**Data Structure**

**Assignment**

**Round Robin**

**Name: Ahmed Mahmoud Ismail**

**ID: 4572**

**General Description:**

To schedule processes fairly, a round-robin scheduler generally employs [time sharing](https://en.wikipedia.org/wiki/Time-sharing), giving each job a time slot or quantum (its allowance of CPU time), and interrupting the job if it is not completed by then. The job is resumed next time a time slot is assigned to that process. If the process terminates or changes its state to waiting during its attributed time quantum, the scheduler selects the first process in the ready queue to execute. In the absence of time-sharing, or if the quanta were large relative to the sizes of the jobs, a process that produced large jobs would be favored over other processes.

Round-robin algorithm is a pre-emptive algorithm as the scheduler forces the process out of the CPU once the time quota expires.

For example, if the time slot is 100 milliseconds, and job1 takes a total time of 250 ms to complete, the round-robin scheduler will suspend the job after 100 ms and give other jobs their time on the CPU. Once the other jobs have had their equal share (100 ms each), job1 will get another allocation of [CPU](https://en.wikipedia.org/wiki/CPU) time and the cycle will repeat. This process continues until the job finishes and needs no more time on the CPU.

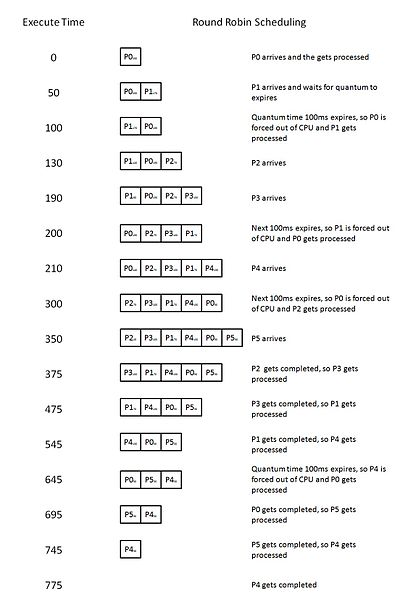
* **Job1 = Total time to complete 250 ms (quantum 100 ms)**.

1. First allocation = 100 ms.
2. Second allocation = 100 ms.
3. Third allocation = 100 ms

|  |  |  |
| --- | --- | --- |
| **Process name** | **Arrival time** | **Execute time** |
| P0 | 0 | 250 |
| P1 | 50 | 170 |
| P2 | 130 | 75 |
| P3 | 190 | 100 |
| P4 | 210 | 130 |
| P5 | 350 | 50 |

1. Total CPU time of job1 = 250 ms

Consider the following table with the arrival time and execute time of the process with the quantum time of 100ms to understand the round-robin scheduling:

[](https://en.wikipedia.org/wiki/File:RoundRobin.jpg)

Another approach is to divide all processes into an equal number of timing quanta such that the quantum size is proportional to the size of the process. Hence, all processes end at the same time. "[CPP Code and Algorithm design for Round Robin Scheduling](http://www.introtoalgo.com/2018/03/round-robin-algorithm.html)".

**The Algorithm:**

* 1. Start to read from the file.
  2. While we don’t get to the end of the file.
  3. We read char char from the file and print it on the screen.
  4. And if c equal to the equal sign we will increase the flag by one and enter it in a loop again unitl the end of the line and we will increase the flag by one to know that the processing time is taken.

And if c equal space skip the loop and enter it again to continue.

And if c not equal to \n means not get to the end of the line we will put the processing time on the string then cast it to int when c equals to the end of the line and break that loop.

* 1. check if the flag bigger than zero means that the processing time we will put the first c in the first process name and reset the flag and counters to use them again
  2. We will make another while loop until not equal the end of the file and also read c from the file and print it on the screen and check.

If c equals to space increase the flag by one and continue the loop and the flag here indicates which part of the line I’m in it.

Else If c is number we will check if the flag equal 1 means it will read the start time and put it in the string of start time and increase the number of items else if the flag equal 2 means it will read the execute time else means there is still no space and this number is the name of the process.

* 1. Else if c equals to \n equal to the end of the line reset every counter and cast the strings start time and execute time to integer and save it in int in the structure process and increase the counter i to receive the next data of the second process with the second loop
  2. Else it’s not a space or number means it the number and put it in the name of the process.
  3. If we reach end of the file end every string with the null and reset the counter and end the file.

And then enter the algorithm of the process.

1. Loop till the end of the processing time.
2. If time equal zero Loop till the number of items in the array and check if the array start time equal zero and have a name then put it the index in the queue.
3. Loop while the queue is not empty the temp will take the first element in the queue and decrease it’s execute time and check if the execute time became zero will print aborted and increase the time and check again if there is item have a start time equal to the now time and put it in the queue.
4. Else if the execution time not equal zero then print it and increase the time and check again if there is item has start time equal to now time then put the element again in the queue.
5. Else the queue is empty print idle and check again the start time of items to put in the queue
6. Then at the end print stop.