CS 4375 Fall 2025
Homework 1: Introduction to xv6
50 points

MD Armanuzzaman

Dude Date: September 22, 2025 (11.59 PM))

This assignment is intended to get you started with xv6. You will learn how to install and boot xv6, how to navigate it, and how to write user-level code that runs in the xv6 environment. In addition, you will learn how to set up your GitHub repository for the class where you will turn in your programming assignments. In addition to your code, you should write a lab report answering any questions given below. For each task, also describe any issues you encountered and what you learned from doing that task.

Xv6 is an operating system developed for teaching operating systems principles. We will be using the RISC-V version. In addition to the xv6 code, you will need the RISC-V versions of QEMU, GDB, GCC, and Binutils. Since we will also be using Linux for some assignments, we recommend installing a virtual machine running Ubuntu Linux on your personal computer, and then installing xv6 and the necessary tools in that environment. Doing this will give you superuser privileges in your Linux environment. For a virtual machine monitor, we recommend using VMware, since it is provided at no cost to UTEP students and faculty.

The hardware and software layers for this setup are shown in Table 1.

<table>
<tr><td colspan="2">User programs</td></tr>
<tr><td colspan="2">Xv6 kernel</td></tr>
<tr><td>QEMU (emulating RISC-V hardware)</td><td>Other user programs</td></tr>
<tr><td colspan="2">Linux (e.g., Ubuntu 22.04)</td></tr>
<tr><td colspan="2">VMM (e.g., VMWare)</td></tr>
<tr><td colspan="2">Native OS (e.g., MacOS, Windows)</td></tr>
<tr><td colspan="2">Hardware</td></tr>
</table>

Table 1: Hardware and software stack for running xv6 on emulated RISC-V hardware

Task 1. Boot xv6 and explore utilities. (25 points)

Since you should already have installed VMWare (or are using a different VMM or Windows Subsystem for Linux (WSL 2)) and Linux, we omit those instructions here. To install QEMU and xv6 over Linux, proceed as follows for Ubuntu Linux:

Install dependencies: Linux or WSL 2 on top of Windows

```
$ sudo apt-get install git build-essential gdb-multiarch qemu-
  system-misc gcc-riscv64-linux-gnu binutils-riscv64-linux-gnu
```

Install dependencies: macOS

```
$ brew install git make gcc gdb qemu riscv-tools
```

Optional: Note if rescv-tools installation is generating faults run the following before:

```
$ brew tap riscv-software-src/riscv
```

Clone and build xv6:

Create a GitHub account if you don't already have one. Follow the instructions at here to create a new private repository (e.g., myxv6) and duplicate the repository at `https://github.com/Tomal-kuet/xv6-riscv-labs`.

Create a separate branch for each homework (hw1 for this assignment) which you can later merge with your main branch if desired. See `https://www.howtogeek.com/714112/how-tocreate-a-new-branch-in-github/` for information on how to create a new branch.

Then type:

```
$ cd xv6-riscv-labs
$ make qemu
```

The above steps will build xv6 and boot it in the QEMU RISCV emulator. If you type ls at the prompt, you should see output similar to the following:

```
$ ls
.               1 1 1024
..              1 1 1024
README          2 2 2226
cat             2 3 23960
echo            2 4 22784
forktest        2 5 13144
grep            2 6 27320
init            2 7 23888
kill            2 8 22744
```

```
ln             2 9 22696
ls             2 10 26192
mkdir          2 11 22848
rm             2 12 22832
sh             2 13 41720
stressfs       2 14 23848
usertests      2 15 156072
grind          2 16 38032
wc             2 17 25088
zombie         2 18 22240
sleep          2 19 22608
ps             2 20 23728
pstree         2 21 24912
pstest         2 22 23880
console        3 24 0
```

These are the files that `mkfs` includes in the initial file system; most are programs you can run. You just ran one of them: ls

xv6 has no ps command (but we will implement one for a future homework assignment!), but if you type `Ctrl-p`, the kernel will print information about each process. If you try it now, you'll see two lines: one for init, and one for sh.

Choose and explore three additional user commands. In your lab report, explain what they do and how they work, and show example results. To explain how they work, you will need to look at and understand the source code.

To quit QEMU type: `Ctrl-a x`.

Task 2. Implement the uptime utility. (25 points)

Tasks 2 involves coding. Please create a separate branch for each homework assignment (hw1 for this assignment), which you can later merge with your main branch if desired.

Implement the UNIX utility `uptime` for xv6. Your `uptime` command should print the number of clock ticks since system start. A tick is a notion of time defined by the xv6 kernel, namely the time between two interrupts from the timer chip. Your solution should be in the file `user/uptime.c`.

Some hints:

- Use the system call `uptime()`.
- See `kernel/sysproc.c` for the kernel code that implements the uptime system call (look for `sys_uptime`), `user/user.h` for the C definition of uptime callable from a user

program, and `user/usys.S` for the assembler code that jumps from user code into the kernel for `uptime`.

- Make sure `main` calls `exit()` in order to exit your program.
- Add your `uptime` program to `UPROGS` in `Makefile`; once you've done that, make `qemu` will compile your program and you'll be able to run it from the xv6 shell.

Run the program from the xv6 shell:

```
$ make qemu
...
init: starting sh
$ uptime
up xxxx clock ticks
$
```

Turn-in procedure and Grading

Please use the accompanying template for your report. Convert your report to PDF format and push your hw1 branch with your code and report to your xv6 GitHub repository by the due date. Also turn in the assignment on blackboard with the URL of your github repo by the due date. Give access to your repo to the and TA.

TA github email: *danielmarin350@gmail.com*

This assignment is worth 50 points. The breakdown of points is given in the task descriptions above. The points for each task will be evaluated based on correctness of your code, proper coding style and adequate comments, and the section of your report for that task.

You may discuss the assignment with other students, but do not share your code. Your code and lab report must be your own original work. Any resources you use should be credited in your report. If we suspect that code has been copied from an online website or github repo, from a book, or from another student, or generated using AI, we will turn the matter over to the Office of Student Conduct for adjudication.