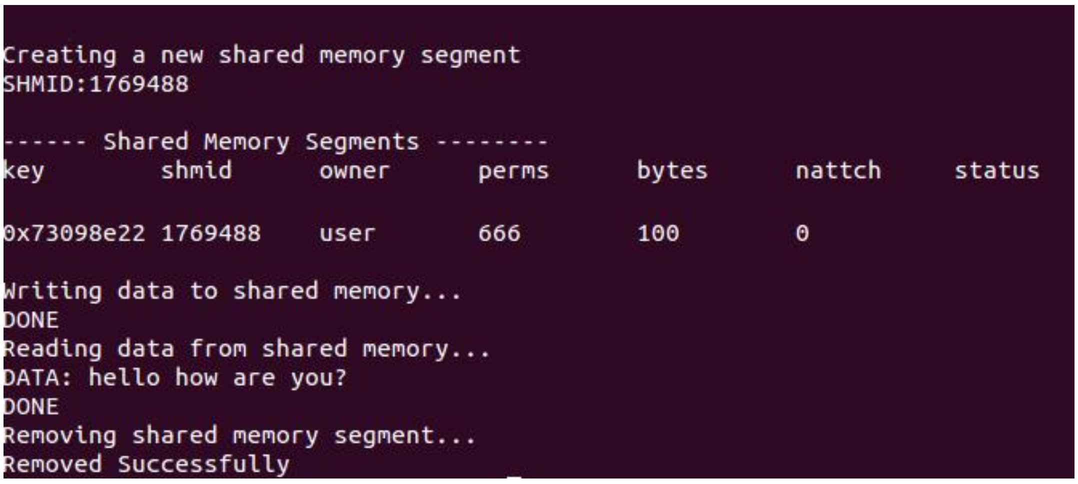
INTER PROCESS COMMUNICATION USING SHARED MEMORY

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<sys/types.h>

#define SEGSIZE 100

int main(int argc, char *argv[])
{
int shmid;
key_t key;
char *segptr;
char buff[] = "hello how are you?";

// Generate a unique key for shared memory
key = ftok(".", 's');

// Try to create a new shared memory segment or get the ID of an
existing one
if ((shmid = shmget(key, SEGSIZE, IPC_CREAT | IPC_EXCL |
0666)) == -1)
{
if ((shmid = shmget(key, SEGSIZE, 0)) == -1)
{
perror("shmget");
exit(1);
}
}
```

```
else
{
printf("Creating a new shared memory segment \n");
printf("SHMID:%d\n", shmid);
}

// Display information about shared memory segments
system("ipcs -m");

// Attach the shared memory segment to the address space of the
calling process
if ((segptr = (char *)shmat(shmid, 0, 0)) == (char *)-1)
{
perror("shmat");
exit(1);
}

// Write data to the shared memory segment
printf("Writing data to shared memory...\n");
strcpy(segptr, buff);
printf("DONE\n");

// Read data from the shared memory segment
printf("Reading data from shared memory...\n");
printf("DATA: %s\n", segptr);
printf("DONE\n");

// Detach the shared memory segment from the address space of the
calling process
if (shmdt(segptr) == -1)
{
perror("shmdt");
exit(1);
}
```

```
// Remove the shared memory segment
printf("Removing shared memory segment...\n");
if (shmctl(shmid, IPC_RMID, 0) == -1)
printf("Can't Remove Shared memory Segment...\n");
else
printf("Removed Successfully\n");

return 0;
}
```

OUTPUT

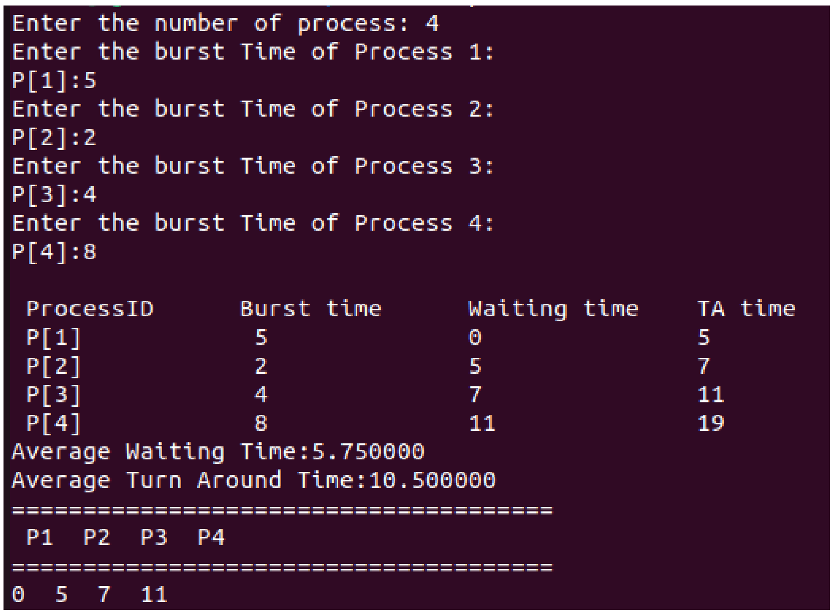


```
Creating a new shared memory segment
SHMID:1769488

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status

0x73098e22 1769488    user       666        100        0

Writing data to shared memory...
DONE
Reading data from shared memory...
DATA: hello how are you?
DONE
Removing shared memory segment...
Removed Successfully
```

FCFS SCHEDULING ALGORITHM

```
#include <stdio.h>

int main()
{
int n;

printf("Enter the number of process\n");
scanf("%d",&n);
int btime[n];
int wtime[n];
int ttime[n];
int i;
float totalw=0;
float totalt=0;

for(i=1; i<=n; i++)
{
        printf("Enter the burst Time of Process %d:",i);
        printf("\nP[%d]:",i);
        scanf("%d",&btime[i]);
}
wtime[1]=0;
for(i=2; i<=n;i++)
{
        wtime[i]=wtime[i-1]+btime[i-1];
}

for(i=1; i<=n; i++)
{
        ttime[i]=btime[i]+wtime[i];
}
printf("\n ProcessID        Burst time    Waiting time          TA time");
for(i=1; i<=n; i++)
printf("\n P[%d]\t\t %d\t\t%d\t\t%d",i,btime[i],wtime[i],ttime[i]);

for(i=1; i<=n; i++)
{
totalw=wtime[i]+totalw;
totalt=ttime[i]+totalt;
}
float avgwt=totalw/n;
float avgtt=totalt/n;
int temp=0;
printf("\nAverage Waiting Time:%f", avgwt);
printf("\nAverage Turn Around Time:%f", avgtt);

printf("\n========================================\n");
for( i=1; i<=n; i++)
{
        printf(" P%d ",i);
}
printf("\n========================================\n");
for( i=1; i<=n; i++)
{
        printf("%d  ",temp);
        temp=temp+btime[i];
}
printf("%d",ttime[n]);

}
```

OUTPUT



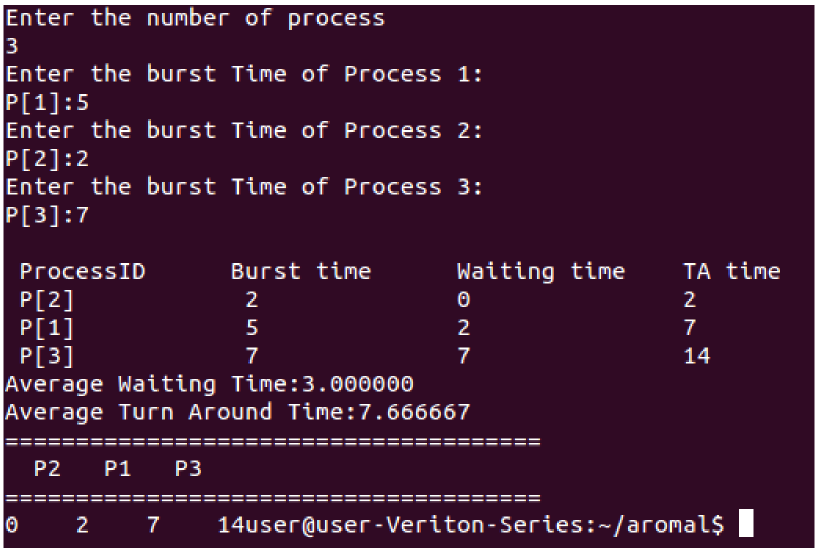
```
Enter the number of process: 4
Enter the burst Time of Process 1:
P[1]:5
Enter the burst Time of Process 2:
P[2]:2
Enter the burst Time of Process 3:
P[3]:4
Enter the burst Time of Process 4:
P[4]:8

 ProcessID      Burst time      Waiting time     TA time
 P[1]            5              0                5
 P[2]            2              5                7
 P[3]            4              7                11
 P[4]            8              11               19
Average Waiting Time:5.750000
Average Turn Around Time:10.500000
======================================
 P1  P2  P3  P4
======================================
0  5  7  11
```

SJF SCHEDULING ALGORITHM

```
#include <stdio.h>
void swap(int *,int *);
int main()
{
        int n;

        printf("Enter the number of process\n");
        scanf("%d",&n);
        int btime[n];
        int wtime[n];
        int ttime[n];
   int temp=0;
        int i,j;
   int pid[n];
        float totalw=0;
        float totalt=0;

        for(i=1; i<=n; i++)
        {
                printf("Enter the burst Time of Process %d:",i);
                printf("\nP[%d]:",i);
                scanf("%d",&btime[i]);
      pid[i]=i;
        }

   for(i=1; i<=n; i++)
        {
        for(j=1; j<=n; j++)
        {
                if(btime[j]>btime[i])
                {
```

```
                //Swapping
          swap(&btime[i],&btime[j]);
          swap(&pid[i], &pid[j]);
        }
          }
    }

         wtime[1]=0;
         for(i=2; i<=n;i++)
         {
                wtime[i]=wtime[i-1]+btime[i-1];
         }

         for(i=1; i<=n; i++)
         {
                ttime[i]=btime[i]+wtime[i];
         }

printf("\n ProcessID        Burst time    Waiting time          TA time");
         for(i=1; i<=n; i++)
printf("\nP[%d]\t\t%d\t\t%d\t\t%d",pid[i],btime[i],wtime[i],ttime[i]);

         for(i=1; i<=n; i++)
         {
         totalw=wtime[i]+totalw;
         totalt=ttime[i]+totalt;
         }
         float avgwt=totalw/n;
         float avgtt=totalt/n;
         printf("\nAverage Waiting Time:%f", avgwt);
         printf("\nAverage Turn Around Time:%f", avgtt);
printf("\n==================================\n");
          for( i=1; i<=n; i++)
          {
                    printf("  P%d ",pid[i]);
          }

printf("\n==================================\n");
    temp=0;
          for( i=1; i<=n; i++)
          {
                    printf("%d     ",temp);
                    temp=temp+btime[i];
          }
    printf("%d",ttime[n]);

}

void swap(int *a,int *b)
{
    int t;
    t=*a;
    *a=*b;
    *b=t;
}
```

OUTPUT



```
Enter the number of process
3
Enter the burst Time of Process 1:
P[1]:5
Enter the burst Time of Process 2:
P[2]:2
Enter the burst Time of Process 3:
P[3]:7

 ProcessID      Burst time      Waiting time     TA time
 P[2]            2              0                2
 P[1]            5              2                7
 P[3]            7              7                14
Average Waiting Time:3.000000
Average Turn Around Time:7.666667
======================================
  P2    P1    P3
======================================
0    2    7    14user@user-Veriton-Series:~/aromal$
```

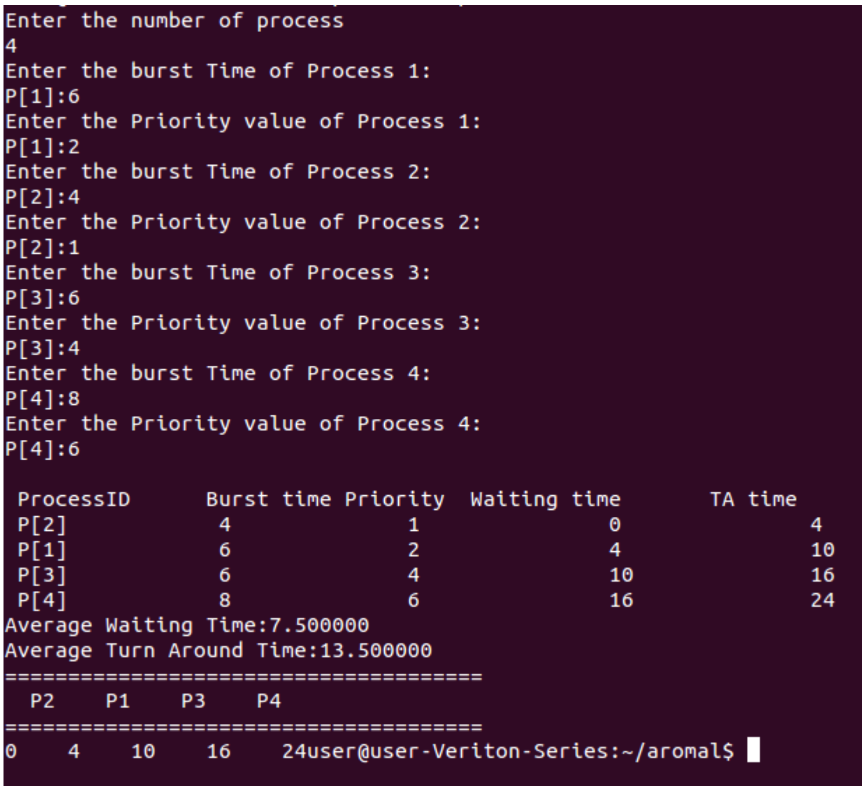
PRIORITY SCHEDULING

```
#include <stdio.h>
void swap(int *,int *);
int main()
{
int n;
printf("Enter the number of process\n");
scanf("%d",&n);
int btime[n];
int wtime[n];
int ttime[n];
int priority[n];
int temp=0;
int i,j;
int pid[n];
float totalw=0;
float totalt=0;
for(i=1; i<=n; i++)
{
 printf("Enter the burst Time of Process %d:",i);
```

```
      printf("\nP[%d]:",i);
      scanf("%d",&btime[i]);
    printf("Enter the Priority value of Process %d:",i);
      printf("\nP[%d]:",i);
      scanf("%d",&priority[i]);
    pid[i]=i;
      }

for(i=1; i<=n; i++)
      {
      for(j=1; j<=n; j++)
      {
      if(priority[j]>priority[i])
       {
      //Swapping
        swap(&priority[i],&priority[j]);
        swap(&btime[i],&btime[j]);
        swap(&pid[i], &pid[j]);
    }
      }
}
      wtime[1]=0;
```

```
for(i=2; i<=n;i++)
{
 wtime[i]=wtime[i-1]+btime[i-1];
}

for(i=1; i<=n; i++)
{
 ttime[i]=btime[i]+wtime[i];
}

printf("\n ProcessID  Burst time Priority  Waiting time    TA time");
for(i=1; i<=n; i++)
 printf("\nP[%d]\t\t%d\t\t%d\t\t%d\t\t%d",pid[i],btime[i],priority[i],w
time[i],ttime[i]);
for(i=1; i<=n; i++)
{
totalw=wtime[i]+totalw;
totalt=ttime[i]+totalt;
}
float avgwt=totalw/n;
float avgtt=totalt/n;
printf("\nAverage Waiting Time:%f", avgwt);
printf("\nAverage Turn Around Time:%f", avgtt);
printf("\n=========================================\n");
for( i=1; i<=n; i++)
{
 printf("  P%d  ",pid[i]);
}
printf("\n=========================================\n");
temp=0;
for( i=1; i<=n; i++)
{
 printf("%d    ",temp);
 temp=temp+btime[i];
}
  printf("%d",ttime[n]);
}
void swap(int *a,int *b)
{
int t;
t=*a;
*a=*b;
*b=t;
 }
```

OUTPUT

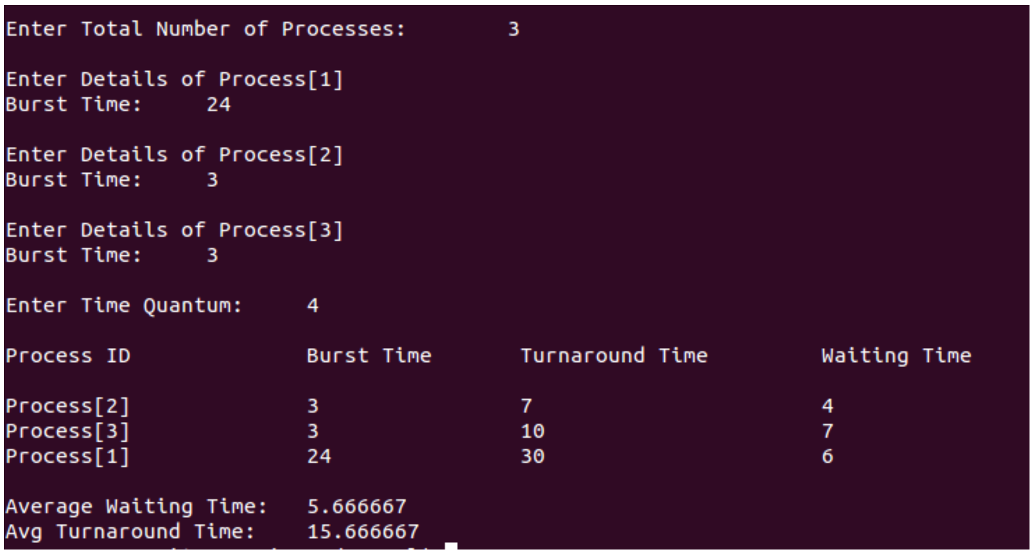


```
Enter the number of process
4
Enter the burst Time of Process 1:
P[1]:6
Enter the Priority value of Process 1:
P[1]:2
Enter the burst Time of Process 2:
P[2]:4
Enter the Priority value of Process 2:
P[2]:1
Enter the burst Time of Process 3:
P[3]:6
Enter the Priority value of Process 3:
P[3]:4
Enter the burst Time of Process 4:
P[4]:8
Enter the Priority value of Process 4:
P[4]:6

 ProcessID      Burst time Priority  Waiting time       TA time
 P[2]            4              1              0               4
 P[1]            6              2              4               10
 P[3]            6              4              10              16
 P[4]            8              6              16              24
Average Waiting Time:7.500000
Average Turn Around Time:13.500000
======================================
  P2     P1     P3     P4
======================================
0    4    10     16     24user@user-Veriton-Series:~/aromal$
```

ROUND ROBIN SCHEDULING

```
#include<stdio.h>
int main()
{
int i, limit, total = 0, x, counter = 0, time_quantum;
int wait_time = 0, turnaround_time = 0,  burst_time[10], temp[10];
float average_wait_time, average_turnaround_time;
printf("\nEnter Total Number of Processes:\t");
scanf("%d", &limit);
x = limit;
for(i = 0; i < limit; i++)
{
printf("\nEnter Details of Process[%d]\n", i + 1);
printf("Burst Time:\t");
scanf("%d", &burst_time[i]);
temp[i] = burst_time[i];
}
printf("\nEnter Time Quantum:\t");
scanf("%d", &time_quantum);
printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting
Time\n");
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] - time_quantum;
total = total + time_quantum;
}
if(temp[i] == 0 && counter == 1)
{
x--;
printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, burst_time[i], total,
total - burst_time[i]);
wait_time = wait_time + total - burst_time[i];
turnaround_time = turnaround_time + total ;
counter = 0;
}
if(i == limit - 1)
{
i = 0;
}
else
{
i++;
}
}
average_wait_time = wait_time * 1.0 / limit;
average_turnaround_time = turnaround_time * 1.0 / limit;
printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
printf("\nAvg Turnaround Time:\t%f\n", average_turnaround_time);
}
```

OUTPUT



```
Enter Total Number of Processes:        3

Enter Details of Process[1]
Burst Time:     24

Enter Details of Process[2]
Burst Time:     3

Enter Details of Process[3]
Burst Time:     3

Enter Time Quantum:     4

Process ID              Burst Time       Turnaround Time          Waiting Time

Process[2]              3                7                        4
Process[3]              3                10                       7
Process[1]              24               30                       6

Average Waiting Time:   5.666667
Avg Turnaround Time:    15.666667
```