Data structures I

ROUND ROBIN SCHEDULING

Marwan ElSafty | 4690 | 28/3/2018

The Round Robin Scheduling:

One of the main functions of the operation system is to schedule processes that the processor executes. One of the very popular algorithms for this scheduling is called Round Robin, where each process enters a queue, executes in a FIFO order, and after a specified time its execution stops and returns at the end of the queue if it still need time to continue execution or aborts if it has finished its required time of execution.

Queue implementation:

- **Header file:**

```
#ifndef QUEUE_H_INCLUDED
#define QUEUE_H_INCLUDED
#define TYPE process typedef struct {
    char name[10];
    int AT, BT;
}  process;
typedef struct node {
    TYPE value;
    struct node * next;
     }  node;t
     typedef struct Queue {
         node * front, * rear;
     }  Queue;
     void initialize(Queue * s);
     void enqueue(Queue * s, TYPE value);
     TYPE dequeue(Queue * s);
     int isEmpty(Queue * s);#
     endif // QUEUE_H_INCLUDED
```

The process struct is a defined structure containing the process' name, arrival time and burst time

(it is defined in the header file to use it as the Queue’s type)

- **Source file:**

```
#  include < stdio.h >
#include < stdlib.h >
#define TYPE process# include "Queue.h"
node * newNode(TYPE value) {
    node * n = malloc(sizeof(node));
    n - > value = value;
    n - > next = NULL;
    return n;
}
      void initialize(Queue * s) { //Queue * s; //s = (Queue *)malloc(sizeof(Queue));
          s - > front = NULL;
          s - > rear = NULL;
          return s;
      }
      void enqueue(Queue * s, TYPE value) {
          node * n = newNode(value);
          if (s - > front == NULL) {
              s - > front = n;
              s - > rear = n;
          } else {
              s - > rear - > next = n;
              s - > rear = n;
          }
      }
      TYPE dequeue(Queue * s) {
          TYPE value;
          /*if(s->front == NULL)        return NULL;*/ // else //{
          value = s - > front - > value;
          node * temp = s - > front;
          s - > front = s - > front - > next;
          free(temp);
          if (!s - > front) s - > rear = NULL; //}
          return value;
      }
      int isEmpty(Queue * s) {
          return s - > front == NULL ? 1 : 0;
      }
```

Functions used:

- **readFile:**

read the watching time slots, the processes, their arrival time and their burst time from the file given and returns the length of the processes.

```
int readFile(FILE * fr, process processes[], int * slots) {
    int i = 0; //Read watching time slots
    fscanf(fr, "%d", slots);
//Read the process' name, its arrival time and its burst time
    while (!feof(fr)) {
        fscanf(fr, "%s %d %d", processes[i].name, & processes[i].AT, & processes[
i].BT);
        i++;
    }
    return i; // return the number of processes
}
```

- **checkAT:**

loop on every process to check its arrival time, if it is the same as the current time slot the process in enqueued at the end of the queue.

```
void checkAT(process processes[], int length, Queue * mainQ, int i) {
    int j;
    for (j = 0; j < length; j++) {
        if (processes[j].AT == i) enqueue(mainQ, processes[j]);
    }
}
```

- **checkBT:**

used to check the burst time of the given process, if it is equal to 0 then the process aborts , if not, the process is re-enqueued to the end of the queue.

```
void checkBT(process x, Queue * mainQ) {
    if (x.BT == 0)
 //if time required in processor is finished the process aborts
        printf("%s aborts", x.name);
    else //if not it is resent to the end of the queue
        enqueue(mainQ, x);
}
```

- **RoundRobin:**

the scheduling function pseudo code:

- Initialize a queue
- checks the arrival time of each process.
- Starts looping of the time slots:
  1. If queue is empty:
     The processor is idle
     Check arrival time of processes
  2. If not empty:
     Dequeue the first inserted process
     Print its name and time slot
     Decrement its burst time
     Check the processes arrival time
     Check the current process burst time
- Stop scheduling

```
1. void RoundRobin(process processes[], int * slots, int length) {
2.     Queue mainQ;
3.     initialize( & mainQ);
4.     int i;
5.     checkAT(processes, length, & mainQ, 0);
6.     for (i = 0; i < * slots; i++) {
7.         if (isEmpty( & mainQ)) {
8.             printf("idle\t(%d-->%d)\n", i, i + 1);
9.             checkAT(processes, length, & mainQ, i + 1);
10.              } else {
11.                  process x = dequeue( & mainQ);
12.                  printf("%s\t(%d-->%d)", x.name, i, i + 1);
13.                  x.BT--;
14.                  checkAT(processes, length, & mainQ, i + 1);
15.                  checkBT(x, & mainQ);
16.                  printf("\n");
17.              }
18.          }
19.          printf("Stop");
20.     }
```