

Searching Text Files with `grep`

Digital Fishing for Text

Scholars in the Middle Ages would have sold their souls for the literary treasures that abound on the Internet today. Now they simply clutter up our hard disks a result of indiscriminate downloading. Just how do you go about finding a passage of text in your digital datastore? The Shell command, `grep`, can help you hunt down that elusive quote.

BY ELISABETH BAUER

If you can't remember things, you should at least put them on your computer." That is not such a bad idea, but in contrast to human memory, which will normally retrieve information stored in it, quite reliably – with the exception of final exams of course – it is not always that easy to find information on a hard disk. If you forget the filename and where you stored the file, you might find yourself wasting a lot of time searching through directories. Even if you know exactly which file contains information you are looking for, this may be of little use in the case of longer text files.

The shell command, `grep`, which locates text patterns within texts, is useful in both cases. In the most simple case, you call `grep` with the search key and the file to be searched. `grep` will respond by outputting all the lines in the specified file that contain the search key.

Imagine you wanted to search the King James Bible for instances of "Garden of Eden". To do so you would type

```
grep Eden bible.txt
```

in the shell, and `grep` would output the appropriate passages from the Bible. If



the search key contains spaces, you need to enclose it within quotes. You need to take some care and watch out for pitfalls with special characters: `*`, `?`, and `!` have a special meaning for the shell. Another group of characters (`.`, `*`, `^`, `$`, and `\`) is not interpreted at face value by `grep`. Instead, the tool will assume a regular expression. The upside is that you can construct powerful and complex queries, although you might want to avoid using these characters with `grep` until you become more accustomed to the tool.

If you are not sure which file contains a text passage you are looking for, you can call `grep` with a wildcard (`*`). The Gutenberg Project's version of Herman Melville's "Moby Dick" comprises of a collection of text files.

```
grep white moby.*
```

will show all the occurrences of the word white in Melville's whaling masterpiece (see Figure 1). The asterisk means any letters. The shell will replace this expression with the names of any files in the current directory with the `moby.` prefix.

If the files you want to search for is distributed across multiple directories,

you can use the `-r` option to tell `grep` to search a folder recursively:

```
grep -r white Melville/works/
```

will search the `works` folder and any sub-directories below it for `white`.

The Trio – `ps`, `grep`, and `kill`

`grep` is not only useful for philosophical and theological text searching, but can also be used in combination with other shell commands. If a command produces a lot of text output, you can use the command followed by a pipe character (`|`) and `grep searchkey` to filter the output, retaining only the parts you are interested in. A typical case where you would

Grep Overview

Command	Action
<code>grep pattern file</code>	searches <code>file</code> for <code>pattern</code>
<code>grep pattern *.htm</code>	searches all files in the current directory that end with the <code>.htm</code> suffix
<code>grep -r pattern folder</code>	performs a recursive search in <code>folder</code> and its subdirectories
<code>grep -i pattern file</code>	ignores case
<code>grep -An</code>	outputs <code>n</code> lines after the line containing the searchkey

```

john@black.box: /home/john/Melville - Shell - Konsole
Session Edit View Settings Help

[john@black Melville]$ wget http://www.ibiblio.org/gutenberg/etext91/moby.zip
--09:58:21--  http://www.ibiblio.org/gutenberg/etext91/moby.zip
=> 'moby.zip'
Resolving www.ibiblio.org... done.
Connecting to www.ibiblio.org[152.2.210.81]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 606,033 [application/zip]

100%[=====] 606,033 59.22K/s ETA 00:00

09:58:32 (59.22 KB/s) - 'moby.zip' saved [606033/606033]

[john@black Melville]$ unzip -q moby.zip
[john@black Melville]$ ls
moby.0  moby.11  moby.121  moby.134  moby.25  moby.37  moby.49  moby.60  moby.73  moby.85  moby.97
moby.1  moby.110  moby.123  moby.14  moby.26  moby.38  moby.5  moby.61  moby.74  moby.86  moby.98
moby.10  moby.111  moby.124  moby.15  moby.27  moby.39  moby.50  moby.62  moby.75  moby.87  moby.99
moby.100  moby.112  moby.125  moby.16  moby.28  moby.40  moby.51  moby.63  moby.76  moby.88  moby.zip
moby.101  moby.113  moby.126  moby.17  moby.29  moby.41  moby.52  moby.64  moby.77  moby.89  README
moby.102  moby.114  moby.127  moby.18  moby.3  moby.41  moby.53  moby.65  moby.78  moby.9
moby.103  moby.115  moby.128  moby.19  moby.30  moby.42  moby.54  moby.66  moby.79  moby.90
moby.104  moby.116  moby.129  moby.2  moby.31  moby.43  moby.55  moby.67  moby.8  moby.91
moby.105  moby.117  moby.13  moby.20  moby.32  moby.44  moby.56  moby.68  moby.80  moby.92
moby.106  moby.118  moby.130  moby.21  moby.33  moby.45  moby.57  moby.69  moby.81  moby.93
moby.107  moby.119  moby.131  moby.22  moby.34  moby.46  moby.58  moby.7  moby.82  moby.94
moby.108  moby.12  moby.132  moby.23  moby.35  moby.47  moby.59  moby.70  moby.83  moby.95
moby.109  moby.120  moby.133  moby.24  moby.36  moby.48  moby.6  moby.71  moby.84  moby.96
[john@black Melville]$ grep white moby.*
moby.100:withdrawing it from the fold that had hidden it, he held up a white arm of
moby.100:a bouncing great whale, with a milky-white head and hump, all crows' feet and
moby.100:great-grandfather, with the white head and hump, runs all afoam into the pod.
moby.100:first, the white hump backed through the wreck, as though it was all chips,
moby.102:great, white, worshipped skeleton lay lounging --a gigantic idler! Yet, as the
moby.103:width, and looks something like a white billiard-ball. I was told that there
moby.108:the white heat for this kind of fine work. Un-a. So he must. I do deem it
moby.109:Matsui, and Siko. With his snow-white new ivory leg braced against the
moby.110:uncontinented seas, interflow with the blue heavens; and so form the white
moby.113:the White Whale? For the white fiend! But now for the barbs; thou must
moby.119:with three tapering white flaves, each of the three tall masts was silently
moby.119:The parted mouth of Tashtego revealed his shark-white teeth, which whaling
moby.119:it; mark it well; the white flame but lights the way to the White Whale!
moby.125:white, for I will not let this go.
moby.126:white bubbles in the blue of the sea. The life-buoy --a long slender cask --was
moby.128:and while they were yet in swift chase to windward, the white hump and head of
moby.128:bubbling white water; and after that nothing more: whence it was concluded

```

Figure 1: Melville's "Moby Dick" is subdivided into a number of text files. The "moby.*" wildcard tells `grep` to search through all the files in the directory

use `grep` in this way is killing a program that has crashed in the shell.

The `ps ax` command displays the processes. You can use a pipe to pass this output to `grep`, to apply a filter for the program you are looking for. Searching for Mozilla will return the following results, for example:

```

> ps ax | grep mozilla
2500 ? S 1:40 /usr/lib/
mozilla-1.3/mozilla-bin
5645 pts/4 S 0:00 grep mozilla

```

`grep` returns two matches containing Mozilla – not only the active browser, but also itself. The part we are interested is output at the start of each line: the process ID, which we need to *kill* the program in the shell. We can now type `kill 2500` to kill the hanging Mozilla browser.

As shell gurus are notoriously lazy, we will want to find a way to avoid typing this command each time we need it: in other words, we need an *alias*. Alias definitions should be saved in `.bashrc` in your home directory. This file is run each time you open an interactive shell. Use your favourite editor to open the file, this could be `write ~/.bashrc &` or `vi ~/.bashrc` in the shell, depending

on your preferences. As `vi` is non-trivial, let's have looked at some simple commands. Typing `G` tells `vi` to go to the end of the file. You can then type `o` to toggle the editor to input mode – in contrast to the `a` and `i` commands, this tells `vi` to start and place the cursor in that line.

You can now enter your alias in the last line of `.bashrc`. Instead of `pss` you could use any name you find easy to remember – but avoid overwriting the name of an existing command:

```
alias pss="ps ax |
grep"
```

Simple as that! You can then press `[Esc] ZZ` or `[Escape] :wq` to store the file and quit `vi`. If you want to use your new alias in the current shell, you need to parse the configuration file. To do so, type:

```
~/.bashrc
```

You can now use the `pss` program command to search for an active program.

An Address Book for Purists

`grep` is extremely flexible. One of my own favorite applications for `grep` is a simple address book.

All you need for this, are the `grep`, `alias`, and `cat` commands, plus a text file where you store the names, phone numbers, email, and snailmail addresses of your friends, acquaintances, and relatives. An address entry might be as follows:

```
Charly Penguin
+12345 678
tux@linux.org
1 South Pole Road
Tux Village, Antarctica
```

Now save the file as `addressbook` in your home directory, and add the following alias to `.bashrc`:

```
alias tel="cat ~/addressbook |
grep -i -A 4"
```

The `cat` command outputs the `addressbook` file. The pipe (`|`) sends this output to `grep`. The `-i` ("case-insensitive") option ensures that the search will not be case-sensitive. And finally, `-A 4` tells `grep` to return the next four lines after the line with the match.

Again

```
~/.bashrc
```

will re-parse your configuration file. In future, you can type `tel name` in the shell to retrieve the address of the person you are looking for – now that's what I call quick!

```

john@black.box: /home/john - Shell - Konsole
Session Edit View Settings Help

[john@black john]$ tel Charly
Charly Penguin
+12345 678
tux@linux.org
1 South Pole Road
Tux Village, Antarctica
[john@black john]$

```

Figure 2: You can use `grep` to quickly construct a shell-based address book. The "tel" alias outputs the address entry that matches the search key