

EARN INUX WITH AVATECH TECHNOLOGY

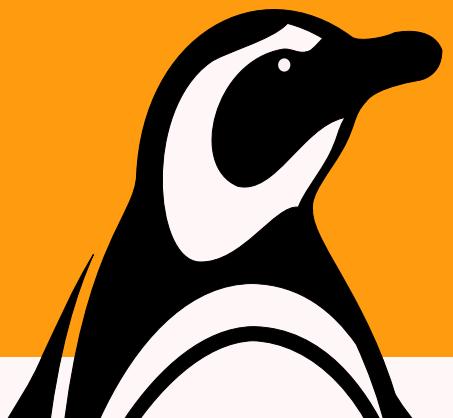
Part 1

Lavatech Technology

Call: 096073 31234

Email: info@lavatechtechnology.com

Website: <https://lavatechtechnology.com>



Copyright © 2022 Lavatech Technology

The contents of this course and all its modules and related materials, including handouts are Copyright ©

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Lavatech Technology.

If you believe Lavatech Technology training materials are being used, copied, or otherwise improperly distributed please e-mail:

info@lavatechtechnology.com

PUBLISHED BY LAVATECH TECHNOLOGY

lavatechtechnology.com

January 2022



YOU CAN

TOTALLY

DO THIS!

Contents

| | | |
|------------|---|-----------|
| 1 | Introduction | 13 |
| 1.1 | Introduction to Operating System | 13 |
| 1.1.1 | What is an Operating System (OS)? | 14 |
| 1.1.2 | Architecture of OS | 15 |
| 1.1.3 | Types of OS | 17 |
| 1.1.4 | Practice | 18 |
| 1.2 | Introduction to Linux | 19 |
| 1.2.1 | What is Open Source Software (OSS)? | 20 |
| 1.2.2 | What is Linux OS? | 21 |
| 1.2.3 | Linux OS v/s Windows OS | 22 |
| 1.2.4 | Understanding console, terminal, command line and shell | 23 |
| 1.2.5 | Linux distributions | 24 |
| 1.2.6 | Practice | 26 |
| 2 | Linux Basics | 29 |
| 2.1 | Linux directory structure | 29 |
| 2.1.1 | Filesystem Hierarchy Standard (FHS) | 30 |

| | | |
|-------|------------------------------------|----|
| 2.1.2 | Important directories in FHS | 31 |
| 2.1.3 | General Linux Shortcuts | 34 |
| 2.1.4 | Practice | 35 |

2.2 Linux commands **38**

| | | |
|-------|------------------------|----|
| 2.2.1 | Basic commands | 39 |
| 2.2.2 | Advance commands | 47 |
| 2.2.3 | Practice | 60 |

3 Editors in Linux **65**

3.1 Gedit & vi/vim editor **65**

| | | |
|-------|--|----|
| 3.1.1 | Editors in Linux | 66 |
| 3.1.2 | Vi/vim in detail | 67 |
| 3.1.3 | Moving around file using vi/vim | 68 |
| 3.1.4 | Searching for text using vi/vim | 69 |
| 3.1.5 | Saving and/or quitting the file using vi/vim | 70 |
| 3.1.6 | More functions in vi/vim | 71 |
| 3.1.7 | Practice | 72 |

4 User & Group Administration **75**

4.1 User & group basics **75**

| | | |
|-------|---|----|
| 4.1.1 | What is a user? | 76 |
| 4.1.2 | What is a group? | 77 |
| 4.1.3 | Files containing user & group details | 79 |
| 4.1.4 | Practice | 86 |

4.2 User & group commands **88**

| | | |
|-------|--------------------------------------|----|
| 4.2.1 | User commands | 89 |
| 4.2.2 | Content of /etc/skel directory | 97 |
| 4.2.3 | Group commands | 98 |
| 4.2.4 | Practice | 99 |

5 Permissions in Linux **103**

5.1 File/Directory permissions **103**

| | | |
|-------|-----------------------------------|-----|
| 5.1.1 | Introduction to permissions | 104 |
|-------|-----------------------------------|-----|

| | |
|----------------------------------|-----|
| 5.1.2 Type of files | 105 |
| 5.1.3 Type of users | 106 |
| 5.1.4 Types of permissions | 107 |
| 5.1.5 Permission commands | 109 |
| 5.1.6 Umask | 118 |
| 5.1.7 Practice | 120 |

6 Advance Permissions 123

6.1 Sudo user 123

| | |
|----------------------------------|-----|
| 6.1.1 What is a sudo user? | 124 |
| 6.1.2 Creating sudo users | 124 |
| 6.1.3 Practice | 127 |

6.2 Special permissions 128

| | |
|------------------------|-----|
| 6.2.1 SUID | 129 |
| 6.2.2 SGID | 132 |
| 6.2.3 Sticky bit | 135 |
| 6.2.4 Practice | 138 |

6.3 Access Control List(ACL) 141

| | |
|--------------------------|-----|
| 6.3.1 What is ACL? | 142 |
| 6.3.2 ACL commands | 143 |
| 6.3.3 Practice | 147 |

7 Processing Commands 149

7.1 Text processing 149

| | |
|----------------------|-----|
| 7.1.1 Commands | 150 |
| 7.1.2 Practice | 160 |

7.2 I/O Redirection 163

| | |
|---|-----|
| 7.2.1 Standard input and standard output device | 164 |
| 7.2.2 Output redirection (>) | 165 |
| 7.2.3 Output append operator (>>) | 165 |
| 7.2.4 Input Redirection (<) | 166 |
| 7.2.5 Error redirection (2>) | 166 |
| 7.2.6 Error append redirection (2>>) | 167 |

| | | |
|--------|---|-----|
| 7.2.7 | Output & error redirection (&>) | 168 |
| 7.2.8 | Output & error append redirection (&>>) | 169 |
| 7.2.9 | Redirection Summary | 170 |
| 7.2.10 | The pipe operator | 171 |
| 7.2.11 | Practice | 172 |

7.3 Archives & Compression 174

| | | |
|-------|----------------|-----|
| 7.3.1 | Commands | 175 |
| 7.3.2 | Practice | 180 |

8 Partition 183

8.1 Introduction to partition 183

| | | |
|-------|--|-----|
| 8.1.1 | What is Hard Disk Drive (HDD) & partition? | 184 |
| 8.1.2 | HDD devices naming in Linux | 185 |
| 8.1.3 | Commands to check HDD | 187 |
| 8.1.4 | Practice | 189 |

8.2 Partition types 191

| | | |
|--------|---------------------------------------|-----|
| 8.2.1 | Partition table in MBR | 192 |
| 8.2.2 | Primary partitions | 193 |
| 8.2.3 | Extended partition | 195 |
| 8.2.4 | Logical partitions | 197 |
| 8.2.5 | Standard formatting filesystems | 200 |
| 8.2.6 | Mounting Partitions | 204 |
| 8.2.7 | Permanent mounting:/etc/fstab | 206 |
| 8.2.8 | Unmounting filesystem | 209 |
| 8.2.9 | Swap partition | 210 |
| 8.2.10 | Setting up swap space | 212 |
| 8.2.11 | Practice | 214 |

9 Logical Volume Management 217

9.1 Introduction to LVM 217

| | | |
|-------|-------------------------|-----|
| 9.1.1 | What is LVM? | 218 |
| 9.1.2 | LVM organisation | 219 |
| 9.1.3 | Volumes practical | 221 |
| 9.1.4 | Physical extends | 228 |

| | |
|----------------------|-----|
| 9.1.5 Practice | 230 |
|----------------------|-----|

10 Links in Linux 233

| | |
|--|------------|
| 10.1 Inodes & Links | 233 |
| 10.1.1 What are inodes? | 234 |
| 10.1.2 Inode number and filesystem | 235 |
| 10.1.3 Links and it's types | 236 |
| 10.1.4 Difference between links | 238 |
| 10.1.5 Practice | 239 |

11 Package Management 241

| | |
|--|------------|
| 11.1 RPM in detail | 241 |
| 11.1.1 What is a package? | 242 |
| 11.1.2 What is an RPM? | 242 |
| 11.1.3 RPM commands | 243 |
| 11.1.4 Drawback of RPM | 246 |
| 11.1.5 Practice | 247 |
| 11.2 YUM & dnf | 249 |
| 11.2.1 What is YUM & dnf? | 250 |
| 11.2.2 Configuration of YUM server | 251 |
| 11.2.3 Configuration of YUM client | 253 |
| 11.2.4 Subscribing RHEL8 server | 254 |
| 11.2.5 YUM commands | 257 |
| 11.2.6 Dnf commands | 260 |
| 11.2.7 Practice | 263 |

12 Process, CPU & Memory 265

| | |
|--|------------|
| 12.1 Process management | 265 |
| 12.1.1 What is a process? | 266 |
| 12.1.2 Types of process | 267 |
| 12.1.3 Linux process states | 272 |
| 12.1.4 How to check Linux process? | 274 |
| 12.1.5 Killing process | 280 |
| 12.1.6 Process priority | 285 |

| | |
|--|------------|
| 12.1.7 Practice | 289 |
| 12.2 CPU management | 292 |
| 12.2.1 What is CPU? | 293 |
| 12.2.2 CPU core & CPU threads | 294 |
| 12.2.3 Types of CPU | 295 |
| 12.2.4 Understanding CPU calculation | 298 |
| 12.2.5 Commands for CPU | 300 |
| 12.2.6 Understanding CPU consumption | 304 |
| 12.2.7 Practice | 305 |
| 12.3 Memory management | 307 |
| 12.3.1 What is RAM? | 308 |
| 12.3.2 What is cache? | 309 |
| 12.3.3 What is buffer? | 310 |
| 12.3.4 Commands for memory | 311 |
| 12.3.5 Practice | 315 |
| 13 Scheduling Jobs | 317 |
| 13.1 Crontab in detail | 317 |
| 13.1.1 Introduction to crontab | 318 |
| 13.1.2 Command to set crontab | 319 |
| 13.1.3 Examples of cronjob | 320 |
| 13.1.4 Practice | 322 |
| 14 Networking | 325 |
| 14.1 Networking essentials | 325 |
| 14.1.1 Networking & it's types | 326 |
| 14.1.2 Network connecting device | 329 |
| 14.1.3 Network interface names | 334 |
| 14.1.4 Practice | 335 |
| 14.2 Introduction to IP | 337 |
| 14.2.1 What is Internet Protocol (IP)? | 338 |
| 14.2.2 What is an IP address? | 341 |
| 14.2.3 What is DNS server? | 342 |

| | |
|--|------------|
| 14.2.4 What is DHCP server? | 343 |
| 14.2.5 Types of IP address | 344 |
| 14.2.6 ISO-OSI model | 346 |
| 14.2.7 Practice | 347 |
| 14.3 IPv4 in detail | 349 |
| 14.3.1 IPv4 address structure | 350 |
| 14.3.2 Network bit & Host bit | 353 |
| 14.3.3 Netmask | 354 |
| 14.3.4 What is classful addressing? | 356 |
| 14.3.5 Loopback & 0.0.0.0 address | 358 |
| 14.3.6 Network address | 359 |
| 14.3.7 Broadcast address | 360 |
| 14.3.8 Calculating number of host IP | 361 |
| 14.3.9 Practice | 365 |
| 14.4 Subnetting | 369 |
| 14.4.1 Drawback of classful addressing | 370 |
| 14.4.2 Subnetting/CIDR | 371 |
| 14.4.3 Working of CIDR | 372 |
| 14.4.4 Practice | 377 |
| 14.5 Networking in action | 378 |
| 14.5.1 Assigning hostname | 379 |
| 14.5.2 /etc/hosts file | 380 |
| 14.5.3 Setting up IP address | 382 |
| 14.5.4 Network monitoring commands | 389 |
| 14.5.5 Practice | 392 |
| 15 Logs in Linux | 395 |
| 15.1 Logging in Linux | 395 |
| 15.1.1 Introduction to logs | 396 |
| 15.1.2 Syslog & rsyslog | 397 |
| 15.1.3 Rules in /etc/rsyslog.conf | 398 |
| 15.1.4 Important log files | 401 |
| 15.1.5 Logger command | 402 |
| 15.1.6 Viewing logs | 403 |

| | |
|-----------------------|-----|
| 15.1.7 Practice | 404 |
|-----------------------|-----|

16 Boot process 407

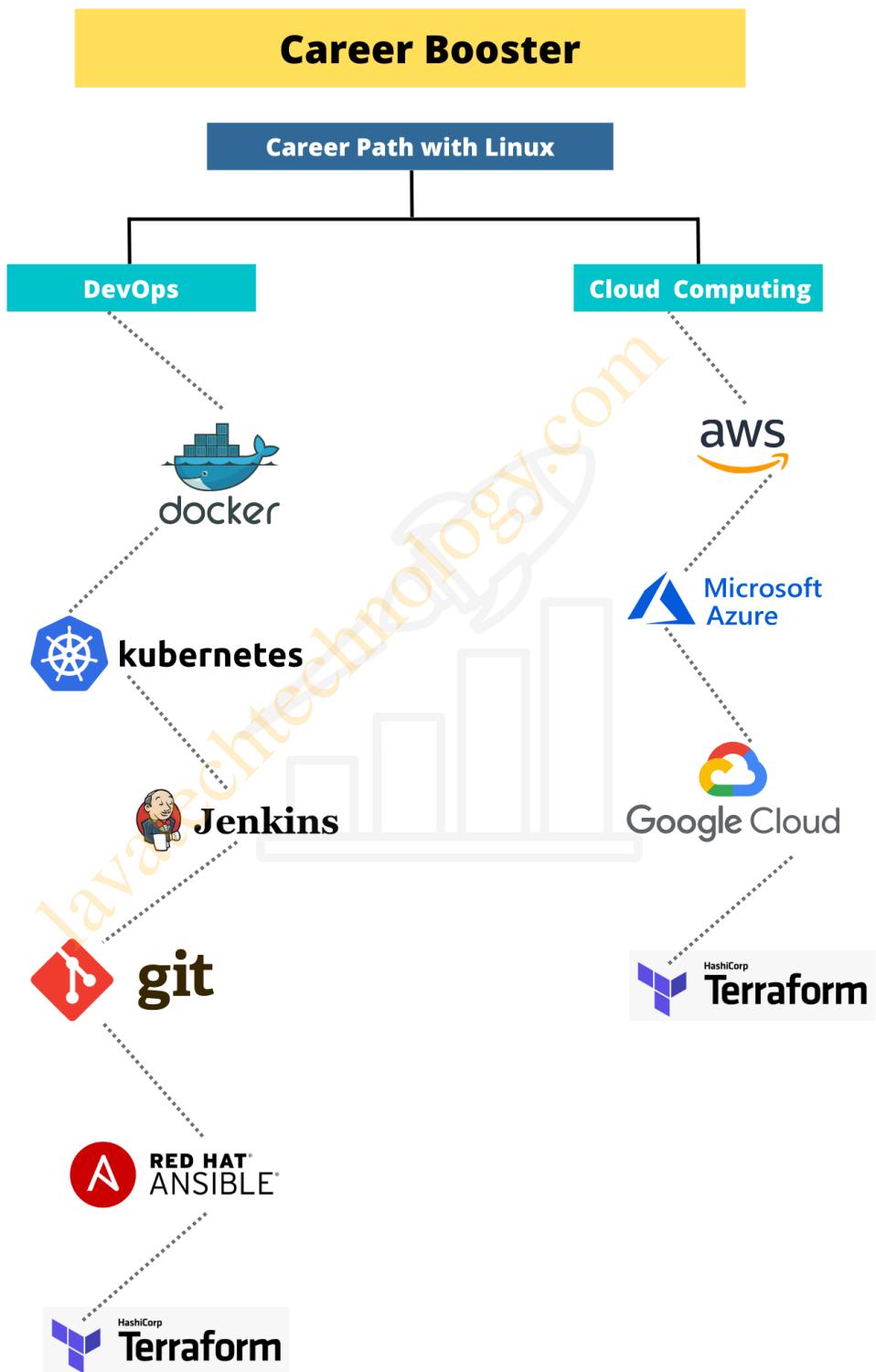
| | |
|------------------------------------|------------|
| 16.1 Booting in Linux | 407 |
| 16.1.1 What is boot process? | 408 |
| 16.1.2 BIOS & POST | 409 |
| 16.1.3 MBR | 411 |
| 16.1.4 Bootloader | 412 |
| 16.1.5 Run levels/Targets | 415 |
| 16.1.6 Systemd | 416 |
| 16.1.7 Practice | 420 |

17 Linux OS Installation 425

| | |
|---|------------|
| 17.1 Installation in detail | 425 |
| 17.1.1 Minimum system requirements for RHEL | 426 |
| 17.1.2 Partition requirement | 427 |
| 17.1.3 Creating a virtual machine | 428 |
| 17.1.4 RHEL OS using Anaconda | 433 |

18 Remote Access in Linux 439

| | |
|---|------------|
| 18.1 SSH, scp & rsync | 439 |
| 18.1.1 What is SSH? | 440 |
| 18.1.2 What is a server & client? | 441 |
| 18.1.3 Setting up SSH server & client on RHEL | 442 |
| 18.1.4 Password based SSH connection | 443 |
| 18.1.5 Passwordless SSH connection | 446 |
| 18.1.6 What is scp command? | 451 |
| 18.1.7 What is rsync command? | 452 |
| 18.1.8 Practice | 453 |



lavatechtechnology.com

LINUX PADHO!

JOB KI CHINTA MAAT KARO



1. Introduction

1.1 Introduction to Operating System

In this section, you are going to learn:

1. What is an Operating System (OS)?
2. Architecture of OS?
3. Types of OS

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

1.1.1 What is an Operating System (OS)?

An Operating System (OS) is an interface between a user and computer hardware.

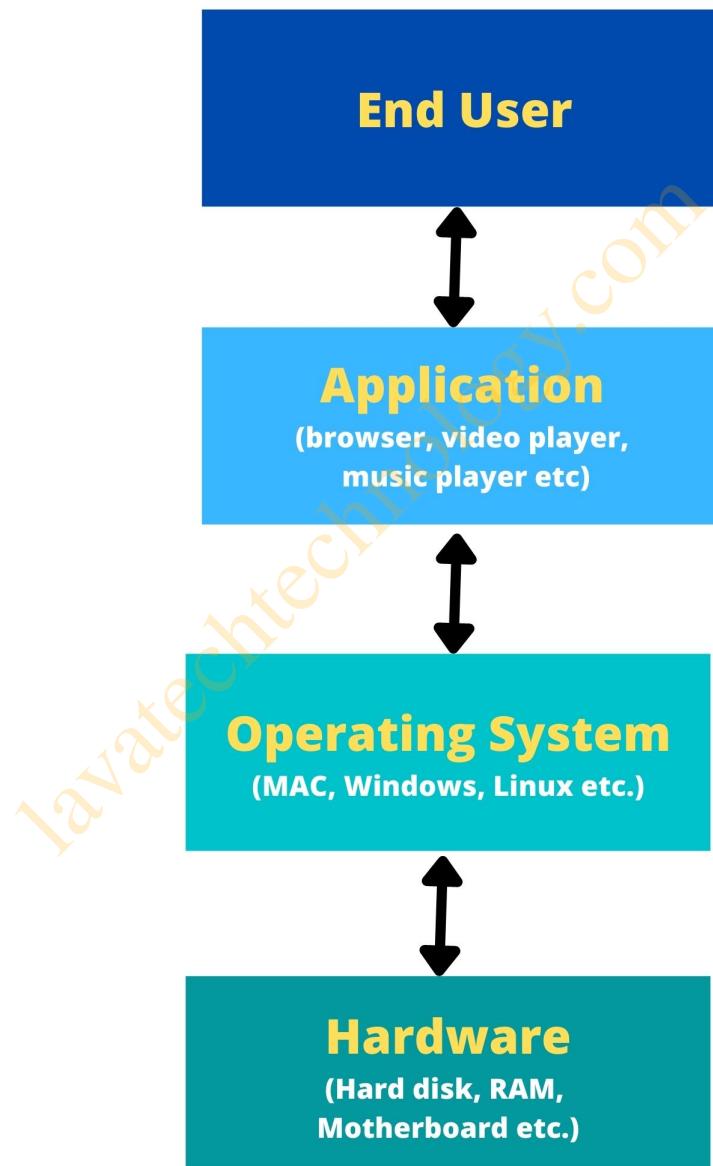


Figure 1.1: Operating System

1.1.2 Architecture of OS

An OS consists of:

- Kernel
- Shell
- Applications

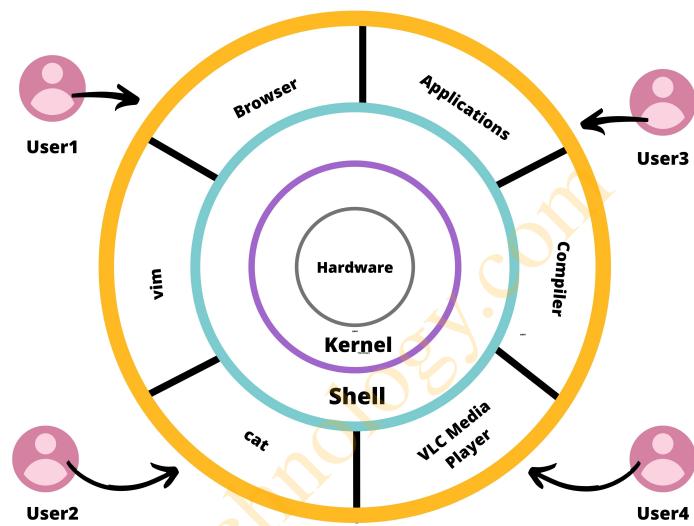


Figure 1.2: OS architecture

Let's see each of these in detail.

1. Kernel -

- Establish communication between hardware and software.

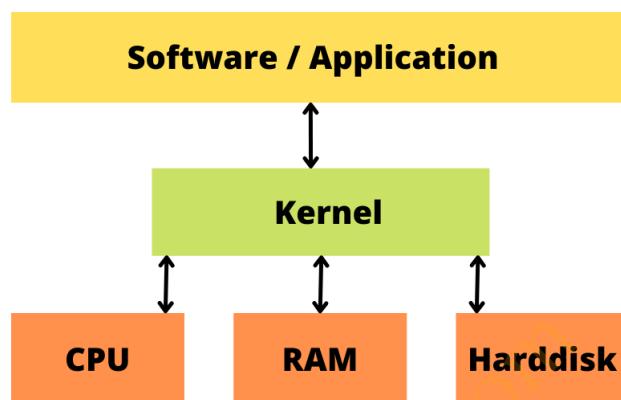


Figure 1.3: Kernel

- Kernel performs:
 - **Process Management:** Manage CPU processes.
 - **Device Management:** Interface between hardware and process.
 - **Memory Management:** Manage RAM.
 - **Handling system calls:** Provides the interface between the processes and OS.

2. **Shell** - An interface to kernel that takes commands from user and executes them.
3. **Utilities/applications** - Programs that can be directly used by users.

1.1.3 Types of OS

There are 3 types of OS:

1. **Server OS** - Designed for server computers that runs 24X7.

Eg:

- Linux server
- Windows server
- Mac OS X server



2. **Desktop OS** - Designed for personal computer.

Eg:

- Windows
- Mac OS

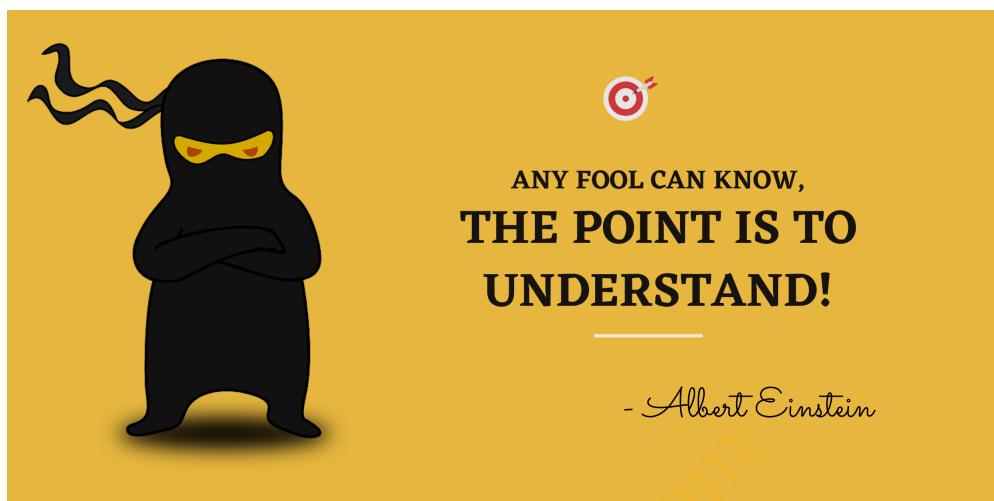


3. **Mobile OS** - Designed to run on mobile devices.

Eg:

- iPhone OS
- Windows mobile
- Anroid



1.1.4 Practice

- 1. What is an Operating System? (Select all that applies.)**
 - (a) A software used for only gaming purpose.
 - (b) Collection of software that acts as an interface between a user and computer hardware.
 - (c) OS consists of kernel to interact with hardware, shell and utilities.
 - (d) OS is a software used for accounting purpose.
- 2. Which of the following are the functions of a kernel? (Select all that applies.)**
 - (a) Perform memory management by keeping track of memory.
 - (b) Perform hardware management.
 - (c) Establish communication between hardware and software.
 - (d) Control all processes of the OS.
- 3. State whether true or false. Kernel is an interface between hardware and software.**
 - (a) True
 - (b) False

1.2 Introduction to Linux

In this section, you are going to learn:

1. **What is Open Source Software (OSS)?**
2. **What is Linux OS?**
3. **Linux OS V/S Windows OS**
4. **What is console, terminal, command line & shell?**
5. **Linux distributions**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

1.2.1 What is Open Source Software (OSS)?

- Open source software (OSS) is code that is publicly accessible.
- OSS is:
 - Free to use
 - Free to modify
 - Free to distribute



Figure 1.4: Open Source Software Logo

Some of the OSS:

| | | | |
|---|---|---|--|
|  Linux OS |  Apache webserver |  VLC media player |  Mozilla firefox |
|  Python |  PHP |  WordPress |  ORACLE VM VirtualBox |

Figure 1.5: Open Source Softwares

1.2.2 What is Linux OS?

- Linux® is an open source operating system.
- Created as a hobby by Linus Torvalds in 1991, as alternative of the MINIX OS (which is variant of Unix).
- Developed for **personal computers, servers, mainframes, mobile devices and embedded devices.**



Figure 1.6: Tux, the Linux mascot, by Larry Ewing

1.2.3 Linux OS v/s Windows OS

| Linux OS | Windows OS |
|--|---|
| Open source OS | Closed source OS |
| Free of cost | Not free |
| File names are case-sensitive | File names are case-insensitive |
| More efficient in resource usage | Less efficient in resource usage |
| More security, no need to install anti-virus | Less security, need to install anti-virus |
| Used for devops, programming, database, cloud-computing, big data hadoop etc | Used mostly for house hold purpose |

1.2.4 Understanding console, terminal, command line and shell

What is a console and terminal?

- A console is a device with a screen and keyboard combined inside it.
- Terminal is the software program inside the console.



Figure 1.7: VT terminals - The first console with terminal

What is a command line?

- It is a blank line and cursor on the screen, allowing the user to type commands to execute.

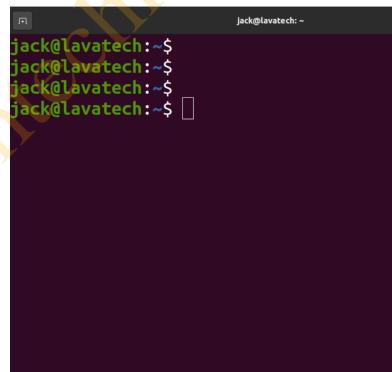


Figure 1.8: Linux command line

What is a shell?

- Shell is an interface to kernel.
- It executes Linux commands & display its result.
- Eg:
 - Shell in Linux OS: bash, fish, zsh, ksh, sh, tsch
 - Shell in Windows OS: PowerShell, pwsh

1.2.5 Linux distributions

- A Linux distribution (or distro) is made from the **Linux kernel and collection of software**.
- Almost one thousand Linux distributions exist.
- Free and community managed distributions are:

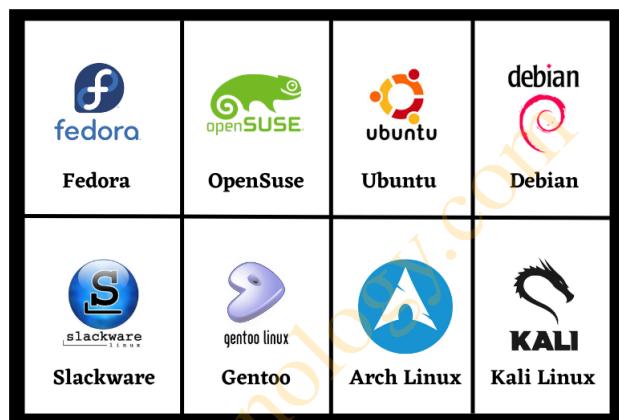


Figure 1.9: Linux distributions

- Popular commercially backed distributions are:

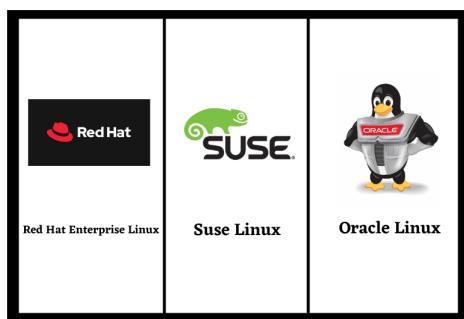
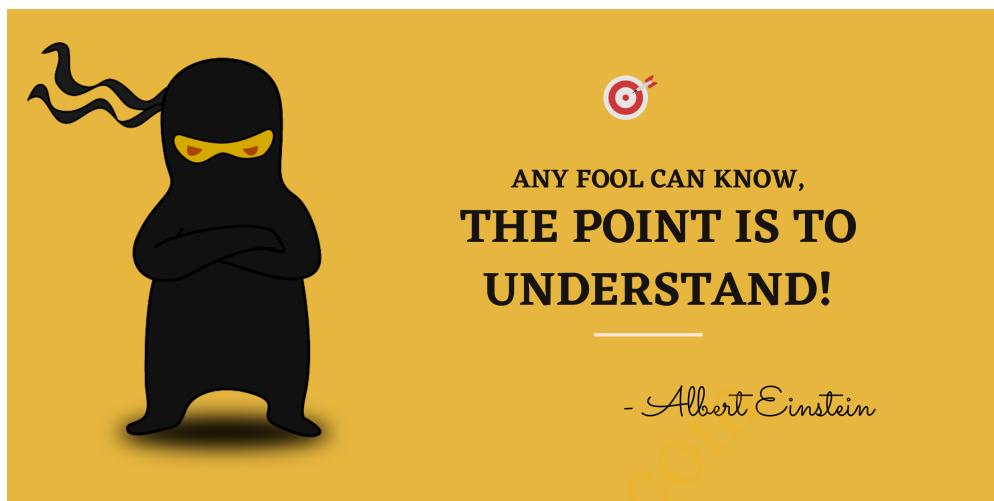


Figure 1.10: Commercial Linux distributions

What is upstream and downstream?

- The term 'upstream' refers to the **original version of a software**.
- Downstream is the **refined product code** based on original software version.
- Eg:
 - **Fedora** is the upstream to **Red Hat Enterprise Linux (RHEL)**.
 - **Debian** is the upstream to **Ubuntu**.

1.2.6 Practice

1. Which of the following are Open Source Software (OSS)? (Select all that applies.)
 - (a) Python
 - (b) Linux
 - (c) Apache webserver
 - (d) MySQL
2. State whether true or false. OSS is free to use, develop, modify and distribute for personal and professional use.
 - (a) True
 - (b) False
3. Select all statement true for Linux OS.
 - (a) Linux is OSS developed by Linus Torvalds.
 - (b) Linux OS cannot be used for house-hold use.
 - (c) Linux OS is variant of MINIX OS.
 - (d) Linux is designed for servers and mainframes computer.

4. Select all statement that are true about Linux and Windows OS.

- (a) File names are case sensitive in Linux OS and case-insensitive in Windows OS.
- (b) Linux OS is less secure than Windows OS.
- (c) Linux OS is OSS while Windows OS is closed source OS.
- (d) Windows OS is expensive compared to Linux OS.

5. Select all statement that are true about shell.

- (a) Bash and ksh are types of shell used in Linux OS.
- (b) Shell takes command from terminal and supplies it to kernel for processing.
- (c) Shell is an interface to kernel.
- (d) Powershell is an example of shell used in Windows OS.

6. State whether true or false. Fedora is upstream of RHEL.

- (a) True
- (b) False



Small steps every day....

2. Linux Basics

2.1 Linux directory structure

In this section, you are going to learn:

1. Filesystem Hierarchy Standard (FHS)
2. Important directories in FHS
3. Some shortcuts to use in Linux OS

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

2.1.1 Filesystem Hierarchy Standard (FHS)

- If you're coming from Windows, you must be aware of C: drive, D: drive etc.
- In Linux, there is **no** C: or D: drive.
- Linux have standard directory structure called Filesystem Hierarchy Standard (FHS).

Note: Folder and directory means the same!

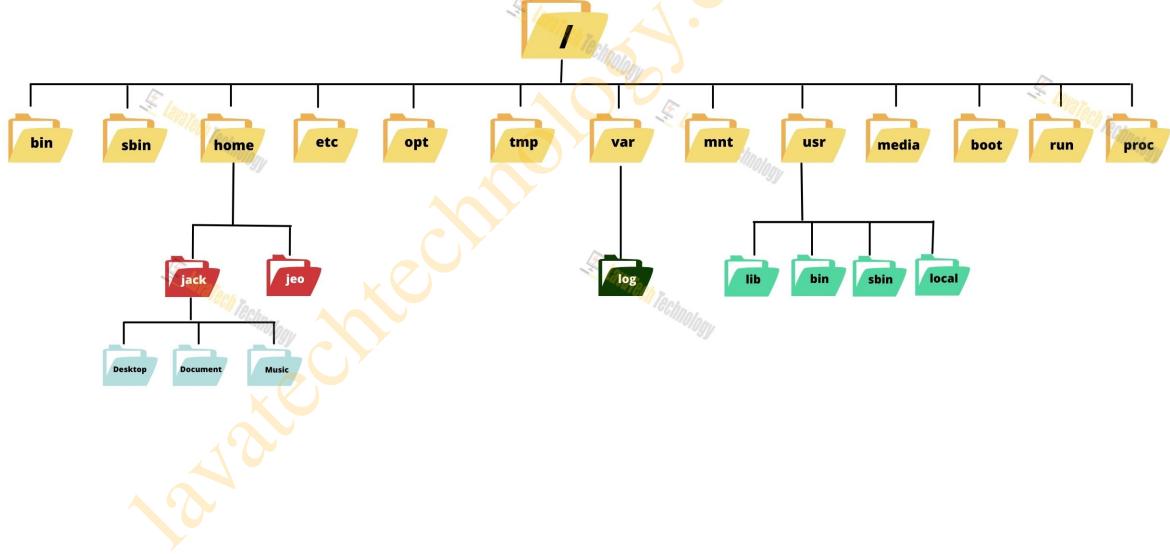


Figure 2.1: Filesystem Hierarchy Standard (FHS)

2.1.2 Important directories in FHS

1. /

- "/" is **top most directory** or **root directory** or **starting point** of FHS.

2. /home

- Contains **home folder** for each user having their personal data & configuration files.
- Eg: Home directory for user **jack** is **/home/jack**.



Figure 2.2: Path separator

Note:

- Symbol "~" refers to the user's home directory.
- Eg: If you are logged in as user "jack", then "~" meant "/home/jack".
- Eg: If you are logged in as user "jill", then ~ meant "/home/jill".

3. /bin

- Contains user binaries (programs).
- Eg: cp, mkdir, pwd, ls, rmdir etc.

4. /sbin

- Contains system administration binaries.
- Eg: fdisk, useradd, userdel, ifconfig etc.

5. /usr

- Contains read-only commands, libraries and data.
 - **/bin** is link to **/usr/bin**
 - **/sbin** is link to **/usr/sbin**
 - **/lib** is link to **/usr/lib**

6. /etc

- Contains system-wide configuration files.
- Eg:
 - **/etc/fstab**
 - **/etc/sysconfig/network-script/**

7. /opt

- Opt stands for optional.
- Used by proprietary 3rd party software to store their settings.

8. /tmp

- Contains temporary files of system.
- Files under this are deleted when system is rebooted.

9. /var

- Var stands for variable files.
- /var includes:
 - System log files: **/var/log**
 - Packages and database files: **/var/lib**
 - Emails: **/var/mail**
 - Lock files: **/var/lock**
 - Temporary files needed across reboots: **/var/tmp**

10. /mnt

- Temporary mount directory.

11. /boot

- Contains grub2 bootloader, kernel, initramfs etc. needed during system boot up.

12. /media

- Mounts temporary media device like USB drive.

13. /proc

- Contains running process informations.
- Eg: /proc/{pid} directory contains information about the process with that particular process.

14. /run

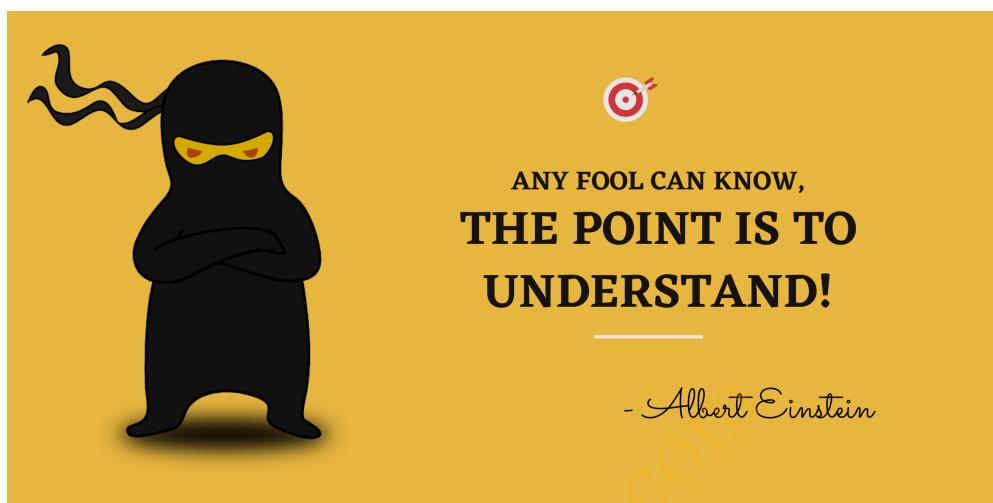
- Used for run-time variable data.
- Eg: Socket files, Process IDs etc.
- Difference between **/run** & **/tmp**:
 - Data in **/run always get deleted at next boot**
 - Data in /tmp may or may not get deleted at next boot.

2.1.3 General Linux Shortcuts

Useful command line-editing shortcuts

| Shortcut | Description |
|-------------------------|---|
| Ctrl + l | Clear the terminal screen. |
| Ctrl + a | Jump to the beginning of the command line. |
| Ctrl + e | Jump to the end of the command line. |
| Ctrl + u | Clear from the cursor to the beginning of the command line. |
| Ctrl + k | Clear from the cursor to the end of the command line. |
| Ctrl + r | Search the history list of commands for a pattern. |
| Ctrl + Shift + c | Copy text from terminal. |
| Ctrl + Shift + v | Paste copied text on terminal. |
| Ctrl + Shift + t | Open new terminal. |
| Alt + Tab | Switch between applications. |

2.1.4 Practice



- 1. Does Linux OS have same directory structure as Windows OS?**
 - (a) Yes
 - (b) No

- 2. Which of the following is the top most directory in Linux OS?**
 - (a) /
 - (b) /home
 - (c) /root
 - (d) /boot

- 3. Which of the following directories contains user binaries & system administration binaries?**
 - (a) /bin
 - (b) /usr/bin
 - (c) /usr/sbin
 - (d) /sbin

- 4. State whether true or false. All files and directories under /tmp may or may not get deleted on system reboot.**

- (a) True
 - (b) False
- 5. Which of the following files/folders are present under /var directory?**
- (a) /var/tmp
 - (b) /var/log
 - (c) /var/mail
 - (d) /var/spool
- 6. Which of the following directory is used for auto-mounting CD-ROM or USB drive?**
- (a) /mnt
 - (b) /media
 - (c) /tmp
 - (d) /run
- 7. Which of the following directory stores information about optional third party application like VLC media player.**
- (a) /mnt
 - (b) /media
 - (c) /opt
 - (d) /run
- 8. The system process in running state stores it's process id and other related information in which of the following directory?**
- (a) /mnt
 - (b) /proc
 - (c) /tmp
 - (d) /run

9. Where is the kernel of Linux OS stored?

- (a) /bin
- (b) /usr
- (c) /proc
- (d) /boot

lavatechtechnology.com

2.2 Linux commands

In this section, you are going to learn:

1. Basic commands in Linux OS
2. Advance commands in Linux OS

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

2.2.1 Basic commands

Before starting with basic commands, let's first understand a few things.

What is command prompt?

- Command prompt is also known as shell prompt.
- Commands are entered in a terminal at the shell prompt.

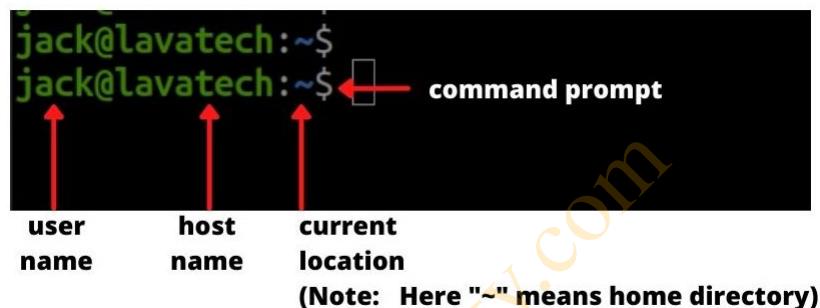


Figure 2.3: Command Prompt

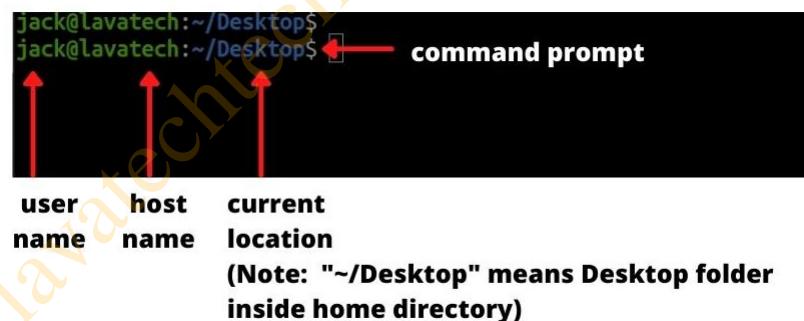


Figure 2.4: Command Prompt

- The shell prompt lists:
 - Username who logged in
 - Server hostname
 - Current directory
 - "\$" prompt (if you are normal user) or "#" prompt (if you are root user)

Command syntax

- Commands entered at the shell prompt have three parts:

Syntax: command [options] [argument]

Explanation:

- **command**: name of command
 - **options**: start with one or two dashes (eg: -a or -all)
 - **arguments**: a target that the command should operate on
 - "[]" means optional.
- Eg:



Figure 2.5: Command syntax

Now let's get started with some actual commands.

1. **pwd**: Shows the user's present working directory.

Eg:

```
# pwd
```

2. **clear**: Clears the contents of the screen.

Eg:

```
# clear
```

Tip: You can use the keyboard shortcut **CTRL+L** as well to clear the screen.

3. **mkdir**: Create a directory.

Syntax: **mkdir [options] foldername**

Eg:

```
# mkdir /home/jack/test
```

Option of **mkdir** command:

- -p: Creates a directory and all its parents too.

Eg:

```
# mkdir -p /home/jack/test/a/b/c
```

4. **ls**: Lists the content of a directory.

Syntax: ls [options] [filename]

Eg:

```
# List current directory content
# ls

# List content of /home directory
# ls /home
```

Options with ls command-

- **-l**: List more details of contents inside directory.

Eg:

```
$ ls -l /home
total 24
drwxr-xr-x 3 jack jack 4096 Feb2 17:02 jack
drwx—— 2 root root 16384 Dec8 14:39 lost+found
```

Output explaination:

- Column 1 – Type & permissions of file
 - Column 2 - Number of links
 - Column 3 - Owner of the file
 - Column 4 - Group under which the file belongs
 - Column 5 - Size of file
 - Column 6 - Date of last update
 - Column 7 – Last updated time of file
 - Column 8 - Name of file/directory
- **-d**: Shows information about directory rather than listing.

Eg:

```
# ls -ld
drwxr-xr-x 3 jack jack 4096 Feb 2 17:02 .
```

- **-a:** Shows all files, including hidden files.

Note: Names of hidden files/folders begin with a dot.

Eg:

```
# ls -a  
. .. .bash_logout .bashrc Desktop .profile
```

5. **cd:** Used to switch between directories.

Syntax: cd [foldername]

Eg:

```
# cd without any argument switch to user's home directory.  
# cd
```

Special characters that can be used with "cd" command:

- "." means current directory
- ".." means parent directory
- "--" would take you to previous working directory
- "~" would take you to home directory of the user

Eg:

```
# cd /home/  
# cd ..  
# cd -  
# cd ~
```

Absolute Path & Relative Path:

- **Absolute path:** Starts with root directory (i.e "/") & is complete path.
- **Relative path:** It is relative to current location & does not start with root directory (i.e "/").

| Absolute Path | Relative Path |
|------------------------|------------------|
| cd /home/jack/Desktop/ | cd jack/Desktop/ |
| cd /var/log/ | cd log/ |

Figure 2.6: Absolute & relative path

6. **cat**: Used to display contents of a file and also to create one.

(a) Creating a file:

Eg:

```
# cat > hello.txt
hi
welcome to unix
# End by pressing Ctrl + d
```

(b) Displaying a file:

Eg:

```
# cat hello.txt
```

7. **touch**: Performs 2 functions:

- (a) Update timestamp of file, if it exists.
- (b) Creates a new file, if it does not exists.

Eg:

```
# touch abc.txt
```

You can create multiple files using brace expansion.

Eg:

```
# touch Sunday,Monday,Tuesday,Wednesday.txt  
# touch file1..3.txt  
# touch filea..h.txt  
# touch filea,b1,2.txt
```

8. cp: Copy file or directory.

Syntax: cp source destination

To copy multiple files to a directory:

```
# cp file1 file2 file3 backup
```

Options with cp command:

- (a) **-r**: Copy directories recursively including all its files and subdirectories.

Eg:

```
# cp -r folder1 newfolder
```

- (b) **Wildcard character "*"**: Copy all files starting with a specific name.

```
# cp file* backup
```

9. mv: Used to move or rename the file/folder.

Syntax: mv source destination

- (a) Renames file(or directory) when the **location of source and destination is same.**

```
# mv old-file new-file
```

- (b) Moves a files/directory to different location if **source and destination is different.**

```
# mv oldfile new-file-location
```

10. **rm:** Deletes one or more files/folders.

Syntax: rm filename

Eg:

```
# rm one.txt
```

Options with **rm** command:

- (a) **-r:** Remove directories and their contents recursively.

```
# rm -r foldername
```

11. **rmdir**: Deletes only empty directory.

Syntax: rmdir directory

Eg:

```
# rmdir /tmp/project/
```

12. **df**: df stands for **disk free**. Displays information about disk usage.

Syntax: df

Options with **df** command:

- (a) **-h**: Display disk-usage size in GBs, MBs etc.

Eg:

```
# df -h
```

- (b) **-T**: Print filesystem of each partition.

Eg:

```
# df -T
```

2.2.2 Advance commands

1. **whoami**: Display username who is logged in.

Eg:

```
# whoami
```

```
jack
```

2. **users**: Display all the username who are currently logged in.

Eg:

```
# users
```

```
jack jill lavatech
```

3. **who**: Displays users currently logged with more details.

Eg:

```
# who
```

```
jack :0 2022-02-02 14:21 (:0)
jill :1 2022-02-03 13:52 (:1)
lavatech pts/0 2022-02-03 14:03 (192.168.0.105)
```

Output explaination:

- Column 1 - Login name
- Column 2 - Login device (TTY or pts)
- Column 3 - Login date
- Column 4 - Login time
- Column 5 - Local device or remote IP address from where the user is logged in

Note:

- TTY stands for **teletypewriter**: It is an input device that

allows alphanumeric character to be typed in and sent to a computer.

- The pts/0 is telling which "**pseudo terminal**" the user is logged in on.

Options with **who** command:

- **-H**: Prints the column headers.

Eg:

```
$ who -H
```

- **-b**: Display the time and date of the last reboot.

Eg:

```
# who -b
```

4. **w**: Shows who is logged and what they are doing.

Eg:

```
# w
14:27:00 up 1 day, 5 min, 2 users, load average: 1.86,
2.44, 2.70
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
jack :0 :0 Wed14 ?xdm? 14:25m 0.03s /usr/lib/g
lavatech :1 :1 13:52 ?xdm? 14:25m 0.00s /usr/lib/g
```

Output explanation:

- Line 1 - Shows below details:
 - Current time
 - How long the system has been running.
 - How many users are currently logged in.
 - System load averages for the past 1, 5, and 15 minutes.
- Line 2 - Header

- Line 3 - Login name, the tty name, the remote host, login time, idle time, JCPU, PCPU, and the command line of their current process.

JCPU - Time used by all processes attached to the tty.

PCPU - Time used by the current process, named in the "what" field.

5. **uname**: Displays the name of OS.

Eg:

```
# uname
```

```
Linux
```

Options with **uname** command:

- **-r**: Displays the current release of OS.

Eg:

```
# uname -r
```

```
3.6.18-194.el8
```

- **-a**: Displays:

- Kernel name
- System name
- Kernel release
- Kernel version
- Machine processor
- Hardware platform

Eg:

```
# uname -a
```

```
Linux lavatech 5.13.0-27-generic #29 20.04.1-  
Ubuntu SMP Fri Jan 14 00:32:30 UTC 2022  
x86_64 x86_64 x86_64 GNU/Linux
```

- **-n**: Prints the hostname of machine.

Eg:

```
# uname -n
```

6. **uptime**: Display belows details in one line:

- Current time
- How long the system has been running
- How many users are currently logged in
- System load averages for the past 1, 5, and 15 minutes

Eg:

```
# uptime
14:48:33 up 1 day, 27 min, 2 users, load average: 2.58, 2.60, 2.76
```

7. **timedatectl**: Control the system time and date

Syntax: timedatectl [options] [arguments]

Eg:

```
root@lavatech:~# timedatectl
          Local time: Tue 2022-05-03 19:09:16 IST
          Universal time: Tue 2022-05-03 13:39:16 UTC
                  RTC time: Tue 2022-05-03 13:39:16
                 Time zone: Asia/Kolkata (IST, +0530)
System clock synchronized: yes
          NTP service: active
      RTC in local TZ: no
```

Figure 2.7: Sample output

Options with **timedatectl** command:

- **list-timezones**: Displays all available timezones.

Eg:

```
# timedatectl list-timezones
```

- **set-timezone:** Sets timezone.

Eg:

```
# timedatectl set-timezone America/Jamaica
```

8. **date:** Display or set the date.

- (a) To display the date:

Eg:

```
# date
Saturday 30 April 2022 12:24:24 PM IST
```

- (b) To set the date, the format is:

Syntax: date "+%formatters"

Eg:

```
# date '+%a %h %d %T'
```

Valid date formatters:

| Date Formatter | Description |
|----------------|---|
| %m | month of year (01-12) |
| %n | prints output to new line |
| %d | day of month (01-31) |
| %y | last two digits of year (00-99) |
| %D | date as mm/dd/yy |
| %H | hour (00-23) |
| %M | minute (00-59) |
| %S | second (00-59) |
| %T | time as HH:MM:SS |
| %j | day of year (001-366) |
| %w | day of week (0-6) Sunday is 0 |
| %a | abbreviated weekday (Sun-Sat) |
| %h | abbreviated month (Jan-Dec) |
| %r | 12-hour time w/ AM/PM (e.g., "03:59:42 PM") |

Options with **date** command:

- **-s datestring**: Sets the time and date to the value specified in the datestring only if you are root user.

Eg:

```
# date -s '11/20/2003 12:48:00'
```

- **-d datestring**: Display the specified date instead of actual current system date.

Eg:

```
# date -d "last friday"
# date -d "next friday"
# date -d "yesterday"
```

9. **cal**: Prints a calendar for the current month.

Syntax: **cal [options] [[month] year]**

Options with **cal** command:

- **-j**: Display julian dates (days numbered 1 to 365, starting from January 1).

Eg: To display calender for month-12 and year-2022 with julian dates.

```
# cal -j 12 2022
```

```
root@lavatech:~# cal -j 12 2022
December 2022
Su Mo Tu We Th Fr Sa
            335 336 337
338 339 340 341 342 343 344
345 346 347 348 349 350 351
352 353 354 355 356 357 358
359 360 361 362 363 364 365
```

Figure 2.8: Sample output

- **-m**: Display specific month.

Eg: To display calender of month-12.

```
# cal -m 12
```

```
root@lavatech:~# cal -m 12
December 2022
Su Mo Tu We Th Fr Sa
                  1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

Figure 2.9: Sample output

- **-y:** Display entire year.
Eg: To display calender of year-2022.

```
# cal -y 2022
```

```
root@lavatech:~# cal -y 2022
                2022
      January           February          March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1       1  2  3  4  5       1  2  3  4  5       1  2  3  4  5
  2  3  4  5  6  7  8     6  7  8  9 10 11 12     6  7  8  9 10 11 12
  9 10 11 12 13 14 15   13 14 15 16 17 18 19   13 14 15 16 17 18 19
16 17 18 19 20 21 22   20 21 22 23 24 25 26   20 21 22 23 24 25 26
23 24 25 26 27 28 29   27 28                      27 28 29 30 31
30 31

      April            May             June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1  2  3  4  5  6  7   1  2  3  4  5  6  7   1  2  3  4
  3  4  5  6  7  8  9     8  9 10 11 12 13 14   5  6  7  8  9 10 11
10 11 12 13 14 15 16   15 16 17 18 19 20 21   12 13 14 15 16 17 18
17 18 19 20 21 22 23   22 23 24 25 26 27 28   19 20 21 22 23 24 25
24 25 26 27 28 29 30   29 30 31                  26 27 28 29 30

      July            August          September
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1  2  3  4  5  6  7   1  2  3  4  5  6       1  2  3
  3  4  5  6  7  8  9     7  8  9 10 11 12 13   4  5  6  7  8  9 10
10 11 12 13 14 15 16   14 15 16 17 18 19 20   11 12 13 14 15 16 17
17 18 19 20 21 22 23   21 22 23 24 25 26 27   18 19 20 21 22 23 24
24 25 26 27 28 29 30   28 29 30 31                  25 26 27 28 29 30
31

      October          November         December
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1  2  3  4  5  6  7   1  2  3  4  5       1  2  3
  2  3  4  5  6  7  8     6  7  8  9 10 11 12   4  5  6  7  8  9 10
  9 10 11 12 13 14 15   13 14 15 16 17 18 19   11 12 13 14 15 16 17
16 17 18 19 20 21 22   20 21 22 23 24 25 26   18 19 20 21 22 23 24
23 24 25 26 27 28 29   27 28 29 30                  25 26 27 28 29 30 31
30 31
```

Figure 2.10: Sample output

10. **ifconfig:** Display the IP address of server.

Eg:

```
# ifconfig
```

lavatechtechnology.com

11. **hostname**: Displays the fully qualified name of server.

Eg:

```
# hostname
```

12. **free**: Displays amount of free and used memory in the system in bytes.

Options with **free** command:

- **-k**: Show the output in Kilobytes
- **-m**: Show the output in Megabytes
- **-g**: Show the output in Gigabytes

Eg:

```
# free  
# free -k  
# free -m  
# free -g
```

13. **shutdown**: Shutdown the machine immediately or schedule a shutdown using 24 hour format.

Syntax: **shutdown [options] [time in 24 hour format]**

- Eg: Shutdown immediately -

```
# shutdown -h now  
or  
# poweroff
```

- Eg: Reboot the system immediately -

```
# shutdown -r now  
or  
# reboot
```

- Eg: Restart OS at specific time, like at 5:30 pm -

```
# shutdown 17:30
```

- Eg: Shutdown OS at specific time, like at 5:30 pm -

```
# shutdown -h 17:30
```

14. **which**: Shows the full path of the command.

Syntax: which command_name

Eg:

```
# which cal  
/usr/bin/cal
```

15. **whereis**: Locate the binary, source, and manual page files for a command.

Syntax: whereis command_name

Eg:

```
# whereis cal
cal: /usr/bin/cal /usr/share/man/man1/cal.1.gz
```

16. **sleep**: Suspends the shell by making it inactive for specified seconds.

Syntax: sleep seconds

Eg:

```
# sleep 10
```

17. **history**: Shows last few commands fired by the current user.

Eg:

```
@lavatech:~$ history
1 ifconfig
2 free -m
3 history
4 cd
5 history
```

Figure 2.11: history command output

Options using **history** command:

-c: Clear the history.

Syntax: history -c

Shortcuts using **history** command:

!: Used to repeat the command executed in past, using its history number.

Syntax: !history-no

Eg:

```

@lavatech:~$ history
1 ifconfig
2 free -m
3 history
4 cd
5 history
6 free -m
7 cd jenkins_master/
8 vagrant ssh
9 history
10 free -m
11 cd
12 history
@lavatech:~$ @lavatech:~$ 16
free -m
      total     used     free   shared  buff/cache available
Mem:      31842      6422    16419      500      9000     24472
Swap:        439         6       432

```

Figure 2.12: history command output

18. **ping**: Used to check if a machine is reachable or not on the network.

Syntax: ping ip-address/hostname

Eg:

```

# ping 192.168.1.0
# ping www.google.com

```

Note: Ping works continuously until ctrl+c is pressed.

Options with **ping** command:

-cN: Send data N number of times using ping command.

Eg:

```

@lavatech:~$ ping -c2 192.168.0.105
PING 192.168.0.105 (192.168.0.105) 56(84) bytes of data.
64 bytes from 192.168.0.105: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 192.168.0.105: icmp_seq=2 ttl=64 time=0.042 ms

--- 192.168.0.105 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1007ms
rtt min/avg/max/mdev = 0.042/0.045/0.049/0.003 ms

```

Figure 2.13: ping command output

19. **man**: Display manual pages of commands.

Syntax: man command_name

Eg:

```
# man date
```

Options with **man** command:

- k: Search for man page using keyword

Syntax: man -k keyword

Eg:

```
jack@lavatech:~$ man -k passwd
chpasswd (8)           - update group passwords in batch mode
chpasswd (8)           - update passwords in batch mode
fgetpwent_r (3)        - get passwd file entry reentrantly
getpwent_r (3)        - get passwd file entry reentrantly
gpasswd (1)            - administer /etc/group and /etc/gshadow
grub-mkpasswd-pbkdf2 (1) - generate hashed password for GRUB
htpasswd (1)           - Manage user files for basic authentication
openssl-passwd (1ssl) - compute password hashes
pam_localuser (8)      - require users to be listed in /etc/passwd
passwd (1)              - change user password
passwd (1ssl)          - compute password hashes
passwd (5)              - the password file
passwd2des (3)         - RFS password encryption
update-passwd (8)       - safely update /etc/passwd, /etc/shadow and /etc/group
jack@lavatech:~$
```

Figure 2.14: Command Prompt

20. **whatis**: Provides very brief descriptions of the command.

Syntax: whatis command_name

Eg:

```
# whatis cal  
cal (1) - displays a calendar and the date of  
Easter
```

21. echo: Display custom message.

Syntax: echo "message"

Eg:

```
# echo "lavatech technology training institute"
```

Executing command along with custom message:

Syntax: echo "message \$(command)"

or

Syntax: echo "message `command`"

Eg:

```
# echo "lavatech technology training institute, date:  
$(date)"  
# echo "lavatech technology training institute, date:  
`date`"
```

22. alias: Sets an alias (similar to nickname) for a command.

Syntax: alias 'shortcutname=command'

Eg:

```
# alias 'cls=clear'
```

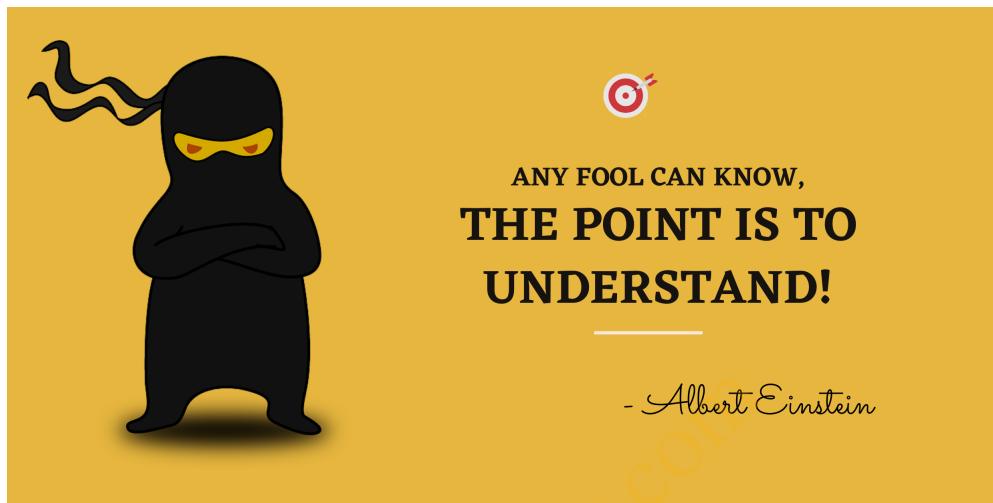
23. **unalias**: Remove an alias.

Syntax: unalias shortcutname

Eg:

```
# unalias cls
```

2.2.3 Practice



- 1. Which of the following command is used to list content of the directory? (Select all that applies.)**
 - (a) pwd
 - (b) mkdir /test1
 - (c) ls
 - (d) ls -l /home

- 2. What are hidden files in Linux?**
 - (a) File/directory having name starting the "*" are hidden files.
 - (b) File/directory having name starting with "." are hidden files.
 - (c) File/directory having name starting with ".." are hidden files.
 - (d) None of above

- 3. Which of the following command is used to create directory along with parent directory?**
 - (a) mkdir /a/b/c
 - (b) mkdir -d /a/b/c
 - (c) mkdir -p /a/b/c
 - (d) mkdir -t /a/b/c

4. If "cd" command is used without any options, where does it take you?
 - (a) Root directory
 - (b) Home directory
 - (c) Parent directory
 - (d) Present working directory
5. Which of the following command can be used to create a file?
(Select all that applies.)
 - (a) cat
 - (b) mkdir
 - (c) cd
 - (d) touch
6. What option of "cp" command is used to copy a directory recursively?
 - (a) cp -d /source_folder /destination_folder
 - (b) cp -p /source_folder /destination_folder
 - (c) cp -r /source_folder /destination_folder
 - (d) cp -t /source_folder /destination_folder
7. Which of the following command is used to rename or move a file/directory?
 - (a) cp
 - (b) ls
 - (c) mv
 - (d) df
8. Which of the following command is used to delete an empty directory? (Select all that applies.)
 - (a) rm -r /foldername
 - (b) rmdir /foldername

- (c) rmdir -e /foldername
- (d) rm /foldername

9. What command is used to display disk usage in Linux? (Select all that applies.)

- (a) df
- (b) df -T
- (c) df -h
- (d) df -Th

10. Which of the following command is used to display details of all logged in users? (Select all that applies.)

- (a) users
- (b) who
- (c) w
- (d) whoami

11. Which of the following command displays details of the OS?

- (a) who
- (b) w
- (c) uname
- (d) df

12. What command displays the JCPU and PCPU in Linux?

- (a) who
- (b) w
- (c) uptime
- (d) alais

13. Select all the valid "date" command examples.

- (a) date -d "yesterday"
- (b) date
- (c) date '+%H'
- (d) date -s "12/20/2022"

14. Which of the following command is used to display memory usage?

- (a) df
- (b) uptime
- (c) w
- (d) free -m

15. What command is used to create shortcut of any command?

- (a) which
- (b) ping
- (c) alias
- (d) echo

16. Which of the following command is used to display IP address?

- (a) history
- (b) ifconfig
- (c) uptime
- (d) free -g

17. What is use of "ping" command?

- (a) Check IP address of OS.
- (b) Check if the IP address is reachable or not.
- (c) Monitor networking.
- (d) Display details about the OS.

18. What command is used to display full path of any command?

- (a) which
- (b) whereis
- (c) how
- (d) echo

lavatechtechnology.com

3.1 Gedit & vi/vim editor

In this section, you are going to learn:

- 1. What is vi/vim editor?**
- 2. Using vim editor:**
 - How to move around in the file?**
 - How to searching for text in the file?**
 - How to save or not save the file?**
 - Other related function related to file editing**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

3.1.1 Editors in Linux

There are 2 major types of editors in Linux:

- **Grahpicl editor**

- **gedit**: Gedit application is a full-featured text editor.

Syntax: gedit filename

Eg:



Figure 3.1: Sample output

- **Text-based editor**

- **vi or vim**: Text based editor used in Linux and Mac OS. Let's see more on this.

3.1.2 Vi/vim in detail

What is vi editor?

- Vi is a free and open source, screen-based text editor program for Linux/Unix OS.

What is vim editor?

- Vim is an advance version of vi.

We are going to explore "vim" editor in detail as it is more advance than "vi".

How to use vi or vim editor?

```
# vim one.txt  
or  
# vi one.txt
```

This will open an editor in front of you as shown:

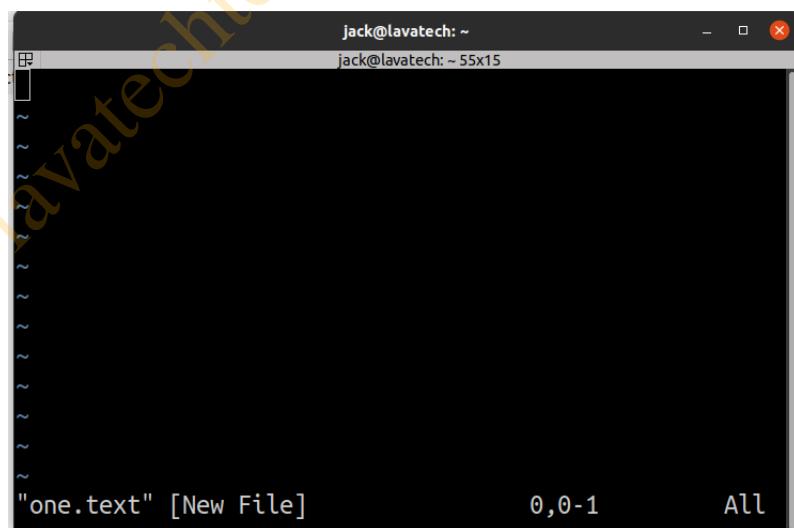


Figure 3.2: Vim Editor

3.1.3 Moving around file using vi/vim

There are two modes using which you can navigate & edit a file in vim:

- **Command mode:**

- Press **Esc** to enter command mode.
- This is default mode.
- File cannot be edited under this mode.
- Below keystrokes will move the cursor:
 - * **\$**: Move cursor to the end of the line.
 - * **17G**: Move cursor to line 17 (i.e type 17 and press shift + g).
 - * **G**: Move cursor to the last line (i.e press shift + g).

- **Insert mode:**

- Press **i** or **a** or **o** to enter insert mode.
- This mode is used to edit the file.
- Below keystrokes will allow you to enter this mode:
 - * **i**: Insert text just **before** the current cursor position.
 - * **a**: Insert text just **after** the current cursor position.
 - * **o**: Insert text into a **new line** below current line.
 - * **I**: Insert text at the **beginning** of the current line.
 - * **A**: Insert text at the **end** of the current line.
 - * **O**: Insert text into a **new line** above current line.

3.1.4 Searching for text using vi/vim

You can search a specific word in vim editor:

Forward search:

- Press **ESC** to switch to command mode.
- Press "/" and type the word to search.

Eg: **/bunny and press enter**: Jump forward to the first occurrence of the string "bunny" after current cursor position.

- **press n**: Jumps to the next occurrence of the word searched by "/".
- **press N**: Jumps to the previous occurrence of the word searched by "/".

Backward search:

- Press **ESC** to switch to command mode.
- Press "?" and type the word to search.

Eg: **?bunny and press enter**: Jump backward to the first occurrence of the string "bunny" before current cursor position.

- **press n**: Jumps to the next occurrence of the word searched by "?".
- **press N**: Jumps to the previous occurrence of the word searched by "?".

3.1.5 Saving and/or quitting the file using vi/vim

File can be saved and/or quit in vim editor.

To do this, **switch to command mode (by pressing Esc)** and then do any of the following:

- **:wq** or **:x** and press enter - **Save & quit** the edited file.
- **:q!** and press enter - **Quit** the editor without saving changes.
- **:w** and press enter - **Save** the edited file without quitting.

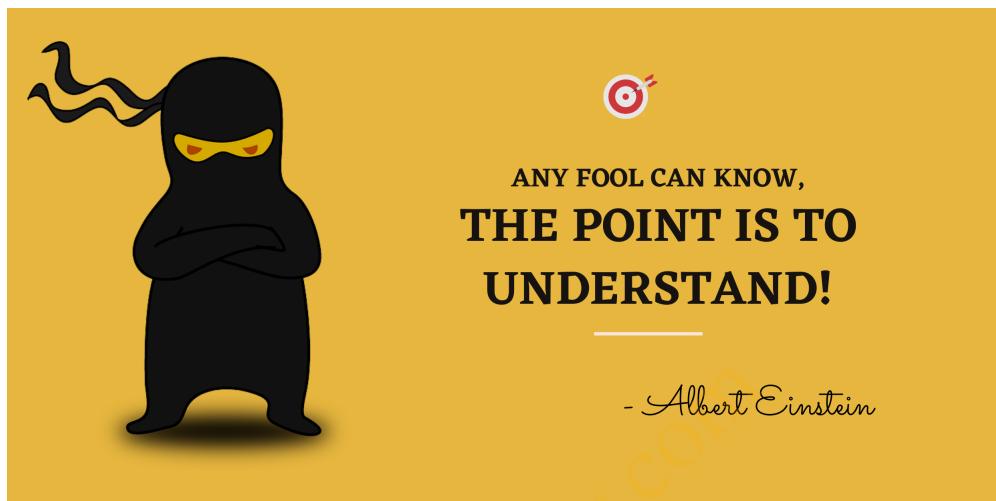
lavatechtechnology.com

3.1.6 More functions in vi/vim

Other useful functions in command mode (**press Esc to switch to command mode**) are:

- **u** - Undo the last change to the file (and type "u" again to re-do the change).
- **U** - Undo all changes to the current line.
- **yy** - Copy a line.
- **p** - Paste a line.
- **dd** - Delete a line.
- **yy** - Copy a single line.
- **Nyy** - Copy N number of lines. Eg: To copy 5 lines, press 5yy.
- **p** - Paste a single line.
- **Np** - Paste N number of times a copied line. Eg: To paste a line 5 times, press 5p.
- **dd** - Delete a single line.
- **Ndd** - Delete N number of lines. Eg: To delete 5 lines, press 5dd.
- **:set nu and press enter** - Show all line numbers .
- **:set nonu and press enter** - Removes all line numbers.
- **:s/Joe/Bob and press enter** - Substitute first occurrence of Joe in current line by Bob.
- **:s/Joe/Bob/g and press enter** - Substitute all occurrences of Joe in current line by Bob.
- **:%s/Joe/Bob/g and press enter** - Substitute every "Joe" to "Bob" throughout the document.

3.1.7 Practice



1. Which of the following combination is used to save and exit a file in vim editor? (Select all that applies.)
 - (a) :x
 - (b) :wq
 - (c) :wq!
 - (d) :w
2. Which of the following keystroke is used to undo changes in vim editor? (Select all that applies.)
 - (a) Press "z" in command mode
 - (b) Press "U" in command mode
 - (c) Press "u" in command mode
 - (d) Press "s" in command mode
3. Which of the following keystroke is used to enter in inset mode in vim editor? (Select all that applies.)
 - (a) Press "a"
 - (b) Press "i"
 - (c) Press "o"

- (d) Press "z"
4. Which of the following keystroke is used to enter in command mode in vim editor?
- (a) Press "CTRL"
 - (b) Press "ATL"
 - (c) Press "SHIFT"
 - (d) Press "ESC"
5. Which of the following keystroke is used to delete a line in vim editor?
- (a) Press "d" in command mode
 - (b) Press "y" in command mode
 - (c) Press "p" in command mode
 - (d) Press "dd" in command mode
6. Which of the following keystroke is used to copy 5 lines in vim editor?
- (a) Press "5y" in command mode
 - (b) Press "5yy" in command mode
 - (c) Press "5p" in command mode
 - (d) Press "5dd" in command mode
7. Which of the following keystroke is used to paste a line 10 times in vim editor?
- (a) Press "10p" in command mode
 - (b) Press "10pp" in command mode
 - (c) Press "10y" in command mode
 - (d) Press "10yy" in command mode
8. Which of the following string helps in setting line numbers in vim editor?

- (a) **:set nu** in command mode
- (b) **:set line** in command mode
- (c) **:s/nu** in command mode
- (d) **:set nonu** in command mode

9. Which of the following string is used substitute a word in entire file in vim editor?

- (a) **:replace/old_word/new_word/** in command mode
- (b) **:s/old_word/new_word/g** in command mode
- (c) **:%s/old_word/new_word/g** in command mode
- (d) **:set nu** in command mode

4.1 User & group basics

In this section, you are going to learn:

1. **What is a user & group?**
2. **Types of group**
3. **Files containing user & group details**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

4.1.1 What is a user?

To login to any UNIX machine, you need a user. There are 3 types of users in Linux:



Figure 4.1: User types

- **Superuser i.e. root:**
 - Automatically created when you install Linux.
 - Superuser can execute commands under **/usr/sbin & /usr/bin**.
- **System users:**
 - Created automatically during application or software installation.
 - Eg: When you install VLC media player in Linux, a user named "vlc" will be created automatically.
 - These accounts exist to allow different services to interact with your computer.
- **Normal or Local users:**
 - Created by root or sudo user.
 - Local user can execute commands under **/usr/bin**.
 - Eg: jack, jill, ram, ravi etc.

4.1.2 What is a group?

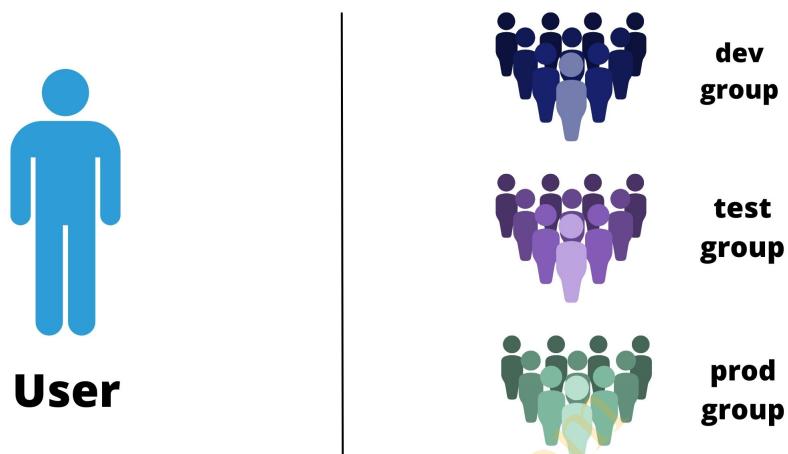


Figure 4.2: User V/S Group

- Group is a collection of users.
- Users are grouped together for granting common permissions to all members of a group.
- A user can be a member of more than one group.

Types of group

- **Primary group:**
 - Primary group are automatically generated while creating a user.
 - A user becomes member of their primary group automatically.
 - Primary group have same name as the username & a unique group ID.
 - **One user can have one and only one primary group.**
- **Secondary or supplementary group:**
 - Secondary group are created separately with the help of **groupadd** command.
 - Multiple users can be added to the secondary group.
 - **One user can have more than one secondary groups.**

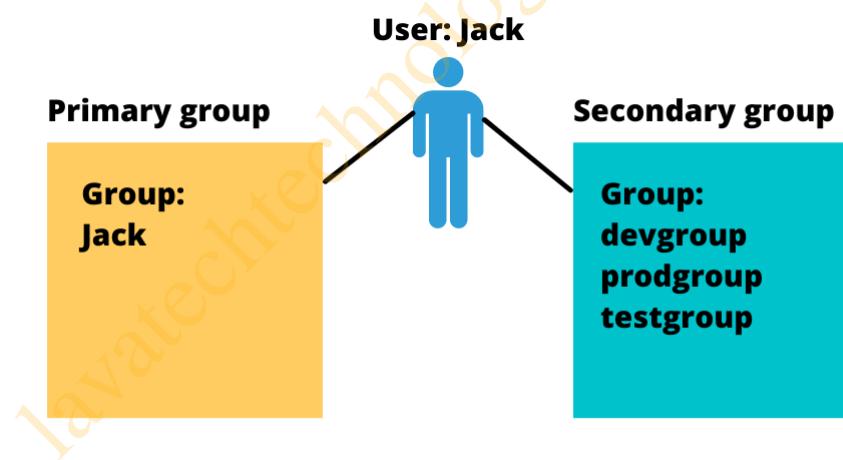


Figure 4.3: Primary and Secondary group

4.1.3 Files containing user & group details

Below diagram shows important files and their short description relating to user & group details:

User details

Filename: /etc/passwd

Description:

- Contains user details like username, uid etc.

Filename: /etc/shadow

Description:

- Contains user's password details
- This file should be highly secure therefore only root can read it.

Group details

Filename: /etc/group

Description:

- Contains group details like groupname etc.

Filename: /etc/gshadow

Description:

- Contains the password information for group accounts.

Figure 4.4: Important user & group files

Let's see each of these file in detail.

Structure of /etc/passwd file

- Fields in this file are separated by a ":" (colon).

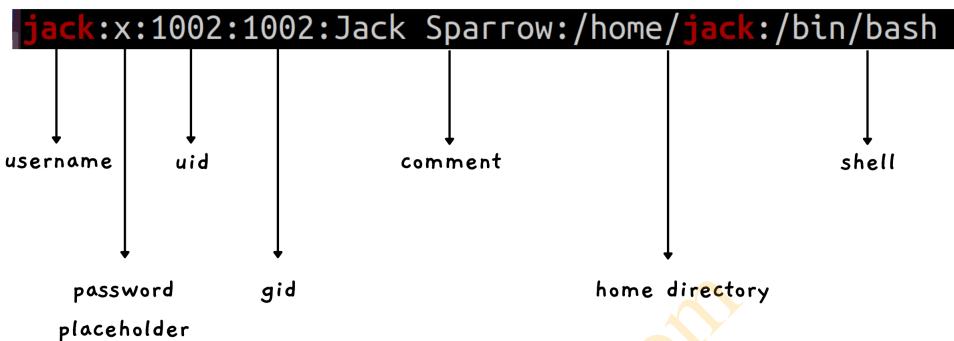


Figure 4.5: Sample /etc/passwd entry

Explanation:

- Username** - Case-sensitive login name of user.
- Password placeholder** - "x" acts as placeholder. Encrypted password are now stored in **/etc/shadow** file.
- User ID: UID ranges:**
 - UID 0 is assigned to root user.
 - UID 1-200 is a range of "system users" assigned to system processes by OS.
 - UID 201-999 is a range of "system users" used by system processes.
 - UID 1000+ is the range available for assignment to regular users.
- Group ID**
- Comment** - More description about user.
- Home directory** - Location of user's personal data.
- Shell** - User login shell, default is **/bin/bash**.

Structure of the /etc/shadow file

- Fields in this file are separated by a ":" (colon).

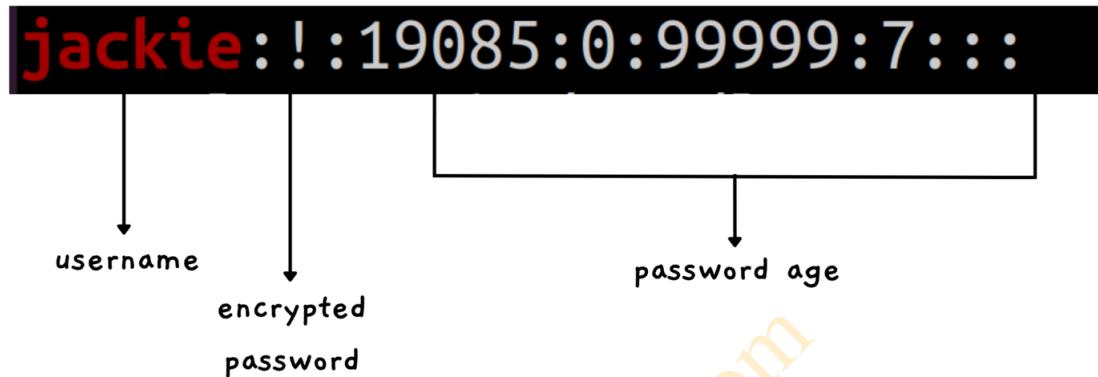


Figure 4.6: Sample /etc/shadow entry

Explanation:

- Username** - A direct match to the username in the /etc/passwd file.
- Encrypted Password** - Store password in encrypted format.

Note:

- “!” in this entry means the account is disabled or locked.
- “!!” mask means password is not assigned.

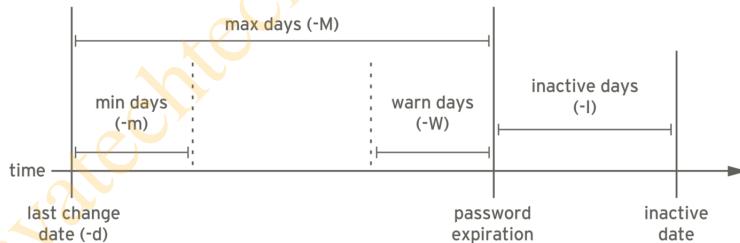
Let's understand the encrypted password in detail.



There are three pieces of information stored in password:

- **Hashing algorithm:** The number 1 indicates an MD5 hash. The number 6 appears when a SHA-512 hash is used.
- **Salt value:** The random salt value used to encrypt the hash. Prevents two users with the same password from having identical entries in the /etc/shadow file.
- **Encrypted hash**

3. Password age:



- (a) **Last password change** - The number of days (since January 1, 1970) since the password was last changed.
- (b) **Minimum days before password change** - The number of days before password may be changed (0 indicates it may be changed at any time).
- (c) **Maximum days before password change** - The number of days after which password must be changed (99999 indicates user can keep his or her password unchanged for many, many years).
- (d) **Password change warning** - The number of days to warn user of an expiring password (7 for a full week).

- (e) **Account active days** - The number of days an account remains active after a password has expired.
- (f) **No. of days account is expired** - The account expiration date, represented as the number of days since 1970.01.01.
- (g) This blank field is reserved for future use.

lavatechtechnology.com

Structure of the /etc/group file

- Fields in this file are separated by a ":" (colon).

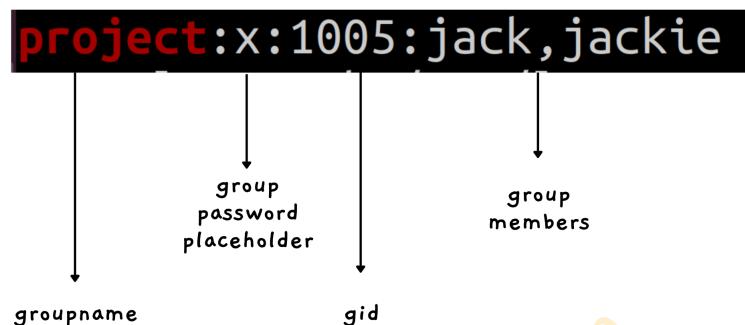


Figure 4.7: Sample /etc/group entry

Explanation

- Column 1: **Groupname** - Name of group. It is case-sensitive.
- Column 2: **Group password placeholder** - "x" acts as placeholder. Encrypted password are stored in /etc/gshadow file.
- Column 3: **Group ID**
- Column 4: **Group members** - Usernames of users belonging to this group.

Structure of the /etc/gshadow file

- Fields in this file are separated by a ":" (colon).

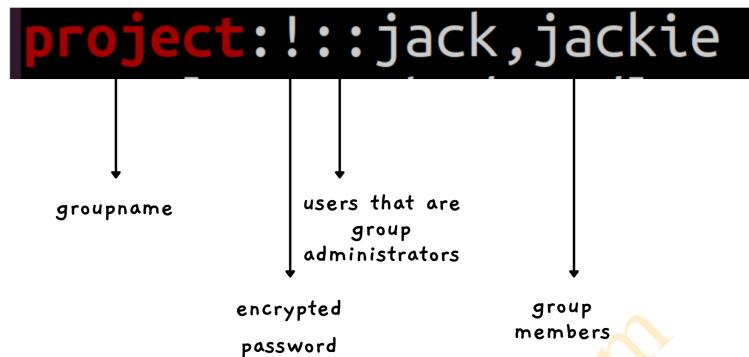


Figure 4.8: Sample /etc/gshadow entry

Explanation:

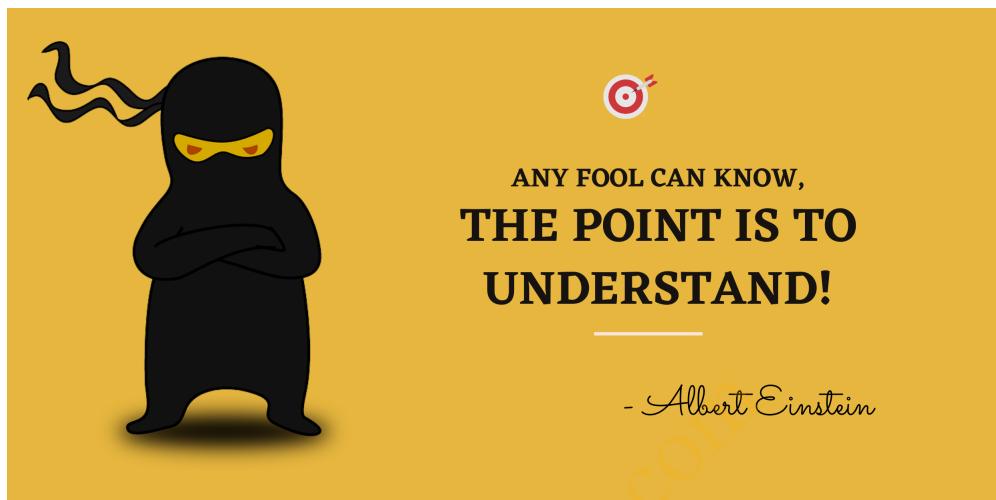
- Column 1: **Groupname** - Name of group. It is case-sensitive.
- Column 2: **Encrypted Password** - Store password in encrypted format.

Note:

- * “!” in this entry means the account is disabled or locked.
- * “!!” mask means password is not assigned.

- Column 3: **Administrators** - Users who can change the password.
- Column 4: **Group members** - Usernames of user belonging to this group.

4.1.4 Practice



1. State whether true or false. One user can have one & only one primary group and 0 or more secondary groups.
 - (a) True
 - (b) False
2. Which of the following file contains user details?
 - (a) /etc/shadow
 - (b) /etc/skel
 - (c) /etc/group
 - (d) /etc/passwd
3. Which of the following file contains user password details?
 - (a) /etc/group
 - (b) /etc/gshadow
 - (c) /etc/shadow
 - (d) /etc/passwd
4. State whether true or false. Default shell of user is /bin/bash.
 - (a) True

(b) False

5. State whether true or false. Group is collection of users.

- (a) True
- (b) False

6. Which of the following are valid fields of /etc/passwd file? (Select all that applies.)

- (a) shell
- (b) home directory
- (c) comment
- (d) username

7. State whether true or false. Primary group is created automatically at the time of user creation.

- (a) True
- (b) False

8. State whether true or false. Secondary group is created automatically at the time of user creation.

- (a) True
- (b) False

4.2 User & group commands

In this section, you are going to learn:

1. **User commands**
2. **Content of /etc/skel directory**
3. **Group commands**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

4.2.1 User commands

Let's see some commands to create, update & delete users in Linux OS.

How to create a new user?

useradd: Create a new user.

Syntax: useradd username

On adding a new user, a user have below things set:

- Home directory: **/home/username**
- Default shell: **/bin/bash**
- Primary group: Named same as **username**
- Password: **No password is set**

Note: Some defaults, such as the UID numbers and default password aging rules, are read from the **/etc/login.defs** file.

Options with **useradd** command:

- (a) **-g:** Assign primary group.

Syntax: useradd -g primary_group_name username

Eg:

```
# useradd -g javadevl shekhar
```

- (b) **-G:** Assign secondary groups.

Syntax: useradd -G secondary_group_name username

Eg:

```
# useradd -G cdevl,perldevl shekhar
```

lavatechtechnology.com

- (c) **-u**: Assign specific UID. UID should be more than 500 for normal users.

Syntax: useradd -u UID username

Eg:

```
# useradd -u 601 shekhar
```

- (d) **-s**: Change user shell.

Syntax: useradd -s shell_name username

Eg:

```
# useradd -s /sbin/nologin shekhar
```

Note: Shell **/sbin/nologin** will not allow username to login.

Usually used for system users like ftp, squid etc.

- (e) **-d**: Change user's home directory.

Syntax: useradd -d directory_name username

Eg:

```
# useradd -d /mnt/shekar shekhar
```

How to assign or change user's password?

passwd: Assign/change user's password.

Syntax: passwd username

On executing the command, you will be asked to set password twice.

Note:

- Root user or superuser can set the password of any user.
- Local user can set their own password, but not of anyone else.

Options with **passwd** command:

- **--stdin:** Change password by providing it on command line itself.

Syntax: echo "new_password" | passwd --stdin username

Eg:

```
# echo "shekhar@12345" | passwd --stdin shekhar
```

How to change user's password attributes?

chage: Changing the password aging information for user.

Syntax: chage username

Eg:

```
root@lavatech:~# chage jack
Changing the aging information for jack
Enter the new value, or press ENTER for the default

        Minimum Password Age [0]:
        Maximum Password Age [99999]:
        Last Password Change (YYYY-MM-DD) [2022-01-25]:
        Password Expiration Warning [7]:
        Password Inactive [-1]:
        Account Expiration Date (YYYY-MM-DD) [-1]:
```

Figure 4.9: Sample output

Options with **chage** command:

- **-l:** Show account aging information.

Syntax: chage -l username

Eg:

```
root@lavatech:~# chage -l jack
Last password change : Jan 25, 2022
Password expires      : never
Password inactive     : never
Account expires       : never
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
```

Figure 4.10: Sample output

Switch between users

- **su:** Allows a user to switch to a different user account. If a username is not specified, the root account is implied.

Syntax: su [-] [username]

Eg:

```
# su - jack
```

What is used of "-" in su command?

- The command "su **username**" starts a non-login shell, while the command "**su - username**" starts a login shell.
- The main distinction is "**su -**" sets up the shell environment as if this were a clean login as that user, while "**su**" just starts a shell as that user with the current environment settings.

Running command as root with sudo

- **sudo:** Allows a user to be permitted to run a command as root, or as another user, based on settings in the /etc/sudoers file.

Syntax: sudo command

Eg:

```
$ sudo useradd raman
```

How to modify an existing user?

usermod: Modify existing user.

Options with **usermod** command:

- (a) **-g:** Change user's primary group.

Syntax: usermod -g primary_group username

Eg:

```
# usermod -g cdevl shekhar
```

- (b) **-G:** Change user's secondary group.

Syntax: usermod -G secondary_group_name username

Eg:

```
# usermod -G javadevl shekhar
```

- (c) **-L:** Lock (i.e. disable) user.

Syntax: usermod -L username

- (d) **-U:** Unlock user account.

Syntax: usermod -U username

- (e) **-s:** Change user's login shell.

Syntax: usermod -s shell_name username

Eg:

```
# usermod -s /bin/ksh shekhar
```

lavatechtechnology.com

- (f) **-d**: Change user's home directory.
- (g) **-m**: Create user's new home directory.

Syntax: usermod -d directory_name username -m

Eg:

```
# usermod -d /opt/java shekhar -m
```

How to delete a user?

Standard Practice: You should not delete a user if they leave the organization. You should lock their account.

userdel: Delete user & remove it's entry from **/etc/passwd** and **/etc/shadow** files.

Syntax: userdel username

Eg:

```
# userdel shekhar
```

Note: userdel command does not delete user's home directory by default.

Options with **userdel** command:

- **-r:** Delete user along with home directory.

Syntax: userdel -r username

Eg:

```
# userdel -r shekhar
```

4.2.2 Content of /etc/skel directory

/etc/skel directory:

- Content of this directory is **automatically copied over to a new user's home directory when it is created.**
- Users gets same initial settings and environment using content of this directory.
- Below are the files under /etc/skel directory that are copied:
 - **/etc/skel/.bash_profile** copied as **~/.bash_profile**:
 - * Configures the bash startup environment.
 - * You can add environment variables to this file.
 - **/etc/skel/.bashrc** copied as **~/.bashrc**:
 - * In this file you can include commands you want to run when you start the bash shell.
 - * You can also add aliases such as: **rm='rm -i'**
 - **/etc/skel/.bash_logout** copied as **~/.bash_logout**:
This file is executed when you exit a bash shell.

Another file containing user's login setting is **/etc/profile**. Note that this file is not copied in the user's home directory.

4.2.3 Group commands

groupadd: Create a new group.

Syntax: groupadd groupname

Eg:

```
# groupadd project
```

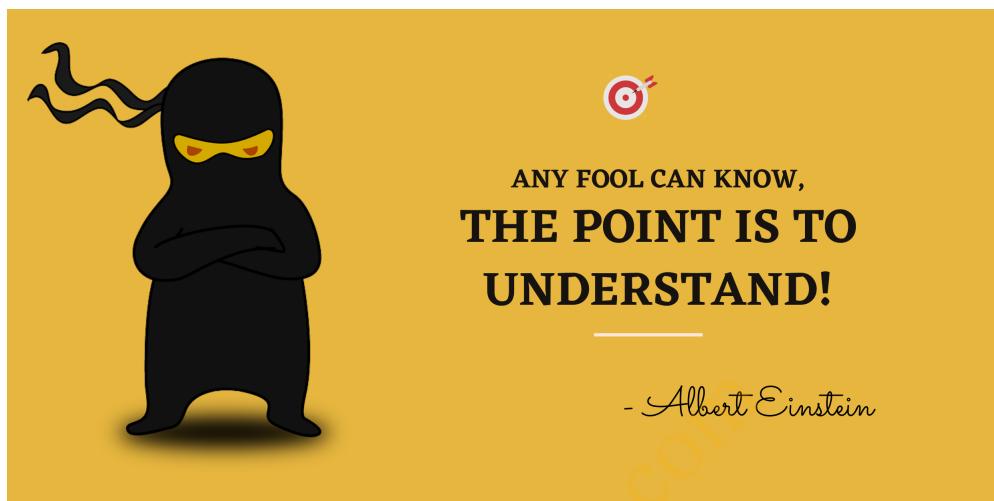
groupdel: Delete a group.

Syntax: groupdel groupname

Eg:

```
# groupdel project
```

4.2.4 Practice



1. Which of the following "useradd" and "usermod" command option is used to assign primary group to user?
 - (a) -g
 - (b) -G
 - (c) -S
 - (d) -s

2. Which of the following command is used to assign password to user?
 - (a) chage
 - (b) passwd
 - (c) useradd
 - (d) usermod

3. Which of the following "userdel" command option is used to delete user along with home directory?
 - (a) -R
 - (b) -m
 - (c) -r

(d) -d

4. Which of the following "useradd" and "usermod" command option is used to assign secondary group to user?
 - (a) -g
 - (b) -G
 - (c) -S
 - (d) -s
5. Which of the following actions are performed during user creation? (Select all that applies.)
 - (a) User home directory is created
 - (b) Default shell of user is /bin/bash
 - (c) Secondary group for user is created
 - (d) User is assigned primary group
6. Which of the following "useradd" command option is used to create and assign home directory to user.
 - (a) -h & -M
 - (b) -h & -m
 - (c) -m & H
 - (d) -H & -M
7. Which of the following "usermod" command option is used to change home directory of the user?
 - (a) -D
 - (b) -h
 - (c) -d
 - (d) -m

8. Which of the following "usermod" command option is used to lock and unlock user account?
- (a) -L & U
 - (b) -l & u
 - (c) -d & m
 - (d) -c & m

lavatechtechnology.com

5.1 File/Directory permissions

In this section, you are going to learn:

1. **What is a permissions?**
2. **Types of files**
3. **Types of users**
4. **Types of permissions**
5. **Permission commands**
6. **Umask**

Finally, there will be a **small excrise** on these topics to check your knowledge.



So let's get started....

5.1.1 Introduction to permissions

- File/directory permissions are used to control who is able to read, write and execute a certain file/directory.
- Command to check file/directory permission:

Syntax: ls -l file/directory

Eg:

```
$ ls -l /home/jack
drwxrwxr-x 2 jack jack 4096 Feb 2 17:02 Desktop
```

The first column of the output shows the permission.

Diagram illustrating the "ls -l" command output. The output is:

```
drwxrwxr-x 2 jack jack 4096 Feb 2 17:02 Desktop
```

A red arrow points from the word "Permission" to the first column of the output, which is highlighted with a blue box.

Figure 5.1: "ls -l" output

Let's see this column in detail.

5.1.2 Type of files

First character from the permissions field defines the file type.



Figure 5.2: First column of permission

There are seven type of files in Linux:

| | |
|------------------------------------|---|
| "-" - Regular file | \$ ls -ld /etc/passwd -rw-r--r-- 1 root root 3118 Feb 3 14:00 /etc/passwd |
| "d" - Directory | \$ ls -ld /home/jack drwxr-xr-x 3 jack jack 4096 Feb 5 17:25 /home/jack |
| "l" - Symbolic link | \$ ls -ld /bin lrwxrwxrwx 1 root root 7 Dec 8 14:39 /bin -> usr/bin |
| "s" - Unix domain socket | \$ ss -xln grep socket \$ ls -ld /run/snapd.socket srw-rw-rw- 1 root root 0 Feb 6 15:48 /run/snapd.socket |
| "p" - Named pipe | \$ mkfifo my_own_pipe # This command will create a new pipe file \$ ls -ld my_own_pipe prw-rw-r-- 1 jack jack 0 Feb 7 05:12 my_own_pipe |
| "c" - Character device file | \$ ls -ld /dev/tty1 crw--w---- 1 root tty 4, 1 Feb 6 15:48 /dev/tty1 |
| "b" - Block device file | \$ ls -ld /dev/sda1 brw-rw---- 1 root disk 259, 6 Feb 6 15:48 /dev/sda1 |

Figure 5.3: File types with example

5.1.3 Type of users

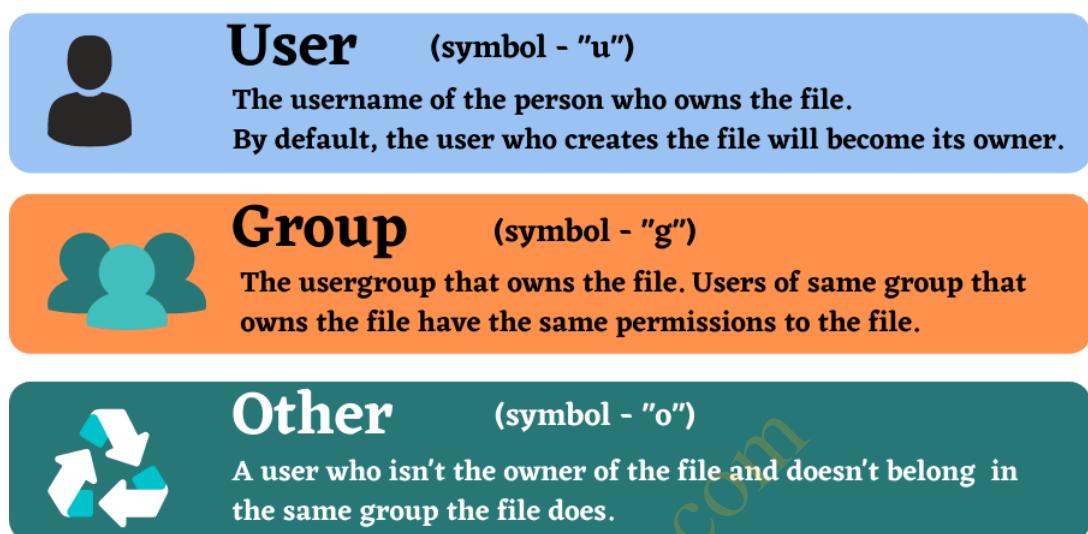


Figure 5.4: User types

Let's map these user with the permissions:

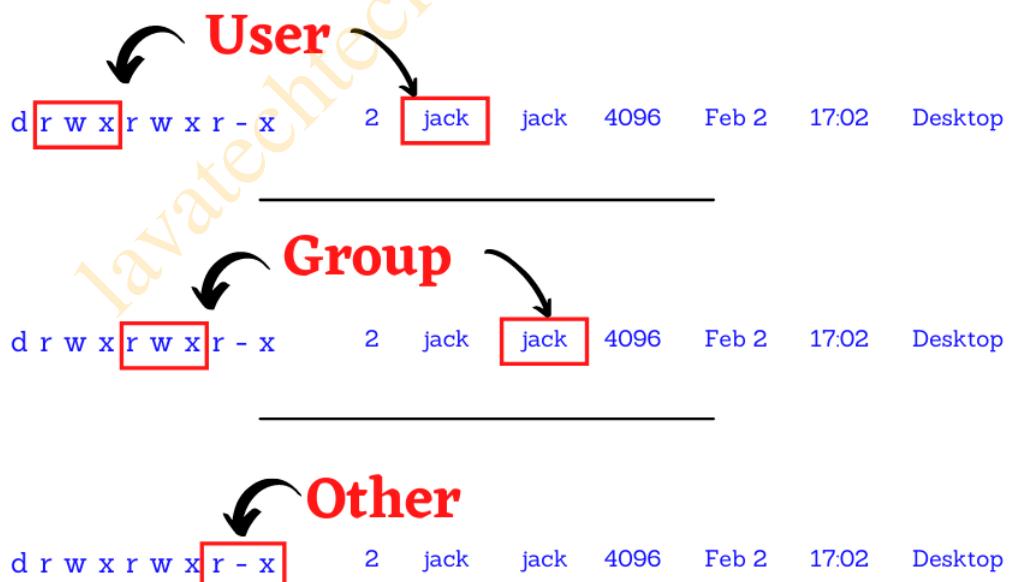


Figure 5.5: Permission mapping

5.1.4 Types of permissions

There are 3 permission available for all types of files:

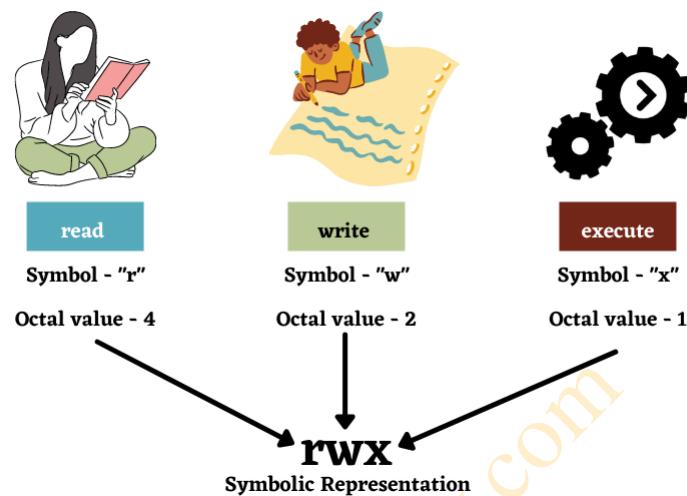


Figure 5.6: File Permission

Let's see what does this means for a normal file and directory.

Read means:

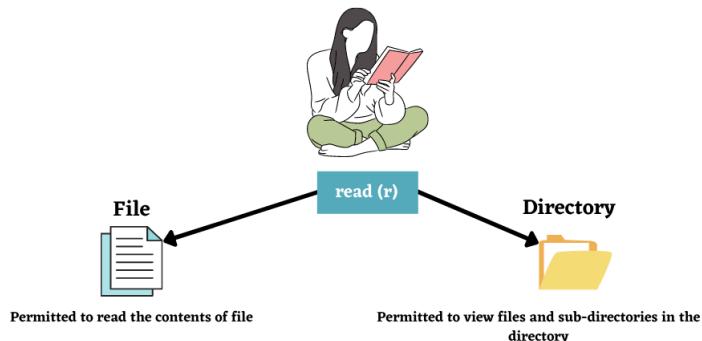


Figure 5.7: Read Permission

Write means:

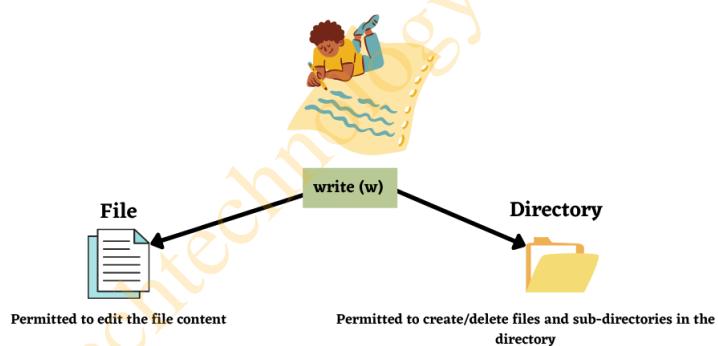


Figure 5.8: Write Permission

Execute means:

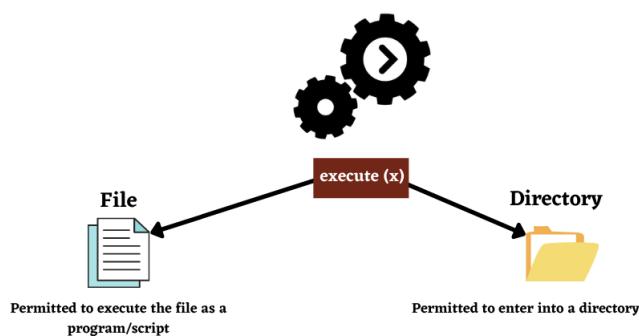


Figure 5.9: Execute Permission

5.1.5 Permission commands

stat: Check file permissions and more details like access time, modify time & change time.

Syntax: stat file_or_directory_name

Eg:

```
[root@server ~]# stat /root
  File: /root
  Size: 4096          Blocks: 8          IO Block: 4096   directory
Device: fd00h/64768d  Inode: 134289537    Links: 3
Access: (0550/dr-xr-x---)  Uid: (    0/      root)  Gid: (    0/      root)
Context: system_u:object_r:admin_home_t:s0
Access: 2022-03-21 10:59:10.804201320 +0000
Modify: 2022-03-19 05:28:51.291139595 +0000
Change: 2022-03-19 05:28:51.291139595 +0000
 Birth: 2022-02-02 01:16:48.735641473 +0000
```

Figure 5.10: Sample output

ls -ld: Check file permission and details like file owner, group owner & timestamp.

Syntax: ls -ld file_or_directory_name

Eg:

```
[root@server ~]# ls -ld /root/
dr-xr-x---. 3 root root 4096 Mar 19 05:28 /root/
```

Figure 5.11: Sample output

chmod: Change the permission of a file or directory using either the octal representation or symbolic representation.

Syntax: chmod permission file_or_directory_name

The permission in chmod command can be supplied using:

- Octal representation
- Symbolic representation

Let's see each of these in detail.

Octal representation**read : 4**

Figure 5.12: Octal representation of read

**write: 2**

Figure 5.13: Octal representation of write

**execute: 1**

Figure 5.14: Octal representation of execute

Let's review the different combinations. Observe the letter representation, the octal representation, and the meaning in the shown image:

| Letter representation | Octal representation | Meaning |
|-----------------------|----------------------|-------------------------|
| rwx | 7 | Read, write and execute |
| rw- | 6 | Read, write |
| r-x | 5 | Read, and execute |
| r-- | 4 | Read, |
| -wx | 3 | Write and execute |
| -w- | 2 | Write |
| --x | 1 | Execute |
| --- | 0 | no permissions |

Figure 5.15: Permission octal representation

Octal permission combination for user, group & other can be as follows:

| Permission | Octal representation | Field |
|------------|----------------------|-------|
| rwx----- | 700 | User |
| ---rwx--- | 070 | Group |
| -----rwx | 007 | Other |

Figure 5.16: Permission combination

Egs:

- Give read, write ($4+2 = 6$) to user and read (4) to group and others.

```
# chmod 644 filename
```

- Give read, execute ($4 + 1 = 5$) to user and read (4) to group, and nothing (0) to others.

```
# chmod 540 filename
```

- Give read, write ($4 + 2 = 6$) to user and nothing (0) to group, and read (4) to others.

```
# chmod 604 filename
```

Symbolic representation

Below image shows all the symbols used in permission:

| Type of user | Operation | Permission |
|--|---|--|
| User - u  | Plus  | Read - r  |
| Group - g  | Minus  | Write - w  |
| Other - o  | Equal to  | Execute - x  |
| All - a  | | |

Figure 5.17: Symbolic representation to assign permissions

Symbols here indicates:

- **u** : User
- **g** : Group
- **o** : Other
- **a** : All
- **+** : Add permission
- **-** : Remove permission
- **=** : Assign permission
- **r** : Read
- **w** : Write
- **x** : Execute

Let's see some of the examples to understand how we can use the symbolic representation.

- Assign user, group and other with read and write permission.

```
# chmod =rw myfile  
or  
# chmod u=rw myfile  
# chmod g=rw myfile  
# chmod o=rw myfile
```

- Remove read and write permission for other.

```
# chmod o=rw myfile
```

- Add read and remove write permission from group.

```
# chmod g+r-w myfile
```

- Assign group with read permission, remove write permission from group. Assign other with read,write & execute permission.

```
# chmod g+r-w,o=rwx myfile
```

- Assign read, write permission to user, group & other.

```
# chmod a=rw myfile  
or  
# chmod =rw myfile
```

Command to change user and group ownership

chown: Changes the user and group ownership of for given file or directory.

Syntax: chown username file_or_directory

Syntax: chown username:groupname file_or_directory

Egs:

- Change file ownership to natasha user.

```
# chown natasha demo.txt
```

- Change file ownership to natasha user and group ownership to devops.

```
# chown natasha:devops demo.txt
```

Options with **chown** command:

-R: Recursively change ownership of directories and their contents.

Syntax: chown -R username directory

Syntax: chown -R username:groupname directory

Eg: Change the owner of a directory and it's subfiles to "shammy".

```
# chown -R shammy /foo
```

Command to change group ownership

chgrp: Changes the group ownership for given file or directory.

Syntax: chgrp groupname file_or_directory

Eg: Change file "**demo.text's**" group ownership to "**devops**" group.

```
# chgrp devops demo.txt
```

Options with **chgrp** command:

-R: Recursively change ownership of directories and their contents.

Syntax: chgrp -R groupname directory

Eg: Change the groupname of "**/foo**" directory and it's subfiles/subfolders to "devops" group.

```
# chgrp -R devops /foo
```

5.1.6 Umask

What is a umask?

- Umask is responsible for the default permission of a file or directory.
- Command to check default umask:

Syntax: umask

Eg:

```
# umask  
0022
```

- Default umask value is **0022**.
- Using the default umask, you can calculate:
 - Default file permission:

Formula: **Complete file permission - Default umask**

Which means: **666 - 022 = 644**

Hence, **default file permission is 644**

- Default folder permission:

Formula: **Complete folder permission - Default umask**

Which means: **777 - 022 = 755**

Hence, **default file permission is 755**

Changing the umask value

- Umask value can be changed using "**umask**" command.

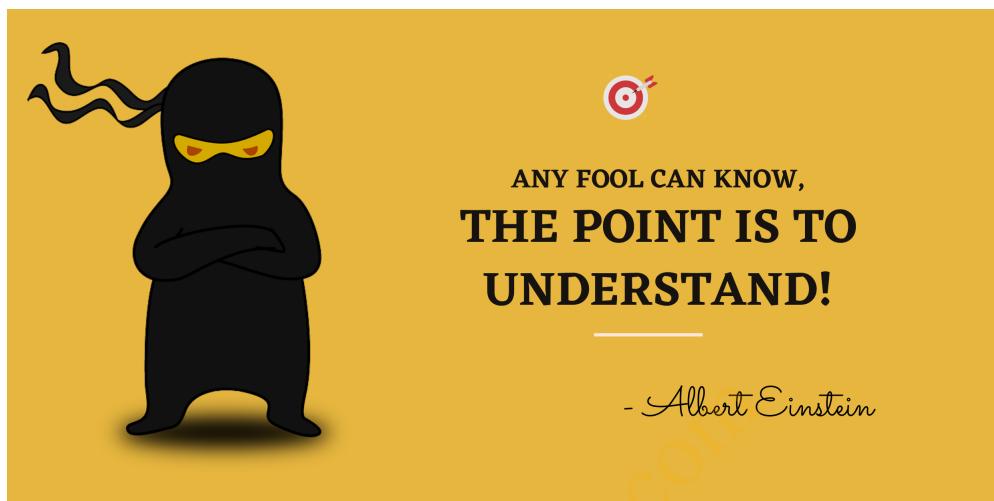
Syntax: umask permission

Eg: If you don't want anybody other than the user (owner) to do anything on the file or directory then you can give umask as 0077.

```
# umask 0077
```

After this, if you create a file or directory, it will have permissions only for the user as shown below:

```
# touch testfile
# ls -l testfile
-rw—— 1 jack jack 0 Mar 17 08:23 testfile
```

5.1.7 Practice

1. Select the character used to represent normal file.
 - (a) =
 - (b) -
 - (c) _
 - (d) ~
2. Which of the following is the default umask?
 - (a) 0222
 - (b) 0202
 - (c) 0220
 - (d) 0022
3. Select the character used to represent block device file.
 - (a) l
 - (b) p
 - (c) s
 - (d) b

4. Select the correct octal representation for:

- user: read,write,execute
 - group: write,execute
 - other: no permission
- (a) 730
(b) 733
(c) 755
(d) 700

5. Select the character used to represent socket file.

- (a) l
(b) p
(c) s
(d) b

6. Which of the following is the correct octal numbering for write, read and execute respectively?

- (a) 2,4,1
(b) 4,2,1
(c) 1,2,3
(d) 1,4,2

7. Select correct combination to give a file permission as : user - read,write & execute permission, other - read and execute permission and group - no permission.

- (a) chmod u=rwx,o+rx-w,g= filename
(b) chmod 705 filename
(c) chmod uo=rx,u+w,g-rwx,o-w filename
(d) chmod 755 filename

8. Select correct combination to give a file permission as: other - read permission, user - read and execute permission.

- (a) chmod u+rx-w,o+r-wx filename
 - (b) chmod ugo=rwx filename
 - (c) chmod u+rx,o+x filename
 - (d) chmod 504 filename
- 9. Which of the following command is used to change ownership of directory to "ravi" and group to "testgrp"?**
- (a) chown ravi directoryname
 - (b) chown ravi: directoryname
 - (c) chown ravi:testgrp directoryname
 - (d) chown ravi directoryname & chgrp testgrp directoryname
- 10. Which of the following command is used to change group ownership of directory to "devops" including all subfiles and subdirectory?**
- (a) chgrp -r devops directoryname
 - (b) chgrp -R devops directoryname
 - (c) chown -R :devops directoryname
 - (d) chown -r :devops directoryname
- 11. Which of the following is the default directory and file permissions?**
- (a) 777 & 666
 - (b) 757 & 606
 - (c) 774 & 600
 - (d) 700 & 600

6.1 Sudo user

In this section, you are going to learn:

1. **What is a sudo user?**
2. **How to create a sudo users?**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

6.1.1 What is a sudo user?

- Sudo users are users who can use **sudo** command.
- The **sudo** command allows local users to **run admin commands without granting them root user's access.**

6.1.2 Creating sudo users

/etc/sudoers: This file is used to create sudo users.

- This file can be edited only using **visudo** command:

```
# visudo
```

- Syntax of lines in this file:

```
users    hosts=(user:group) commands
OR
users    hosts=(user)   commands
OR
users    hosts=      commands
OR
# To avoid password prompt everytime user runs
admin commands
users    hosts=(user:group) NOPASSWD:   commands
```

- Sample entry: Root can run all commands as per below entry in **/etc/sudoers**. Do not comment out this line.

```
root    ALL=(ALL:ALL)  ALL
```

This means, **root** user can run **ALL commands** on **ALL hosts** as **ALL user and ALL group**.

Creating sudo user with all command access

This can be done in 2 ways:

- Way I:

- Find below line in the file **/etc/sudoers** & remove comment(#).

```
# visudo  
%wheel  ALL=(ALL)  ALL
```

Line explanation:

1. %wheel: "%" indicates group is wheel.
 2. The line means, **ALL members of wheel group** can run **ALL commands on ALL hosts** as **ALL user** and **ALL group**.
- Add the user you want to give sudo access to (eg. neo) to the wheel group.

Eg:

```
# usermod -aG wheel neo
```

- Way II:

To give user **bob** and **bunny** sudo access without password prompt:

```
# visudo  
bob,bunny  ALL=(ALL)  NOPASSWD:  ALL
```

Granting specific command's sudo access to user

- Give user **Peter** access to **/bin/kill**, **/usr/bin/kill** & **/usr/bin/pkill** command:

```
# visudo  
Peter  ALL=/bin/kill,  /usr/bin/kill,  /usr/bin/pkill
```

Using aliases in the sudoers file

- Users can be grouped and assigned a nickname or alias which is used throughout the **/etc/sudoers** file.
- Commands can also be grouped and assigned an aliases.

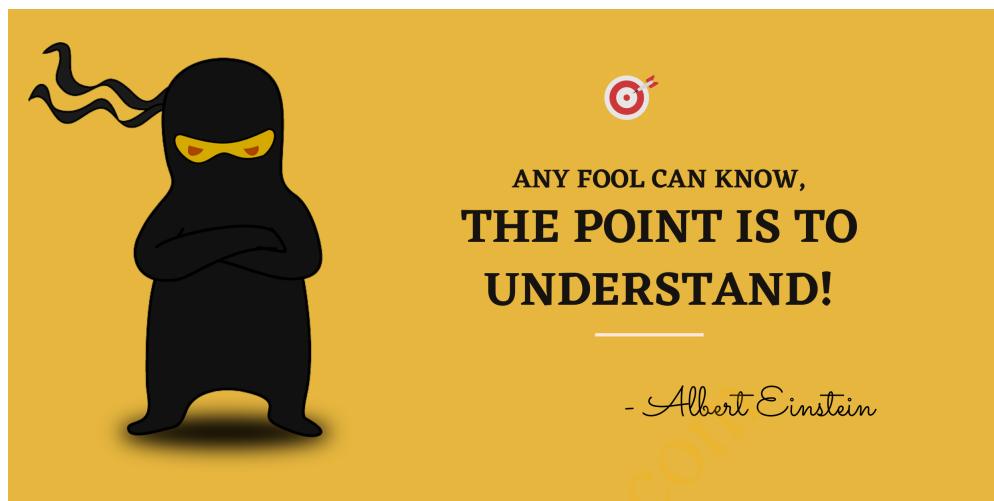
```
# visudo  
User_Alias  ADMINS = peter, bob  
Cmnd_Alias  CMND = /usr/bin/useradd, /usr/bin/userdel  
ADMINS  ALL=CMND
```

sudo command

- Sudo users can run admin commands by using **sudo** command in front of every command.

Eg:

```
$ sudo useradd testuser
```

6.1.3 Practice

- 1. Which of the following commands can be executed by sudo user with all admin access? (Select all that applies.)**
(a) useradd ravi
(b) userdel ravi
(c) sudo useradd ravi
(d) sudo userdel ravi

- 2. Which of the following file is used to edit sudo users settings?**
(a) /etc/sudo
(b) /etc/sudos
(c) /etc/sudoers
(d) /etc/su

- 3. Which of the following command is used to open sudo configuration file?**
(a) vi
(b) vim
(c) vi sudo
(d) visudo

6.2 Special permissions

In this section, you are going to learn:

1. What is SUID permission?
2. How to apply SUID permission?
3. What is SGID?
4. How to apply SGID permission?
5. What is sticky bit?
6. How to apply sticky bit permission?

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

6.2.1 SUID

What is SUID?

- SUID stands for Set User ID.
- **SUID can be applied only on command binaries.**
- **SUID is applicable only on user.**
- SUID allows user to run a **command binary** with the permissions of the **command binary owner** rather than the user who runs it.
- SUID is denoted as "s", if user has execute permission.

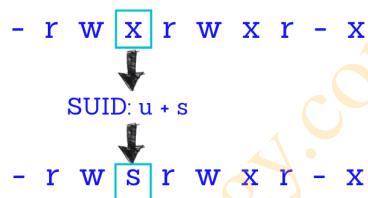


Figure 6.1: SUID permission

- SUID is denoted as "S", if no execute permission is applied for user.

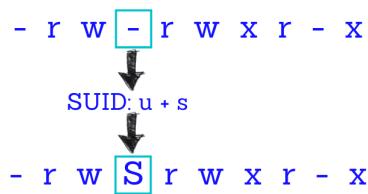


Figure 6.2: SUID permission

- Octal representation of SUID permission is "4".

Real world example for SUID

The **passwd** command has SUID applied on it.

```
$ ls -ld /usr/bin/passwd  
-rwsr-xr-x 1 root root 68208 Jul 15 2021 /usr/bin/passwd
```

Explanation:

- The **passwd** command is used to change password.
- The command tries to edit files such as **/etc/passwd**, **/etc/shadow** etc. while changing the password.
- These files **can be accessed by root & not local users**.
- The **SUID** permission allows local user to execute passwd command as root & edit **/etc/passwd**, **/etc/shadow** files.
- Hence passwd command can be used by local user to change their own password.

How to apply SUID permission?

Command:

Syntax: chmod u+s command_binary

Let's take example of **fdisk** command.

The **fdisk command cannot be executed by normal user and is owned by root user**. Is there a way to allow local user execute **fdisk** commnd?

Solution:

- Apply SUID on the **fdisk** command binary.
- SUID can be set in two ways:

```
# chmod u+s /usr/sbin/fdisk
or
# chmod 4750 /usr/sbin/fdisk

# ls -ld /usr/sbin/fdisk
-rwsr-xr-x 1 root root 153880 Jul 21 2020
/usr/sbin/fdisk
```

- Check if **normal user jack** is able to execute **fdisk** command:

```
jack@lavatech: $ fdisk -l
```

How to remove SUID permission?

Command:

Syntax: chmod u-s command_binary

Eg:

```
# chmod u-s /usr/sbin/fdisk
```

OR

lavatechtechnology.com

6.2.2 SGID

What is SGID?

- SGID stands for Set Group ID.
- SGID can be applied only on directories.
- SGID is applicable only on group ownership.
- When SGID permission is set on a directory, all the new (future) files/folders created under that directory will have the same group owner as that of the parent directory.
- Subdirectories created in future will also have SGID bit on them.
- SGID is denoted as "s" for group, if group has execute permission:



Figure 6.3: SGID permission

- SGID is denoted as "S" for group, if no execute permission is applied for group:

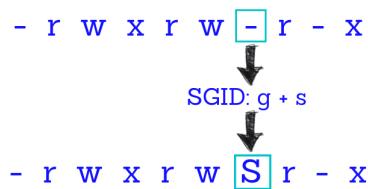


Figure 6.4: SGID permission

- Octal representation of SGID is "2".

How to apply SGID permission?

Command:

Syntax: chmod g+s directory_name

Eg: Create a folder named "/project" with below conditions:

- Owner: raj
- Group: devops
- SUID: yes

Solution:

- Create the folder:

```
# mkdir /project
```

- Assign owner as raj and group as devops:

```
# chown raj:devops /project
```

- Set SGID on /project folder:

```
# chmod g+s /project
```

or

```
# chmod 2770 /project
```

- To confirm the effect of SGID, switch to root user and create a file named "/project/sample.txt". Confirm the group ownership is set to "devops".

```
root@lavatech:/project# touch /project/sample.txt
root@lavatech:/project#
root@lavatech:/project# ls -lh /project/
total 0
-rw-r--r-- 1 root devops 0 Apr  3 23:14 sample.txt
root@lavatech:/project#
```

Figure 6.5: Sample output

How to remove SGID permission?

Command:

Syntax: chmod g-s directory_name

Eg:

```
# chmod g-s /tmp/test
or
# chmod 0750 /tmp/test
```

6.2.3 Sticky bit

- Sticky bit can be applied only on directories.
- Content of directory having sticky bit on it **can be only deleted by root or the user who created that file.**
- Sticky bit is applicable only on **other users.**
- Sticky bit is denoted as '**t**' for other, if **other** has execute permission.

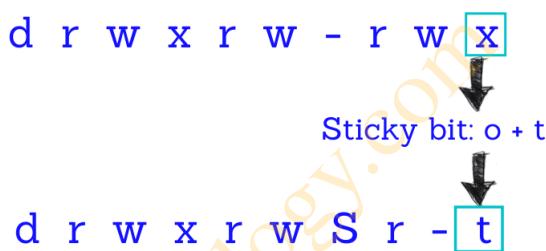


Figure 6.6: Sticky bit with execute permission

- Sticky bit is denoted as '**T**' for other, if **other** has no execute permission.

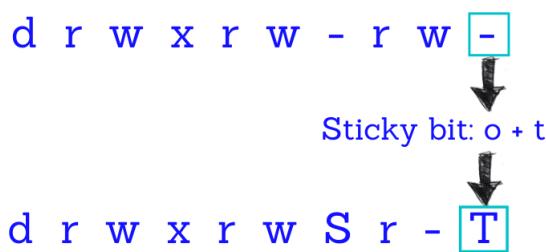


Figure 6.7: Sticky bit without execute permission

- Eg: **/tmp** directory is having sticky bit permission on it, so that only root or the owner of the files under /tmp can delete its content. **/tmp/**.
- Octal representation of sticky bit is **1**.

How to apply sticky bit permission?

Command:

Syntax: chmod o+t directory_name

Eg: Create a folder named **/opt/dump** and provide it with permission **757** and **sticky bit**.

```
# mkdir /opt/dump/  
$ chmod o+t,u=rwx,g=r-x,o+rwx /opt/dump/  
or  
$ chmod 1757 /opt/dump/
```

As local user named "raj", **create a file named /opt/dump/sample.txt**.

```
# su - raj  
$ touch /opt/dump/sample.txt
```

As local user "ravi", **delete the file /opt/dump/sample.txt** and confirm below error:

```
# su - ravi  
$ rm /opt/dump/sample.txt  
rm: cannot remove '/opt/test/sample.txt': Operation not  
permitted
```

How to remove sticky bit permission?

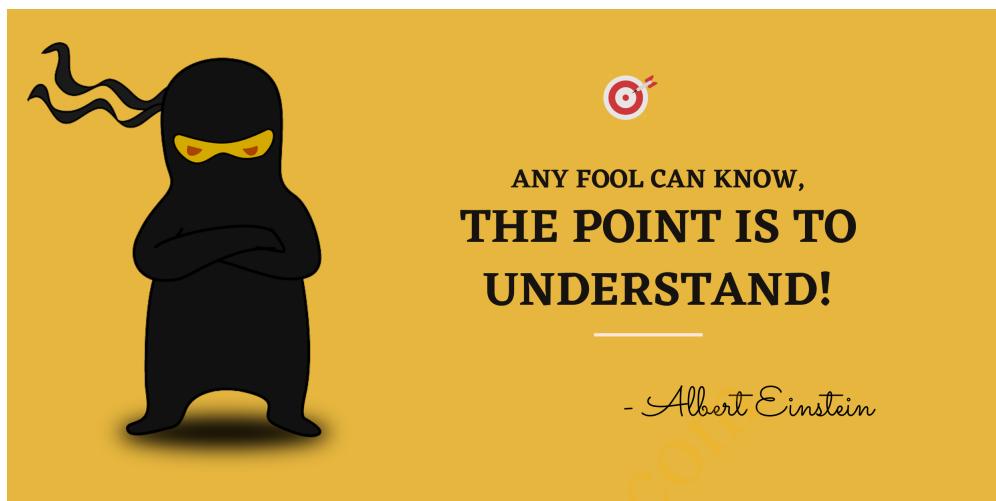
Command:

Syntax: chmod o-t directory_name

Eg:

```
$ chmod o-t /opt/dump/  
or  
$ chmod 0757 /opt/dump/
```

6.2.4 Practice



1. Select the correct octal representation for:

- user: read,write,execute
 - group: write,execute
 - other: no permission
 - SUID for user
- (a) 4730
(b) 2730
(c) 1730
(d) 0730

2. Select all statement true about SUID permission.

- (a) SUID is used to provide same user ownership to all files in folder.
- (b) SUID allows local users to run binaries with the permissions of the binary owner.
- (c) SUID provides read, write and execute permission to all users.
- (d) SUID can be applied on command binaries.

- 3. Which of the following commands have SUID applied on themselves by default? (Select all that applies.)**
(a) passwd
(b) umount
(c) su
(d) useradd
- 4. Which of the following permissions can be applied on directories? (Select all that applies.)**
(a) SUID
(b) SGID
(c) Sticky bit
(d) None of above
- 5. Select the correct octal representation for SUID, SGID and sticky bit respectively.**
(a) 4,2,1
(b) 1,2,3
(c) 1,2,4
(d) 2,1,3
- 6. Which of the following directory have sticky bit applied on it by default?**
(a) /var
(b) /mnt
(c) /tmp
(d) /run
- 7. State whether true or false. SGID can be applied on files and directories.**
(a) True
(b) False

8. State whether true or false. Sticky bit can be applied only on directories.

- (a) True
- (b) False

lavatechtechnology.com

6.3 Access Control List(ACL)

In this section, you are going to learn:

1. What is Access control list(ACL)?
2. setfacl & getfacl command
3. Default ACL
4. Removing ACL

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

6.3.1 What is ACL?

- Access Control Lists (ACL) allows a file to be **owned by several users & groups**, instead of single user & group.
- Consider a directory named **project**. You need to provide permissions to this directory as below:
 - User named **John** should have read, write and execute permission.
 - User named **Jimmy** should have read and execute permission.
 - Group named **managers** should have read, write and execute permission.
 - Group named **testers** should have read and execute permission.

This is possible by applying ACL rules.

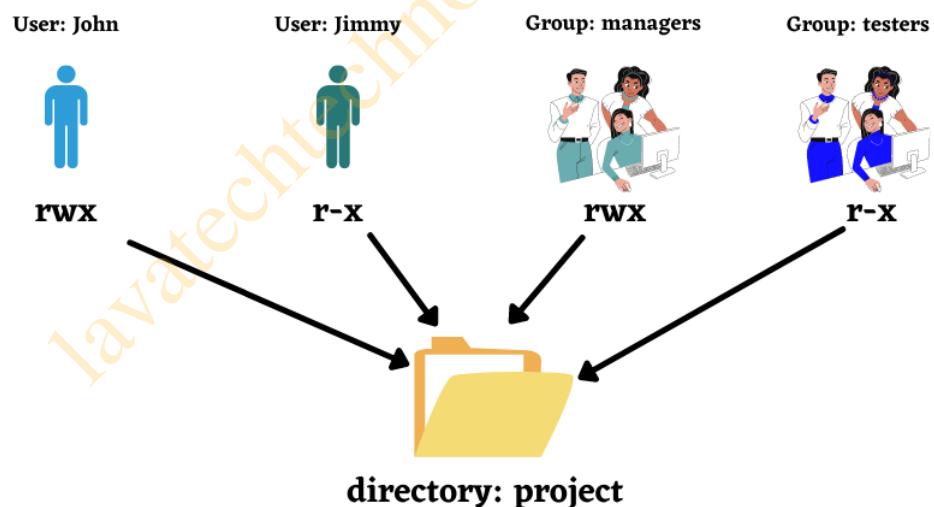


Figure 6.8: ACL example

6.3.2 ACL commands

setfacl: Used to add a new ACL rule or modify an existing rule on a file or directory.

Syntax for ACL rule on user:

```
Syntax: setfacl -m u:user_name:permissions file_or_directory
```

Syntax for ACL rule on group:

```
Syntax: setfacl -m g:group_name:permissions file_or_directory
```

Eg: Apply ACL rule on directory **project**, such as:

- User named **John** should have read, write and execute permission
- User named **Jimmy** should have read and execute permission
- Group named **managers** should have read, write and execute permission
- Group named **testers** should have read and execute permission

```
# setfacl -m u:John:rwx,u:Jimmy:r-x project  
# setfacl -m g:managers:rwx,g:testers:r-x project
```

getfacl: Used to check ACL rules applied directory or files.

Syntax: getfacl file_or_directory

Eg:

```
[root@server ~]# getfacl project
# file: project
# owner: root
# group: root
user::rwx
user:John:rwx
user:Jimmy:r-x
group::r-x
group:managers:rwx
group: testers:r-x
mask::rwx
other::r-x
```

Figure 6.9: Sample output

Default ACL rules:

- For directories, you can set ACL rules that will be assigned by defaults to files and directories created inside it.
- To do so, use the default identifier using **-R** option and **d** character in setfacl command.
- Syntax for default ACL rule for user:

Syntax: setfacl -R -m d:u:user_name:permissions directory

- Syntax for default ACL rule for group:

Syntax: setfacl -R -m d:g:group_name:permissions directory

- Eg: Allow user **peter** to read all files and directories in folder **project**:

```
# setfacl -R -m d:u:peter:rwx project
```

```
[root@server ~]# getfacl project/
# file: project/
# owner: root
# group: root
user::rwx
user:John:rwx
user:Jimmy:r-x
group::r-x
group:managers:rwx
group: testers:r-x
mask::rwx
other::r-x
default:user::rwx
default:user:peter:rwx
default:group::r-x
default:mask::rwx
default:other::r-x
```

Figure 6.10: Sample output

Remove ACL permission

- Syntax to remove ACL rule applied for user and group from file or directory:

Syntax: setfacl -x u:user_name file_or_directory

Syntax: setfacl -x g:group_name file_or_directory

Eg: Remove ACL rule for file named **secretfile** for user **john**:

```
# setfacl -x u:john secretfile
```

- Syntax to remove default ACL rule applied for user and group from a directory:

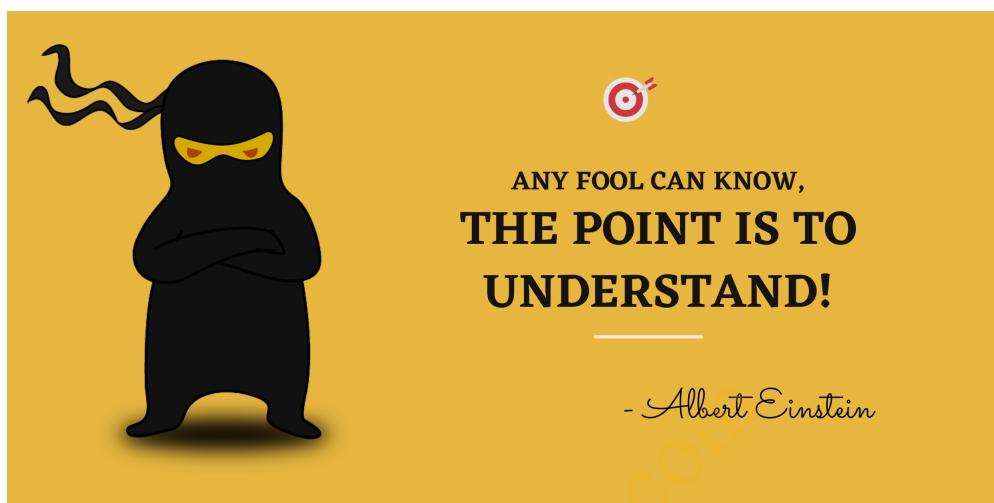
Syntax: setfacl -R -x u:user_name directory

Syntax: setfacl -R -x g:group_name directory

Eg: Remove ACL rule on **project** directory for user **john**:

```
# setfacl -R -x u:john project
```

6.3.3 Practice



1. Which of the following command is used to display ACL rule of file or directory?
 - (a) cat
 - (b) getfacl
 - (c) visudo
 - (d) setfacl
2. Select the correct syntax of setfacl command to add ACL rule.
 - (a) setfacl -m user:username:permission file_or_directory
 - (b) setfacl -d -m user:username:permission file_or_directory
 - (c) setfacl -m g:groupname:permission file_or_directory
 - (d) setfacl -x -m g:groupname:permission file_or_directory
3. Select the correct syntax of setfacl command to remove ACL rule.
 - (a) setfacl -x u:username file_or_directory
 - (b) setfacl -x g:groupname file_or_directory
 - (c) setfacl -m g:groupname:permission file_or_directory
 - (d) setfacl -R -m g:groupname:permission file_or_directory

4. Select the correct syntax of setfacl command to apply default ACL rule.

- (a) setfacl -d u:username file_or_directory
- (b) setfacl -d g:groupname file_or_directory
- (c) setfacl -R -m d:u:username:permission directory
- (d) setfacl -R -m d:g:groupname:permission directory

lavatechtechnology.com

Don't focus on the pain ->



Focus on progress ->



- Dwayne Johnson

7. Processing Commands

7.1 Text processing

In this section, you are going to learn text processing commands like:

- **find & grep**
- **head & tail**
- **more & wc**
- **sort, cut & uniq**

There will be a **small exercise** on these topics to check your knowledge.



So let's get started....

7.1.1 Commands

- **locate:** Searches for file names or file paths and returns the results instantly.

Syntax: locate filename/foldername

Eg:

```
# locate passwd
```

- **find:** Searches the directory tree containing a specific file.

Syntax: find directory_name options expression

Options with **find** command:

- **-name:** Supply an expression to search the directory for.

Syntax: find directory_name -name expression

Eg: Search for files having name "sample_file" in directory **/root:**

```
# find /root -name "sample_file"
```

Special characters in **-name** option:

- * **"?" in expression:** The question-mark is a wild card which represents 'any one character'.

```
# find /root -name "sample?"
```

- * **"*"** in expression: Represent any number of multiple characters.

```
# find /root -name "sample*"
```

- **-iname**: Perform a case-insensitive search for a given file name.

Syntax: find directory_name -iname expression

- **-user**: Find files owned by a specific user.

Syntax: find directory_name -user user_name

Eg: Find all files/directories owned by user "**root**" in **/var** directory.

```
# find /var -user root
```

- **-group**: Find files owned by a specific group.

Syntax: find directory_name -group group_name

Eg: Find all files/directories owned by group "**root**" in **/var** directory.

```
# find /var -group root
```

- **-type**: Find files of specific type like **file**, **directory**, **symbolic link** etc.

Syntax: find directory_name -type [f,d,l,s,c,b]

Eg: Find all directories under /tmp directory.

```
# find /tmp -type d
```

- **-atime**: Find files according to their access time.

Syntax: find directory_name -atime [argument]

Eg:

```
# Find files whose access time 3 days hours ago.  
$ find /tmp -atime 3  
  
# Find files whose access time 3 days hours ago and prior to that.  
$ find /tmp -atime +3  
  
# Find files whose access time between now and upto 3 days ago.  
$ find /tmp -atime -3
```

- **-perm**: Find files according to specific permission.

Syntax: find directory_name -perm argument

Eg:

```
# Find in the current directory the files having exact permissions of 644.
```

```
$ find . -perm 644
```

```
# Find in the current directory the files having either rw to user OR r to group OR r to others. Any one permission match will do.
```

```
$ find . -perm /644
```

```
# Find in the current directory the files having minimum 664 permissions.
```

```
$ find . -perm -664
```

- **grep:** Stands for "Global Regular Expression Print". Searches the file for lines containing a match to the given expression.

Syntax: grep expression file_name

Eg: Search word "shakher" in file **/etc/passwd**:

```
# grep shakher /etc/passwd
shakher:x:1000:1000::/home/shakher:/bin/bash
shakher_suman:x:1000:1000::/home/shakher_suman:/bin/bash
```

Options with **grep** command:

- **-i:** Force grep to ignore word case.

```
# grep -i shakher /etc/passwd
shakher:x:1000:1000::/home/shakher:/bin/bash
shakher_suman:x:1000:1000::/home/shakher_suman:/bin/bash
```

- **-r or -R:** Search recursively i.e. search all files under each directory for a string.

```
# grep -r "192.168.1.5" /etc/
/etc/ppp/options:# ms-wins 192.168.1.50
/etc/ppp/options:# ms-wins 192.168.1.51
```

- **-w:** Select only those lines containing matches that form whole words.

```
# grep -w "shakher" /etc/passwd
shakher:x:1000:1000::/home/shakher:/bin/bash
```

- **-c:** Count a particular word in file.

```
# grep -c "Error" logfile.txt  
4
```

- **-n**: Display line number of lines matching the word.

```
# grep -n "Error" logfile
```

Eg:

```
root@server:~# grep -n "Error" logfile  
1:Error: Server not reachable! IP address: 172.135.23.23  
3:Error: Server not reachable! IP address: 172.135.23.23  
10:Error: Server not reachable! IP address: 172.135.23.23  
14:Error: Server not reachable! IP address: 172.135.23.23
```

Figure 7.1: Sample output

- **-v**: Display only those lines that **do not** contain the given word.

```
# grep -v "nep" logfile
```

- **-l**: Displays only the file names which matches the given pattern.

```
# grep -l "main" *.java
```

Special characters in grep command:

- **^** : Matches only the lines having the starting word mentioned in the expression.

```
# grep -v "^UUID" /etc/fstab
```

- **\$** : Matches only the lines having the last word mentioned in the expression.

```
# grep -v "UUID$" /etc/fstab
```

- **head:** Display starting lines of the file. By default, displays the top 10 lines of file.

Syntax: head file_name

Eg:

```
# head /etc/passwd
```

Options for **head** command:

- n: Provide the number of lines to be displayed from the start of the file.

```
# head -n5 /etc/passwd
```

- **tail:** Display ending lines of the file. By default, displays the bottom 10 lines of file.

Syntax: tail file_name

Eg:

```
# tail /etc/passwd
```

Options for **tail** command:

- n: Provide the number of lines to be displayed from the end of the file.

```
# tail -n5 /etc/passwd
```

- **more:** Easily read a file without using an editor. Press "q" to quit reading the file.

Syntax: more file_name

Eg:

```
# more /etc/passwd
```

- **wc:** Display number of **lines, words and bytes** respectively in a file.

Syntax: wc filename

Eg:

```
$ wc myfile  
6 7 39 myfile
```

Options with **wc** command:

- **-l:** Display number of lines in a file
- **-w:** Display number of words in a file
- **-c:** Display number of characters in a file

Eg:

```
# wc -l myfile  
# wc -w myfile  
# wc -c myfile
```

- **sort:** Sorts file content by default in ascending order.

Syntax: sort file_name

Eg: Notice the file content "phonebook":

```
$ cat phonebook
Smith,Brett 5554321
Doe,John 5551234
Doe,Jane 5553214
Avery,Cory 5554321
Fogarty,Suzie 5552314
```

Let's sort the content of "phonebook" file.

```
$ sort phonebook
Avery,Cory 5554321
Doe,Jane 5553214
Doe,John 5551234
Fogarty,Suzie 5552314
Smith,Brett 5554321
```

Options with **sort** command:

- r: Sort file in reverse order.

```
$ sort -r phonebook
```

- **uniq**: Display file content by removing consecutive duplicate lines from the file.

Syntax: uniq file_name

Eg: Find all unique lines in below file:

```
# cat cities.txt
Pune
Kolhapur
Kolhapur
Pune
```

The **uniq** command is used find uniq consecutive lines:

```
# uniq cities.txt
Pune
Kolhapur
Pune
```

- **cut**: Extract a certain range of characters from a line or file.

Options with **cut** command:

- **-c**: Display specific character or range of characters.

Syntax: `cut -cn file_name`

Eg: Consider below file:

```
# cat company.data
406378:Sales:Itorre:Jan
031762:Marketing:Nassium:Jim
636496:Research:Ancholie:Mel
396082:Sales:Jucacion:Ed
```

To display the 6th characters of the file:

```
# cut -c6 company.data
8
```

Display the range of characters like 2nd to 6th character of the file:

```
# cut -c2-6 company.data
06378
31762
36496
96082
```

Display only 2nd & 6th character of the file:

```
# cut -c2,6 company.data
08
32
36
92
```

- **-f**: Specifies a field list, the line being cut is supposed to be comprised of fields
- **-d**: The ‘fields’ in the line are determined by delimiter specified by ‘-d’ option

Syntax: `cut -fn -d[delimiter] file_name`

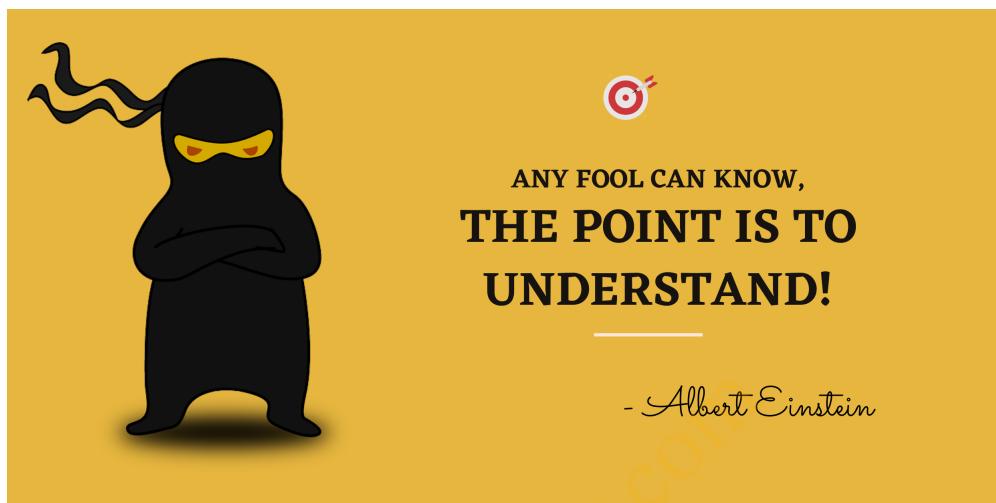
Eg: **Cut** 3rd field as per delimiter ":" -

```
# cut -d":" -f3 company.data
Itorre
Nasium
Ancholie
Jucacion
```

Eg: If you want to access multiple fields like 1st and 3rd field, you can separate them by comma:

```
# cut -d":" -f1,3 company.data
406378:Itorre
031762:Nasium
636496:Ancholie
396082:Jucacion
```

7.1.2 Practice



1. Which of the following command is used to search a file in a directory?
 - (a) grep
 - (b) sort
 - (c) cut
 - (d) find
2. Which of the following command is used to search an expression in a file?
 - (a) find
 - (b) locate
 - (c) grep
 - (d) sort
3. Which of the following are valid options of find command?
(Select all that applies.)
 - (a) -user
 - (b) -name
 - (c) -type

(d) -atime

4. Which of the following is valid options for grep command? (Select all that applies.)
- (a) -i
 - (b) -r or -R
 - (c) -v
 - (d) -n
5. Which of the following command is used to search all lines in /etc/fstab not having letter "#"?
- (a) grep -v "#" /etc/pfstab
 - (b) grep -w "#" /etc/pfstab
 - (c) grep -n "#" /etc/pfstab
 - (d) grep -i "#" /etc/pfstab
6. Which of the following command is used to display first 20 lines of /etc/passwd file?
- (a) head -2 /etc/passwd
 - (b) tail -20 /etc/passwd
 - (c) head -20 /etc/passwd
 - (d) tail -2 /etc/passwd
7. Which of the following command is used to sort all lines of /etc/passwd file in descending order?
- (a) sort -d /etc/passwd
 - (b) sort -r /etc/passwd
 - (c) sort -n /etc/passwd
 - (d) sort -R /etc/passwd

8. Which of the following command is used to count number of words in a file?

- (a) sort
- (b) wc
- (c) cut
- (d) tail

9. Which of the following command is valid to display only UID of all users from /etc/passwd file?

- (a) cut -d":" -f3 /etc/passwd
- (b) cut -d"." -f2 /etc/passwd
- (c) cut -d":" -f1 /etc/passwd
- (d) cut -d":" -f4 /etc/passwd

7.2 I/O Redirection

In this section, you are going to learn:

1. **What is standard input device & standard output device?**
2. **Output redirection**
3. **Output append operator**
4. **Input redirection**
5. **Error redirection**
6. **Error append redirection**
7. **Output & error redirection**
8. **Output & error append redirection**
9. **Redirection summary**
10. **The pipe operator**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

7.2.1 Standard input and standard output device

- Commands read input using standard input device and produce output on standard output device.
- Standard input device: **Keyboard**
- Standard output device: **Screen** or **Desktop** or **Terminal window**



Figure 7.2: Standard input and output device

7.2.2 Output redirection (>)

- Output redirection saves the output of a command in a file instead of displaying it on the screen.
- Redirection is done using the ">" (greater-than symbol).

Syntax: command > output_file

- ">" performs below operations:
 - If the file does not exist, the file will be created.
 - If the file exists, the original contents are overwritten and output of command is saved in the file.
- Eg:

```
# ls > my_files # [press enter]
```

7.2.3 Output append operator(>>)

- Append operator is similar to output redirection.
- Append redirection is done using the ">>".

Syntax: command >> output_file

- ">>" performs below operations:
 - If the file exists, it will **append** new content to the existing content.
 - If the file does not exist, ">>" operator will create it.
- Eg:

```
# uname > my_files
# echo "Hello World!" >> my_files

# Read content of my_files
```

Linux

Hello World!

lavatechtechnology.com

7.2.4 Input Redirection (<)

- Input redirection ‘<’ (less-than symbol) is used with a program which accepts user input from the keyboard.

Syntax: command < input_file

- Eg: wc command can accept file content using input redirection:

```
# wc < filename
```

7.2.5 Error redirection (2>)

- Error redirection "2>" is used to redirect only standard error to some file.

Syntax: command 2> error_input_file

- "2>" performs below operations:
 - It creates the file if it does not exists.
 - It overwrites the file if it exists.
- Eg: Redirect error "**No such file or directory**" error of ls command to some file:

```
# ls foo 2> error_file

# Display content of error_file
# cat error_file
ls: cannot access 'foo': No such file or directory
```

7.2.6 Error append redirection (2>>)

- Error append redirection **2>>** is used to redirect & append standard error to some file.

Syntax: command 2>> error_input_file

- "**2>>**" performs below operations:
 - It creates the file if it does not exists.
 - If the file exists, it will **append** new content to the existing content.
- Eg: Redirect error "**No such file or directory**" of **ls** command to some file:

```
# ls foo 2>> error_file
# ls foo 2>> error_file

# Read content of error_file
$ cat error_file

ls: cannot access 'foo': No such file or directory
ls: cannot access 'foo': No such file or directory
```

7.2.7 Output & error redirection (&>)

- Redirects output & error of command to some file.

Syntax: command &> output_error_input_file

- "&>" performs below operations:
 - It creates the file if it does not exists.
 - It overwrites the file if it exists.
- Eg: Redirect output & error of a command to some file:

```
# ls -l fin /root &> one.txt  
  
# Read content of error_file  
# cat one.txt  
ls: cannot access 'fin': No such file or directory  
/root:  
total 4  
-rw-r--r--. 1 root root 51 Apr 10 13:36 one.txt
```

7.2.8 Output & error append redirection (&>>)

- Redirects & appends output & error of command to some file.

Syntax: command &>> output_error_input_file

- "&>>" performs below operations:
 - It creates the file if it does not exists.
 - Appends the content to the file if it exists.
- Eg: Redirect & append output & error of a command to some file:

```
# ls -l fin /root &>> one.txt  
  
# Read content of error_file  
# cat one.txt  
ls: cannot access 'fin': No such file or directory  
/root:  
total 4  
-rw-r--r--. 1 root root 51 Apr 10 13:36 one.txt
```

7.2.9 Redirection Summary

| Redirection Operator | Resulting Operation |
|----------------------|---|
| command > file | stdout written to file, overwriting if file exists |
| command >> file | stdout written to file, appending if file exists |
| command < file | input read from file |
| command 2> file | stderr written to file, overwriting if file exists |
| command 2>> file | stderr written to file, appending if file exists |
| command &> file | stdout & stderr written to file, overwriting if file exists |
| command &>> file | stdout & stderr written to file, appending if file exists |

7.2.10 The pipe operator

- The "!" character is pipe operator.
- A pipe is a technique for passing output of one command to another command:



Figure 7.3: Pipe operator

Syntax: command1 | command2 | command3

Eg:

```
# who | wc -l [press Enter]
4
```

Pipelines, redirection, and tee

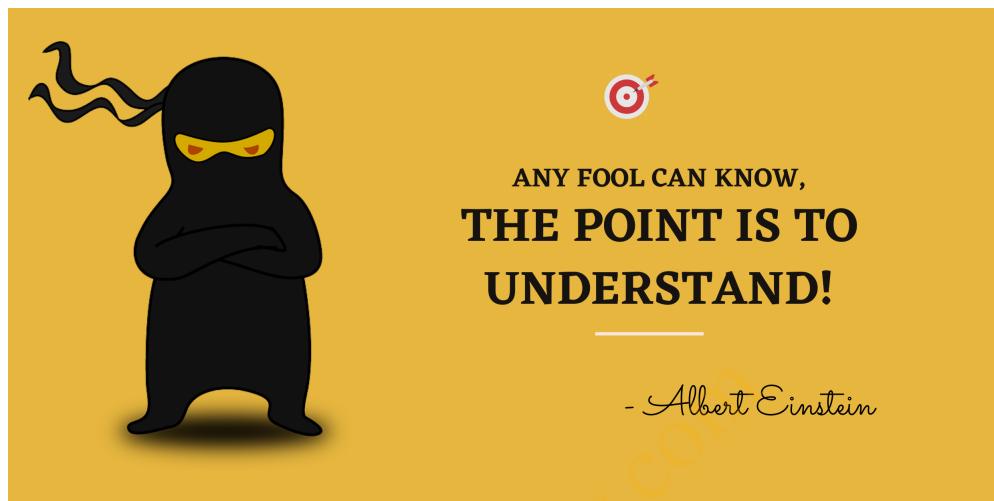
tee: In a pipeline, tee will copy its standard input to its standard output(i.e monitor screen) and will also redirect its standard output to the files named as arguments to the command.

Syntax: command1 | tee file-name

Eg:

```
# ls -t | head -n 10 | tee /tmp/ten-last-changed-files
# ls -l | tee /tmp/saved-output | less
```

7.2.11 Practice



1. Which of the following command is valid to redirect "find" command output to /tmp/data file?
 - (a) find / -type f > /tmp/data
 - (b) find / -type f < /tmp/data
 - (c) grep / -type f > /tmp/data
 - (d) locate / -type f > /tmp/data
2. Which of the following command appends the output of "echo" command to /tmp/data file?
 - (a) echo "Welcome" > /tmp/data
 - (b) echo "Welcome" >> /tmp/data
 - (c) echo "Welcome" < /tmp/data
 - (d) echo "Welcome" &> /tmp/data
3. Which of the following is a valid command to append errors of "find" command to /tmp/data?
 - (a) find / -type f 2>> /tmp/data
 - (b) find / -type f 2> /tmp/data
 - (c) find / -type f >> /tmp/data

- (d) find / -type f < /tmp/data
4. Which of the following command is used to count number of lines in "ifconfig" command?
- (a) ifconfig | sort -n
 - (b) ifconfig | wc -l
 - (c) ifconfig | wc -w
 - (d) ifconfig | sort -w
5. Which of the following are valid commands to redirect output & error to a file? (Select all that applies.)
- (a) ls -l file1 /root &> /tmp/output.txt
 - (b) ls -l file1 /root &>> /tmp/output.txt
 - (c) ls -l file1 /root 2> /tmp/output.txt
 - (d) ls -l file1 /root 2>> /tmp/output.txt

7.3 Archives & Compression

In this section, you are going to learn:

1. **zip and unzip command to compress file/directory**
2. **File compression commands: gzip, bzip2**
3. **File uncompression commands: gunzip, bunzip2**
4. **File & directory compression command: tar**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

7.3.1 Commands

Let's see some commands to archive and compress file or directory.

- **zip & unzip:**

1. **zip:** Compresses file/directory. It is available on Linux, Windows, and Mac OS.

Syntax: zip zip_file_name *

Eg:

```
# mkdir testing
# cd testing
# touch {1..10}
# mkdir -p a/b/c

# To compress testing folder, execute zip
command inside the folder:
# zip data *

# Confirm the creation of data.zip file
# ls
1 10 2 3 4 5 6 7 8 9 a data.zip
```

Note: zip command creates **data.zip** file automatically.

Options with **zip** command:

- **-r:** To zip up an entire directory (including all subdirectories):

Syntax: zip -r zip_file_name *

Eg:

```
# zip -r data *
```

2. **unzip**: Used to decompress files/directories compressed using zip utility.

Syntax: unzip argument

Eg:

```
# unzip data.zip
```

- **gzip & gunzip:**

1. **gzip:** Can compress only files. Available on Linux and Mac OS. It gives compressed files ".gz" extension.

Syntax: gzip filename

Eg:

```
# gzip file.txt
```

Note: This command replaces **file.txt** with **file.txt.gz**

2. **gunzip:** Uncompress the files with ".gz" extension.

Syntax: gunzip filename.gz

Eg:

```
# gunzip file.txt.gz
```

- **bzip2 & bunzip2:**

1. **bzip2:** Can compress only files. Available on Linux and Mac OS. It gives compressed files ".bz2" extension.

Syntax: bzip2 filename

Eg:

```
# bzip2 file.txt
```

Note: This command replaces **file.txt** with **file.txt.bz2**

2. **bunzip2:** Uncompress the files with ".bz2" extension.

Syntax: bunzip2 filename.bz2

Eg:

```
# bunzip2 file.txt.bz2
```

- **tar:** Stands for "tape archive". Used to create an archive of a directory.

Syntax: tar [options] archive-file-name.tar directory_name

Options with **tar** command:

1. **-c:** Create archive

-v: Stands for verbose. Tells tar to print all the filenames added to archive.

-f: The name of the archive Eg:

```
# tar -cvf MyProject.tar MyProject
```

Note: The **-cvf** will only archive the directory, but not compress it. To compress it further, you will need to use "**-z or -j**" options as shown below.

2. **-z:** Compress a tar archive with the gzip command. You need to add extension **".tar.gz"**

Eg:

```
# tar -cvzf MyProject.tar.gz MyProject
```

3. **-j:** Compress a tar archive with bzip2 command. You need to add extension **".tar.bz2"**.

Eg:

```
# tar -cjvf MyProject.tar.bz2 MyProject
```

To create an archive in a different directory:

```
$ tar -cjvf /tmp/MyProject.tar.bz2 MyProject
```

4. **-t**: List the contents of a tar file.

Eg:

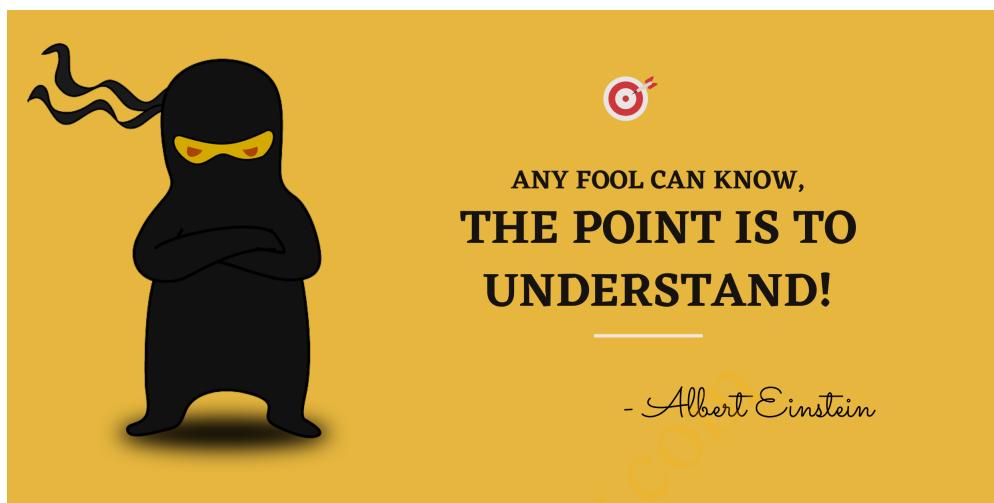
```
# tar -tvf MyProject.tar  
# tar -tzvf MyProject.tar.gz  
# tar -tjvf MyProject.tar.bz2
```

5. **-x**: Extract the contents of a tar archive.

Eg:

```
# tar -xvf MyProject.tar  
# tar -xzvf MyProject.tar.gz  
# tar -xjvf MyProject.tar.bz2
```

7.3.2 Practice



1. Which of the following is valid command to archive & compress a directory using gzip?
 - (a) tar -cjvf sample.tar.gz sample_directory
 - (b) tar -czvf sample.tar.gz sample_directory
 - (c) tar -clvf sample.tar.gz sample_directory
 - (d) tar -ckvf sample.tar.gz sample_directory
2. Which of the following command can be used to compress file/directory in Linux, Unix, Windows and Mac OS?
 - (a) tar
 - (b) bzip2
 - (c) gzip
 - (d) zip
3. Which of the following command is valid to compress a file in Linux?
 - (a) gzip filename
 - (b) gzip filename.gz
 - (c) gunzip filename

- (d) gunzip filename.gz
4. Which of the following command is valid to uncompress a bzip2 compressed file in Linux?
- (a) bunzip2 filename
 - (b) bzip2 filename.bz
 - (c) bunzip2 filename.bz2
 - (d) gunzip filename.bz2
5. State whether true or false. gzip and bzip2 command replaces file with file.gz and file.bz2 respectively?
- (a) True
 - (b) False
6. Which of the following are valid options of tar command? (Select all that applies.)
- (a) -c
 - (b) -v
 - (c) -f
 - (d) -z
7. Which of the following is valid command to display content of tar archive?
- (a) tar -tvf sample.tar
 - (b) tar -lvf sample.tar
 - (c) tar -nfv sample.tar
 - (d) tar -pvf sample.tar
8. Which of the following is valid command to extract a tar file? (Select all that applies.)
- (a) tar -xvf sample.tar

- (b) tar -xzvf sample.tar.gz
- (c) tar -xjvf sample.tar.bz2
- (d) tar -xvf sample.tar.bz2

lavatechtechnology.com

8.1 Introduction to partition

In this section, you are going to learn:

1. **What is a hard disk drive (HDD) & partition?**
2. **HDD devices naming in Linux**
3. **Commands to check HDD**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

8.1.1 What is Hard Disk Drive (HDD) & partition?

- A **hard disk drive (HDD) or hard drive**, is a data storage device that stores and retrieves digital data in computer.
- HDD brands in market: Seagate, Western Digital, Hitachi etc.



Figure 8.1: Hard disk

- **Partitioning** means to divide a HDD into many logical drives.

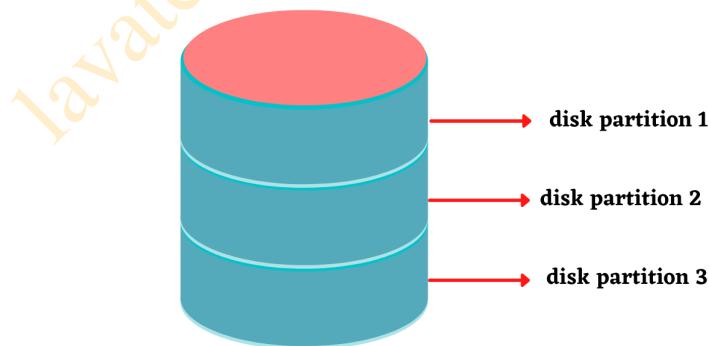


Figure 8.2: Hard disk partitions

8.1.2 HDD devices naming in Linux

- In Linux, both HDD and partitions are represented as **block device files**.
- Block device files are located under **/dev**.
- Command to find all block device files under /dev:

```
# find /dev -type b
```

Disk drives naming

- **HDD names starting with "sd":**
 1. The **IDE/SATA/SCSI type of disk drive** are represented with name starting from "sd".
 2. Eg: **/dev/sda, /dev/sdb, /dev/sdc** and so on.
- **HDD names starting with "vd":**
 1. The **paravirtualized disk driver** are represented with name starting from "vd".
 2. Eg: **/dev/vda, /dev/vdb, /dev/vdc** and so on.

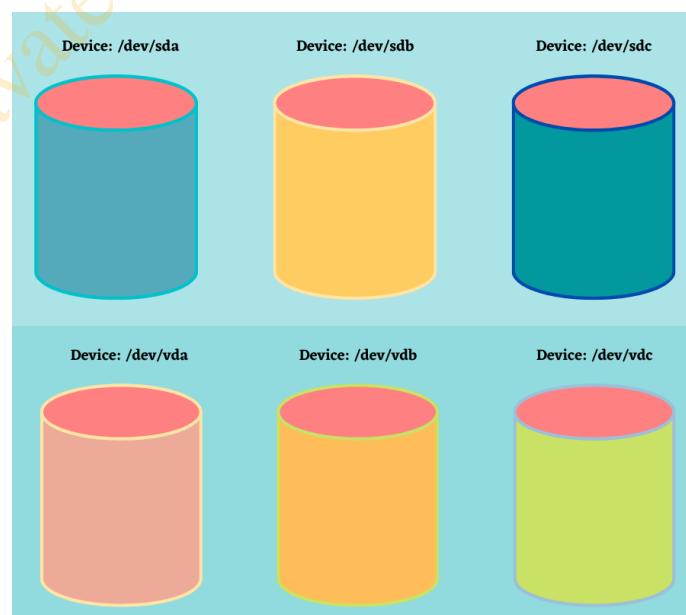


Figure 8.3: Hard disk partitions

Partition Names

- The partitions are represented as numbers at the end of the HDD names.
- Eg: For HDD "/dev/sda", partitions will be numbered as **/dev/sda1**, **/dev/sda2** and so on.

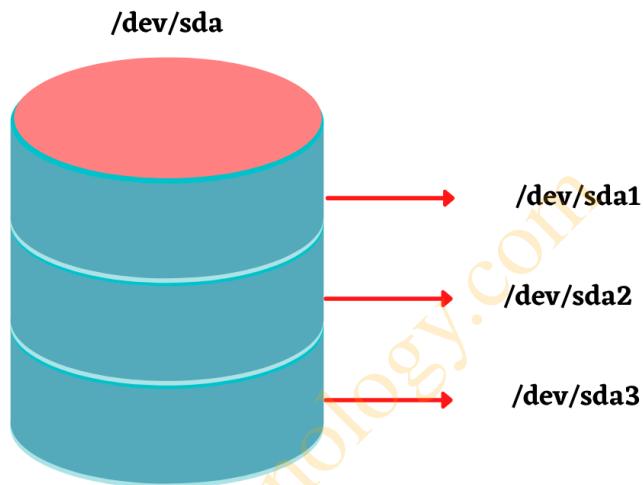


Figure 8.4: Partition names

8.1.3 Commands to check HDD

- **lsblk:** Lists information about all available block devices.

Syntax: lsblk

Eg:

```
[root@rhel8 ~]# lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda        8:0    0 128G  0 disk
└─sda1     8:1    0   1G  0 part /boot
└─sda2     8:2    0 127G  0 part
  └─rhel_rhel8-root 253:0    0 70G  0 lvm   /
  └─rhel_rhel8-swap 253:1    0 2.1G 0 lvm   [SWAP]
sdb        8:16   0   8G  0 disk
```

Figure 8.5: lsblk command output

- **fdisk -l:** Display more detailed information about all disk drives.

Syntax: fdisk -l

Eg:

- **du:** Estimate file space usage.

Syntax: du [option] [file/folder]

Eg:

```
# du /home
```

Options with **du** command:

- **-sh:** Display only a total for each argument in human readable format.

Syntax: du -sh [folder/file]

```
[root@rhel8 ~]# fdisk -l
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/sda: 128 GiB, 137438953472 bytes, 268435456 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xdf46eef

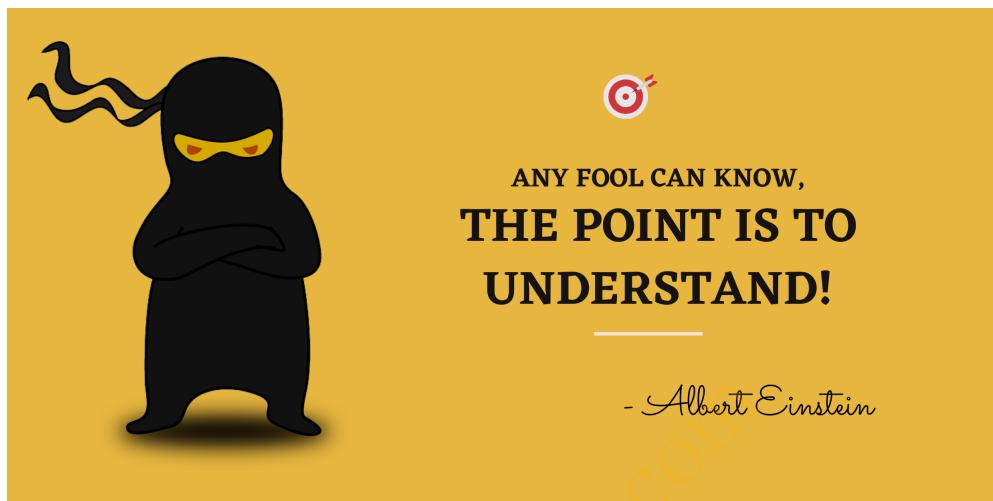
Device      Boot   Start     End   Sectors  Size Id Type
/dev/sda1    *      2048 2099199 2097152   1G 83 Linux
/dev/sda2        2099200 268435455 266336256 127G 8e Linux LVM
```

Figure 8.6: "fdisk -l" command output

Eg:

```
# du -sh /home
16K  /home
```

8.1.4 Practice



- 1. What is a hard disk drive(HDD)? (Select all that applies.)**
 - (a) Data storage device used to store and retrieves digital data.
 - (b) Place to store and access data on a short-term basis.
 - (c) Regulates and integrates the computer's operations.
 - (d) Short term memory where data is stored as the processor needs it.
- 2. What does partitioning of HDD mean? (Select all that applies.)**
 - (a) Applying filesystem on HDD.
 - (b) Splitting a large hard disk into multiple drives.
 - (c) Dividing HDD into logical drives.
 - (d) Converting a HDD into file and directory to make it useable.
- 3. Where are all HDD stored in Linux directory structure?**
 - (a) /mnt
 - (b) /dev
 - (c) /tmp
 - (d) /root
- 4. Which of the following are correct partition names? (Select all that applies.)**

- (a) Partition of /dev/sda can be /dev/sda1
 - (b) Partition of /dev/sdb can be /dev/vdb1
 - (c) Partition of /dev/sdc can be /dev/sdc5
 - (d) Partition of /dev/vda can be /dev/vda5
5. Which of the following command is used to display all block devices of computer? (Select all that applies.)
- (a) display
 - (b) lsblk
 - (c) free
 - (d) fdisk

8.2 Partition types

In this section, you are going to learn:

1. **What is MBR & partition table?**
2. **Types of partition:**
 - Primary partition
 - Extended partition
 - Logical partition
3. **What is filesystem? How to implement it on partition?**
4. **Temporary & permanent mounting**
5. **Swap partition**
6. **Setting up swap partition**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

8.2.1 Partition table in MBR

What is an MBR?

- MBR stands for **Master Boot Recorder**.
- MBR is the first **512 bytes** of HDD.
- MBR consists of:
 - Bootloader (446 bytes in size) - More on this in chapter 17 under section 17.1.4.
 - **Partition Table** (64 bytes in size) - Stores entry of maximum 4 partitions (16 byte entry for each partition).
 - Magic number (2 bytes in size) - More on this in chapter 17 under section 17.1.3.

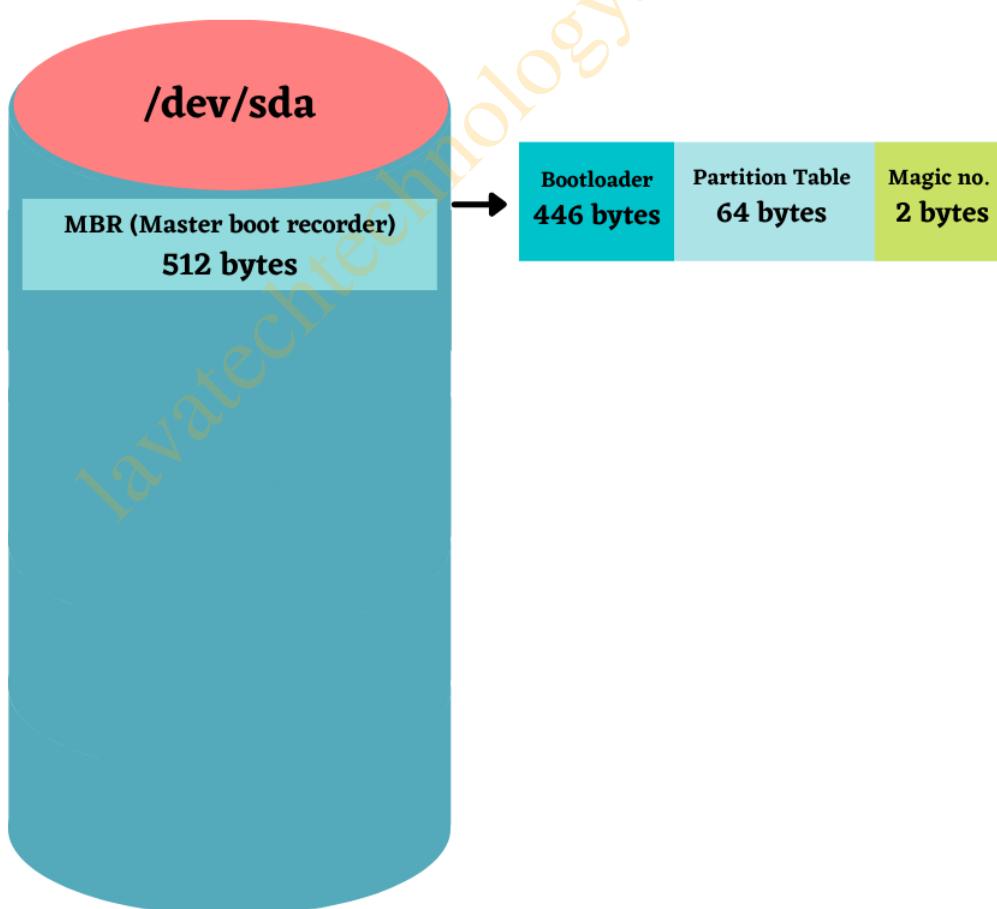


Figure 8.7: MBR

8.2.2 Primary partitions

- The partition table in MBR can hold only **4 entries**.
- Hence, there can be only **four partitions in hard disks**.
- These 4 partitions are called primary partitions.
- OS can be installed on the primary partition.

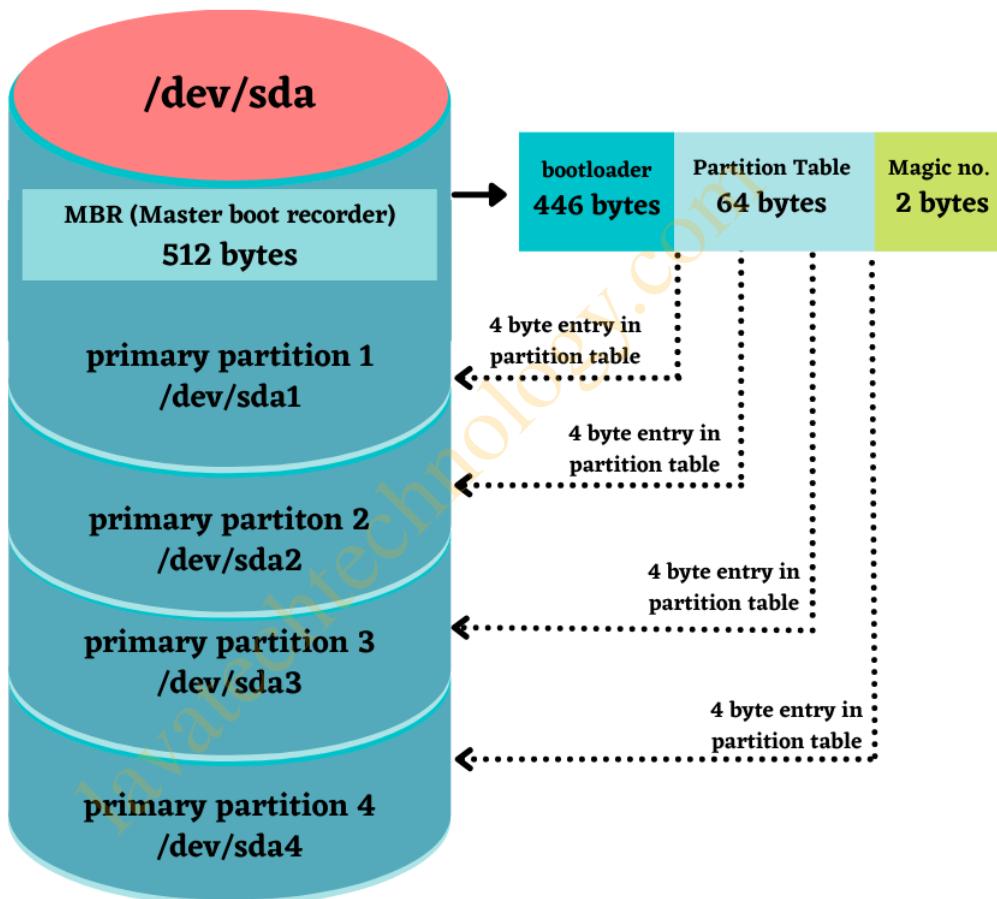


Figure 8.8: Primary partitions

Command to create primary partition

fdisk: Used to change the partition table.

Syntax: fdisk device_name

Options for **fdisk** command:

- **p:** Print the partition table
- **n:** Create a new partition
- **w:** Write the new partition table and exit

Eg:

```
[root@rhel8 ~]# fdisk /dev/sdb <- Command to create partition on /dev/sdb
Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x68ee4a0a.

Command (m for help): p <- Display partition table
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x68ee4a0a

Command (m for help): n <- Create new partition
Partition type
  p  primary (0 primary, 0 extended, 4 free)
  e  extended (container for logical partitions)
Select (default p): p <- Create primary partition
Partition number (1-4, default 1): 1 <- Select partition number
First sector (2048-16777215, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-16777215, default 16777215): +1G <- Type partition size

Created a new partition 1 of type 'Linux' and of size 1 GiB.

Command (m for help): p <- Display partition table
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x68ee4a0a

Device      Boot Start   End Sectors Size Id Type
/dev/sdb1        2048 2099199 2078752    1G 83 Linux

Command (m for help): w <- The partition table has been altered.
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Figure 8.9: Creating primary partitions

8.2.3 Extended partition

- Although there can only be **four primary partitions**, it is possible to create additional partitions.
- This is possible by creating an **extended partition**.
- An extended partition is divided up to create more partitions.

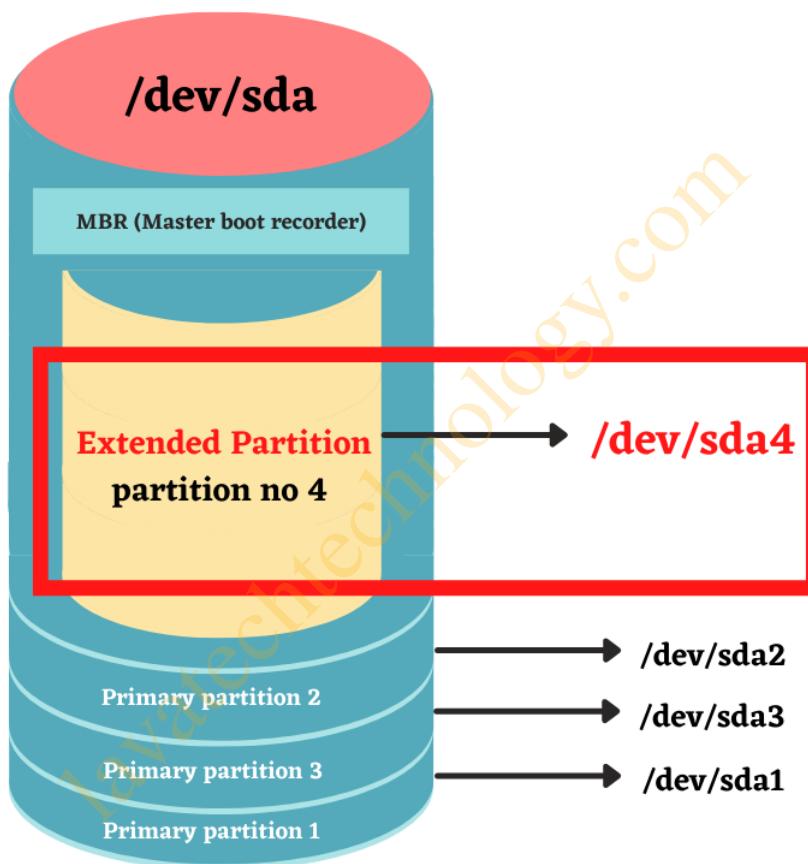


Figure 8.10: Extended partitions

Note: Out of 4 primary partition, there can be one and only one extended partition.

fdisk command option to create extended partition

- e: Create an extended partition

Eg:

```
[root@rhel8 ~]# fdisk /dev/sdb <-- Command to create extended partition on /dev/sdb
Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): n <-- Create new partition by typing "n"
Partition type
  p  primary (1 primary, 0 extended, 3 free)
  e  extended (container for logical partitions)
Select (default p): e <-- Create extended partition by typing "e"
Partition number (2-4, default 2): 3 <-- Type partition number
First sector (2099200-16777215, default 2099200):
Last sector, +sectors or +size{K,M,G,T,P} (2099200-16777215, default 16777215): +5G <-- Type partition size
Created a new partition 3 of type 'Extended' and of size 5 GiB.

Command (m for help): p <-- Display partition table by typing "p"
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x68ee4a0a

Device      Boot   Start     End  Sectors  Size Id Type
/dev/sdb1        2048 2099199 2097152   1G 83 Linux
/dev/sdb3    2099200 12584959 10485760   5G  5 Extended

Command (m for help): w <-- Save & quit by typing "w"
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Figure 8.11: Creating extended partitions

8.2.4 Logical partitions

- Logical partitions are partitions created inside the extended partition.

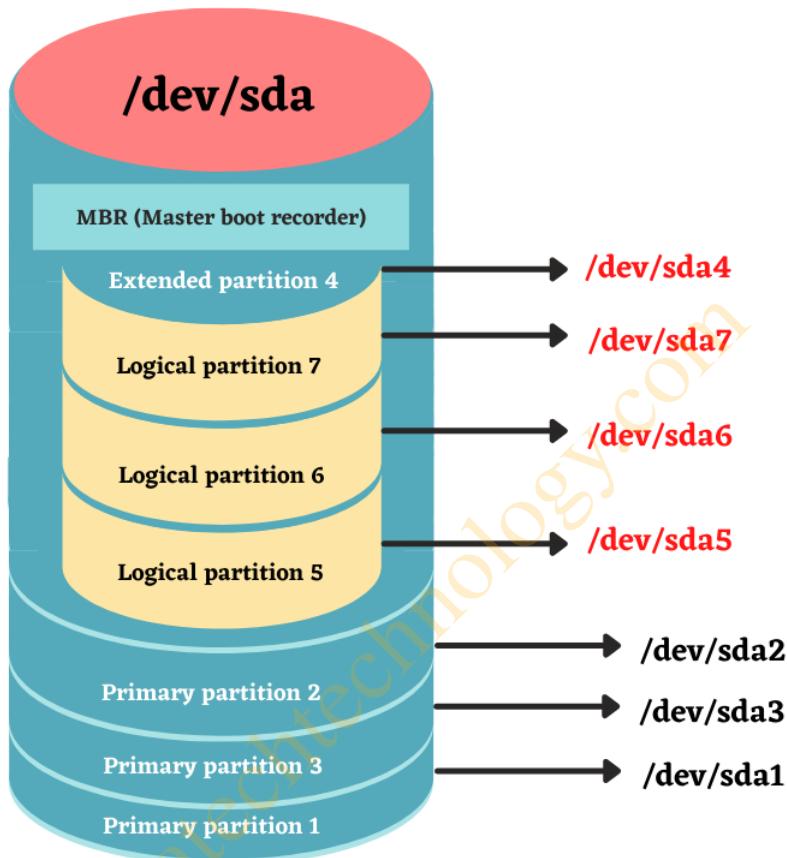


Figure 8.12: Logical partitions

Note:

- Number of logical partitions is unlimited
- However, Linux imposes limit on the total number of partitions on a drive
- There can be total 15 partitions on an SCSI disk and 63 total on an IDE disk

fdisk command option to create logical partition

- I: Create a logical partition

Eg:

```
[root@rhel8 ~]# fdisk /dev/sdb <-- Command to create partition on /dev/sdb
Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): n <-- Create new partition by typing "n"
Partition type
  p  primary (1 primary, 1 extended, 2 free)
  l  logical (numbered from 5)
Select (default p): l <-- Create new partition by typing "l"

Adding logical partition 5 <-- Partition number is auto-selected
First sector (2101248-12584959, default 2101248):
Last sector, +sectors or +size[K,M,G,T,P] (2101248-12584959, default 12584959): +2G <-- Type partition size
Created a new partition 5 of type 'Linux' and of size 2 GiB.

Command (m for help): p <-- Display partition table
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x68ee4a0a

Device      Boot   Start     End   Sectors  Size Id Type
/dev/sdb1        2048 2099199  2097152   1G 83 Linux
/dev/sdb3        209920 12584959 10485760   5G  5 Extended
/dev/sdb5        2101248 6295551  4194304   2G 83 Linux <-- Notice logical partition entry

Command (m for help): w <-- Save & quit by typing "w"
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Figure 8.13: Creating logical partition

fdisk command option to delete any partition

- **d:** Delete partition

Eg:

```
[root@rhel8 ~]# fdisk /dev/sdb    <-- Command to edit /dev/sdb
Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): d    <-- Delete a partition by typing "d"
Partition number (1,3,5, default 5) 5    <-- Select the partition to delete

Partition 5 has been deleted.

Command (m for help): p    <-- Display partition table by typing "p"
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x68ee4a0a

Device      Boot   Start     End   Sectors  Size Id Type
/dev/sdb1        2048 2099199 2097152   1G 83 Linux
/dev/sdb3    2099200 12584959 10485760   5G  5 Extended

Command (m for help): w    <-- Save & quit by typing "w"
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Figure 8.14: Deleting partition

8.2.5 Standard formatting filesystems

- The organizational structure inside a partition is called **filesystem**.
- Once a partition is created, it should be given a **filesystem** to make the partition ready to use.
- The first Linux operating systems used the **extended (i.e ext) filesystem**.

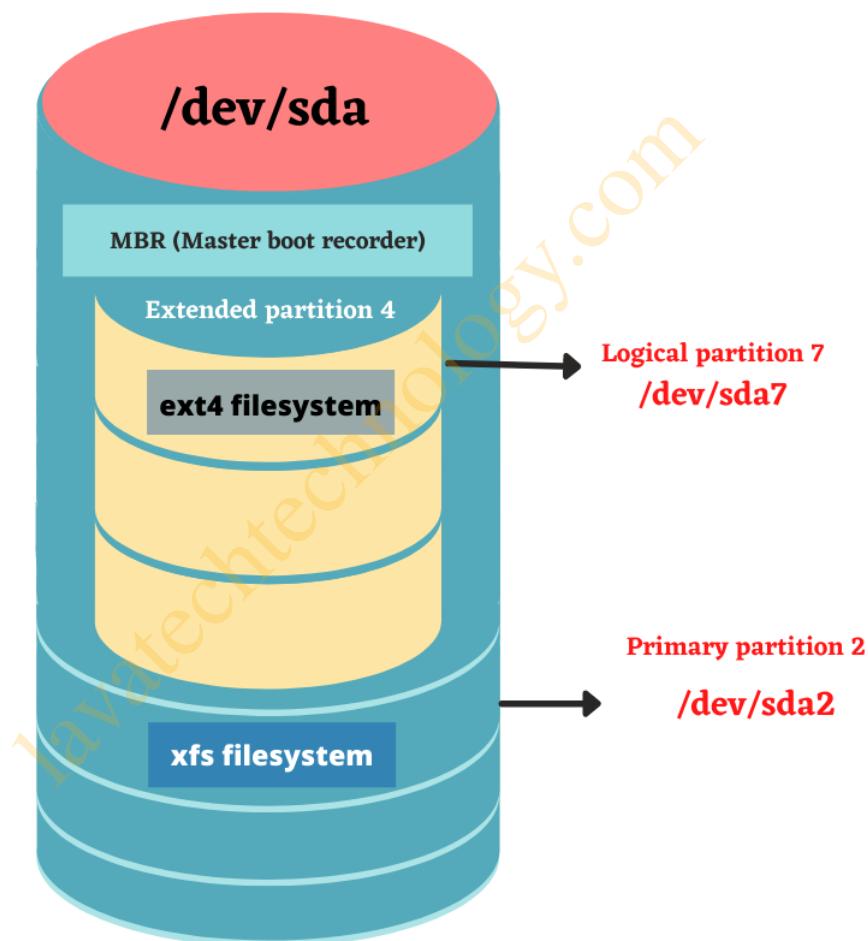


Figure 8.15: Filesystem

Note: Without applying filesystem, the partitions of HDD should not be used.

Filesystem types

| Type | Filesystem description |
|------|--|
| ext | <ul style="list-style-type: none">• First Linux filesystem |
| ext2 | <ul style="list-style-type: none">• Second extended filesystem• Maximum size of partition can be 32TB• Maximum file size can be 2TB |
| ext3 | <ul style="list-style-type: none">• Third extended filesystem• Maximum size of partition can be 32TB• Maximum file size can be 2TB• Journaling option available |
| ext4 | <ul style="list-style-type: none">• Fourth extended filesystem• Maximum size of partition can be 1PB• Maximum file size can be 16TB• Journaling option available• Defragmentation option available |
| xfs | <ul style="list-style-type: none">• High-performance 64-bit journaling filesystem• Maximum file size and partition size can be 8EB (exbibytes)• Maximum number of files that can be created are 264 |

What is journaling filesystem?

- Journaling provides fault tolerance in file systems.
- It keeps track of all changes in a log before saving it to the HDD.
- Journaling helps in recovering system crashes and power failures.
- Permanent data loss is avoided.

What is defragmentation in filesystem?

- Defragmentation makes sure unused space on the hard disk is all together.

Command to apply filesystem on a partition

mkfs: Stands for **Make Filesystem**. It applies a filesystem on a partition of HDD, USB drive, etc.

Syntax: mkfs -t filesystem_name device_name

Eg: Apply "xfs" filesystem on partition /dev/sdb1:

```
# mkfs -t xfs /dev/sdb1
```

Filesystem UUID

Once you have applied a filesystem to a partition, it is allotted with a **universally unique identifier** called as **UUID**.

Command to check **UUID** of all filesystem:

Syntax: blkid

Eg:

```
# blkid  
/dev/sdb1: PARTUUID="4f7c2ba4-01"  
/dev/sda1: UUID="cd73be54-75ea-4705" TYPE="ext4"  
/dev/sda2: UUID="7db79e4a-5144-43c4" TYPE="swap"
```

8.2.6 Mounting Partitions

Mounting a partition means attaching it to some directory so that it can be used.

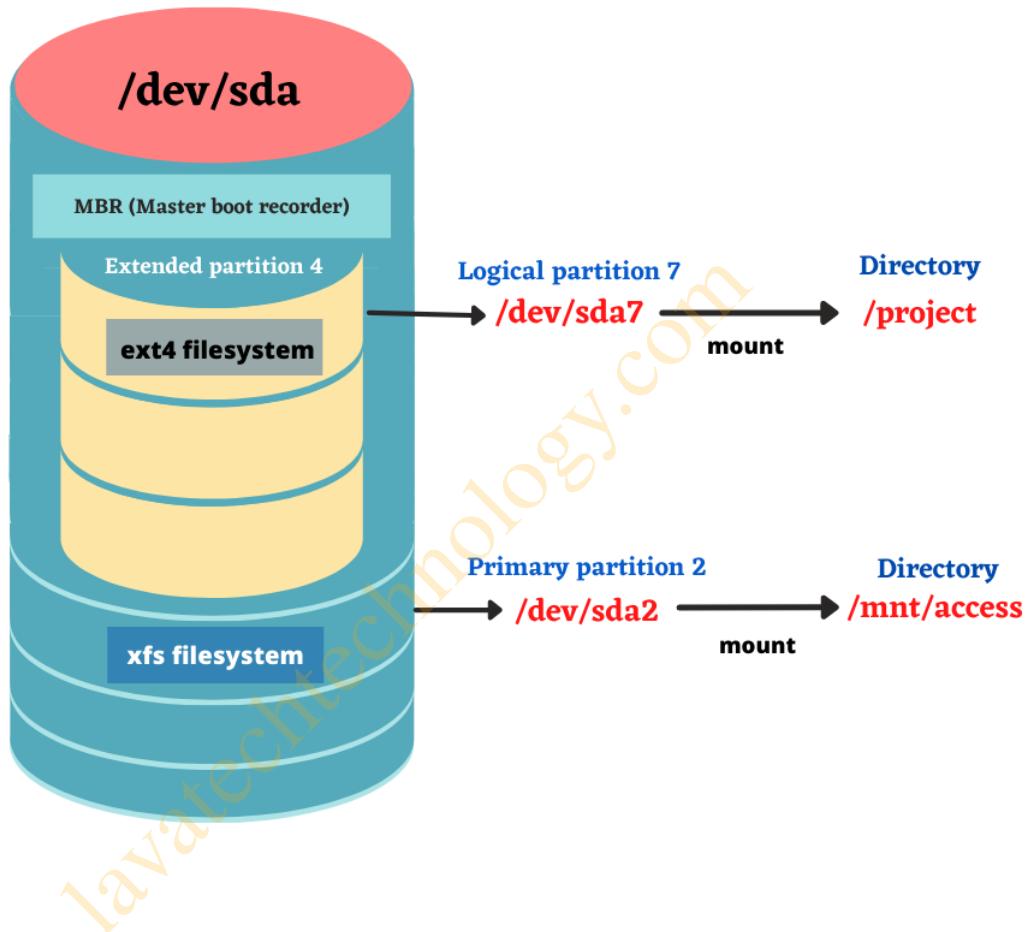


Figure 8.16: Mounting filesystem

Command to mount a partition

- Mount a linux partition:

Syntax: `mount device_name directory_name`

- Check mounted partition:

Syntax: `df -Th`

- Eg: Create a directory to mount a partition -

```
# mkdir /mnt/access
```

Mount `/dev/sda2` partition on `/mnt/access` directory -

```
# mount /dev/sda2 /mnt/access
```

Check mounted partition:

```
# df -Th
```

Note: Mounting done using **mount** command is **temporary**. After system reboot, the partition and directory will get dissociated. This will cause partition to be **unmounted**.

8.2.7 Permanent mounting:/etc/fstab

- Information about your local and remote mounted filesystems is stored in **/etc/fstab**.
- When the system boots, the entries in **/etc/fstab** are read and partitions are mounted permanently.
- This file consists of 6 important entries as shown below:

| Field 1 Filesystem | Field 2 Mount point | Field 3 Filesystem type | Field 4 Options | Field 5 Filesystem dump | Field 6 Filesystem check |
|---|---|--|---------------------------|-----------------------------------|------------------------------------|
| /dev/sda2 | /access | xfs | defaults | 0 | 1 |
| Can be: • Device name • UUID • Remote IP | Can be: • ext • ext2 • ext3 • ext4 • vfat • nfs | Can be: • ro • rw • noauto • dev • exec • auto | Can be: • 0 • 1 | Can be: • 0 • 1 • 2 | |

Figure 8.17: Sample /etc/fstab entries

- Command to mount all entries from **/etc/fstab** manually:

```
# mount -a
```

- Command to know more about **/etc/fstab** file, check it's manual page:

```
# man fstab
```

Understanding 6 important entries of /etc/fstab

| Filed Name | Description |
|-----------------|---|
| Label | Label of device to be mounted. Can be: <ul style="list-style-type: none"> • Device name • UUID • Remote IP address |
| Mount point | The directory where the filesystem will be mounted |
| Filesystem type | Eg: ext, ext2, ext3, xfs, vfat, swap etc. |
| Mount options | Permissions while mounting filesystem. Basic filesystem options are: <ul style="list-style-type: none"> • defaults: use default options like rw, uid, dev, exec, auto, nouser, and async. • noauto: means do not mount when "mount -a" is given (e.g., at boot time) • user: allow a user to mount • owner: allow device owner to mount • nofail: do not report errors for this device if it does not exist |

To be continued on next page...

Continue..

| Filed Name | Description |
|------------------------|---|
| Dump Value | <p>Value either 0 or 1</p> <ul style="list-style-type: none">• 0 means no backup• 1 means dump utility backup of a partition <p>This is an outdated backup method and should NOT be used.</p> |
| Filesystem check order | <p>Value either 0, 1 or 2</p> <ul style="list-style-type: none">• 0 means not to check filesystem during the Linux boot process.• The number 1 & 2 determines the order that filesystems are checked by fsck during the boot process.• The root directory (/) filesystem should be set to 1, and other local filesystems should be set to 2. |

8.2.8 Unmounting filesystem

- **lsof:**

- Lists all open files and the process accessing them in the mounted directory.
- It is useful to identify which processes currently prevent the file system from successful unmounting.

Syntax: lsof directory-name

Eg:

```
# lsof /project
COMMAND   PID   USER   FD      TYPE   DEVICE
SIZE/OFF NODE   NAME
bash    5514  jack   cwd   DIR    259,6  4096   2  /project
```

- **umount:** Unmounts the filesystem.

Syntax: umount directory-name

Eg:

```
# umount /project
```

8.2.9 Swap partition

- The swap partition serves as overflow space for your RAM.
- If your RAM fills up completely, applications will use the **swap partition rather than RAM**.

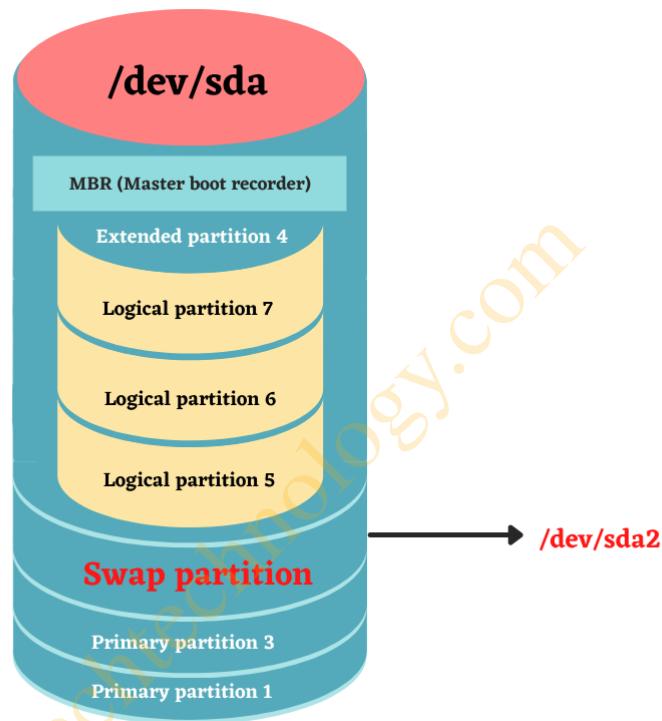


Figure 8.18: Swap partitions

Recommended swap partition size

| Amount of RAM installed in system | Recommended swap space |
|-----------------------------------|------------------------|
| 2GB | 2x RAM |
| 2GB – 8GB | = RAM |
| 8GB – 64GB | 4G to 0.5x RAM |
| >64GB | Minimum 4GB |

fdisk command option to create a swap partition

- **t:** Change type of partition.

Eg:

```
[root@rhel8 ~]# fdisk /dev/sdb] <-- Command to creation partition on /dev/sdb
Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): n <-- Create new partition by typing "n"
Partition type
  p  primary (1 primary, 1 extended, 2 free)
  l  logical (numbered from 5)
Select (default p): p <-- Create new primary partition by typing "p"
Partition number (2,4, default 2): 2 <-- Select partition number
First sector (12584960-16777215, default 12584960):
Last sector, +sectors or +size[K,M,G,T,P] (12584960-16777215, default 16777215): +1G
Created a new partition 2 of type 'Linux' and of size 1 GiB. Type partition size
Command (m for help): t <-- Change the type of partition by typing "t"
Partition number (1-3,5, default 5) 2 <-- Select partition number
Hex code (type L to list all codes): 82 <-- Type "82" to change partition type to swap
Changed type of partition 'Linux' to 'Linux swap / Solaris'.

Command (m for help): p <-- Display partition table
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x68ee4a0a

Device      Boot   Start     End   Sectors  Size Id Type
/dev/sdb1        2048 2099199 2097152  1G 83 Linux
/dev/sdb2    12584960 14682111 2097152  1G 82 Linux swap / Solaris <-- Notice the entry
/dev/sdb3    2099200 12584959 10485760 5G 5 Extended
/dev/sdb5    2101248 6295551 4194304  2G 83 Linux

Partition table entries are not in disk order.

Command (m for help): w <-- Save & quit by typing "w"
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Figure 8.19: Creating swap partitions

8.2.10 Setting up swap space

After the swap partition is created, you will need to perform below steps to proceed:

- Step 1: Setup partition as swap area using "**mkswap**" command.

Syntax: mkswap partition_name

Eg:

```
# mkswap /dev/sdb5
```

- Step 2: Mount swap partition permanently. Add the partition in "**/etc/fstab**" file.

Sample entry:

```
# cat /etc/fstab  
/dev/hdb2 swap swap defaults 0 0
```

- Step 3: Enable the swap partition for usage using "**swapon**" command.

Syntax: swapon partition_name

Eg:

```
# swapon /dev/sdb5
```

Options with **swapon** command:

- **-s**: Verify swap area is available for your use

```
# swapon -s
```

- **-a:** Enable all swap partitions for use in **/etc/fstab** file

```
# swapon -a
```

lavatechtechnology.com

- You can also disable swap partition in **/etc/fstab** file if you don't want to use swap space.

Syntax: swapoff partition_name

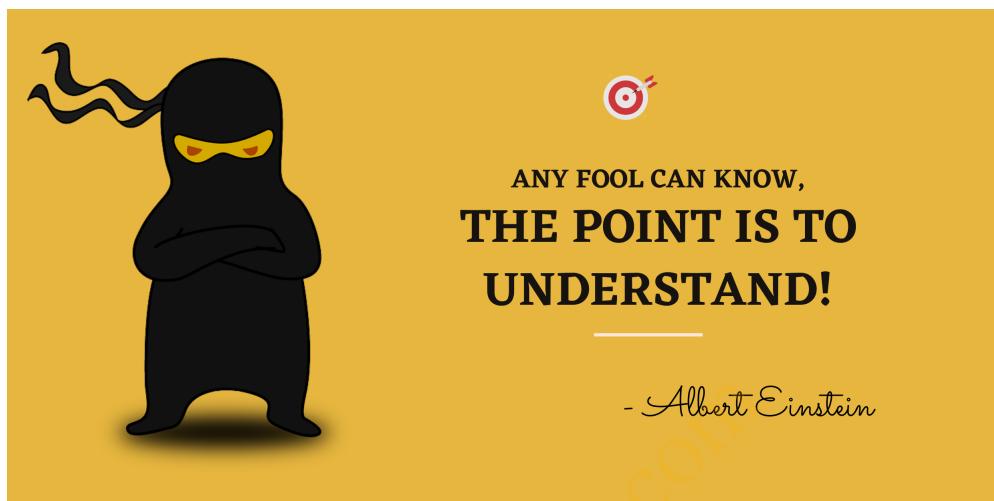
Eg:

```
# swapoff /dev/sdb5
```

Options with **swapoff** command:

- **-a**: Disable all swap partitions for use in **/etc/fstab** file

```
# swapoff -a
```

8.2.11 Practice

- 1. What is full form of MBR?**
 - (a) Master Bost Reader
 - (b) Matter Boot Reader
 - (c) Master Boot Recorder
 - (d) Master Beat Reader

- 2. What is the size of MBR?**
 - (a) 255 bytes
 - (b) 64 bytes
 - (c) 446 bytes
 - (d) 512 bytes

- 3. What is the maximum number of primary partition that can be created?**
 - (a) 2
 - (b) 8
 - (c) 16
 - (d) 4

4. What is the maximum number of extended partition that can be created?
 - (a) 2
 - (b) 8
 - (c) 1
 - (d) 4

5. State whether true or false. You can use a partition without applying a filesystem on it.
 - (a) True
 - (b) False

6. Which of the following are valid filesystem in Linux? (Select all that applies.)
 - (a) ext
 - (b) xfs
 - (c) ext2
 - (d) ext3

7. Which of the following command is used to apply filesystem on any partition?
 - (a) blkid
 - (b) mkfs
 - (c) fdisk
 - (d) lsblk

8. Which of the following command is used to display UUID of any partition with filesystem applied on it?
 - (a) blkid
 - (b) mkfs
 - (c) fdisk
 - (d) mount

9. Which of the following command is used to mount a partition?
- (a) blkid
 - (b) mkfs
 - (c) fdisk
 - (d) mount
10. Which of the following file is used for permanent mounting of a partition?
- (a) /etc/tab
 - (b) /etc/sudoers
 - (c) /etc/fstab
 - (d) /etc/passwd
11. Which of the following partition is used when the RAM fills up?
- (a) primary partition
 - (b) logical partition
 - (c) extended partition
 - (d) swap partition
12. Which of the following command is used to setup swap partition?
- (a) mkfs
 - (b) mkswap
 - (c) swapon
 - (d) blkid
13. Which of the following command is used to disable swap partition?
- (a) swapoff
 - (b) mkswap
 - (c) swapon
 - (d) blkid

9.1 Introduction to LVM

In this section, you are going to learn:

1. **What is LVM?**
2. **LVM organisation:**
 - **Physical volumes, Volumes groups & Logical volume**
 - **Physical extends**
 - **Device mapper**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

9.1.1 What is LVM?

Logical Volume Management (LVM) makes it easier to manage disk space.

Problem with normal partition:

- Once a partition is created, it cannot be resized.
- If a partition needs to be reduced or expanded, it's not possible.

LVM is used for the following purposes:

- Combine multiple HDD storage and use them as single HDD.
- If existing partition size is 100% full, you can extend it without the need to unmount the partition.
- If existing partition size is very large and resulting in space wasteage, you can reduce it without the need to unmount the partition.

9.1.2 LVM organisation

LVM are organized into:

- Physical Volumes (PVs)
- Volume Groups (VGs)
- Logical Volumes (LVs)

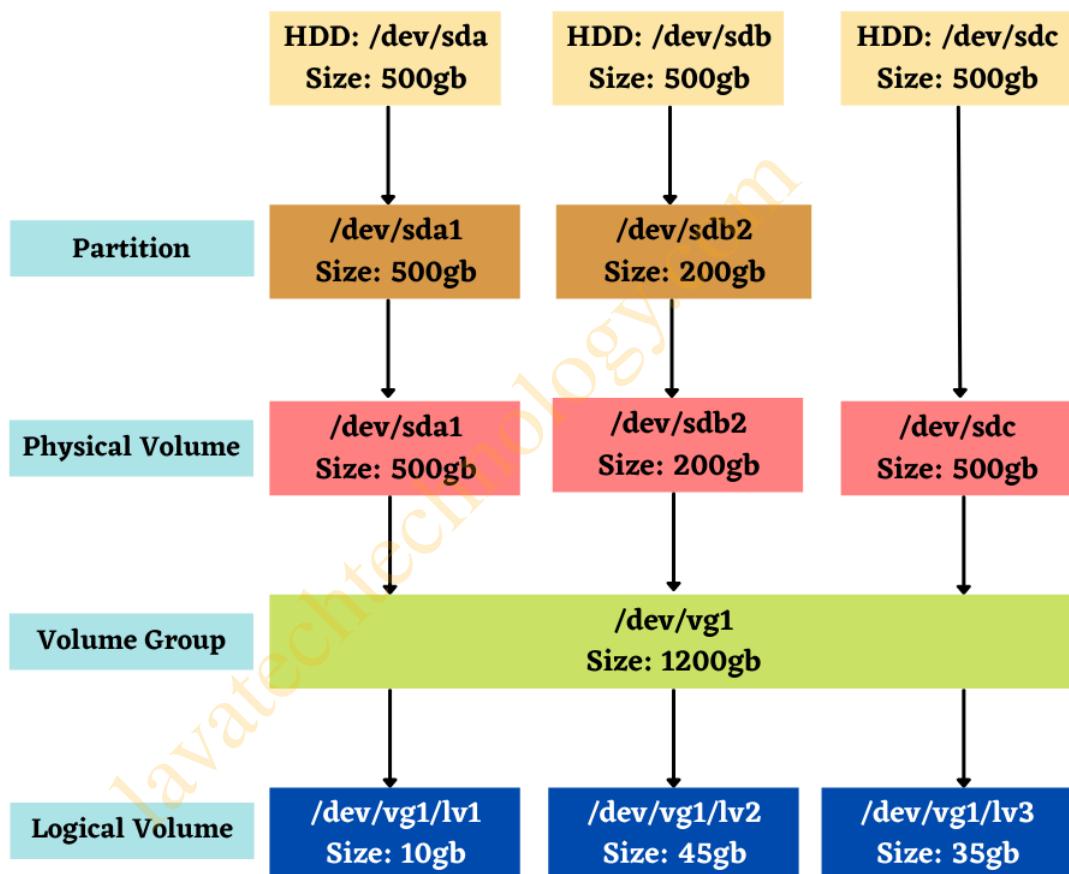


Figure 9.1: LVM

Let's see each of these in detail.

Physical Volumes (PVs)

- Physical volumes are physical HDD or physical disk partitions (eg: /dev/sda or /dev/sdb1).
- In order to create LVM, you first need to create physical volumes out of HDD or partition to proceed.

Volume Groups (VGs)

- A volume group is a collection of physical volumes.
- All volume groups would be created under **/dev** filesystem.
- Eg: **/dev/vg1**

Logical Volumes (LVs)

- A volume group can be partitioned into logical volumes.
- Every logical volume can be located under it's VG.
- Eg: **/dev/vg1/lv1** , **/dev/vg1/lv2** etc.

Device Mapper

- Device mapper is a Linux kernel module.
- Its job is to map devices names correctly to the physical devices.
- Eg: If the logical volume is named as **/dev/vg1/lv1**, device mapper will create mapping with name **/dev/mapper/vg1-lv1**.

9.1.3 Volumes practical

Physical Volumes (PVs) Commands:

- **How to create physical volumes (PVs)?**

pvcreate: Initialize physical volume(s) for use by LVM

Syntax: `pvcreate device_name`

Eg: Create physical volume of HDD `/dev/sda`:

```
# pvcreate /dev/sda
```

Eg: Create physical volume of HDD `/dev/sdb9`:

```
# pvcreate /dev/sdb9
```

- **How to display all physical volumes (PVs)?**

pvdisplay: Display various attributes of physical volume(s)

Syntax: `pvdisplay [device_name]`

Syntax: `pvs [device_name]`

Eg: Display all the available physical volumes:

```
# pvdisplay
```

Eg: Display a specific physical volume:

```
# pvdisplay /dev/sda1
```

- **How to delete a physical volumes (PVs)?**

pvremove: Delete a physical volume(s).

Syntax: pvremove device_name

Eg: Remove **/dev/sda1** physical volumes:

```
# pvremove /dev/sda1
```

lavatechtechnology.com

Volume Group (VGs) Commands

- **How to create new volume group (VGs)?**

vgcreate: Create a volume group.

Syntax: `vgcreate VG_new PV`

Eg: Create volume group named **vg0** of PV **/dev/sda**:

```
# vgcreate vg0 /dev/sda
```

Eg: Create volume group named **vg0** of PV **/dev/sdb9** and **/dev/sdc**:

```
# vgcreate vg0 /dev/sdb9 /dev/sdc
```

- **How to extend an existing volumes groups (VGs)?**

vgextend: Add one or more initialized PVs to an existing VG to extend it in size

Syntax: `vgextend [VG_name] [PV_name]`

Eg: Extend VG named **vg0** by adding PV **/dev/sdb2**:

```
# vgextend vg0 /dev/sdb2
```

- **How to remove a PV from an existing volumes groups (VGs)?**

vgreduce: Remove a PV from a VG.

Syntax: `vgreduce [VG_name] [PV_name]`

Eg: Remove PV named **/dev/sdb2** from VG named **vg0**

```
# vgreduce vg0 /dev/sdb2
```

- **How to display all volumes groups (VGs)?**

vgdisplay: Display various attributes of volume group(s).

Syntax: **vgdisplay [VG_name]**

Syntax: **vgs [VG_name]**

Eg: Display all the available volume groups:

```
# vgdisplay
```

Eg: Display a specific volume group:

```
# vgdisplay vg0
```

- **How to delete a volume group (VGs)?**

vgremove: Remove a specific volume group.

Syntax: **vgremove VG_name**

Eg: Remove **vg0** volume group:

```
# vgremove vg0
```

Logical Volume (LV) Commands

- **How to create new logical volume (LV)?**

lvcreate: Create a new logical volume

Syntax: `lvcreate -L size[G,M,K] -n vg_name lv_name`

Eg: Create logical volume named **lv1** from volume group **vg0** of size 50GB.

```
# lvcreate -L 50G -n vg0 lv1
```

- **How to display logical volumes (LVs)?**

lvdisplay: Check for the LV properties

Syntax: `lvdisplay [LV_name]`

Syntax: `lvs [LV_name]`

Eg: Display all the available logical volumes:

```
# lvdisplay
```

Eg: Display a specific logical volume:

```
# lvdisplay /dev/vg0/lv1
```

or

```
# lvdisplay /dev/mapper/vg0-lv1
```

- **Once LV partition is ready, you can give it filesystem like normal partition.**

Eg: Apply **xfs** filesystem to LV named **/dev/mapper/vg0-lv1**

```
# mkfs -t xfs /dev/mapper/vg0-lv1
```

- Mount the LV partition like normal partition.

Eg: Temporary mounting

```
# mount /dev/mapper/vg0-lv1 /mnt
```

Eg: Permanent mounting

```
# tail -1 /etc/fstab
/dev/mapper/vg0-lv1 /mnt xfs defaults 0 0
```

- How to delete an existing logical volume (LVs)?

lvremove: Removes one or more logical volumes.

Syntax: lvremove [LV_name]

Eg: Remove logical volume **/dev/mapper/vg0-lv1**

```
# lvremove /dev/mapper/vg0-lv1
```

Warning: Make sure to unmount the LV partition before deleting it.

- How to extend an existing logical volume (LVs)?

- Step 1:

lvextend: Extend the logical volume(s)

To give the final size to logical volume:

Syntax: lvextend -L size[G,M,K] logical_volume_name

To give the additional size to logical volume:

Syntax: lvextend -L +size[G,M,K] logical_volume_name

Eg: Extend logical volume with total size of 8GB:

```
# lvextend -L 8G /dev/test-volume/data
```

Eg: Provide additional 5GB size to logical volume:

```
# lvextend -L +5G /dev/test-volume/data
```

Note: No need to unmount the LV partition while extending it.

- Step 2:

Once the LV partition is extended, it is important to resize the filesystem.

- * To extend **ext2, ext3, ext4** filesystems:

resize2fs: This command is used to resize ext, ext2, ext3 and ext4 filesystems

Syntax: resize2fs logical_volume_name

Eg: To resize filesystem of logical volume:

```
# resize2fs /dev/mapper/vg0-lv1
```

- * To extend **xfs** filesystem:

xfs_growfs: Expands an existing XFS filesystem.

Syntax: xfs_growfs logical_volume_name

Eg: To resize filesystem of logical volume:

```
# xfs_growfs /dev/mapper/vg0-lv1
```

- How to reduce an existing logical volume (LVs)?

Note: You cannot reduce **xfs** filesystem. You can reduce **ext2**, **ext3** & **ext4** filesystems.

- Step 1:

Unmount the LV partition.

Eg:

```
# umount /dev/mapper/vg0-lv1
```

- Step 2:

e2fsck: Used to check the ext2/ext3/ext4 family of file systems for any errors. The **-f** option is used to force the checking.

Syntax: **e2fsck -f logical_volume_name**

Eg: Check the filesystem of logical volume:

```
# e2fsck -f /dev/mapper/vg0-lv1
```

- Step 3:

resize2fs: Allows you to resize ext2, ext3, or ext4 file systems

Syntax: **resize2fs logical_volume_name [desired size]**

Eg: Resize the filesystem of LV:

```
# resize2fs /dev/mapper/vg0-lv1 3G
```

- Step 4:

lvreduce: Shrink the LV and the filesystem applied on it

Syntax: `lvreduce -L size[G,M,K] logical_volume_name`

Eg: Resize the filesystem of logical volume:

```
# lvreduce -L 3G /dev/mapper/vg0-lv1
```

- Step 5:

Finally mount the filesystem again:

```
# mount /dev/mapper/vg0-lv1 /mnt
```

9.1.4 Physical extends

- VG are segmented into small, fixed-size chunks called **physical extents (PE)**.
- LV are created by adding up these PEs in VG.

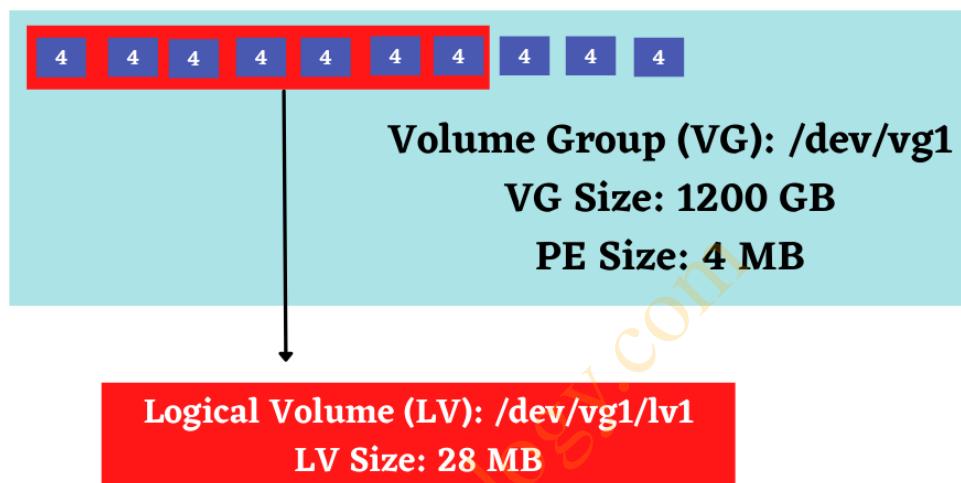


Figure 9.2: Physical extend

- Default PE size is 4MB.
- Command to check PE size of VG:

```
[root@server ~]# vgdisplay
--- Volume group ---
VG Name          vg0
System ID        lvm2
Format           lvm2
Metadata Areas   1
Metadata Sequence No 1
VG Access        read/write
VG Status        resizable
MAX LV           0
Cur LV           0
Open LV          0
Max PV           0
Cur PV           1
Act PV           1
VG Size          <8.00 GiB
PE Size          4.00 MiB
total PE         2047
Alloc PE / Size  0 / 0
Free  PE / Size  2047 / <8.00 GiB
VG UUID          6lXVRP-BGkC-ctNF-aDJ3-kHec-cdaX-TvTaeh
```

Figure 9.3: Sample output of vgdisplay

- You can change the PE size for your volume group at the time of VG creation

Command to change PE size:

vgcreate -s: The -s option allows to change the PE size at the time of VG creation

Syntax: vgcreate -s PE_size vg_name pv_name

Eg: Resize the filesystem of logical volume:

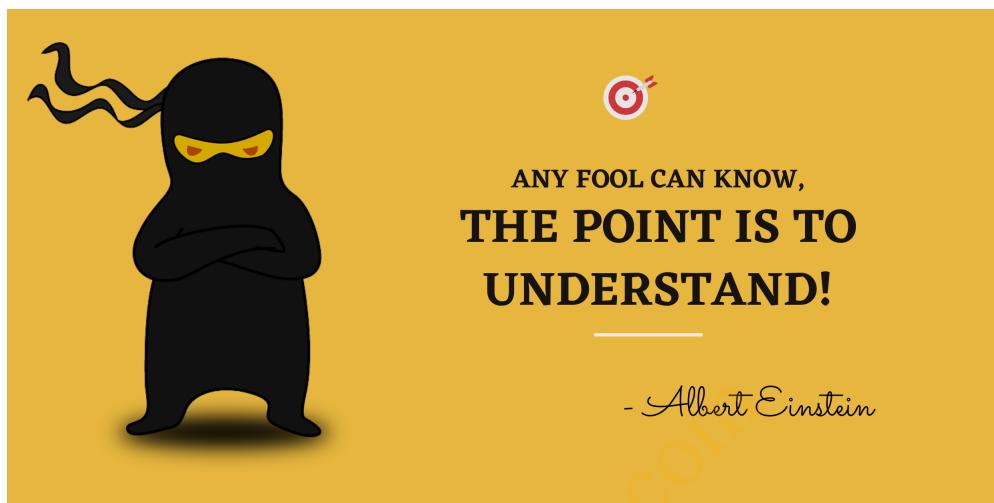
```
# vgcreate -s 8 vg0 /dev/sdb
```

```
[root@server ~]# vgdisplay
--- Volume group ---
VG Name          vg0
System ID
Format           lvm2
Metadata Areas   1
Metadata Sequence No  1
VG Access        read/write
VG Status        resizable
MAX LV
Cur LV
Open LV
Max PV
Cur PV
Act PV
VG Size          7.99 GiB
PE Size          8.00 MiB
Total PE         1023
Alloc PE / Size  0 / 0
Free  PE / Size  1023 / 7.99 GiB
VG UUID          C1TDrE-1Gnh-BD6Z-dJ6T-xQKu-ZUqq-gEtGNZ
```

Figure 9.4: PE size changed

- You cannot change PE size once the VG is created.

9.1.5 Practice



1. State whether true or false. Once a filesystem is full, you cannot resize it if it's not an LVM partition.
 - (a) True
 - (b) False
2. Which of the following are valid LVM components? (Select all that applies.)
 - (a) Physical volume
 - (b) Volume logical
 - (c) Volume group
 - (d) Logical volume
3. State whether true or false. Volume group is collection of logical volumes.
 - (a) True
 - (b) False

4. Which of the following statements are true about device mapper?

- (a) It is a Linux kernel module.
- (b) It creates a mapping for logical volume named as /dev/vg1/lv5 with name /dev/mapper/vg1-lv5.
- (c) It creates a mapping for logical volume named as /dev/vg1/lv5 with name /dev/vg1-lv5.
- (d) Its job is to map logical volume's name correctly to physical devices.

5. Which of the following are valid physical volume commands?

(Select all that applies.)

- (a) pvcreate, pvdisk, pvs, pvremove
- (b) pv-create, pv-display, pv-s, pv-remove
- (c) pv create, pv display, pv s, pv remove
- (d) pcreate, pdisk, ps, premove

6. Which of the following are valid volume group commands? (Select all that applies.)

- (a) vg-create, vg-display, vg-s, vg-remove
- (b) vg-create, vg-display, vg-s, vg-remove , vg-extend
- (c) vg create, vg display, vg s, vg remove , vg extend
- (d) vgcreate, vgextend, vgreduce, vgdisplay, vgs

7. Which of the following are valid syntax for logical volume commands? (Select all that applies)

- (a) lvcreate -L 5G -n volumename lvname
- (b) lvextend -L +5G lvname
- (c) lvreduce -L 5G lvname
- (d) lvremove lvname

8. State whether true or false. You can reduce an LVM having ext2, ext3 or ext4 filesystem, but you cannot reduce an LVM having xfs filesystem.
 - (a) True
 - (b) False
9. Which of the following command is used to resize ext2, ext3 & ext4 filesystem?
 - (a) blkid
 - (b) e2fsck
 - (c) resize2fs
 - (d) e4fsck
10. What is the default PE size?
 - (a) 8 MB
 - (b) 16 MB
 - (c) 6 4MB
 - (d) 4 MB
11. Which of the following command is used to change PE size?
 - (a) vgcreate -s
 - (b) vgextend -s
 - (c) vgmodify -s
 - (d) vgdevelop -s

10.1 Inodes & Links

In this section, you are going to learn:

1. What are inodes?
2. Inode numbers and filesystem
3. Types of links

Finally, there will be a **small excrise** on these topics to check your knowledge.



So let's get started....

10.1.1 What are inodes?

- Each file contains metadata & an identification number, called an inode number.
- The inode number refers to the **physical file & the data stored in a particular location.**
- An inode number contains files metadata as shown below:

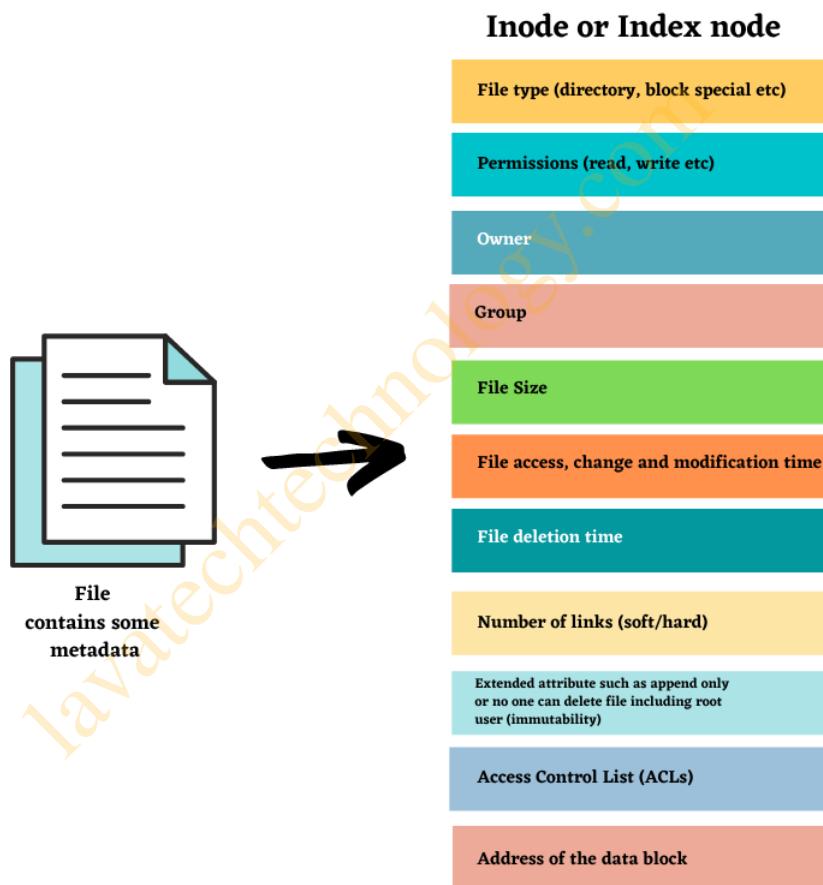


Figure 10.1: File metadata

10.1.2 Inode number and filesystem

- A filesystem is divided into two parts – data blocks and inode number.

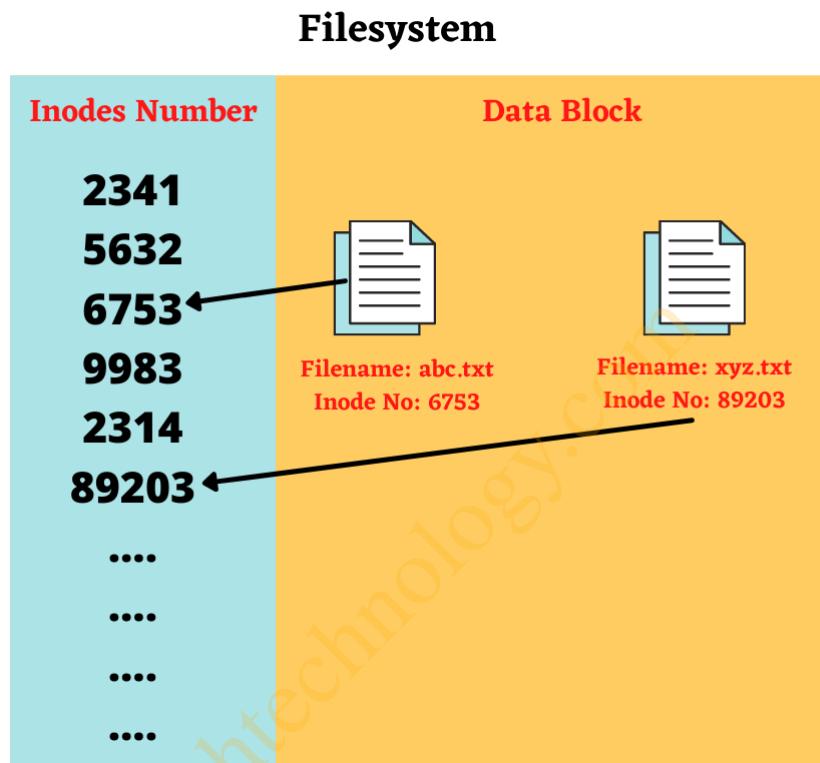


Figure 10.2: Filesystem and inodes

- How do I see a file's inode number?

Solution: You can use "ls -i" command to see inode number of file:

```
# ls -i /etc/passwd  
32820 /etc/passwd
```

10.1.3 Links and it's types

What is a link?

- A link allows two or more names for the same file/directory.
- There are two types of links:
 - **Hard link:**

- * Can be created only for file and points to the file's inode number.
- * Cannot be created across filesystem.
- * Inode for all the hard links are same .

Syntax: `ln source_filename hard_link_name`

Eg:

```
# Create hard link for abc.txt file:  
# ln abc.txt /tmp/new_link.txt  
  
# Check the inode of abc.txt & /tmp/new_link.txt  
# ls -i abc.txt /tmp/new_link.txt  
3052945 abc.txt  
3052945 /tmp/new_link.txt
```

Note: Notice how the inode number for hard link are same.

- Soft link:

- * Can be created for file as well as directory.
- * Can be created across filesystems.
- * Inode for soft link are different.

Syntax: `ln -s source_file_or_directory softlink_name`

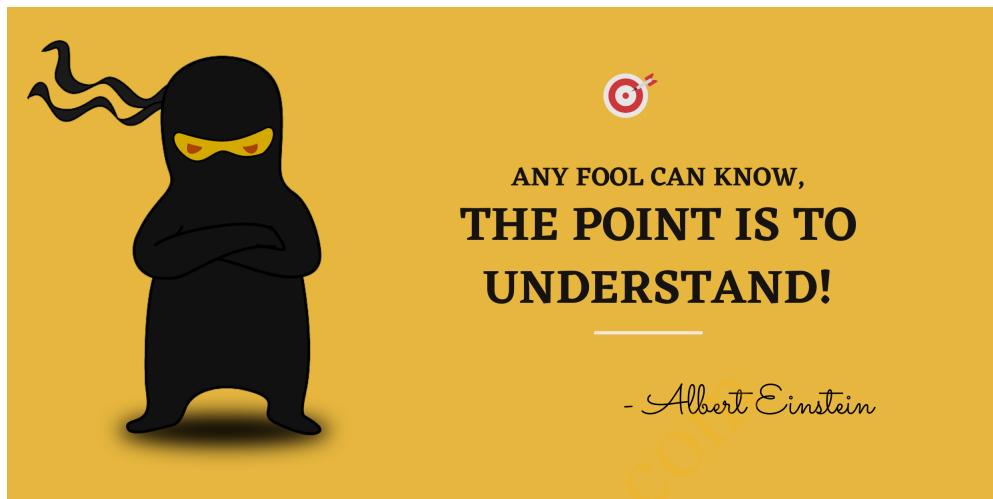
Eg:

```
# Create soft link for /tmp folder:  
$ ln -s /tmp /home/sample/  
  
# Check the inode of /tmp & /home/sample  
$ ls -i /tmp /home/sample  
3052047 /tmp  
2052047 /home/sample
```

Note: Notice how the inode number for soft link are different.

10.1.4 Difference between links

| Hard link | Soft link |
|--|---|
| Can be created only for files. | Can be created for files & directories. |
| Cannot be created across the different filesystems. | Can be created across filesystem |
| Point to another file by inode. | Point to another file or directory by name rather than by inode. |
| Deleted only if all hard links are deleted | Deleting a soft link does not delete the target file or directory. |

10.1.5 Practice

- 1. What is an inode number?**
 - (a) Unique identification number used to refer to a file and its content
 - (b) Number used to denote file permission
 - (c) Used to denote whether the file is corrupt or not
 - (d) Used to recover data for a file in case of data loss
- 2. Which of the following command is used to find a file's inode number?**
 - (a) ls -ld filename
 - (b) ls -l filename
 - (c) ls -i filename
 - (d) ls -a filename
- 3. Select the correct type of links in Linux. (Select all that applies.)**
 - (a) Short link
 - (b) Hard link
 - (c) I link
 - (d) Soft link

4. State whether true or false. Hard link can be created across filesystem.
 - (a) True
 - (b) False
5. State whether true or false. If there are 5 soft links for a file, they all have same inode number.
 - (a) True
 - (b) False
6. State whether true or false. Hard link can be created only for file.
 - (a) True
 - (b) False
7. Which of the following command is used to create soft link?
 - (a) ln -soft filename_or_directoryname softlink_name
 - (b) ln filename_or_directoryname softlink_name
 - (c) ln -s filename_or_directoryname softlink_name
 - (d) ln -l filename_or_directoryname softlink_name
8. Which of the following command is used to create hard link?
 - (a) ln -hard filename hardlink_name
 - (b) ln filename hardlink_name
 - (c) ln -s filename hardlink_name
 - (d) ln -l filename hardlink_name

11.1 RPM in detail

In this section, you are going to learn:

1. **What is a package?**
2. **What is an RPM?**
3. **RPM commands**
4. **Drawback of RPM**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

11.1.1 What is a package?

- A package is how Linux OS delivers a **software**.
- Linux packages are maintained through software repositories.

Note: A repository basically mean collection of something. It can be collection of codes, or collection of files, or collection of data or collection of anything else.

11.1.2 What is an RPM?

- RPM is a collection of different files that constitutes a package.
- RPM stands for **Red Hat Package Manager**.
- It was developed by Red Hat company.
- RPM is used on Red Hat-based Linux operating systems (like Fedora, CentOS, RHEL, etc.)
- An RPM package uses the **.rpm** extension.
- Syntax of package name:

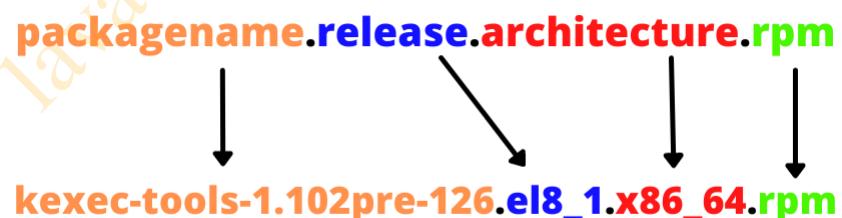


Figure 11.1: Package name syntax

11.1.3 RPM commands

rpm: Used to install packages with **.rpm** extension.

Syntax: rpm option package_name

Options with "rpm" command:

- **-i:** Installs the package, if it isn't already installed
- **-v:** Provide more detailed output
- **-h:** Print hash marks to display progress

Syntax: rpm -ivh package_name

Eg: Download package named "cvs" and install it:

```
# For rhel/centos7, download & install cvs package
# wget https://rpmfind.net/linux/centos/7.9.2009/os/x86_64/
/Packages/cvs-1.11.23-35.el7.x86_64.rpm
# rpm -ivh cvs-1.11.23-35.el7.x86_64.rpm

# For rhel/centos8, download & install cvs package
# wget https://rpmfind.net/linux/epel/8/Everything/x86_64/
/Packages/c/cvs-1.11.23-52.el8.x86_64.rpm
# rpm -ivh cvs-1.11.23-52.el8.x86_64.rpm
```

- **-U:** Upgrades any existing package or installs it if an earlier version isn't already installed.

Syntax: rpm -U package_name

- **-F:** Upgrades only existing packages. It does not install a package if it wasn't previously installed.

Syntax: rpm -F package_name

- **-e:** Un-install a package.

Syntax: rpm -e package_name

Eg: Un-install package named "cvs".

```
# rpm -e cvs
```

Note: **rpm -e** fails with an error if there's some dependent package issues.

- **--nodeps:** Ignore dependency error and uninstall the package (which may break the package dependent on it).

Syntax: rpm -e --nodeps package_name

Eg:

```
# rpm -e --nodeps cvs
```

- **-qa:** Lists all installed packages. Here **-q** stands for **query** and **-a** stands for all.

Syntax: rpm -qa

- **-qf:** Identifies the package associated with a specific file/directory.

Syntax: rpm -qf /path/to/file

Eg: Find package associated with **/etc/ssh** file or **/etc** folder:

```
# rpm -qf /etc/ssh  
# rpm -qf /etc
```

- **-qc:** Lists configuration files that come with the package installation.

Syntax: rpm -qc packagename

Eg: Find all configuration files associated with package **openssh**:

```
# rpm -qc openssh
```

- **-qi:** Displays basic information for package name.

Syntax: rpm -qi packagename

Eg: Find basic information associated with package **openssh**:

```
# rpm -qi openssh
```

- **-qR:** Display all package dependencies.

Syntax: rpm -qR packagename

Eg: Display all dependency associated with package **openssh**:

```
# rpm -qR openssh
```

lavatechtechnology.com

11.1.4 Drawback of RPM

1. Dependency problem while installing a package:

- The "rpm" commands can handle only 1 package at a time.
- However, packages can be dependent on each other.
- Eg: Notice error while installing "**NetworkManager**" package using **rpm** command:

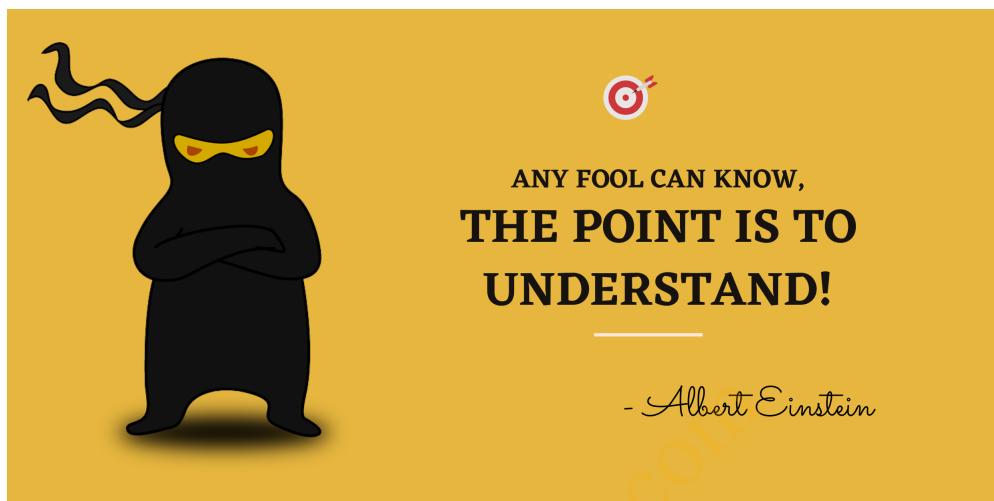
```
# wget https://rpmfind.net/linux/centos/8-stream/BaseOS/x86_64/os/Packages/NetworkManager-1.36.0-0.7.el8.x86_64.rpm
# rpm -ivh NetworkManager-1.36.0-0.7.el8.x86_64.rpm

warning: NetworkManager-1.36.0-0.7.el8.x86_64.rpm:
Header V3 RSA/SHA256 Signature, key ID
8483c65d: NOKEY
error: Failed dependencies:
NetworkManager-libnm(x86-64) = 1:1.36.0-0.7.el8 is
needed.....
```

2. Dependency problem while removing a pacakge:

- While "uninstalling" & "removing" a package, **rpm** command cannot remove a package if it has dependency issues.
- Eg: Notice the error while removing "**openssh**" package using **rpm** command:

```
# rpm -e openssh
error: Failed dependencies:
openssh = 8.0p1-10.el8 is needed by (installed)
openssh-clients-8.0p1-10.el8.x86_64
openssh = 8.0p1-10.el8 is needed by (installed)
openssh-server-8.0p1-10.el8.x86_64
```

11.1.5 Practice

- 1. What is the full form of RPM?**
 - (a) Read Package Memory
 - (b) Red Hat Package Manager
 - (c) Read Program Meta
 - (d) Random Program Meta
- 2. Which of the following is correct RPM package syntax?**
 - (a) package.architecture.rpm
 - (b) package.release.rpm
 - (c) package.architecture.release.rpm
 - (d) package.release.architecture.rpm
- 3. Select the correct RPM command to install a package.**
 - (a) rpm -install packagename
 - (b) rpm -ivh packagename
 - (c) rpm -vh packagename
 - (d) rpm –install packagename

4. Select the correct RPM command to upgrade a package.

- (a) rpm -upgrade packagename
- (b) rpm –upgrade packagename
- (c) rpm -U packagename
- (d) rpm -Uvh packagename

5. Select the correct RPM command to uninstall a package.

- (a) rpm -uninstall packagename
- (b) rpm -e packagename
- (c) rpm -remove packagename
- (d) rpm -r packagename

6. Select the correct RPM command to list all installed packages.

- (a) rpm -l
- (b) rpm -q
- (c) rpm -qa
- (d) rpm -a

7. Select the correct RPM command to list all configuration file install with a package.

- (a) rpm -qc
- (b) rpm -q
- (c) rpm -c
- (d) rpm -a

11.2 YUM & dnf

In this section, you are going to learn:

1. **What is YUM & DNF?**
2. **Advantages of YUM over RPM**
3. **Configuration of YUM server and client**
4. **Subscribing RHEL8 server**
5. **YUM commands**
6. **DNF commands**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

11.2.1 What is YUM & dnf?

What is YUM?

- YUM stands for **Yellow dog Updater, Modified**.
- YUM is a command-line as well as graphical-based package management tool for RPM.
- YUM allows to install, update, remove & search packages easily.

What is DNF?

- The DNF command (Dandified yum) is the next version of the YUM.
- It is the default package manager for Fedora 22, CentOS8, and RHEL8.
- It is intended to be a replacement for YUM.

Advantages of YUM over RPM

1. Dependency resolution:
 - If a package installation requires additional packages, YUM can list these dependencies and prompt the user to install them.
2. Locate package from multiple locations & install them:
 - YUM can be configured to search for software packages in more than one location.
3. YUM allows to specify particular software versions or architectures while installing or removing the package.

11.2.2 Configuration of YUM server

You can create a private software repository that can be used everytime you want to install a package by configuring **yum server**.

Below are the steps to configure a **yum server** using **FTP (file transfer protocol)**:

1. First attach the RHEL ISO image to the virtual machine(VM) while the machine is up and running.

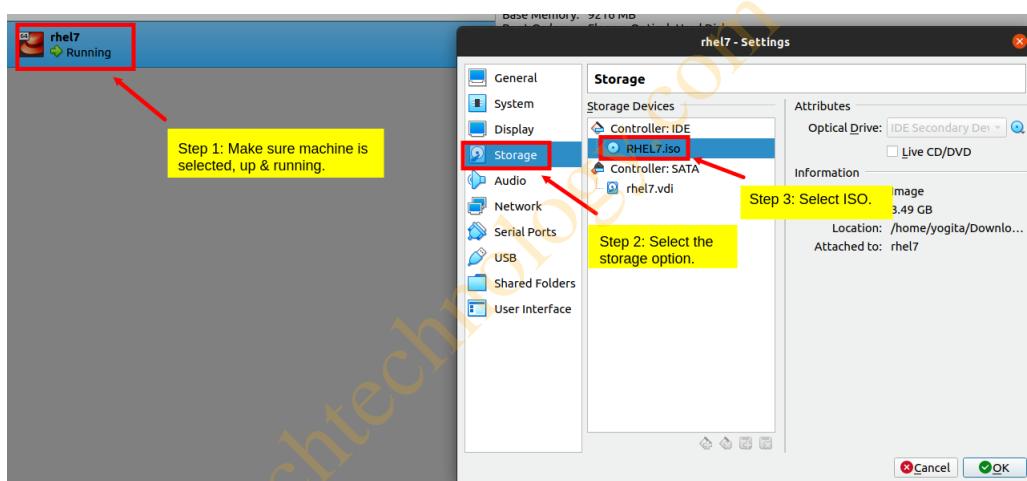


Figure 11.2: Upload RHEL7 ISO

2. Unmount **/dev/sr0** & mount the iso image attach on device **/dev/sr0** to "**/mnt**" folder.

```
# umount /dev/sr0
# mount /dev/sr0 /mnt
```

3. Install following packages from attached ISO image.

```
# rpm -ivh /mnt/Packages/ftp-xxxx
# rpm -ivh /mnt/Packages/vsftpd-xxxx
# rpm -ivh /mnt/Packages/createrepo-xxxx
```

4. Copy all packages from attached ISO image to pub directory:

```
# cp -rv /mnt/Packages/* /var/ftp/pub
```

5. Create a repo in pub directory.

```
# createrepo -v /var/ftp/pub/
```

6. Start and enable the vsftpd service.

```
# systemctl restart vsftpd  
# systemctl enable vsftpd
```

7. Browse the pub directory using browser(like firefox or google chrome) with URL **ftp://(server IP address)/pub**.

Eg: "ftp://192.168.10.111/pub"

8. Create a repository file named **/etc/yum.repos.d/new.repo**:

```
[Title_name] # Add title name for this repository  
name = Repo_name # Add repository name  
enabled = 1  
gpgcheck = 0  
baseurl = ftp://(server-IP-address)/pub
```

Note:

- Location of repository file should be **/etc/yum.repos.d**
- Name of the file can be anything
- Extension of the file of should be **".repo"**

9. Clean all the cached files from any enabled repository.

```
# yum clean all
```

10. Refresh all the enabled repository:

```
# yum update all
```

11. Now you can install any packages using the YUM server.

Eg:

```
# yum install httpd -y
```

lavatechtechnology.com

11.2.3 Configuration of YUM client

You can start using yum server for package installation on any client, by following below steps:

1. Create a repository file named **/etc/yum.repos.d/new.repo**:

```
[Title_name] # Add title name for this repository
name = Repo_name # Add repository name
enabled = 1
gpgcheck = 0
baseurl = ftp://(server-IP-address)/pub
```

Note:

- Location of repository file should be **/etc/yum.repos.d**
- Name of the file can be anything
- Extension of the file of should be **".repo"**

2. Clean all the cached files from any enabled repository.

```
# yum clean all
```

3. Refresh all the enabled repository:

```
# yum update all
```

4. Now you can install any packages using the YUM server.

Eg:

```
# yum install httpd -y
```

11.2.4 Subscribing RHEL8 server

Starting with RHEL8, you can directly install package from Red Hat Network.

For this, you need join a free Red Hat Developer program.

Follow along for step-by-step guide:

- Browse to <https://developers.redhat.com/register>.
- Create an account as shown in the image:

The screenshot shows the 'Create a Red Hat account' page. It includes fields for 'Enter username', 'Enter email address', 'Select job role' (set to 'Network Administrator'), 'Enter password', 'Accept terms and conditions' (with three checkboxes checked), and 'Email opt-in' (with one checkbox checked). A red arrow points to the third checkbox under 'Accept terms and conditions'.

Red Hat
Create a Red Hat account
Sign up and use this Red Hat account to access all of Red Hat's applications, communities, support, and more.
Red Hat will collect your contact and account information to create your Red Hat account. We use your personal data to identify you and to provide you with information, support, and customer service. For more information, please see Red Hat's [privacy statement](#).
* Required fields
Choose a Red Hat login *
Enter username
Your login is a user ID for accessing your account across all Red Hat sites. It must be at least 5 characters and cannot be changed once created.
Email address *
Enter email address
Job role *
Select job role
Network Administrator
Choose a password *
Enter password
Your password must include at least 8 characters. A strong password combines lowercase letters, uppercase letters, numbers, and symbols.
 I have read and agree to all the terms and conditions below (check all boxes).
 * I have read and agree to the [Enterprise Agreement](#).
 * I have read and agree to the [Red Hat Developer Subscription for Individuals](#).
Accept terms and conditions
Email opt-in
 Receive email notifications of Red Hat Developer services and events, including invitations and reminders.
CREATE MY ACCOUNT

Figure 11.3: Sample output

- Once you have created your account, Red Hat should send a verification email as shown. Click on the link provided in email to confirm your email address.

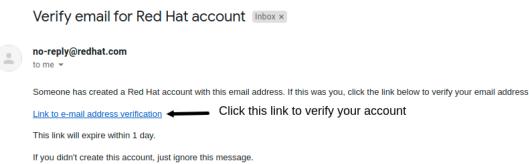


Figure 11.4: Sample output of verification email

- Now register your RHEL8 server to Red Hat Network. Enter valid username and password that you created during account creation.

Syntax: subscription-manager register

Eg:

```
[root@rhel8 yum.repos.d]# subscription-manager register
Registering to: subscription.rhsm.redhat.com:443/subscription
Username: [REDACTED]
Password: [REDACTED]
The system has been registered with ID: e5f57771-77b9-46ad-badd-55b9f2539f50
The registered system name is: rhel8.localdomain
```

Figure 11.5: Sample output

- Confirm whether the system is successfully registered.

Syntax: subscription-manager status

Eg:

```
[root@rhel8 yum.repos.d]# subscription-manager status
+-----+
| System Status Details |
+-----+
Overall Status: Current
System Purpose Status: Not Specified
```

Figure 11.6: Sample output

- Finally, attach the subscriptions to your RHEL8 server.

Syntax: subscription-manager attach --auto

Eg:

```
[root@rhel8 yum.repos.d]# subscription-manager attach --auto
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux for x86_64
Status: Subscribed
```

Figure 11.7: Sample output

- Clear the yum cache.

Syntax: yum clean all

- Refresh yum repository.

Syntax: yum update all

Eg:

```
[root@rhel8 ~]# yum update all
Updating Subscription Management repositories.
Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)                               351 kB/s | 45 MB   02:09
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)                            171 kB/s | 39 MB   03:53
No match for argument: all
Error: No packages marked for upgrade.
```

Figure 11.8: Sample output

- Install httpd package to confirm you are connected to Red Hat Network.

```
# yum install httpd -y
```

11.2.5 YUM commands

yum: Used to install, update, remove & search package.

Syntax: yum [options] command [package_name]

Command and options with **yum** command:

- **install:** Install package along with it's dependency.

-y: Automatically answer yes for all questions

Syntax: yum install package_name -y

Eg:

```
# yum install httpd -y
```

- **remove:** Uninstall a package along with this dependency.

Syntax: yum remove package_name

Eg:

```
# yum remove httpd
```

- **info:** Display the information of a package.

Syntax: yum info package_name

Eg:

```
# yum info httpd
```

- **groupinfo:** List all packages provided by the group.

Syntax: yum groupinfo group_name

Eg:

```
# yum groupinfo "Development Tools"
```

- **groupinstall**: Install all packages provided by the group.

Syntax: yum groupinstall group_name

Eg:

```
# yum groupinstall "Development Tools"
```

- **groupremove**: Remove all packages provided by the group.

Syntax: yum groupremove group_name

Eg:

```
# yum groupremove "Development Tools"
```

- **update**: Update all packages and their dependencies.

Syntax: yum update

- **history**: Display a list of all the latest yum transactions.

Syntax: yum history

- **history info**: Examine a particular yum transaction.

Syntax: yum history info transactionID

Eg:

```
# yum history info 8
```

- **history undo:** Revert a particular transaction.

Syntax: yum history undo transactionID

Eg:

```
# yum history undo 8
```

- **history redo:** Repeat a particular transaction.

Syntax: yum history redo transactionID

Eg:

```
# yum history redo 8
```

- **list all:** List all installed packages.

Syntax: yum list all

Eg: List all installed pacakges associated with httpd.

```
# yum list all | grep httpd
```

- **localinstall:** Install locally downloaded package.

Syntax: yum localinstall package.rpm

Eg: Download cvs package using command "wget

https://rpmfind.net/linux/epel/8/Everything/x86_64/Packages/c/cvs-1.11.23-52.el8.x86_64.rpm

Install downloaded package using command:

```
# yum localinstall cvs-1.11.23-52.el8.x86_64.rpm -y
```

lavatechtechnology.com

11.2.6 Dnf commands

dnf: Used to install, update, remove & search package.

Syntax: `dnf [options] command [package_name]`

Command and options with **dnf** command:

- **install:** Install package along with it's dependency.

-y: Automatically answer yes for all questions.

Syntax: `dnf install package_name -y`

Eg:

```
# dnf install httpd -y
```

- **remove:** Uninstall a package along with this dependency.

Syntax: `dnf remove package_name`

Eg:

```
# dnf remove httpd
```

- **info:** Display the information of a package.

Syntax: `dnf info package_name`

Eg:

```
# dnf info httpd
```

- **groupinfo:** List all packages provided by the group.

Syntax: dnf groupinfo group_name

Eg:

```
# dnf groupinfo "Development Tools"
```

- **groupinstall**: Install all packages provided by the group.

Syntax: dnf groupinstall group_name

Eg:

```
# dnf groupinstall "Development Tools"
```

- **groupremove**: Remove all packages provided by the group.

Syntax: dnf groupremove group_name

Eg:

```
# dnf groupremove "Development Tools"
```

- **update**: Update all packages and their dependencies.

Syntax: dnf update

- **history**: Display a list of all the latest dnf transactions

Syntax: dnf history

- **history info**: Examine a particular dnf transaction.

Syntax: dnf history info transactionID

Eg:

```
# dnf history info 8
```

- **history undo:** Revert a particular transaction.

Syntax: dnf history undo transactionID

Eg:

```
# dnf history undo 8
```

- **history redo:** Repeat a particular transaction.

```
Syntax: dnf history redo transactionID
```

Eg:

```
# dnf history redo 8
```

- **list all:** List all installed packages.

```
Syntax: dnf list all
```

Eg: List all installed packages associated with httpd.

```
# dnf list all | grep httpd
```

- **localinstall:** Install locally downloaded package.

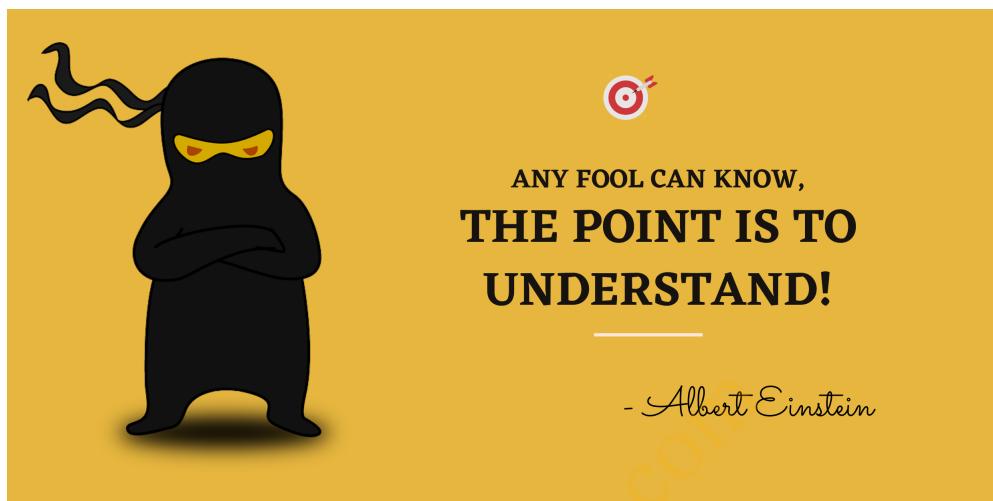
```
Syntax: dnf localinstall package.rpm
```

Eg: Download cvs package using command "wget

https://rpmfind.net/linux/epel/8/Everything/x86_64/Packages/c/cvs-1.11.23-52.el8.x86_64.rpm"

Install downloaded package using command:

```
# dnf localinstall cvs-1.11.23-52.el8.x86_64.rpm -y
```

11.2.7 Practice

- 1. Which of the following statement is true about yum & rpm command? (Select all that applies.)**
 - (a) The **rpm** command can install a package, but cannot resolve dependency issue of the package.
 - (b) The **yum** command can install a package while resolving its dependency issue.
 - (c) The **yum** command cannot resolve dependency issue of a package.
 - (d) The **rpm** command can resolve dependency issue of a package.
- 2. Which of the following is the correct location for repository file?**
 - (a) /etc/yum/repos.d
 - (b) /etc/yum/
 - (c) /etc/yum.repos.d/
 - (d) /etc/repos.d
- 3. Select the correct YUM command to install a package.**
 - (a) yum install package_name -y
 - (b) yum -install package_name -y

- (c) yum -i package_name -y
 - (d) yum –install package_name -y
- 4. Select the correct YUM command to uninstall a package.**
- (a) yum –remove package_name -y
 - (b) yum remove package_name -y
 - (c) yum uninstall package_name -y
 - (d) yum -e package_name -y
- 5. Select the correct DNF command display package details.**
- (a) dnf display package_name
 - (b) dnf show package_name
 - (c) dnf info package_name
 - (d) dnf list package_name
- 6. Select the correct DNF command to list all installed package.**
- (a) dnf list all
 - (b) dnf report all
 - (c) dnf show all
 - (d) dnf list

12.1 Process management

In this section, you are going to learn:

1. **What is a process?**
2. **Types of process**
3. **Linux process states**
4. **How to check Linux process?**
5. **How to kill a Linux process?**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

12.1.1 What is a process?

- A process is a program running in the OS using memory & CPU.
- A Linux process is also called **service or daemon**.
- Every process has many details associated with it, some of them are:
 - Process ID (PID)
 - Process name
 - A program associated with it
 - Process state
 - User owning the process
 - Parent Process ID (PPID)

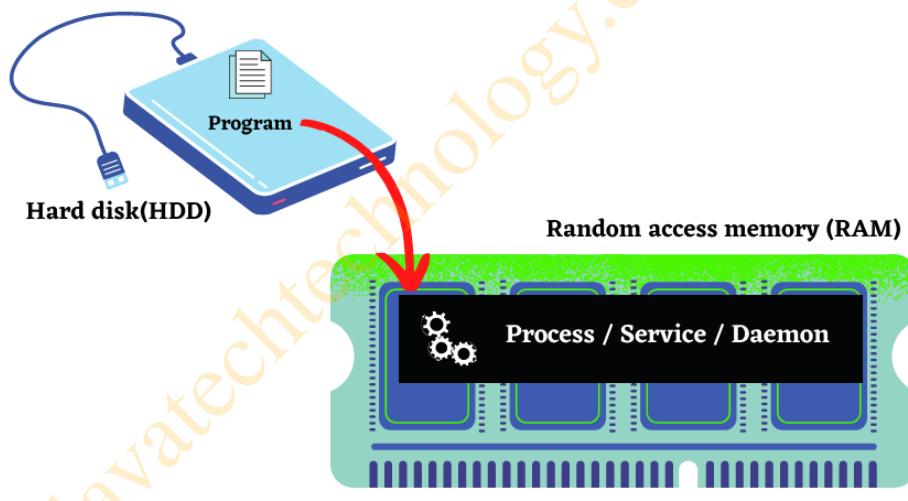
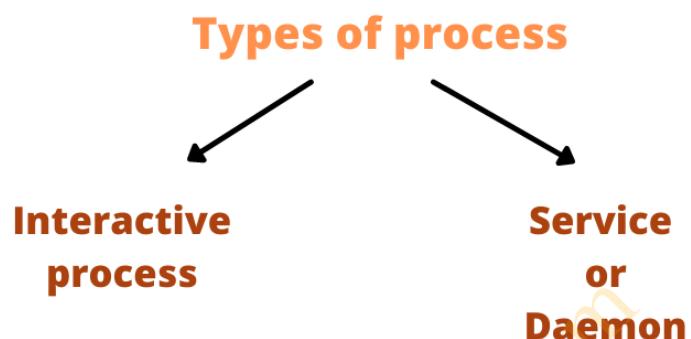


Figure 12.1: Process/Service/Daemon

- Keeping unused Linux process running in the system is **a waste of RAM & CPU**.
- Unused process can expose your system to **security threat**.

12.1.2 Types of process

There are 2 types of process:



Let's see each of these in detail.

1. Interactive process

- Interactive process are invoked by a user and can interact with the user.
- Classified into two types:

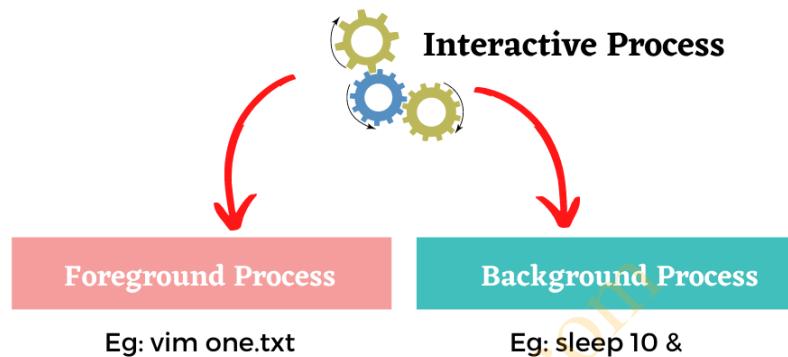


Figure 12.2: Types of interactive process

- Foreground process:** Process that you are currently interacting with using the terminal.
- Background process:**
 - These process do not interact with the user & they run in the background.
 - To run process in background, use "&" by appending it at the end of command.

Syntax: command &

Eg: Run **sleep** command in background:

```
# sleep 60 &
```

- **Commands for interactive process**

- (a) **jobs**: To check all interactive process.

Syntax: jobs

Eg: If there are any interactive process, **jobs** will list all process:

```
# jobs  
[1]+ Stopped vim one.txt  
[2]- Running sleep 10 &
```

Explanation of output:

- [1],[2]: Interactive process ids.
- "+": Job used as default for the fg or bg utilities.
- "-": Default job if the current default job were to exit.

- (b) **bg**: To change the state of a background process from "Stopped" to "Running".

Syntax: bg %[job_id]

Eg: Execute **sleep** command as shown and press **CTRL+Z** to push the process in background in "Stopped" state.

```
# sleep 300  
Press CTRL+Z
```

Next, execute **jobs** command to confirm shown output:

```
# jobs  
[1]+ Stopped sleep 300
```

Execute **bg** command to change the state of the command to "Running".

```
# bg %1  
[1]+ sleep 300 &  
# jobs  
[1]+ Running sleep 300 &
```

(c) **fg**: To run the background process in foreground.

Syntax: fg %[job_id]

Eg: Execute **sleep** command in background as shown:

```
# sleep 300 &
```

Next, execute jobs command to confirm shown output:

```
# jobs  
[1]+ Running sleep 300 &
```

Execute **fg** command to bring **sleep** command in foreground.

```
# fg %1  
[1]+ sleep 300 &  
# jobs  
[1]+ Running sleep 300 &
```

2. Daemon

- Daemon are process that are running in background on the computer.
- They provide some services, but they do not interact with the console.
- **Server software are implemented as a daemon.**
- Eg: httpd, sshd, nfs etc.
- Command to check daemon status:

Syntax: systemctl status daemon_name

Eg:

```
root@server:~# systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2022-04-06 12:46:30 UTC; 15h ago
    Process: 1211 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 1250 (sshd)
     Tasks: 1 (limit: 1104)
    CGroup: /system.slice/ssh.service
           └─1250 /usr/sbin/sshd -D

Apr 06 12:46:29 server.example.com systemd[1]: Starting OpenBSD Secure Shell server...
Apr 06 12:46:30 server.example.com sshd[1250]: Server listening on 0.0.0.0 port 22.
Apr 06 12:46:30 server.example.com sshd[1250]: Server listening on :: port 22.
Apr 06 12:46:30 server.example.com systemd[1]: Started OpenBSD Secure Shell server.
Apr 06 12:46:30 server.example.com sshd[1271]: Accepted publickey for vagrant from 10.0.2.2 port 51454 ssh2: RSA
Apr 06 12:46:30 server.example.com sshd[1271]: pam_unix(sshd:session): session opened for user vagrant by (uid=0)
Apr 06 12:49:43 server.example.com sshd[1848]: Accepted publickey for vagrant from 10.0.2.2 port 51456 ssh2: RSA
Apr 06 12:49:43 server.example.com sshd[1848]: pam_unix(sshd:session): session opened for user vagrant by (uid=0)
```

Figure 12.3: Sample output

12.1.3 Linux process states

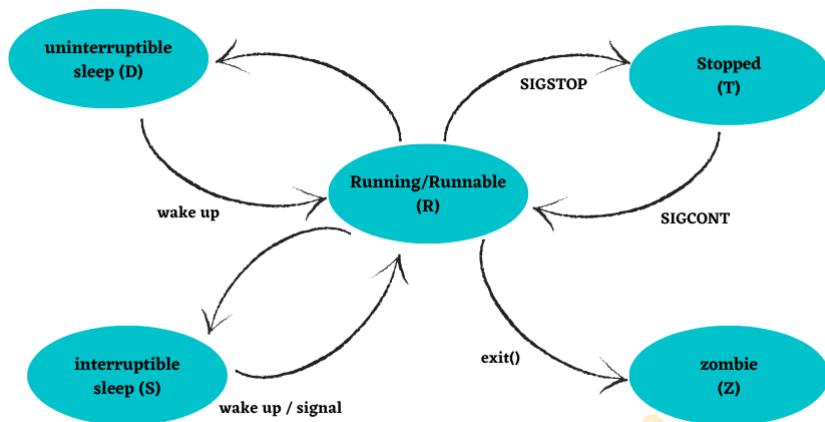


Figure 12.4: Process State

- **Running:**
 - Process is either **running or ready to execute**.
 - Denoted as "**R**".
- **Interruptible sleep:**
 - Indicates the process is blocked.
 - The process is waiting indirectly for some event.
 - Eg: Some process is waiting for completion of an I/O operation.
 - Denoted as "**S**".
- **Uninterruptible sleep:**
 - Indicates the process is blocked.
 - A process is waiting directly on hardware conditions.
 - Eg: Printer running out of page.
 - Denoted as "**D**".
- **Stopped:**
 - Halted process.
 - Eg: A process that is being troubleshooted can be put into the stopped state.
 - Denoted as "**T**".

- **Dead:**

- Permanently stopped process.
 - Dead process do not use RAM, CPU etc.
 - Denoted as "X".

- **Zombie:**

- Zombie process are dead process but they still consume CPU & RAM.
 - Zombie process does not have process ID, but their process name is still present.
 - They may result in CPU & RAM wastage.
 - Denoted as "Z".

12.1.4 How to check Linux process?

Commands to display process related details:

- **ps**: Display current processes.

Syntax: ps [options]

Eg: **ps** command without any option display current terminal process.

```
# ps
PID  TTY   TIME  CMD
190566 pts/3  00:00:00 bash
190577 pts/3  00:00:00 bc
197680 pts/3  00:00:00 ps
```

Options with **ps** command:

- **Display all process running in all terminals by all user.**

Syntax: ps -aux

where,

- a**: List all processes with a terminal (tty).
- x**: List all processes owned by all users.
- u**: Display user-oriented format.

Eg:

| USER | PID | %CPU | %MEM | VSZ | RSS | TTY | STAT | START | TIME | COMMAND |
|------|-----|------|------|-------|------|-----|------|-------|------|----------------|
| root | 1 | 0.0 | 0.5 | 77848 | 5812 | ? | Ss | Apr06 | 0:01 | /sbin/init |
| root | 2 | 0.0 | 0.0 | 0 | 0 | ? | S | Apr06 | 0:00 | [kthreadd] |
| root | 4 | 0.0 | 0.0 | 0 | 0 | ? | I< | Apr06 | 0:00 | [kworker/0:0H] |
| root | 6 | 0.0 | 0.0 | 0 | 0 | ? | T | Apr06 | 0:00 | [rcu_bh] |

Figure 12.5: Sample output

Output explanation:

- * **USER**: User owning the process
- * **PID**: Process ID

- * **%CPU:** CPU utilisation
- * **%MEM:** Memory utilisation
- * **RSS:** The resident set size, the non-swapped physical memory that the process has used in kilobytes
- * **VSZ:** Gives the virtual size of the process, code, data and stack segments
- * **TTY:**
 - Terminal on which the process is running
 - "?" means process is running in background
- * **STAT:** Display process state as shown below:
Process states:
 - **D:** Uninterruptible sleep (usually IO)
 - **I:** Idle kernel thread
 - **R:** Running process
 - **S:** Interruptible sleep (waiting for an event to complete)
 - **T:** Stopped process
 - **t:** Stopped by debugger during the tracing
 - **X:** Dead (should never be seen)
 - **Z:** Defunct ("zombie") process, terminated but not reaped by its parent
- * Additional characters:
 - <: High-priority
 - +: Foreground process
 - N: Low-priority
 - s: Session leader
 - L: Has pages locked into memory
 - l: Multi-threaded process
- * **START:** Start time of process
- * **TIME:** How long the process takes to execute
- * **COMMAND:** Program in running state

- Display all processes along with it's parent process ID (PPID).

Syntax: ps -ef

where,

- e:** Display all the processes.
- f:** Display full format listing along with PPID (i.e parent process ID).

Eg:

| UID | PID | PPID | C | STIME | TTY | TIME | CMD |
|------|-----|------|---|-------|-----|----------|-------------------|
| root | 1 | 0 | 0 | 12:09 | ? | 00:00:01 | /sbin/init splash |
| root | 2 | 0 | 0 | 12:09 | ? | 00:00:00 | [kthreadd] |
| root | 3 | 2 | 0 | 12:09 | ? | 00:00:00 | [rcu_gp] |
| root | 4 | 2 | 0 | 12:09 | ? | 00:00:00 | [rcu_par_gp] |

Figure 12.6: Sample output

- Display all process related to a specific user.

Syntax: ps -u user_name

where,

- u:** Display all process related to a specific user.

Eg:

| PID | TTY | TIME | CMD |
|-----|-----|----------|------------|
| 1 | ? | 00:00:01 | systemd |
| 2 | ? | 00:00:00 | kthreadd |
| 3 | ? | 00:00:00 | rcu_gp |
| 4 | ? | 00:00:00 | rcu_par_gp |

Figure 12.7: Sample output

- Find all process of a specific process name.

```
Syntax: ps -C process_name
```

where,

-C: Display all process related to a specific process name.

Eg:

```
root@lavatech:~# ps -C chrome
  PID TTY      TIME CMD
390536 ?    00:23:11 chrome
390571 ?    00:00:00 chrome
390572 ?    00:00:00 chrome
390576 ?    00:00:00 chrome
390597 ?    00:26:01 chrome
```

Figure 12.8: Sample output

- Find process name of a specific process ID (PID).

```
Syntax: ps -f -p PID
```

where,

-p: Display all process related to a PID.

-f: Display full detailed information.

Eg:

```
root@lavatech:~# ps -f -p 410857
UID        PID  PPID  C STIME TTY      TIME CMD
jack      410857  390576  0 16:59 ?    00:00:28 /opt/google/chrome/chrome
root@lavatech:~#
```

Figure 12.9: Sample output

- **pstree** - Display a tree of processes.

Syntax: pstree

Eg: Option with **pstree** command:

```
jack@lavatech:~$ pstree
systemd—ModemManager—2*[{ModemManager}]
          | NetworkManager—2*[{NetworkManager}]
          | accounts-daemon—2*[{accounts-daemon}]
          | acpid
          | avahi-daemon—avahi-daemon
          | blkmapd
          | bluetoothd
```

Figure 12.10: Sample output

- **-u**: Display all process related to a specific user.

Syntax: pstree -u username

Eg:

```
# pstree -u jack
```

- **fuser**: Find PID of running process accessing specific file/directory.

Syntax: fuser -v file_folder

Eg:

```
root@lavatech:~# fuser -v /var
                  USER      PID ACCESS COMMAND
/var:            root      kernel mount /var
root@lavatech:~#
root@lavatech:~# fuser -v /home
                  USER      PID ACCESS COMMAND
/home:           root      kernel mount /home
```

Figure 12.11: Sample output

- **pidof:** Find PID of all running process using the process name.

Syntax: pidof process_name

Eg:

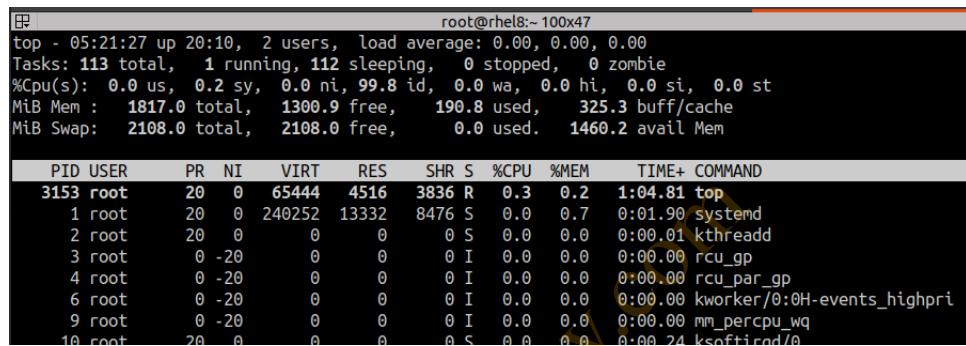
```
root@lavatech:~# pidof chrome
417510 414505 410857 408821 403111 393128 392899 392885 391949 391935
391873 391838 391818 391763 391688 391675 391647 391639 391611 39157
4 391516 391507 391504 391503 391478 391477 391446 391334 391155 3910
87 391086 390810 390784 390776 390766 390757 390747 390716 390696 390
683 390655 390600 390598 390597 390576 390572 390571 390536
root@lavatech:~# _
```

Figure 12.12: Sample output

- **top:** Displays real time process with CPU load, memory usage etc.

Syntax: top

Eg: Press "**CTRL+C**" or "q" to exit from the command output.



```

root@rhel8:~ 100x47
top - 05:21:27 up 20:10, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 113 total, 1 running, 112 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.2 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1817.0 total, 1300.9 free, 199.8 used, 325.3 buff/cache
MiB Swap: 2108.0 total, 2108.0 free, 0.0 used. 1460.2 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
3153 root 20 0 65444 4516 3836 R 0.3 0.2 1:04.81 top
1 root 20 0 240252 13332 8476 S 0.0 0.7 0:01.90 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kthreadd
3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-events_highpri
9 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
10 root 20 0 0 0 0 S 0.0 0.0 0:00.24 ksoftirqd/0

```

Figure 12.13: Top command output

Options with **top** command:

-u: Display all process of specific user.

Syntax: top -u username

Eg: **top** command to display all process associated with "**root**" user.

```
# top -u root
```

12.1.5 Killing process

Commands to kill a process:

- **kill**: Kills a specific process by sending some signal to it.

Syntax: kill [options] [process_id]

Eg: Start a process and kill it with it's PID.

```
[root@rhel8 ~]# sleep 1000 &
[2] 34325
[root@rhel8 ~]#
[root@rhel8 ~]#
[root@rhel8 ~]# ps -aux | grep sleep
root  34325  0.0  0.0    7308   840 pts/0      S     04:04   0:00 sleep 1000
root  34327  0.0  0.0   12136  1116 pts/0      S+    04:04   0:00 grep --color=auto sleep
[root@rhel8 ~]#
[root@rhel8 ~]#
[root@rhel8 ~]# kill 34325
[root@rhel8 ~]#
```

Figure 12.14: Sample output

Options with **kill** command:

- **-l**: List all signal associated with kill command.

Syntax: kill -l

Eg:

```
[root@rhel8 ~]# kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE      9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Figure 12.15: Sample output

- **-signal_number:** Supply signal while killing a process.

Syntax: kill -signal_number process_id

Eg:

```
[root@rhel8 ~]# sleep 1000 &
[2] 34421
[root@rhel8 ~]#
[root@rhel8 ~]#
[root@rhel8 ~]# ps -aux | grep sleep
root      34421  0.0  0.0  7308  896 pts/0    S     04:19   0:00 sleep 1000
root      34423  0.0  0.0 12136 1144 pts/0    S+    04:19   0:00 grep --color=auto sleep
[root@rhel8 ~]#
[root@rhel8 ~]#
[root@rhel8 ~]# kill -9 34421
[root@rhel8 ~]#
[2]- Killed                  sleep 1000
[root@rhel8 ~]#
```

Figure 12.16: Sample output

Meaning of some important signals:

* **SIGHUP (1):**

- Process hang due to death of controlling process.
- Eg:

```
# sleep 6000 &
[1] 421019

# kill -1 421019
```

* **SIGINT (2):**

- Issued if the user sends an interrupt signal (Ctrl + C).
- Eg:

```
# sleep 6000 &
[1] 421019

# kill -2 421019
[1]+ Interrupt sleep 6000
```

* **SIGQUIT (3):**

- Issued if the user sends a quit signal (Ctrl + D).
- Eg:

```
# sleep 6000 &
[1] 421019

# kill -3 421019
[1]+  Quit (core dumped) sleep 6000
```

* **SIGKILL (9):**

- If a process gets this signal it must quit immediately and will not perform any clean-up operations.
- Eg:

```
# sleep 6000 &
[1] 421019

# kill -9 421019
[1]+ Killed sleep 6000
```

* **SIGTERM (15):**

- Software termination signal (sent by kill by default).
- Eg:

```
# sleep 6000 &
[1] 421019

# kill -15 421019
[1]+ Killed sleep 6000
```

*** SIGSTOP (19):**

- Issued if the user sends a stop signal (Ctrl + Z).
- Eg:

```
# sleep 6000 &
[1] 421019

# kill -19 421019
[1]+ Stopped sleep 6000
```

*** SIGCONT (18):**

- Issued if the user sends a resume signal.
- Eg:

```
# sleep 6000 &
[1] 421019

# kill -18 421019
# jobs
[1]+ Running sleep 6000 &
```

- **killall:** Kill multiple running process by process name instead of it's process ID. Zombie process needs to be killed using process name as they won't have process ID.

Syntax: killall [options] process_name

Eg:

```
root@rhel8 ~]# sleep 10000 &
[2] 34475
root@rhel8 ~]#
root@rhel8 ~]# ps -aux | grep sleep
root      34475  0.0  0.0    7308   868 pts/0    S     04:26   0:00 sleep 10000
root      34479  0.0  0.0   12136  1192 pts/0    S+    04:26   0:00 grep --color=auto sleep
root@rhel8 ~]#
root@rhel8 ~]# killall sleep
[2]- Terminated                 sleep 10000
root@rhel8 ~]#
```

Figure 12.17: Sample output

- **pkill:** The pkill command, like killall, can signal multiple processes.

Syntax: pkill process_name

Eg:

```
# pkill sleep
```

12.1.6 Process priority

- The priority value is the process's actual priority used by the Linux kernel to schedule a task.
- Priorities are 0 to 139 in which **0 to 99** is for real-time and **100 to 139** for users.

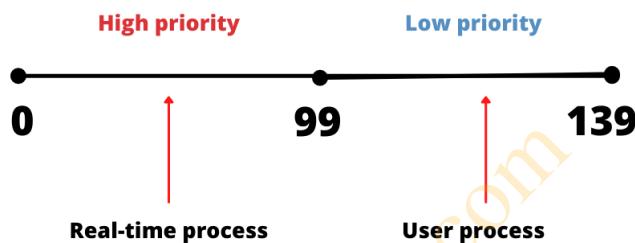


Figure 12.18: Priority values

Command to check priority of process:

Syntax: top

Eg:

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|------|------|----|-----|--------|-------|------|---|------|------|---------|-----------------------------|
| 3153 | root | 20 | 0 | 65444 | 4516 | 3836 | R | 0.3 | 0.2 | 1:04.81 | top |
| 1 | root | 20 | 0 | 240252 | 13332 | 8476 | S | 0.0 | 0.7 | 0:01.90 | systemd |
| 2 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.01 | kthreadd |
| 3 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | rcu_gp |
| 4 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | rcu_par_gp |
| 6 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | kworker/0:0H-events_highpri |
| 9 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | mm_percpu_wq |
| 10 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.24 | ksfticed/0 |

Figure 12.19: Priority values

How to start a process with a higher or lower priority than default priority?

- **Solution: Nice value**
- Regular user can change priorities of their own jobs.
- Priorities assigned by regular user is known as '**niceness**'.
- The nice value range is -20 to +19 where **-20 is highest, 0 default and +19 is lowest**.

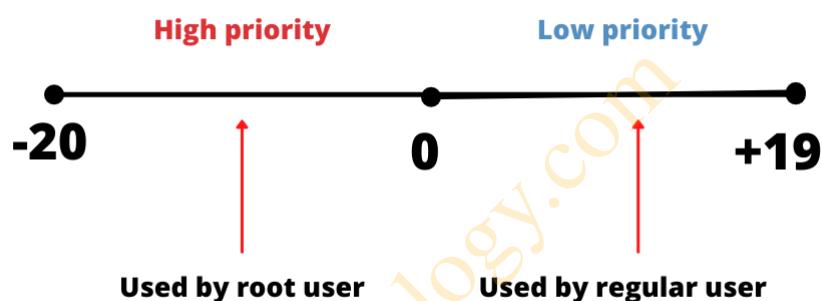


Figure 12.20: Nice priority values

Command to check priority & nice value of process:

Syntax: ps -eo user,priority,nice,comm | grep command_name

Eg:

```
root@lavatech:~# ps -eo user,priority,nice,comm | grep sleep
root      20   0 sleep
root@lavatech:~#
```

Figure 12.21: Sample output

Running a command through the ‘nice’ command

- nice: Run a program with modified scheduling priority.

Syntax: nice -n level command

Eg:

```
root@lavatech:~# nice -n 17 sleep 50000 &
[1] 446961
root@lavatech:~#
root@lavatech:~# ps -eo user,priority,nice,comm | grep sleep
root      37  17 sleep
root@lavatech:~#
```

Figure 12.22: Sample output

The relation between nice value and priority is:

Formula: Priority_value = Nice_value + 20

Notice in the above screenshot, priority is 17+20 which is 37.

How to change the priority of a running process?

- **Solution: renice command**
- renice: Alter priority of running processes.

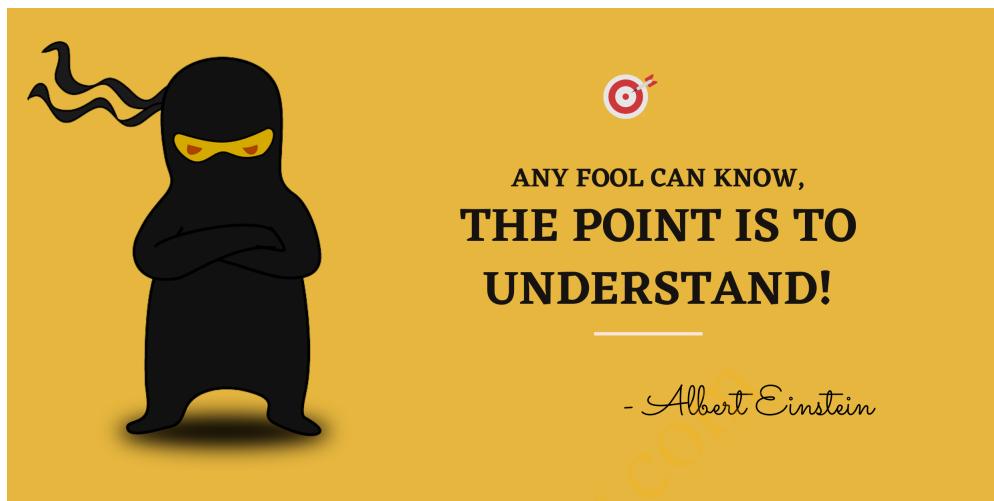
Syntax: renice -n level PID

Eg:

```
# renice -n 15 446961
```

```
root@lavatech:~# ps -eo pid,user,priority,nice,comm | grep sleep
446961 root      37  17 sleep
root@lavatech:~# renice -n 15 446961
446961 (process ID) old priority 17, new priority 15
root@lavatech:~#
root@lavatech:~# ps -eo pid,user,priority,nice,comm | grep sleep
446961 root      35  15 sleep
root@lavatech:~#
```

Figure 12.23: Sample output

12.1.7 Practice

- 1. Which of the following statement is true about Linux process? (Select all that applies.)**
 - (a) A process is a program running in RAM and utilizing CPU.
 - (b) A daemon is Linux process that runs in background.
 - (c) A process has process name, PID, process state, PPID.
 - (d) A process that is dead but still utilizing computer memory and CPU is a zombie process.
- 2. Which of the following command will be executed in the background?**
 - (a) sleep 4000 &
 - (b) cat /etc/passwd &&
 - (c) ifconfig
 - (d) ls -ld /etc/repos.d
- 3. Which of the following command is used to check status of a daemon or service?**
 - (a) systemctl daemon_name status
 - (b) systemctl daemon_name report

- (c) systemctl report daemon_name
 - (d) systemctl status daemon_name
- 4. Select the correct symbol representing running process.**
- (a) R
 - (b) D
 - (c) T
 - (d) W
- 5. Select the correct symbol representing stopped process.**
- (a) S
 - (b) D
 - (c) T
 - (d) W
- 6. Select the correct symbol representing zombie process.**
- (a) R
 - (b) D
 - (c) Z
 - (d) W
- 7. Which of the following command provides real time process details with CPU load, memory usage etc.?**
- (a) iostat
 - (b) netstat
 - (c) top
 - (d) ps
- 8. Which of the following command displays all process executed by user named "ravi"? (Select all that applies.)**
- (a) iostat -u ravi

- (b) netstat -u ravi
 - (c) top -u ravi
 - (d) ps -u ravi
9. Which of the following command kills a process using process ID?
- (a) kill
 - (b) killall
 - (c) terminate
 - (d) stop
10. Which of the following command kills a process using process name?
- (a) kill
 - (b) killall
 - (c) terminate
 - (d) stop
11. Which of the following command display all kill signals?
- (a) kill -l
 - (b) kill -s
 - (c) kill -p
 - (d) kill -t

12.2 CPU management

In this section, you are going to learn:

1. **What is CPU?**
2. **CPU core & CPU threads**
3. **Types of CPU**
4. **Understanding CPU calculation**
5. **Checking CPU**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

12.2.1 What is CPU?

- CPU stands for **Central Processing Unit**.
- CPU is the brain of the computer.
- CPU is also called as the **processor**.
- It performs:
 - Arithmetic operations such as addition, subtraction etc.
 - Logical operations i.e. whether something is true or false, etc.
 - Synchronizes computer operations.
- Intel and AMD are leading in processor manufacturing.



Figure 12.24: Intel dual-core processor

12.2.2 CPU core & CPU threads

What is a CPU cores?

- A “core” is a single CPU.
- A CPU core is a physical hardware component.

What is a CPU thread?

- A core can be virtually splitted into parts called **thread**.
- A core uses threads to offer more power to specific programs.
- A thread virtually divides one CPU core into 2 parts giving the effect of 2 cores.
- A thread implements **hyperthreading** which is responsible for multi-tasking.

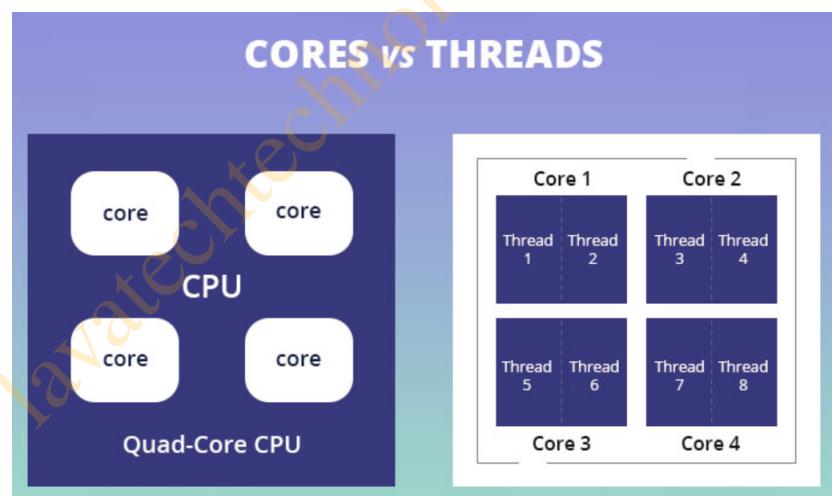


Figure 12.25: Core & Thread

12.2.3 Types of CPU

- **Single-core processors:**
 - A single CPU with **one core**.
 - Can execute only one command at a time.
 - It is not efficient in multi-tasking.

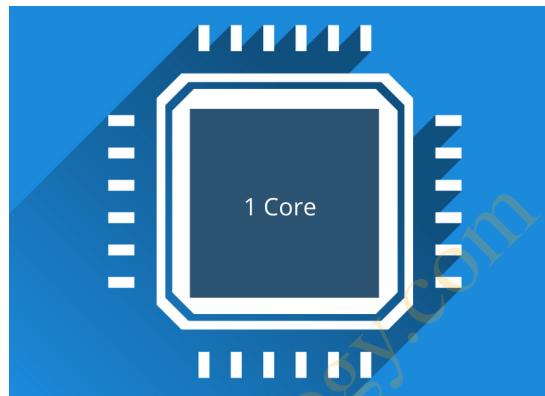


Figure 12.26: Single-core CPU

- **Dual-core(2 cores) processors:**
 - A single CPU with **two cores**.
 - Functions like dual CPU acting like one.

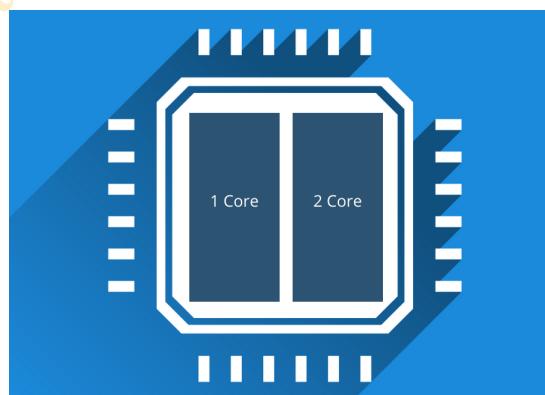


Figure 12.27: Dual-core CPU

- **Quad-core(4 cores) processors:**

- The quad-core CPU comes with **four cores** on a single CPU.
- Such types of CPU are used by gamers.

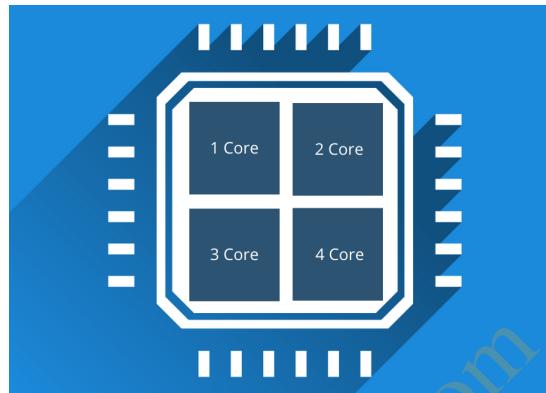


Figure 12.28: Quad-core CPU

- **Hexa Core(6 cores) processors:**

- The hexa-core CPU comes with **six cores** on a single CPU.
- Intel has launched with Inter core i7 in 2010 with Hexa core processor.

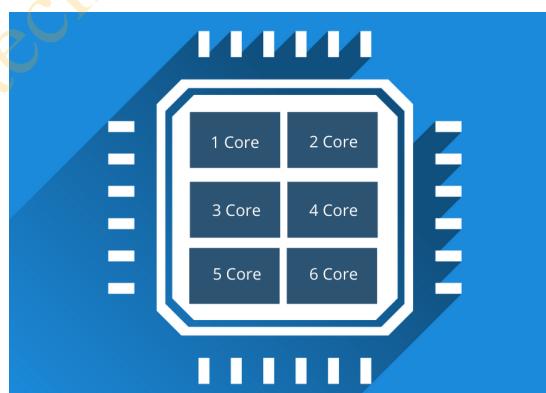


Figure 12.29: Hexa-core CPU

- **Octa Core(8 cores) processors:**
 - It is multiple core processor with 8 cores.
 - For gaming, video editing, and other processor-intensive applications, eight cores or more is ideal.

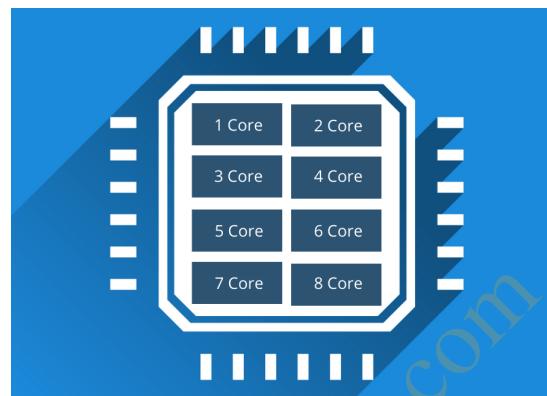


Figure 12.30: Octa-core CPU

- **Deca-Core(10 cores) processors:**
 - It is multiple core processor with 10 cores.
 - Ideal for gamers and users who do heavy multitasking.

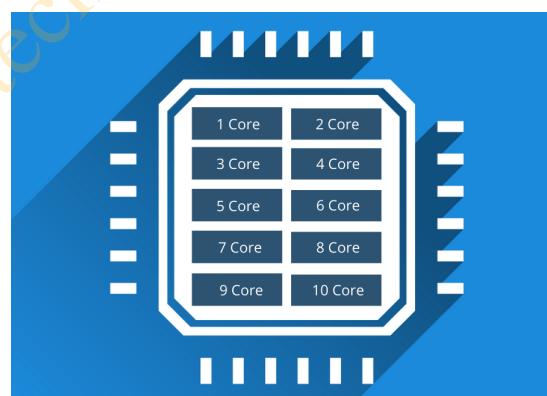


Figure 12.31: Deca-core CPU

12.2.4 Understanding CPU calculation

When you say "**How many CPU(s) are present in your system?**", it either means:

- How many physical CPU(s) are present?

To calculate physical CPU(s), the formula is:

Formula: No. of physical cpu = No. of socket X No. of cores

Here,

- Socket means physical CPU(s) present on the motherboard.
- Core means full CPU cores there are present per physical CPU.

- How many logical CPU(s) are present?

To calculate logical CPU(s), the formula is:

Formula: No. of physical cpu = No. of socket X No. of cores X
No. of threads

Here,

- Socket means physical CPU(s) present on the motherboard.
- Core means full CPU cores there are present per physical CPU.
- Threads means number of threads present on single core.

How to check number of CPU cores, number of threads and number of sockets in OS?

Solution: All these details are stored in /proc/cpuinfo.

Display number of socket:

```
Syntax: grep "physical id" /proc/cpuinfo | sort | uniq | wc -l
```

Display number of logical CPU:

```
Syntax: grep "^processor" /proc/cpuinfo | wc -l
```

Display number of CPU cores:

```
Syntax: grep "cpu cores" /proc/cpuinfo | uniq
```

12.2.5 Commands for CPU

- lscpu: Command to display CPU details.

Syntax: lscpu

Eg:

```
jack@lavatech:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
Address sizes:         39 bits physical, 48 bits virtual
CPU(s):                12
On-line CPU(s) list:  0-11
Thread(s) per core:   2
Core(s) per socket:   6
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 158
Model name:            Intel(R) Core(TM) i7-9850H CPU @ 2.60GHz
Stepping:               13
```

Figure 12.32: Sample output

Output explanation: Here,

- Number of physical CPU(s) = 6
- Number of logical CPU(s) = 12

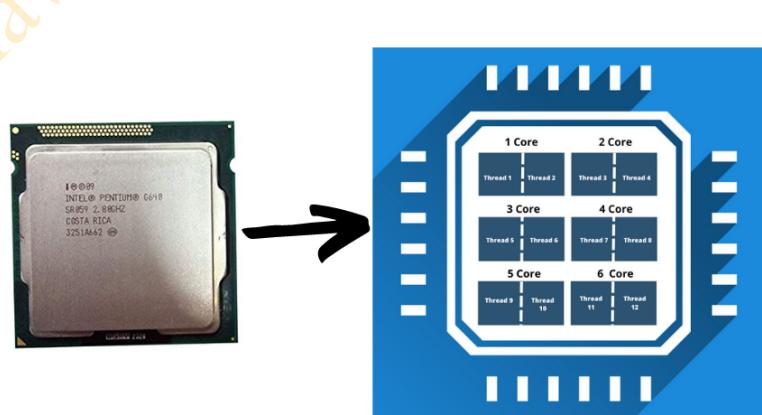


Figure 12.33: Sample output

- **iostat:** Reports CPU statistics and I/O statistics for devices and partitions.

Options with **iostat** command:

- **-c:** Displays CPU utilization report.

Syntax: iostat -c

Eg:

```
[root@server ~]# iostat -c
Linux 4.18.0-348.el8.x86_64 (server.example.com) 03/26/2022 _x86_64_ (2 CPU)

avg-cpu: %user %nice %system %iowait %steal %idle
          0.06   0.00   0.28   0.02   0.00  99.65
```

Figure 12.34: Sample output

Output explanation:

- * **%user:** Show CPU utilization of user application.
- * **%nice:** Show CPU utilization of user application with nice priority.
- * **%system:** Show CPU utilization of the system kernel.
- * **%iowait:** Show CPUs idle percentage during outstanding disk I/O request.
- * **%steal:** Show wait by the virtual CPUs while the hypervisor was servicing another virtual processor.
- * **%ideal:** Show CPUs idle percentage when the system did not have an outstanding disk I/O request.

- **vmstat:**

- Stands for Virtual Memory STATistics.
 - It is a computer monitoring tool that collects and displays system memory, processes, interrupts, pagging and block I/O.

Option with **vmstat** command:

-s: Displays a table of event counters and memory statistics.

Syntax: vmstat -s

Eg:

```
[root@server ~]# vmstat -s
 1860564 K total memory
 191584 K used memory
  93116 K active memory
 283124 K inactive memory
1343668 K free memory
   3164 K buffer memory
 322148 K swap cache
2158588 K total swap
     0 K used swap
2158588 K free swap
 2435 non-nice user cpu ticks
    73 nice user cpu ticks
  5505 system cpu ticks
4128160 idle cpu ticks
   667 IO-wait cpu ticks
  4202 IRQ cpu ticks
 1490 softirq cpu ticks
      0 stolen cpu ticks
261350 pages paged in
116617 pages paged out
      0 pages swapped in
      0 pages swapped out
1419089 interrupts
2113022 CPU context switches
1648281184 boot time
 12761 forks
```

Figure 12.35: Sample output

- **sar:**

- The **sar** command is a performance monitoring utility.
- It collects reports ongoing basis and saves the performance data.

Syntax: sar

Eg:

```
[root@server ~]# sar
Linux 4.18.0-348.el8.x86_64 (server.example.com)      03/27/2022      _x86_64_      (2 CPU)
05:49:50 AM   CPU   %user   %nice   %system   %iowait   %steal   %idle
05:50:00 AM   all    0.10    0.10    0.42     0.00     0.00   99.38
06:00:01 AM   all    0.06    0.01    0.25     0.02     0.00   99.66
Average:      all    0.06    0.01    0.26     0.02     0.00   99.65
```

Figure 12.36: Sample output

Refer **iostat** command for output explaination.

12.2.6 Understanding CPU consumption

- On a server with 6 cores and 2 threads per core (total of 12 CPUs), the usage could go up to 1200%.
- **Command to display process by maximum CPU consumption:**

Syntax: top

Press "P" to sort task list by processor usage

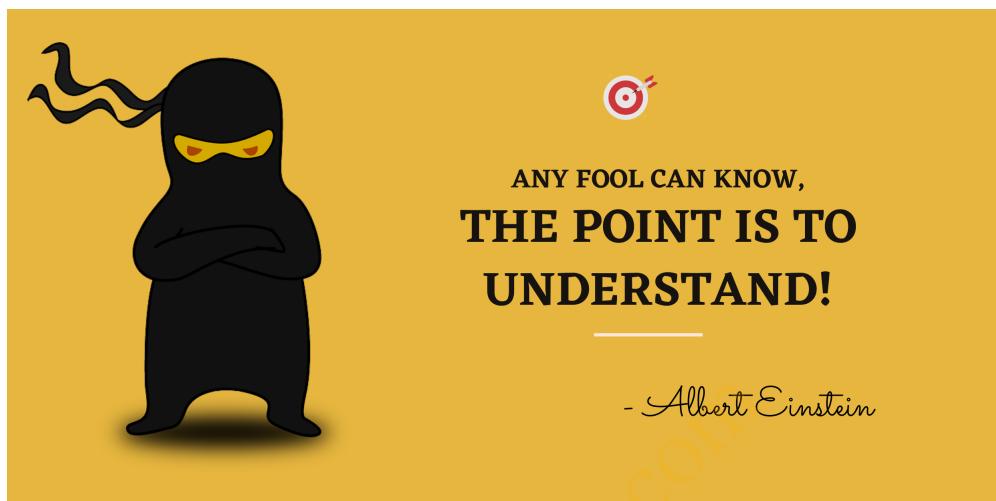
Eg:

| top - 16:59:22 up 7 days, 9:49, 1 user, load average: 2.18, 2.73, 2.81 | | | | | | | | | | |
|---|------|-----------|----|--------------|-------------|---------------|---|-------------|------------|------------------------|
| Tasks: 433 total, 2 running, 423 sleeping, 8 stopped, 0 zombie | | | | | | | | | | |
| %Cpu(s): 20.9 us, 1.2 sy, 0.0 ni, 77.4 id, 0.4 wa, 0.0 hi, 0.1 si, 0.0 st | | | | | | | | | | |
| MiB Mem : 31842.8 total, 11332.3 free, 13311.2 used, 7199.3 buff/cache | | | | | | | | | | |
| MiB Swap: 439.7 total, 344.6 free, 95.1 used. 14375.2 avail Mem | | | | | | | | | | |
| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ COMMAND |
| 404825 | root | 20 | 0 | 16.4g | 127816 | 41116 | S | 142.9 | 0.4 | 119:55.49 chrome |
| 403306 | root | 20 | 0 | 32.5g | 2.8g | 123968 | R | 82.4 | 9.1 | 48:59.72 chrome |
| 390597 | root | 20 | 0 | 16.8g | 257948 | 149316 | S | 15.3 | 0.8 | 15:22.74 chrome |
| 390598 | root | 20 | 0 | 16.3g | 132340 | 93732 | S | 6.0 | 0.4 | 18:00.55 chrome |
| 391503 | root | 20 | 0 | 25.3g | 246972 | 107444 | S | 5.6 | 0.8 | 7:40.18 chrome |
| 390536 | root | 20 | 0 | 16.9g | 497764 | 201404 | S | 4.7 | 1.5 | 18:45.50 chrome |
| 391446 | root | 20 | 0 | 16.6g | 75496 | 64472 | S | 1.3 | 0.2 | 1:08.60 chrome |

Figure 12.37: Sample output

In "top" what are us, sy, ni, id, wa, hi, si and st (for CPU usage)?

- us - %CPU time spent in user space
- sy - %CPU time spent in kernel space
- ni - %CPU time spent on low priority processes
- id - %CPU time spent idle
- wa - %CPU time spent in wait (on disk)
- hi - %CPU time spent with hardware interrupts
- si - %CPU time spent with software interrupts
- st - %CPU time stolen from a virtual machine

12.2.7 Practice

- 1. Which of the following is true about CPU? (Select all that applies.)**
 - (a) CPU stands for Central Processing Unit.
 - (b) CPU is also known as processor.
 - (c) CPU perform arithmetic & logical operations of computer.
 - (d) CPU has CPU cores & threads.

- 2. How many CPU cores are present in hexa core processor?**
 - (a) 6 cores
 - (b) 4 cores
 - (c) 2 cores
 - (d) 8 cores

- 3. Which of the following formula calculates total number of logical CPU?**
 - (a) No. of cores X No. of threads
 - (b) No. of socket X No. of threads
 - (c) No. of socket X No. of cores
 - (d) No. of socket X No. of cores X No. of threads

4. Which of the following command is used to display CPU details?

- (a) free -w
- (b) lscpu
- (c) fdisk -l
- (d) top

5. Which of the following file stores CPU details?

- (a) /proc/meminfo
- (b) /proc/cpu
- (c) /proc/cpuinfo
- (d) /proc/processor

12.3 Memory management

In this section, you are going to learn:

1. **What is RAM?**
2. **What is cache?**
3. **What is buffer?**
4. **Checking memory**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

12.3.1 What is RAM?

- RAM stands for Random Access Memory.
- Usage of RAM:
 - CPU uses RAM as **short term memory** for data storage.
 - Used by computer to store actively used data for quick access.
- How much RAM is recommended according to usage?
 - **6GB - 8GB** - Normal usage and internet browsing.
 - **16GB** - For spreadsheets and other office programs.
 - **32GB** or more - For gamers and multimedia creators.
 - **64GB & more** - For servers.



Figure 12.38: RAM

- Compared to HDD, RAM is expensive. RAM price according to size:
 - 2GB - Approx Rs. 950
 - 4GB - Approx Rs. 1750
 - 8GB - Approx Rs. 2940
- All memory details are stored in **/proc/meminfo**.

Syntax: less /proc/meminfo

12.3.2 What is cache?

- Cache is high speed storage area.
- Two types of cache are:
 - **Memory cache** also called CPU memory:

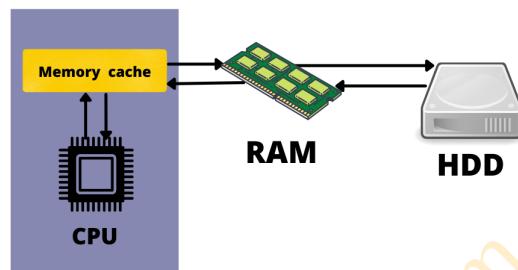


Figure 12.39: Memory cache

- * Memory cache is located inside the CPU.
- * Multiple running processes stored their information in memory cache, instead of searching the disk everytime.
- **Disk cache:**

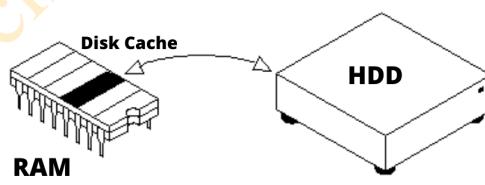


Figure 12.40: Disk cache

- * Disk cache can be part of the HDD or RAM.
- * Disk cache holds data that is frequently or recently read and is likely to be accessed next.

12.3.3 What is buffer?

- Buffer is located inside RAM.

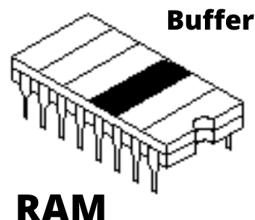


Figure 12.41: Buffer

- Buffer act as a temporary holding area for the CPU to manipulate data before transferring it to a device.
- Buffer usage eg:
 - **Copy operation:** In case of copying data from 1 device to another with device having different speeds, data is stored in buffer.
 - **Saving data to HDD:** Word processors write the data first in the buffer, and later updates the file in the disk with the contents of the buffer.
 - **Printing:** The documents to be printed is stored in a buffer, and the printer can then access this information at its own pace.

12.3.4 Commands for memory

Commands to display RAM related details:

- **free**: Display information about the total amount of the physical & swap memory in bytes.

Syntax: free

Eg:

```
jack@lavatech:~$ free
              total        used        free      shared  buff/cache   available
Mem:       32607060     19870024     1388500      3340640    11348536     8932856
Swap:      450260          107284      342976
```

Figure 12.42: Sample output

Understanding the output:

- **total** - Total RAM.
- **used** - Used memory. It is calculated as: **used = total - free - buffers - cache**.
- **free** - Free / unused memory.
- **shared** - It has no meaning. It is here only for backward compatibility.
- **buff/cache** - The combined memory used by the buffers and cache. This memory can be reclaimed at any time if needed by the applications.
- **available** - Amount of memory available for starting new applications, without swapping.

Note: You should always refer the **available** column to understand how much memory is available.

Options with **free** command:

- **-h**: Human readable format.

Syntax: free -h

Eg:

```
jack@lavatech:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:       31Gi       18Gi       1.3Gi      3.2Gi       10Gi       8.5Gi
Swap:    439Mi      104Mi      334Mi
```

Figure 12.43: Sample output

Notice the size is displayed in "**Gi**" and not **GB**. Let's understand difference between "**Gi**" & "**GB**".

UnitsPolicy

- * Base-2 units:
 - 1 KiB (Kibibyte) = 1,024 bytes (Note: big k)
 - 1 MiB (Mebibyte) = 1,024 KiB
 - **1 GiB (Gibibyte) = 1,024 MiB**
 - 1 TiB (Tebibyte) = 1,024 GiB
- * Base-10 units:
 - 1 kB (Kilobyte) = 1,000 bytes (Note: small k)
 - 1 MB (Megabyte) = 1,000 kB
 - **1 GB (Gigabyte) = 1,000 MB**
 - 1 TB (Terabytes) = 1,000 GB

To conclude,

- * 1 GB = 0.93 GiB
- * 1 GiB = 1.07 GB
- * This is approx 7% difference and can make huge difference.

- w: Display buffer and cache separately.

Syntax: free -w

Eg:

```
jack@lavatech:~$ free -h -w
              total        used        free      shared  buffers     cache available
Mem:       31Gi       18Gi       1.5Gi    3.7Gi  1.2Gi     10Gi     8.8Gi
Swap:      439Mi      147Mi      292Mi
```

Figure 12.44: Sample output

- Display top 10 processes consuming highest amount of RAM:

Syntax: ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head

Eg:

```
jack@lavatech:~$ ps -eo pid,ppid,cmd,\%mem,\%cpu --sort=-\%mem | head
  PID   PPID CMD                      %MEM %CPU
114181  1880 texstudio                12.7  0.3
  1643  1632 /usr/bin/pulseaudio --daemon 2.4  1.4
390536  1880 /opt/google/chrome/chrome - 1.3  3.1
  1880  1632 /usr/bin/gnome-shell       0.9  2.0
391688  390576 /opt/google/chrome/chrome - 0.9  1.2
179932  179864 /usr/share/code/code --type 0.8  0.3
391503  390576 /opt/google/chrome/chrome - 0.6  0.8
180143  179989 /usr/share/code/code --ms-e 0.6  0.0
391675  390576 /opt/google/chrome/chrome - 0.5  0.5
```

Figure 12.45: Sample output

- Command to sort all process with maximum memory consumption:

Syntax: top

Press "M" to sort task list by processor usage

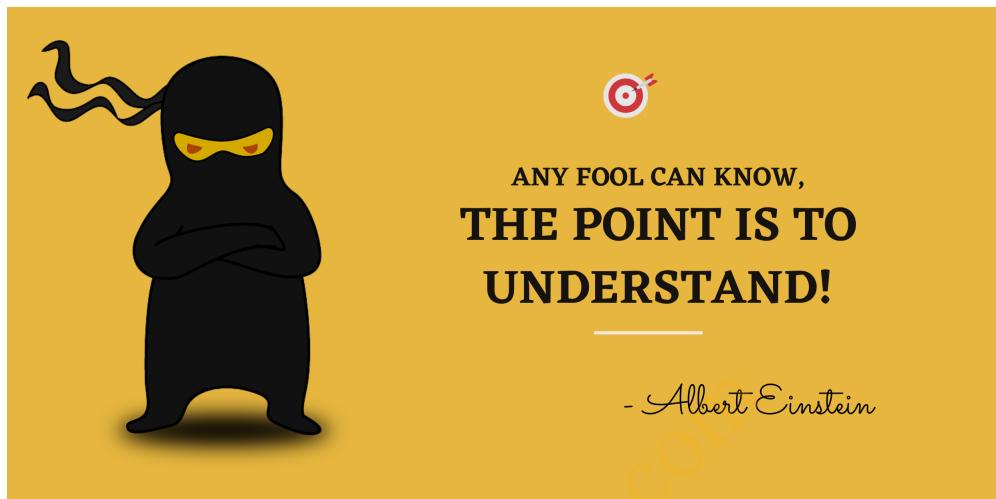
Eg:

| root@lavatech: ~ 82x41 | | | | | | | | | | | | | |
|--|------|----|----|---------|--------|--------|---|------|------|-----------|------------|--|--|
| top - 20:22:57 up 8 days, 13:12, 1 user, load average: 0.57, 0.47, 0.51 | | | | | | | | | | | | | |
| Tasks: 472 total, 1 running, 463 sleeping, 8 stopped, 0 zombie | | | | | | | | | | | | | |
| %Cpu(s): 1.9 us, 0.8 sy, 0.0 ni, 97.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st | | | | | | | | | | | | | |
| MiB Mem : 31842.8 total, 4215.3 free, 13726.5 used, 13901.1 buff/cache | | | | | | | | | | | | | |
| MiB Swap: 439.7 total, 347.0 free, 92.7 used. 8107.5 avail Mem | | | | | | | | | | | | | |
| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND | | |
| 114181 | root | 20 | 0 | 9159444 | 4.4g | 110304 | S | 0.0 | 14.1 | 39:01.83 | texstudio | | |
| 1643 | root | 20 | 0 | 5655448 | 1.0g | 11476 | S | 1.7 | 3.2 | 157:47.71 | pulseaudio | | |
| 410237 | root | 20 | 0 | 2784056 | 973156 | 942052 | S | 1.7 | 3.0 | 11:57.35 | VBoxHeadl+ | | |
| 390536 | root | 20 | 0 | 17.0g | 683804 | 210844 | S | 0.0 | 2.1 | 34:40.72 | chrome | | |
| 441962 | root | 20 | 0 | 26.0g | 511464 | 122940 | S | 13.6 | 1.6 | 20:37.72 | chrome | | |
| 1880 | root | 20 | 0 | 5173496 | 350380 | 62456 | S | 0.3 | 1.1 | 247:44.67 | gnome-she+ | | |

Figure 12.46: Sample output

When should I start to worry about my RAM?

- Below values in **free** command shows **healthy Linux system**:
 - **Free memory** is close to **0**.
 - **Used memory** is close to **total**.
 - **Available memory** should have approx. **20%+ of total**.
 - Swap used does not change.
- Below values are **warning signs** of a genuine low memory situation:
 - **Available memory** is close to **zero**.
 - **Swap used increases or fluctuates**.
 - Command "dmesg | grep oom-killer" shows the **OutOfMemory-killer** at work.

12.3.5 Practice

- 1. Which of the following is true about RAM? (Select all that applies.)**
 - (a) RAM stands for Random Access Memory.
 - (b) RAM is a computer's short-term memory, which it uses to handle all active tasks and apps.
 - (c) RAM is used to save file content permanently.
 - (d) RAM is used to perform arithmetic & logical processing.
- 2. Which of the following is CPU memory?**
 - (a) RAM
 - (b) Buffer
 - (c) Memory cache
 - (d) Disk cache
- 3. Which of the following operations uses buffer? (Select all that applies.)**
 - (a) Copying data from one device to another having different data transfer speed.
 - (b) Print operation.

- (c) Holding frequently access data from HDD.
 - (d) Writing data into a file & saving it to HDD.
- 4. Which of the following command is used to display physical & swap memory?**
- (a) vmstat
 - (b) free
 - (c) fdisk
 - (d) netstat
- 5. Which of the following file contains memory details?**
- (a) /proc/cpuinfo
 - (b) /proc/meminfo
 - (c) /etc/fstab
 - (d) /proc/pid
- 6. Which of the following are the warning signs of low memory situation?**
- (a) Swap usage is increasing or fluctuating frequently.
 - (b) Available memory is nearing 0.
 - (c) Used memory is close to total memory.
 - (d) Free memory is close to 0.



BELIEVE IN
YOURSELF

13. Scheduling Jobs

13.1 Crontab in detail

In this section, you are going to learn:

1. **Introduction to crontab**
2. **Command to set crontab**
3. **Cronjobs examples**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

13.1.1 Introduction to crontab

- The cron daemon runs jobs/processes on your system at a scheduled date/time.
- Cron daemon reads the **crontab (cron tables)** for commands and scripts.
- Every user can set crontab for themselves.
- Crontab related files:
 - Configuration file for crontab: **/etc/crontab**
 - User specific crontab files: **/var/spool/cron/{user_name}**
 - Cron log file: **/var/log/cron**
- Syntax of crontab jobs:

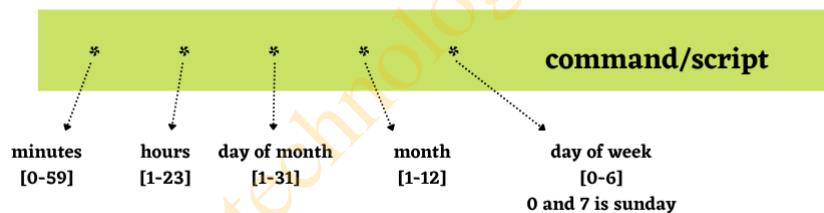


Figure 13.1: crontab file syntax

13.1.2 Command to set crontab

crontab: Command to maintains crontab files for individual users.

- **-e:** Edit cron file
- **-u:** Username

Syntax: `crontab -u username -e`

Eg:

```
# crontab -u jack -e
*/1 * * * * date > /tmp/one.txt
```

Note: Above command would configure cronjob in **/var/spool/cron/jack** file

- **-l:** List all the cronjob for current user.

Syntax: `crontab -u username -l`

- **-r:** Remove all cronjob at a time for a particular user.

Syntax: `crontab -u username -r`

13.1.3 Examples of cronjob

- Configure cronjob that would execute script **/home/jack/one.sh** every 2 minutes:

```
*/2 * * * * /home/jack/one.sh
```

Note: Use "*/x" to execute a job every "x" minutes.

- Configure cronjob that would execute script **/home/jack/one.sh** at 1:30pm every day:

```
30 13 * * * /home/jack/one.sh
```

- Configure cronjob that would execute script **/home/jack/one.sh** at 9:30am & 2:30pm every day:

```
30 9,14 * * * /home/jack/one.sh
```

- Configure cronjob that would execute script **/home/jack/one.sh** on 25th of each month at 7am:

```
* 7 25 * * /home/jack/one.sh
```

- Configure cronjob that would execute script **/home/jack/one.sh** from 15 to 21 date of Feb month at 7am:

```
* 7 15-21 2 * /home/jack/one.sh
```

Note: For range, use "-".

- Configure cronjob that would execute script **/home/jack/one.sh** for first 6 months on 5,10,15,29 date at 12am:

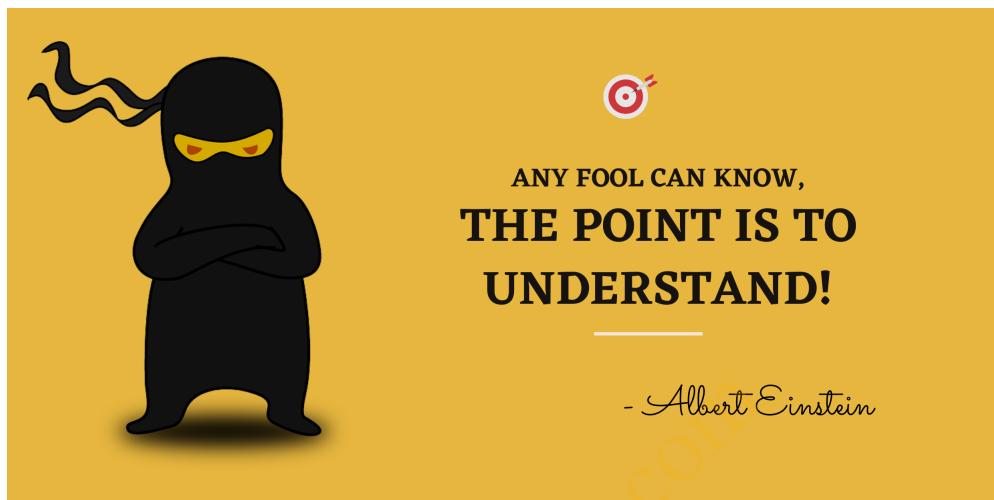
```
* 12 5,10,15,20 1-6 * /home/jack/one.sh
```

- Configure cronjob that would execute script **/home/jack/one.sh** that run on each monday & sunday of each month at 4:30am:

```
30 4 * * 0,1 /home/jack/one.sh
```

- Configure cronjob that would execute script **/home/jack/one.sh** that run on 3rd week of monday in each month at 7:10pm:

```
10 19 15-21 * monday /home/jack/one.sh
```

13.1.4 Practice

- 1. Which of the following file stores cron logs?**
 - (a) /var/log/cronjobs
 - (b) /var/log/cron
 - (c) /var/log/crontab
 - (d) /var/log/crond
- 2. Which of the following directory stores user specific crontab files?**
 - (a) /etc/crontab.d/
 - (b) /var/spool/crond/
 - (c) /var/spool/cron/
 - (d) /etc/crontab/
- 3. Select the correct command to edit cron file.**
 - (a) crontab -u username -e
 - (b) cronjob -u username -e
 - (c) cron -u username -e
 - (d) crontab –user username -e
- 4. Select the correct command to list cronjob of specific user.**

- (a) crontab -u username -list
- (b) cronjob -u username list
- (c) crontab -u username -l
- (d) crontab –user username -l

5. Select the correct command to remove cronjob of specific user.

- (a) crontab -u username -r
- (b) cronjob -u username remove
- (c) crontab -u username –remove
- (d) crontab –user username -r



Work hard
in silence,
LET SUCCESS
make the noise



14. Networking

14.1 Networking essentials

In this section, you are going to learn:

1. What is a network?
2. Types of network
3. Network connecting device

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

14.1.1 Networking & it's types

What is a network?

A network **consists of two or more computers that are linked in order to share resources (such as printers and CDs)**, exchange files, or allow electronic communications.

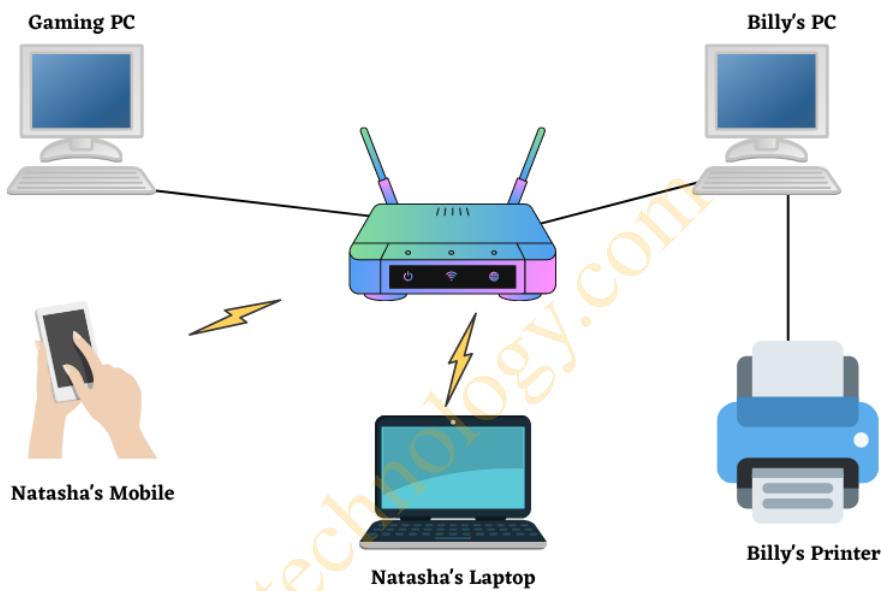
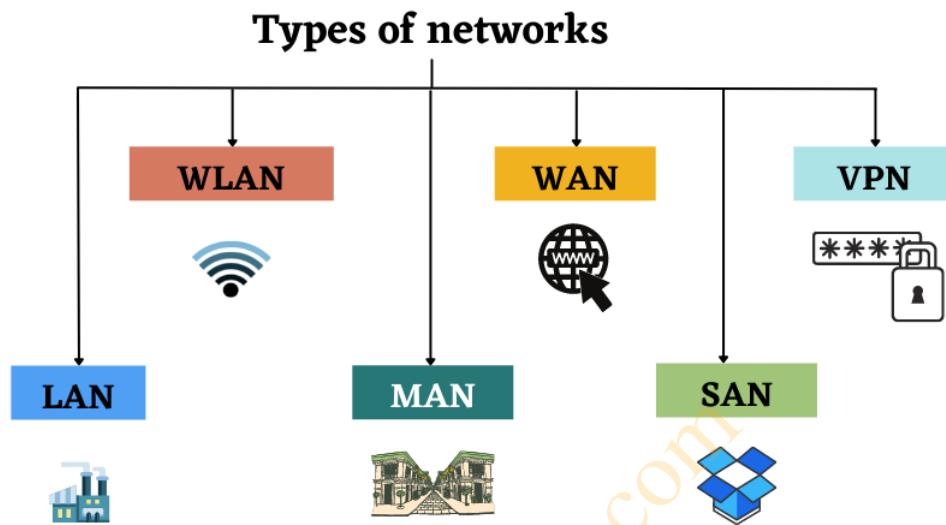


Figure 14.1: Network of device

Let's see the types of network available:



- **Local Area Network (LAN):**
 - Connect computers across short distances (like group of buildings close to each other) to share information/resources.
 - Companies uses LANs.
- **Wireless Local Area Network (WLAN):**
 - Functions like a LAN.
 - WLANs make use of wireless network technology, such as Wi-Fi.
- **Wide Area Network (WAN):**
 - Connects computers together across longer physical distances.
 - Eg: The Internet
 - Maintained by multiple administrators or the public.
- **Metropolitan Area Network (MAN):**
 - Larger than LANs but smaller than WANs.
 - Span an **entire geographic area** (eg. town or city).
 - Handled by single person or company.

- **Storage-Area Network (SAN):**
 - Connects shared pools of storage devices to several servers.
 - SAN doesn't rely on a LAN or WAN.
 - Storage resources are placed into high-performance network.
- **Virtual Private Network (VPN):**
 - Private network across the Internet.
 - Lets its users send and receive data over private network using Internet.

lavatechtechnology.com

14.1.2 Network connecting device

- **Modem & Router:**
 - **Modem:** Send or receive **data over telephone or cable lines.**
 - **Router:** **Supply the Internet connection** provided by modem to your wired & wireless devices.

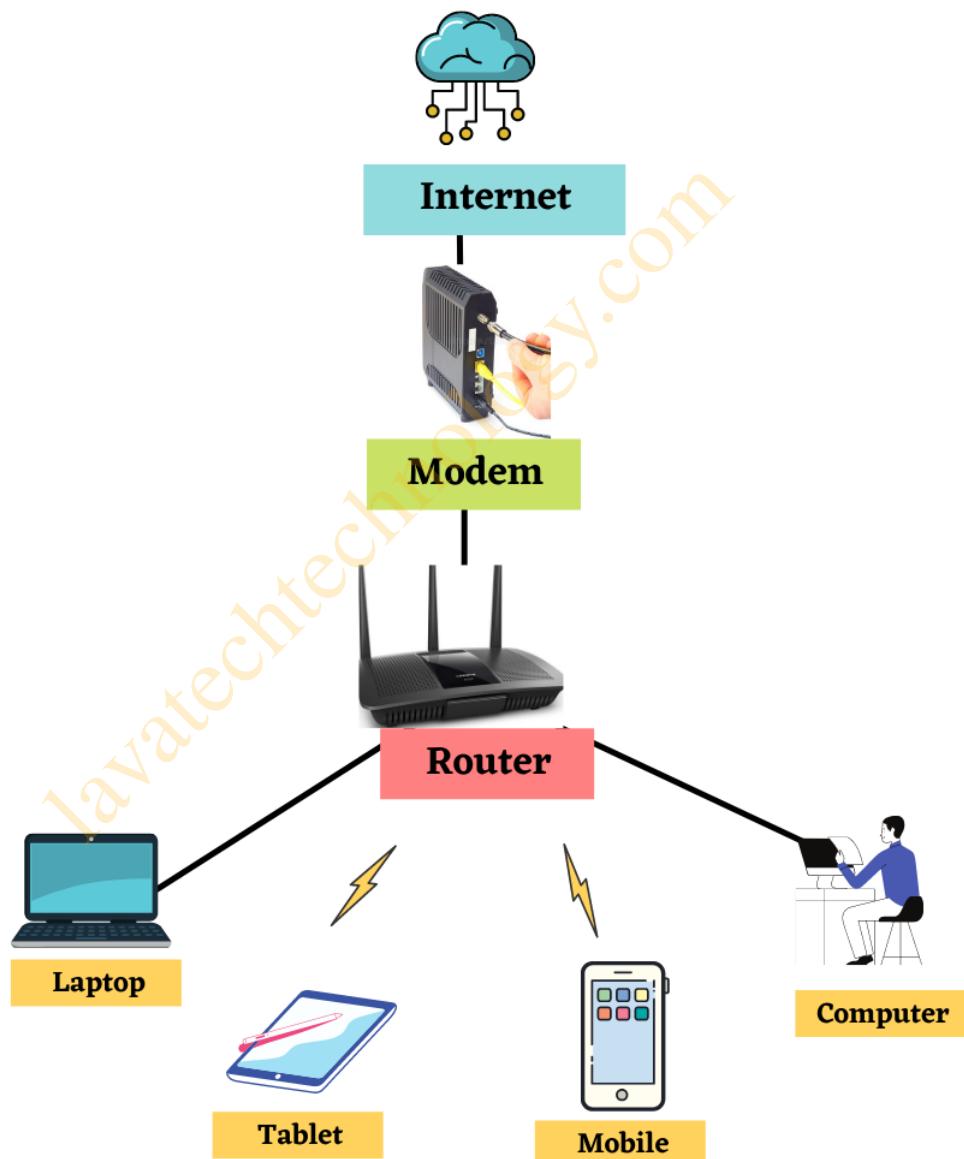


Figure 14.2: Modem & router

- **Router & Switch:**

- **Router:** Used for inter-network communication.
- **Switch:** A switch is used to provide additional ports, expanding the capability of the router.

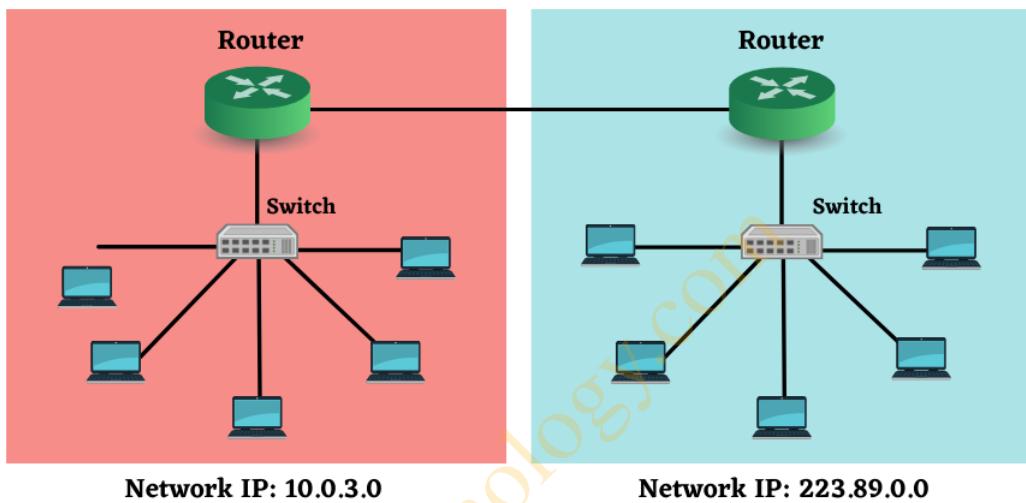


Figure 14.3: Router & Switch

- **RJ45 Connector:**

- RJ45 stands for **Registered Jack 45**.
 - 8-pin jack used to **connects a computer to a local area network (LAN)**.



Figure 14.4: RJ45 connector

- **Ethernet card or NIC:**
 - Also called a "network interface card" (NIC).
 - A card that plugs into a slot on the motherboard and enables a computer to access an Ethernet network (LAN).



Figure 14.5: NIC

- **Every NIC have a MAC address:**
 - * MAC address stands for **Media Access Control** address
 - * It is a **unique 48 bit** number identifier assigned to network interfaces card (NIC)



Figure 14.6: Mac address

- * Eg of MAC address: **FF.01.AA.EE.4F.ED**
- * **ifconfig** command displays the MAC address of NIC card.

```
[root@rhel8 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        ether 08:00:27:23:fe:8a txqueuelen 1000 (Ethernet)
          RX packets 27116 bytes 38127629 (36.3 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 4603 bytes 332789 (324.9 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 14.7: MAC address

- **Gateway:**

- Used to connect 2 different networks with each other.
- By default, the router acts as default gateway for all the PC in the network.

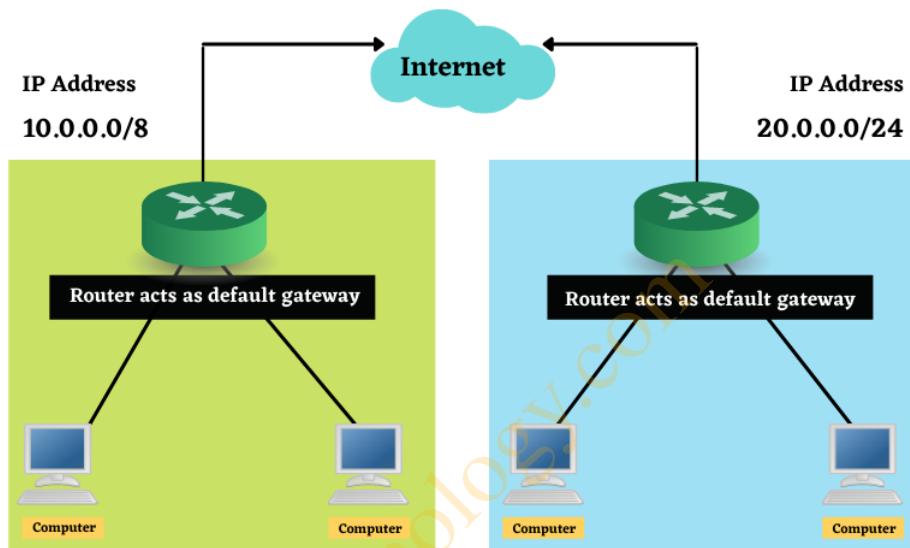


Figure 14.8: Gateway

14.1.3 Network interface names

- In older OS, network interfaces were named as eth0, eth1, eth2, and so on.
- New names of ethernet cards are assigned based on firmware, device topology, and device type.
- Interface names have the following characters:
 - The beginning character can be:
 - * **en**: For ethernet interfaces.
 - * **wl**: For WLAN interfaces.
 - * **ww**: For WWAN interfaces.
 - The next character can be:
 - * **o**: For on-board.
 - * **s**: For hotplug slot.
 - * **p**: For PCI geographic location
 - Finally, a number N is used to represent an index, ID, or port.
- Eg: A PCI card network interface may be named enp2s0.

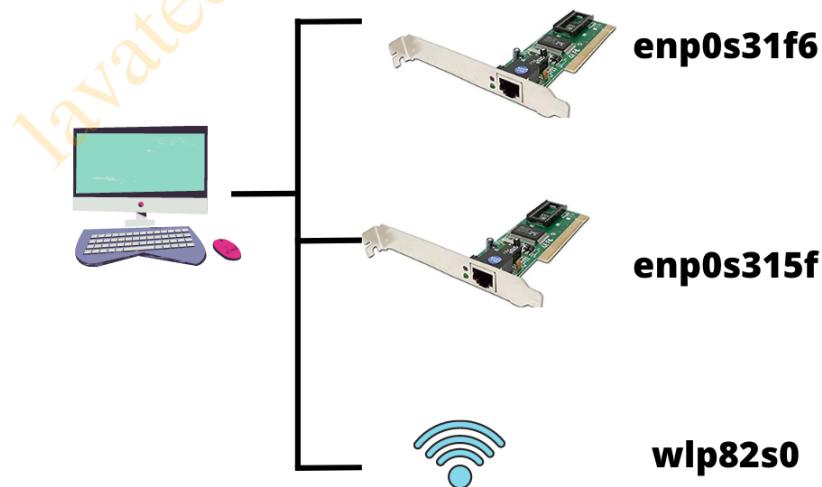
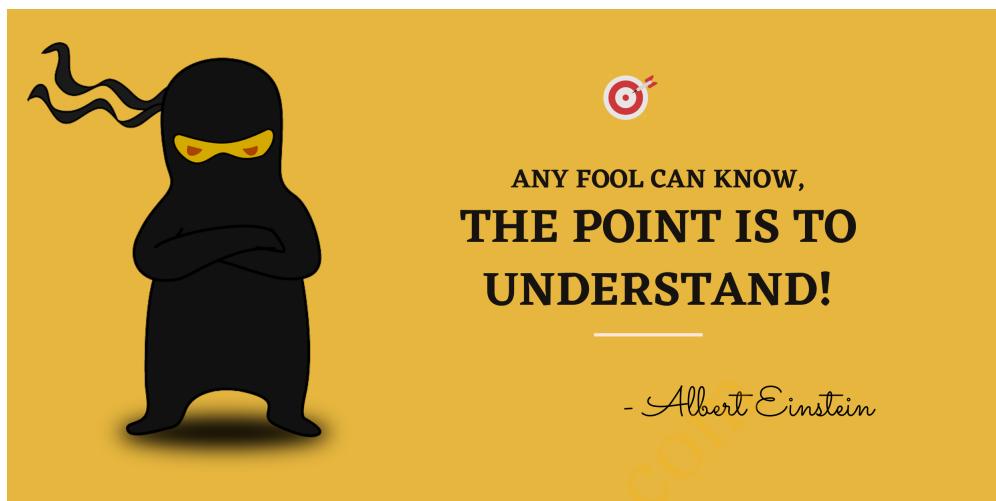


Figure 14.9: Ethernet card names

14.1.4 Practice

1. Which of the following device is used to send & receive data over telephone line?
 - (a) router
 - (b) modem
 - (c) switch
 - (d) hub
2. Which of the following device is used for inter-network communication?
 - (a) router
 - (b) modem
 - (c) switch
 - (d) hub
3. Which of the following device is used to provide additional ports to router?
 - (a) gateway
 - (b) modem
 - (c) switch

(d) hub

4. What is a MAC address?

- (a) MAC (Media Acces Control) address is 48 bit number assigned to ethernet card.
- (b) MAC (Media Acces Control) address is 64 bit number assigned to router.
- (c) MAC (Media Acces Control) address is 32 bit number assigned to switch.
- (d) MAC (Media Acces Control) address is 48 bit number assigned to router.

14.2 Introduction to IP

In this section, you are going to learn:

1. **What is Internet Protocol (IP)?**
2. **What is IP address?**
3. **Types of IP address**
4. **ISO-OSI model**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

14.2.1 What is Internet Protocol (IP)?

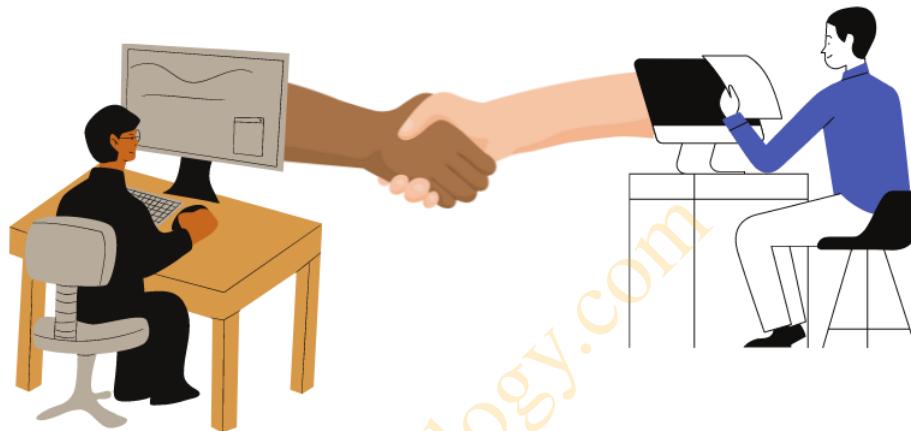
Internet Protocol (IP) is the method by which data is sent from one computer to another on the internet.

There are 2 methods of sending data:

- **TCP:**
 - Stands for "**Transmission Control Protocol**"
- **UDP**
 - Stands for "**User Datagram Protocol**"

Let's see each of these protocol.

TCP connection

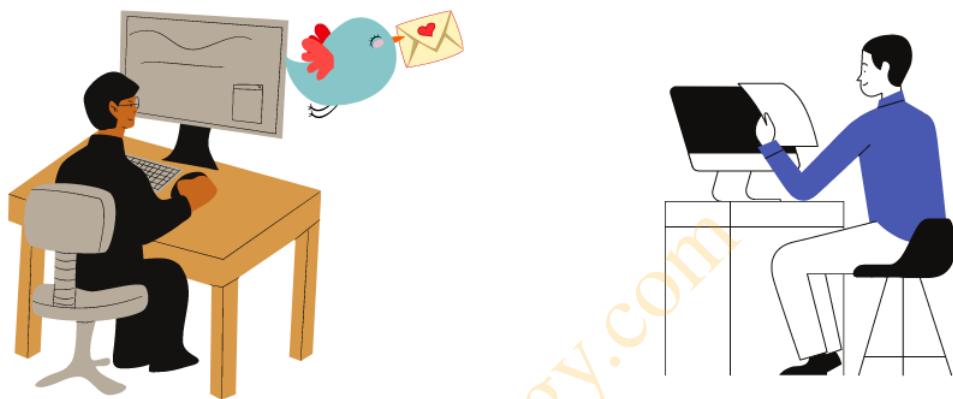


Secure data transfer

Figure 14.10: Secure TCP connection

- TCP provides **reliable transmission, error control and in order** receiving of the data.
- Provides point to point connection.
- Eg : **Whatsapp, Instagram, Google Chat, Emails, browsing**
- Used when:
 - Cannot tolerate the loss of data
 - **Order** of data is importance

UDP connection



Unsecure data transfer

Figure 14.11: Unsecure UDP connection

- UDP provides fast but **non-guaranteed transfer** of the data.
- No point to point connection.
- There is no connection establishment or termination on UDP.
- Eg: **Online gaming, Live streaming**

14.2.2 What is an IP address?

- An IP address is a **unique address that identifies a device** on the internet or a local network.
- Command to check IP address:

Syntax: ifconfig

or

Syntax: ip addr

- Eg:

```
[root@rhel8 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 brd 10.0.2.255 netmask 255.255.255.0
        ether 08:00:27:23:e8:8a txqueuelen 1000 (Ethernet)
        RX packets 27116 bytes 38127629 (36.3 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 4603 bytes 332789 (324.9 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 brd 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 14.12: Sample output

14.2.3 What is DNS server?

- The IP protocol uses IP addresses to communicate on internet, but human beings cannot remember IP addresses.
- DNS, the **Domain Name System**, is a servers that maps hostnames to IP addresses.
- Eg: If you type **https://www.gmail.com** in your browser, it is DNS server who will find the actual IP address of **www.gmail.com** host.

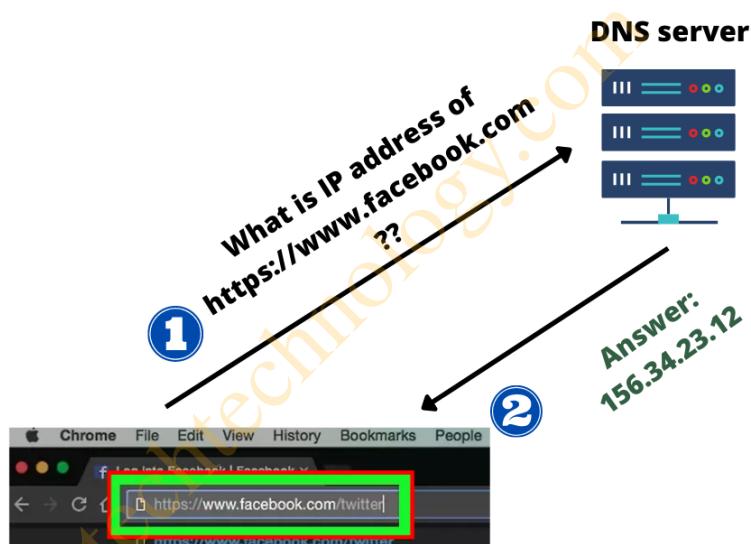


Figure 14.13: DNS name resolution

- Configuration of DNS server will be covered in **Learn Linux with Lavatech Technology Part-2** book.

14.2.4 What is DHCP server?

- DHCP stands for **Dynamic Host Control Protocol**.
- Using DHCP server, systems can obtain IP address automatically at boot time.
- If a DHCP server is not available, the system uses a static configuration from a local configuration file.
- Only one address can be assigned per NIC card with DHCP.
- In your home, router acts like a DHCP server and assign IP address to all connected device.

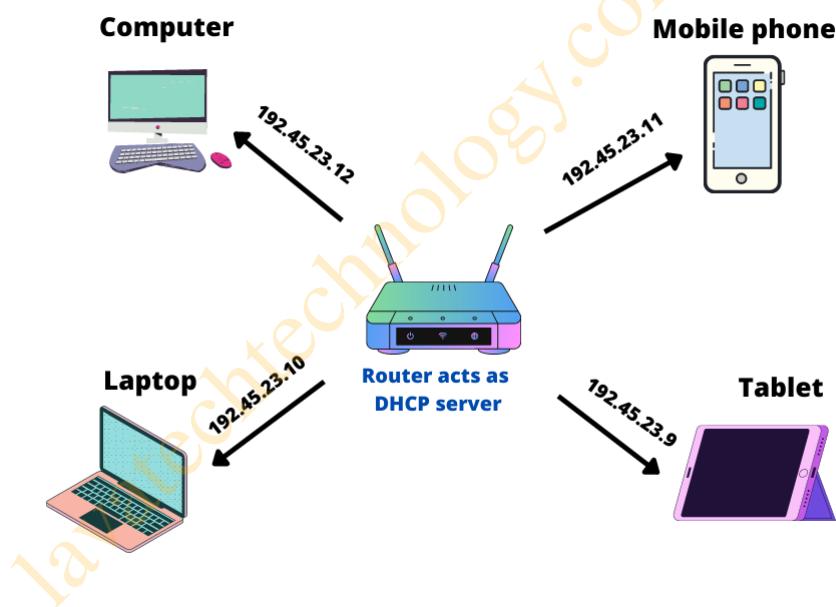


Figure 14.14: DHCP server

14.2.5 Types of IP address

Type of IP address depends on:

- **Location:** There are 2 types of IP address:

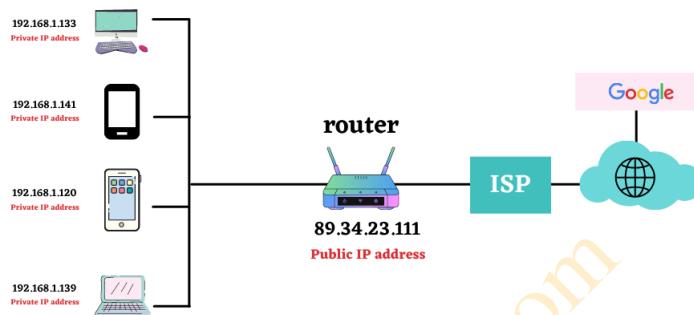


Figure 14.15: Public & Private IP

- **Public IP address:** Used **outside of a network**
- **Private IP address:** Used **inside a network**
- **Permanancy:** There are 2 types of IP address:
 - **Static IP address:** **Won't change and created manually.**
Provided by internet service provider (ISP) for external usage.
 - **Dynamic IP address:** **Can change anytime** and assigned by a **DHCP** server or router.
- **IPV4 & IPV6**

IPV4

- IPv4 is used to identify devices on a network using an addressing system.
- IPv4 is a 32-bit address.
- IPV4 can have approximate 4 million address.
- IPV4 carries 94% of Internet traffic.

IPv6

- This new IP address version solves the need for more Internet addresses.
- IPv6 is a 128-bit hexadecimal address.
- IPv6 allows 340 undecillion unique address space.

lavatechtechnology.com

14.2.6 ISO-OSI model

- ISO stands for **International Organization of Standardization**.
- ISO is model for **Open System Interconnection (OSI)**.
- The ISO-OSI model is a **seven** layer architecture.
- It defines seven layers or levels in a **complete communication system**.

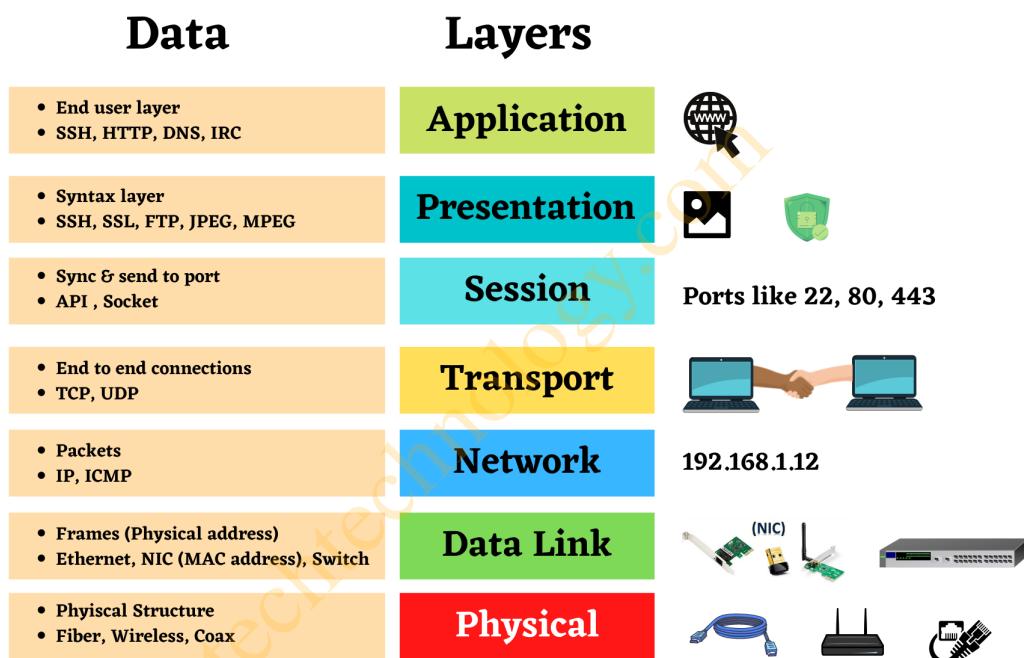
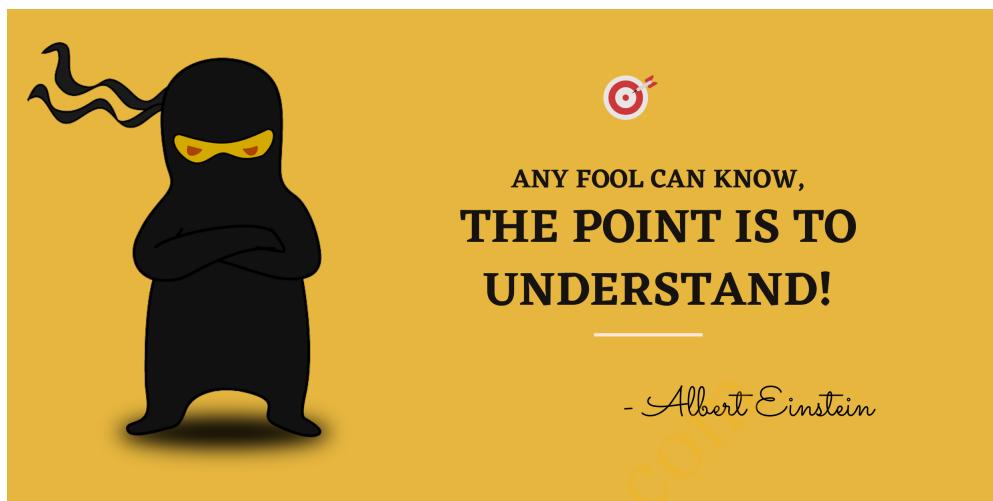


Figure 14.16: ISO OSI model

Let's see IP address in detail.

14.2.7 Practice

- 1. What is the full form of TCP?**
 - (a) Transmission Control Protocol
 - (b) Transmission Command Protocol
 - (c) Transfer Control Protocol
 - (d) Transfer Command Protocol

- 2. Which of the following statement is true about TCP protocol? (Select all that applies.)**
 - (a) TCP protocol allows unordered data transmission
 - (b) TCP protocol provides reliable data transmission
 - (c) TCP protocol provides ordered data transmission
 - (d) TCP protocol provides point to point connection

- 3. Which of the following software uses UDP protocol for data transfer? (Select all that applies.)**
 - (a) Whatsapp
 - (b) Google Chat
 - (c) Online gaming
 - (d) Live streaming

4. Which of the following IP address can change anytime and assigned by DHCP server?
- (a) Static IP address
 - (b) Dynamic IP address
 - (c) Public IP address
 - (d) Private IP address

lavatechtechnology.com

14.3 IPv4 in detail

In this section, you are going to learn:

1. **IPV4 address structure**
2. **Network bit & Host bit**
3. **Netmask**
4. **What is classful addressing?**
5. **What is loopback and 0.0.0.0 address?**
6. **What is network address & broadcast address?**
7. **How to calculate number of host IP?**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

14.3.1 IPv4 address structure

- IP address consist of 4 numbers (between 0-255), separate by ".":

Structure: 0-255 . 0-255 . 0-255 . 0-255

- But why the range of numbers is 0-255?

Let's see more on this.

Language of Computer

Computer's language is binary. **Binary language is of bits:**

- A bit is 0 1.
- 0 means off bit.
- 1 means on bit.

In computer' language, IP address is made of 32 bits as shown:

8bits . 8bits . 8bits . 8bits

$$8 + 8 + 8 + 8 = 32$$

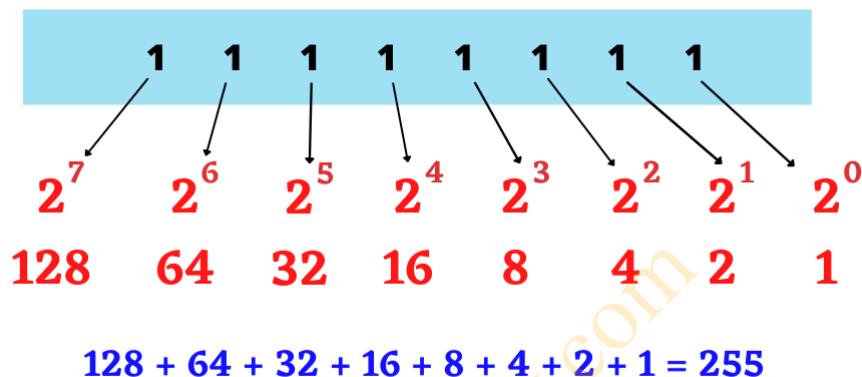
Example

11000000 . 01011010 . 11001100 . 00110011

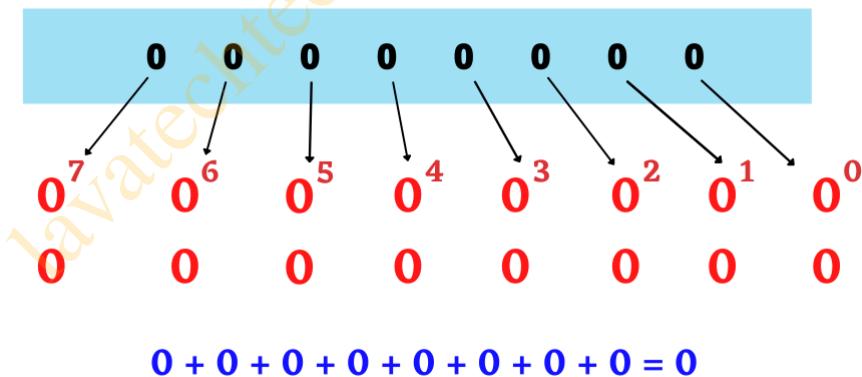
But how to understand numeric value behind bits? Let see more on this.

Calculating numeric form of IP:

- Numbers in IP are calculated by bit's position.
- Formula to calculate IP address number if any bit is **ON** :



- Formula to calculate IP address number if any bit is **OFF** :



Examples of calculating number of IP:

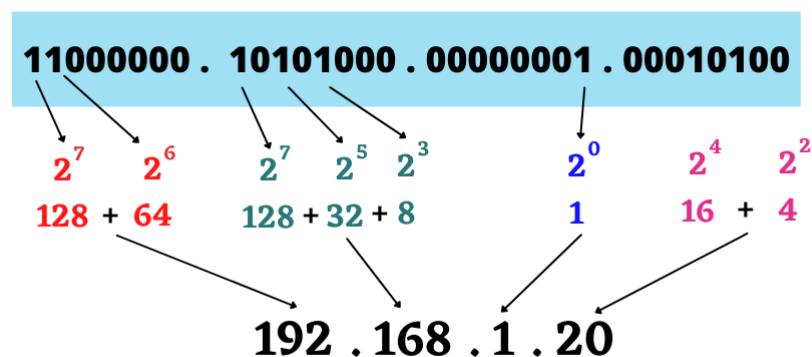


Figure 14.17: Binary to numeric conversion of IP

14.3.2 Network bit & Host bit

- IP address consists of 2 parts:
 - **Network bit:** Defines the network of your IP
 - **Host bit :** Defines the IP address of your machine

32-bit IP address

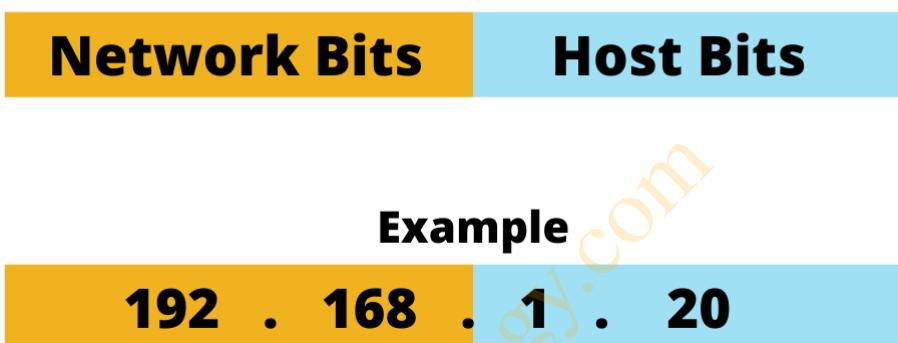


Figure 14.18: Network bit & Host bit

- How do we decide what part of IP address is network bit & host bit?
- Answer: Netmask

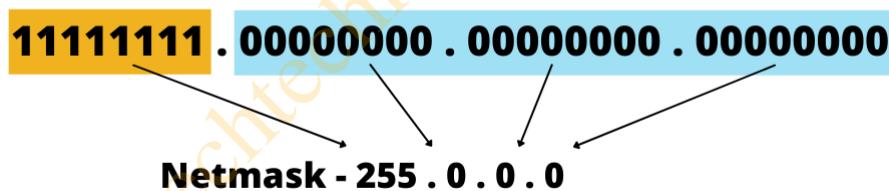
14.3.3 Netmask

- Netmask also called **subnet mask**:
 - Divides one big network into small small network
 - **Decides network bit & host bit**
 - The **on bits** are **network bit** in netmask
 - The **on bits** decides the **netmask prefix**
 - The **off bits** are **host bit** in netmask

Note: Netmask bits are always made on from left to right.

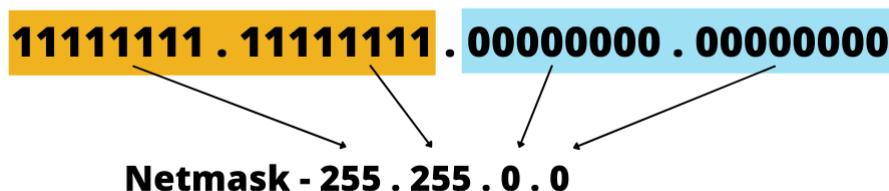
Netmask Prefix - 8

Network bit = 8 Host bit = $32-8=24$



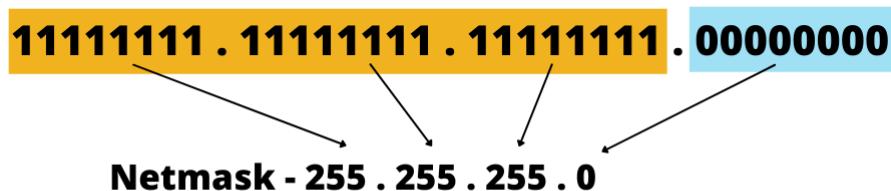
Netmask Prefix - 16

Network bit = 16 Host bit = $32-16=16$



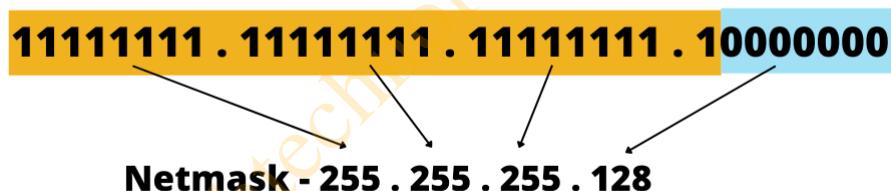
Netmask Prefix - 24

Network bit = **24** Host bit = **32-24=8**



Netmask Prefix - 25

Network bit = **25** Host bit = **32-25=7**



14.3.4 What is classful addressing?

Classful addressing:

- Was introduced in 1981
- Divides the IP address into five separate classes:
 - Class A
 - Class B
 - Class C
 - Class D
 - Class E
- Below image shows the details of each class in detail:

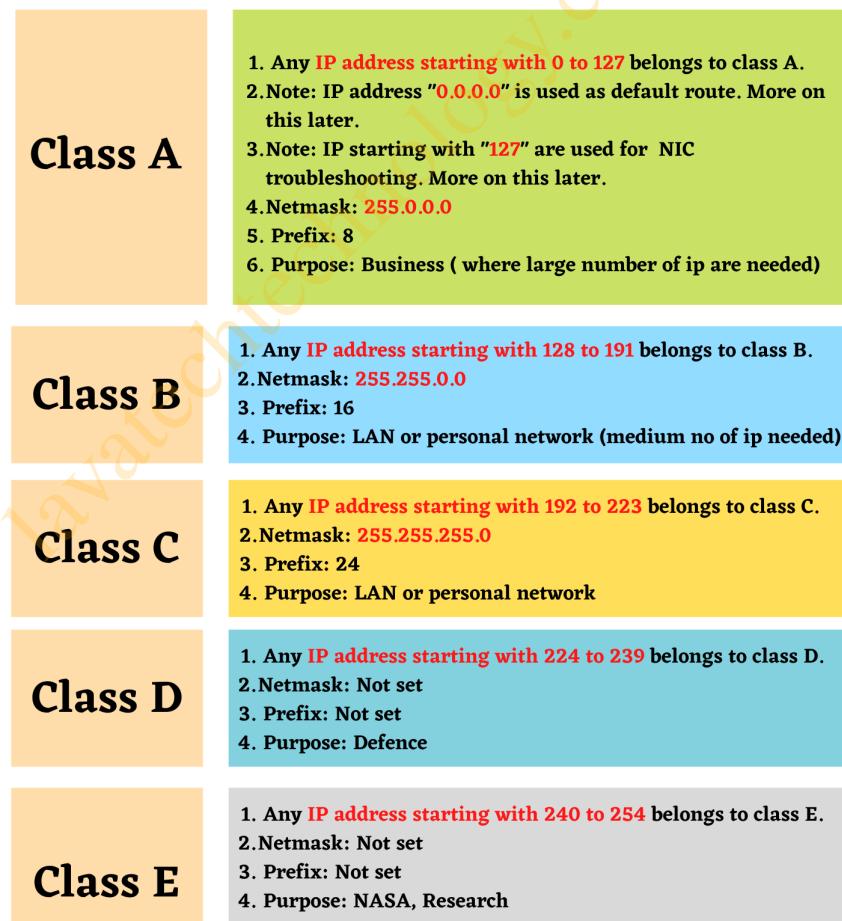


Figure 14.19: IP address classes

Example of IP address and their class:

| IP address | Class | Network part | Host part |
|--------------|---------|--------------|-----------|
| 1.2.3.4 | Class A | 1 | 2.3.4 |
| 134.67.23.12 | Class B | 134.67 | 23.12 |
| 171.78.91.0 | Class B | 171.78 | 91.0 |
| 221.67.23.16 | Class C | 221.67.23 | 16 |
| 192.56.23.1 | Class C | 192.56.23 | 1 |

Figure 14.20: IP address classification according to class

14.3.5 Loopback & 0.0.0.0 address

What is loopback address?

- **127.0.0.1** is loopback IP address also called **localhost**.
- The **Class A**, 127.0.0.0 network address is reserved for loopback testing.
- Used to test whether NIC card is working properly or not.

```
[root@rhel8 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        ether 08:00:27:23:e8:8a txqueuelen 1000 (Ethernet)
          RX packets 8565 bytes 12331382 (11.7 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 1129 bytes 87260 (85.2 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
          RX packets 0 bytes 0 (0.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 0 bytes 0 (0.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 14.21: Localhost IP

- You can ping localhost IP to check if your NIC card is working properly or not.
- Eg:

```
root@lavatech:~# ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.021 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.024 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.016 ms
64 bytes from localhost (127.0.0.1): icmp_seq=4 ttl=64 time=0.018 ms
^C
--- localhost ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3069ms
rtt min/avg/max/mdev = 0.016/0.019/0.024/0.003 ms
```

Figure 14.22: Localhost IP

What is 0.0.0.0 address?

- Can mean "**all IPv4 addresses on the system**".
- Eg: If a system has two IP addresses, 192.168.1.1 and 10.1.2.1, and a server running on the system is configured to listen on 0.0.0.0, it will be reachable at both of those IP addresses.

14.3.6 Network address

- Network address represents entire network.
- It represents group of IP addresses that can be used on that network.
- **It is not a usable IP.**
- **It is only used by routers** for packet transmission.

How to write network address for a given IP?

- Solution: Network address is decided according to the netmask as follows:
 - Keep the network part of the IP address fix
 - Replace the host part of the IP address with 0
 - Eg 1:

IP address: **195.34.23.128**

Netmask: **255.255.255.0**

Network Address: **195.34.23.0**

- Eg 2:

IP address: **10.4.2.3**

Netmask: **255.0.0.0**

Network Address: **10.0.0.0**

- Eg 3:

IP address: **129.89.32.12**

Netmask: **255.255.0.0**

Network Address: **129.89.0.0**

14.3.7 Broadcast address

Broadcast address is used to transmit data to all of the hosts on the local subnet.

It is used by DHCP server to dynamically assign an IP address to computers on a network.

How to write broadcast address for a given IP?

- Solution: Broadcast address is decided according to the netmask as follows:
 - Keep the network part of the IP address fix
 - Replace the host part of the IP address with 255
 - Eg 1:

IP address: **195.34.23.128**

Netmask: **255.255.255.0**

Broadcast Address: **195.34.23.255**

- Eg 2:

IP address: **10.4.2.3**

Netmask: **255.0.0.0**

Broadcast Address: **10.255.255.255**

- Eg 3:

IP address: **228.34.23.1**

Netmask: **255.255.255.0**

Broadcast Address: **228.34.23.255**

14.3.8 Calculating number of host IP

How many IP address can be generated in a network?

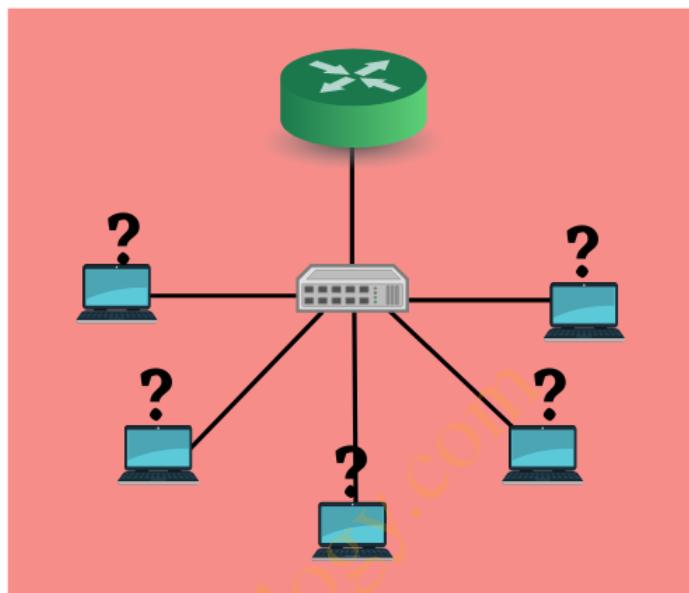


Figure 14.23: Host IP

- To calculate number of host IP address in any network:

Formula: $2^n - 2$

In this formula:

- **n = host id bits**
- The **-2** is to remove network address and broadcast address

Examples 1

If you have an IP address **18.6.7.8**, how will you determine total number of host IP address in it's network?

Solution: From IP address **18.6.7.8**, you can conclude -

Class: **A**

Netmask: **255.0.0.0**

Host ID bits: **24**

Total number of host IP address: $2^{24} - 2 = 1,67,77,214$

| Network address: 18.0.0.0 | | | |
|--|-------------------|-------------------|-------------------------|
| Broadcast address: 18.255.255.255 | | | |
| 18.0.0.1 | 18.0.1.1 | 18.0.2.1 | 18.255.255.0 |
| 18.0.0.2 | 18.0.1.2 | 18.0.2.2 | 18.255.255.1 |
| 18.0.0.3 | 18.0.1.3 | 18.0.2.3 | 18.255.255.2 |
| 18.0.0.4 | 18.0.1.4 | 18.0.2.4 | 18.255.255.3 |
| . | . | . | . |
| . | . | . | . |
| 18.0.0.255 | 18.0.1.255 | 18.0.2.255 | 18.0.255.255.254 |

Figure 14.24: Total IP address for 18.0.0.0 network

Examples 2

If you have an IP address **160.7.27.10**, how will you determine total number of host IP address in it's network?

Solution: From IP address **160.7.27.10**, you can conclude:

Class: **B**

Netmask: **255.255.0.0**

Host ID bits: **16**

Total number of host IP address: $2^{16} - 2 = 65534$

Network address: 160.7.0.0
Broadcast address: 160.7.255.255

| | | | |
|-------------|-------------|-------------|---------------|
| 160.7.0.1 | 160.7.1.0 | 160.7.2.0 | 160.7.255.0 |
| 160.7.0.2 | 160.7.1.1 | 160.7.2.1 | 160.7.255.1 |
| 160.7.0.3 | 160.7.1.2 | 160.7.2.2 | 160.7.255.2 |
| 160.7.0.4 | 160.7.1.3 | 160.7.2.3 | 160.7.255.3 |
| . | . | . | . |
| . | . | . | . |
| 160.7.0.255 | 160.7.1.255 | 160.7.2.255 | 160.7.255.254 |

Figure 14.25: Total IP address for 160.7.0.0 network

Examples 3

If you have an IP address **223.17.28.0**, how will you determine total number of host IP address in it's network?

Solution: From IP address **223.17.28.0**, you can conclude:

Class: **C**

Netmask: **255.255.255.0**

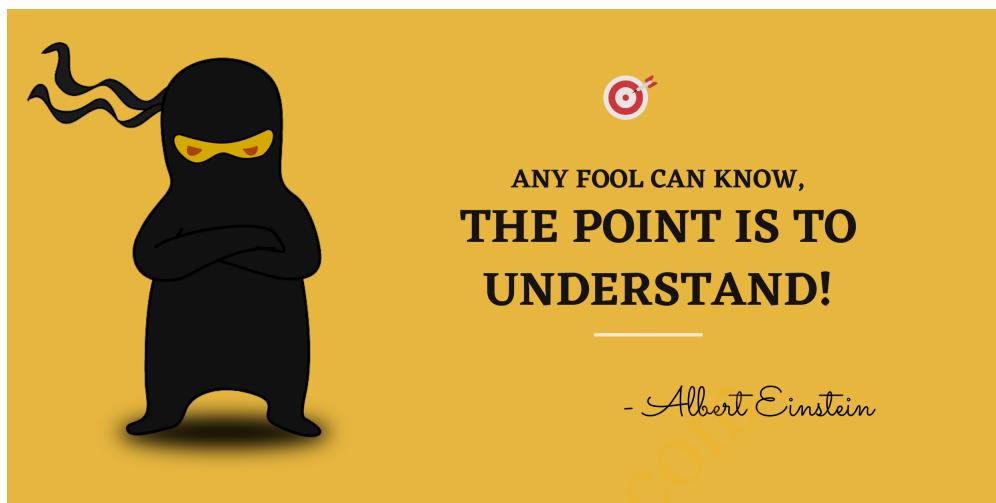
Host ID bits: **8**

Total number of host IP address: $2^8 - 2 = 254$

Network address: 223.17.28.0
Broadcast address: 223.17.28.255

223.17.28.1
223.17.28.2
223.17.28.3
223.17.28.4
.
. .
223.17.28.254

Figure 14.26: Total IP address for 223.17.28.0 network

14.3.9 Practice

- 1. IP address is made up of how many bits?**
 - (a) 64 bit
 - (b) 48 bit
 - (c) 32 bit
 - (d) 24 bit

- 2. Which of the following are valid IP address? (Select all that applies.)**
 - (a) 555.34.21.10
 - (b) 243.678.12.100
 - (c) 89.45.12.23
 - (d) 99.45.244.123

- 3. State whether true or false. Network bit defines the network of your IP address.**
 - (a) True
 - (b) False

4. What is netmask? (Select all that applies.)

- (a) Netmask defines the network bit & host bit of an IP address
- (b) Netmask divides one big network into small networks
- (c) Netmask defines how to assign IP address to a system
- (d) Netmask defines the network address of your system

5. What is netmask prefix for netmask -

"11111111.11111111.11111111.00000000"?

- (a) 26
- (b) 25
- (c) 24
- (d) 18

6. What is netmask prefix for netmask -

"11111111.11111111.00000000.00000000"?

- (a) 16
- (b) 25
- (c) 24
- (d) 18

7. What is the class of IP address - 192.168.1.12?

- (a) Class A
- (b) Class B
- (c) Class C
- (d) Class D

8. What is the class of IP address - 129.12.23.12?

- (a) Class A
- (b) Class B
- (c) Class C
- (d) Class D

9. What is the class of IP address - 10.0.2.15?

- (a) Class A
- (b) Class B
- (c) Class C
- (d) Class D

10. Which of the following is loopback address?

- (a) 127.90.1.0
- (b) 127.0.0.1
- (c) 128.0.0.1
- (d) 129.0.0.1

11. What is the network address of IP - 172.67.45.10?

- (a) 172.67.45.0
- (b) 172.67.0.0
- (c) 172.0.0.0
- (d) 172.67.45.1

12. What is the broadcast address of IP - 172.67.45.10?

- (a) 172.67.45.255
- (b) 172.67.255.255
- (c) 172.255.255.255
- (d) 172.255.255.0

13. What is the broadcast address of IP - 10.78.23.10?

- (a) 10.255.255.255
- (b) 10.78.255.255
- (c) 10.78.23.255
- (d) 10.78.23.0

14. What is the network address of IP - 10.78.23.10?

- (a) 10.78.0.0
- (b) 10.78.23.0
- (c) 10.78.23.1
- (d) 10.0.0.0

15. Which of the following is a formula to calculate total number of host IP address in any network?

- (a) $2^n - 1$
- (b) $2^n - 4$
- (c) $2^n - 8$
- (d) $2^n - 2$

14.4 Subnetting

In this section, you are going to learn:

1. **Drawback of classful addressing**
2. **What is CIDR?**
3. **Working of CIDR**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

14.4.1 Drawback of classful addressing

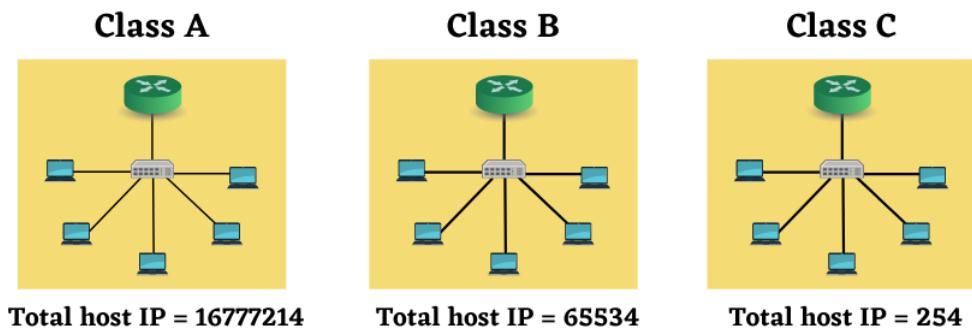


Figure 14.27: Total host IP per class

- Using classful addressing results in wasting many IP address.
- Half of the IP address allocated to network are not needed by anybody.
- Hence classful addressing don't exist anymore.

Solution: Classless Inter-domain Routing (CIDR). Let see more on this.

14.4.2 Subnetting/CIDR

- Introduced in 1993.
- CIDR (Classless Inter-Domain Routing) is also known as **supernetting is logical division of network.**
- It is a method of distributing IP addresses effectively.
- It replaces the Class A, Class B & Class C networks.
- Netmask prefix for CIDR are anything between 1-32, except 8, 16 & 24 which are used for classful addressing.

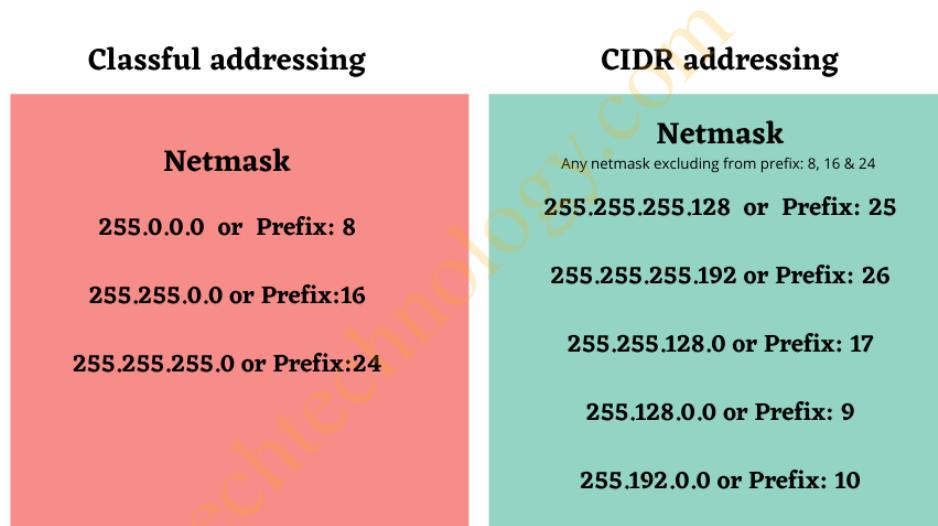


Figure 14.28: Netmask as per classful & CIDR addressing

14.4.3 Working of CIDR

Use case 1

- Imagine there is a requirement to create 2 separate networks for **Finance & HR department**.
- **Finance department** needs 50 IP address.
- **HR department** needs 30 IP address.
- Team manager have purchased a **network address - "16.0.0.0"**.
- **How would you use network address 16.0.0.0 and create 2 different network for finance & HR department?**

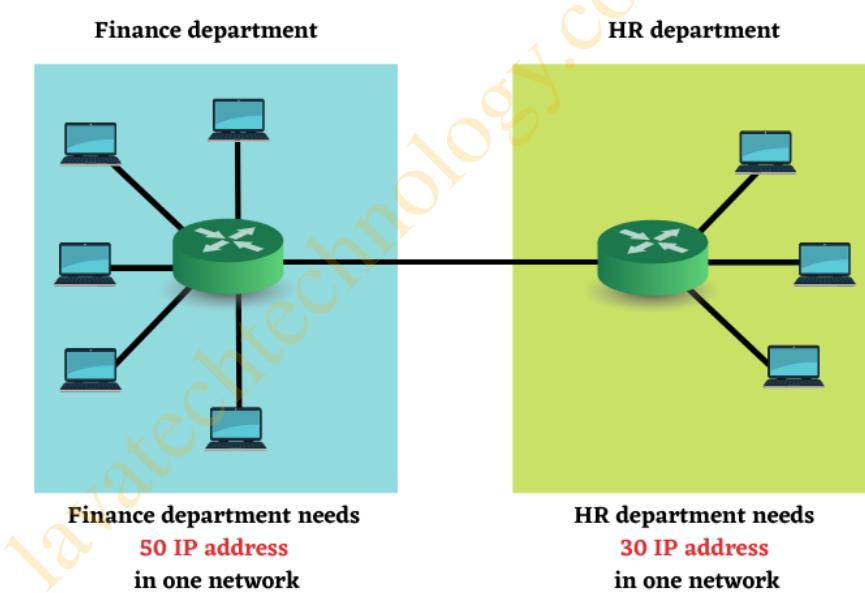


Figure 14.29: Custom IP address requirement

- Let's see how to do this.

Solution

Using Class A, B or C IP address range would result in lots of IP address wastage.

The solution is to logically divide the **16.0.0.0** network address into multiple different networks using CIDR.

Steps for CIDR:

- **Select a number "n":**

- Decide maximum number of IP address you need in a single network. Here, **maximum number of IP needed is 50**.
- Select number "n" that can fit into below formula and is greater than or equal to 50.

$$\text{Formula: } 2^n - 2 \geq \text{number_of_host_ip_needed}$$

- Here, $2^n - 2 \geq 50$
- Solution: $2^6 - 2 \geq 50$, where **n=6**

- **Calculate new netmask:**

- To calculate new netmask:

Formula:

OFF **n** bits from right to left

ON rest of the bits

- Here n=6, so answer is:

- * Binary form: **1111111.1111111.1111111.11000000**
- * Netmask: **255.255.255.192**
- * Netmask Prefix: **26**

- Total host IP in a network:

- Using value of "**n**", total number of host IP are:

$$\text{Formula: } 2^n - 2$$

- Here, answer is: $2^6 - 2 = 62$

lavatechtechnology.com

- All networks possible for **16.0.0.0/26**: .

All 4 of the Possible /26 Networks for 16.0.0.*

| Network Address | Usable Host Range | Broadcast Address: |
|-----------------|-------------------------|--------------------|
| 16.0.0.0 | 16.0.0.1 - 16.0.0.62 | 16.0.0.63 |
| 16.0.0.64 | 16.0.0.65 - 16.0.0.126 | 16.0.0.127 |
| 16.0.0.128 | 16.0.0.129 - 16.0.0.190 | 16.0.0.191 |
| 16.0.0.192 | 16.0.0.193 - 16.0.0.254 | 16.0.0.255 |

Figure 14.30: 16.0.0.0/26 possible network address

Use case 2

- Imagine there is a requirement to create 2 separate networks for **IT & HR department**.
- **IT department** needs 24 IP address.
- **HR department** needs 10 IP address.
- Team manager have purchased an **IP address - "34.67.4.0"**.
- **How would you use IP address "34.67.4.0" and create 2 different network for IT & HR department?**

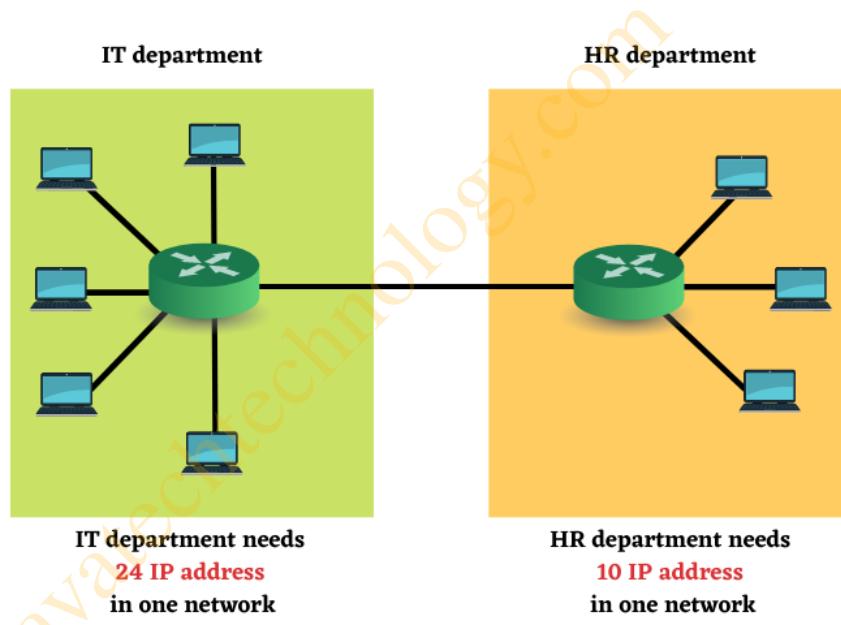


Figure 14.31: Custom IP address requirement

- Let's see how to do this.

Solution

Steps for CIDR:

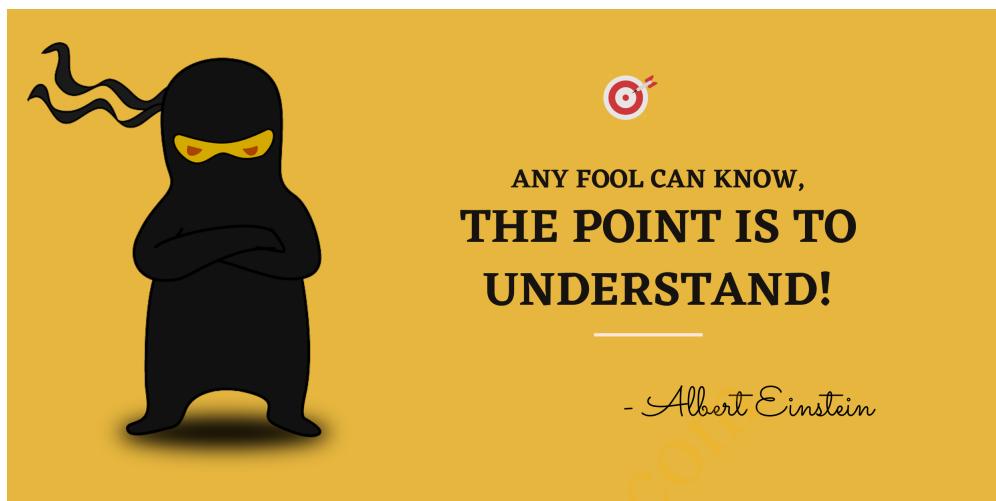
- **Select a number "n":**
 - Here, **maximum number of IP needed is 24.**
 - Select number "n" that can fit into below formula and is greater than or equal to 24.
 - Here, $2^n - 2 \geq 24$
 - Solution: $2^5 - 2 \geq 24$, where **n=5**
- **Calculate new netmask:**
 - Here n=5, so answer is:
 - * Binary form: **1111111.1111111.1111111.11100000**
 - * Netmask: **255.255.255.224**
 - * Netmask Prefix: **27**
- Total host IP in a network:
 - Using value of "n", total number of host IP are:

Formula: $2^n - 2$
 - Here, answer is: $2^5 - 2 = 30$
- All networks possible for **34.67.4.0/27**: .

All 8 of the Possible /27 Networks for 34.67.4.*

| Network Address | Usable Host Range | Broadcast Address: |
|-----------------|---------------------------|--------------------|
| 34.67.4.0 | 34.67.4.1 - 34.67.4.30 | 34.67.4.31 |
| 34.67.4.32 | 34.67.4.33 - 34.67.4.62 | 34.67.4.63 |
| 34.67.4.64 | 34.67.4.65 - 34.67.4.94 | 34.67.4.95 |
| 34.67.4.96 | 34.67.4.97 - 34.67.4.126 | 34.67.4.127 |
| 34.67.4.128 | 34.67.4.129 - 34.67.4.158 | 34.67.4.159 |
| 34.67.4.160 | 34.67.4.161 - 34.67.4.190 | 34.67.4.191 |
| 34.67.4.192 | 34.67.4.193 - 34.67.4.222 | 34.67.4.223 |
| 34.67.4.224 | 34.67.4.225 - 34.67.4.254 | 34.67.4.255 |

Figure 14.32: 34.67.4.0/27 possible network address

14.4.4 Practice

1. **What is the full form of CIDR?**
 - (a) Classless Inter-domain Redirection
 - (b) Classless Inter-department Routing
 - (c) Classfull Inter-domain Routing
 - (d) Classless Inter-domain Routing

2. **Which of the following statement is true about CIDR? (Select all that applies.)**
 - (a) CIDR is a method of effectively distributing IP address.
 - (b) CIDR is also known as supernetting.
 - (c) CIDR replaces classful addressing.
 - (d) CIDR is outdated method of allotting IP address.

14.5 Networking in action

In this section, you are going to learn:

1. **Assigning hostname**
2. **Setting up IP address**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

14.5.1 Assigning hostname

- Hostname is the name of your computer.
- **Fully qualified domain name (FQDN)** is hostname plus the domain.
- Example of hostname, domain name and FQDN is:
 - **Hostname:** server
 - **Domain:** example.com
 - **FQDN:** server.example.com
- Command to assign hostname:

```
Syntax: hostnamectl set-hostname FQDN
```

- Eg:

```
# hostnamectl set-hostname server.example.com
```

- Command to check hostname status:

```
Syntax: hostnamectl status
```

- Eg:

```
[root@rhel8 ~]# hostnamectl status
Static hostname: server.example.com
Icon name: computer-vm
Chassis: vm
Machine ID: 67803279ab354eedb87f639abb59bdcf
Boot ID: 197c7f6701f84ee7bbb7815a5c787bdf
Virtualization: oracle
Operating System: Red Hat Enterprise Linux 8.5 (Ootpa)
CPE OS Name: cpe:/o:redhat:enterprise_linux:8::baseos
Kernel: Linux 4.18.0-348.el8.x86_64
Architecture: x86-64
```

Figure 14.33: Sample output

- In RHEL 8, the hostname is stored in **/etc/hostname** file. Changes in this file will be reflected only after server reboot.

14.5.2 /etc/hosts file

- The **/etc/hosts** file is used to **temporarily** resolve host names to IP addresses or the reverse.
- **/etc/hosts do not** perform permanent name resolution like DNS server.

```
[root@rhel8 ~]# cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6
```

Figure 14.34: Content of /etc/hosts file

- You can resolve your hostname to IP address by adding an entry to this file as shown:

```
[root@rhel8 ~]# cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.0.104  server.example.com
```

Figure 14.35: Add entry to /etc/hosts file

- Test host name resolution with the **/etc/hosts** file using below command:

Syntax: `getent hosts hostname`

If an entry is not found in that file, the **getent** looks for the information from a DNS nameserver.

Eg:

```
# getent hosts server.example.com
fe80::a00:27ff:fe95:49b4  server.example.com
```

- Test host name resolution that are resolved only using DNS and not **/etc/hosts**:

Syntax: host hostname

Eg:

```
[root@rhel8 ~]# host www.facebook.com
www.facebook.com is an alias for star-mini.c10r.facebook.com.
star-mini.c10r.facebook.com has address 31.13.79.35
star-mini.c10r.facebook.com has IPV6 address 2a03:2880:f12f:183:face:b00c:0:25de
[root@rhel8 ~]#
[root@rhel8 ~]#
[root@rhel8 ~]# host server.example.com
Host server.example.com not found: 3(NXDOMAIN)
```

Figure 14.36: Sample output

14.5.3 Setting up IP address

nmcli command

- nmcli stands for **network manager command line**.
- It is a command-line tool for controlling NetworkManager.
- nmcli save configuration files in the **/etc/sysconfig/network-scripts** directory.
- Using "**nmcli**" command, you can assign/change IP address to network interface card.
- Options with **nmcli** command:
 - Display all available LAN card devices:

```
Syntax: nmcli device show  
or  
Syntax: nmcli dev show
```

Eg:

```
root@server ~]# nmcli device show  
GENERAL.DEVICE:           eth0  
GENERAL.TYPE:             ethernet  
GENERAL.HWADDR:          08:00:27:23:E8:8A  
GENERAL.MTU:              1500  
GENERAL.STATE:            100 (connected)  
GENERAL.CON-PATH:         System eth0  
WIRED-PROPERTIES.CARRIER: on  
IP4.ADDRESS[1]:           10.0.2.15/24  
IP4.GATEWAY:              10.0.2.2  
IP4.ROUTE[1]:             dst = 0.0.0.0/0, nh = 10.0.2.2, mt = 101  
IP4.ROUTE[2]:             dst = 10.0.2.0/24, nh = 0.0.0.0, mt = 101  
IP4.DNS[1]:                10.0.2.3  
IP6.GATEWAY:  
  
GENERAL.DEVICE:           eth1  
GENERAL.TYPE:             ethernet  
GENERAL.HWADDR:          08:00:27:68:D6:8F  
GENERAL.MTU:              1500  
GENERAL.STATE:            30 (disconnected)  
GENERAL.CON-PATH:         --  
WIRED-PROPERTIES.CARRIER: on  
  
GENERAL.DEVICE:           lo  
GENERAL.TYPE:             loopback  
GENERAL.HWADDR:          00:00:00:00:00:00  
GENERAL.MTU:              65536  
GENERAL.STATE:            10 (unmanaged)  
GENERAL.CON-PATH:         --  
IP4.ADDRESS[1]:           127.0.0.1/8  
IP4.GATEWAY:              --  
IP6.GATEWAY:              --
```

Figure 14.37: List of available devices

- Display all available connections

Syntax: nmcli connection show

or

Syntax: nmcli con show

Eg:

```
[root@server ~]# nmcli connection show
NAME           UUID                                  TYPE      DEVICE
System eth1    9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  ethernet  eth1
System eth0    5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
```

Figure 14.38: List of available interfaces

- Add new connection to an interface:

Syntax: nmcli connection add con-name <connection_name> ifname <interface_name> type ethernet

Eg:

```
# nmcli connection add con-name newconnection
ifname eth1 type ethernet
```

- Provide dynamic IP address (i.e IP from DHCP server) to existing connection:

Syntax: nmcli connection modify <connection_name> ipv4.method auto

Syntax: nmcli con mod <connection_name> ipv4.method auto

Eg: Eg:

```
# nmcli con mod newconnection ipv4.method auto
```

- Provide IP address to existing connection:

Syntax: nmcli connection modify <connection_name> ipv4.addresses <ip_address> ipv4.method manual

Syntax: nmcli connection up <connection_name>

The above command results the changes in below config file:

Configuration file:

/etc/sysconfig/network-scripts/ifcfg-<connection_name>

Eg:

```
# nmcli connection modify newconnection
ipv4.addresses 192.168.120.20/24 ipv4.method manual

# nmcli connection up newconnection
```

Notice the changes are reflected in configuration file:

/etc/sysconfig/network-scripts/ifcfg-newconnection.

```
[root@server ~]# cat /etc/sysconfig/network-scripts/ifcfg-newconnection
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=newconnection
UUID=3d4b4943-566f-479e-b685-8d6207bdd685
DEVICE=eth1
ONBOOT=yes
IPADDR=192.168.120.20
PREFIX=24
```

Figure 14.39: Sample output

- Command to set gateway & dns:

Syntax: nmcli connection modify <connection_name>
 ipv4.gateway <ip_address> ipv4.dns <ip_address>

Eg:

```
# nmcli connection modify newconnection  
    ipv4.gateway 192.168.122.1 ipv4.dns 192.168.122.254
```

- Delete connection:

Syntax: nmcli connection delete <connection_name>
or
Syntax: nmcli con del <connection_name>

Eg:

```
# nmcli con del newconnection
```

Editing interface configuration files

- **Dynamic configuration:**

- Create file with name -

/etc/sysconfig/network-scripts/ifcfg-<name> along with below settings:

```
# cat /etc/sysconfig/network-scripts/ifcfg-eth0
NAME="newcon"
TYPE=Ethernet
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

- A description of some of these configuration parameters follows:
 - * **TYPE=device_type:** The type of network interface device.
 - * **BOOTPROTO=protocol:** Where protocol is one of the following:
 - **none:** No boot-time protocol is used.
 - **bootp:** Use BOOTP (bootstrap protocol).
 - **dhcp:** Use DHCP (Dynamic Host Configuration Protocol).
 - * **ONBOOT=answer:** Where answer is one of the following:
 - **yes:** This interface is activated at boot time.
 - **no:** This interface is not activated at boot time.
 - * **DEVICE=interface-name:** Ethernet interface name.

```
# nmcli con down eth0
# nmcli con up eth1
```

- **Static configuration:**

- Create file with name -

/etc/sysconfig/network-scripts/ifcfg-<name> along with below settings:

```
# cat /etc/sysconfig/network-scripts/ifcfg-eth0
NAME="newcon2"
TYPE=Ethernet
IPADDR0=172.25.4.5
PREFIX0=24
GATEWAY0=172.25.4.254
DEVICE=eth1
ONBOOT=yes
BOOTPROTO=none
DNS0=172.25.254.254
```

- A description of some of these configuration parameters follows:
 - * **NAME=connection-name:** Connection name for the interface.
 - * **TYPE=device_type:** The type of network interface device.
 - * **IPADDRN=address:** The IPv4 address assigned to the interface
 - * **PREFIXN=N:** Length of the IPv4 netmask value
 - * **GATEWAYN=address:** The IPv4 gateway address assigned to the interface.
 - * **DNSN=address:** The address of the Domain Name Servers (DNS)

Note: Because an interface can be associated with several combinations of IP address, network mask prefix length, and gateway address, these are numbered starting from 0.

- * **BOOTPROTO=protocol:** Where protocol is one of the

following:

- **none**: No boot-time protocol is used.
 - **bootp**: Use BOOTP (bootstrap protocol).
 - **dhcp**: Use DHCP (Dynamic Host Configuration Protocol).
 - * **ONBOOT=answer**: Where answer is one of the following:
 - **yes**: This interface is activated at boot time.
 - **no**: This interface is not activated at boot time.
 - * **DEVICE=interface-name**: Ethernet interface name.
- Reflect the changes done in the interface configuration file by bringing down the interface and bringing it up again.

```
# nmcli con down eth0
# nmcli con up eth1
```

14.5.4 Network monitoring commands

- **netstat**: Produce information related to network connections, routing tables, interface statistics etc.

Options with **netstat** command:

- **-r**: Display routing table

Syntax: netstat -r

Eg:

```
[root@server ~]# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt Iface
default         _gateway       0.0.0.0        UG        0 0          0 eth0
10.0.2.0        0.0.0.0        255.255.255.0  U         0 0          0 eth0
192.168.120.0   0.0.0.0        255.255.255.0  U         0 0          0 eth1
```

Figure 14.40: Sample output

- **-t**: Display **tcp** sockets
- **-u**: Display **udp** sockets
- **-l**: Display listening server sockets
- **-p**: Display PID/Program name for sockets
- **-n**: Don't resolve names

Syntax: netstat -tulpn

Eg:

```
[root@server ~]# netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp     0      0 0.0.0.0:22              0.0.0.0:*          LISTEN    942/sshd
tcp6    0      0 ::1:22                  :::*                 LISTEN    942/sshd
udp     0      0 127.0.0.1:323           0.0.0.0:*          LISTEN    879/chronyd
udp6    0      0 ::1:323                :::*                 LISTEN    879/chronyd
udp6    0      0 fe80::acf6:ea37:694:546  :::*          2575/NetworkManager
```

Figure 14.41: Sample output

- **tcpdump:**

- Captures network package for future analysis.

Option with **tcpdump** command:

- **-w:** Capture the network package in a file with ".pcap" extension.
- **-i:** Interface name
- **-c:** Number of network packets to capture.

Syntax: `tcpdump -c number_of_packets -w filename.pcap -i interface_name`

Eg:

```
# tcpdump -c 3 -w 0001.pcap -i eth1
```

```
[root@server ~]# [root@server ~]# tcpdump -c 3 -w 0001.pcap -i eth1
dropped privs to tcpcap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
3 packets captured
3 packets received by filter
0 packets dropped by kernel
[root@server ~]#
[root@server ~]# ls -ld 0001.pcap
-rw-r--r--. 1 tcpcap tcpcap 576 Mar 27 06:59 0001.pcap
[root@server ~]#
```

Figure 14.42: Sample output

- **-r:** Read .pcap file

Syntax: `tcpdump -r filename.pcap`

Eg:

```
[root@server ~]# tcpdump -r 0001.pcap
reading from file 0001.pcap, link-type EN10MB (Ethernet)
dropped privs to tcpcap
06:59:56.684259 IP6 fe80::52d4:f7ff:fedc:1916 > ip6-allnodes: ICMP6, router advertisement, length 24
06:59:57.992277 IP lavatech.32843 > 239.255.255.250.ssdp: UDP, length 171
06:59:58.993263 IP lavatech.32843 > 239.255.255.250.ssdp: UDP, length 171
[root@server ~]#
```

Figure 14.43: Sample output

- **ip route:** Show routing information on the system.

Syntax: ip route

- **traceroute or tracepath:** To trace the path to a remote host, use either traceroute or tracepath.

Syntax: tracepath hostname

Eg:

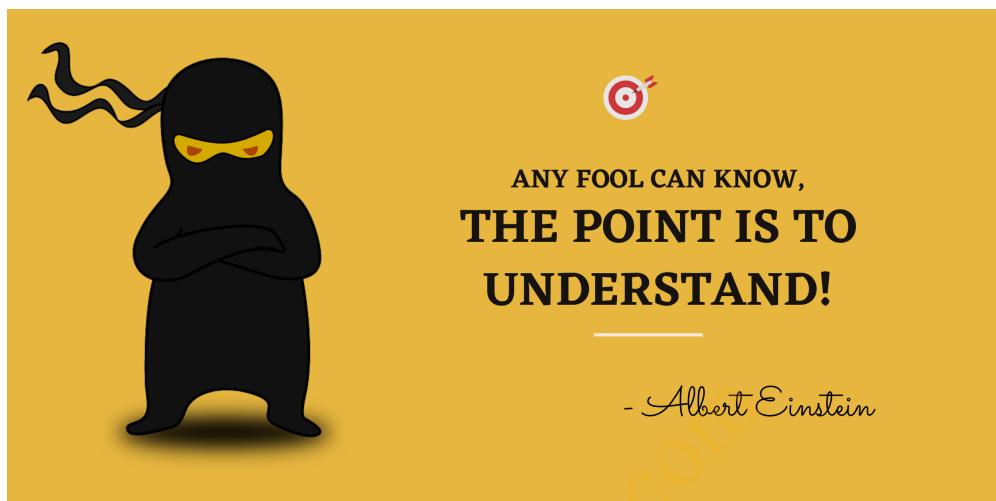
```
# tracepath www.facebook.com
```

```
@lavatech:~$ tracepath www.facebook.com
1?: [LOCALHOST] pmtu 1500
1: _gateway                                2.954ms
1: _gateway                                2.433ms
2: no reply
3: 192.168.25.168                           29.484ms
4: 192.168.25.172                           24.456ms
5: 202.88.186.66                            21.417ms
6: 202.88.186.61                            13.914ms
7: 125.99.43.254                           66.034ms
8: 125.99.43.253                           28.002ms asymm 6
9: 136.232.32.29.static.jio.com            16.881ms asymm 7
```

Figure 14.44: Sample output

Each line in the output of tracepath represents a router or hop that the packet passes through between the source and the final destination.

14.5.5 Practice



- 1. What is the full form of FQDN?**
 - (a) Fully qualified domain name
 - (b) Fully qualified department name
 - (c) Fully qualified domain number
 - (d) Fully qualified department number
- 2. Which of the following is used to assign hostname to a server?**
 - (a) hostnamectl sethostname FQDN
 - (b) hostnamectl set-host FQDN
 - (c) hostnamectl hostname FQDN
 - (d) hostnamectl set-hostname FQDN
- 3. Which of the following file stores hostname details in RHEL8?**
 - (a) /etc/hosts
 - (b) /etc/passwd
 - (c) /etc/fstab
 - (d) /etc/hostname

- 4. Which of the following command is used to display all LAN card devices? (Select all that applies.)**
(a) nmcli con show
(b) nmcli dev show
(c) nmcli device show
(d) nmcli con diplay
- 5. Which of the following command is used to display all network connections? (Select all that applies.)**
(a) nmcli con show
(b) nmcli connection show
(c) nmcli connection display
(d) nmcli con diplay
- 6. Which of the following directory stores network interface card settings?**
(a) /etc/sysconfig/networkscripts/
(b) /etc/network-scripts/
(c) /etc/sysconfig/network-scripts/
(d) /etc/network-scripts/sysconfig/
- 7. Which of the following command is used to add a new connection? (Select all that applies.)**
(a) nmcli con add con-name <connection_name> ifname
 <interface_name> type ethernet
(b) nmcli connection add <connection_name> ifname
 <interface_name> type ethernet
(c) nmcli connection add <connection_name> <interface_name> type
 ethernet
(d) nmcli connection add con-name <connection_name> ifname
 <interface_name> type ethernet

8. Which of the following command is used to set static IP address to a connection? (Select all that applies.)
- (a) nmcli connection mod <connection_name> ipv4.addresses
<ip_address> ipv4.method manual
 - (b) nmcli connection modify <connection_name> ipv4.addresses
<ip_address> ipv4.method manual
 - (c) nmcli connection add <connection_name> ipv4.addresses
<ip_address> ipv4.method manual
 - (d) nmcli connection type <connection_name> ipv4.addresses
<ip_address> ipv4.method manual
9. Which of the following command is used to display all open tcp & udp ports along with PID?
- (a) iostat -c
 - (b) sar
 - (c) netstat -tulpn
 - (d) netstat -r
10. Which of the following command is used to display routing table?
- (a) iostat -c
 - (b) sar
 - (c) netstat -tulpn
 - (d) netstat -r
11. Which of the following command is used to capture network packets?
- (a) iostat
 - (b) tcpdump
 - (c) iostat
 - (d) vmstat

15.1 Logging in Linux

In this section, you are going to learn:

1. **Introduction to logs**
2. **Syslog & rsyslog**
3. **Rules in /etc/rsyslog.conf**
4. **Important log files**
5. **Logger command**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

15.1.1 Introduction to logs

What are logs in Linux?

- Linux logs provide a timeline of some important events.
- Logs are used for troubleshooting issues & problems.
- Location of log files: **/var/log** directory.
- There are Linux logs for: **system, kernel, package managers, boot processes, Xorg, Apache, MySQL**, etc
- Sample log entry from **/var/log/messages** file:

```
Mar 23 11:25:01 server systemd[27221]: Reached target Timers.  
Mar 23 11:25:01 server systemd[27221]: Reached target Paths.  
Mar 23 11:25:01 server systemd[27221]: Listening on D-Bus User Message Bus Socket.  
Mar 23 11:25:01 server systemd[27221]: Reached target Sockets.  
Mar 23 11:25:01 server systemd[27221]: Reached target Basic System.  
Mar 23 11:25:01 server systemd[27221]: Reached target Default.
```

Figure 15.1: Sample log

Understanding log entry:

- Column 1: **Timestamp**
- Column 2: The **hostname of the server where the log is generated**
- Column 3: The **daemon generating the logs**
- Column 4: The **log message**

15.1.2 Syslog & rsyslog

Syslog daemon

- It is the default log manager in Linux.
- It redirects log sorted by **facility and severity** to files under **/var/log** directory.

Rsyslog daemon

- Rsyslog is an advanced version of syslog.
- Configuration file for rsyslog is **/etc/rsyslog.conf**.
- Rsyslog has more advance features like:
 - Lot of modules.
 - Discriminate the log filtering by program, source, message, pid etc.
 - Discard message by configuring rules.

15.1.3 Rules in /etc/rsyslog.conf

The rules in /etc/rsyslog.conf defines what files the logs are redirected according to **facility & severity** using below syntax:

Syntax: facility.severity logfile-name

Eg:

```
##### RULES #####
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                     /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none      /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                    /var/log/secure

# Log all the mail messages in one place.
mail.*                                         - /var/log/maillog

# Log cron stuff
cron.*                                         /var/log/cron

# Everybody gets emergency messages
*.emerg                                         :omusrmsg:*

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                  /var/log/spooler

# Save boot messages also to boot.log
local7.*                                        /var/log/boot.log
```

Figure 15.2: Facility & severity setting in /etc/rsyslog.conf

Let's see more on **facility & severity**.

Facility levels: The facility represents the machine process that created the syslog event. Below are the available facility levels:

| Facility Number | Keyword | Facility Description |
|-----------------|----------|--|
| 0 | kern | kernel messages |
| 1 | user | user-level messages |
| 2 | mail | mail system |
| 3 | daemon | system daemons |
| 4 | auth | security/authorization messages |
| 5 | syslog | messages generated internally by syslogd |
| 6 | lpr | line printer subsystem |
| 7 | news | network news subsystem |
| 8 | uucp | UUCP subsystem |
| 9 | | clock daemon |
| 10 | authpriv | security/authorization messages |
| 11 | ftp | FTP daemon |
| 12 | - | NTP subsystem |
| 13 | - | log audit |
| 14 | - | log alert |
| 15 | cron | clock daemon |
| 16 | local0 | local use 0 (local0) |
| 17 | local1 | local use 1 (local1) |
| 18 | local2 | local use 2 (local2) |
| 19 | local3 | local use 3 (local3) |
| 20 | local4 | local use 4 (local4) |
| 21 | local5 | local use 5 (local5) |
| 22 | local6 | local use 6 (local6) |
| 23 | local7 | local use 7 (local7) |

Figure 15.3: Log facility

Severity also known as Priority: The severity level describes the severity of the log in question.

| Code | Severity | Keyword | Description |
|-------------|-----------------|-------------------|-----------------------------------|
| 0 | Emergency | emerg (panic) | System is unusable. |
| 1 | Alert | alert | Action must be taken immediately. |
| 2 | Critical | crit | Critical conditions. |
| 3 | Error | err (error) | Error conditions. |
| 4 | Warning | warning (warn) | Warning conditions. |
| 5 | Notice | notice | Normal but significant condition. |
| 6 | Informational | info | Informational messages. |
| 7 | Debug | debug | Debug-level messages. |

Figure 15.4: Log Severity

15.1.4 Important log files

- **/var/log/messages:** Stores global system messages during system startup.
- **/var/log/secure:** Store successful and failed logins details & authentication logs.
- **/var/log/boot.log:** All information related to system boot & startup.
- **/var/log/maillog:** Mail servers logs.
- **/var/log/kern:** Stores kernel logs and warning data.
- **/var/log/dmesg:** Messages relating to device drivers. Use **dmesg** command to view messages in this file.
- **/var/log/cron:** Stores all crond-related logs.
- **/var/log/dnf.log:** Stores all yum related logs.
- **/var/log/httpd/** directory: Contains **error_log** and **access_log** files of the apache webserver.
 - The **error_log** contains error logs.
 - The **access_log** contains a record of all requests received over HTTP.
- **/var/log/mysqld.log:** Logs MySQL database related errors & information.

15.1.5 Logger command

- **logger**: Allows a local user to write to log files owned by the root user with default facility **user** & priority **notice**.

Syntax: logger log_message

or

Syntax: logger -p facility.severity log_message

Eg:

```
# logger "Sample log entry 1..."  
# logger -p user.info "Sample log entry 2..."  
  
# Check the log entry in /var/log/messages  
# tail -1 /var/log/messages  
Mar 26 11:27:00 server vagrant[8956]: Sample log  
entry 1...  
Mar 26 11:27:00 server vagrant[8956]: Sample log  
entry 2...
```

15.1.6 Viewing logs

Commands to display all logs:

- **journalctl**: Shows the full system journal logs.

Syntax: `journalctl [options]`

Eg:

```
root@lavatech:~ 11x58
-- Logs begin at Tue 2022-05-03 14:22:25 IST, end at Tue 2022-05-03 18:05:00 IST. --
May 03 14:22:25 lavatech kernel: microcode: microcode updated early to revision 0xea, date = 2021-01-05
May 03 14:22:25 lavatech kernel: Linux version 5.13.0-27-generic (build@lqw01-amd64-045) (gcc (Ubuntu 9.3.0-17ubuntu1) 9.3.0, ld version 2.34-17ubuntu1)
May 03 14:22:25 lavatech kernel: Command line: BOOT_IMAGE=/vmlinuz-5.13.0-27-generic root=UUID=cc3b6551-19bb-4d4c-8f3e-4a2a2a2a2a2a
May 03 14:22:25 lavatech kernel: KERNEL supported cpus:
May 03 14:22:25 lavatech kernel:   Intel GenuineIntel
May 03 14:22:25 lavatech kernel:   AMD AuthenticAMD
May 03 14:22:25 lavatech kernel:   Hygon HygonGenuine
May 03 14:22:25 lavatech kernel:   Centaur CentaurHauls
May 03 14:22:25 lavatech kernel:   sha256: 8baebf
```

Figure 15.5: Sample output

Options with **journalctl** command:

- **-p**: Display logs according to specific priority.

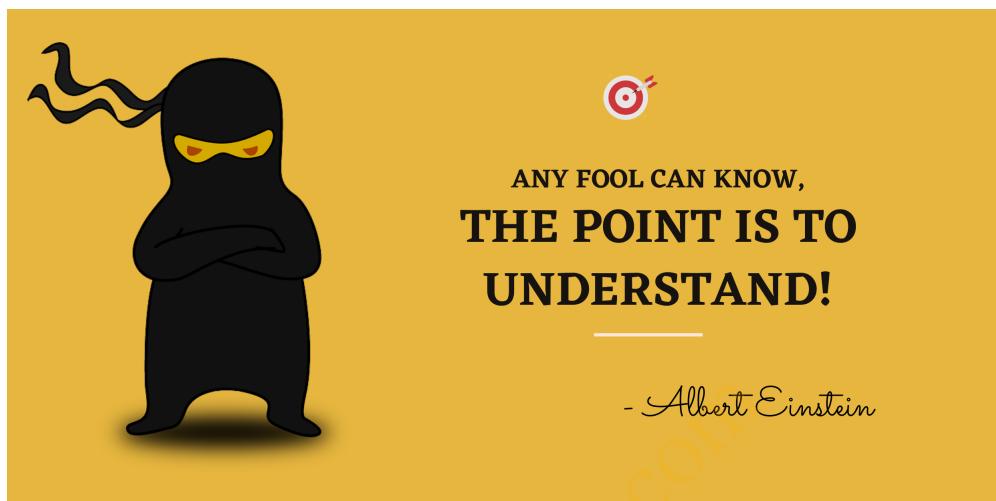
Syntax: `journalctl -p priority`

Eg:

```
# journalctl -p err
```

- **-b**: Show the log messages since the last boot of the system.

Syntax: `journalctl -b`

15.1.7 Practice

- 1. Which of the following directory stored all logs in Linux?**
 - (a) /var/spool/log
 - (b) /var/log
 - (c) /var/tmp
 - (d) /var/tmp/log
- 2. Which of the following is the configuration file for rsyslog?**
 - (a) /etc/syslog.config
 - (b) /etc/rsyslog
 - (c) /etc/rsyslog.conf
 - (d) /etc/rsyslog.config
- 3. Which of the following are valid facility level? (Select all that applies.)**
 - (a) kern
 - (b) cron
 - (c) mail
 - (d) auth

4. Which of the following are valid priority level? (Select all that applies.)

- (a) emerg
- (b) crit
- (c) error
- (d) info

5. Which of the following log file is used to store booting & startup logs?

- (a) /var/log/secure
- (b) /var/log/messages
- (c) /var/log/boot.log
- (d) /var/log/boot

6. Which of the following log file is used to store authentication logs?

- (a) /var/log/secure
- (b) /var/log/messages
- (c) /var/log/boot.log
- (d) /var/log/boot

7. Which of the following log file is used to store yum logs?

- (a) /var/log/secure
- (b) /var/log/messages
- (c) /var/log/yum.log
- (d) /var/log/dnf.log

8. Which of the following log file is used to store cronjobs related logs?

- (a) /var/log/crontab
- (b) /var/log/crony
- (c) /var/log/cron
- (d) /var/log/cron.log

9. Which of the following command is used to create a log with facility kern & priority info?

- (a) logger -p kern.info log_message
- (b) logger kern.info log_message
- (c) logger -s kern.info log_message
- (d) logger -d kern.info log_message

lavatechtechnology.com

16.1 Booting in Linux

In this section, you are going to learn:

1. **Introduction to boot process**
2. **BIOS & POST**
3. **MBR**
4. **Bootloader**
5. **Run levels / Targets**
6. **Commands for targets**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

16.1.1 What is boot process?

In computing, booting is the process of starting a computer.

Below diagram shows detailed steps involved in boot process:

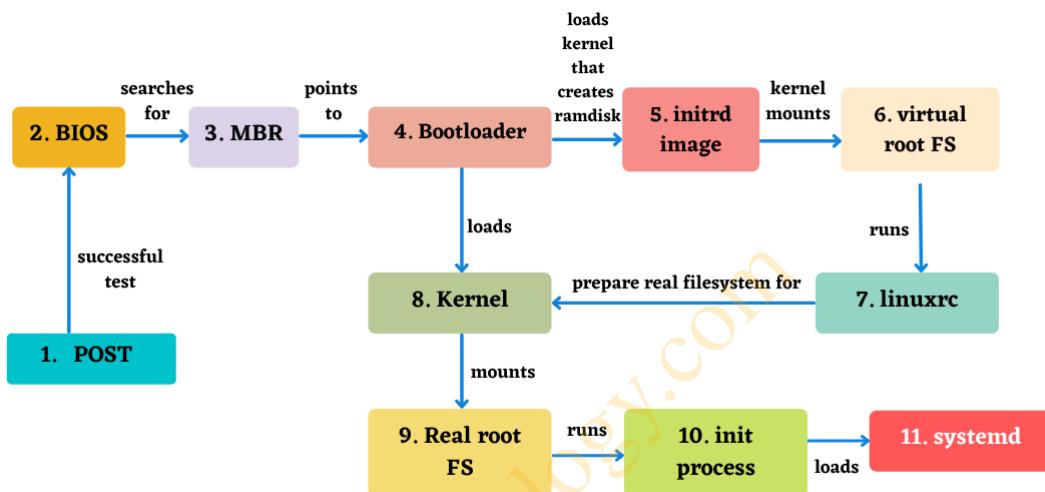


Figure 16.1: Boot process

Let's see each of these steps in detail.

16.1.2 BIOS & POST



Figure 16.2: Powering on PC

Step 1: BIOS

- When you press the start button, the computer first reads a program called as **BIOS**.
- BIOS stands for **Basic Input Output System**.
- BIOS first performs **POST**.
- POST stands for **Power-On Self-Test**.

Step 2: POST

- POST determine if the computer keyboard, RAM, disk drives, and other hardware are working correctly.

Step 3: BIOS

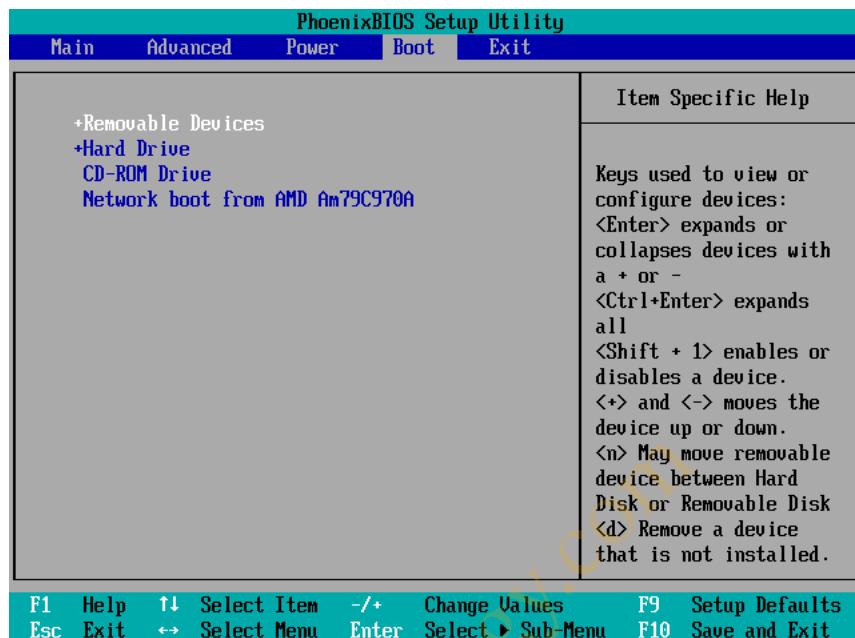


Figure 16.3: BIOS

- Once the POST check is completed successfully, BIOS will look CMOS (stands for Complementary metal–oxide–semiconductor) settings to know what is the boot order:
 - First preference is **Removable Device**.
 - If it's not present, the bios will look at the second device from the boot order settings.
 - The second device is your **hard disk**.

Note: By default, BIOS is not displayed during system startup. According to your hardware, you might have to press **F12** or **F10** or similar key combinations at the start of system boot to see the BIOS settings.

16.1.3 MBR

Step 4: MBR

- BIOS checks for the **MBR** in the hard disk for the location of the **boot loader**.
- Let's see MBR in detail:



First 512 bytes of HDD is MBR

Figure 16.4: MBR

- MBR stands **Master boot recorder**.
- MBR is the first 512 bytes of hard drive.
- It contains:
 - * **Bootloader:** Also called **stage 1 bootloader**. It is of 446 bytes in size. More on this in next page.
 - * **Partition table:** Contains details about primary & extended partition. Refer **Chapter 8: Partition** under section **8.2.1** for more detail on this.
 - * **Boot signature also called magic number:**
 - Final 2 bytes are called the boot signature.
 - Determine if the selected boot drive is bootable or not.
 - On a disk that contains valid bootstrap code, the last two bytes will always be **0x55 0xAA**.

16.1.4 Bootloader

Step 5: Stage 1 bootloader

- This first 512 bytes of HDD is too small to fit an entire boot loader program to it.
- Hence only the stage 1 bootloader program is present in initial 446 bytes of MBR.
- This stage 1 bootloader load the stage 2 boot loader & file system drivers to the RAM.

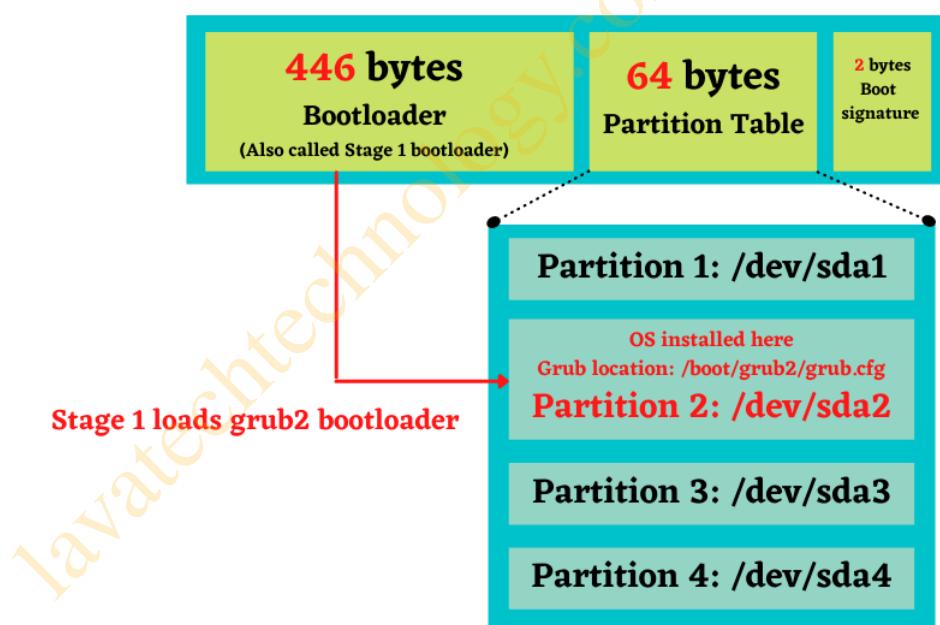


Figure 16.5: Stage 1 bootloader loading stage 2

Step 6: Stage 2 bootloader also known as **Grub2**

- **GRUB2 (GRand Unified Boot Loader version 2)** does the job of loading the kernel (Eg: /boot/vmlinuz-4.18.0-348.el8.x86_64) in RAM.
- Grub 2 configuration file: **/boot/grub/grub.conf**.
- At this stage, you are presented with a **TUI (Terminal user interface)**, where you can select your operating system kernel and press enter to boot it.

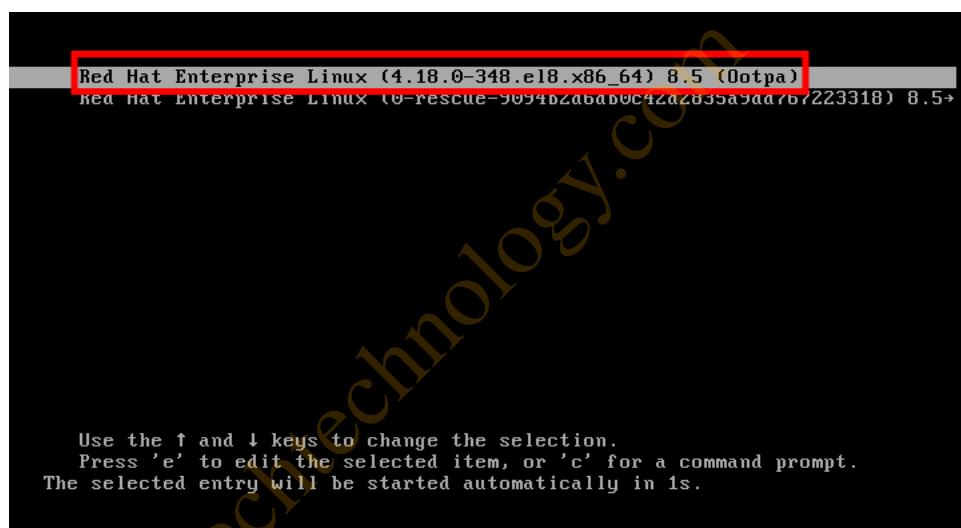


Figure 16.6: TUI

Step 7: Initrd image also known as **initramfs**

- The Linux kernel loads initrd (Initial Ramdisk).
- The kernel uses initrd image files (Eg: /boot/initramfs-4.18.0-348.el8.x86_64.img) as a **temporary virtual filesystem** in the memory.

Step 8: Linuxrc file

- Next, the linuxrc script is executed that creates temporary device nodes in /dev, waits and mount rootfs, switches to real root.

Step 9: Kernel stage

- The Linux kernel based on the result of linuxrc, mount the real root file system i.e "/".
- The **kernel will then start the init process with process ID "1"** as first background process.

Step 10: init/systemd process

- On RHEL 8, /sbin/init is link to **/lib/systemd/systemd**.
- **systemd** looks for **default target** and starts all the units under it.

Let's see more on **runlevels/target** in Linux.

16.1.5 Run levels/Targets

- A run level defines what system services are operating in whole OS.
- There are total 7 run levels.
- Starting with RHEL 7, **run level concept is removed and is replaced with "target units"**.
- Below shows the list of all runlevels and their equivalent targets:

| Run level number | Target Units | Description | File location | Command |
|------------------|------------------------------------|--|---|---------------|
| 0 | runlevel0.target poweroff.target | Shut down and power off the system | /usr/lib/systemd/system/poweroff.target | init 0 |
| 1 | runlevel1.target rescue.target | Set up a rescue shell | /usr/lib/systemd/system/rescue.target | init 1 |
| 2 | runlevel2.target multi-user.target | Set up a non-graphical multi-user system | /usr/lib/systemd/system/multi-user.target | init 2 |
| 3 | runlevel3.target multi-user.target | Set up a non-graphical multi-user system | /usr/lib/systemd/system/multi-user.target | init 3 |
| 4 | runlevel4.target multi-user.target | Set up a non-graphical multi-user system | /usr/lib/systemd/system/multi-user.target | init 4 |
| 5 | runlevel5.target graphical.target | Set up a graphical multi-user system | /usr/lib/systemd/system/graphical.target | init 5 |
| 6 | runlevel6.target reboot.target | Shut down and reboot the system | /usr/lib/systemd/system/reboot.target | init 6 |

Figure 16.7: Targets in Linux

- In RHEL 8, all target unit files are stored under location **/usr/lib/systemd/system/** or **/etc/systemd/system/** and have extension **".target"**.

16.1.6 Systemd

- **Systemd is the first process** that starts on system boot up.
- Systemd is a system and service manager for Linux OS.
- Systemd introduces the concept of systemd units.
- Units are represented as unit configuration files located under:
 - /usr/lib/systemd/system/
 - /run/systemd/system/
 - /etc/systemd/system/

List all systemd units:

Syntax: `systemctl -t help`
Syntax: `systemctl list-unit-files`

```
[root@server ~]# systemctl -t help
Available unit types:
service
socket
target
device
mount
automount
swap
timer
path
slice
scope
```

Figure 16.8: Sample output

Managing services & targets with systemd

systemctl: Controls the systemd system and services.

Syntax: `systemctl [options] [arguments]`

Options with **systemctl** command:

- **start**: Start a service/daemon.

Syntax: `systemctl start [service-name]`

- **status**: Display status of a service/daemon.

Syntax: `systemctl status [service-name]`

Eg:

```
[root@server ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service)
   Active: active (running) since Sat 2022-04-16 11:27:09 UTC; 1 day ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Main PID: 937 (sshd)
      Tasks: 1 (limit: 11402)
        Memory: 8.1M
       CGroup: /system.slice/sshd.service
               └─937 /usr/sbin/sshd -D -u0 -oCiphers=aes256-gcm@openssh.com

Apr 16 11:27:09 server.example.com sshd[937]: Server listening on ::1
Apr 16 11:27:09 server.example.com systemd[1]: Started OpenSSH server.
Apr 16 11:27:16 server.example.com sshd[1151]: Accepted publickey for
Apr 16 11:27:16 server.example.com sshd[1151]: pam_unix(sshd:session)
Apr 16 11:27:39 server.example.com sshd[2710]: Accepted publickey for
Apr 16 11:27:39 server.example.com sshd[2710]: pam_unix(sshd:session)
Apr 17 07:07:44 server.example.com sshd[9498]: Accepted publickey for
Apr 17 07:07:44 server.example.com sshd[9498]: pam_unix(sshd:session)
Apr 17 15:12:43 server.example.com sshd[19780]: Accepted publickey for
Apr 17 15:12:43 server.example.com sshd[19780]: pam_unix(sshd:session)

[lines 1-21/21 (END)]
```

Figure 16.9: Sample output

- **stop**: Stop a service/daemon.

Syntax: `systemctl stop [service-name]`

- **restart:** Start & stop a service/daemon. The PID of daemon will change.

Syntax: `systemctl restart [service-name]`

- **reload:** Reload a service/daemon. The process ID of daemon will not change.

Syntax: `systemctl reload [service-name]`

- **enable:** Enables a service/daemon permanently, which means service/daemon will automatically start on system boot up.

Syntax: `systemctl enable [service-name]`

- **disable:** Disable a service/daemon permanently, which means service/daemon will not automatically start on system boot up.

Syntax: `systemctl disable [service-name]`

- **get-default:** Get default target/runlevel.

Syntax: `systemctl get-default`

Eg:

```
# systemctl get-default  
multi-user.target
```

- **set-default:** Set a default target/runlevel so that the system boots into that specific target.

```
Syntax: systemctl set-default target_name
```

Eg:

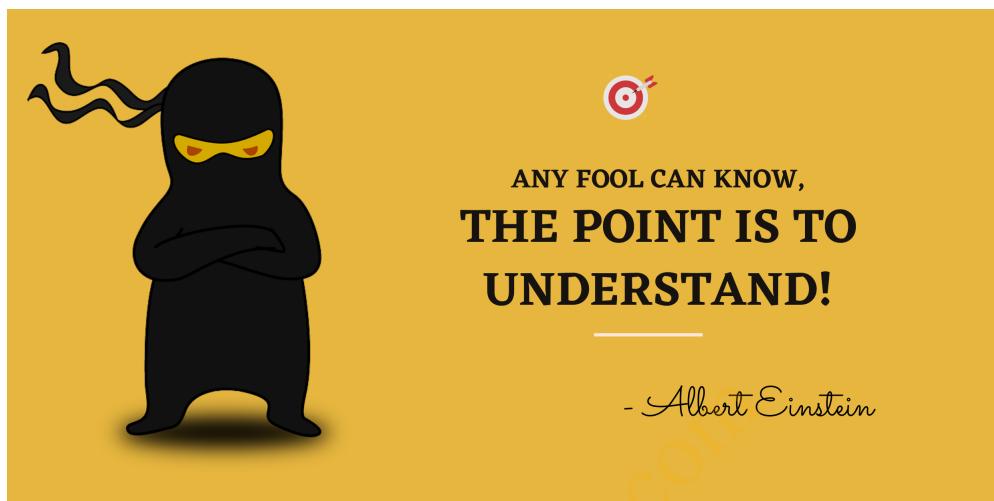
```
# systemctl set-default multi-user.target  
or  
# systemctl set-default graphical.target
```

- **isolate:** Switch to a specific target instantly.

```
Syntax: systemctl isolate target_name
```

Eg:

```
# systemctl isolate multi-user.target  
or  
# systemctl isolate graphical.target
```

16.1.7 Practice

- 1. What is the full form of BIOS?**
 - (a) Basic Input Output System
 - (b) Begin Input Output System
 - (c) Basic Intake Outtake System
 - (d) Basic Input Output State
- 2. What is the full form of GRUB?**
 - (a) GRand Unified Bootloader
 - (b) GRand Unified Bootloader
 - (c) GRand Union Bootloader
 - (d) GRand Union Bootloader
- 3. What is the full form of POST?**
 - (a) Power On Self Terminator
 - (b) Power On System Test
 - (c) Pity On Self Test
 - (d) Power On Self Test

4. What is the full form of MBR?

- (a) Mini Boot Recorder
- (b) Mini Boot Reader
- (c) Mini Booting Recorder
- (d) Master Boot Recorder

5. What is the size of MBR?

- (a) 512 bytes
- (b) 528 bytes
- (c) 446 bytes
- (d) 518 bytes

6. What is the size of partition table in MBR?

- (a) 512 bytes
- (b) 412 bytes
- (c) 64 bytes
- (d) 446 bytes

7. What is magic number?

- (a) Final 12 bytes in MBR that contains boot signature
- (b) Final 12 bytes in MBR that contains boot signature
- (c) Final 64 bytes in MBR that contains boot signature
- (d) Final 446 bytes in MBR that contains boot signature

8. What is the location of grub2 bootloader?

- (a) /etc/boot/grub2/grub.cfg
- (b) /boot/grub.cfg
- (c) /boot/grub2/grub.cfg
- (d) //grub2/grub.cfg

- 9. Who loads the /boot/initramfs-xxxxx.img file?**
 - (a) Linux modules
 - (b) Linux kernel
 - (c) Linux bootloader grub2
 - (d) linuxrc

- 10. Which is the first processs initiated by Linux kernel?**
 - (a) systemd
 - (b) graphical.target
 - (c) initramfs
 - (d) initrd

- 11. Which directory stores all target unit files in RHEL? (Select all that applies.)**
 - (a) /usr/lib/systemd/system/
 - (b) /etc/systemd/system/
 - (c) /usr/share/lib/systemd/system
 - (d) /usr/lib/system

- 12. Which of the following are valid targets units? (Select all that applies.)**
 - (a) multi-user.target
 - (b) graphical.target
 - (c) poweroff.target
 - (d) reboot.target

- 13. Which of the following command is used to switch to multi-user.target in Linux?**
 - (a) init 5
 - (b) init 6
 - (c) init 3
 - (d) init 0

14. Which of the following command is used to poweroff server in Linux?

- (a) init 5
- (b) init 6
- (c) init 3
- (d) init 0

15. Which of the following command is used to switch to graphical.target in Linux?

- (a) init 5
- (b) init 6
- (c) init 3
- (d) init 0

16. Which of the following command is used to reboot server in Linux?

- (a) init 5
- (b) init 6
- (c) init 3
- (d) init 0

17. Which of the following command is used to check default runlevel/target in Linux?

- (a) systemctl get-default
- (b) systemctl set-default
- (c) systemctl give-default
- (d) systemctl display-default

18. Which of the following command is used to set default runlevel/target in Linux?
- (a) systemctl setdefault target_name
 - (b) systemctl set-default target_name
 - (c) systemctl set target_name
 - (d) systemctl set default target_name

lavatechtechnology.com

17.1 Installation in detail

In this section, you are going to learn:

1. **Minimum System Requirements for RHEL**
2. **Partition requirement**
3. **Creating a virtual machine**
4. **RHEL OS installation using anaconda installation**



So let's get started....

17.1.1 Minimum system requirements for RHEL

- **RAM:** 4 GB
- **HDD:** 20 GB unallocated disk space
- **Architecture:** Supports 64-bit x86 or ARM System

32 v/s 64 bit Architecture:

- A 32-bit system can access 2^{32} different memory addresses, i.e 4 GB of RAM or more.
- A 64-bit system can access 2^{64} different memory addresses, i.e actually 18-Quintillion bytes (137,438,953,472 Gb) of RAM.

17.1.2 Partition requirement

/: This is the compulsory partition. Minimum size of this partition is **20 GB**.

Below partition can be created separately to have more structured filesystem:

- **/boot**: Minimum size for this partition is 512 MB.
- **swap**: Recommended swap partition size:

| Amount of RAM installed in system | Recommended swap space |
|-----------------------------------|---|
| 2GB | 2xRAM |
| 2GB – 8GB | = RAM |
| 8GB – 64GB | 4G to 0.5xRAM |
| >64GB | Minimum 4G, depends on type of application installed, number of user etc. |

- **/var**: It is recommended to have **/var** as separate partition to have a separate space for logs, libraries & temporary files. Minimum space required is **5GB**.
- **/home**: It is recommended to have **/home** as separate partition. Minimum recommended size of **8GB**.

Note: It is recommended to install Linux OS using LVM rather than normal partitions.

17.1.3 Creating a virtual machine

- Download VirtualBox hypervisor from <https://www.virtualbox.org/wiki/Downloads> according to your OS:



- Install VirtualBox by running the executable file.
- Next, download RHEL8 ISO from <https://developers.redhat.com/products/rhel/download>

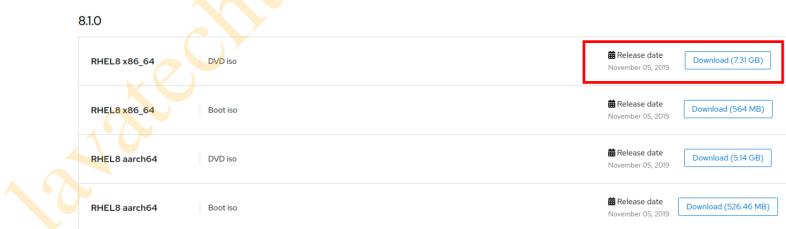
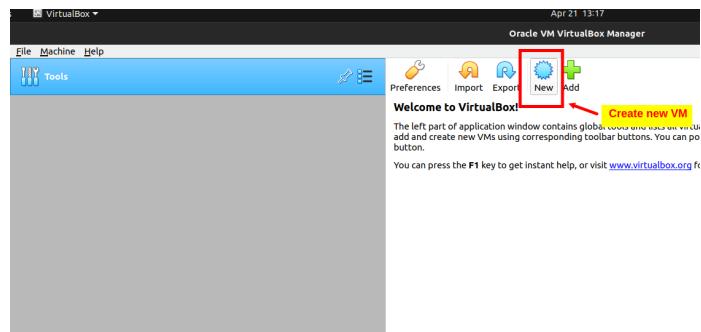
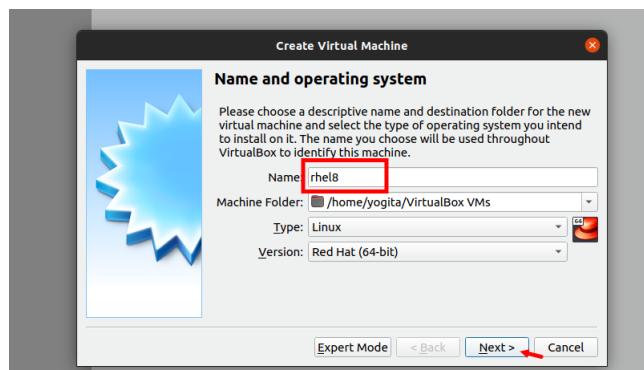


Figure 17.1: Download ISO by clicking on red box

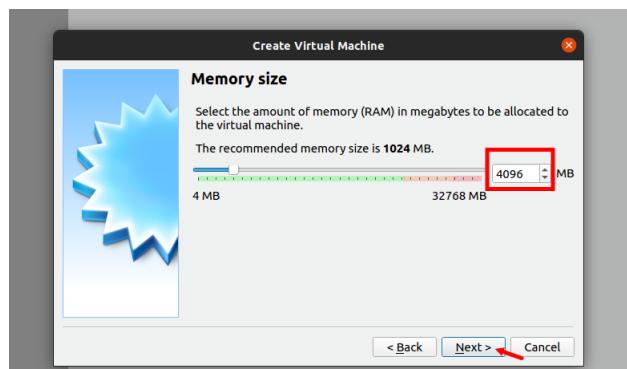
- Open the VirtualBox application and create a new virtual machine(VM) by clicking on new button:



- Provide name to the VM and click next.



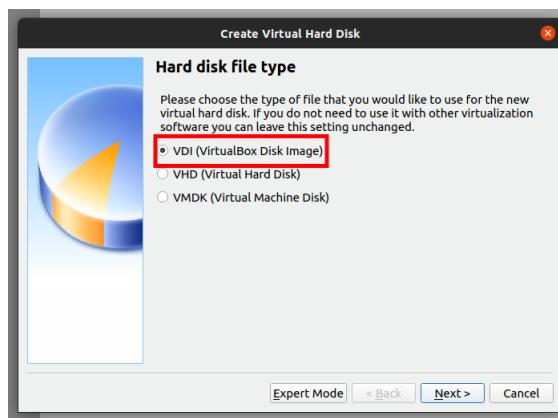
- Enter the RAM size for the VM.



- Create a virtual HDD as shown.



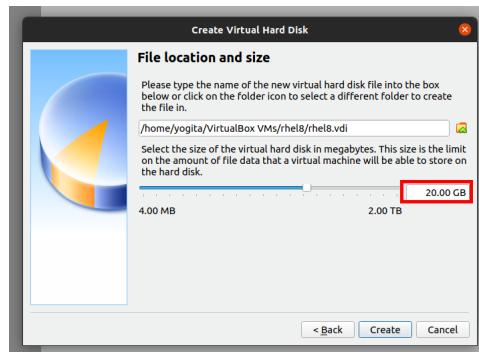
- Select HDD file type as VDI as shown in the image:



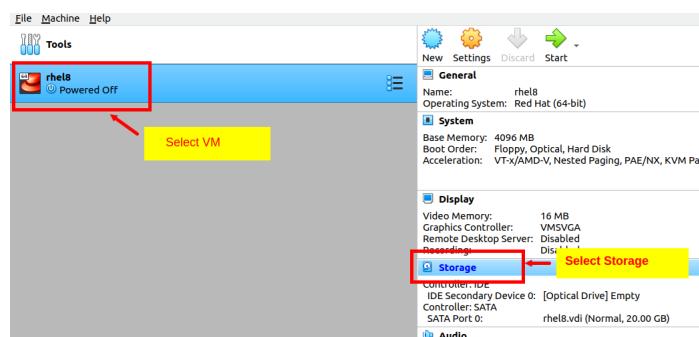
- Select the "**Dynamically allocated**" option so that VM will use physical HDD space as it fills up.



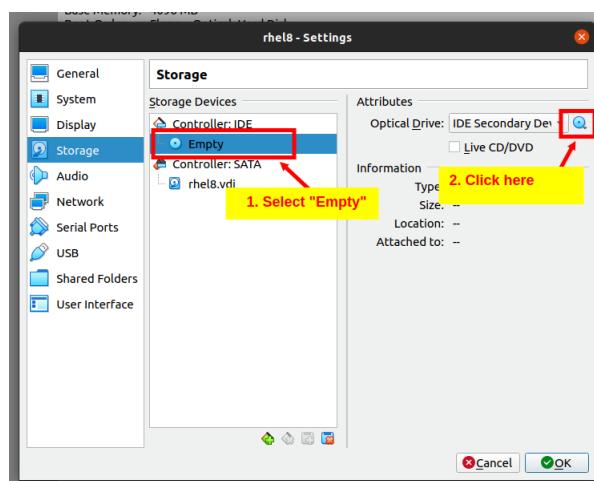
- Enter the virtual HDD size for the VM as per your requirement. In the screenshot shown, the HDD size entered is 20GB.



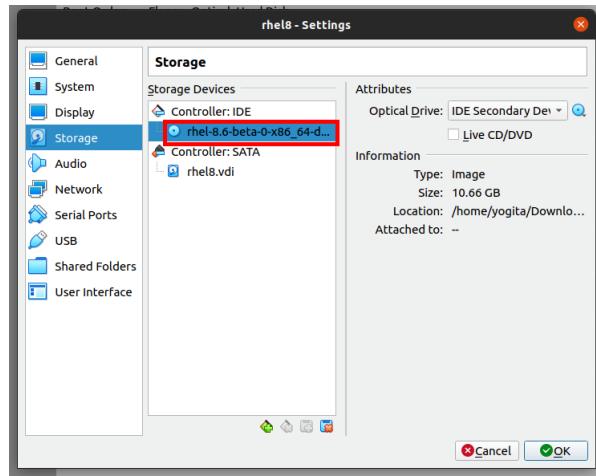
- Now select "Storage" option to upload RHEL8 ISO.



- Upload RHEL 8 ISO by selecting the drive icon as shown.



This is how the uploaded ISO should look like:



17.1.4 RHEL OS using Anaconda

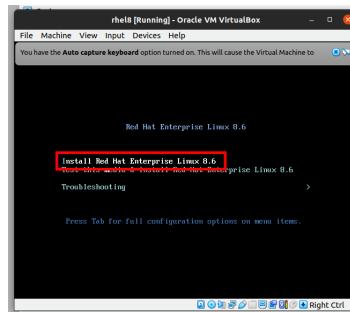
Anaconda is the **installation program** used by Fedora, Red Hat Enterprise Linux and some other distributions. In this chapter, RHEL8 is installed in a virtual machine.

Follow along to install RHEL8 OS:

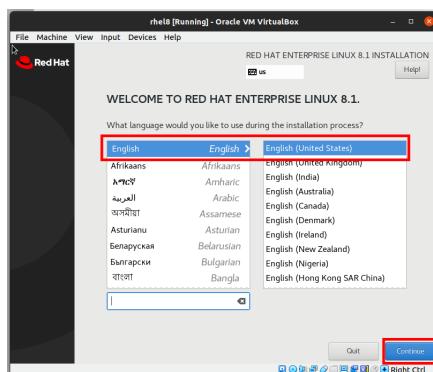
- Select the start button to "ON" the VM.



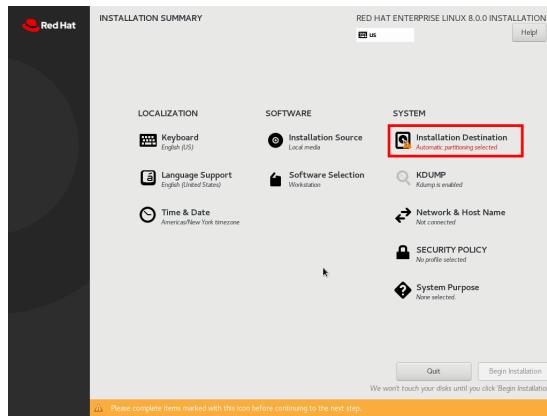
- Select the "Install Red Hat Enterprise Linux 8.1.0" option using **up/down arrow keys on the keyboard**.



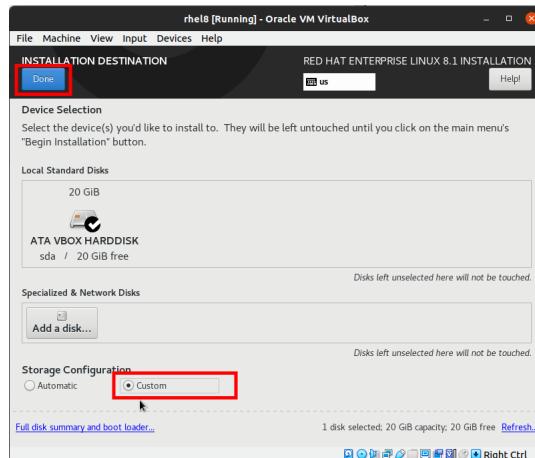
- Select the language of your preference and press continue button.



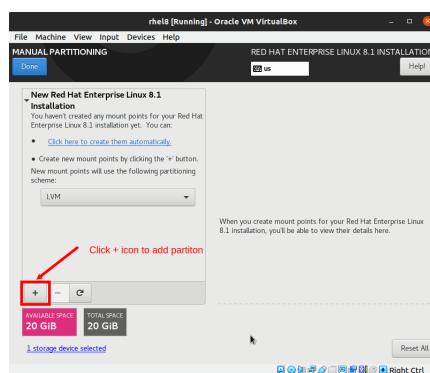
- Select the "Installation Destination" as shown.



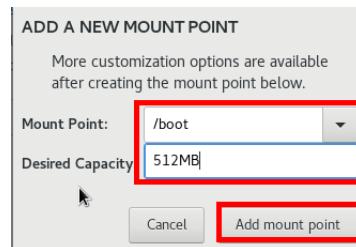
- Select the "Custom" option and click on "Done" as shown.



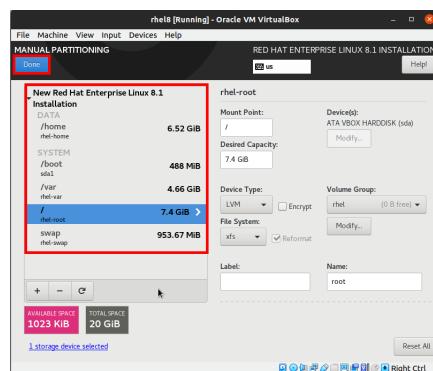
- Add new partition by clicking on "+" icon.



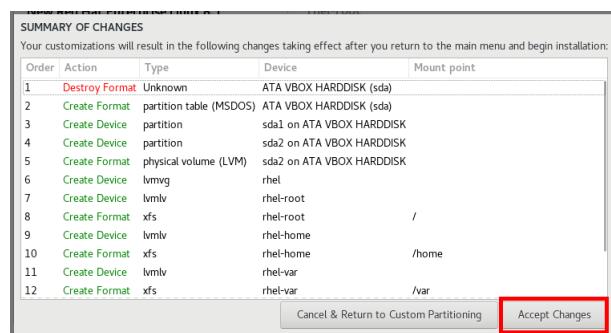
- Create "/boot" partition of size "512MB" as shown.



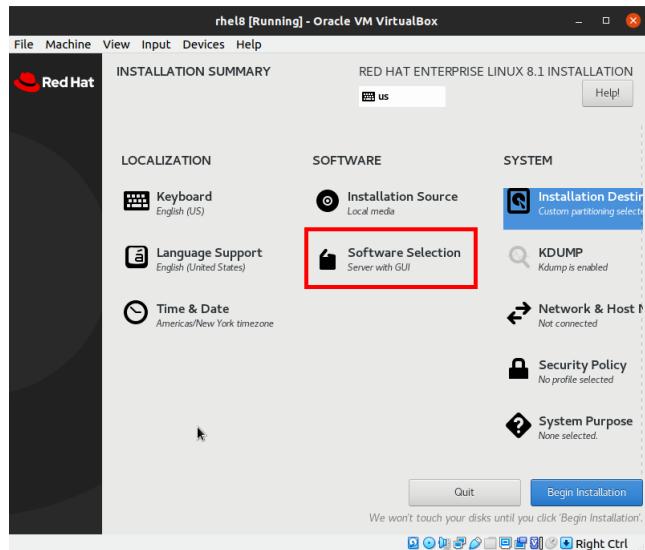
- Similarly create below partitions:
 - "swap" partition of size "1GB"
 - "/var" partition of size "5GB"
 - "/home" partition of size "7GB"
 - Remaining space to "/" partition
- You should see summary of all partitions as shown in the image:



- Accept all the changes as shown:



- Next, select the "Software Selection":



- Select the "Server with GUI" & "Development Tools" as highlighted:

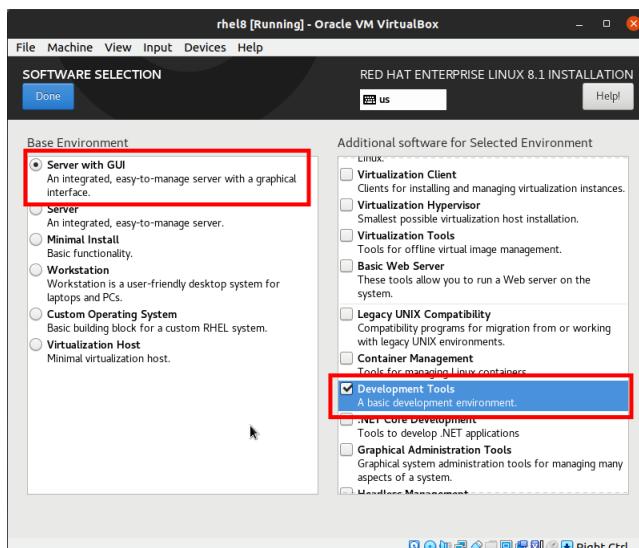
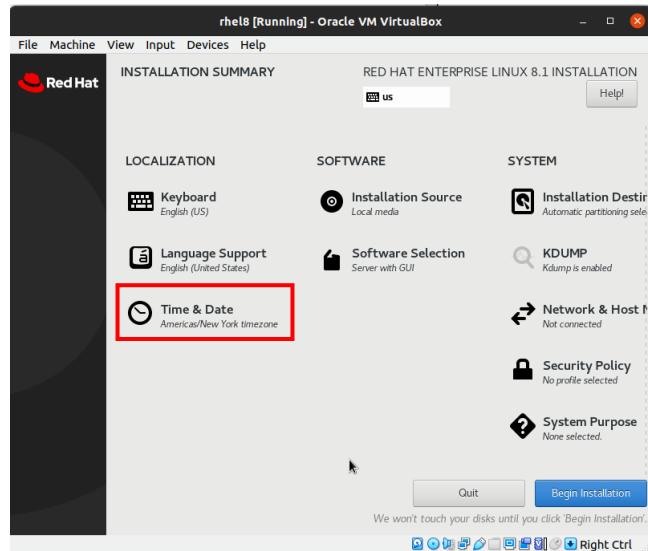
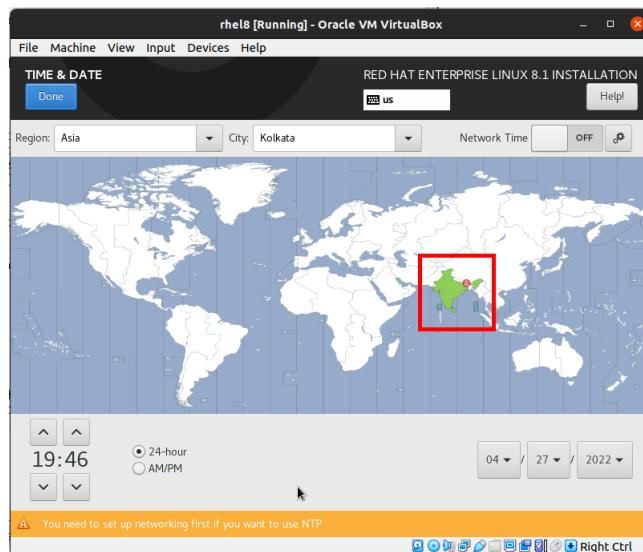


Figure 17.2

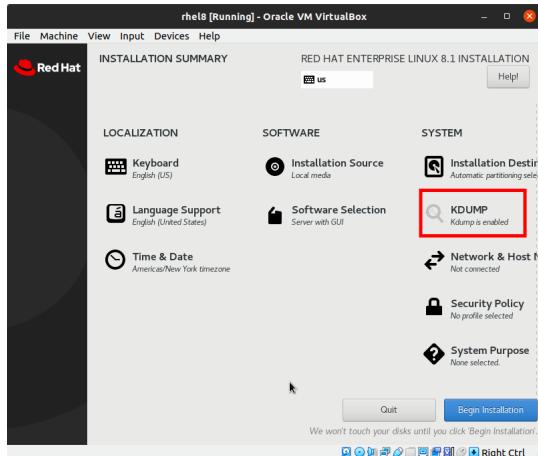
- Select the "Time and Date":



- Select appropriate date and time as shown:



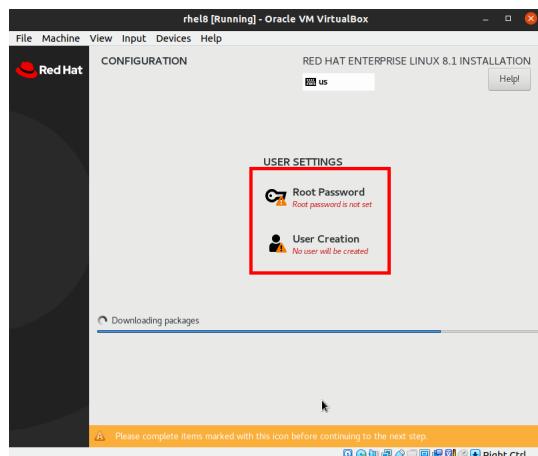
- Next, select "**KDUMP**" to disable it:



- Uncheck the "**Enable kdump**" checkbox to save the memory and click on "**Done**":



- Finally start the installation. During OS installation, set the root password and create user as per your requirement:



18.1 SSH, scp & rsync

In this section, you are going to learn:

1. **What is SSH?**
2. **Setting up SSH server & client**
3. **Password based SSH connection**
4. **Passwordless SSH connection**
5. **What is SCP?**

Finally, there will be a **small exercise** on these topics to check your knowledge.



So let's get started....

18.1.1 What is SSH?

- SSH stand for Secure SHell.
- SSH allows secure **remote access** to Linux server.
- It provides a **secure and encrypted communication over a network**.

Linux to Linux remote connection

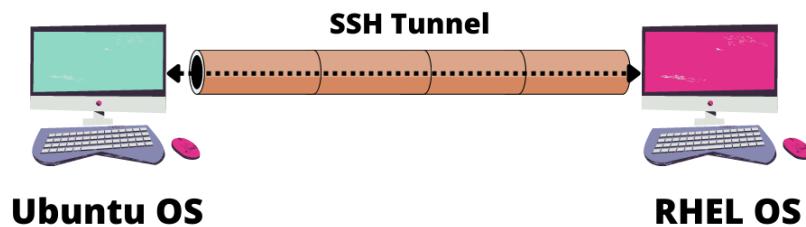


Figure 18.1: SSH connection

- Types of SSH connection:
 - Password based SSH
 - Passwordless or Key based SSH

We shall see more on this coming sections.

18.1.2 What is a server & client?

Server

- Server is a computer system that provides a specific service 24x7.

Client

- Client is a computer system that accesses a specific service from a server.

For eg:

- When you connect to youtube, your system becomes client to youtube server.
- When you connect to facebook, your system becomes client to facebook server.
- When you connect to database, your system becomes client to Database server.

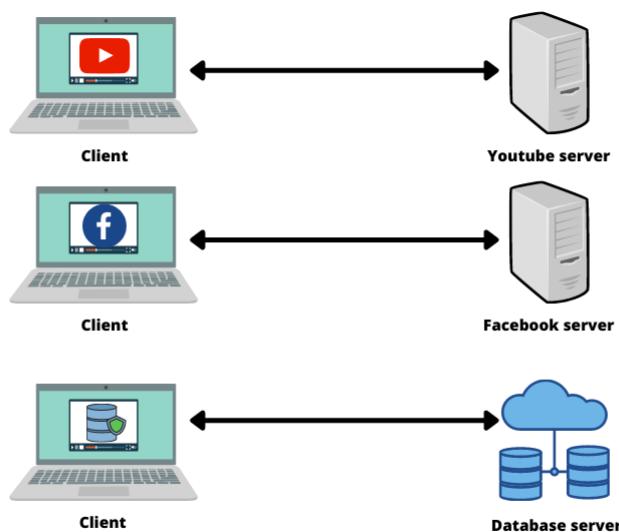


Figure 18.2: Server & client

18.1.3 Setting up SSH server & client on RHEL

SSH server settings

- Install SSH server package.

```
# yum install openssh-server -y
```

- The package installation provides configuration file named **/etc/ssh/sshd_config**.
- Important entries in **/etc/ssh/sshd_config** configuration file:
 - **Line 17 - "Port 22"**: You can change port on this line, default SSH port is 22.
 - **Line 43 - "PermitRootLogin yes"**: You can disable/enable root login on this line.
 - **Line 70 - "PasswordAuthentication no"**: You can disable/enable password authentication on this line.
- Changes in the configuration file are reflected only after the restarting SSH daemon named "**sshd**":

```
# systemctl restart sshd
```

SSH client settings

- Install SSH client package.

```
# yum install openssh-client -y
```

18.1.4 Password based SSH connection

- SSH command:

Syntax: ssh [options] [username@]remote_server_ip

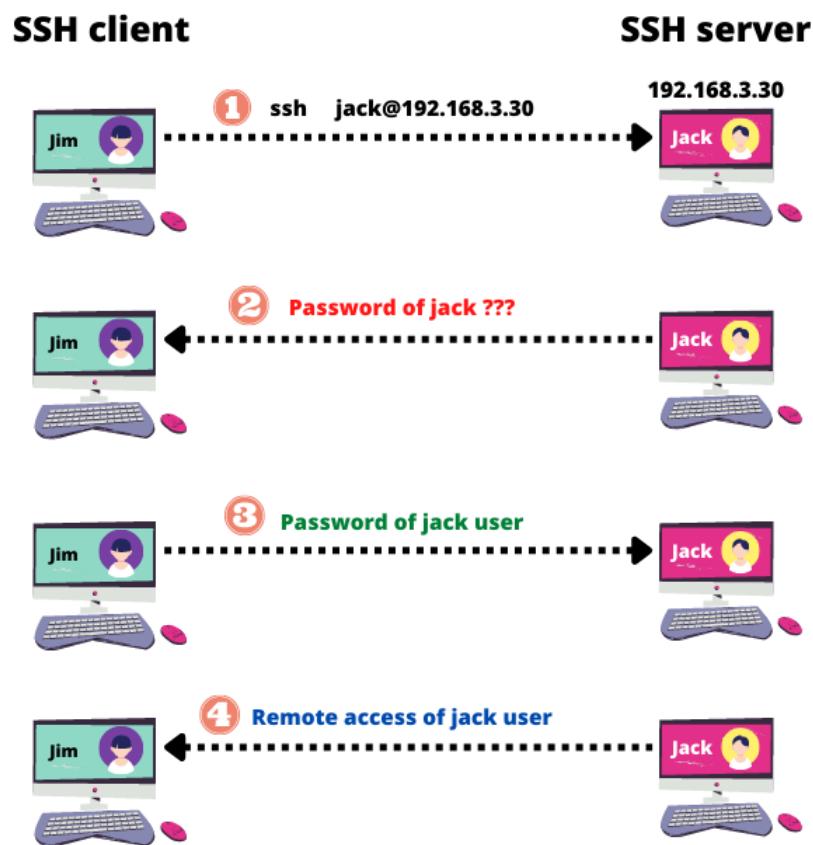


Figure 18.3: SSH connection using ssh command

- **Known host authentication**

- The first time SSH client takes access of SSH server:
 - * The host public key of SSH client is stored in SSH server's "**known_hosts**" file.
 - * This file authenticates the SSH client to the SSH server.
 - * Location of known_host file: "**~/.ssh/known_hosts**".
- Eg:

```
root@lavatech:~# ssh jack@192.168.0.105
The authenticity of host '192.168.0.105 (192.168.0.105)' can't be established.
ECDSA key fingerprint is SHA256:z8dWAdvi4qWp+7YSsENOJ8s8Ndw0ekhtni89mtqaqYk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.105' (ECDSA) to the list of known hosts.
jack@192.168.0.105's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.13.0-27-generic x86_64)
```

Figure 18.4: Sample output

- Execute command on remote system using SSH:

Syntax: ssh [username@]remote_server_ip [command]

Eg:

```
# ssh jack@192.168.0.105 "logger 'Test log from remote system'"
```

- Options with **ssh** command:

- **-v**: Verbose mode to display debugging messages. Maximum 3 "v" options can be supplied.

Syntax: ssh -v [username@]remote_server_ip

Syntax: ssh -vv [username@]remote_server_ip

Syntax: ssh -vvv [username@]remote_server_ip

Eg:

```
# ssh -v jack@192.168.0.105
# ssh -vvv jack@192.168.0.105
```

- **-p**: Used to change port while taking remote access.

Syntax: ssh -p port_number [username@]remote_server_ip

Eg:

```
# ssh -p 8080 jack@192.168.0.105
```

18.1.5 Passwordless SSH connection

- Problem with password based SSH is, you need to **share remote user's account password** in order to take its access.
- This can be a security problem.
- **Solution: Passwordless or Key based SSH.**
- Passwordless SSH uses a public & private key-pair instead of user's account password while taking remote access.

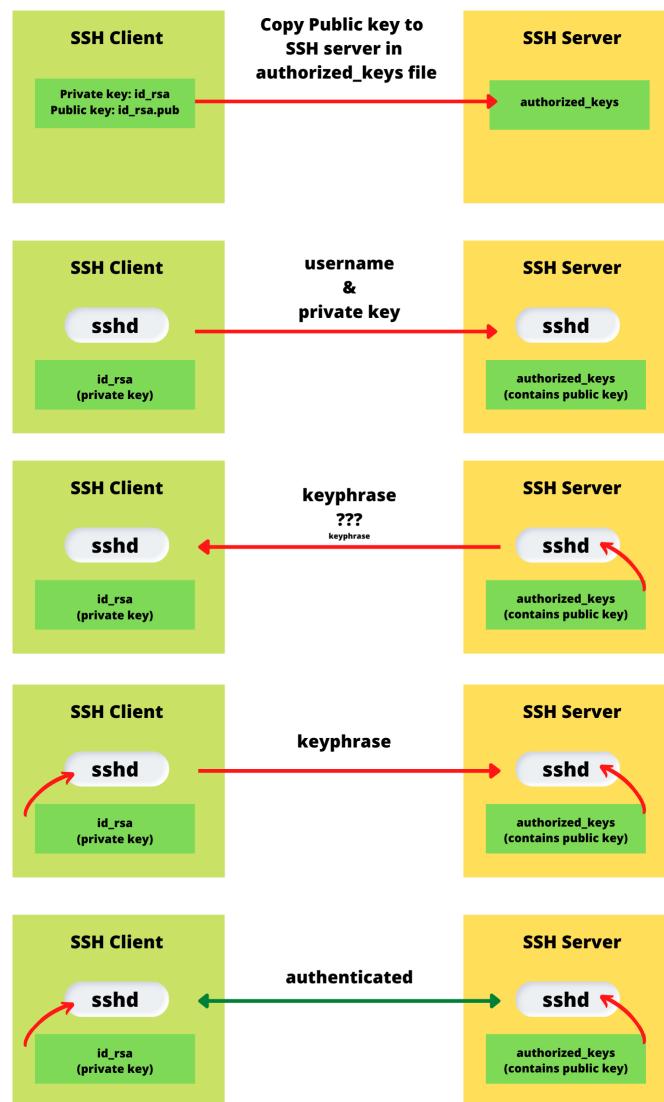


Figure 18.5: Working of Passwordless SSH

Steps to configure passwordless SSH

On SSH client:

- Generate public-private key-pair:
 - **ssh-keygen**: Used to generate the key pair.

Syntax: ssh-keygen

- Public key is generated as **~/.ssh/id_rsa.pub** & private key is generated as **~/.ssh/id_rsa**.
- By default, RSA algorithm is used key-pair. Other algorithms are DSA, EDSA etc.

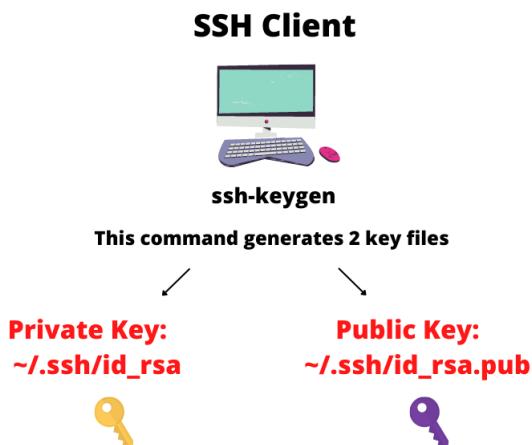


Figure 18.6: Public & Private key pair

Eg:

```

[jim@client ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jim/.ssh/id_rsa):
Enter passphrase (empty for no passphrase): Enter passphrase
Enter same passphrase again: Enter passphrase
Your identification has been saved in /home/jim/.ssh/id_rsa.
Your public key has been saved in /home/jim/.ssh/id_rsa.pub.
The key fingerprint is:
  
```

Figure 18.7: Sample output

- **Copy public-key from SSH client to SSH server:**
 - **ssh-copy-id:** Copy public key (eg. `~/.ssh/id_rsa.pub`) to SSH server in `~/.ssh/authorized_keys` file.
 - This command will ask you the **password of user's account on SSH server.**

Note: In order to copy keys, SSH server's `/etc/ssh/sshd_config` file should have setting: "**PasswordAuthentication yes**".

Syntax: `ssh-copy-id username@server_ip_address`

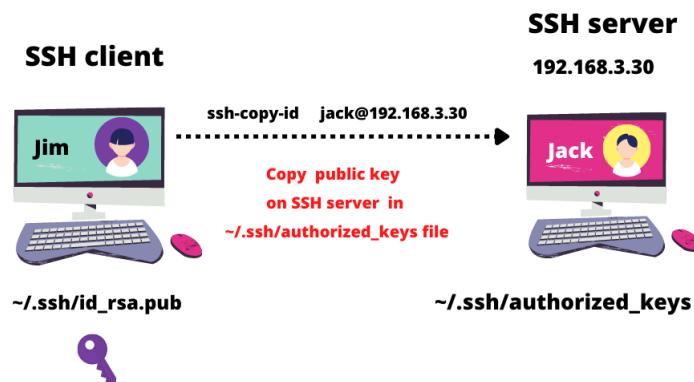


Figure 18.8: Copy public key from SSH client to SSH server

Eg:

```
[jim@client ~]$ ssh-copy-id jack@192.168.0.108
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/jim/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it
install the new keys
jack@192.168.0.108's password: ← Enter password for jack
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'jack@192.168.0.108'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 18.9: Sample output

- Finally from SSH client, take passwordless SSH of SSH server: Eg:

```
[jim@client ~]$ ssh jack@192.168.0.108
Last login: Tue Apr 19 17:37:02 2022
[jack@server ~]$
[jack@server ~]$
```

Figure 18.10: Sample output

On SSH server

- Once you have configured passwordless SSH, disable password based login permanently.

```
# grep -w PasswordAuthentication /etc/ssh/sshd_config
PasswordAuthentication no
```

- Restart sshd daemon to reflect the changes in configuration file.

```
# systemctl restart sshd
```

Important things to note about SSH server & client

| | |
|---|-------------------------|
| SSH Port | 22 |
| SSH server package name | openssh-server |
| SSH client package name | openssh-client |
| Configuration file | /etc/ssh/sshd_config |
| Daemon name | sshd |
| Location of public-private key-pair on SSH client | ~/.ssh/ |
| Passwordless ssh command executed on SSH client | ssh-keygen, ssh-copy-id |

18.1.6 What is scp command?

- **scp:** Stands for Secure CoPy. It is used to copy data securely from one Linux server to another.
scp command will ask for remote SSH server's user account password or key passphrase.
- **Copy file using scp:**

```
Syntax:      scp      [option]      [user@]SRC_HOST:]file1  
[user@]DEST_HOST:]file2
```

Eg: Copy file from remote SSH server to SSH client's /tmp folder.

```
#  scp  jack@192.168.0.108:  /one.txt  /tmp
```

Eg: Copy file from SSH client's /home/jim/data.txt to remote SSH server's /tmp folder.

```
#  scp  /home/jim/data.txt  jack@192.168.0.108:/tmp
```

- **Copy folder using scp:**

```
Syntax:      scp      -R      [user@]SRC_HOST:]folder1  
[user@]DEST_HOST:]folder2
```

Eg: Copy folder from remote SSH server to SSH client's /tmp folder.

```
#  scp  -R  jack@192.168.0.108:  /data  /tmp
```

Eg: Copy folder /home/jim/data of SSH client to remote SSH server's /tmp folder.

```
# scp -R /home/jim/data jack@192.168.0.108:/tmp
```

18.1.7 What is rsync command?

- **rsync:** Securely copy files from one system to another. It differs from scp in that if two files or directories are similar between two systems, rsync only needs to copy the differences between the systems, while scp would need to copy everything.
- **Copy file using rsync:**

```
Syntax:      rsync      [option]      [user@]SRC_HOST:]file1  
[user@]DEST_HOST:]file2
```

Eg: Copy file from remote SSH server to SSH client's /tmp folder.

```
# rsync jack@192.168.0.108: /one.txt /tmp
```

Eg: Copy file from SSH client's /home/jim/data.txt to remote SSH server's /tmp folder.

```
# rsync /home/jim/data.txt jack@192.168.0.108:/tmp
```

- **Copy folder using rsync:**

```
Syntax:      rsync      -r      [user@]SRC_HOST:]folder1  
[user@]DEST_HOST:]folder2
```

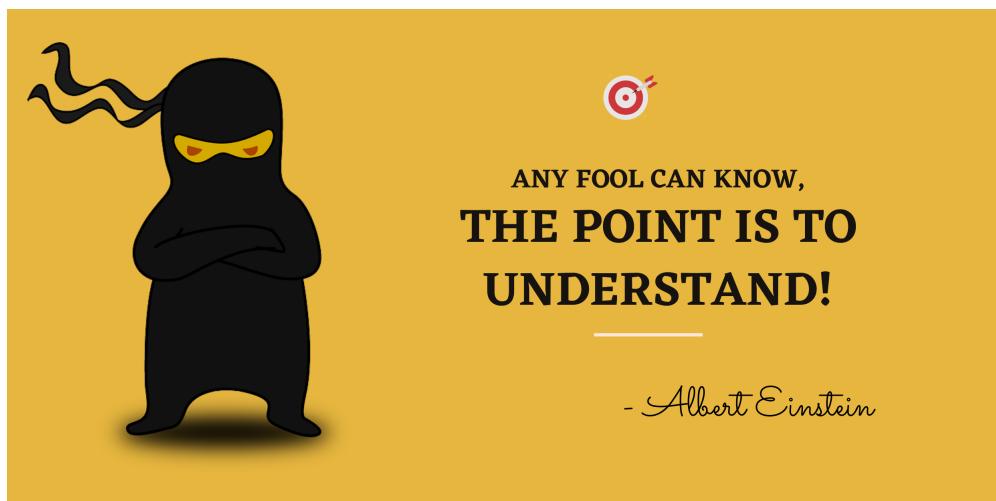
Eg: Copy folder from remote SSH server to SSH client's /tmp folder.

```
# rsync -r jack@192.168.0.108: /data /tmp
```

Eg: Copy folder /home/jim/data of SSH client to remote SSH server's /tmp folder.

```
# rsync -r /home/jim/data jack@192.168.0.108:/tmp
```

18.1.8 Practice



- 1. What is the full form of SSH?**
 - (a) Secure Smart Shell
 - (b) Secure Shell
 - (c) Secure Support Shell
 - (d) Smart Shell
- 2. Which of the following is configuration file for SSH?**
 - (a) /etc/ssh_config
 - (b) /etc/sshd/ssh_config
 - (c) /etc/ssh/sshd_config
 - (d) /etc/sshd/sshd_config
- 3. Which of the following option is used to display verbose debugging message while taking ssh connection?**
 - (a) ssh -v username@remote_server_ip
 - (b) ssh -vv username@remote_server_ip
 - (c) ssh -vvv username@remote_server_ip
 - (d) ssh -p username@remote_server_ip

- 4. Which of the following command is used to create public & private key pair using SSH?**
 - (a) ssh-keygen
 - (b) ssh-copy-id
 - (c) ssh -key
 - (d) ssh key-gen
- 5. Which of the following command is used to copy SSH public key from SSH client to SSH server?**
 - (a) ssh-keygen
 - (b) ssh-copy-key
 - (c) ssh -key
 - (d) ssh-copy-id
- 6. What is the following command is used to copy file/folder securely from server to client?**
 - (a) scp
 - (b) rsync
 - (c) ssh-copy
 - (d) ssh-copy-data

