

File Expressions

file1 - ef file2 file1 and file2 have the same inode numbers (the two filenames refer to the same file by hard linking).

file1 - nt file2 file1 is newer than file2.

file1 - ot file2 file1 is older than file2.

-b file file exists and is a block-special (device) file.

-c file file exists and is a character-special (device) file.

-d file file exists and is a directory.

-e file file exists.

-f file file exists and is a regular file.

-g file file exists and is set-group-ID.

-G file file exists and is owned by the effective group ID.

-k file file exists and has its "sticky bit" set.

-L file file exists and is a symbolic link.

-O file file exists and is owned by the effective user ID.

-p file file exists and is a named pipe.

-r file file exists and is readable (has readable permission for the effective user).

-s file file exists and has a length greater than zero.

-S file file exists and is a network socket.

File Expressions (cont)

-t fd fd is a file descriptor directed to/from the terminal. This can be used to determine whether standard input/output/error is being redirected.

-u file file exists and is setuid.

-w file file exists and is writable (has write permission for the effective user).

-x file file exists and is executable (has execute/search permission for the effective user).

Logical Expressions

AND -a

OR -o

NOT !

Integer Expressions

integer1 -eq integer2 integer1 is equal to integer2.

integer1 -ne integer2 integer1 is not equal to integer2.

integer1 -le integer2 integer1 is less than or equal to integer2.

integer1 -lt integer2 integer1 is less than integer2.

integer1 -ge integer2 integer1 is greater than or equal to integer2.

integer1 -gt integer2 integer1 is greater than integer2.



String Expressions

<code>string</code>	<i>string is not null.</i>
<code>-n string</code>	<i>The length of string is greater than zero.</i>
<code>-z string</code>	<i>The length of string is zero.</i>
<code>string1 = string2</code>	<i>string1 and string2 are equal.</i>
<code>string1 == string2</code>	
<code>string1 != string2</code>	<i>string1 and string2 are not equal.</i>
<code>string1 > string2</code>	<i>string1 sorts after string2.</i>
<code>string1 < string2</code>	<i>string1 sorts before string2.</i>

Modern Version of test

Instead of using `[ expression ]` is going to be `[[ expression ]]` and one designed for integers is `(( expression ))`.

With `[[ expression ]]` we can have an expression matching a regular expression:

`string =~ regex`.

For the integers in this way we can use the conventional way of other programming languages expressions like:

`int == 0`.

`int < 0`.

`int > 0`.

`int % 0`.

And of course the logical operators, instead of using the older options of test we can use:

`&&` for and.

`||` for or.

`!` for not.

