

A Quick Overview of COP 2341 – Linux Shell Scripting Fall, 2013, Jerry Reed

What is this document?

This is not the syllabus – that's coming. That's long, and full of details about grading, rules and so forth. This is just an informal FAQ to help you get an overview of what the class will be like.

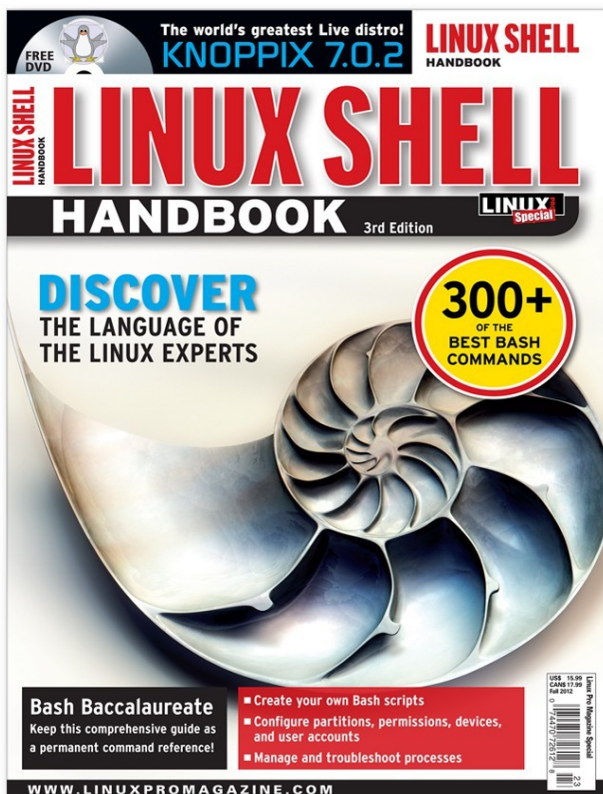
Please feel free to e-mail me with any questions or comments.

What text are we using?

There is NO REQUIRED textbook for this class. All of the information you will need to succeed is available on-line, mostly from sites on the Internet you find via Google and from on-line discussions with your classmates, and from asking me questions.

That said, I STRONGLY RECOMMEND that you purchase this:

<http://shop.linuxnewmedia.com/us/magazines/special-editions/eh32022.html>



It's a special edition of *Linux Pro Magazine* that does a really excellent job of covering almost all of what you need to know for this class. The articles are short, by textbook standards, and that means they can be challenging to read, but they get right to the important material immediately. At \$15, it's a bargain.

This edition might still be available over the counter at larger bookstores, but you can always order it on-line. It comes bundled with a DVD of one Linux distribution, but you probably want to check out the section below on “What are my Linux options?” below before installing.

A free sample article is here:

http://www.linuxpromagazine.com/content/download/70475/581816/version/1/file/090-095_scripting_SE10.pdf

and another here:

http://www.linuxpromagazine.com/content/download/70476/581819/version/1/file/009-011_filemanagement.pdf

What is Shell scripting?

Shell scripting is like learning a specialized programming language. The “shell” is just a buzzword for a command-line programming language also known as Bash. This language is built-in to almost every Linux system, and also runs on OSX and can even be added to Windows (with Cygwin).

You know about if and while and variables from your other programming classes, but shell scripting adds powerful Linux commands and utilities to these basic programming constructs. The result is that you can write really short programs (“scripts”) that accomplish a lot more than a much longer program in C or Java.

In some cases, you can connect the output of one command to the input of the next, forming a pipeline that does four or five things in the same line. You can save the command combinations you use in files and run them whenever you need, or even schedule them to run unattended.

The available commands run from simple, like cat, which copies information from a source to a destination, to full-fledged programming languages, like awk, which is a C-like language for processing and parsing data from files. Scripts assemble combinations of these commands together and allow iterating (repeating) commands, or executing them conditionally (if). The result can be a very powerful “program” of twenty or so lines that automates a complex task and that would take literally hundreds of lines of C or Java.

What will we do in this class?

The best way to learn about the shell commands and using them in shell scripts is to experiment with developing some useful and entertaining scripts yourself. Towards that end, a lot of your time will be spent working on projects I assign (or that you come up with).

Here's a hypothetical example of a moderately simple project:

Here's the output from some Bash command:

```
total 228
-rwxr-xr-x 1 jerry jerry    130 2007-06-01 21:57 buildLuaExt.sh
-rw-r--r-- 1 jerry jerry    207 2007-06-06 09:13 collapse.awk
-rw-r--r-- 1 jerry jerry   1515 2007-06-06 15:12 luabridge.h
-rw-r--r-- 1 jerry jerry  15350 2007-06-06 15:14 luabridge.c
-rw-r--r-- 1 jerry jerry   2296 2007-06-07 19:35 avg.lua
drwxrwxr-x 2 jerry jerry   4096 2012-08-27 13:52 cop2800
-rw-rw-r-- 1 jerry jerry  10638 2013-02-19 09:01 Untitled 1.odt
-rw-rw-r-- 1 jerry jerry  13084 2013-02-19 09:28 project_directions2.odt
-rw-rw-r-- 1 jerry jerry     80 2013-03-13 08:53 p3_input_file.txt~
-rw-rw-r-- 1 jerry jerry     74 2013-03-13 09:04 p3_input_file.txt
-rw-rw-r-- 1 jerry jerry    720 2013-03-13 09:05 sample_scanf.c~
-rw-rw-r-- 1 jerry jerry   1253 2013-03-13 09:08 sample_scanf.c
-rwxrwxr-x 1 jerry jerry   7449 2013-03-13 09:08 a.out
-rw-rw-r-- 1 jerry jerry 124366 2013-04-10 12:00 kern-1.log
drwxrwxr-x 2 jerry jerry   4096 2013-04-24 13:17 cts2321
drwxrwxr-x 2 jerry jerry   4096 2013-04-25 16:25 cop2220
drwxrwxr-x 2 jerry jerry   4096 2013-04-26 16:11 cop2224
drwxrwxr-x 2 jerry jerry   4096 2013-05-28 10:33 cop2341
```

Please answer the following questions and provide links to resources you used in answering these questions?

- What command produced this output?
- Who owns all the files here? (easy).
- Note the order of the files (newest file last, oldest file last). What command flags were used to do this? Which of these entries are directories (like folders on Windows)? How can you tell?
- Which file is the largest? The smallest?

Here's a broad hint: “the command you need starts with an l (lowercase lletter el)”.

Submit your answers as an assignment, and then post the links you used in the discussion forum set up for that purpose. (That way you are helping others who might be “stuck”).

Here's a hypothetical example of an advanced project:

Ping any network host you like, no more than once a minute. Log the responses, and produce a script that can be run daily to graph the resulting network delays. You might like to use gnuplot to produce the graph.

After you submit this assignment, please post your graphical output, along with any links you used to figure all this out, in the designated Discussion Forum in Blackboard.

Here's the typical workflow for a project like this:

1. Find out what *ping* does (Google it, example: <http://www.wikihow.com/Ping-in-Linux>)
2. Find out about *gnuplot* (Google it, example: <http://gnuplot.sourceforge.net/>)
3. Try ping out on the command line to see the results.
4. Try some gnuplot examples.

5. Plan your script. (Where will I put the output from ping? How will I run gnuplot daily?)
6. Experiment and develop your script by running the needed commands directly.
7. Look at automating this with cron (Google again, example: <http://www.thegeekstuff.com/2009/06/15-practical-crontab-examples/>).
8. Submit your script on the relevant assignment in Blackboard
9. Post the links and output in the designated Blackboard discussion forum.

Both #1 (ping) and #7 (cron) are covered in the recommended *Linux Pro Magazine* issue, too.

So we'll have projects due roughly weekly, plus discussion postings and two exams.

I'll post some of my own material and examples, along with “starter” links and a few videos, but most of what you need to know you'll find from the web, or the magazine. There are tons of videos about Linux shell on YouTube, and so my videos won't duplicate those so much as provide brief examples of my particular point of view (and biases :-)) about Linux.

Where can I go for help if I get stuck or just have questions?

You can ask a question in the Questions and Answers discussion forum (best), or e-mail me, or come to the West Campus IT lab.

Asking in the Questions and Answers Blackboard forum is the best option because:

Discussion Board
 Forums are made up of individual discussion threads that can be organized around a particular subject. Create Forums to organize discussions.
[More Help](#)

Create Forum Search ↑

Forum	Description	Total Posts	Unread Posts	Total Participants
<input type="checkbox"/> Questions and Answers	Please post questions of general interest here so everyone can see and respond. You might want to subscribe to this forum so you are notified via your Atlas e-mail when someone posts or replies.	112	0	11

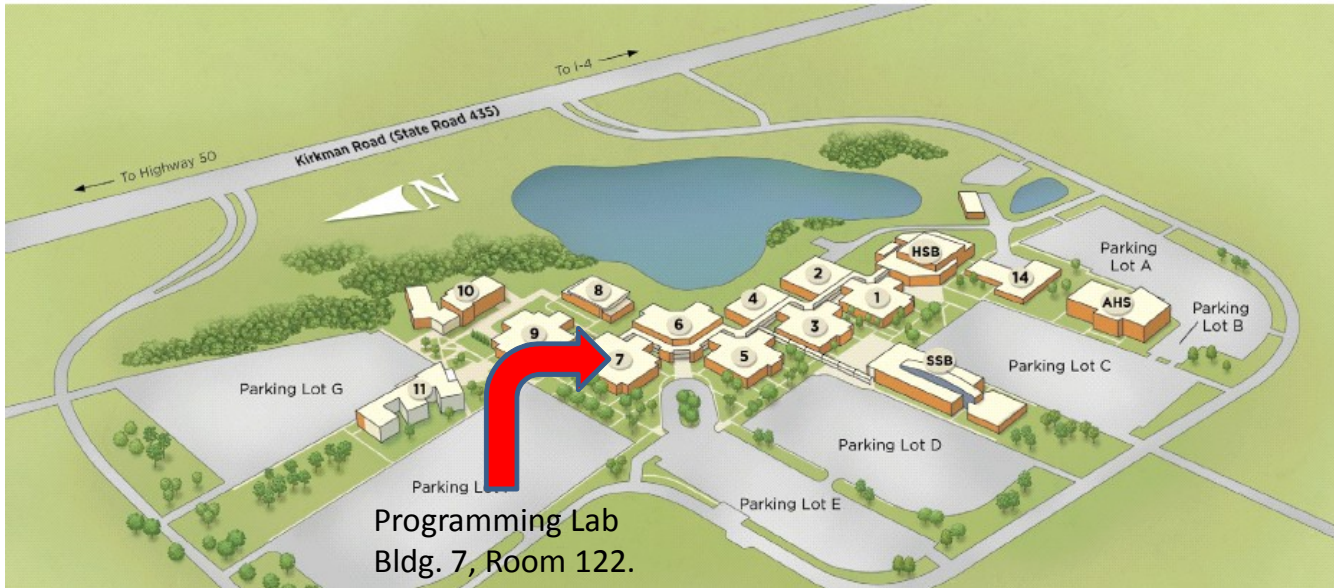
1. I subscribe to that forum, and you should too. That way as soon as something is posted, we'll get an e-mail with the post inside. Those go to my phone. I can usually respond to these within a couple of hours, between about 8:00 AM and 10:00 PM. *Please do not use the Messages facility in Blackboard (I try to hide it) because it does notify me and I will forget to check it in a timely fashion.*

2. Since there are almost certain to be a few folks in our class who already know a fair amount about Linux, or who have already dealt with similar issues in their own work, I'm not the only one who may answer and provide help. Interaction with other folks on-line is a crucial part of this class.

3. You may find that your question has already been asked and answered already. If you see a question where you can contribute, please jump in and don't wait for me, or feel free to correct or extend answers from others and from me.

E-mailing me is fine, too, (greed9@valenciacollege.edu) but is best reserved for questions that you don't feel comfortable sharing more publicly, such as questions about your grades or progress in the class. I check pretty regularly, but not so much on weekends and nights, so the post with a push to my phone works much faster, in general.

Coming to the lab is logistically difficult, I know but for some problems, there is no substitute for hands-on assistance. The lab staff can help with the majority of shell scripting problems, and they can consult me if needed as well. This is particularly useful if you are having trouble getting your Linux system set up initially.



<http://valenciacollege.edu/west/engineering/LabHours.cfm>

What can I learn in this class?

Please see the attached *Learning Outcomes* document. It has a number of links to related pages, although some may be broken at the moment.

What are my options for a Linux environment?

The good news is that there are more and more options for installing Linux every day, it seems. The bad news is that you need to decide how you want to proceed.

I. If you have the memory and CPU, and the patience for a bit more complex installation, then the absolutely most versatile way to go is to create a *Linux Virtual Machine (VM)* using software such as Virtual Box from Oracle:

<http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html#vbox>

and

<http://www.virtualbox.org/manual/ch02.html>

or the non-open-source (but still free and robust) VMware:

<http://www.vmware.com/products/player/overview.html>

Advantages:

“Bullet-proof”. If you take snapshots at regular intervals, then no matter how badly you mess up your virtual machine, you can always revert to a prior state and start over. Now it is unlikely that you'll mess up your Linux install in this class, but for the Linux System Administration class (CTS 2321), where there is a lot of configuring and experimenting expected, we always push this option.

“Portable”. You can get Virtual Box for OSX, Linux and Windows, meaning that any of these three OS'es can host your virtual Linux installation. You can even export your VM and move to another type of host if needed. If you're running Windows 8, this may be your best option.

Disadvantages:

“Resource intensive”. You really do need at least 4 GB of RAM, and perhaps 50 GB of free hard disk space. Dual-core or greater class processors are strongly recommended. You can squeeze a small VM (say something like TinyCore, Slitaz or DSL) into 1 GB RAM and 2 GB of disk on a single-core processor, but you won't be happy with the performance and the compromises in features and ease of use you will need to make.

With a 4 GB machine, and dual-core processor, you can basically bisect the machine, with Windows getting 2 GB and one core, and the VM getting the other GB and the other core. The disk space is used to store the Linux file system, as well as to provide space to save the machine when it is shutdown or when you take a snapshot or create and export an appliance.

2. Bare-metal install. This means that you install Linux directly on a hard drive in your system, and then reboot from Windows into it. (Not easily applicable to OSX or to Windows 9 yet).

Advantages:

“Good performance”. Even on low-end hardware, say an aging laptop, a direct, bare-metal install of Linux will yield quite acceptable performance, like better than Windows on the same machine, if the resources are limited. Should work to dual-boot with Windows 8, or Windows 7.

Disadvantages:

“Install can be scary”. If you still intend to keep the current Windows installation on the computer, you will be asked to partition the disk during the install to allow Linux to co-exist and boot. This is quite safe, if you follow the instructions. If you rush through recklessly, you could clobber your Windows install :-).

3. Wubi installation. Simulates a dual-boot, bare-metal install but doesn't partition your hard drive, just puts all of Linux in one gigantic file on Windows.

<http://www.ubuntu.com/download/desktop/windows-installer>

Advantages:

“Easy installation”. Provided you're on Windows 7 (or XP), very easy installation, and subsequent uninstall if desired. No scary disk partitioning dialogs, since everything “lives” on the Windows side.

Disadvantages:

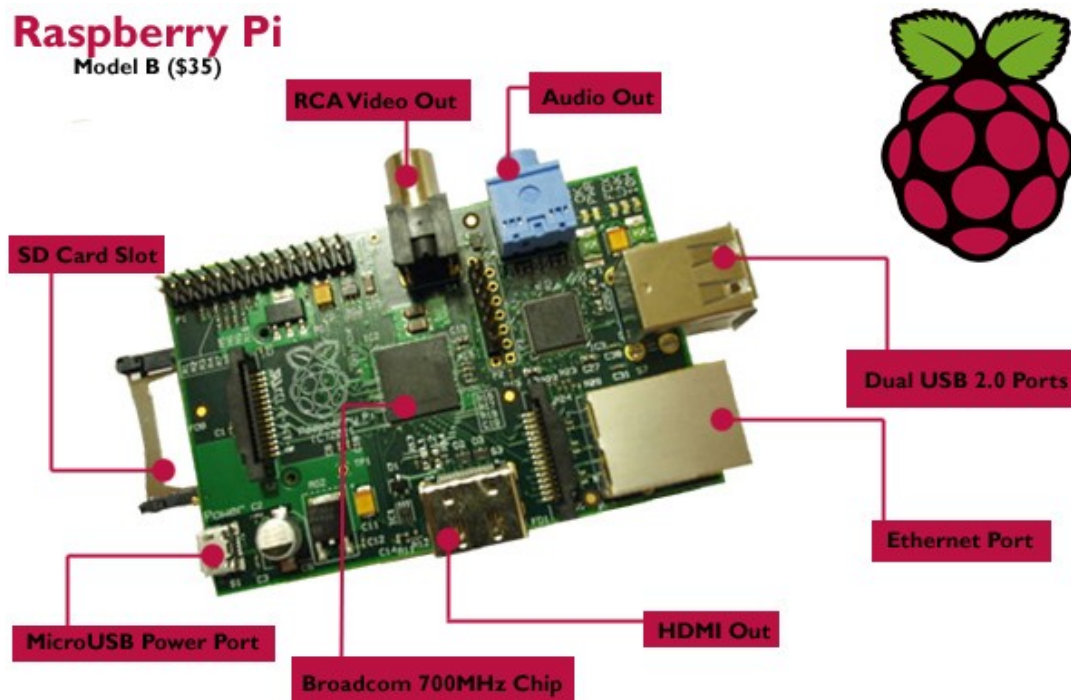
“Ok but not great performance”. Performance is slowed a bit by having to go through the Windows file system, relative to bare-metal, but still pretty good.

Doesn't work with Windows 8.

4. Get a Raspberry Pi: If you are the kind of person who is comfortable plugging together a bit of hardware, and maybe even has say an old USB keyboard and mouse laying around, you might want to invest the \$45 or so and pick yourself up your own tiny, dedicated Linux computer.

Raspberry Pi

Model B (\$35)



Wildly popular on the Internet, the Raspberry Pi is a 700 Mhz ARM processor (like your phone, likely) with 512 MB of RAM and an SD card for “disk”. It plugs into any HDMI-capable monitor or TV and runs recent versions of Linux, including of course, the Bash shell we need for this class.

<http://www.raspberrypi.org/quick-start-guide>

and

<http://www.engadget.com/2012/09/04/raspberry-pi-getting-started-guide-how-to/>

and

<http://www.designspark.com/knowledge-item/raspberry-pi---getting-started-guide>

and

a million others.

Advantages:

“Fun factor”: The Pi is everything some of us weren't in High School – cute and popular. :-) Your small investment buys a complete Linux computer that you can dedicate to this class, and subsequently to any one of a large number of projects like these you might decide to undertake:

<http://www.linuxuser.co.uk/features/amazing-raspberry-pi-projects-part-1>

and

http://www.reddit.com/r/AskReddit/comments/1f607z/owners_of_a_raspberry_pi_what_do_you_use_it_for/

Disadvantages:

“Total cost of ownership”. This is still a low total , but to make the Pi work, you need

- the board itself (\$45 or so, plus shipping),
- a USB keyboard,
- a USB mouse, a monitor or TV that accepts HDMI input,
- a SD card with a Linux distribution such as Raspbian on it,
- Ethernet cable,
- and a USB powered hub.

Total is closer to \$100 I expect, if you have to buy all these except the TV.

The SD card holds the Linux OS, and any files you create or install. You can buy one with Linux already installed:

http://www.adafruit.com/products/1121?gclid=CIGBuJ_K67cCFXRp7AodejUABA

or you can make your own like this from Windows if your PC/laptop has a slot for an SD card or you use an adapter:

http://elinux.org/RPi_Easy_SD_Card_Setup

The keyboard, mouse and TV/monitor are necessary to interact with the Pi during initial setup. After you have configured things initially and plugged the Pi into your router with an Ethernet cable, you can dispense with the borrowed TV, keyboard and Mouse, and interact with the Pi remotely, like this:

http://elinux.org/RPi_Remote_Access

The USB powered hub is needed to supply 5 volt current to the Pi, and to support plugging lots of other USB peripherals into the Pi if you need.

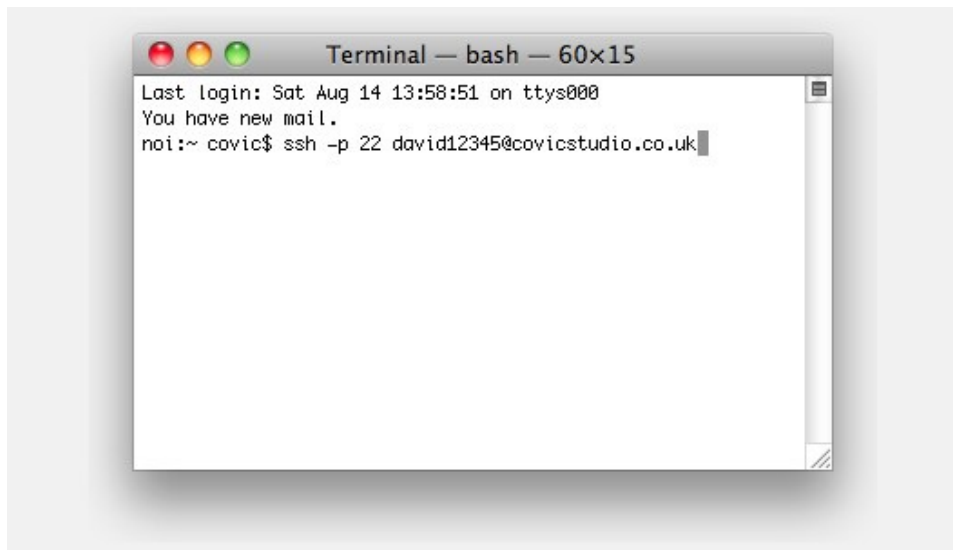
“Techy”: You should be comfortable plugging in cables, and the like to attempt this approach.

I have set up several Raspberry Pi's and I would be glad to provide advice as needed. Obviously if you're a PC-head who has lots of keyboards and cables laying around, you cost can be pretty much limited to the Pi itself.

Plus, we have several Raspberry Pi's in the Computer Programming and Applications Lab on West, so feel free to stop by and check out a demo if you like. Details on the lab hours and location are under the “Where can I go for help...” question above.

5. Something else: the Terminal app on OSX, Cygwin on Windows, remote access to cop2341.valenciacollege.edu via putty.

If you've got a Mac, in theory you can use the Terminal application there to develop Bash scripts.



<http://guides.macrumors.com/Terminal>

This will work and I have had at least one student complete this course successfully using this approach. You should be aware that OSX is not Linux, and although the two are quite similar internally, there are lots of little differences between the two when it comes to exact commands.

<http://stackoverflow.com/questions/8051145/is-the-terminal-in-mac-and-linux-the-same>

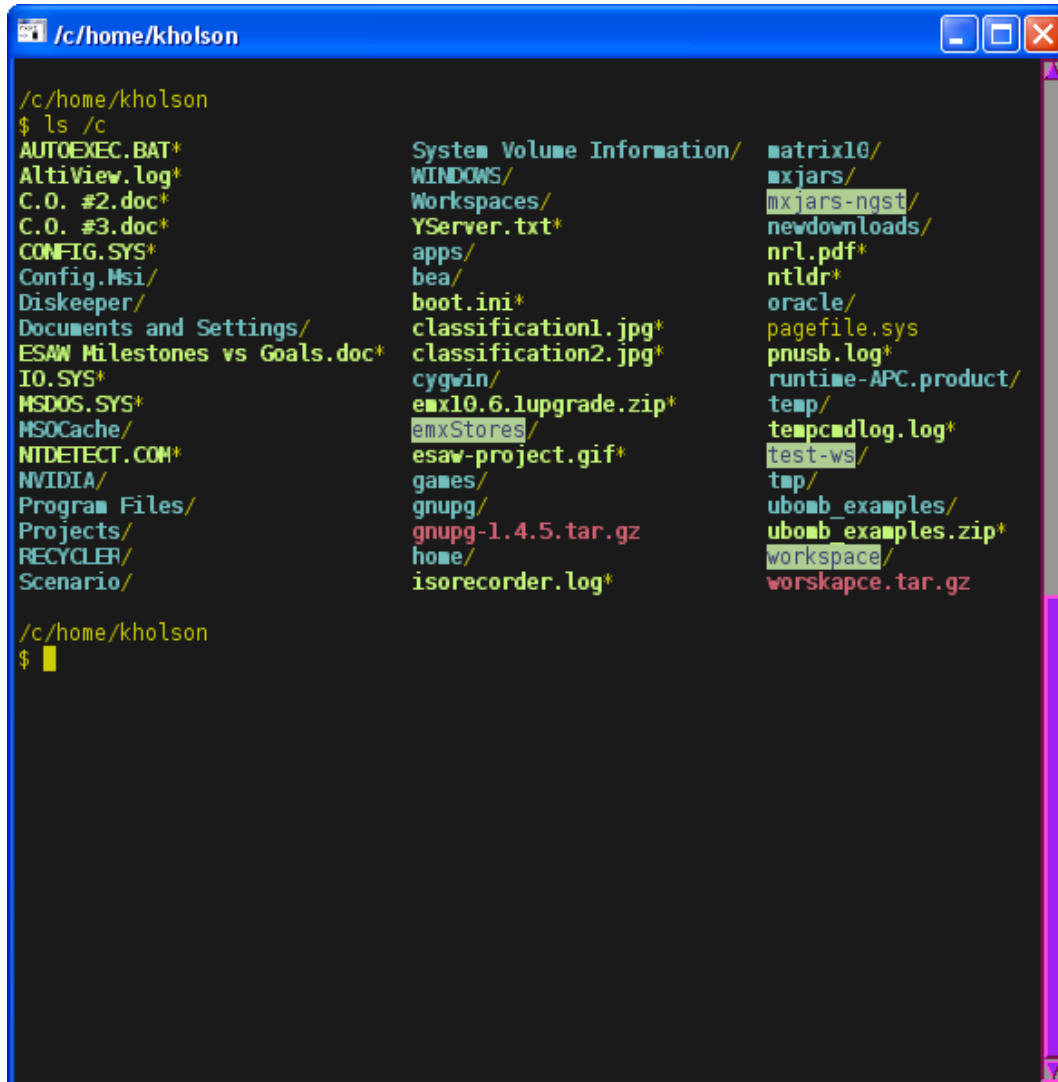
You may have to wrestle with the tendency of Mac users to embed spaces in file and directories, forcing a lot of quotes around things.

If you can work around this, and you have a modern (10.3.x) version of OSX, then you're set already.

If you really want to stick with Windows, and are willing to put up with some annoying, if small differences, then consider installing Cygwin.

<http://www.cygwin.com/install.html>

Cygwin is a POSIX compatibility layer for Windows (Google it). It provides a bash shell under Windows XP, 7 or 8. Almost all Linux commands and utilities are available and work identically to “real” Linux. I use it all the time in my work.



```
/c/home/kholson
$ ls /c
AUTOEXEC.BAT*          System Volume Information/  matrix10/
AltView.log*          WINDOWS/                   mxjars/
C.O. #2.doc*          Workspaces/                mxjars-ngst/
C.O. #3.doc*          YServer.txt*               newdownloads/
CONFIG.SYS*           apps/                      nrl.pdf*
Config.Msi/           bea/                       ntldr*
Diskeeper/            boot.ini*                  oracle/
Documents and Settings/ classification1.jpg*        pagefile.sys
ESAM Milestones vs Goals.doc* classification2.jpg*        pnuhb.log*
IO.SYS*               cygwin/                    runtime-APC.product/
MSDOS.SYS*            emx10.6.lupgrade.zip*     temp/
MSOCache/             emxStores/                 tempcmdlog.log*
NTDETECT.COM*        esaw-project.gif*         test-ws/
NVIDIA/              games/                     tap/
Program Files/        gnupg/                    ubomb_examples/
Projects/             gnupg-1.4.5.tar.gz        ubomb_examples.zip*
RECYCLER/             home/                      workspace/
Scenario/             isorecorder.log*          worskapce.tar.gz

/c/home/kholson
$
```

As with the OSX terminal, there are small and somewhat distracting differences in the way files are named and located, with the Windows backslash (\) giving way to the Linux forward slash (/) [C:\blah](#) blah becoming “/cygdrive/c/blah blah” and so on.

I've had two students try this in the past, and each switched to Linux half-way through the class, but I understand this class has been taught entirely using Cygwin before, so it is doable.

Last but not least, the college maintains a Linux machine that you can access remotely to do the work of this class. It's cop2341.valenciacollege.edu and is accessible via the Internet.

Using a program such as putty:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

and login credentials we supply, you could log into this college computer remotely from your Windows PC and create, edit and run Bash shell scripts. This is not the most user-friendly setup, but is workable and many students have used it successfully in the past. (For OSX you probably already have all the software you need to connect).

Please get in touch with me if you need details of this approach.