COMP3511 (2019 Spring) –

Project 2

Modifications and Additional Guidelines

The modifications:

1. In Task 3, **no need** to work out the response time.
   - When calculating the waiting time and turn-around time, you should count the time between the arrival of thread to the ready list And the time as the thread has terminated.
2. The outputs from the readme file does not match with the actual output. You are expected to submit the output based on the input data suggested in the Project2 menu.

The clarifications:

- When the pre-emption and the thread is ended at the same time ticking, the output result would be a weird like the following:-

  Switching from thread "threadC" to thread "threadA"
  threadA, Still 0 to go. Priority 1. Quantum 4. Elapesd ticks: total 8
  threadA, Done with burst. Elapesd ticks: total 8
  Switching from thread "threadA" to thread "threadD"

  This is caused by the checking of preemption by scheduler **comes before** the checking of thread ending. Your scheduler implementation should not be affected by this behavior of the Nachos system. You do not need to change the skeleton code to get a more clear output message.

Points to note:

The following are the common queries from the class, please see the additional guideline below.

1. **In test.3.cc:-**
   a. you need to set the "numThreads" according to the input data
   b. Initialize the scheduler with the input data startTime[], burstTime[], priority[]
   c. Set the number of queues and Policy.
      - Scheduler->SetNumOfQueues(?) Scheduler->SetSchedPolicy(?)

2. **In the scheduler.cc – Scheduler::ReadyToRun( ), in the case of Pre-emptive Priority Policy:-**
   You need to use readyList -> SortedInsert(arg1, arg2) to put new thread in the ready list according to the priority vaue of the thread. Arg1 is the thread, Arg2 is the "sorted key" which indicates the ordering of the thread, (from 0 to 20, with 0 as the first one in the Ready List). This sorted key can be worked out as (20 – the thread's priority value).

3. **In the scheduler.cc – Scheduler::ReadyToRun( ), in the case of MLFQ:-**

Three list are set up in the MLFQ, MultiLevelList[level], where level = 0..2 which corresponds to 0..2 of the priority of the thread, with 2 the highest priority.

One way of implementation:

If (the quantum of the running thread is used up or the thread is just arrived, i.e. <=0)

- Append the thread to the MultiLevelList[level], where level = the thread's current priority.
- Set the thread's quantum according to the level of the list: level=2, set q=4; level=1, set q=8; level=0, FCFS, i.e. set q=thread's remaining burst time

Else // the running thread is pre-empted

- Prepend the thread to the current level of the MultiLevelList[level].

4. **In the scheduler.cc – Scheduler::InterruptHandler( ),**

This method handles time interrupts (invoke in each cpu time tick). Only threads running in RR policy and MLFQ policy is will be affected. (So FCFS and Prempt Priority policy would be invoked by this interrupts.)

- In RR:
    1. Decrement the quantum
    2. If q <= 0,
        - set the thread to q=4;
        - the thread is pre-empted and do the context switching by calling **interrupt->YieldOnReturn;**
- In MFLQ:
    1. Decrement the quantum
    2. If q <= 0, // the running thread's quantum is used up
        - Decrement the thread's priority by 1;
        - the thread is pre-empted and do the context switching by calling **interrupt->YieldOnReturn;**