CS 550 Operating Systems, Spring 2023
Programming Project 1 (PROJ1)

Out: 2/5/2023, SUN
Due date: 2/25/2023, SAT 23:59:59

In this project, you will implement shutdown functionality for xv6 and answer the questions related to system call implementation in xv6. The goal of this project is to learn how system calls are implemented in modern OSes and how system calls are invoked from a user program in xv6.

1 GitHub account username

All the projects in this semesters will be administered collectively using GitHub classroom and Brightspace. By default, our grading script assumes that your GitHub username is the same as your BU username (if your BU email address is "`abc@binghamton.edu`", "`abc`" is your BU username). If your GitHub username is different from your BU username, please inform us by completing the following form (so that our grading script can be updated accordingly). If your GitHub username is not submitted before grading starts, 5 points will be deducted from your final score of this project.

Form link: `https://forms.gle/nCYuwiQMPYJQnWn47`

2 Baseline source code

For this and all the later projects, you will be working on the baseline code that needs to be cloned/downloaded from your own private GitHub repository. Please make sure you read this whole section, as well as the grading guidelines (Section 7), before going to the following link at the end of this section.

- Go to the link at the end of this section to accept the assignment.
- **Work on and commit your code to the default branch of your repository. Do not create a new branch. Failure to do so will lead to problems with the grading script and 5 points off of your project grade.**

Assignment link: `https://classroom.github.com/a/Ea9FY9CW`

3 Build and run xv6

The following provides instructions on how to build and run xv6 using either a CS machine or your own computer.

(1) Log into a CS machine (i.e., a local machine or a remote cluster machine).

(2) Clone or download the baseline xv6 code.

(3) Compile and run xv6:

```
$ make qemu-nox
```

After the compiling the kernel source and generating the root file system, the Makefile will start a QEMU VM to run the xv6 kernel image just compiled (you can read the Makefile for more details). Then you will see the following output (disregard the warning message(s)), indicating you have successfully compiled and run the xv6 system.

```
xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$
```

Troubleshooting: if you see the following error message

```
make: execvp: ./sign.pl: Permission denied
```

Solve it by running the following command:

```
chmod ugo+x ./sign.pl
```

Then

```
make clean
make qemu-nox
```

4 Shutdown commands and system calls (60 points)

The original xv6 system doesn't have a "`shutdown`" command to properly turn off the machine. In this part, you are going to add two "shutdown" commands, which essentially do the same thing - to shut down the machine.

4.1 `shutdown` command and the system call

The first command you need complete is named "shutdown", which simply shuts down the machine. In the baseline code, the file "shutdown.c" provides the implementation of this command. What you need to do is to complete the corresponding system call and its user space wrapper routine.

4.2 `shutdown2` command and the system call

The first command you need complete is named "shutdown2", which prints a shutdown message provided by the user **within the system call implementation** before shutting down the machine. In the baseline code, the file "shutdown2.c" provides the implementation of this command. What you need to do is to complete the corresponding system call and its user space wrapper routine.

4.3 Requirements and hints

- You implementation should not change the code in shutdown.c and shutdown2.c other than commenting out the "`STUB_FUNCS`" macro definition to remove the stub function. **Failure to follow this requirement will lead to zero point of this part**.
- To shut down the xv6 virtual machine in your system call, you only need two lines of code:

```
outw(0xB004, 0x0|0x2000);
outw(0x604, 0x0|0x2000);
```

- Reading and understanding how the existing user commands and the associated system calls are implemented will be helpful. For example, you can look at how the `cat` user command is implemented. The `cat` user command is implemented in `cat.c`, in which the system call `open()` is called. The actual work of the `open()` system call is done in the `sys_open()` function defined in sysfile.c.
- Understanding the mechanism is important. Pay attention to all the related files, which include assembly files (relax, the related assembly file is very easy to read). You may be requested to explain how things work in xv6 in exams.

5 xv6 system call implementation Q&A (40 points)

Answer the following questions about system call implementation.

(1) (10 points) What is the name of the (user space) wrapper function of the system call for the `shutdown2` command (2 points)? Where is this wrapper function invoked (3 points)? Where is this function defined (5 points)?

(2) (10 points) Explain (with the actual code) how the above wrapper function triggers the system call.

(3) (10 points) Explain (with the actual code) how the OS kernel locates a system call and calls it (i.e., the kernel-level operations of calling a system call).

(4) (10 points) How are arguments of a system call passed from user space to OS kernel?

Key in your answers to the above questions with any the editor you prefer, export them in a PDF file named "xv6-syscall-mechanisms.pdf", and submit the file to the assignment link in Brightspace.

6 Submit your work

Once your code in your GitHub private repository is ready for grading, submit a text file named "DONE" (and the the previous "xv6-syscall-mechanisms.pdf") to the assignment link in Brightspace.

If you have referred to any form of online materials or resources when completing this project (code and Q&A), please state all the references in this "DONE" file. Failure to do so, once dectected, will lead to zero point for the entire project, and possibly further penalty depending on the severity of the violation.

Suggestion: Test your code thoroughly on a CS machine before submitting.

7 Grading

The following are the general grading guidelines for this and all future projects.

(1) **The code in your repository will not be graded until a "DONE" file is submitted to Brightspace**.

(2) The submission time of the "DONE" file shown on the Brightspace system will be used to determine if your submission is on time or to calculate the number of late days. Late penalty is 10% of the points scored for each of the first two days late, and 20% for each of the days thereafter.

(3) If you are to compile and run the xv6 system on the department's remote cluster, remember to use baseline xv6 source code provided by our GitHub classroom. Compiling and running xv6 source code downloaded elsewhere can cause 100% CPU utilization on QEMU.

Removing the patch code from the baseline code will also cause the same problem. So make sure you understand the code before deleting them.

If you are reported by the system administrator to be running QEMU with 100% CPU utilization on QEMU, 10 points off.

(4) If the submitted patch cannot successfully patched to the baseline source code, or the patched code does not compile:

```
1   TA will try to fix the problem (for no more than 3 minutes);
2   if (problem solved)
3     1%-10% points off (based on how complex the fix is, TA's discretion);
4   else
5     TA may contact the student by email or schedule a demo to fix the problem;
6     if (problem solved)
7       11%-20% points off (based on how complex the fix is, TA's discretion);
8     else
9       All points off;
```

So in the case that TA contacts you to fix a problem, please respond to TA's email promptly or show up at the demo appointment on time; otherwise the line 9 above will be effective.

(5) If the code is not working as required in the project spec, the TA should take points based on the assigned full points of the task and the actual problem.

(6) **Lastly but not the least, stick to the collaboration policy stated in the syllabus: you may discuss with you fellow students, but code should absolutely be kept private. Any kind of cheating will result in zero point on the project, and further reporting.**