CSCE-611: Operating Systems Machine Problem 5

Name: Vamsi Tallam UIN: 432001932

Files modified/added:

1. thread.C
2. thread.H
3. kernel.C
4. simple_timer.C
5. scheduler.C
6. scheduler.H

Attempted **BONUS1** correct handling of interrupts and **BONUS2** Round Robin Scheduler.

The code submitted has the Round Robin Scheduler as default and to test FIFO scheduler do the following steps:

1. Uncomment the SYSTEM_SCHEDULER = new Scheduler() in kernel.C and comment the SYSTEM_SCHEDULER = new RRScheduler(50)
2. Uncomment lines 67-74 in simple_timer.C and comment the following part lines 81-97, the Round Robin scheduler. Lines to be comments are mentioned in the code.

Or you can clone the repo with the commit name Testing "FIFO scheduler testing" from the repository

https://github.tamu.edu/CSCE-410-611-Fall-2022/VamsiTallam_CSCE611/tree/main/mp5

Design Steps:

1. Whenever a thread is created, preempted, resumed, or deleted interrupts are checked and properly handled – **BONUS1** and we can see that threads get properly scheduled based on the interrupt masks set.
2. First-In-First-Out (FIFO) queue is implemented using a linked list of FIFO objects and RR scheduler has a similar functionality with time quantum as an add-on.
3. All updates/modifications to the files are listed above and functions changed are listed below:

The following are the changes made:

1. Thread.H
    a. Contains all include statements, and initializations using the extern keyword.

2. Thread.C
    a. The thread creation and destruction using the shutdown function and thread start functions.

3. Scheduler.H
    a. Data structure to store threads is declared and Round-Robin scheduler is declared.

4. Scheduler.C
    a. Creates the FIFO or Round Robin scheduler object and each thread created is added to the queue. After a specified time quantum, the thread gets preempted and the next thread in the queue gets the CPU.
5. Simple_timer.C
    a. Handle_interrupts Functionality has been tweaked to trigger an interrupt after every time quantum.

6. Kernel.C
    a. Added a line to test the functionality for Round-Robin Scheduler **BONUS2.**
    b. Uncommented uses scheduler and uses terminating functions

Bonus Section:

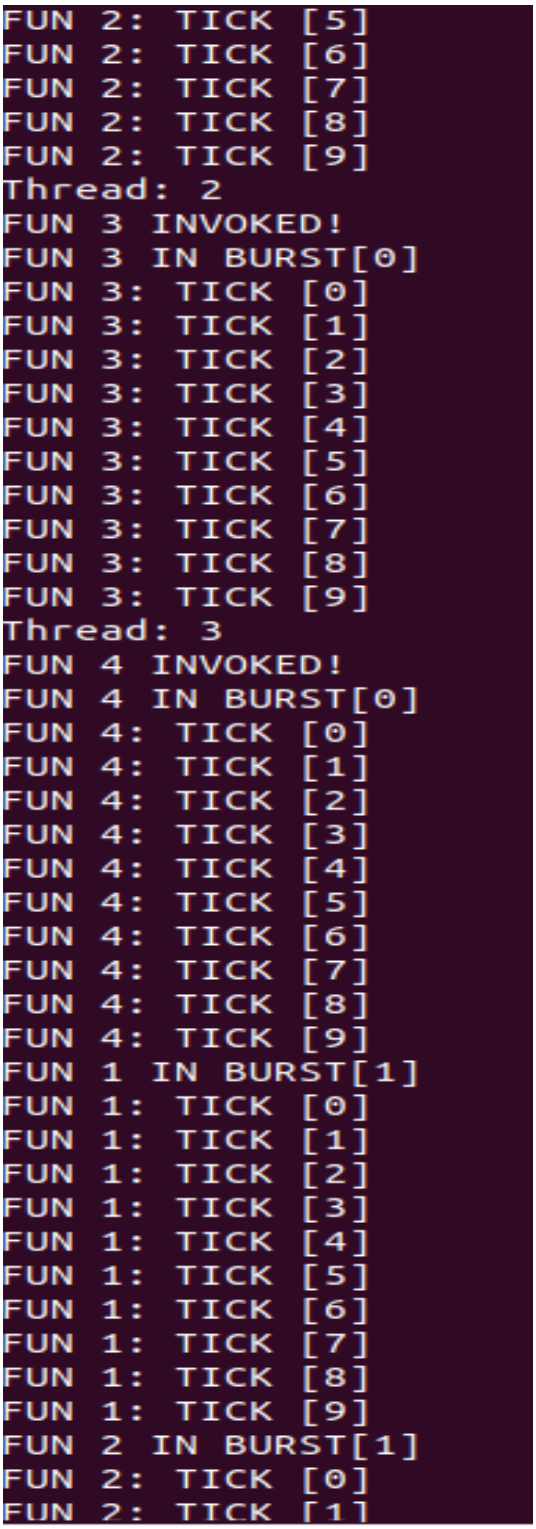
- Correct Handling of Interrupts:
    - The status of interrupts is checked every time a new thread is scheduled, thereby making sure that the interruption is handled correctly
    - Conditional statements are incorporated to handle the interrupts correctly.

- Implemented Round Robin Scheduler:
    - Time quantum of 50ms is checked to assess the correctness of the RR scheduler.
    - The class is derived from the Scheduler and also the simple_timer and interrupts for every elapsed quantum is handled similarly to the one-second elapse as given in the handout by minor modifications to reflect the 50ms quantum.
    - Minor changes are made to handle_interrupts functionality in SimpleTimer.C to check the elapse of 50ms quantum and preempt the current thread and move to the next thread in the queue for RR implementation.

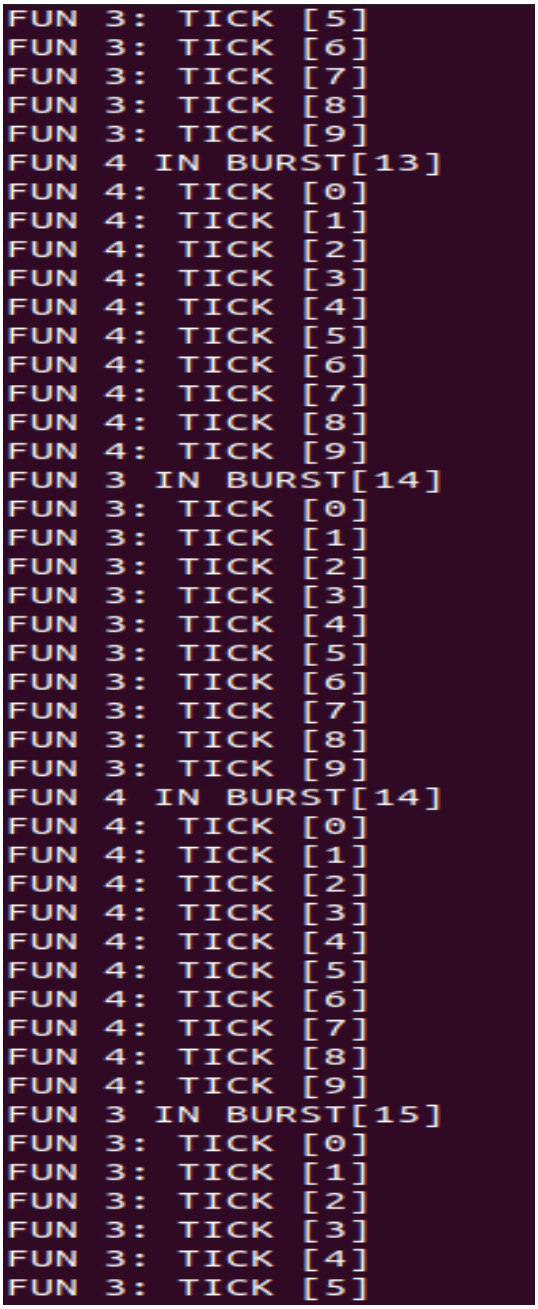
Results:

Initializing the FIFO scheduler and terminating functions:

```
Please choose one: [6] 6
00000000000i[      ] installing x module as the Bochs GUI
00000000000i[      ] using log file bochsout.txt
Installed exception handler at ISR <0>
Allocating Memory Pool... done
Installed interrupt handler at IRQ <0>
Constructeed Scheduler.
Hello World!
CREATING THREAD 1...
esp = <2098108>
done
DONE
CREATING THREAD 2...esp = <2099156>
done
DONE
CREATING THREAD 3...esp = <2100204>
done
DONE
CREATING THREAD 4...esp = <2101252>
done
DONE
STARTING THREAD 1 ...
Thread: 0
FUN 1 INVOKED!
FUN 1 IN BURST[0]
FUN 1: TICK [0]
FUN 1: TICK [1]
FUN 1: TICK [2]
FUN 1: TICK [3]
FUN 1: TICK [4]
FUN 1: TICK [5]
FUN 1: TICK [6]
FUN 1: TICK [7]
FUN 1: TICK [8]
FUN 1: TICK [9]
Thread: 1
FUN 2 INVOKED!
FUN 2 IN BURST[0]
FUN 2: TICK [0]
FUN 2: TICK [1]
FUN 2: TICK [2]
FUN 2: TICK [3]
FUN 2: TICK [4]
```



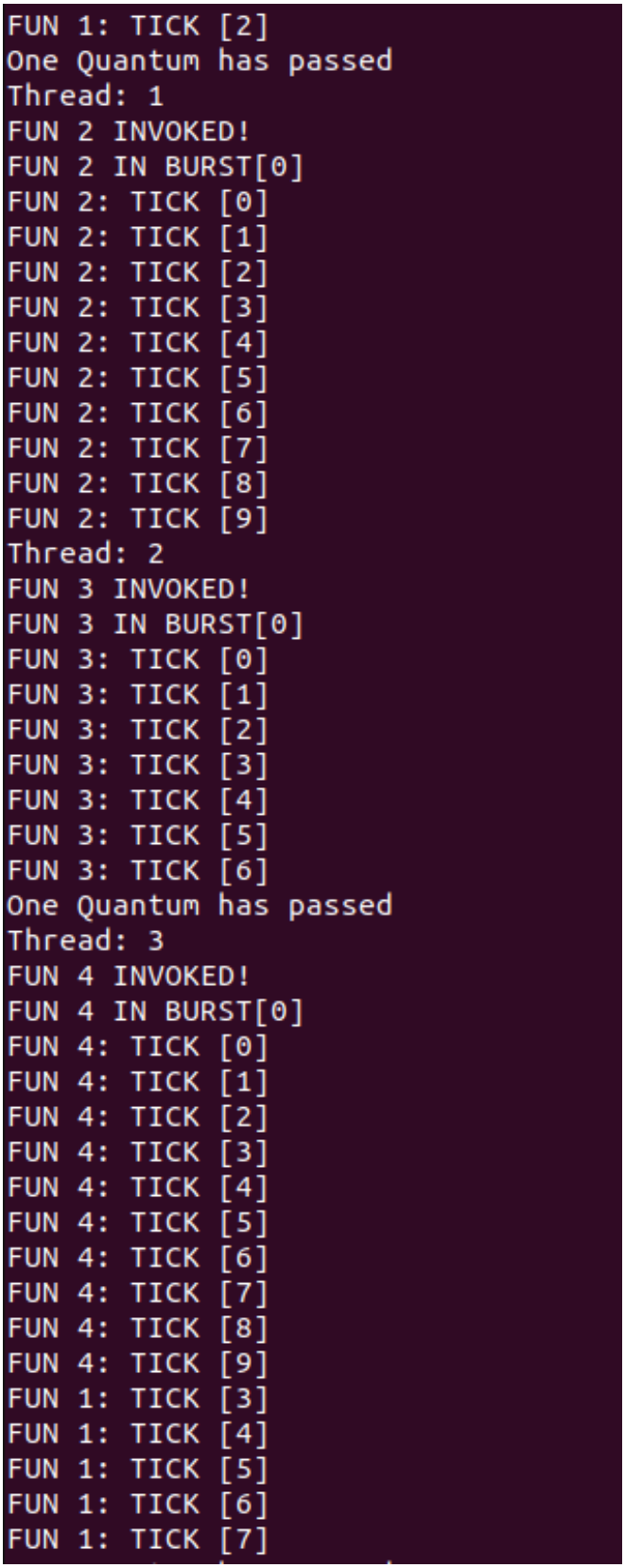
FIFO scheduler in action



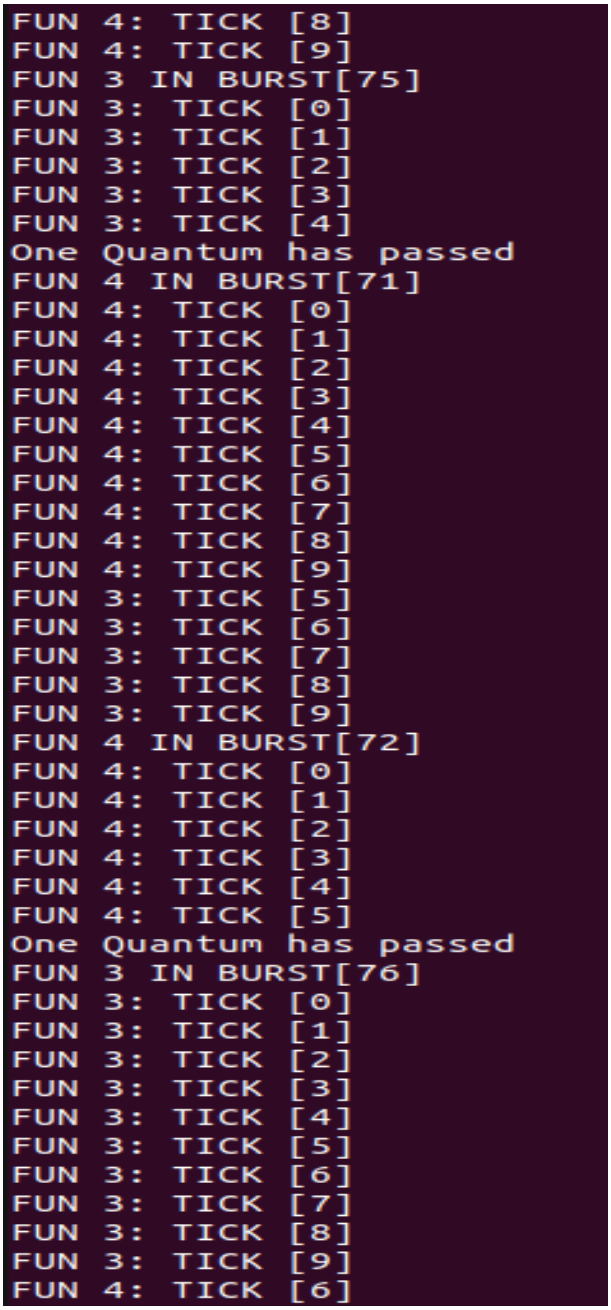
As Fun1 and Fun2 are terminating functions only fun3 and fun4 run indefinitely as they are non-terminating functions.

Initializing Round Robin Scheduler

```
Please choose one: [6]
000000000000i[      ] installing x module as the Bochs GUI
000000000000i[      ] using log file bochsout.txt
Installed exception handler at ISR <0>
Allocating Memory Pool... done
Installed interrupt handler at IRQ <0>
Constructeed Scheduler.
Installed interrupt handler at IRQ <0>
Constructed Round Robin Scheduler
Hello World!
CREATING THREAD 1...
esp = <2098132>
done
DONE
CREATING THREAD 2...esp = <2099180>
done
DONE
CREATING THREAD 3...esp = <2100228>
done
DONE
CREATING THREAD 4...esp = <2101276>
done
DONE
STARTING THREAD 1 ...
Thread: 0
FUN 1 INVOKED!
FUN 1 IN BURST[0]
FUN 1: TICK [0]
FUN 1: TICK [1]
FUN 1: TICK [2]
One Quantum has passed
Thread: 1
FUN 2 INVOKED!
FUN 2 IN BURST[0]
FUN 2: TICK [0]
FUN 2: TICK [1]
FUN 2: TICK [2]
```



RR scheduler in action.



As Fun1 and Fun2 are terminating functions only fun3 and fun4 run indefinitely as they are non-terminating functions. They get preempted after 50ms and the other one gets CPU and after the quantum, the former thread continues from where it stopped.