Project Summary

# Simulation overview

1. Before running scheduling algorithms, we first wait for the process generator to check if there is any arrived process in this clock cycle.
2. Assuming there will be x process arrived in this clock cycle – x ꞓ [0,…) – we loop over x receiving those processes, if x is zero then no process will be received and the 2 files keep going
3. After receiving each process, we push it to a waiting queue where we can test memory allocation of this process if possible before adding it to the suitable ready queue to be scheduled next.
4. After checking incoming processes, we schedule the next process or keep the current one running depending on the current scheduling algorithm described in algorithm section below.
5. In parallel with these operations the process file which is running if it is its turn to run or suspended using SIGSTOP if it’s not its turn to run yet.

# Data structures:

|  |  |
| --- | --- |
| Task | Data structure used |
| Processes Arrival | Sequential queue |
| Ready processes in Round Robin | Sequential queue |
| Ready Processes in SRTN & HPF | Priority Queue implemented using binary heap |
| Waiting queue (processes not allocated yet) | Sequential queue |
| Finished processes | Sequential queue |
| Memory representation | Bitmap implement using normal array |

# Algorithms overview

## Shortest Remaining Time Next SRTN

1. After checking the waiting queue (step 3 in the overview) it is inserted to the queue with a priority representing the remaining time for it to finish execution
2. Now the queue is ready and has the shortest remaining time process on top of it.
3. Here we have two cases: -

* If the process on top is different from the currently running one, then it means we need to preempt it, by sending it a SIGSTOP and sending the new Process SIGCONT congratulation preempted :)
* If the process was the same, then to need for preemption and keep going.

1. Next decrease its heap key of the top process by one

## Highest Priority First HPF

1. After checking the waiting queue (step 3 in the overview) it is pushed to the queue with a priority representing the priority for it red from the file
2. Now the queue is ready and has the shortest remaining time process on top of it.
3. Now we check if there is a currently running process decrease its remaining time and just go as it is not a preemption algorithm.
4. If there is no currently running process, then get the top process of the queue set its priority to some low priorities so that it is always on the top of the queue then run it using SIGCONT.

## Round Robin RR

1. This is simpler step than the previous algorithms, just push it to the tail of the queue after memory allocating it.
2. Each clock cycle we check for two things.

* If the quantum has reached zero take the current process and throw it to the tail of the queue, else do nothing just decrease the remaining time by one in the pcb structure.
* If the process sent termination signal, then remove it from the queue and reset the quantum to the given quantum so that the next process takes its quantum.

## Memory Allocation Algorithms

## 1)First Fit

We loop on the memory array and see if we can find available consecutive bytes for the process. If we find it, we allocate the process.

## 2)Buddy Memory Allocation

1. Round the size of the process up to the nearest power of 2.
2. loop on the memory array to find the minimum number of consecutive bytes (To find the best fit block) that can hold the process.
3. I found, allocate the process.

# Assumptions

* A process decreases its remaining time when it finishes the execution of it, which means if a process started at 2 it will decrease one at the beginning of second 3
* No negative priorities.

# Workload distribution

|  |  |  |
| --- | --- | --- |
| Task | Implementer | Time taken |
| Process Generator and Process files | Abdallah |  |
| Process generator synchronization with scheduler | Mohammed Adel | 30 mins implementing it,  I did not count thinking of it :D |
| Round Robin Algorithm | Mohammed Adel | 2 hours |
| Highest Priority First | Fares | 1 Hour |
| Shortest Remaining Time Next | Abdallah | 12 Hours |
| Process Synchronization | Whole team | 2 Hours |
| First Fit Memory Allocation | Omar | 3 Hours |
| Buddy Memory Allocation | Fares & Omar | 1 Hour |
| Testing, error and corner cases handling | Whole team | Many Hours |