NAME

getopt – parse command options (enhanced)

SYNOPSIS

getopt *optstring parameters*

getopt [options] [−−] *optstring parameters*

getopt [options] **−o**|**−−options** *optstring* [options] [−−] *parameters*

DESCRIPTION

getopt is used to break up (*parse*) options in command lines for easy parsing by shell procedures, and to check for valid options. It uses the GNU **getopt**(3) routines to do this.

The parameters **getopt** is called with can be divided into two parts: options which modify the way **getopt** will do the parsing (the *options* and the *optstring* in the **SYNOPSIS**), and the parameters which are to be parsed (*parameters* in the **SYNOPSIS**). The second part will start at the first non−option parameter that is not an option argument, or after the first occurrence of '−−'. If no '−o' or '−−options' option is found in the first part, the first parameter of the second part is used as the short options string.

If the environment variable **GETOPT_COMPATIBLE** is set, or if the first *parameter* is not an option (does not start with a '−', the first format in the **SYNOPSIS**), **getopt** will generate output that is compatible with that of other versions of **getopt**(1). It will still do parameter shuffling and recognize optional arguments (see the **COMPATIBILITY** section for more information).

Traditional implementations of **getopt**(1) are unable to cope with whitespace and other (shell−specific) special characters in arguments and non−option parameters. To solve this problem, this implementation can generate quoted output which must once again be interpreted by the shell (usually by using the **eval** command). This has the effect of preserving those characters, but you must call **getopt** in a way that is no longer compatible with other versions (the second or third format in the **SYNOPSIS**). To determine whether this enhanced version of **getopt**(1) is installed, a special test option (−**T**) can be used.

OPTIONS

−a, **−−alternative**
Allow long options to start with a single '−'.

−l, **−−longoptions** *longopts*
The long (multi−character) options to be recognized. More than one option name may be specified at once, by separating the names with commas. This option may be given more than once, the *longopts* are cumulative. Each long option name in *longopts* may be followed by one colon to indicate it has a required argument, and by two colons to indicate it has an optional argument.

−n, **−−name** *progname*
The name that will be used by the **getopt**(3) routines when it reports errors. Note that errors of **getopt**(1) are still reported as coming from getopt.

−o, **−−options** *shortopts*
The short (one−character) options to be recognized. If this option is not found, the first parameter of **getopt** that does not start with a '−' (and is not an option argument) is used as the short options string. Each short option character in *shortopts* may be followed by one colon to indicate it has a required argument, and by two colons to indicate it has an optional argument. The first character of shortopts may be '+' or '−' to influence the way options are parsed and output is generated (see the **SCANNING MODES** section for details).

−q, **−−quiet**
Disable error reporting by **getopt**(3).

−Q, **−−quiet−output**
>   Do not generate normal output. Errors are still reported by **getopt**(3), unless you also use **−q**.

−s, **−−shell** *shell*
>   Set quoting conventions to those of *shell*. If the **−s** option is not given, the **BASH** conventions are
>   used. Valid arguments are currently '**sh**', '**bash**', '**csh**', and '**tcsh**'.

−T, **−−test**
>   Test if your **getopt**(1) is this enhanced version or an old version. This generates no output, and sets the
>   error status to 4. Other implementations of **getopt**(1), and this version if the environment variable
>   **GETOPT_COMPATIBLE** is set, will return '−−' and error status 0.

−u, **−−unquoted**
>   Do not quote the output. Note that whitespace and special (shell−dependent) characters can cause
>   havoc in this mode (like they do with other **getopt**(1) implementations).

−h, **−−help**
>   Display help text and exit.

−V, **−−version**
>   Print version and exit.

PARSING

>   This section specifies the format of the second part of the parameters of **getopt** (the *parameters* in the
>   **SYNOPSIS**). The next section (**OUTPUT**) describes the output that is generated. These parameters were
>   typically the parameters a shell function was called with. Care must be taken that each parameter the shell
>   function was called with corresponds to exactly one parameter in the parameter list of **getopt** (see the
>   **EXAMPLES**). All parsing is done by the GNU **getopt**(3) routines.

>   The parameters are parsed from left to right. Each parameter is classified as a short option, a long option, an
>   argument to an option, or a non−option parameter.

>   A simple short option is a '−' followed by a short option character. If the option has a required argument, it
>   may be written directly after the option character or as the next parameter (i.e., separated by whitespace on
>   the command line). If the option has an optional argument, it must be written directly after the option
>   character if present.

>   It is possible to specify several short options after one '−', as long as all (except possibly the last) do not
>   have required or optional arguments.

>   A long option normally begins with '−−' followed by the long option name. If the option has a required
>   argument, it may be written directly after the long option name, separated by '=', or as the next argument
>   (i.e., separated by whitespace on the command line). If the option has an optional argument, it must be
>   written directly after the long option name, separated by '=', if present (if you add the '=' but nothing behind
>   it, it is interpreted as if no argument was present; this is a slight bug, see the **BUGS**). Long options may be
>   abbreviated, as long as the abbreviation is not ambiguous.

>   Each parameter not starting with a '−', and not a required argument of a previous option, is a non−option
>   parameter. Each parameter after a '−−' parameter is always interpreted as a non−option parameter. If the
>   environment variable **POSIXLY_CORRECT** is set, or if the short option string started with a '+', all
>   remaining parameters are interpreted as non−option parameters as soon as the first non−option parameter is
>   found.

OUTPUT

>   Output is generated for each element described in the previous section. Output is done in the same order as
>   the elements are specified in the input, except for non−option parameters. Output can be done in *compatible*

(*unquoted*) mode, or in such way that whitespace and other special characters within arguments and non−option parameters are preserved (see **QUOTING**). When the output is processed in the shell script, it will seem to be composed of distinct elements that can be processed one by one (by using the shift command in most shell languages). This is imperfect in unquoted mode, as elements can be split at unexpected places if they contain whitespace or special characters.

If there are problems parsing the parameters, for example because a required argument is not found or an option is not recognized, an error will be reported on stderr, there will be no output for the offending element, and a non−zero error status is returned.

For a short option, a single '−' and the option character are generated as one parameter. If the option has an argument, the next parameter will be the argument. If the option takes an optional argument, but none was found, the next parameter will be generated but be empty in quoting mode, but no second parameter will be generated in unquoted (compatible) mode. Note that many other **getopt**(1) implementations do not support optional arguments.

If several short options were specified after a single '−', each will be present in the output as a separate parameter.

For a long option, '−−' and the full option name are generated as one parameter. This is done regardless whether the option was abbreviated or specified with a single '−' in the input. Arguments are handled as with short options.

Normally, no non−option parameters output is generated until all options and their arguments have been generated. Then '−−' is generated as a single parameter, and after it the non−option parameters in the order they were found, each as a separate parameter. Only if the first character of the short options string was a '−', non−option parameter output is generated at the place they are found in the input (this is not supported if the first format of the **SYNOPSIS** is used; in that case all preceding occurrences of '−' and '+' are ignored).

QUOTING

In compatibility mode, whitespace or 'special' characters in arguments or non−option parameters are not handled correctly. As the output is fed to the shell script, the script does not know how it is supposed to break the output into separate parameters. To circumvent this problem, this implementation offers quoting. The idea is that output is generated with quotes around each parameter. When this output is once again fed to the shell (usually by a shell **eval** command), it is split correctly into separate parameters.

Quoting is not enabled if the environment variable **GETOPT_COMPATIBLE** is set, if the first form of the **SYNOPSIS** is used, or if the option '−**u**' is found.

Different shells use different quoting conventions. You can use the '−**s**' option to select the shell you are using. The following shells are currently supported: '**sh**', '**bash**', '**csh**' and '**tcsh**'. Actually, only two 'flavors' are distinguished: sh−like quoting conventions and csh−like quoting conventions. Chances are that if you use another shell script language, one of these flavors can still be used.

SCANNING MODES

The first character of the short options string may be a '−' or a '+' to indicate a special scanning mode. If the first calling form in the **SYNOPSIS** is used they are ignored; the environment variable **POSIXLY_CORRECT** is still examined, though.

If the first character is '+', or if the environment variable **POSIXLY_CORRECT** is set, parsing stops as soon as the first non−option parameter (i.e., a parameter that does not start with a '−') is found that is not an option argument. The remaining parameters are all interpreted as non−option parameters.

If the first character is a '−', non−option parameters are outputted at the place where they are found; in normal operation, they are all collected at the end of output after a '−−' parameter has been generated. Note

that this '−−' parameter is still generated, but it will always be the last parameter in this mode.

COMPATIBILITY

This version of **getopt**(1) is written to be as compatible as possible to other versions. Usually you can just replace them with this version without any modifications, and with some advantages.

If the first character of the first parameter of getopt is not a '−', **getopt** goes into compatibility mode. It will interpret its first parameter as the string of short options, and all other arguments will be parsed. It will still do parameter shuffling (i.e., all non−option parameters are output at the end), unless the environment variable **POSIXLY_CORRECT** is set, in which case, **getopt** will prepend a '+' before short options automatically.

The environment variable **GETOPT_COMPATIBLE** forces **getopt** into compatibility mode. Setting both this environment variable and **POSIXLY_CORRECT** offers 100% compatibility for 'difficult' programs. Usually, though, neither is needed.

In compatibility mode, leading '−' and '+' characters in the short options string are ignored.

RETURN CODES

getopt returns error code **0** for successful parsing, **1** if **getopt**(3) returns errors, **2** if it does not understand its own parameters, **3** if an internal error occurs like out−of−memory, and **4** if it is called with **−T**.

EXAMPLES

Example scripts for (ba)sh and (t)csh are provided with the **getopt**(1) distribution, and are installed in */usr/share/doc/util−linux* directory.

ENVIRONMENT

POSIXLY_CORRECT

This environment variable is examined by the **getopt**(3) routines. If it is set, parsing stops as soon as a parameter is found that is not an option or an option argument. All remaining parameters are also interpreted as non−option parameters, regardless whether they start with a '−'.

GETOPT_COMPATIBLE

Forces **getopt** to use the first calling format as specified in the **SYNOPSIS**.

BUGS

getopt(3) can parse long options with optional arguments that are given an empty optional argument (but cannot do this for short options). This **getopt**(1) treats optional arguments that are empty as if they were not present.

The syntax if you do not want any short option variables at all is not very intuitive (you have to set them explicitly to the empty string).

AUTHOR

Frodo Looijaard <frodo@frodo.looijaard.name>

SEE ALSO

bash(1), **tcsh**(1), **getopt**(3)

REPORTING BUGS

For bug reports, use the issue tracker at https://github.com/util−linux/util−linux/issues.

AVAILABILITY

The **getopt** command is part of the util−linux package which can be downloaded from Linux Kernel Archive <https://www.kernel.org/pub/linux/utils/util−linux/>.