Unix Tutorial 7: Piping Commands

Today, we are going to practise chaining together simple commands using the Unix pipe facility, which is written as a vertical bar i.e. |. Most of our pipelines will have the following shape—there is a 'source' command that generates some textual output, which is then filtered by a number of other commands. In this tutorial, we will work through two examples.

Counting Word Occurrences

Download a textfile containing *The Adventures of Sherlock Holmes* from the web, using this command[1].

```
$ curl -O http://www.gutenberg.org/files/1661/1661-0.txt
$ ls *.txt
```

Notice that a file `1661-0.txt` has been downloaded to your filespace, into your working directory. You may inspect the contents of file using the less command:

```
$ less 1661-0.txt
```

Press space bar to scroll down one screen and b to scroll back one screen. Press q to quit the less program.

Sometimes, you will want to look at just the first (or last) few lines of a long textfile, e.g. a log file. Use the head (or tail) command for this:

```
$ head -n 10 1661-0.txt
$ tail -n 10 1661-0.txt
```

To print the whole text file out to the console output, use the cat command as follows:

```
$ cat 1661-0.txt
```

We are going to use cat as the source command in our pipeline. Suppose we want to find all the instances where Holmes says, 'My dear Watson.' Let's build a pipeline that searches for this phrase in our text file:

```
$ cat 1661-0.txt | grep -i ”my dear watson”
```

When you run this command, it should print out three lines which are the three occasions when Sherlock utters his memorable phrase to his friend. Notice the | is the pipe operator, sending the output from the cat command to the grep command directly. Notice the `-i` flag for grep, which makes the search case-insensitive. There are plenty of other useful flags for grep. You might want to

[1] Sorry, you will have to type in this command rather than cutting+pasting. The PDF version of this file uses different character encodings for – and other characters, that the console will not recognize.

print out a line or two of context before each quotation, or the position in the source file where each expression occurs. Try this command:

```
$ cat 1661-0.txt | grep -inC3 "my dear watson"
```

which will print line numbers for the matching lines, and include 3 lines of context each side of the matching line. Notice that we can run together the flags after a single dash—this saves typing `-i -n -C3` as separate flags.

Here are some more search terms for you—try looking for **elementary** and **cocaine**. How many times does each word occur in the text file?

Counting Word Occurrences

For a second exercise, we are going to find out the current temperature in Glasgow from the Met Office website. We fetch the Glasgow weather page with a curl command:

```
$ curl https://www.metoffice.gov.uk/weather/forecast/gcuvz3bch
```

This command prints out the whole HTML webpage to the console standard output. We want to filter this command, so it just prints out the relevant information. Let's use the grep command to search for the latest observation of weather[2].

```
$ curl -s https://www.metoffice.gov.uk/
weather/forecast/gcuvz3bch | grep tempText |
head -n 1
```

Note the `-s` flag to curl makes it operate *silently* so it does not print out its progress information as it downloads the data. The `grep` command prints out matching lines for the tempText paragraph class.

Now we need to use some search-and-replace commands (`sed`) to strip out the HTML tags, with their angle-bracket start and end characters:

```
$ curl -s https://www.metoffice.gov.uk/
weather/forecast/gcuvz3bch | grep tempText |
head -n 1 | sed -e 's/<[^>]*>/ /g'
```

This returns the headline weather forecast for Glasgow.

[2]This is a long command, so it may spread over multiple lines, but you can carry on typing directly on a single line that gets wrapped by the console.

Finally, let's save the weather report to a textfile called glasgowtemp.txt. We can use the > operator to redirect the output of the pipeline to a file.

```
curl -s https://www.metoffice.gov.uk/
weather/forecast/gcuvz3bch | grep tempText
| head -n 1 | sed -e 's/<[^>]*>/ /g' >
glasgowtemp.txt
```

Some more challenges:

1. (various approaches possible) Adapt this pipeline to print out the temperature in London.

Further Reading

For further investigation today, use the man command to find out the flags for all the commands you have already executed.

There are lots of grep tutorials online, e.g. see `http://www.thegeekstuff.com/2009/03/15-practical-unix-grep-command-examples/`.