OPERATING SYSTEMS LAB - PRACTICAL 5

Name - Sakshi Soni
Roll No - 13

AIM -

Given the list of processes, their CPU burst time, arrival time, priority, and time quantum. Display/print the Gantt chart; compute the average waiting time and average turnaround time for the given scheduling policy

PROGRAM AND OUTPUT -

```
#include <stdio.h>
#include<limits.h>
struct P{
        int bt;
        int at;
        int prio;
        int rt;
        int wt;
        int tat;
};

void fcfs(struct P processes[], int n) {
        int currentTime = 0;
        float avgwt = 0;
        float avgtat = 0;

        printf("Gantt Chart:\n");
        printf("------------\n");
        for (int i = 0; i < n; i++) {
        processes[i].wt = currentTime - processes[i].at;
        if (processes[i].wt < 0) {
```

```
    currentTime = processes[i].at;
    processes[i].wt = 0;
    }
    processes[i].tat = processes[i].wt + processes[i].bt;
    currentTime += processes[i].bt;

    printf("| P%d ", i + 1);

    avgwt += processes[i].wt;
    avgtat += processes[i].tat;
    }

    avgwt /= n;
    avgtat /= n;

    printf("|\n");
    printf("\nAverage Waiting Time: %.2f\n", avgwt);
    printf("Average Turnaround Time: %.2f\n", avgtat);
}

void sjn(struct P processes[], int n) {
    int currentTime = 0;
    float avgwt = 0;
    float avgtat = 0;

    for (int i = 0; i < n; i++) {
    processes[i].rt = processes[i].bt;
    }

    printf("Gantt Chart:\n");
    printf("------------\n");
    while (1) {
    int shortestJobIndex = -1;
    int shortestJobTime = INT_MAX;
```

```
        for (int i = 0; i < n; i++) {
        if (processes[i].at <= currentTime && processes[i].rt <
shortestJobTime && processes[i].rt > 0) {
                shortestJobTime = processes[i].rt;
                shortestJobIndex = i;
        }
        }

        if (shortestJobIndex == -1)
        break;

        processes[shortestJobIndex].wt = currentTime -
processes[shortestJobIndex].at;
        if (processes[shortestJobIndex].wt < 0) {
        currentTime = processes[shortestJobIndex].at;
        processes[shortestJobIndex].wt = 0;
        }
        processes[shortestJobIndex].tat = processes[shortestJobIndex].wt +
processes[shortestJobIndex].bt;
        currentTime += processes[shortestJobIndex].bt;
        processes[shortestJobIndex].rt = 0;

        printf("| P%d ", shortestJobIndex + 1);

        avgwt += processes[shortestJobIndex].wt;
        avgtat += processes[shortestJobIndex].tat;
        }

        avgwt /= n;
        avgtat /= n;

        printf("|\n");
        printf("\nAverage Waiting Time: %.2f\n", avgwt);
        printf("Average Turnaround Time: %.2f\n", avgtat);
}
```

```
void ps(struct P processes[], int n) {
        int currentTime = 0;
        float avgwt = 0;
        float avgtat = 0;

        for (int i = 0; i < n; i++) {
        processes[i].rt = processes[i].bt;
        }

        printf("Gantt Chart:\n");
        printf("------------\n");
        while (1) {
        int highestprioIndex = -1;
        int highestprio = INT_MAX;

        for (int i = 0; i < n; i++) {
        if (processes[i].at <= currentTime && processes[i].prio < highestprio
&& processes[i].rt > 0) {
                highestprio = processes[i].prio;
                highestprioIndex = i;
        }
        }

        if (highestprioIndex == -1)
        break;

        processes[highestprioIndex].wt = currentTime -
processes[highestprioIndex].at;
        if (processes[highestprioIndex].wt < 0) {
        currentTime = processes[highestprioIndex].at;
        processes[highestprioIndex].wt = 0;
        }
        processes[highestprioIndex].tat = processes[highestprioIndex].wt +
processes[highestprioIndex].bt;
```

```
        currentTime += processes[highestprioIndex].bt;
        processes[highestprioIndex].rt = 0;

        printf("| P%d ", highestprioIndex + 1);

        avgwt += processes[highestprioIndex].wt;
        avgtat += processes[highestprioIndex].tat;
        }

        avgwt /= n;
        avgtat /= n;

        printf("|\n");
        printf("\nAverage Waiting Time: %.2f\n", avgwt);
        printf("Average Turnaround Time: %.2f\n", avgtat);
}

void rr(struct P processes[], int n, int tq) {
        int currentTime = 0;
        float avgwt = 0;
        float avgtat = 0;
        int completedProcesses = 0;

        printf("Gantt Chart:\n");
        printf("------------\n");
        while (completedProcesses < n) {
        for (int i = 0; i < n; i++) {
        if (processes[i].at <= currentTime && processes[i].rt > 0) {
                if (processes[i].rt <= tq) {
                processes[i].wt += currentTime - processes[i].at;
                processes[i].tat = processes[i].wt + processes[i].rt;
                currentTime += processes[i].rt;
                processes[i].rt = 0;
                completedProcesses++;
                printf("| P%d ", i + 1);

                avgwt += processes[i].wt;
                avgtat += processes[i].tat;
                } else {
                processes[i].wt += currentTime - processes[i].at;
                processes[i].rt -= tq;
                currentTime += tq;

                printf("| P%d ", i + 1);
                }
        }
        }
        }

        avgwt /= n;
        avgtat /= n;

        printf("|\n");
        printf("\nAverage Waiting Time: %.2f\n", avgwt);
        printf("Average Turnaround Time: %.2f\n", avgtat);
}

int main() {
        int n;
        int tq;

        printf("Enter the number of processes: ");
        scanf("%d", &n);

        struct P processes[n];

        printf("Enter the CPU burst time, arrival time, and prio for each
process:\n");
        for (int i = 0; i < n; i++) {
```

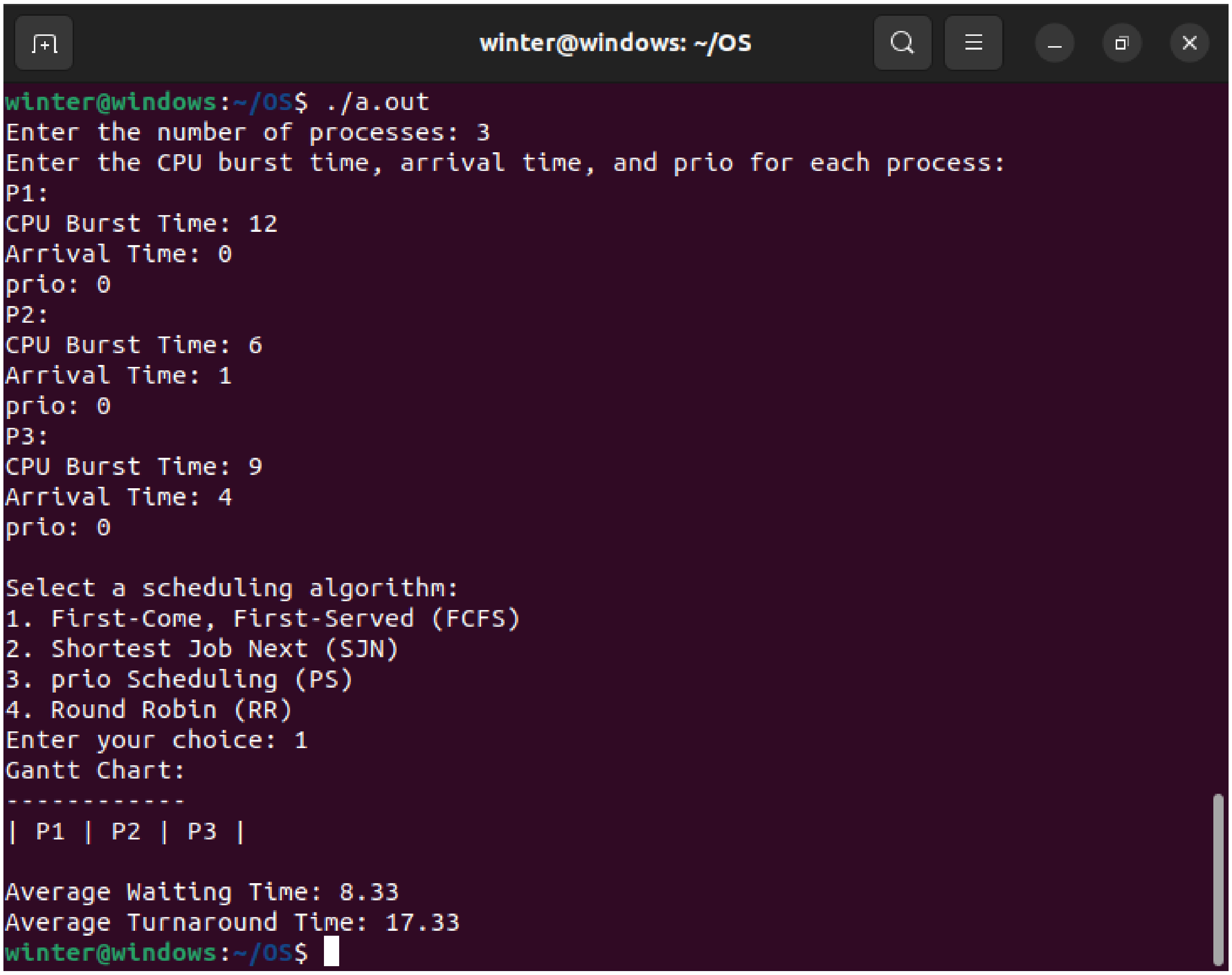
```
printf("P%d:\n", i + 1);
printf("CPU Burst Time: ");
scanf("%d", &processes[i].bt);
printf("Arrival Time: ");
scanf("%d", &processes[i].at);
printf("prio: ");
scanf("%d", &processes[i].prio);
}

printf("\nSelect a scheduling algorithm:\n");
printf("1. First-Come, First-Served (FCFS)\n");
printf("2. Shortest Job Next (SJN)\n");
printf("3. prio Scheduling (PS)\n");
printf("4. Round Robin (RR)\n");
printf("Enter your choice: ");

int choice;
scanf("%d", &choice);

switch (choice) {
case 1:
fcfs(processes, n);
break;
case 2:
sjn(processes, n);
break;
case 3:
ps(processes, n);
break;
case 4:
printf("Enter the time quantum: ");
scanf("%d", &tq);
rr(processes, n, tq);
break;
default:
        printf("Invalid choice!\n");
        break;
        }

        return 0;
}
```



```
winter@windows:~/OS$ ./a.out
Enter the number of processes: 3
Enter the CPU burst time, arrival time, and prio for each process:
P1:
CPU Burst Time: 12
Arrival Time: 0
prio: 0
P2:
CPU Burst Time: 6
Arrival Time: 1
prio: 0
P3:
CPU Burst Time: 9
Arrival Time: 4
prio: 0

Select a scheduling algorithm:
1. First-Come, First-Served (FCFS)
2. Shortest Job Next (SJN)
3. prio Scheduling (PS)
4. Round Robin (RR)
Enter your choice: 1
Gantt Chart:
------------
| P1 | P2 | P3 |

Average Waiting Time: 8.33
Average Turnaround Time: 17.33
winter@windows:~/OS$ 
```

RESULT -

Linux C programs on different CPU scheduling policies have been implemented.