CS 537 Spring 2019, Project 1b: Intro To Kernel Hacking

To develop a better sense of how an operating system works, you will also do a few projects *inside* a real OS kernel. The kernel we'll be using is a port of the original Unix (version 6), and is runnable on modern x86 processors. It was developed at MIT and is a small and relatively understandable OS and thus an excellent focus for simple projects.

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, **getopenedcount()**, simply returns how many times that the **open()** system call has been called by user processes since the time that the kernel was booted.

Background

If you haven't watched the discussion video (https://www.youtube.com/watch?v=vR6z2QGcoo8), you might want to read this background section (https://github.com/remzi-arpacidusseau/ostep-projects/blob/master/initial-xv6/background.md).

More information about xv6, including a very useful book written by the MIT folks who built xv6, is available here (https://pdos.csail.mit.edu/6.828/2017/xv6.html). Do note, however, that we use a slightly older version of xv6 (for various pedagogical reasons), and thus the book may not match our code base exactly.

Setting Up XV6

You may extract the xv6 source using of the CSL machines. To do so follow the steps below:

- Log into a CSL machine
- Navigate the directory where you will develop your system call
- Copy the xv6 source code to your working directory:

```
> cp -r /p/course/cs537-shivaram/xv6-sp19 .
```

- Build the source

```
> make
```

- Run xv6 using qemu

```
> make qemu
```

Your System Call

Your new system call should look have the following return codes and parameters:

```
int getopenedcount(void)
```

Your system call returns the value of a counter (perhaps called **openedcount** or something like that) which is incremented every time any process calls the **open()** system call. That's it!

Tips

Watch this discussion video (https://www.youtube.com/watch?v=vR6z2QGcoo8) – it contains a detailed walk-through of all the things you need to know to unpack xv6, build it, and modify it to make this project successful.

One good way to start hacking inside a large code base is to find something similar to what you want to do and to carefully copy/modify that. Here, you should find some other system call, like **getpid()** (or any other simple call). Copy it in all the ways you think are needed, and 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!

Detecting Potentially Unclosed Files (Not Graded)

By applying the same technique used to create **getopenedcount()**, create a second systemcall **getclosedcount()** which supplies the number of times which the **close()** system call has been called by user processes.

Comparing the values of both **getopenedcount()** and **getclosedcount()** allows the system a way to estimate if user processes have closed the files which have been opened. However, these counts alone are not enough for a system or individual user processes to know if all of the opened files have been closed. What additional information might be useful to determine this?

Due Date

This project is due on February, 8th, 2019 at 11:59PM.

Submitting Your Solution

To submit your solution, copy all of the xv6 files and directories with your changes into `~cs537-1/handin/<cs-login>/p1b/` . One way to do this is to navigate to your solution's working directory and execute the following command:

```
cp -r . ~cs537-1/handin/<cs-login>/p1b/
```

When executing `ls -l` in the `~cs537-1/handin/<cs-login>/p1b/` directory the contents should be similar to what is shown below:

```
$ ls -l
total 28
-rw-r----- 1 <user-name> <user-name>  223 Feb 27  2012 FILES
drwxr-x--- 2 <user-name> <user-name> 2048 Jan 24 16:04 include/
drwxr-x--- 2 <user-name> <user-name> 6144 Jan 28 12:01 kernel/
-rw------- 1 <user-name> <user-name> 4823 Jan 23 14:51 Makefile
-rw------- 1 <user-name> <user-name> 4809 Feb 27  2012 Makefile~
-rw-r----- 1 <user-name> <user-name> 1793 Feb 27  2012 README
drwxr-x--- 2 <user-name> <user-name> 2048 Jan 28 12:01 tools/
drwxr-x--- 2 <user-name> <user-name> 4096 Jan 28 12:01 user/
-rw-r----- 1 <user-name> <user-name>   22 Feb 27  2012 version
```