NAME

virt−tail − Follow (tail) files in a virtual machine

SYNOPSIS

```
 virt-tail [--options] -d domname file [file ...]

 virt-tail [--options] -a disk.img [-a disk.img ...] file [file ...]
```

DESCRIPTION

`virt-tail` is a command line tool to follow (tail) the contents of `file` where `file` exists in the named virtual machine (or disk image). It is similar to the ordinary command `tail -f`.

Multiple filenames can be given, in which case each is followed separately. Each filename must be a full path, starting at the root directory (starting with '/').

The command keeps running until:

- The user presses the ^C or an interrupt signal is received.
- None of the listed files was found in the guest, or they all get deleted.
- There is an unrecoverable error.

EXAMPLE

Follow */var/log/messages* inside a virtual machine called `mydomain`:

```
 virt-tail -d mydomain /var/log/messages
```

OPTIONS

−−help
Display brief help.

−a file
−−add file
Add *file* which should be a disk image from a virtual machine. If the virtual machine has multiple block devices, you must supply all of them with separate *−a* options.

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
If using libvirt, connect to the given *URI*. If omitted, then we connect to the default libvirt hypervisor.

If you specify guest block devices directly (*−a*), then libvirt is not used at all.

−d guest
−−domain guest
Add all the disks from the named libvirt guest. Domain UUIDs can be used instead of names.

−−echo−keys
When prompting for keys and passphrases, virt-tail 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.

−f
−−follow
This option is ignored. virt-tail always behaves like **tail**(1) *−f*. You don’t need to specify the *−f* option.

−−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.

For example:

```
 virt-tail --format=raw -a disk.img file
```

forces raw format (no auto-detection) for *disk.img*.

```
 virt-tail --format=raw -a disk.img --format -a another.img file
```

forces raw format (no auto-detection) for *disk.img* and reverts to auto-detection for *another.img*.

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).

−−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.

−m dev[:mountpoint[:options[:fstype]]]
−−mount dev[:mountpoint[:options[:fstype]]]
Mount the named partition or logical volume on the given mountpoint.

If the mountpoint is omitted, it defaults to */*.

Specifying any mountpoint disables the inspection of the guest and the mount of its root and all of its mountpoints, so make sure to mount all the mountpoints needed to work with the filenames given as arguments.

If you don’t know what filesystems a disk image contains, you can either run guestfish without this option, then list the partitions, filesystems and LVs available (see ‘‘list-partitions’’, ‘‘list-filesystems’’ and ‘‘lvs’’ commands), or you can use the **virt−filesystems**(1) program.

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
```

Using this flag is equivalent to using the `mount-options` command.

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.

−v
−−verbose
Enable verbose messages for debugging.

−V
−−version
Display version number and exit.

−x Enable tracing of libguestfs API calls.

LOG FILES

To list out the log files from guests, see the related tool **virt−log**(1). It understands binary log formats such as the systemd journal.

WINDOWS PATHS

`virt−tail` has a limited ability to understand Windows drive letters and paths (eg. *E:\foo\bar.txt*).

If and only if the guest is running Windows then:

- Drive letter prefixes like `C:` are resolved against the Windows Registry to the correct filesystem.
- Any backslash (`\`) characters in the path are replaced with forward slashes so that libguestfs can process it.
- The path is resolved case insensitively to locate the file that should be displayed.

There are some known shortcomings:

- Some NTFS symbolic links may not be followed correctly.
- NTFS junction points that cross filesystems are not followed.

EXIT STATUS

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

SEE ALSO

guestfs(3), **guestfish**(1), **virt−copy−out**(1), **virt−cat**(1), **virt−log**(1), **virt−tar−out**(1), **tail**(1), http://libguestfs.org/.

AUTHOR

Richard W.M. Jones http://people.redhat.com/~rjones/

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.