Process Scheduler Phase 1 Report

🚩 Team Number 3

🚩 Team Members

Name	Code	Sec
Abdelruhman Sami Mohammad	9220430	1
Ahmed Hamed Gaber	9220027	1
Amir Anwar Bkhiet	9220166	1
George Magdy Tawfek	9220231	1

🚩 Data Structure Used

1. Min-Heap

- **Implementation**:
  Implemented in a generic way where it stores data in array of void* and takes compare function to be able to compare data stored to apply it's logic
- **Usage**:
  Used to implement priority-queue to be able which is used in scheduling algorithms as :
  - SRTN algorithm
  - HPF algorithm

2. List

- **Implementation**
  Implemented as a doubly linked list to provide fast insertion and deletion to be compatible with it's usage
- **Usage**
  Handles the dynamic allocations and maintaining a building block for queue ds

3. Queue

- **Implementation**
  It's all about an interface above list ds to restrict & specify normal queue operations as push/pop/empty...etc
- **Usage**
  Used in RR scheduling algorithm

🚩 Algorithm Explanation & Results

For All Algorithms

- One ready-queue and an algorithm variable are set in a switch case - in the scheduler main - function according to input to decide the type of the ready-queue & set the algorithm function to be called to apply scheduling directly, providing a generic and extensible way of implementation.
- For SRTN & HPF algorithms, each one has a compare function to be sent to the priority queue.

1. HPF Algorithm

- **Explanation**
  It's a non-preemptive algorithm, where if process entered execution, it's completes until it finishes execution unless it gets stop signal.
- **Results**
  - The HPF provided more concerning on handling processes with the ones who's priority is higher.
  - This caused somehow starvation for less prior processes resulting in high `avg waiting time` and high `avg WTA`.
  - It's not recommended at all to use it alone for managing process. It can be used in a hybrid system.

2. SRTN Algorithm

- **Explanation**
  It's a preemptive algorithm, each clock cycle it checks on the arrival of a new process and compares it's remaining time with the one running to decide whether applying context switch or complete executing the current process.
- **Results**
  - It provided a very efficient `avg WTA time` as it minimized it so much.
  - However, the `avg waiting time` is relatively high compared to that of RR algorithm with low quantum but it's better than that of HPF.
  - This happens due to starvation of processes with long runtime.
  - Processes are not processed according to their level of priority as HPF.

3. RR Algorithm

- **Explanation**
  New processes are received in a **FIFO queue**, context switch is applied each quantum time/on process termination.
- **Results**
  - It provides fairness between processes where it got the best `avg waiting time` between all scheduling algorithms.

- Each quantum, context switch occurs and another process is being processed.
- This affected negatively somehow the value of `avg WTA`.
- On increasing quantum time, the `avg waiting time` & `avg WTA` increases.
- It also provided the best CPU utilization.

🚩 Assumptions

- We assume the max number of input processes is 1000 process. We do this to save terminated processes' WTA to be used to calculate standard deviation.
- In RR, if only one process is available, it will start and resume at the same time.
- In SRTN, if a process is running with remaining time = 3 & another process arrived with remaining time = 3, the one already running will continue as it is.

🚩 Work Load

Assignee	Task
Ahmed	List Data Structure
	Scheduler main setup
	3 scheduling alogrithms
	Bug hunting
George	Queue Data Structure
	Handling output file
	Calculated performance results
Abdelruhman	Min-Heap Data Structure
	Signal Handling between processes & scheduler
	Process Class
	Bug hunting
Amir	process generator class
	Making pretty console output format
	Signal handling between process generator & scheduler
	Preempting
	Bug hunting