Week 02

More about the command line and shell scripting

Melvyn Ian Drag

January 27, 2020

Abstract

For tonight's class we'll learn more about the Linux command line and shell scripting.

Working on the Linux Command Line

You know some commands already:

1. cat
2. ls
3. cd
4. cp
5. cut
6. cut -c
7. cut -b
8. echo
9. touch
10. **What else?**

Here are some more commands that are useful:

1. sort
2. uniq
3. tee
4. history
5. grep

these commands are particularly useful in the context of **pipes**. Pipes are one of the defining features of Unix systems, they are one of the things that make unix great. A major tenet of Unix philosophy is that programs should be very simple and do one thing - just one simple thing - but do it incredibly efficiently and always get the correct result. Then these simple programs can be strung together to make more interesting programs that are fast and correct, because the tiny programs (like cat and sort and ls and cd, etc.) are fast and correct.

Pipes

Consider the following file:

```
10
1
2
3
4
8
11
91
871
78
11
-10
11
200
8000
10000
-2
-3
```

There are many things you can do with this file. By the way, I have it saved in my current working directory as 'numbers.txt' - you should do the same.

```
melvyn@laptop$ cat numbers.txt | sort
melvyn@laptop$ cat numbers.txt | sort -n
melvyn@laptop$ cat numbers.txt | sort -g
melvyn@laptop$ cat numbers.txt | sort | uniq
melvyn@laptop$ cat numbers.txt | uniq # note that this does not work.
```

So you see the idea of pipes? We used a few linux commands together, separated by a '—' to achieve a cool result. I showed you above a few ways to sort, and then the correct and the wrong way to find the uniq numbers in a file.

Let's continue with this discussion of pipes:

```
melvyn@laptop$ echo hello | cut -b1-2
melvyn@laptop$ echo hello | cut -c1-4
melvyn@laptop$ echo "hello world" | cut -d" " -f1
melvyn@laptop$ te how cut with d and f is finicky
melvyn@laptop$ echo "hello world" | cut -d" " -f2 # many spaces between hello and world.
melvyn@laptop$ default cut uses TAB ( \t ) as delimiter.
melvyn@laptop$ cat file | md5sum
melvyn@laptop$ md5sum file
melvyn@laptop$ cat file | sha256
melvyn@laptop$ history | grep "that cool command I can only remember part of" # I do this at least 50
    times a day
```

...this bit about 'cut' is super important.... let's play with it more to make sure you understand it... heck... to make sure *I* understand it.

Spend 5 minutes playing with cut and pipes to extract various bits of information

1 7:20 - 7:30 About this class

runthroughthesyllabus

I'll not give 'bathroom breaks' in this class because it usually leads to the class getting derailed. Some folks don't have to pee and then get bored of 10 minutes of nothing, others decide to take a walk, then come back late having missed alot of material and then disturb the lecture, I get anxious sitting here doing nothing, etc. So I'll just punctuate lectures with periods of activities for you all to do and you can leave the class during those periods of time. Of course you can leave whenever, but I want to make sure you be able to see the light at the end of the tunnel if you have to pee - if you can hold it, there will be a break in 20 minutes or so, so you don't need to worry. Also I don't want you to sit for 3 hours listening to me talk.

We're going to use digital ocean servers in this class. I don't have time to show you one right now, next week we'll discuss it. This will either be free to you or at most cost $50 or so for the semester. It will probably be free though. More details next week. I'm not requiring a text book for this class so you've already saved about $150 right there, and I don't think you'll need to pay for these servers we'll use, but if you do it will be a tiny amount of money, especially compared to the crazy stuff I'm going to show you how to do with this cheap machine.

2 7:30 - 8:00 Getting started with the terminal in Linux

1. ls
2. ls -l
3. ls -ltr
4. ls -a
5. ls -la
6. cd
7. cd
8. cd .
9. cd ..
10. cd ../../
11. cp
12. cp -r
13. mv
14. mkdir
15. mkdir -p
16. touch
17. rm
18. rm -r
19. which

3 8:00 - 8:05 5 minutes to practice commands.

- practice the above commands
-

putsomestructuredactivitieshere

4 8:05 - 8:30 Bash commands part 2

1. cat $file1
2. cat $file2 $file3
3. echo “hello”
4. echo “my path $PATH”
5. man
6. wc
7. wc -c $fileName
8. wc -l
9. wc -w
10. md5sum
11. more
12. less
13. unzip
14. zip
15. tar
16. tar -xvzf
17. cut -b
18. cut -c
19. tree
20. lsblk (plug in a usb stick and see that it changes)
21. du - list disk usage globally
22. df - list disk usage by file
23. wget

Note about cut: cut -c is just like cut -b on every machine I’ve seen. In the future -b (byte) and -c (character) will be different. Ill give some insight on that later in the class, and if you know Java + C you will likely have a bit of insight on the definitions of byte and character, where they are the same, and where they are different. There is no time for that now, and if you already have insight, just keep it to yourself for now haha.

8:30 - 9:00 vim

4.1 8:30 - 8:35 Why learn vim?

As you’ve seen a major part of working on Linux is doing stuff on the command line. If you’re on your laptop, you might use other graphical tools. You can create a directory, for example, by right clicking and going to ’create directory’ or whatever as you do on windows, but you’ll often work on the command line, or create directories in your code, and things like ’mkdir’ are useful for that. If you are a web developer maintaining a website, youll often connect to a command line interface and do things there, because you won’t have a monitor in front of you. If you remotely connect to a computer, you’ll maybe need to edit a file. for this you’ll need a command line editor. Here’s your first dose of linux/unix/bsd culture! There is alot of playful fighting in the nix world about the best way to edit files on the command line. Mainly the war is around two editors, `https://en.wikipedia.org/wiki/Editor_war,emacsandvim`. I like vim. Some folks like emacs. I started on emacs a decade ago, but when I learned vim I found that I liked it more and I haven’t looked back in about 10 years. I’m going to teach you vim, and if you want to learn Emacs you’ll have to do it on your own time. If you get a job and have to edit a file on a server, and you fire up vim and edit it smoothly you’ll instantaneously become the celebrity of the office. You’ll be one of the cool cats. Emacs would also get you cool points.

But vim is featured in every standard linux distro, while emacs is not! Whatever, too much said already. We’re using vim for this class. There are simpler editors like pico and nano, but I know vim by heart so that’s what I use and teach!

8:35 - 8:40 Basic Vim workflow

Look at the stupid vim memes. They are dumb and frustrating. Memes in the foloder here. Memes about not knowing how to exit vim. So many ‘hackers’ want to complain that they can’t even remember the following:

To open a file with vim type

```
melvyn@computer$ ls
filename otherfilename dirname otherstuff
melvyn@computer$ vim filename
# vim will open a file with name 'filename'
# if the file exists, it will create the file.
```

Vim is a 'mode based' editor. Hit i to enter insert mode. Start typing as normal. Hit esc to get out of insert mode. Type :wq to save and quit. Thats 99% of what you'll do in this class.

8:40 - 9:00 More vim commands to make your work better.

- h, j, k, l
- I prefer to just use the arrow keys
- modes i, esc
- quit with :q. To save and quit you use :wq or :x. If you want to know the difference, google the diff between :wq and :x. there is a slight difference, but it doesnt matter.
- dd to delete a line
- y is copy.
- p is paste after the cursor
- shift p is paste before the cursor
- u is undo
- CTRL + r is to redo.

Here is some more functionality that is so powerful and unique to vim that it will make you feel scared. You've never had this power before and this is going to make you think vim is hard. It's not. You can ignore the following for now, but within 2 weeks when you've mastered the above you'll be ready to appreciate how great the below commands are. I'm just showing you now to plan the seed so that it can start to develop in your subconscious.

- w to go forward a word
- e to go to the end of the next word
- b to go to the previous word beginning.
- to go forward 5 words, 5w
- To go back 5 words, 5b.

9:00 - 9:05 Experiment with Vim

Now is your time to try allthe things I've shown you and ask me questions if you don't get it.

9:05 - 9:10 Bash Part 3

Return value of last command is : $?. All these linux commands I've shown you return information to the terminal. They tell the terminal if the command was successful or not. To check the return code you type 'echo $?'.

```
melvyn@machine$ echo "hello world"
hello world
melvyn@machine$? echo $?
0
melvyn@machine$ ehco "aksljdg"
# error
melvyn@machine$ echo $?
127
#the 127 indicates an error. There was an error because ehco is not a command,
echo is.
```

5 9:10 - 9:30 More vim and the inevitable questions

The coupe de gras. Registers in vim. Vim remembers the things you've yanked in the past. Type :reg See that vim remembers the stuff you've put on your clipboard in the past. Exercise Type: Hello World Foo Bar Baz dd Hello dd World highlight and yank the other three Then look at clip board. Paste world Paste Baz Foo Bar Baz Look at :reg and see what it remembers. I think it remembers the dd stuff but only remembers the latest yanked stuff. Paste the things from your registers with ¡reg id¿p e.g. (when in normal mode, not insert mode) "1p, see what happens.

9:30 - 9:45 Homework Discussion and further questions and answers

Look at and discuss homework.