getopt - Man Page

command option parsing

Prolog

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

Synopsis

```
#include <unistd.h>

int getopt(int argc, char * const argv[], const char *optstr
extern char *optarg;
extern int opterr, optind, optopt;
```

Description

The *getopt*() function is a command-line parser that shall follow Utility Syntax Guidelines 3, 4, 5, 6, 7, 9, and 10 in the Base Definitions volume of POSIX.1-2017, *Section 12.2*, *Utility Syntax Guidelines*.

The parameters *argc* and *argv* are the argument count and argument array as passed to *main*() (see *exec*()). The argument *optstring* is a string of recognized option characters; if a character is followed by a <colon>, the option takes an argument. All option characters allowed by Utility Syntax Guideline 3 are allowed in *optstring*. The implementation may accept other characters as an extension.

getopt() shall update it when it finishes with each element of *argv*[]. If the application sets *optind* to zero before calling *getopt*(), the behavior is unspecified. When an element of *argv*[] contains multiple option characters, it is unspecified how *getopt*() determines which options have already been processed.

The *getopt*() function shall return the next option character (if one is found) from *argv* that matches a character in *optstring*, if there is one that matches. If the option takes an argument, *getopt*() shall set the variable *optarg* to point to the option-argument as follows:

1. If the option was the last character in the string pointed to by an element of *argv*, then *optarg* shall contain the next element of *argv*, and *optind* shall be incremented by 2. If the resulting value of *optind* is greater than *argc*, this indicates a missing option-argument, and *getopt*() shall return an error indication.
2. Otherwise, *optarg* shall point to the string following the option character in that element of *argv*, and *optind* shall be incremented by 1.

If, when *getopt*() is called:

```
 argv[optind]  is a null pointer
*argv[optind]  is not the character -
 argv[optind]  points to the string "-"
```

getopt() shall return -1 without changing *optind*. If:

```
argv[optind]   points to the string "--"
```

getopt() shall return -1 after incrementing *optind*.

If *getopt*() encounters an option character that is not contained in *optstring*, it shall return the <question-mark> (**'?'**) character. If it detects a missing option-argument, it shall return the <colon>

shall set the variable *optopt* to the option character that caused the error. If the application has not set the variable *opterr* to 0 and the first character of *optstring* is not a <colon>, *getopt*() shall also print a diagnostic message to *stderr* in the format specified for the *getopts* utility, unless the *stderr* stream has wide orientation, in which case the behavior is undefined.

The *getopt*() function need not be thread-safe.

Return Value

The *getopt*() function shall return the next option character specified on the command line.

A <colon> (**':'**) shall be returned if *getopt*() detects a missing argument and the first character of *optstring* was a <colon> (**':'**).

A <question-mark> (**'?'**) shall be returned if *getopt*() encounters an option character not in *optstring* or detects a missing argument and the first character of *optstring* was not a <colon> (**':'**).

Otherwise, *getopt*() shall return -1 when all command line options are parsed.

Errors

If the application has not set the variable *opterr* to 0, the first character of *optstring* is not a <colon>, and a write error occurs while *getopt*() is printing a diagnostic message to *stderr*, then the error indicator for *stderr* shall be set; but *getopt*() shall still succeed and the value of *errno* after *getopt*() is unspecified.

The following sections are informative.

getopt - Man Page

-

Parsing Command Line Options

The following code fragment shows how you might process the arguments for a utility that can take the mutually-exclusive options *a* and *b* and the options *f* and *o*, both of which require arguments:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int
main(int argc, char *argv[ ])
{
    int c;
    int bflg = 0, aflg = 0, errflg = 0;
    char *ifile;
    char *ofile;
    . . .
    while ((c = getopt(argc, argv, ":abf:o:")) != -1) {
        switch(c) {
        case 'a':
            if (bflg)
                errflg++;
            else
                aflg++;
            break;
        case 'b':
            if (aflg)
                errflg++;
            else
                bflg++;
            break;
        case 'f':
            ifile = optarg;
```

```
                break;
            case ':':       /* -f or -o without operand */
                fprintf(stderr,
                    "Option -%c requires an operand\n", opto
                errflg++;
                break;
            case '?':
                fprintf(stderr,
                    "Unrecognized option: '-%c'\n", optopt);
                errflg++;
            }
        }
        if (errflg) {
            fprintf(stderr, "usage: . . . ");
            exit(2);
        }
        for ( ; optind < argc; optind++) {
            if (access(argv[optind], R_OK)) {
        . . .
    }
```

This code accepts any of the following as equivalent:

```
    cmd -ao arg path path
    cmd -a -o arg path path
    cmd -o arg -a path path
    cmd -a -o arg -- path path
    cmd -a -oarg path path
    cmd -aoarg path path
```

Selecting Options from the Command Line

The following example selects the type of database routines the user wants to use based on the *Options* argument.

getopt - Man Page

```
const char *Options = "hdbtl";
...
int dbtype, c;
char *st;
...
dbtype = 0;
while ((c = getopt(argc, argv, Options)) != -1) {
    if ((st = strchr(Options, c)) != NULL) {
        dbtype = st - Options;
        break;
    }
}
```

Application Usage

The *getopt*() function is only required to support option characters included in Utility Syntax Guideline 3. Many historical implementations of *getopt*() support other characters as options. This is an allowed extension, but applications that use extensions are not maximally portable. Note that support for multi-byte option characters is only possible when such characters can be represented as type **int**.

Applications which use wide-character output functions with *stderr* should ensure that any calls to *getopt*() do not write to *stderr*, either by setting *opterr* to 0 or by ensuring the first character of *optstring* is always a <colon>.

While *ferror*(*stderr*) may be used to detect failures to write a diagnostic to *stderr* when *getopt*() returns **'?'**, the value of *errno* is unspecified in such a condition. Applications desiring more control over handling write failures should set *opterr* to 0 and independently perform output to *stderr*, rather than relying on *getopt*() to do the output.

getopt - Man Page

The *optopt* variable represents historical practice and allows the application to obtain the identity of the invalid option.

The description has been written to make it clear that *getopt*(), like the *getopts* utility, deals with option-arguments whether separated from the option by <blank> characters or not. Note that the requirements on *getopt*() and *getopts* are more stringent than the Utility Syntax Guidelines.

The *getopt*() function shall return -1, rather than EOF, so that *<stdio.h>* is not required.

The special significance of a <colon> as the first character of *optstring* makes *getopt*() consistent with the *getopts* utility. It allows an application to make a distinction between a missing argument and an incorrect option letter without having to examine the option letter. It is true that a missing argument can only be detected in one case, but that is a case that has to be considered.

Future Directions

None.

See Also

exec

The Base Definitions volume of POSIX.1-2017, *Section 12.2*, *Utility Syntax Guidelines*, <unistd.h>

The Shell and Utilities volume of POSIX.1-2017, getopts

Referenced By

getopts(1p), getsubopt(3p), stdio.h(0p), unistd.h(0p).

2017 IEEE/The Open Group POSIX Programmer's Manual

getopt - Man Page

Home Blog About