Computer Practicum 1

Introduction to Bash scripts

Vida Groznik

Source: Ryan‘s tutorials.

For loops

```
1. #!/bin/bash
2. # Basic for loop
3.
4. names='Alen Marko Aleks'
5.
6. for name in $names
7. do
8.     echo $name
9. done
```

For each of the items in a given list, perform the given set of commands. It has the following syntax:

```
for var in <list>
do
        <commands>
done
```

The **for** loop will take each item in the **list** (in order, one after the other), assign that item as the value of the variable **var**, execute the commands between **do** and **done** then go back to the top, grab the next item in the list and repeat over.

The list is defined as a series of strings, separated by spaces.

Line 4 - Create a simple list which is a series of names.

Line 6 - For each of the items in the list `$names` assign the item to the variable `$name` and do the following commands.

Line 8 - echo the name to the screen.

For loops - ranges

```
1. #!/bin/bash
2. # Basic range in for loop
3.
4. for value in {1..5}
5. do
6.     echo $value
7. Done
```

```
1. #!/bin/bash
2. # Basic range with steps for loop
3.
4. for value in {10..0..2}
5. do
6.     echo $value
7. done
```

We can process a series of numbers. To specify a range use curly braces `{No1..No2}` in a for loop and specify any number you like for the starting value and the ending value. There are **no spaces** between the curly braces!

The first value may also be larger than the second in which case it will count down.

It is also possible to specify a value to increase or decrease by each time. You do this by adding another two dots (`..`) and the value to step by.

Break & continue

Sometimes we may need to intervene and slightly alter the running of a loop. We can do this using **`break`** and **`continue`** statements.

`break`

The **`break`** statement tells Bash to **leave the loop straight away.**

It may be that there is a normal situation that should cause the loop to end but there are also exceptional situations in which it should end as well. For instance, maybe we are copying files but if the free disk space get's below a certain level we should stop copying.

`continue`

The **`continue`** statement tells Bash to **stop running through this iteration of the loop and begin the next iteration.**

Sometimes there are circumstances that stop us from going any further. For instance, maybe we are using the loop to process a series of files but if we happen upon a file which we don't have the read permission for we should not try to process it.

Select

```
1.  #!/bin/bash
2.  # A symple menu system with select
3.
4.  names='Alen Marko Aleks Quit'
5.
6.  PS3='Select person: '
7.
8.  select name in $names
9.  do
10.     if [ $name == 'Quit' ]
11.     then
12.         break
13.     fi
14.     echo Hello $name
15. done
16.
17. echo Bye
```

The **`select`** mechanism allows you to create a simple menu system. It has the following format:

```
select var in <list>
do
        <commands>
done
```

It takes all the items in list and present them on the screen with a number before each item. A prompt will be printed after this allowing the user to select a number. When they select a number and hit enter the corresponding item will be assigned to the variable `var` and the commands between `do` and `done` are run.

Once finished a prompt will be displayed again so the user may select another option.

Line 4 - Set up a variable with the list of names and a last option which we may select to quit.

Line 6 - Change the value of the system variable PS3 with something more descriptive. (By default it is #?)

Lines 10 - 13 - If the last option, 'Quit', is selected break out of the select loop.

Exercises

Write a script using **for loop** to print the following patterns on the screen:

a)

```
1
22
333
4444
55555
```

b)

```
1
12
123
1234
12345
```

c)

```
*
**
***
****
*****
```

d)

```
*
**
***
****
*****
*****
****
***
**
*
```

e)

```
    1
   22
  333
 4444
55555
666666
7777777
88888888
```

f)

```
       *
      **
     ***
    ****
   *****
  ******
 *******
********
```