|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **16.3 test: Check file types and compare values**  test returns a status of 0 (true) or 1 (false) depending on the evaluation of the conditional expression *expr*. Each part of the expression must be a separate argument.  test has file status checks, string operators, and numeric comparison operators.  test has an alternate form that uses opening and closing square brackets instead a leading ‘test’. For example, instead of ‘test -d /’, you can write ‘[ -d / ]’. The square brackets must be separate arguments; for example, ‘[-d /]’ does not have the desired effect. Since ‘test *expr*’ and ‘[ *expr* ]’ have the same meaning, only the former form is discussed below.  Synopses:  test *expression*  test  [ *expression* ]  [ ]  [ *option*  Due to shell aliases and built-in test functions, using an unadorned test interactively or in a script may get you different functionality than that described here. Invoke it via env (i.e., env test …) to avoid interference from the shell.  If *expression* is omitted, test returns false. If *expression* is a single argument, test returns false if the argument is null and true otherwise. The argument can be any string, including strings like ‘-d’, ‘-1’, ‘--’, ‘--help’, and ‘--version’ that most other programs would treat as options. To get help and version information, invoke the commands ‘[ --help’ and ‘[ --version’, without the usual closing brackets. See [Common options](https://www.gnu.org/software/coreutils/manual/html_node/Common-options.html#Common-options).  Exit status:  0 if the expression is true,  1 if the expression is false,  2 if an error occurred.   |  |  |  | | --- | --- | --- | | • [File type tests](https://www.gnu.org/software/coreutils/manual/html_node/File-type-tests.html#File-type-tests): |  | -[bcdfhLpSt] | | • [Access permission tests](https://www.gnu.org/software/coreutils/manual/html_node/Access-permission-tests.html#Access-permission-tests): |  | -[gkruwxOG] | | • [File characteristic tests](https://www.gnu.org/software/coreutils/manual/html_node/File-characteristic-tests.html#File-characteristic-tests): |  | -e -s -nt -ot -ef | | • [String tests](https://www.gnu.org/software/coreutils/manual/html_node/String-tests.html#String-tests): |  | -z -n = == != | | • [Numeric tests](https://www.gnu.org/software/coreutils/manual/html_node/Numeric-tests.html#Numeric-tests): |  | -eq -ne -lt -le -gt -ge | | • [Connectives for test](https://www.gnu.org/software/coreutils/manual/html_node/Connectives-for-test.html#Connectives-for-test): |  | ! -a –o |  * Advance: [[ "string" =~ pattern ]] * operator =~ which performs a regular expression match of the string to its left to the expression pattern on its right. * Note that the string should be quoted, and that the regular expression shouldn't be quoted. * **Example**: check file extension if match or not: [[ ! ${FILENAME} =~ \.sig$ ]] |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| What is the difference between test, [ and [[ ? [ ("test" command) and [[ ("new test" command) are used to evaluate expressions. [[ works only in Bash, Zsh and the Korn shell, and is more powerful; [ and test are available in POSIX shells. Here are some examples:  #POSIX  [ "$variable" ] || echo 'variable is unset or empty!' >&2  [ -f "$filename" ] || printf 'File does not exist or is not a regular file: %s\n' "$filename" >&2  if [[ ! -e $file ]]; then  echo "File doesn't exist or is in an inaccessible directory or is a symlink to a file that doesn't exist: $file" >&2  fi  if [[ $file0 -nt $file1 ]]; then  printf 'file %s is newer than %s\n' "$file0" "$file1"  fi  To cut a long story short: test implements the old, portable syntax of the command. In almost all shells (the oldest Bourne shells are the exception), [ is a synonym for test(but requires a final argument of ]). Although all modern shells have built-in implementations of [, there usually still is an external executable of that name, e.g. /bin/[. [POSIX](http://pubs.opengroup.org/onlinepubs/9699919799/utilities/test.html) defines a mandatory feature set for [, but almost every shell offers extensions to it. So, if you want portable code, you should be careful not to use any of those extensions.  [[ is a new, improved version of it, and it is a keyword rather than a program. This makes it easier to use, as shown below. [[ is understood by [KornShell](http://mywiki.wooledge.org/KornShell), Zsh and [BASH](http://mywiki.wooledge.org/BASH) (as of version 2.03), but not by other POSIX shell implementations (e.g. posh, yash, or dash) or the [BourneShell](http://mywiki.wooledge.org/BourneShell).  Although [ and [[ have much in common and share many expression operators like "-f", "-s", "-n", and "-z", there are some notable differences. Here is a comparison list:   |  |  |  |  | | --- | --- | --- | --- | | **Feature** | **new test** [[ | **old test** [ | **Example** | | string comparison | > | \> [(\*)](http://mywiki.wooledge.org/BashFAQ/031#np) | [[ a > b ]] || echo "a does not come after b" | | < | \< [(\*)](http://mywiki.wooledge.org/BashFAQ/031#np) | [[ az < za ]] && echo "az comes before za" | | = (or ==) | = | [[ a = a ]] && echo "a equals a" | | != | != | [[ a != b ]] && echo "a is not equal to b" | | integer comparison | -gt | -gt | [[ 5 -gt 10 ]] || echo "5 is not bigger than 10" | | -lt | -lt | [[ 8 -lt 9 ]] && echo "8 is less than 9" | | -ge | -ge | [[ 3 -ge 3 ]] && echo "3 is greater than or equal to 3" | | -le | -le | [[ 3 -le 8 ]] && echo "3 is less than or equal to 8" | | -eq | -eq | [[ 5 -eq 05 ]] && echo "5 equals 05" | | -ne | -ne | [[ 6 -ne 20 ]] && echo "6 is not equal to 20" | | conditional evaluation | && | -a [(\*\*)](http://mywiki.wooledge.org/BashFAQ/031#np2) | [[ -n $var && -f $var ]] && echo "$var is a file" | | || | -o [(\*\*)](http://mywiki.wooledge.org/BashFAQ/031#np2) | [[ -b $var || -c $var ]] && echo "$var is a device" | | expression grouping | (...) | \( ... \) [(\*\*)](http://mywiki.wooledge.org/BashFAQ/031#np2) | [[ $var = img\* && ($var = \*.png || $var = \*.jpg) ]] && echo "$var starts with img and ends with .jpg or .png" | | Pattern matching | = (or ==) | (not available) | [[ $name = a\* ]] || echo "name does not start with an 'a': $name" | | [RegularExpression](http://mywiki.wooledge.org/RegularExpression) matching | =~ | (not available) | [[ $(date) =~ ^Fri\ ...\ 13 ]] && echo "It's Friday the 13th!" |   (\*) This is an extension to the POSIX standard; some shells may have it, others may not.  (\*\*) The -a and -o operators, and ( ... ) grouping, are defined by POSIX but only for strictly limited cases, and are marked as deprecated. Use of these operators is discouraged; you should use multiple [ commands instead:   * if [ "$a" = a ] && [ "$b" = b ]; then ... * if [ "$a" = a ] || { [ "$b" = b ] && [ "$c" = c ];}; then ...   Special primitives that [[ is defined to have, but [ may be lacking (depending on the implementation):   |  |  |  | | --- | --- | --- | | **Description** | **Primitive** | **Example** | | entry (file or directory) exists | -e | [[ -e $config ]] && echo "config file exists: $config" | | file is newer/older than other file | -nt / -ot | [[ $file0 -nt $file1 ]] && echo "$file0 is newer than $file1" | | two files are the same | -ef | [[ $input -ef $output ]] && { echo "will not overwrite input file: $input"; exit 1; } | | negation | ! | [[ ! -u $file ]] && echo "$file is not a setuid file" | |