CSCE 410
MP5

Kranthi Kumar Katam
225009204

Files changed:
Kernel.C
Scheduler.H
Scheduler.C
Thread.H
Thread.C
simple_timer.C

In this MP i have implemented FIFO Scheduler. Then extended my code to Option 1 and Option 2 i.e., handling interrupts correctly i think and implementing Round robin scheduler i think.

FIFO Scheduler:
First, initialize a new list of ready queue with Thread as the value. Initialize head and tail nodes for the ready queue. Initialize these head and tail to NULL.

Yield() : Whenever a running thread calls yield function it means that this thread is leaving the critical section. So, we have to move the head of the queue to the next thread in the queue and dispatch the running thread and delete the node from the queue.
{For correct handling of threads we have disable interrupts at the beginning of fun and revoke interrupts at the end → Option 1}

Resume() : Create a new node of linked list ready queue i.e., new_thread. Add _thread to this new_thead node and next points to null. Now move the head to new_thread if head is null else move tail to new_thread.
{For correct handling of threads we have disable interrupts at the beginning of fun and revoke interrupts at the end → Option 1}

Add() : Just resume the _thread

Terminate() : This is a tricky part. We have to first change the Thread.C code in thread_shutdown function we have to disable interrupts. Include an extern scheduler called SYSTEM_SCHEDULER and this will call the terminate(current thread).
In the terminate function we can delete the _thread and call the yield.

Option 1 : For correct handling of errors we have to first enable interrupts in the Thread.c. Whenever a new thread is starting in the thread_start enable interrupts first and initiate the thread on port 0x20 and write data 0x20. And in scheduler change enabling and disabling interrupts as mentioned above.

Option 2 : **Round Robin Scheduler** : We have create a new class RRScheduler in Scheduler.H and create a linked list of ready_queue of threads in the RRScheduler. The Yield, Resume, Add and Terminate functions are same as the FIFO scheduler.
The only modifications are :
We have to create a simple time which will tick for every 50ms time quantum. This timer is implemented as an interrupt handler.
In Simple_Timer interrupt handler we have handle thread interrupts. We have to have a extern system scheduler for doing this. For every 50ms timer interrupt will come up and the system scheduler will resume the current thread and listens on port 20. It then calls the yield function.

I think implementation of Round Robin scheduler is correct. If it is not correct then we should change Kernel.C to operate on FIFO scheduler.