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.

```c
#include <stdio.h>
#include <string.h>

#define MAX_PROCESSES 100

typedef struct {
    int id;
    int burst_time;
    char type[10];
} Process;

void calculateTimes(Process processes[], int n, int wt[], int tat[]) {
    wt[0] = 0;

    for (int i = 1; i < n; i++) {
        wt[i] = 0;
        for (int j = 0; j < i; j++) {
            wt[i] += processes[j].burst_time;
        }
    }

    for (int i = 0; i < n; i++) {
        tat[i] = processes[i].burst_time + wt[i];
    }
}

void printResults(Process processes[], int n) {
    int wt[n], tat[n];
    calculateTimes(processes, n, wt, tat);

    printf("Process ID  Type    Burst Time  Waiting Time  Turnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("   %d        %s       %d          %d             %d\n",
               processes[i].id, processes[i].type, processes[i].burst_time, wt[i], tat[i]);
```

```
    }
}

void multiLevelQueueScheduling(Process processes[], int n) {
    Process systemQueue[MAX_PROCESSES];
    Process userQueue[MAX_PROCESSES];
    int systemCount = 0, userCount = 0;

    for (int i = 0; i < n; i++) {
        if (strcmp(processes[i].type, "System") == 0) {
            systemQueue[systemCount++] = processes[i];
        } else {
            userQueue[userCount++] = processes[i];
        }
    }

    printf("System Processes:\n");
    printResults(systemQueue, systemCount);

    printf("\nUser Processes:\n");
    printResults(userQueue, userCount);
}

int main() {
    Process processes[] = {
        {1, 10, "System"},
        {2, 5, "User"},
        {3, 8, "System"},
        {4, 6, "User"},
        {5, 3, "System"}
    };
    int n = sizeof(processes) / sizeof(processes[0]);

    printf("Multi-Level Queue Scheduling:\n");
    multiLevelQueueScheduling(processes, n);

    return 0;
}
```

```
Multi-Level Queue Scheduling:
System Processes:
Process ID  Type    Burst Time  Waiting Time  Turnaround Time
   1        System      10          0            10
   3        System      8           10           18
   5        System      3           18           21

User Processes:
Process ID  Type    Burst Time  Waiting Time  Turnaround Time
   2        User        5           0            5
   4        User        6           5            11
PS C:\Users\Rudradeep\Desktop\OS Assignment\Os Lab>
```