INDEX

SL. No.	Date of Submission	Title	Page No.	Remarks
1.	23/07/19	Linux Commands	1 - 7	
2.	20/08/19	Write a shell script program to implement basic calculator.	8 - 9	
3.	27/8/19	Write a shell script program to displayed the following thing i) Details of all file & attribute. ii) Current date & time iii) Currently log-in users. iv) process status v) present working status.	10 - 13	
4.	27/08/19	Write a shell script program to convert Farhenheite to celcius.	14	

NO	TITLE	DATE	PG NO	SIGNATURE
5.	Write a shell script program to find gross salary.	27/08/19	15-16	
6.	Write a shell script program to Bubble sort.	03/09/19	17-18	
7.	Write a shell script program to fibonacci series.	03/09/19	19	
8.	Write a shell script program to check a number is prime or not.	03/09/19	20	
9.	Write a c program to calculate waiting time and Average turn arround time in FCFS algorithm.	17/09/19	21-23	
10.	Write a c program to calculate waiting time & ATAT in SJF algorithm.	24/09/19	24-26	
11.	Write a c program to calculate Average wt and ATAT in Round robin algo.	24/09/19	27-30	
12.	Write a c program to calculate average waiting time and average turn arround time in Priority scheaduling algorithm.			

LINUX COMMANDS ⇒

1. echo :- It is used to display the message / text / string that are passed as an argument. This is a build in command that is mostly used in shell shell scripts to display output to the screen.

Ex: $ echo "My name is Ram."
My name is Ram.

2. tty :- tty command is used to know the name of the device file on which user is working.

Ex: $ tty
/dev/pts/1

3. Who :- use to display the log-in details of the current user.

Ex: $ who
ram :0 2019-09-13 22:49 (:0)

4. whoami ⇒ use to get the details of the self log-in.

Ex: $ whoami

ram

5. date ⇒ shows the current date among with the time.

Ex: $ date

sat sep 14 00:18:07 IST 2019

6. cal ⇒ show the calender.

Ex: $ cal

september 2019

SU	MO	TU	WE	TH	FR	SA
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

7. cat ⇒ It-is one of the more usefull for create a small file.
cat stands for "concatenate" It-is used to allows us to create single or multiple files, view contain of files, concatenat files & redirect output in terminal or files.

Ex

```
$ cat > abc
My self Ramprasad Mandal.
$ cat abc
My self Ramprasad Mandal.
$ cat >> abc
This is my first linux command.
$ cat abc
My self Ramprasad Mandal.
This is my first linux command.
```

8. man ⇒ man command in Linux is used to display the user manual of any command that we can run on the terminal.
It provides a details view of the command which includes → NAME, SYNOPSIS, DESCRIPTION, OPTION, EXIT VALUE, RETURNS VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS, and SEE also.

Ex ⇒ $ man echo

Output ⇒

ECHO (1)

NAME

echo - display a line of text

SYNOPSIS

echo [SHORT-OPTION]... [STRING]....

DESCRIPTION

Echo the STRING(S) to standard output.

-n do not output the trailing newline.

-e enable interpretation of backslash eseapes.

-E disable interpretation of backslash escapes (default)

--help display this help and exit

--version

output version information and exit.

9. LS ⇒ gt is used to list information about files and directories within the file system.

Ex $ ls

abc	cal.sh	exp.desktop	public
all.sh	Desktop	for.sh	Music
a.sh	pocument	fibbo.sh	videos

10. ls -l ⇒ In this command is used to give the detailed list of folder in the directory, it will display the size, Date in which it is modified and ownership of the file and the permission allowed in the folder/file.

Ex: $ ls -l

total 84

-rw-r--r--	1 ram	58	sep 14	19.57	abc
-rwxr-xr-x	1 ram	359	sep 13	23.36	all.sh
drwxr-xr-x	2 ram	4096	Jul 29	11.26	Music
drwxr-xr-x	2 ram	4096	Jul 29	11.26	videos.

11. LS -a ⇒ This command is used to display all hidden file & folder.

Ex: $ LS -a

.	.bashrc	Downloads	.ICEauthority
..	Bubble.sh	exp.desktop	.local
abc	.cache	.mozilla	videos
.bash	.profile	prime.sh	

12. mkdir ⇒ mkdir commands in Linux allows the user to create directories. This command can create multiple directories at once as well as set the permission for the directories.

Ex $ mkdir Aliah

or, $ mkdir -p Aliah/a/b # use to create sub directory.

13. cd ⇒ The cd command is used to change the current directory (i.e., the directory in which the user is currently working) in Linux.

Ex ram@Ram:~$ cd Desktop

ram@Ram:~/Desktop$

14. chmod ⇒ chmod command is used to change the access permissions of the system objects (file and directories). It is also used to change special mode flags. The request is filtered by the umask. The name is an abbreviation of change mode.

Ex: $ ls -l

drwxr-xr-x 2 ram ram 4096 JUL 29 11:26 MUSIC

$ chmod 764 MUSIC

drwxrw-r-- 2 ram ram 4096 JUL 29 11:26 MUSIC

2. Write a shell script program to implement basic calculator.

```
echo "1 for add"
echo "2 for sub"
echo "3 for Mul"
echo "4 for div"
echo "Enter your choice:"
read ch
echo "Enter two numbers:"
read a
read b
case $ch in
    1) res='echo $a \+ $b|bc';;
    2) res='echo $a \- $b|bc';;
    3) res='echo $a \* $b|bc';;
    4) res='echo "scale=2; $a/$b"|bc';;
    *) echo "Unknown input";;
esac
echo "$res"
```

(1) Output ⇒

```
1 for add.
2 for sub.
3 for Mul.
4 for Div.
Enter your choise.
1
Enter two numbers.
22
11
33
```

(2) output ⇒

```
1 for add.
2 for sub.
3 for Mul
4 for Div
Enter your choice.
4
22
11
2.00
```

(3) output ⇒

```
1 for add.
2 for sub.
3 for Mul.
4 for Div
Enter your choice.
3
11
11
121
```

(4) output ⇒

```
1 for add
2 for sub
3 for Mul
4 for Div
Enter your choice
2
22
11
11
```

5. Write a shell script program to elasted the following thing ⇒

i) Details of all file & Attribute
ii) current Date & Time
iii) process status
iv) Currently Log-in user
v) To display the present working dir.

```
clear
echo "1. Details of all file & folder."
echo "2. Current date & time."
echo "3. process status."
echo "4. Currently log-in user."
echo "5. Display present working dir."
echo "----------"
echo "Press any one."

read n
case $n in
1) ls -l ;;
2) date ;;
3) PS ;;
4) who ;;
5) pwd ;;
*) "Wrong input!"
   echo
esac
```

(1) Output =>

1. Details of all file & folder.
2. current date & time.
3. process status.
4. currently Log-in user.
5. Display present working directory.

---

press any one.

1

total 80

-rwxr-xr-x 1 ram ram 359 sep 13 12.36 all.sh

-rw-r-- 1 ram ram 33 sep 8 16.39 a.sh

⋮

drwxr-xr-x 2 ram ram 4096 JUL 29 11.26 videos

(2) Output =>

1. Details of all file & folder.
2. current date & time.
3. process status.
4. currently Log-in user.
5. Display present working directory.

---

press any one.

2

Fri sep 13 23.43.50 IST 2019

(3) output ⇒

1. Details of all file & folder.
2. current date & time.
3. process status.
4. currently log-in user.
5. Display present working directory.

---

press any one.

3

```
  PID  TTY          TIME CMD
 1717 pts/1    00:00:00 bash
 2098 pts/1    00:00:00 bash
 2099 pts/1    00:00:00 PS
```

(4) output ⇒

1. details of all file & folder
2. current date & time
3. process status
4. currently log-in user.
5. Display present working directory.

---

press any one.

4

```
ram      :0      2019-09-13 22:49 (:0)
```

(5) Output :⇒

1. Details of all file & folder.
2. current date & time.
3. process status.
4. Currently Log-in user.
5. Display present working directory.

---

press any one.

5

/home/ram

(6) output ⇒

1. Details of all file & folder.
2. current date & time
3. process status.
4. Currently log-in user.
5. Display present working directory.

---

press any one.

6

Wrong input!

4. Write a shell script program to convert fahrenheit into celcius.

```
clear
echo "convert farhenheit temparature
      into celcius"
echo "Enter a number: "
read f
c=$(echo "scale=2;(((($f-32)*5)/9)"
                                  |bc)
echo "$f f = $c c"
```

output:

```
convert farheheit temparature into
                                celcius

Enter a number:
42
42 f = 5.5 c
```

5 Write a shell script program to find gross salary.

```
clear
echo "Enter basic salary:"
read sal
if [ $sal -le 1500]    # If (($sal < 1500))
then
    hr = 'expr $sal \* 10 / 100'
    da = 'expr $sal \* 90 / 100'
    gs = 'expr $sal \+ $hr \+ $da'
echo "Your Basic salary is: $$sal"
echo "your HR is:" $hr"
echo "your DA is: $da"
echo "your gross salary is: $gs"
else
    hr = 'expr $sal \+ 500'
    da = 'expr $sal \* 98 / 100'
    gs = 'expr $sal \+ $hr \+ $da"
echo "your basic salary is: $sal"
echo "your HR is: $hr"
echo "your DA is: $da"
echo "your gross salary is: $gs"
fi
```

① output:

```
Enter basic salary is :
1280
Your basic salary is : 1280
Your HR is : 128
Your DA is : 1152
Your gross salary is : 2560
```

output_2:

```
Enter basic salary :
4950
your basic salary is : 4950
your HR is :5450
your DA is : 4851
your gross salary is : 15251
```

6. Write a shell script program to Bubble sort.

```
clear
echo "Enter the total no: "
read n
echo "Enter the no one by one:"
declare -a a.
for((i=0; i<n; i++))
do
   read a[$i]
done
for((i=1; i<n; i++))
do
      for((j=0; j<n-1; j++))
      do
          if((a[j] > a[j+1]))
          then
              temp = ${a[j]}
              a[$j] = ${a[j+1]}
              a[$j+1] = $temp
          fi
      done
done
echo "The Bubble sort list are:"
for((i=0; i<n; i++))
do
   echo "${a[i]}
done
```

output =>

```
Enter the total no:
6
Enter the no one by one:
12
 6
55
77
34
16
The Bubble sort list are:
 6
 12
 16
 34
 55
 77
```

7. Write a shell script program to fibonacci series.

```
clear
echo "Enter a number: "
read n
a=0;
b=1;
echo "The fibonacci series: "
for ((i=0; i<n; i++))
 do
      echo "$a"
      fn=$((a+b))
      a=$b
      b=$fn
done
```

Output:

```
Enter a number:
9
The fibonacci series is:
0
1
1
2
3
5
8
13
21
```

8. Write a program to check whether a no is prime or not.

```
clear
echo "Enter a number:"
read n
f=0
for ((i=1; i<=n; i++))
do
  if [$((n%i)) -eq 0]
  then
    f='expr $f \+ 1'
  fi
done
if [$f -eq 2]
then
echo "prime"
else
 echo "Not prime"
fi
```

(1) output ⇒ Enter a number
15
Not prime

(2) output ⇒

Enter a number
19
prime

9. Write a c program to calculate average waiting time and average turn arround time for the following algorithm

FCFS

```
#include <stdio.h>
#include <conio.h>
#define MAX 30
void main()
{
int i, j, n, bt[MAX], at[MAX], w[MAX],
    tat[MAX], temp[MAX];
float awt=0, atat=0;
printf("Enter the no of process");
scanf("%d", &n);
printf("Enter the burst time of the
                                process");
for(i=0; i<n; i++)
  {
   scanf("%d", &bt[i]);
  }
printf("Enter the arrival time of the
                                process");
for(i=0; i<n; i++)
   {
    scanf("%d", &at[i]);
   }
temp[0]=0;
```

```
Printf ("process \t burst time \t arrival
    time \t waiting time \t turn
    arround time \n");
for (i=0; i<n; i++)
{
    wt [i] = 0;
    tat [i] = 0;
    temp [i+1] = temp [i] + bt [i];
    wt [i] = temp [i] - at [i];
    tat [i] = wt [i] + bt [i];
    awt = awt + wt [i];
    atat = atat + tat [i];
    printf ("%.d \t %.d \t\t %.d \t\t %.d
        \t\t %.d \n", i+1, bt [i], at [i],
        wt [i], tat [i]);
}
awt = awt /n;
atat = atat/n;
printf ("Average waiting time = %.f \n", awt);
printf ("Average turn arround time = %.f
    \n", atat );
}
```

P.T.O →

output =>

Enter the no of process. 4

Enter burst time. 8 4 9 5

Enter arrival time. 0 1 2 3

process	burst time	arrival time	waiting time	turn arround time
1	8	0	0	8
2	4	1	7	11
3	9	2	10	19
4	5	3	18	23

average waiting time = 8.750000

average turn arround time = 15.250000

Write a c program to calculate Average Waiting time and ATA in SJF Algorithm.

```
#include <stdio.h>
int main()
{
    int i, n, j;
    printf("Enter the no of process:");
    scanf("%d", &n);
    int burst_time[n], temp, wt[n], tat[n],
        ct[n], at[n];
    printf("Enter the burst time for the process");
    for (i=0; i<n; i++)
    {
        printf("\n Burst time of p %d :", i+1);
        at[i]=0;
    }
    for (i=0; i<n; i++)
    {
        for (j=0; j<n-1-i; j++)
        {
            if (burst_time[j] > burst[j+1]
            {
                temp = burst_time[j];
                burst_time[j] = burst_time[j+1];
                burst_time[j+1] = temp;
            }
        }
    }
    ct[0] = burst_time[0];
```

```
    ct[i+1] = ct[i] + burst_time[i+1];
    tat[p] = ct[p] - at[p];
    wt[i] = tat[i] - burst_time[i];
    awt += wt[i];
    atat += tat[i];
}
printf("\n Burst time \t waiting time \t
        Turn around time: \n");
for (i=0; i<n; i++)
{
    printf("%d \t\t %d \t\t %d \n",
        burst_time[i], wt[i], tat[i]);
}
printf("\n Average waiting time: %.2f
    \n Average turn around
    time: %.2f", awt/n, atat/n);

return 0;

}
```

output :-

```
Enter the no of processes : 4
Enter the burst time of the processes:
Burst time of P1 : 8
Burst time of P2 : 4
Burst time of P3 : 9
Burst time of P4 : 5
```

Burst time	waiting time	turnaround time
4	0	4
5	4	9
8	9	17
9	17	26

Average waiting time : 7.50
Average Turn around time : 14.00

Write a c program to calculate Average wt and Average TAT.

```
#include <stdio.h>
int main()
{
int i, limit, total=0, x, counter=0, time_
   quantum, wait_time=0, turnaround_time=0,
   arrival_time[10], burst_time[10], temp[10];
float average_waiting_time, average_turna
   round_time;
printf("\n Enter the total no of processes:");
scanf("%d", &limit);
x = limit;
for(i=0; i< limit; i++)
{
    printf("\n Arrival time of P %d:",
                i+1);
    scanf("%d", &arrival_time[i]);
    printf("\n Burst_time of P %d:", i+1);
    scanf("%d", &burst_time[i]);
    temp[i] = burst_time[i];
}
printf("\n Enter the quantum:");
scanf("%d", &time_quantum);
printf("\n process \t\t Burst_time \t
 Turnaround Time \t waiting time");
```

```
for (total=0; i=0; x!=0)
{
 if (temp[i] <= time_quantum && temp[i]>0)
 {
  total = total + temp[i];
  temp[i] = 0;
  counter = 1;
 }
else if (temp[i] > 0)
{
  temp[i] = temp[i] - temp_quantum;
  total = total + time_quantum;
}
if (temp[i]==0 && counter == 1)
{
  x--;
  printf("\n p %d \t\t %d \t\t %d \t\t %d", i+1,
    burst_time[i], total_arrival_time[i],
    total_arrival_time[i] - burst_time[i]);
  wait_time = wait_time + total_arrival_time[i]
              - burst_time[i];
  turn arround_time = turn_arround_time
        + total_arrival_time[i];
  counter = 0;
}
if (i == limit-1)
{
  i=0;
}
else if (arrival_time[i+1] <= total)
```

```
    } i++;
    }
    else
    {
      i=0;
    }
}
average_wait_time = wait_time * 1.0 / limit;
average_turnaround_time = turnaround_time
                * 1.0 / limit;
printf ("\n Average waiting time : %.2f",
                        average_wait_time);
printf ("\n Avg Turn around Time : %.2f",
                        average_turnaround_time);

return 0;
}
```

output :-

Enter the total no of processes : 4
Arrival Time of P_1 : 0
Burst Time of P_1 : 8
Arrival Time of P_2 : 1
Burst Time of P_2 : 4
Arrival Time of P_3 : 2
Burst Time of P_3 : 9
Arrival Time of P_4 : 3
Burst Time of P_4 : 5
Enter time quantum : 4

process	Burst Time	Turnaround Time	Waiting Time
P_2	4	7	3
P_1	8	20	12
P_4	5	22	17
P_3	9	24	15

Average Waiting Time : 11.75
Average Turnaround Time : 18.25