

## WEEK 3

Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories  $\pm$  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 process 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 process 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 %d\n", i + 1, sptat[i], spwt[i]);
printf("ATAT(System Processes): %.2f\n", spatat / n1);
printf("AWT(System Processes): %.2f\n", spawt / n1);

printf("User Processes:\n");
for (i = 0; i < n2; i++)
    printf("UP%d %d %d\n", i + 1, uptat[i], upwt[i]);
printf("ATAT(User Processes): %.2f\n", upatat / n2);
printf("AWT(User Processes): %.2f\n", upawt / n2);

return 0;
}

```

## OBSERVATION:

12/7/23

### Lab-3

Write a C program to simulate multi-level queue scheduling algorithm.

```
#include <stdio.h>
int spat[10], upat[10], i, n1, n2, p1[10], p2[10];
int sppt[10], uppt[10], time=0, op=0, q, 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) {
        opt = sppt[x];
        sptat[x] = op - spat[x];
        sppt[x] = 0;
        spwt[x] = sptat[x] - p1[x];
        spatat += sptat[x];
        spawt += spwt[x];
    }
    else {
        opt = 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 no. of system processes:");
    scanf("%d", &n1);
    printf("Enter no. of user processes:");
    scanf("%d", &n2);
    printf("Enter arrival time for system process");
    for (i=0; i<n1; i++)
        scanf("%d", &spat[i]);
    printf("Enter process time for system process");
    for (i=0; i<n1; i++)
        scanf("%d", &sppt[i]);
    printf("Enter arrival time for user process");
    for (i=0; i<n2; i++)
        scanf("%d", &upat[i]);
    printf("Enter process time for user process");
    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] = spat[i];
    for (i=0; i<n2; i++)
        p2[i] = upat[i];

    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++;
}
}
```

~~Process~~  
10/10

```
printf("%d", op);
```

```
printf("System Processes : ");
for (i=0; i<M; i++)
    printf("SP%d %d", i+1, spat[i], sppt[i]);
printf("ATAI (System Processes) : %.2f", spatot/n1);
printf("ALOT (System Processes) : %.2f", spawot/n1);
printf("\n");
```



```

printf("user procses: ");
for (i=0; i<n2; i++)
    printf("ATAT (user Procses): %.2f ", upatat/n2);
printf("AWT (user Procses): %.2f ", upawt/n2);
return 0;
}

```

OUT PUT:-

Enter the number of system procs : 3

Enter the number of User procs : 1

Enter the arrival times for system procses:

0 0 10

Enter the procs times for system procses:

4 3 5

Enter the arrival times for user procses:

0

Enter the procs times for user procses:

8

0 SP1 4 SP2 7 UP1 15 SP3 20

System procses:

SP1 4 0

SP2 7 4

SP3 10 5

ATAT (system procses): 7.00

AWT (system procses): 3.00

User procses:

UP1 15 7

ATAT (user procses): 15.00

AWT (user procses): 7

## OUTPUT:

```
"C:\Users\ysrmo\OneDrive - Base PU College\Desktop\4thsem\OS\oslab\lab3\bin\Debug\lab3.exe"
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 process times for System Processes:
4 3 5
Enter the arrival times for User Processes:
0
Enter the process times for User Processes:
8

0 SP1 4 SP2 7 UP1 15 SP3 20
System Processes:
SP1 4 0
SP2 7 4
SP3 10 5
ATAT(System Processes): 7.00
AWT(System Processes): 3.00

User Processes:
UP1 15 7
ATAT(User Processes): 15.00
AWT(User Processes): 7.00

Process returned 0 (0x0)   execution time : 51.340 s
Press any key to continue.

```