NAME

guestmount − Mount a guest filesystem on the host using FUSE and libguestfs

SYNOPSIS

```
 guestmount [−−options] −a disk.img −m device [−−ro] mountpoint

 guestmount [−−options] −a disk.img −i [−−ro] mountpoint

 guestmount [−−options] −d Guest −i [−−ro] mountpoint
```

WARNING

Using `guestmount` in write mode on live virtual machines, or concurrently with other disk editing tools, can be dangerous, potentially causing disk corruption. The virtual machine must be shut down before you use this command, and disk images must not be edited concurrently.

Use the *−−ro* (read-only) option to use `guestmount` safely if the disk image or virtual machine might be live. You may see strange or inconsistent results if running concurrently with other changes, but with this option you won't risk disk corruption.

DESCRIPTION

The guestmount program can be used to mount virtual machine filesystems and other disk images on the host. It uses libguestfs for access to the guest filesystem, and FUSE (the "filesystem in userspace") to make it appear as a mountable device.

Along with other options, you have to give at least one device (*−a* option) or libvirt domain (*−d* option), and at least one mountpoint (*−m* option) or use the *−i* inspection option or the *−−live* option. How this works is better explained in the **guestfish** (1) manual page, or by looking at the examples below.

FUSE lets you mount filesystems as non-root. The mountpoint must be owned by you. The filesystem will not be visible to any other users unless you make configuration changes, see "NOTES" below.

To unmount the filesystem, use the **guestunmount** (1) command.

EXAMPLES

For a typical Windows guest which has its main filesystem on the first partition:

```
 guestmount −a windows.img −m /dev/sda1 −−ro /mnt
```

For a typical Linux guest which has a /boot filesystem on the first partition, and the root filesystem on a logical volume:

```
 guestmount −a linux.img −m /dev/VG/LV −m /dev/sda1:/boot −−ro /mnt
```

To get libguestfs to detect guest mountpoints for you:

```
 guestmount −a guest.img −i −−ro /mnt
```

For a libvirt guest called "Guest" you could do:

```
 guestmount −d Guest −i −−ro /mnt
```

If you don't know what filesystems are contained in a guest or disk image, use **virt−filesystems** (1) first:

```
 virt−filesystems −d MyGuest
```

If you want to trace the libguestfs calls but without excessive debugging information, we recommend:

```
 guestmount [...] −−trace /mnt
```

If you want to debug the program, we recommend:

```
 guestmount [...] −−trace −−verbose /mnt
```

To unmount the filesystem after using it:

```
 guestunmount /mnt
```

NOTES

Other users cannot see the filesystem by default

If you mount a filesystem as one user (eg. root), then other users will not be able to see it by default. The fix is to add the FUSE `allow_other` option when mounting:

```
sudo guestmount [...] -o allow_other /mnt
```

and to enable this option in */etc/fuse.conf*.

Enabling FUSE

On some distros, you may need to add yourself to a special group (eg. `fuse`) before you can use any FUSE filesystem. This is necessary on Debian and derivatives.

On other distros, no special group is required. It is not necessary on Fedora or Red Hat Enterprise Linux.

fusermount error: "Device or resource busy"

You can see this error when another process on the system jumps into the mountpoint you have just created, holding it open and preventing you from unmounting it. The usual culprits are various GUI "indexing" programs.

The popular workaround for this problem is to retry the `fusermount -u` command a few times until it works (**guestunmount**(1) does this for you). Unfortunately this isn't a reliable fix if (for example) the mounted filesystem is particularly large and the intruding program particularly persistent.

A proper fix is to use a private mountpoint by creating a new mount namespace using the Linux-specific **clone**(2)/**unshare**(2) flag `CLONE_NEWNS`. Unfortunately at the moment this requires root and we would also probably need to add it as a feature to guestmount.

Race conditions possible when shutting down the connection

When **guestunmount**(1)/**fusermount**(1) exits, guestmount may still be running and cleaning up the mountpoint. The disk image will not be fully finalized.

This means that scripts like the following have a nasty race condition:

```
guestmount -a disk.img -i /mnt
# copy things into /mnt
guestunmount /mnt
# immediately try to use 'disk.img' ** UNSAFE **
```

The solution is to use the *--pid-file* option to write the guestmount PID to a file, then after guestunmount spin waiting for this PID to exit.

```
guestmount -a disk.img -i --pid-file guestmount.pid /mnt

# ...
# ...

# Save the PID of guestmount *before* calling guestunmount.
pid="$(cat guestmount.pid)"

# Unmount the filesystem.
guestunmount /mnt

timeout=10

count=$timeout
while kill -0 "$pid" 2>/dev/null && [ $count -gt 0 ]; do
    sleep 1
    ((count--))
done
if [ $count -eq 0 ]; then
    echo "$0: wait for guestmount to exit failed after $timeout seconds"
```

```
     exit 1
 fi

 # Now it is safe to use the disk image.
```

Note that if you use the `guestfs_mount_local` API directly (see ''MOUNT LOCAL'' in **guestfs**(3)) then it is much easier to write a safe, race-free program.

OPTIONS

−a IMAGE
−−add IMAGE

Add a block device or virtual machine image.

The format of the disk image is auto-detected. To override this and force a particular format use the *−−format=..* option.

−a URI
−−add URI

Add a remote disk. See ''ADDING REMOTE STORAGE'' in **guestfish**(1).

−−blocksize=512
−−blocksize=4096
−−blocksize

This parameter sets the sector size of the disk image. It affects all explicitly added subsequent disks after this parameter. Using*−−bloc ksize* with no argument switches the disk sector size to the default value which is usually 512 bytes. See also ''guestfs_add_drive_opts'' in **guestfs**(3).

−c URI
−−connect URI

When used in conjunction with the *−d* option, this specifies the libvirt URI to use. The default is to use the default libvirt connection.

−d LIBVIRT-DOMAIN
−−domain LIBVIRT-DOMAIN

Add disks from the named libvirt domain. If the *−−ro* option is also used, then any libvirt domain can be used. However in write mode, only libvirt domains which are shut down can be named here.

Domain UUIDs can be used instead of names.

−−dir−cache−timeout N

Set the readdir cache timeout to *N* seconds, the default being 60 seconds. The readdir cache [actually, there are several semi-independent caches] is populated after a **readdir**(2) call with the stat and extended attributes of the files in the directory, in anticipation that they will be requested soon after.

There is also a different attribute cache implemented by FUSE (see the FUSE option *−o attr_timeout*), but the FUSE cache does not anticipate future requests, only cache existing ones.

−−echo−keys

When prompting for keys and passphrases, guestfish normally turns echoing off so you cannot see what you are typing. If you are not worried about Tempest attacks and there is no one else in the room you can specify this flag to see what you are typing.

−−fd=FD

Specify a pipe or eventfd file descriptor. When the mountpoint is ready to be used, guestmount writes a single byte to this file descriptor. This can be used in conjunction with*−−no−fork* in order to run guestmount captive under another process.

−−format=raw|qcow2|..
−−format

The default for the *−a* option is to auto-detect the format of the disk image. Using this forces the disk format for *−a* options which follow on the command line. Using *−−format* with no argument switches back to auto-detection for subsequent *−a* options.

If you have untrusted raw-format guest disk images, you should use this option to specify the disk format. This avoids a possible security problem with malicious guests (CVE−2010−3851). See also ‘‘guestfs_add_drive_opts’’ in **guestfs** (3).

−−fuse−help
Display help on special FUSE options (see *−o* below).

−−help
Display brief help and exit.

−i
−−inspector
Using **virt−inspector** (1) code, inspect the disks looking for an operating system and mount filesystems as they would be mounted on the real virtual machine.

−−key SELECTOR
Specify a key for LUKS, to automatically open a LUKS device when using the inspection. `ID` can be either the libguestfs device name, or the UUID of the LUKS device.

−−key `ID`:key:KEY_STRING
Use the specified `KEY_STRING` as passphrase.

−−key `ID`:file:FILENAME
Read the passphrase from *FILENAME*.

−−keys−from−stdin
Read key or passphrase parameters from stdin. The default is to try to read passphrases from the user by opening */dev/tty*.

If there are multiple encrypted devices then you may need to supply multiple keys on stdin, one per line.

−−live
Connect to a live virtual machine. (Experimental, see ‘‘ATTACHING TO RUNNING DAEMONS’’ in **guestfs** (3)).

−m dev[:mountpoint[:options[:fstype]]
−−mount dev[:mountpoint[:options[:fstype]]]
Mount the named partition or logical volume on the given mountpoint **in the guest** (this has nothing to do with mountpoints in the host).

If the mountpoint is omitted, it defaults to */*. You have to mount something on */*.

The third (and rarely used) part of the mount parameter is the list of mount options used to mount the underlying filesystem. If this is not given, then the mount options are either the empty string or `ro` (the latter if the *−−ro* flag is used). By specifying the mount options, you override this default choice. Probably the only time you would use this is to enable ACLs and/or extended attributes if the filesystem can support them:

```
 -m /dev/sda1:/:acl,user_xattr
```

The fourth part of the parameter is the filesystem driver to use, such as `ext3` or `ntfs`. This is rarely needed, but can be useful if multiple drivers are valid for a filesystem (eg: `ext2` and `ext3`), or if libguestfs misidentifies a filesystem.

−−no−fork
Don’t daemonize (or fork into the background).

−n
−−no−sync
By default, we attempt to sync the guest disk when the FUSE mountpoint is unmounted. If you specify this option, then we don’t attempt to sync the disk. See the discussion of autosync in the **guestfs** (3) manpage.

−o OPTION
−−option OPTION
Pass extra options to FUSE.

To get a list of all the extra options supported by FUSE, use the command below. Note that only the FUSE *−o* options can be passed, and only some of them are a good idea.

```
 guestmount --fuse-help
```

Some potentially useful FUSE options:

−o allow_other
Allow other users to see the filesystem. This option has no effect unless you enable it globally in */etc/fuse.conf*.

−o attr_timeout=N
Enable attribute caching by FUSE, and set the timeout to *N* seconds.

−o kernel_cache
Allow the kernel to cache files (reduces the number of reads that have to go through the **guestfs** (3) API). This is generally a good idea if you can afford the extra memory usage.

−o uid=N −o gid=N
Use these options to map all UIDs and GIDs inside the guest filesystem to the chosen values.

−o use_ino
Preserve inode numbers from the underlying filesystem.

Without this option, FUSE makes up its own inode numbers. The inode numbers you see in **stat** (2), `ls −i` etc aren't the inode numbers of the underlying filesystem.

Note this option is potentially dangerous if the underlying filesystem consists of multiple mountpoints, as you may see duplicate inode numbers appearing through FUSE. Use of this option can confuse some software.

−−pid−file FILENAME
Write the PID of the guestmount worker process to `filename`.

−r
−−ro
Add devices and mount everything read-only. Also disallow writes and make the disk appear read-only to FUSE.

This is highly recommended if you are not going to edit the guest disk. If the guest is running and this option is *not* supplied, then there is a strong risk of disk corruption in the guest. We try to prevent this from happening, but it is not always possible.

See also "OPENING DISKS FOR READ AND WRITE" in **guestfish** (1).

−−selinux
This option is provided for backwards compatibility and does nothing.

−v
−−verbose
Enable verbose messages from underlying libguestfs.

−V
−−version
Display the program version and exit.

−w
−−rw
This changes the *−a*, *−d* and *−m* options so that disks are added and mounts are done read-write.

See "OPENING DISKS FOR READ AND WRITE" in **guestfish** (1).

−x
−−trace
Trace libguestfs calls and entry into each FUSE function.

This also stops the daemon from forking into the background (see *−−no−fork*).

FILES

`$XDG_CONFIG_HOME`/libguestfs/libguestfs−tools.conf
`$HOME`/.libguestfs−tools.rc
`$XDG_CONFIG_DIRS`/libguestfs/libguestfs−tools.conf
/etc/libguestfs−tools.conf
This configuration file controls the default read-only or read-write mode (*−−ro* or *−−rw*).

See **libguestfs−tools.conf** (5).

EXIT STATUS

This program returns 0 if successful, or non-zero if there was an error.

SEE ALSO

guestunmount (1), **fusermount** (1), **guestfish** (1), **virt−inspector** (1), **virt−cat** (1), **virt−edit** (1), **virt−tar** (1), **libguestfs−tools.conf** (5), ‘‘MOUNT LOCAL’’ in **guestfs** (3), http://libguestfs.org/, http://fuse.sf.net/.

AUTHORS

Richard W.M. Jones (`rjones at redhat dot com`)

COPYRIGHT

LICENSE

BUGS

To get a list of bugs against libguestfs, use this link: https://bugzilla.redhat.com/buglist.cgi?component=libguestfs&product=Virtualization+Tools

To report a new bug against libguestfs, use this link: https://bugzilla.redhat.com/enter_bug.cgi?component=libguestfs&product=Virtualization+Tools

When reporting a bug, please supply:

- The version of libguestfs.
- Where you got libguestfs (eg. which Linux distro, compiled from source, etc)
- Describe the bug accurately and give a way to reproduce it.
- Run **libguestfs−test−tool** (1) and paste the **complete, unedited** output into the bug report.