Assigment 1 Report

Jan Ocampo & Thomas Griffin

CS433 Xiaoyu Zhang

February 13th, 2017

Files included in project:

Source files: main.cpp, PCB.cpp, PCB.h, queue.cpp, queue.h, readyqueue.cpp, readyqueue.h

Executable: assignment1

Instructions: To build the executable just type make assignment1, to run type ./assignemnt1

During runtime you will be asked to enter 1 or 2, enter 1 to run test 1 and enter 2 to run test 2

debrief:

From tests, the program seems to function for test one and mostly functions for test two.

It used to throw a segmentation fault randomly every few times it was ran but we're pretty sure its been resolved. For the most part we didn't really use the status of a process because we weren't exactly sure how exactly you wanted that to be implemented. We implemented each feature of the ready queue that was required and we implemented the ready queue with something of our own design. We made an array of 50 queues, each queue corresponds to a pid, that way if two processes of the same pid are added the first one that was added is removed. We split the array of 50 into 3 theoretical different priority levels (high, medium, and low). High priority processes have a pid of 1 to 15, medium priority processes have a pid of 16 to 32, and low priority processes have a pid of 33 to 50. The time complexity for this method should be fairly low. Inserts should be almost instant as the process is just added to the queue of it's corresponding pid so o(1) time? The remove function is what takes the longest as it needs to update the new highest value. Since it only has to search within a given priority level i.e of n indexes it would have a worst case time complexity of o(n), its best case is that there are two processes with the same pid so the time complexity would be o(1). Since indexes are broken up into 3 different levels for this test even our worst case is pretty fast since at max it only needs to check about 16 queues if they're empty. We looked back at our cs311 code to get our queue class functioning. Overall our implementation was really convoluted and we probably should have just implemented the priority queue using a heap or some other balanced binary tree since those have really good worst case time complexities instead of trying to reinvent the wheel. As far as comments for the instructor go, the prompt wasn't exactly descriptive. At some points (like how you wanted the statuses of the processes to be implemented) we weren't sure what you wanted and just kind of went with the flow (thought this is our bad as well since we could of just emailed you) .