



## 8.117. Getopt::Long

Lets your program accept command-line options with long names, introduced by `--`. Standard single-character options are also accepted. Options that start with `--` may have an argument appended, following a space or an equals sign (=):

```
--foo=bar
--foo bar
```

Provides two functions: `GetOptions` and `config`.

### `config`

---

```
Getopt::Long::config(optionlist)
```

Sets the variables in *optionlist* to change the default behavior of `GetOptions`. The following options are available:

#### `$Getopt::Long::autoabbrev`

If true, option names can be invoked with unique abbreviations. Default is 1 (true) unless the environment variable `POSIXLY_CORRECT` has been set.

#### `$Getopt::Long::getopt_compat`

If true, options can start with `+`. Default is 1 unless the environment variable `POSIXLY_CORRECT` has been set.

#### `$Getopt::Long::order`

Value indicates whether options and non-options may be mixed on the command line:

#### `$PERMUTE`

Non-options may be mixed with options. The default if `POSIXLY_CORRECT` is not set.

#### `$REQUIRE_ORDER`

Mixing is not allowed. The default if `POSIXLY_CORRECT` is set.

#### `$Getopt::Long::ignorecase`

If true, ignore case when matching options. Default is 1.

#### `$Getopt::Long::VERSION`

The version number of this `Getopt::Long` implementation in the format *major.minor*.

#### `$Getopt::Long::error`

Internal error flag. May be incremented from a callback routine to cause options-parsing to fail.

### **\$Getopt::Long::debug**

If true, enables debugging output. Default is 0 (false).

## **GetOptions**

---

```
$result = GetOptions(option-descriptions)
```

Uses descriptions from *option-descriptions* to retrieve and process the command-line options with which your Perl program was invoked. The options are taken from @ARGV. After `GetOptions` has processed the options, @ARGV contains only command-line arguments that were not options. Returns 0 if errors are detected. Each option description consists of two elements:

### *Option specifier*

Defines the option name and optionally a value as an argument specifier.

### *Option linkage*

A reference to a variable that is set when the option is present.

`GetOptions` can also take as a first argument a reference to a hash that describes the linkage for the options. The linkage specified in the argument list takes precedence over the one specified in the hash. Thus, the following are equivalent:

```
%optctl = (size => \ $offset);
&GetOptions(\%optctl, "size=i");
```

and:

```
&GetOptions("size=i" => \ $offset);
```

## **Option specifiers**

Each option specifier consists of an option name and possibly an argument specifier. The name can be a name, or a list of names separated by |; the first name in the list is the true name of the option, and the others are treated as aliases. Option names may be invoked with the shortest unique abbreviation. Values for argument specifiers are:

### **<none>**

Option takes no argument. The option variable is set to 1.

### **!**

Option does not take an argument and may be negated, that is, prefixed by `no`.

### **=s**

Option takes a mandatory argument that is a string that will be assigned to the option variable. Even if the argument starts with `-` or `--`, it is assigned to the option variable rather than treated as another option.

**:s**

Option takes an optional string argument. If the option is invoked with no argument, an empty string ("") is assigned to the option variable. If the argument starts with `-` or `--`, it is treated as another option rather than assigned to the option variable.

**=i**

Option takes a mandatory integer argument, which may start with `-` to indicate a negative value.

**:i**

Option takes an optional integer argument that may start with `-` to indicate a negative value. With no argument, the value 0 is assigned to the option variable.

**=f**

Option takes a mandatory floating-point argument that may start with `-` to indicate a negative value.

**:f**

Option takes an optional floating-point argument that may start with `-` to indicate a negative value. With no argument, the value 0 is assigned to the option variable.

A hyphen (`-`) by itself is considered an option whose name is the empty string. A double hyphen (`--`) by itself terminates option processing. Any options following the double hyphen remain in `@ARGV` when `GetOptions` returns. If an argument specifier ends with `@` (e.g., `=s@`), then the option is treated as an array.

The special option specifier `<>` can be used to designate a subroutine to handle non-option arguments. For this specifier to be used, the variable `$Getopt::Long::order` must have the value of the predefined and exported variable, `$PERMUTE`. See the description of `Getopt::Long::config` below.

## Linkage specification

The linkage specifier can be a reference to any of the following:

### *Scalar*

The new value is stored in the referenced variable. If the option occurs more than once, the previous value is overwritten.

### *Array*

The new value is appended (pushed) onto the referenced array.

### *Subroutine*

The referenced subroutine is called with two arguments: the option name, which is always the true name, and the option value.

If no linkage is explicitly specified, but a hash reference is passed, `GetOptions` puts the value in the hash. For array options, a reference to an anonymous array is generated.

If no linkage is explicitly specified, and no hash reference is passed, `GetOptions` puts the value into a global variable named after the option, prefixed by `opt_`. Characters that are not part of the variable syntax are translated to

underscores. For example, `--fpp-struct-return` sets the variable `$opt_fpp_struct_return`.

---

**◀ PREVIOUS**

**HOME**

**NEXT ▶**

8.116. GDBM\_File

**BOOK INDEX**

8.118. Getopt::Std

---



[Copyright © 2002](#) O'Reilly & Associates. All rights reserved.