Conditionals

coursera.org/learn/linux-tools-for-developers/supplement/f8cTz/conditionals

bash permits control structures like:

```
1
2
3
4
5
6
if condition
then
   statements
else
   statements
fi
```

There is also an **elif** statement.

Note that the variable **$?** holds the exit status of the previous command.

The condition can be expressed in several equivalent ways:

```
1
2
3
if [[ -f file.c ]] ; then ... ; fi
if [-ffile.c] ;then...;fi
if test -f file.c ; then ... ; fi
```

Remember to put spaces around the **[ ]** brackets.

The first form with double brackets is preferred over the second form with single brackets, which is now considered deprecated. For example, a statement such as:

```
1

if [ $VAR == "" ]
```

will produce a syntax error if **VAR** is empty, so you have to do:

```
1

if [ "$VAR" == "" ]
```

to avoid this.

The test form is also deprecated for the same reason and it is more clumsy as well. However, it is common to see these older conditional forms in many legacy scripts.

You will often see the **&&** and || operators (AND and OR, as in C) used in a compact shorthand:

```
1

2

$ make && make modules_install && make install

$ [[ -f /etc/foo.conf ]] || echo 'default config' >/etc/foo.conf
```

The **&&**s (ANDs) in the first statement say stop as soon as one of the commands fails; it is often preferable to using the **;** operator. The ||' (ORs) in the second statement says stop as soon as one of the commands succeeds.

The **&&** operator can be used to do compact conditionals; e.g. the statement:

```
1
[[ $STRING == mystring ]] && echo mystring is "$STRING"
```

is equivalent to:

```
1
2
3
if [[ $STRING == mystring ]] ; then
    echo mystring is "$STRING"
fi
```

File Conditionals

There are many conditionals which can be used for various logical tests. Doing **man 1 test** will enumerate these tests. Grouped by category we find:

Test	Meaning
-e file	**file** exists?
-d file	**file** is a directory?
-f file	**file** is a regular file?
-s file	**file** has non-zero size?

Test	Meaning
-g file	**file** has **sgid** set?
-u file	**file** has **suid** set?
-r file	**file** is readable?
-w file	**file** is writeable?
-x file	**file** is executable?

String Comparisons

If you use single square brackets or the test syntax, be sure to enclose environment variables in quotes in the following:

Test	Meaning
string	**string** not empty?
string1 == string2	**string1** and **string2** same?
string1 != string2	**string1** and **string2** differ?
-n string	**string** not null?
-z string	**string** null?

Arithmetic Comparisons

These take the form:

```
1

exp1 -op exp2
```

where the operation **(-op)** can be:

```
1
-eq, -ne, -gt, -ge, -lt, -le
```

Note that any condition can be negated by use of **!**.

case

This construct is similar to the **switch** construct used in C code. For example:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
#!/bin/sh
echo "Do you want to destroy your entire file system?"
read response
```

```
case "$response" in
   "yes")              echo "I hope you know what you are doing!" ;;
   "no" )              echo "You have some comon sense!" ;;
   "y" | "Y" | "YES" ) echo "I hope you know what you are doing!" ;
                       echo 'I am going to type: " rm -rf /"';;
   "n" | "N" | "NO" )  echo "You have some comon sense!" ;;
   *   )               echo "You have to give an answer!" ;;
esac
exit 0
```

Note the use of the **read** command, which reads user input into an environment variable.

Completed