Lecture Notes on Operating Systems

Lab: Understanding the xv6 Filesystem

Goal

The goal of this lab is to familiarize yourself with the xv6 filesystem.

Before you begin

- Download, install, and run the xv6 OS. You can use your regular desktop/laptop to run xv6; it runs on an x86 emulator called QEMU that emulates x86 hardware on your local machine.
- First, download the publicly available version of xv6, then copy the patch provided as part of this lab onto the original xv6 code.
- For this lab, you will need to understand the filesystem code in xv6, which is mainly contained in the following files: `sysfile.c`, `file.c`, `fs.c`, and `bio.c`.
- Learn how to write your own test programs in xv6. We have provided a simple test program `testcase.c` as part of our patch. This test program is compiled by our patched `Makefile` and you can run it on the xv6 shell by typing `testcase`. You must be able to write other such test programs to test your code. Note that the xv6 OS itself does not have any text editor or compiler support, so you must write and compile the code in your host machine, and then run the executable in the xv6 QEMU emulator.

Exercises

In this lab, you will add the following system calls to xv6. These system calls print out various statistics about the xv6 file system. Below are the system calls and a description of the information that must be printed out. You may choose any sensible, readable display format of your choice for displaying the output to the screen. Note that none of the system calls have a return value.

1. The system call `printDiskStats()` should print out the following information about the disk to the screen: the total number of inodes in the file system, the number of free inodes, the total number of data blocks, and the number of free data blocks.
2. The system call `printBufferCacheStats()` should print out the following information about the disk buffer cache to the screen: the total number of buffers, and number of valid buffers containing data from disk (whether clean or dirty).

3. The system call `printFileTableStats()` should print out the following information about the global open file table to the screen: the total number of slots in the file table, number of occupied slots (occupied by any type of file), and the maximum inode number (across those slots that are occupied by regular files).

4. The system call `printFDStats()` should print out the following information about the file descriptor array of a process to the screen: the total number of slots in the file descriptor array, and the number of occupied slots.

For all these system calls, you may assume that the disk device to be used is the device that is hosting the current working directory of the current process. The xv6 patch provided to you contains a simple test program and the expected output from it. The test program invokes the above system calls before and after performing various filesystem operations. While the format of the output can vary in your case, the summary of the information printed out, and the change in the information after various operations, should be similar.

Submission instructions

- You must submit all modified xv6 files. We expect them to be the following: `syscall.h`, `syscall.c`, `sysfile.c`, `user.h`, `usys.S`, `defs.h`. `bio.c`, `file.c`, `fs.c`.

- Place these files and any other files you wish to submit in your submission directory, with the directory name being your roll number (say, 12345678).

- Tar and gzip the directory using the command `tar -zcvf 12345678.tar.gz 12345678` to produce a single compressed file of your submission directory. Submit this tar gzipped file on Moodle.

Grading

We will check correctness of your code using various test cases, including those beyond what given you. We will also read through your code to ensure that you have adhered to the problem specification in your implementation.