NAME

getmntopts, **getmntpoint**, **chkdoreload**, **build_iovec**, **build_iovec_argf**, **free_iovec**, **checkpath**, **rmslashes** - mount point operations

SYNOPSIS

#include <mntopts.h>

void
getmntopts(*const char *options, const struct mntopt *mopts, int *flagp, int *altflagp*);

*struct statfs **
getmntpoint(*const char *name*);

int
chkdoreload(*struct statfs *mntp, void (*prtmsg)(const char *fmt, ...)*);

void
build_iovec(*struct iovec **iov, int *iovlen, const char *name, void *val, size_t len*);

void
build_iovec_argf(*struct iovec **iov, int *iovlen, const char *name, const char *fmt, ...*);

void
free_iovec(*struct iovec **iov, int *iovlen*);

int
checkpath(*const char *path, char *resolved*);

void
rmslashes(*char *rrpin, char *rrpout*);

DESCRIPTION

The **mntopts** functions support operations associated with a mount point. For historic reasons are in a file in the sources for the mount(8) program. Thus, to access them the following lines need to be added to the **Makefile** of the program wanting to use them:

```
SRCS+= getmntopts.c
MOUNT= ${SRCTOP}/sbin/mount
CFLAGS+= -I${MOUNT}
.PATH: ${MOUNT}
```

The **getmntopts**() function takes a comma separated option list and a list of valid option names, and computes the bitmask corresponding to the requested set of options.

The string *options* is broken down into a sequence of comma separated tokens. Each token is looked up in the table described by *mopts* and the bits in the word referenced by either *flagp* or *altflagp* (depending on the *m_altloc* field of the option's table entry) are updated. The flag words are not initialized by **getmntopts**(). The table, *mopts*, has the following format:

```
struct mntopt {
        char *m_option;     /* option name */
        int m_inverse;      /* is this a negative option, e.g., "dev" */
        int m_flag;         /* bit to set, e.g., MNT_RDONLY */
        int m_altloc;       /* non-zero to use altflagp rather than flagp */
};
```

The members of this structure are:

m_option the option name, for example "suid".

m_inverse tells **getmntopts**() that the name has the inverse meaning of the bit. For example, "suid" is the string, whereas the mount flag is MNT_NOSUID. In this case, the sense of the string and the flag are inverted, so the *m_inverse* flag should be set.

m_flag the value of the bit to be set or cleared in the flag word when the option is recognized. The bit is set when the option is discovered, but cleared if the option name was preceded by the letters "no". The *m_inverse* flag causes these two operations to be reversed.

m_altloc the bit should be set or cleared in *altflagp* rather than *flagp*.

Each of the user visible MNT_ flags has a corresponding MOPT_ macro which defines an appropriate *struct mntopt* entry. To simplify the program interface and ensure consistency across all programs, a general purpose macro, MOPT_STDOPTS, is defined which contains an entry for all the generic VFS options. In addition, the macros MOPT_FORCE and MOPT_UPDATE exist to enable the MNT_FORCE and MNT_UPDATE flags to be set. Finally, the table must be terminated by an entry with a NULL first element.

The **getmntpoint**() function takes the pathname of a possible mount point or of a device (with or without */dev/* prepended to it). If the pathname is a directory or a file, **getmntpoint**() checks to see if the mount point currently has a filesystem mounted on it. If the pathname is a device, **getmntpoint**() checks to see if it is currently mounted. If there is an associated mount, a pointer to a *struct statfs* is returned. The

returned result is stored in a static buffer that is over-written each time the **getmntpoint**() function or the getmntinfo(3) library routine is called. If no mount is found, NULL is returned.

The **chkdoreload**() function takes a pointer to a *struct statfs*. If the filesystem associated with the mount point is mounted read-only, **chkdoreload**() requests the filesystem to reload all of its metadata from its backing store. The second parameter is the function to call to print an error message if the reload fails. If no error message is desired, a NULL can be passed as the second argument. The **chkdoreload**() function returns zero on success or non-zero on failure.

The **build_iovec**() function adds a parameter to a list of parameters to be passed to the nmount(2) system call. The parameter list is built up in *iov* and its length is kept in *iovlen*. Before the first call to **build_iovec**(), *iov* should be set to NULL and *iovlen* should be set to 0. The parameter name is passed in *name*. The value of the parameter name is pointed to by *val*. The size of the value is passed in *len*. If the value is a string, a *len* of -1 is passed to indicate that the length should be determined using strlen(3). If the parameter has no value, *name* should be NULL and *len* should be 0.

The **build_iovec_argf**() function adds a formatted parameter to a list of parameters to be passed to the nmount(2) system call. The parameter list is built up in *iov* and its length is kept in *iovlen*. Before the first call to **build_iovec_argf**(), *iov* should be set to NULL and *iovlen* should be set to 0. The parameter name is passed in *name*. The value of the parameter name is described by a format string pointed to by *fmt*. If the parameter has no value, *name* should be NULL.

The **free_iovec**() function frees the memory in the *iov* vector of the length specified in *iovlen* that was previously allocated by the **build_iovec**() and / or **build_iovec_argf**() functions. The *iov* is set to NULL and the *iovlen* is set to 0 to indicate that the space has been freed.

The **checkpath**() function uses realpath(3) to verify that its *path* argument is valid and references a directory. The **checkpath**() function returns zero on success or non-zero on failure.

The **rmslashes**() function removes all double slashes and trailing slashes from its *rrpin* pathname parameter and returns the resulting pathname in its *rrpout* parameter.

EXAMPLES

Most commands will use the standard option set. Local file systems which support the MNT_UPDATE flag, would also have an MOPT_UPDATE entry. This can be declared and used as follows:

```
#include "mntopts.h"

struct mntopt mopts[] = {
        MOPT_STDOPTS,
```

```
        MOPT_UPDATE,
        { NULL }
};

        ...
        mntflags = mntaltflags = 0;
        ...
        getmntopts(options, mopts, &mntflags, &mntaltflags);
        ...
```

DIAGNOSTICS

If the external integer variable *getmnt_silent* is zero, then the **getmntopts**() function displays an error message and exits if an unrecognized option is encountered. Otherwise unrecognized options are silently ignored. By default *getmnt_silent* is zero.

SEE ALSO

err(3), mount(8), nmount(8)

HISTORY

The **getmntopts**() function appeared in 4.4BSD. The **build_iovec**(), **build_iovec_argf**(), **free_iovec**(), **checkpath**(), and **rmslashes**() functions were added with nmount(8) in FreeBSD 5.0. The **getmntpoint**() and **chkdoreload**() functions were added in FreeBSD 14.0.