UNIT 4

SYSTEM ADMINISTRATION

Linux was intended for use by more than one person at a time. Multiuser features enable many people to have accounts on a single Linux system, with their data kept secure from others. Multitasking enables many people to run many programs on the computer at the same time, with each person able to run more than one program. The person assigned to manage all of a Linux system's resources is called the system administrator.

There are mainly three types of users in Linux: root user, regular user, and system user

Root user: This is also called super user and would have complete and unfettered control of the system. A super user can run any commands without any restriction. This user should be assumed as a system administrator. The default prompt for the root user is a pound sign (also called a hash mark):#.

Regular user: Regular users have the necessary privileges to perform standard tasks on a Linux computer such as running word processors, databases, and Web browsers. They can store files in their own home directories. Since regular users do not normally have administrative privileges, they cannot accidentally delete critical operating system configuration files. The default prompt for a regular user is simply a dollar sign: **$.**

System user: System accounts are those needed for the operation of system-specific components for example mail accounts. These accounts are usually needed for some specific function on your system, and any

modifications to them could adversely affect the system. (Software / application users created automatically by the system).

If you are the system administrator of a Linux system, you generally log in as a regular user account and then ask for administrative privileges when you need them. This is often done with one of the following:

- **su** command — Often *su* is used to open a shell as root user. Once open, the administrator can run multiple commands and then exit to return to a shell as a regular user.
- **sudo** command — With *sudo*, a regular user is given root privileges, but only when that user runs the *sudo* command. After running that one command with *sudo*, the user is immediately returned to a shell, and will be acting as the regular user again.
- **Graphical windows** — Many graphical administration windows, which can be launched from the System or Applications menu, can be started by a regular user. When root privilege is needed, you are prompted for the root password.

The following is a list of common features that a system administrator is expected to manage:

- File systems — When you first install Linux, the directory structure is set up to make the system usable. However, if you later want to add extra storage or change the file system layout, you need administrative privileges to do that. Also, the root user has permission to access files owned by any user. As a result, the root user can copy, move, or change any other user's files
- Software installation — Because malicious software can harm your system or make it insecure, you need root privilege to install software so it is available to all users on your system. Regular users can still install some software in their own directories and can list information about installed system software.

- User accounts — Only the root user can add and remove user accounts and group accounts.
- Network interfaces — It used to be totally up to the root user to configure network interfaces as well as to start and stop those interfaces. Now, many Linux desktops allow regular users to start and stop network interfaces from their desktop using Network Manager.
- Servers — Configuring web servers, fileservers, domain name servers, mail servers, and dozens of other servers require root privilege, as does starting and stopping those services. Often, services run as non-root users, and content, such as web pages, can be added to servers by non-root users if you configure your system to allow that.
- Security features — Setting up security features, such as firewalls and user access lists, is usually done by the root user. It's also up to the root user to monitor how the services are being used and make sure that server resources are not exhausted or abused.

Common Administrative Tasks

The Common Administrative Tasks can be classified as:

System automation which includes: the free disk space checking and reporting periodic backups, data for the system performance, user account maintenance (creation, deletion etc), Business specific functions (pushing data to a web server, running monthly/quarterly/yearly report, etc.)

Documentation: A good system administrator should document Policies, Procedures, and Changes.

Communication: A good system administrator should be a good communicator too. All the users should be aware of at least three points about a system administrator: what he is going to do, what he is doing, what he has done.

Identifying Administrative Files Configuration and Log files

Linux keep track of itself. Linux files are grouped according to their functionality and usage, so there are separate directories for configuration and log files. It will keep the log file to store the data of logins and the root user can monitor the system to see if people are trying to access the computer illegally. These files are dealing with system admin. The functions of administrative files are the following:

- Setting the Run Level
- System Services
- User Management
- Network Settings
- Scheduling Jobs
- Quota Management
- Backup and Restore
- Adding and Removing software/packages
- Setting a Printer
- Monitoring the system (general, logs)
- Monitoring any specific services running. Eg. DNS, DHCP, Web, NIS, NPT, Proxy etc.

Configuration Files

Configuration files on a Linux system that control user permissions, system applications, daemons, services, and other administrative tasks in a multi-user, multi-tasking environment. These tasks include managing user accounts, allocating disk quotas, managing e-mails and newsgroups, and configuring kernel parameters. In Linux each programme is free to choose the configuration file format.

Almost everything you set up for your particular computer — user accounts, network addresses, or GUI preferences— is stored in *plaintext* files (configuration file). This has some advantages and some disadvantages. The advantage of plaintext files is that it's easy to read and change them.

Any text editor will do. The downside, however, is that as you edit configuration files, no error checking is going on. You have to run the program that reads these files to find out whether you set up the files correctly. A comma or a quote in the wrong place can sometimes cause an entire interface to fail.

The two major locations of configuration files are your **home directory** (where your personal configuration files are kept) and the **/etc** directory (which holds system-wide configuration files).The following are some interesting configuration files in */etc*:

- **/etc/passwd** — Stores account information for all valid users for the system. Also includes other information, such as the home directory and default shell.
- **/etc/printcap**— Contains definitions for the printers configured for your computer.
- **/etc/shadow** — Contains encrypted passwords for users who are defined in the *passwd* file.
- **/etc/shells** — Lists the shells that are available on the system, as well as their locations.
- **/etc/group** — Identifies group names and group IDs (GIDs) that are defined on the system.
- **/etc/gshadow** — Contains shadow passwords for groups.
- **/etc/hosts** — Contains IP addresses and host names that you can reach from your computer.
- **/etc/inittab** — Contains information that defines which programs start and stop when Linux boots, shuts down, or goes into different states in between. This configuration file is the first one read when Linux starts the *init* process.
- **/etc/login.defs** - Configuration file for the login command.
- **/etc/skel**- Contains files and directories that are automatically copied over to a new user's home directory when new user is created.
- **/etc/default/useradd** - Contains default values for adding mew users.

Log Files

Log files are files that contain **messages about the system**, including the kernel, services, and applications running on it. There are different log files for different information. For example, there is a default system log file, a log file just for security messages, and a log file for *cron* tasks. (*Cron* allows users to run commands or scripts at a given date and time).

Log files can be very useful when trying to troubleshoot a problem with the system such as trying to load a kernel driver or when looking for unauthorized login attempts to the system. So we can use log files to help tracking the different problems. Most log files are located in the **/var/log/** directory. The following are some interesting log files in */var/log:*

- **/var/log/messages**: Contains many general informational messages about the system.
- **/var/log/secure**: All user authentication events are logged here. It stores all security related messages including authentication failures.
- **/var/log/boot.log**: This is the repository of booting related information and messages logged during system startup process.
- **/var/log/kern.log**: It contains information logged by the kernel.
- **/var/log/faillog**:This file contains information on failed login attempts.
- **/var/log/cron**:This log file records information on cron jobs. Whenever a cron job runs, this log file records all relevant information including successful execution and error messages in case of failures.
- **/var/log/maillog**: All mail servers related logs are stored here. Track all the emails that were sent or received during a particular period.

Role of system administrator

The system administrator is responsible for ensuring that the Linux system provides a reasonable service to its users. This involves a variety of activities; following are the most important ones:

- Adding new users to the system and configuring their home directories and basic privileges.
- Installing any new software, including applications, new versions of the operating system, or bug fixes.
- Monitoring the usage of the file system, ensuring that no one is using too much disk space and that all backups are carried out properly.
- Responding to problems from users, attempting to track down bugs, and meeting with suppliers as appropriate.
- Installing new hardware components.
- Ensuring the smooth operation of any network services, such as electronic mail or remote access to other machines.

Managing user accounts-adding & deleting users

Adding and managing users are common tasks for Linux systems administrator. User accounts keep boundaries between the people who use your systems and between the processes that run on your systems. Groups are a way of assigning rights to your system that can be assigned to multiple users at once.

Users who want to log in to a Linux computer must have an existing user account, which consists of properties that allow a user to access files and folders stored on the computer. This account information can be created and stored on the computer itself or on another computer on the network. Accounts stored on the computer are called *local user accounts*. Accounts stored in eDirectory are called *eDirectory user accounts*.

Adding users - *useradd*

The command to add new user is *useradd*. The synatx is:

useradd [options] <username>

If the system administrator uses *useradd* command for creating a user, then he must set an initial password for the user using *passwd* command.

passwd [options] <username>

The new user's information is stored in the */etc/passwd* file. Also inserts necessary entries in /etc/*shadow* and */etc/group files.*

Options used with useradd command

Options	Meaning
-c "comment"	Provide a description of the new user account. Typically, this will be the person's full name. Replace comment with the name of the user account
-d home_dir	Set the home directory to use for the account. The default is to name is the same as the login name and to place it in /home. Replace home_dir with the directory name to use
-D	Rather than create a new account, save the supplied information as the new default settings for any new accounts that are created.
-e expire_date	Assign the expiration date for the account in YYYY-MM-DD format. Replace expire_date with a date you want to use.
-f days	Set the number of days after a password expires until the account is permanently disabled. The default, -1, disables the option. Setting this to 0 disables the account immediately after the password has expired. Replace -1 (that's minus one) with the number to use.
-g group	Set the primary group the new user will be in. Replace group with the group name. Without this option, a new group is created that is the same as the user name and is used as that user's primary group.
-G grouplist	Set the list of groups that user belongs to.
-m	Automatically create the user's home directory and copy the files in theskeleton directory (/etc/skel) to it.
-M	Do not create the new user's home directory, even if the default behaviour is set to create it.
-n	A group having the same name as the user being added to the system will be created by default.

-o	Allow the creation of a user account with a duplicate (non-unique) UID.
-p passwd	Enter a password for the account you are adding. This must be an encrypted password.
-s shell	Specify the command shell to use for this account
-u user_id	Specify the user ID number for the account

Options used with passwd command

Options	Meaning
-l	Lock the password of specified user and is available to root only.
-u	Unlock the user password

Examples:

```
useradd Torvalds
```

Add new user Torvalds.

```
useradd –c "Linus Torvalds" Torvalds
```

Here a new user Linus Torvalds with login name Torvalds is created.

```
passwd Torvalds
Changing password for user Torvalds
New password:******
Retype new password : ******
```

/etc.passwd file contains new user entry as follows:

```
Torvals:x: 1001:1001:Linus Torvalds:/home/Torvalds:/bin/bash
```

The above entry contains a set of seven colon-separated fields; each field has its own meaning. The fields are User name, Password(x) stored in */etc/shadow* file, User ID, Group ID, User Info, Home directory, Shell.

In creating the account for Torvalds, the *useradd* command performs several actions:

- Reads the */etc/login.defs* and */etc/default/useradd* files to get default values to use when creating accounts.
- Checks command-line parameters to find out which default values to override.
- Creates a new user entry in the */etc/passwd* and */etc/shadow* files based on the default values and command-line parameters.
- Creates any new group entries in the */etc/group* file.
- Creates a home directory, based on the user's name, in the */home* directory.
- Copies any files located within the */etc/skel* directory to the new home directory. This usually includes login and application start up scripts.

Example2:

```
useradd -m -g users -G Linux,Unix -s /bin/tcsh -c "Linus Torvalds" Torvalds
```

The *useradd* command is told to create a home directory for *Torvalds* (-m), make users theprimary group *Torvalds* belongs to (-g), add to the groups *Linux* and *Unix* and assign *tcsh* as primary command shell (-s).

```
[salmiya@localhost ~]$ sudo useradd -c "Linus Torvalds" Torvalds
[salmiya@localhost ~]$ sudo passwd Torvalds
Changing password for user Torvalds.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[salmiya@localhost ~]$
```

Modifying users with *usermod* command

After creating user accounts, in some scenarios where we need to change the attributes of an existing user such as, change user's home directory, login name, login shell, password expiry date, etc, where in such case '*usermod*' command is used. Syntax is as follows:

usermod [options] <username>

Options	Meaning
-c "comment"	Provide a description of the new user account. Typically, this will be the person's full name. Replace comment with the name of the user account
-d home_dir	Set the home directory to use for the account. The default is to name it the same as the login name and to place it in /home. Replace home_dir with the directory name to use
-e expire_date	Assign the expiration date for the account in YYYY-MM-DD format. Replace expire_date with a date you want to use.
-f days	Set the number of days after a password expires until the account is permanently disabled. The default, -1, disables the option. Setting this to 0 disables the account immediately after the password has expired. Replace -1 (that's minus one) with the number to use.
-g group	Set the primary group the new user will be in. Replace group with the group name. Without this option, a new group is created that is the same as the user name and is used as that user's primary group.
-G grouplist	Set the list of groups that user belongs to.
-m	moving the contents of the home directory from existing home dir to new dir.
-n	A group having the same name as the user being added to the system will be created by default.
-p passwd	Enter a password for the account you are adding. This must be an encrypted password.
-s shell	Specify the command shell to use for this account
-u user_id	Specify the user ID number for the account
-l new-name	change login name of account to the name supplied after –l option.

Example:

usermod -s /bin/csh Torvalds

Deleting users- *userdel*

The *userdel* command is used to remove a user from the system, which removes all the entries pertaining to the specified user from the three files- */etc/passwd*, */etc/shadow*, and */etc/group*. General format is:

userdel [options] <username>

This command only removes the user account, but the user's files remains as they are, not deleted. The system administrator can delete these files separately if required.

Options:

Options	Meaning
-f	This option forces the removal of the user, even if she is still logged in. It also forces *userdel* to remove the user's home directory or her mail spool, even if another user uses the same home directory or if the mail spool is not owned by the specified user.
-r	Files in the user's home directory will be removed along with the home directory itself and the user's mail spool. Files located in other file systems will have to be searched for and deleted manually.

Files owned by the deleted user but not located in the user's home directory will not be deleted. The system administrator must search for and delete those files manually using the *find* command.

Examples:

```
userdel Torvalds
```

This command deletes user account with login name Torvalds

```
userdel –r Torvalds
```

This command wipes out Torvald's home directory along with his account.

Changing Permissions and Ownerships

Every file and directory on your Linux system is assigned 3 types of owner, given below.

1. **User** : A user is the owner of the file. i.e, the person who created a file.
2. **Group**: A user- group can contain multiple users.
3. **Other** : Any other user who has access to a file. When you set the permission for others, it is also referred as set permissions for the world.

Every file and directory in Linux system has following 3 permissions for user, group and other.

- **Read** : This permission give you the authority to **open and read a file**.
- **Write** : The right permission gives you the authority to **modify the contents of a file**. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory.
- **Execute** : Allows you to **execute the file**.

Permissions associated with files and directories in Linux were designed to keep users from accessing other users' private files and to protect important system files. The **9 bits** assigned to each file for permissions define the access that you and others have to your file. Permission bits for a regular file appear as **-rwxrwxrwx**. Those bits are used to define who can read, write, or execute the file. Of the nine-bit permissions, the first three bits apply to the owner's permission, the nextthree apply to the group assigned to the file, and the last three apply to all others. The **'r'** stands for *read*, the **'w'** stands for *write*, and the **'x'** stands for *execute* permissions. If a dash (-) appears instead of the letter, it means that permission is turned off for that associated read, write, or execute bit.

In order to change the permissions of the file, you need to use the **chmod** command. One thing to keep in mind is that you must be the owner of the file or the super user in order to change its permissions. General format is:

chmod <permission> <filename>

The *chmod* utility support two modes for modifying permissions:

- Symbolic Mode
- Absolute(Numeric) Mode

1. Symbolic Mode

The symbolic mode uses **letters and some operators** to set the permission. When using the symbolic mode the *chmod* command has the following syntax:

chmod [u g o a][+ - =]<permissions><filename>

Each letter between the first set of brackets specifies to whom to apply the permissions and have the following meanings:

- **u** : specifies the user who owns the file (the first set of permissions).
- **g** : specifies the group which owns the file (the second set of permissions).
- **o** : specifies the other users which are not members of the group or owners of the file(the third set of permissions).
- **a**: specifies all users and available on the system (all three sets of permissions).

The next 3 values between the second brackets are used as operators to manipulate permissions. They have the following meanings:

- + : add a permission to the file
- - : removes the selected permissions from the file.
- = : overwrite the existing permissions of the file and add a new one.

Examples:

1. Let's assume we have a file named *sample.sh* in our current folder with the default permissions (rw-r--r--) and we want to mark it as executable:

```
chmod a+x sample.sh
```

This will make the script executable for everyone (a). Now the permission bits will look like this (rwxr-xr-x).

2. To restrict the execute permission to the owner only run the following command:

```
chmod u+x sample.sh
```

After this the permissions will look like this (rwxr--r--).

3. Another way to achieve the same result is by removing the executable bit from the group and others

```
chmod go-x sample.sh
```

4. To simply duplicate the permission from the owner of the file to the group we can simply run something like this:

```
chmod g=u ssample.sh
```

Now the permissions will look like this (rwxrwx---). As you can see by using the '=' operator the read permission from the other users have been removed

5. To assign read and write permissions for everybody:

```
chmod ugo+rw sample.sh
```

The permission will look like this (rw-rw-rw-).

```
[salmiya@localhost ~]$ chmod -v u+x sample.sh
mode of 'sample.sh' changed from 0644 (rw-r--r--) to 0744 (rwxr--r--)
[salmiya@localhost ~]$
[salmiya@localhost ~]$ chmod -v g=u sample.sh
mode of 'sample.sh' changed from 0744 (rwxr--r--) to 0774 (rwxrwxr--)
[salmiya@localhost ~]$
[salmiya@localhost ~]$ chmod -v ugo-wx sample.sh
mode of 'sample.sh' changed from 0774 (rwxrwxr--) to 0444 (r--r--r--)
[salmiya@localhost ~]$
[salmiya@localhost ~]$ chmod -v ugo+rw sample.sh
mode of 'sample.sh' changed from 0444 (r--r--r--) to 0666 (rw-rw-rw-)
[salmiya@localhost ~]$
[salmiya@localhost ~]$ chmod -v a+x sample.sh
mode of 'sample.sh' changed from 0666 (rw-rw-rw-) to 0777 (rwxrwxrwx)
[salmiya@localhost ~]$
[salmiya@localhost ~]$
```

2. Absolute Mode

The absolute mode uses **numerical values (0-7)** to assign permission. Using the absolute syntax, every group of 3 bits can have a value between 0 and 7. Each permission (read, write, and execute) is assigned a number — r=4, w=2, and x=1 no permission=0 — and you use each set's total number to establish the permission. The next table displays the eight possible combinations for each set of three bits:

Number	Permission type	Permission
0	No permission	---
1	Execute	--x
2	Wrie	-w-
3	Write+execute	-wx
4	Read	r--
5	Read+execute	r-x
6	Read+write	rw-
7	Read+write+execute	rwx

For example the "rwx" permission is calculated by adding the individual values for each bit, i.e,4+2+1 =7.

Examples:

```
chmod 644 scriptfile.sh
```

After running this the permission bits will have these values (rw-r--r--).

1. To set read and write access for all users use:

```
chmod 666 scriptfile.sh
```

The permission bits will look like this (rw-rw-rw-).

2. To set full permissions to the owner of the file:

```
chmod 700 scriptfile.sh
```

The permission bits will look like this (-rwx------).

By running the *chmod* utility without any parameter you apply the changes to only a single file or directory. If you wish to apply changes recursively to multiple file and directories you must use the *chmod* command with the **'-R'** option.

```
chmod -R 755 /HOME/mydirectory
```

In this example the permissions will be applied recursively to all file and folders below the *mydirectory* directory in your home directory. You can also pass the '**-v**' parameter to display the changes made on the screen.

```
[salmiya@localhost ~]$ chmod -v 644 sample.sh
mode of 'sample.sh' changed from 0666 (rw-rw-rw-) to 0644 (rw-r--r--)
[salmiya@localhost ~]$
[salmiya@localhost ~]$ chmod -v 666 sample.sh
mode of 'sample.sh' changed from 0644 (rw-r--r--) to 0666 (rw-rw-rw-)
[salmiya@localhost ~]$
[salmiya@localhost ~]$ chmod -v 700 sample.sh
mode of 'sample.sh' changed from 0666 (rw-rw-rw-) to 0700 (rwx------)
[salmiya@localhost ~]$
[salmiya@localhost ~]$
```

Managing ownership- *chown*

Ownership refers to the user and group to which owns a file or directory. The **chown** command is used to change the user and group ownership of a

given file. Remember that you must be the owner of the file or the super user in order to change its owners. The syntax for the *chown* command is:

chown [options] [user] [:group] filename

Here, *user* is the user name or the user ID of the new owner, *group* is the name of the group or the group ID (GID).

Options	Meaning
-c	Reports when a file ownership is changed.
-f	Do not print error messages about files whose ownership cannot be changed.
-R	Recursively change ownership of directories and their contents.
-v	It is used to show the verbose information for every file processed.

Examples:

```
chown Torvalds sample.sh
```

In this example the owner of the file is the username 'Torvalds'.

```
chown Torvalds : group1 sample.txt
```

In this example the owner of the file is the username 'Torvalds' and the group is 'group1'.

To change only the group use:

```
chown :group1 sample.sh
```

In this example only the group is specified preceded by a colon ':'.

Creating and managing groups

Group accounts are useful if you want to share a set of files with multiple users. You can create a group and change the set of files to be associated with that group. The root user can assign users to that group so they can have access to files based on that group's permission.

Every user is assigned to a primary group. By default, that group is a new group with the same name as the user.

Creating group - *groupadd*

To create new groups use **groupadd** command. When you add a group to the system, the system places the group's name in the */etc/group* file and gives it a group ID number. This command creates the group category. Users are individually added to the group. The syntax is:

groupadd [options] groupname

Options	Meaning
-f	This option causes the command to simply exit with success status if the specified group already exists.
-g	used to provide a group id (numeric) to the new group, and it should be non-negative and unique
-o	This option permits to add a group with a non-unique GID.
-p	To set the encrypted password for the new group. The default is to disable the password.
-r	Create a system group.
-h	Displays help message and exit.

Examples:

```
groupadd newgroup
```

To create new group with specific groupid, use the following:

```
groupadd –g 9090 newgroup
```

Deleting group - *groupdel*

The **groupdel** command is used to delete an existing group. It will delete all entry that refers to the group, modifies the system account files, and it is handled by super user or root user. The syntax is:

groupdel <groupname>

Example:

```
groupdel newgroup
```

Modifying group- *groupmod*

The **groupmod** command in Linux is used to modify or change the existing group on Linux system. It can be handled by super user or root user. The syntax is:

groupmod [options]<groupname>

Options	Meaning
-n	The name of group will change into newname.
-g	To change the group ID
-o	To change the group GID to a non-unique value.
-p	To change the encrypted password

Example:

groupadd oldgroup	//Creates new group *oldgroup*
groupmod -n newgroup oldgroup	//Change the group name to *newgroup*
groupmod –g 9090 newgroup	//Change the group ID of *newgroup* to 9090

Temporary disabling of users accounts

Linux systems allow you to disable access to particular user account without changing anything from the account. This might be useful if you do not want to remove user account permanently but, you just want it disabled and no longer able to use the system. The disabled user will still receive emails for example, but he will not be able to login and check them out. A user account can be temporarily disabled or permanently removed. There are two methods to temporary disable user account:

1. Editing */etc/shadow* file
2. using *passwd* command

1. Editing /etc/shadow file

You can disable or lock a user account temporarily by just putting an asterisk "*" or "!" at the beginning of the second field in the file */etc/shadow*.

This means that "*" and "!" won't permit login for this account. Whenever you want to enable the account, just erase the asterisk and the user account is back in operation, with its old password.

For example you want to disable user "Torvalds" then you can do this as follows:

vi /etc/shadow

```
Torvalds:*$1$narMEFm6$fhA1puOU422HiSL5aggLI/
:11193:0:99999:7:-1:-1:134539228
```

Here, the second field is the encrypted password. You can prefix the password with "*" or "!". This will render user account inaccessible and it will mean that no login is permitted for that user.

2. Using passwd command

passwd command can be used to disable the user account. To lock the user use the following command.

```
passwd –l Torvalds
```

Above command changes the shadow file and adds "!!" in front of the user password.

```
Torvalds:!!$1$eFd7EIOg$EeCk6XgKktWSUgi2pGUpk.:13852:0:99999:7:::
```

Now in case if you want enable the account just unlock it using –u option as follows:

```
passwd -u Torvalds
```

You can also enable account by removing manually the "!" character from the user's password line in */etc/shadow.*

Creating and mounting file system

A file system is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk. The word is also used to refer to a partition or disk that is used to store the files or the type of the file system.

Before any partition or disk is used as file system, it is initialized first, and actual data structures need to be written to the disk which keeps track of all the records on the disk. This process is called *making a file system*.

A file system is either the device file associated with the partition or device or is the directory where the file system is mounted. The basic requirement to mount a partition or to use a partition is that the file system must first be installed on it. This is also a process of creating *inodes* and *data blocks*.

Creating a file system writes information to the device and creates order of the empty space. This file system–related data consumes a small percentage of the space. The remaining space on the disk drive is split into small, consistently sized segments called **blocks**. Linux supports a number of file system types, some of which are described as follows.

File system	Description
ext2	High performance for fixed disk and removable media
ext3	Journaling version of ext2
ext4	Supports larger files and file system sizes
Vfat	MS-DOS file system useful when sharing files between Windows and Linux
XFS	High-performance journaling file system
Btrfs	Addresses scalability requirements of large storage systems

Creating a file system

The command to build a Linux file system on a device, or hard disk partition, is **mkfs**. The syntax for the command is:

mkfs [options] device

There is actually a separate program for each file system type. The **mkfs** is just a front end that runs the appropriate program depending on the desired file system type. The most important options are summarized below.

Options	Meaning
-t fstype	Select the type of the filesystem.(ext2,ext3 ...) , the default file system type is ext2.
-c	Search for bad blocks and initialize the bad block list accordingly.
-V, —verbose	Produce verbose output, including all file system-specific commands that are executed. Specifying this option more than once inhibits execution of any file system-specific commands. This is really only useful for testing.
-V, —version	Display version information and exit. Option -V displays version information only when it is the only parameter, otherwise it will work as —verbose.

Example: To make a ext2 files system on a hard disk, use the following command:

```
mkfs -t ext2 /dev/sda0
```

The default file system type created when using the *mkfs* command is ext2. Following commands create an ext2 file system on the specified device. Here sda2 means second partition of the first HDD:

```
mkfs /dev/sda2
mke2fs /dev/sda2
mkfs.ext2 /dev/sda2
```

To create an ext3 file system, use any of the following commands:

```
mkfs –t ext3 /dev/sda2
mke2fs –t ext3 /dev/sda2
mkfs.ext3 /dev/sda2
```

To create an ext4 file system, use any of the following commands:

```
mkfs –t ext4 /dev/sda2
mke2fs –t ext4 /dev/sda2
mkfs.ext4 /dev/sda2
```

To see which supported file system types are installed in your system, use the **ls /sbin/mkfs*** command.

Mounting file systems

The meaning of Mounting is to attach the newly created file System to the currently accessible file System in Linux. On Linux OS, the directory structure begins with the root directory, which is the directory that contains all other directories and files on the system and is referred by a forward slash (/). The currently accessible file system is the file system that can be accessed at given time without special programming or settings.

File systems on different partitions and removable devices, such as CDs, DVDs, or USB flash drives, must be attached to the directory hierarchy to be accessed. To attach a partition or device, a **mount point** must be created. Mount point refers to **the empty directory in the currently available file system with which a newly created or additional file system is mounted**. The ***/mnt*** directory already exists in Linux. This directory with its subdirectories like */mnt/floppy, /mnt/usb* are intended specifically to be used as mount points for removable media such as CDROMS, USB key drives and floppy disks. So a mount point is simply a directory created with the *mkdir* command. After a directory, or mount point, is created, attach the partition by using the **mount** command. Syntax for the mount command is:

mount <device_file> <mount-point>

The mount command takes two arguments. The first one is the device file corresponding to the disk or partition containing the file system. The second one is the directory below which it will be mounted. When used without any argument, the mount command will display all currently attached file systems.

mount

Options	Meaning
-a	*Mount all filesystems (of the given types) mentioned in /etc/fstab.*
-l	Lists all the file systems mounted yet.
-V	Displays the version information.
-v	Verbose mode.
-t	Type of filesystem device uses.

The following example creates a mount point */test* and attaches the partition:

```
mkdir /test
mount /dev/sda2  /test
```

Unmounting File system

When a file system no longer needs to be mounted, it can be unmounted with **umount** command. The *umount* command takes one argument: either the device file or the mount point. Syntax is:

umount device-file/mount-point

For example, to unmount the directories of the previous example, one could use the commands

```
umount /dev/sda2
umount /test
```

An updated list of the devices that are mounted on the Linux can be seen at the */etc/fstab* file. This text file also displays the mount points and other information about the devices. It can be viewed by using the command:cat /etc/fstab.

Checking and monitoring system performance

If your Linux system is being used as a multiuser computer, sharing the processing power of that computer can be a major issue. Likewise, any time you can stop a runaway process or reduce the overhead of an unnecessary program running, your Linux server can do a better job serving files, Web pages, or e-mail to the people that rely on it.

Utilities are included with Linux that can help you monitor the performance of your Linux system. The kinds of features you want to monitor in Linux include CPU usage, memory usage (RAM and swap space), and overall load on the system. Different commands are available to monitor and find the actual causes of performance problem.

1. Top

Linux Top command is a performance monitoring program which is used frequently by many system administrators to monitor Linux performance and it is available under many Linux/Unix like operating systems. The top command used to display all the running and active real-time processes in ordered list and updates it regularly. It display CPU usage, Memory usage, Swap Memory, Cache Size, Buffer Size, Process PID, User, Commands and much more. It also shows high memory and CPU utilization of running processes. The top command is much useful for system administrator to monitor and take correct action when required.

2. VmStat

Linux VmStat command is used to display statistics of virtual memory, kernel, threads, disks, system processes, I/O blocks, interrupts, CPU activity and much more.

3. Iostat

IoStat is simple tool that will collect and show system input and output storage device statistics. This tool is often used to trace storage device performance issues.

4. free

Shows memory statistics for both main memory and swap.

5. Netstat

Netstat is a command line tool for monitoring incoming and outgoing network packets statistics as well as interface statistics. It is very useful tool for every system administrator to monitor network performance and troubleshoot network related problems.

6. Uptime

Uptime gives a one line display of the following information. The current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

7. **ps**

Reports a snapshot of the current process. The details include PID of the process, CPU utilization of the process, Percentage of memory in KB used by the process, Terminal (console) associated with the process, Current state of the process, Process start time or date, etc.

Becoming super user using su

Though the normal way to become the super user is to log in as root, sometimes that is not convenient. For example, you may be logged into a regular user account and just want to make a quick administrative change to your system without having to log out and log back in. In this situation, you can use **su** command.

From any Terminal window or shell, you can simply type: su

After you enter the **su** command (without arguments), the system prompts you for the root password. If you type the password correctly, the prompt for the regular user (**$**) will be changed to the super user prompt (#). At this point, you have successfully become super user and you have full permission to run any command and use any file on the system.

You may exit from the super user account with *exit* command or press Ctrl+D.

```
[salmiya@localhost ~]$
[salmiya@localhost ~]$
[salmiya@localhost ~]$ su
Password:
[root@localhost salmiya]#
[root@localhost salmiya]#
[root@localhost salmiya]# exit
exit
[salmiya@localhost ~]$
```

Getting system information with uname, host name

While using Linux, there might arises a need to know about the system you are on or the hardware specifications you are using. As a normal Linux user or a software developer, it is important for you to check the compatibility of a software or hardware system that you want to install. Linux command

line comes with multiple built-in commands for you to get familiar with the software and hardware platform you are working on.

uname command

The *uname* command is used to print **information about the current system**. Syntax is :

uname [options]

If *uname* is executed without option, it prints the kernel name. If no option is specified, by default *uname* assumes the **-s** option.

Options	Meaning
-a	It prints all the system information in the following order: Kernel name, network node hostname, kernel release date, kernel version, machine hardware name, hardware platform, operating system
-s	print the kernel name
-n	print the network node hostname
-k	print the kernel release date
-v	print the kernel version
-m	print the machine hardware name
-p	Print the type of the processor.
-i	print the platform of the hardware.
-o	print the name of the operating system.

hostname

A *hostname* command is used to display or set a computer's hostname or domain name. These names are used by many of the networking programs to identify the machine. Syntax is:

hostname [options]

If you want to change or set hostname of your Linux system, simply run:

```
hostname new-hostname
```

Options	Meaning
-a	Display the alias name of the host
-d	print the domain name
-i	Display the IP address
-s	Display the short host name. This is the host name cut at the first dot.

Disk Partitioning

Large storage devices are divided into separate sections called partitions. Partitioning also allows you to divide your hard drive into isolated sections, where each section behaves as its own hard drive. Partitioning is particularly useful if you run multiple operating systems.

If you have added a new disk to your system, you can simply format entire disk and create it as a single disk. But it's a good idea to create smaller partitions on large size disks.

The **fdisk** command that stands for *Format-disk* or *Fixed-disk* is basically used to create or delete hard disk partitions. The syntax is:

fdisk [options] <device-name>

Options	Meaning
-b sectorsize	Specify the sector size of the disk.
-C cyls	Specify the number of cylinders of the disk.
-H heads	Specify the number of heads of the disk.
-S sects	Specify the number of sectors per track of the disk.
-l	List the partition tables for the specified devices and then exit.
-s	Display the size of the partition (in blocks)

Example 1: To lists all the partitions on your system, use:

```
fdisk -l
```

Example 2:To list partitions on the specific device, use

```
fdisk -l /dev/sda
```

To work on a disk's partitions, you have to enter in command mode. It will display all commands which are available for *fdisk*. The following command enters command mode for the first disk device:

fdisk /dev/sda

```
[salmiya@localhost]$ fdisk /dev/sda
Welcome to fdisk
Changes will remain in memory only, until you decide to write them
sectors (command 'u').
Command (m for help):
```

Type '**m**' to see the list of all available commands of *fdisk* which can be operated on */dev/sda* hard disk. After pressing **m**, you will see the all available options for *fdisk* that you can be used on the */dev.*

```
[salmiya@localhost]$ fdisk /dev/sda
Welcome to fdisk
Changes will remain in memory only, until you decide to write them
    sectors (command 'u').
Command (m for help): m
Command action
  a  toggle a bootable flag
  b  edit bsd disklabel
  c  toggle the dos compatibility flag
  d  delete a partition
  l  list known partition types
  m  print this menu
  n  add a new partition
  o  create a new empty DOS partition table
  p  print the partition table
  q  quit without saving changes
```

```
s  create a new empty Sun disklabel
t  change a partition's system id
u  change display/entry units
v  verify the partition table
w  write table to disk and exit
x  extra functionality (experts only)
```

Command (m for help):

Command mode option	Meaning
m	To print list of commands
n	To create a new partition
d	To delete a partition
p	To print current partition table

Create a New Disk Partition:

To create a new partition, type '**n**' in command mode. While creating a new partition, it will ask you two options 'extended' or 'primary' partition creation. Press '**e**' for extended partition and '**p**' for primary partition. You will then be prompted to enter the first cylinder or sector number of the partition to be created. You may press *Enter* to accept the defaults, which is the first free sector on the disk. Then, specify the last sector number of the partition to be created. You may press *Enter* to use all available space after the first sector or enter a specific size such as +2G or +256M for a 2 gigabyte or 256 megabyte partition respectively.

After creating a new partition, you should run '**w**' command to alter and save new changes to partition table and finally reboot your system to verify newly created partition.

After the new partition is created, you need to format the newly created partition using 'mkfs' command.

mkfs.ext4 /dev/sda2

After formatting new partition, check the size of that partition using flag 's' (displays size in blocks) with *fdisk* command. This way you can check size of any specific device.

fdisk –s /dev/sda2

```
[salmiya@localhost]$ fdisk /dev/sda
Welcome to fdisk
Changes will remain in memory only, until you decide to write them
    sectors (command 'u').
Command (m for help): n
Partition type:
  p  primary partition (1-4)
  e  extended

Select (default p): p
Partition number (2-4): 2
First sector (71848- 1953525167, default 71848):
Last sector, +sectors or +size {K,M,G}(71848- 1953525167, default
1953525167):

Partition 2 of type Linux and of size 931.2GiB is created
Command m for help:
```

Delete a Hard Disk Partition:

To delete a partition for the hard disk and free up space occupied by that partition enter '**d**' in command mode. Then you will be prompted to enter partition number that you want to delete. Then run '**w**' command to alter and save new changes to partition table and reboot for changes. For example, if you want to delete the partition 4 from */dev/sda* hard disk, then just type 4. Then it will delete partition number '4' (i.e. /dev/sda4) disk and shows free space in partition table.

```
[salmiya@localhost]$ fdisk /dev/sda
Welcome to fdisk
Changes will remain in memory only, until you decide to write them
    sectors (command 'u').
Command (m for help): d
Partition number (1-4): 4
Partition 4 is deleted

Command (m for help): w
The partition table has been altered!
```

Installing and removing packages with rpm command

An RPM package is a consolidation of files needed to provide a feature, such as a word processor, a photo viewer, or a file server. Inside an RPM can be the commands, configuration files, and documentation that make up the software feature. However, an RPM file also contains metadata that stores information about the contents of that package, where the package came from, what it needs to run, and other information. Each software Package is actually an RPM package, consisting of an archive of software files and information about how to install those files. Each archive resides as a single file with a name that ends with *.rpm*, indicating it is a software package that can be installed by the *RedHat Package Manager*.

RPM (Red Hat Package Manager) is a default open source and most popular package management utility for Red Hat based systems like (RHEL, CentOS and Fedora). The tool allows system administrators and users **to install, update, uninstall, query, verify and manage system software packages** in Unix/Linux operating systems. The RPM formerly known as **.rpm file**, that includes compiled software programs and libraries needed by the packages. This utility only works with packages that built on *.rpm* format.

- RPM is free and released under GPL (General Public License).
- RPM keeps the information of all the installed packages under */var/lib/rpm* database.
- RPM is the only way to install packages under Linux systems.
- RPM deals with .rpm files, which contains the actual information about the packages such as: what it is, from where it comes, dependencies info, version info etc.

There are five basic modes for RPM command

1. **Install** : It is used to install any RPM package.
2. **Remove** : It is used to erase, remove or un-install any RPM package.
3. **Upgrade** : It is used to update the existing RPM package.
4. **Verify** : It is used to verify an RPM packages.
5. **Query** : It is used query any RPM package.

Syntax: ***rpm [options] <packagename>***

Options	Meaning
-i	Install the package
-U	update package, same as install but any previous version is removed.
-e	Remove the installed package
-q	Query
-h	Show real time progress
-v	Display detailed information about running operation.

RPM Command Query Operation: (Options used with query operation)

Option	Description
-q	Perform query operation
-a	List all installed packages in system
-c	List all configuration files from package
-d	List all documentation files from package
-R	List all dependent packages
-i	Provide information about package
-l	List all files from package
-f	Find the package which belong to specified file
-p	Perform query in individual package instead of RPM database

1. To install a package we use -i options.

   rpm –ivh <packagename>

   Eg: rpm –ivh mozilla-mail-1.7.5-17.i586.rpm

2. To remove a package we use –e options.

   rpm –evh <packagename>

   Eg: rpm –evh mozilla-mail

3. To update a package we use –Uvh options.

   rpm –Uvh packagename

   Eg: rpm –Uvh mozilla-mail

4. To check whether a particular package is installed in system or not we can use following command:

   rpm –q <PackageName>

   Eg: rpm –q MySql

5. To list all installed packages in system, we can use following command.

   rpm –qa

6. We can use –qa option with –last, it will list all the recently installed rpm packages.

   rpm –qa —last

7. To get more detailed information about a package such as version, release, architecture, group, size, build date, install date, license and vendor we can use following command.

   rpm -qi <PackageName>

   Eg: rpm –qi MySql

8. To view all the files of an installed rpm packages, use the -ql (query list) with rpm command.

   rpm -ql <Package Name>

REVIEW QUESTIONS

Part A

1. Define the term super user.
2. What are the different users in Linux?
3. Differentiate sudo and su.
4. Define useradd command.
5. How to set a password for the new user?
6. Define groupadd command.
7. Define userdel command.
8. What is chmod command?
9. What is the use of chown command?

10. How to delete a group?
11. Which command is used for mounting file system?
12. Differentiate uname and hostname commands.
13. What is rpm?
14. What is the use of mkfs command?
15. What is disk partitioning?

Part B

16. What are the roles of system administrator?
17. Write note on configuration and log files.
18. Write note on how to add and delete a user.
19. How to set permission for a file?
20. How to change the ownership of a file?
21. Write note on how to add and delete a group.
22. What are the different methods to temporary disable a user account?
23. What are the commands used for monitoring system performance?
24. Write note on creating and mounting file system.
25. How disk partitioning is performed?
26. Write note on rpm packages.

Part C

27. Explain the common administrative tasks in Linux.
28. What is file system? Explain how to create and mount a file system in Linux.
29. Explain the following commands:
    a. Useradd b) userdel c) Groupadd d)groupdel
    b. Chmod and chown