7. Loops for, while and until

In this section you'll find for, while and until loops.

The **for** loop is a little bit different from other programming languages. Basically, it let's you iterate over a series of 'words' within a string.

The **while** executes a piece of code if the control expression is true, and only stops when it is false (or a explicit break is found within the executed code.

The **until** loop is almost equal to the while loop, except that the code is executed while the control expression evaluates to false.

If you suspect that while and until are very similar you are right.

7.1 For sample

```
#!/bin/bash
for i in $( ls ); do
    echo item: $i
done
```

On the second line, we declare i to be the variable that will take the different values contained in $(ls).

The third line could be longer if needed, or there could be more lines before the done (4).

'done' (4) indicates that the code that used the value of $i has finished and $i can take a new value.

This script has very little sense, but a more useful way to use the for loop would be to use it to match only certain files on the previous example

7.2 C-like for

fiesh suggested adding this form of looping. It's a for loop more similar to C/perl... for.

```
#!/bin/bash
for i in `seq 1 10`;
do
        echo $i
done
```

7.3 While sample

```
#!/bin/bash
COUNTER=0
while [  $COUNTER -lt 10 ]; do
    echo The counter is $COUNTER
    let COUNTER=COUNTER+1
```

```
done
```

This script 'emulates' the well known (C, Pascal, perl, etc) 'for' structure

7.4 Until sample

```
#!/bin/bash
COUNTER=20
until [  $COUNTER -lt 10 ]; do
    echo COUNTER $COUNTER
    let COUNTER-=1
done
```

Next Previous Contents