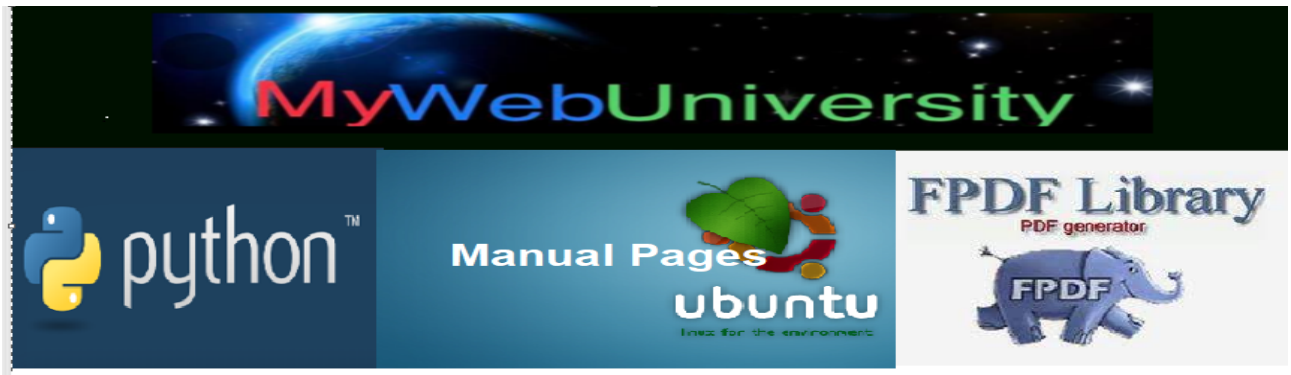

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'opterr.3'

$ man opterr.3

getopt(3) Library Functions Manual getopt(3)

NAME

getopt, optarg, optind, opterr, optopt - Parse command-line options

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <unistd.h>

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

extern char *optarg;

extern int optind, opterr, optopt;
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

getopt():

_POSIX_C_SOURCE >= 2 || _XOPEN_SOURCE

DESCRIPTION

The getopt() function parses the command-line arguments. Its arguments argc and argv are the argument count and array as passed to the main() function on program invocation. An el? ement of argv that starts with '-' (and is not exactly "-" or "--") is an option element. The characters of this element (aside from the initial '-') are option characters. If getopt() is called repeatedly, it returns successively each of the option characters from each of the option elements.

The variable optind is the index of the next element to be processed in argv. The system initializes this value to 1. The caller can reset it to 1 to restart scanning of the same

argv, or when scanning a new argument vector.

If getopt() finds another option character, it returns that character, updating the external variable optind and a static variable nextchar so that the next call to getopt() can resume the scan with the following option character or argv-element.

If there are no more option characters, getopt() returns -1. Then optind is the index in argv of the first argv-element that is not an option.

optstring is a string containing the legitimate option characters. A legitimate option character is any visible one byte ascii(7) character (for which isgraph(3) would return nonzero) that is not '-', ':', or ';'. If such a character is followed by a colon, the op? tion requires an argument, so getopt() places a pointer to the following text in the same argv-element, or the text of the following argv-element, in optarg. Two colons mean an op? tion takes an optional arg; if there is text in the current argv-element (i.e., in the same word as the option name itself, for example, "-oarg"), then it is returned in optarg, other? wise optarg is set to zero. This is a GNU extension. If optstring contains W followed by a semicolon, then -W foo is treated as the long option --foo. (The -W option is reserved by POSIX.2 for implementation extensions.) This behavior is a GNU extension, not available with libraries before glibc 2.

By default, getopt() permutes the contents of argv as it scans, so that eventually all the nonoptions are at the end. Two other scanning modes are also implemented. If the first character of optstring is '+' or the environment variable POSIXLY_CORRECT is set, then op? tion processing stops as soon as a nonoption argument is encountered. If '+' is not the first character of optstring, it is treated as a normal option. If POSIXLY_CORRECT behavior is required in this case optstring will contain two '+' symbols. If the first character of optstring is '-', then each nonoption argv-element is handled as if it were the argument of an option with character code 1. (This is used by programs that were written to expect op? tions and other argv-elements in any order and that care about the ordering of the two.)

The special argument "--" forces an end of option-scanning regardless of the scanning mode.

While processing the option list, getopt() can detect two kinds of errors: (1) an option character that was not specified in optstring and (2) a missing option argument (i.e., an option at the end of the command line without an expected argument). Such errors are han? dled and reported as follows:

? By default, getopt() prints an error message on standard error, places the erroneous op? tion character in optopt, and returns '?' as the function result.

? If the caller has set the global variable opterr to zero, then getopt() does not print an error message. The caller can determine that there was an error by testing whether the function return value is '?'. (By default, opterr has a nonzero value.)

? If the first character (following any optional '+' or '-' described above) of optstring is a colon (':'), then getopt() likewise does not print an error message. In addition, it returns ':' instead of '?' to indicate a missing option argument. This allows the caller to distinguish the two different types of errors.

RETURN VALUE

If an option was successfully found, then getopt() returns the option character. If all command-line options have been parsed, then getopt() returns -1. If getopt() encounters an option character that was not in optstring, then '?' is returned. If getopt() encounters an option with a missing argument, then the return value depends on the first character in opt? string: if it is ':', then ':' is returned; otherwise '?' is returned.

ENVIRONMENT

POSIXLY_CORRECT

If this is set, then option processing stops as soon as a nonoption argument is en? countered.

ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

Interface	Attribute	Value
getopt()	Thread safety	MT-Unsafe race:getopt env

VERSIONS

POSIX specifies that the argv array argument should be const, but these functions permute its elements unless the environment variable POSIXLY_CORRECT is set. const is used in the actual prototype to be compatible with other systems; however, this page doesn't show the qualifier, to avoid confusing readers.

STANDARDS

getopt()

POSIX.1-2008.

The use of '+' and '-' in optstring is a GNU extension.

HISTORY

getopt()

POSIX.1-2001, and POSIX.2.

On some older implementations, getopt() was declared in <stdio.h>. SUSv1 permitted the dec? laration to appear in either <unistd.h> or <stdio.h>. POSIX.1-1996 marked the use of <stdio.h> for this purpose as LEGACY. POSIX.1-2001 does not require the declaration to ap? pear in <stdio.h>.

Very old versions of glibc were affected by a _PID_GNU_nonoption_argv_flags_ environment variable.

NOTES

A program that scans multiple argument vectors, or rescans the same vector more than once, and wants to make use of GNU extensions such as '+' and '-' at the start of optstring, or changes the value of POSIXLY_CORRECT between scans, must reinitialize getopt() by resetting optind to 0, rather than the traditional value of 1. (Resetting to 0 forces the invocation of an internal initialization routine that rechecks POSIXLY_CORRECT and checks for GNU ex? tensions in optstring.)

Command-line arguments are parsed in strict order meaning that an option requiring an argu? ment will consume the next argument, regardless of whether that argument is the correctly specified option argument or simply the next option (in the scenario the user mis-specifies the command line). For example, if optstring is specified as "1n:" and the user specifies the command line arguments incorrectly as prog -n -1, the -n option will be given the optarg value "-1", and the -1 option will be considered to have not been specified.

EXAMPLES

getopt()

The following trivial example program uses getopt() to handle two program options: -n, with no associated value; and -t val, which expects an associated value.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int
main(int argc, char *argv[])
{
    int flags, opt;
```

```
        int nsecs, tfnd;
        nsecs = 0;
        tfnd = 0;
        flags = 0;
        while ((opt = getopt(argc, argv, "nt:")) != -1) {
            switch (opt) {
            case 'n':
                flags = 1;
                break;
            case 't':
                nsecs = atoi(optarg);
                tfnd = 1;
                break;
            default: /* '?' */
                fprintf(stderr, "Usage: %s [-t nsecs] [-n] name\n",
                        argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        printf("flags=%d; tfnd=%d; nsecs=%d; optind=%d\n",
               flags, tfnd, nsecs, optind);
        if (optind >= argc) {
            fprintf(stderr, "Expected argument after options\n");
            exit(EXIT_FAILURE);
        }
        printf("name argument = %s\n", argv[optind]);
        /* Other code omitted */
        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

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