|  |  |  |  |
| --- | --- | --- | --- |
|  | 3/7/2015 |  | |
|  | |  | |
| Project 2 Report  *Round-Robin Scheduler in Linux* | | | |
|  |  | |  |
|  |  | | Tyler Steiner, Michael Esry |

Project 2 Report

Round-Robin Scheduler in Linux

# Introduction

## Problem at hand

The problem at hand for this project was to write a round-robin scheduler for the linux operating system. Though linux already has a round-robin scheduler available for selection, the purpose of this project is to also familiarize us with programming and testing a linux kernel.

## Proposed Solution

The proposed solution involves making changes to the sched.c file that is already a part of the linux kernel. We will also need to modify the sched\_other\_rr.c file that was given to us for the project. A few other miscellaneous files (syscall\_table\_32.S, unistd\_32.h, and syscalls.h) also had to be slightly modified, however the bulk of the changes were done to sched\_other\_rr.c.

# Implementation

## Discussion

Most of the implementation of this project was done within the sched\_other\_rr.c file. We were supplied all of the files and all of the methods that were needed to complete this project. It was only a matter of adding functional code to the files to complete this project.

We did run into one odd difficulty while implementing. If sched\_other\_rr.c is included in sched.c or sched.h, the kernel will not compile and will instead give odd errors about pointers. Once we removed the include, it all worked fine.

# Testing

## Procedures

Testing of the kernel changes consisted of 2 parts. The first was compilation and installation of the kernel. If there were any syntactical errors, they would show up at compilation time. The built kernel was then installed via command line and the system rebooted.

After reboot, we used the thread\_runner testing program that was provided to test the project. The results of the testing using thread\_runner are below.

## Results

**Command:**  
./thread\_runner -s normal 4 6000k  
**Result:**  
normal (CFS) scheduler selected

thread: 3 writing 1536000 d's

thread: 2 writing 1536000 c's

thread: 1 writing 1536000 b's

thread: 0 writing 1536000 a's

completed 4 threads -- processing shared memory segment

Thread: 0 wrote 1535998 a's

Thread: 1 wrote 1535999 b's

Thread: 2 wrote 1535996 c's

Thread: 3 wrote 1535996 d's

dumping the val\_buf (aggregate=10000):

ddddddddddddddcccccccccccccccccccbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaddddddddd

ddddcccccccccccccbbbbbbbbbbbbbdddddddddddddaaaaaaaaaaaaacccccccccccccbbbbbbbbbbb

bbdddddddddddddaaaaaaaaaaaaacccccccccccccbbbbbbbbbbbbbdddddddddddddaaaaaaaaaaaaa

ccccccccccccbbbbbbbbbbbbdddddddddddddaaaaaaaaaaaaacccccccccccccbbbbbbbbbbbbbdddd

dddddddddaaaaaaaaaaacccccccccccccbbbbbbbbbbbbbaaaaaaaaaaaaadddddddddddddcccccccc

cccccbbbbbbbbbbbbbaaaaaaaaaaaaadddddddddddddcccccccccccccbbbbbbbbbbbaaaaaaaaaaaa

adddddddddddddcccccccccccccbbbbbbbbbbbbbaaaaaaaaaaaaadddddddddddddcccccccccccccb

bbbbbbbbbbbbaaaaaaaaaaaaadddddddddddccccccccbbbbbbbbbbaaaaaaaaaa

**Command:**  
./thread\_runner -s other\_rr --quantum 2 4 6000k  
**Result:**

other\_rr scheduler selected, quantum=2

thread: 0 writing 1536000 a's

thread: 1 writing 1536000 b's

thread: 2 writing 1536000 c's

thread: 3 writing 1536000 d's

completed 4 threads -- processing shared memory segment

Thread: 0 wrote 1535999 a's

Thread: 1 wrote 1535997 b's

Thread: 2 wrote 1535999 c's

Thread: 3 wrote 1535997 d's

dumping the val\_buf (aggregate=10000):

aaaaaaaaaaaaabbbbbbbbbbbbbbbbbbccccccccccccccccccddddddddddddddddddaaaaaaaaaaaaa

aaaaaaabbbbbbbbbbbbbbbbbbbbccccccccccccccccccccdddddddddddddddddddaaaaaaaaaaaaaa

aaaaaabbbbbbbbbbbbbbbbbbbbccccccccccccccccccccdddddddddddddddddddaaaaaaaaaaaaaaa

aaaaabbbbbbbbbbbbbbbbbbbbccccccccccccccccccccdddddddddddddddddddaaaaaaaaaaaaaaaa

aaaabbbbbbbbbbbbbbbbbbbbccccccccccccccccccccdddddddddddddddddddaaaaaaaaaaaaaaaaa

aaabbbbbbbbbbbbbbbbbbbccccccccccccccccccccdddddddddddddddddddaaaaaaaaaaaaaaaaaaa

abbbbbbbbbbbbbbbbbbbbccccccccccccccccccccdddddddddddddddddddaaaaaaaaaaaaaaaaaaaa

bbbbbbbbbbbbbbbbbbbccccccccccccccccccccdddddddddddddddddddaaaaaaaabbbccdddd

**Command:**  
./thread\_runner –s other\_rr –quantum 0 4 6000k  
**Result:**

other\_rr scheduler selected, quantum=0 (FCFS policy)

thread: 0 writing 1536000 a's

thread: 1 writing 1536000 b's

thread: 2 writing 1536000 c's

thread: 3 writing 1536000 d's

completed 4 threads -- processing shared memory segment

Thread: 0 wrote 1536000 a's

Thread: 1 wrote 1536000 b's

Thread: 2 wrote 1536000 c's

Thread: 3 wrote 1536000 d's

dumping the val\_buf (aggregate=10000):

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbb

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbcccccccccccc

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccdddddddddddddddddd

dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd

dddddddddddddddddddddddddddddddddddddddddddddddddddddddd

# Concluding Remarks

Based off of the tests done, the Round-Robin scheduler we implemented worked as requested. If the time quantum was set to 0, it functioned as a FCFS policy. If the time quantum was set to anything else, it functioned as a Round-Robin scheduler with the desired quantum.

Perhaps an interesting and difficult part of this project was the compiling. With most, if not all other projects done in classes, our programs have been small enough that compiling took a matter of seconds. This small time allowed for minute changes to be done and checked with high frequency and at little cost of time. However, compiling and building this kernel takes a lot of time, even if most of the files were not changed and so did not need to be recompiled. Therefore, it was best to make as many and as precise of changes as possible before recompiling and building the program.