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, **getreadcount()**, simply returns how many times that the **read()** 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, you might want to read this background section.

More information about xv6, including a very useful book written by the MIT folks who built xv6, is available here. 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.

Your S

The tests assume that xv6 source code is found in the `src/` subdirectory. If it's not there, the script will complain.

The test script does a one-time clean build of your xv6 source code using a newly generated makefile called `Makefile.test`. You can use this when debugging (assuming you ever make mistakes, that is), e.g.:

```
prompt> cd src/
prompt> make -f Makefile.test qemu-nox
```

You can suppress the repeated building of xv6 in the tests with the `-s` flag. This should make repeated testing faster:

```
prompt> ./test-getreadcounts.sh -s
```

The other usual testing flags are also available. See the testing README for details.