**Experiment 1(a):**

**Aim:** Simulate CPU Scheduling algorithm***Round Robin***

**Objective:** We create*n*no.of processes, each process one has*burst time*and*arrival time.*

Based on input of *time quantum,* processes are scheduled.

**Algorithm**

Step 1:Start

2: Declare exeCount,i,rounds:=1,finish=0,totalwaiting=0 3:Repeat steps 4 to 10 While(not isAllFinished(p,n))

4: Print “After Round #”,rounds++

5: Repeat steps 6 to 10 For I:=0 to n step 1

1. Repeat steps 7 to 9 For exeCount:=1 to exeCount<=q AND p[i].remainingTime!=0
2. Set p[i].remainingTime--
3. Set finish:=finish+1
4. if(p[i].remainingTime=0) Then → Set p[i].finishTime:=finish

[End For – exeCount]

10: Print “Remaining time ',p[i].remainingTime

[End For – I]

[End While]

11: Repeat steps 12,13 For I:=0 to n step 1

1. Print p[i].finishTime, p[i].finishTime-p[i].burstTime //this is waiting time

subtraction

1. Set totalwaiting+=p[i].finishTime-p[i].burstTime //adding total waiting

[End For]

14: Print totalwaiting, totalwaiting/n //n for no.of processes

15: Stop

**Program:**

#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<string.h> void main()

{

char p[10][5];

int et[10],wt[10],timer=3,count,pt[10],rt,i,j,totwt=0,t,n=5,found=0,m; float avgwt;

clrscr();

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

{

printf("enter the process name : "); scanf("%s",&p[i]);

printf("enter the processing time : "); scanf("%d",&pt[i]);

}

m=n;

wt[0]=0;

i=0;

do

{

if(pt[i]>timer)

{

rt=pt[i]-timer; strcpy(p[n],p[i]); pt[n]=rt; et[i]=timer; n++;

}

else

{

et[i]=pt[i];

}

i++; wt[i]=wt[i-1]+et[i-1]; }while(i<n);

count=0;

for(i=0;i<m;i++)

{

for(j=i+1;j<=n;j++)

{

if(strcmp(p[i],p[j])==0)

{

count++;

found=j;

}

}

if(found!=0)

{

wt[i]=wt[found]-(count\*timer); count=0;

found=0;

}

}

for(i=0;i<m;i++)

{

totwt+=wt[i];

}

avgwt=(float)totwt/m;

for(i=0;i<m;i++)

{

printf("**\n**%s**\t**%d**\t**%d",p[i],pt[i],wt[i]);

}

printf("**\n**total waiting time %d**\n**",totwt); printf("total avgtime %f",avgwt);

}

**Expected Input/Output**

INPUT :

enter the process name : aaa

enter the processing time : 4

enter the process name : bbb

enter the processing time : 3

enter the process name : ccc

enter the processing time : 2

enter the process name : ddd

enter the processing time : 5

enter the process name : eee

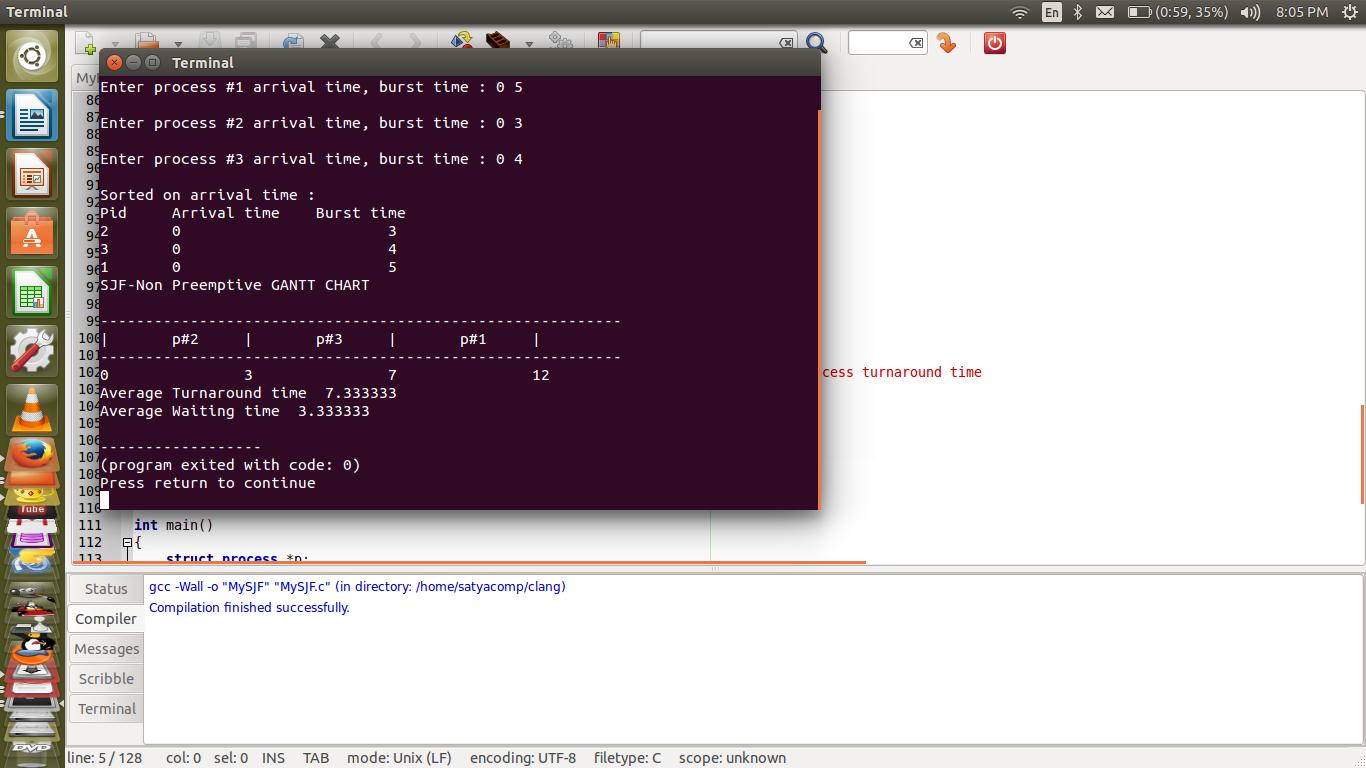
enter the processing time : 1

|  |  |  |
| --- | --- | --- |
| **OUTPUT** : |  |  |
| p\_name | p\_time | w\_time |
| aaa | 4 | 9 |
| bbb | 3 | 3 |
| ccc | 2 | 6 |
| ddd | 5 | 10 |
| eee | 1 | 11 |

total waiting time : 39

average waiting time : 7.8000

**Original input/output:**



**Viva-voice Questions**

**1. Define CPU scheduling.**

CPU scheduling is the process of switching the CPU among various processes. CPU scheduling is the basis of multi programmed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.

1. **What is a Dispatcher?**

The dispatcher is the module that gives control of the CPU to theprocess selected by the short-term scheduler. This function involves: • Switching context • Switching to user mode • Jumping to the proper location in the user program to restart that program.

1. **What is turnaround time?**

Turnaround time is the interval from the time of submission tothe time of completion of a process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

1. **Define dispatch latency?**

The time taken for the dispatcher to stop one process and start another running process is known as dispatch latency

**5. What is job scheduling?**

If several jobs are ready to be brought in to memory, and if there is not enough room for all of them, then the system must choose among them. Making this decision is job scheduling

**Experiment 1(b):**

**Aim:** Simulate CPU Scheduling algorithm***Shortest Job First***

**Objective:** Create*n*no. of processes, each process has*arrival time*and*burst time.*Sort on*arrival time*then by*burst time.* Process least burst time first and highest burst time next.

**Algorithm**

|  |  |  |
| --- | --- | --- |
| Step 1: Start |  |  |
| 2: Declare | wt:=0,tt:=0,i | //wt-for waiting time, tt-turnaround time, i -index repeating |
| 3: Repeat | step 4 For I:=0 to n step 1 | |
| 4: Print p[i].pid | | //printing each process id in Gantt chart |
| [End for] |  |  |
| 5: Repeat Steps 6 to 10 | | For i=0 to n step 1 |
| 6: print wt | //priting waiting time | |

1. Set p[i].watitingtime:=wt
2. Repeat Steps 9,10 while p[i].RemainingTime>0 //each process remainingtime is initially burst time
3. Set p[i].RemainingTime:=p[i].RemaminingTime-1

10: Set Wt:=Wt+1 //increase waiting time by 1

[End While]

[End For]

11: print Wt //last process waiting time

1. Set p[i].waitingTime:=Wt
2. Repeat steps 14 to 16 For I:=0 to n step 1
3. set p[i].turnAroundTime:=(p[i].waitingTime+p[i].burstTime)-p[i].ArrivalTime //Each process turnaround time

15: Set tt:=tt+p[i].TurnaroundTime //calculating total turnaround time

1. Set wt:=wt+p[i].waitingTime //calculating total waiting time [End For]
2. Print “Avg turnaround time : “,tt/n
3. Print “Avg waiting time : “,wt/n
4. Stop

**Program:**

#include<stdio.h>

#include<conio.h>

#include<process.h> void main()

{

char p[10][5],temp[5];

int tot=0,wt[10],pt[10],i,j,n,temp1; float avg=0;

clrscr();

printf("enter no of processes:"); scanf("%d",&n); for(i=0;i<n;i++)

{

printf("enter process%d name:**\n**",i+1); scanf("%s",&p[i]);

printf("enter process time"); scanf("%d",&pt[i]);

}

for(i=0;i<n-1;i++)

{

for(j=i+1;j<n;j++)

{

if(pt[i]>pt[j])

{

temp1=pt[i];

pt[i]=pt[j];

pt[j]=temp1;

strcpy(temp,p[i]);

strcpy(p[i],p[j]);

strcpy(p[j],temp);

}

}

}

wt[0]=0;

for(i=1;i<n;i++)

{

wt[i]=wt[i-1]+et[i-1]; tot=tot+wt[i];

}

avg=(float)tot/n;

printf("p\_name**\t** P\_time**\t** w\_time**\n**");

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

printf("%s**\t**%d**\t**%d**\n**",p[i],et[i],wt[i]);

printf("total waiting time=%d**\n** avg waiting time=%f",tot,avg);

getch();

}

**Expected input/output:**

enter no of processes: 5

enter process1 name: aaa

enter process time: 4

enter process2 name: bbb

enter process time: 3

enter process3 name: ccc

enter process time: 2

enter process4 name: ddd

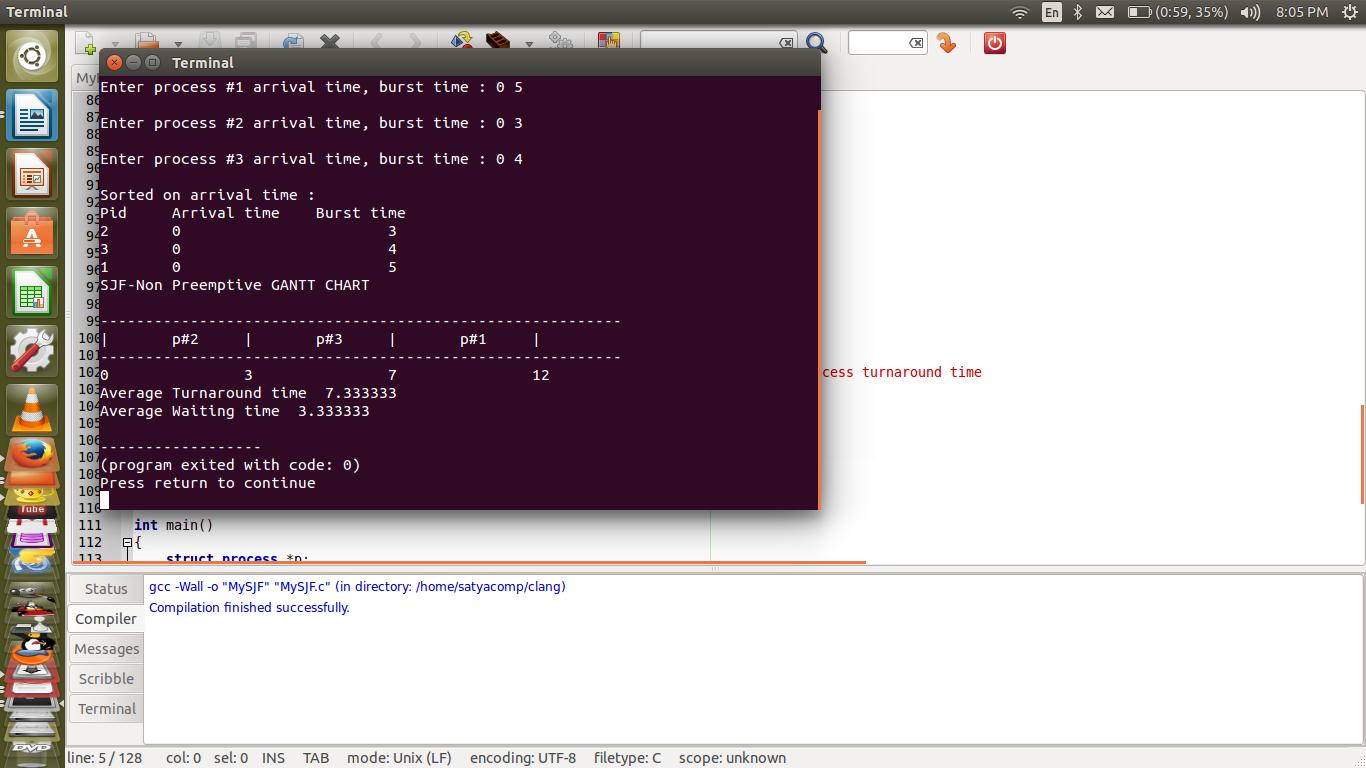
enter process time: 5

enter process5 name: eee enter process time: 1

|  |  |  |
| --- | --- | --- |
| p\_name | P\_time | w\_time |
| eee | 1 | 0 |
| ccc | 2 | 1 |
| bbb | 3 | 3 |
| aaa | 4 | 6 |
| ddd | 5 | 10 |

total waiting time=20 avg waiting time=4.00

**Original input/output:**



**Viva-voice Questions**

**1. What is symmetric multiprocessing?**

Each processor runs an identical copy of the operating system, and these copies communicate with one another as needed.

**2. List out the types in mainframe systems**

1. Batch system
2. Multiprogrammed systems
3. Time-sharing system

**3. What are file-server systems?**

File-server system provides a file system interface where clients can create, update, read, and delete files

**4. What is job scheduling?**

If several jobs are ready to be brought in to memory, and if there is not enough room for all of them, then the system must choose among them. Making this decision is job scheduling

**Experiment 1(c):**

**Aim:** Simulate CPU Scheduling algorithm***First Come First Serve***

**Objective:** Create*n*no.of processes, each process has*arrivaltime*and*bursttime.*Sort on*arrivaltime.* First entered process is processed first.

**Algorithm**

Step 1: Start

1. Declare wt:=0, I //Waiting Time and i for index
2. Repeat step 4 for I:=0 to n step 1
3. Print p[i].pid //printing each process in Gantt chart [End for]
4. Repeat steps 6 to 10 For I:=0 to n step 1

|  |  |  |
| --- | --- | --- |
| 6: | Print wt | // process each waiting time |
| 7: | Set p[i].waitingTime:=wt-p[i].arrivalTime //update each process waiting | |

time

1. Repeat Step 9,10 while p[i].burstTime>0 loop
2. p[i].bursttime-- //process each bursttime until reaches to 0
3. Set wt:=wt+1

[End While]

[End For]

11: Print wt //last proccess waiting time

12: Set p[i].waitingTime:=wt

13:Repeat step 14 for I:=0 to n step 1

1. Set wt:=wt+p[i].waitingTime //sum of all waiting times [End for]
2. print “Average waiting time : “,wt/n
3. Stop

**Program:**

#include<stdio.h>

#include<conio.h>

#include<process.h> void main()

{

char p[10][5];

int tot=0,wt[10],i,n;

float avg=0; clrscr();

printf("enter no of processes:"); scanf("%d",&n); for(i=0;i<n;i++)

{

printf("enter process%d name:**\n**",i+1); scanf("%s",&p[i]);

printf("enter process time"); scanf("%d",&pt[i]);

}

wt[0]=0;

for(i=1;i<n;i++)

{

wt[i]=wt[i-1]+et[i-1]; tot=tot+wt[i];

}

avg=(float)tot/n;

printf("p\_name**\t** P\_time**\t** w\_time**\n**"); for(i=0;i<n;i++) printf("%s**\t**%d**\t**%d**\n**",p[i],et[i],wt[i]);

printf("total waiting time=%d**\n** avg waiting time=%f",tot,avg); getch();

}

**Expected input/output:**

enter no of processes: 5

enter process1 name: aaa

enter process time: 4

enter process2 name: bbb

enter process time: 3

enter process3 name: ccc

enter process time: 2

enter process4 name: ddd

enter process time: 5

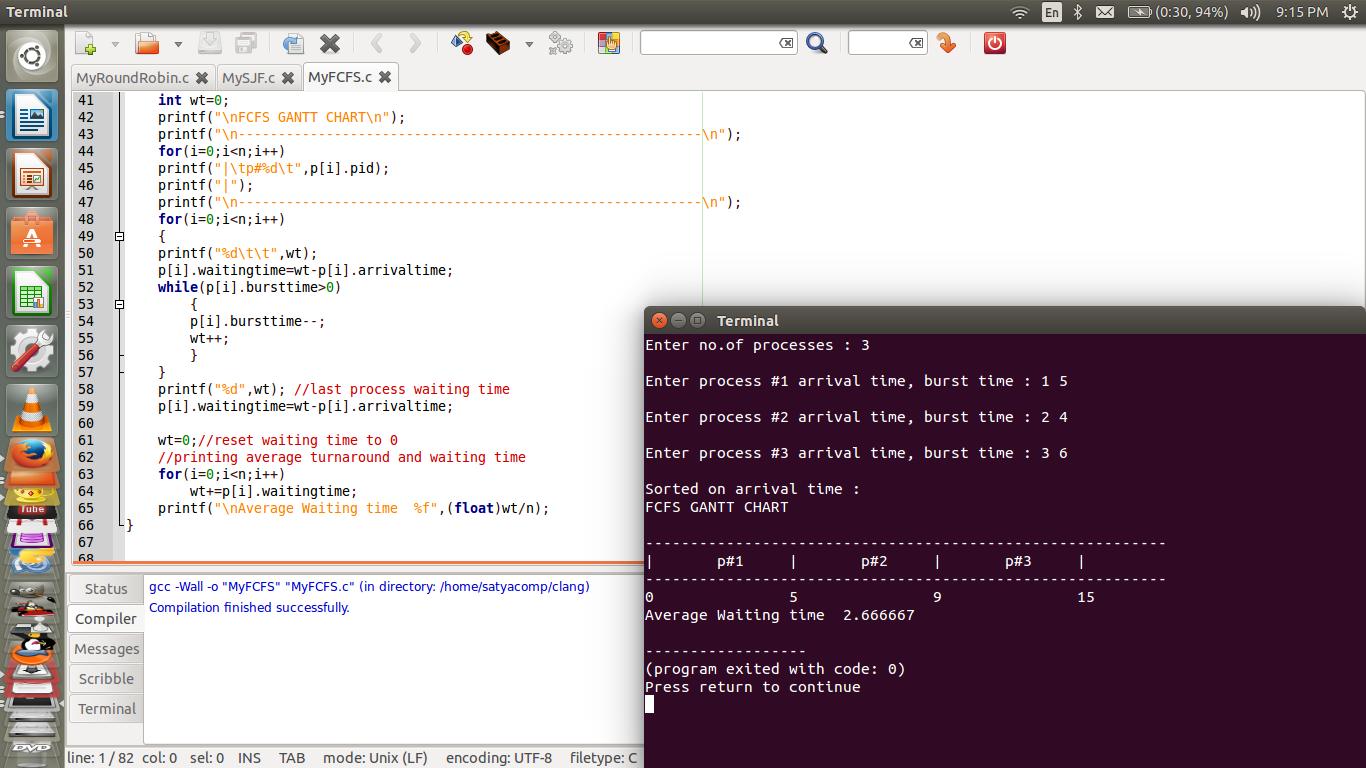
enter process5 name: eee

enter process time: 1

|  |  |  |
| --- | --- | --- |
| p\_name | P\_time | W\_time |
| aaa | 4 | 0 |
| bbb | 3 | 4 |
| ccc | 2 | 7 |
| ddd | 5 | 9 |
| eee | 1 | 14 |

total waiting time=34 avg waiting time=6.80

**Original output:**



**Viva-voice Questions**

**1. Define CPU scheduling.**

CPU scheduling is the process of switching the CPU among various processes. CPU scheduling is the basis of multi programmed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.

1. **What is a Dispatcher?** The dispatcher is the module that gives control of the CPU to theprocess selected by the short-term scheduler. This function involves: • Switching context • Switching to user mode • Jumping to the proper location in the user program to restart that program.
2. **What is turnaround time?** Turnaround time is the interval from the time of submission tothe time of completion of a process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.
3. **Define dispatch latency?**

The time taken for the dispatcher to stop one process and start another running process is known as dispatch latency.

**Experiment 1(d):**

**Aim:** Simulate CPU Scheduling algorithm***Priority Scheduling***

**Objective:** Create*n*no. of processes, each process has*priority no*and*burst time.*Sort on*priorityno.* Lowest priority number indicates that is processes first. If multiple process havesame priority number, then FCFS is used.

Algorithm for Priority Scheduling (Process \*p,integer n)

Step 1: Start

1. Declare wt:=0, I //Waiting Time and i for index
2. Repeat step 4 for I:=0 to n step 1
3. Print p[i].pid //printing each process in Gantt chart [End for]
4. Repeat steps 6 to 10 For I:=0 to n step 1

|  |  |  |
| --- | --- | --- |
| 6: | Print wt | // process each waiting time |
| 7: | Set p[i].waitingTime:=wt-p[i].arrivalTime //update each process waiting | |

time

1. Repeat Step 9,10 while p[i].burstTime>0 loop
2. p[i].bursttime-- //process each bursttime until reaches to 0
3. Set wt:=wt+1

[End While]

[End For]

11: Print wt //last proccess waiting time

12: Set p[i].waitingTime:=wt

13:Repeat step 14 for I:=0 to n step 1

1. Set wt:=wt+p[i].waitingTime //sum of all waiting times [End for]
2. print “Average waiting time : “,wt/n
3. Stop

**Program:**

#include<stdio.h>

#include<conio.h> void main()

{

char p[10][5],temp[5];

int i,j,pt[10],wt[10],totwt=0,pr[10],temp1,n; float avgwt;

clrscr();

printf("enter no of processes:"); scanf("%d",&n);

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

{

printf("enter process%d name:",i+1); scanf("%s",&p[i]);

printf("enter process time:"); scanf("%d",&pt[i]);

printf("enter priority:"); scanf("%d",&pr[i]);

}

for(i=0;i<n-1;i++)

{

for(j=i+1;j<n;j++)

{

if(pr[i]>pr[j])

{

temp1=pr[i];

pr[i]=pr[j];

pr[j]=temp1;

temp1=pt[i];

pt[i]=pt[j];

pt[j]=temp1;

strcpy(temp,p[i]);

strcpy(p[i],p[j]);

strcpy(p[j],temp);

}

}

}

wt[0]=0;

for(i=1;i<n;i++)

{

wt[i]=wt[i-1]+et[i-1]; totwt=totwt+wt[i];

}

avgwt=(float)totwt/n;

printf("p\_name**\t** p\_time**\t** priority**\t** w\_time**\n**"); for(i=0;i<n;i++)

{

printf(" %s**\t** %d**\t** %d**\t** %d**\n**" ,p[i],pt[i],pr[i],wt[i]);

}

printf("total waiting time=%d**\n** avg waiting time=%f",tot,avg); getch();

}

**Expected input/output:**

enter no of processes: 5

enter process1 name: aaa enter process time: 4 enter priority:5

enter process2 name: bbb enter process time: 3 enter priority:4

enter process3 name: ccc enter process time: 2 enter priority:3

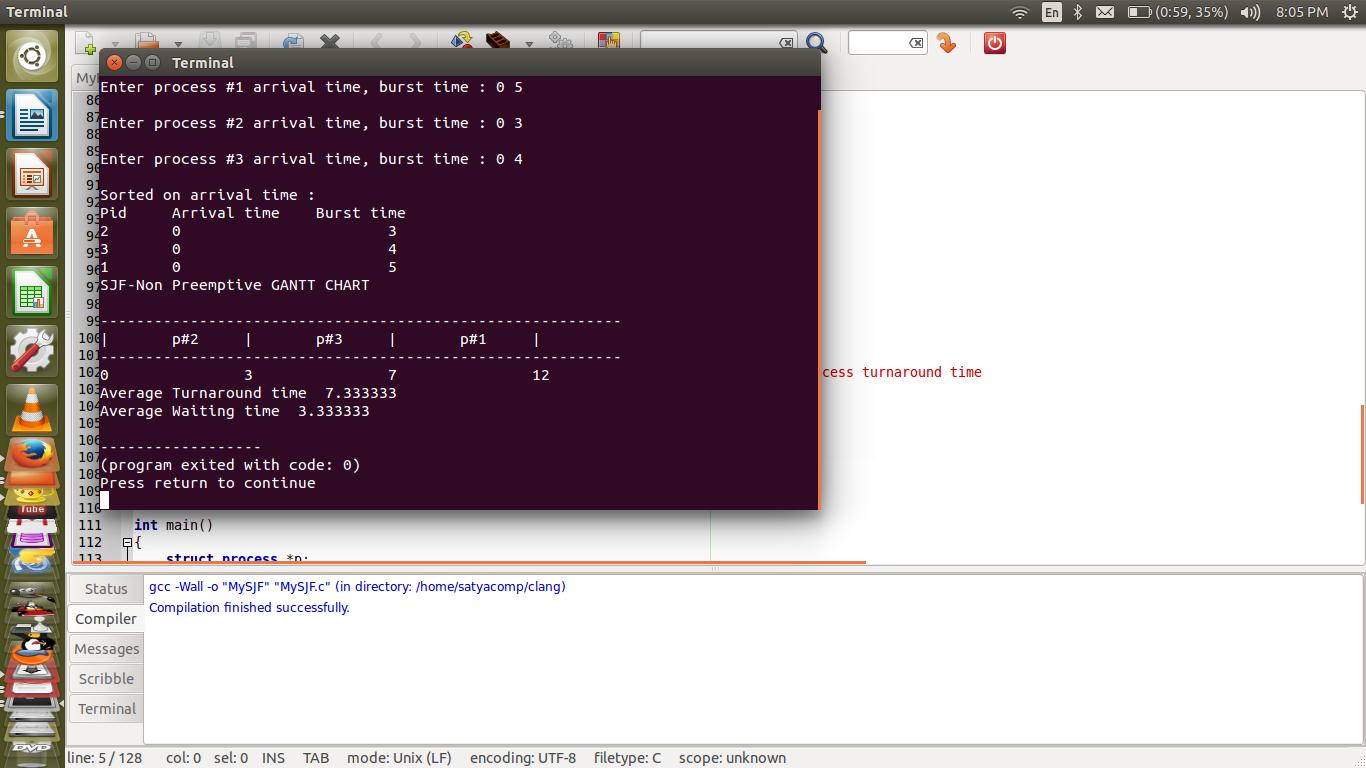
enter process4 name: ddd enter process time: 5 enter priority:2

enter process5 name: eee enter process time: 1 enter priority:1

|  |  |  |
| --- | --- | --- |
| p\_name | P\_time priority | w\_time |
| eee | 1 | 1 0 |
| ddd | 5 | 2 1 |
| ccc | 2 | 3 6 |
| bbb | 3 | 4 8 |
| aaa | 4 | 115 |

total waiting time=26 avg waiting time=5.20

**Output:**



**Viva-voice Questions**

**1. Define CPU scheduling.**

CPU scheduling is the process of switching the CPU among various processes. CPU scheduling is the basis of multi programmed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.

**2. What is a Dispatcher?**

The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves: • Switching context • Switching to user mode Jumping to the proper location in the user program to restart that program.

**3. What is turnaround time?**

Turnaround time is the interval from the time of submission to the time of completion of a process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

**4. Define dispatch latency?**

The time taken for the dispatcher to stop one process and start another running process is known as dispatch latency.