WEEK 3

Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system is divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. Use FCFS scheduling for the processes in each queue.

CODE :

```
#include <stdio.h>

int spat[10], upat[10], i, n1, n2, p1[10], p2[10];
int sppt[10], uppt[10], time = 0, op = 0, y, z, pt;
int sptat[10], uptat[10];
int spwt[10], upwt[10];
float spatat = 0, spawt = 0;
float upatat = 0, upawt = 0;

void process(int x, int isSystem) {
   if (isSystem) {
      op += sppt[x];
      sptat[x] = op - spat[x];
      sppt[x] = 0;
      spwt[x] = sptat[x] - p1[x];
      spatat += sptat[x];
      spawt += spwt[x];
```

```
    } else {
        op += uppt[x];
        uptat[x] = op - upat[x];
        uppt[x] = 0;
        upwt[x] = uptat[x] - p2[x];
        upatat += uptat[x];
        upawt += upwt[x];
    }
}

int main() {
    printf("Enter the number of System Processes: ");
    scanf("%d", &n1);

    printf("Enter the number of User Processes: ");
    scanf("%d", &n2);

    printf("Enter the arrival times for System Processes:\n");
    for (i = 0; i < n1; i++)
        scanf("%d", &spat[i]);

    printf("Enter the burst times for System Processes:\n");
    for (i = 0; i < n1; i++)
        scanf("%d", &sppt[i]);

    printf("Enter the arrival times for User Processes:\n");
```

```
for (i = 0; i < n2; i++)
   scanf("%d", &upat[i]);

printf("Enter the burst times for User Processes:\n");
for (i = 0; i < n2; i++)
   scanf("%d", &uppt[i]);

for (i = 0; i < n1; i++)
   time += sppt[i];

for (i = 0; i < n2; i++)
   time += uppt[i];

for (i = 0; i < n1; i++)
   p1[i] = sppt[i];

for (i = 0; i < n2; i++)
   p2[i] = uppt[i];

printf("\n");

while (op < time) {
   y = -1;
   z = -1;

   for (i = 0; i < n1; i++) {
```

```
        if (op >= spat[i] && sppt[i] != 0) {
            y = i;
            break;
        }
    }

    for (i = 0; i < n2; i++) {
        if (op >= upat[i] && uppt[i] != 0) {
            z = i;
            break;
        }
    }

    if (y != -1) {
        printf("%d SP%d ", op, y + 1);
        process(y, 1);
    } else if (z != -1) {
        printf("%d UP%d ", op, z + 1);
        process(z, 0);
    } else {
        op++;
    }
}
printf("%d ",op);

printf("\n");
```

```
    printf("System Processes:\n");
    for (i = 0; i < n1; i++)
        printf("SP%d %d 0\n", i + 1, sptat[i]);
    printf("Average Turnaround Time (System Processes): %.2f\n", spatat / n1);
    printf("Average Waiting Time (System Processes): 0\n");
    printf("User Processes:\n");
    for (i = 0; i < n2; i++)
        printf("UP%d %d %d\n", i + 1, uptat[i], upwt[i]);
    printf("Average Turnaround Time (User Processes): %.2f\n", upatat / n2);
    printf("Average Waiting Time (User Processes): %.2f\n", upawt / n2);

    return 0;
}
```

OBSERVATION BOOK :

12/7/23

Week 3

Write a C program to simulate queue scheduling algorithm for system process with higher priority and user process. Use FCFS algorithm for the process in each queue.

```
#include <stdio.h>
int sat[10], uat[10], i, n1, n2, p1[10], p2[10]
int sbt[10], ubt[10], time = 0, op=0, x, y, z, pt;
int stat[10], swt[10], utat[10], uwt[10];
void process (int n, uSys)
{
  if (uSys)
  {
    op= op+sbt[x];
    stat[x]= op-stat[x];
    swt[x]= stat[x]- p1[x];
    satat = satat+ stat[x];
    spawt = spawt + swt[x];
  }
  else
  {
    op= op+abt[x];
    utat[x]= op-uat[x];
    ubt[x]= utat[x] p2[x];
    uatat = utat[x];
    upawt = upawt + twt[x];
  }
}
```

```
int main()
{
  printf("Enter no of system and user process");
  scanf("%d %d", &n1, &n2);
  printf("Enter arrival and burst time for
          system process");
  for(int i=0; i<n1; i++)
  {
    scanf("%d", &at[i]);
    scanf("%d", &bt[i]);
  }
  printf("Enter arrival and burst time for us
          process");
  for(int i=0; i<n2; i++)
  {
    scanf("%d", &uat[i]);
    scanf("%d", &ubt[i]);
  }
  for(i=0; i<n1; i++)
      time = time + bt[i];
  for(i=0; i<n2; i++)
     time += ubt[i];
  for(i=0; i<n1; i++)
     p1[i]=bt[i];
  for(i=0; i<n2; i++)
     p2[i]=ubt[i];
  while(op<time)
  {
    y=-1;
    z=-1;
    for(i=0; i<n1; i++)
  {
```

```
        if (op>=sat[i] && sbt[i]!=0)
        {
            y=i;
            break;
        }
    }
}
for(i=0; i<n2; i++)
{
    if (op>=uat[i] && ubt[i]!=0)
    {
        x=i;
        break;
    }
}
if (y!=-1)
{
    fruntf ("\d SP \d", op, y+1);
    process(y, 1);
}
else if (x!=-1)
{
    fruntf ("\d UP \d", op, x+1);
    process(x,0);
}
else
    op++;
fruntf ("System process :");
for (i=0; i<n1; i++)
    fruntf ("System process \d \d \n", i+1, stat[i]);
fruntf ("Average TAT : \f", stat[n]);
fruntf ("Average WT : \f", swt[n1]);
```

```
    printf("User processes");
    for(i=0; i<n2; i++)
        printf("User Process %d %d %d", i+1, ubt[i], ...
    printf("Average TAT: %.2f ", utat/n);
    printf("Average WT: %.2f", uwt/n);
    return 0;
}
```

Output:

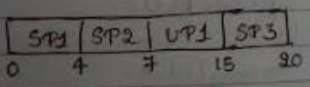
```
Enter the number of system and user processes
3 1
Enter arrival and burst times of system process
0 0 10
4 3 5
Enter arrival and burst times of user process
0
8
System processes:
SP1  4  0
SP2  7  0
SP3  10  0
ATAT: 7.00
AWT: 0.00

User processes:
UP1  15  7
ATAT= 15.00
AWT= 0.0
```

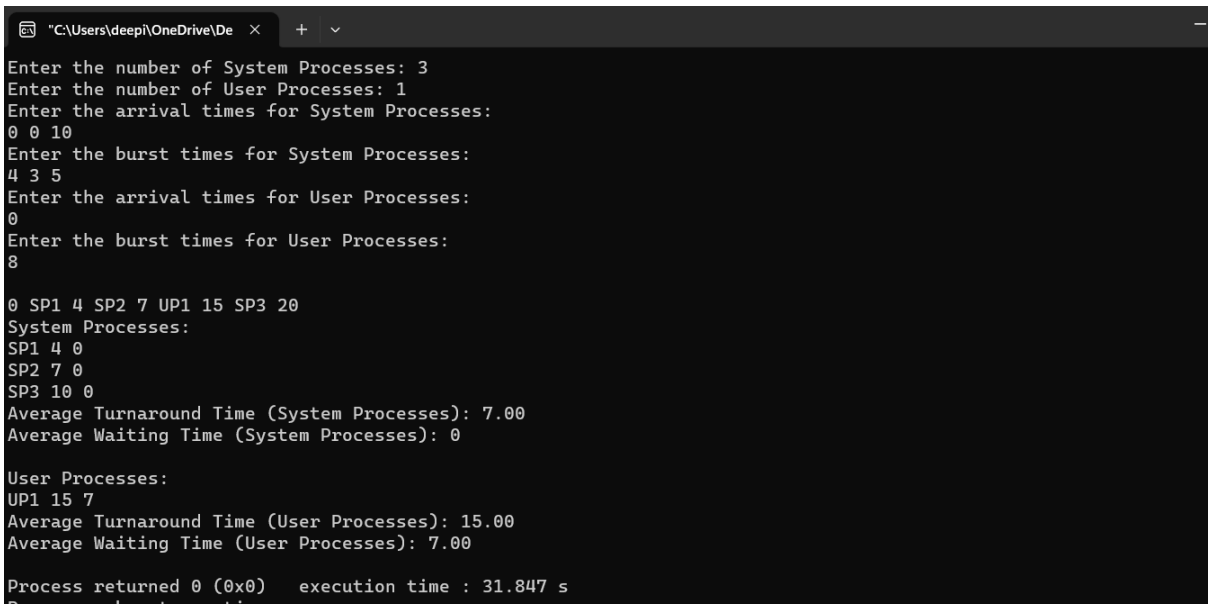
10/10

14/7/23

Gantt chart:



OUTPUT :


"C:\Users\deepi\OneDrive\De
Enter the number of System Processes: 3
Enter the number of User Processes: 1
Enter the arrival times for System Processes:
0 0 10
Enter the burst times for System Processes:
4 3 5
Enter the arrival times for User Processes:
0
Enter the burst times for User Processes:
8
0 SP1 4 SP2 7 UP1 15 SP3 20
System Processes:
SP1 4 0
SP2 7 0
SP3 10 0
Average Turnaround Time (System Processes): 7.00
Average Waiting Time (System Processes): 0
User Processes:
UP1 15 7
Average Turnaround Time (User Processes): 15.00
Average Waiting Time (User Processes): 7.00
Process returned 0 (0x0) execution time : 31.847 s