Searching with grep, egrep, fgrep, (b)zgrep

Text in a Haystack

Commands belonging to the grep family are useful if you need to search text files for words and expressions. They either return the lines discovered by the search, the names of the files that contain the search string, or the number of matches. BY HEIKE JURZIK

Forgotten the file where you stored those oh-so-useful tips on using xy just a while back? A *grep* family command may be your last hope. The *grep* ("global regular expression print/parser") command line tool searches for search strings in text files and returns the lines containing matches as screen output.

The command can accept both simple text strings, and **regular expressions** as search keys, allowing users who ask the right questions to find the proverbial needle in a haystack.

Simple Searching

In the simplest of all cases, *grep* searches the specified file(s) (*test.txt* in our example below) for a single word, and outputs the lines which contain that word on screen:

```
$ grep the test.txt
Other than that he ...
Peter kicked the ball ...
```

The command is case-sensitive by default. However, you can set the *-i* flag to change this. As this tells *grep* to find words that contain the substring "the" (e.g. "Other"), you need to set the *-w* flag, to tell *grep* to search for whole words:

```
$ grep -iw the test.txt
The start of winter ...
Peter kicked the ball ...
```

Setting the *-n* flag provides a bit more information. It tells *grep* to also display the line number:

```
$ grep -n "the" test.txt
109:Other than that he ...
207:Peter kicked the ball ...
```

You do not need quotes if you are searching for a single word, but you do need them if your search string contains space characters, for example.

To gain a clearer overall view, you might like to try the *-num* option, replacing *num* with the number of lines both preceding and following the match to display. *grep -10 "the" test.txt* tells the powerful utility to start displaying the output ten lines before the match, and to end the display output ten lines after the match.

Table 1: Regular Expressions

Search string	Explanation
abc	Exactly this string: "abc"
[abc]	One of these characters: "a", "b" or "c"
[^abc]	None of these characters permitted to appear
[a-c]	One character between "a" and "c"
.	Any character
?	The character preceding the "?" is permitted to occur once or not at all (*egrep* only)
*	The character can occur any number of times or not at all
+	The character can occur any number of times, but must occur at least once (*egrep* only)
|	Link two search patterns to alternatively search for one (*egrep* only)
{n}	The character must occur exactly *n* times
{,n}	The character can occur no more than *n* times
{n,}	The character must occur exactly *n* times
{n,m}	The character must occur at least *n* times, and at most *m* times

Too much information at once? In that case, enter *grep -c* to output the number of matches. If you just need to know the name of the file that matches the search string, specify the *-l* option:

```
$ grep -l command *.html
grep.html
commandline.html
```

The above example searches files with the *.html* suffix in the current directory. If you need to search a complete tree structure, specify the *-r* (for "recursive") option:

```
$ grep -r command *
LMUK/df.html:The command line...
LMUK/magic.html:Some command ...
[...]
```

Sometimes it makes sense to remove individual results from the search list; for example, if you are searching through logfiles, and want to filter some details. The *-v* flag (see Listing 1) is useful in this case. It tells *grep* to search for any lines in the results that do *not* contain the second search string.

The */var/log/auth.log* file logs user logon entries. Many programs, such as *login* and *sshd*, use PAM (Pluggable Authentication Modules), to authenticate users. Listing 1 shows *grep* searching lines that contain the name of the user *huhn*, and throwing out any lines that contain the search string *pam* (*grep -v pam*).

Listing 1: Searching in the Search Results

```
$ grep huhn /var/log/auth.log | grep -v pam
Mar 18 12:53:21 asteroid sshd[1965]: Accepted password for huhn from
   192.168.2.15 port 1034 ssh2
Mar 18 12:57:06 asteroid sshd[2025]: Accepted password for huhn from
   192.168.2.15 port 1035 ssh2
Mar 18 13:00:13 asteroid sudo:     huhn : TTY=pts/0 ; PWD=/home/huhn
   ; USER=root ; COMMAND=/bin/bash
```

More Precision!

If you know that the text is at the start of a line in a file somewhere, you can formulate an appropriate regular expression to match:

```
$ grep "^The" test.txt
The start of winter ...
```

The circumflex ^ indicates the beginning of the line. Note that you need to use a backslash (\) to escape this meta-character, or place the whole expression in double quotes, to prevent the shell from interpreting the meta-character. Use a dollar sign to search at the end of a line:

```
$ grep "goal$" test.txt
... and scored the goal
```

^ and $ can be combined. If you specify a text string between these characters, *grep* will search for a line that matches the string exactly. If you prefer to search for single expressions rather than whole line, you can use \< and \> to delimit words.

Some other regular expressions only work with *egrep* (or *grep -E*) (see Table 1). *egrep* stands for "extended *grep*". It is quite easy to search a text file for the words "Linus" and "Linux" with *egrep*:

```
egrep "Lin(us|ux)" file1.txt
```

Other Family Members

The *grep* family contains a few more commands. The *fgrep* command (or alternatively *grep -F*) uses a quicker search algorithm, but does not support regular expressions. The command is useful if you need to search through a large amount of data as quickly as possible.

If you need to search in compressed files, try *zgrep* and *bzgrep*, instead. *zgrep* will search *gzip*ped files, and *bzgrep* does the same for *bzip2* files:

```
bzgrep Linux file1.txt.bz2
```

That can certainly save you a lot of time unpacking and repacking the files before and after searching. ■

GLOSSARY

Regular expression: *Provides a method of formulating searches that can handle parameters such as "any lines that start with A or B, and end in g". The regular expression to match the above statement is as follows: "^[AB].*g$". Refer to Table 1 for an explanation of the characters used in the regex. Additional details are available from [1] and http://en.wikipedia.org/wiki/Regular_expressions.*

INFO

[1] Marc André Selig: "Needles in a Haystack", Linux Magazine Issue 24 *http://www.linux-magazine.com/issue/24/RegularExpressions.pdf*