CS107 Handout 03
Autumn 2000 September 28, 2000

Section Handout 1: UNIX Basics

This handout was written by Andy Gray, past CS107 and CS193D. Miss you, Andy!

Section Agenda

Except this time, sections meet Thursdays in 420-041 for an hour starting at 4:00 pm. This week we'll cover some of the basics of using the Leland UNIX environment to edit, compile, and submit your programs:

- ⇨ Logging in, class directory, starter files
- ⇨ Editing your source files
- ⇨ Compiling your program
- ⇨ Submitting your program

If you find that you need more individualized help getting started, be sure to come to office hours this week or early next week. We'll have plenty of time to help you before the mad rush of assignments begins. Office hours will be posted to the web site over the weekend, after CS107 TAs have finalized their own academic schedules.

Getting Started

You'll need a **leland** account to submit the assignments for this class. If you don't already have an account, talk to the consultants on the second floor of Sweet Hall to open one.

In addition to submitting, most of you will probably do all of your development on the **leland** systems. The tools on the UNIX machines are very good, although somewhat more difficult to get started with. Although we will be able to offer some support for those of you using Microsoft Visual C++ for the PC and Metrowerks CodeWarrior for the Macintosh, I suggest you use the UNIX tools wherever possible. Even though ANSI C++ is highly portable, we will be grading your submissions on the **leland** systems, so it is your responsibility to make sure that your program written on platform X works there as well. In addition, working on the **leland** machines makes your files available from any remote machine and backs up your work each night.

Once you've logged on, you'll find yourself in your home directory. Here's a quick overview of changing directories, creating directories, and copying files. If you're completely new to UNIX, it would probably be a good idea to stop by office hours next week, so that one of us can show you around. There are also several good quick reference sheets at Sweet Hall located near the consulting desk.

You can create a new directory for all of your projects with the following (text that you type is bold):

```
saga5:~> mkdir cs107
```

and then change to the directory with

```
saga5:~> cd cs107
saga5:~/cs107>
```

Notice that the prompt changes to reflect the directory you are currently working in. The saga5 part indicates the name of the machine I'm currently logged into, but there's nothing particularly special about this host.

The directory for class materials is /usr/class/cs107/. In here you'll find starter files which we provide for the assignments. You'll also find code samples. Let's copy one of the sample programs over to your newly-created 107 directory:

```
saga5:~/cs107> cd /usr/class/cs107/assignments
saga5:/usr/class/cs107> ls
hw0/
saga5:/usr/class/cs107> cp -r hw0 ~/cs107
```

The last command copies all of the files in the sample directory recursively (-r) into your cs107 directory (~ is the same as your home directory, whatever it may be).

Now, we've copied the sample directory over to your own cs107 directory.

Later when it's time to start working on the assignments, you'll copy the starting files from /usr/class/cs107/assignments the same way.) You can get to your copy of the sample directory this way:

```
saga5:/usr/class/cs107> cd ~/cs107/hw0
saga5:~/cs107/hw0> ls
Makefile  main.cc  stack.cc  stack.h
saga5:~/cs107/hw0>
```

Now that you've copied these files over, you can start editing the source code and compiling the program.

Editing Files

There are a number of editors you can use in the UNIX environment, but we'll just go through a brief overview of the most popular, emacs. Although it's incredibly powerful and really nice to use once you've figured it out, emacs can be a bit intimidating at first. If you go to Sweet Hall to work (or run an X server on your local machine), you can run emacs in a graphical mode, or better yet, use its sister program xemacs. Both emacs and xemacs support plain old telnet sessions too, but you'll need to rely on the command keys since you won't have any menus to use.

To run emacs, you just type emacs at the prompt (or xemacs for, surprisingly enough, xemacs):

```
saga5:~/cs107/hw0> xemacs
```

This brings up the program. At Sweet Hall, a new window will pop up for the program. If you're running it over telnet, it will just take over the telnet window. You can open a file for editing by using the Files|Open File... menu, or you can type C-x C-f (where C is the control key) and then the name of the file you want. If that seems a bit cryptic, it is. If you're new to this, you'll find it easier if you use the menus at first and then gradually introduce the keyboard shortcuts into your routine until you don't need the menus at all.

To open a file for editing on the command line when you start the program, just type the name(s) of the file(s):

```
saga5:~/cs107/hw0> xemacs main.cc
```

This brings up xemacs with main.cc already in it. Here's a little tip if you work at Sweet Hall or run an X server: typing an ampersand at the end of a command puts it in the background and frees up the shell for other tasks. Otherwise, the shell is tied up with xemacs and you don't get the prompt back.

```
saga5:~/cs107/hw0> xemacs main.cc &
```

If you try this when you're running emacs in a telnet window, however, the program is immediately suspended and it looks as if it didn't start at all.

Now you have the sample C++ program open for editing. You can take a look at the code, and make changes to it if you want to. If you look over main.cc, you'll quickly notice that it is definitely not a C program. Don't worry about the details for now; the main thing to recognize is that an ordinary C compiler **will not** compile main.cc. This may prove useful for some of you in testing whether or not your own compiler is handling C++ code correctly. We'll be programming in C initially, but we WILL transition to C++ at some point, so this distinction needs to be clear. We're using C++ here for the purposes of illustration, because CS193D students are also watching.

Compiling

Now the real fun begins. If you've used Makefiles in other classes, then there's really nothing new to learn here. For each of the assignments we will provide you with a Makefile for the project (although in some cases you may need to add your own files to the SRCS line). To compile with a Makefile, go to the directory which contains the source files and the Makefile itself, and run `make`:

```
saga5:~/cs107/hw0> make
/usr/newsw/bin/g++ -g -Wall -MM main.cc stack.cc > Makefile.dependencies
/usr/newsw/bin/g++  -g -Wall  -c -o main.o main.cc
/usr/newsw/bin/g++  -g -Wall  -c -o stack.o stack.cc
/usr/newsw/bin/g++ -o foo main.o stack.o
saga5:~/cs107/hw0>
```

If all goes well, you'll get a few lines of intermediate output but no warnings or errors. If none of the source files have changed since the last compile, `make` will let you know (this often comes up when you forget to save your changes before recompiling).

The `g++` above refers to the Gnu C Compiler, `gcc`. It's actually both a C and C++ compiler (aside: C++ is a strict superset of C), but if you invoke it as `g++` you have a better chance of getting it to recognize your C++ code as such (as opposed to C code) and linking against the right libraries. If you stick to using the Makefiles that we provide, then you won't need to worry about this issue (at least, not for now). If you have questions about how to edit Makefiles, email us at the question queue and we'll be happy to help you.

To clean up your workspace (delete `.o` object files, etc.) run `make clean`. The submission script described below will require you to do this before turning in your work.

Submitting

Before you submit your program you will need to write a README file that describes your submission. In it, you should describe how to run your program, and note any extra features and bugs (if any).

Once you've finished writing your program, there's just one more step you need to complete: electronic submission. Fortunately, it's very simple. Our submit program is

To submit your hw0, you should do this in the directory where your code lives:

```
saga5:~/cs107/hw0> /usr/class/cs107/bin/submit
```

Answer the questions when prompted. The submission script checks your files, makes sure you included a README and cleaned up your project with `make clean`, and then copies the files over to our submission area:

When the script asks for the directory where your files are located and you are running the script from your working directory, you can just type . (a single period—it's a shorthand for the current directory).

If after submitting you decide that you need to make some additional changes, you can run the submit program again and send us the new version of your code. We will look at the last submission we receive from you (and we'll determine the number of late days based on when you turned in the last submission).

The submission script will let you know how your attempt turned out. If you're getting repeated failures and you're not sure why, send email to the course staff and we'll help you out.