```
HOWTO zum Test von Bedingungen in Shell-Skripten

(C) 2006-2013 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
              OSTC Open Source Training and Consulting GmbH
              http://www.ostc.de

$Id: shell-test-HOWTO.txt,v 1.8 2019/11/26 19:37:07 tsbirn Exp $

Dieses Dokument beschreibt die verschiedenen MÃ¶glichkeiten, die das Kommando
"test" (bzw. "[...]") zum Testen von Bedingungen in der Shell bietet.

_____________________________________________________________________________

INHALTSVERZEICHNIS

1) Das Kommando "test"
2) Mit "test" Ã¼berprÃ¼fbare Bedingungen
2.1) Nur in der Bash vorhandene Bedingungen
_____________________________________________________________________________


1) Das Kommando "test"
----------------------

Das Kommando "test" Ã¼berprÃ¼ft eine Bedingung, die ihm als Argumente angegeben
wird und gibt folgende Exit-Stati zurÃ¼ck:

  +---+-------------------------------------+
  | 0 | Bedingung erfÃ¼llt ("wahr")          |
  | 1 | Bedingung NICHT erfÃ¼llt ("falsch") |
  | 2 | Syntaxfehler                        |
  +---+-------------------------------------+

Die Syntax des "test"-Kommandos lautet (Kommando "[" ist Link auf "test"):

  test COND ...      # Kommando-Form
  [ COND ] ...       # Leerzeichen um eckige Klammern notwendig!


2) Mit "test" Ã¼berprÃ¼fbare Bedingungen
--------------------------------------

Folgende Bedingungen COND sind mÃ¶glich, die einzelnen Elemente einer Bedingung
mÃ¼ssen mit Leerzeichen getrennt werden:

* Dateiexistenz und -grÃ¶Ã-^_e-Vergleich:

  -e FILE            # Datei FILE existiert [exists]
  -s FILE            # Datei FILE existent und NICHT leer [size]
  \! -s FILE         # Datei FILE existent und leer [size]

* Dateityp-Vergleich (gleichzeitig Existenz-Test):

  -f FILE            # Datei FILE normale Datei [file]
  -d FILE            # Datei FILE [directory]
  -b FILE            # Datei FILE blockorientiert [block device]
  -c FILE            # Datei FILE zeichenorientiert [character device]
  -L FILE            # Datei FILE ein symbolischer Link [link]
  -p FILE            # Datei FILE eine Named Pipe [pipe]
  -S FILE            # Datei FILE ein Socket [socket]
  -t FILE            # Datei FILE ein Terminal [terminal]

* Dateirechte-Vergleich:

  -r FILE            # Datei FILE lesbar [readable]
  -w FILE            # Datei FILE schreibbar [writable]
  -x FILE            # Datei FILE ausfÃ¼hrbar [executable]
  -g FILE            # Datei FILE hat Group-ID Recht gesetzt [group]
  -k FILE            # Datei FILE hat Sticky-Bit gesetzt [sticky]
  -u FILE            # Datei FILE hat User-ID Recht gesetzt [user]

* Text-Vergleich:

  "TEXT"             # TEXT NICHT leer (nicht mit !/-a/-o kombinierbar!)
  -n "TEXT"          # TEXT NICHT leer (mit !/-a/-o kombinierbar!) [nonzero]
  -z "TEXT"          # TEXT leer [zero]
  "TEXT1" = "TEXT2"  # TEXT1 und TEXT2 gleich
  "TEXT1" != "TEXT2" # TEXT1 und TEXT2 verschieden

* Numerischer Vergleich (nur Zahlen, kein Text oder leer erlaubt):

  NUM1 -eq NUM2      # Zahl NUM1 gleich NUM2 [equal]
```

```
  NUM1 -ne NUM2       # Zahl NUM1 NICHT gleich NUM2 [not equal]
  NUM1 -le NUM2       # Zahl NUM1 kleiner gleich NUM2 [less equal]
  NUM1 -lt NUM2       # Zahl NUM1 kleiner NUM2 [less than]
  NUM1 -ge NUM2       # Zahl NUM1 grÃ¶Ã-^_er gleich NUM2 [greater equal]
  NUM1 -gt NUM2       # Zahl NUM1 grÃ¶Ã-^_er NUM2 [greater than]

* Logische VerknÃ¼pfung (Vorrang von oben nach unten):

  \( ... \)          # Klammerung (quotiert wg. Shell!)
  \! EXPR            # Negation von EXPR [not]
  EXPR1 -a EXPR2     # EXPR1 und EXPR2 [and]
  EXPR1 -o EXPR2     # EXPR1 oder EXPR2 [or]


2.1) Nur in der Bash vorhandene Bedingungen
-------------------------------------------

* Dateiexistenz und -grÃ¶Ã-^_e-Vergleich:

  -a FILE            # Datei FILE existiert (analog "-e") [available]

* Dateityp-Vergleich (gleichzeitig Existenz-Test):

  -h FILE            # Datei FILE ein symbolischer Link (analog "-L")

* Dateibesitzer/besitzergruppe-Vergleich:

  -G FILE            # Datei FILE gehÃ¶rt effektiver Group-ID [Group]
  -O FILE            # Datei FILE gehÃ¶rt effektiver User-ID [Owner]

* Dateialter-Vergleich bzgl. Alter/Inode-Vergleich:

  -N FILE1           # Datei FILE1 seit letztem Lesen verÃ¤ndert [New]
  FILE1 -nt FILE2    # Datei FILE1 neuer als FILE2 [newer than]
  FILE1 -ot FILE2    # Datei FILE1 Ã¤lter als FILE2 [older than]
  FILE1 -ef FILE2    # Dateien haben ident. device/inode-Nummer [equal file]

* Text-Vergleich:

  "TEXT1" == "TEXT2" # TEXT1 und TEXT2 gleich (analog "=")
  "TEXT1" \< "TEXT2" # TEXT1 alphabetisch kleiner als TEXT2 (quot. wg. Shell]
  "TEXT1" \> "TEXT2" # TEXT1 alphabetisch grÃ¶Ã-^_er als TEXT2 (quot. wg. Shell]

* Shell-Variable testen:

  -v VAR             # Variable VAR gesetzt/hat einen Wert [variable]
  \! -v VAR          # Variable VAR NICHT gesetzt/undefiniert [variable]

* Shell-Option testen:

  -o OPT             # Option OPT gesetzt [option]
  \! -o OPT          # Option OPT NICHT gesetzt [option]
```