Introduction to Linux

Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. The defining component of Linux is the Linux kernel, an operating system kernel first released in 1991, by Linus Torvalds.

Today, Linux systems are used in every domain, from embedded systems to supercomputers. 90% of all cloud infrastructure is powered by Linux, including supercomputers and cloud providers. 74% of smartphones in the world are Linux-based.

I. FILE HANDLING UTILITIES

cat

cat command concatenates files and print it on the standard output.

SYNTAX:

The Syntax is

cat [OPTIONS] [FILE]...

OPTIONS:

-A	Show all.
-b	Omits line numbers for blank space in the output.
-e	A $ character will be printed at the end of each line prior to a new line.
-E	Displays a $ (dollar sign) at the end of each line.
-n	Line numbers for all the output lines.
-s	If the output has multiple empty lines it replaces it with one empty line.
-T	Displays the tab characters in the output.
-v	Non-printing characters (with the exception of tabs, new-lines and form- feeds) are printed visibly.

EXAMPLE:

1. To Create a new file:
   cat > file1.txt
   This command creates a new file named file1.txt. Start typing the file content onto the display screen. After typing, press <ctrl>d to end and save the file.
2. To Append data into the file:
   cat >> file1.txt
   To append data into the same file, use append operator >> to write into the file, else the file will be overwritten (i.e., all of its contents will be erased).
3. To display a file:
   cat file1.txt
   This command displays the data in the file.
4. To concatenate several files and display

1. cat file1.txt file2.txt

   The above cat command will concatenate the two files (file1.txt and file2.txt) and it will display the output in the screen. Sometimes the output may not fit the monitor screen. In such situation you can print those files to a new file or display the file using less command.

   cat file1.txt file2.txt | less
2. To concatenate several files and to transfer the output to another file.

   cat file1.txt file2.txt > file3.txt

   In the above example the output is redirected to new file file3.txt. The cat command will create new file file3.txt and store the concatenated output into file3.txt.

rm

rm Linux command is used to remove/delete the file from the directory

Caution: Always use *rm -i* not *rm*

SYNTAX:

The Syntax is

rm [options..] [file | directory]

OPTIONS:

-f	Remove all files in a directory without prompting the user.
-i	Interactive. With this option, rm prompts for confirmation before removing any files.
-r (or) -R	Recursively remove directories and subdirectories in the argument list. The directory will be emptied of files and removed. The user is normally prompted for removal of any write-protected files which the directory contains.

EXAMPLE:

1. To Remove / Delete a file:

   rm file1.txt

   Here rm command will remove/delete the file file1.txt.
2. To delete a directory tree:

   rm -ir tmp

   This rm command recursively removes the contents of all subdirectories of the tmp directory, prompting you regarding the removal of each file, and then removes the tmp directory itself.
3. To remove multiple files

   rm file1.txt file2.txt

   rm command removes file1.txt and file2.txt files at the same time.

cd

cd command is used to change the directory.

Syntax is:

cd [directory | ~ | ./ | ../ | -]

OPTIONS:

-L	Use the physical directory structure.
-P	Forces symbolic links.

EXAMPLE:

1. cd sub
   This command will take you to the sub-directory called sub from its parent directory.
2. cd **..**
   This will change to the parent-directory from the current working directory/sub-directory.

   Note: Double dots **..** refers to parent directory. Single dot **.** refers to current directory
3. cd ~
   This command will move to the user's home directory which is "/home/username".

cp

cp command copy files from one location to another

If the destination is an existing file, then the file is overwritten; if the destination is an existing directory, the file is copied into the directory (the directory is not overwritten).

SYNTAX:

The Syntax is

cp [OPTIONS]... SOURCE DEST
cp [OPTIONS]... SOURCE... DIRECTORY
cp [OPTIONS]... --target-directory=DIRECTORY SOURCE...

OPTIONS:

-a	same as -dpR.
-v	verbose
-b	like --backup but does not accept an argument.
-f	if an existing destination file cannot be opened, remove it and try again.
-p	same as –preserve mode, ownership, timestamps.

EXAMPLE:

1. Copy a file:
   cp file1 file2
   The above cp command copies the content of file1.php to file2.php.
2. To backup the copied file:
   cp -b file1.php file2.php
   Backup of file1.php will be created with '~' symbol as file2.php~.
3. Copy folder and subfolders:
   cp -R srcdir destdir
   cp -R command is used for recursive copy of all files and directories in source directory
4. cp -Rv dev bak

ls

ls command lists the files and directories in current working directory

SYNTAX:

The Syntax is

ls [OPTIONS]... [FILE]

OPTIONS:

-l	Lists all the files, directories and their mode, Number of links, owner of the file, file size, Modified date and time and filename.
-t	Lists in order of last modification time.
-a	Lists all entries including hidden files.
-d	Lists directory files instead of contents.
-p	Puts slash at the end of each directories.
-u	List in order of last access time.
-i	Display inode information.
-ltr	List files order by date.
-lSr	List files order by file size.

EXAMPLE:

1. Display root directory contents:
   ls /
   lists the contents of root directory.
2. Display hidden files and directories:
   ls -a
   lists all entries including hidden files and directories.
3. Display inode information:
   ls -i
   7373073 book.gif
   7373074 clock.gif
   The above command displays filename with inode value.

ln

ln command is used to create link to a file (or) directory.

It helps to provide soft link (shortcut) for desired files.

Inode will be different for source and destination.

SYNTAX:

The Syntax is

ln [options] existingfile(or directory)name newfile(or directory)name

OPTIONS:

-f	Link files without questioning the user, even if the mode of target forbids writing. This is the default if the standard input is not a terminal.
-n	Does not overwrite existing files.
-s	Used to create soft links.

EXAMPLE:

1. ln -s file1.txt file2.txt
   Creates a symbolic link to 'file1.txt' with the name of 'file2.txt'. Here inode for 'file1.txt' and 'file2.txt' will be different.
2. ln -s nimi nimi1
   Creates a symbolic link to 'nimi' with the name of 'nimi1'.

mkdir

mkdir command is used to create one or more directories.

SYNTAX:

The Syntax is

mkdir [options] directories

OPTIONS:

-m	Set the access mode for the new directories.
-p	Create intervening parent directories (path) if they don't
-v	Print help message for each directory created.

EXAMPLE:

1. Create directory:
   mkdir test
   The above command is used to create the directory 'test'.
2. Create directory and set permissions:
   mkdir -m 666 test
   The above command is used to create the directory 'test' and set the read and write permission.

rmdir

rmdir command is used to delete/remove a directory and its subdirectories.

SYNTAX:

The Syntax is

rmdir [options..] Directory

OPTIONS:

-p	Allow users to remove the directory dirname and its parent directories that become empty.

EXAMPLE:

1. To delete/remove a directory
   rmdir tmp
   rmdir command will remove/delete the directory tmp if the directory is empty.
2. To delete a directory tree:
   rm -ir tmp
   This command recursively removes the contents of all subdirectories of the tmp directory, prompting you regarding the removal of each file, and then removes the tmp directory itself.

mv

mv command is short for move. It is used to move/rename file

mv command is different from cp command as it completely removes the file from the source and moves to the directory specified, where cp command just copies the content from one file to another.

SYNTAX:

The Syntax is

mv [-f] [-i] oldname newname

OPTIONS:

-f	This will not prompt before overwriting (equivalent to --reply=yes). mv -f will move the file(s) without prompting even if it is writing over an existing target.
-i	Prompts before overwriting another file.

EXAMPLE:

1. To Rename / Move a file:
   mv file1.txt file2.txt
   This command renames file1.txt as file2.txt
2. To move a directory
   mv hscripts tmp
   In the above line mv command moves all the files, directories and sub-directories from hscripts folder/directory to tmp directory if the tmp directory already exists. If there is no tmp directory it renames the hscripts directory as tmp directory.
3. To Move multiple files/More files into another directory
   mv file1.txt tmp/file2.txt newdir
   This command moves the files file1.txt from the current directory and file2.txt from the tmp folder/directory to newdir.

diff

diff command is used to find differences between two files.

SYNTAX:

The Syntax is

diff [options..] from-file to-file

OPTIONS:

-a	Treat all files as text and compare them line-by-line.
-b	Ignore changes in amount of white space.
-c	Use the context output format.
-e	Make output that is a valid ed script.
-H	Use heuristics to speed handling of large files that have numerous scattered small changes.
-i	Ignore changes in case; consider upper- and lower-case letters equivalent.
-n	Prints in RCS-format, like -f except that each command specifies the number of lines affected.
-q	Output RCS-format diffs; like -f except that each command specifies the number of lines affected.
-r	When comparing directories, recursively compare any subdirectories found.
-s	Report when two files are the same.
-w	Ignore white space when comparing lines.
-y	Use the side by side output format.

EXAMPLE:

1. Compare files ignoring white space: diff -w file1.txt file2.txt
   This command will compare the file file1.txt with file2.txt ignoring white/blank space and it will produce output.

2. Compare the files side by side, ignoring white space: diff -by file1.txt file2.txt
   This command will compare the files ignoring white/blank space, It is easier to differentiate the files.

3. Compare the files ignoring case. diff -iy file1.txt file2.txt
   This command will compare the files ignoring case(upper-case and lower-case).

wc

Short for word count, wc displays a count of lines, words, and characters in a file.

Syntax

wc [*-c* | *-m* | *-C*] [*-l*] [*-w*] [*file* ...]

-c	Count bytes.
-m	Count characters.
-C	Same as -m.
-l	Count lines.

Examples

wc myfile.txt - Displays information about the file myfile.txt.

comm

Select or reject lines common to two files.

Syntax

comm [-1] [-2] [-3] file1 file2

-1	Suppress the output column of lines unique to file1.
-2	Suppress the output column of lines unique to file2.
-3	Suppress the output column of lines duplicated in file1 and file2.
file1	Name of the first file to compare.

Examples

comm myfile1.txt myfile2.txt

The above example would compare the two files myfile1.txt and myfile2.txt.

II. SECURITY BY FILE PERMISSIONS

This section will cover the following commands:

- chmod - modify file access rights
- su - temporarily become the superuser
- chown - change file ownership
- chgrp - change a file's group ownership

File permissions

Linux uses the same permissions scheme as UNIX. Each file and directory on your system is assigned access rights for the owner of the file, the members of a group of related users, and everybody else. Rights can be assigned to read a file, to write a file, and to execute a file (i.e., run the file as a program).

To see the permission settings for a file, we can use the `ls` command as follows:

```
$ ls -l filename
```

Let's try another example. We will look at the`bash` program which is located in the `/bin` directory:

```
$ ls -l /bin/bash
```

```
-rwxr-xr-x 1 root root 1183448 Feb 1  2023 /bin/bash
```

Here we can see:

- The file "/bin/bash" is owned by user "root"
- The superuser has the right to read, write, and execute this file
- The file is owned by the group "root"
- Members of the group "root" can also read and execute this file
- Everybody else can read and execute this file

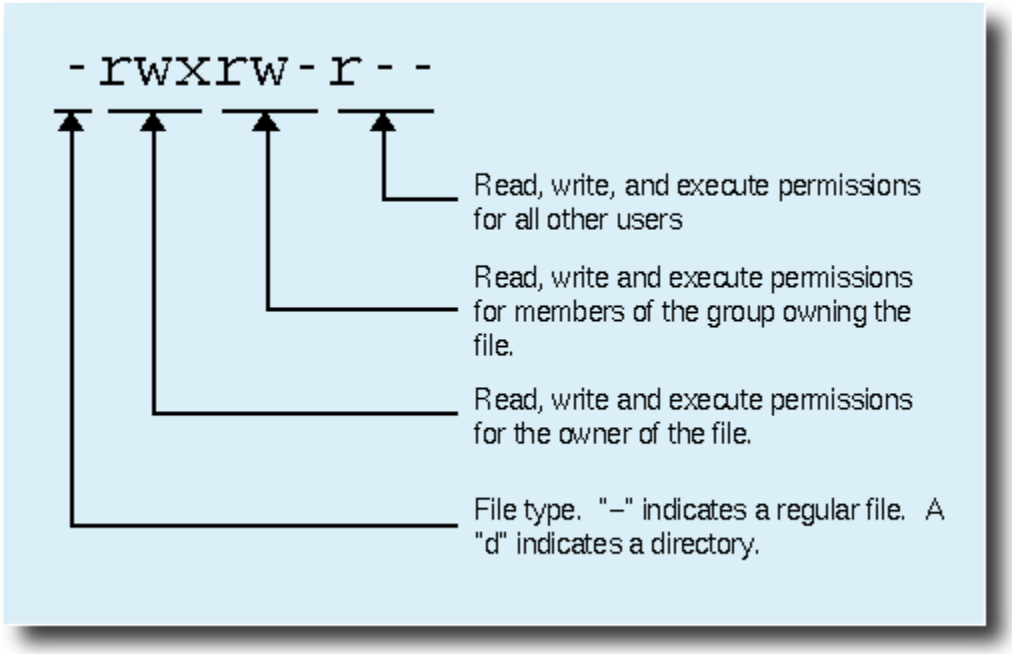
File Permission

#	File Permission
0	none
1	execute only
2	write only
3	write and execute
4	read only
5	read and execute
6	read and write
7	set all permissions

OPTIONS:

-c	Displays names of only those files whose permissions are being changed

In the diagram below, we see how the first portion of the listing is interpreted. It consists of a character indicating the file type, followed by three sets of three characters that convey the reading, writing and execution permission for the owner, group, and everybody else.



chmod

The `chmod` command is used to change the permissions of a file or directory. To use it, you specify the desired permission settings and the file or files that you wish to modify. There are more ways to specify the permissions.

It is easy to think of the permission settings as a series of bits (which is how the computer thinks about them). Here's how it works:

rwx rwx rwx = 111 111 111
rw- rw- rw- = 110 110 110
rwx --- --- = 111 000 000

and so on...

rwx = 111 in binary = 7
rw- = 110 in binary = 6
r-x = 101 in binary = 5
r-- = 100 in binary = 4

Now, if you represent each of the three sets of permissions (owner, group, and other) as a single digit, you have a pretty convenient way of expressing the possible permissions settings. For example, if we wanted to set `some_file` to have read and write permission for the owner, but wanted to keep the file private from others, we would:

```
$ chmod 600 some_file
```

Here is a table of numbers that covers all the common settings. The ones beginning with "7" are used with programs (since they enable execution) and the rest are for other kinds of files.

Value	*Meaning*
777	*(rwxrwxrwx)* No restrictions on permissions. Anybody may do anything. Generally not a desirable setting.
755	*(rwxr-xr-x)* The file's owner may read, write, and execute the file. All others may read and execute the file. This setting is common for programs that are used by all users.
700	*(rwx------)* The file's owner may read, write, and execute the file. Nobody else has any rights. This setting is useful for programs that only the owner may use and must be kept private from others.
666	*(rw-rw-rw-)* All users may read and write the file.
644	*(rw-r--r--)* The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change.
600	*(rw-------)* The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private.

Directory permissions

The `chmod` command can also be used to control the access permissions for directories. In most ways, the permissions scheme for directories works the same way as they do with files. However, the execution permission is used in a different way. It provides control for access to file listing and other things. Here are some useful settings for directories:

Value	*Meaning*
777	*(rwxrwxrwx)* No restrictions on permissions. Anybody may list files, create new files in the directory and delete files in the directory. Generally not a good setting.
755	*(rwxr-xr-x)* The directory owner has full access. All others may list the directory, but cannot create files nor delete them. This setting is common for directories that you wish to share with other users.
700	*(rwx------)* The directory owner has full access. Nobody else has any rights. This setting is useful for directories that only the owner may use and must be kept private from others.

-c Change the permission for each file.

su

Becoming the superuser for a short while

It is often useful to become the superuser to perform important system administration tasks, but as you have been warned (and not just by me!), you should not stay logged on as the superuser. In most distributions, there is a program that can give you temporary access to the superuser's privileges. This program is called `su`(short for substitute user) and can be used in those cases when you need to be the superuser for a small number of tasks. To become the superuser, simply type the `su` command. You will be prompted for the superuser's password:

```
$ su Password:
[root@Linuxbox me]#
```

After executing the `su`command, you have a new shell session as the superuser. To exit the superuser session, type `exit` and you will return to your previous session.

sudo

In some distributions, most notably Ubuntu, an alternate method is used. Rather than using `su`, these systems employ the `sudo` command instead. With `sudo`, one or more users are granted superuser privileges on an as needed basis. To execute a command as the superuser, the desired command is simply preceeded with the`sudo` command. After the command is entered, the user is prompted for the user's password rather than the superuser's:

```
$ sudo some_command Password:
```

III. PROCESS UTILITIES

ps

ps command is used to report the process status. ps is the short name for Process Status.

SYNTAX:

The Syntax is ps
[options]

OPTIONS:

-a	List information about all processes most frequently requested: all those except process group leaders and processes not associated with a terminal..
-A or e	List information for all processes.
-d	List information about all processes except session leaders.
-e	List information about every process now running.

-f Generates a full listing.
-j Print session ID and process group ID.
-l Generate a long listing.

EXAMPLE:

1. ps

Output:

```
PID TTY          TIME CMD
2540 pts/1    00:00:00 bash
2621 pts/1    00:00:00 ps
```

In the above example, typing ps alone would list the current running processes.

kill

kill command is used to kill the process.

SYNTAX:

The Syntax is

kill [-s] [-l] %pid

OPTIONS:

-s	Specify the signal to send. The signal may be given as a signal name or number.
-l	Write all values of signal supported by the implementation, if no operand is given.
-pid	Process id or job id.
-9	Force to kill a process.

EXAMPLE:

Step by Step process:

- Open a process music player.
  xmms
  press ctrl+z to stop the process.
- To know group id or job id of the background task.
  jobs -l
- It will list the background jobs with its job id as,
- xmms 3956
  kmail 3467
- To kill a job or process.
  kill 3956
  kill command kills or terminates the background process xmms.

at

Schedules a command to be ran at a particular time

Syntax

at executes commands at a specified time.

atq lists the user's pending jobs, unless the user is the superuser; in that case, everybody's jobs are listed. The format of the output lines (one for each job) is: Job number, date, hour, job class.

atrm deletes jobs, identified by their job number.

batch executes commands when system load levels permit; in other words, when the load average drops below 1.5, or the value specified in the invocation of atrun.

at [-c | -k | -s] [-f filename] [-q queuename] [-m] -t time [date] [-l] [-r]

-c C shell. csh(1) is used to execute the at-job.

-k Korn shell. ksh(1) is used to execute the at-job.

-s Bourne shell. sh(1) is used to execute the at-job.

-f filename Specifies the file that contains the command to run.

-m Sends mail once the command has been run.

-t time Specifies at what time you want the command to be ran. Format hh:mm. am / pm indication can also follow the time otherwise a 24-hour clock is used. A timezone name of GMT, UCT or ZULU (case insensitive) can follow to specify that the time is in Coordinated Universal Time. Other timezones can be specified using the TZ environment variable. The below quick times can also be entered:

midnight - Indicates the time 12:00 am (00:00).
noon - Indicates the time 12:00 pm.
now - Indicates the current day and time. Invoking at - now will submit submit an at-job for potentially immediate execution.

date Specifies the date you wish it to be ran on. Format month, date, year. The following quick days can also be entered:

today - Indicates the current day.
tomorrow - Indicates the day following the current day.

-l Lists the commands that have been set to run.

-r Cancels the command that you have set in the past.

Examples

at -m 01:35 < atjob

Run the commands listed in the 'atjob' file at 1:35AM, in addition to all output that is generated from job mail to the user running the task.

III. FILTERS

more

more command is used to display text in the terminal screen.

SYNTAX:

The Syntax is

more [options] filename

OPTIONS:

-c	Clear screen before displaying.
-e	Exit immediately after writing the last line of the last file in the argument list.
-n	Specify how many lines are printed in the screen for a given file.
+n	Starts up the file from the given number.

EXAMPLE:

1. more -c index.php
   Clears the screen before printing the file .
2. more -3 index.php
   Prints first three lines of the given file. Press **Enter** to display the file line by line.

head

head command is used to display the first ten lines of a file, and also specifies how many lines to display.

SYNTAX:

The Syntax is

head [options] filename

OPTIONS:

-n	To specify how many lines you want to display.
-n number	The number option-argument must be a decimal integer whose sign affects the location in the file, measured in lines.
-c number	The number option-argument must be a decimal integer whose sign affects the location in the file, measured in bytes.

EXAMPLE:

1. head index.php
   This command prints the first 10 lines of 'index.php'.
2. head -5 index.php

The head command displays the first 5 lines of 'index.php'.

3. head -c 5 index.php

The above command displays the first 5 characters of 'index.php'.

tail

tail command is used to display the last or bottom part of the file. By default it displays last 10 lines of a file.

SYNTAX:

The Syntax is

tail [options] filename

OPTIONS:

-l	To specify the units of lines.
-b	To specify the units of blocks.
-n	To specify how many lines you want to display.
-c number	The number option-argument must be a decimal integer whose sign affects the location in the file, measured in bytes.
-n number	The number option-argument must be a decimal integer whose sign affects the location in the file, measured in lines.

EXAMPLE:

1. tail index.php

It displays the last 10 lines of 'index.php'.

2. tail -2 index.php

It displays the last 2 lines of 'index.php'.

3. tail -n 5 index.php

It displays the last 5 lines of 'index.php'.

4. tail -c 5 index.php

It displays the last 5 characters of 'index.php'.

cut

cut command is used to cut out selected fields of each line of a file. The cut command uses delimiters to determine where to split fields.

SYNTAX:

The Syntax is cut

[options]

OPTIONS:

-c	Specifies character positions.
-b	Specifies byte positions.
-d flags	Specifies the delimiters and fields.

EXAMPLE:

1. cut -c1-3 text.txt

   Output:

   Thi

   Cut the first three letters from the above line.

2. cut -d, -f1,2 text.txt

   Output:

   This is, an example program

   The above command is used to split the fields using delimiter and cut the first two fields.

paste

paste command is used to paste the content from one file to another file. It is also used to set column format for each line.

SYNTAX:

The Syntax is paste [options]

OPTIONS:

-s	Paste one file at a time instead of in parallel.
-d	Reuse characters from LIST instead of TABs

EXAMPLE:

1. paste test.txt>test1.txt

   Paste the content from 'test.txt' file to 'test1.txt' file.

2. ls | paste - - - -

   List all files and directories in four columns for each line.

sort

sort command is used to sort the lines in a text file.

SYNTAX:

The Syntax is sort [options] filename

OPTIONS:

-r	Sorts in reverse order.
-u	If line is duplicated display only once.
-o filename	Sends sorted output to a file.

EXAMPLE:

1. sort test.txt

   Sorts the 'test.txt'file and prints result in the screen.

2. sort -r test.txt

   Sorts the 'test.txt' file in reverse order and prints result in the screen.

uniq

Report or filter out repeated lines in a file.

Syntax

uniq [-c | -d | -u] [-f fields] [-s char] [-n] [+m] [input_file [output_file]]

-c	Precede each output line with a count of the number of times the line occurred in the input.
-d	Suppress the writing of lines that are not repeated in the input.
-u	Suppress the writing of lines that are repeated in the input.
-f fields	Ignore the first fields fields on each input line when doing comparisons, where fields is a positive decimal integer. A field is the maximal string matched by the basic regular expression: [[:blank:]]*[^[:blank:]]* If fields specifies more fields than appear on an input line, a null string will be used for comparison.
-s char	Ignore the first chars characters when doing comparisons, where chars is a positive decimal integer. If specified in conjunction with the -f option, the first chars characters after the first fields fields will be ignored. If chars specifies more characters than remain on an input line, a null string will be used for comparison.
-n	Equivalent to -f fields with fields set to n.
+m	Equivalent to -s chars with chars set to m.
input_file	A path name of the input file. If input_file is not specified, or if the input_file is -, the standard input will be used.
output_file	A path name of the output file. If output_file is not specified, the standard output will be used. The results are unspecified if the file named by output_file is the file named by input_file.

Examples

uniq myfile1.txt > myfile2.txt - Removes duplicate lines in the first file1.txt and outputs the results to the second file.

IV. General Commands

date

date command prints the date and time.

SYNTAX:

The Syntax is

date [options] [+format] [date]

OPTIONS:

-a	Slowly adjust the time by sss.fff seconds (fff represents fractions of a second). This adjustment can be positive or negative.Only system admin/ super user can adjust the time.
-s date-string	Sets the time and date to the value specfied in the datestring. The datestr may contain the month names, timezones, 'am', 'pm', etc.
-u	Display (or set) the date in Greenwich Mean Time (GMT-universal time).

Format:

%a	Abbreviated weekday(Tue).
%A	Full weekday(Tuesday).
%b	Abbreviated month name(Jan).
%B	Full month name(January).
%c	Country-specific date and time format..
%D	Date in the format %m/%d/%y.
%j	Julian day of year (001-366).
%n	Insert a new line.
%p	String to indicate a.m. or p.m.
%T	Time in the format %H:%M:%S.
%t	Tab space.
%V	Week number in year (01-52); start week on Monday.

EXAMPLE:

1. date command
   date
   The above command will print
2. To use tab space:
   date +"Date is %D %t Time is %T"
   The above command will remove space and print as To know the week number of the year, date -V
3. To set the date,
   date -s "1/28/2023 11:37:23"

echo

echo command prints the given input string to standard output.

SYNTAX:

The Syntax is

echo [options..] [string]

OPTIONS:

-n	do not output the trailing newline
-e	enable interpretation of the backslash-escaped characters listed below
-E	disable interpretation of those sequences in STRINGs

Without -E, the following sequences are recognized and interpolated:

\NNN	the character whose ASCII code is NNN (octal)
\a	alert (BEL)
\\	backslash
\b	backspace
\c	suppress trailing newline
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab

EXAMPLE:

1. echo command
   echo "CS 4440 OS"
   The above command will print as CS 4440 OS.
2. To use backspace:
   echo -e "CS \4440 \OS"
   The above command will remove space and print as CS4440OS.
3. To use tab space in echo command
   echo -e "CS\t4440\tOS"
   The above command will print as CS 4440 OS

passwd

passwd command is used to change your password.

SYNTAX:

The Syntax is

passwd [options]

OPTIONS:

-a	Show password attributes for all entries.
-l	Locks password entry for name.
-d	Deletes password for name. The login name will not be prompted for password.
-f	Force the user to change password at the next login by expiring the password for name.

EXAMPLE:

1. passwd

   Entering just passwd would allow you to change the password. After entering passwd you will receive the following three prompts:

   Current Password:

   New Password:

   Confirm New Password:

   Each of these prompts must be entered correctly for the password to be successfully changed.

pwd

pwd - Print Working Directory. pwd command prints the full filename of the current working directory.

SYNTAX:

The Syntax is

pwd [options]

OPTIONS:

-P	The pathname printed will not contain symbolic links.
-L	The pathname printed may contain symbolic links.

EXAMPLE:

1. Displays the current working directory.
   pwd
   If you are working in home directory then, pwd command displays the current working directory as /home.

cal

cal command is used to display the calendar.

SYNTAX:

The Syntax is

cal [options] [month] [year]

OPTIONS:

-1	Displays single month as output.
-3	Displays prev/current/next month output.
-s	Displays sunday as the first day of the week.
-m	Displays Monday as the first day of the week.
-j	Displays Julian dates (days one-based, numbered from January 1).
-y	Displays a calendar for the current year.

EXAMPLE:

1. cal

cal command displays the current month calendar.

2. cal -3 5 2023