Project 1: xv6 Intro

We'll be doing kernel hacking projects in **xv6.** Xv6 is a port of a classic version of unix to a modern processor, Intel's x86. It is a clean and beautiful little kernel, and thus a perfect object for our study and usage.

This first project is just a warmup, and thus relatively light on work. The goal of the project is simple: to add a system call to xv6. Your system call, **getprocsinfo()** , simply returns the PID value and process name for each of the processes that exists in the system at the time of the call.

Details

The function prototype for your syscall should be

int getprocsinfo(struct procinfo*)

struct procinfo is a structure with two data members, an integer **pid** and a string **pname**. The syscall should write the data into the passed argument for each existing process at the time of the call. The syscall should return the number of existing processes, or -1 if there is some error.

The syscall should be added with the next available syscall index. This *should* be 22. If it is not, you should check your xv6 version.

Your system call returns the PID value and process name for each of the processes that exists in the system at the time of the call.

The OS kernel underlies every user application. The kernel developer/maintainer should avoid making modifications that add significant overhead (space/runtime) to existing kernel facilities. Please ensure that you avoid adding overhead to main kernel operations and the kernel size in order to implement your syscall.

Tips

Find some other system call, like **getpid()** or any other simple call. Basically, copy it in all the ways you think are needed. Then modify it to do what you need.

Most of the time will be spent on understanding the code. There shouldn't be a whole lot of code added.

Using gdb (the debugger) may be helpful in understanding code, doing code traces, and is helpful for later projects too. Get familiar with this fine tool!

The Code and Environment

The source code for xv6 (and associated README) can be found in https://pdos.csail.mit.edu/6.828/2017/xv6.html.

You will need to QEMU emulator and a compiler toolchain. If you use the linux machine in the lab, the compiler toolchain should be there already. I am talking with system adminitrators to install QEMU for you. Otherwise, you can install QEMU under your own directory.

Details on how to use xv6 can be found on the references.

Submission

Submit your **getprocsinfo(struct procinfo*)** to canvas before due time. Create test cases and test your **getprocsinfo(struct procinfo*)** before submission. There is an existing test executable (**usertests**) included with xv6. Your tests should follow the general format of **usertests** to create a test executable **testgetprocsinfo**. The grader should be able to build your kernel, launch qemu, and run **testgetprocsinfo** from the command line in the same way as **usertests**. **testgetprocsinfo** should demonstrate that your syscall works. Your modifications to the kernel source should **not** break tests within **usertests**. Please submit the full OS source as a single archive (tar or zip). Please include a file readme-<username> with your source that details what changes you made and argues that your modifications have minimal impact on kernel overhead.