	SHRI SANT GAJANAN MAHARAJ COLLEGE ENGG. SHEGAON		LABORATORY MANUAL
	PRACTICAL EXPERIMENT INSTRUCTION SHEET		
	EXPERIMENT TITLE: **Basic Linux Commands**		
EXPERIMENT NO.: SSGMCE/WI/ASH/01/1A6/01		ISSUE NO.:	**ISSUE DATE:**26.08.23
REV. DATE: 26.08.23	REV. NO.: 00	DEPTT. : INFORMATION TECHNOLOGY.	
LABORATORY: OPERATING SYSTEMS		SEMESTER : IV	PAGE NO. 01 OF 05

Date:

BASIC LINUX COMMANDS

01. AIM :

To study basic Commands in a Linux Operating System.

02. FACILITIES :

1. **Linux Operating System** (e.g., Ubuntu, CentOS).
2. **Terminal/CLI** for executing commands.
3. **Text Editor** (e.g., Nano, Vim) for writing scripts.

03. SCOPE :

This experiment explores fundamental Linux commands, including:
1. **File Management**: Commands for handling files and directories.
2. **Permissions and Ownership**: Commands for modifying file permissions and ownership.
3. **Text Processing**: Commands for manipulating and searching through text files.

Additional commands related to **System Monitoring**, **User Management**, and **Networking** are also referenced.

04 THEORY:

Linux Commands:

Linux commands are the basic tools used to interact with Linux on an individual level. They are used to perform a variety of tasks, including displaying information about files and directories. All users should be familiar with most of these commands as they are required for most operating system tasks and computer programming.

File Management Commands:

These commands are used for handling files and directories within the file system. They allow you to list, copy, move, delete, and create files and directories, helping you manage your system's file structure. Common commands include `ls`, `cp`, `mv`, `rm`, and `mkdir`.

System Monitoring Commands:

System monitoring commands are used to check system information, resource usage, and the status of running processes. They help you track performance, memory, CPU usage, and active tasks. Examples include `top`, `cal`, `df`, and `free`.

Permissions and Ownership Commands:

These commands help you manage file permissions and ownership. You can control who can read, write, or execute a file, as well as change the file's owner or group. Common commands include **`chmod`**, **`chown`**, **`chgrp`**, and **`umask`**.

Text Processing Commands:

Text processing commands are used to manipulate and analyze text data. They allow you to search, filter, format, and transform text files or output from other commands. Important commands include **`cat`**, **`grep`**, **`awk`**, **`sed`**, and **`cut`**.

Some Basic Linux Commands:

1. **`ls` command:**
   The **`ls` command** is commonly used to identify the files and directories in the working directory. This command can be used by itself without any arguments and it will provide us the output with all the details about the files and the directories in the current working directory. There is a lot of flexibility offered by this command in terms of displaying data in the output.

```
localhost:~# ls
bench.py    hello.c     hello.js    readme.txt
localhost:~# ls -l
total 16
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
localhost:~# ls -a
.             .ash_history  .mozilla      bench.py      hello.js
..            .cache        .wine         hello.c       readme.txt
localhost:~# ls -t
readme.txt  bench.py    hello.c     hello.js
localhost:~# ls -tl
total 16
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
localhost:~# ls -hl
total 16K
-rw-r--r--    1 root     root         114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root          76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root          22 Jun 26  2020 hello.js
-rw-r--r--    1 root     root         151 Jul  5  2020 readme.txt
```

Commonly Used Options in `ls` command in Linux

Options	Description
`-l`	known as a long format that displays detailed information about files and directories.
`-a`	Represent all files Include hidden files and directories in the listing.
`-S`	Sort files and directories by their sizes, listing the largest ones first.
`-R`	List files and directories recursively, including subdirectories.
`-i`	known as inode which displays the index number (inode) of each file and directory.
`-g`	known as group which displays the group ownership of files and directories instead of the owner.
`-h`	Print file sizes in human-readable format (e.g., 1K, 234M, 2G).
`-d`	List directories themselves, rather than their contents.

2. `mkdir` command:

In Linux, the 'mkdir' command is like a magic wand for creating folders super easily. 'mkdir' stands for "make directory," and it helps you organize your computer stuff by creating folders with just one command.

This command can create multiple directories at once as well as set the permissions for the directories. It is important to note that the user executing this command must have enough permission to create a directory in the parent directory, or he/she may receive a 'permission denied' error.

Syntax: `mkdir [options...] [directory_name]`

```
localhost:~# mkdir sample
localhost:~# ls -l
total 20
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
drwxr-xr-x    2 root     root            37 Feb 12 19:57 sample
localhost:~#
```

3. `cd` command:

The **`cd` command in Linux** is one of the most basic and frequently used commands for Linux users. The `cd` command allows you to **change directories in Linux**, making it easier to navigate through the file system and manage your

files efficiently. Whether you're a beginner or an experienced user, understanding how to **use the `cd` command in Linux** is essential for smooth command-line operations.

Description	Notes
`cd [directory]`	Basic Syntax. Defaults to home.
`cd /path/to/directory`	Navigation to a Specific Directory.
`cd or cd ~`	Navigation to Home Directory.
`cd -`	Navigation to a Previous Directory.
`cd ..`	Navigation to Parent Directory.

```
localhost:~# ls
bench.py    hello.c     hello.js    readme.txt  sample
localhost:~# cd sample
localhost:~/sample# cd -
/root
localhost:~# cd ~
localhost:~# cd ..
localhost:/# ls
bin    etc    lib    mnt    proc   run    srv    tmp    var
dev    home   media  opt    root   sbin   sys    usr
localhost:/#
```

4. **`mv` command:**

The `mv` command in Linux is like a superhero tool that can do a bunch of cool stuff with your files and folders. Think of it as a digital moving truck that helps you shift things around in your computer. Whether you want to tidy up your folders, give your files new names, or send them to different places, `mv` is the go-to friend for the job.

Syntax:

```
mv [options(s)] [source_file_name(s)] [Destination_file_name]
```

Here,
source_file_name(s) = The name of the files that we want to rename or move.
Destination_file_name = The name of the new location or the name of the file.

```
localhost:~# ls
bench.py    hello.c     hello.js    readme.txt  sample
localhost:~# mv sample new
localhost:~# ls
bench.py    hello.c     hello.js    new         readme.txt
localhost:~#
```

5. **`touch` command:**

The **`touch`** command is a standard command used in the UNIX/Linux operating system which is used to create, change and modify the timestamps of a file. It is used to create a file without any content. The file created using the touch command is empty. This command can be used when the user doesn't have data to store at the time of file creation.

Options	Description
`-a`	This option changes the access time only.
`-c`	Suppresses file creation if the file does not exist.
`-d`	Sets the access & modification times using the specified STRING.
`-m`	This option changes the modification time only.
`-r`	Uses the access and modification times from the reference file.

```
localhost:~# ls
bench.py    hello.c     hello.js    readme.txt
localhost:~# ls -l
localhost:~# ls
bench.py    hello.c     hello.js    new         readme.txt
localhost:~# touch f1 f2 f3 f4 f5
localhost:~# ls
bench.py    f2          f4          hello.c     new
f1          f3          f5          hello.js    readme.txt
```

6. **`cat` command:**

The **`cat`** command in Linux is more than just a simple tool, it's a versatile companion for various file-related operations, allowing **users to view, concatenate, create, copy, merge**, and **manipulate file contents**.

Syntax: `cat [OPTION] [FILE]`

Options	Description
`cat [filename]`	Display the contents of the file
`cat > [newfile]`	Create a new file and enter contents or copy contents from one file to another.
`cat >> [existingfile]`	Append new content to an existing file.
`cat -n [filename]`	Display the line numbers of the lines.

```
localhost:~# ls
bench.py    hello.c     hello.js    readme.txt
localhost:~# ls -l
localhost:~# ls -l
total 28
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            23 Feb 13 00:06 file.txt
-rw-r--r--    1 root     root            22 Feb 13 00:06 file1.txt
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
drwxr-xr-x    2 root     root            37 Feb 12 19:57 new
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
localhost:~# cat file.txt
Hello, this is a file.
localhost:~# cat file.txt file1.txt
Hello, this is a file.
Hello, this is file1.
localhost:~# cat > new.txt
This is a new file.
This is a new file.
localhost:~# ls -l
total 32
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            23 Feb 13 00:06 file.txt
-rw-r--r--    1 root     root            22 Feb 13 00:06 file1.txt
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
drwxr-xr-x    2 root     root            37 Feb 12 19:57 new
-rw-r--r--    1 root     root            20 Feb 13 00:10 new.txt
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
localhost:~# cat >> file.txt
This is new text.
localhost:~# cat file.txt
Hello, this is a file.
This is new text.
localhost:~# cat file.txt file1.txt > combined.txt
localhost:~# cat combined.txt
Hello, this is a file.
This is new text.
Hello, this is file1.
localhost:~# ls -l
total 36
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            63 Feb 13 00:12 combined.txt
-rw-r--r--    1 root     root            41 Feb 13 00:11 file.txt
-rw-r--r--    1 root     root            22 Feb 13 00:06 file1.txt
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
drwxr-xr-x    2 root     root            37 Feb 12 19:57 new
-rw-r--r--    1 root     root            20 Feb 13 00:10 new.txt
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
localhost:~#
```

7. `cal` command:

The `cal` command is a calendar command in Linux which is used to see the calendar of a specific month or a whole year. By default, entering cal in the terminal shows the calendar of the current month, with today's date highlighted. This provides a quick overview of the month at hand.

Syntax: `cal [ [ month ] year]`

```
localhost:~# cal june 2024
      June 2024
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
```

8. `chmod` command:

The `chmod` command in Linux is a powerful tool that allows you to modify the permissions of files and directories, controlling who can read, write, or execute them. Understanding how to use symbolic and numeric modes is essential for securing files and directories in a Linux environment.

Command	Description
`chmod +x file,` `chmod u+x file`	Adds execute permission for **everyone** or for the **user** (owner).
`chmod g-w file`	Removes write permission for the **group**.
`chmod u=rwx,g=rx,o=r file`	Set exact permissions for **user** (rwx), **group** (rx), and **others** (r).
`chmod 755 file,` `chmod 644 file`	Set permissions in numeric form (e.g., 755 gives `rwx` for user, `rx` for group and others).

```
localhost:~# ls -l
total 36
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            63 Feb 13 00:12 combined.txt
-rw-r--r--    1 root     root            41 Feb 13 00:11 file.txt
-rw-r--r--    1 root     root            22 Feb 13 00:06 file1.txt
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
drwxr-xr-x    2 root     root            37 Feb 12 19:57 new
-rw-r--r--    1 root     root            20 Feb 13 00:10 new.txt
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
localhost:~# chmod u+x file.txt
localhost:~# ls -l
total 36
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            63 Feb 13 00:12 combined.txt
-rwxr--r--    1 root     root            41 Feb 13 00:11 file.txt
-rw-r--r--    1 root     root            22 Feb 13 00:06 file1.txt
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
drwxr-xr-x    2 root     root            37 Feb 12 19:57 new
-rw-r--r--    1 root     root            20 Feb 13 00:10 new.txt
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
localhost:~#
```

08 CONCLUSION:

This experiment helped familiarize users with essential Linux commands for managing files, modifying permissions, processing text, and performing basic system administration tasks. Mastery of these commands is fundamental for navigating the Linux environment, and these tools form the basis for more advanced tasks, such as system automation and scripting.

09 VIVA QUESTIONS:

PREPARED BY **H. P. Amle**	**APPROVED BY** **HOD**