Name: Pratyush Kumar
Reg. No.: 19BCE0506
Course: CSE2005, Operating Systems
Slot: L41+L42

DIGITAL ASSIGNMENT - 1

1. CPU Scheduling Algorithms

a. FCFS Scheduling Algorithm

Aim: We are given with the n number of processes i.e. P1, P2, P3,.......,Pn and their corresponding burst times. The task is to find the average waiting time and average turnaround time using FCFS CPU Scheduling algorithm.

Algorithm:

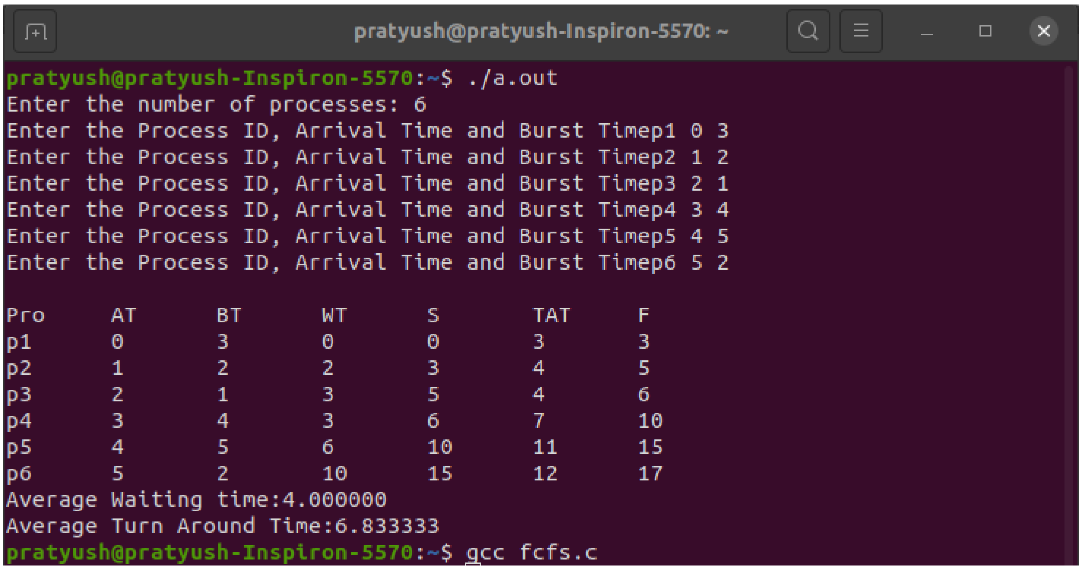
```
Start
    Accept the number of process
    Then the process_id, arrival time and its burst time
Then,
    Sort the process table in ascending order of their arrival
times using a temp variable
        temp=arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
        temp=bur[i];
        bur[i]=bur[j];
        bur[j]=temp;
        strcpy(t,pn[i]);
        strcpy(pn[i],pn[j]);
        strcpy(pn[j],t);
Then,
    using the arrays star[] and finish[], calculate
the start and finish time of each process
    We use the formulae,
        TAT = Completion time – AT
        WT = TAT – BT
Average TAT = Sum of tat[i]/number of process
Average WT = Sum of WT[i]/number of process
```

Code:

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
int main()
{
	char pn[10][10], t[10];
	int arr[10], bur[10], star[10], finish[10], tat[10], wt[10], i, j, n, temp;
```

```
        int totwt = 0, tottat = 0;
        printf("Enter the number of processes: ");
        scanf("%d", &n);
        for (i = 0; i < n; i++)
        {
                printf("Enter the Process ID, Arrival Time and Burst Time");
                scanf("%s%d%d", pn[i], &arr[i], &bur[i]);
        }
        for (i = 0; i < n; i++)
        {
                for (j = 0; j < n; j++)
                {
                        if (arr[i] < arr[j])
                        {
                                temp = arr[i];
                                arr[i] = arr[j];
                                arr[j] = temp;
                                temp = bur[i];
                                bur[i] = bur[j];
                                bur[j] = temp;
                                strcpy(t, pn[i]);
                                strcpy(pn[i], pn[j]);
                                strcpy(pn[j], t);
                        }
                }
        }
        for (i=0; i<n; i++)
        {
        if (i == 0)
                        star[i] =arr[i];
        else
                        star[i] = finish[i-1];
        wt[i] = star[i] - arr[i];
        finish[i] = star[i] +bur[i];
        tat[i] = finish[i] - arr[i];
        }
        printf("\nPro  AT      BT      WT      S       TAT     F");
        for(i=0; i<n; i++)
        {
        printf("\n%s\t%d\t%d\t%d\t%d\t%d\t%d", pn[i], arr[i], bur[i], wt[i], star[i], tat[i],
finish[i]);
        totwt += wt[i];
        tottat += tat[i];
        }
        printf("\nAverage Waiting time:%f",(float)totwt/n);
        printf("\nAverage Turn Around Time:%f",(float)tottat/n);
        printf("\n");
        return 0;
}
```

Output:



```
pratyush@pratyush-Inspiron-5570:~$ ./a.out
Enter the number of processes: 6
Enter the Process ID, Arrival Time and Burst Timep1 0 3
Enter the Process ID, Arrival Time and Burst Timep2 1 2
Enter the Process ID, Arrival Time and Burst Timep3 2 1
Enter the Process ID, Arrival Time and Burst Timep4 3 4
Enter the Process ID, Arrival Time and Burst Timep5 4 5
Enter the Process ID, Arrival Time and Burst Timep6 5 2

Pro     AT      BT      WT      S       TAT     F
p1      0       3       0       0       3       3
p2      1       2       2       3       4       5
p3      2       1       3       5       4       6
p4      3       4       3       6       7       10
p5      4       5       6       10      11      15
p6      5       2       10      15      12      17
Average Waiting time:4.000000
Average Turn Around Time:6.833333
pratyush@pratyush-Inspiron-5570:~$ gcc fcfs.c
```

b. Shortest-Job-First Scheduling Algorithm (SJF)

Aim: Given process, the burst time of a process repsecively and a quantum limit; the task is to find and print the waiting time, turnaround time and their average time using Shortest Job First Scheduling non-preemptive method.

Algorithm:

1. Sort all the process according to the arrival time.

2. Then select that process which minimum arrival time and minimum Burst time.

3. After completion of process make a pool of process which after till the completion of previos process and select that process among the pool which is having minimum Burst Time.

How to compute below times in SJF using a program

1. Completion time: Time at which process compmletes its execution

2. Turn Around Time: Completion time – Arrival time

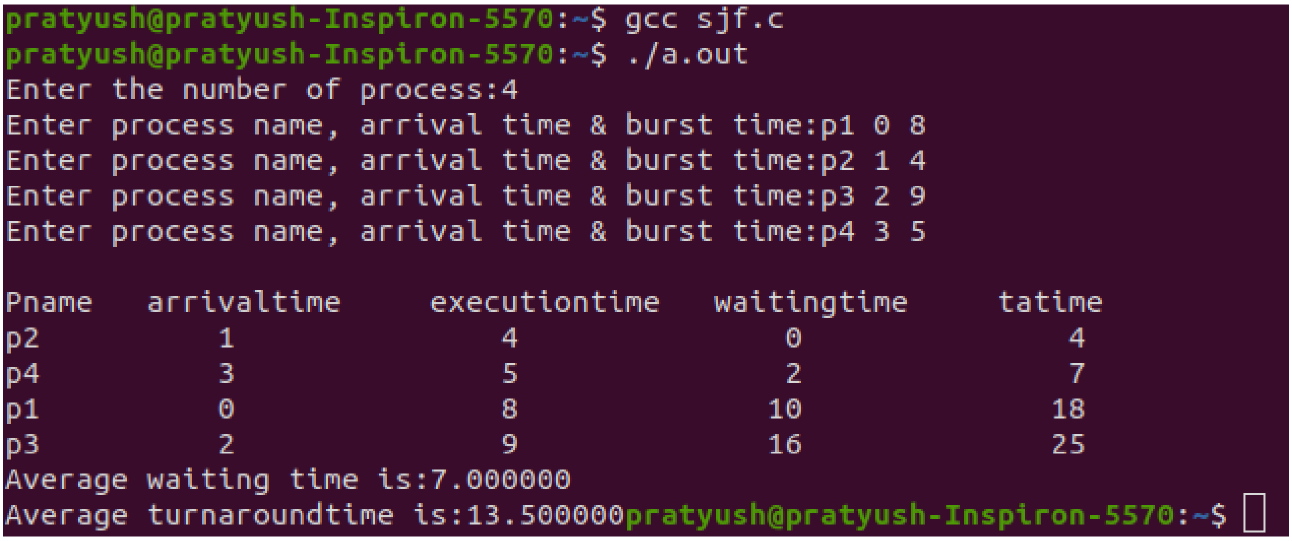
3. Waiting Time: TAT – Burst Time

Code:

```
#include<stdio.h>
#include<string.h>
#include<unistd.h>
int main()
{
   int bt[20], at[10], n, i, j, temp, st[10], ft[10], wt[10], tat[10];
        //bt[]->burst time, at[]-> arrival time, st[]->start time, wt[]->waiting time, tat[]->turn
around time
   int totwt = 0, tottat = 0;
```

```
    float awt, atat;
    char pn[10][10], t[10];//pn -> process number
    printf("Enter the number of process:");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        printf("Enter process name, arrival time & burst time:");
        scanf("%s%d%d", pn[i], &at[i], &bt[i]);
    }
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
        {
            if (bt[i] < bt[j])
            {
                temp = at[i];
                at[i] = at[j];
                at[j] = temp;
                temp = bt[i];
                bt[i] = bt[j];
                bt[j] = temp;
                strcpy(t, pn[i]);
                strcpy(pn[i], pn[j]);
                strcpy(pn[j], t);
            }
        }
    for (i = 0; i < n; i++)
    {
        if (i == 0)
            st[i] = at[i];
        else
            st[i] = ft[i-1];
        wt[i] = st[i] - at[i];
        ft[i] = st[i] + bt[i];
        tat[i] = ft[i] - at[i];
        totwt += wt[i];
        tottat += tat[i];
    }
    awt = (float)totwt/n;
    atat = (float)tottat/n;
    printf("\nPname\tarrivaltime\texecutiontime\twaitingtime\ttatime");
    for (i = 0; i < n; i++)
        printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d", pn[i], at[i], bt[i], wt[i], tat[i]);
    printf("\nAverage waiting time is:%f", awt);
    printf("\nAverage turnaroundtime is:%f", atat);
    return 0;
}
```

Output:



```
pratyush@pratyush-Inspiron-5570:~$ gcc sjf.c
pratyush@pratyush-Inspiron-5570:~$ ./a.out
Enter the number of process:4
Enter process name, arrival time & burst time:p1 0 8
Enter process name, arrival time & burst time:p2 1 4
Enter process name, arrival time & burst time:p3 2 9
Enter process name, arrival time & burst time:p4 3 5

Pname   arrivaltime     executiontime   waitingtime     tatime
p2          1               4               0               4
p4          3               5               2               7
p1          0               8              10              18
p3          2               9              16              25
Average waiting time is:7.000000
Average turnaroundtime is:13.500000pratyush@pratyush-Inspiron-5570:~$ 
```

c. Shortest Remaining Time First Scheduling Algorithm (SRTF)

Aim: To implement SJF scheduling algorithm in preemptive method

Algorithm:

```
1. Traverse until all process gets completely executed.
        a. Find process with minimum remaining time at every single time lap..
        b. Reduce its time by 1.
        c. Check if its remaining time becomes 0.
        d. Increment the counter of process completion.
        e. Complettion time of current process = current_time + 1.
        f. Calculate waiting time for each completed process.
                wt[i] = completion time – arrival_time - burst_time
        g. increment time lap by one
2. Find turn around time (waiting time + burst_time).
```

Code:

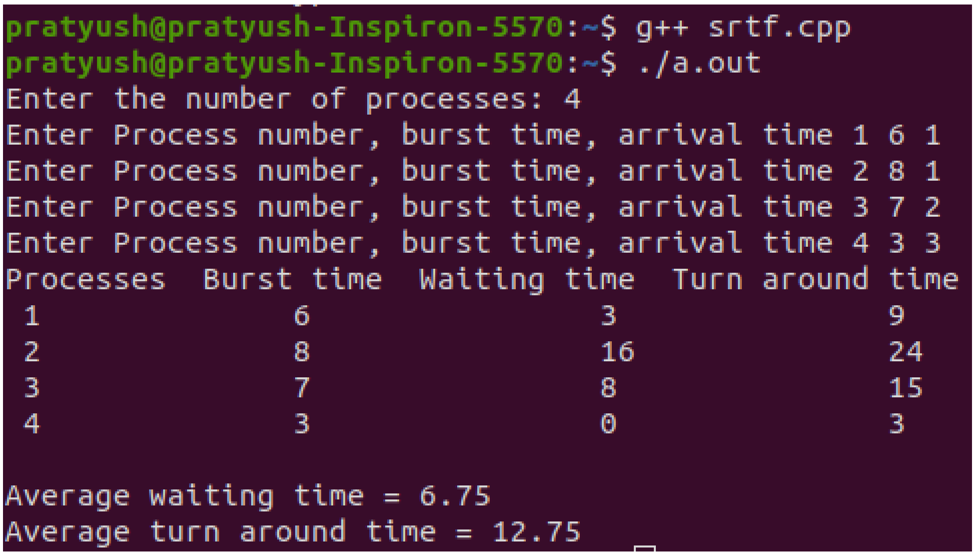
```
#include <bits/stdc++.h>
#include <unistd.h>
using namespace std;

struct Process {
        int pid; // Process ID
        int bt; // Burst Time
        int art; // Arrival Time
};

// Function to find the waiting time for all processes
void findWaitingTime(Process proc[], int n, int wt[])
{
        int rt[n];
```

```
// Copy the burst time into rt[]
for (int i = 0; i < n; i++)
    rt[i] = proc[i].bt;

int complete = 0, t = 0, minm = INT_MAX;
int shortest = 0, finish_time;
bool check = false;

// Process until all processes gets completed
while (complete != n) {

    // Find process with minimum
    // remaining time among the
    // processes that arrives till the
    // current time`
    for (int j = 0; j < n; j++) {
        if ((proc[j].art <= t) &&
        (rt[j] < minm) && rt[j] > 0) {
            minm = rt[j];
            shortest = j;
            check = true;
        }
    }

    if (check == false) {
        t++;
        continue;
    }

    // Reduce remaining time by one
    rt[shortest]--;

    // Update minimum
    minm = rt[shortest];
    if (minm == 0)
        minm = INT_MAX;

    // If a process gets completely
    // executed
    if (rt[shortest] == 0) {

        // Increment complete
        complete++;
        check = false;

        // Find finish time of current
        // process
        finish_time = t + 1;

        // Calculate waiting time
        wt[shortest] = finish_time -
                    proc[shortest].bt -
```

```
                                                            proc[shortest].art;

                                    if (wt[shortest] < 0)
                                            wt[shortest] = 0;
                            }
                            // Increment time
                            t++;
                    }
            }

            // Function to calculate turn around time
            void findTurnAroundTime(Process proc[], int n, int wt[], int tat[])
            {
                    // calculating turnaround time by adding
                    // bt[i] + wt[i]
                    for (int i = 0; i < n; i++)
                            tat[i] = proc[i].bt + wt[i];
            }

            // Function to calculate average time
            void findavgTime(Process proc[], int n)
            {
                    int wt[n], tat[n], total_wt = 0, total_tat = 0;

                    // Function to find waiting time of all
                    // processes
                    findWaitingTime(proc, n, wt);

                    // Function to find turn around time for
                    // all processes
                    findTurnAroundTime(proc, n, wt, tat);

                    // Display processes along with all
                    // details
                    cout << "Processes "
                            << " Burst time "
                            << " Waiting time "
                            << " Turn around time\n";

                    // Calculate total waiting time and
                    // total turnaround time
                    for (int i = 0; i < n; i++) {
                            total_wt = total_wt + wt[i];
                            total_tat = total_tat + tat[i];
                            cout << " " << proc[i].pid << "\t\t"
                                    << proc[i].bt << "\t\t " << wt[i]
                                    << "\t\t " << tat[i] << endl;
                    }

                    cout << "\nAverage waiting time = "
                            << (float)total_wt / (float)n;
                    cout << "\nAverage turn around time = "
```

```
                << (float)total_tat / (float)n;
        cout << "\n";
}

// Driver code
int main()
{
        int n;
        cout << "Enter the number of processes: ";
        cin >> n;
        Process proc[n];
        for (int i = 0; i < n; i++)
        {
                cout << "Enter Process number, burst time, arrival time ";
                cin >> proc[i].pid >> proc[i].bt >> proc[i].art;
        }
        findavgTime(proc, n);
        return 0;
}
```

Output:



```
pratyush@pratyush-Inspiron-5570:~$ g++ srtf.cpp
pratyush@pratyush-Inspiron-5570:~$ ./a.out
Enter the number of processes: 4
Enter Process number, burst time, arrival time 1 6 1
Enter Process number, burst time, arrival time 2 8 1
Enter Process number, burst time, arrival time 3 7 2
Enter Process number, burst time, arrival time 4 3 3
Processes  Burst time  Waiting time  Turn around time
 1            6             3              9
 2            8             16             24
 3            7             8              15
 4            3             0              3

Average waiting time = 6.75
Average turn around time = 12.75
```

d. Priority Scheduling Algorithm (Non-Preemptive)

Aim: To implemet non-preemptive priority Cpu scheduling algorthm in C++ language. In Non-Preemptive Priority Scheduling there is a priority assigned to each process and processes are executed according to their priority and since it is non-preemptive so a process can't be preempted by another process in the midst of execution of a process.

Algorithm:

1. First input the processes with their arrival time, burst time and priority.

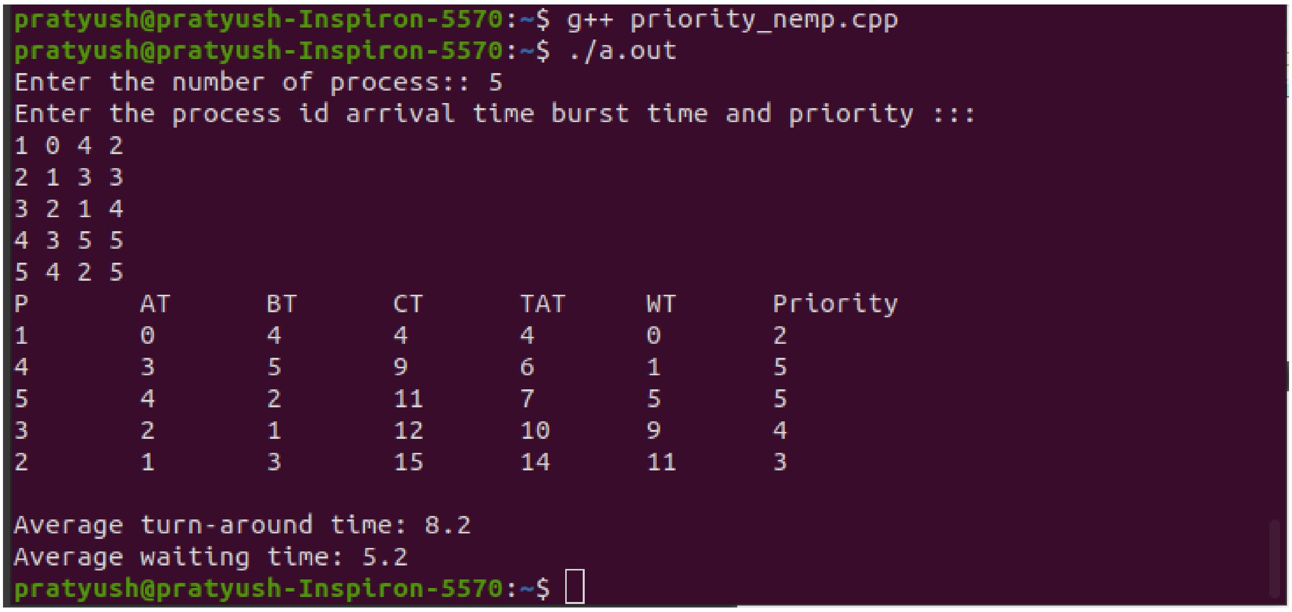
2. Sort the processes, according to arrival time if two process arrival time is same then sort according process priority if two process priority are same then sort according to process number.
3. Now simply apply FCFS algorithm.

Code:

```
//Implementation of Priority(Non-Preeemptive)
#include <iostream>
#include <algorithm>
#include <unistd.h>
using namespace std;

typedef struct proccess
{
 int at,bt,pr,ct,ta,wt;
 string pro_id;

     /*
 artime = Arrival time,
 bt = Burst time,
 ct = Completion time,
 ta = Turn around time,
 wt = Waiting time
 */

}process;

bool compare(process a,process b)
{
 return a.at < b.at;
   /* This schedule will always return TRUE
if above condition comes*/

}

bool compare2(process a,process b)
{
 return a.pr > b.pr;
  /* This schedule will always return TRUE
if above condition comes*/

}

int main()
{
 process pro[10];
 int n, i, j;
 cout << "Enter the number of process:: ";
 cin >> n;
 cout << "Enter the process id arrival time burst time and priority ::: ";
 for ( i=0; i < n; i++)
 {
  cin >> pro[i].pro_id;
  cin >> pro[i].at;
  cin >> pro[i].bt;
  cin >> pro[i].pr;
 }

 sort(pro,pro+n,compare);

  /*sort is a predefined function  defined in algorithm.h header file,
it will sort the schedules according to their arrival time*/

 pro[0].ct = pro[0].bt + pro[0].at;
 pro[0].ta = pro[0].ct - pro[0].at;
 pro[0].wt = pro[0].ta - pro[0].bt;
 i = 1;

 while(i < n-1)
 {

  for (j = i; j < n; j++)
  {
   if (pro[j].at > pro[i-1].ct)
   break;
  }
  sort (pro+i,pro+i+(j-i),compare2);
  pro[i].ct = pro[i-1].ct + pro[i].bt;
  pro[i].ta = pro[i].ct - pro[i].at;
  pro[i].wt = pro[i].ta - pro[i].bt;
  i++;
  }
  pro[i].ct = pro[i-1].ct + pro[i].bt;
  pro[i].ta = pro[i].ct - pro[i].at;
  pro[i].wt = pro[i].ta - pro[i].bt;

 cout << "P    AT      BT      CT      TAT     WT      Priority\n";
 for (i = 0; i < n; i++)
 {
  //displaying all the values
  cout << pro[i].pro_id << "\t" << pro[i].at << "\t"<<pro[i].bt << "\t" << pro[i].ct << "\t" <<
pro[i].ta << "\t" << pro[i].wt << "\t" << pro[i].pr;
  cout << endl;
 }
        float avg_TAT = 0, avg_WT = 0;
        for (i = 0; i < n; i++)
        {
                avg_TAT += pro[i].ta;
                avg_WT += pro[i].wt;
        }
        avg_TAT = (float)avg_TAT/n;
```

```
          avg_WT = (float) avg_WT/n;
          cout << "\nAverage turn-around time: " << avg_TAT;
          cout << "\nAverage waiting time: " << avg_WT;
          cout << "\n";
 return 0;
}
```

Output:



```
pratyush@pratyush-Inspiron-5570:~$ g++ priority_nemp.cpp
pratyush@pratyush-Inspiron-5570:~$ ./a.out
Enter the number of process:: 5
Enter the process id arrival time burst time and priority :::
1 0 4 2
2 1 3 3
3 2 1 4
4 3 5 5
5 4 2 5
P       AT      BT      CT      TAT     WT      Priority
1       0       4       4       4       0       2
4       3       5       9       6       1       5
5       4       2       11      7       5       5
3       2       1       12      10      9       4
2       1       3       15      14      11      3

Average turn-around time: 8.2
Average waiting time: 5.2
pratyush@pratyush-Inspiron-5570:~$ 
```

e. Priority Scheduling Algorithm (Preemptive)

Aim: To implement Priority Scheduling algorithm in Preemptive mode

Algorithm:

First input the processes with their arrival time, burst time and priority.

Sort the processes, according to arrival time if two process arrival time is same then sort according process priority if two process priority are same then sort according to process number.

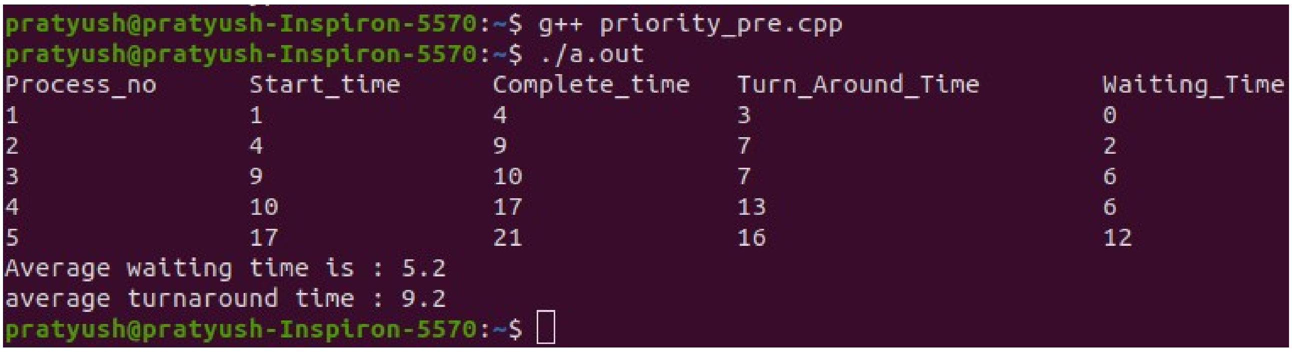
Now simply apply FCFS algorithm.

Code:

```
//C++ implementation for Priority Scheduling with
//Different Arrival Time priority scheduling
/*1. sort the processes according to arrival time
2. if arrival time is same the acc to priority
3. apply fcfs
*/

#include <bits/stdc++.h>
#include <unistd.h>
```

```
using namespace std;

#define totalprocess 5

// Making a struct to hold the given input

struct process
{
        int at,bt,pr,pno;
};

process proc[50];

/*
Writing comparator function to sort according to priority if
arrival time is same
*/

bool comp(process a,process b)
{
        if(a.at == b.at)
        {
                return a.pr<b.pr;
        }
        else
        {
                return a.at<b.at;
        }
}

// Using FCFS Algorithm to find Waiting time
void get_wt_time(int wt[])
{
// declaring service array that stores cumulative burst time
        int service[50];

// Initilising initial elements of the arrays
        service[0] = proc[0].at;
        wt[0]=0;


        for(int i = 1; i < totalprocess; i++)
        {
                service[i] = proc[i-1].bt + service[i-1];

                wt[i] = service[i] - proc[i].at;

// If waiting time is negative, change it into zero

                if(wt[i]<0)
                {
                        wt[i]=0;
                }
        }
}

void get_tat_time(int tat[],int wt[])
{
        // Filling turnaroundtime array

        for(int i = 0; i < totalprocess; i++)
        {
                tat[i] = proc[i].bt + wt[i];
        }
}

void findgc()
{
        //Declare waiting time and turnaround time array
        int wt[50],tat[50];

        double wavg=0,tavg=0;

        // Function call to find waiting time array
        get_wt_time(wt);
        //Function call to find turnaround time
        get_tat_time(tat,wt);

        int stime[50],ctime[50];

        stime[0] = proc[0].at;
        ctime[0] = stime[0]+tat[0];

        // calculating starting and ending time
        for(int i = 1; i < totalprocess; i++)
        {
        stime[i] = ctime[i-1];
        ctime[i] = stime[i] + tat[i] - wt[i];
        }

        cout<<"Process_no\tStart_time\tComplete_time\tTurn_Around_Time\
tWaiting_Time"<<endl;

        // display the process details

        for (int i = 0; i < totalprocess; i++)
        {
        wavg += wt[i];
        tavg += tat[i];

        cout << proc[i].pno << "\t\t" << stime[i] << "\t\t" << ctime[i] << "\t\t"<<          tat[i]<<"\t\
t\t"<<wt[i]<<endl;
        }
      // display the average waiting time
      //and average turn around time

        cout << "Average waiting time is : ";
        cout << wavg/(float)totalprocess<<endl;
        cout << "average turnaround time : ";
        cout << tavg/(float)totalprocess<<endl;

}

int main()
{
        int arrivaltime[] = { 1, 2, 3, 4, 5 };
        int bursttime[] = { 3, 5, 1, 7, 4 };
        int priority[] = { 3, 4, 1, 7, 8 };

        for(int i=0;i<totalprocess;i++)
        {
                proc[i].at=arrivaltime[i];
                proc[i].bt=bursttime[i];
                proc[i].pr=priority[i];
                proc[i].pno=i+1;
        }

        //Using inbuilt sort function

        sort(proc,proc+totalprocess,comp);

        //Calling function findgc for finding Gantt Chart

        findgc();

        return 0;
}
```

Output:



```
pratyush@pratyush-Inspiron-5570:~$ g++ priority_pre.cpp
pratyush@pratyush-Inspiron-5570:~$ ./a.out
Process_no      Start_time      Complete_time   Turn_Around_Time        Waiting_Time
1               1               4               3                       0
2               4               9               7                       2
3               9               10              7                       6
4               10              17              13                      6
5               17              21              16                      12
Average waiting time is : 5.2
average turnaround time : 9.2
pratyush@pratyush-Inspiron-5570:~$ 
```

f. Round-Robin Scheduling Algorithm

Aim: To implement Round Robin CPU Scheduling Algorithm with arrival time variant in C language

Algorithm:

1. The queue structure in ready queue is of First In First Out (FIFO) type.

2. A fixed time is allotted to every process that arrives in the queue. This fixed time is known as time slice or time quantum.

3. The first process that arrives is selected and sent to the processor for execution. If it is not able to complete its execution within the time quantum provided, then an interrupt is generated using an automated timer.

4. The process is then stopped and is sent back at the end of the queue. However, the state is saved and context is thereby stored in memory. This helps the process to resume from the point where it was interrupted.

5. The scheduler selects another process from the ready queue and dispatches it to the processor for its execution. It is executed until the time Quantum does not exceed.

6. The same steps are repeated until all the process are finished.

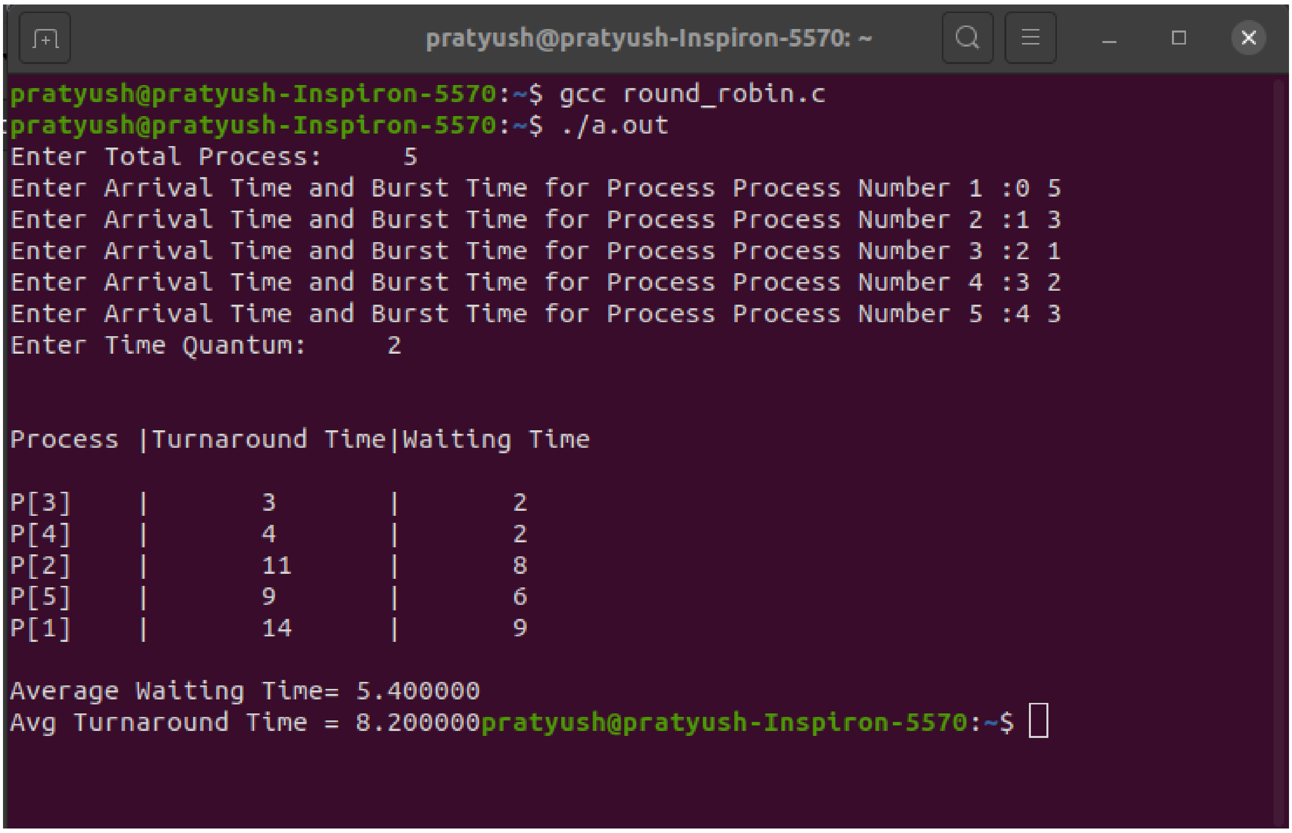
The round robin algorithm is simple and the overhead in decision making is very low. It is the best scheduling algorithm for achieving better and evenly distributed response time.

Code:

```
#include <stdio.h>

#include <unistd.h>

int main()

{


        int count,j,n,time,remain,flag=0,time_quantum;

        int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];

        printf("Enter Total Process:\t ");

        scanf("%d",&n);

        remain=n;

        for(count=0;count<n;count++)

        {

                printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);

                scanf("%d",&at[count]);
        scanf("%d",&bt[count]);
        rt[count]=bt[count];
    }
    printf("Enter Time Quantum:\t");
    scanf("%d",&time_quantum);
    printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
    for(time=0,count=0;remain!=0;)
    {
        if(rt[count]<=time_quantum && rt[count]>0)
        {
            time+=rt[count];
            rt[count]=0;
            flag=1;
        }
        else if(rt[count]>0)
        {
            rt[count]-=time_quantum;
            time+=time_quantum;
        }
        if(rt[count]==0 && flag==1)
        {
            remain--;
            printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-
bt[count]);
            wait_time+=time-at[count]-bt[count];
            turnaround_time+=time-at[count];
            flag=0;
        }
        if(count==n-1)
          count=0;
        else if(at[count+1]<=time)
          count++;
```

```
                else
                        count=0;
        }
        printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
        printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

        return 0;
}
```

<u>Output</u>:

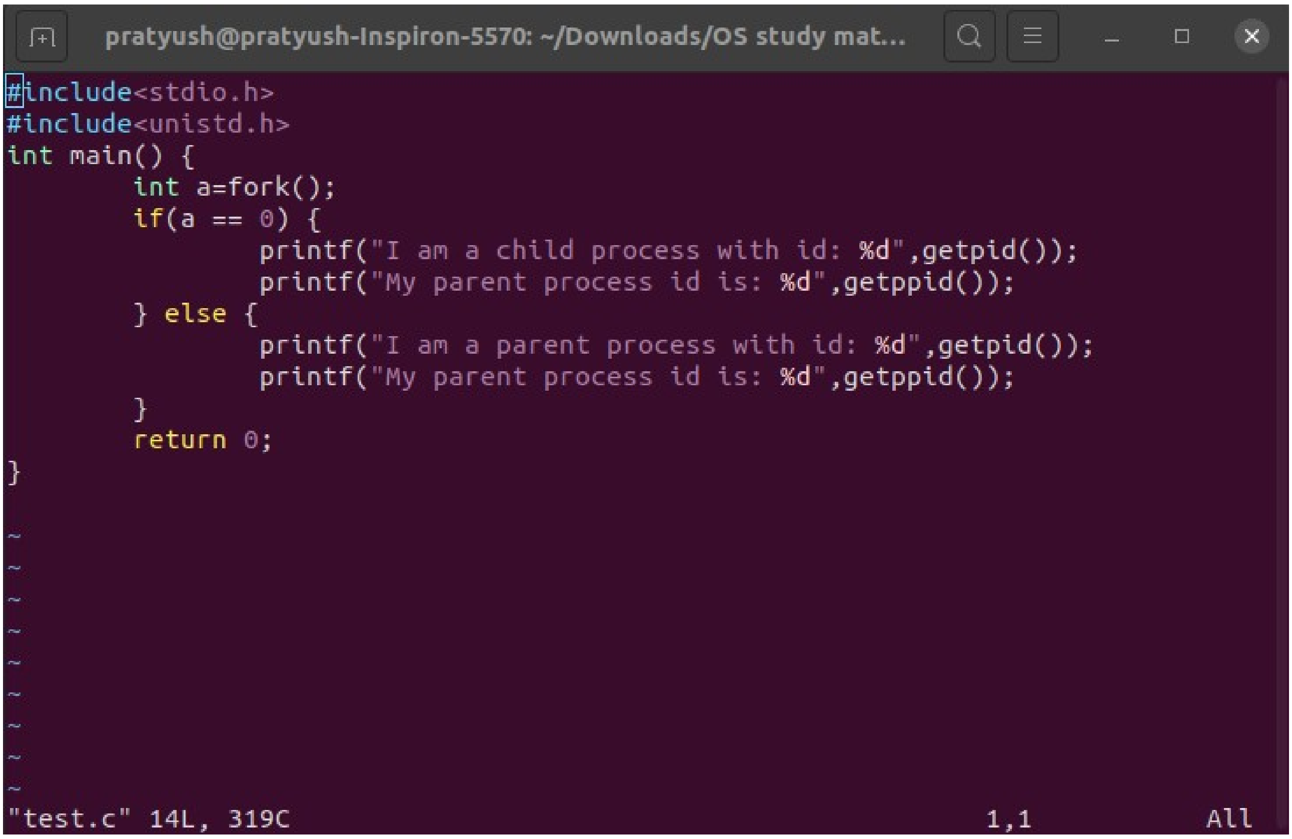


```
pratyush@pratyush-Inspiron-5570:~$ gcc round_robin.c
pratyush@pratyush-Inspiron-5570:~$ ./a.out
Enter Total Process:     5
Enter Arrival Time and Burst Time for Process Process Number 1 :0 5
Enter Arrival Time and Burst Time for Process Process Number 2 :1 3
Enter Arrival Time and Burst Time for Process Process Number 3 :2 1
Enter Arrival Time and Burst Time for Process Process Number 4 :3 2
Enter Arrival Time and Burst Time for Process Process Number 5 :4 3
Enter Time Quantum:     2


Process |Turnaround Time|Waiting Time

P[3]    |       3       |       2
P[4]    |       4       |       2
P[2]    |       11      |       8
P[5]    |       9       |       6
P[1]    |       14      |       9

Average Waiting Time= 5.400000
Avg Turnaround Time = 8.200000pratyush@pratyush-Inspiron-5570:~$ 
```

2. Process

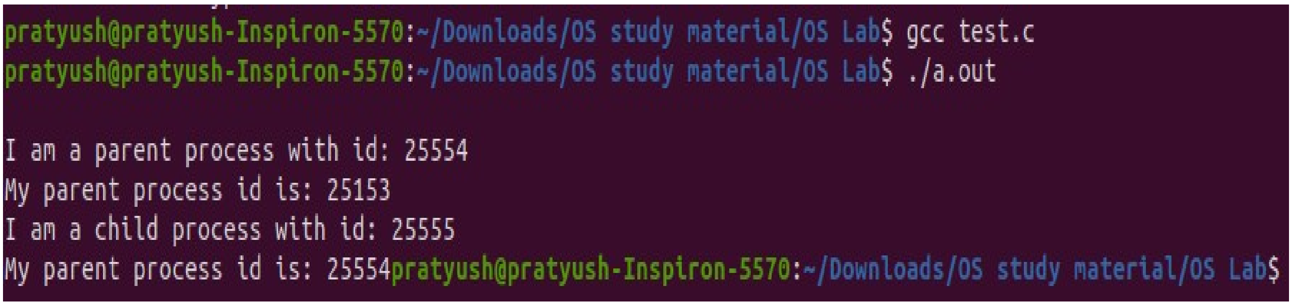
a. Process Creation

Code:


pratyush@pratyush-Inspiron-5570: ~/Downloads/OS study mat...

```
#include<stdio.h>
#include<unistd.h>
int main() {
        int a=fork();
        if(a == 0) {
                printf("I am a child process with id: %d",getpid());
                printf("My parent process id is: %d",getppid());
        } else {
                printf("I am a parent process with id: %d",getpid());
                printf("My parent process id is: %d",getppid());
        }
        return 0;
}

~
~
~
~
~
~
~
~
~
"test.c" 14L, 319C                                            1,1           All
```

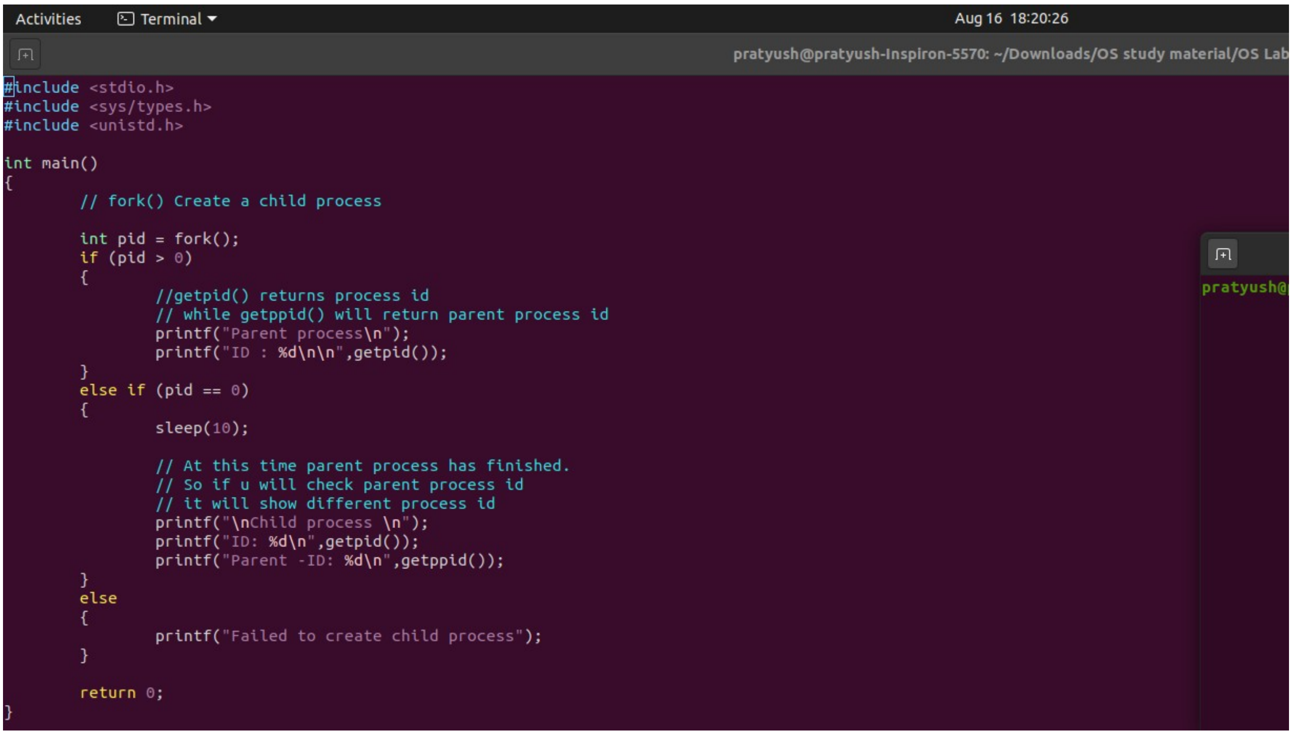
Output:



```
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ gcc test.c
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ./a.out

I am a parent process with id: 25554
My parent process id is: 25153
I am a child process with id: 25555
My parent process id is: 25554pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$
```

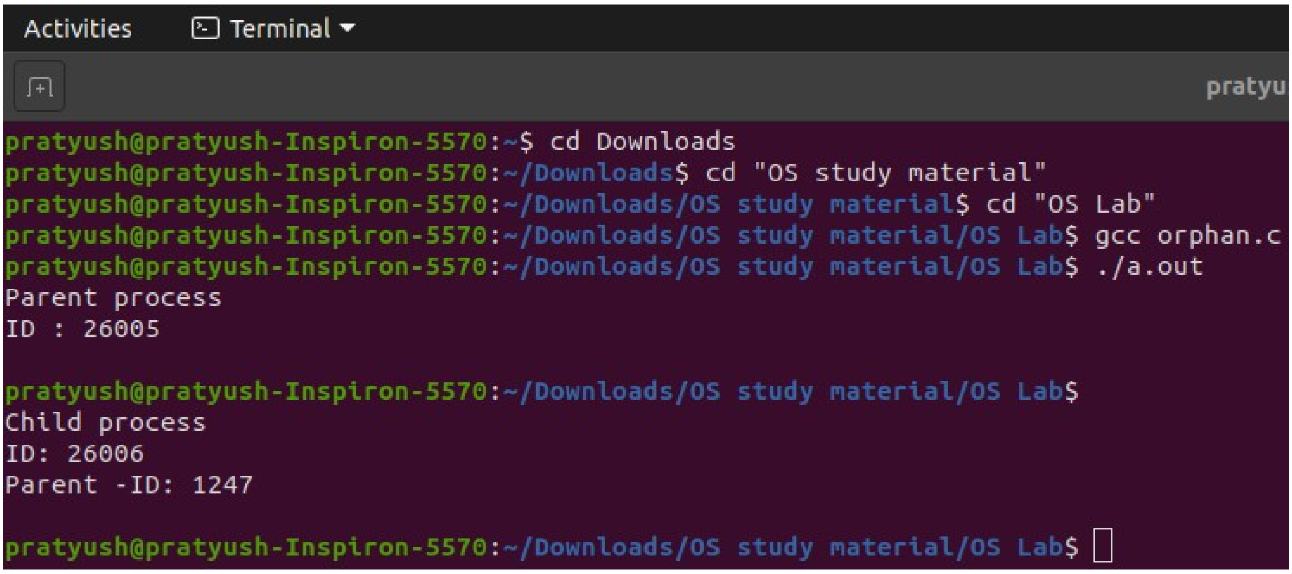
b. Orphan Process

Code:


Activities
Terminal ▾
Aug 16 18:20:26
pratyush@pratyush-Inspiron-5570: ~/Downloads/OS study material/OS Lab
pratyush@

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
        // fork() Create a child process

        int pid = fork();
        if (pid > 0)
        {
                //getpid() returns process id
                // while getppid() will return parent process id
                printf("Parent process\n");
                printf("ID : %d\n\n",getpid());
        }
        else if (pid == 0)
        {
                sleep(10);

                // At this time parent process has finished.
                // So if u will check parent process id
                // it will show different process id
                printf("\nChild process \n");
                printf("ID: %d\n",getpid());
                printf("Parent -ID: %d\n",getppid());
        }
        else
        {
                printf("Failed to create child process");
        }

        return 0;
}
```

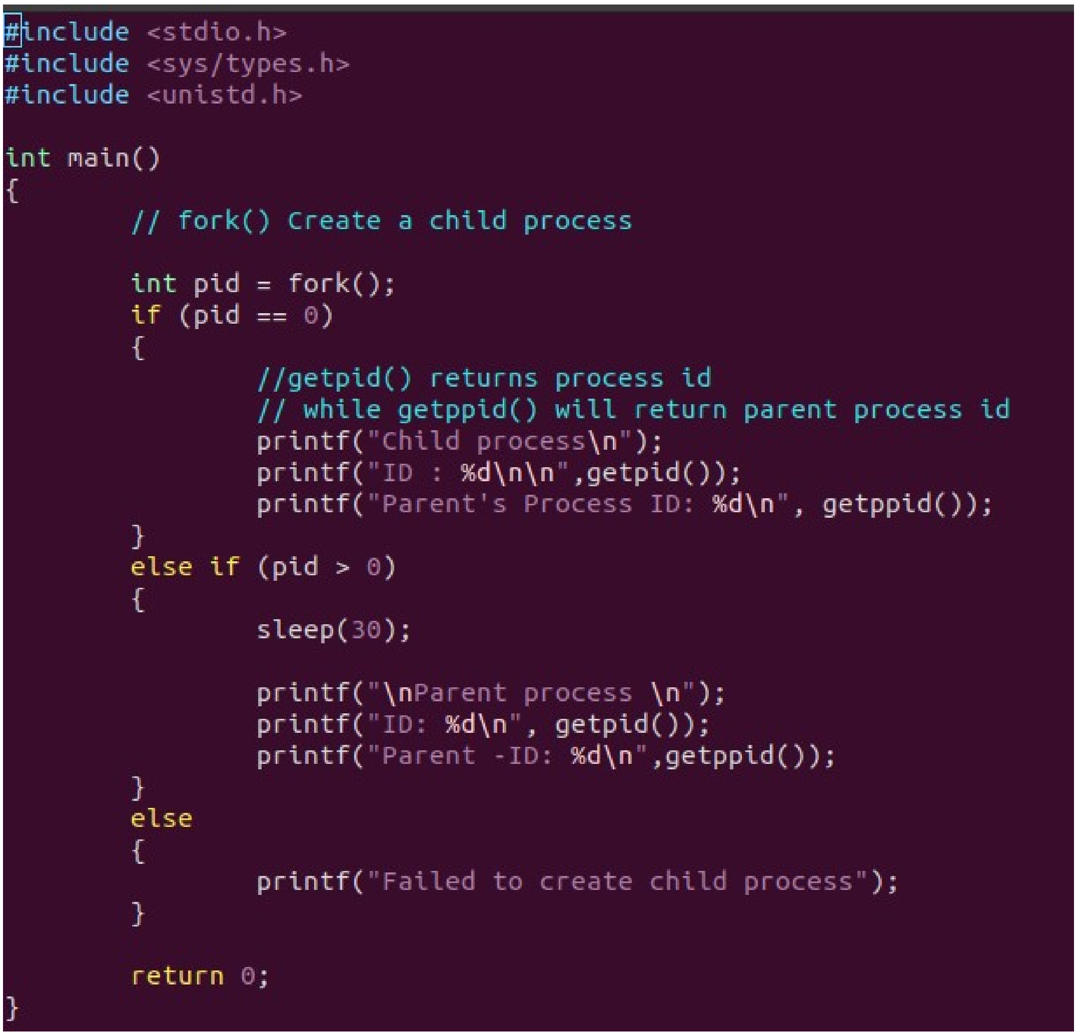
Output:



```
pratyush@pratyush-Inspiron-5570:~$ cd Downloads
pratyush@pratyush-Inspiron-5570:~/Downloads$ cd "OS study material"
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material$ cd "OS Lab"
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ gcc orphan.c
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ./a.out
Parent process
ID : 26005

pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$
Child process
ID: 26006
Parent -ID: 1247

pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ 
```

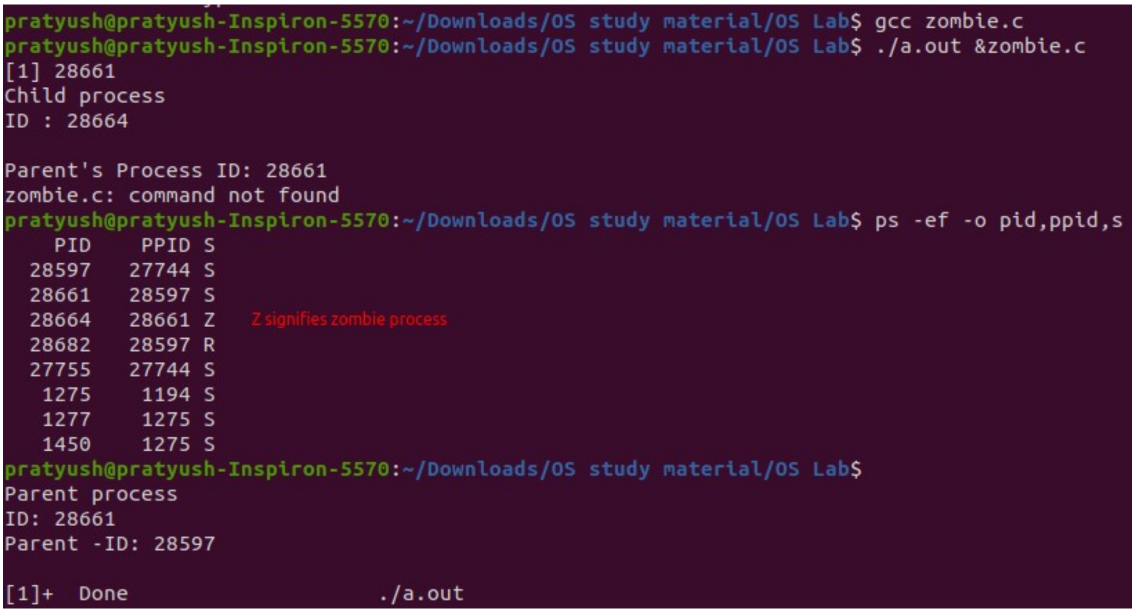
c. Zombie Process

Code:



```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
        // fork() Create a child process

        int pid = fork();
        if (pid == 0)
        {
                //getpid() returns process id
                // while getppid() will return parent process id
                printf("Child process\n");
                printf("ID : %d\n\n",getpid());
                printf("Parent's Process ID: %d\n", getppid());
        }
        else if (pid > 0)
        {
                sleep(30);

                printf("\nParent process \n");
                printf("ID: %d\n", getpid());
                printf("Parent -ID: %d\n",getppid());
        }
        else
        {
                printf("Failed to create child process");
        }

        return 0;
}
```

Output:



```
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ gcc zombie.c
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ./a.out &zombie.c
[1] 28661
Child process
ID : 28664

Parent's Process ID: 28661
zombie.c: command not found
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ps -ef -o pid,ppid,s
    PID    PPID S
  28597   27744 S
  28661   28597 S
  28664   28661 Z    Z signifies zombie process
  28682   28597 R
  27755   27744 S
   1275    1194 S
   1277    1275 S
   1450    1275 S
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$
Parent process
ID: 28661
Parent -ID: 28597

[1]+  Done                    ./a.out
```

3. Shell Scripting

a. Shell script to find greatest of three numbers

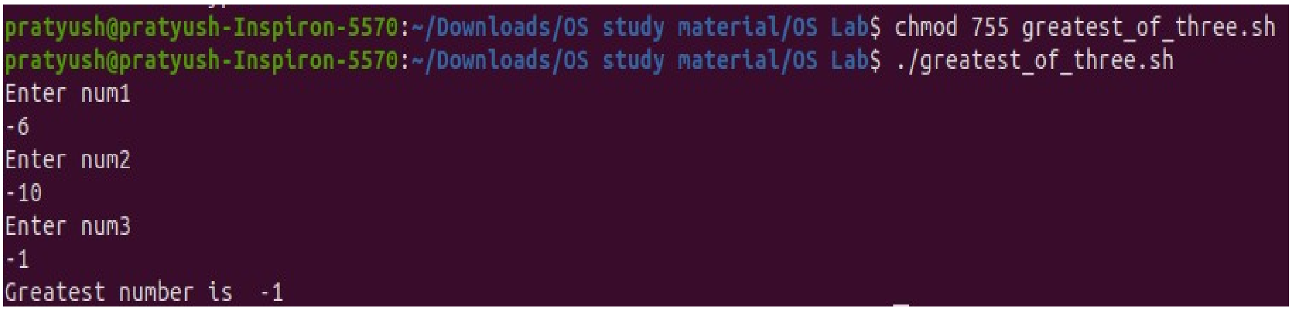
Algorithm:

```
1. Read three integers from the user; num1, num2 and num3
2, Then print out the greatest of the integers by
        if (num1 > num2 and num1 > num3)
                then greatest number is num1
        else if (num2 > num1 and num2 > num3)
                then greatest number is num2
        else
                greatest number is num3
```

Code:

```
echo "Enter num1"
read num1
echo "Enter num2"
read num2
echo "Enter num3"
read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
        echo "Greatest number is " $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
        echo "Greatest number is " $num2
else
        echo "Greatest number is " $num3
fi
```

<u>Output</u>:



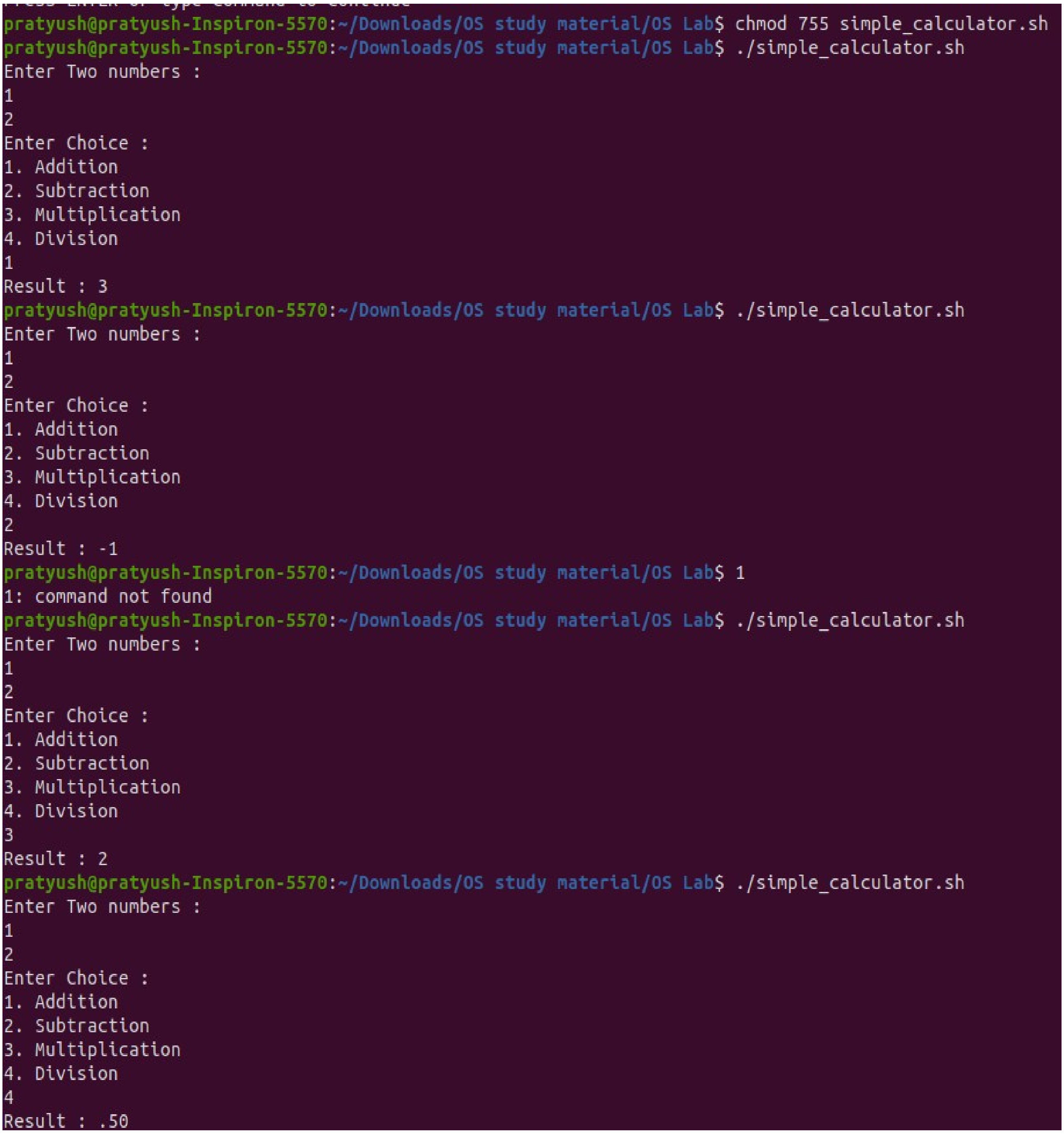
```
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ chmod 755 greatest_of_three.sh
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ./greatest_of_three.sh
Enter num1
-6
Enter num2
-10
Enter num3
-1
Greatest number is  -1
```

b. To perform basic arithmetic operations between two numbers

<u>Code</u>:

```
echo "Enter Two numbers : "
read a
read b

# Input type of operation
echo "Enter Choice :"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"

read ch

# Switch Case to perform
# calulator operations
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`
  ;;
esac
echo "Result : $res"
```

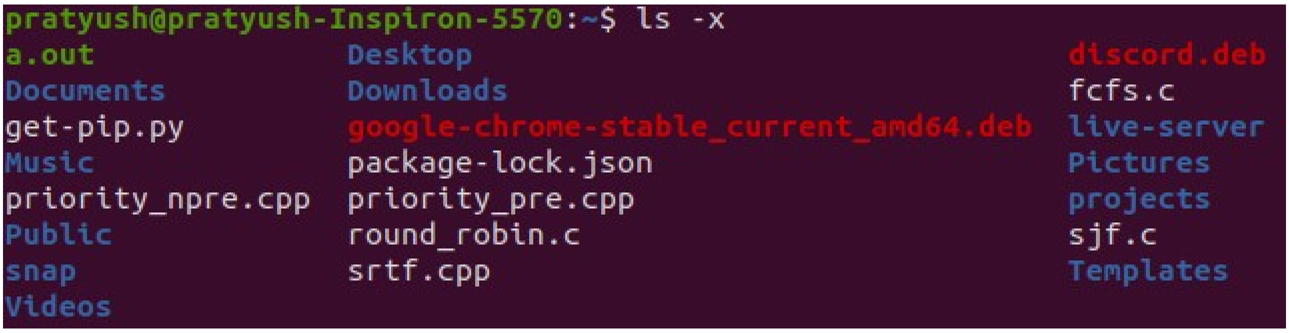
Output:



```
Press ENTER or type command to continue
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ chmod 755 simple_calculator.sh
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ./simple_calculator.sh
Enter Two numbers :
1
2
Enter Choice :
1. Addition
2. Subtraction
3. Multiplication
4. Division
1
Result : 3
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ./simple_calculator.sh
Enter Two numbers :
1
2
Enter Choice :
1. Addition
2. Subtraction
3. Multiplication
4. Division
2
Result : -1
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ 1
1: command not found
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ./simple_calculator.sh
Enter Two numbers :
1
2
Enter Choice :
1. Addition
2. Subtraction
3. Multiplication
4. Division
3
Result : 2
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ./simple_calculator.sh
Enter Two numbers :
1
2
Enter Choice :
1. Addition
2. Subtraction
3. Multiplication
4. Division
4
Result : .50
```

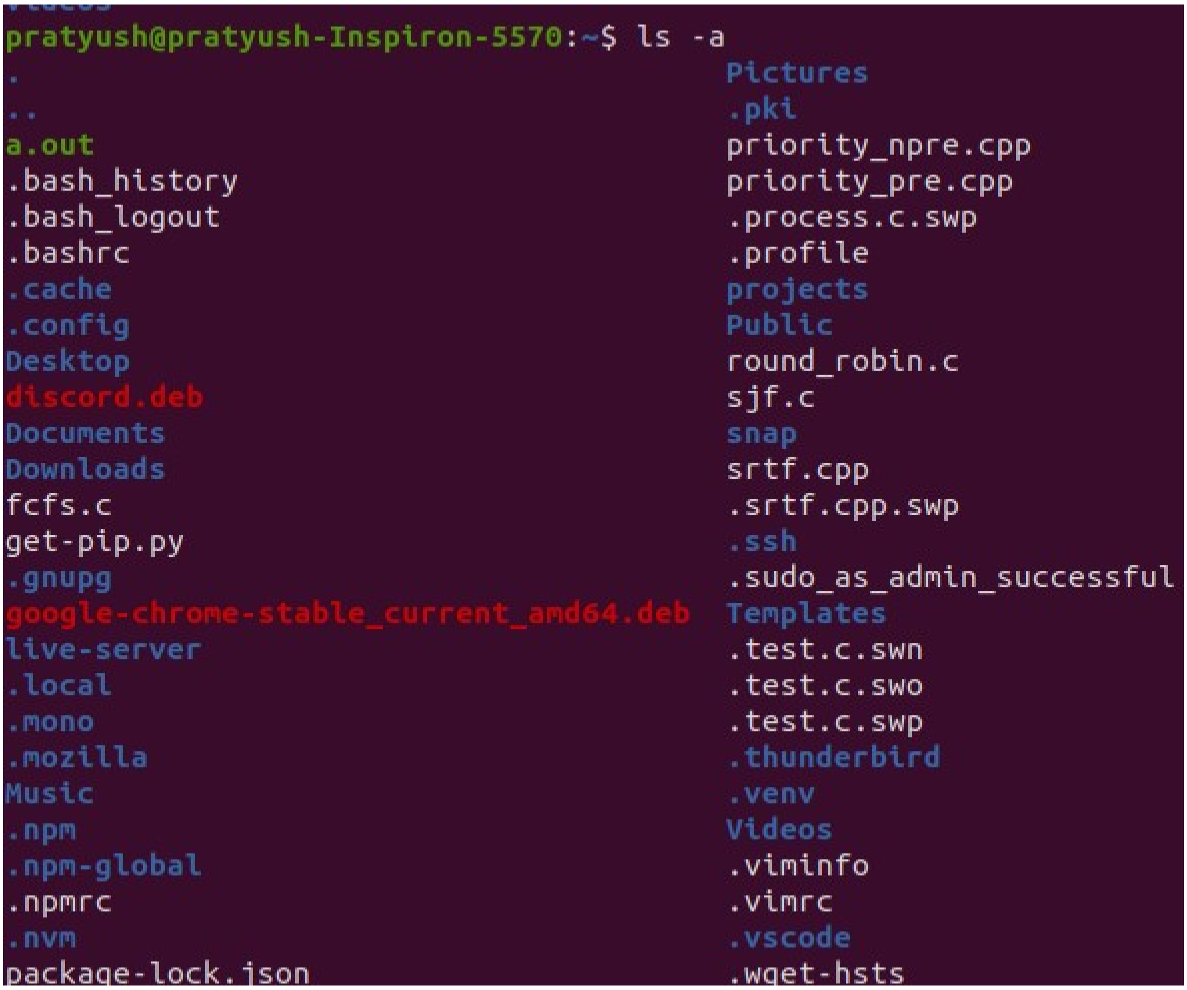
4. 15 Basic Linux Commands

a. ls -x (display files in many columns)



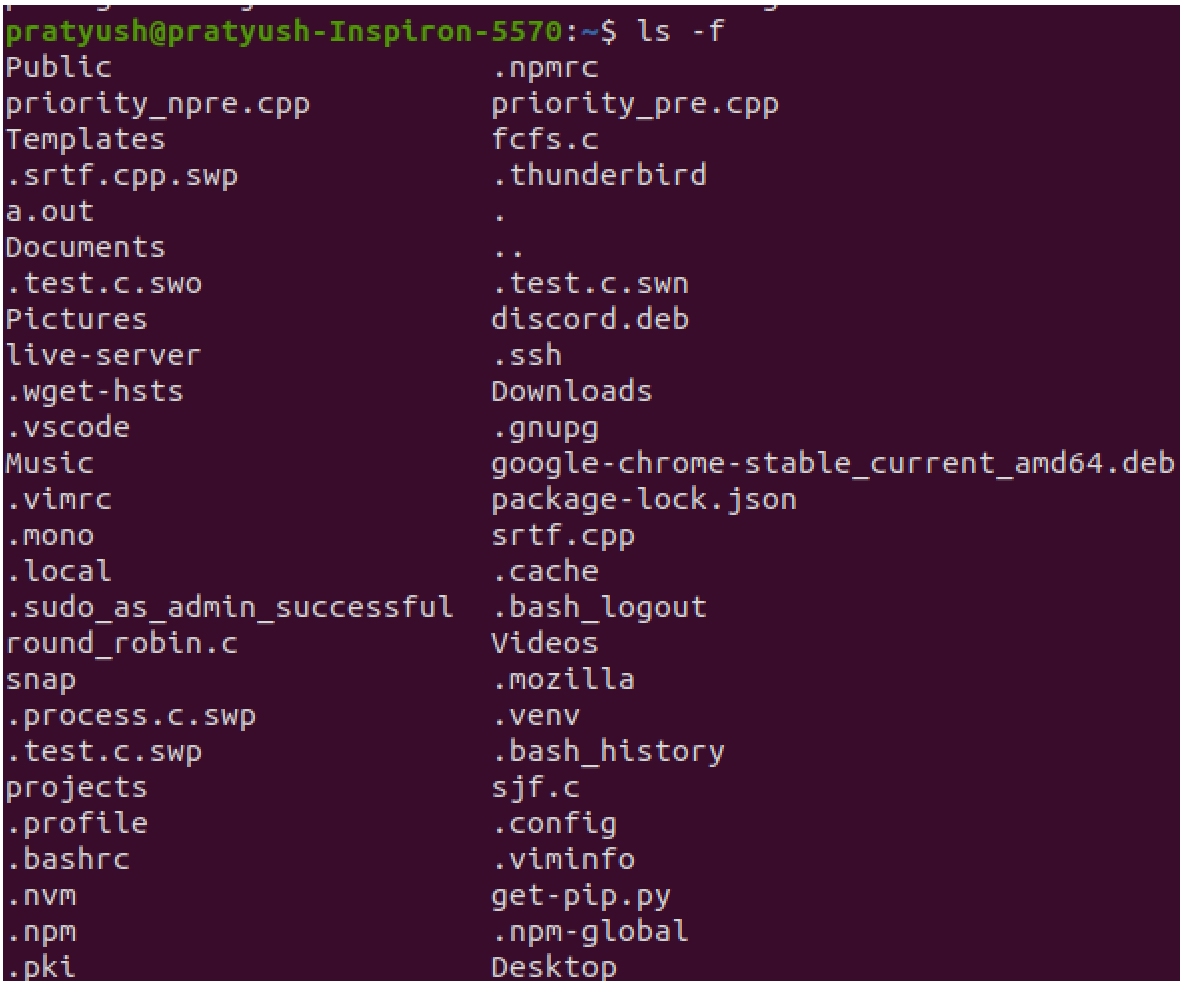
```
pratyush@pratyush-Inspiron-5570:~$ ls -x
a.out              Desktop                                 discord.deb
Documents          Downloads                               fcfs.c
get-pip.py         google-chrome-stable_current_amd64.deb  live-server
Music              package-lock.json                       Pictures
priority_npre.cpp  priority_pre.cpp                        projects
Public             round_robin.c                           sjf.c
snap               srtf.cpp                                Templates
Videos
```

b. ls -a (hidden files)



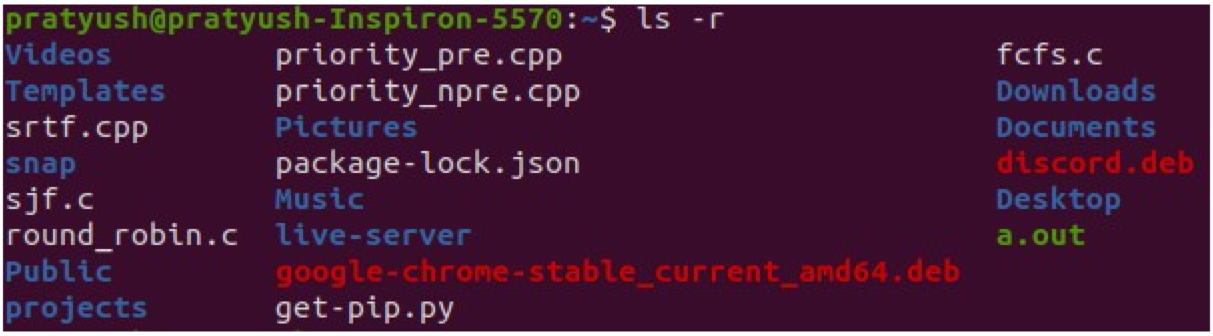
```
pratyush@pratyush-Inspiron-5570:~$ ls -a
.                                       Pictures
..                                      .pki
a.out                                   priority_npre.cpp
.bash_history                           priority_pre.cpp
.bash_logout                            .process.c.swp
.bashrc                                 .profile
.cache                                  projects
.config                                 Public
Desktop                                 round_robin.c
discord.deb                             sjf.c
Documents                               snap
Downloads                               srtf.cpp
fcfs.c                                  .srtf.cpp.swp
get-pip.py                              .ssh
.gnupg                                  .sudo_as_admin_successful
google-chrome-stable_current_amd64.deb  Templates
live-server                             .test.c.swn
.local                                  .test.c.swo
.mono                                   .test.c.swp
.mozilla                                .thunderbird
Music                                   .venv
.npm                                    Videos
.npm-global                             .viminfo
.npmrc                                  .vimrc
.nvm                                    .vscode
package-lock.json                       .wget-hsts
```

c. ls -f (shows / for directory, * for exe file)



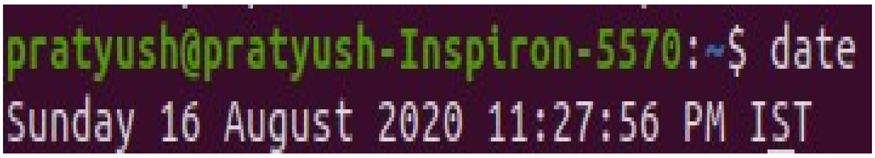
```
pratyush@pratyush-Inspiron-5570:~$ ls -f
Public                          .npmrc
priority_npre.cpp               priority_pre.cpp
Templates                       fcfs.c
.srtf.cpp.swp                   .thunderbird
a.out                           .
Documents                       ..
.test.c.swo                     .test.c.swn
Pictures                        discord.deb
live-server                     .ssh
.wget-hsts                      Downloads
.vscode                         .gnupg
Music                           google-chrome-stable_current_amd64.deb
.vimrc                          package-lock.json
.mono                           srtf.cpp
.local                          .cache
.sudo_as_admin_successful       .bash_logout
round_robin.c                   Videos
snap                            .mozilla
.process.c.swp                  .venv
.test.c.swp                     .bash_history
projects                        sjf.c
.profile                        .config
.bashrc                         .viminfo
.nvm                            get-pip.py
.npm                            .npm-global
.pki                            Desktop
```

d. ls -r (list in reverse order)



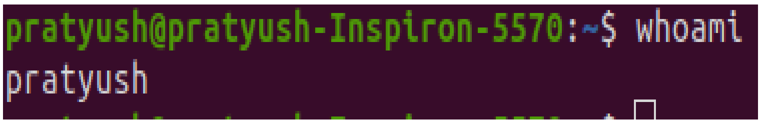
```
pratyush@pratyush-Inspiron-5570:~$ ls -r
Videos         priority_pre.cpp                        fcfs.c
Templates      priority_npre.cpp                       Downloads
srtf.cpp       Pictures                                Documents
snap           package-lock.json                       discord.deb
sjf.c          Music                                   Desktop
round_robin.c  live-server                             a.out
Public         google-chrome-stable_current_amd64.deb
projects       get-pip.py
```

e. date (show date and time)



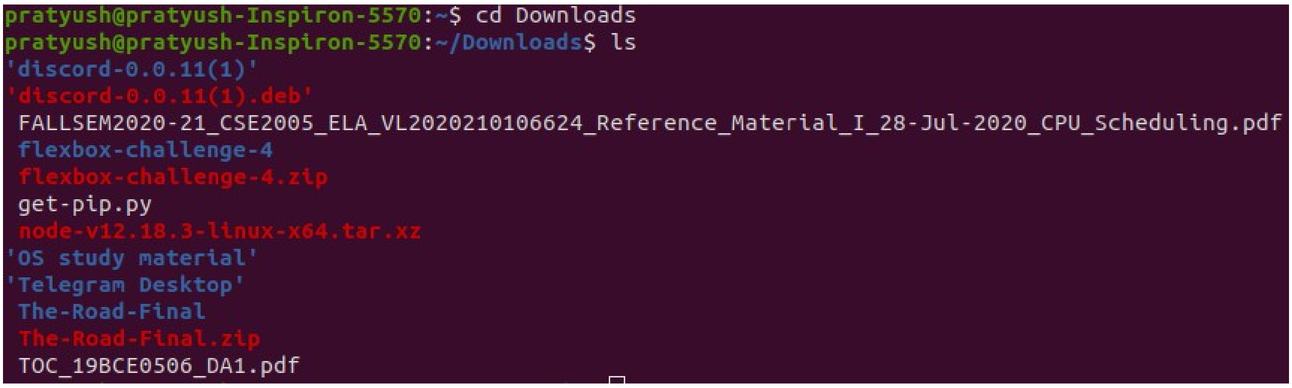
```
pratyush@pratyush-Inspiron-5570:~$ date
Sunday 16 August 2020 11:27:56 PM IST
```

f. whoami (who is logged onto this terminal)



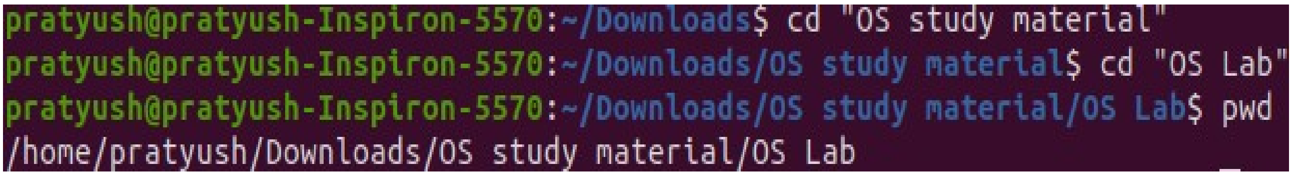
```
pratyush@pratyush-Inspiron-5570:~$ whoami
pratyush
```

g. cd (change directory)



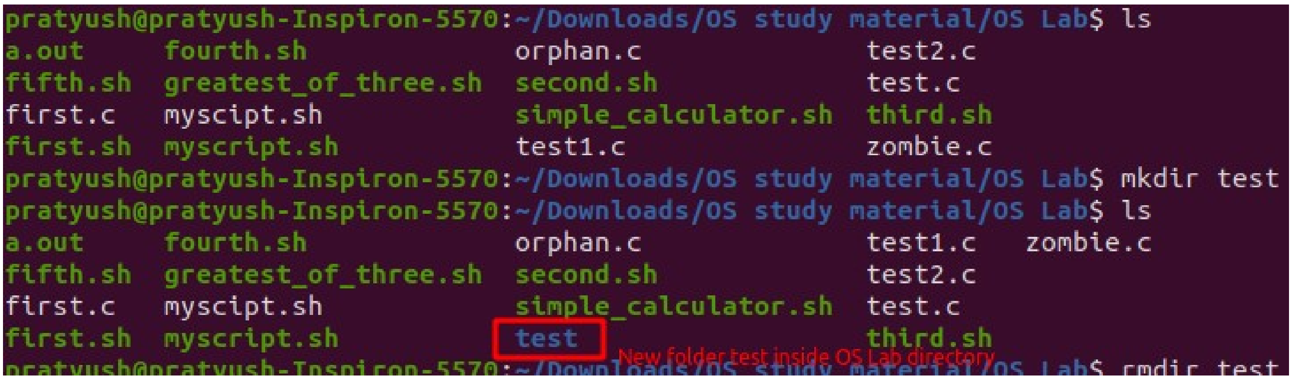
```
pratyush@pratyush-Inspiron-5570:~$ cd Downloads
pratyush@pratyush-Inspiron-5570:~/Downloads$ ls
'discord-0.0.11(1)'
'discord-0.0.11(1).deb'
 FALLSEM2020-21_CSE2005_ELA_VL2020210106624_Reference_Material_I_28-Jul-2020_CPU_Scheduling.pdf
 flexbox-challenge-4
 flexbox-challenge-4.zip
 get-pip.py
 node-v12.18.3-linux-x64.tar.xz
'OS study material'
'Telegram Desktop'
 The-Road-Final
 The-Road-Final.zip
 TOC_19BCE0506_DA1.pdf
```

h. pwd (show current directory)



```
pratyush@pratyush-Inspiron-5570:~/Downloads$ cd "OS study material"
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material$ cd "OS Lab"
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ pwd
/home/pratyush/Downloads/OS study material/OS Lab
```

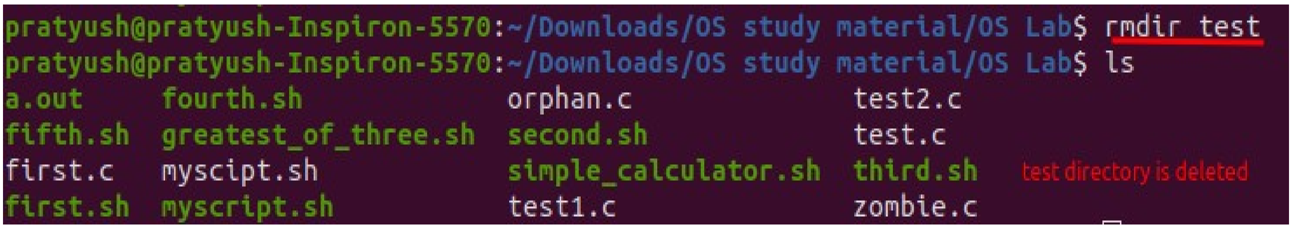
i. mkdir (create new directory)



```
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ls
a.out      fourth.sh             orphan.c                test2.c
fifth.sh   greatest_of_three.sh  second.sh               test.c
first.c    myscipt.sh            simple_calculator.sh    third.sh
first.sh   myscript.sh           test1.c                 zombie.c
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ mkdir test
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ls
a.out      fourth.sh             orphan.c                test1.c   zombie.c
fifth.sh   greatest_of_three.sh  second.sh               test2.c
first.c    myscipt.sh            simple_calculator.sh    test.c
first.sh   myscript.sh           test                    third.sh
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ rmdir test
```

New folder test inside OS Lab directory

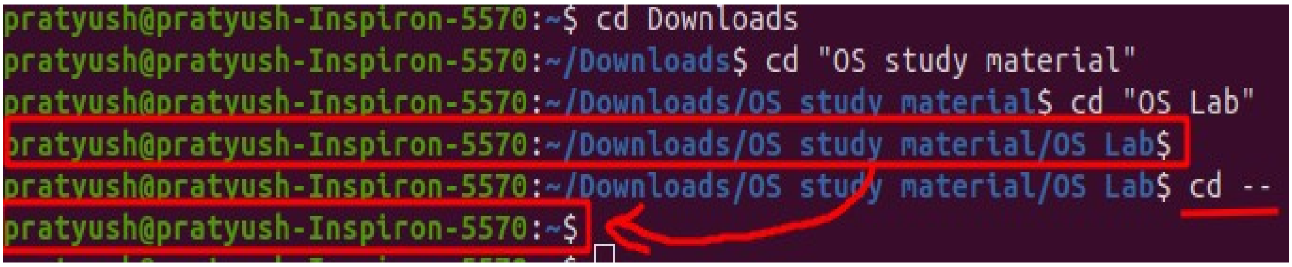
j. rmdir (remove directory)



```
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ rmdir test
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ ls
a.out      fourth.sh             orphan.c                test2.c
fifth.sh   greatest_of_three.sh  second.sh               test.c
first.c    myscipt.sh            simple_calculator.sh    third.sh
first.sh   myscript.sh           test1.c                 zombie.c
```

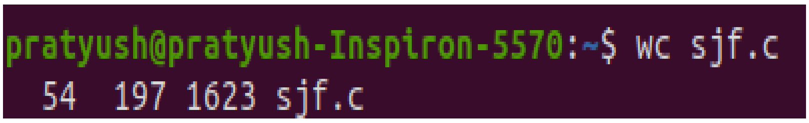
test directory is deleted

k. cd – (go back to root directory)



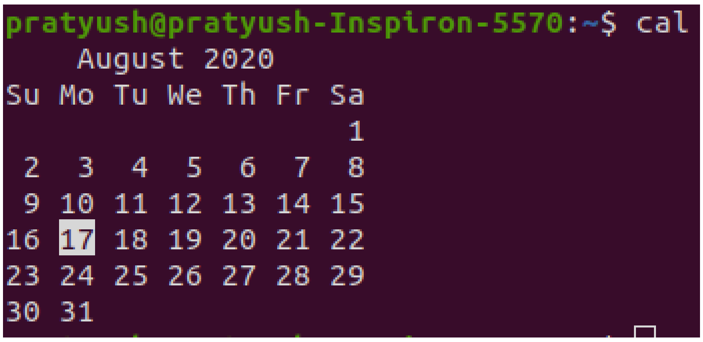
```
pratyush@pratyush-Inspiron-5570:~$ cd Downloads
pratyush@pratyush-Inspiron-5570:~/Downloads$ cd "OS study material"
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material$ cd "OS Lab"
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$
pratyush@pratyush-Inspiron-5570:~/Downloads/OS study material/OS Lab$ cd --
pratyush@pratyush-Inspiron-5570:~$
```

l. wc (count characters, words and lines in a file)



```
pratyush@pratyush-Inspiron-5570:~$ wc sjf.c
  54  197 1623 sjf.c
```

m. cal (print the calendar)



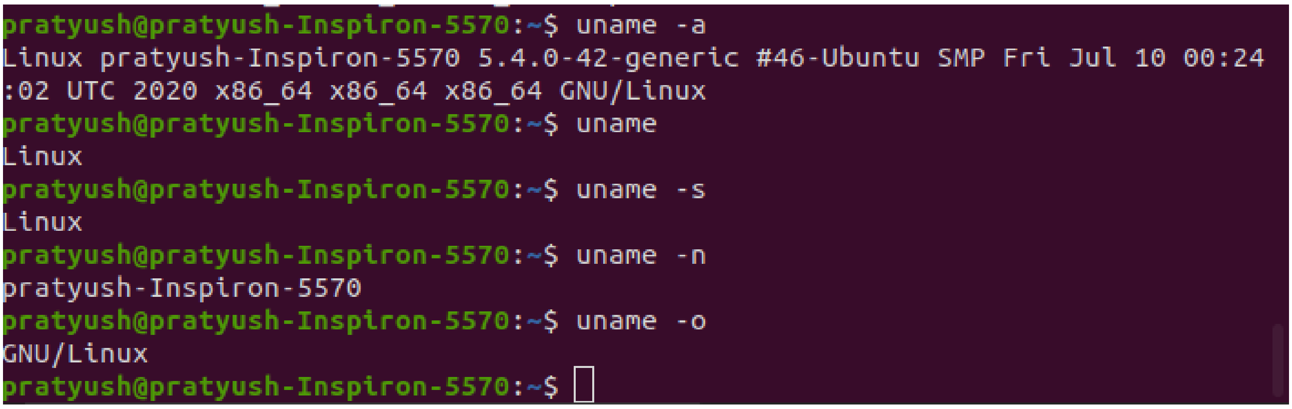
```
pratyush@pratyush-Inspiron-5570:~$ cal
    August 2020
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

n. UNAME

Print system information

uname –a all

options –s (print kernel name)**, -n** (network name)**, -e** (hardware platform) **,-o** (os)



```
pratyush@pratyush-Inspiron-5570:~$ uname -a
Linux pratyush-Inspiron-5570 5.4.0-42-generic #46-Ubuntu SMP Fri Jul 10 00:24
:02 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
pratyush@pratyush-Inspiron-5570:~$ uname
Linux
pratyush@pratyush-Inspiron-5570:~$ uname -s
Linux
pratyush@pratyush-Inspiron-5570:~$ uname -n
pratyush-Inspiron-5570
pratyush@pratyush-Inspiron-5570:~$ uname -o
GNU/Linux
pratyush@pratyush-Inspiron-5570:~$
```

o.

echo $$

process id of current shell

ps

process status (pid, name, tty, time)

ps –f

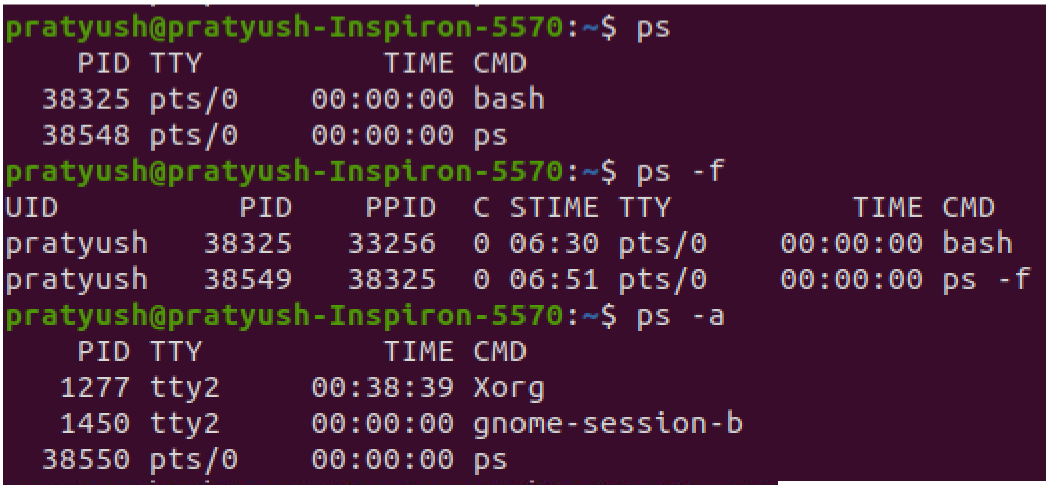
full information(uid, ppid etc)

ps –f –u cra

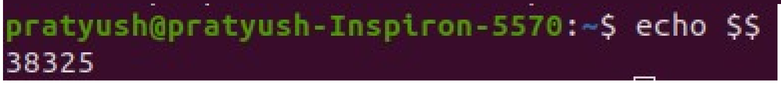
full information of user cra [here cra = pratyush]

ps –a

processes of all users



```
pratyush@pratyush-Inspiron-5570:~$ ps
    PID TTY          TIME CMD
  38325 pts/0    00:00:00 bash
  38548 pts/0    00:00:00 ps
pratyush@pratyush-Inspiron-5570:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
pratyush   38325   33256  0 06:30 pts/0    00:00:00 bash
pratyush   38549   38325  0 06:51 pts/0    00:00:00 ps -f
pratyush@pratyush-Inspiron-5570:~$ ps -a
    PID TTY          TIME CMD
   1277 tty2     00:38:39 Xorg
   1450 tty2     00:00:00 gnome-session-b
  38550 pts/0    00:00:00 ps
```



```
pratyush@pratyush-Inspiron-5570:~$ echo $$
38325
```