Edith Cowan University
School of Science

CSI6203 Scripting Languages

Module 4

Snippets, Calculations and More bash commands

Contents

- Code Snippets
- Arithmetic expressions in bash
- More bash commands
- Redirection/colours

Learning Objectives

After finishing this module, you should be able to:

- Use arithmetic in bash scripts
- Customise your text editor to improve efficiency
- Use grep, find and ping
- Use redirections for input and output

Code Snippets

Code Snippets

- There are several structures in scripting that involve typing the same code many times (such as the shebang #! line)

- To improve efficiency, many text editors have the ability to automatically insert pre-prepared chunks of code without needing to type them all manually

Code Snippets in VSCode

- Visual Studio code provides a method for creating custom snippets to improve your scripting experience

Code Snippets in VSCode

- The syntax for creating snippets uses a structure called “JSON” which allows scripters to easily create snippets for anything they want

```
"add a shebang": {
        "prefix": "shebang",
        "body": [
            "#!/bin/bash"
        ],
        "description": "Add shebang to script"
    }
```

Code Snippets in VSCode

- To insert a snippet, start typing the prefix and then hit <tab> to insert it

Code Snippets in VSCode

- Snippets can also contain multiple lines and editable fields for convenience.

```
"add a decision": {
    "prefix": "decision",
    "body": [
        "if $1; then",
        "    $2",
        "fi"
    ],
    "description": "Add an if statement to a script"
}
```

Code Snippets in VSCode

- Snippets can also contain multiple lines and editable fields for convenience

ARITHMETIC OPERATIONS

((

- In bash scripting, double parentheses are used to evaluate arithmetic and perform mathematical operations

```
$ a=$(( 5 + 5 ))
```

- The same results can be accomplished using the “let” command

```
$ let a=5+5
```

((

- Bash supports the following arithmetic operations

Operator	Purpose
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation (powers)
%	Modulus (remainders)

((

- All numeric values in bash are treated as integers
- Bash does not understand floating point numbers (eg. 5.5)
- In bash, eleven divided by two is five

```
echo $(( 11 / 2 ))
5
```

- Any fractional information is truncated

Increment and Decrement

- bash also has numeric increment and decrement operators

- These add one to a variable or subtract one from a variable respectively

```
$ count=1
$ $(( count++ ))
$ echo count
2
```

numeric comparisons

- Double parentheses can also be used to compare numbers using boolean operators (Such as > or <)

```
if (( count > 1 )); then
    echo 'count is bigger than one!'
fi
```

More useful bash commands

Command Types

- There are five different types of command in bash
  - Alias
    - a shortcut to another command
  - Function
    - a series of pre-defined commands (covered later in the unit)
  - Shell Built-in
    - commands provided as part of the bash shell
  - Keyword
    - not a command but part of bash syntax
  - File
    - a program or script or other executable command

Command Types

- There are five different types of command in bash
  - Alias
    - a shortcut to another command
  - Function
    - a series of pre-defined commands (covered later in the unit)
  - Shell Built-in
    - commands provided as part of the bash shell
  - Keyword
    - not a command but part of bash syntax
  - File
    - a program or script or other executable command

type

- The `type` command can be used to find out the type of a command

```
$ type ls
ls is aliased to `ls --color=auto'
```

- Some commands can have multiple types which can be seen using the –a flag

```
$ type –a ls
ls is aliased to `ls --color=auto'
ls is /bin/ls
```

find

- The `find` command can be used to search for specific files

```
$ find -name 'script.sh'
```

- It can also be used in scripts to apply a command to every file it has found using the -exec option

```
#!/bin/bash
find -name "*.sh" -exec cp {} backups/ \;
```

find

- In this example, the command locates all files that end in “.sh”

```
#!/bin/bash
find -name "*.sh" -exec cp {} backups/ \;
```

- For each of those files, it will execute the “cp” command to copy the files to the “backups” folder.
- In this case, {} refers to the name of each file that has been found

ping

- ping is a simple network command that sends an ICMP ping request to an IP address

```
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=56 time=56.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=56 time=56.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=56 time=56.5 ms
```

- This can be used to test network connectivity and check latency

wget

- wget is a simple network command connects to a web address and downloads the contents
- This can be used to download files and websites from URLs
- Very useful in web scraping scripts

wget

```
wget http://google.com

http://google.com/
Resolving google.com (google.com)... 216.58.203.110, 2404:6800:4006:804::200e
Connecting to google.com (google.com)|216.58.203.110|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://www.google.com/ [following]
http://www.google.com/
Resolving www.google.com (www.google.com)... 172.217.167.100,
2404:6800:4006:809::2004
Connecting to www.google.com (www.google.com)|172.217.167.100|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]Saving to: ‘index.html’

index.html         [ <=>     ]  11.10K  --.-KB/s    in 0s

(55.8 MB/s) - ‘index.html’ saved [11368]
```

grep

- grep is a powerful search tool that can find text within files or other text streams

```
# Find all lines that contain the
# word “ponies” in script.sh
grep ponies script.sh
```

- It is used very often in bash scripting to locate specific text within files or output

More scripting syntax

Redirection

- By default, the standard output for a script is the terminal screen

- By default, the standard input for a script is the terminal input (typed in by the user)

Redirection

- The standard input and standard output can be redirected to use files instead using the redirection operators < and >

```
$ ./backup.sh > backuplog.txt
```

- Instead of outputting any "echo" commands to the screen, this will save it into the "backuplog.txt" file

Redirection

- The input can also be replaced

```
$ ./hello.sh < name.txt
```

- Instead of getting input from a user, any "`read`" command will use whatever text is in the "name.txt" file

Redirection

- The output of one command can also be redirected to be the input of another

```
$ ./script1.sh | ./script2.sh
```

- This is called “piping” and uses the “|” pipe operator
- It’s very common to see this with the “grep” command and in other text processing

```
$ ./viewLogs.sh | grep error
```

Colours

- Most terminals support ANSI colour codes.
- These can be used in bash scripts using special escape character codes for setting colours and resetting colours

```
echo -e "\033[31mERROR\033[0m"
```

- \033 is the code for a terminal escape character
- [31m is the code to set RED colour
- [0m is the code to reset to the previous colour

Colours

- It’s very common to create variables for them to save time

```
Black='\033[30m'
Red='\033[31m'
Green='\033[32m'
Brown='\033[33m'
Blue='\033[34m'
Purple='\033[35m'
Cyan='\033[36m'
White='\033[37m'
Clear='\033[0m'
echo -e "${Purple}this is purple text${Clear}"
```

Colours

- Here are some common colour codes

Colour	Code
Black	`\033[30m`
Red	`\033[31m`
Green	`\033[32m`
Brown	`\033[33m`
Blue	`\033[34m`
Purple	`\033[35m`
Cyan	`\033[36m`
Grey	`\033[37m`

Summary

- Terms to review and know include:
  - Snippets
  - integers
  - arithmetic expressions
  - type, find, ping and grep
  - redirection
  - colour codes

References and Further Reading

- Ebrahim, M. and Mallet, A. (2018) Mastering Linux Based Scripting (2nd Ed) Chapter 4 and 5