Lab_03_1: Searching with `find` and `grep` using Globbing and Regex

Otago Polytechnic, IN616 Operating Systems Concepts,

Semester 1 – 2020

1 Objectives

- Searching the file system (file and directory names)
- Searching file content
- Refining searching with globbing and regular expressions

2 Searching the File System

An important task of working in different Linux systems is the ability to **quickly find directories and files**. For example, finding configuration files, log files or important documents.

2.1 Basic Commands

There are many commands to perform searching, but in this lab we will look at the most fundamental command: `find`. The basic syntax of the `find` command is:

```
find <location> <options> <expression>
```

Below is a basic example of the find command. Try and execute the following command in your Linux system. When executing the command, make sure you are in your home directory (`cd ~`).

```
find . -name ".*"
```

Based on the command above, answer the following 3 questions.

Q1. What location in the file system is the `find` command searching? Explain how you came to this answer.

Q2. What options are you using with the `find` command?

HINT: Remember to use the help menu or man page of a command to determine information about command options and arguments. You can use `man find`, or `find -help`.

Q3. What is the search term of the executed `find` command? Describe what this search term will find and give at least one example of files the command found.

In Linux systems, the current directory is indicated by a dot (`.`). Furthermore, hidden files start with a dot (`.`). The asterisk (`*`) is a wildcard for any following character. Basically, the summary of the search term we used above is: *anything that starts with a dot (`.`).*

> **PRO TIP:** The standard `ls` command does not display hidden files, files that start with a dot (`.`). Run `ls` with the option `-a` to see all files, including hidden files.

Also, in Linux systems, the parent directory is indicated by a dot dot (`..`). This is why we use `cd ..` to move up one directory.

We are going to investigate some more `find` commands to get a good idea of how we can use it to find files and directories.

Q4. What command would you run to look for all hidden files in the parent directory of your current folder? Note: try to write a command that will do this without changing the directory using the `cd` command.

HINT: Remember the difference between relative and absolute paths. Also think of how you would change to the parent directory using the `cd` command.

In the above command you should see files in two different home directories, one user is named `student` and the other user is named `manager`.

Q5. What is the command for finding the file named: `hostname` in the `/etc` folder?

Note that you may get some *permission denied* errors in addition to the valid results from your answer to Question 5. The reason for this is that `find` works recursively by default (and some sub folders are restricted to sudo). Recursive means it includes all subfolders of the given folder in the search. Use the parameter `-maxdepth` to constrain the search depth instead of being recursive. For example, the following command only searches one level in the `/var/log` directory.

```
find /etc -maxdepth 1 -name "s*"
```

If you want to reduce the number of results (independent of the depth) you can append `| head` to your command. By default this will only show the first ten results. For example:

```
find /etc -name "s*" | head
```

You further use the `-n` parameter to specify the number of lines; for example, `| head -n 15` for the first 15 lines. The `tail` command does the opposite: it shows the last lines of output. `head` and `tail` are very powerful commands and work with any form of output. At any point where there are tool many results to fit on the screen, try these commands.

2.2 Searching Directories or Files

By default `find` not only explores arbitrary levels of depth (recursive), but also finds any type of entry - this includes directories, as well as files. You can use the parameter `-type f` or `-type d` to identify only files or only directories. This helps to find specific types! For example, only search file names and not directory names.

Q6. How can we identify all files that start with `host` in the `/etc` directory, without its subdirectories)?

HINT: Use your knowledge of all previous options we have used for the `find` command.

Q7. What is the command to show all hidden files in a given directory?

Q8. What is the command to show all directories that start with the letter `s` in the `/etc` directory?

3 Searching File Content with grep

Until now we have concentrated on file names, or directory names, when searching. However, it is equally important to search files for content. For this purpose, Linux systems have the powerful tool `grep`.

3.1 Introducing grep

The `grep` command can search through files and directories and check which lines in those files match a given regular expression. `grep` will output the file names and the line numbers, or the actual lines, that matched the search term. It is an exceptionally powerful tool that is used on a daily basis by most Linux users.

The `grep` command has follows the syntax:

```
grep <pattern> <file/s or directory>
```

`grep` offers a wide range of options (or paramters). For example, the `-i` option performs a case-insensitive search (it will match uppercase and lowercase characters). Additional information on the command can be found in the man page or help menu.

The following example is a very common `grep` command.

```
grep -r "config" /etc
```

The above command will search for all files that contain the string `config` in the `/etc` directory. Watch out, the output is log as there are many files that contain the keyword `config`! Make sure you use the scroll up and scroll down hotkeys to review the output. Look back at last classes lab if you have forgotten these hotkeys.

Q9. In the above command, what does the `-r` option achieve? Describe what this option does in your own words, and provide an example.

Q10. Why are you getting permission errors when you run the above command? How would you solve this issue?

OK, so we mentioned in the lecture that `grep` uses regular expressions... Interestingly, the name stands for: *globally search a regular expression and print*. However, we have not even used a regular expression yet! This is because you don't always need to resort to regular expressions, and can achieve quite a lot with simple strings and globbing - even in complex search scenarios...

Q11. What is the command to search for the content of files in the `/etc` directory. You command should find files that contain `start`, but avoiding terms such as `restarting`, `restart` or `started`?

HINT: Don't over think it! You do not need a regular expression. Also, you might need to search recursively! Feel free to ask fellow colleagues or the tutor if you are stuck.

Now we want to expand on the command we just ran. This time we should try the command with additional options. Try adding the following option to the command you answered for Question 11: `-l` and `-n`.

Q12. What does the `-l` option do?

Nice work so far! Now we want to search a specific file, instead of a directory of files. In most Linux systems, user information is stored in the `/etc/passwd` file. Try viewing the file using the following command:

```
cat /etc/passwd
```

Now we want to make a command that will grab lines from this file when they match a specific pattern.

Q13A. What is the command to identify all users that use the bash shell? Try to make your `grep` command as specific as possible.

Q13B. Based on your answer to Question 13A, what users have the `/bin/bash/` shell as a default.

So far we have performed a search technique known as *globbing*, however, we have only touched the surface of what is possible. A brief overview of generally used operators is given in https://en.wikipedia.org/wiki/Glob_(programming).

3.2 Using Regular Expressions

A more powerful mechanism to match content are regular expressions (or regex). You will find regular expression libraries in a wide variety of programming languages, and unfortunately some are a little different from others - basically, some regex libraries use different expressions to perform the same thing. Regular expressions are useful because they allow much more complex matching.

To become more comfortable with regular expressions, try your skills using the following interactive tutorial:

- http://regexone.com/

Please try to complete as much of this tutorial as possible. Make sure you read the instructions provided on each page, as the exercises do not make any sense without reading the purpose of the exercise! Try to finish up to question 5, at least.

Take note that there may not always be a single solution to the exercises, but various ways to match expressions. Also, as mentioned in the lecture, regular expressions do not necessarily need to match the entire string, just subsets of it (portions).

Once you are comfortable with the basic principles of regular expressions, continue with the following tasks.

Other than just working with files, grep works on any kind of input piped into it. Piping is an amazing tool for using with Linux commands, so try to get used to the concept. Piping in Linux commands is achieved with the pipe symbol (|). Using a pipe, you can filter the output from commands (such as find, ls and others) by performing a grep search of the first commands output. For example, the following command lists all files in the /etc directory and searches for any output that begins with host:

```
ls /etc | grep "host*"
```

3.3 Finding Installed Packages using grep and Regular Expressions

The following command will list all available packages on an Ubuntu system:

```
apt-cache pkgnames
```

Try running the above command without any additional arguments... The list is long! So it makes sense to try to search it. A good method is to perform expression matching on those packages using `grep`.

We are going to start simple, by piping the output of the `apt-cache pkgnames` to grep and *globbing* the keyword `ubuntu`

Q14. Write a `grep` command to find all package names that contain `ubuntu`.

HINT: Make sure to use the `apt-cache pkgnames` command first, followed by a pipe, then your `grep` command.

Q15. Write a `grep` command that uses a regular expression to find all package names that start with `ubuntu` and contain anything else that keyword. For example: `ubuntu-upload-manager`.

> **PRO TIP:** A good resource to debug your regular expressions quickly is `http://regexr.com/`. Simply copy or type text you want to match into the text area. Construct your regex without escape characters (\) until it works. Once done, adapt the regex to use it in bash (i.e. escape parentheses and | with \)

Q16. Write a command to identify all packages that end with a single numeric digit.

Q17. Identify all packages that contain `lib` in their names and version numbers in the format x.xx.x. For example: `2.83.0` or `4.40.7`.

Q18. Refine the previous command. Identify all packages that begin with either `libboost` or `libgtk` and end with the version numbers in the format x.xx.x. For example: `2.83.0` or `4.40.7`.

HINT: Remember: You need to escape the dot symbol (.).

3.4 Exploring the `syslog` using `grep`

An important target for regular expression matching is the syslog, and all other log files. This is because log files are... well, long! The syslog is the most general logging mechanism in Linux systems. In Debian-based distributions (like Ubuntu) you find it in the following location:

```
/var/log/syslog
```

Explore it using the following command:

```
cat /var/log/syslog | tail -n 100
```

Make sure to get an idea of the format and structure of the syslog file. This includes date and time stamps, and the different columns in the file.

Q19. Write a command to list all entries added in the last two weeks (check the entry format) and contain the keyword: `dhclient`.

Q20. Write a command to match all entries that contain `DHCP` commands.

Q21. Write a command to match all *DHCP offers* (OFFER) and *DHCP acknowledgements* (ACK), but not *DHCP requests*.

4 Challenge Questions

CQ1. Using `http://regexr.com/` try to construct a simple regex that matches an e-mail address. As you know, there is no single solution, but try to keep it simple.

CQ2. Write a regex that matches URLs.

5 Additional Resources

A range of further command line features, including input, output and error streams as well as grep are introduced in Chapter 8 of the Linux Essentials course on Netacad.

If not done before, have a look at the interactive tutorial under `http://regexone.com/` to develop your understanding of regular expressions.