Erik Roeckel
2/7/20
CS 1550

Pros and Cons of Using Priority Queue with Semaphore

There are a few pros of using a priority queue within a semaphore. First of all, a priority queue allows us to solve the indefinite blocking/ starvation problem with semaphores. The starvation problem occurs when two or greater processes end up waiting indefinitely for a lock to be released by another waiting process. This lock will never be released and will ultimately lead to a deadlock. This problem normally occurs when a last-in, first-out queue is used within a semaphore to order the processes. Using a priority queue solves this problem because the higher priority process will always be at the top of the queue. Another pro of using a priority queue is that it allows us to solve the process ordering problem with semaphores. When using semaphores with a regular queue there is no sense of priority. Being that there is no sense of priority there is no way to put certain processed in front of other processes in the queue, even if they should be executed before the others. With a priority queue, there is a process priority tied to each node. Using this priority we can use a priority queue inside of our semaphore implementation, in order to ensure that the process with the highest priority is always executed first. However, we can’t always guarantee that the process with the highest priority will be run first, this is a large disadvantage to using a priority queue within a semaphore.

One major con to using a priority queue within a semaphore is the large scheduling challenge that arises. This scheduling challenge arises when a high priority process enters the queue but a lower priority process or processes are already accessing the kernel. Being that these lower priority processes already have access to read and write to the kernel, this kernel space will be protected by a lock. This kernel space will be locked until one lower priority process unlocks the kernel data, which then means that the priority queue didn’t serve its purpose. This problem is formally known as priority inversion. Another con to using a priority queue within a semaphore is a situation when you have three processes in the queue. Let’s say the first process is utilizing a resource that another process with a high priority needs. This process will wait until this resource is released by the other process. But now say a lower priority process arrives that doesn’t need this shared resource. Now this lower priority process will cause the other high priority process to wait until it’s done executing.