CIS 450 Project 2 Report

1. Our implementation of the multi-level feedback queue scheduler involves using six pointers to manage the next process in each queue. In the scheduler function, a for loop loops through every queue starting from the highest priority queue, q1, and then in a second for loop starting from the queue’s pointer, begins looking for the next process in the ptable whose queue variable is equal to that queue.

If a process is found and it is runnable, the scheduler will run until the time is up based on which queue it is and how much quantum time each process gets in that, or until the process is done or an interrupt occurs. The queue’s pointer is then increased to the next process in the ptable, so it knows where to begin looking next time it loops through.

If the quantum time is up and the process is still runnable, then the process will be sent to the next queue, unless it is in queue 6, in which case it will be sent to the back of the queue as the pointer will increase to the next process and therefore this process will not be run again until every other queue 6 process has had a chance to run.

If the process becomes a zombie its queue variable is set to 0 to denote that this process is not in any queue. At this point we break from both loops and the scheduler logic runs from the beginning.

We keep a boost timer and a quantum timer to determine when to boost every process to q1, and when to stop running a process on the cpu and send it to the next queue, respectively. Almost all of our scheduler’s code is in proc.c’s scheduler function, with a few helper functions at the bottom of proc.c. Proc.h has the new variable, int queue, added to the process struct.

1. Evan Shriner – wrote initial scheduler logic, bugfixing, testing.

Edward Urban – wrote boost code, revised scheduler logic to use six pointers, report, bugfixing.