Multilevel-Feedback-Queue scheduler

Operating systems project

Group members

Name	Tasks
Shahad Ahmed Alqarni 2111214	• Queue 2 (FCFS) • Display Process information
Shahd Ali Alshikhi 2111228	• Queue 0 (RR) • Queue 1 (RR)

Table of contents

01 Introduction

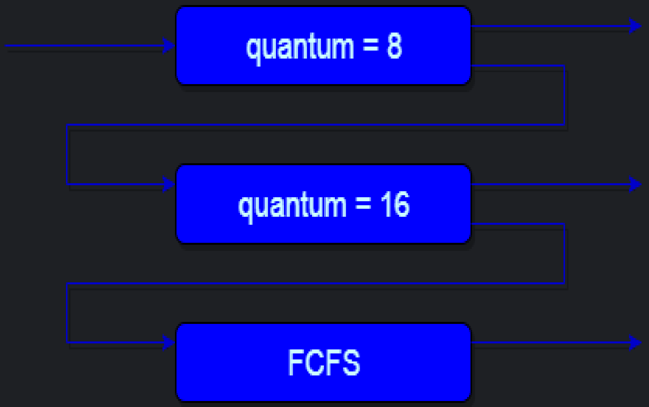
- All three different type of processes has their own queue. Each queue has its own Scheduling algorithm
- New job enters queue Q0
- What will happen if all the queues have some processes? Which process should get the CPU? To determine this Scheduling among the queues is necessary:

  1-Fixed priority preemptive scheduling

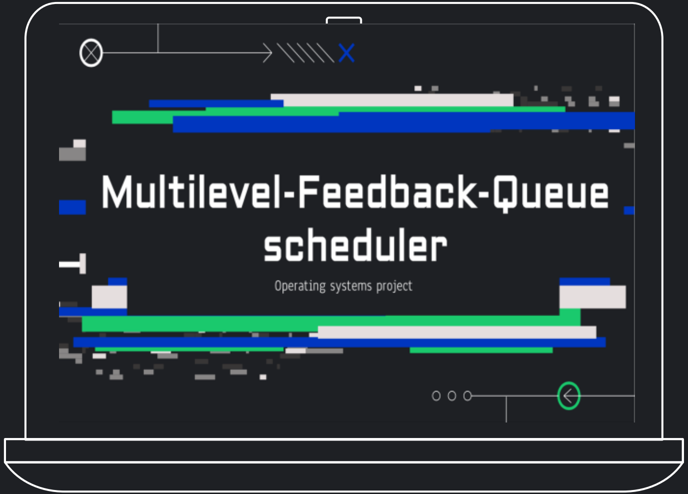
  queue 0 > queue 1 > queue 2

  2-Time slicing

  each queue gets a certain portion of CPU time and can use it to schedule its own processes



02 Features and capabilities


Multilevel-Feedback-Queue scheduler

Operating systems project

- The processes moves between 3 different queues.
- apply various kind of scheduling for different processes.
- Fixed priority preemptive scheduling.
- Calculate the total time taken and throughput then display it to user.

03 Simple user manual

1 User input

Each time user run the program they will be asked to enter number of processes they wish to schedule

After the first input user will be asked to enter brust time for each process to schedule

2 Program calculations

For each process the queue in which processes has finished is determined and the waiting time, turnaround time, response time, is calculated and represented in the output

04

Project code

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

#define MAX_PROCESSES 100
#define QUANTUM1 8
#define QUANTUM2 16

//processes struct to hold information of each process
typedef struct Process {
  int responseTime;
  int turnArountTime;
  int BurstTime;
  int waitingTime;
  char Q[3];
} Process;
```

04 Project code

```
//first round robin queue Q0
//the queue will let the process enter queue for 8ms
void RR1(Process *a, int length, int *timer) { //timer to keep track of process
  for (int i = 0; i < length; i++) {
    a[i].responseTime = *timer; //response to process request
    if (a[i].BurstTime <= QUANTUM1) { //check brust time if it less than or equal to the queue
quantum 8
      *timer += a[i].BurstTime;
      a[i].turnAroundTime = *timer; //arrival time is zero so the turnaround time will equal timer
      a[i].BurstTime = 0;
      strcpy(a[i].Q, "Q0");
      continue;
    }
    a[i].BurstTime -= QUANTUM1; //decrease brust time by the quantum of the first queue
    *timer += QUANTUM1; //increase timer by quantum of the first queue 8
  }
}
```

04

Project code

```
/second round robin queue Q1
//the queue will let the process enter queue for 16ms
void RR2(Process *a, int length, int *timer) {
  for (int i = 0; i < length; i++) {
    if (a[i].BurstTime == 0) continue; // check if the process has reminaning brust time if not then the
proccess finished continue
    if (a[i].BurstTime <= QUANTUM2) { // check if the brust time is less than or equal to the queue
quantum 16
      *timer += a[i].BurstTime; //increase timer by brust time of the process
      a[i].turnArountTime = *timer; //the turnaround time of the process will equal timer
      a[i].BurstTime = 0; //the process done exceuting so the brust will be zero
      strcpy(a[i].Q, "Q1");
      continue; //continue to another process
    }
    a[i].BurstTime -= QUANTUM2; //if the brust is not less than or equal to the queue quantum 16
decrease brust time
    *timer += QUANTUM2; //increasse timer for the process by  quantum of the second queue 16
  }
}
```

04 Project code

```
//third queue Q2 first come first serve
//the queue will exceute the riminaning processes
void FCFS(Process *a, int length, int *timer) {
  for (int i = 0; i < length; i++) {
    if (a[i].BurstTime == 0) continue; //if the process finished continue to the next
    *timer += a[i].BurstTime; //increase timer by the remining brust time for the process
    a[i].BurstTime = 0;//the process done exceuting so the brust will be zero
    a[i].turnArountTime = *timer; //the turnaround time of the process will equal timer
    strcpy(a[i].Q, "Q2");
  }
}
```

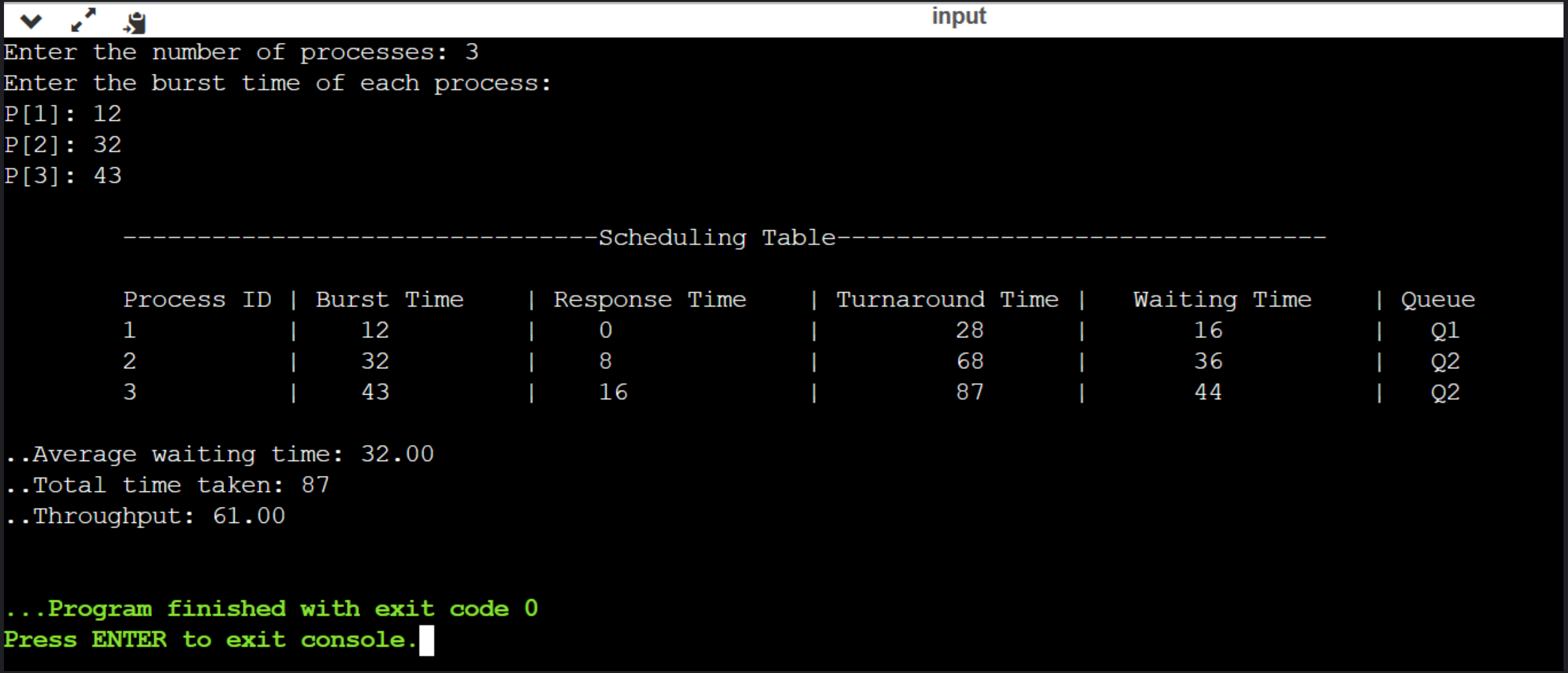
04

Project code

```c
// display the processes and their information
void printProcesses(Process *processes, int length, int bursttime[MAX_PROCESSES]) {
  printf("\n\t--------------------------------Scheduling Table--------------------------------\n\n");
  printf("\tProcess ID | Burst Time    | Response Time    | Turnaround Time |   Waiting Time    |
Queue\n");
  int total_waiting_time = 0;
  for (int i = 0; i < length; i++) {
    processes[i].waitingTime = processes[i].turnAroundTime - bursttime[i]; //calculate waiting time of
each process
    total_waiting_time += processes[i].waitingTime;
    printf("\t%d\t   | \t%d\t   | \t%d\t      | \t%d\t| \t%d\t    |\t%s\n", i + 1,bursttime[i],
processes[i].responseTime, processes[i].turnAroundTime, processes[i].waitingTime,  processes[i].Q);
}
printf("\n..Average waiting time: %.2f\n", (float)total_waiting_time / length); //calculate avarege
waiting time of all processes
}
```

02

Project code



```
Enter the number of processes: 3
Enter the burst time of each process:
P[1]: 12
P[2]: 32
P[3]: 43

        --------------------------------Scheduling Table------------------------------------

        Process ID | Burst Time     | Response Time     | Turnaround Time |  Waiting Time     | Queue
        1          |    12          |    0              |        28       |     16            |  Q1
        2          |    32          |    8              |        68       |     36            |  Q2
        3          |    43          |    16             |        87       |     44            |  Q2

..Average waiting time: 32.00
..Total time taken: 87
..Throughput: 61.00


...Program finished with exit code 0
Press ENTER to exit console.
```

Thanks!