Command Line for the Skeptical Studio

It's time to learn that scary beast we call command line. Today's studio will walk through obtaining a UNIX terminal, file browsing, file creation, input redirection, and terminal source control.

Let's begin!

Obtaining a UNIX Terminal

Are you on **Mac**? We will connect to the WUSTL UNIX systems just to make sure you have `svn` installed.

1. Open up Terminal.
2. Connect to the UNIX boxes WUSTL provides by typing `ssh YOURWUSTLUSERNAME@shell.cec.wustl.edu`.
3. If asked, `yes`, you want to save the RSA key.
4. Enter your computer password and your WUSTL password, as prompted.
5. When the connection opens, type `qlogin` and enter your password again when prompted.
6. Done.

Are you on **Windows**? Windows is not a UNIX machine, so we need to *connect remotely* to one.

1. Download PuTTY
2. Under **Host Name**, type `YOURWUSTLUSERNAME@shell.cec.wustl.edu`.
3. Click `Open`, entering your WUSTL password when prompted.
4. When the connection opens, type `qlogin` and enter your password again when prompted.
5. Done!

File Browsing

Well, you got yourself a terminal window. You might want to get your bearings.

What's the current directory?

1. Print the working directory with the `pwd` command.
2. What files are in the current directory? `ls` will tell you.
3. Write these in your coversheet (For now, just write this stuff down somewhere. You will download the coversheet later in the lab).

That directory will not do. We will be creating several files today, and we don't want to clutter the home directory.

Breaking new ground

Create a new directory, then browse to it!

1. `mkdir DIRECTORYNAME` with whatever directory name you want
2. `cd DIRECTORYNAME` to browse into the new directory.
3. Is there anything in your new directory? Write the answer in your cover sheet.
4. Make another directory within this new one, then see if you can browse back to your home directory (`cd ..` goes up a directory, and `cd ABSOLUTEPATH` will move to some absolute path).
5. What's the absolute path of your new directory? What's the path, relative to your home directory? Write these in your coversheet.

File creation & modification

It's really easy to make files in command line.

Leave a message

1. Go to the second directory you created, using `cd`.
2. Use `touch README.txt` to create a new file called `README.txt`.
3. If you use `cat README.txt` to print out the content of `README.txt`, you will see no output. `README.txt` is empty.
4. Let's fix this. `echo 'text'` will print `text` to `STDOUT`, your terminal's default output. You can redirect content from `STDOUT` to a file by following a command with `> FILENAME`. Use this **input redirection** to leave a helpful message in `README.txt`.
5. Make sure your message got written by using `cat` to print out the content of `README.txt`

Let's duplicate our data because no one actually reads README*s*.

1. Use `man COMMANDNAME` to learn how `cp` works. You don't need any arguments with dashes, and you should be able to figure it out on the first page. Of course, `j` and `k` browse up and down the helpfile, and `q` will return to treminal.
2. Copy `README.txt` to `PLEASE_README.txt`.
3. Are they the same? Verify using `cat`.
4. Append some text (`>>` instead of `>`) saying that people should read the real `README.txt` next time.

Pass the data along.

I want to show you how to pass content from one program to another.

1. `history` shows a list of all your most recent commands. Try it now!
2. Since its output changes every time you use it, output it to a file. I called mine `history.txt`.
3. You can **pipe** output from one command into another command as its input. For example, `cat history.txt` will output the contents of `history.txt` to `STDOUT`, but if I pipe it to `grep` (`cat history.txt | grep 'man'`), it sends the file's contents as input to `grep`, a search function. Now `STDOUT` only shows lines that have `man` in them; lines where you used the manual command.

   How many times did you use the `man` command today? Use `wc` (and the manual entry for it) to count the number of lines that `grep` returns. You will need to pipe twice, once to grep, and once to `wc`. You can do this in the same line. Write the number you get in your `cover_page.txt`.

Get a coversheet

1. Use `curl` to download the cover page for this assignment (`https://raw.githubusercontent.com/citelao/102R/master/Lecture%204a%20-%20Command%20Line%20Can%20Be%20Not%20Terrifying%20and%20Source%20Control%20for%20the%20Skeptical/cover-page.txt`). Look up the documentation using `man`. As before, you shouldn't need any special arguments.
2. Save it to a file `cover-page.txt` in the first directory you created (the parent directory).

Mark it up

File editing in UNIX is really hard. Like, it sucks until you get good.

There are a couple editors, and all of them are weird. Choose one for this next bit. I like `vi`, but I also think it's cool that `man 7 ascii` gives you a list of the ASCII characters and spend my free time reading the Stack Overflow newsletter.

1. Read the guide to your new favorite text editor on the 330 wiki.
2. Answer all questions you can in the `cover-page.txt`. You can edit a specific file by passing the filename as the argument when you open your editor, so `vi cover-page.txt`, for example.

Source Control

Time to get your stuff up on the internet.

Checkout the code

You need to checkout your code into a local copy so you can make edits.

1. In your home directory, create a directory to hold your SVN repos. I called mine `svn/` because I'm creative.
2. Browse into that directory, and then type `svn checkout WUSTLUSERNAME@svn.seas.wustl.edu/repositories/WUSTLUSERNAME/cse132_fl15/`.
3. Enter your WUSTL password when prompted. Do **not** save it unencrypted. That would be bad.
4. `cd` to the new SVN directory. It should have `arduino/` and `java/` folders.

Realize what log messages are for

1. Use `svn log` to list the commit messages to your repo.
2. This should be enough to indicate why descriptive commit messages are good :)

   Don't worry, some of my recent commits:

   - "autograder is salty"
   - "i guess there's a typo?"
   - "spaces always getcha"
   - "Now with real non-hacky things"
3. Write your funniest commit message on the cover page.

Move your code to a new home

It's time to move the directories you created earlier into your repo.

1. Make a new directory next to `arduino/` and `java/` called `shell/`.
2. Figure out the `mv` command and move the directories you created earlier into that `shell/` directory. Again, you need no special arguments to do this.

Sputnik Niner-Niner, what's your status?

1. Within the local copy of your SVN repo, type `svn status` to see the status of your local copy. You should see your `shell/` directory with a question mark before it.

   If you committed now, you would not push your new directory to the server. You need to tell SVN to track that new directory.
2. `svn add shell` to track that new directory.
3. `svn status` should now show all the files in the `shell/` directory, with `A`s next to them.

Get it online!

1. `svn commit -m "YOUR MESSAGE HERE"`
2. Make sure your code got pushed by going to your repository in a browser.

You should have a `shell/` directory with:

- a subdirectory
  - `README.txt`
  - `PLEASE_README.txt`
- `cover-page.txt`

Once that's done, you're ready to get checked out. Congratulations! You know bare-bones command line!