**Faculty of Engineering - Cairo University**

**Credit Hours System**

**Fall 2022**

**CMPN302: Operating Systems**

Project Report

**Students:**

Karim Hafez (1190432)

Seifallah Mohamed Kishtah (1190243)

Ahmed Hany Abdelwahab (1200017)

Mai Abdelhameed (1190365)

**Submitted To**

• Eng.Philopeteer Nabil

• Eng.Berlnty Kerlos

**Data structures:**

* Struct of algorithm contains name(char c[3]), Quantum(int), preemptive(bool), process\_count(integer)
* Struct of PCB contains id(int), pid(int), finishtime(int), priority(int), runningtime(int),remaingtime(int), start\_time(int) &arrival time(int)
* Priority Queue adjusted to 3 types

1. Shortest time Queue
2. Priority Queue
3. Normal Queue

Declaration -> Node\*PQueue;

* Message Queue-> for sending and receiving process between processes
* Shared Memory-> for saving remaining time to both scheduler and process
* Boolean to indicate if process generator is done->is\_generator\_done
* Array of Priority queues with 11 priorities[0->10] for Multilevel feedback loop->MFL\_Queue[11]
* **Algorithms**

1. Shortest Job First

Data Structures Used:

Input File

First Input file

1 6 29 3

2 7 12 5

3 8 7 2

Second Input file

1 6 29 3

2 14 12 5

3 21 7 2

4 23 2 6

Output File

<SJF_case.txt>

<SJF_preemtive_case_1.txt>

<SJF_preemptive_case_2.txt>

Explanation

1. Highest Priority First

Data Structures Used

Input file

First Input file

1 6 29 3

2 7 12 5

3 8 7 2

Second Input file

1 6 29 3

2 14 12 5

3 21 7 2

4 23 2 6

Output file

<HPF_case.txt>

<HPF_preemptive_case_1.txt>

[HPF\_preemptive\_case\_2 .txt](HPF_preemptive_case_2%20.txt)

Explanation

1. Round Robin

Data Structures Used

Single normal queue which enqueues process at end of the queue -> Node\*ready\_qeueu

Input File

Input file

1 6 29 3

2 7 12 5

3 8 7 2

Case 1 with Quantum = 1

Case 2 with Quantum = 2

Case 3 with Quantum = 3

Output File

[RR\_case\_1 .txt](RR_case_1%20.txt)

<RR_case_2.txt>

<RR_case_3.txt>

Explanation

1. Multi-level Feedback Queue

Data Structures Used

Array of priority queues-> Node\* MFL\_queue[11];

Shared memory for remaining time storage-> shmaddr\_process;

Last-level refers to the prior priority of the running process before it took its quantum

Input File

First Input file

1 6 29 3

2 7 12 5

3 8 7 2

Second Input file

1 6 29 3

2 14 12 5

3 21 7 2

4 23 2 6

Output File

[MFL\_case\_1.txt](MFL_case_2.txt)

<MFL_case_2.txt>

Explanation

The each queue refers to a priority and each dequeue of the head happens during a quantum as then inserted at the end of a quantum at the end of the other queue below it in priority.

dequeue(&MFL\_queue[i]);

Last\_level refers to the level of the priority that the running process was extracted from so that it is incremented and reused to enqueue the running process in the level below it

enqueue(&MFL\_queue[last\_level + 1]);

if a new process is enqueue in a priority higher than the processes in the current queue the for loop forces the function to start working on that process until it reaches the process as the other processes.

* **Calculations**

[**code-20221225T160239Z-001\code\process\_logs.txt**](code-20221225T160239Z-001/code/process_logs.txt)

**Workload Division:**

|  |  |
| --- | --- |
| Seifallah Eshta | Shortest Job first |
| Karim Hafez | Round Robin |
| Ahmed Hany | Multilevel feedback queue |
| Mai Abdelhameed | Highest Priority |