

NAME

getopt_long, **getopt_long_only** - get long options from command line argument list

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <getopt.h>

```
extern char *optarg;
extern int optind;
extern int optopt;
extern int opterr;
extern int optreset;
```

int

```
getopt_long(int argc, char * const *argv, const char *optstring, const struct option *longopts,
             int *longindex);
```

int

```
getopt_long_only(int argc, char * const *argv, const char *optstring, const struct option *longopts,
                  int *longindex);
```

DESCRIPTION

The **getopt_long**() function is similar to **getopt(3)** but it accepts options in two forms: words and characters. The **getopt_long**() function provides a superset of the functionality of **getopt(3)**. The **getopt_long**() function can be used in two ways. In the first way, every long option understood by the program has a corresponding short option, and the option structure is only used to translate from long options to short options. When used in this fashion, **getopt_long**() behaves identically to **getopt(3)**. This is a good way to add long option processing to an existing program with the minimum of rewriting.

In the second mechanism, a long option sets a flag in the *option* structure passed, or will store a pointer to the command line argument in the *option* structure passed to it for options that take arguments. Additionally, the long option's argument may be specified as a single argument with an equal sign, e.g.,

```
myprogram --myoption=somevalue
```

When a long option is processed, the call to **getopt_long**() will return 0. For this reason, long option processing without shortcuts is not backwards compatible with **getopt(3)**.

It is possible to combine these methods, providing for long options processing with short option equivalents for some options. Less frequently used options would be processed as long options only.

The **getopt_long()** call requires a structure to be initialized describing the long options. The structure is:

```
struct option {
    char *name;
    int has_arg;
    int *flag;
    int val;
};
```

The *name* field should contain the option name without the leading double dash.

The *has_arg* field should be one of:

no_argument	no argument to the option is expected
required_argument	an argument to the option is required
optional_argument	an argument to the option may be presented

If *flag* is not NULL, then the integer pointed to by it will be set to the value in the *val* field. If the *flag* field is NULL, then the *val* field will be returned. Setting *flag* to NULL and setting *val* to the corresponding short option will make this function act just like **getopt(3)**.

If the *longindex* field is not NULL, then the integer pointed to by it will be set to the index of the long option relative to *longopts*.

The last element of the *longopts* array has to be filled with zeroes.

The **getopt_long_only()** function behaves identically to **getopt_long()** with the exception that long options may start with '-' in addition to '--'. If an option starting with '-' does not match a long option but does match a single-character option, the single-character option is returned.

RETURN VALUES

If the *flag* field in *struct option* is NULL, **getopt_long()** and **getopt_long_only()** return the value specified in the *val* field, which is usually just the corresponding short option. If *flag* is not NULL, these functions return 0 and store *val* in the location pointed to by *flag*.

These functions return ':' if there was a missing option argument and error messages are suppressed, '?'

if the user specified an unknown or ambiguous option, and -1 when the argument list has been exhausted. The default behavior when a missing option argument is encountered is to write an error and return '?'. Specifying ':' in *optstr* will cause the error message to be suppressed and ':' to be returned instead.

In addition to ':', a leading '+' or '-' in *optstr* also has special meaning. If either of these are specified, they must appear before ':'.

A leading '+' indicates that processing should be halted at the first non-option argument, matching the default behavior of `getopt(3)`. The default behavior without '+' is to permute non-option arguments to the end of *argv*.

A leading '-' indicates that all non-option arguments should be treated as if they are arguments to a literal '1' flag (i.e., the function call will return the value 1, rather than the char literal '1').

ENVIRONMENT

`POSIXLY_CORRECT` If set, option processing stops when the first non-option is found and a leading '-' or '+' in the *optstring* is ignored.

EXAMPLES

```
int bflag, ch, fd;
int daggerset;
```

```
/* options descriptor */
static struct option longopts[] = {
    { "buffy",no_argument,          NULL,          'b' },
    { "fluoride",    required_argument, NULL,          'f' },
    { "daggerset",   no_argument,      &daggerset,    1 },
    { NULL,          0,                NULL,          0 }
};
```

```
bflag = 0;
while ((ch = getopt_long(argc, argv, "bf:", longopts, NULL)) != -1) {
    switch (ch) {
        case 'b':
            bflag = 1;
            break;
        case 'f':
            if ((fd = open(optarg, O_RDONLY, 0)) == -1)
                err(1, "unable to open %s", optarg);
```

```

        break;
    case 0:
        if (daggerset) {
            fprintf(stderr, "Buffy will use her dagger to "
                "apply fluoride to dracula's teeth\n");
        }
        break;
    default:
        usage();
    }
}
argc -= optind;
argv += optind;

```

IMPLEMENTATION DIFFERENCES

This section describes differences to the GNU implementation found in glibc-2.1.3:

- Setting of *optopt* for long options with *flag* != NULL:

GNU

sets *optopt* to *val*.

BSD

sets *optopt* to 0 (since *val* would never be returned).

- Setting of *optarg* for long options without an argument that are invoked via '-W' ('W;' in option string):

GNU

sets *optarg* to the option name (the argument of '-W').

BSD

sets *optarg* to NULL (the argument of the long option).

- Handling of '-W' with an argument that is not (a prefix to) a known long option ('W;' in option string):

GNU

returns '-W' with *optarg* set to the unknown option.

BSD

treats this as an error (unknown option) and returns '?' with *optopt* set to 0 and *optarg* set to NULL (as GNU's man page documents).

- BSD does not permute the argument vector at the same points in the calling sequence as GNU does. The aspects normally used by the caller (ordering after -1 is returned, value of *optind* relative to current positions) are the same, though. (We do fewer variable swaps.)

SEE ALSO

getopt(3)

HISTORY

The **getopt_long()** and **getopt_long_only()** functions first appeared in the GNU libiberty library. The first BSD implementation of **getopt_long()** appeared in NetBSD 1.5, the first BSD implementation of **getopt_long_only()** in OpenBSD 3.3. FreeBSD first included **getopt_long()** in FreeBSD 5.0, **getopt_long_only()** in FreeBSD 5.2.

BUGS

The *argv* argument is not really *const* as its elements may be permuted (unless POSIXLY_CORRECT is set).

The implementation can completely replace getopt(3), but right now we are using separate code.

getopt_long makes the assumption that the first argument should always be skipped because it's typically the program name. As a result, setting *optind* to 0 will indicate that **getopt_long** should reset, and *optind* will be set to 1 in the process. This behavior differs from getopt(3), which will handle an *optind* value of 0 as expected and process the first element.