Chapter 3b. scripting loops

3b.1. test [ ]

The **test** command can test whether something is true or false. Let's start by testing whether 10 is greater than 55.

```
[clim@sop ~]$ test 10 -gt 55 ; echo $? 1

[clim@sop ~]$
```

The test command returns 1 if the test fails. And as you see in the next screenshot, test returns 0 when a test succeeds.

```
[clim@sop ~]$ test 56 -gt 55 ; echo $? 0

[clim@sop ~]$
```

If you prefer true and false, then write the test like this.

```
[clim@sop ~]$ test 56 -gt 55 && echo true || echo false
true

[clim@sop ~]$ test 6 -gt 55 && echo true || echo false
false
```

The test command can also be written as square brackets, the screenshot below is identical to the one above.

```
[clim@sop ~]$ [ 56 -gt 55 ] && echo true || echo false
true

[clim@sop ~]$ [ 6 -gt 55 ] && echo true || echo false
false
```

Below are some example tests. Take a look at **man test** to see more options for tests.

```
[ -d foo ]              Does the directory foo exist ?
                        Does the file bar exist ?
[ -e bar ]
                        Is the string /etc equal to the variable $PWD ?
[ '/etc' = $PWD ]       Is the first parameter different from secret ?
                        Is 55 less than the value of $bar ?
[ $1 != 'secret' ]
[ 55 -lt $bar ]         Is the value of $foo greater or equal to 1000 ?
                        Does abc sort before the value of $bar ?
[ $foo -ge 1000 ]
[ "abc" < $bar ]        Is foo a regular file ?
                        Is bar a readable file ?
[ -f foo ]
                        Is file foo newer than file bar ?
[ -r bar ]              Is the shell option nounset set ?

[ foo -nt bar ]
[ -o nounset ]
```

Tests can be combined with logical AND and OR.

```
clim@sop:~$ [ 66 -gt 55 -a 66 -lt 500 ] && echo true || echo false
true

clim@sop:~$ [ 66 -gt 55 -a 660 -lt 500 ] && echo true || echo false
false

clim@sop:~$ [ 66 -gt 55 -o 660 -lt 500 ] && echo true || echo false
true
```

3b.2. if then else

The **if then else** construction is about choice. If a certain condition is met, then execute something, else execute something else. The example below tests whether a file exists, and if the file exists then a proper message is echoed.

```
#!/bin/bash



if [ -f isit.txt ]

then echo isit.txt exists!
else echo isit.txt not found!
fi
```

If we name the above script 'choice', then it executes like this.

```
[clim@sop scripts]$ ./choice
isit.txt not found!

[clim@sop scripts]$ touch isit.txt
[clim@sop scripts]$ ./choice isit.txt
exists!

[clim@sop scripts]$
```

3b.3. if then elif

You can nest a new **if** inside an **else** with **elif**. This is a simple example.

```
#!/bin/bash
count=42

if [ $count -eq 42 ]
then

  echo "42 is correct."
elif [ $count -gt 42 ]
then

  echo "Too much."
else

  echo "Not enough."

fi
```

3b.4. for loop

The example below shows the syntax of a classical **for loop** in bash.

```
for i in 1 2 4
do

   echo $i
done
```

An example of a **for loop** combined with an embedded shell.

```
#!/bin/ksh

for counter in `seq 1 20`
do

   echo counting from 1 to 20, now at $counter
   sleep 1

done
```

The same example as above can be written without the embedded shell using the bash **{from..to}** shorthand.

```
#!/bin/bash

for counter in {1..20}
do

   echo counting from 1 to 20, now at $counter
   sleep 1

done
```

This **for loop** uses file globbing (from the shell expansion). Putting the instruction on the command line has identical functionality.

```
stud1000@srvUbuntu$
ls count.ksh
           go.ksh

stud1000@srvUbuntu$ for file in *.ksh ; do cp $file $file.backup ;
done stud1000@srvUbuntu$ ls

count.ksh  count.ksh.backup  go.ksh  go.ksh.backup
```

3b.5. while loop

Below a simple example of a **while loop**.

```
i=100;

while [ $i -ge 0 ] ;
do

   echo Counting down, from 100 to 0, now at $i;
   let i--;

done
```

Endless loops can be made with **while true** or **while :** , where the **colon** is the equivalent of **no operation** in the **Korn** and **bash** shells.

```
#!/bin/ksh

# endless loop
while :

do

 echo hello
 sleep 1

done
```

3b.6. until loop

Below a simple example of an **until loop**.

```
let i=100;

until [ $i -le 0 ] ;
do

   echo Counting down, from 100 to 1, now at $i;
   let i--;

done
```

3b.7. practice: scripting tests and loops

3b.1.1. Write a script that uses a **for** loop to count from 3 to 7.

3b.2.1. Write a script that uses a **for** loop to count from 1 to 17000.

3b.3.1. Write a script that uses a **while** loop to count from 3 to 7.

3b.4.1. Write a script that uses an **until** loop to count down from 8 to 4.

3b.5.1. Write a script that counts the number of files ending in **.txt** in the current directory.

3b.6.1. Wrap an **if** statement around the script so it is also correct when there are zero files ending in **.txt**.