CS 550 Operating Systems, Spring 2018
Programming Project 0

Out: 2/1/2018, Thursday
Due date: 2/11/2018, Sunday 23:59:59

In this warm-up project, you are going to make some small changes to the xv6 code, then build and run xv6. The goal is for you to get acquainted with the xv6 system. Section 2 gives instructions on how to build and run the xv6 system. Section 3 and 4 describe the tasks of this project. Section 5 gives submission instructions. Section 6 clarifies how this project will be graded.

1 Baseline xv6 source code

For this and all the later projects, you will be working on the baseline xv6 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 6) item (3) and (4), before going to the following link at the end of this section.

- First, log into your GitHub account whose username is the same your BU username.
- Go to the link at the end of this section to accept the assignment. This will create a private repository of your own on the instructor's teaching organization, and start importing the baseline code into your private repository.

  Attention: Sometimes the import process may take a long time. Do not click the "Cancel" button, which will lead to an empty repository. You can just close the page and come back later.

  If you have clicked the "Cancel" button or it already took really long (e.g., more than two hours), contact the instructor and the TAs so we can delete your repository, and you can click the assignment link to import again.
- Once the import process finishes, you can clone or download the repository into your computer and start working

Assignment link: `https://classroom.github.com/a/djO8iRJi`

2 Build and run xv6

You can build and run xv6 using either a CS machine or your own computer.

2.1 Using a CS machine

(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...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$
```

2.2 Using your own computer

You will need a Linux distribution that supports GNU C, which is targeted at the x86 architecture and using ELF binaries.

(1) Install QEMU in your computer. For example, on Ubuntu, just run:

```
    $ sudo apt-get install qemu
```

(2) Perform Section 2.1 step (2).

(3) Perform Section 2.1 step (3).

3 (60 points) Add a new shell command (print messages in user space)

Add a new shell command (i.e., a user space program) named "proj0" to xv6. After this command is added, if doing an `ls` in the root directory, you will see:

```
$ ls
.              1 1 512
..             1 1 512
README         2 2 1973
cat            2 3 13636
echo           2 4 12601
forktest       2 5 8205
grep           2 6 15548
init           2 7 13502
kill           2 8 12721
ln             2 9 12627
ls             2 10 15483
mkdir          2 11 12750
rm             2 12 12739
sh             2 13 25383
stressfs       2 14 13721
usertests      2 15 67232
wc             2 16 14242
zombie         2 17 12355
proj0          2 18 12742
console        3 19 0
```

The second to the last line is the new program. The new command works like `echo`. But it prints a fix string ("CS550 proj0 print in user space: ") before the user-supplied string. For example, here should be the outputs if we run the program three times with different strings supplied:

```
$ proj0 Hello World
CS550 proj0 print in user space: Hello World
$ proj0 CS 550 Operating Systems
CS550 proj0 print in user space: CS 550 Operating Systems
$ proj0
CS550 proj0 print in user space:
```

Note: the system should operatable after boot (i.e., should not freeze).

4 (40 points) Print messages in kernel space

When compiling and booting the baseline xv6, the outputs during boot looks like:

```
xv6...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 ...
init: starting sh
$
```

In this task, add a message "CS550 proj0 printing in kernel space" just before "cpu0: starting" is printed. The desired output is:

```
xv6...
cpu1: starting
CS550 proj0 printing in kernel space
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 ...
init: starting sh
$
```

Note: the system should operatable after boot (i.e., should not freeze).

A few tips:

- "`make clean`" allows you to compile everything from scratch.
- Start early and let us know if you have any problems/questions.
- Check in your code to your private GitHub repository regularly as you work on the project, so that we can monitor the project progress.
- You can use either `git` command line or the GitHub web interface for code check-in. But the former is more convenient and professional.

5 Submit your work

A submission link be available on the MyCourses website. **Once your code in your GitHub private repository is ready for grading, submit a file named "DONE", which minimally**

contains your name and B-number, to that link. Additionally, you can include the following info in this file:

- The status of your implementation (especially, if not fully complete).
- A description of how your code works, if that is not completely clear by reading the code (note that this should not be necessary, ideally your code should be self-documenting).
- Possibly a log of test cases which work and which don't work.
- Any other material you believe is relevant to the grading of your project.

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

6 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 MyCourses.**

(2) The submission time of the "DONE" file shown on the MC 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) Your GitHub username should be the same as your BU username. If your BU username has been used by someone else, contact the instructor for the solution. Any grading issues caused by using incorrect GitHub account will lead to 10 points off.

(4) If you have multiple GitHub accounts, please make sure you log into the one which has your BU username as the GitHub username before you accept the assignment.

Once you accept the assignment, you can check what account you used to accept the assignment. For example, if your the GitHub account you use to accept the assignment is "abc", the name of your private repository will be "cs550-18s-projX-abc" (with X=0, 1, 2, 3, or 4). If you use a GitHub account whose username is different from you BU username, our script will not be able to locate your repository. If the TA needs to correct it manually, 10 points off.

If you made the mistake, you can correct it by logging into the correct GitHub account, and go to the assignment link and accept again.

(5) 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.

(6) 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.

(7) 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.

(8) 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.