Assignment 2: OS161 FIFO and Preemptive Priority Scheduler

Version 1.0

Dr. Mosaddek Hossain Kamal Tushar and Dr. Mamun Rashid

Last updated on November 6, 2019

Due Date: November 15, 2019

Contents

1 OS161 Scheduler **1**

2 What to do? **2**

3 Testing **2**

4 Submission **2**

1 OS161 Scheduler

OS161 scheduler by default is a simple round-robin schedule. Function *hardclock()* in *../kern/thread/clock.c* is periodically called by hardware clock interrupt. Thereafter, function hardclock() call three functions (i) schedule(), (ii) thread_consider_migration(), and (iii) thread_yield(). Currently, *schedule()* function is empty therefore round-robin scheduling is default. Function *thread_consider_migration()* is to use migrate thread from one cpu to another cpu and *thread_yield()* used to switching from current thread to the next thread.

In *../kern/thread/thread.c*, the *thread_switch()* is doing the actual job. The first argument of the *thread_switch()* is **new_state** which have given a hints about the scheduling. Thread states are S_RUN, S_READY, S_SLEEP, and S_ZOMBIE are refers to running, ready to run, sleeping, and zombie; exited but not yet deleted. When the value of **new_state** is changed to **S_READY** then current running thread is consume all its time slots and forced to **yield** another thread.

$curcpu \rightarrow c\_runqueue$ is a doubly linked list which contains the collection threads queued to run. In fact $curcpu \rightarrow c\_runqueue$ is the running/ready queue of the operation system. Each of the processor has at least one running queue. The queue is defined in **cpu.h** as **"struct threadlist c_runqueue"**, where "struct threadlist" is defined in *threadlist.h* and all the related functions are implemented in *threadlist.c*.

Running process – the current running process is in the head of the *curcpu → c_runqueue* queue, *thread_yield()* move the currently running process to the tail of the queue, and next process starts running (if the state of the process is S_READY).

2 What to do?

You are asked to implement two scheduling algorithms (i) FIFO and (ii) Preemptive priority scheduling. You will write your most of the code in **schedule()** function and modify other functions and data structures as needed. However, you must record all the changes which should be added to your design document.

For the FIFO and preemptive priority scheduling, you must implement the algorithms or techniques discussed in the class. However, you must consider the aging of the process to alleviate the priority. Again, when a process takes more than 16-time slots, than increase its priority number by '5', which eventually decreases the priority of the process/thread. For the aging, the priority of the process/thread would be improved by decreasing the priority number by one while the process is not running for 64 units of time.

The priority number range is from 1 to 200; beyond this, the OS161 should terminate the process/thread immediately. However, you must take care of the priority while implementing aging and decrease the priority of a process.

Your modification must incorporate functions and necessary data structure to keep some statistics of the thread (i) start time of the first run, (ii) time for resume running state, (ii) time for exiting from running state, (iii) termination time. All the time recoded should be in the form of the time of the day.

3 Testing

A multithreading prime number generator will be used for testing. Each of the threads (generator) usually generates primes between a range of numbers starting from 1000000. The prime number generator should display the statistics (i) total run time (ii) waiting time etc.

4 Submission

Demonstrate your implementation and submit the design document.