Project #0

** This instruction is validated on Ubuntu 16.04.7 LTS.

1. Introduction to xv6

The xv6 is a reimplementation of Unix Version 6 (v6), loosely follows the structure and style of v6, but implemented in ANSI C for an x86-based multiprocessor. Like pintos, xv6 also runs on `qemu` or `bochs`, an x86 emulator.

For more detailed information, see x86 book, written by the same team that ported xv6 to x86.

2. Prerequisite for xv6

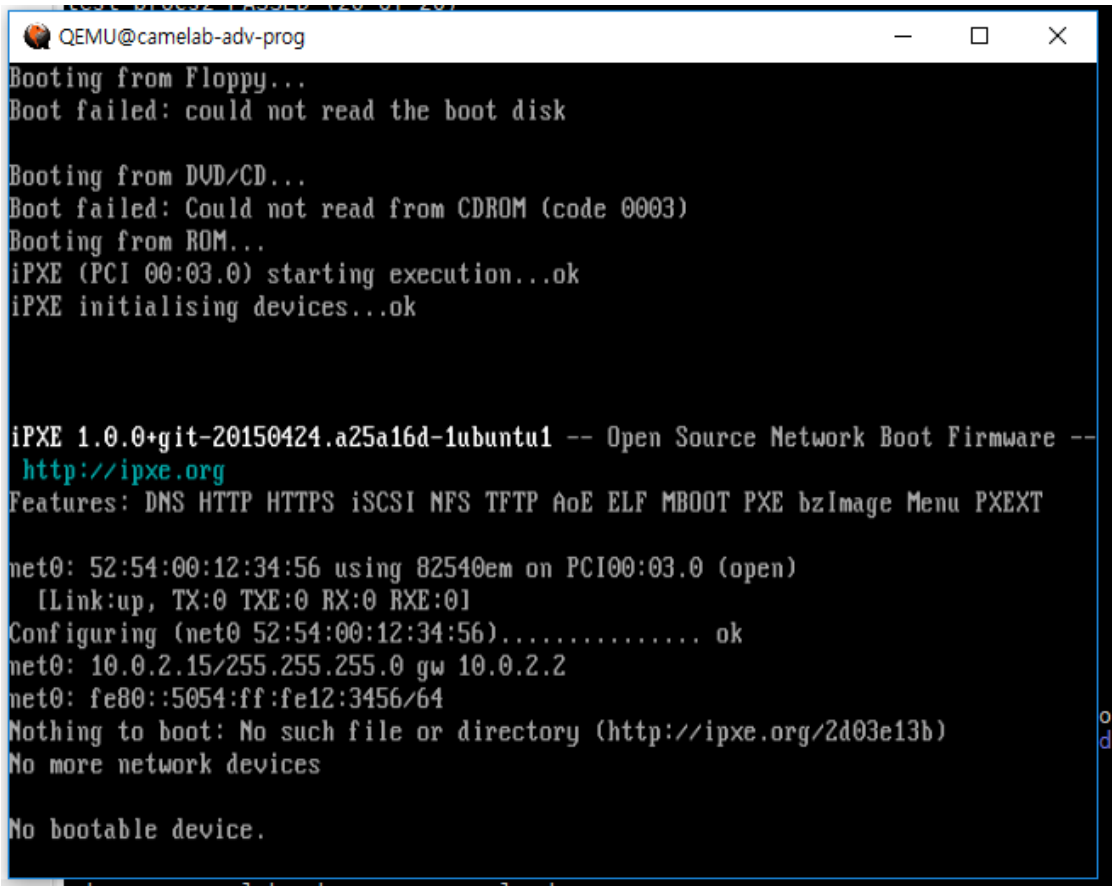
In this section, we will explain what is need to build and run xv6. You need to prepare your own Linux machine for the project. If you don't have a Linux machine, you can use a public PC at Haedong Lounge. (Accounts will be created and announced after 8th march.) If it is your first time using the Haedong Lounge server, please refer to the PDF file named "How_to_use_Haedong_Lounge_Server" uploaded on the KLMS.

On Linux, install `qemu`. After installation, you can run `qemu` by executing `qemu-system-x86_64` (or `qemu-system-x86` or `qemu` – this depends on your Linux)

```
# On Debian-like system
sudo apt update
sudo apt install qemu

# On RedHat-like system
sudo yum install qemu
```

You can see window shows monitor of guest operating system. Because no disk image specified, qemu bios tries to boot from network.



3. Build and run xv6

In this section, we will explain how to build and execute xv6. First, you have to download "xv6.tar.gz" file from KLMS (https://klms.kaist.ac.kr/mod/resource/view.php?id=509893). After unzip the file, you can see following directory structure.

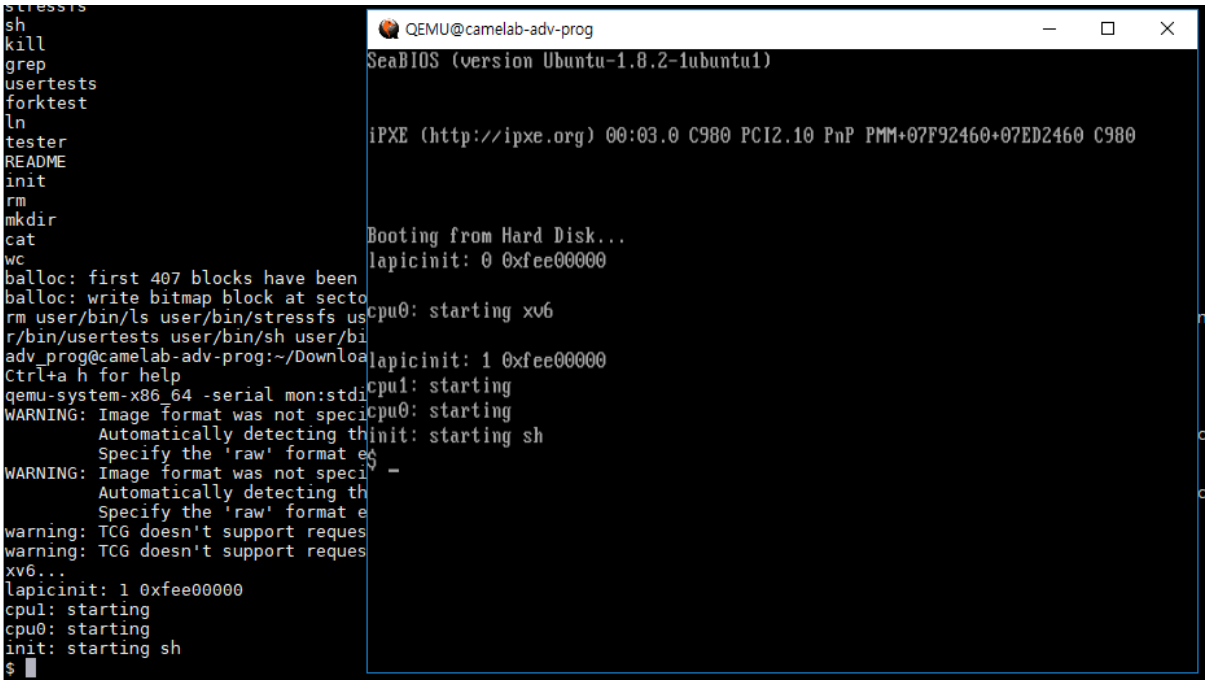
```
# Your home folder
/home/usrname
└── xv6
    ├── FILES
    ├── fspatch.patch     # Will be used for the latter project.
    ├── include           # Include header files
    ├── kernel            # Kernel source codes
    ├── Makefile          # Build script
    ├── README
    ├── tools
    ├── user              # User-mode library
    └── version
```

Before execute `make`, we have to specify where the `qemu` is. Edit Makefile using your favorite editor to change line 58:

```
# Makefile:58
# Edit following line
#QEMU :=
# To
QEMU := qemu-system-x86_64
```

Now, we are ready to build xv6. Build procedure is very simple, just execute `make`.

Build should be completed without any error. To boot xv6, execute `make qemu`. Then you can see terminal screen.



If you got SDL error message saying 'No display detected', you can try following command: `make qemu-nox`, or enable X11WindowForwarding on ssh client.

4. Testing programs from user-level

If you want to create your own program to test, write your code and save file to `user` folder. Then add name of your code to `USER_PROGS` in `user/makefile.mk` file. After `make` and `make qemu`, you can see your program in root directory.

5. Tips for QEMU

If you want to terminate simulation, just close the QEMU window. If you want to use `make qemu-nox` which is useful when you cannot use X Window forwarding, terminate simulation can be done with `[Ctrl + A, X]`. For more commands, hit `[Ctrl + A, H]`.

6. Add new system call function to xv6

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, `getprocs()` , simply returns how many processes exist in the system (=number of processes whose state is not "UNUSED") at the time of the.

Prototype of function: `int getprocs(void)`

Before implementing new system call, find some other calls, like `getpid()` or `fork()`. Most of the time will be spent on understanding the code. There shouldn't be a whole lot of code added.

7. Grading

Please submit a compressed file named "`proj0_<StudentID>_<Name>.tar.gz`" that contains your xv6 directory on the KLMS. If you used tokens or have somethings to say about the project, mention them on the "`proj0_README`" file in the xv6 directory and compress it together.

For grading, we will use automatic grading scripts. For example, Project 0 has six tests. Here is the example output of grading script and we will use points as your score.

```
*****************************************************************************
Summary:
test build PASSED
 (build xv6 using make)

test procs1 PASSED (20 of 20)
 (call getprocs() from a user program)

test procs2 PASSED (20 of 20)
 (call fork() before getprocs())

test procs3 PASSED (20 of 20)
 (call getprocs() with a full process table)

test procs4 PASSED (20 of 20)
 (call getprocs() with a process table filled with zombies)

test procs5 PASSED (20 of 20)
 (call getprocs() after some entries turn back to UNUSED)

Passed 6 of 6 tests.
Overall 6 of 6
Points 100 of 100
```