Nicholas Everekyan, Kevin Gallagher
CS433: Operating Systems
Professor Zhang
September 10th, 2024

Programming Assignment 1 Report

Choice of Data Structure:

The data structure we chose to implement for our ReadyQueue was a binary heap. With a binary heap, we can assure that processes with the highest priority can be quickly accessed, which is essential for scheduling in operating systems. Binary heaps also provide ample resources between insertion and removal operations, which are common in a ready queue scenario.

Expected Time Complexity:

Insertion (addPCB): In a binary heap, inserting an element (a new PCB) takes O(logn) time, where n is the number of processes in the queue. This is because the heap maintains its balanced structure after each insertion.

Removal (removePCB): Removing the highest-priority element, which is the root of the heap, also takes O(logn) time. This includes removing the root and then calling reheapify to maintain its properties.

Accessing the Highest Priority Element: Accessing the root element (the process with the highest priority) is O(1), as it's always at the top of the heap.

Size and Display Operations: Checking the size of the queue is O(1), and displaying all elements would take O(n).

Timing Results:

With using a binary heap, the insertion and removal operations scale logarithmically. This means that as the number of processes grows, the time for each operation will increase. There would only be noticeable delays for a very large number of processes. Here are a set of results of time from test2 without doing any optimization in the Makefile:

- Time taken: 0.040151 seconds
- Time taken: 0.0411131 seconds
- Time taken: 0.0391007 seconds

Average: 0.0401216 seconds

Due to the scalability of binary heap, the timing measures would align with an expected O(logn) behavior.