NAME

mount – mount a file system

SYNOPSIS

#include <sys/types.h>
#include <sys/mount.h>

int mount(const char **spec*, **const char** **dir*, **int** *mflag*, **/* char** **fstype*,
const char **dataptr*, **int** *datalen* **/* **. . .);**

DESCRIPTION

mount( ) requests that a removable file system contained on the block special file identified by *spec* be mounted on the directory identified by *dir*. *spec* and *dir* are pointers to path names. *fstype* is the file system type, which can be determined by the **sysfs**(2) function. If both the **MS_DATA** and **MS_FSS** flag bits of *mflag* are off, the file system type defaults to the root file system type. Only if either flag is on is *fstype* used to indicate the file system type.

If the **MS_DATA** flag is set in *mflag* the system expects the *dataptr* and *datalen* arguments to be present. Together they describe a block of file-system specific data at address *dataptr* of length *datalen*. This is interpreted by file-system specific code within the operating system and its format depends on the file system type. If a particular file system type does not require this data, *dataptr* and *datalen* should both be zero. Note that **MS_FSS** is obsolete and is ignored if **MS_DATA** is also set, but if **MS_FSS** is set and **MS_DATA** is not, *dataptr* and *datalen* are both assumed to be zero.

After a successful call to **mount( )**, all references to the file *dir* refer to the root directory on the mounted file system.

The low-order bit of *mflag* is used to control write permission on the mounted file system: if **1**, writing is forbidden; otherwise writing is permitted according to individual file accessibility.

The **mount( )** system call may only be invoked only by processes with super-user privileges.

RETURN VALUES

Upon successful completion a value of **0 is returned.** Otherwise, a value of –**1** is returned and **errno** is set to indicate the error.

ERRORS

mount( ) fails if one or more of the following are true:

EBUSY	*dir* is currently mounted on, is someone's current working directory, or is otherwise busy.
EBUSY	The device associated with *spec* is currently mounted.
EBUSY	There are no more mount table entries.
EFAULT	*spec*, *dir*, or *datalen* points outside the allocated address space of the process.
EINVAL	The super block has an invalid magic number or the *fstype* is invalid.

ELOOP	Too many symbolic links were encountered in translating *spec* or *dir*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines and the file system type does not allow it.
ENAMETOOLONG	The length of the *path* argument exceeds {**PATH_MAX**}, or the length of a *path* component exceeds {**NAME_MAX**} while {**_POSIX_NO_TRUNC**} is in effect.
ENOENT	None of the named files exists or is a null pathname.
ENOTBLK	*spec* is not a block special device.
ENOTDIR	*dir* is not a directory.
ENOTDIR	A component of a path prefix is not a directory.
EPERM	The effective user ID is not super-user.
EREMOTE	*spec* is remote and cannot be mounted.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENXIO	The device associated with *spec* does not exist.
EROFS	*spec* is write protected and *mflag* requests write permission.
ENOSPC	The file system state in the super-block is not **FsOKAY** and *mflag* requests write permission.

SEE ALSO

mount(1M), **sysfs**(2), **umount**(2)