NAME

guestunmount − Unmount a guestmounted filesystem

SYNOPSIS

```
guestunmount mountpoint

guestunmount --fd=<FD> mountpoint
```

DESCRIPTION

guestunmount is a utility to clean up mounted filesystems automatically. **guestmount**(1) mounts filesystems using libguestfs. This program unmounts the filesystem when a program or script has finished with it.

guestunmount is a wrapper around the FUSE **fusermount**(1) program, which must exist on the current `PATH`.

There are two ways to use guestunmount. When called as:

```
guestunmount mountpoint
```

it unmounts `mountpoint` immediately.

When called as:

```
guestunmount --fd=FD mountpoint
```

it waits until the pipe `FD` is closed. This can be used to monitor another process and clean up its mountpoint when that process exits, as described below.

FROM PROGRAMS

You can just call `guestunmount mountpoint` from the program, but a more sophisticated way to use guestunmount is to have it monitor your program so it can clean up the mount point if your program exits unexpectedly.

In the program, create a pipe (eg. by calling **pipe**(2)). Let`FD` be the file descriptor number of the read side of the pipe (ie. `pipefd[0]`).

After mounting the filesystem with **guestmount**(1) (on `mountpoint`), fork and run guestunmount like this:

```
guestunmount --fd=FD mountpoint
```

Close the read side of the pipe in the parent process.

Now, when the write side of the pipe (ie. `pipefd[1]`) is closed for any reason, either explicitly or because the parent process exits, guestunmount notices and unmounts the mountpoint.

If your operating system supports it, you should set the `FD_CLOEXEC` flag on the write side of the pipe. This is so that other child processes don't inherit the file descriptor and keep it open.

Guestunmount never daemonizes itself.

FROM SHELL SCRIPTS

Since bash doesn't provide a way to create an unnamed pipe, use a trap to call guestunmount on exit like this:

```
trap "guestunmount mountpoint" EXIT INT QUIT TERM
```

OPTIONS

--fd=FD
: Specify the pipe file descriptor to monitor, and delay cleanup until that pipe is closed.

--help
: Display brief help and exit.

-q

--quiet
Don't display error messages from fusermount. The return status is still set (see ''EXIT STATUS'' below).

--no-retry
--retry=N
By default, guestunmount will retry the fusermount operation up to 5times (that is, it will run it up to 6times = 1try + 5retries).

Use *--no-retry* to make guestunmount run fusermount only once.

Use *--retry=N* to make guestunmount retry `N` times instead of 5.

guestunmount performs an exponential back-off between retries, waiting 1second, 2seconds, 4seconds, etc before each retry.

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

ENVIRONMENT VARIABLES

`PATH`
The **fusermount** (1) program (supplied by FUSE) must be available on the current `PATH`.

EXIT STATUS

This program returns 0 if successful, or one of the following error codes:

`1` Program error, eg. could not allocate memory, could not run fusermount. See the error message printed for more information.

`2` The mount point could not be unmounted even after retrying. See the error message printed for the underlying fusermount error.

`3` The mount point is not mounted.

SEE ALSO

guestmount (1), **fusermount** (1), **pipe** (2), ''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.