of PHP files

Setting Up PHP

To run PHP you need

- Web Server
- PHP
- Database like MySQL

PHP Syntax

In .php file, php script can be written at any place

- PHP starts with <?php
- Ends with ?>
- PHP statements ends with semicolon ;
- PHP keywords are not case sensitive
- Variables are case sensitive e.g $color, $COLOR, $ ColoR are different variables

I.e

```
<?php
//codes
?>
```

Example

```
<html>
<head>
<title>PHP Practice </title>
</head>
<body>
<h1> Heading </h1>
<? Php
Echo “Hello PHP”;
```

```
?>
</body>
</html>
```

PHP Comments

Single Line Comment

- Starts with // or
- Starts with #

Multi Line Comment

- Enclosed within /* and */

PHP Variables

- Variable starts with $ sign and variable name
- Example $name=”Abhash”; $x=5;
- When text is assigned in variable, “..” is used
- Variables are use to store data
- PHP variables are case sensitive
- “echo” statement is used for output
- Example

```
<?
$x=11;
$y=12
$sum=$x+$y;
Echo $sum;
?>
```

- PHP is loosely typed language (we don’t need to assign data type)
- PGP Variable have three scope
    - Local [declared within function]
    - Global [declared outside function]
    - Static

PHP echo and print Statements

- both used to output data
- Echo has no return value
- Print has return value 1
- Echo can take multiple parameters
- Print can take 1 argument

PHP Data Types

String

- Sequence if characters like “Japan”
- Texted inside quotes “ “ (single or double quote can be used)

Integer

- non-decimal number between -2,147,483,648 and 2,147,483,647

Float

- Number with decimals

Boolean

- Represents two state, true or false
- Used in conditional testing

Array

- Stores multiple value in a variable

Object

- Class and object concept

Null

- Has one value NULL
- No value assigned to it
- IF variable with no value is created, NULL is automatically assigned

PHP Operators

Arithmetic Operators

- +,-,*,/,%,**,

Assignment Operators

Comparison Operators

- ==
- ===
- !=
- !==
- <>
- >
- <
- >=
- <=
- ⇔

Increment / Decrement

- ++
- - -

Logical Operators

- And
- Or
- Xor
- &&
- ||

- !

String Operators

- .
- .=

Array Operators

- +
- ==
- ===
- !=
- <>
- !==

Conditional Assignment Operators

- ?:
- ??

PHP Flow Control

- If
- If ..else
- If..elseif..else
- Switch

PHP Loop

- While
- Do..while
- For
- Foreach
-