PRACTICAL:05

AIM: Write a program to print the total waiting time, average waiting time, total turnaround time, average turnaround time & Gantt Chart using Priority CPU scheduling policy.

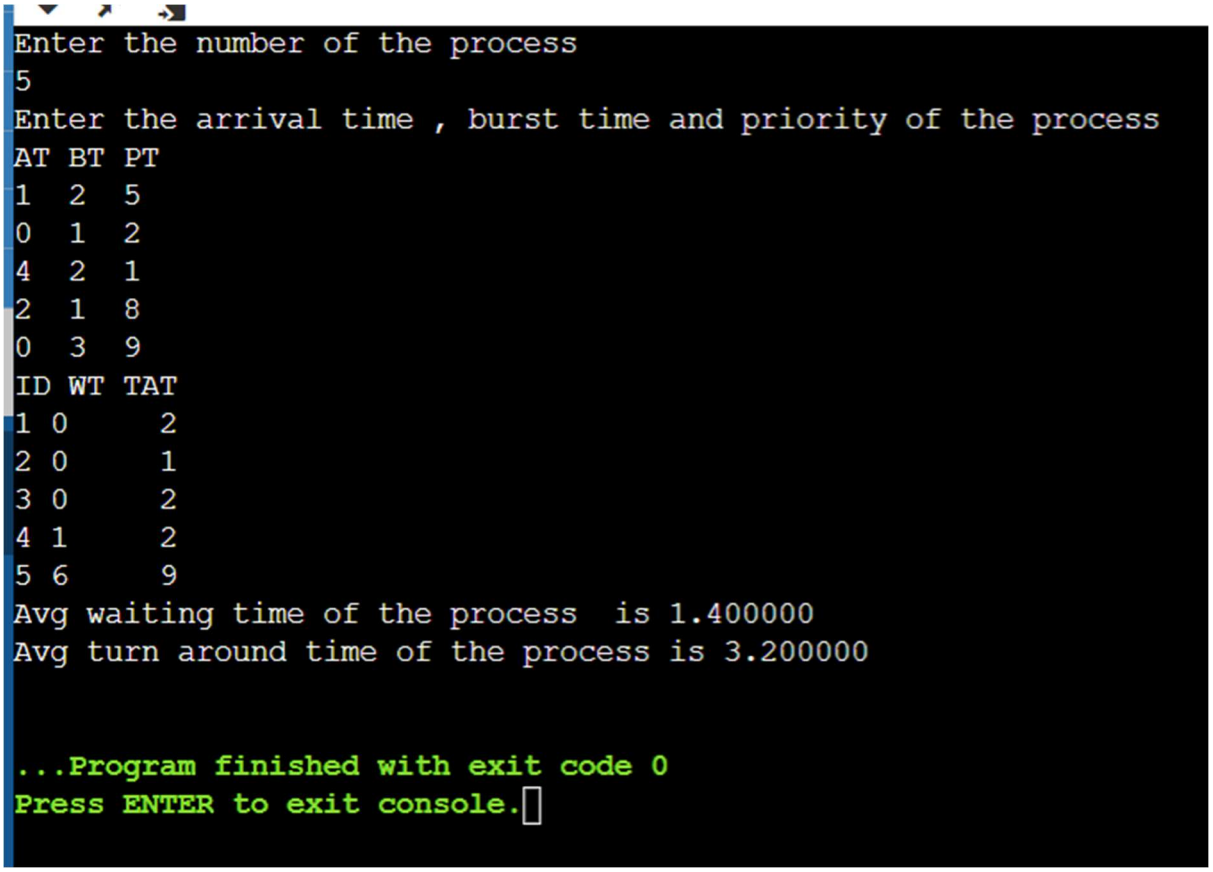
Source code:

```
#include<stdio.h>
struct process
{
   int WT,AT,BT,TAT,PT;
};

struct process a[10];

int main()
{
   int n,temp[10],t,count=0,short_p;
   float total_WT=0,total_TAT=0,Avg_WT,Avg_TAT;
   printf("Enter the number of the process\n");
   scanf("%d",&n);
   printf("Enter the arrival time , burst time and priority of the process\n");
   printf("AT BT PT\n");
   for(int i=0;i<n;i++)
   {
      scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PT);
```

```
    temp[i]=a[i].BT;
}

a[9].PT=10000;

for(t=0;count!=n;t++)
{
    short_p=9;
    for(int i=0;i<n;i++)
    {
        if(a[short_p].PT>a[i].PT && a[i].AT<=t && a[i].BT>0)
        {
            short_p=i;
        }
    }

    a[short_p].BT=a[short_p].BT-1;

    if(a[short_p].BT==0)
    {
        count++;
        a[short_p].WT=t+1-a[short_p].AT-temp[short_p];
        a[short_p].TAT=t+1-a[short_p].AT;
```

```
        total_WT=total_WT+a[short_p].WT;
        total_TAT=total_TAT+a[short_p].TAT;

    }
  }

  Avg_WT=total_WT/n;
  Avg_TAT=total_TAT/n;

  printf("ID WT TAT\n");
  for(int i=0;i<n;i++)
  {
    printf("%d %d\t%d\n",i+1,a[i].WT,a[i].TAT);
  }

  printf("Avg waiting time of the process  is %f\n",Avg_WT);
  printf("Avg turn around time of the process is %f\n",Avg_TAT);

  return 0;
}
```

Output:


Enter the number of the process
5
Enter the arrival time , burst time and priority of the process
AT BT PT
1 2 5
0 1 2
4 2 1
2 1 8
0 3 9
ID WT TAT
1 0 2
2 0 1
3 0 2
4 1 2
5 6 9
Avg waiting time of the process is 1.400000
Avg turn around time of the process is 3.200000
...Program finished with exit code 0
Press ENTER to exit console.

Practical-08

AIM: To study vi-editor in detail.

UNIX: vi Editor

General Introduction:

The vi editor (short for visual editor) is a screen editor which is available on almost all Unix systems. Once you have learned vi, you will find that it is a fast and powerful editor. vi has no menus but instead uses combinations of keystrokes in order to accomplish commands.

Starting vi:

To start using vi, at the Unix prompt type **vi** followed by a file name. If you wish to edit an existing file, type in its name; if you are creating a new file, type in the name you wish to give to the new file.

%vi *filename*

vi's Modes and Moods

vi has two modes: the command mode and the insert mode. It is essential that you know which mode you are in at any given point in time. When you are in command mode, letters of the keyboard will be interpreted as commands. When you are in insert mode the same letters of the keyboard will type or edit text..

General Command Information:

As mentioned previously, vi uses letters as commands. It is important to note that in general vi commands:

- are case sensitive - lowercase and uppercase command letters do different things
- are not displayed on the screen when you type them
- generally do not require a **Return** after you type the command.

You will see some commands which start with a colon (**:**). These commands are *ex* commands which are used by the *ex* editor. *ex* is the true editor which lies underneath vi -- in other words, vi is the interface for the ex editor.

Moving One Character at a Time:

Try using your direction keys to move up, down, left and right in your file. Sometimes, you may find that the direction keys don't work. If that is the case, to move the cursor one character at the time, you may use the **h**, **j**, **k**, and **l** keys. These keys move you in the following directions:

h left one space
l right one space
j down one space
k up one space

If you move the cursor as far as you can in any direction, you may see a screen flash or hear a beep.

Moving among Words and Lines:

While these four keys (or your direction keys) can move you just about anywhere you want to go in your file, there are some shortcut keys that you can use to move a little more quickly through a document. To move more quickly among words, you might use the following:

w moves the cursor forward one word
b moves the cursor backward one word (if in the middle of a word, b will move you to the beginning of the current word).
e moves to the end of a word.

Command Keys and Case:

You will find when using vi that lower case and upper case command keys are interpreted differently. For example, when using the lower case **w**, **b**, and **e** commands, words will be defined by a space or a punctuation mark. On the other hand, **W**, **B**, and **E** commands may be used to move between words also, but these commands ignore punctuation.

Shortcuts:

Two short cuts for moving quickly on a line include the **$** and the **0** (zero) keys. The **$** key will move you to the end of a line, while the **0** will move you quickly to the beginning of a line.

Screen Movement:

To move the cursor to a line within your current screen use the following keys:

H moves the cursor to the top line of the screen.
M moves the cursor to the middle line of the screen.
L moves the cursor to the last line of the screen.

To scroll through the file and see other screens use:

ctrl-f scrolls down one screen
ctrl-b scrolls up one screen
ctrl-u scrolls up a half a screen
ctrl-d scrolls down a half a screen

Two other useful commands for moving quickly from one end to the other of a document are **G** to move to the end of the file and **1G** to move to the beginning of the file. If you precede **G** with a number, you can move to a specific line in the document (e.g. 15G would move you to line 15).

Deleting (or Cutting) Characters, Words, and Lines:

To delete a character, first place your cursor on that character. Then, you may use any of the following commands

x deletes the character under the cursor.
X deletes the character to the left of your cursor.
dw deletes from the character selected to the end of the word.
dd deletes all the current line.
D deletes from the current character to the end of the line.

Preceding the command with a number will delete multiple characters. For example, **10x** will delete the character selected and the next 9 characters; **10X** will delete the 10 characters to the left of the currently selected character. The command **5dw** will delete 5 words, while **4dd** deletes four lines.

Replacing or Changing Characters, Words, and Lines:

When you are using the following commands to replace text, you will be put temporarily into insert mode so that you can change a character, word, line, or paragraph of text.

r replaces the current character with the next character you enter/type. Once you enter the character you are returned to command mode.

R puts you in overtype mode until you hit **ESC** which will then return you to command mode.

cw changes and replaces the current word with text that you type. A dollar sign marks the end of the text you're changing. Pressing **ESC** when you finish will return you to command mode.

Inserting Text:

If you wish to insert new text in a line, first position the cursor to the right of where you wish the inserted text to appear. Type **i** to get into insert mode and then type in the desired text (note that the text is inserted before the cursor). Press **ESC** to return to command mode.

Inserting a Blank Line:

To insert a blank line below the line your cursor is currently located on, use the **o** key and then hit **ESC** to return to the command mode . Use **O** to insert a line above the line the cursor is located on.

Appending Text:

You can use the append command to add text at any place in your file. Append (**a**) works very much like Insert (**i**) except that it insert text *after* the cursor rather than before it. Append is probably used most often for adding text to the end of a line. Simply place your cursor where you wish to append text and press **a.** Once you've finished appending, press **ESC** to go back to command mode.

Joining Lines:

Since vi does not use automatic word wrap, it is not unusual in editing lines to end up with lines that are too short and that might be improved if joined together. To do this, place your cursor on the first line to be joined and type **J**. As with other commands, you can precede **J** with a number to join multiple lines (**4J** joins 4 lines).

Undoing:

Be sure to remember this command. When you make a mistake you can undo it. **DO NOT** move the cursor from the line where you made the change. Then try using one of the following two commands:

u undoes the last change you made anywhere in the file. Using **u** again will "undo the undo".

U undoes all recent changes to the current line. You can not have moved from the line to recover the original line.

Closing and Saving Files

When you edit a file in vi, you are actually editing a copy of the file rather than the original. The following sections describe methods you might use when closing a file, quitting vi, or both.

Quitting and Saving a File:

The command **ZZ** (notice that it is in uppercase) will allow you to quit vi and save the edits made to a file. You will then return to a Unix prompt. Note that you can also use the following commands:

:w to save your file but not quit vi (this is good to do periodically in case of machine crash!).

:q to quit if you haven't made any edits.

:wq to quit and save edits (basically the same as ZZ).

Quitting without Saving Edits:

Sometimes, when you create a mess (when you first start using vi this is easy to do!) you may wish to erase all edits made to the file and either start over or quit. To do this, you can choose from the following two commands:

:e! reads the original file back in so that you can start over.
:q! wipes out all edits and allows you to exit from vi.

More about Combining Commands, Objects, and Numbers

Now that you've learned some basic vi commands you might wish to expand your skills by trying some fancy combination steps. Some commands are generally used in combination with a text object. We've already seen some examples of this. For example, when you use the command **dw** to delete a word, that combines the delete (**d**) command with the word (**w**) text object. When you wish to delete multiple words, you might add a number to this combination. If you wished to delete 2 words you might use **2dw** or **d2w**. Either of these combinations would work. So, as you can see, the general format for a command can be

(number) (command) (text object) **or** *(command) (number) (text object)*

You might wish to try out some of the following combinations of commands and objects:

Command	**Text Object**
d (delete)	**w** (word to the left)
y (yank/copy)	**b** (word to the right or backward)
c (change)	**e** (end of word)
	H (top of the screen)
	L (bottom of the screen)
	M (middle of the screen)
	0 (zero - first character on a line)
	$ (end of a line)
	((previous sentence)
	) (next sentence)
	[(previous section)
	] (next section)

Repeating a Command:

If you are doing repetitive editing, you may wish to use the same command over and over. vi will allow you to use the dot (.) to repeat the last basic command you issued. If for example, you wished to deleted several lines, you could use **dd** and then **.** (dot) in quick succession to delete a few lines.

Useful vi Commands

Cut/Paste Commands:

x	delete one character (destructive backspace)
dw	delete the current word (Note: ndw deletes n numbered words)
dd	delete the current line (Note: ndd deletes n numbered lines)
D	delete all content to the right of the cursor
d$	same as above
:u	undo last command
p,P	paste line starting one line below/above current cursor location
J	combine the contents of two lines
"[a-z]nyy	yank next n lines into named buffer [a-z]
"[a-z]p/P	place the contents of selected buffer below/above the current line

Extensions to the Above Commands:

:3,18d	delete lines 3 through 18
16,25m30	move lines 16 through 25 to after line 30
23,29co62	copy specified lines and place after line 62

Cursor Relocation commands:

:[n]	goto line [n]
shift g	place cursor on last line of text
h/l/j/k	move cursor left, right, down and up
^f/^b	move forward, backward in text, one page
^u/^d	move up, down one half page
$	move to end of line
0	move to beginning of line

Extensions to the Above:

b	move backwards one word (Note: nb moves back n number of words)
e	move to end of current word

(	move to beginning of curent block
)	move to the end of current block

Searching and Substitution commands:

/ [string]	search forward for string
? [string]	search backwards for string
n	repeat last search
N	repeat search in opposite direction
cw	change the contents of the current word, (use ESC to stop replacement mode)
c$	Replace all content to the right of cursor (exit replacement mode with ESC)
c0	Replace all content to the left of cursor (exit with ESC)
:1,$s/s1/s2/g	(Yow!) global replacement of string1 with string2
r	replace current character with next character typed

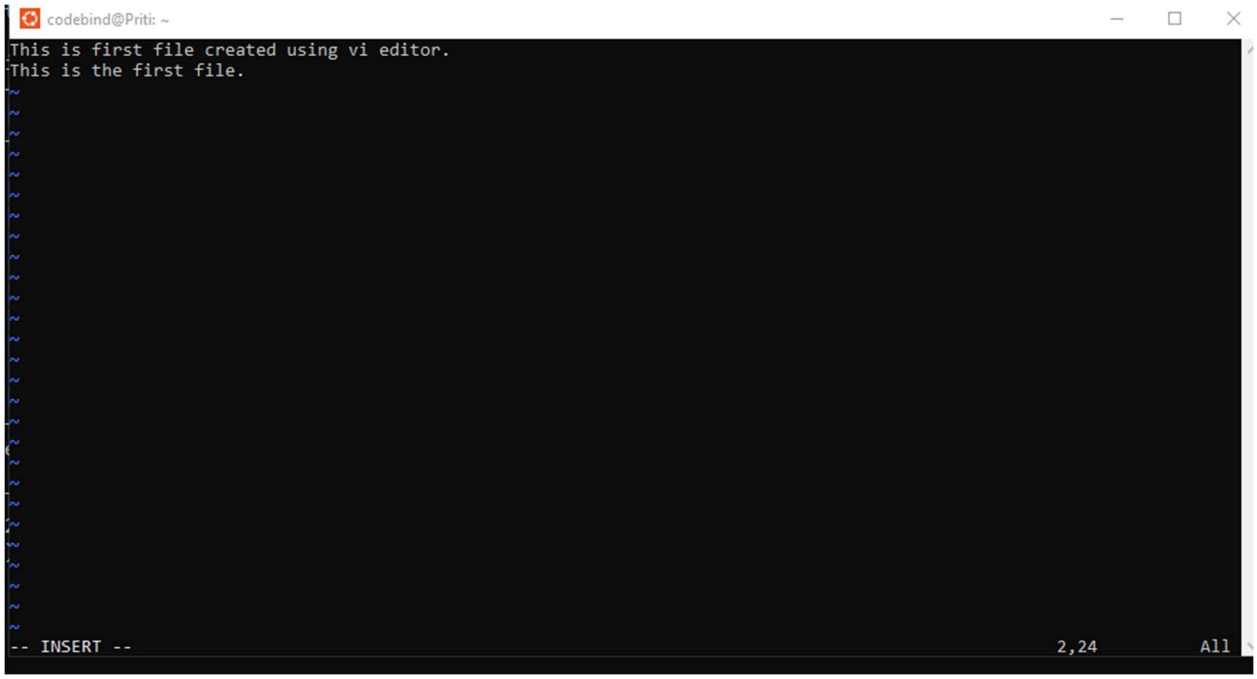
Entering the Insert Mode:

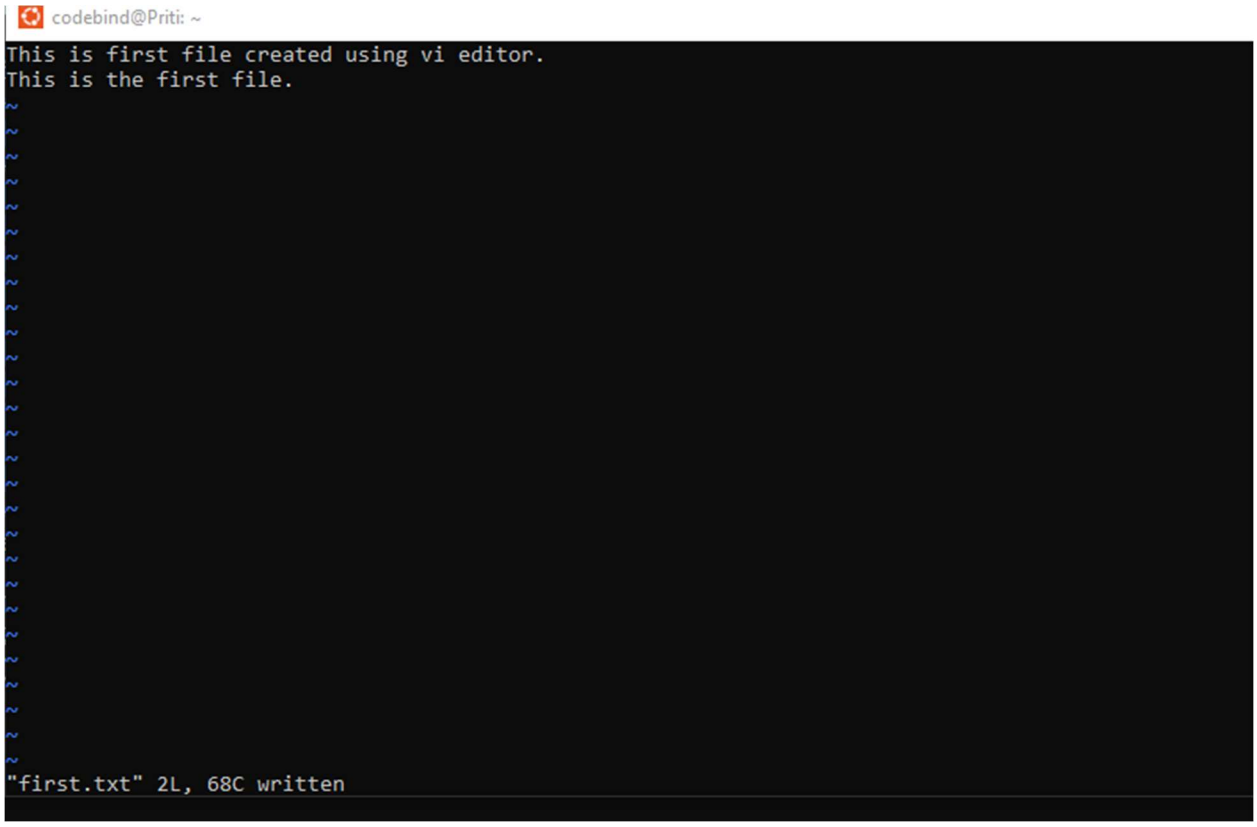
i	Begin inserting text at current cursor location
I	Begin inserting text at the beginning of the current line
a	Begin appending text, one character to the right of current cursor location
A	Begin appending text at the end of the current line
o/O	Begin entering text one line below\above current line
ESC	Exit insertion mode and return to command mode

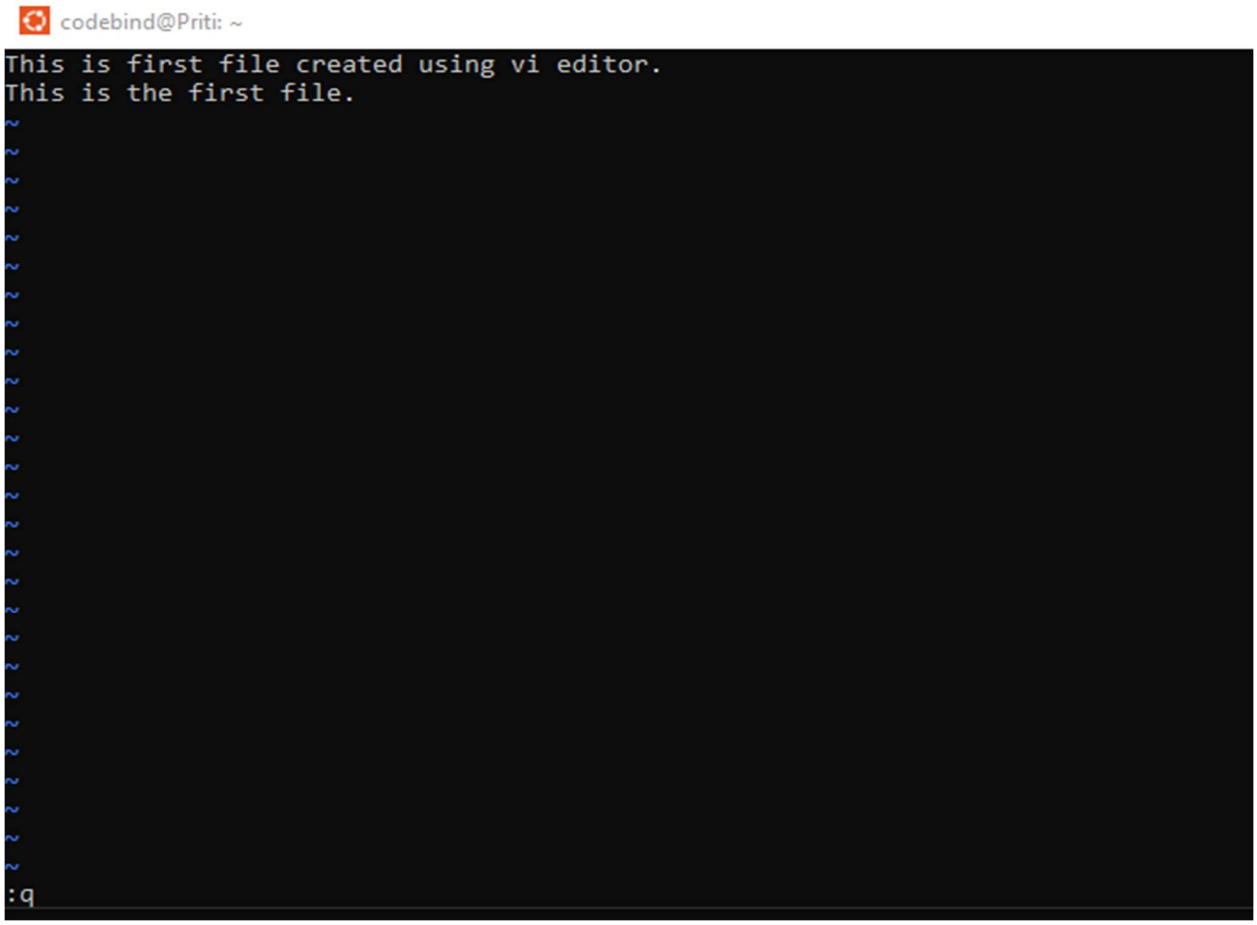
Exiting and Entering VI:

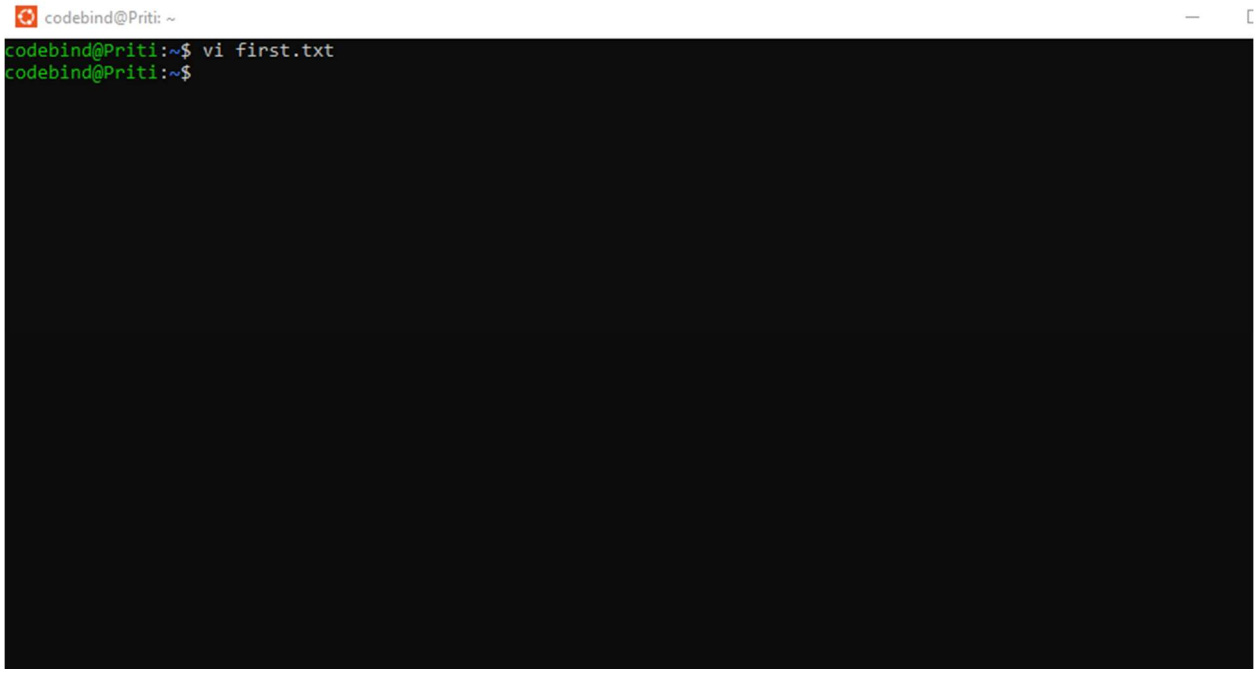
ZZ	save file and exit VI
:wq	same as above
:e!	return to last saved version of current file
:q	quit without save, (Note :q! is required if changes have been made)
:w	write without exit (:w! to force write)

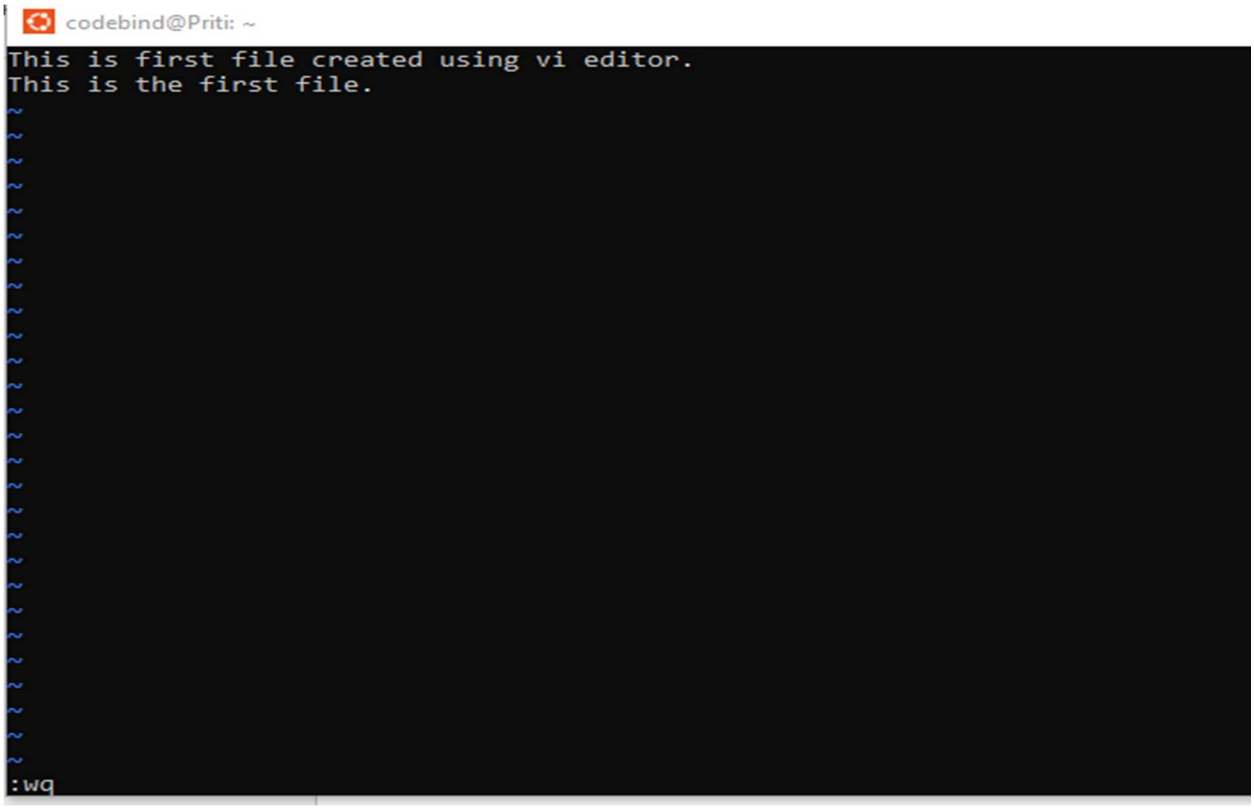
Implementation:

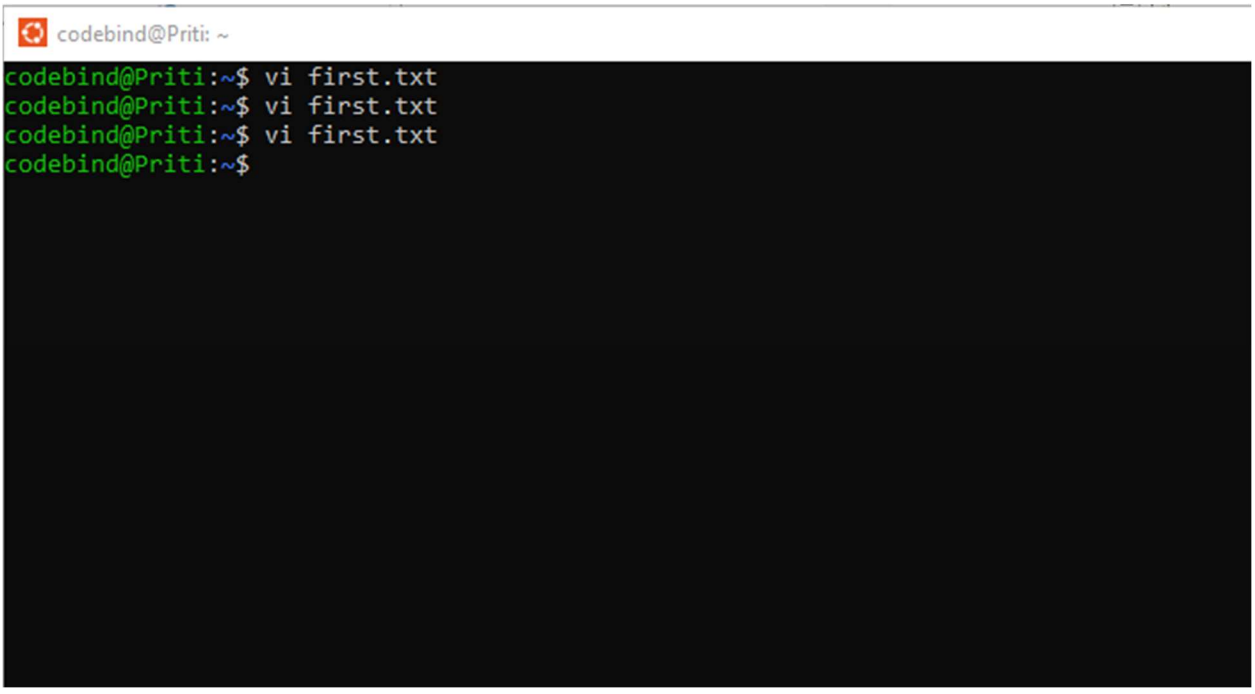





codebind@Priti: ~
This is first file created using vi editor.
This is the first file.
:q


codebind@Priti: ~
codebind@Priti:~$ vi first.txt
codebind@Priti:~$


codebind@Priti: ~
This is first file created using vi editor.
This is the first file.
:wq


codebind@Priti: ~
codebind@Priti:~$ vi first.txt
codebind@Priti:~$ vi first.txt
codebind@Priti:~$ vi first.txt
codebind@Priti:~$

PRACTICAL:06

AIM: Write a program to print the total waiting time, average waiting time, total turnaround time, average turnaround time & Gantt Chart using Round Robin CPU scheduling policy.

Source code:

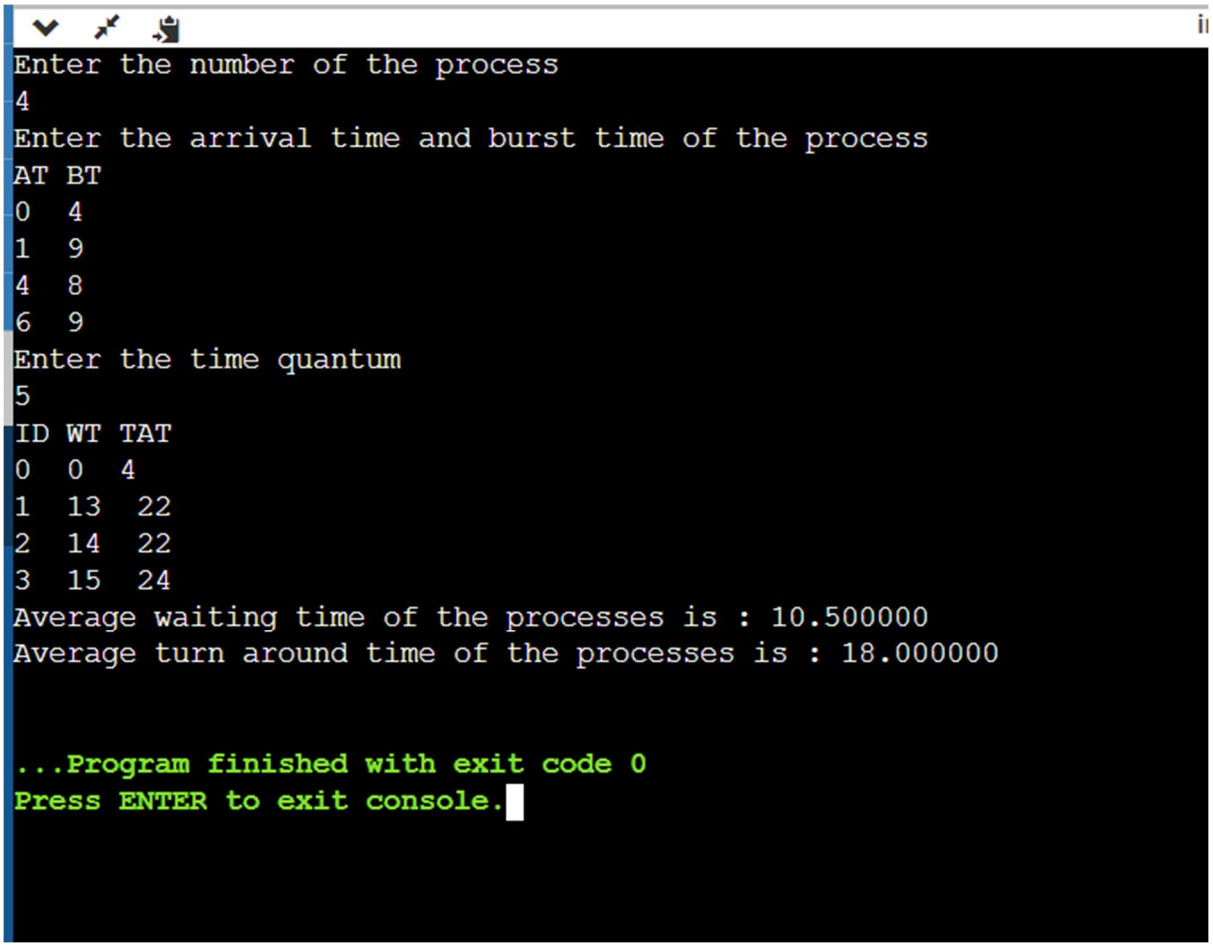
```
#include<stdio.h>
struct process
{
   int id,AT,BT,WT,TAT;
};

struct process a[10];
int queue[100];
int front=-1;
int rear=-1;
void insert(int n)
{
   if(front==-1)
    front=0;
   rear=rear+1;
   queue[rear]=n;
}
int delete()
{
   int n;
   n=queue[front];
```

```
    front=front+1;
    return n;
}
int main()
{
    int n,TQ,p,TIME=0;
    int temp[10],exist[10]={0};
    float total_wt=0,total_tat=0,Avg_WT,Avg_TAT;
    printf("Enter the number of the process\n");
    scanf("%d",&n);
    printf("Enter the arrival time and burst time of the process\n");
    printf("AT BT\n");
    for(int i=0;i<n;i++)
    {
        scanf("%d%d",&a[i].AT,&a[i].BT);
        a[i].id=i;
        temp[i]=a[i].BT;
    }
    printf("Enter the time quantum\n");
    scanf("%d",&TQ);
    insert(0);
    exist[0]=1;
    while(front<=rear)
    {
```

```
p=delete();
if(a[p].BT>=TQ)
{
   a[p].BT=a[p].BT-TQ;
   TIME=TIME+TQ;
}
else
{
   TIME=TIME+a[p].BT;
   a[p].BT=0;
}
for(int i=0;i<n;i++)
{
   if(exist[i]==0 && a[i].AT<=TIME)
   {
      insert(i);
      exist[i]=1;
   }
}
if(a[p].BT==0)
{
   a[p].TAT=TIME-a[p].AT;
   a[p].WT=a[p].TAT-temp[p];
   total_tat=total_tat+a[p].TAT;
```

```
            total_wt=total_wt+a[p].WT;
        }
        else
        {
            insert(p);
        }
    }

    Avg_TAT=total_tat/n;
    Avg_WT=total_wt/n;
    printf("ID WT TAT\n");
    for(int i=0;i<n;i++)
    {
        printf("%d  %d  %d\n",a[i].id,a[i].WT,a[i].TAT);
    }
    printf("Average waiting time of the processes is : %f\n",Avg_WT);
    printf("Average turn around time of the processes is : %f\n",Avg_TAT);
    return 0;
}
```

Output:


Enter the number of the process
4
Enter the arrival time and burst time of the process
AT BT
0 4
1 9
4 8
6 9
Enter the time quantum
5
ID WT TAT
0 0 4
1 13 22
2 14 22
3 15 24
Average waiting time of the processes is : 10.500000
Average turn around time of the processes is : 18.000000
...Program finished with exit code 0
Press ENTER to exit console.

Practical –7

AIM: - To study various filters.

1. WC

wc (short for **w**ord **c**ount) is a command in Unix-like operating systems.

The program reads either standard input or a list of files and generates one or more of the following statistics: newline count, word count, and byte count. If a list of files is provided, both individual file and total statistics follow.

Sample execution of wc:

$ wc foo bar

```
  40    149    947 foo
2294  16638  97724 bar
2334  16787  98671 total
```

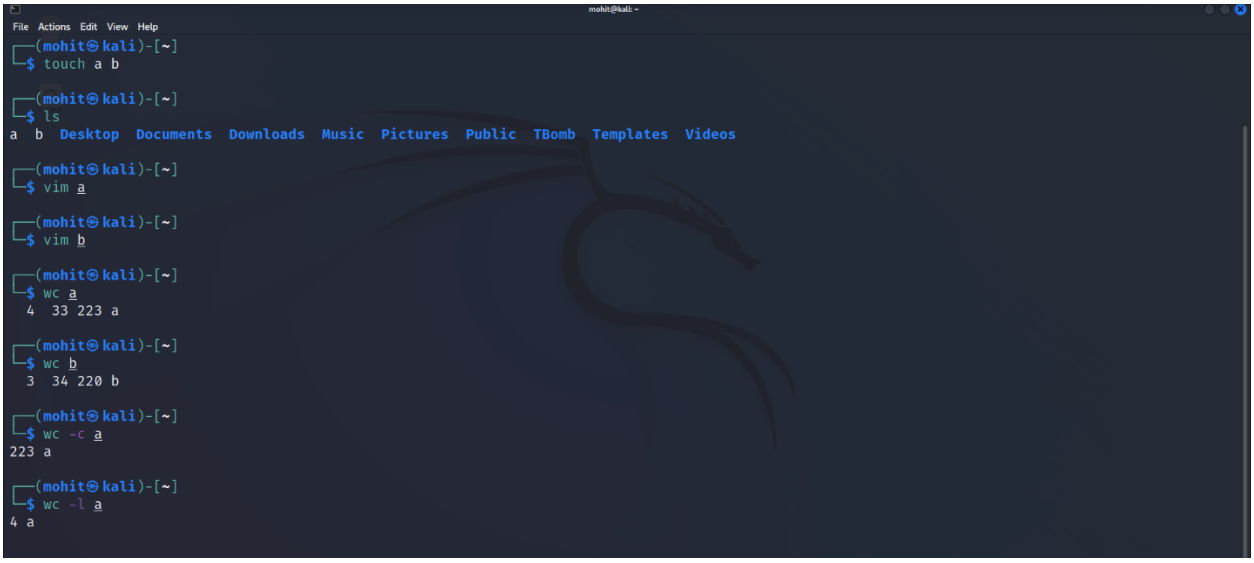
The first column is the count of newlines, meaning that the text file foo has 40 newlines while bar has 2294 newlines- resulting in a total of 2334 newlines. The second column indicates the number of words in each text file showing that there are 149 words in foo and 16638 words in bar- giving a total of 16787 words. The last column indicates the number of characters in each text file, meaning that the file foo has 947 characters while bar has 97724 characters- 98671 characters all in all.

Newer versions of wc can differentiate between byte and character count. This difference arises with Unicode which includes multi-byte characters. The desired behaviour is selected with the -c or -m switch.

GNU wc used to be part of the GNU textutils package; it is now part of GNU coreutils.

options

- wc -l <filename> print the line count
- wc -c <filename> print the byte count
- wc -m <filename> print the character count
- wc -L <filename> print the length of longest line
- wc -w <filename> print the word count



2.CMP

cmp is a command line utility for computer systems that use Unix or a Unix-like operating system. It compares two files of any type and writes the results to the standard output. By default, cmp is silent if the files are the same; if they differ, the byte and line number at which the first difference occurred is reported.

options

-b, --print-bytes
: Print the differing bytes. Display control bytes as a '^' followed by a letter of the alphabet and precede bytes that have the high bit set with 'M-' (which stands for "meta").

-i SKIP, --ignore-initial=SKIP
: Skip the first SKIP bytes of input.

-i SKIP1:SKIP2, --ignore-initial=SKIP1:SKIP2
: Skip the first SKIP1 bytes of FILE1 and the first SKIP2 bytes of FILE2.

-l, --verbose
: Output the (decimal) byte numbers and (octal) values of all differing bytes, instead of the default standard output. Also, output the EOF message if one file is shorter than the other.

-n LIMIT, --bytes=LIMIT
: Compare at most LIMIT bytes.

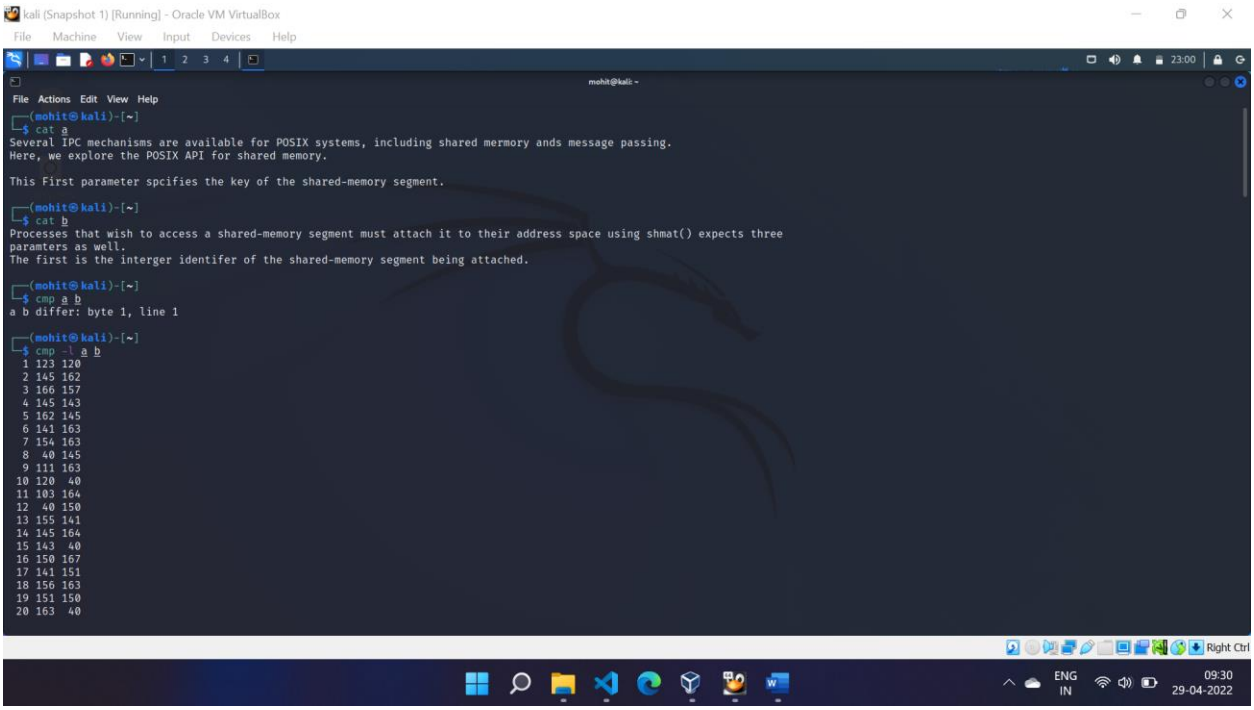
-s, --quiet, --silent
: Output nothing; yield exit status only.

-v, --version
: Output version info.

--help
: Outputs a help file.

Return values

- 0 — files are identical
- 1 — files differ
- 2 — inaccessible or missing argument



3. DIFF

SYNOPSIS

Tag	Description
filterdiff [-i *PATTERN*] [-p *n*] [--strip=*n*] [--addprefix=*PREFIX*] [-x *PATTERN*] [--verbose] [-v] [-z] [[-# *RANGE*] [--hunks=*RANGE*]] [--lines=*RANGE*] [--files=*RANGE*] [--annotate] [--format=*FORMAT*] [--as-numbered-lines=*WHEN*] [--remove-timestamps] [*file*...] **filterdiff** {[--help] [--version] [--list] [--grep ...]}	

DESCRIPTION

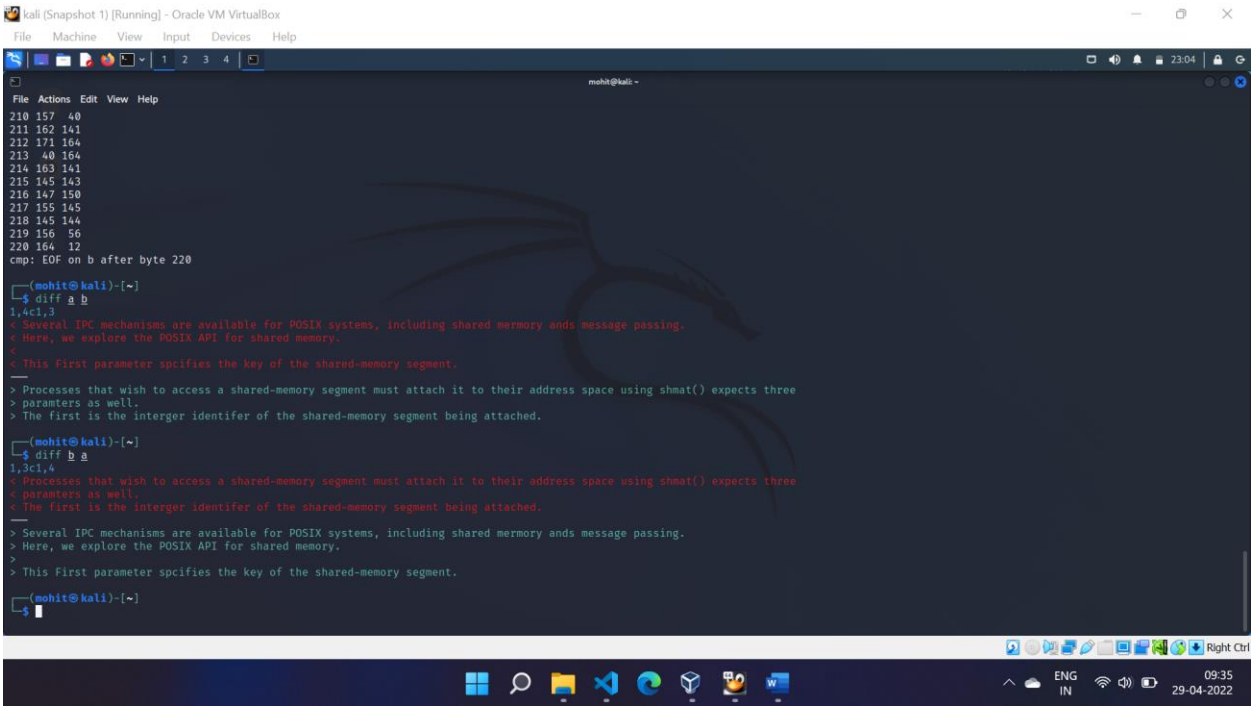
You can use filterdiff to obtain a patch that applies to files matching the shell wildcard *PATTERN* from a larger collection of patches. For example, to see the patches in *patch-2.4.3.gz* that apply to all files called *lp.c*:

filterdiff -z -i '*/lp.c' patch-2.4.3.gz

If neither **-i** nor **-x** options are given, **-i '*'** is assumed. This way **filterdiff** can be used to clean up an existing diff file, removing redundant lines from the beginning (eg. the text from the mail body) or between the chunks (eg. in CVS diffs). To extract pure patch data, use a command like this:

filterdiff message-with-diff-in-the-body > patch

Note that the interpretation of the shell wildcard pattern does not count slash characters or periods as special (in other words, no flags are given to **fnmatch**). This is so that \(lq*/basename\(rq-type patterns can be given without limiting the number of pathname components. You can use both unified and context format diffs with this program.



4. COMM

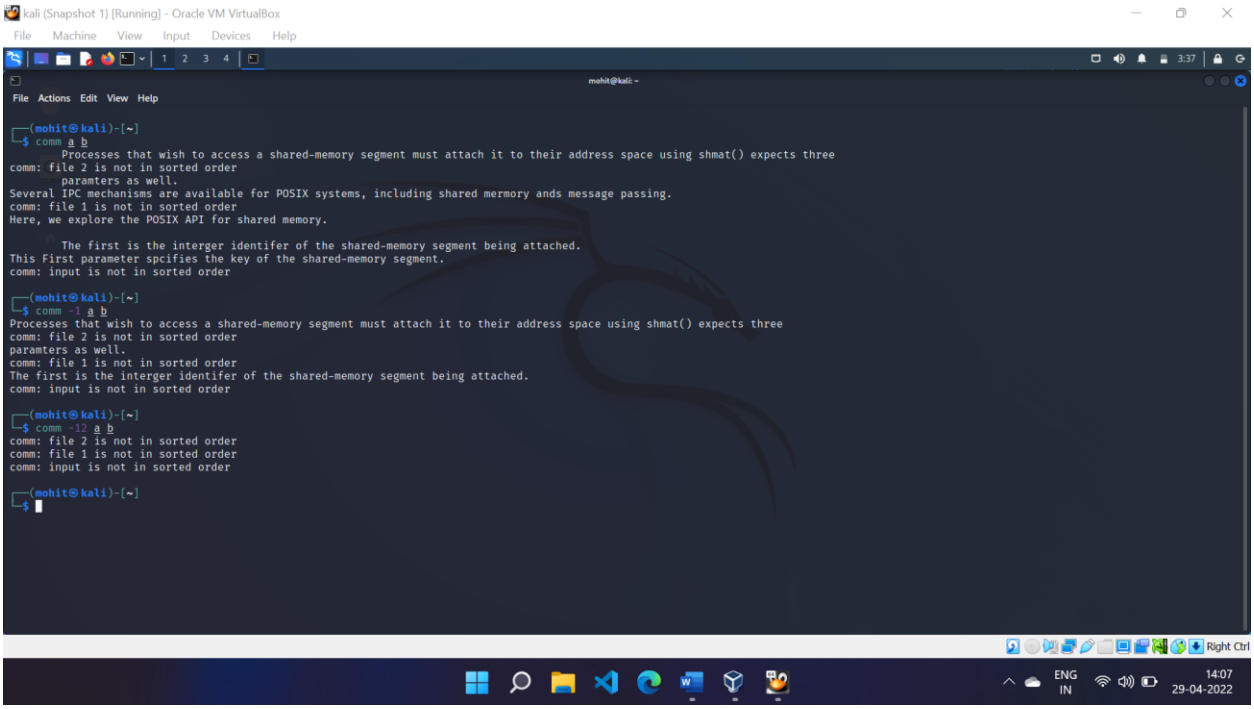
comm reads two files as input, regarded as lines of text. comm outputs one file, which contains three columns. The first two columns contain lines unique to the first and second file, respectively. The last column contains lines common to both. This functionally is similar to diff.

Return code:-Unlike diff, the return code from comm has no logical significance concerning the relationship of the two files. A return code of 0 indicates success, a return code >0 indicates an error occurred during processing.

Comparison to diff

In general terms, diff is a more powerful utility than comm. The simpler comm is best suited for use in scripts.The primary distinction between comm and diff is that comm discards information about the order of the lines prior to sorting.

A minor difference between comm and diff is that comm will not try to indicate that a line has "changed" between the two files; lines are either shown in the "from file #1", "from file #2", or "in both" columns. This can be useful if one wishes two lines to be considered different even if they only have subtle differences.



5. HEAD

head is a program on Unix and Unix-like systems used to display the first few lines of a text file or piped data. The command syntax is:

head [options] <file_name>

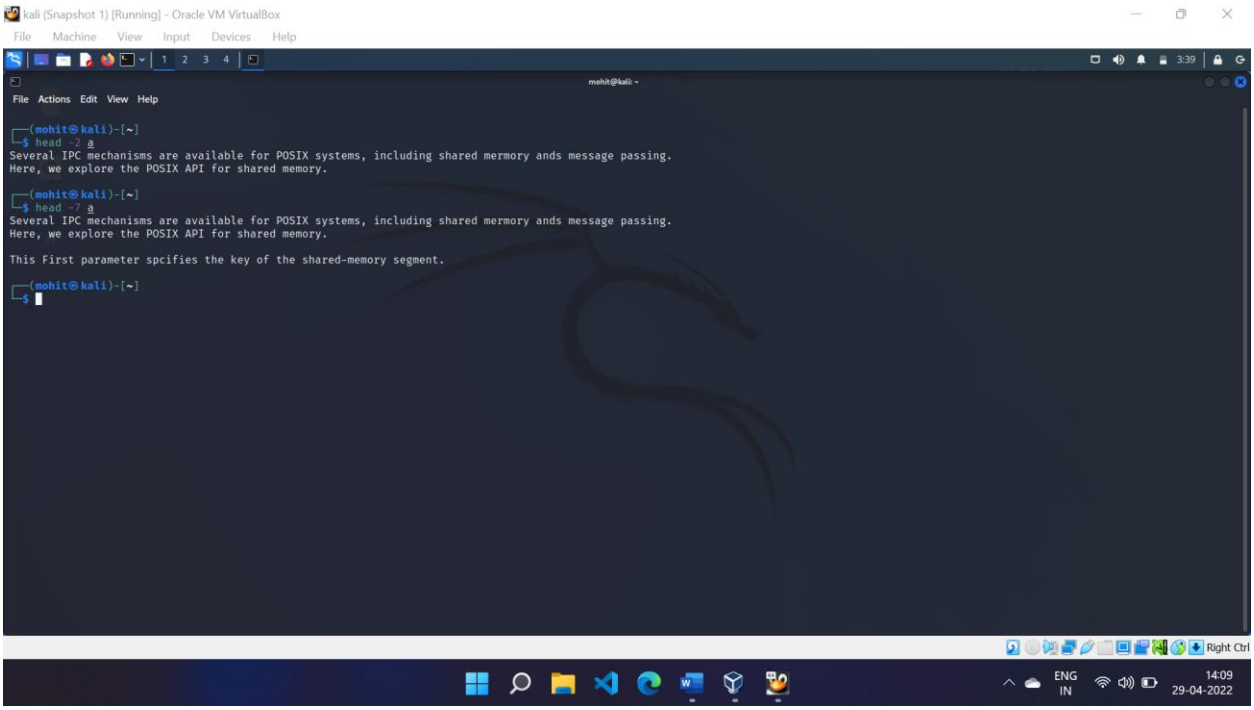
By default, head will print the first 10 lines of its input to the standard output. The number of lines printed may be changed with a command line option. The following example shows the first 20 lines of *filename*:

head -n 20 *filename*

This displays the first 5 lines of all files starting with *foo*:

head -n 5 *foo**

Some versions omit the n and just let you say -5.



6. TAIL

tail is a program on Unix and Unix-like systems used to display the last few lines of a text file or piped data.

Syntax:- **tail [options] <file_name>**

By default, tail will print the last 10 lines of its input to the standard output.

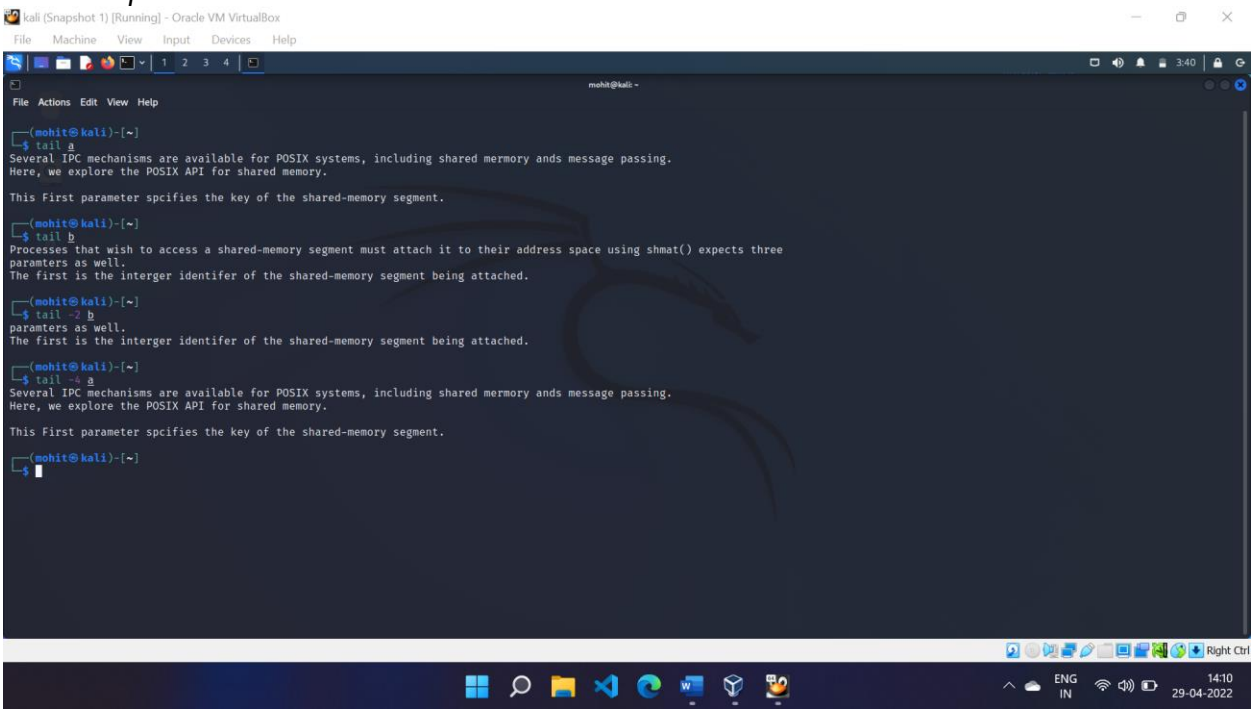
tail -n 20 *filename*

This example shows the last 15 bytes of all files starting with *foo*:

tail -c 15 foo*

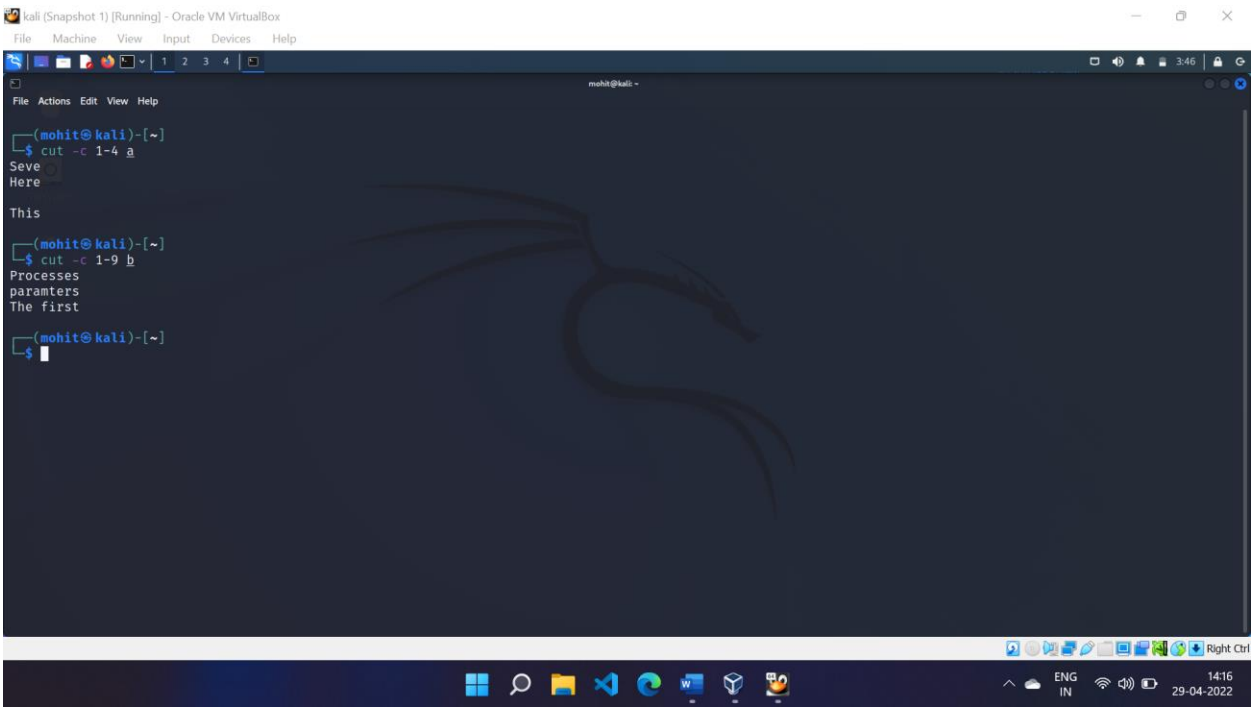
This example shows all lines of *filename* from the second line onwards:

tail -n +2 *filename*



7. CUT

In computing, **cut** is a Unix command line utility which is used to extract sections from each line of input — usually from a fileExtraction of line segments can typically be done by bytes (-b), characters (-c), or fields (-f) separated by a delimiter (-d — the tab character by default).



8. PASTE

paste [-s] [-d delim-list] [--serial] [--delimiters=delim-list] [--help] [--version] [file...]

paste prints lines consisting of sequentially corresponding lines of each specified file. In the output the original lines are separated by TABs. The output line is terminated with a newline.

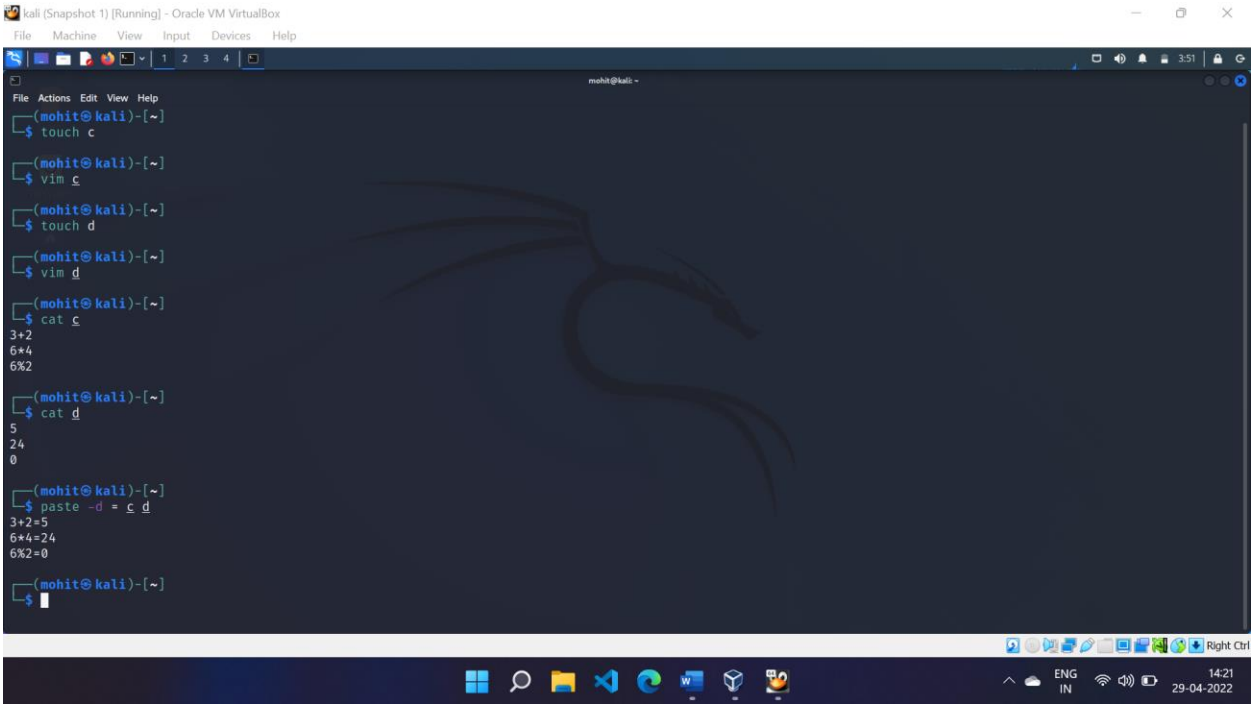
OPTIONS

-s, --serial
Paste the lines of one file at a time rather than one line from each file.

-d, --delimiters delim-list
Consecutively use the characters in delim-list instead of TAB to separate merged lines. When delim-list is exhausted, start again at its beginning.

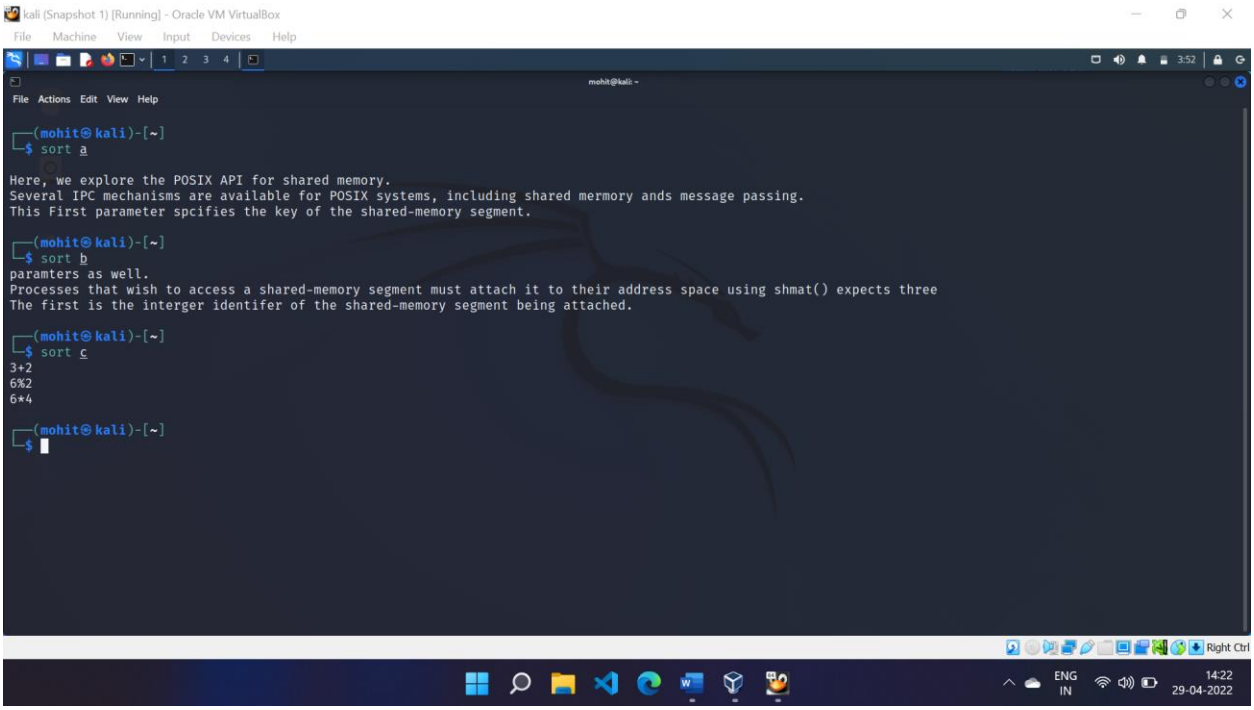
--help
Print a usage message and exit with a status code indicating success.

--version
Print version information on standard output then exit.



9. SORT

In Unix-like operating systems, **sort** is a standard command line program that prints the lines of its input or concatenation of all files listed in its argument list in sorted order. Sorting is done based on one or more sort keys extracted from each line of input. By default, the entire input is taken as sort key. Blank space is the default field separator.



10. NL

nl is a Unix utility for numbering lines, either from a file or from standard input, reproducing output on standard output.

It has a number of switches:

- a - number all lines
- t - number lines with printable text only
- n - no line numbering
- string - number only those lines containing the regular expression defined in the *string* supplied.

The default applied switch is *t*.

nl also supports a number of command line options.

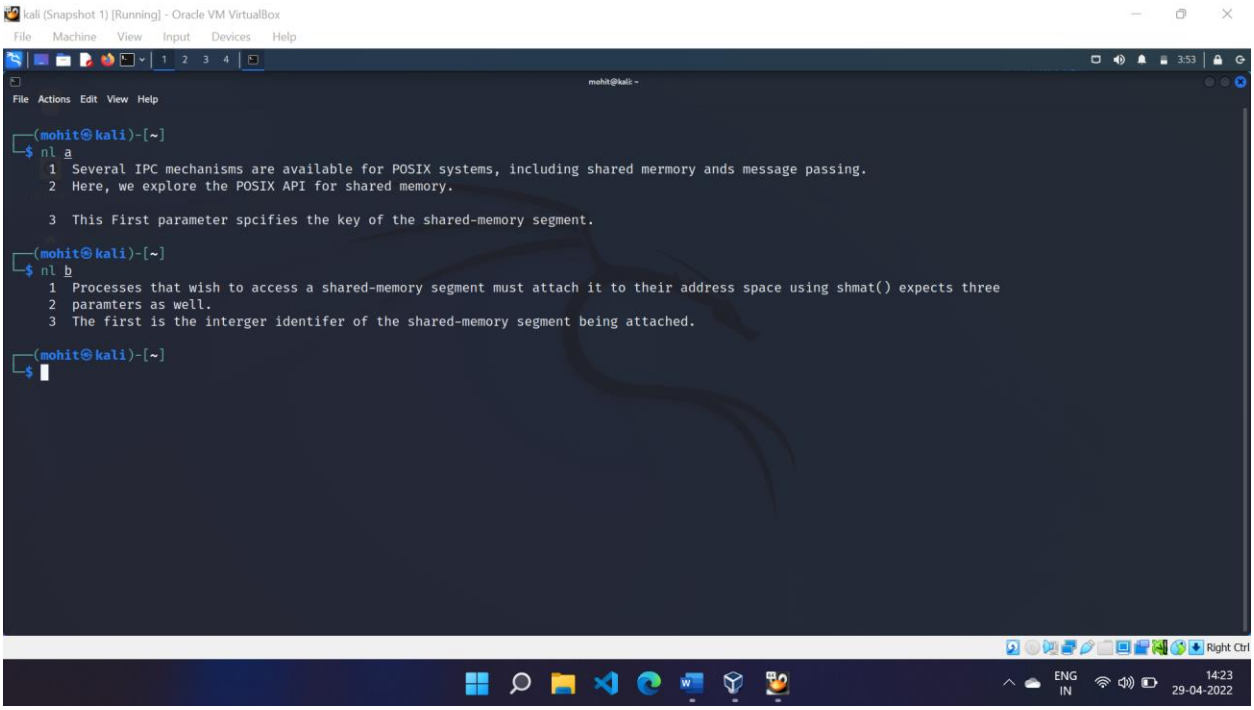
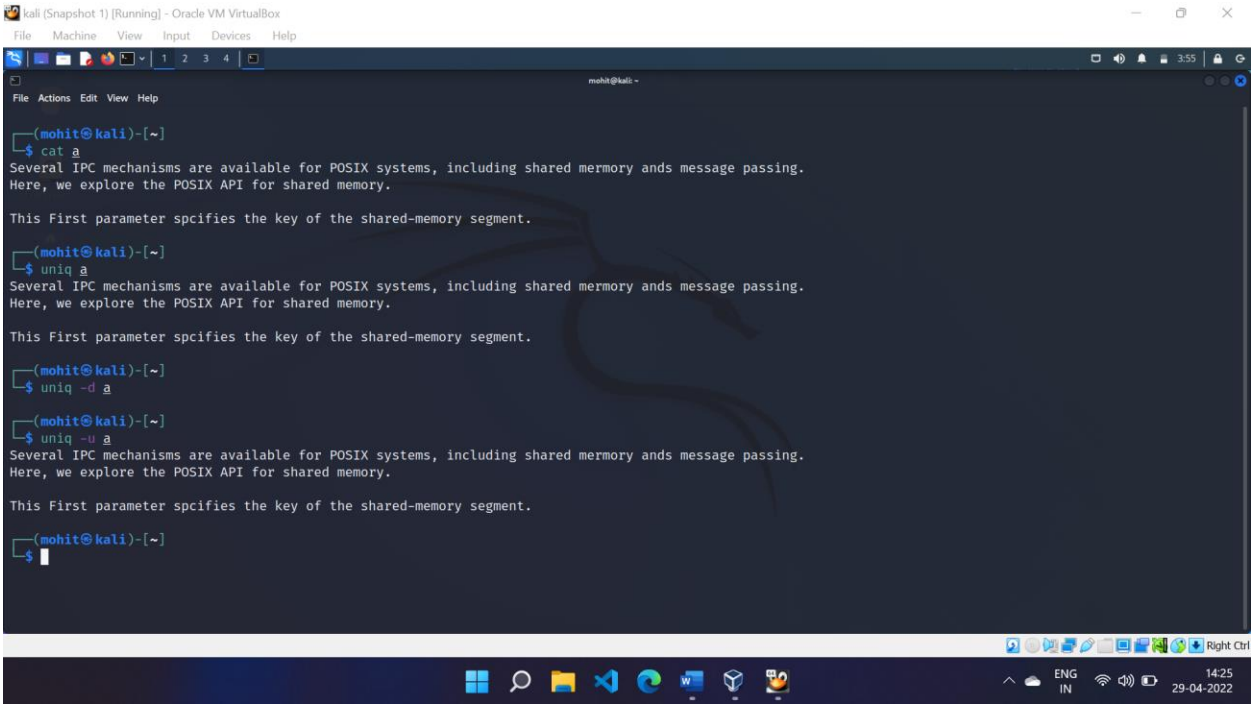


Fig:-nl filter

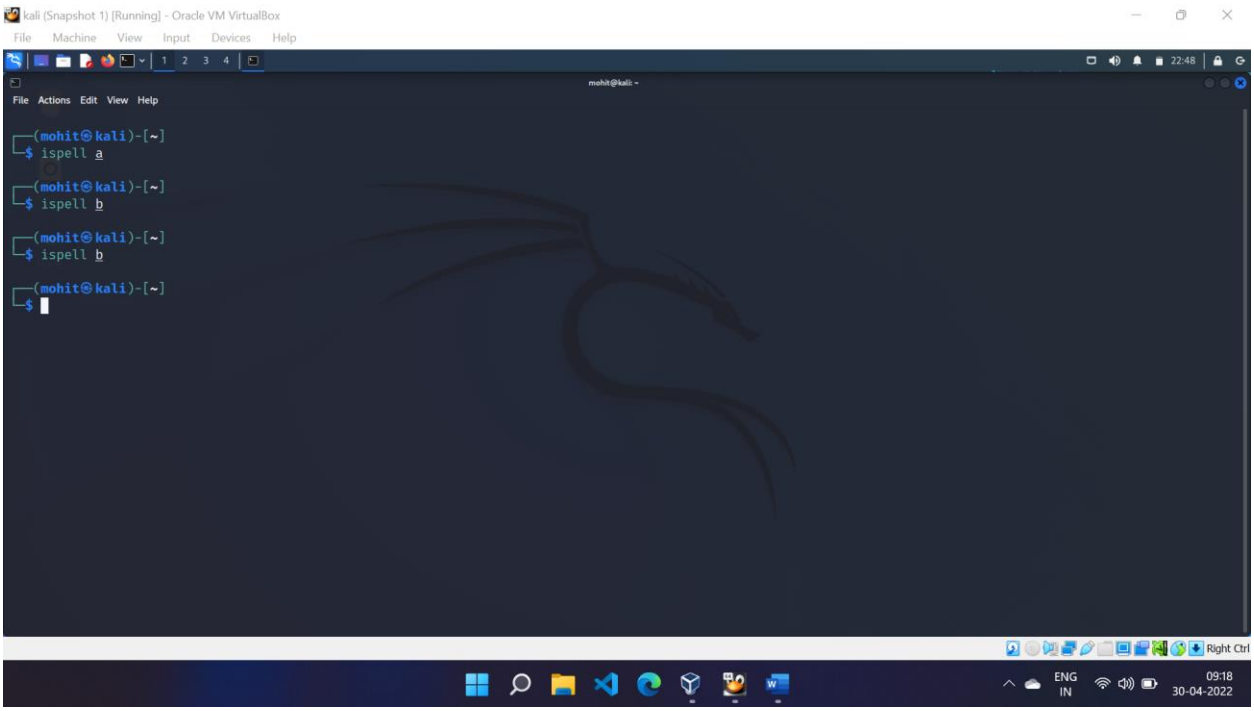
11. UNIQ

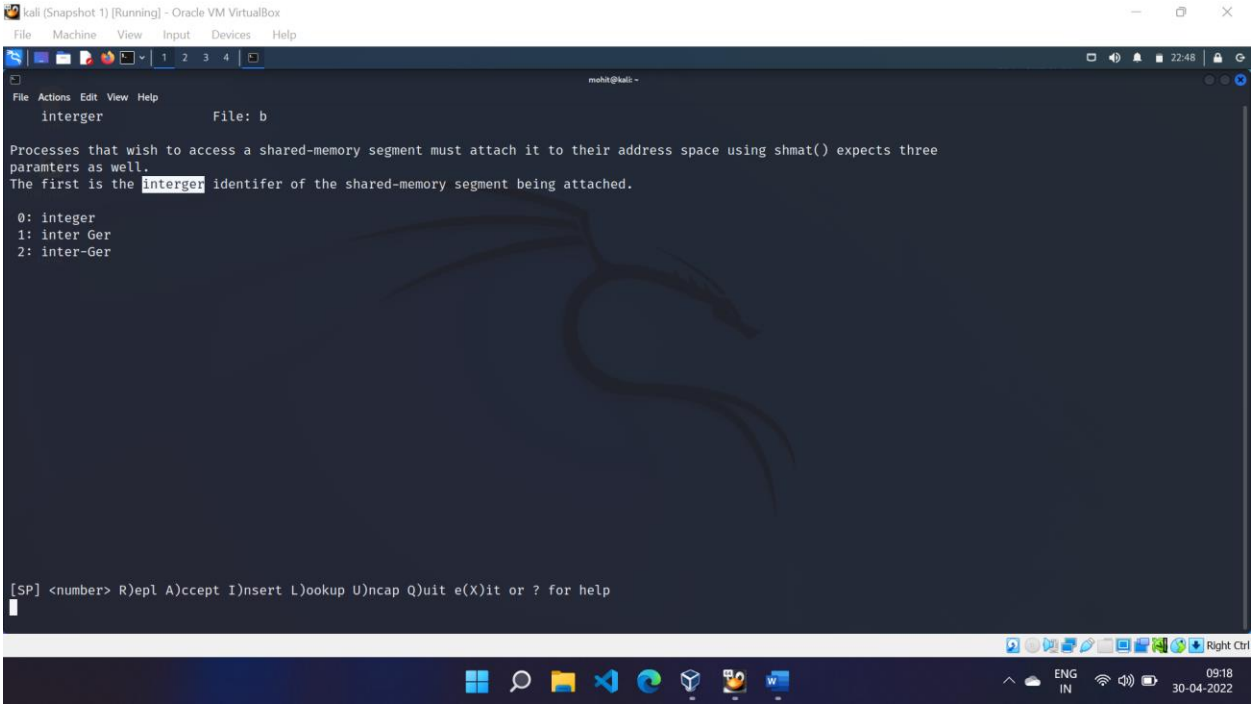
uniq is a Unix utility which, when fed a text file, outputs the file with adjacent identical lines collapsed to one. It is a kind of filter program. Typically it is used after sort. It can also output only the duplicate lines (with the -d option), or add the number of occurrences of each line (with the -c option).



12. SPELL

spell compares words typed in at the keyboard or the contents of a text file to those in a built-in dictionary file and then writes all of the words not in the dictionary (and, therefore, presumably misspelled) to standard output (which is the display screen by default).




kali (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
mohit@kali: ~
File Actions Edit View Help
interger File: b
Processes that wish to access a shared-memory segment must attach it to their address space using shmat() expects three
paramters as well.
The first is the interger identifer of the shared-memory segment being attached.
0: integer
1: inter Ger
2: inter-Ger
[SP] <number> R)epl A)ccept I)nsert L)ookup U)ncap Q)uit e(X)it or ? for help
Right Ctrl
ENG
IN
09:18
30-04-2022

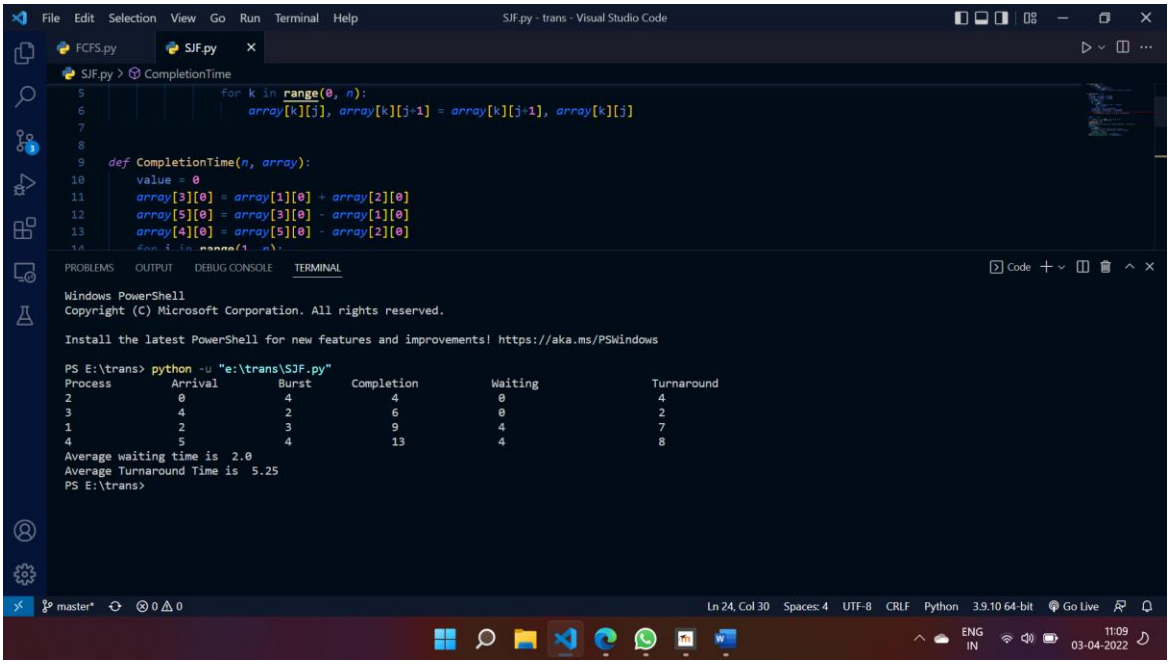
Practical-4

Aim: - List of processes / jobs along with their arrival times & CPU burst times is given. Write a program to print the total waiting time, average waiting time, total turnaround time, average turnaround time & Gantt Chart using Shortest Job First (SJF) CPU scheduling policy.

Code:-

```
def arrangeArrival(n, array):
    for i in range(0, n):
        for j in range(i, n-i-1):
            if array[1][j] > array[1][j+1]:
                for k in range(0, n):
                    array[k][j], array[k][j+1] = array[k][j+1], array[k][j]


def CompletionTime(n, array):
    value = 0
    array[3][0] = array[1][0] + array[2][0]
    array[5][0] = array[3][0] - array[1][0]
    array[4][0] = array[5][0] - array[2][0]
    for i in range(1, n):
        temp = array[3][i-1]
        mini = array[2][i]
        for j in range(i, n):
            if temp >= array[1][j] and mini >= array[2][j]:
                mini = array[2][j]
                value = j
        array[3][value] = temp + array[2][value]
        array[5][value] = array[3][value] - array[1][value]
```

```
                array[4][value] = array[5][value] - array[2][value]
                for k in range(0, 6):
                        array[k][value], array[k][i] = array[k][i], array[k][value]
if __name__ == '__main__':
        n = 4
        arr = [[int(i) for i in range(1, n+1)], [2, 0, 4, 5],
                [3, 4, 2, 4], [0]*n, [0]*n, [0]*n]
        arrangeArrival(n, arr)
        CompletionTime(n, arr)
        print("Process \tArrival\t\tBurst\t  Completion\t\tWaiting  \t\tTurnaround ")
        waitingtime = 0
        turaroundtime = 0
        for i in range(0, n):
                print(arr[0][i], "\t\t", arr[1][i], "\t\t", arr[2][i],
                        "\t\t", arr[3][i], "\t\t", arr[4][i], "\t\t\t", arr[5][i])
                waitingtime += arr[4][i]
                turaroundtime += arr[5][i]
        print("Average waiting time is ", (waitingtime/n))
        print("Average Turnaround Time is ", (turaroundtime/n))
```



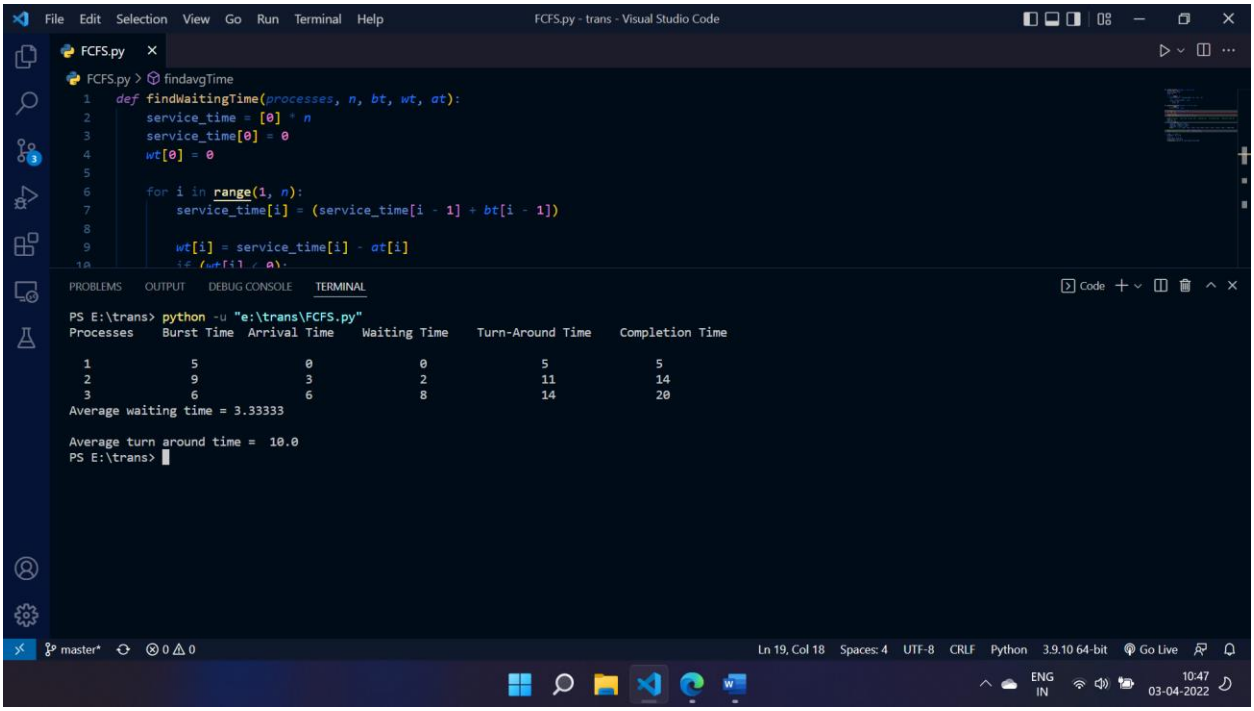
Practical-3

Aim: - List of processes / jobs along with their arrival times & CPU burst times is given. Write a program to print the total waiting time, average waiting time, total turnaround time, average turnaround time & Gantt Chart using First Come First Serve(FCFS) CPU scheduling policy.

Code:-

```
def findWaitingTime(processes, n, bt, wt, at):
        service_time = [0] * n
        service_time[0] = 0
        wt[0] = 0
        for i in range(1, n):
                service_time[i] = (service_time[i - 1] + bt[i - 1])
                wt[i] = service_time[i] - at[i]
                if (wt[i] < 0):
                        wt[i] = 0
def findTurnAroundTime(processes, n, bt, wt, tat):
        for i in range(n):
                tat[i] = bt[i] + wt[i]

def findavgTime(processes, n, bt, at):
        wt = [0] * n
        tat = [0] * n
        findWaitingTime(processes, n, bt, wt, at)
        findTurnAroundTime(processes, n, bt, wt, tat)
        print("Processes    Burst Time  Arrival Time    Waiting Time    Turn-Around Time    Completion
Time \n")
        total_wt = 0
        total_tat = 0
        for i in range(n):
                total_wt = total_wt + wt[i]
                total_tat = total_tat + tat[i]
```

```
            compl_time = tat[i] + at[i]
            print(" ", i + 1, "\t\t", bt[i], "\t\t", at[i],"\t\t", wt[i], "\t\t ", tat[i], "\t\t ",  compl_time)
        print("Average waiting time = %.5f "%(total_wt /n))
        print("\nAverage turn around time = ", total_tat / n)
if __name__ =="__main__":
        processes = [1, 2, 3]
        n = 3
        burst_time = [5, 9, 6]
        arrival_time = [0, 3, 6]
        findavgTime(processes, n, burst_time,arrival_time)
```



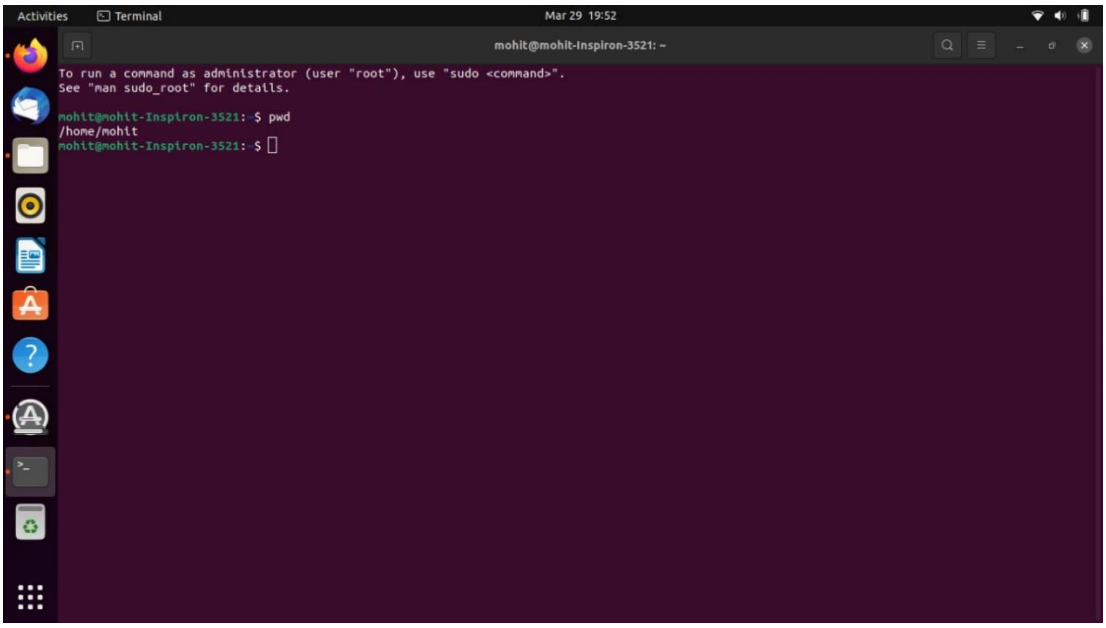
Practical-2

Aim: - To study various file access commands.

Commands:-

1. pwd:-

pwd (print working directory) command displays your location currently you are working on. It will give the whole path starting from the root ending to the directory.



2. cat:-

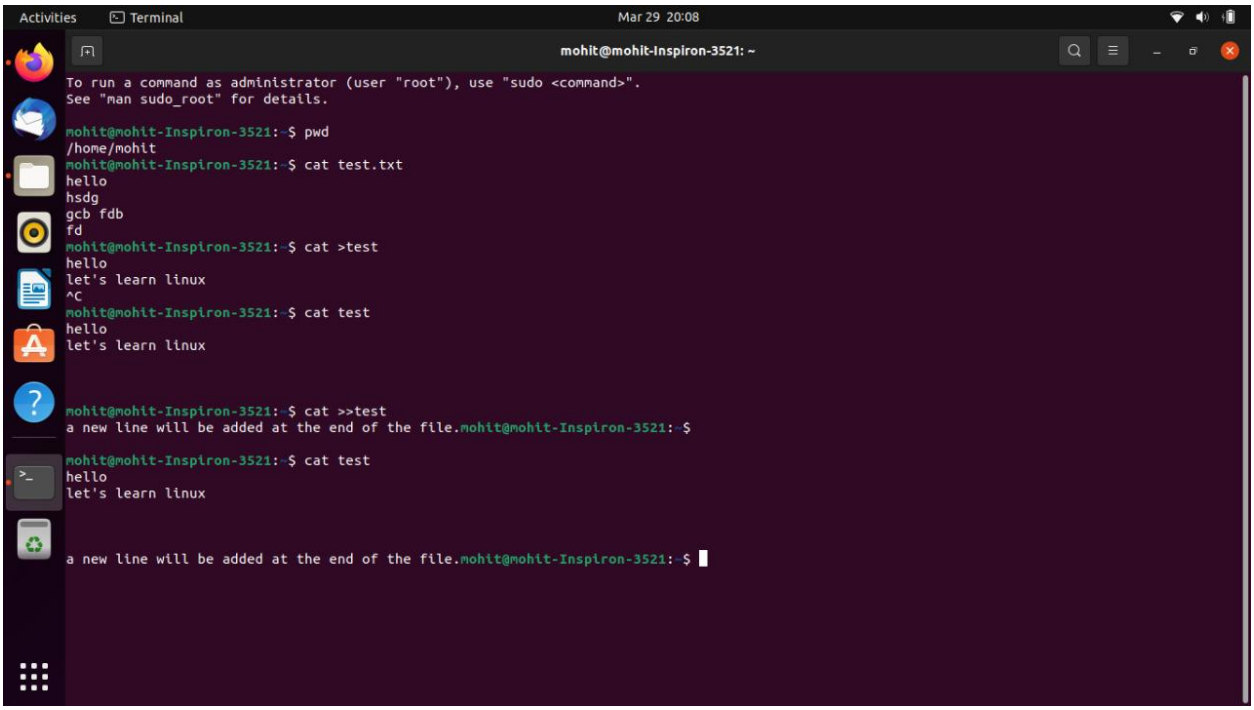
The 'cat' command can be used to display the content of a file.
The 'cat' command is the most universal and powerful tool. It is considered to be one of the most frequently used commands. It can be used to display the content of a file, copy content from one file to another, concatenate the contents of multiple files, display the line number, display $ at the end of the line, etc.

cat command (to create a file)

syntax: - cat **>** **<file** name**>**

To Append the Content of A File

syntax: - cat **>>** (file name)



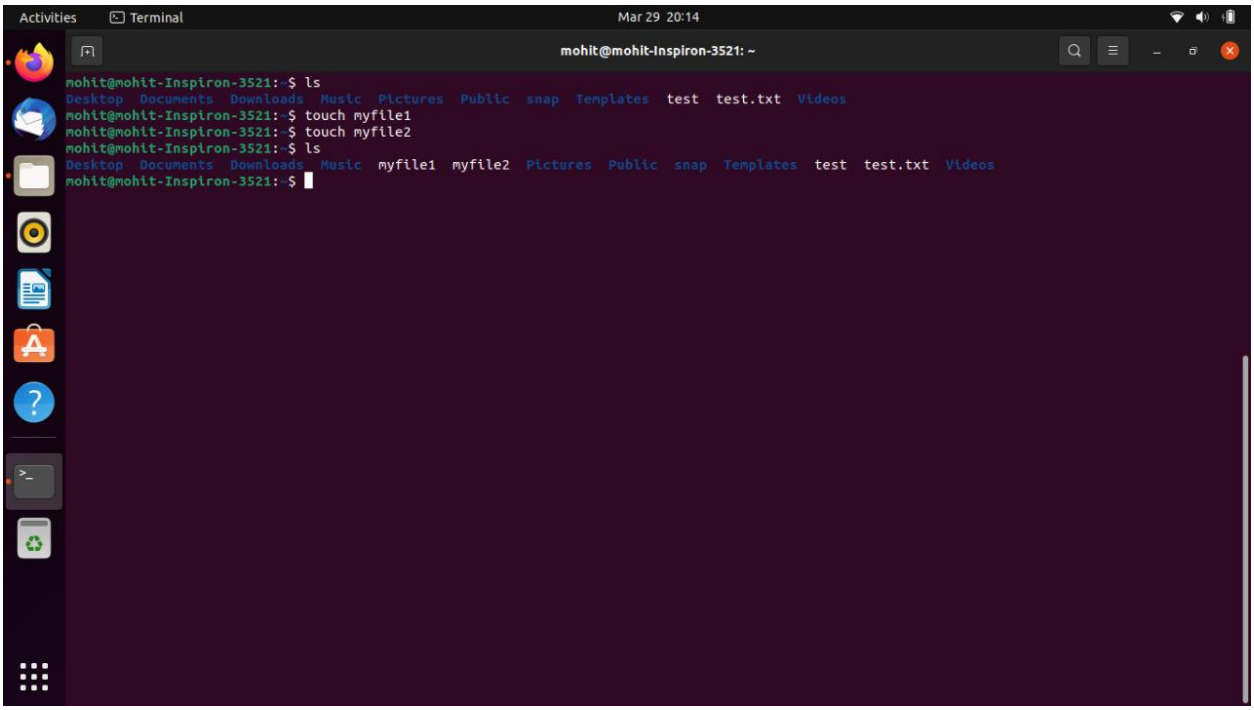
Option	Function
cat > [fileName]	To create a file.
cat [oldfile] > [newfile]	To copy content from older to new file.
cat [file1 file2 and so on] > [new file name]	To concatenate contents of multiple files into one.
cat -n/cat -b [fileName]	To display line numbers.
cat -e [fileName]	To display $ character at the end of each line.
cat [fileName] <<EOF	Used as page end marker.

3. touch:-

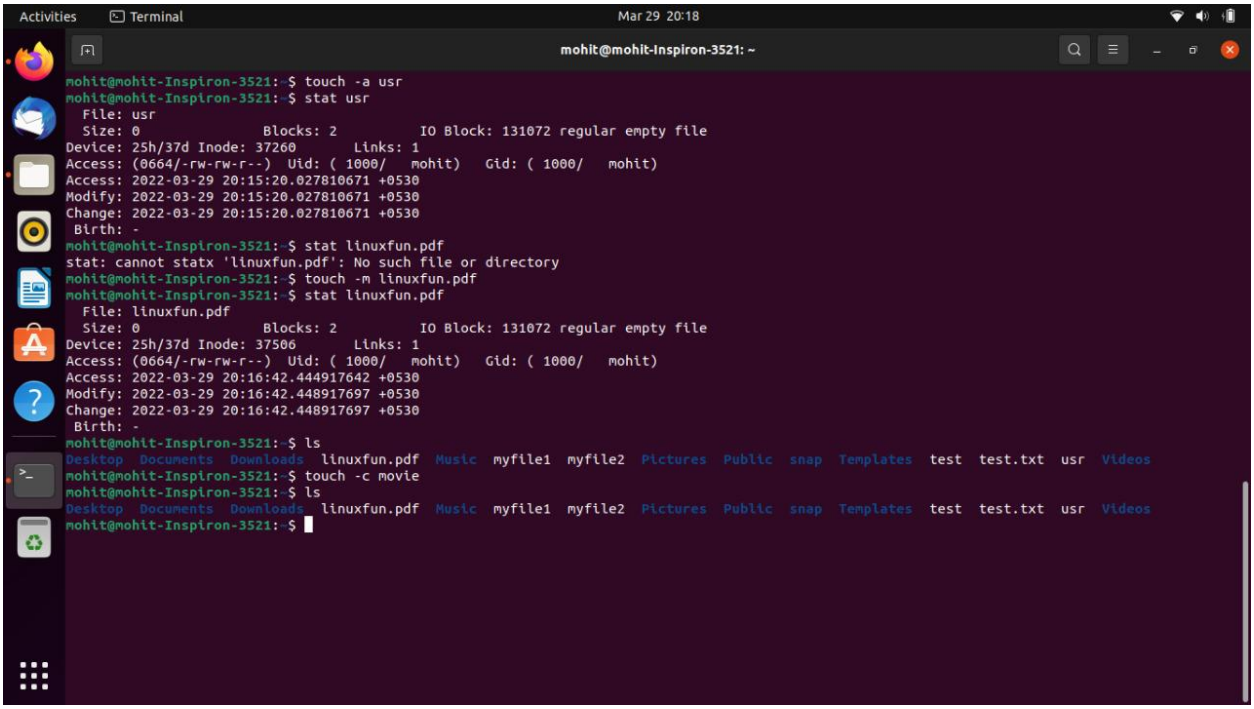
touch command is a way to create empty files (there are some other mehtods also). You can update the modification and access time of each file with the help of touch command.

Syntax:

touch **<filename>**



Option	Function
touch -a	To change file access and modification time.
touch -m	It is used to only modify time of a file.
touch -r	To update time of one file with reference to the other file.
touch -t	To create a file by specifying the time.
touch -c	It does't create n empty file.



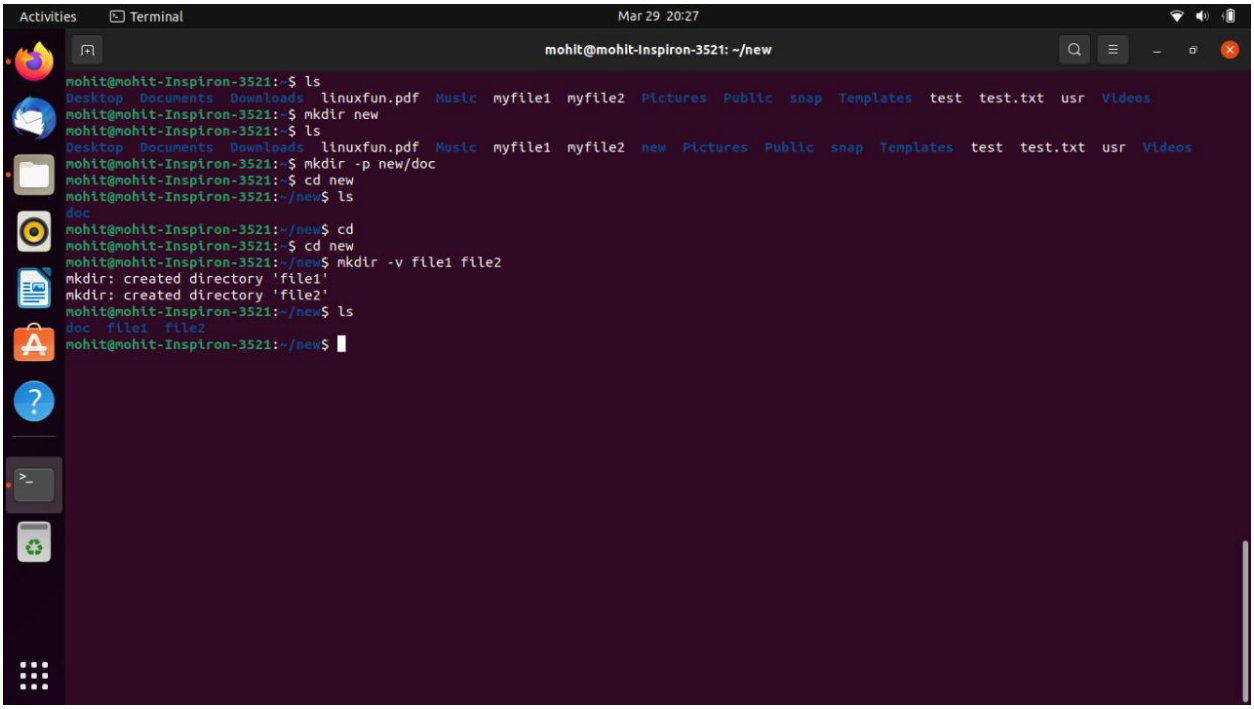
4.mkdir:-

The mkdir stands for 'make directory'. With the help of mkdir command, you can create a new directory wherever you want in your system. Just type **"mkdir <dir name>** , in place of <dir name> type the name of new directory, you want to create and then press enter.

Syntax:

mkdir **<dirname>**

Options	Description
mkdir -p, -parents	Add directory including its sub directory.
mkdir -v, -verbose	Print a message for each created directory.
mkdir -m -mode=MODE	Set access privilege.



5.cd:-

Linux **cd** command is used to change the current working directory (i.e., in which the current user is working). The "cd" stands for **'change directory**.' It is one of the most frequently used commands in the Linux terminal.

Syntax:

cd **<dirname>**

1. **Change from the current directory to a new directory**

cd **< current** directory**>** **<specified** directory**>**

```
mohit@mohit-Inspiron-3521:~$ pwd
/home/mohit
mohit@mohit-Inspiron-3521:~$ cd /home/mohit/Desktop
mohit@mohit-Inspiron-3521:~/Desktop$
```

2. **Change directory using an absolute path**

```
mohit@mohit-Inspiron-3521:~$ cd /home/mohit/Documents/files
mohit@mohit-Inspiron-3521:~/Documents/files$
```

3. **Change directory using a relative path**

```
mohit@mohit-Inspiron-3521:~$ cd /home/mohit/Documents
mohit@mohit-Inspiron-3521:~/Documents$ cd files
mohit@mohit-Inspiron-3521:~/Documents/files$
```

4. **Change to the home directory**

```
mohit@mohit-Inspiron-3521:~/Documents/files$ cd ~
mohit@mohit-Inspiron-3521:~$
```

5. **Change to the previous directory**

```
mohit@mohit-Inspiron-3521:~$ cd /home/mohit/Documents/files
mohit@mohit-Inspiron-3521:~/Documents/files$ cd -
/home/mohit
mohit@mohit-Inspiron-3521:~$
```

6. **Change to Parent Directory**

```
mohit@mohit-Inspiron-3521:~$ cd /home/mohit/Documents/files
mohit@mohit-Inspiron-3521:~/Documents/files$ cd ..
mohit@mohit-Inspiron-3521:~/Documents$
```

7. **Change to the root directory**

```
mohit@mohit-Inspiron-3521:~$ cd /home/mohit/Documents/files
mohit@mohit-Inspiron-3521:~/Documents/files$ cd /
mohit@mohit-Inspiron-3521:/$
```

8. **Change to another user's home directory**

```
mohit@mohit-Inspiron-3521:~$ cd /home/mohit/Documents/files
mohit@mohit-Inspiron-3521:~/Documents/files$ cd ~mohit
mohit@mohit-Inspiron-3521:~$
```

6. rmdir:-

This command is used to delete a directory. But will not be able to delete a directory including a sub-directory. It means, a directory has to be empty to be deleted.

```
mohit@mohit-Inspiron-3521:~/new$ ls
doc  file1  file2
mohit@mohit-Inspiron-3521:~/new$ rmdir file1
mohit@mohit-Inspiron-3521:~/new$ ls
doc  file2
mohit@mohit-Inspiron-3521:~/new$ cd
mohit@mohit-Inspiron-3521:~$ rmdir new/file2
mohit@mohit-Inspiron-3521:~$
```

7.rm:-

The 'rm' means remove. This command is used to remove a file. The command line doesn't have a recycle bin or trash unlike other GUI's to recover the files. Hence, be very much careful while using this command. Once you have deleted a file, it is removed permanently.

Syntax:

rm **<filename>**

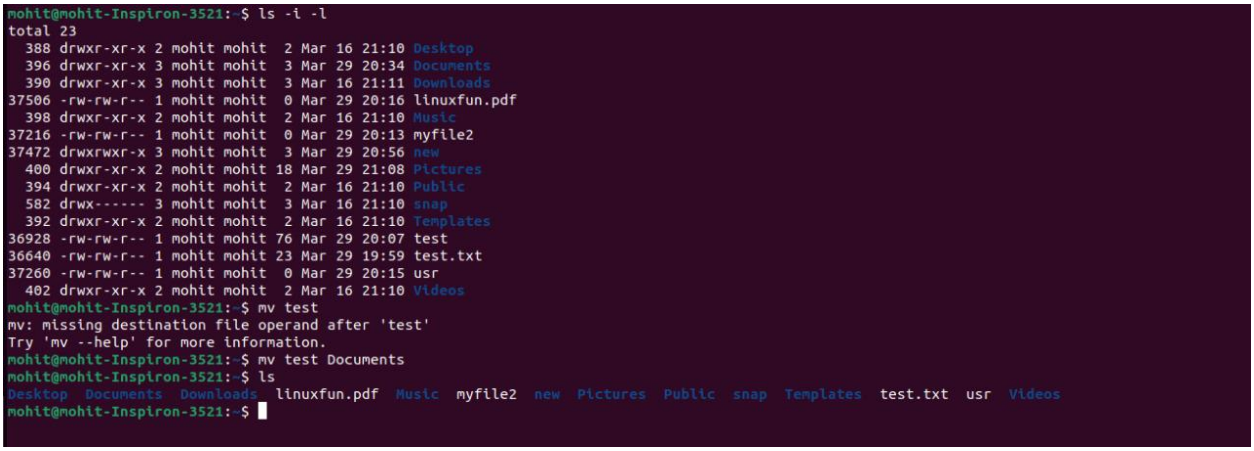
```
mohit@mohit-Inspiron-3521:~$ ls
Desktop  Documents  Downloads  linuxfun.pdf  Music  myfile1  myfile2  new  Pictures  Public  snap  Templates  test  test.txt  usr  Videos
mohit@mohit-Inspiron-3521:~$ rm myfile1
mohit@mohit-Inspiron-3521:~$ ls
Desktop  Documents  Downloads  linuxfun.pdf  Music  myfile2  new  Pictures  Public  snap  Templates  test  test.txt  usr  Videos
mohit@mohit-Inspiron-3521:~$
```

Option	Description
rm *extension	Used to delete files having same extension.
rm -r or R	To delete a directory recursively.
rm -i	Remove a file interactively.
rm -rf	Remove a directory forcefully.

8.mv:-

Linux mv command is used to move existing file or directory from one location to another. It is also used to rename a file or directory. If you want to rename a single directory or file then **'mv'** option will be better to use.

Option	Function
mv -i	Asks for permission to over write.
mv *	Move multiple files to a specific directory.
mv --suffix	Used to take backup before over writing.
mv -u	Only move those files that doesn't exist.



9. **cp: -**

'cp' means copy. 'cp' command is used to copy a file or a directory. Syntax will be,

cp **<existing** file name**>** **<new** file name**>**

Option	Function
cp -r	To copy a directory along with its sub dirctories.
cp file1 file 2 directory name	To copy multiple file or directories in a directory.
cp -backup	To backup the existing file before over writing it.
cp -i	Asks for confirmtion.
cp -l	To create hard link file.
cp -p	Preserves attribute of a file.

cp -u -v	To make sure source file is newer then destination file.

10. chmod:-

chmod changes the permissions of each given file according to mode, where mode describes the permissions to modify. Mode can be specified with octal numbers or with letters.

chmod [Options]... Mode [,Mode]... file...

chmod [Options]... Numeric_Mode file...

chmod [Options]... --reference=RFile file...

OPTIONS

Tag	Description
-f, --silent, --quiet	suppress most error messages
-v, --verbose	output a diagnostic for every file processed
-c, --changes	like verbose but report only when a change is made
-c, --reference=RFile	use RFile's mode instead of MODE values
-R, --recursive	change files and directories recursively
--help	display help and exit
--version	output version information and exit

Numeric mode

The format of a numberic mode is 'augo'

A numeric mode is from one to four octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. Any omitted digits are assumed to be leading zeros. The first digit selects the set user ID (4) and set group ID (2) and sticky (1) attributes. The second digit selects permissions for the user who owns the file: read (4), write (2), and execute (1); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values.

Read by owner only

```
mohit@mohit-Inspiron-3521:~$ chmod 400 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
-r-------- 1 mohit mohit 23 Mar 29 19:59 test.txt
mohit@mohit-Inspiron-3521:~$
```

Read by group only

```
mohit@mohit-Inspiron-3521:~$ chmod 040 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
----r----- 1 mohit mohit 23 Mar 29 19:59 test.txt
```

Read by anyone

```
mohit@mohit-Inspiron-3521:~$ chmod 004 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
-------r-- 1 mohit mohit 23 Mar 29 19:59 test.txt
mohit@mohit-Inspiron-3521:~$
```

Write by owner only

```
mohit@mohit-Inspiron-3521:~$ chmod 200 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
--w------- 1 mohit mohit 23 Mar 29 19:59 test.txt
mohit@mohit-Inspiron-3521:~$
```

Write by group only

```
mohit@mohit-Inspiron-3521:~$ chmod 020 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
-----w---- 1 mohit mohit 23 Mar 29 19:59 test.txt
mohit@mohit-Inspiron-3521:~$
```

Write by anyone

```
mohit@mohit-Inspiron-3521:~$ chmod 002 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
--------w- 1 mohit mohit 23 Mar 29 19:59 test.txt
mohit@mohit-Inspiron-3521:~$
```

Execute by owner only

```
mohit@mohit-Inspiron-3521:~$ chmod 100 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
---x------ 1 mohit mohit 23 Mar 29 19:59 test.txt
mohit@mohit-Inspiron-3521:~$
```

Execute by group only

```
mohit@mohit-Inspiron-3521:~$ chmod 010 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
------x--- 1 mohit mohit 23 Mar 29 19:59 test.txt
mohit@mohit-Inspiron-3521:~$
```

Execute by anyone

```
mohit@mohit-Inspiron-3521:~$ chmod 001 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
---------x 1 mohit mohit 23 Mar 29 19:59 test.txt
mohit@mohit-Inspiron-3521:~$
```

Allow read permission to owner and group and anyone.

```
mohit@mohit-Inspiron-3521:~$ chmod 444 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
-r--r--r-- 1 mohit mohit 23 Mar 29 19:59 test.txt
mohit@mohit-Inspiron-3521:~$
```

Allow everyone to read, write, and execute file.

```
mohit@mohit-Inspiron-3521:~$ chmod 777 test.txt
mohit@mohit-Inspiron-3521:~$ ls -l *.txt
-rwxrwxrwx 1 mohit mohit 23 Mar 29 19:59 test.txt
mohit@mohit-Inspiron-3521:~$
```

11. umask: -

`umask` command is used to set default file and folder permission in Linux operating system. File and folder permissions are important because the permission enables or deny different users, groups and others to access, read, write and execute of the given file. In this tutorial, we will learn how to use `umask` command to set default read, write and execute permissions.

Linux is a file-based operating system where there are 3 permissions.

- `r` means read where reading given file or folder is specified
- `x` means execute where execution of the given file specified. As folders can not be executed we can not use this permission for folders or directories.
- `w` means write where writing or appending to the specified file or folder.

The permissions has also numeric representation like below.

number	permission
4	read
2	write
1	execute

umask Permission Digit Presentation

We can use the following table to set umask permission digit presentation.

umask digit	default file permissions	default directory permissions
	rw	rwx
1	rw	rw
2	r	rx

3	r	r
4	w	wx
5	w	w
6	x	x
7	(no permission allowed)	(no permission allowed)

Practical-1

Aim: - To study about various utility commands.

Commands:-

1. passwd:-

NAME

passwd - change user password

SYNOPSIS

passwd [options] [LOGIN]

DESCRIPTION

The passwd command changes passwords for user accounts. A normal user may only change the password for his/her own account, while the super user may change the password for any account. passwd also changes the account or associated password validity period.As a general guideline, passwords should consist of 6 to 8 characters including one or more characters from each of the following sets:

- lower case alphabetics
- digits 0 thru 9
- punctuation marks

Care must be taken not to include the system default erase or kill characters. passwd will reject any password which is not suitably complex.

OPTIONS

The options which apply to the passwd command are:

-a, --all

This option can be used only with -S and causes show status for all users.

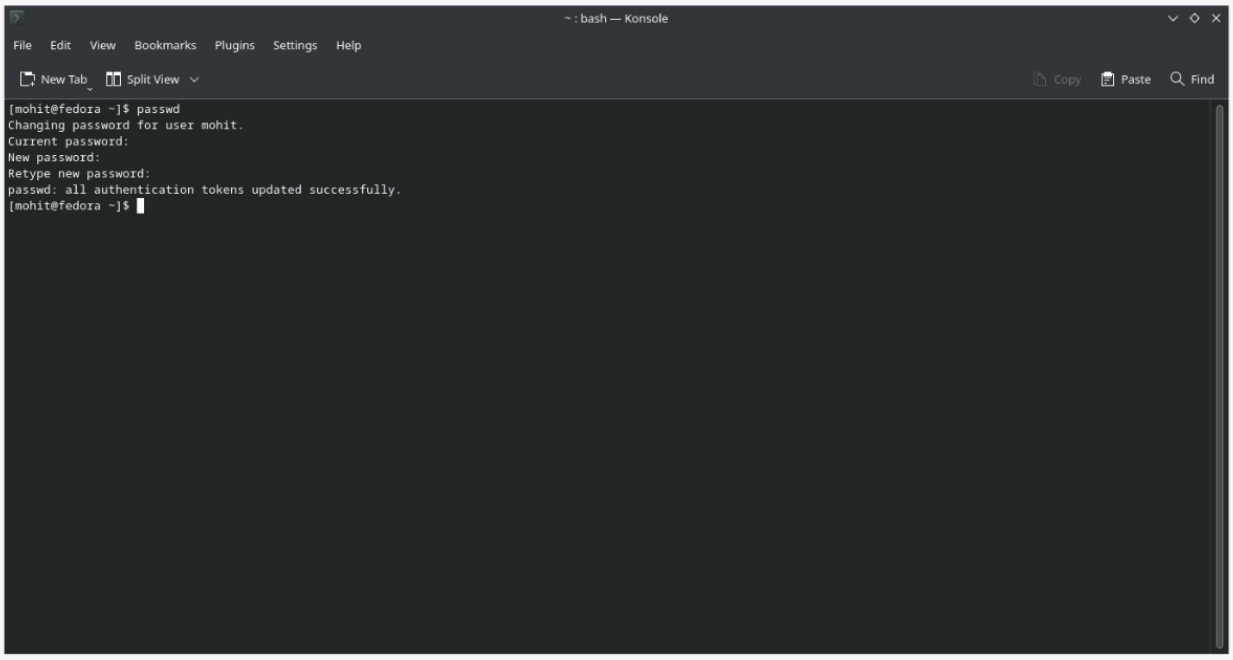
-d, --delete

Delete a user's password (make it empty). This is a quick way to disable a password for an account. It will set the named account passwordless.

-e, --expire

Immediately expire an account's password. This in effect can force a user to change his/her password at the user's next login.

-h, --help Display help message and exit.



2. who:-

NAME

who - show who is logged on

SYNOPSIS

who [OPTION]... [FILE | ARG1 ARG2]

DESCRIPTION

Print information about users who are currently logged in.

OPTIONS

-a, --all
same as -b -d --login -p -r -t -T -u

-b, --boot
time of last system boot

-H, --heading
print line of column headings

--ips print ips instead of hostnames. with --lookup, canonicalizes based on stored IP, if available, rather than stored hostname

-l, --login
print system login processes

-p, --process
print active processes spawned by init

-q, --count
all login names and number of users logged on

-t, --time
print last system clock change

-u, --users
list users logged in

3. clear:-

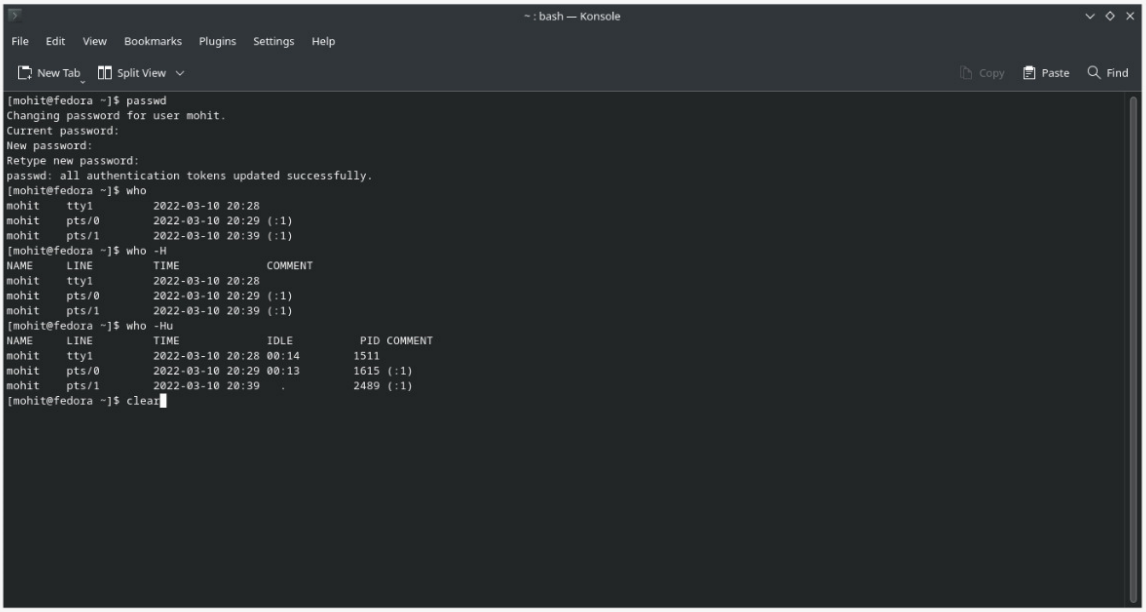
NAME

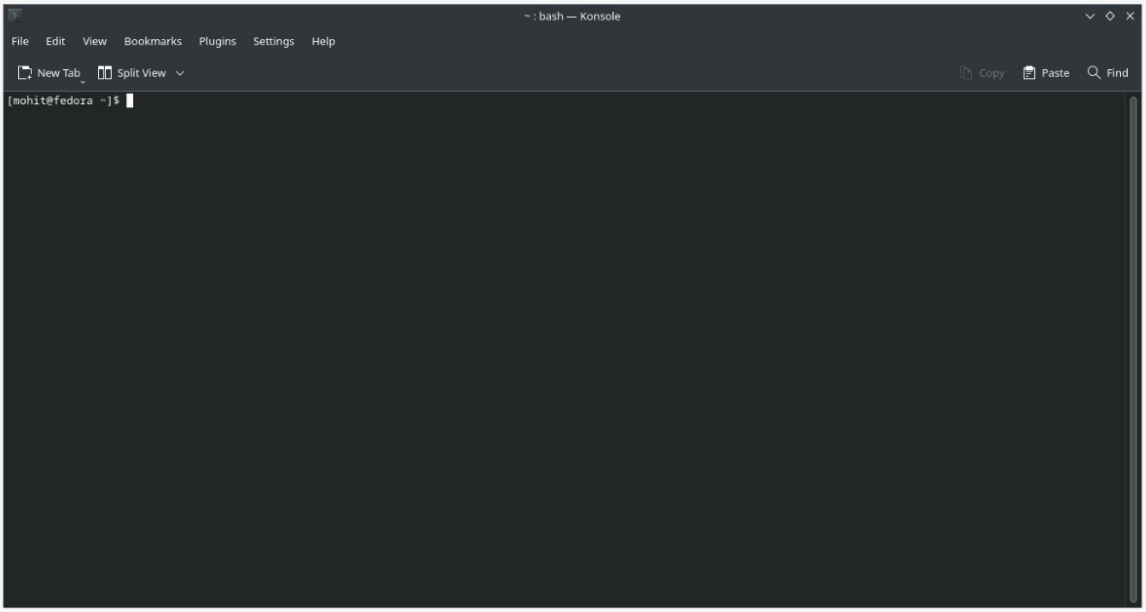
clear - clear the terminal screen

SYNOPSIS clear

DESCRIPTION

Clear clears your screen if this is possible. It looks in the environment for the terminal type and then in the terminfo database to figure out how to clear the screen. Clear ignores any command-line parameters that may be present.


~ : bash — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View
Copy Paste Find
[mohit@fedora ~]$ passwd
Changing password for user mohit.
Current password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[mohit@fedora ~]$ who
mohit tty1 2022-03-10 20:28
mohit pts/0 2022-03-10 20:29 (:1)
mohit pts/1 2022-03-10 20:39 (:1)
[mohit@fedora ~]$ who -H
NAME LINE TIME COMMENT
mohit tty1 2022-03-10 20:28
mohit pts/0 2022-03-10 20:29 (:1)
mohit pts/1 2022-03-10 20:39 (:1)
[mohit@fedora ~]$ who -Hu
NAME LINE TIME IDLE PID COMMENT
mohit tty1 2022-03-10 20:28 00:14 1511
mohit pts/0 2022-03-10 20:29 00:13 1615 (:1)
mohit pts/1 2022-03-10 20:39 . 2489 (:1)
[mohit@fedora ~]$ clear


~ : bash — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View
Copy Paste Find
[mohit@fedora ~]$

4.tput:-

NAME

tput, reset - initialize a terminal or query terminfo database

SYNOPSIS

tput [-Ttype] capname [parms ...]
tput [-Ttype] init
tput [-Ttype] reset
tput [-Ttype] longname
tput-S << tput-V

DESCRIPTION

The tput utility uses the terminfo database to make the values of terminal-dependent capabilities and information available to the shell (see sh(1)), to initialize or reset the terminal, or return the long name of the requested terminal type. tput outputs a string if the attribute (capability name) is of type string, or an integer if the attribute is of type integer. If the attribute is of type boolean, tput simply sets the exit code (0 for TRUE if the terminal has the capability, 1 for FALSE if it does not), and produces no output

5.tty:-

NAME

tty - print the file name of the terminal connected to standard input

SYNOPSIS

tty [OPTION]...

DESCRIPTION

Print the file name of the terminal connected to standard input.

-s, --silent, --quiet
print nothing, only return an exit status

6. uname:-

NAME

uname - print system information

SYNOPSIS

uname [OPTION]...

DESCRIPTION

Print certain system information. With no OPTION, same as -s.

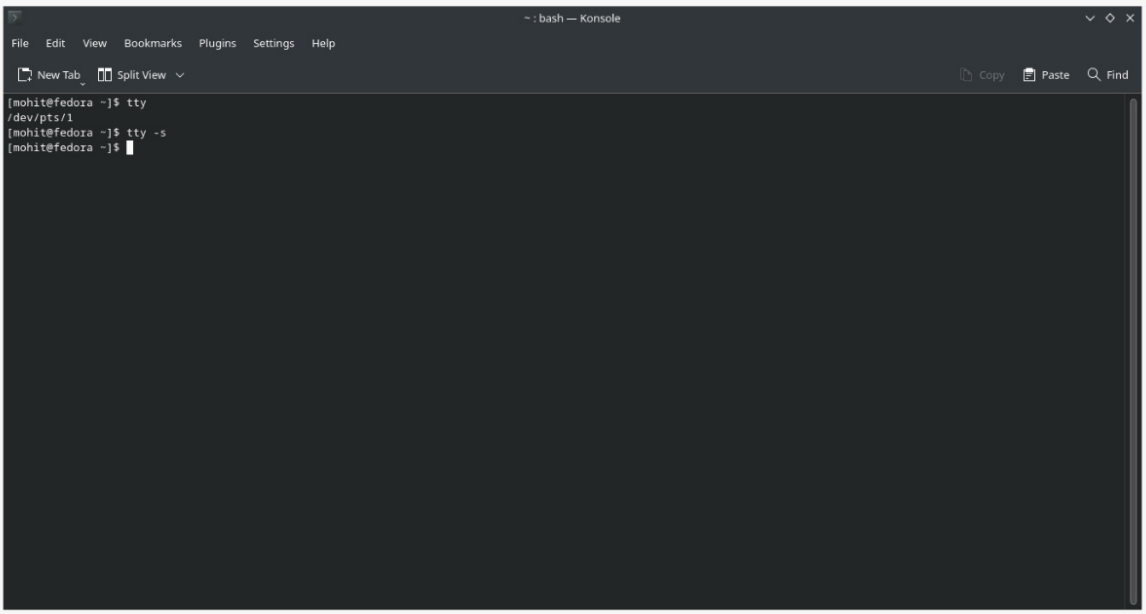
-a, --all
print all information, in the following order, except omit -p and -i if unknown:

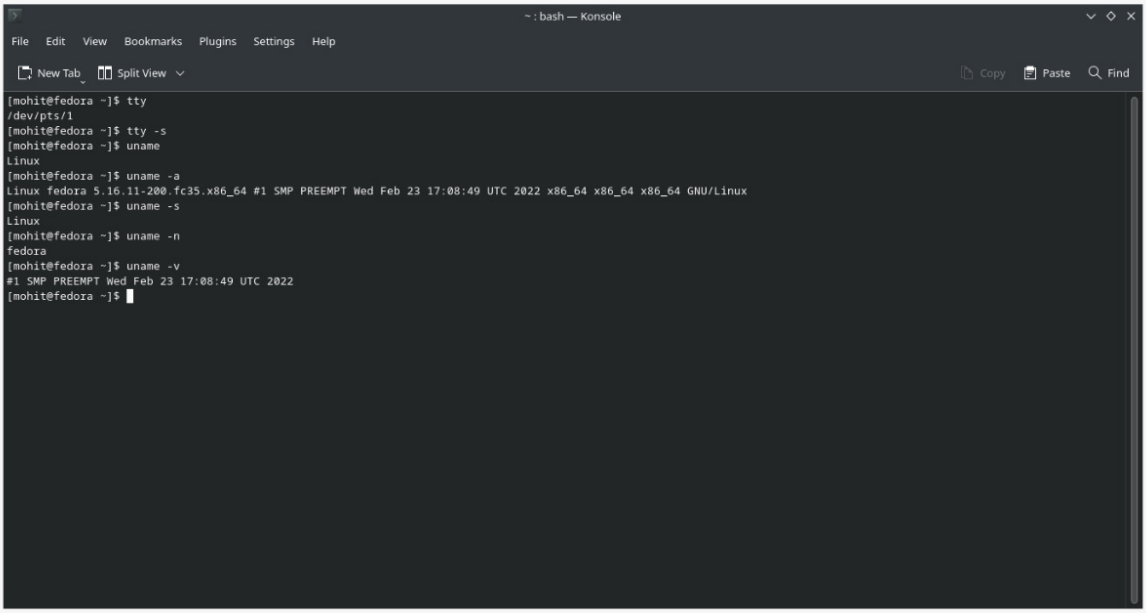
-s, --kernel-name
print the kernel name

-n, --nodename
print the network node hostname

-r, --kernel-release
print the kernel release

-v, --kernel-version print the kernel version


~ : bash — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View
Copy Paste Find
[mohit@fedora ~]$ tty
/dev/pts/1
[mohit@fedora ~]$ tty -s
[mohit@fedora ~]$


~ : bash — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View
Copy Paste Find
[mohit@fedora ~]$ tty
/dev/pts/1
[mohit@fedora ~]$ tty -s
[mohit@fedora ~]$ uname
Linux
[mohit@fedora ~]$ uname -a
Linux fedora 5.16.11-200.fc35.x86_64 #1 SMP PREEMPT Wed Feb 23 17:08:49 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
[mohit@fedora ~]$ uname -s
Linux
[mohit@fedora ~]$ uname -n
fedora
[mohit@fedora ~]$ uname -v
#1 SMP PREEMPT Wed Feb 23 17:08:49 UTC 2022
[mohit@fedora ~]$

7.cal:-

NAME

cal, ncal — displays a calendar and the date of Easter

SYNOPSIS

cal [-3hjy] [-A number] [-B number] [[month] year]
cal [-3hj] [-A number] [-B number] -m month [year]
ncal [-3bhjJpwySM] [-A number] [-B number] [-s country_code] [[month] year]
ncal [-3bhJeoSM] [-A number] [-B number] [year]
ncal [-CN] [-H yyyy-mm-dd] [-d yyyy-mm]

DESCRIPTION

The cal utility displays a simple calendar in traditional format and ncal offers an alternative layout, more options and the date of Easter. The new format is a little cramped but it makes a year fit on a 25x80 terminal. If arguments are not specified, the current month is displayed.

The options are as follows:

-h Turns off highlighting of today.

-J Display Julian Calendar, if combined with the -e option, display date of Easter according to the Julian Calendar.

-e
Display date of Easter (for western churches).

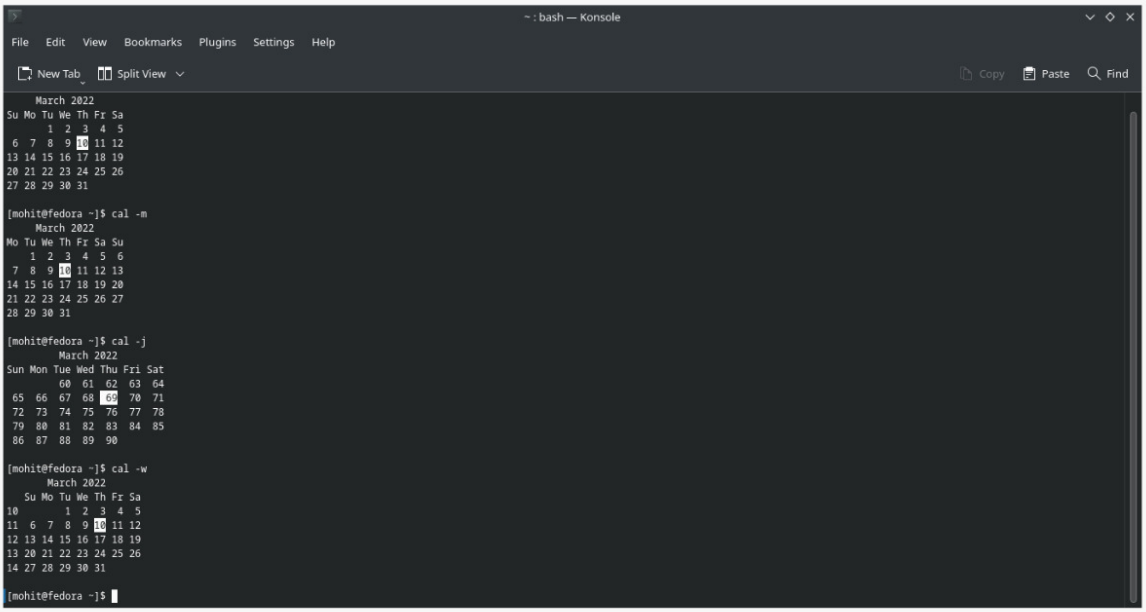
-j
Display Julian days (days one-based, numbered from January 1).

-m month
Display the specified month. If month is specified as a decimal number, it may be followed by the letter 'f' or 'p' to indicate the following or preceding month of that number, respectively.

-w
Print the number of the week below each week column.

-y
Display a calendar for the specified year.

-A number
Display the number of months after the current month.



8.date:-

NAME

date - print or set the system date and time

SYNOPSIS

date [OPTION]... [+FORMAT]

date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION

Display the current time in the given FORMAT, or set the system date.

-d, --date=STRING

display time described by STRING, not `now'

-f, --file=DATEFILE

like --date once for each line of DATEFILE

-r, --reference=FILE

display the last modification time of FILE

-R, --rfc-2822

output date and time in RFC 2822 format. Example: Mon, 07 Aug 2006 12:34:56 -0600

--rfc-3339=TIMESPEC

output date and time in RFC 3339 format. TIMESPEC=`date', `seconds', or `ns' for date and time to the indicated precision. Date and time components are separated by a single space: 2006-08-07 12:34:56-06:00

-s, --set=STRING

set time described by STRING

-u, --utc, --universal

print or set Coordinated Universal Time

9.bc:-

NAME

bc - An arbitrary precision calculator language

SYNTAX

bc [-hlwsqv] [long-options] [file ...]

DESCRIPTION

bc is a language that supports arbitrary precision numbers with interactive execution of statements. There are some similarities in the syntax to the C programming language. A standard math library is available by command line option. If requested, the math library is defined before processing any files. bc starts by processing code from all the files listed on the command line in the order listed. After all files have been processed, bc reads from the standard input. All code is executed as it is rea (If a file contains a command to halt the processor, bc will never read from the standard input.)

OPTIONS OF BC

-h, --help

Print the usage and exit.

-i, --interactive

Force interactive mode.

-l, --mathlib

Define the standard math library.

-w, --warn

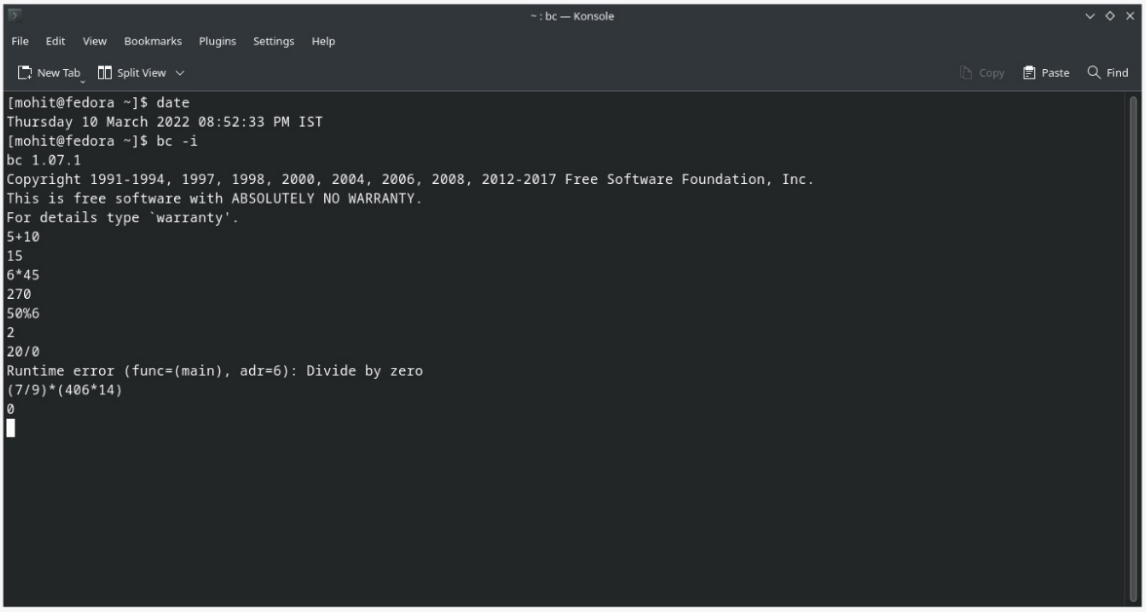
Give warnings for extensions to POSIX bc.

-s, --standard

Process exactly the POSIX bc language.

-q, --quiet

Do not print the normal GNU bc welcome.



10.script:-

NAME

script — make typescript of terminal session

SYNOPSIS

script [-a] [-c command] [-e] [-f] [-q] [-t[=file]] [-V] [-h] [file]

DESCRIPTION

script makes a typescript of everything printed on your terminal. It is useful for students who need a hardcopy record of an interac- tive session as proof of an assignment, as the typescript file can be printed out later with lpr.If the argument file is given, script saves all dialogue in file. If no file name is given, the typescript is saved in the file typescript.

Options:

-a, --append

Append the output to file or typescript, retaining the prior contents.

-c, --command command

Run the command rather than an interactive shell. This makes it easy for a script to capture the output of a program that behaves differently when its stdout is not a tty.

-e, --return

Return the exit code of the child process.Uses the same format as bash termination on signal termination exit code is 128+n.

-f, --flush

Flush output after each write. This is nice for telecooperation: one person does `mkfifo foo; script -f foo', and another can supervise real-time what is being done using `cat foo'.

--force

Allow the default output destination, i.e. the typescript file, to be a hard or symbolic link. The command will follow a symbolic link.

-q, --quiet

Be quiet.

11.echo:-

NAMEecho - display a line of text

SYNOPSIS

echo [SHORT-OPTION]... [STRING]…

echo LONG-OPTION

DESCRIPTION

Echo the STRING(s) to standard output.

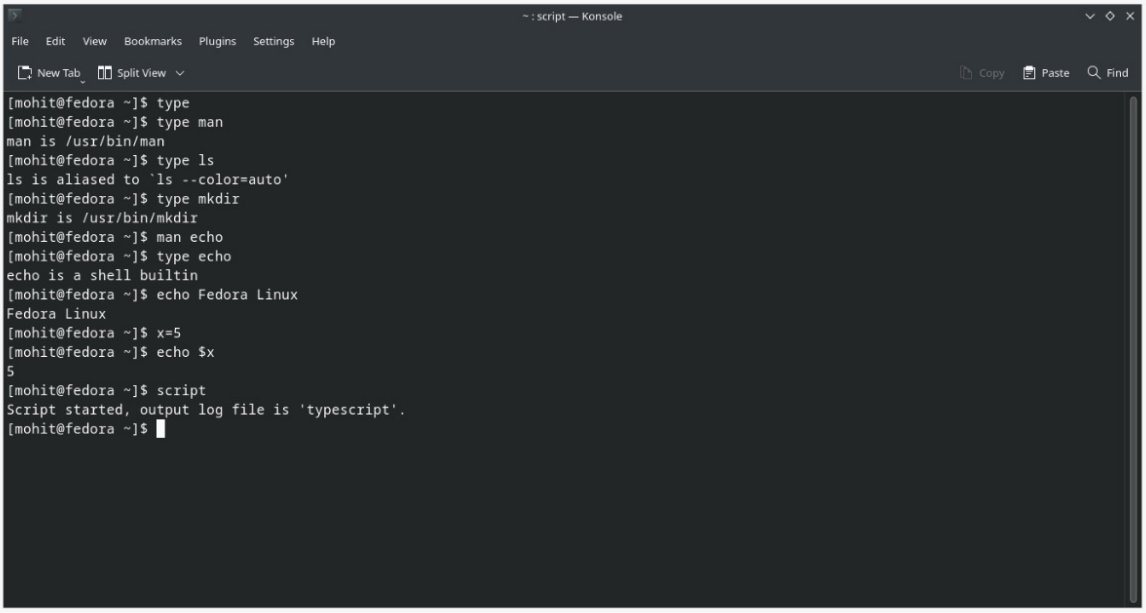
Options:-

-n do not output the trailing newline

-e enable interpretation of backslash escapes

-E disable interpretation of backslash escapes (default)

--help display this help and exit



12.lock:-

NAME

lock – lock the user logged in

SYNOPSIS lock

DESCRIPTION

Thelockcommand appeared for the first time in 3.0BSD, decades ago, and some version of it has existed in the major BSD Unix systems that have been available ever since. The versions oflockincluded in the base systems of FreeBSD, NetBSD, and OpenBSD differ slightly in the command line options they provide today.

All three of them lock a standard virtual console. FreeBSD's lock in particular, distributed under copyfree terms (a BSD License), behaves as follows, by default.

1. It asks for a "key", or password, that can be used to unlock the terminal at any time.
2. It locks the terminal for fifteen minutes, or until it is unlocked with that key — whichever comes first.

13.type:-

NAME type - display a line of text

SYNOPSIS type command

DESCRIPTIONType is a Unix command that describes how its arguments would be interpreted if used as command names. Where applicable, type will display the command name's path. Possible command types are:

- shell built-in
- function
- alias
- hashed command
- keyword

The command returns a non-zero exit status if command names cannot be found.