Deeksha Juneja

Cpre308

Lab 8

Lab 8 was about understanding and implementing different scheduling algorithms. It was interesting to see how each one worked.

The first algorithm was the first come first serve and it was given to us and was the base of all the algorithms. It was pretty easy to understand and was implemented like a queue is implemented. Hence, I didn't run into a lot of troubles with that.

After that came in the concept of time Quantas with Round Robin. We were given a step by step implementation of Round Robin. This was a very helpful step in understanding how the code actually works. During this implementation, I went around the folders and tried to figure out where everything really was. Moreover, I also got a closer look at task.h which is the most important component in building further algorithms.

Next was priority round robin. This one was a little tricky as I was trying to implement all the priorities using only one queue and hence it wasn't working well. It was getting stuck in many places. Hence, after talking to Kris, I make a separate queue for each priority. Now I am enqueueing and dequeueing based the priority of the task that came in. Another mistake that I made was that I was setting my priority 0 as the default case and many times that queue was empty. However, that was an easy fix and I detected it using valgrind.

Now, before I talk about SRTN, I will mention that I have done the extra credit because it would be easier to talk about that first. Moreover, it is more closely related to fcfs as well. My algorithm for that is rather basic and is based on the principle of stack. I am using first come last serve or a stack algorithm. It was pretty straight forward and I didn't really run into any problems while implementing it. I rather like the idea of designing a creative algorithm.

Shortest remaining time next is a very interesting algorithm which executes the schedules the task with shortest remaining time left. I started the algorithm with the following way: -

- TaskEnqueue
  - If task->scheduler_data == null then
    - malloc an object and store in this pointer
    - predict the remaining-time as (2*(4 - task->task_info->priority)), store this into task->scheduler_data
  - If the remaining-time of task is <= zero
    - use the last remaining time and the last run time to calculate a new remaining time as explained in the lab
  - insert this task in the linked list sorted by remaining time
  - store the current task->task_info->run_time in the task->scheduler_data
  - return the head of the list / the task with the shortest remaining time
- TaskDequeue
  - Find how much time this this task ran by subtracting task->task_info->run_time from the last run-time in task->scheduler_data
  - subtract that value from remaining time
  - save the last run-time value to task->scheduler_data
  - remove the task from the list

    - store the current task->task_info->run_time in the task->scheduler_data
    - return the head of the list / the task with the shortest remaining time

I don’t have much experience with structs so after implementing my enqueue I was getting a lot of errors regarding the struct that I had made. However, I was able to fix them with the help of my TA Tom. They were mainly regarding incorrect declarations and typecasting.

However, my thinking for Dequeue was not entirely correct. I had a lot of trouble understanding on how to implement it correctly. Even after I did implement it, I didn’t take into consideration the circular nature of the queue. That was the next thing I corrected, checking if the circular queue was empty or not. However, my dequeuer implementation was still flawed due to me removing the wrong task. After I corrected that, I tried to run the program. The make is working. However, the problem arises in when I try to run the srtn.cfg file because I am getting an invalid read of size 8 error. I really doubt I will be able to make more progress on this because I don’t have any clue about this error.

However, I have come pretty far and I am happy about the progress I have made. If I would have had a little more time, then I am pretty sure I would have been able to complete the lab.