Linux tutorial and gdb walkthrough notes.

Since we didn't end up using slides for the overall presentation, this will just be a quick guide to the topics covered. We encourage you to try these out on your own, and to use the 390 lecture slides or the 303 lecture notes from past quarters for a more thorough coverage of many of these topics (and more!).

SSH clients:
On a Mac or Linux machine, you can simply use the Terminal application; Windows users will want to download PuTTY or similar clients. To connect to a remote machine (for our purposes, we'll say attu), you will need to ssh in:

```
ssh [CSE username]@attu.cs.washington.edu
```

You'll need your password to login. If you don't know it, go talk to the front desk or email support.

Help:
If you get lost, Unix systems have a very helpful documentation system; simply type man [command] to see the help pages. For further tutorials or in-depth walkthroughs, google is very helpful: these are really old systems about which lots of helpful documentation has been written up. Also, the slides and notes linked above will prove helpful.

Navigation:
Once you're on the command line, you need to be able to look around, move between directories, and edit files. Some generally useful commands:

`ls`	displays (non-hidden) files in the current directory
`pwd`	displays the location of the current directory
`cd [directory name]`	change to another directory
`cd ..`	change to the parent directory
`mkdir [name]`	create a new director
`rmdir [name]`	remove the specified directory
`rm [file name]`	remove the specified file
`mv [file name] [dest]`	moves the file to the provided location
`diff [file1] [file2]`	returns the differences between the two files
`less [file name]`	shows the file contents in a readable format; q to exit
`exit`	logs out of the shell

Text editors:
The two most famous command-line text editors are emacs and vi/vim. If you're interested in learning one of these (highly recommended for your future career!), the

internet has a number of thorough guides to both. However, they're simply too complicated for us to spend much time on here.

If you aren't interested in mastering a tricky (but very useful) editor like either of the above, you may want to stick to the small editors pico and nano, which wear their commands on their sleeves.

C:
For the purposes of this class, we'll be using a number of C programs. Some helpful C commands include:

`gcc [file name]`	compiles the provided file into the executable a.out
`gcc -o [executable name] [name]`	as above, but you provide the name of the executable
`gdb`	the GNU debugger; more information on this later
`./[executable name]`	runs the program in the provided file

File transfer:
If you need to transfer files between a remote machine (attu) and your personal computer, the scp command is a useful way to do this. Many personal computers don't allow external machines to get ahold of them, so it may be easiest to rely on transferring files through the command-line on your personal computer.

To transfer *from* attu, *to* your own computer, use the following:

```
scp [username]@attu.cs.washington.edu:[complete path to desired file] [location on personal computer]
```

To reverse the transfer (going from your own computer, to attu), simply reverse the command:

```
scp [complete path to desired file]  [username] @attu.cs.washington.edu:[location on attu where you want the file to end up]
```

The . character refers to the directory you're currently in.

On Windows machines, it may not be possible to transfer files over the command line, so you might want to use WinSCP, a file transfer client.

gdb:
I have attached a pdf containing a number of screenshots of stepping through the disassemblies of a pair of small hello world-type programs, which may be helpful in navigating gdb. However, I have only touched on the basics, as gdb is a very complicated and powerful program, so you may wish to seek more information elsewhere. A useful cheat sheet is linked from the course website, and as with any of

the other topics covered in these notes, much more information is available at a casual search.

Piping:
I promised those people who attended the lecture a brief demonstration of piping, which allows you to run one command and send its console output as input to a second. For a cute demonstration of this, you can try something like ls l less, or (alternatively and perhaps slightly more dramatic) ls l sort -r. (Sort, which we haven't discussed here, does exactly what it sounds like; the -r flag sorts in reverse order. You can read more about the other possible flags to sort in man sort, of course.)

Key commands:
It's useful to remember that programmers are generally lazy, and try to avoid doing difficult things, like moving the mouse or entering input, whenever a reasonable alternative presents itself. Accordingly, some useful shortcuts to avoid undue clicking and typing:

`Ctrl-A`	moves cursor to the front of the current line
`Ctrl-E`	moves cursor to the end of the current line
`up and down arrows`	scroll through the commands you've entered previously

Additionally, some commands you may find helpful while switching between programs:

`Ctrl-Z`	suspend (fg to resume)
`Ctrl-D`	signifies end of input
`Ctrl-C`	terminates current process