Sr. No: 11
Name: Soham Desai
XIE ID: 202003021

EXP. No 3.

A) Study of Unix file system (tree structure), file and directory permissions, single and multiuser environment.

B) Execution of File System Management Commands like ls, cd, pwd, cat, mkdir, rmdir, rm, cp, mv, chmod, wc, piping and redirection, grep, tr, echo, sort, head, tail, diff, less, more, file, type, wc, split, cmp, tar, find, gzip, bzip2, unzip, locate, etc.

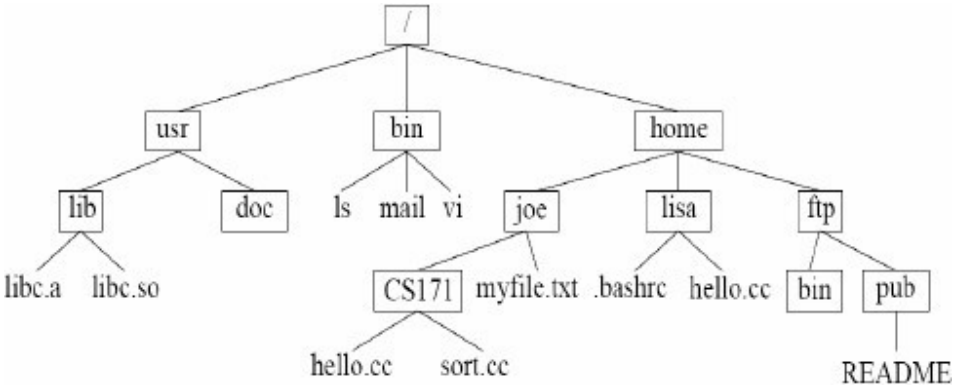
C) Execution of User Management Commands like who, whoami, su, sudo, login, logout, exit, passwd, useradd/adduser, usermod, userdel, groupadd, groupmod, groupdel, gpasswd, chown etc

Objective:- To understand file system management and user management commands in Unix.

Outcome:- Apply Unix commands for system administrative tasks such as file system management and user management

Description-

Linux Filesystem



A simple example of a hierarchical file system is shown in the above figure. Each boxed name represents a directory, while the unboxed names are files. Linux file names are case sensitive and may contain almost any character. File names may or may not be followed by an extension like .txt or .cc or html. In fact, the period in a file name is not given any special significance by a shell, and extensions are rarely required for a file to be opened by a particular application. However, it is usually a good idea to include an extension for a file so it is easier for you to figure out what kind of file it is. By convention, executable programs in Linux usually have no extension.

Any directory that is not the root is usually called a subdirectory. For example, in the figure, usr is a subdirectory of / and doc is a subdirectory of usr. The directory usr is also called the parent directory of doc and / is the parent directory of usr. The root directory is the only directory without a parent; by convention, the root directory is its own parent.

In a Linux filesystem, the bin subdirectory contains programs that correspond to core Linux commands. The usr subdirectory contains many other parts of the basic Linux system. The home subdirectory contains the

home directories of all the users with accounts on the system. If your username were joe, you could store your files in the joe subdirectory of home.

The pathname of a file contains a sequence of directories to follow to reach the file. For example, the pathname of the joe subdirectory is /home/joe. The pathname of the file myfile.txt in the joe subdirectory is /home/joe/myfile.txt. The pathnames above are called absolute pathnames because they contain all the information needed to find a file. On the other hand, a relative pathname gives the information necessary to find a file from a particular point in the tree. For example, from the directory /home, the relative pathname of myfile.txt is just joe/myfile.txt. Notice that you can tell the difference between an absolute and a relative pathname by looking for the leading forward slash.

Linux Disks and Partitions

Linux treats its devices as files. The special directory where these "files" are maintained is "/dev".

DISKS

Floppy (a:) /dev/fd0
Floppy (b:) /dev/fd1
1st Hard disk (master, IDE-0) /dev/hda
Hard disk (slave, IDE-0) /dev/hdb
Hard disk (master, IDE-1) /dev/hdc, etc.
1st SCSI hard disk /dev/sda
2nd SCSI hard disk /dev/sdb, etc.

PARTITIONS

1st Hard disk (master, IDE-0) /dev/had
1st Primary partition /dev/hda1
2nd Primary partition /dev/hda2, etc.
1st Logical drive (on ext'd part) /dev/hda5
2nd Logical drive /dev/hda6, etc.
2nd Hard disk (slave, IDE-0) /dev/hdb
1st Primary partition /dev/hdb1, etc
CDROM or 3rd disk (master, IDE-1) /dev/hdc
CDROM (SCSI) /dev/scd0
1st SCSI disk /dev/sda
1st Primary partition /dev/sda1, etc.

The pattern described above is fairly easy to follow. If you are using a standard IDE disk, it will be referred to as "hdx" where the "x" is replaced with an "a" if the disk is connected to the primary IDE controller as master and a "b" if the disk is connected to the primary IDE controller as a slave device. In the same way, the IDE disks connected to the secondary IDE controller as master and slave will be referred to as "hdc" and "hdd" respectively Note: Before a filesystems on devices can be used, they must be mounted. In order to

mount them, you must know what they arr called. So for example, if you use a parallel ZIP drive or USB disk (thumb drive, memory stick, etc.), it will be accessed as /dev/sda (assuming no other SCSI devices) or /dev/sdb.

A) Linux File Permissions

Every file or folder in Linux has access permissions. There are three types of permissions (what allowed to do with a file):

read access
write access
execute access

Permissions are defined for three types of users:

the owner of the file
the group that the owner belongs to
other users

Thus, Linux file permissions are nine bits of information (3 types x 3 type of users), each of them may have just one of two values: allowed or denied. Simply put, for each file it can be specified who can read or write from/to the file. For programs or scripts it also can be set if they are allowed to be executed. It is used in Linux long directory listings. It consists of 10 characters. The first character shows the file type. Next 9 characters are permissions, consisting of three groups: owner, group, others. Each group consists of three symbols: **rwx** (in this order), if some permission is denied, then a dash "-" is used instead. Example:
-rwxr--r--
0123456789
Symbol in the position 0 ("-") is the type of the file. It is either:

d = directory
- = regular file
l = symbolic link
s = Unix domain socket
p = named pipe
c = character device file
b = block device file

Symbols in positions 1 to 3 ("rwx") are permissions for the owner of the file.
Symbols in positions 4 to 6 ("r--") are permissions for the group.
Symbols in positions 7 to 9 ("r--") are permissions for others.

r Read access is allowed
w Write access is allowed
x Execute access is allowed

Replaces "r", "w" or "x" if according access type is denied

Examples:
-rwxr-xr-x

File,
Owner has read, write, and execute permissions,
Group: only read and execute permissions,
Others: only read and execute permissions.
dr-x------
Directory,
owner has read and execute access, group
and others have no access

Sr. No: 11
Name: Soham Desai
XIE ID: 202003021

If a numeric representation is used (like in chmod command, for example), then it is in the octal format (with the base of 8), and digits involved are 0 to 7. Octal format is used for the simplicity of understanding: every octal digit combines read, write and execute permissions together. Respective access rights for owner, group and others (in this order) are the last three digits of the numeric file permissions representation. Example: "0644". Here the second digit ("6" in the example) stands for rights of the owner, the third digit ("4" in the example) stands for rights of the group, the fourth digit ("4" in the example) stands for rights of others.

This table shows what numeric values mean:

Octal	Text	Binary	Meaning
0	---	000	All types of access are denied
1	--X	001	Execute access is allowed only
2	-W-	010	Write access is allowed only
3	-WX	011	Write and execute access are allowed
4	R--	100	Read access is allowed only
5	R-X	101	Read and execute access are allowed
6	RW-	110	Read and write access are allowed
7	RWX	111	Everything is allowed

B) Commands Related to file system-

Directories

Linux "folders" are called *directories*. The top-level, root directory is called /. Your home directory is **/home/***username*. From anywhere you can get back there by typing simply **cd** The short-hand name for the directory you happen to be in at any time is called "**.**" and the directory in which the current directory resides is called "**..**". Typing "**cd ..**" will move you to the next higher level directory. Several useful commands for directories are listed below.

Command	Function	Examples
cd	Change directory	cd, cd .., cd /home/catyp
pwd	Print working directory	pwd
mkdir	Make a new subdirectory	**mkdir** newdirectory
rmdir	Remove a directory	**rmdir** emptydirectory
ls	List files in a directory	ls, ls –l

Command	Function	Examples
mv	Rename (move) a file	**mv** *oldname newname*
cp	Copy a file	**cp** *oldname newname*
		cp *oldname dirname/*
rm	Delete (remove) a file	**rm** *filename,* **rm** *file1 file2 file3,* **rm** *-r dirname*
cat	Output the contents of a file	**cat** *filename* to the screen
file	Identify the type of file	**file** *filename*
head	Display the first few lines	**head** *filename* of a text file.
tail	Display the last few lines	**tail** *filename* of a text file.
chmod	Change access permissions on files	**chmod** mode *filename*
ln	Creates symbolic link	**ln -s** targetfile *linkname*

Sr. No: 11
Name: Soham Desai
XIE ID: 202003021

grep Print lines in file containing the **grep** *PATTERN* *file*
search

USER Management Commands-

There are three types of accounts on a Unix system –

Root account This is also called superuser and would have complete and unfettered control of the system. A superuser can run any commands without any restriction. This user should be assumed as a system administrator. System accounts System accounts are those needed for the operation of system-specific components for example mail accounts and the sshd accounts. These accounts are usually needed for some specific function on your system, and any modifications to them could adversely affect the system. User accounts User accounts provide interactive access to the system for users and groups of users. General users are typically assigned to these accounts and usually have limited access to critical system files and directories. Unix supports a concept of Group Account which logically groups a number of accounts. Every account would be a part of another group account. A Unix group plays important role in handling file permissions and process management. Managing Users and Groups There are four main user administration files –

/etc/passwd − Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system. /etc/shadow − Holds the encrypted password of the corresponding account. Not all the systems support this file.

/etc/group − This file contains the group information for each account.

/etc/gshadow − This file contains secure group account information. Check all the above files using the cat command. The following table lists out commands that are available on majority of Unix systems to create and manage accounts and groups

Create a Group -We will now understand how to create a group. For this, we need to create groups before creating any account otherwise, we can make use of the existing groups in our system. We have all the groups listed in /etc/groups file.

groupadd [-g gid [-o]] [-r] [-f] groupname

$ groupmod -n new_modified_group_name old_group_name

$ groupmod -n developer developer_2

$ groupdel developer

$ useradd -d /home/chhaya -g developers -s /bin/ksh chhaya

$ passwd chhaya

$ Changing password for user chhaya.

$New UNIX password: $Retype new UNIX password: ,

$passwd: all authentication tokens updated successfully, $ userdel -r chhaya

Sr. No: 11
Name: Soham Desai
XIE ID: 202003021

```
touch A1.txt
touch A2.txt
pwd
find /home/anshuman09 -name "*.txt"
locate "*.txt"
cat >> anshu2.txt
grep Xavier *.txt
ls
ls | grep "h"
ls > myfiles
cat myfiles
ls -l
umask
chmod u+x A1.txt
ls -l
chmod g+x A1.txt
ls -l
chmod o+wx A1.txt
ls -l
*
ls -l
chmod 664  A1.txt
ls -l
chmod 661  A1.txt
ls -l
chmod 456 A1.txt
ls -l
ls -l
chmod 740 A1.txt
ls -l
chmod 441 A1.txt
ls -l
chmod 174 A1.txt
ls -l
chmod 777 A1.txt
ls -l
cat >> SEIT
head SEIT
tail SEIT
tail -5 SEIT
wc -l SEIT
wc -c SEIT
wc -w SEIT
mkdir afile50
ls
cd afile50
touch ad1 ad2
cd ..
tar -cf com1.tar afile50
ls
tar -cf com1.tar
tar -xf com1.tar
ls
gzip A2.txt
ls
```

Conclusion: We have learned and successfully performed different file management commands and User Management commands .