Session 8

Unix programmers use the word ‘grep’ in order to search their machine. ‘grep’ is a contraction of ‘global/regular expression/print’.

grep will find a specified file and print lines in that file which match a pattern. We can demonstrate this using the Haikus. Change you working directory to the one that contains the haiku (within the writing file in the file tree).

$ cat haiku.txt

The Tao that is seen

Is not the true Tao, until

You bring fresh toner.

With searching comes loss

and the presence of absence:

"My Thesis" not found.

Yesterday it worked

Today it is not working

Software is like that.

We can find lines that contain the word ‘the’:

$ grep the haiku.txt

The Tao that is seen

Is not the true Tao, until

and the presence of absence:

In this example the pattern we are searching for is the. Grep will search a file and look to match the pattern we have specified with a pattern in the file. The syntax here is to type grep followed by the pattern followed by the file name.

As an output we have returned all the lines in the file that contain the letters ‘the’

Grep will search for a pattern in a case-sensitive way by default. In addition, the search pattern that is selected doesn’t have to form a complete word.

Take for example: ‘The’. Here we are searching for the same word but with a capitalised T

$ grep The haiku.txt

The Tao that is seen

"My Thesis" not found.

This time, two lines that include the letters ‘The’ are outputted, one of which contained our search pattern within a larger word, ‘Thesis’.

We can use flags to restrict what grep returns. To return just the full word ‘The’, we use the -w  option

$ grep **-w** The haiku.txt

The Tao that is seen

We can search for a phrase by using quotations.

$ grep **-w** "that is" haiku.txt

The Tao that is seen

It is best practice to always use quote marks, regardless of if you’re search for a phrase or a single word.

Another useful option is -n, which numbers the lines that match:

$ grep **-n** "it" haiku.txt

5:With searching comes loss

9:Yesterday it worked

10:Today it is not working

Here, we can see that lines 5, 9, and 10 contain the letters ‘it’.

As usual we can combine options. We can combine the option -w to find the lines that contain the word ‘the’ and -n to number the lines that match:

$ grep **-n** **-w** "the" haiku.txt

2:Is not the true Tao, until

6:and the presence of absence:

Now we want to use the option -i to make our search case-insensitive:

$ grep **-n** **-w** **-i** "the" haiku.txt

1:The Tao that is seen

2:Is not the true Tao, until

6:and the presence of absence:

grep has lots of other options. To find out what they are, we can type:

$ grep **--help**

Usage: grep [OPTION]... PATTERN [FILE]...

Search for PATTERN in each FILE or standard input.

PATTERN is, by default, a basic regular expression (BRE).

Example: grep -i 'hello world' menu.h main.c

Regexp selection and interpretation:

-E, --extended-regexp PATTERN is an extended regular expression (ERE)

-F, --fixed-strings PATTERN is a set of newline-separated fixed strings

-G, --basic-regexp PATTERN is a basic regular expression (BRE)

-P, --perl-regexp PATTERN is a Perl regular expression

-e, --regexp=PATTERN use PATTERN for matching

-f, --file=FILE obtain PATTERN from FILE

-i, --ignore-case ignore case distinctions

-w, --word-regexp force PATTERN to match only whole words

-x, --line-regexp force PATTERN to match only whole lines

-z, --null-data a data line ends in 0 byte, not newline

Miscellaneous:

... ... ...

### Quick Questions:

1. Which command would result in the following output:

and the presence of absence:

1. grep "of" haiku.txt
2. grep -E "of" haiku.txt
3. grep -w "of" haiku.txt
4. grep -i "of" haiku.txt

### Wildcards

Options aren’t where grep’s real power lies. It’s real power comes from the fact that patterns can include wildcards. (The technical name for these is **regular expressions**, which is what the ‘re’ in ‘grep’ stands for.) Regular expressions are both complex and powerful.

$ grep **-E** "^.o" haiku.txt

You bring fresh toner.

Today it is not working

Software is like that.

We use the -E option and put the pattern in quotes to prevent the shell from trying to interpret it. (If the pattern contained a \*, for example, the shell would try to expand it before running grep.) The ^ in the pattern anchors the match to the start of the line. The . matches a single character (just like ? in the shell), while the o matches an actual ‘o’.

### Quick Questions:

1. Leah has several hundred data files saved in one directory, each of which is formatted like this:

2013-11-05,deer,5

2013-11-05,rabbit,22

2013-11-05,raccoon,7

2013-11-06,rabbit,19

2013-11-06,deer,2

She wants to write a shell script that takes a species as the first command-line argument and a directory as the second argument. The script should return one file called species.txt containing a list of dates and the number of that species seen on each date. For example using the data shown above, rabbit.txt would contain:

2013-11-05,22

2013-11-06,19

Put these commands and pipes in the right order to achieve this:

cut **-d** : **-f** 2

>

|

grep **-w** $1 **-r** $2

|

$1.txt

cut **-d** , **-f** 1,3

Hint: use man grep to look for how to grep text recursively in a directory and man cut to select more than one field in a line.

### Listing and Finding: The similarities and differences

ls and find can be made to do similar things given the right options, but under normal circumstances, ls lists everything it can, while find searches for things with certain properties and shows them.

We can combine them and put the find command inside $():

$ wc **-l** **$(**find . **-name** "\*.txt"**)**

11 ./haiku.txt

300 ./data/two.txt

21022 ./data/LittleWomen.txt

70 ./data/one.txt

21403 total

Upon executing this command, the shell will first run everything inside the brackets, $(). The brackets are then replaced with whatever the prior output was, allowing the shell to then execute the rest of the command since the output of find is the four filenames ./data/one.txt, ./data/LittleWomen.txt, ./data/two.txt, and ./haiku.txt.

$ wc **-l** ./data/one.txt ./data/LittleWomen.txt ./data/two.txt ./haiku.txt

which is what we wanted. This expansion is exactly what the shell does when it expands wildcards like \* and ?, but lets us use any command we want as our own ‘wildcard’.

It’s very common to use find and grep together. The first finds files that match a pattern; the second looks for lines inside those files that match another pattern. Here, for example, we can find PDB files that contain iron atoms by looking for the string ‘FE’ in all the .pdb files above the current directory:

$ grep "FE" **$(**find .. **-name** "\*.pdb"**)**

../data/pdb/heme.pdb:ATOM 25 FE 1 -0.924 0.535 -0.518

### Listing and Finding: The similarities and differences

* find finds files with specific properties that match patterns.
* grep selects lines in files that match patterns.
* --help is an option supported by many bash commands, and programs that can be run from within Bash, to display more information on how to use these commands or programs.
* man command displays the manual page for a given command.
* $(command) inserts a command’s output in place