|  |  |
| --- | --- |
| Operating Systems  COP4600-001 | |
| Date: | January 30, 2017 |
| Name: | Tyler Simoni  Andrew Rodiek |
| UNumber: | U25068858  U84332681 |
| Approx Hours Spent: | 8 hours |
| Difficulty: | Moderate |

# Project Objectives

The project objectives were to write system\_call.c, which is to measure the time it takes to do a system call, and context\_switch.c, which is to measure the time it takes to do a context switch. Also to be coded, was a makefile for easy compilation. The code was written in C.

# Project Code

## system\_call.c

## For system\_call.c, we chose to make our system call a for loop calculating the number of multiples of 5 up to 100,000. This for loop is nested inside another for loop in order to get an average time measurement between 100,000 iterations. Due to this, there are a total of 10 billion iterations.

## The time is measured using the time.h library. Two clock\_t variables were declared as t1 and t2 as start and stop, respectively. The timers (t1 = clock(); and t2+=clock() -t1;) were placed around the inside for loop. This allowed the clock time to be taken at the beginning of each loop and then once the outside loop completed, the resultant t2 value was divided by MAX to get the average. This was then divided by CLOCKS\_PER\_SEC to get the time in milliseconds.

The output of the code is below:



## context\_switch.c

For context\_switch.c, the code starts out by creating a group of processors to use and setting the group to empty. We then select the CPU and set the schedule affinity for that single CPU. A read/write pipe is then declared and initialized, a child process ID and our clock\_t variables were declared.

There is then a simultaneous creation of a fork and error check to make sure the fork was created properly. The first clock\_t variable was then initialized to clock(). A for loop loops through the read/writes of the child and parent, measuring the time in between each of them. At the end, the code will output the average time between the context switches.

The output is below:



# Conclusions

Both pieces of code proved to be a challenge to code. We made an error while coding system\_call.c, in that were doing “num1 % i” instead of the other way around, which created an error in the output of the multiples. This also caused an error in the count. Context\_switch.c took a lot of research and trial and error in order to get compiled and working correctly. Initially, we did not know where to set the timers to correctly grab the time between switches. In the end, we believe to have correctly grabbed it.