NAME

guestfs−building − How to build libguestfs from source

DESCRIPTION

This manual page describes how to build libguestfs from source.

The main steps are:

- Install the requirements.
- Build, either from the git repository or from a tarball.
- Run the tests.
- Run the tools from the source directory, or install.

REQUIREMENTS

Short cut for Fedora or Red Hat Enterprise Linux (RHEL) users

On Fedora, use **dnf** (8) to install all the requirements:

```
dnf builddep libguestfs
dnf install autoconf automake libtool gettext-devel
```

On systems still using **yum** (8), do:

```
yum-builddep libguestfs
yum install autoconf automake libtool gettext-devel
```

Short cut for Debian or Ubuntu users

Use APT to install all the requirements:

```
apt-get build-dep libguestfs
apt-get install autoconf automake libtool-bin gettext
```

If that command doesn't work, take a look at the Debian source package http://packages.debian.org/source/libguestfs, at the list of `build-depends` and `build-depends-indep`, and install everything listed there.

Full list of requirements

appliance/packagelist.in

Install as many package names found in this file as possible. (It is not strictly required to install all of them).

Note: If you build libguestfs followed by installing appliance packages, the build will not pick them up automatically, even if you do `makeclean`. You have to do this command to clean the old supermin appliance and force a new one to be prepared:

```
make -C appliance clean-supermin-appliance
```

qemu ≥ 1.3.0

Required.

qemu-img ≥ 1.3.0

Required.

kernel ≥ 2.6.34

Required. The following features must be enabled: `virtio-pci`, `virtio-serial`, `virtio-block`, `virtio-net`.

supermin ≥ 5.1.18

Required. For alternatives, see "USING A PREBUILT BINARY APPLIANCE" below.

glibc

Required. We use the custom printf formatters extension of glibc (see "DAEMON CUSTOM PRINTF FORMATTERS" in **guestfs−hacking** (1)).

XDR (tirpc, glibc or other)
: *Required*. We use the XDR implementation from `<rpc/xdr.h>`, which may come from glibc, tirpc or another library.

  The `rpcgen` tool is optional, except if you want to compile from git and/or patch libguestfs with new APIs.

Gcc or Clang
: *Required*. We use `__attribute__((cleanup))` which is a GCC extension also supported by Clang.

Perl
: *Required*. Various build steps and tests are written in Perl. Perl is not needed at runtime except if you need to run a small number of virt tools which are still written in Perl.

Perl `Pod::Man`
Perl `Pod::Simple`
: *Required*. Part of Perl core.

OCaml ≥ 4.04
OCaml findlib
: *Required*.

autoconf
automake
gettext
: *Required* if compiling from git. Optional if compiling from tarball.

cpio
: *Required*.

gperf
: *Required*.

flex
bison
: *Required*.

Perl-compatible Regular Expressions (PCRE2) library
: *Required*.

xorriso, genisoimage or mkisofs
: One of these is *Required*.

libxml2
: *Required*.

ncurses
: *Required*.

augeas ≥ 1.2.0
: *Required*.

xz *Required*.

Jansson ≥ 2.7
: *Required*.

po4a
: *Required* if compiling from git. Optional if compiling from tarball.

hivex ≥ 1.2.7
ocaml-hivex
: *Required*. ocaml-hivex is the OCaml binding for hivex, which is required when building the daemon.

libmagic
: *Required*. This is the library used by the **file**(1) command.

libvirt ≥ 0.10.2
: Optional. Always use the latest possible version of libvirt.

xmllint
: Optional. Used only for tests.

libconfig
: Optional. Used to parse libguestfs's own config files, eg. */etc/libguestfs−tools.conf*.

libselinux
: Optional. Used by the libvirt backend to securely confine the appliance (sVirt).

systemtap
: Optional. For userspace probes.

readline
: Optional. For nicer command line editing in **guestfish**(1).

acl
: Optional. Library and programs for handling POSIX ACLs.

libcap
: Optional. Library and programs for handling Linux capabilities.

libldm
: Optional. Library and **ldmtool**(1) for handling W indows Dynamic Disks.

sd-journal
: Optional. Library for accessing systemd journals.

gdisk
: Optional. GPT disk support.

netpbm
: Optional. Render icons from guests.

icoutils
: Optional. Render icons from Windows guests.

librpm
: Optional. To parse the list of applications from RPM-based guests.

Perl `Expect`
: Optional. Perl module used to test **virt−r escue**(1).

FUSE
: Optional. **fusermount**(1), libfuse and kernel module are all needed if you want **guestmount**(1) and/or mount-local support.

static glibc
: Optional. Used only for testing.

qemu-nbd
nbdkit ≥ 1.12
: Optional. qemu-nbd is used for testing.

uml_mkcow
: Optional. For the UML backend.

curl
: Optional. Used by virt-builder for downloads.

GNU Privacy Guard (GnuPG, gpg) v1 or v2
: Optional. Used by virt-builder for checking digital signatures.

liblzma
: Optional. If available, virt-builder will use this library for fast, parallel uncompression of templates.

python-evtx
: Optional. Used by **virt-log**(1) to parse Windows Event Log files.

OCaml gettext
: Optional. For localizing OCaml virt tools.

ocaml-ounit ≥ 2.0.0
: Optional. For testing the common OCaml modules.

Perl `Module::Build` ≥ 0.19
Perl `Test::More`
: Optional. Used to build and test the Perl bindings.

Python ≥ 3.6
: Optional. Used to build the Python bindings. Python2 support was removed in libguestfs1.42.1.

Python `unittest`
: Optional. Used to run the Python testsuite.

Ruby
rake
rubygem-minitest
rubygem-rdoc
: Optional. Used to build the Ruby bindings.

Java ≥ 1.6
: Optional. Java, JNI and jpackage-utils are needed for building Java bindings.

GHC
: Optional. Used to build the Haskell bindings.

PHP
phpize
: Optional. Used to build the PHP bindings.

glib2
gobject-introspection
gjs
: Optional. Used to build and test the GObject bindings.

vala
: Optional. Used to build the Vala bindings.

LUA
: Optional. Used to build the LUA bindings.

Erlang ≥ 23
ei
: Optional. Used to build the Erlang bindings. Note that Erlang ≤ 22 will not work unless you use libguestfs ≤ 1.42.

golang ≥ 1.1.1
: Optional. Used to build the Go bindings.

valgrind
: Optional. For testing memory problems.

libvirt-python
: Optional. For testing Python libvirt/libguestfs interactions.

Perl `libintl`
: Optional.

bash-completion
: Optional. For tab-completion of commands in bash.

libtsk
: Optional. Library for filesystem forensics analysis.

yara ≥ 4.0.0
: Optional. Tool for categorizing files based on their content.

BUILDING FROM GIT

You will need to install additional dependencies `autoconf`, `automake`, `gettext`, OCaml findlib and po4a when building from git.

```
git clone https://github.com/libguestfs/libguestfs
cd libguestfs
git submodule update --init
autoreconf -i
./configure CFLAGS=-fPIC
make
```

BUILDING FROM TARBALLS

Tarballs are downloaded from http://download.libguestfs.org/. Stable tarballs are signed with the GnuPG key for `rich@annexia.org`, see https://pgp.mit.edu/pks/lookup?op=vindex&search=0x91738F73E1B768A0. The fingerprint is`F777 4FB1 AD07 4A7E 8C87 67EA 9173 8F73 E1B7 68A0`.

Download and unpack the tarball.

```
cd libguestfs-1.xx.yy
./configure
make
```

RUNNING THE TESTS

DO NOT run the tests as root! Libguestfs can be built and tested as non-root. Running the tests as root could even be dangerous, don't do it.

To sanity check that the build worked, do:

```
make quickcheck
```

To run the basic tests, do:

```
make check
```

There are many more tests you can run. See **guestfs-hacking** (1) for details.

INSTALLING

DO NOT use `make install`! You'll end up with conflicting versions of libguestfs installed, and this causes constant headaches for users. See the next section for how to use the *./run* script instead.

Distro packagers can use:

```
make INSTALLDIRS=vendor DESTDIR=[temp-build-dir] install
```

THE ./run SCRIPT

You can run **guestfish** (1), **guestmount** (1) and the virt tools without needing to install them by using the *./run* script in the top directory. This script works by setting several environment variables.

For example:

```
./run guestfish [usual guestfish args ...]

./run virt-inspector [usual virt-inspector args ...]
```

The *./run* script adds every libguestfs binary to the `$PATH`, so the above examples run guestfish and virt-inspector from the build directory (not the globally installed guestfish if there is one).

You can use the script from any directory. If you wanted to run your own libguestfs-using program, then the following command will also work:

```
/path/to/libguestfs/run ./my_program [...]
```

You can also run the C programs under valgrind like this:

```
./run valgrind [valgrind opts...] virt−cat [virt−cat opts...]
```

or under gdb:

```
./run gdb −−args virt−cat [virt−cat opts...]
```

This also works with sudo (eg. if you need root access for libvirt or to access a block device):

```
sudo ./run virt−cat −d LinuxGuest /etc/passwd
```

To set environment variables, you can either do:

```
LIBGUESTFS_HV=/my/qemu ./run guestfish
```

or:

```
./run env LIBGUESTFS_HV=/my/qemu guestfish
```

*local** FILES

Files in the top source directory that begin with the prefix *local** are ignored by git. These files can contain local configuration or scripts that you need to build libguestfs.

I have a file called *localconfigure* which is a simple wrapper around *configure* containing local configure customizations that I need. It looks like this:

```
. localenv
./configure.sh \
    −C \
    −−enable−werror \
    "$@"
```

So I can use this to build libguestfs:

```
./localconfigure && make
```

If there is a file in the top build directory called *localenv*, then it will be sourced by `make`. This file can contain any local environment variables needed, eg. for skipping tests:

```
# Skip this test, it is broken.
export SKIP_TEST_BTRFS_FSCK=1
```

Note that *localenv* is included by the top Makefile (so it's a Makefile fragment). But if it is also sourced by your *localconfigure* script then it is used as a shell script.

SELECTED ./configure SETTINGS

There are many `./configure` options. Use:

```
./configure −−help
```

to list them all. This section covers some of the more important ones.

−−disable−appliance −−disable−daemon
See "USING A PREBUILT BINARY APPLIANCE" below.

−−disable−erlang
−−disable−gobject
−−disable−golang
−−disable−haskell
−−disable−lua
−−disable−ocaml

−−disable−perl
−−disable−php
−−disable−python
−−disable−ruby

Disable specific language bindings, even if `./configure` finds all the necessary libraries are installed so that they could be compiled.

Note that disabling OCaml (bindings) or Perl will have the knock-on effect of disabling parts of the test suite and some tools.

OCaml is required to build libguestfs and this requirement cannot be removed. Using *−−disable−ocaml* only disables the bindings and OCaml tools.

−−disable−fuse

Disable FUSE support in the API and the **guestmount** (1) tool.

−−disable−static

Don't build a static linked version of the libguestfs library.

−−enable−install−daemon

Normally **guestfsd** (8) is not installed by `make install`, since that wouldn't be useful (instead it is ''installed'' inside the supermin appliance). However if packagers are building ''libguestfs live'' then they should use this option.

−−enable−werror

This turns compiler warnings into errors (ie. `−Werror`). Use this for development, especially when submitting patches. It should generally *not* be used for production or distro builds.

−−with−default−backend=libvirt

This controls the default method that libguestfs uses to run qemu (see ''BACKEND'' in **guestfs** (3)). If not specified, the default backend is `direct`, which means libguestfs runs qemu directly.

Fedora and Red Hat Enterprise Linux (RHEL) ≥ 7 use this flag to change the default backend to `libvirt`, because (especially in RHEL) the policy is not to allow any program to run qemu except via libvirt.

Note that despite this setting, all backends are built into libguestfs, and you can override the backend at runtime by setting the `$LIBGUESTFS_BACKEND` environment variable (or using API methods).

−−with−distro=REDHAT|DEBIAN|...

Libguestfs needs to know which Linux distro is in use so it can choose package names for the appliance correctly (see for example *appliance/packagelist.in*). It normally does this automatically.

However if you can building or packaging libguestfs on a new distro then you can use *−−with−distro* to specify that the distro is similar to an existing one (eg. *−−with−distro=REDHAT* if the distro is a new Red Hat or CentOS derivative).

Note that if your distro is completely new then it may still require upstream modifications.

−−with−extra=''*distroname*=*version*,libvirt,...''
−−with−extra=''local''

This option controls the ''extra'' field returned by ''guestfs_version'' in **guestfs** (3) and also printed by virt tools' *−−version* option. It is a free text field, but a good idea is to encode a comma-separated list of facts such as the distro name and version, whether libvirt is the default backend, and anything else that may help with debugging problems raised by users.

For custom and/or local builds, this can be set to `local` to indicate this is *not* a distro build.

−−without−libvirt

Compile libguestfs without libvirt support, even if libvirt development libraries are installed.

−−with−qemu="bin1 bin2 ...**"**
Provide an alternate qemu binary (or list of binaries). This can be overridden at runtime by setting the `LIBGUESTFS_HV` environment variable.

−−with−supermin−packager−config=*yum.conf*
This passes the −−*packager−config* option to **supermin**(1).

The most common use for this is to build the appliance using an alternate repository (instead of using the installed yum/dnf/apt/etc configuration to find and download packages). You might need to use this if you want to build libguestfs without having a network connection. Examples of using this can be found in the Fedora `libguestfs.spec` file (see "BUILDING A PACKAGE FOR FEDORA" below for resources).

−−with−supermin−extra−options="−−opt1 −−opt2 ...**"**
Pass additional options to **supermin**(1). See*appliance/mak e.sh.in* to understand precisely what this does.

PYTHON
This environment variable may be set to point to a python binary (eg. `python3`). When `./configure` runs, it inspects this python binary to find the version of Python, the location of Python libraries and so on.

SUPERMIN
This environment variable can be set to choose an alternative **supermin**(1) binary. This might be used, for example, if you want to use a newer upstream version of supermin than is packaged for your distro, or if supermin is not packaged at all. On RHEL 7, you must set `SUPERMIN=/usr/bin/supermin5` when compiling libguestfs.

NOTES ABOUT QEMU AND KVM

A common problem is with broken or incompatible qemu releases.

Different versions of qemu have problems booting the appliance for different reasons. This varies between versions of qemu, and Linux distributions which add their own patches.

If you find a problem, you could try using your own qemu built from source (qemu is very easy to build from source), with a "qemu wrapper". See "QEMU WRAPPERS" in **guestfs**(3).

By default the configure script will look for qemu-kvm (KVM support). KVM is much faster than using plain qemu.

You may also need to enable KVM support for non-root users, by following these instructions: http://www.linux−kvm.org/page/FAQ#How_can_I_use_kvm_with_a_non−privileged_user.3F

On some systems, this will work too:

```
 chmod 0666 /dev/kvm
```

On some systems, the chmod will not survive a reboot, and you will need to make edits to the udev configuration.

USING CLANG (LLVM) INSTEAD OF GCC

```
 export CC=clang
 ./configure
 make
```

USING A PREBUILT BINARY APPLIANCE

To understand what the libguestfs appliance means, see **guestfs−internals**(1).

If you are using non-Linux, or a Linux distribution that does not have **supermin**(1) support, or simply if you don't want to build your own libguestfs appliance, then you can use one of the prebuilt binary appliances that we supply: http://libguestfs.org/download/binaries/appliance

Build libguestfs like this:

```
./configure --disable-appliance --disable-daemon
make
```

Set `$LIBGUESTFS_PATH` to the path where you unpacked the appliance tarball, eg:

```
export LIBGUESTFS_PATH=/usr/local/lib/guestfs/appliance
```

and run the libguestfs programs and virt tools in the normal way, eg. using the *./run* script (see above).

BUILDING A PACKAGE FOR FEDORA

The Fedora spec file is stored under: http://pkgs.fedoraproject.org/cgit/rpms/libguestfs.git/

Libguestfs is built in Fedora using the ordinary Fedora build system (Koji).

BUILDING A PACKAGE FOR RED HAT ENTERPRISE LINUX

Red Hat Enterprise Linux (RHEL) builds of libguestfs are heavily patched. There are broadly two types of patches we apply:

- We disable many features that we do not wish to support for RHEL customers. For example, the ‘‘libguestfs live’’ feature is disabled.
- We backport upstream features.

The patches we apply to RHEL releases are available publically in the upstream git repository, in a branch called `rhel-x.y`

For example, the RHEL 7.3 patches are available here: https://github.com/libguestfs/libguestfs/commits/rhel−7.3

The sources and spec files for RHEL versions of libguestfs are available on https://git.centos.org/project/rpms, and see also https://wiki.centos.org/Sources.

SEE ALSO

guestfs (3), **guestfs−examples** (3), **guestfs−hacking** (1), **guestfs−internals** (1), **guestfs−performance** (1), **guestfs−release−notes** (1), **guestfs−testing** (1), **libguestfs−test−tool** (1), **libguestfs−make−fixed−appliance** (1), http://libguestfs.org/.

AUTHORS

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

COPYRIGHT

Copyright (C) 2009−2020 Red Hat Inc.

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.