

Assignment - 3

OPERATING SYSTEM

TOPIC: Process Scheduling, - PART1

NAME: Devesh Tulshyan

MCA SEM-3

SECTION A

Roll Number: 22

-
1. Write a C programme to simulate the following **non-preemptive** CPU scheduling algorithms to find the turnaround time and waiting time for the above problem.
 - a. FCFS
 - b. SJF
 - c. Priority

★ FCFS CPU SCHEDULING ALGORITHM

- For the FCFS scheduling algorithm, read the number of processes/jobs in the system, and their CPU burst times.
- The scheduling is performed based on the arrival time of the processes, irrespective of their other parameters.
- Each process will be executed according to its arrival time.
- Calculate the waiting time and turnaround time of each of the processes accordingly.

★ SJF CPU SCHEDULING ALGORITHM

- For the SJF scheduling algorithm, read the number of processes/jobs in the system, and their CPU burst times.
- Arrange all the jobs in order with respect to their burst times.
- Two jobs may be in queue with the same execution time, and then the FCFS approach will be performed.
- Each process will be executed according to the length of its burst time.
- Then calculate each process's waiting time and turnaround time accordingly.

★ PRIORITY CPU SCHEDULING ALGORITHM

- For the priority scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times, and the priorities.

- Arrange all the jobs in order with respect to their priorities.
- There may be two jobs in queue with the same priority, and then FCFS approach will be performed.
- Each process will be executed according to its priority.
- Calculate the waiting time and turnaround time of each of the processes accordingly.

CODE :-

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_PROCESSES 100

typedef struct {
    int id;
    int burst_time;
    int priority;
    int arrival_time;
    int waiting_time;
    int turnaround_time;
} Process;

void calculate_fcfs(Process processes[], int n);
void calculate_sjf(Process processes[], int n);
void calculate_priority(Process processes[], int n);

void swap(Process *a, Process *b) {
    Process temp = *a;
    *a = *b;
    *b = temp;
}

void print_results(Process processes[], int n) {
    printf("Process ID\tBurst Time\tArrival Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\t%d\n", processes[i].id, processes[i].burst_time,
        processes[i].arrival_time, processes[i].waiting_time, processes[i].turnaround_time);
    }
}

int main() {
    int n;
```

```

Process processes[MAX_PROCESSES];

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

printf("Enter the arrival times and burst times for each process:\n");
for (int i = 0; i < n; i++) {
    processes[i].id = i + 1;
    printf("Process %d:\n", i + 1);
    printf("Arrival Time: ");
    scanf("%d", &processes[i].arrival_time);
    printf("Burst Time: ");
    scanf("%d", &processes[i].burst_time);
    processes[i].priority = 0;
}

printf("\nFCFS Scheduling:\n");
calculate_fcfs(processes, n);
print_results(processes, n);

printf("\nSJF Scheduling:\n");
for (int i = 0; i < n; i++) {
    processes[i].waiting_time = 0;
    processes[i].turnaround_time = 0;
}
calculate_sjf(processes, n);
print_results(processes, n);

printf("\nPriority Scheduling:\n");
printf("Enter the priorities for each process:\n");
for (int i = 0; i < n; i++) {
    printf("Process %d:\n", i + 1);
    printf("Priority: ");
    scanf("%d", &processes[i].priority);
}
for (int i = 0; i < n; i++) {
    processes[i].waiting_time = 0;
    processes[i].turnaround_time = 0;
}
calculate_priority(processes, n);
print_results(processes, n);

return 0;
}

```

```

void calculate_fcfs(Process processes[], int n) {
    int time = 0;
    for (int i = 0; i < n; i++) {
        if (time < processes[i].arrival_time) {
            time = processes[i].arrival_time;
        }
        processes[i].waiting_time = time - processes[i].arrival_time;
        processes[i].turnaround_time = processes[i].waiting_time + processes[i].burst_time;
        time += processes[i].burst_time;
    }
}

```

```

void calculate_sjf(Process processes[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (processes[i].burst_time > processes[j].burst_time) {
                swap(&processes[i], &processes[j]);
            }
        }
    }
}

```

```

    int time = 0;
    for (int i = 0; i < n; i++) {
        if (time < processes[i].arrival_time) {
            time = processes[i].arrival_time;
        }
        processes[i].waiting_time = time - processes[i].arrival_time;
        processes[i].turnaround_time = processes[i].waiting_time + processes[i].burst_time;
        time += processes[i].burst_time;
    }
}

```

```

void calculate_priority(Process processes[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (processes[i].priority < processes[j].priority) {
                swap(&processes[i], &processes[j]);
            } else if (processes[i].priority == processes[j].priority) {
                if (processes[i].arrival_time > processes[j].arrival_time) {
                    swap(&processes[i], &processes[j]);
                }
            }
        }
    }
}

```

```
}  
  
int time = 0;  
for (int i = 0; i < n; i++) {  
    if (time < processes[i].arrival_time) {  
        time = processes[i].arrival_time;  
    }  
    processes[i].waiting_time = time - processes[i].arrival_time;  
    processes[i].turnaround_time = processes[i].waiting_time + processes[i].burst_time;  
    time += processes[i].burst_time;  
}  
}
```

OUTPUT :-

```
devesh225@devesh: ~  
devesh225@devesh:~$ gcc scheduling_algorithms.c  
devesh225@devesh:~$ ./a.out  
Enter the number of processes: 5  
Enter the arrival times and burst times for each process:  
Process 1:  
Arrival Time: 1  
Burst Time: 5  
Process 2:  
Arrival Time: 2  
Burst Time: 3  
Process 3:  
Arrival Time: 6  
Burst Time: 4  
Process 4:  
Arrival Time: 2  
Burst Time: 4  
Process 5:  
Arrival Time: 5  
Burst Time: 3
```

FCFS Scheduling:

Process ID	Burst Time	Arrival Time	Waiting Time	Turnaround Time
1	5	1	0	5
2	3	2	4	7
3	4	6	3	7
4	4	2	11	15
5	3	5	12	15

SJF Scheduling:

Process ID	Burst Time	Arrival Time	Waiting Time	Turnaround Time
2	3	2	0	3
5	3	5	0	3
4	4	2	6	10
3	4	6	6	10
1	5	1	15	20

Priority Scheduling:

Enter the priorities for each process:

Process 1:

Priority: 2

Process 2:

Priority: 1

Process 3:

Priority: 2

Process 4:

Priority: 1

Process 5:

Priority: 3

Process ID	Burst Time	Arrival Time	Waiting Time	Turnaround Time
1	5	1	0	5
4	4	2	4	8
2	3	2	8	11
5	3	5	8	11
3	4	6	10	14

devesh225@devesh:~\$