NAME

getopt - get option character from command line argument list

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <unistd.h>

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

int
getopt(*int argc*, *char * const argv[]*, *const char *optstring*);

DESCRIPTION

The **getopt**() function incrementally parses a command line argument list *argv* and returns the next *known* option character. An option character is *known* if it has been specified in the string of accepted option characters, *optstring*.

The option string *optstring* may contain the following elements: individual characters, and characters followed by a colon to indicate an option argument is to follow. If an individual character is followed by two colons, then the option argument is optional; *optarg* is set to the rest of the current *argv* word, or NULL if there were no more characters in the current word. This is a GNU extension. For example, an option string "x" recognizes an option "**-x**", and an option string "x:" recognizes an option and argument "**-x** *argument*". It does not matter to **getopt**() if a following argument has leading white space.

On return from **getopt**(), *optarg* points to an option argument, if it is anticipated, and the variable *optind* contains the index to the next *argv* argument for a subsequent call to **getopt**(). The variable *optopt* saves the last *known* option character returned by **getopt**().

The variables *opterr* and *optind* are both initialized to 1. The *optind* variable may be set to another value before a set of calls to **getopt**() in order to skip over more or less argv entries.

In order to use **getopt**() to evaluate multiple sets of arguments, or to evaluate a single set of arguments multiple times, the variable *optreset* must be set to 1 before the second and each additional set of calls to **getopt**(), and the variable *optind* must be reinitialized.

The **getopt**() function returns -1 when the argument list is exhausted. The interpretation of options in the argument list may be cancelled by the option '--' (double dash) which causes **getopt**() to signal the end of argument processing and return -1. When all options have been processed (i.e., up to the first non-option argument), **getopt**() returns -1.

RETURN VALUES

The **getopt**() function returns the next known option character in *optstring*. If **getopt**() encounters a character not found in *optstring* or if it detects a missing option argument, it returns '?' (question mark). If *optstring* has a leading ':' then a missing option argument causes ':' to be returned instead of '?'. In either case, the variable *optopt* is set to the character that caused the error. The **getopt**() function returns -1 when the argument list is exhausted.

EXAMPLES

```
#include <unistd.h>
int bflag, ch, fd;

bflag = 0;
while ((ch = getopt(argc, argv, "bf:")) != -1) {
        switch (ch) {
        case 'b':
                bflag = 1;
                break;
        case 'f':
                if ((fd = open(optarg, O_RDONLY, 0)) < 0) {
                        (void)fprintf(stderr,
                            "myname: %s: %s\n", optarg, strerror(errno));
                        exit(1);
                }
                break;
        case '?':
        default:
                usage();
        }
}
argc -= optind;
argv += optind;
```

DIAGNOSTICS

If the **getopt**() function encounters a character not found in the string *optstring* or detects a missing option argument it writes an error message to the stderr and returns '?'. Setting *opterr* to a zero will disable these error messages. If *optstring* has a leading ‘:’ then a missing option argument causes a ‘:’ to be returned in addition to suppressing any error messages.

Option arguments are allowed to begin with "-"; this is reasonable but reduces the amount of error checking possible.

SEE ALSO

getopt(1), getopt_long(3), getsubopt(3)

STANDARDS

The *optreset* variable was added to make it possible to call the **getopt**() function multiple times. This is an extension to the IEEE Std 1003.2 ("POSIX.2") specification.

HISTORY

The **getopt**() function appeared in 4.3BSD.

BUGS

The **getopt**() function was once specified to return EOF instead of -1. This was changed by IEEE Std 1003.2-1992 ("POSIX.2") to decouple **getopt**() from <*stdio.h*>.

A single dash "-" may be specified as a character in *optstring*, however it should *never* have an argument associated with it. This allows **getopt**() to be used with programs that expect "-" as an option flag. This practice is wrong, and should not be used in any current development. It is provided for backward compatibility *only*. Care should be taken not to use ‘-’ as the first character in *optstring* to avoid a semantic conflict with GNU **getopt**(), which assigns different meaning to an *optstring* that begins with a ‘-’. By default, a single dash causes **getopt**() to return -1.

It is also possible to handle digits as option letters. This allows **getopt**() to be used with programs that expect a number ("-3") as an option. This practice is wrong, and should not be used in any current development. It is provided for backward compatibility *only*. The following code fragment works in most cases.

```
int ch;
long length;
char *p, *ep;

while ((ch = getopt(argc, argv, "0123456789")) != -1)
        switch (ch) {
        case ’0’: case ’1’: case ’2’: case ’3’: case ’4’:
        case ’5’: case ’6’: case ’7’: case ’8’: case ’9’:
```

```
                p = argv[optind - 1];
                if (p[0] == '-' && p[1] == ch && !p[2]) {
                        length = ch - '0';
                        ep = "";
                } else if (argv[optind] && argv[optind][1] == ch) {
                        length = strtol((p = argv[optind] + 1),
                            &ep, 10);
                        optind++;
                        optreset = 1;
                } else
                        usage();
                if (*ep != '\0')
                        errx(EX_USAGE, "illegal number -- %s", p);
                break;
        }
```