7.2. File test operators

Returns true if...

-e

file exists

-a

file exists

This is identical in effect to -e. It has been "deprecated," [1] and its use is discouraged.

-f

file is a *regular* file (not a directory or device file)

-s

file is not zero size

-d

file is a directory

-b

file is a block device

-c

file is a character device

```
device0="/dev/sda2"    # /   (root directory)
if [ -b "$device0" ]
then
  echo "$device0 is a block device."
fi

# /dev/sda2 is a block device.



device1="/dev/ttyS1"   # PCMCIA modem card.
if [ -c "$device1" ]
then
  echo "$device1 is a character device."
fi

# /dev/ttyS1 is a character device.
```

-p

file is a pipe

```
function show_input_type()
{
   [ -p /dev/fd/0 ] && echo PIPE || echo STDIN
}

show_input_type "Input"                          # STDIN
echo "Input" | show_input_type                   # PIPE

# This example courtesy of Carl Anderson.
```

-h

file is a symbolic link

-L

file is a symbolic link

-S

file is a socket

-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

If a directory has the *sgid* flag set, then a file created within that directory belongs to the group that owns the directory, not necessarily to the group of the user who created the file. This may be useful for a directory shared by a workgroup.

-u

set-user-id (suid) flag set on file

A binary owned by *root* with *set-user-id* flag set runs with *root* privileges, even when an ordinary user invokes it. [2] This is useful for executables (such as **pppd** and **cdrecord**) that need to access system hardware. Lacking the *suid* flag, these binaries could not be invoked by a *non-root* user.

```
-rwsr-xr-t    1 root       178236 Oct  2  2000 /usr/sbin/pppd
```

A file with the *suid* flag set shows an *s* in its permissions.

-k

sticky bit set

Commonly known as the *sticky bit,* the *save-text-mode* flag is a special type of file permission. If a file has this flag set, that file will be kept in cache memory, for quicker access. [3] If set on a directory, it restricts write permission. Setting the sticky bit adds a *t* to the permissions on the file or directory listing. This restricts altering or deleting specific files in that directory to the owner of those files.

```
drwxrwxrwt    7 root         1024 May 19 21:26 tmp/
```

If a user does not own a directory that has the sticky bit set, but has write permission in that directory, she can only delete those files that she owns in it. This keeps users from inadvertently overwriting or deleting each other's files in a publicly accessible directory, such as `/tmp`. (The *owner* of the directory or *root* can, of course, delete or rename files there.)

-O

you are owner of file

-G

group-id of file same as yours

-N

file modified since it was last read

f1 -nt f2

file *f1* is newer than *f2*

f1 -ot f2

file *f1* is older than *f2*

f1 -ef f2

files *f1* and *f2* are hard links to the same file

!

"not" -- reverses the sense of the tests above (returns true if condition absent).

Example 7-4. Testing for broken links

```
#!/bin/bash
# broken-link.sh
# Written by Lee bigelow <ligelowbee@yahoo.com>
# Used in ABS Guide with permission.

#  A pure shell script to find dead symlinks and output them quoted
#+ so they can be fed to xargs and dealt with :)
#+ eg. sh broken-link.sh /somedir /someotherdir|xargs rm
#
#  This, however, is a better method:
#
#  find "somedir" -type l -print0|\
#  xargs -r0 file|\
#  grep "broken symbolic"|
#  sed -e 's/^\|: *broken symbolic.*$/"/g'
```

```
#
#+ but that wouldn't be pure Bash, now would it.
#  Caution: beware the /proc file system and any circular links!
################################################################


#  If no args are passed to the script set directories-to-search
#+ to current directory.  Otherwise set the directories-to-search
#+ to the args passed.
######################

[ $# -eq 0 ] && directorys=`pwd` || directorys=$@


#  Setup the function linkchk to check the directory it is passed
#+ for files that are links and don't exist, then print them quoted.
#  If one of the elements in the directory is a subdirectory then
#+ send that subdirectory to the linkcheck function.
##########

linkchk () {
    for element in $1/*; do
      [ -h "$element" -a ! -e "$element" ] && echo \"$element\"
      [ -d "$element" ] && linkchk $element
    # Of course, '-h' tests for symbolic link, '-d' for directory.
    done
}

#  Send each arg that was passed to the script to the linkchk() function
#+ if it is a valid directoy.  If not, then print the error message
#+ and usage info.
##################
for directory in $directorys; do
    if [ -d $directory ]
        then linkchk $directory
        else
            echo "$directory is not a directory"
            echo "Usage: $0 dir1 dir2 ..."
    fi
done

exit $?
```

Example 31-1, Example 11-8, Example 11-3, Example 31-3, and Example A-1 also illustrate uses of the file test operators.

Notes

[1] Per the 1913 edition of *Webster's Dictionary*:

```
Deprecate
...

To pray against, as an evil;
to seek to avert by prayer;
to desire the removal of;
to seek deliverance from;
to express deep regret for;
to disapprove of strongly.
```

[2] Be aware that *suid* binaries may open security holes. The *suid* flag has no effect on shell scripts.

[3] On Linux systems, the sticky bit is no longer used for files, only on directories.