ENCE360 Lab 10: xv6

Objectives

This lab is a sort of victory lap. We're going to experiment with a toy operating system called xv6, designed for teaching purposes and written in ANSI C.

This lab will only touch a few aspects of this OS, but hopefully you now understand enough about operating systems to imagine how you might implement most of them.

Preparation

As well as this handout, `lab10xv6.zip` should contain another `.zip`, called `xv6`. Extract this into its own directory. Have a look at the file names, and take a quick guess as to what each might do.

xv6 runs on 32-bit x86, but the lab machines are x86-64; a different architecture! Therefore, we will use an emulator called QEMU to run the xv6, instead of a VMM like VirtualBox. You can run xv6 in the emulator with

```
make qemu
```

When you make modifications to the user programs (e.g. `cp.c`), you'll need to `make clean` and then re-run the emulator so that the changes are included in the filesystem.

Many of the standard Unix shell commands are present in xv6, though some are conspicuously absent. There is also no shell auto-complete, so keep your filenames short and simple! Here's an example set of commands, the output of which should make sense to you. Give them a go in xv6.

```
ls
wc README
cat README | wc

mkdir example
ls
rm example

echo "hello world"
echo "hello world" | wc
```

xv6 comes with a much smaller set of system calls than Linux, but it's still enough to do any task. The list of them can be seen inside `user.h`.

Program `cp.c`

xv6 comes with no command to copy files, though something hacky can be done with piping, e.g.

```
cat srcFile > destFile
```

We've given you a skeleton for the copy command, `cp.c`. Implement it. If you need hints, have a look inside `cat.c`. Note that xv6 has no buffered I/O, so you will have to use the lower-level `open()`/`close()` and `read()`/`write()` system calls.

You can test your implementation by using `cat` to have a look at a file's contents. For example:

```
echo "hello world" > myFile
cp myFile myFile2
cat myFile2
```

You should also test your `cp` on larger files, such as `README`.

Program `pipetest.c`

Just like Linux, xv6 has pipes. `pipetest.c` is a minimal program which shows how to use the `dup()` system call to redirect input, in this case to the `wc` program.

xv6 does not have a `dup2()` system call. Instead, we close `stdin` manually (using `close(0)`) and then call `dup()`, which will copy the file handle into the first free file descriptor (in this case 0).

Modify `pipetest.c` so that is uses two pipes instead of one. The child should run `wc`, but take input via a pipe from the parent and output via a pipe to the parent. The parent should pass input to the child and then print its output, similar to what a shell program does.