Grand Canyon University

Project 3: Short-Term Process Scheduler

Evan Lee and Ryan Scott

CST-315: Operating Systems Lecture and Lab

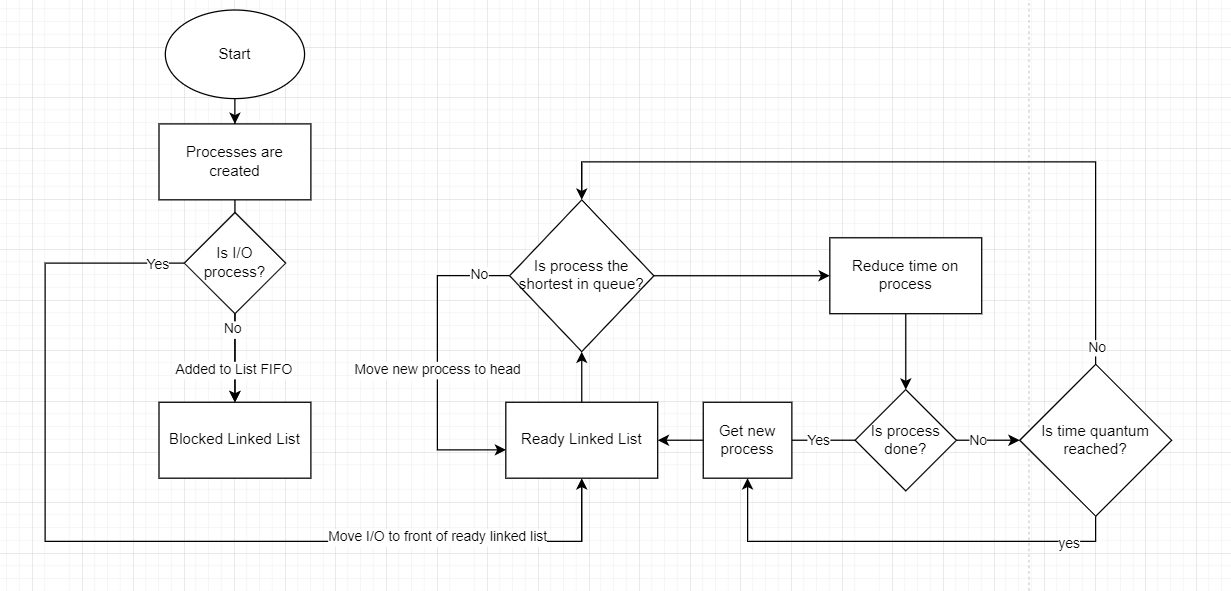
Dr. Ricardo Citro

March 29, 2022

**Implementation and Introduction:**

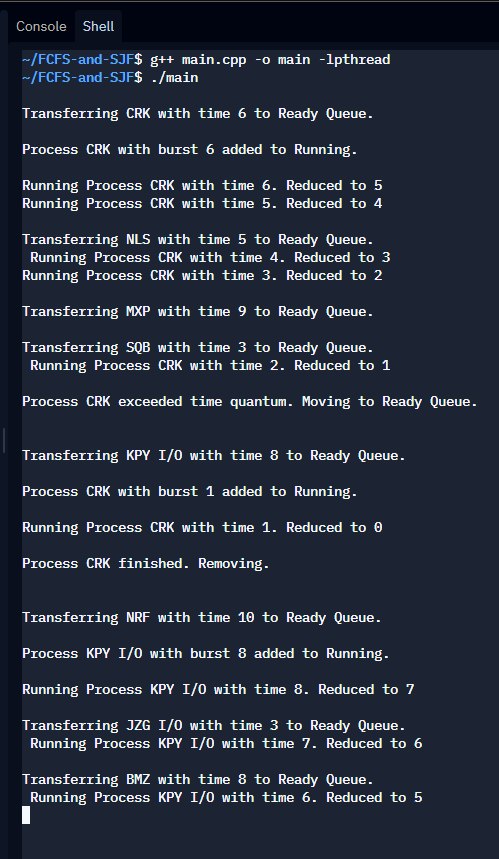
When deciding how to implement a scheduler into our shell, the type of scheduling algorithm must be considered. We decided to use both First In First Out (FIFO) and Shortest Job First (SJF) in our scheduler. When processes are created and initially sent to the blocked queue, the scheduler will use FIFO when determining which processes will be sent to the ready queue. Then in the ready queue the scheduler will sort the processes by their burst time with the shortest being at the head of the linked list while the longest are sent to the back of the linked list. The head of the ready list will be the node in which the process is running and getting CPU time. When creating processes, we have a two-thirds chance for the next process made to be a user process and a one-third chance for the next process to be an I/O process in which it needs to take priority and be sent to the head of the ready list as soon as possible.

**Explanation of Design Flowchart on Short-Term Process Scheduler:**

****

As processes are created, they are checked to see if they are an I/O process or not. If they are, they are moved directly to the head of the ready list and are given the state READY. If they not, they are added to the blocked list with the state BLOCKED and will be brought into the ready list using FIFO. Once in the ready linked list, a SJF scheduling algorithm will be used to determine which process will be moved to the head of the list, in which the state of the node will be switched from READY to RUNNING. Once a process is in the RUNNING state, the CPU would be allocating resources to it in which the burst time of the process will decrease by one second. The process in the RUNNING position will change if one of three things happens: the process finishes, an I/O process is added, or the time quantile is reached. When this happens the next process in the linked list will be set to RUNNING while the previous process will either be deleted or set to READY or BLOCKED depending on what needs to be done.

**Code Output:**



The first process that is added is process CRK that has a burst time of 6. It is added directly to the running queue since it is the first process. Then after some time process NLS is added to the ready queue but is not sent to the running queue because it has a higher burst time. The same happens with process MXP. Further down, process KPY I/O is added to the ready queue, and since it is an I/O it takes priority. Since our scheduler is not preemptive, when the next process is finished the process KPY I/O is added to the running queue and begins to execute.

**Test and Validate:**

When using linked lists, it is especially important that whenever a new node is created as a temporary node to traverse a list, it must be deleted when it is finished. Otherwise, the program could suffer from memory overflow and crash. One problem that we needed to validate was that our ready linked list was indeed being sorted correctly with the I/O processes first, then sorted by shortest job to slowest job. This was done by exclusively printing out our ready linked list as we ran our program to double check that the correct processes are being executed.

**Analysis and Conclusion on Short-Term Process Scheduler:**

Through our implementation of the Short-Term Process Scheduler, it was found that it can be useful to use different types of scheduling algorithms depending on how the processes want to be handled. Using FIFO can help include processes that were starting sooner priority while SJF can give priority to processes that have a shorter burst time which can lower the average waiting time for processes. Using pthreads is a must for a scheduler to be able to handle moving processes, creating processes, and executing a process at the same time.