IT Scripting and Automation

Bash Scripting
-IF statements (test)-

Lecturer: Art Ó Coileáin

Conditional Programming

- One of the most important features of all programming languages is the ability to execute commands depending on previously defined conditions.

- The **if** construction allows you to specify such conditions.

- **For more info check: man test**

If/Else:

- The syntax of an **if/else** is as follows: (Square brackets are required)

```
if [ condition ]
then
        statement1
        statement2
        .........
else
        statement3
        .........
fi
```

Conditional Programming

Else if:

- while the structure of an "**else if**" is as follows:

```
if [ condition ]
then
        statement1
        .........
elif [ condition2 ]
then
        statement3
        ........
elif [ condition3 ]
then
        statement4
        ........
fi
```

Conditional Programming

- Example (myBashIF.sh):

```
#!/bin/bash
echo Type the value of X
read X
echo Type the value of Y
read Y
if [ $X -lt $Y ]
then
        echo ‘The value of $X is less than the value of $Y’
else
        echo ‘The value of $X is greater or equal than the value of $Y’
fi
```

Quick Reference: Conditional Programming

Operator	Produces true if …	Number of operands
-n	operand non zero length	1
-z	operand has zero length	1
-d	there exists a directory whose name is *operand*	1
-f	there exists a file whose name is *operand*	1
-eq	the operands are integers and they are equal	2
-ne	the opposite of -eq	2
=	the operands are equal (as strings)	2
!=	opposite of =	2
-lt	*operand1* is strictly less than *operand2* (both operands should be integers)	2
-gt	*operand1* is strictly greater than *operand2* (both operands should be integers)	2
-ge	*operand1* is greater than or equal to *operand2* (both operands should be integers)	2
-le	*operand1* is less than or equal to *operand2* (both operands should be integers)	2

File operators (test)

- Syntax:
  - [condition-to-test-for]
- Example:
  - **[-e /etc/passwd]**
- Remember syntax for if/else and else if statements:

```
if [ condition ]
then
    statement1
else
    statement2
fi
```

Note: Please remember the spaces inside the square brackets.

File operators (test)

-d FILE	#True if file is a directory.
-e FILE	#True if file exists.
-f FILE	#True if file exists and is a regular file.
-r FILE	#True if file is readable by you.
-s FILE	#True if file exists and is not empty.
-w FILE	#True if file is writable by you.
-x FILE	#True if file is executable by you.

- For more info check: man test

String operators (test)

- -z VARIABLE   #True if VARIABLE is empty.
- -n VARIABLE   #True if VARIABLE is not empty.
- VARIABLE1=VARIABLE2 #True if the variables are equal.
- VARIABLE1!=VARIABLE2   #True if the strings are not equal.

Arithmetic Operators (test)

- arg1 -eq arg2 #True if arg1 is equal to arg2.
- arg1 -ne arg2 #True if arg1 is not equal to arg2.
- arg1 -lt arg2 #True if arg1 is less than arg2.
- arg1 -le arg2 #True if arg1 is less than or equal to arg2.
- arg1 -gt arg2 #True if arg1 is greater than arg2.
- arg1 -ge arg2 #True if arg1 is greater than arg2 or equal to arg2.

Exercise

- Create a file called “file1” using VIM editor.
- Type your name on it, save the file and quit.
- Create a empty file called “file2” using touch command.
- Create a script called “scriptIF.sh”.
- Using if statements check the following (provide appropriate message in each case):
  - Check if both files exist.
  - Check if file1 or file2 is a directory
  - Check if file1 has execution permissions
  - Check what file is older (check man test if needed)
  - Check if file2 is empty