



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

## ***Rocky Enterprise Linux 9.2 Manual Pages on command 'fakeroot.1'***

**\$ man fakeroot.1**

fakeroot(1)

Debian manual

fakeroot(1)

### NAME

fakeroot - run a command in an environment faking root privileges for file manipulation

### SYNOPSIS

fakeroot [-l|--lib library] [--faked faked-binary] [-i load-file] [-s save-file] [-u|--un?

known-is-real ] [-b|--fd-base ] [-h|--help ] [-v|--version ] [-] [command]

### DESCRIPTION

fakeroot runs a command in an environment wherein it appears to have root privileges for file manipulation. This is useful for allowing users to create archives (tar, ar, .deb etc.) with files in them with root permissions/ownership. Without fakeroot one would need to have root privileges to create the constituent files of the archives with the correct permissions and ownership, and then pack them up, or one would have to construct the ar? chives directly, without using the archiver.

fakeroot works by replacing the file manipulation library functions (chmod(2), stat(2) etc.) by ones that simulate the effect the real library functions would have had, had the user really been root. These wrapper functions are in a shared library /usr/lib/\*/libfake? root-\* .so or similar location on your platform. The shared object is loaded through the LD\_PRELOAD mechanism of the dynamic loader. (See ld.so(8))

If you intend to build packages with fakeroot, please try building the fakeroot package first: the "debian/rules build" stage has a few tests (testing mostly for bugs in old fakeroot versions). If those tests fail (for example because you have certain libc5 pro? grams on your system), other packages you build with fakeroot will quite likely fail too, but possibly in much more subtle ways.

Also, note that it's best not to do the building of the binaries themselves under fake? root. Especially configure and friends don't like it when the system suddenly behaves dif? ferently from what they expect. (or, they randomly unset some environment variables, some of which fakeroott needs).

## OPTIONS

**-l library, --lib library**

Specify an alternative wrapper library.

**--faked binary**

Specify an alternative binary to use as faked.

**[--] command**

Any command you want to be ran as fakeroott. Use ?--? if in the command you have other options that may confuse fakeroott's option parsing.

**-s save-file**

Save the fakeroott environment to save-file on exit. This file can be used to re? store the environment later using -i. However, this file will leak and fakeroott will behave in odd ways unless you leave the files touched inside the fakeroott alone when outside the environment. Still, this can be useful. For example, it can be used with rsync(1) to back up and restore whole directory trees complete with user, group and device information without needing to be root. See /usr/share/doc/fakeroott/README.saving for more details.

**-i load-file**

Load a fakeroott environment previously saved using -s from load-file. Note that this does not implicitly save the file, use -s as well for that behaviour. Using the same file for both -i and -s in a single fakeroott invocation is safe.

**-u, --unknown-is-real**

Use the real ownership of files previously unknown to fakeroott instead of pretend? ing they are owned by root:root.

**-b fd** Specify fd base (TCP mode only). fd is the minimum file descriptor number to use for TCP connections; this may be important to avoid conflicts with the file de? scriptors used by the programs being run under fakeroott.

**-h** Display help.

**-v** Display version.

## EXAMPLES

Here is an example session with fakeroot. Notice that inside the fake root environment file manipulation that requires root privileges succeeds, but is not really happening.

```
$ whoami
joost
$ fakeroot /bin/bash
# whoami
root
# mknod hda3 b 3 1
# ls -ld hda3
brw-r--r-- 1 root root 3, 1 Jul 2 22:58 hda3
# chown joost:root hda3
# ls -ld hda3
brw-r--r-- 1 joost root 3, 1 Jul 2 22:58 hda3
# ls -ld /
drwxr-xr-x 20 root root 1024 Jun 17 21:50 /
# chown joost:users /
# chmod a+w /
# ls -ld /
drwxrwxrwx 20 joost users 1024 Jun 17 21:50 /
# exit
$ ls -ld /
drwxr-xr-x 20 root root 1024 Jun 17 21:50 //
$ ls -ld hda3
-rw-r--r-- 1 joost users 0 Jul 2 22:58 hda3
```

Only the effects that user joost could do anyway happen for real.

fakeroot was specifically written to enable users to create Debian GNU/Linux packages (in the deb(5) format) without giving them root privileges. This can be done by commands like dpkg-buildpackage -rfakeroot or debuild -rfakeroot (actually, -rfakeroot is default in de? build nowadays, so you don't need that argument).

## SECURITY ASPECTS

fakeroot is a regular, non-setuid program. It does not enhance a user's privileges, or de? crease the system's security.

/usr/lib/\*/libfakeroot-\* .so The shared library containing the wrapper functions.

## ENVIRONMENT

### FAKEROOTKEY

The key used to communicate with the fakeroot daemon. Any program started with the right LD\_PRELOAD and a FAKEROOTKEY of a running daemon will automatically connect to that daemon, and have the same "fake" view of the file system's permissions/own? erships. (assuming the daemon and connecting program were started by the same user).

### LD\_LIBRARY\_PATH

### LD\_PRELOAD

Fakeroot is implemented by wrapping system calls. This is accomplished by setting LD\_LIBRARY\_PATH=/usr/lib/fakeroot and LD\_PRELOAD=libfakeroot.so.0. That library is loaded before the system's C library, and so most of the library functions are in? tercepted by it. If you need to set either LD\_LIBRARY\_PATH or LD\_PRELOAD from within a fakeroot environment, it should be set relative to the given paths, as in

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/foo/bar/
```

## LIMITATIONS

### Library versions

Every command executed within fakeroot needs to be linked to the same version of the C library as fakeroot itself.

### open()/create()

fakeroot doesn't wrap open(), create(), etc. So, if user joost does either

```
touch foo
```

```
fakeroot
```

```
ls -al foo
```

or the other way around,

```
fakeroot
```

```
touch foo
```

```
ls -al foo
```

fakeroot has no way of knowing that in the first case, the owner of foo really should be joost while the second case it should have been root. For the Debian packaging, defaulting to giving all "unknown" files uid=gid=0, is always OK. The real way around this is to wrap open() and create(), but that creates other prob?

lems, as demonstrated by the libtricks package. This package wrapped many more functions, and tried to do a lot more than fakeroott. It turned out that a minor upgrade of libc (from one where the stat() function didn't use open() to one with a stat() function that did (in some cases) use open()), would cause unexplainable segfaults (that is, the libc6 stat() called the wrapped open()), which would then call the libc6 stat(), etc). Fixing them wasn't all that easy, but once fixed, it was just a matter of time before another function started to use open(), never mind trying to port it to a different operating system. Thus I decided to keep the number of functions wrapped by fakeroott as small as possible, to limit the likelihood of collisions?.

#### GNU configure (and other such programs)

fakeroott, in effect, is changing the way the system behaves. Programs that probe the system like GNU configure may get confused by this (or if they don't, they may stress fakeroott so much that fakeroott itself becomes confused). So, it's advisable not to run "configure" from within fakeroott. As configure should be called in the "debian/rules build" target, running "dpkg-buildpackage -rfakeroott" correctly takes care of this.

#### BUGS

It doesn't wrap open(). This isn't bad by itself, but if a program does open("file", O\_WRONLY, 000), writes to file "file", closes it, and then again tries to open to read the file, then that open fails, as the mode of the file will be 000. The bug is that if root does the same, open() will succeed, as the file permissions aren't checked at all for root. I choose not to wrap open(), as open() is used by many other functions in libc (also those that are already wrapped), thus creating loops (or possible future loops, when the implementation of various libc functions slightly change).

#### COPYING

fakeroott is distributed under the GNU General Public License. (GPL 2.0 or greater).

#### AUTHORS

joost witteveen

<joostje@debian.org>

Clint Adams

<clint@debian.org>

Timo Savola

## MANUAL PAGE

mostly by J.H.M. Dassen <jdassen@debian.org> Rather a lot mods/additions by joost and Clint.

## SEE ALSO

[faked\(1\)](#) [dpkg-buildpackage\(1\)](#), [debuild\(1\)](#) [/usr/share/doc/fakeroot/DEBUG](#)

Debian Project

5 October 2014

[fakeroot\(1\)](#)