bash(1) — Linux manual page

NAME top

bash - GNU Bourne-Again SHell

SYNOPSIS top

bash [options] [command_string | file]

COPYRIGHT top

DESCRIPTION top

Bash is an **sh**-compatible command language interpreter that executes commands read from the standard input or from a file. **Bash** also incorporates useful features from the *Korn* and *C* shells (**ksh** and **csh**).

Bash is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification (IEEE Standard 1003.1). **Bash** can be configured to be POSIX-conformant by default.

OPTIONS top

All of the single-character shell options documented in the description of the **set** builtin command, including **-o**, can be used as options when the shell is invoked. In addition, **bash** interprets the following options when it is invoked:

-c If the **-c** option is present, then commands are read from the first non-option argument *command_string*. If there are arguments after the *command_string*, the first argument is assigned to **$0** and any remaining arguments are assigned to the positional parameters. The assignment to **$0** sets the

[[*expression* **]]**

Return a status of 0 or 1 depending on the evaluation of the conditional expression *expression*. Expressions are composed of the primaries described below under **CONDITIONAL EXPRESSIONS**. Word splitting and pathname expansion are not performed on the words between the **[[** and **]]**; tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution, process substitution, and quote removal are performed. Conditional operators such as **-f** must be unquoted to be recognized as primaries.

When used with **[[**, the **<** and **>** operators sort lexicographically using the current locale.

When the **==** and **!=** operators are used, the string to the right of the operator is considered a pattern and matched according to the rules described below under **Pattern Matching**, as if the **extglob** shell option were enabled. The **=** operator is equivalent to **==**. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters. The return value is 0 if the string matches (**==**) or does not match (**!=**) the pattern, and 1 otherwise. Any part of the pattern may be quoted to force the quoted portion to be matched as a string.

An additional binary operator, **=~**, is available, with the same precedence as **==** and **!=**. When it is used, the string to the right of the operator is considered a POSIX extended regular expression and matched accordingly (as in regex(3)). The return value is 0 if the string matches the pattern, and 1 otherwise. If the regular expression is syntactically incorrect, the conditional expression's return value is 2. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters. Any part of the pattern may be quoted to force the quoted portion to be matched as a string. Bracket expressions in regular expressions must be treated carefully, since normal quoting characters lose their meanings between brackets. If the pattern is stored in a shell variable, quoting the variable expansion forces the entire pattern to be matched as a string. Substrings matched by parenthesized subexpressions within the regular expression are saved in the array variable **BASH_REMATCH**. The element of **BASH_REMATCH** with index 0 is the portion of the string matching the entire regular expression. The element of **BASH_REMATCH** with index *n* is the portion of the string matching the *n*th parenthesized subexpression.

Expressions may be combined using the following operators, listed in decreasing order of precedence:

(*expression* **)**
: Returns the value of *expression*. This may be used to override the normal precedence of operators.

! *expression*
: True if *expression* is false.

expression1 **&&** *expression2*
: True if both *expression1* and *expression2* are true.