Bash Test Operators

Enjoy this cheat sheet at its fullest within Dash, the macOS documentation browser.

Everything that can be useful in test constructs (if statements) in a bash environment.

This cheat sheet is based on the Advanced Bash-Scripting Guide by Mendel Cooper.

Compound Comparison

`-a`	logical `and` Similar to `&&`
`-o`	logical `or` Similar to `||`

Integer Comparison

`-eq`	is equal to `if [ "$a" -eq "$b" ]`
`-ne`	is not equal to `if [ "$a" -ne "$b" ]`
`-gt`	is greater than `if [ "$a" -gt "$b" ]`
`-ge`	is greater than or equal to `if [ "$a" -ge "$b" ]`
`-lt`	is less than `if [ "$a" -lt "$b" ]`
`-le`	is less than or equal to `if [ "$a" -le "$b" ]`
`<`	is less than (within double parentheses) `(("$a" < "$b"))`
`<=`	is less than or equal to (within double parentheses) `(("$a" <= "$b"))`
`>`	is greater than (within double parentheses) `(("$a" > "$b"))`
`>=`	is greater than or equal to (within double parentheses) `(("$a" >= "$b"))`

String Comparison

`=`

is equal to

`==`

The == comparison operator behaves differently within a double-brackets test than within single brackets.

```
[[ $a == z* ]]   # True if $a starts with an "z" (pattern matching).
[[ $a == "z*" ]] # True if $a is equal to z* (literal matching).

[ $a == z* ]     # File globbing and word splitting take place.
[ "$a" == "z*" ] # True if $a is equal to z* (literal matching).
```

`!=`

is not equal to

```
if [ "$a" != "$b" ]
```

This operator uses pattern matching within a [[...]] construct.

`<`

is less than, in ASCII alphabetical order

Note that the `<` needs to be escaped within a [ ] construct.

```
if [[ "$a" < "$b" ]]
if [ "$a" \< "$b" ]
```

`>`

is greater than, in ASCII alphabetical order.

Note that the `>` needs to be escaped within a [ ] construct.

```
if [[ "$a" > "$b" ]]
if [ "$a" \> "$b" ]
```

`-z`

string is null

that is, has zero length

```
if [ -z "$s" ]
```

`-n`

string is not null.

```
if [ -n "$s" ]
```

File Test Operators

`-e`	file exists
`-a`	`-a` is deprecated and its use is discouraged.
`-f`	file is a regular file (not a directory or device file)
`-d`	file is a directory
`-h` `-L`	file is a symbolic link
`-b`	file is a block device
`-c`	file is a character device
`-p`	file is a pipe
`-S`	file is a socket
`-s`	file is not zero size
`-t`	file (descriptor) is associated with a terminal device This test option may be used to check whether the `stdin [ -t 0 ]` or `stdout [ -t 1 ]` in a given script is a terminal.
`-r`	file has read permission (for the user running the test)
`-w`	file has write permission (for the user running the test)
`-x`	file has execute permission (for the user running the test)
`-g`	set-group-id (sgid) flag set on file or directory
`-u`	set-user-id (suid) flag set on file
`-k`	sticky bit set
`-O`	you are owner of file
`-G`	group-id of file same as yours
`-N`	file modified since it was last read
`-nt`	file f1 is newer than f2 `if [ "$f1" -nt "$f2" ]`

`-ot`	file f1 is older than f2 `if [ "$f1" -ot "$f2" ]`
`-ef`	files f1 and f2 are hard links to the same file `if [ "$f1" -ef "$f2" ]`
`!`	"not" -- reverses the sense of the tests above (returns true if condition absent).

Notes

Based on the Advanced Bash-Scripting Guide by Mendel Cooper.

You can modify and improve this cheat sheet here