Red Hat Enterprise Linux Release 9.2 Manual Pages on 'mount.fuse3.8' command

$ man mount.fuse3.8

fuse(8)   System Manager's Manual   fuse(8)

NAME

   fuse - configuration and mount options for FUSE file systems

DESCRIPTION

   FUSE (Filesystem in Userspace) is a simple interface for userspace pro?

   grams to export a virtual filesystem to the Linux kernel. FUSE also

   aims to provide a secure method for non privileged users to create and

   mount their own filesystem implementations.

DEFINITIONS

   FUSE   The in-kernel filesystem that forwards requests to a user-space

        process.

   filesystem

        The user-space process that responds to requests received from

        the kernel.

   libfuse

        The shared library that most (user-space) filesystems use to

        communicate with FUSE (the kernel filesystem). libfuse also pro?

        vides the fusermount3 (or fusermount if you have older version

        of libfuse) helper to allow non-privileged users to mount

        filesystems.

   filesystem owner

        The user that starts the filesystem and instructs the kernel to

        associate it with a particular mountpoint. The latter is typi?

cally done by the filesystem itself on start-up. When using lib?

fuse, this is done by calling the fusermount3 utility.

client Any process that interacts with the mountpoint.

CONFIGURATION

Some options regarding mount policy can be set in the file

/etc/fuse.conf. Currently these options are:

mount_max = NNN

Set the maximum number of FUSE mounts allowed to non-root users.

The default is 1000.

user_allow_other

Allow non-root users to specify the allow_other or allow_root

mount options (see below).

These limits are enforced by the fusermount3 helper, so they can be

avoided by filesystems that run as root.

OPTIONS

Most of the generic mount options described in mount are supported (ro,

rw, suid, nosuid, dev, nodev, exec, noexec, atime, noatime, sync,

async, dirsync). Filesystems are mounted with nodev,nosuid by default,

which can only be overridden by a privileged user.

General mount options:

These are FUSE specific mount options that can be specified for all

filesystems:

default_permissions

This option instructs the kernel to perform its own permission

check instead of deferring all permission checking to the

filesystem. The check by the kernel is done in addition to any

permission checks by the filesystem, and both have to succeed

for an operation to be allowed. The kernel performs a standard

UNIX permission check (based on mode bits and ownership of the

directory entry, and uid/gid of the client).

This mount option is activated implicitly if the filesystem en?

ables ACL support during the initial feature negotiation when

opening the device fd. In this case, the kernel performs both

ACL and standard unix permission checking.

Filesystems that do not implement any permission checking should

generally add this option internally.

allow_other

This option overrides the security measure restricting file ac?

cess to the filesystem owner, so that all users (including root)

can access the files.

rootmode=M

Specifies the the file mode of the filesystem's root (in octal

representation).

blkdev Mount a filesystem backed by a block device. This is a privi?

leged option. The device must be specified with the fsname=NAME

option.

blksize=N

Set the block size for the filesystem. This option is only valid

for 'fuseblk' type mounts. The default is 512.

In most cases, this option should not be specified by the

filesystem owner but set internally by the filesystem.

max_read=N

With this option the maximum size of read operations can be set.

The default is infinite, but typically the kernel enforces its

own limit in addition to this one. A value of zero corresponds

to no limit.

This option should not be specified by the filesystem owner. The

correct (or optimum) value depends on the filesystem implementa?

tion and should thus be set by the filesystem internally.

This mount option is deprecated in favor of direct negotiation

over the device fd (as done for e.g. the maximum size of write

operations). For the time being, libfuse-using filesystems that

want to limit the read size must therefore use this mount option

and set the same value again in the init() handler.

fd=N The file descriptor to use for communication between the

userspace filesystem and the kernel. The file descriptor must

have been obtained by opening the FUSE device (/dev/fuse).
This option should not be specified by the filesystem owner. It
is set by libfuse (or, if libfuse is not used, must be set by
the filesystem itself).

user_id=N

group_id=N Specifies the numeric uid/gid of the mount owner.
This option should not be specified by the filesystem owner. It
is set by libfuse (or, if libfuse is not used, must be set by
the filesystem itself).

fsname=NAME

Sets the filesystem source (first field in /etc/mtab). The de?
fault is the name of the filesystem process.

subtype=TYPE

Sets the filesystem type (third field in /etc/mtab). The default
is the name of the filesystem process. If the kernel suppports
it, /etc/mtab and /proc/mounts will show the filesystem type as
fuse.TYPE
If the kernel doesn't support subtypes, the source field will be
TYPE#NAME, or if fsname option is not specified, just TYPE.

libfuse-specific mount options:

These following options are not actually passed to the kernel but in?
terpreted by libfuse. They can be specified for all filesystems that
use libfuse:

allow_root

This option is similar to allow_other but file access is limited
to the filesystem owner and root. This option and allow_other
are mutually exclusive.

auto_unmount

This option enables automatic release of the mountpoint if
filesystem terminates for any reason. Normally the filesystem is
responsible for releasing the mountpoint, which means that the
mountpoint becomes inaccessible if the filesystem terminates
without first unmounting.

At the moment, this option implies that the filesystem will also

be mounted with nodev and nosuid (even when mounted by root).

This restriction may be lifted in the future.

High-level mount options:

These following options are not actually passed to the kernel but in‐

terpreted by libfuse. They can only be specified for filesystems that

use the high-level libfuse API:

kernel_cache

This option disables flushing the cache of the file contents on

every open(2). This should only be enabled on filesystems,

where the file data is never changed externally (not through the

mounted FUSE filesystem). Thus it is not suitable for network

filesystems and other "intermediate" filesystems.

NOTE: if this option is not specified (and neither direct_io)

data is still cached after the open(2), so a read(2) system call

will not always initiate a read operation.

auto_cache

This option is an alternative to kernel_cache. Instead of uncon‐

ditionally keeping cached data, the cached data is invalidated

on open(2) if the modification time or the size of the file has

changed since it was last opened.

umask=M

Override the permission bits in st_mode set by the filesystem.

The resulting permission bits are the ones missing from the

given umask value. The value is given in octal representation.

uid=N Override the st_uid field set by the filesystem (N is numeric).

gid=N Override the st_gid field set by the filesystem (N is numeric).

entry_timeout=T

The timeout in seconds for which name lookups will be cached.

The default is 1.0 second. For all the timeout options, it is

possible to give fractions of a second as well (e.g. entry_time‐

out=2.8)

negative_timeout=T

The timeout in seconds for which a negative lookup will be

cached. This means, that if file did not exist (lookup returned

ENOENT), the lookup will only be redone after the timeout, and

the file/directory will be assumed to not exist until then. The

default is 0.0 second, meaning that caching negative lookups are

disabled.

attr_timeout=T

The timeout in seconds for which file/directory attributes are

cached. The default is 1.0 second.

ac_attr_timeout=T

The timeout in seconds for which file attributes are cached for

the purpose of checking if auto_cache should flush the file data

on open. The default is the value of attr_timeout

noforget

remember=T

Normally, libfuse assigns inodes to paths only for as long as

the kernel is aware of them. With this option inodes are instead

assigned for at least T seconds (or, in the case of noforget,

the life-time of the filesystem). This will require more memory,

but may be necessary when using applications that make use of

inode numbers.

modules=M1[:M2...]

Add modules to the filesystem stack. Modules are pushed in the

order they are specified, with the original filesystem being on

the bottom of the stack.

mount.fuse3 options:

These options are interpreted by mount.fuse3 and are thus only avail‐

able when mounting a file system via mount.fuse3 (such as when mounting

via the generic mount(1) command or /etc/fstab). Supported options are:

setuid=USER

Switch to USER and its primary group before launching the FUSE

file system process. mount.fuse3 must be run as root or with

CAP_SETUID and CAP_SETGID for this to work.

drop_privileges

Perform setup of the FUSE file descriptor and mounting the file

system before launching the   FUSE   file   system   process.

mount.fuse3 requires privilege to do so, i.e. must be run as

root or at least with CAP_SYS_ADMIN and CAP_SETPCAP. It will

launch the file system process fully unprivileged, i.e. without

capabilities(7) and prctl(2) flags set up such that privileges

can't be reacquired (e.g. via setuid or fscaps binaries). This

reduces risk in the event of the FUSE file system process get?

ting compromised by malicious file system data.

FUSE MODULES (STACKING)

Modules are filesystem stacking support to high level API. Filesystem

modules can be built into libfuse or loaded from shared object

iconv

Perform file name character set conversion. Options are:

from_code=CHARSET

Character set to convert from (see iconv -l for a list of possi?

ble values). Default is UTF-8.

to_code=CHARSET

Character set to convert to. Default is determined by the cur?

rent locale.

subdir

Prepend a given directory to each path. Options are:

subdir=DIR

Directory to prepend to all paths. This option is mandatory.

rellinks

Transform absolute symlinks into relative

norellinks

Do not transform absolute symlinks into relative. This is the

default.

SECURITY

The fusermount3 program is installed set-user-gid to fuse. This is done

to allow users from fuse group to mount their own filesystem implemen?

tations.   There must however be some limitations, in order to prevent

Bad User from doing nasty things. Currently those limitations are:

1.   The user can only mount on a mountpoint, for which it has write
    permission

2.   The mountpoint is not a sticky directory which isn't owned by
    the user (like /tmp usually is)

3.   No other user (including root) can access the contents of the
    mounted filesystem.

NOTE

FUSE filesystems are unmounted using the fusermount3(1) command (fuser?

mount3 -u mountpoint).

AUTHORS

FUSE is currently maintained by Nikolaus Rath <Nikolaus@rath.org>

The original author of FUSE is Miklos Szeredi <mszeredi@inf.bme.hu>.

This man page was originally written by Bastien Roucaries <rou?

caries.bastien+debian@gmail.com> for the Debian GNU/Linux distribution.

SEE ALSO

fusermount3(1) fusermount(1) mount(8)

<div align="center">fuse(8)</div>