Project 1: Warm-up Project

Due: Tuesday 6/27/17, 11:59 pm

NO LATE WORK ACCEPTED IN THIS COURSE. NO EXCEPTIONS.

Questions?

Please post your questions on Piazza so that your classmates with similar problems can benefit from it. However, if the question is about your code, visiting during **office hours** is the most efficient way of getting help. You should **not** post your code on Piazza.

Overview

There are two parts to this project:

- Sorting: to be done on a CS Linux machine so you can learn more about programming in C on a typical UNIX-based platform (Linux).
- Kernel Intro: 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.

Learning C

If you are not completely comfortable coding in C, **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.

You may also want to watch this video for a C tutorial. The tutorial is given by a former CS 537 instructor and goes over details of C that are especially useful for the projects in CS 537. The

following topics are covered: pointers and arrays, structs, function pointers, preprocessor, multiple files, and gdb. Again, you might want to pause the video while you try out the examples yourself.

Notes

This project 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.

Keep your source code in a private directory. An easy way to do this is to log into your account and first change directories into `private/` and then make a directory therein (say `cs537/p1` , by typing `mkdir -p cs537/p1` after you've typed `cd private/` to change into the private directory). However, you can always check who can read the contents of your AFS directory by using the `fs` command. For example, by typing in `fs listacl .` you will see who can access files in your current directory. If you see that `system:anyuser` can read (r) files, your directory contents are readable by anybody. To fix this, you would type `fs setacl . system:anyuser ""` in the directory you wish to make private. The dot "." referred to in both of these examples is just shorthand for the current working directory.

Handing It In

For the C/Linux part of this project (sorting), you should turn in one file, called `varsort.c` . We will compile it in the following way:

```
shell% gcc -O -Wall -o varsort varsort.c
```

so make sure it compiles in such a manner. You should copy this file into your handin directory into the subdirectory called `linux` .

The handin directory is `~cs537-1/handin/$USER/p1` where `$USER` is your CS login. Copying of these files is accomplished with the `cp` program, as follows:

```
shell% cp varsort.c ~cs537-1/handin/$USER/p1/linux/
```

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 copy everything into the destination directory, then type `make` to make sure it builds, and then type `make clean` to remove unneeded files.

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

Finally, into your p1 directory, please include a README file. In there, describe what you did a little bit (especially if you ran into problems and did not implement something). The most important bit, at the top, however, should be the authorship of the project.