NAME

getsubopt − parse suboption arguments from a string

SYNOPSIS

#include <stdlib.h>

int getsubopt(char *****optionp*, char * const ****tokens***, char *****valuep***);**

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

getsubopt():
_XOPEN_SOURCE >= 500
|| /* Since glibc 2.12: */ _POSIX_C_SOURCE >= 200809L

DESCRIPTION

getsubopt() parses the list of comma-separated suboptions provided in *optionp*. (Such a suboption list is typically produced when getopt(3) is used to parse a command line; see for example the *-o* option of mount(8).) Each suboption may include an associated value, which is separated from the suboption name by an equal sign. The following is an example of the kind of string that might be passed in *optionp*:

```
ro,name=xyz
```

The *tokens* argument is a pointer to a NULL-terminated array of pointers to the tokens that **getsubopt**() will look for in *optionp*. The tokens should be distinct, null-terminated strings containing at least one character, with no embedded equal signs or commas.

Each call to **getsubopt**() returns information about the next unprocessed suboption in *optionp*. The first equal sign in a suboption (if any) is interpreted as a separator between the name and the value of that suboption. The value extends to the next comma, or (for the last suboption) to the end of the string. If the name of the suboption matches a known name from *tokens*, and a value string was found, **getsubopt**() sets **valuep* to the address of that string. The first comma in *optionp* is overwritten with a null byte, so **valuep* is precisely the "value string" for that suboption.

If the suboption is recognized, but no value string was found, **valuep* is set to NULL.

When **getsubopt**() returns, *optionp* points to the next suboption, or to the null byte ('\0') at the end of the string if the last suboption was just processed.

RETURN VALUE

If the first suboption in *optionp* is recognized, **getsubopt**() returns the index of the matching suboption element in *tokens*. Otherwise, −1 is returned and **valuep* is the entire *name***[=***value***]** string.

Since **optionp* is changed, the first suboption before the call to **getsubopt**() is not (necessarily) the same as the first suboption after **getsubopt**().

ATTRIBUTES

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

Interface	Attribute	Value
getsubopt()	Thread safety	MT-Safe

CONFORMING TO

POSIX.1-2001, POSIX.1-2008.

NOTES

Since **getsubopt**() overwrites any commas it finds in the string **optionp*, that string must be writable; it cannot be a string constant.

EXAMPLE

The following program expects suboptions following a "−o" option.

```
#define _XOPEN_SOURCE 500
#include <stdlib.h>
#include <assert.h>
#include <stdio.h>
```

```
int
main(int argc, char **argv)
{
enum {
RO_OPT = 0,
RW_OPT,
NAME_OPT
};
char *const token[] = {
[RO_OPT]   = "ro",
[RW_OPT]   = "rw",
[NAME_OPT] = "name",
NULL
};
char *subopts;
char *value;
int opt;

int readonly = 0;
int readwrite = 0;
char *name = NULL;
int errfnd = 0;

while ((opt = getopt(argc, argv, "o:")) != -1) {
switch (opt) {
case 'o':
subopts = optarg;
while (*subopts != '\0' && !errfnd) {

switch (getsubopt(&subopts, token, &value)) {
case RO_OPT:
readonly = 1;
break;

case RW_OPT:
readwrite = 1;
break;

case NAME_OPT:
if (value == NULL) {
fprintf(stderr, "Missing value for "
"suboption '%s'\n", token[NAME_OPT]);
errfnd = 1;
continue;
}

name = value;
break;

default:
fprintf(stderr, "No match found "
"for token: /%s/\n", value);
errfnd = 1;
break;
}
}
if (readwrite && readonly) {
fprintf(stderr, "Only one of '%s' and '%s' can be "
```

```
"specified\n", token[RO_OPT], token[RW_OPT]);
errfnd = 1;
}
break;

default:
errfnd = 1;
}
}

if (errfnd || argc == 1) {
fprintf(stderr, "\nUsage: %s -o <suboptstring>\n", argv[0]);
fprintf(stderr, "suboptions are 'ro', 'rw', "
"and 'name=<value>'\n");
exit(EXIT_FAILURE);
}

/* Remainder of program... */

exit(EXIT_SUCCESS);
}
```

SEE ALSO

getopt(3)

COLOPHON

This page is part of release 4.16 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man−pages/.