CMPN303 – Operating Systems

Cairo University - Faculty of Engineering

Project Report

Submitted to:

Dr. Mona Farouk - Eng. Hussein Fadl - Eng. Omar Farrag

Submitted By:

Aliaa Khalifa Ali – 1170315

Aalaa Mohamed Mohamed – 1170106

Shorouk Mohamed Roshdy - 1170110

Lina Ayman Hamza - 1170113

Submission Date:

18/1/2021

- ## Algorithm for Highest Priority Non-preemtive (HPF):

Data Structure:

Ascending (priority) sorted linked list (because low priority value maps to a high priority process.

Algorithm:

1- We get the number of processes in the queue then loop while number > 0.
2- If we received a message meaning that the process arrived, we update its remaining time and push it in the priority list.
3- Then we check if there is no process running now (was forked), and check the clock to get the process that is supposed to run at this time
4- If the priority list is not empty, we update the running process and its data
5- Remove it from the list.
6- Update the flag to indicate that a process is now running (forked).

Assumptions:

1- If two processes arrive at the same time, we take the first in the file.

- ## Algorithm for Round Robin (RR) & Shortest Remaining Time First (SRTF)

Both methods have common algorithm for receiving processes from the message queue, but they differ in the data structure used to stored messages, and their preemption reason.

Data Structure:

- For RR: Queue
- For SRTF: Priority Linked-list, sorted according to the remaining time for each process (ascendingly).

Preemption When:

- For RR: When the running process runs for time = the quantum.
- For SRTF: When a process has less remaining time, than the currently running process.

Common Code Algorithm:

- **Memory Management part:** Initially check if the received process can be allocated in the memory (phase 2 part), if so, it's inserted in the message queue/list. If not, it's inserted in the waiting queue. Each time a process is finished, we check again if we can allocate a process from the waiting queue in the message queue/list.
- The code keeps executing until we reach the total number of processes (no more processes to be executed).
- In the while loop where apply the previous condition, we check on the first process (say process **x)** of the queue/list. We have different states. So, if:
  1. **No currently running process:**
     If this was the first time process x is executed (state= -1), it's forked, and its data is initialized (start time, waiting time, etc), and setting running process state with 1.

     If process x was executed before (but interrupted), we update its data, and set the running process state with 1.

  2. **If a process is currently running:**
     We compare the currently running process with the process x according to the preemption condition for each method (RR or SRTF).

     If the preemption condition is satisfied, the running process is stopped and replaced by process x (running process is now process x), process x data is also updated (start time, waiting time, …etc.) and forked it was its first time running. The previously running process data is also updated and inserted back into the queue/list.

     If the preemption condition isn't satisfied, the loop keeps executing until on the previous cases occur.

RR Assumption: If there are 2 processes with the same arrival time, they're inserted in the queue according to their order in the input file.

- **Work Load distribution:**

Team Member	Task Assigned
Aliaa Khalifa	HPF, RR, & phase 3
Aalaa Mohamed Mohamed	HPF, RR, & phase 2
Lina Ayman Hamza	HPF, SRTF, & phase 2
Shorouk Mohamed Roshdy	HPF, SRTF, & phase 2

- **Phases Time Table:**

Tasks	Days Taken
Phase 1	4 days
Phase 2	2 1/2 days
Phase 3	1 day