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

**whether 10 is greater than 55.**

[Tux@Linux]$test 10 -gt 55 ; echo $?1

[paul@RHEL4b ~]$

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

test returns 0 when a test succeeds.

[Tux@Linux]$test 56 -gt 55 ; echo $?

0

[paul@RHEL4b ~]$

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

[Tux@Linux]$test 56 -gt 55 && echo true || echo false

true

[Tux@Linux]$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.

[Tux@Linux]$[ 56 -gt 55 ] && echo true || echo false

true

[Tux@Linux]$[ 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 ?

[ -e bar ] Does the file bar exist ?

[ '/etc' = $PWD ] Is the string /etc equal to the variable $PWD ?

Password  
[ $1 != 'Password' ] Is the first parameter different from secret ?

[ 55 -lt $bar ] Is 55 less than the value of $bar ?

[ $foo -ge 1000 ] Is the value of $foo greater or equal to 1000 ?

[ "abc" < $bar ] Does abc sort before the value of $bar ?

[ -f foo ] Is foo a regular file ?

[ -r bar ] Is bar a readable file ?

[ foo -nt bar ] Is file foo newer than file bar ?

[ -o nounset ] Is the shell option nounset set ?

Tests can be combined with logical AND and OR.

[Tux@Linux]$ [ 66 -gt 55 -a 66 -lt 500 ] && echo true || echo false

true

[Tux@Linux]$ [ 66 -gt 55 -a 660 -lt 500 ] && echo true || echo false

false

[Tux@Linux]$ [ 66 -gt 55 -o 660 -lt 500 ] && echo true || echo false

true

**20.2. 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.

[paul@RHEL4a scripts]$ ./choice

isit.txt not found!

[paul@RHEL4a scripts]$ touch isit.txt

[paul@RHEL4a scripts]$ ./choice

isit.txt exists!

[paul@RHEL4a scripts]$

**20.3. 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

20.4. 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.

kahlan@solexp11$ ls

count.ksh go.ksh

kahlan@solexp11$ for file in \*.ksh ; do cp $file $file.backup ; done

kahlan@solexp11$ ls

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

**20.5. 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

20.6. **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