Alexander Lerma

CS 450

Due: March 3, 2017

Programming Assignment 2

Build xv6:

- make clean; make; make qemu-nox

New Files:

- callcount.c
    - Runs test program
    - Start up xv6
    - just enter callcount

Files changed:

- syscall.h
    - #define SYS_callcount 22
    - #define NUM_CALLS  22
- syscall.c
    - extern int sys_callcount(void);
        - defined in sys_callcount(void) from sysproc.c
    - added [SYS_callcount] sys_callcount to syscalls function array
    - wrapped up valid syscall check into valid_syscall(int num);
    - proc->callcount[num] += 1; increment a specific syscall count for a process
    - proc->callcount[0] += 1; increment the total syscalls for a process
- sysproc.c
    - int sys_callcount(void); system call for callcount defined in proc.c;
    - prints my name and student id
    - checks for callcount argument
    - returns callcount for a syscall for a process
- proc.h
    - int callcount[NUM_CALLS + 1]; callcount[0] is total counts, holds total counts for each call for a process
- proc.c
    - int callcount(int num); get the number of calls for a sys call
    - allocproc(); added memset(p->callcount, 0, NUM_CALLS + 1); for callcount initialization
- usys.S
    - SYSCALL(callcount) added user can reference my callcount systemcall

About lab:

Adding a System call:

Overall this lab allowed me to explore xv6 in depth. I learned how other system calls, such as kill or sleep, are exposed to the user and how they are passed to the kernel. The code was surprisingly simple to understand and I was happy to find that adding a new system call doesn't require all that much work.

Vagrant for Linux Compiling / MacOS Compiling:

I decided to use vagrant to set up a linux vm. Vagrant is super easy to set up and has so many boxes available to use on their site. Will definitely be using this a lot for various projects.
I use macos and managed to find some nice homebrew formulas that let me compile xv6 without issue natively. Got to skip the tedious setup.