MP5 Design

Spencer Rawls

I implemented a ready queue as a rotating array by adding the variables readyQueue, readyStart, readyEnd, readyCapacity, and empty to the Scheduler class. ReadyQueue is an array of thread pointers, initially allocated space for 4 thread pointers. ReadyStart stores the index in readyQueue of the next-up thread, and readyEnd stores the index one after the end of the list. ReadyCapacity stores the number of elements that there is space allocated for in readyQueue. By iterating through indices from readyStart until reaching readyEnd, adding 1 to the index and modding it by readyCapacity, you will visit all of queued the threads in order.

When a thread is added to the queue, I first check that there is room in the current queue to add another element. If not, I reallocate a new array twice the size of the old one and copy over the elements from the old one, without the offset, changing readyStart to 0. This is where the empty bool comes in. If readyStart = readyEnd, the queue is either completely empty, or completely full. When a thread is added, I set empty to false, and when a thread is removed making readyStart = readyEnd, I set empty to true. The value of empty is used to tell whether I need to reallocate the array. When that is done, the new thread is placed at the index of readyEnd, and readyEnd is incremented mod readyCapacity.

When a thread is removed from the queue, I increment readyStart mod readyCapacity.

In my yield function, I save a pointer to the thread at readyStart, remove that thread from the queue, and dispatch_to that thread.

In my resume function, I add the given thread to the ready queue.

In my add function, I call the resume function because they do the exact same thing.

My terminate function has 2 steps: search through the ready queue and remove the given thread if it is in the ready queue, and then dispatch the next thread. The first step shouldn’t have to happen because the running thread shouldn’t be in the ready queue, but I implemented it anyway by iterating through the queue in the way I described previously, and if I found the given thread, moved all the previous elements in the array forward one index and incremented readyStart mod readyCapacity. I implemented the second step by simply calling yield.

Other than that, I changed the thread_shutdown function to call terminate on the current thread.

I didn’t attempt either of the bonus point options.