Worksheet 8: Shell Scripting

Updated: 19th August, 2018

The objectives of this practical are:

- To gain experience writing UNIX shell scripts.
- To become familiar with regular expressions.

Practical Exercises

Note: For all of these exercises, ensure you adequately test your scripts!

Also, don't be afraid to experiment at the command-line. Everything you put in a script file is a valid command, and every valid command can be put in a script file.

1. `if` and "`[`"

Write a set of simple bash scripts to perform the following tasks:

(a) `newerthan` — prompt the user to enter the names of two files, and determine which one is newer (i.e. modified more recently).

(b) `permissions` — take a filename from the command-line, and determine whether the file is readable, writable and/or executable.

2. `for` and `$*`

Write a set of simple bash scripts to perform the following tasks:

(a) `allperms` — take a list of files from the command-line, and determine whether each file is readable, writable and/or executable.

(b) `identical` — take a list of files from the command-line, and report any *pairs* in the list that have identical contents (i.e. they contain exactly the same bytes).

Note: You will need the exit status of the `diff` command.

(c) `reverse` — take any number of command-line parameters (just strings, not necessarily filenames), and output them in reverse order.

Note: Think in terms of string manipulation. You should not need to use any indexing or counting for this.

3. Command Substitution and Piping

(a) Modify your `allperms` and `identical` scripts to work on all files in all subdirectories, instead of filenames provided on the command line.

> **Note:** The command `find -type f` will output all (normal) files in all subdirectories.

(b) Write a simple bash script to recursively scan subdirectories (starting at the current directory), listing the names of all files where the name contains the current day-of-the-month. Do not hard-code anything.

> **Note:** Consult the man pages for the commands `find` and `date`. You *may* also need `grep`.

4. Regular Expressions

Taking the sample log file from last week's worksheet, write and test (with `grep`) regular expressions to find the following information:

(a) All log entries recorded on 20 August from 10:21:00am to 10:21:59am. For instance:

```
Aug 20 10:21:11 acpid: 1 client rule loaded
```

(b) All log entries containing the words "warning", "error" or "fail" (nb. this should be a *single* regex, not three separate ones).

(c) All log entries recorded at a time where minutes is zero. For instance:

```
Aug 22 03:00:21 avahi-daemon[1135]: Invalid query packet.
```

At the same time, avoid false positives.

(d) All log entries containing an IPv4 address — 4 integers separated by periods (e.g. "`82.13.55.116`").

(e) All log entries containing at least *two* IPv4 addresses.

End of Worksheet