CS 3100 Lab 3: Create A System Call
Add a New System Call to XV6

In Lab3, you will download and install onto your Linux system an instance of XV6, a very small but complete operating system written in C. You will then add a new system call `getsyscallinfo()`, which returns the value of a global kernel integer named `callCount`, which is incremented each time a system call is executed.

Learning Objectives

- Install and configure git, QEMU and XV6
- Run XV6 under QEMU and execute shell commands to verify your installation
- Modify XV6 to add a new system call `getsyscallinfo()` and add a global integer to the kernel named `callCount`, initialized to zero and incremented each time a system call is made.
- Run provided program `peep` that calls `getsyscallinfo()` a few times interspersed with other system calls, and prints out either Success or a diagnostic message to aid in debugging.

Assignment

Begin by installing the tools you will need for this assignment:

```
mkdir -p ~/cs3100/lab3
cd ~/cs3100/lab3
sudo apt-get update # refresh the apt-get database
sudo apt-get install git
sudo apt-get install qemu
git clone git://github.com/WeberState/xv6-public.git
scp USER@icarus.cs.weber.edu:/var/classes/cs3100/lab3/Makefile .
make get            # retrieve the files we will be editing (once)
make                # this will build xv6 and launch it under qemu
cat README.TED      # displays a XV6 README.TED file
CTRL-A X            # press and release CTRL-A, then type X
                    #   which will terminate QEMU and XV6
```

The next step is to modify the source files needed to implement the new system call. These are the files:

```
syscall.c syscall.h sysproc.c trap.c user.h usys.S peep.c
```

You are to implement the following:
- a new system call called `getsyscallinfo()` which retrieves the current number of times any system call has been issued since boot (hint: `sysproc.c`, `usys.h` and `user.h`).
- a global kernel integer named `callCount` (`int callCount = 0;`) that is incremented each time any system call is issued (hint: `trap.c`)
- a new entry at the end of the system call table representing `getsyscallinfo()` (hint: `syscall.c` and `syscall.h`)

Your workflow is to retrieve these source files from `xv6-public` so you can edit them, then copy them back into `xv6-public` for compilation. The `Makefile` you retrieved from `icarus` has these commands that facilitate your workflow:

`make` `(or make install):` copies your copy of the source files from `~/cs3100/lab3`, builds the XV6 kernel and launches `qemu` to load and run `xv6`. You will receive a `$` prompt and you can enter a very limited numer of shell commands, including `peep`. Type `CTRL-A X` to terminate `xv6` and `qemu`.

`make get:` copies the specific source files you will need from `~/cs3100/lab3/xv6-public` to `~/cs3100/lab3` so you can edit them outside of the `xv6-public` folder. This copy is non-destructive, meaning that no files will be overwritten if you have already executed this command before.

A note about file `peep.c`: this is a program written by your instructor to test the implementation of your new system call. However, as delivered, all of the important code is commented out. This was done so you wouldn't get compilation errors when building your `xv6` kernel. When you have updated the other source files with the code needed for this lab, remove the `/* */` from `peep.c` and `make install`. When `xv6` displays the `$` prompt, type `peep 20` (or some other number greater than 1) and see what `peep` reports. If you see the word `"Success"`, your new system call should be working properly. If it reports a failure, you will receive additional information. If you receive nothing at all, you forgot to remove the comments. We reserve the right to enhance `peep.c` throughout the semester, even for this assignment. If we do, instructions will be provided.

Example output

Here is an example of how your `xv6` should behave when compiled and run:

```
tcowan@tcowan-Ubuntu:~/cs3100/lab3$ make
```

I typed "make"

```
cp syscall.c syscall.h sysproc.c trap.c user.h usys.S peep.c xv6-
public/
cd xv6-public; make; make qemu-nox
```

Another makefile is run

```
make[1]: Entering directory `/home/tcowan/cs3100/lab3/xv6-public'
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -fvar-tracking -
fvar-tracking-assignments -O0 -g -Wall -MD -gdwarf-2 -m32 -Werror -fno-
omit-frame-pointer -fno-stack-protector   -c -o syscall.o syscall.c
          .
          .
          .
          .
```

A bunch of compilation steps take place

```
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
```

The xv6 file system image is created

```
5120000 bytes (5.1 MB) copied, 0.053437 s, 95.8 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000103275 s, 5.0 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
353+1 records in
353+1 records out
180820 bytes (181 kB) copied, 0.00047917 s, 377 MB/s
make[1]: Leaving directory `/home/tcowan/cs3100/lab3/xv6-public'
make[1]: Entering directory `/home/tcowan/cs3100/lab3/xv6-public'
qemu-system-i386 -nographic -hdb fs.img xv6.img -smp 2 -m 512
xv6...
cpu1: starting
```

qemu loads and runs xv6

```
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32
bmap start 58
init: starting sh
```

$ is the xv6 shell prompt

```
$
$
$ peep 20
```

peep was called but it still is totally commented out

```
$
$ QEMU: Terminated
```

CTRL-A X was typed

```
make[1]: Leaving directory `/home/tcowan/cs3100/lab3/xv6-public'
tcowan@tcowan-Ubuntu:~/cs3100/lab3$
```

Hints and Suggestions

- Search xv6 for a simple system call and see how it is implemented in the files mentioned above. Suggestion: `getpid()`.
- qemu is a complex virtualization software product with a ton of command-line parameters and configuration options. Use `make` to run it until you know more about what you are doing. See `man qemu`.
- xv6 has documentation! A PDF file is attached to the module containing this assignment in Canvas.

Grading

I use Cucumber scripts to help grade programs. Please note the following:

- The cucumber tests will fail if you do not follow the naming instructions, so please be sure that the programs are named correctly. As long as the source file is named correctly, and you have to upload it to Canvas more than once, the displayed name may have a number after it. That is just for display purposes and your file should still be named correctly internally.
- Upload the files you modified for this assignment to Canvas when you are ready for me to grade your work. You should upload only the following, using the Firefox web browser in ubuntu:
  ```
  syscall.c syscall.h sysproc.c trap.c user.h usys.S
  ```
- It is not necessary to upload your files to icarus for this assignment.

NOTE: Just because cucumber reports a certain number of points for your submission of the assignment, the instructor reserves the right to inspect your code, run your submission with additional tools and manually grade your assignment. The decision of the instructor is final. Test your code thoroughly to ensure all requirements are fully met.

Here is how you earn points for this assignment:

FEATURES	POINTS
Must-Have Features	
Files are named correctly and only the following files are uploaded to Canvas: `syscall.c syscall.h sysproc.c trap.c user.h usys.S`	50
xv6 compiles and builds a usable kernel and file system without errors or warnings with your implementation of system call `getsyscallinfo()`, which uses a kernel integer variable named `callCount`.	50
xv6 runs with system call `getsyscallinfo()` under qemu, displays the `$` command prompt and accepts and processes shell commands	50
`peep` compiles and runs under xv6 with calls to `getsyscallinfo()`, and each time it is passed an integer > 1, displays "Success"	50
Grand Total	200