First time logging into tux.cs.drexel.edu

I. First, you have to be connected to the Internet (dial-up is fine).

II. Next, you need an account on the CS machines. If you are taking a CS course, then you are eligible. If you've already created your account, skip down to step III.

1. Before we get there, you need your ID that the university gave you. It will be 4 or 5 characters, your initials followed by some numbers.
2. You'll need that password that goes w/that ID, the one you use to read your Drexel mail.
3. Now, go to this link: https://www.cs.drexel.edu/Account.php
4. You'll see this screen:

Type in your ID and your password (it is masked, so feel safe).

5. Click the **OK** button **once**.

If you get any noise about certificates, just click through them.

III. Now, you need a program that will, at a minimum, give you a secure terminal interface to another computer. SSH (Secure Shell Client) and Putty can be used on Windows machines. If you have Mac OS X (or Linux) you already have SSH, just open a terminal window, type:

```
ssh userId@tux.cs.drexel.edu
```

SSH on Windows

You may download SSH from Drexel's download site. It does things such as encrypt your userid and password, and all subsequent traffic.

Use your university id, e.g., js123, as your username, and probably the password that gets you into DrexelOne.

Then just look around, some good stuff in there. SSH also comes w/a secure FTP program that you may use.

SSH (and SFTP) on Windows have command-line interfaces, and GUI clients (I recommend using the SSH GUI client, or load cygwin).

PuTTY

PuTTY is free, and runs stand-alone (doesn't require installation). It can be found at http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html, along with other useful binaries, like `pscp` (putty scp), `psftp` (putty sftp), and `puttygen`, which will generate keys for you (for SSH to use).

If you have cygwin installed, I recommend using puttycyg, which is precisely just PuTTY, but includes an xterm interface to cygwin. Download and documentation can be found a http://code.google.com/p/puttycyg/ .

I also recommend you visit http://www.portableapps.com . You'll find a bunch of programs which have been modified to be run from a stick. Settings are stored on the device, rather than in the registry, etc. Some of the useful ones: Firefox, PuTTY, gimp, VLC, Thunderbird, Sumatra PDF reader, gnucash, and many others.

IV. Whichever program you use you will connect to **tux.cs.drexel.edu**. That is a server farm. It's actually a bunch of (64-bit) Linux boxes, you'll be sent to one that's not too crowded.

```
kschmidt@caucus Ref> ssh js123@tux.cs.drexel.edu
tux64-06(1)#
```

Note that you're now logged into tux, and when you type it will be tux that responds. You are done interacting with SSH (directly) for the moment.

I think we now start you in the **bash** shell (may be **tcsh**), the program that allows you to talk to the OS. You can change that, if you have a preference. It is just one of several common shells, and the one that I use, but you'll have to learn and pick your own. Things I show you here are common to all shells.

V. Getting used to Unix, and compiling your first program:

Take a look at my Unix cheat sheet

1. First, use pwd to see where you are (I'll make stuff that you type this color):

```
tux64-06(1)# pwd
/home/kschmidt
```

This is your home directory. It's your space. All your stuff will go under here somewhere, hopefully into a subdirectory. **Note:** This does not mean that other people can't see your files. We'll examine this shortly.

2. This next step is just a little file that you'll copy from me, to make your VI experience a little more pleasant.

```
tux64-06(2)# cp ~kschmidt/Public/.vimrc .vimrc
tux64-06(3)# ls
```

Hey, where is it? That 3rd command you entered was supposed to list out your directory. Where is that file you just copied? (If you've already used your CS account, your results may differ a little.)

```
tux64-06(4)# ls -a
./    ../    .bashrc    .bash_profile    .bash_history
```

Ah, okay. Files that start w/a period (.) are hidden, by default.

Note well: Capitalization matters to Unix, both in commands and in file (and directory) names.

3. Now, let's create a directory that you will use for **all** of your work in my class, and lock it down so that only you have access to your homework files. (I will, occasionally, run a script that searches all of your directories for current homework. If I can see it, then others can too. Trouble.)

```
tux64-06(5)# mkdir CS260
tux64-06(6)# chmod 700 CS260
tux64-06(7)# ls
CS260/
```

Okay, we see the directory there. (Note that the '/' character isn't really part of the name, it's just a more legible format that you asked for.)

4. Now we'll go into that directory w/the `cd` command and take a look around:

```
tux64-06(8)# cd CS260
tux64-06(9)# pwd
```

```
/home/kschmidt/CS260
tux64-06(10)# ls
```

5. Hey, there's nothing there! Well, it's a brand-new directory. Let's create a simple little program there:

```
tux64-06(11)# pico hello.cc
```

You'll be in the pico editor. In lieu of creating the screen, I'll just stick a screenshot here:

You'll want to type in a simple program:

```
#include <iostream>
using std::cout;

int main()
{
  cout << "\nHello!\n\n";

  return 0;
}
```

You save the file by hitting ctrl-O, and following the prompts. You can then exit by hitting ctrl-X.

6. Do a list, see your source code there:

```
tux64-06(12)# ls
hello.cc
```

7. Okay, now we can compile the program:

```
tux64-06(13)# g++ hello.cc
tux64-06(14)# ls -CF
```

```
a.out*   hello.cc
tux64-06(15)# ls
a.out   hello.cc
```

a.out is the name of your program. The asterisk isn't part of the name, it's just pretty formatting telling you that that file has execute permissions on it. Give it an **ls** to see it without the formatting.

8. To run it, simply type the name of the program:

```
tux64-06(16)# ./a.out

Hello!
```

Note that the leading **./** tells the shell to look in the current directory for the command. See the PATH environment variable in the bash man (or info) pages[1]. See also the variable PWD and the command **pwd**

[1]**Note:** If you add the current directory (.) to your path, add it at the end.

9. Since program files are comparatively large (do an **ls -l** to see the sizes) and you have the source code, go ahead and remove the program you just made:

```
tux64-06(17)# rm a.out
```

10. You can rename the program from **a.out**, or tell the compiler what name you'd like the program to have using the **-o** option:

```
tux64-06(18)# g++ -o myProgram hello.cc
tux64-06(19)# ls
hello.cc myProgram*
tux64-06(20)# ./myProgram

Hello!

tux64-06(21)# rm myProgram
```

11. When you're done don't forget to exit the shell. Don't simply kill the program that is providing the connection. You want to tell tux that you're leaving. Do this by typing **exit**. It will be obvious when you are disconnected.

```
tux64-06(22)# exit
```

Your files are safe. In fact, some very capable people in the department back up all our files nightly, and, you can get to them from anywhere, as long as you have access to the Internet.

Last updated 7/24/2011