Lab 1 Due 2/14/2020

CS 4375 OS

A measurement

- We will be discussing context swapping in the next few lectures. The OS does this for us, behind the scenes. We want to measure the cost (i.e., time) it takes for the OS to do this.
- We will run an experiment to do this.
- This requires writing some code.

What you will deliver:

- A paper (yes, you get to write some papers in this class).
  - Introduction (which lays out the goals of the experiment and a brief bit of background)
  - Methods (which describes what you did to measure the context swap)
  - Results (which show the results of your experiment)
  - Discussion (where you explain what the results mean and discuss possible explanations, including why your measurements might not be accurate)
- Source code and make file
  - I expect to see well-written code with comment blocks at the top of each file and procedure
  - The longer it takes me to read it, the lower your grade

Stuff you need to know:

- Linux.
- The programming language C and make.
- File operations open(), close(), read(), write(), flush(), pipe().
- System time utility gettimeofday().
- Process creation fork()
- schedsetaffinity()

My recommend

- Start small and work your way to the final
- Here are six steps

1.A Getting started with Linux

- Get a linux system running. Either use the CS3432 Virtual Machine, down load a virtual machine that I have available, or install Linux on a PC. Do not delay in starting this process. It isn’t difficult, but there are many ways it can go wrong.

1.B Getting started with C

- Read the OSTEP tutorial on separate compilation and make: http://pages.cs.wisc.edu/~remzi/OSTEP/lab-tutorial.pdf
- Write a simple program that reads from stdin (see gets()) and writes to stdout. Create a make file that compiles this program. Break your program into two functions, and put the functions in separate files. A common naming convention is to name the file the same as the function (e.g., a read_input() function would be in the file named “read_input.c” and have a header “read_input.h”).

1.C Fun with fork()

- Write a program that uses the fork() method to split one process into two. You can confirm there are two processes if you have each of the processes sleep() long enough for you to run the ps command. (You may need to use the “-e” option. See the ps man page.)
-

1.D pipe()

- Extend the previous program with a pipe. (See the pipe() man page.) In general, you will create an array of file descriptors, call the pipe() command, then fork. The basic outline of the code is

```
int main (int *argv[], int argc)
 {  int my_pipe[2];
    pipe(my_pipe);
    p = fork();
    if (p>0)   // parent
     { close (my_pipe[0];          // close the read side. parent only writes
        write(my_pipe[1], “some string from parent”, 24);
         fflush(my_pipe[1]);
         close(my_pipe[1]);
          wait();
       }
       else   // child process
       { close (my_pipe[1]);
          char input_str[100]);
          read(my_pipe[0], input_str, 100);
          close(my_pipe[0]);
          exit (0);
       }
   }
```

1.E First experiment

- Write a program that calls gettimeofday() or clock() twice with no intervening code. Subtract these two (and, in the case of clock(), convert to seconds) to get an estimate of the precision of the timing facility. Repeat this experiment enough times that you have a good estimate (how will you know you have enough measurements).

1.F Actual context swap measurement

- Two processes, two pipes:
  - A read from pipe 0 and writes to pipe 1
  - B read from pipe 1 and writes to pipe 0
- B will read then write
- A will write then read
- Measure the time it takes to read
- Part of the lab is for you to explain why this is measuring a context swap.

1.F Actual context swap measurement

- At least one problem: what if you have multiple CPUs?
- You should ensure that your context-switching processes are located on the same processor. Check out the Linux schedsetaffinity() call.
-

1.F Actual context swap measurement

- Science: it is likely that if you run the experiment more than once, you'll get different measurements.
  - Run it many times
  - Collect data for each experiment
  - Analyze the data (statistics)
  - Explain the variance