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

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'getsubopt.3p' command

$ man getsubopt.3p

GETSUBOPT(3P) POSIX Programmer's Manual GETSUBOPT(3P)

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.

NAME

getsubopt ? parse suboption arguments from a string

SYNOPSIS

#include <stdlib.h>

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

DESCRIPTION

The getsubopt() function shall parse suboption arguments in a flag ar? gument. Such options often result from the use of getopt().

The getsubopt() argument optionp is a pointer to a pointer to the op? tion argument string. The suboption arguments shall be separated by <comma> characters and each may consist of either a single token, or a token-value pair separated by an <equals-sign>.

The keylistp argument shall be a pointer to a vector of strings. The end of the vector is identified by a null pointer. Each entry in the vector is one of the possible tokens that might be found in *optionp.

Since <comma> characters delimit suboption arguments in optionp, they should not appear in any of the strings pointed to by keylistp. Simi?

larly, because an <equals-sign> separates a token from its value, the application should not include an <equals-sign> in any of the strings pointed to by keylistp. The getsubopt() function shall not modify the keylistp vector.

The valuep argument is the address of a value string pointer.

If a <comma> appears in optionp, it shall be interpreted as a suboption separator. After <comma> characters have been processed, if there are one or more <equals-sign> characters in a suboption string, the first <equals-sign> in any suboption string shall be interpreted as a separa? tor between a token and a value. Subsequent <equals-sign> characters in a suboption string shall be interpreted as part of the value.

If the string at *optionp contains only one suboption argument (equiva? lently, no <comma> characters), getsubopt() shall update *optionp to point to the null character at the end of the string. Otherwise, it shall isolate the suboption argument by replacing the <comma> separator with a null character, and shall update *optionp to point to the start of the next suboption argument. If the suboption argument has an asso? ciated value (equivalently, contains an <equals-sign>), getsubopt() shall update *valuep to point to the value's first character. Other? wise, it shall set *valuep to a null pointer. The calling application may use this information to determine whether the presence or absence of a value for the suboption is an error.

Additionally, when getsubopt() fails to match the suboption argument with a token in the keylistp array, the calling application should de? cide if this is an error, or if the unrecognized option should be pro? cessed in another way.

RETURN VALUE

The getsubopt() function shall return the index of the matched token string, or -1 if no token strings were matched.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

Parsing Suboptions

The following example uses the getsubopt() function to parse a value argument in the optarg external variable returned by a call to getopt().

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int do_all;
const char *type;
int read_size;
int write_size;
int read_only;
enum
{
    RO_OPTION = 0,
    RW_OPTION,
    READ_SIZE_OPTION,
    WRITE_SIZE_OPTION
};
const char *mount_opts[] =
{
    [RO_OPTION] = "ro",
    [RW_OPTION] = "rw",
    [READ_SIZE_OPTION] = "rsize",
    [WRITE_SIZE_OPTION] = "wsize",
    NULL
};
int
main(int argc, char *argv[])
{
    char *subopts, *value;
    int opt;
    while ((opt = getopt(argc, argv, "at:o:")) != -1)
```

```
switch(opt)
    {
    case 'a':
        do_all = 1;
        break;
    case 't':
        type = optarg;
        break;
    case 'o':
        subopts = optarg;
        while (*subopts != ' ')
        {
            char *saved = subopts;
            switch(getsubopt(&subopts, (char **)mount_opts,
                &value))
            {
            case RO_OPTION:
                read_only = 1;
                break;
            case RW_OPTION:
                read_only = 0;
                break;
            case READ_SIZE_OPTION:
                if (value == NULL)
                    abort();
                read_size = atoi(value);
                break;
            case WRITE_SIZE_OPTION:
                if (value == NULL)
                    abort();
                write_size = atoi(value);
                break;
            default:
```

```
                    /* Unknown suboption. */
                    printf("Unknown suboption `%s'\n", saved);
                    abort();
                  }
                }
                break;
              default:
                abort();
              }
            /* Do the real work. */
            return 0;
          }
```

If the above example is invoked with:

```
program -o ro,rsize=512
```

then after option parsing, the variable do_all will be 0, type will be a null pointer, read_size will be 512, write_size will be 0, and read_only will be 1. If it is invoked with:

```
program -o oops
```

it will print:

```
"Unknown suboption `oops'"
```

before aborting.

APPLICATION USAGE

The value of *valuep when getsubopt() returns -1 is unspecified. Historical implementations provide various incompatible extensions to allow an application to access the suboption text that was not found in the keylistp array.

RATIONALE

The keylistp argument of getsubopt() is typed as char * const * to match historical practice. However, the standard is clear that implementations will not modify either the array or the strings contained in the array, as if the argument had been typed const char * const *.

FUTURE DIRECTIONS

None.

SEE ALSO

getopt()

The Base Definitions volume of POSIX.1?2017, <stdlib.h>

COPYRIGHT