**OS​ ​Scheduler**report

Team members:

* Mohamed Gamal
* Mostafa Yasser
* Berlnty Kerlos

**Algorithm explanation:**

* At first, the process generator starts then asks the user about his preferred algorithm

1. non-preemptive HPF.
2. Shortest Remaining Time Next.
3. Round Robin (for this choice the user is asked for the quantum).

* Then the process generator runs the clock and scheduler (send the quantum if needed) children.
* Then sends every process in its due time.
* Waits for the scheduler to be terminated then it clear resources and terminates itself.

1. HPF algorithm:

* Data structure used:
  + - * priority queue as a container for ready processes.
      * Overloaded operator < to sort the queue according to the higher priority.
* The scheduler is a while loop of 3 operations
  + - * Read incoming processes and push them in the ready queue.
      * Apply the HPF logic on the available processes and run the appropriate one.
      * Pause itself till the running process is terminated.
* Print the related information through the program.
* Finally terminates itself.

1. Shortest Remaining Time Next:

* Data structure used:
* Priority queue as a container for ready processes.
* Overloaded operator < to sort the queue according to the shortest remaining time.
* First before receiving a message in the while loop I checks if the queue size is greater than 0 as if so, the queue contains process so I stop the running process after it finishes , so I calculate the remaining time of the process and pop it . If queue is not empty, I adding the wait time for each process in the queue.
* Second if I received a message from scheduler so I add the process in the queue.
* Third I checks if all messages finished and queue is empty to ensure that Scheduler has finished
* Forth thing after receive message from scheduler , I check if queue is not empty , I set the current to the future running process , if this process is new so its default pid=-3 so I fork it else I continue executing it.
* After finishing all process I terminate in step 3.

1. Round Robin:

* Data structure used:
  + - * Queue as a container for the ready processes.
* Assumption:
  + - * After running a process and it is not terminated yet but the quantum is over the following sequence takes place:
        + Send SIGSTOP to the process and reduce its remaining time.
        + Read the incoming processes and push them in the ready queue.
        + Then push this process in the queue.
        + Finally run the RR algorithm to run the next process.
* At first, the scheduler receives the quantum value sent by the process generator.
* The scheduler is a while loop of 3 operations
  + - * Read incoming processes and push them in the ready queue.
      * Apply the RR logic on the available processes and run the appropriate one.
      * Pause itself till the running process is terminated or the quantum is over.
* In case that the quantum is over (SIGALRM is triggered) the SIGALRM handler turn off the alarm, sends SIGSTOP to the running process, reduce the remaining time of the process and push the process back in the queue.
* In case that the process is terminated (SIGCHLD is triggered) the SIGCHLD handler turn off the alarm and print required data of the process.
* Print the related information through the program.
* Finally terminates itself.

**Results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | CPU utilization | AVvg waiting | Avg WTA | Std WTA |
| **HPF** | **100%** | **3.61** | **10.57** | **7.76** |
| **SRTN** |  |  |  |  |
| **RR** | **81.40%** | **2.94** | **613.86** | **11.51** |

**Time Taken for each task:**

|  |  |
| --- | --- |
| task | time |
| HPF | 2 days |
| SRTN | 4 days |
| RR | 3 days |

**Workload Distribution:**

1. Reading processes & Constructing Process Generator : Berlnty Kerlos & Mohamed Gamal
2. Designing Schedule : Berlnty Kerlos & Mohamed Gamal
3. Designing HPF : Berlnty Kerlos & Mohamed Gamal
4. Designing SRTN : Mohamed Gamal & Mostafa Yasser
5. Designing RR : Berlnty Kerlos
6. Displaying results (brief & files.log) : Berlnty Kerlos & Mostafa Yasser