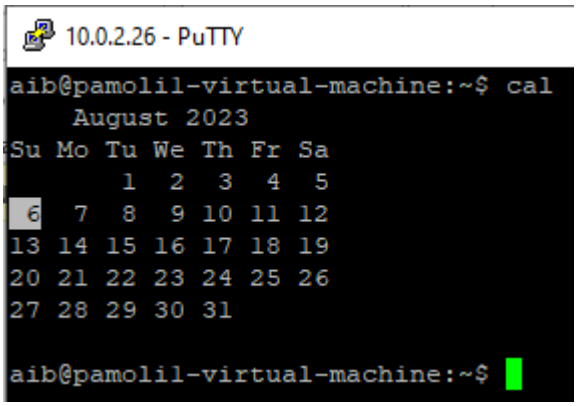
Date- 31.07.2023
Monday

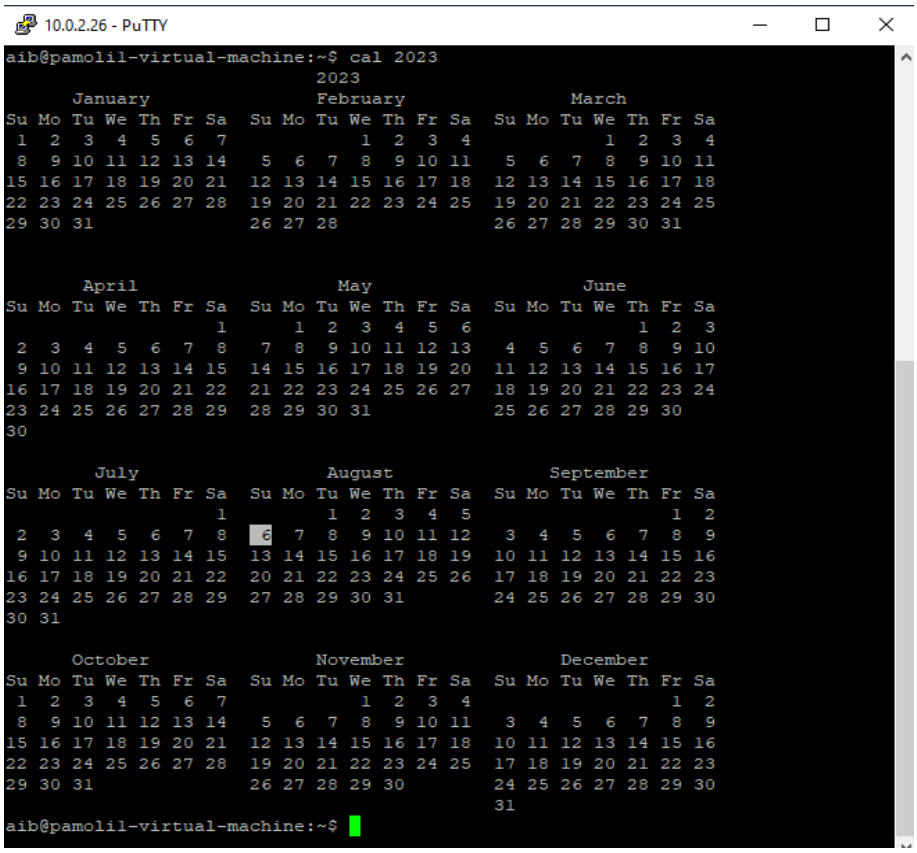
Experiment -1

Himanshu Kumar saw
A023119822010

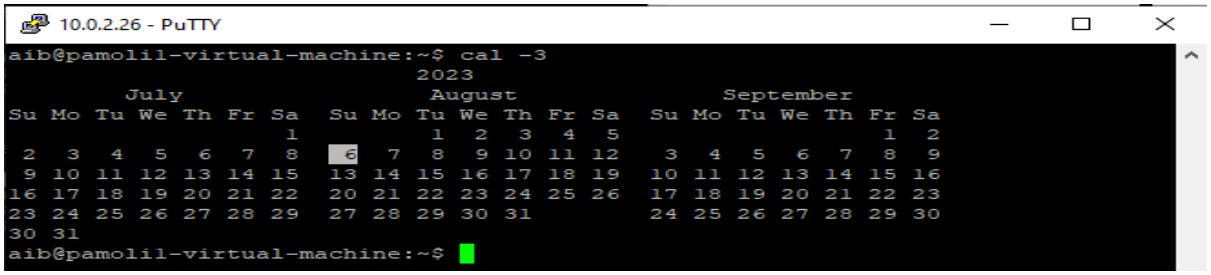
1.$ Cal:- It is used to see calendar of specific month or year.



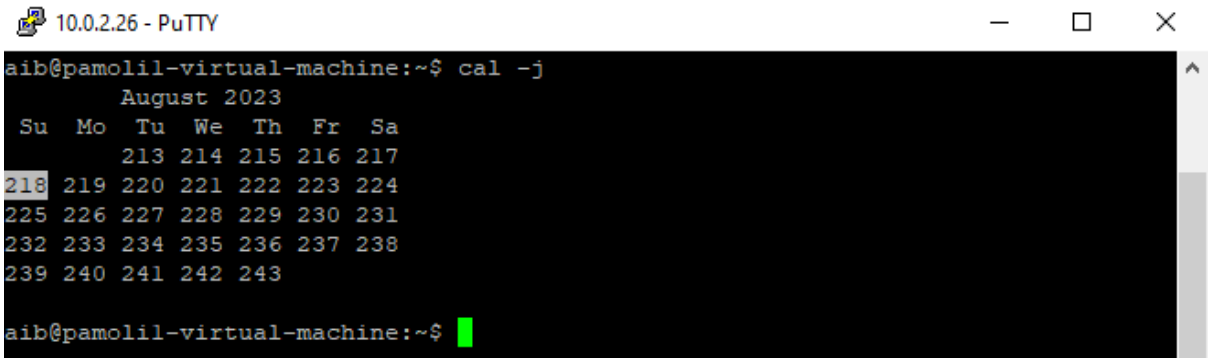
$ cal 2023:- It is used to get the current calendar of the specified year.



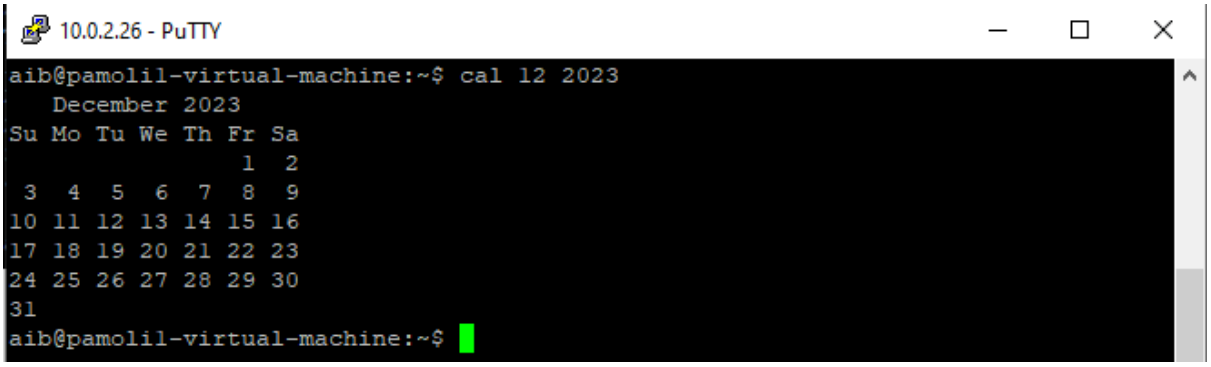
$ cal -3:- It is used the get the calendar of first specified month.



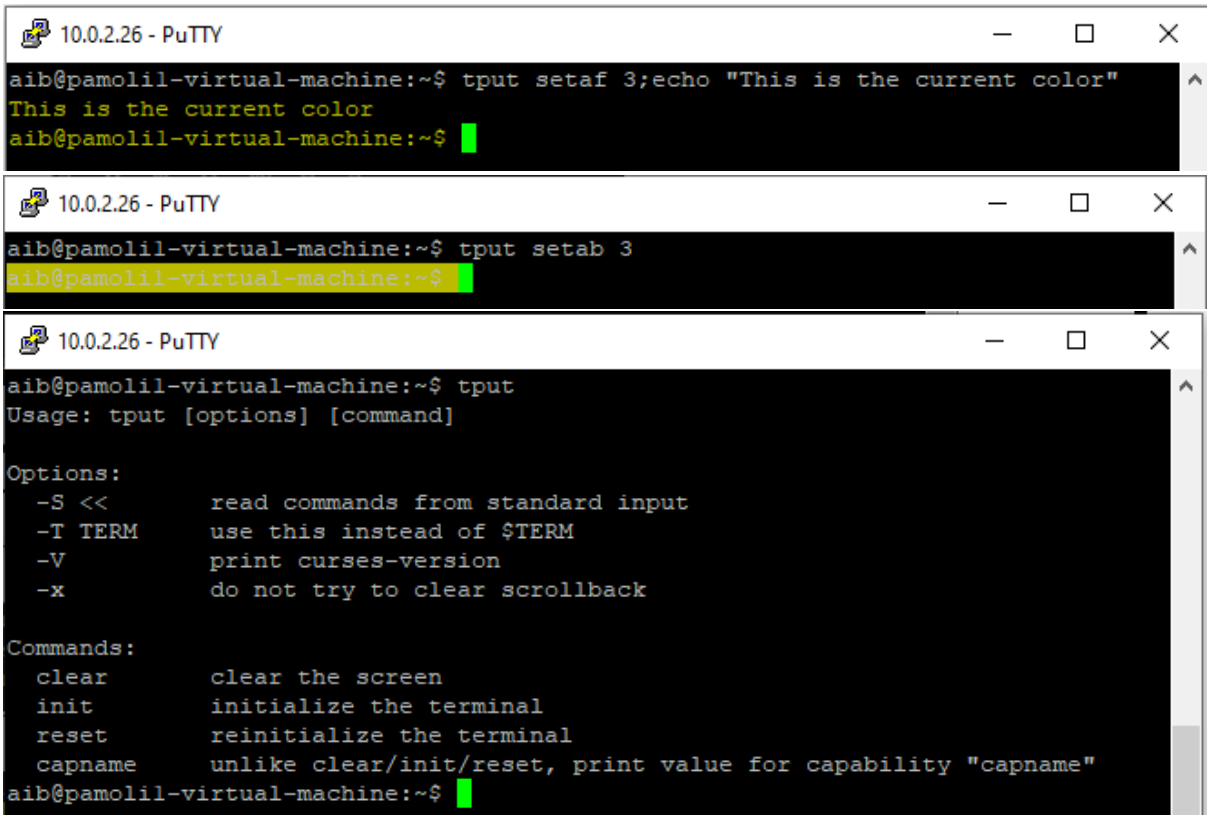
$ cal -j:- It is used to get the calendar of current month with current number of day in the year.



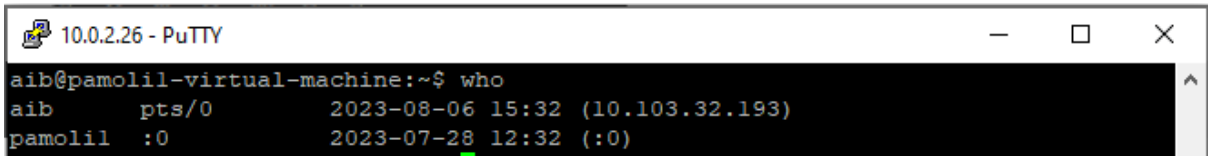
$ cal [month][year]:- It is used to get the calendar of specified month and year.



2. $ tput:- It is used to access terminal properties and manipulates how properties such as colors,position are displayed in it.

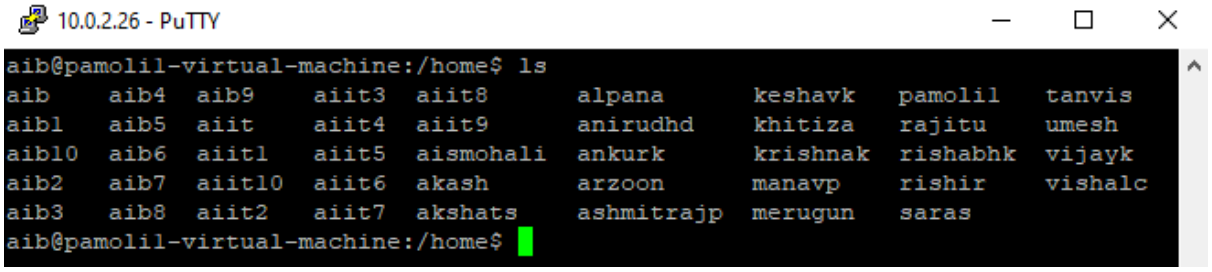


3.$ who:- It is used to get the details of all the user currently logged in the server or computer.

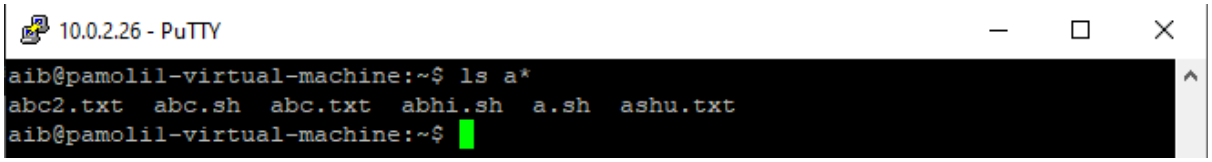


4.$ Ps:- It is used to list the currently running processes and their PIDs along with some other information.

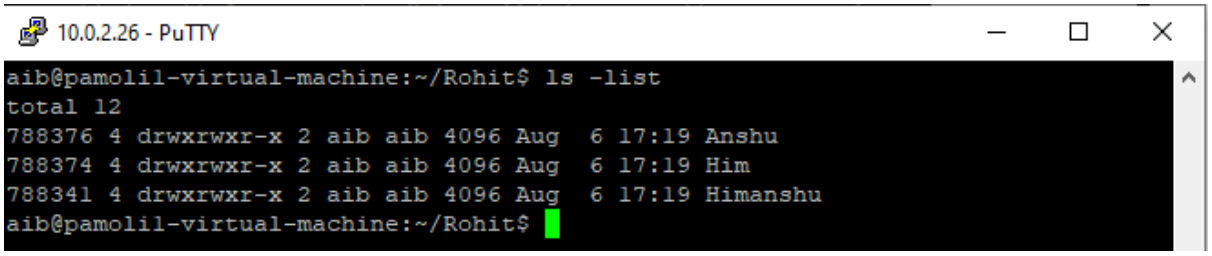
5.ls:- It is used to get the names of all the files and directories present inside the current working directory.



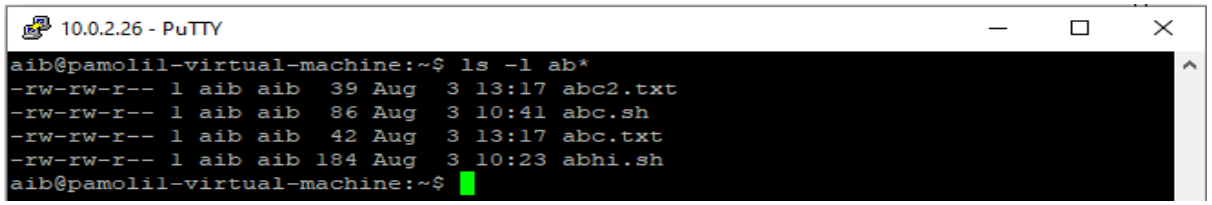
$ls a*:- It is used to get the names of the files starting with ‘a’ in the current directory.



$ls -list:- It is used to get the details of the files inside the current directory.

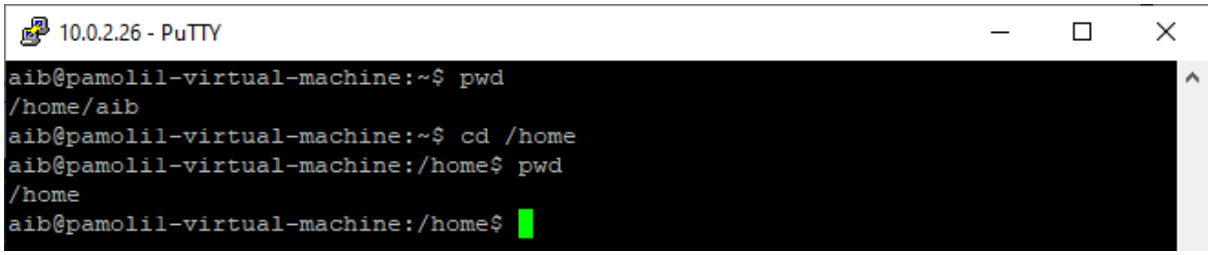


$ls -l ab*:- It is used to get the details of the files starting with ‘ab’ in the current directory.

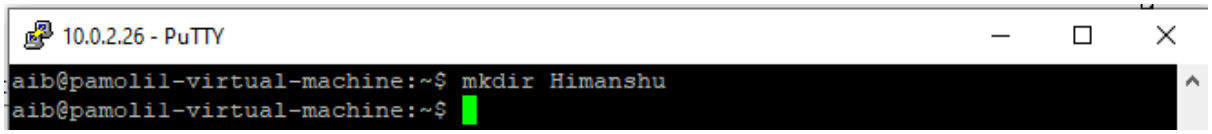


6. $ pwd:- It is used to get the current working directory.

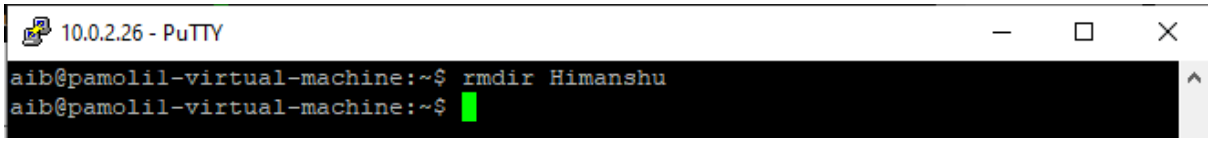
7.$cd :- It is used the change the current working directory.



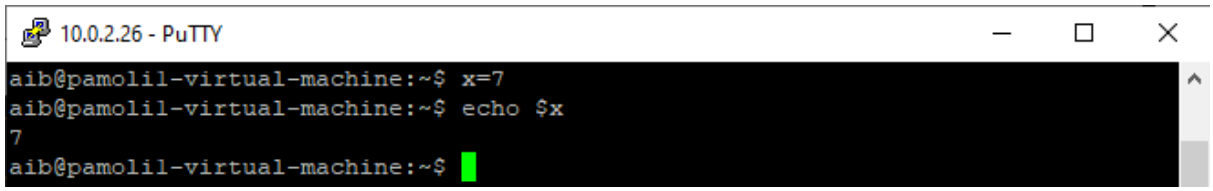
8. $mkdir :- It is used to make the directory inside the current working directory.



9. $rmdir:- It is used to delete an unempty directory inside the current directory.



10. $echo:- It is used to print the variable or string.

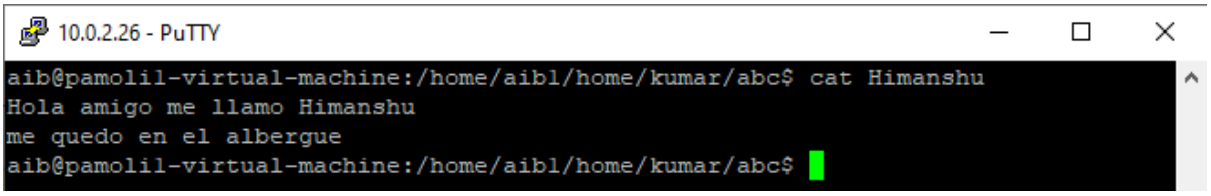


Experiment-2

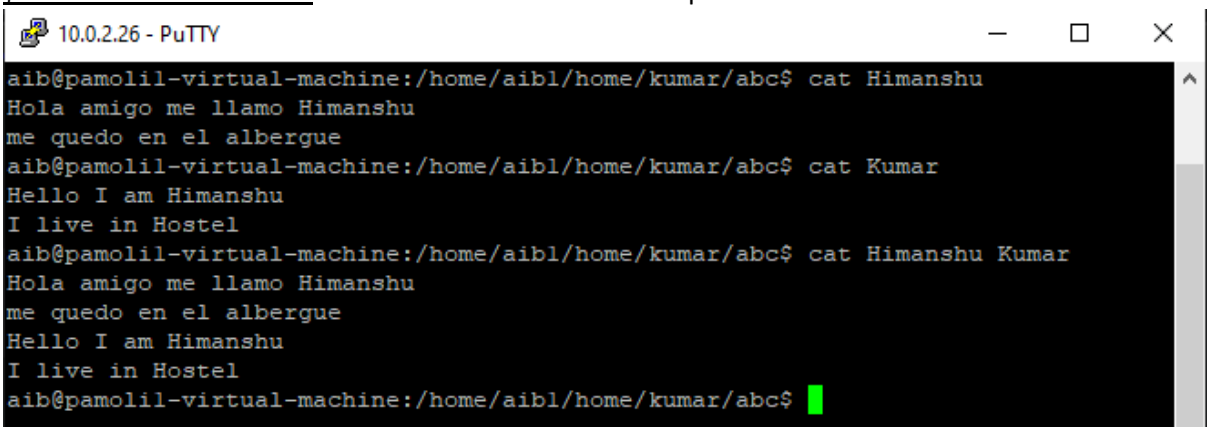
1.$cat:- It provides us space to write content in our own language. It helps to create ,read and concatenate files.

We used crtl+D to exit from the write mode.

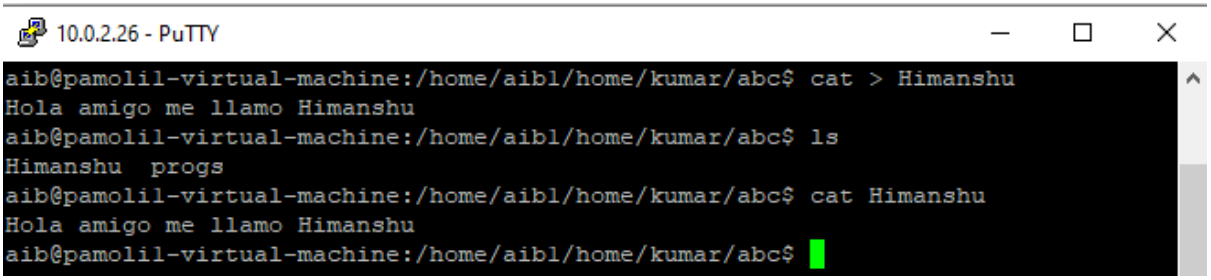
$cat filename:- It is used to read a single file within a directory.



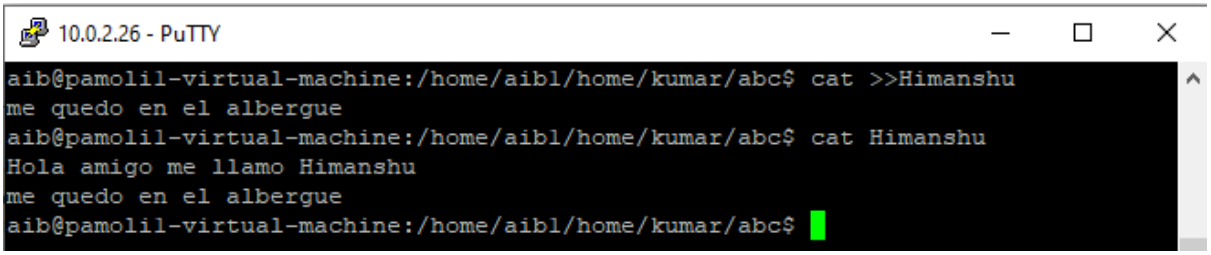
$cat filename1 filename 2:- It is used to read two or multiple file.



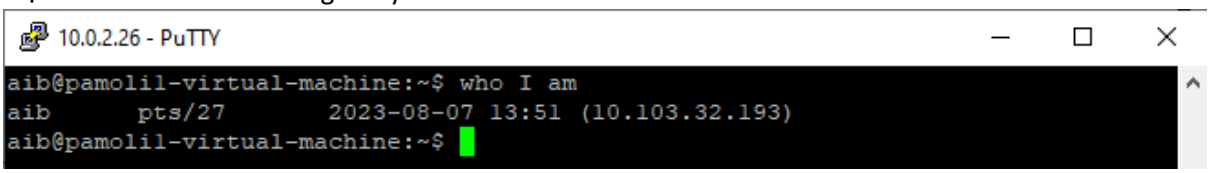
$cat > newfilename:- It is used to create a new file inside the current directory.



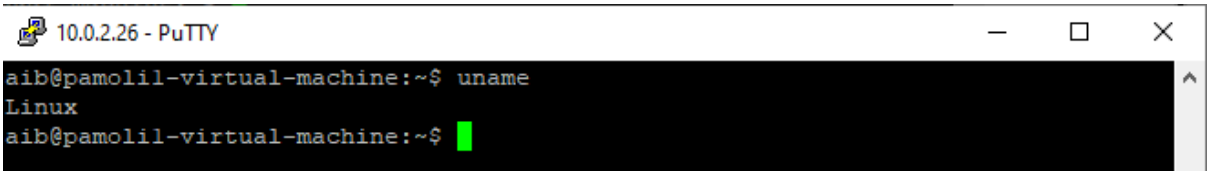
$cat >> filename:- It is used to add content in the existing file.



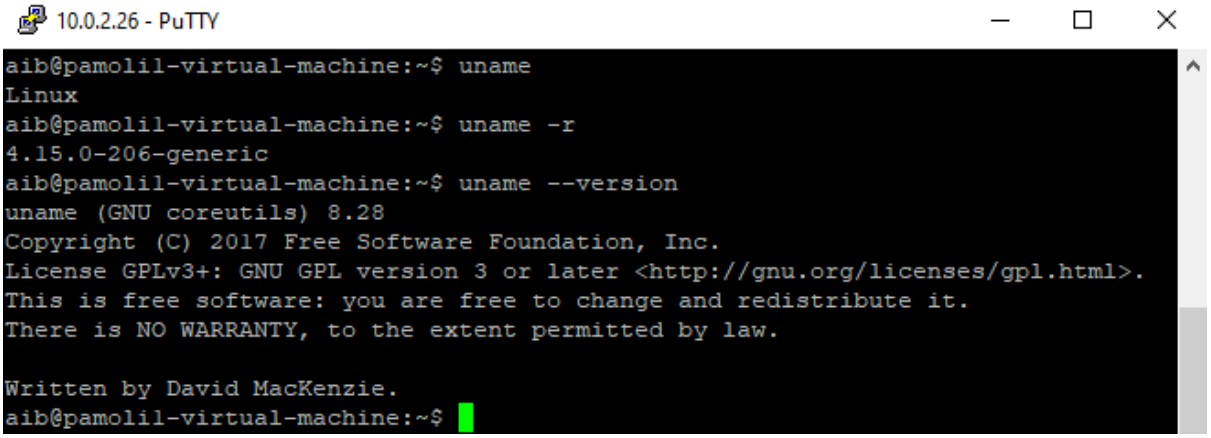
2.$who I am:- It is used to get my current user details.



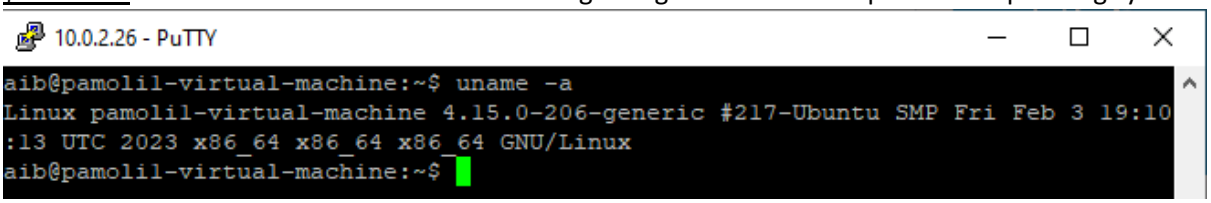
3.$uname:- It is used to provide the name of current operating system.



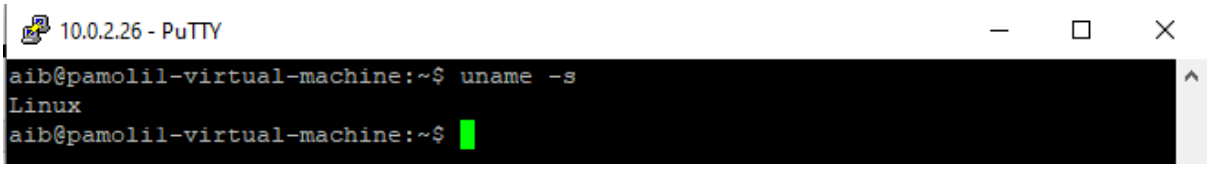
$uname –version:- It is used to get the current version of operating system.



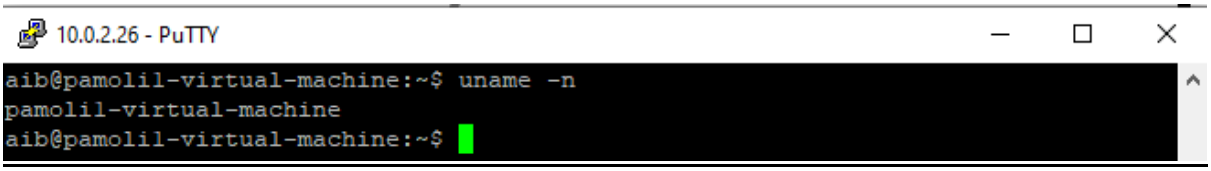
$uname -a:- It is used to view the information regarding the current computer and operating system.



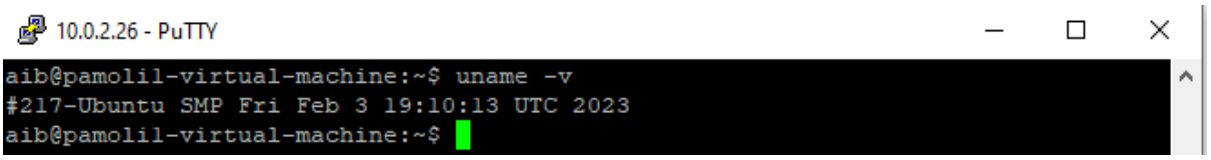
$uname -s:- It is used to used to display only kernal name.



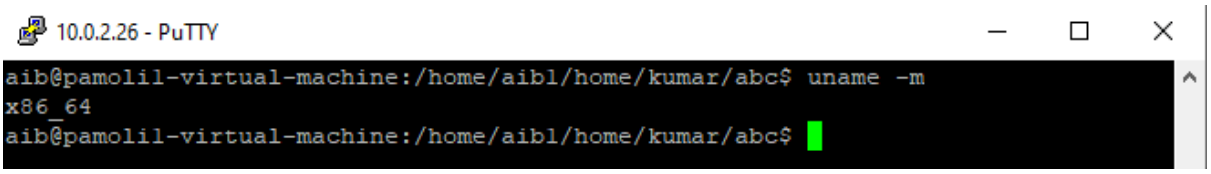
$uname -n:- Hostname is the name by which the computer is identified in the local network. To display the hostname with the uname command, run the command.



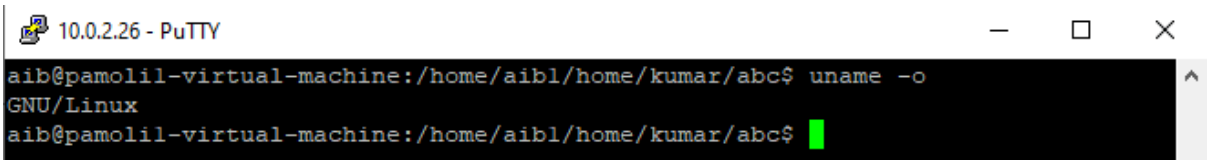
$uname -v:- It is used to print the current version of kernal.



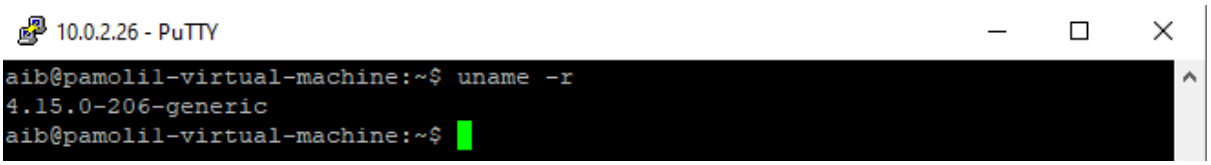
$uname -m:- Machine name is the computer's hardware, which can be displayed using the "**-m**" option with the uname command.



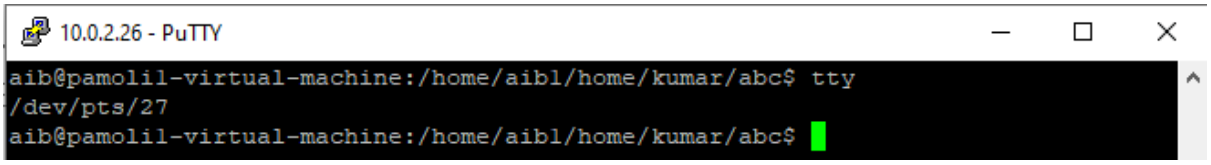
$uname -o:- It is used to get the information about the current operating system.



$uname -r**:-** It is used to release the current kernal.

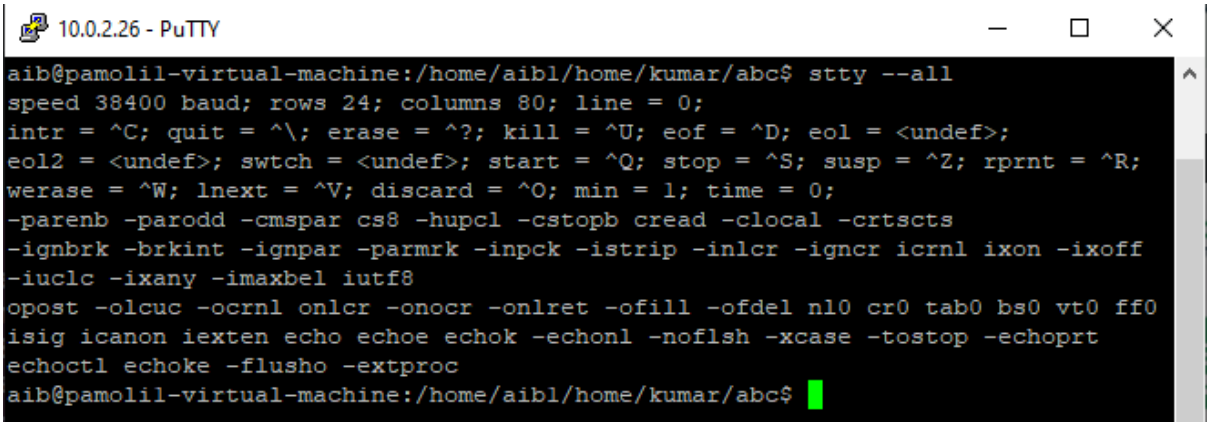


4.$tty:- It is used to get the current terminal name.tty is short form of teletype also know as terminal.

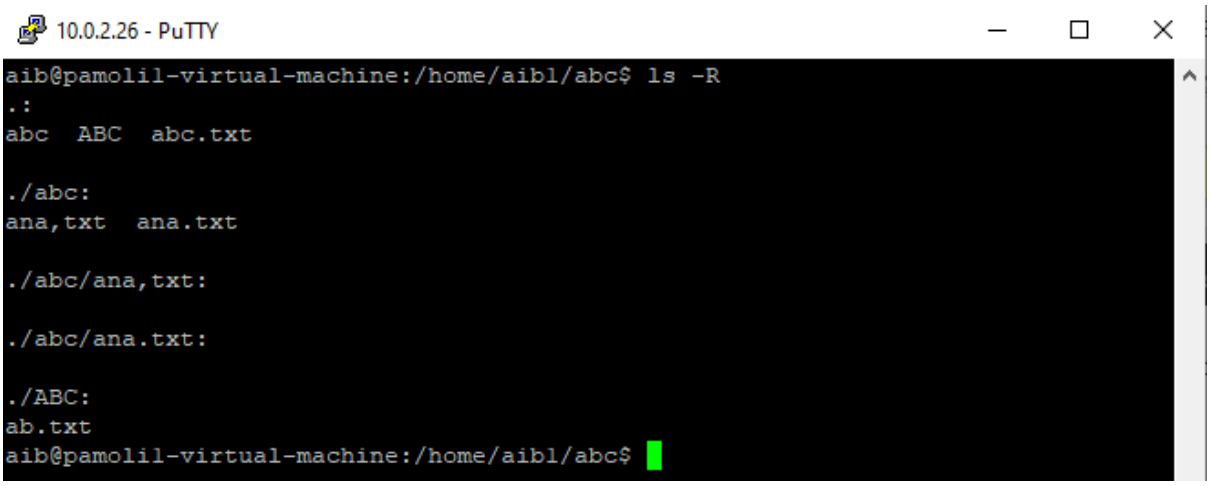


5.$stty:- It is used to change or print the current terminal line setting.

$stty –all:- It is used to get all the current setting of terminal in human readable format.

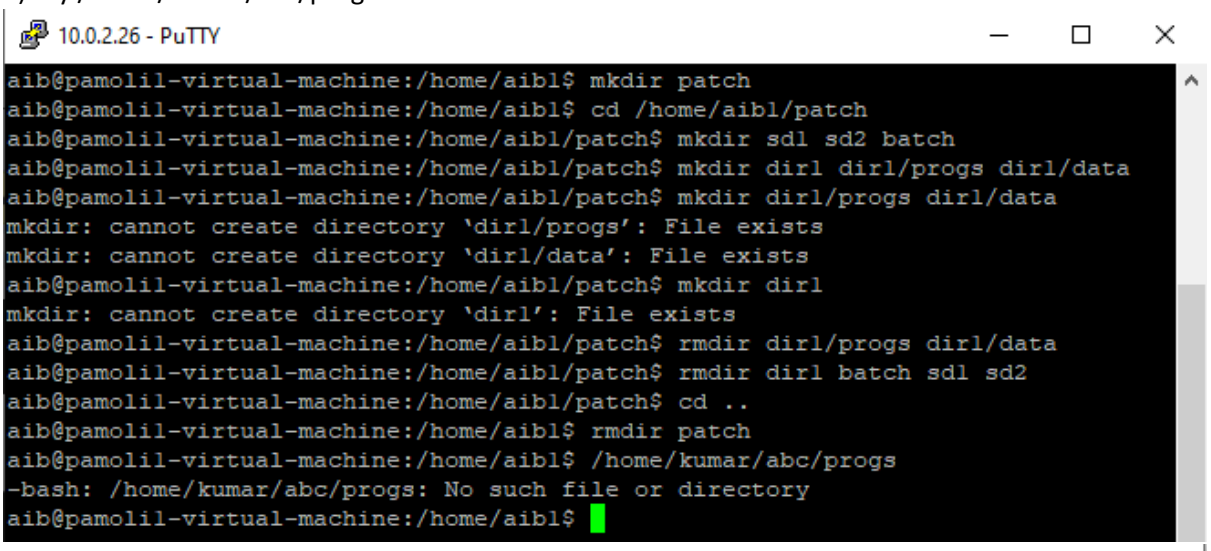


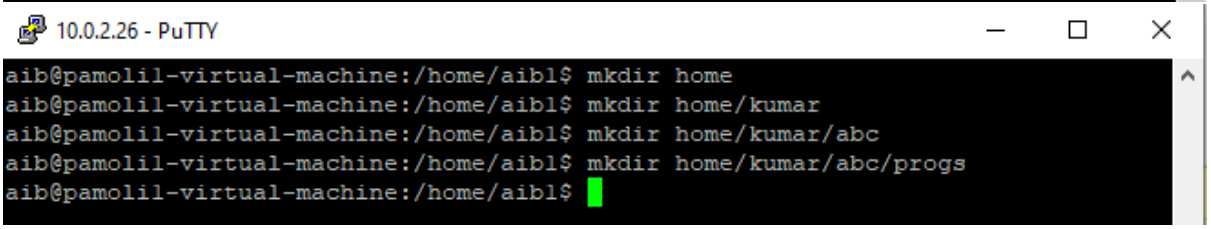
6.$ls -R:-It is used to get all the directories name including its all subdirectories.



Task :- 2

1)create a directory patch under current directory
2)create three subdirectories batch, sd1, sd2 with single commands
3)create a directory tree Dir 1 Dir1/progs Dir/data
4)mkdir dir1/progs dir1/data whether it will give error or not
5) Try to create an already existing directory
6)Remove your directory which you have creates in task 3
7) Try /home/kumar/abc/progs





Experiment-3

Write a shell script to check whether a given number is even or odd

Program:

```
1 echo "Enter the number: "
2 read n
3 r='expr $n % 2'
4 if [ $r -eq 0 ]
5 then
6 echo "$n is Even number"
7 else
8 echo "$n is odd number"
9 fi
10
```

Ouput:

```
┌──(kali㉿kali)-[~/Desktop/Operating System]
└─$ ./even_odd_check.sh
Enter the number:
3
./even_odd_check.sh: 4: [: expr: unexpected operator
3 is odd number
```

Experiment-4

Write a shell script to swap to number.

Program:

```
echo "enter first number"
read a
echo "enter second number"
read b
echo "a before swapping is $a and b is $b"
#swapping
a=$((a+b))
b=$((a - b))
a=$((a-b))
echo "a after swapping is  $a and b is $b"

```

Output:

```
┌──(kali㉿kali)-[~/Desktop/Operating System]
└─$ ./Swaping.sh
enter first number
2
enter second number
3
a before swapping is 2 and b is 3
a after swapping is  3 and b is 2
```

Experiment-5

Write a shell script to calculate the factorial of a given number.

Program:

```
echo "Enter a number"
read num
fact=1
while [ $num -gt 1 ]
do
  fact=$((fact * num))  #fact = fact * num
  num=$((num - 1))      #num = num - 1
done
echo $fact

```

Output:

```
┌──(kali㉿kali)-[~/Desktop/Operating System]
└─$ ./Factorial.sh
Enter a number
4
24
```

Experiment-6

Q2. Write a shell script to check whether a given year is leap or not.

Program:

```
1 echo "LEAP YEAR SHELL SCRIPT"
2 echo -n "Enter a year:"
3 read year_checker
4 if [ `expr $year_checker % 4` -eq 0 ]
5 then
6         echo "$year_checker is a leap year"
7 else
8         echo "$year_checker is not a leap year"
9 fi
10
```

Output:

```
┌──(kali㉿kali)-[~/Desktop/Operating System]
└─$ ./Leap_check.sh
LEAP YEAR SHELL SCRIPT
Enter a year:2020
2020 is a leap year

┌──(kali㉿kali)-[~/Desktop/Operating System]
└─$ ./Leap_check.sh
LEAP YEAR SHELL SCRIPT
Enter a year:2022
2022 is not a leap year
```

Experiment-7

Write a program to implement FCFS scheduling algorithm

Code:

```
#include<stdio.h>
int main(){
    int bursttime[20],waitingtime[20],tat[20],i,n;
    float avgtat,avgwt;
    printf("Enter the processes: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        printf("Enter the burst time for process id[%d]: ",i+1);
        scanf("%d",&bursttime[i]);
    }
    waitingtime[0]=avgwt=0;
    tat[0]=avgtat=bursttime[0];
    for(int i=1;i<n;i++){
        waitingtime[i]=waitingtime[i-1]+bursttime[i-1];
        tat[i]=tat[i-1]+bursttime[i];
        avgwt=avgwt+waitingtime[i];
        avgtat=(avgtat+tat[i]);
    }
    printf("\tProcesses\tBurst Time\tWating Time\tTurnaround Time");
    for(i=0;i<n;i++)
printf("\n\tP%d\t\t%d\t\t\t%d\t\t%d",i,bursttime[i],waitingtime[i],tat[i]);
    printf("\nAverage waiting time: %f",avgwt/n);
    printf("\nAverage turnaround time: %f",avgtat/n);
}
```

Output:

```
Enter the processes: 5
Enter the burst time for process id[1]: 2
Enter the burst time for process id[2]: 4
Enter the burst time for process id[3]: 3
Enter the burst time for process id[4]: 3
Enter the burst time for process id[5]: 1
    Processes    Burst Time    Wating Time    Turnaround Time
    P0              2             0          2
    P1              4             2          6
    P2              3             6          9
    P3              3             9          12
    P4              1             12         13
Average waiting time: 5.800000
Average turnaround time: 8.400000
```

Experiment – 8

Write a program to implement SJF scheduling algorithm

Code:

```
#include<stdio.h>
int main(){
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
    float avg_wt,avg_tat;
    printf("Enter number of processes: ");
    scanf("%d",&n);
    printf("Enter Burst time: \n");
    for(i=0;i<n;i++){
        printf("P%d: ",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;
    }
    for(i=0;i<n;i++){
        pos=i;
        for(j=i+1;j<n;j++){
            if(bt[j]<bt[pos]) pos=j;
        }
        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;
        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }
    wt[0]=0;
    for(i=1;i<n;i++){
        wt[i]=0;
        for(j=0;j<i;j++) wt[i]+=bt[j];
        total+=wt[i];
    }
    avg_wt=(float)total/n;
    total=0;
    printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++){
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("\nP%d\t %d\t\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }
    avg_tat=(float)total/n;
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\n\nAverage Turnaround Time=%f",avg_tat);
}
```

Output:

Enter number of processes: 5

Enter Burst time:

P1: 6

P2: 2

P3: 4

P4: 3

P5: 1

Process	Burst Time	Waiting Time	Turnaround Time
P5	1	0	1
P2	2	1	3
P4	3	3	6
P3	4	6	10
P1	6	10	16

Average Waiting Time=4.000000

Average Turnaround Time=7.200000

Experiment-9

Write a program to implement round robin scheduling algorithm

Code:

```
#include<stdio.h>
void main(){
    int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10],
temp[10];
    float avg_wt, avg_tat;
    printf(" Total number of process in the system: ");
    scanf("%d", &NOP);
    y = NOP;
    printf("Enter the Arrival Time of: \n");
    for(i=0; i<NOP; i++){
        printf("P[%d]: ",i+1);
        scanf("%d",&at[i]);}
    printf("Enter the Burst Time of: \n");
    for(i=0; i<NOP; i++){
        printf("P[%d]: ",i+1);
        scanf("%d", &bt[i]);
        temp[i] = bt[i];}
    printf("Enter the Time Quantum for the process: ");
    scanf("%d",&quant);
    printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
    for(sum=0, i = 0; y!=0; ){
        if(temp[i] <= quant && temp[i] > 0){
            sum = sum + temp[i];
            temp[i] = 0;
            count=1;
        }
        else if(temp[i] > 0){
            temp[i] = temp[i] - quant;
            sum = sum + quant;
        }
        if(temp[i]==0 && count==1){
            y--;
            printf("\nProcess No[%d] \t\t %d\t\t\t%d\t\t\t %d", i+1, bt[i],
sum-at[i], sum-at[i]-bt[i]);
            wt = wt+sum-at[i]-bt[i];
            tat = tat+sum-at[i];
            count =0;
        }
        if(i==NOP-1) i=0;
        else if(at[i+1]<=sum) i++;
        else i=0;
    }
    avg_wt = wt * 1.0/NOP;
    avg_tat = tat * 1.0/NOP;
    printf("\n Average Turn Around Time: \t%f", avg_wt);
    printf("\n Average Waiting Time: \t%f", avg_tat);}
```

Output:

Total number of process in the system: 4

Enter the Arrival Time of:

P[1]: 0

P[2]: 1

P[3]: 2

P[4]: 3

Enter the Burst Time of:

P[1]: 8

P[2]: 5

P[3]: 10

P[4]: 11

Enter the Time Quantum for the process: 6

Process No	Burst Time	TAT	Waiting Time
Process No[2]	5	10	5
Process No[1]	8	25	17
Process No[3]	10	27	17
Process No[4]	11	31	20

Average Turn Around Time: 14.750000

Average Waiting Time: 23.250000

Experiment-10

Write a program to implement Preemptive priority scheduling round robin scheduling algorithm.

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

3

Enter the arrival time , burst time and priority of the process

AT BT PT

0 7 2

2 2 1

4 1 3

ID WT TAT

1 2 9

2 0 2

3 5 6

Avg waiting time of the process is 2.333333

Avg turn around time of the process is 5.666667