Week02 - Pipes, I/O Redirection, and a Few More Tools

January 22, 2021

This week we'll look a couple of quintessential parts of Unix scripting, which are pipes and I/O (input/output) redirection. Then we'll spend a few minutes looking a few command line programs we were'nt able to cover last week.

---

1 Setup

Turn on a Debian 10 server and connect with ssh.

After the class you can try to go through these notes with your Macbook, with the gitbash terminal you installed on your PC, or with a different Linux/BSD distribution. You'll find most (if not all) of the things we do will work just as well on those platforms.

2 Learn as we go

I said I'd teach you a few new commands tonight. We'll just learn them as we go. You'll learn to use:

- history
- grep
- head
- tail
- cut
- sort
- uniq

3 Pipes

Pipes are one of the famous/important/fantastic command line tools you use on Linux. They allow you to create beautiful little programs by stringing together the basic programs you know like "cat", "echo", "cut", etc..

The pipe symbol is "| ".

You put this between the different commands you want to run to cause the data to flow through them like water through a pipe.

https://www.youtube.com/watch?v=bKzonnwoR2ILater you could watch this video of Brian Kernighan talking about pipes. He's one of the creators of many of the things you are learning this semester.

Here are a few examples in no particular order.

3.1 Searching your history.

history - this commands tells you all the things you've typed into your terminal. Go ahead and try it! Type history and you'll see all the things you previously typed.

grep - this cool program is for searching.

You might rename a PNG image to a new name. Now you can't remember what you renamed it to. So use a pipe!

```
root@machine$ history
# all of the things you typed
root@machine$ history | grep "mv"
# all of the mv commands you typed
root@machine$ history | grep "mv" | grep "png"
# all of the mv commands you typed that also contain the
    letters "png"
```

I often forget the ip address of the last machine I ssh-ed into. Use the same trick with history, grep and a pipe!

```
root@machine$ history | grep ssh
# all of the old ssh connections I made
```

3.2 Count the letters in the second word

Maybe you want to know how many letters are in the second word of a sentence.

```
root@machine$ sentence="the quick brown fox jumps over the
    lazy dog"
root@machine$ echo $sentence
the quick brown fox jumps over the lazy dog
root@machine$ echo $sentence | cut -d" " -f2 | wc -c
6
```

Actually we know the word "quick" has 5 letters - don't forget that cut appends a "
n" character.

In the above example we also learned a little bit more about "cut". "cut" takes a '-d' flag, which means 'delimiter'. In this case our delimiter was " ". It also takes an '-f' flag, which means 'field'. We cut the sentence on spaces, and grabbed the second one.

3.3 How many words on the third line?

Assume you want to know how many words are in the third line of a text file.

```
root@machine$ cat sentences.txt
whose woods these are I think I know
his house is in the valley though
he will not see me stopping here
to watch his woods fill up with snow
root@machine$ head -n3 | tail -n1 | wc -w
7
```

The head command gives us the first three lines. From there we use the tail command to trim off just the last line. Then we use wordcount to count the words.

3.4 What is the largest number in Column 2?

Assume we have a CSV file (Comma Separated Values) and for some reason we want to know the largest value in column 2.

```
root@machine$ cat data.csv
1,2,3,4
4,5,6,7
7,8,9,0
4,4,4,4
root@machine$ cut -d "," -f2 data.csv | sort -gr | head -n1
8
```

In this example we use cut to extract the second column. Then we pass the second column on to the sort program and we sort it in general numeric order (g) and descending (r), and then we pass that sorted data on to 'head' and take the first line. This gives us the largest number.

3.5 How many unique numbers in numbers.txt?

The uniq command takes a bunch of lines of sorted data and outputs just the unique elements. The input must must must be sorted

```
root@machine$ cat numbers.txt
1
2
3
2
1
root@machine$ uniq numbers.txt
1
2
3
2
1
```

See? It doesn't work. Make sure to sort first.

```
root@machine$ cat numbers.txt
1
2
```

```
3
2
1
root@machine$ sort -g numbers.txt | uniq
1
2
3
```

3.6 The Message

All of the above examples show that we can quickly write useful little programs on the command line by just piping data through the basic programs we already know.

This is a major part of Unix philosophy.

We have a bunch of simple, seemingly boring tools - but then we glue them together into a more meaningful program. And it is easy!

How long would it take you to write, compile and run a Java program to read a .csv file, find the largest value in a column, and then print the output? And how much typing would you have to do?

4 IO Redirection

```
1 public void run_exam(){
2   System.out.println("the function body goes here");
3 }
```