CHAPTER 1

INTRODUCTION

* 1. Objective:

Redhat enterprise linux introduces the Linux operating system. The installation and description of different Linux OS along with file organizations are covered in this internship. The commands for file handing and filter commands along with user and group permissions are also covered in this internship.

* 1. Overview:

The Linux system administrator is responsible for installing the server of the system, updating configuration, operating and maintaining the hardware and software of the system. He/She has to manage failures in the system, troubleshoot and maintain the servers, the user accounts, security, etc. He/ She is also responsible for managing the disk space as well as the backup. The system administrator is responsible for changing the contents of the file or its attributes like its ownership or permission and can delete any file even if it is write-protected. This professional also needs to reset new password and constantly monitor the special services and security of the servers. They are expected to remain informed with the recent knowledge in Linux system in order to assess the hardware and software technologies.

* 1. Scope:

For IT professionals, various courses and certification programs are organized and offered by reputed institutions in collaboration with top brands the world. **RHCSA Certification** is also one of the most demanding types, of course, that proves your ability to securely deploy and configure networking services on Linux servers running Red Hat Enterprise Linux. It is also the right option to enhance our knowledge and skill in the areas of Linux system automation and administration. For those who want to give a strong start to their career with a good salary hike and designation in an esteemed organization, pursuing such courses is beneficial in a number of ways. A good scope is visible with secure career growth and good fortune.

CHAPTER 2

ORGANIZATION PROFILE

2.1 KGTTI

Karnataka German Technical Training Institutes (KGTTIs) are set up by Karnataka German Multi Skill Development Centre (KGMSDC), a Society promoted by Government of India and Government of Karnataka with technical support of Deutsche Gesellschaft fur Internationale Zusammenarbeit GmbH – International Services (GIZ-InS).

KGTTIs are located at Bengaluru, Kalaburagi, Belagavi, Mangaluru and Hubballi. The primary mission of KGTTIs is to provide broad-based multidisciplinary world class training programs in various technical fields, directed towards development of specialized skills in alignment with the industry requirements. KGTTIs will train and produce high-quality skilled workforce relevant to current and emerging labour market needs and promote entrepreneurship skills for the youth of Karnataka.

VISION:

To develop KGMSDCs as world class training centers that offer specialized skills training program in alignment with the Industry requirements in Karnataka and beyond.

MISION:

KGMSDC will generate high quality skilled manpower in close association with the industry, while operating with operational flexibility and striving financial self-sustainability.

2.2 Red Hat Enterprise Linux

Red Hat Enterprise Linux is a Linux distribution developed by Red Hat and targeted toward the commercial market. Red Hat Enterprise Linux is released in server versions for x86-64, Power Architecture, ARM64, and IBM Z, and a desktop version for x86-64. All of Red Hat’s official support and training, together with the Red Hat Certification Program, focuses on the Red Hat Enterprise Linux platform. Red Hat Enterprise Linux is often abbreviated to RHEL. The first version of Red Hat Enterprise Linux to bear the name originally came into the market as “Red Hat Linux Advanced Server”.

In 2003, Red Hat rebranded Red Hat Linux Advanced Server to “Red Hat Enterprise Linux AS”, and added two more variants, Red Hat Enterprise Linux ES and Red Hat Enterprise Linux WS.

Red Hat uses strict trademark rules to restrict free re-distribution of their officially supported versions of Red Hat Enterprise Linux, and yet freely provides its source code. Third-party derivatives can be built and redistributed by stripping away the non-free components like Red Hat’s trademarks.

Examples include community-supported distributions like CentOS and Scientific Linux, and commercial forks like Oracle Linux. To maintain a stable application binary interface (ABI), Red Hat does not update the kernel version, but instead backports new features to the same kernel version with which a particular version of RHEL has been released.

New features are backported throughout the Production 1 phase of the RHEL lifecycle. Consequently, RHEL

may use a Linux kernel with a dated version number, yet the kernel is up-to-date regarding not only security fixes but also certain features.

CHAPTER 3

LINUX OPERATING SYSTEM

Linux operating system is one of the popular versions of the UNIX operating system, which is designed to offer a free or low cost operating system for personal computer users. It gained the reputation as a fast performing and very efficient system. This is a remarkably complete operating system, including a GUI (graphical user interface), TCP/IP, the Emacs editor, X Window System, etc.

Linux is a family of free and opensource software operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991 by Linus Torvalds. Linux is typically packaged in a Linux distribution (or distro for short). Distributions include the Linux kernel and supporting system software and libraries, many of which are provided by the GNU Project.

Many Linux distributions use the word “Linux” in their name, but the Free Software Foundation uses the name GNU/Linux to emphasize the importance of GNU software. Linux was originally developed for personal computers based on the Intel x 86 architecture, but has since been ported to more platforms than any other operating system.

Linux is the leading operating system on servers and other big iron systems such as mainframe computers and is the only OS used on TOP 500 supercomputer (since November 2017, having gradually eliminated all competitors).

Linux is one of the most prominent examples of free and opensource software collaboration. The source code may be used, modified and distributed-commercially or non-commercially by anyone under the terms of its respective licenses, such as the GNU General Public License.

3.1 THE LINUX ARCHITECTURE

3.1.1 Components of Linux System

Linux Operating System has primarily the following three components as shown in figure 3.1:

1. Kernel: Kernel is the core part of Linux. It is responsible for all major activities of Linux operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
2. System Library: System libraries are special functions or programs using which application programs or system utilities access Kernel’s features. These libraries implement most of the functionalities of the operating system and do not require kernel module’s code access rights.
3. System Utility: System Utility programs are responsible to do specialized, individual level tasks.

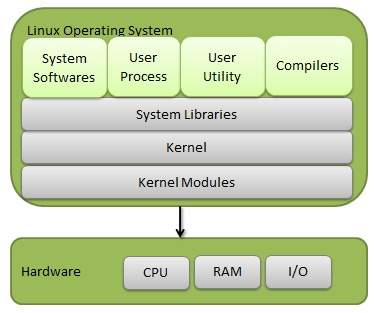


Fig 3.1 Components of Linux OS

3.1.2 ARCHITECTURE OF A LINUX SYSTEM

The following illustration shows the architecture of a Linux system in figure 3.2

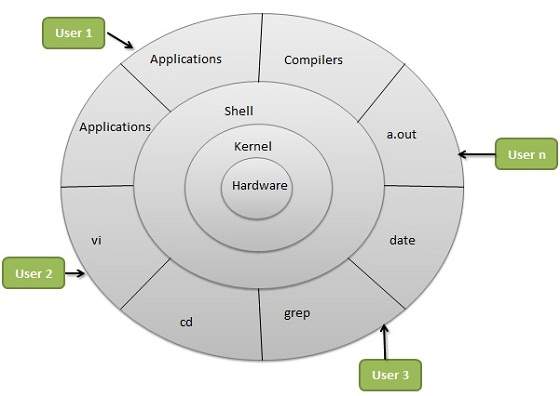


Fig 3.2 Architecture of Linux System

The architecture of a Linux System consists of the following layers –

**Hardware Layer**: Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).

**Kernel**: It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.

**Shell**: An interface to kernel, hiding complexity of kernel’s functions from users. The shell takes commands from the user and executes kernel’s functions.

**Utilities**: Utility programs that provide the user most of the functionalities of an operating system.

3.2 BASIC FEATURES OF LINUX OPERATING SYSTEM

Following are some of the important features of Linux Operating System.

* **Portable**: Portability means software can work on different types of hardware in the same way. Linux kernel and application programs support their installation on any kind of hardware platform.
* **Open Source**: Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
* **Multi-User**: Linux is a multiuser system which means multiple users can access system resources like memory / RAM / application programs at the same time.
* **Multiprogramming**: Linux is a multiprogramming system which means multiple applications can run at the same time.
* **Hierarchical File System**: Linux provides a standard file structure in which system files/ user files are arranged.
* **Shell**: Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations such as call application programming etc.
* **Security**: Linux provides user security by using authentication features like password protection/ controlled access to specific files/ encryption of data. Linux is fast, free and easy to use, power laptops and servers around the world. Linux has many more features such as:
* Live CD/USB: Almost all Linux distributions have Live CD/USB feature by which user can run/try the OS even without installing it on the system.
* Graphical user interface (X Window System): People think that Linux is a command line OS; it is somewhat true also but not necessarily, Linux has packages which can be installed to make the whole OS graphic based as Windows.
* Supports most national or customized keyboards: Linux is used worldwide and hence it is available in multiple languages, and it supports most of their custom national keyboards.

3.3 SOFTWARE AND HARDWARE REQUIREMENTS

3.3.1 SOFTWARE REQUIREMENT

To use your local computer to develop your server, you must install a Linux system. Windows can also be used to create & deploy servers but carrying these tasks in windows becomes difficult. It’s recommended to use Linux system. RedHat Enterprise Linux 8.0.0 is one of the best Linux OS that can be used.

INSTALLIG THE REDHAT ENTERPRISE LINUX

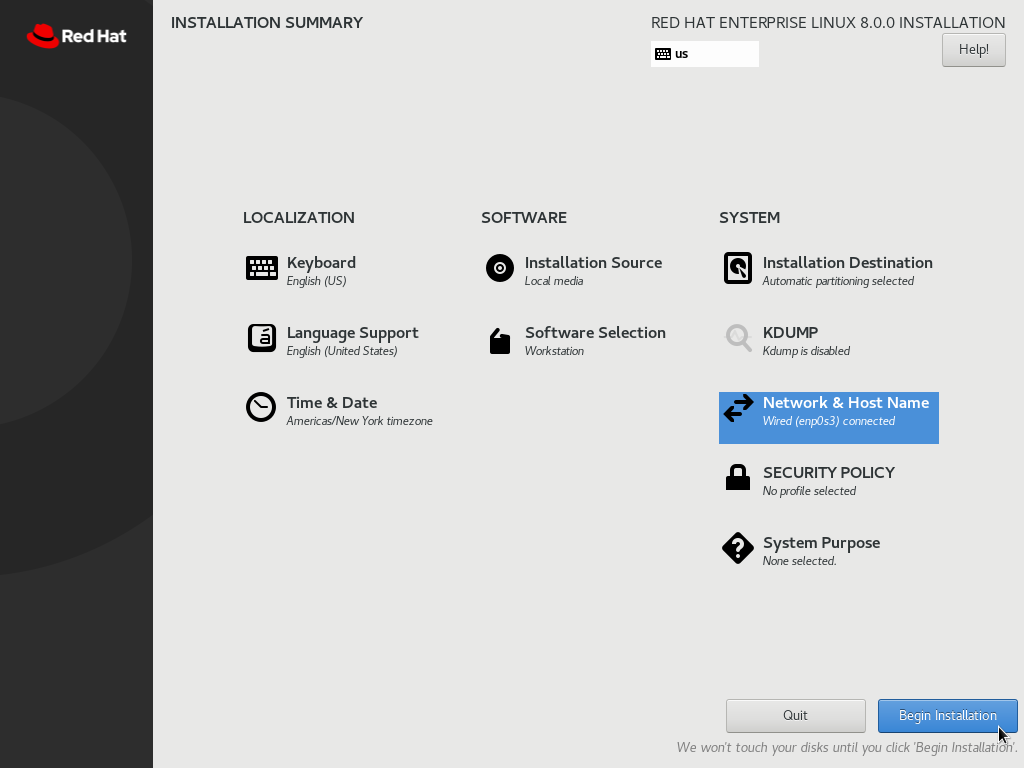


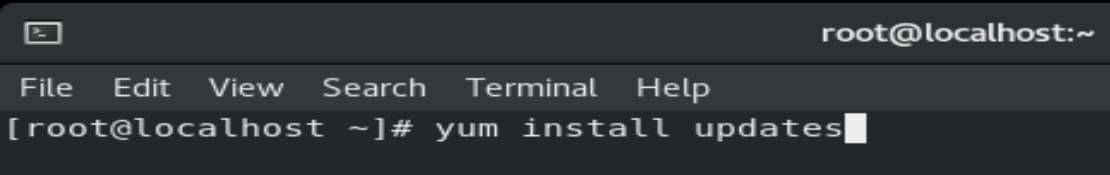
Fig 3.3 Installing Redhat Enterprise Linux

Installing a Linux system is easy and fast task. There is one more reason to use Linux system is because it’s free.

Configuring the system:

As the Linux system is installed i.e RHEL 8.0.0, log in as root. Now we’ve to configure it by installing some additional packages and upgrading the system packages.

Open the Terminal and type following commands to install updates:



3.3.2 HARDWARE EQUIREMENT

Minimum requirement is Pentium 4 or AMD or Celeron Processor. All the processors above this configuration would be very well working to go with Linux. So, the processors like Core 2 Duo Processor, Dual Core Processor, Dual core i3, Dual core i5, Dual core i7, AMD Duron, AMD Sempron, AMD Turion, MD Opteron, AMD Phenom 1, and Celeron III are recommended. Minimum of 512 MB RAM is required and the RAM above this size would be recommended.

CHAPTER 4

NETWORK CONFIGURATION

To configure a network interface using Network Manager, we can use following tools:

* the text user interface tool, nmtui.
* the command-line tool, nmcli.
* the graphical user interface tools, GNOME GU

4.1 NMTUI

To configure a network interface IPv4 address, start by invoking the **nmtui** tool.  A graphical user interface that provides just a subset of features compated to nmcli. Using nmtui, you can edit a connection, activate a connection or change the hostname of your computer.

The **nmtui** tool is used in a terminal window. It is contained in the NetworkManager-tui package, but it is not installed along with **NetworkManager** by default. To install NetworkManager-tui:

# yum install NetworkManager-tui

Type nmtui in the terminal

#nmtui

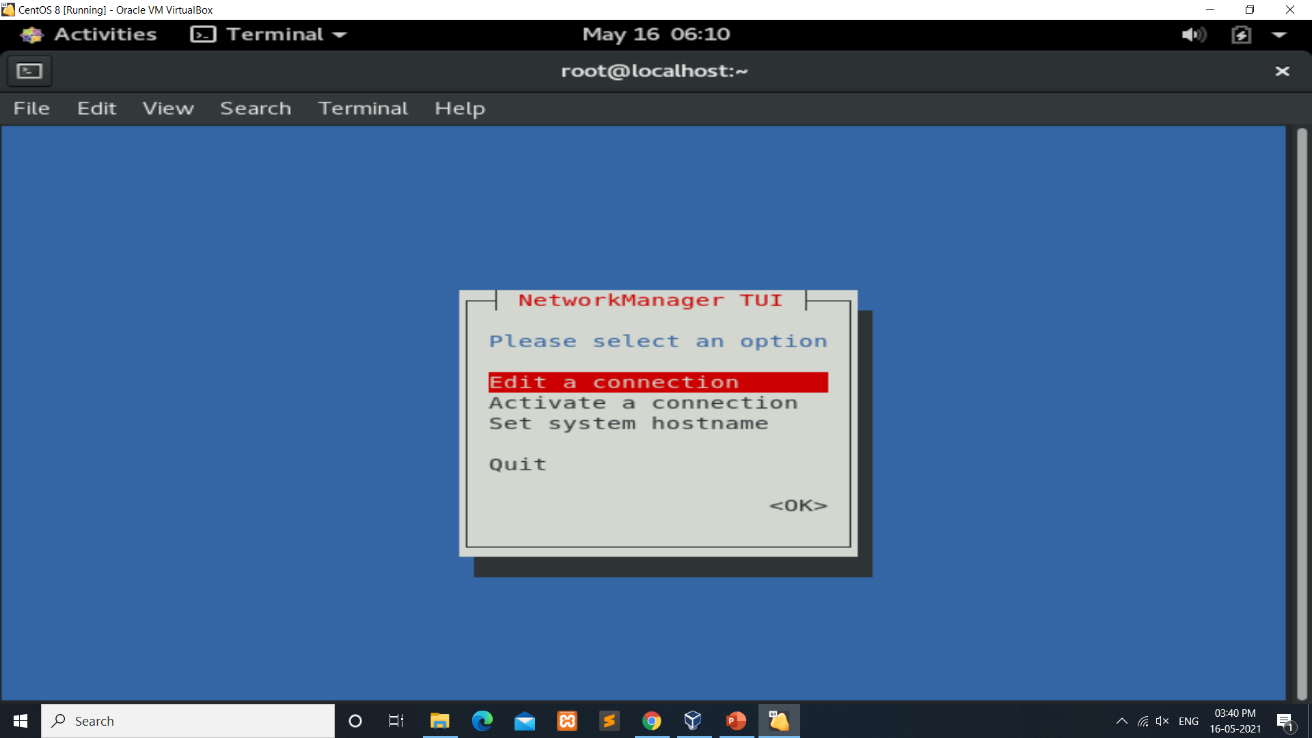


Fig 4.1 NMTUI command

In this graphical menu, you have three options:

* Edit a connection: where you are able to select network interfaces and modify parameters assigned to them (such as the DNS, the IP address or the gateway address);
* Activate a connection: but also, disactivate existing connections. As a reminder, connections are only active or enabled whenever they are assigned to a specific device;
* Set system hostname: like the “hostnamectl” or the “hostname”, you can set the PC name over a network.

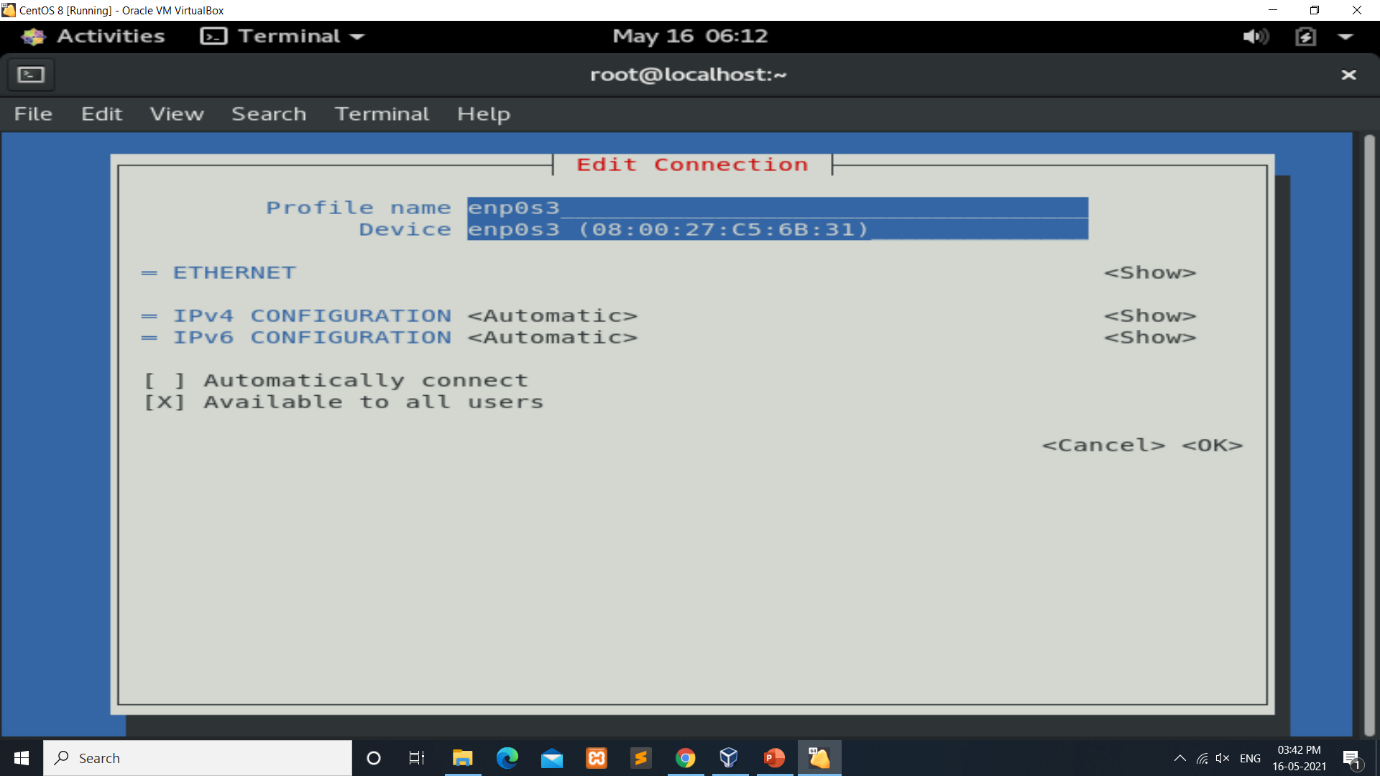


Fig 4.2 Network configuration using NMTUI

4.2 NMCLI

The **nmcli** (NetworkManager Command Line Interface) command-line utility is used for controlling NetworkManager and reporting network status. It can be utilized as a replacement for **nm-applet** or other graphical clients. **nmcli** is used to create, display, edit, delete, activate, and deactivate network connections, as well as control and display network device status.

The **nmcli** utility can be used by both users and scripts for controlling networkmanager:

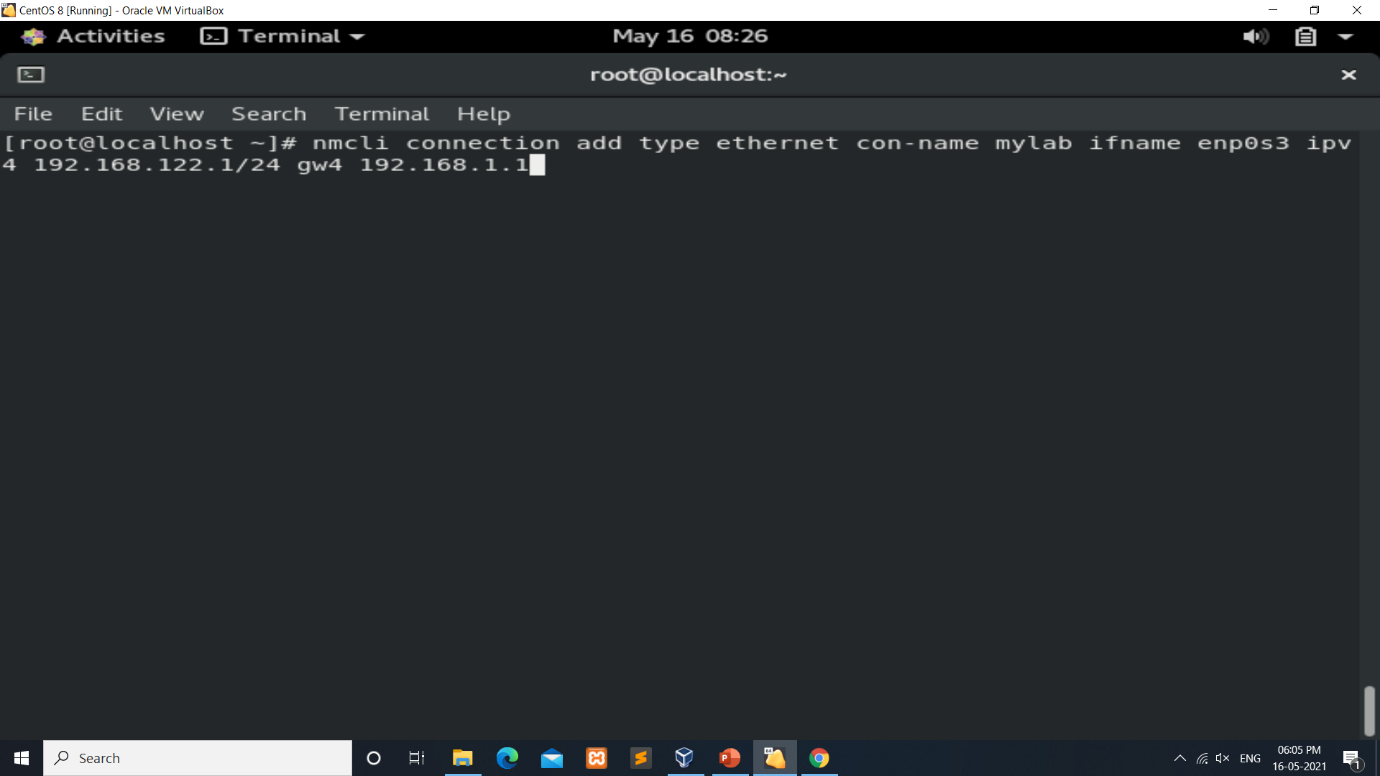
* For servers, headless machines, and terminals, **nmcli** can be used to control **NetworkManager** directly, without GUI, including creating, editing, starting and stopping network connections and viewing network status.
* For scripts, **nmcli** supports a terse output format which is better suited for script processing. It is a way to integrate network configuration instead of managing network connections manually.

The basic format of a **nmcli** command is as follows:

# nmcli [options] OBJECT {COMMAND | help }

where OBJECT can be one of the following options:

general, networking, radio, connection, device, agent, and monitor. You can use any prefix of these options in your commands. For example, nmcli con help, nmcli c help, nmcli connection help generate the same output.

 Fig 4.3 NMCLI Command

CHAPTER 5

DISK PARTITIONING

A hard disk can be divided into several partitions. Each partition functions as if it were a separate hard disk. The idea is that if you have one hard disk, and want to have two operating systems on it, you can divide the disk into two partitions.

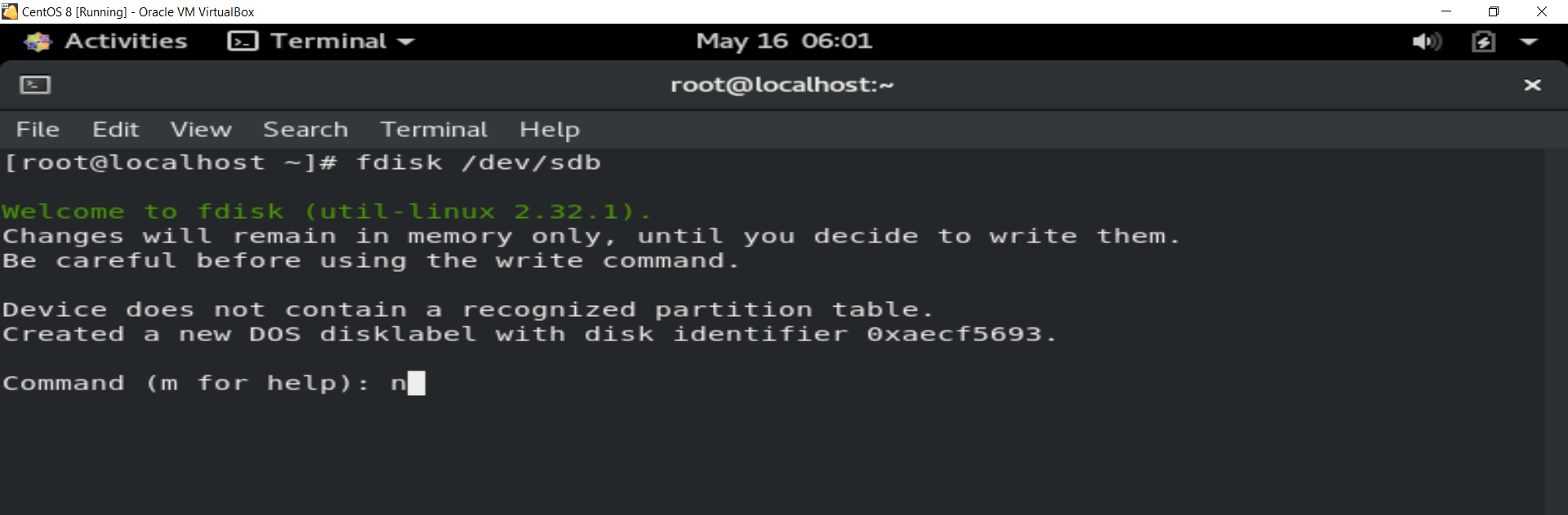
Each operating system uses its partition as it wishes and doesn’t touch the other ones. In this way the two operating systems can co-exist on the same hard disk. The information about how a hard disk has been partitioned is stored in its first sector (that is, the first sector of the first track on the first disk surface). The first sector is the master boot record (MBR) of the disk; this is the sector that the BIOS reads in and starts when the machine is first booted.

The master boot record contains a small program that reads the partition table, and checks which partition is active (that is, marked bootable), and reads the first sector of that partition, the partition’s boot sector (the MBR is also a boot sector, but it has a special status and therefore a special name). This boot sector contains another small program that reads the first part of the operating system stored on that partition (assuming it is bootable), and then it starts it.

The partitioning scheme is not built into the hardware, or even into the BIOS. It is only a convention that many operating systems follow. Not all operating systems do follow it, but they are the exceptions. Some operating systems support partitions, but they occupy one partition on the hard disk, and use their internal partitioning method within that partition.

The latter type exists peacefully with other operating systems (including Linux), and does not require any special measures, but an operating system that doesn’t support partitions cannot co-exist on the same disk with any other operating syste

* Type ‘m’ to see the list of all available commands of fdisk.
* To create a new partition, we have to use ‘n’ command.

 Fig 5.1 fdisk command

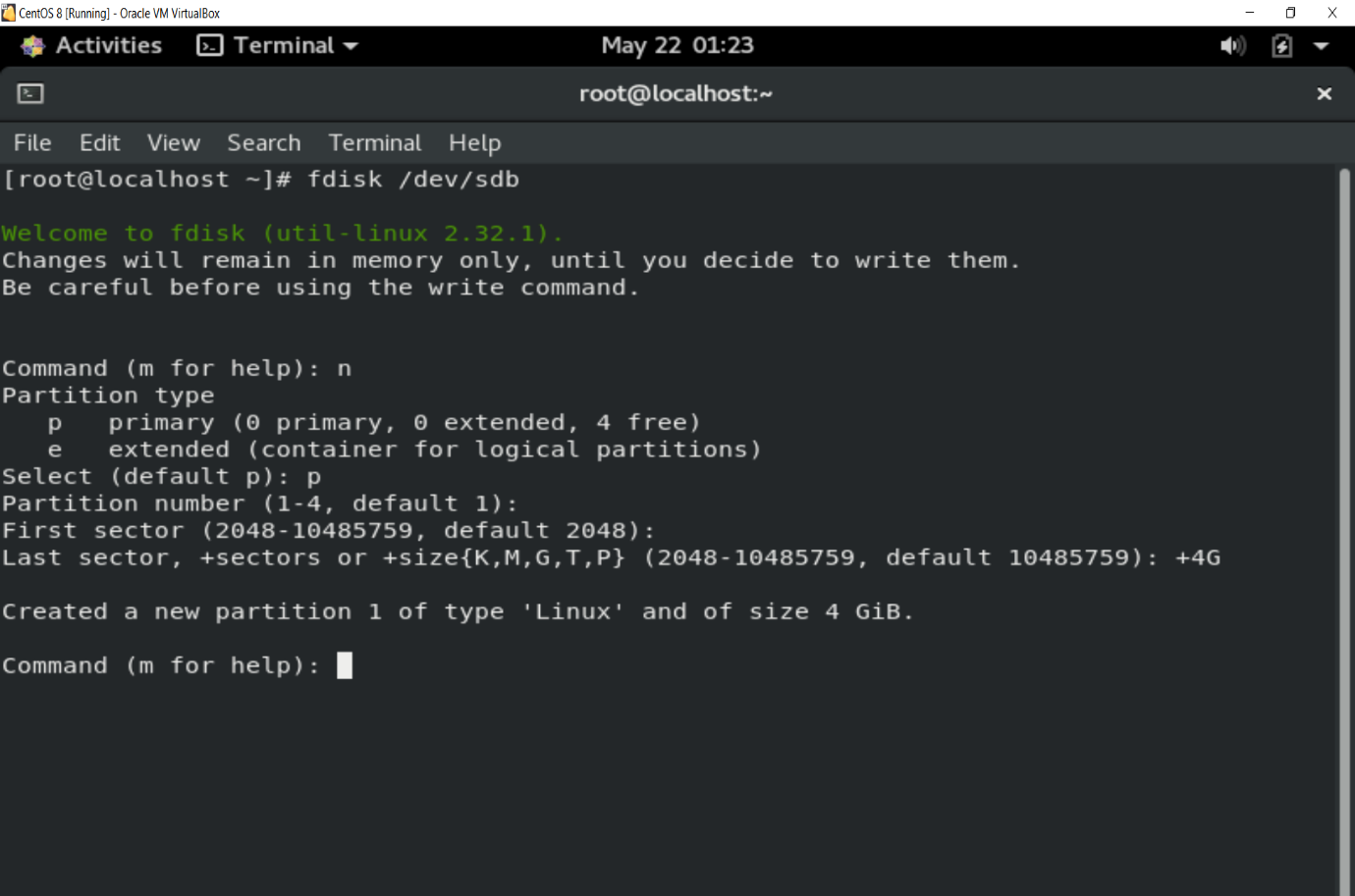
The original partitioning scheme for PC hard disks allowed only four partitions. This quickly turned out to be too little in real life, partly because some people want more than four operating systems (Linux, MS-DOS, OS/2, Minix, FreeBSD, NetBSD, or Windows/NT, to name a few), but primarily because sometimes it is a good idea to have several partitions for one operating system.

For example, swap space is usually best put in its own partition for Linux instead of in the main Linux partition for reasons of speed. To overcome this design problem, extended partitions were invented.

This trick allows partitioning a primary partition into sub-partitions. The primary partition thus subdivided is the extended partition; the sub-partitions are logical partitions.

They behave like primary partitions but are created differently. There is no speed difference between them. By using an extended partition you can now have up to 15 partitions per disk.A partition can be used in lots of different ways. One of the common ways is to use it as an ordinary filesystem. There are 3 steps you can do this:

1. Create the partition – this is done using the “fdisk” tool. We’ll cover how to do this in this article.
2. Format the partition – This is also known as installing a filesystem. It is done using the “mkfs” command
3. Mount the partition – done using the “mount” tool (and automate it by updating the fstab config file)

 Fig 5.2 Disk Partitioning

A disk partition must be formatted and mounted before use. The formatting process can also be done for several other reasons, such as changing the file system, fixing errors, or deleting all data.

There are three ways to format disk partitions using the mkfs command, depending on the file system type:

* ext4
* FAT32
* NTFS

The general syntax for formatting disk partitions in Linux is:

mkfs [options] [-t type fs-options] device [size]

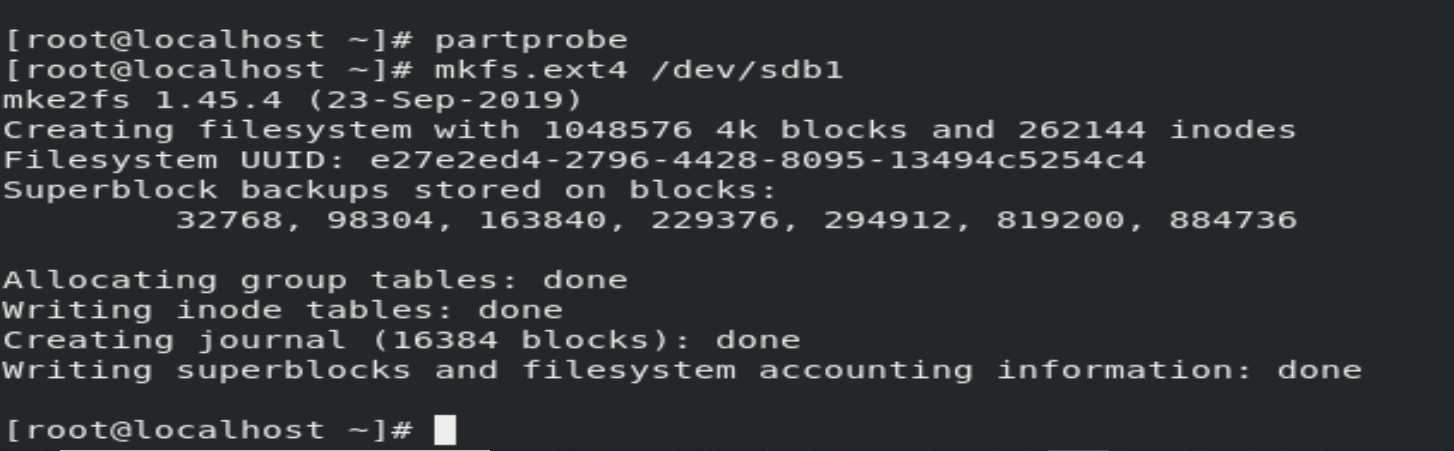


Fig 5.3 Disk Formatting

Before using the disk, create a mount point and mount the partition to it. A mount point is a directory used to access data stored in disks.

1. Create a mount point by entering:

mkdir -p [mountpoint]

2. After that, mount the partition by using the following command:

mount -t auto /dev/sdb1 [mountpoint]

The mount command attaches the filesystem of an external device to the filesystem of a system.

It instructs the operating system that filesystem is ready to use and associate it with a particular point in the system's hierarchy. Mounting will make files, directories and devices available to the users.

It mounts the external storage devices like hard disks, pen drives, USBs etc.

Conversely, umount command unmount the mount point and detach the device from the system.

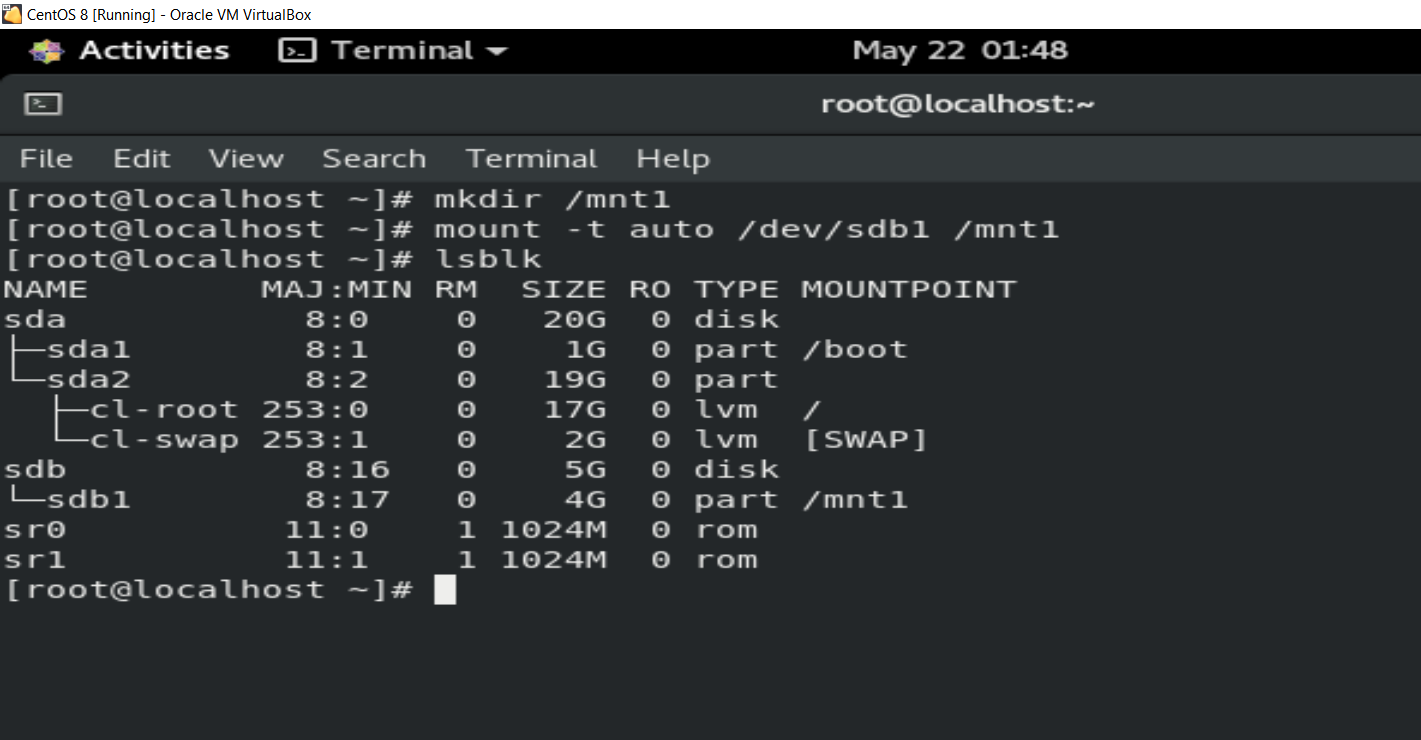


Fig 5.4 Mounting

Verify if the partition is mounted using the following command:

# lsblk -l or # fdisk -l

CHAPTER 6

USERS AND GROUPS

6.1 INTRODUCTION TO USER AND GROUPS

While users can be either people (meaning accounts tied to physical users) or accounts that exist for specific applications to use, groups are logical expressions of organization, tying users together for a common purpose. Users within a group share the same permissions to read, write, or execute files owned by that group.

Each user is associated with a unique numerical identification number called a User ID (**UID**). Likewise, each group is associated with a Group ID (**GID**). A user who creates a file is also the owner and group owner of that file. The file is assigned separate read, write, and execute permissions for the owner, the group, and everyone else. The file owner can be changed only by root, and access permissions can be changed by both the root user and file owner.

### 6.1.1 USER PRIVATE GROUPS

Red Hat Enterprise Linux uses a user private group (UPG) scheme, which makes UNIX groups easier to manage. A user private group is created whenever a new user is added to the system. It has the same name as the user for which it was created and that user is the only member of the user private group.

User private groups make it safe to set default permissions for a newly created file or directory, allowing both the user and the group of that user to make modifications to the file or directory.

The setting which determines what permissions are applied to a newly created file or directory is called a umask and is configured in the /etc/bashrc file. Traditionally on UNIX-based systems, the umask is set to 022, which allows only the user who created the file or directory to make modifications. Under this scheme, all other users, including members of the

creator’s group, are not allowed to make any modifications. However, under the UPG scheme, this "group protection" is not necessary since every user has their own private group.

### 6.1.2 SHADOW PASSWORDS

In environments with multiple users, it is very important to use shadow passwords provided by the shadow-utils package to enhance the security of system authentication files. For this reason, the installation program enables shadow passwords by default.

The following is a list of the advantages shadow passwords have over the traditional way of storing passwords on UNIX-based systems:

* Shadow passwords improve system security by moving encrypted password hashes from the world-readable /etc/passwd file to /etc/shadow, which is readable only by the root user.
* Shadow passwords store information about password aging.
* Shadow passwords allow to enforce some of the security policies set in the /etc/login.defs file.

### 6.1.3 ADDING A NEW USER AND NEW GROUP

To add a new user to the system, type the following at a shell prompt as root:

useradd options username

By default, the useradd command creates a locked user account. To unlock the account, run the following command as root to assign a password:

passwd username

To add a new group to the system, type the following at a shell prompt as root:

groupadd options group\_name

**Table 6.1 Common useradd command-line options**

|  |  |
| --- | --- |
| -c ‘comment’ | comment can be replaced with any string. This option is generally used to specify the full name of a user. |
| -d home\_directory | Home directory to be used instead of default /home/username. |
| -e date | Date for the account to be disabled in the format YYYY-MM-DD. |
| -g group\_name | Group name or group number for the user’s default (primary) group. The group must exist prior to being specified here. |
| -G group\_list | List of additional (supplementary, other than default) group names or group numbers, separated by commas, of which the user is a member. The groups must exist prior to being specified here. |
| -N | Do not create a user private group for the user. |
| -u | User ID for the user, which must be unique and greater than 999. |

6.1.4 SETTING USER AND GROUP OWNERSHIP OF FILES AND ACCESS PERMISSIONS

The Linux files access is controlled. There are three types of access (permissions):

* read
* write
* execute

To view the permissions for files and directories, use the ls -l or ls – n commands.

Permission Set:

* Each permission group has three permissions, called a permission set.
* Each set consists of read, write, and execute permissions.
* Each file or directory has three permission sets for the three types of permission groups.
* The first permission set represents the owner permissions, the second set represents the group permissions, and the last set represents the other permissions.
* The read, write, and execute permissions are represented by the characters r, w, and x, respectively.
* The presence of any of these characters, such as r, indicates that the particular permission is granted.
* A dash (–) symbol in place of a character in a permission set indicates that a particular permission is denied.
* Linux assigns initial permissions automatically when a new file or directory is created.

chmod Command

The chmod command is used to alter the permissions of a file. It may be used to add or remove permissions symbolically. For example, to add execute permissions for the owner of a file you would run: $ chmod u+x filename

Or, to add read and write permissions for the group that owns the file, you would run:

$ chmod g+rw filename

Instead of adding permissions, the symbolic syntax of chmod can also be used to subtract or set to some absolute value as shown in these examples:

$ chmod o-w filename

$ chmod u=rwx, g=rx, o= filename

chown Command

Every file is owned by a specific user (or UID) and a specific group (or GID). The chown command can be used to change just the user, or the user and group of a file. Here is an example of changing the owner of file test to user and its group to user.

$ ls -l test

-rw-r—r— 1 root root 0 Nov 20 00:43 test

$ chown user:user test

$ ls -l test -rw-r—r— 1 user user 0 Nov 20 00:43 test

The basic format for the chown command is as follows:

$ chown user: group filename

A period (.) can be used in place of the colon (:) separator character. Also, either the user or group name can be omitted. If the username is omitted (but the separator character is present), then the chown command behaves like the chgrp command, and only the group ownership is changed. If the group name is omitted but the separator character is present, then the group will be set to the login group of the specified user. If both the group name and the separator character are omitted, then only the user name is changed. For example, to change only the owner you could run the following: $ chown user filename

CHAPTER 7

LINUX FILE SYSTEM

In Linux system, all data and programs are stored in file. There is not any restriction on the structure of a file. It is a collection of related data. The size of a Linux file is the exact size of the contents that stored inside the file. That means it does not need its own size for storage. A Linux file or a file system is said to be the heart of it. The files in a file system are connected logically.

A file system manages how the files are stored and can be retrieved easily. It is not possible for an operating system to know the positioning of a file i.e., where a file starts and where ends, without a file system. It is a method through which operating system keeps track of how files are organized in a disk partition so that it can be found easily.

There are different file systems present for Linux OS. They are different from each other based on the structure, logic and other important properties like flexibility, speed, size etc. We will study some of the file systems that are being commonly used for better understanding of the functionalities of a file system. File System Types: Any Linux distribution has different types of file system choices to partition a disk. Some of them are given below:

* EXT
* EXT2
* EXT3
* EXT4
* XFS
* IFS
* REISER FS
* ISO9660 etc.

7.1 FILE ORGANIZATION

7.1.1 ABSOLUTE PATHNAMES

Each directory and file in Linux is indicated by its pathname beginning from the root directory. i.e., the pathname specifies the location of a file or directory where it is found. Absolute pathname is the complete path of a directory or file starting from the root. For example, /usr/abc is the absolute pathname of the directory abc. The figure 6.1 shows it. The first forward slash(/) indicates the root directory. Thus, the pathname /usr/abc means that the directory abc is found under the usr directory which is in turn found under the root directory. Similarly, the absolute pathname of the directory abc3 is /usr/abc/abc3. To move directly to a particularly directory or file you must have to use the absolute pathname. The following commands show the use of absolute pathname.

7.1.2 RELATIVE PATHNAMES

Relative pathname represents the location of a directory or file with respect to the current working directory. For example, if your current directory is /usr/abc, then you can move to a directory say abc3 under the abc directory by using only the directory name abc3 with cd command. It is performed in command line as–

$ pwd

/usr/abc

$ cd abc3

$ pwd

/usr/abc/abc3

Here, in the command line ‘‘cd abc3’’, abc3 has been used as a relative pathname. It can be understood using the figure 6.1. Similarly, if the current directory is /usr/abc/abc3, then by using the relative pathname of a directory (i.e., only the directory name) resides under the directory abc3 to move to that particular directory.

7.1.3 FILE SYSTEM MOUNTING AND UNMOUNTING

In Linux, once a file system is created, it is logically separated with its own tree structure and root directory. But during the time of booting, all these file systems are required to be united to become a single file system. Then the root file system become the main file system to which all the secondary file systems are attached. File system mounting is the process of attaching all the secondary file systems (i.e., the file systems other than the root or main file system) to the main file system at different points. Here, the secondary files attach themselves to the file system. File system mounting is done by using the mount command. The file systems are mounted during the system start up.

File system unmounting is the process of detaching the secondary file systems from the main file system. The file systems are unmounted when the system is shut down. File system unmounting is not possible if there is a file open in it.

7.2 STANDARD DIRECTORIES AND THEIR CONTENTS

The standard directories are located directly under the root directory. These directories are essential to the startup and continuous operation of the system. The following are the standard directories of the Linux file system.

**/bin:** It contains the programs needed for using and managing the system. The commands found in this directory are date (displays today’s date), ls (lists the contents of a directory), and cp (makes a copy of a file).

**/boot:** It contains the static files of the boot loader.

**/dev:** It contains system device files. A device file provides an interface to a particular device. Examples of devices having device files in /dev are disk drives, tape drives, or CDROM drives. **/etc:** It contains system specific configuration files, and files essential for system startup.

**/home:** This is the directory where the home directories for all users of the system are stored. **/lib:** Shared libraries and kernel modules are under this directory.

**/mnt:** This is the directory where temporary file systems are mounted. It may contain subdirectories like cdrom, floppy, and disk.

**/opt:** It contains software files that are not installed when the operating system is installed. This directory usually contains products provided by third-party software vendors.

**/sbin:** Programs for administering a system are located in this directory. The commands fdisk (used to partition a disk), fsck (used to check the integrity of a file system), and shutdown (used for stopping a system) are found under this directory.

**/tmp:** This directory is used to hold temporary files. This directory is generally referred to as a scratch directory, and can be used by all system users.

**/usr:** It contains programs and files related to the users of a system. The data in /usr is typically read-only, and may be shared with other computer systems on a network.

**/var:** Files with varying content are stored in this directory. It includes system log files, mail system files, and print spooling system files.

**/srv:** It contains data for services provided by the system.

CHAPTER 8

CONTAINERS IN LINUX

Linux Containers have emerged as a key opensource application packaging and delivery technology, combining lightweight application isolation with the flexibility of image-based deployment methods. Red Hat Enterprise Linux 7 implements Linux Containers using core technologies such as Control Groups (Cgroups) for Resource Management, Namespaces for Process Isolation, SELinux for Security, enabling secure multi-tenancy and reducing the potential for security exploits. All this is meant to provide you with an environment to producing and running enterprise-quality containers.

## 8.1 LINUX CONTAINERS ARCHITECTURE

Several components are needed for Linux Containers to function correctly, most of them are provided by the Linux kernel. Kernel namespaces ensure process isolation and cgroups are employed to control the system resources. SELinux is used to assure separation between the host and the container and also between the individual containers. Management interface forms a higher layer that interacts with the aforementioned kernel components and provides tools for construction and management of containers.

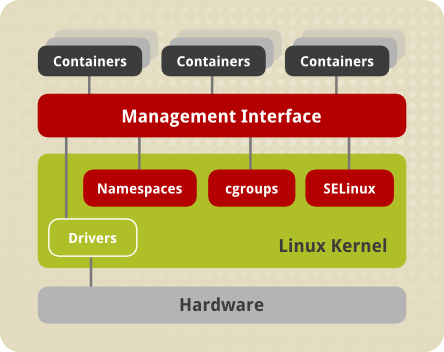


Fig 8.1 architecture of Linux Containers

**Namespaces**

The kernel provides process isolation by creating separate namespaces for containers. Namespaces enable creating an abstraction of a particular global system resource and make it appear as a separated instance to processes within a namespace. Consequently, several containers can use the same resource simultaneously without creating a conflict. There are several types of namespaces:

* **Mount namespaces** isolate the set of file system mount points seen by a group of processes so that processes in different mount namespaces can have different views of the file system hierarchy. With mount namespaces, the mount() and umount() system calls cease to operate on a global set of mount points (visible to all processes) and instead perform operations that affect just the mount namespace associated with the container process. For example, each container can have its own /tmp or /var directory or even have an entirely different userspace.
* **UTS namespaces** isolate two system identifiers – nodename and domain name, returned by the uname() system call. This allows each container to have its own hostname and NIS domain name, which is useful for initialization and configuration scripts based on these names. You can test this isolation by executing the hostname command on the host system and a container – the results will differ.
* **IPC namespaces** isolate certain interprocess communication (IPC) resources, such as System V IPC objects and POSIX message queues. This means that two containers can create shared memory segments and semaphores with the same name, but are not able to interact with other containers memory segments or shared memory.
* **Network namespaces** provide isolation of network controllers, system resources associated with networking, firewall and routing tables. This allows container to use separate virtual network stack, loopback device and process space. You can add virtual or real devices to the container, assign them their own IP Addresses and even full iptables rules. You can view the different network settings by executing the ip addr command on the host and inside the container.
* **PID namespaces** allow processes in different containers to have the same PID, so each container can have its own init (PID1) process that manages various system initialization tasks as well as containers life cycle. Also, each container has its unique /proc directory. Note that from within the container you can monitor only processes running inside this container. In other words, the container is only aware of its native processes and cannot "see" the processes running in different parts of the system. On the other hand, the host operating system is aware of processes running inside of the container, but assigns them different PID numbers. For example, run the ps -eZ | grep systemd$ command on host to see all instances of systemd including those running inside of containers.

**Control Groups (cgroups)**

The kernel uses cgroups to group processes for the purpose of system resource management. Cgroups allocate CPU time, system memory, network bandwidth, or combinations of these among user-defined groups of tasks. In Red Hat Enterprise Linux 7, cgroups are managed with systemd slice, scope, and service units.

**SELinux**

SELinux provides secure separation of containers by applying SELinux policy and labels. It integrates with virtual devices by using the sVirt technology.

CONCLUSION

This internship helped me to acquire the following skills,

* Understand and use essential tools for handling files, directories, command-line environments, and documentation
* Configure local storage using partitions and logical volumes
* Create and configure file systems and file system attributes, such as permissions, encryption, access control lists, and network file systems
* Manage users and groups, including use of a centralized directory for authentication
* Manage security, including basic firewall and SELinux configuration

REFERENCES

[1]: <https://developers.redhat.com/products/developertoolset/overview/>

[2]: https://www.redhat.com/en

[3]: https://developers.redhat.com/products/softwarecollections/overview