COURSEWARE

Pipes and Filters

Contents

Overview

A **pipe** connects two commands together, taking the `stdout` of the first command and passing it as an `stdin` for the second command.

A **filter** reads data from the stdin and uses it to write a stdout.

These are very useful things to use together as we shall see in this module.

Pipes

A Quick Recap

Recall:

- **stdin** is short for `standard input`, and is what a process takes in.
- **stdout** is short for `standard output`, and is what a process gives out.
- **stderr** is short for `standard error`, and is what the process gives out if there's an error with the process.

About Pipes

Currently, if you want to take the stdout of one command, and then run another command on that stdout, you would need to create a temporary file to store the stdout.
That can mean that a process takes a long time if you have to copy a lot of data to a file each time.
Also if you forget to delete the files, your storage will fill up very quickly.

Pipes provide a solution to these problems as it does not create any temporary files.
This reduces the number of in/out operations required, reducing the chance of mistakes and increasing the speed of execution.

This is easiest to see in an example.
Assume we want to see how many users are currenly logged on to the system.
This can be done by running the command `who`, and then seeing how many entries there are.
Currently, that process would look like this:

```
$ who > /tmp/tmpWho
$ wc -l /tmp/tmpWho
$ rm /tmp/tmpWho
```

This a lot of commands to run every time you want to do this.

With **pipes** we can get the same output in one command:

```
$ who | wc -l
```

This is much easier

We can also combine pipes together as many times as we like, these are called **Multistage pipes**.

So if we wanted to see how many users are logged in through the character interface (tty), we could run

```
$ who | grep tty | wc -l
```

Multistage pipes are useful, but if you are using more than two or three, then maybe consider writing a script instead.

Filters

This section discussing some useful filters that we can use in conjunction with pipes to sort through large amounts of data, quickly.

tee

The `tee` command is similar to the `cat` command, except that it can save the stdin to a file before passing it to stdout.

```
$ who | tee -a users.list | wc -l
```

`tee` receives the stdin, appends that stdin to a list called *users.list* then passes the stdin to stdout to continue along the pipe.

grep

The `grep` command searches for a specified pattern in some text.

```
$ grep -v john /etc/passwd
```

`grep` searches for all occurrences of 'john' in the file */etc/passwd*, the `-v` option (verbose) then displays all of the entries except the ones found by `grep`.

sort

Performs a sort on a datastream.

```
$ sort -t: -k1 /etc/passwd
```

- `-t` defines what character is used to separate different fields.
- `-k` specifies which field to start with

uniq

Looks for duplicated lines of data within a stream.

```
$ cut -d: -f7 /etc/passwd | uniq -d
```

In the */etc/passwd* file, the seventh part of each entry references the default command shell for that user.
Adding the filter `uniq -d` shows one copy of all the duplicated shells in that file.

tr

The translate, `tr`, command lets you translate one set of characters to another.

```
$ tr 'a-z' 'A-Z' < file
```

Translates everything in *file* to upper case.

Tutorial

We will practice using pipes and filters with a text file.
Create a `.txt` file called *country.txt*.
Add to the file the following text:

```
england,london,gbp,english

usa,washington,usd,english

china,beijing,rnb,chinese

germany,berlin,euro,german

france,paris,euro,french

italy,rome,euro,italian

canada,ottawa,cad,english/French
```

Using `grep` and `cut` we can show which countries speak english, and all languages they speak.

```
$ cut -d, -f1,4 country.txt | grep english
```

Using `sort` we can sort the countries by the languages they speak in alphabetical order.

```
$ sort -t, -k4 country.txt | cut -d, -f1,4
```

Using `cut`, `sort` and `uniq`, we can find all the different currencies used.

```
$ cut -d, -f3 country.txt | sort -k2 | uniq
```

Exercises

On the same data set, try to manipulate the data in the following ways:

- Get all the countries which do not speak english, and put them in to a file.
- Find which country has the capital city 'rome'.