Brent Mitchell & Tyler Yoder
EECS 678 – Heechul Yun
Second Project Report
26 April 2015

Project 2 – Round Robin Scheduler
Brent Mitchell & Tyler Yoder

Brent Mitchell & Tyler Yoder
EECS 678 – Heechul Yun
Second Project Report
26 April 2015

Introduction

In this project, we implemented a round robin scheduler in the Linux kernel. This was done by modifying the scheduler classes, finishing the implementation of the round robin scheduling class, adding a system call to set the quantum, and analyzing the new kernel build by using a multithreaded program on a custom build on KVM.

Implementation

The general implementation of the project followed the provided guide. First, sched_other_rr.c was modified at several functions, as described below. Then we modified sched_setscheduler to select our new round robin scheduler, and added a system call for setting the quantum to the three files listed on the guide. Finally, we built and debugged the kernel, and tested it using the threadrunner program.

Functions

Several sched_other_rr.c functions were modified. The modifications are explained briefly below.

enqueue_task_other_rr

This function adds a task to the end of the run queue. First, the time slice is set to the default quantum. Then the task is added to the run queue. Then the count of running tasks is incremented.

dequeue_task_other_rr

This function removes a task from the run queue. First the task is removed from the run queue, then the count of running tasks is decremented.

yield_task_other_rr

This function ends a task's control of the CPU and adds it to the end of the run queue. This is done by using the requeue_task_other_rr function, which has already been defined in the program.

pick_next_task_other_rr

This function selects the next task to run (the head of the run queue). First, it checks if the queue is empty. If it isn't, the next pointer is set to the run queue's head item. Then the clock is started on that item.

task_tick_other_rr

This function decrements the current task's time slice by one time unit (jiffie). First, it checks if the default quantum is 0. If it is, that means the round robin scheduler is set to behave like FCFS, so the function returns automatically. Otherwise, the time slice is decremented by 1. If the time slice is now 0, then the reschedule flag is set and the current task is yielded.

Testing

For each build, we first made any changes to the code that we found necessary. This was mostly done by intuition; since the entire implementation only changed a few things it was easy to find where our

problems were coming from and addressing them without using gdb. Once the changes were made, we built a new build and installed the kernel. Then, we first ran threadrunner on the default settings, for a FCFS behavior. Then, we ran threadrunner with the –quantum flag to set the quantum to some integer value, usually 10. If both threadrunners succeeded without faulting, our implementation was deemed correct. If not, it usually broke with the –quantum flag, so the output (print and printk) was analyzed to see what was malfunctioning. The code was fixed and this process was repeated.

Difficulties in Implementation

There were bugs in our first build. In the function that selects the next task, we were always returning NULL, even if a task was available. That caused a continuous stream of printouts whenever we tried to log into the build. The timer function also had a bug; we originally only checked to see if the time slice equaled zero, then continued if it didn't. We didn't cover the unlikely case that the time slice could pass zero and become less than zero.

Our second build ran threadrunner fine for the default quantum, but segfaulted when the quantum was changed. We tracked the error to our syscall implementation, which was returning void. The syscall was defined as returning a long, so we returned the new quantum. This might have fixed the current segfault, but the build still segfaulted.

During lab, the T.A. showed his implementation of the project, which included transferring the provided config file to the linux directory. We realized that we had not done that, so included it and created our third build. Including the config file, in conjunction with using the –quantum instead of –q flag, allowed for threadrunner to successfully complete using our round robin scheduler implementation.