NAME

mount_fusefs - mount a Fuse file system daemon

SYNOPSIS

mount_fusefs [**-A**] [**-S**] [**-v**] [**-D** *fuse_daemon*] [**-O** *daemon_opts*] [**-s** *special*] [**-m** *node*] [**-h**] [**-V**]
[**-o** *option ...*] *special node* [*fuse_daemon ...*]

DESCRIPTION

Basic usage is to start a fuse daemon on the given *special* file. In practice, the daemon is assigned a *special* file automatically, which can then be identified via fstat(1). That special file can then be mounted by **mount_fusefs**.

However, the procedure of spawning a daemon will usually be automated so that it is performed by **mount_fusefs**. If the command invoking a given *fuse_daemon* is appended to the list of arguments, **mount_fusefs** will call the *fuse_daemon* via that command. In that way the *fuse_daemon* will be instructed to attach itself to *special*. From that on mounting goes as in the simple case. (See *DAEMON MOUNTS*.)

The *special* argument will normally be treated as the path of the special file to mount.

However, if *auto* is passed as *special*, then **mount_fusefs** will look for a suitable free fuse device by itself.

Finally, if *special* is an integer it will be interpreted as the number of the file descriptor of an already open fuse device (used when the Fuse library invokes **mount_fusefs**. See *DAEMON MOUNTS*).

The options are as follows:

-A, **--reject-allow_other**
: Prohibit the **allow_other** mount flag. Intended for use in scripts and the sudoers(5) (*ports/security/sudo*) file.

-S, **--safe**
: Run in safe mode (i.e., reject invoking a filesystem daemon).

-v Be verbose.

-D, **--daemon** *daemon*
: Call the specified *daemon*.

-O, **--daemon_opts** *opts*
Add *opts* to the daemon’s command line.

-s, **--special** *special*
Use *special* as special.

-m, **--mountpath** *node*
Mount on *node*.

-h, **--help**
Show help.

-V, **--version**
Show version information.

-o Mount options are specified via **-o**. The following options are available (and also their negated versions, by prefixing them with "no"):

allow_other
Do not apply *STRICT ACCESS POLICY*. Only root can use this option.

async I/O to the file system may be done asynchronously. Writes may be delayed and/or reordered.

default_permissions
Enable traditional (file mode based) permission checking in kernel.

intr Allow signals to interrupt operations that are blocked waiting for a reply from the server. When this option is in use, system calls may fail with EINTR whenever a signal is received.

max_read=*n*
Limit size of read requests to *n*.

neglect_shares
Do not refuse unmounting if there are secondary mounts.

private
Refuse shared mounting of the daemon. This is the default behaviour, to allow sharing, explicitly use **-o noprivate**.

push_symlinks_in
Prefix absolute symlinks with the mountpoint.

subtype=*fsname*
Suffix *fsname* to the file system name as reported by statfs(2). This option can be used to identify the file system implemented by *fuse_daemon*.

Besides the above mount options, there is a set of pseudo-mount options which are supported by the Fuse library. One can list these by passing **-h** to a Fuse daemon. Most of these options only have effect on the behavior of the daemon (that is, their scope is limited to userspace). However, there are some which do require in-kernel support. Currently the options supported by the kernel are:

direct_io
Bypass the buffer cache system.

kernel_cache
By default cached buffers of a given file are flushed at each open(2). This option disables this behaviour.

DAEMON MOUNTS

Usually users do not need to use **mount_fusefs** directly, as the Fuse library enables Fuse daemons to invoke **mount_fusefs**. That is,

```
fuse_daemon device mountpoint
```

has the same effect as

```
mount_fusefs auto mountpoint fuse_daemon
```

This is the recommended usage when you want basic usage (eg, run the daemon at a low privilege level but mount it as root).

STRICT ACCESS POLICY

The strict access policy for Fuse filesystems lets one use the filesystem only if the filesystem daemon has the same credentials (uid, real uid, gid, real gid) as the user.

This is applied for Fuse mounts by default and only root can mount without the strict access policy (i.e., the **allow_other** mount option).

This is to shield users from the daemon "spying" on their I/O activities.

Users might opt to willingly relax strict access policy (as far as they are concerned) by doing their own secondary mount (See *SHARED MOUNTS*).

SHARED MOUNTS

A Fuse daemon can be shared (i.e., mounted multiple times). When doing the first (primary) mount, the spawner and the mounter of the daemon must have the same uid, or the mounter should be the superuser.

After the primary mount is in place, secondary mounts can be done by anyone unless this feature is disabled by **private**. The behaviour of a secondary mount is analogous to that of symbolic links: they redirect all filesystem operations to the primary mount.

Doing a secondary mount is like signing an agreement: by this action, the mounter agrees that the Fuse daemon can trace her I/O activities. From then on she is not banned from using the filesystem (either via her own mount or via the primary mount), regardless whether **allow_other** is used or not.

The device name of a secondary mount is the device name of the corresponding primary mount, followed by a ’#’ character and the index of the secondary mount; e.g., */dev/fuse0#3*.

SECURITY

System administrators might want to use a custom mount policy (ie., one going beyond the *vfs.usermount* sysctl). The primary tool for such purposes is sudo(8) (*ports/security/sudo*). However, given that **mount_fusefs** is capable of invoking an arbitrary program, one must be careful when doing this. **mount_fusefs** is designed in a way such that it makes that easy. For this purpose, there are options which disable certain risky features (**-S** and **-A**), and command line parsing is done in a flexible way: mixing options and non-options is allowed, but processing them stops at the third non-option argument (after the first two have been utilized as device and mountpoint). The rest of the command line specifies the daemon and its arguments. (Alternatively, the daemon, the special and the mount path can be specified using the respective options.) Note that **mount_fusefs** ignores the environment variable POSIXLY_CORRECT and always behaves as described.

In general, to be as scripting / sudoers(5) (*ports/security/sudo*) friendly as possible, no information has a fixed position in the command line, but once a given piece of information is provided, subsequent arguments/options cannot override it (with the exception of some non-critical ones).

ENVIRONMENT

MOUNT_FUSEFS_SAFE	This has the same effect as the **-S** option.
MOUNT_FUSEFS_VERBOSE	This has the same effect as the **-v** option.

MOUNT_FUSEFS_IGNORE_UNKNOWN
: If set, **mount_fusefs** will ignore unknown mount options.

MOUNT_FUSEFS_CALL_BY_LIB
: Adjust behavior to the needs of the FUSE library. Currently it effects help output.

Although the following variables do not have any effect on **mount_fusefs** itself, they affect the behaviour of fuse daemons:

FUSE_DEV_NAME	Device to attach. If not set, the multiplexer path */dev/fuse* is used.
FUSE_DEV_FD	File descriptor of an opened Fuse device to use. Overrides FUSE_DEV_NAME.
FUSE_NO_MOUNT	If set, the library will not attempt to mount the filesystem, even if a mountpoint argument is supplied.

FILES

/dev/fuse	Fuse device with which the kernel and Fuse daemons can communicate.
/dev/fuse	The multiplexer path. An open(2) performed on it automatically is passed to a free Fuse device by the kernel (which might be created just for this puprose).

EXAMPLES

Mount the example filesystem in the Fuse distribution (from its directory): either

```
./fusexmp /mnt/fuse
```

or

```
mount_fusefs auto /mnt/fuse ./fusexmp
```

Doing the same in two steps, using */dev/fuse0*:

```
FUSE_DEV_NAME=/dev/fuse ./fusexmp &&
mount_fusefs /dev/fuse /mnt/fuse
```

A script wrapper for fusexmp which ensures that **mount_fusefs** does not call any external utility and also provides a hacky (non race-free) automatic device selection:

```
#!/bin/sh -e

FUSE_DEV_NAME=/dev/fuse fusexmp
mount_fusefs -S /dev/fuse /mnt/fuse "$@"
```

SEE ALSO

fstat(1), mount(8), sudo(8) (*ports/security/sudo*), umount(8)

HISTORY

mount_fusefs was written as the part of the FreeBSD implementation of the Fuse userspace filesystem framework (see **https://github.com/libfuse/libfuse**) and first appeared in the *sysutils/fusefs-kmod* port, supporting FreeBSD 6.0. It was added to the base system in FreeBSD 10.0.

CAVEATS

This user interface is FreeBSD specific. Secondary mounts should be unmounted via their device name. If an attempt is made to unmount them via their filesystem root path, the unmount request will be forwarded to the primary mount path. In general, unmounting by device name is less error-prone than by mount path (although the latter will also work under normal circumstances).

If the daemon is specified via the **-D** and **-O** options, it will be invoked via system(3), and the daemon's command line will also have an "&" control operator appended, so that we do not have to wait for its termination. You should use a simple command line when invoking the daemon via these options.

BUGS

special is treated as a multiplexer if and only if it is literally the same as *auto* or */dev/fuse*. Other paths which are equivalent with */dev/fuse* (eg., */../dev/fuse*) are not.