Project 1: Warm-up Project

Important Dates

~~**P1a (Table and Keypad) Due:** Sunday, 2/08, by 11:59pm.~~

P1b (xv6 Intro) Due: Sunday, 2/15, by 11:59pm.

Questions?

Post your questions in piazza (or, of course, visit us in person during office hours!). Please do refrain from posting your entire code in piazza.

Overview

There are two parts to this project:

- Table and Keypad (P1a): to be done on the lab machines, so you can learn more about programming in C on a typical UNIX-based platform (Linux)
- Kernel Intro (P1b): to be done in our xv6 hacking environment, so you can learn more about what actually goes on in a real kernel.

Click on the above links to learn more about what you should do. READ EACH CAREFULLY! Note: it will take a long time to read each and really make sure not to miss anything.

Notes

Before beginning, **read this tutorial.** It has some useful tips for programming in the C environment.

Read K+R (the course textbook on C) as it is a great introduction to the language. Familiarizing yourself with these topics will help you complete the programming projects:

- character arrays, null terminated strings, character pointers (1.9, 5.5, 5.10)
- pointer arithmetic (also called address arithmetic) (5.4)
- using array notation on pointers and vice-versa (5.3)
- pointer arrays (5.6)
- function pointers (pointers to functions) (5.11, 5.12)
- external declarations and the difference between definition and declaration (4.3, 4.4, A8)
- preprocessor file inclusion (#include) and macro substitution (#define) (for constant style definitions) (4.5, 4.11)

These topics highlight the main differences between C and other languages like Java. They are generally what cause issues for programmers new to C.

This project, unlike all others this semester, must be done alone. Note that it is always OK to talk to others about your code, as well as help them debug their code. Copying code, however, is considered cheating. Don't do it! How will you learn that way? Read this for more info on what is OK and what is not.

Handing It In

You are free to modify the contents of your handin directory prior to the due date.

Linux

For the C/Linux part of this project, you should turn in your source files and Makefile. We will compile your code by calling make on your handin directory. This implies that your Makefile's `all` target should generate the expected binaries.

Provide all files necessary for a successful compilation. This means that for tnine project, submit the header file and the object file that was provided.

There will be separate directories made available for the two parts of P1A within the `p1/linux` folder. Copy the files for the Multiplication Table program into the folder `linux/multable` and those for the T9 program into `linux/tnine`

The handin directory (for Section 1) is `~cs537-1/handin/login/p1` where `login` is your login; for Section 2, it is the same but with a 2 (~cs537-2/handin/login/p1) . For example, Sankar's login is `sankarp`, and thus he would copy his beautiful code into `~cs537-1/handin/sankarp/p1/linux`. Copying of these files is accomplished with the `cp` program, as follows:

```
shell%
cp multable.c ~cs537-1/handin/sankarp/p1/linux/multable
```

xv6

For the xv6 part of the project, copy all of your source files (but not .o files, please, or binaries!) into the `xv6/` subdirectory of your p1 directory. A simple way to do this is to do make clean (to remove all *.o files) and then copy everything into the destination directory. Now type `make` in the handin directory to make sure it builds, and then type `make clean` to remove unneeded files.

```
shell% make clean
shell% cp -r . ~cs537-1/handin/sankarp/p1/xv6
shell% cd ~cs537-1/handin/sankarp/p1/xv6
shell% make
```

```
shell% make clean
```

Readme

Finally, into each of your p1 directories, please make a README file. In there, describe what you did a little bit. The most important bit, at the top, however, **should be the authorship of the project.**

To generate this README file, please make use of the template available at `/u/c/s/cs537-1/public/samples/README`. Your README should **strictly follow** this template as an automated grader scores your projects, and identifies authorship from this file to assign scores. **An incorrect README will cause failure while parsing and result in a score of 0 on the project.**

Have a separate readme file for each subparts of the project.

For xv6, you could overwrite the README file already present in the xv6 folder.

Tests

We will be releasing a few tests for both the linux and xv6 projects. These will be a guide to help you validate a basic implementation. By no means are the released tests exhaustive, which means you will have to test your code on edge cases, keeping the specification in mind. The actual tests used to grade your programs will be more exhaustive, and will attempt to test these corner cases in your program.

You will be notified via class mailing list, piazza and a post on this page, when the tests are released. Instructions on how to run the tests will be made clear post-release.

Verification Scripts

Apart from the test scripts, we will also be providing you a verification script. This verification script will help you make sure that you have turned in all the required files correctly. You should run this script after you think you have copied all the necessary source files including the makefile and README file to the handin directory.

The following are the sanity checks that the verification script will run:-

- Check if the README file is present in the expected location and is in the correct format.
- The Makefile is present in the expected location.
- Running make generates the binaries expected by the test scripts in the correct location.

If you've not handed in your code, Makefile or the README file in the

correct location, this script will be able to notify you when you execute it, so that you can move them in to the correct location.