Assignment - 4

OPERATING SYSTEM

TOPIC: Process Scheduling, - PART2

QUESTION

Write a C program to simulate a multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are 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.

DESCRIPTION

A multi-level queue scheduling algorithm is used in scenarios where the processes can be classified into groups based on properties like process type, CPU time, IO access, memory size, etc. In a multi-level queue scheduling algorithm, there will be 'n' number of queues, where 'n' is the number of groups the processes are classified into. Each queue will be assigned a priority and will have its own scheduling algorithm like round-robin scheduling or FCFS. For the process in a queue to execute, all the queues of priority higher than it should be empty, meaning the process in those high-priority queues should have completed its execution. In this scheduling algorithm, once assigned to a queue, the process will not move to any other queues.

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_PROCESSES 10

typedef struct {
    int process_id;
    int arrival_time;
    int burst_time;
} Process;
void sortByArrivalTime(Process queue[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (queue[j].arrival_time > queue[j + 1].arrival_time) {
                Process temp = queue[j];
                queue[j] = queue[j + 1];
                queue[j + 1] = temp;
            }
        }
    }
}
void executeQueue(Process queue[], int n, const char *queue_name) {
    printf("\nExecuting %s Queue (FCFS Scheduling):\n", queue_name);
    int currentTime = 0;
    for (int i = 0; i < n; i++) {
```

```
        if (queue[i].arrival_time > currentTime) {
            currentTime = queue[i].arrival_time;
        }
        printf("Process ID: %d | Arrival Time: %d | Burst Time: %d | Start Time: %d | Finish Time: %d\n",
               queue[i].process_id,
               queue[i].arrival_time,
               queue[i].burst_time,
               currentTime,
               currentTime + queue[i].burst_time);
        currentTime += queue[i].burst_time;
    }
}
int main() {
    int n;
    Process systemQueue[MAX_PROCESSES], userQueue[MAX_PROCESSES];
    int systemCount = 0, userCount = 0;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        int process_id, arrival_time, burst_time, type;
        printf("\nEnter details for process %d\n", i + 1);
        printf("Process ID: ");
        scanf("%d", &process_id);
        printf("Arrival Time: ");
        scanf("%d", &arrival_time);
        printf("Burst Time: ");
        scanf("%d", &burst_time);
        printf("Process Type (0 for System, 1 for User): ");
        scanf("%d", &type);
        if (type == 0) {
            systemQueue[systemCount].process_id = process_id;
            systemQueue[systemCount].arrival_time = arrival_time;
            systemQueue[systemCount].burst_time = burst_time;
            systemCount++;
        } else {
            userQueue[userCount].process_id = process_id;
            userQueue[userCount].arrival_time = arrival_time;
            userQueue[userCount].burst_time = burst_time;
            userCount++;
        }
    }
    sortByArrivalTime(systemQueue, systemCount);
    sortByArrivalTime(userQueue, userCount);
    if (systemCount > 0) {
        executeQueue(systemQueue, systemCount, "System");
    } else {
        printf("\nNo system processes to execute.\n");
    }
    if (userCount > 0) {
        executeQueue(userQueue, userCount, "User");
    } else {
        printf("\nNo user processes to execute.\n");
```

```
    }

    return 0;
}
```

OUTPUT:

```
namrata@NamraRio:~/MCA2023/Namrata_B_34/assignment4$ nano multiLevelQueue.c
namrata@NamraRio:~/MCA2023/Namrata_B_34/assignment4$ gcc -o multiLevelQueue multiLevelQueue.c
namrata@NamraRio:~/MCA2023/Namrata_B_34/assignment4$ ./multiLevelQueue
Enter the number of processes: 4

Enter details for process 1
Process ID: 1
Arrival Time: 0
Burst Time: 8
Process Type (0 for System, 1 for User): 1

Enter details for process 2
Process ID: 4
Arrival Time: 5
Burst Time: 4
Process Type (0 for System, 1 for User): 0

Enter details for process 3
Process ID: 3
Arrival Time: 5
Burst Time: 1
Process Type (0 for System, 1 for User): 0

Enter details for process 4
Process ID: 2
Arrival Time: 4
Burst Time: 5
Process Type (0 for System, 1 for User): 1

Executing System Queue (FCFS Scheduling):
Process ID: 4 | Arrival Time: 5 | Burst Time: 4 | Start Time: 5 | Finish Time: 9
Process ID: 3 | Arrival Time: 5 | Burst Time: 1 | Start Time: 9 | Finish Time: 10

Executing User Queue (FCFS Scheduling):
Process ID: 1 | Arrival Time: 0 | Burst Time: 8 | Start Time: 0 | Finish Time: 8
Process ID: 2 | Arrival Time: 4 | Burst Time: 5 | Start Time: 8 | Finish Time: 13
```