CSCE 611 – Operating Systems - MP5

Garuda Suma Pranavi – UIN: 926009146

In this machine problem, I have implemented a simple First In First Out Scheduler (FIFO) and I think I was able to get the get the interrupts working correctly (bonus part 1).

Scheduler():

I've implemented the ready queue using a linked list approach and initialized the head and tail nodes as NULL.

yield():

Since I've tried to add interrupt functionality, I disabled interrupts at the start of the function for the correct handling of the threads. Yield means that the current thread has to give up the CPU and the next thread from the ready queue is loaded, so we pop (actually delete, just named pop) the thread at the head of the ready queue and move the head to the next thread in the ready queue. Finally, we call the dispatcher function to do the context switch and again enable the interrupts.

resume():

Just like yield before, we need to disable interrupts at the start of the function for the correct handling of the threads. Resume is called when threads are waiting for an event to happen or had to give up the CPU in response to a pre-emption, once this thread returns we need to add it to the ready queue. So we create a new node called new_thread and if the queue is empty i.e. head is NULL, we just move the head to new_thread, but if we already have some thread control blocks, we move the tail to the new_thread and the next points to NULL. We enable the interrupts again.

Add():

Making a thread runnable in our implementation simply implies adding it to the ready queue i.e. calling the function resume() that already has this functionality in place.

Terminate():

We first find the given thread which was asked to terminate, if we find it we change the order of the list and delete the thread to be terminated (also remembering to disable and enable the interrupts).

In the thread.C code, we make changes thread_shutdown(), first disabling the interrupts, then calling the terminate function in scheduler.c and then the yield function in scheduler.c. (We also enabled interrupts in thread_start()). I faced a lot of issues in the declaration of SYSTEM_SCHEDULER and was finally able to make it work through the extern keyword, I've initialized it in multiple places, not sure which declaration got the code to work.

I've run the code and not found any errors in the FIFO implementation. Other bonus parts have not been implemented.