

**NAME**

schroot – securely enter a chroot environment

**SYNOPSIS**

```
schroot [-h|--help | -V|--version | -l|--list | -i|--info | --config | --location | --automatic-session
| -b|--begin-session | --recover-session | -r|--run-session | -e|--end-session] [-f|--force] [-n session-name|--session-name=session-name] [-d directory|--directory=directory] [-u user|--user=user]
[-p|--preserve-environment] [-q|--quiet | -v|--verbose] [-c chroot|--chroot=chroot | --all |
--all-chroots | --all-source-chroots | --all-sessions] | --exclude-aliases [--] [COMMAND [
ARG1 [ ARG2 [ ARGn]]]]
```

**DESCRIPTION**

**schroot** allows the user to run a command or a login shell in a chroot environment. If no command is specified, a login shell will be started in the user’s current working directory inside the chroot.

The command is a program, plus as many optional arguments as required. Each argument may be separately quoted.

The directory the command or login shell is run in depends upon the context. See `--directory` option below for a complete description.

All chroot usage will be logged in the system logs. Under some circumstances, the user may be required to authenticate themselves; see the section “*Authentication*”, below.

If no chroot is specified, the chroot name or alias ‘default’ will be used as a fallback. This is equivalent to “`--chroot=default`”.

**OVERVIEW**

There is often a need to run programs in a virtualised environment rather than on the host system directly. Unlike other virtualisation systems such as **kvm** or **Xen**, schroot does not virtualise the entire system; it only virtualises the filesystem, and some parts of the filesystem may still be shared with the host. It is therefore fast, lightweight and flexible. However, it does not virtualise other aspects of the system, such as shared memory, networking, devices etc., and so may be less secure than other systems, depending upon its intended use. Some examples of existing uses for schroot include:

- Running an untrusted program in a sandbox, so that it can’t interfere with files on the host system; this may also be used to limit the damage a compromised service can inflict upon the host
- Using a *defined* or *clean* environment, to guarantee the reproducibility and integrity of a given task
- Using different versions of an operating system, or even different operating systems altogether, e.g. different GNU/Linux distributions
- Running 32-bit programs using a 32-bit chroot on a 64-bit host system
- Automatic building of Debian packages using **sbuid**(1), which builds each package in a pristine chroot snapshot when using LVM snapshots or unions
- Supporting multiple system images in a cluster setup, where modifying the base image is time-consuming and/or supporting all the required configurations needed by users is difficult: different chroots can support all the different configurations required, and cluster users may be given access to the chroots they need (which can include root access for trusted users to maintain their own images)

A chroot may be used directly as root by running **chroot**(8), but normal users are not able to use this command. **schroot** allows access to chroots for normal users using the same mechanism, but with permissions checking and allowing additional automated setup of the chroot environment, such as mounting additional filesystems and other configuration tasks. This automated setup is done through the action of *setup scripts* which may be customised and extended to perform any actions required.

**OPTIONS**

**schroot** accepts the following options:

**Actions****-h, --help**

Show help summary.

**-V, --version**

Print version information.

**-l, --list**

List all available chroots.

**-i, --info**

Print detailed information about the specified chroots.

**--config**

Print configuration of the specified chroots. This is useful for testing that the configuration in use is the same as the configuration file. Any comments in the original file will be missing.

**--location**

Print location (path) of the specified chroots. Note that chroot types which can only be used within a session will not have a location until they are active.

**General options****-q, --quiet**

Print only essential messages.

**-v, --verbose**

Print all messages.

**Chroot selection****-c, --chroot=*chroot***Specify a chroot or active session to use. This option may be used multiple times to specify more than one chroot, in which case its effect is similar to *--all*. The chroot name may be prefixed with a *namespace*; see the section “*Chroot Namespaces*”, below.**-a, --all**Select all chroots, source chroots and active sessions. When a command has been specified, the command will be run in all chroots, source chroots and active sessions. If *--info* has been used, display information about all chroots. This option does not make sense to use with a login shell (when no command has been specified). This option is equivalent to “*--all-chroots --all-source-chroots --all-sessions*”.**--all-chroots**Select all chroots. Identical to *--all*, except that source chroots and active sessions are not considered.**--all-sessions**Select all active sessions. Identical to *--all*, except that chroots and source chroots are not considered.**--all-source-chroots**Select all source chroots. Identical to *--all*, except that chroots and sessions are not considered.**--exclude-aliases**

Do not select aliases in addition to chroots. This ensures that only real chroots are selected, and are only listed once.

**Chroot environment****-d, --directory=*directory***Change to *directory* inside the chroot before running the command or login shell. If *directory* is not available, schroot will exit with an error status.

The default behaviour is as follows (all directory paths are inside the chroot). A login shell is run in the current working directory. If this is not available, it will try \$HOME (when

`--preserve-environment` is used), then the user's home directory, and / inside the chroot in turn. A command is always run in the current working directory inside the chroot. If none of the directories are available, schroot will exit with an error status.

**-u, --user=*user***

Run as a different user. The default is to run as the current user. If required, the user may be required to authenticate themselves with a password. For further information, see the section "Authentication", below.

**-p, --preserve-environment**

Preserve the user's environment inside the chroot environment. The default is to use a clean environment; this option copies the entire user environment and sets it in the session. The environment variables allowed are subject to certain restrictions; see the section "Environment", below.

### Session actions

**--automatic-session**

Begin, run and end a session automatically. This is the default action, so does not require specifying in normal operation.

**-b, --begin-session**

Begin a session. A unique session identifier (session ID) is returned on standard output. The session ID is required to use the other session options. Note that the session identifier may be specified with the `--session-name` option.

**--recover-session**

Recover an existing session. If an existing session has become unavailable, for example becoming unmounted due to a reboot, this option will make the session available for use again, for example by remounting it. The session ID is specified with the `--chroot` option.

**-r, --run-session**

Run an existing session. The session ID is specified with the `--chroot` option.

**-e, --end-session**

End an existing session. The session ID is specified with the `--chroot` option.

### Session options

**-n, --session-name=*session-name***

Name a session. The specified *session-name* replaces the default session name containing an automatically-generated session ID. The session name must not contain a namespace qualifier, since sessions are always created within the 'session:' namespace. The session name is also subject to the chroot naming restrictions documented in **schroot.conf(5)**.

**-f, --force**

Force a session operation, even if it would otherwise fail. This may be used to forcibly end a session, even if it has active users. This does not guarantee that the session will be ended cleanly; filesystems may not be unmounted, for example.

### Separator

**--** End of options. Used to indicate the end of the schroot options; any following options will be passed to the command being run, rather than to schroot.

## AUTHENTICATION

If the user is not an allowed user, or a member of the allowed groups (or if changing to root, the allowed root users or allowed root groups) for the specified chroot(s), permission will be immediately denied. If switching users, and the user running the command has access, the user will be required to authenticate themselves using the credentials of the user being switched to.

On systems supporting Pluggable Authentication Modules (PAM), schroot will use PAM for authentication and authorisation of users. If and when required, schroot will prompt for a password. If PAM is not available, all authentication will automatically fail (user switching is *not* supported without PAM).

Note that when PAM is in use, the root user is not granted any special privileges by default in the program.

However, the default PAM configuration permits root to log in without a password (*pam\_rootok.so*), but this may be disabled to prevent root from accessing any chroots except if specifically permitted. In such a situation, root must be added to the allowed users or groups as for any other user or group. If PAM is not available, the root user will be permitted to access all chroots, even when not explicitly granted access.

## CHROOT NAMESPACES

### Namespace basics

There are three different types of chroot: regular chroots, source chroots and session chroots. These different types of chroot are separated into different *namespaces*. A namespace is a prefix to a chroot name. Currently there are three namespaces: ‘chroot:’, ‘source:’ and ‘session:’. Use `--list --all` to list all available chroots in all namespaces. Because ‘:’ is used as the separator between namespace and chroot names, it is not permitted to use this character in chroot names.

Depending upon the action you request schroot to take, it may look for the chroot in one of the three namespaces, or a particular namespace may be specified. For example, a chroot named “sid” is actually named “chroot:sid” if the namespace is included, but the namespace may be omitted for most actions.

### Source chroots

Some chroot types, for example LVM snapshots and Btrfs snapshots, provide session-managed copy-on-write snapshots of the chroot. These also provide a *source chroot* to allow easy access to the filesystem used as a source for snapshotting. These are regular chroots as well, just with the snapshotting disabled. For a chroot named “sid-snapshot” (i.e. with a fully qualified name of “chroot:sid-snapshot”), there will also be a corresponding source chroot named “source:sid-snapshot”. Earlier versions of schroot provided source chroots with a ‘-source’ suffix. These are also provided for compatibility. In this example, this would be called “chroot:sid-snapshot-source”. These compatibility names will be dropped in a future version, so programs and scripts should switch to using the namespace-qualified names rather than the old suffix.

### Session chroots

All sessions created with `--begin-session` are placed within the ‘session:’ namespace. A session named with `--session-name` may have any name, even the same name as the chroot it was created from, providing that it is unique within this namespace. This was not permitted in previous versions of schroot which did not have namespaces.

### Actions and default namespaces

All actions use ‘chroot:’ as the default namespace, with some session actions being the exception. `--run-session`, `--recover-session` and `--end-session` use ‘session:’ as the default namespace instead, since these actions work on session chroots. The upshot is that the namespace is usually never required except when you need to work with a chroot in a namespace other than the default, such as when using a source chroot. To make chroot selection unambiguous, it is always possible to use the full name including the namespace, even when not strictly required.

## PERFORMANCE

Performance on some filesystems, for example Btrfs, is bad when running dpkg due to the amount of fsync operations performed. This may be mitigated by installing the eatmydata package and then adding eatmydata to the *command-prefix* configuration key, which disables all fsync operations. Note that this should only be done in snapshot chroots where data loss is not an issue. This is useful when using a chroot for package building, for example.

## EXAMPLES

### List available chroots

```
% schroot -l
chroot:default
chroot:etch
chroot:sid
chroot:testing
chroot:unstable
```

**Get information about a chroot**

```
% schroot -i -c sid
— Chroot —
Name                sid
Description         Debian sid (unstable)
Type                plain
Priority             3
Users               rleigh
Groups              sbuild
Root Users
Root Groups         sbuild
Aliases             unstable unstable-sbuild unstable-p
owerpc-sbuild
Environment Filter  ^(BASH_ENV|CDPATH|ENV|HOSTALIASES|IFS|KRB5_CONFIG|KRB5CONFDIR|KRB5KFILE|KRB5_CONF|LD_*|LOCALDOMA
IN|NLSPATH|PATH_LOCALE|RES_OPTIONS|TERMINFO|TERMINFO_DIRS|TE
RMPATH)$
Run Setup Scripts   true
Script Configuration script-defaults
Session Managed     true
Personality         linux32
Location            /srv/chroot/sid
```

Use `--all` or `-c` multiple times to use all or multiple chroots, respectively.

**Running commands in a chroot**

```
% schroot -c sid /bin/ls
[sid chroot] Running command: "/bin/ls"
CVS                sbuild-chroot.c   sbuild-session.h  schroot.conf.5
Makefile           sbuild-chroot.h   schroot.1          schroot.conf.5.in
Makefile.am        sbuild-config.c   schroot.1.in
Makefile.in        sbuild-config.h   schroot.c
pam                sbuild-session.c  schroot.conf
```

```
% schroot -c sid -- ls -l | head -n 5
```

```
[sid chroot] Running command: "ls -l"
```

```
ABOUT-NLS
AUTHORS
COPYING
ChangeLog
INSTALL
```

Use `--` to allow options beginning with `'-` or `'--` in the command to run in the chroot. This prevents them being interpreted as options for schroot itself. Note that the top line was echoed to standard error, and the remaining lines to standard output. This is intentional, so that program output from commands run in the chroot may be piped and redirected as required; the data will be the same as if the command was run directly on the host system.

**Switching users**

```
% schroot -c sid -u root
Password:
[sid chroot] (rleigh→root) Running login shell: "/bin/bash"
#
```

If the user `'rleigh'` was in `root-users` in `/usr/local/etc/schroot/schroot.conf`, or one of the groups he belonged to was in `root-groups`, they would be granted root access without authentication, but the PAM authorisation step is still applied.

## Sessions

A chroot may be needed to run more than one command. In particular, where the chroot is created on the fly from an LVM LV or a file on disc, there is a need to make the chroot persistent while a given task (or set of tasks) is performed. Sessions exist for this purpose. For simple chroot types such as ‘plain’ and ‘directory’, sessions may be created but are not strictly necessary.

Let’s start by looking at a session-capable chroot:

```
% schroot -i -c sid-snap
— Chroot —
Name                sid-snap
Description          Debian sid snapshot
Type                 lvm-snapshot
Priority              3
Users                maks rleigh
Groups               sbuild
Root Users
Root Groups          root sbuild
Aliases
Environment Filter  ^(BASH_ENV|CDPATH|ENV|HOSTALIASES|IFS|KRB5_CONFIG|KRB5CONFDIR|KRBTKFILE|KRB_CONF|LD_*|LOCALDOMAIN|NLSPATH|PATH_LOCALE|RES_OPTIONS|TERMINFO|TERMINFO_DIRS|TE
RMPATH)$
Run Setup Scripts   true
Script Configuration script-defaults
Session Managed      true
Personality          linux
Device               /dev/hda_vg/sid_chroot
Mount Options        -o atime,async,user_xattr
Source Users
Source Groups        root rleigh
Source Root Users
Source Root Groups   root rleigh
LVM Snapshot Options --size 2G -c 128
```

Note that the *Session Managed* option is set to ‘true’. This is a requirement in order to use session management, and is supported by most chroot types. Next, we will create a new session:

```
% schroot -b -c sid-snap
sid-snap-46195b04-0893-49bf-beb8-0d4ccc899f0f
```

The session ID of the newly-created session is returned on standard output. It is common to store it like this:

```
% SESSION=$(schroot -b -c sid-snap)
% echo $SESSION
sid-snap-46195b04-0893-49bf-beb8-0d4ccc899f0f
```

The session may be used just like any normal chroot. This is what the session looks like:

```
% schroot -i -c sid-snap-46195b04-0893-49bf-beb8-0d4ccc899f0f
— Session —
Name                sid-snap-46195b04-0893-49bf-beb8-0d
4ccc899f0f
Description          Debian sid snapshot
Type                 lvm-snapshot
Priority              3
Users                maks rleigh
Groups               sbuild
```

```

Root Users
Root Groups          root sbuild
Aliases
Environment Filter   ^(BASH_ENV|CDPATH|ENV|HOSTALIASES|IFS|KRB5_CONFIG|KRBCONFDIR|KRBTXFILE|KRB_CONF|LD_*|LOCALDOMAIN|NLSPATH|PATH_LOCALE|RES_OPTIONS|TERMINFO|TERMINFO_DIRS|TE
RMPATH)$
Run Setup Scripts    true
Script Configuration script-defaults
Session Managed      true
Personality          linux
Mount Location       /var/lib/schroot/mount/sid-snap-461
95b04-0893-49bf-beb8-0d4ccc899f0f
Path                 /var/lib/schroot/mount/sid-snap-461
95b04-0893-49bf-beb8-0d4ccc899f0f
Mount Device         /dev/hda_vg/sid-snap-46195b04-0893-
49bf-beb8-0d4ccc899f0f
Device               /dev/hda_vg/sid_chroot
Mount Options        -o atime,async,user_xattr
Source Users
Source Groups        root rleigh
Source Root Users
Source Root Groups   root rleigh
LVM Snapshot Device /dev/hda_vg/sid-snap-46195b04-0893-
49bf-beb8-0d4ccc899f0f
LVM Snapshot Options --size 2G -c 128

```

Now the session has been created, commands may be run in it:

```

% schroot -r -c sid-snap-46195b04-0893-49bf-beb8-0d4ccc899f0f -- uname
-sr
I: [sid-snap-46195b04-0893-49bf-beb8-0d4ccc899f0f chroot] Running com-
mand: "uname -sr"
Linux 2.6.18-3-powerpc
% schroot -r -c $SESSION -- uname -sr
I: [sid-snap-fe170af9-d9be-4800-b1bd-de275858b938 chroot] Running com-
mand: "uname -sr"
Linux 2.6.18-3-powerpc

```

When all the commands to run in the session have been performed, the session may be ended:

```

% schroot -e -c sid-snap-46195b04-0893-49bf-beb8-0d4ccc899f0f.
% schroot -e -c $SESSION.

```

Finally, the session names can be long and unwieldy. A name may be specified instead of using the automatically generated session ID:

```

% schroot -b -c sid-snap -n my-session-name.
my-session-name

```

## TROUBLESHOOTING

If something is not working, and it's not clear from the error messages what is wrong, try using the `--debug=level` option to turn on debugging messages. This gives a great deal more information. Valid debug levels are 'none', and 'notice', 'info', 'warning' and 'critical' in order of increasing severity. The lower the severity level, the more output.

If you are still having trouble, the developers may be contacted on the mailing list:

```

Debian buildd-tools Developers
<buildd-tools-devel@lists.alioth.debian.org>

```

## BUGS

On the **mips** and **mipsel** architectures, Linux kernels up to and including at least version 2.6.17 have broken **personality**(2) support, which results in a failure to set the personality. This will be seen as an “Operation not permitted” (EPERM) error. To work around this problem, set *personality* to ‘undefined’, or upgrade to a more recent kernel.

## ENVIRONMENT

By default, the environment is not preserved, and the following environment variables are defined: HOME, LOGNAME, PATH, SHELL, TERM (preserved if already defined), and USER. The environment variables SCHROOT\_COMMAND, SCHROOT\_USER, SCHROOT\_GROUP, SCHROOT\_UID and SCHROOT\_GID are set inside the chroot specifying the command being run, the user name, group name, user ID and group ID, respectively.

The following, potentially dangerous, environment variables are removed for safety by default: BASH\_ENV, CDPATH, ENV, HOSTALIASES, IFS, KRB5\_CONFIG, KRB5\_CONFDIR, KRBTKFILE, KRB\_CONF, LD\_.\*, LOCALDOMAIN, NLSPATH, PATH\_LOCALE, RES\_OPTIONS, TERMINFO, TERMINFO\_DIRS, and TERMPATH. If desired, the *environment-filter* configuration key will allow the exclusion list to be modified; see **schroot.conf**(5) for further details.

## FILES

### Configuration files

*/usr/local/etc/schroot/schroot.conf*

The system-wide chroot definition file. This file must be owned by the root user, and not be writable by other.

*/usr/local/etc/schroot/chroot.d*

Additional chroot definitions may be placed in files under this directory. They are treated in exactly that same manner as */usr/local/etc/schroot/schroot.conf*. Each file may contain one or more chroot definitions. Note that the files in this directory follow the same naming rules as **run-parts**(8) when run with the *--lsbsysinit* option.

*/usr/local/etc/schroot/setup.d*

The system-wide chroot setup script directories. See **schroot-setup**(5).

*/etc/pam.d/schroot*

PAM configuration.

### System directories

*/usr/local/libexec/schroot*

Directory containing helper programs used by setup scripts.

### Session directories

Each directory contains a directory or file with the name of each session. Not all chroot types make use of all the following directories.

*/usr/local/var/lib/schroot/session*

Directory containing the session configuration for each active session.

*/usr/local/var/lib/schroot/mount*

Directory used to mount the filesystems used by each active session.

*/usr/local/var/lib/schroot/union/underlay*

Directory used for filesystem union source (underlay).

*/usr/local/var/lib/schroot/union/overlay*

Directory used for filesystem union writeable overlay.

*/usr/local/var/lib/schroot/unpack*

Directory used for unpacking file chroots.

## AUTHORS

Roger Leigh.

**COPYRIGHT**

Copyright © 2005–2011 Roger Leigh <rleigh@debian.org>

**schroot** is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

**SEE ALSO**

**dchroot(1)**, **sbuid(1)**, **chroot(2)**, **run-parts(8)**, **schroot-setup(5)**, **schroot-faq(7)**, **schroot.conf(5)**.

## NAME

schroot.conf – chroot definition file for schroot

## DESCRIPTION

*schroot.conf* is a plain UTF-8 text file, describing the chroots available for use with schroot.

Comments are introduced following a '#' ("hash") character at the beginning of a line, or following any other text. All text right of the '#' is treated as a comment.

The configuration format is an INI-style format, split into groups of key-value pairs separated by section names in square brackets.

### General options

A chroot is defined as a group of key-value pairs, which is started by a name in square brackets on a line by itself. The file may contain multiple groups which therefore define multiple chroots.

A chroot definition is started by the name of the chroot in square brackets. For example,

```
[sid]
```

The name is subject to certain naming restrictions. For further details, see the section "*Chroot Names*" below.

This is then followed by several key-value pairs, one per line:

***type***=*type*

The type of the chroot. Valid types are 'plain', 'directory', 'file', 'loopback', 'block-device', 'btrfs-snapshot' and 'lvm-snapshot'. If empty or omitted, the default type is 'plain'. Note that 'plain' chroots do not run setup scripts and mount filesystems; 'directory' is recommended for normal use (see "*Plain and directory chroots*", below).

***description***=*description*

A short description of the chroot. This may be localised for different languages; see the section "*Localisation*" below.

***priority***=*number*

Set the priority of a chroot. *number* is a positive integer indicating whether a distribution is older than another. For example, "oldstable" and "oldstable-security" might be '0', while "stable" and "stable-security" are '1', "testing" is '2' and "unstable" is '3'. The values are not important, but the difference between them is. This option is deprecated and no longer used by schroot, but is still permitted to be used; it will be obsoleted and removed in a future release.

***message-verbosity***=*verbosity*

Set the verbosity of messages printed by schroot when setting up, running commands and cleaning up the chroot. Valid settings are 'quiet' (suppress most messages), 'normal' (the default) and 'verbose' (show all messages). This setting is overridden by the options *--quiet* and *--verbose*.

***users***=*user1,user2,...*

A comma-separated list of users which are allowed access to the chroot. If empty or omitted, no users will be allowed access (unless a group they belong to is also specified in *groups*).

***groups***=*group1,group2,...*

A comma-separated list of groups which are allowed access to the chroot. If empty or omitted, no groups of users will be allowed access.

***root-users***=*user1,user2,...*

A comma-separated list of users which are allowed **password-less** root access to the chroot. If empty or omitted, no users will be allowed root access without a password (but if a user or a group they belong to is in *users* or *groups*, respectively, they may gain access with a password). See the section "*Security*" below.

***root-groups***=*group1,group2,...*

A comma-separated list of groups which are allowed **password-less** root access to the chroot. If empty or omitted, no users will be allowed root access without a password (but if a user or a group

they belong to is in *users* or *groups*, respectively, they may gain access with a password). See the section “*Security*” below.

**aliases**=*alias1,alias2,...*

A comma-separated list of aliases (alternate names) for this chroot. For example, a chroot named “sid” might have an ‘unstable’ alias for convenience. Aliases are subject to the same naming restrictions as the chroot name itself.

**run-setup-scripts**=*true|false*

Set whether chroot setup scripts will be run. The default is to run setup scripts for all chroot types except ‘plain’. Setup scripts are **required** to mount and configure the chroot environment. This option is deprecated and no longer used by schroot, but is still permitted to be used; it will be obsoleted and removed in a future release.

**run-exec-scripts**=*true|false*

Set whether chroot execution scripts will be run. The default is the same as the default for the *run-setup-scripts* key. This option was called *run-session-scripts* in versions prior to 0.2.5. This option is deprecated and no longer used by schroot, but is still permitted to be used; it will be obsoleted and removed in a future release.

**script-config**=*filename*

The behaviour of the chroot setup scripts may be customised on a per-chroot basis by providing a shell script which the scripts will source. The filename is relative to *@SCHROOT\_SYSCONF\_DIR@*. The default filename is ‘default/config’. Alternatives are ‘minimal/config’ (minimal configuration), ‘desktop/config’ (for running desktop applications in the chroot, making more functionality from the host system available in the chroot) and ‘sbuid/config’ (for using the chroot for Debian package building).

Desktop users should note that the *fstab* file *desktop/fstab* will need editing if you use gdm3. The *preserve-environment* key should also be set to ‘true’ so that the environment is preserved inside the chroot.

If none of the configuration profiles provided above meet your needs, then they may be edited to further customise them, and/or copied and used as a template for entirely new profiles. Settings for specific chroots may also be set in a single script by using conditionals checking the chroot name and/or type. Note that the script will be sourced once for each and every script invocation, and must be idempotent. The file format is documented in **schroot-script-config(5)**.

Note that the different profiles have different security implications; see the section “*Security*” below for further details.

**command-prefix**=*command,option1,option2,...*

A comma-separated list of a command and the options for the command. This command and its options will be prefixed to all commands run inside the chroot. This is useful for adding commands such as *nice*, *ionice* or *eatmydata* for all commands run inside the chroot. *nice* and *ionice* will affect CPU and I/O scheduling. *eatmydata* ingores filesystem *fsync* operations, and is useful for throwaway snapshot chroots where you don’t care about dataloss, but do care about high speed.

**personality**=*persona*

Set the personality (process execution domain) to use. This option is useful when using a 32-bit chroot on 64-bit system, for example. Valid options on Linux are ‘bsd’, ‘hpux’, ‘irix32’, ‘irix64’, ‘irixn32’, ‘iscr4’, ‘linux’, ‘linux32’, ‘linux\_32bit’, ‘osf4’, ‘osr5’, ‘riscos’, ‘scorvr3’, ‘solaris’, ‘sunos’, ‘svr4’, ‘uw7’, ‘wysev386’, and ‘xenix’. The default value is ‘linux’. There is also the special option ‘undefined’ (personality not set). For a 32-bit chroot on a 64-bit system, ‘linux32’ is the option required. The only valid option for non-Linux systems is ‘undefined’. The default value for non-Linux systems is ‘undefined’.

**preserve-environment**=*true|false*

By default, the environment will not be preserved inside the chroot, instead a minimal environment will be used. Set to *true* to always preserve the environment. This is useful for example

when running X applications inside the chroot, which need the environment to function correctly. The environment may also be preserved using the `--preserve-environment` option.

***environment-filter=regex***

The environment to be set in the chroot will be filtered in order to remove environment variables which may pose a security risk. Any environment variable matching the specified POSIX extended regular expression will be removed prior to executing any command in the chroot.

Potentially dangerous environment variables are removed for safety by default using the following regular expression: “`^(BASH_ENV|CDPATH|ENV|HOSTALIASES|IFS|KRB5_CONFIG|KRBCONFDIR|KRBTKFILE|KRB_CONF|LD_*|LOCALDOMAIN|NLSPATH|PATH_LOCALE|RES_OPTIONS|TERMINFO|TERMINFO_DIRS|TERMPATH)$`”.

### Plain and directory chroots

Chroots of type ‘plain’ or ‘directory’ are directories accessible in the filesystem. The two types are equivalent except for the fact that *directory* chroots run setup scripts, whereas plain chroots do not. In consequence, filesystems such as */proc* are not mounted in plain chroots; it is the responsibility of the system administrator to configure such chroots by hand, whereas *directory* chroots are automatically configured. Additionally, *directory* chroots implement the **filesystem union chroot** options (see “*Filesystem Union chroot options*”, below).

These chroot types have an additional (mandatory) configuration option:

***directory=directory***

The directory containing the chroot environment. This is where the root will be changed to when executing a login shell or a command. The directory must exist and have read and execute permissions to allow users access to it. Note that on Linux systems it will be bind-mounted elsewhere for use as a chroot; the directory for ‘plain’ chroots is mounted with the `--rbind` option to **mount(8)**, while for ‘directory’ chroots `--bind` is used instead so that sub-mounts are not preserved (they should be set in the *fstab* file just like in */etc/fstab* on the host).

This option was previously named *location*, but was renamed to avoid ambiguity with the option by the same name for **mountable chroot** options (see “*Mountable chroot options*”, below). The name *location* is deprecated, but still valid; it will be obsoleted and removed in a future release. It is recommended to use *directory* rather than *location*. Note that it is an error to use both *directory* and *location* at the same time.

### File chroots

Chroots of type ‘file’ are files on the current filesystem containing an archive of the chroot files. They implement the **source chroot** options (see “*Source chroot options*”, below). Note that a corresponding source chroot (of type ‘file’) will be created for each chroot of this type; this is for convenient access to the source archive, e.g. for the purpose of updating. These additional options are also implemented:

***file=filename***

The file containing the archived chroot environment (mandatory). This must be a tar (tape archive), optionally compressed with gzip or bzip2. The file extensions used to determine the type are *.tar*, *.tar.gz*, *.tar.bz2*, *.tgz*, and *.tbz*. This file must be owned by the root user, and not be writable by other. Note that zip archives are no longer supported; zip was not able to archive named pipes and device nodes, so was not suitable for archiving chroots.

***location=path***

This is the path to the chroot *inside* the archive. For example, if the archive contains a chroot in */squeeze*, you would specify “*/squeeze*” here. If the chroot is the only thing in the archive, i.e. */* is the root filesystem for the chroot, this option should be left blank, or omitted entirely.

### Loopback chroots

Chroots of type ‘loopback’ are a filesystem available as a file on disk, accessed via a loopback mount. The file will be loopback mounted and unmounted on demand. Loopback chroots implement the **mountable chroot** and **filesystem union chroot** options (see “*Mountable chroot options*” and “*Filesystem Union chroot options*”, below), plus an additional option:

**file=filename**

This is the filename of the file containing the filesystem, including the absolute path. For example “/srv/chroot/sid”.

### Block device chroots

Chroots of type ‘block-device’ are a filesystem available on an unmounted block device. The device will be mounted and unmounted on demand. Block device chroots implement the **mountable chroot** and **filesystem union chroot** options (see “*Mountable chroot options*” and “*Filesystem Union chroot options*”, below), plus an additional option:

**device=device**

This is the device name of the block device, including the absolute path. For example, “/dev/sda5”.

### Btrfs snapshot chroots

Chroots of type ‘btrfs-snapshot’ are a Btrfs snapshot created from an existing Btrfs subvolume on a mounted Btrfs filesystem. A snapshot will be created from this source subvolume on demand at the start of a session, and then the snapshot will be mounted. At the end of the session, the snapshot will be unmounted and deleted. This chroot type implements the **source chroot** options (see “*Source chroot options*”, below). Note that a corresponding source chroot (of type ‘directory’) will be created for each chroot of this type; this is for convenient access to the source volume. These additional options are also implemented:

**btrfs-source-subvolume=directory**

The directory containing the source subvolume.

**btrfs-snapshot-directory=directory**

The directory in which to store the snapshots of the above source subvolume.

### LVM snapshot chroots

Chroots of type ‘lvm-snapshot’ are a filesystem available on an LVM logical volume (LV). A snapshot LV will be created from this LV on demand, and then the snapshot will be mounted. At the end of the session, the snapshot LV will be unmounted and removed.

LVM snapshot chroots implement the **source chroot** options (see “*Source chroot options*”, below), and all the options for ‘block-device’. Note that a corresponding source chroot (of type ‘block-device’) will be created for each chroot of this type; this is for convenient access to the source device. This additional option is also implemented:

**lvm-snapshot-options=snapshot\_options**

Snapshot options. These are additional options to pass to `lvcreate(8)`. For example, “-L 2g” to create a snapshot 2 GiB in size. **Note:** the LV name (`-n`), the snapshot option (`-s`) and the original LV path may not be specified here; they are set automatically by `schroot`.

### Source chroot options

The ‘btrfs-snapshot’, ‘file’ and ‘lvm-snapshot’ chroot types implement source chroots. Additionally, chroot types with union support enabled implement source chroots (see “*Filesystem Union chroot options*”, below). These are chroots which automatically create a copy of themselves before use, and are usually session managed. These chroots additionally provide an extra chroot in the `source:` namespace, to allow convenient access to the original (non-snapshotted) data, and to aid in chroot maintenance. I.e. for a chroot named `wheezy` (`chroot:wheezy`), a corresponding `source:wheezy` chroot will be created. For compatibility with older versions of `schroot` which did not support namespaces, a chroot with a `-source` suffix appended to the chroot name will be created in addition (i.e. `wheezy-source` using the above example). Note that these compatibility names will be removed in `schroot 1.5.0`, so the use of the `source:` namespace is preferred over the use of the `-source` suffix form. See `schroot(1)` for further details.

These chroots provide the following additional options:

**source-clone=true|false**

Set whether the source chroot should be automatically cloned (created) for this chroot. The default is `true` to automatically clone, but if desired may be disabled by setting to `false`. If

disabled, the source chroot will be inaccessible.

**source-users**=*user1,user2,...*

A comma-separated list of users which are allowed access to the source chroot. If empty or omitted, no users will be allowed access. This will become the *users* option in the source chroot.

**source-groups**=*group1,group2,...*

A comma-separated list of groups which are allowed access to the source chroot. If empty or omitted, no users will be allowed access. This will become the *groups* option in the source chroot.

**source-root-users**=*user1,user2,...*

A comma-separated list of users which are allowed **password-less** root access to the source chroot. If empty or omitted, no users will be allowed root access without a password (but if a user is in *users*, they may gain access with a password). This will become the *root-users* option in the source chroot. See the section “*Security*” below.

**source-root-groups**=*group1,group2,...*

A comma-separated list of groups which are allowed **password-less** root access to the source chroot. If empty or omitted, no users will be allowed root access without a password (but if a user’s group is in *groups*, they may gain access with a password). This will become the *root-groups* option in the source chroot. See the section “*Security*” below.

### Mountable chroot options

The ‘block-device’, ‘loopback’ and ‘lvm-snapshot’ chroot types implement device mounting. These are chroots which require the mounting of a device in order to access the chroot. These chroots provide the following additional options:

**mount-options**=*options*

Mount options for the block device. These are additional options to pass to **mount(8)**. For example, “-o atime, sync, user\_xattr”.

**location**=*path*

This is the path to the chroot *inside* the filesystem on the device. For example, if the filesystem contains a chroot in */chroot/sid*, you would specify “/chroot/sid” here. If the chroot is the only thing on the filesystem, i.e. / is the root filesystem for the chroot, this option should be left blank, or omitted entirely.

### Filesystem Union chroot options

The ‘block-device’, ‘directory’ and ‘loopback’ chroot types allow for the creation of a session using filesystem unions to overlay the original filesystem with a separate writable directory. The original filesystem is read-only, with any modifications made to the filesystem made in the overlying writable directory, leaving the original filesystem unchanged. A union permits multiple sessions to access and make changes to a single chroot simultaneously, while keeping the changes private to each session. To enable this feature, set *union-type* to any supported value. If enabled, the chroot will also be a **source chroot**, which will provide additional options (see “*Source chroot options*”, above). All entries are optional.

**union-type**=*type*

Set the union filesystem type. Currently supported filesystems are ‘aufs’, ‘overlayfs’ and ‘unionfs’. The default is ‘none’, which disables this feature.

**union-mount-options**=*options*

Union filesystem mount options (branch configuration), used for mounting the union filesystem specified with *union-type*. This replaces the complete “-o” string for mount and allows for the creation of complex filesystem unions. Note that ‘aufs’, ‘overlayfs’ and ‘unionfs’ each have different supported mount options. **Note:** One can use the variables “\${CHROOT\_UNION\_OVERLAY\_DIRECTORY}” and “\${CHROOT\_UNION\_UNDERLAY\_DIRECTORY}” to refer to the writable overlay session directory and read-only underlying directory which are to form the union. See **schroot-setup(5)** for a complete variable list.

***union-overlay-directory=directory***

Specify the directory where the writeable overlay session directories will be created. The default is '@SCHROOT\_OVERLAY\_DIR@'.

***union-underlay-directory=directory***

Specify the directory where the read-only underlying directories will be created. The default is '@SCHROOT\_UNDERLAY\_DIR@'.

### Localisation

Some keys may be localised in multiple languages. This is achieved by adding the locale name in square brackets after the key name. For example:

```
description[en_GB]=British English translation
```

This will localise the *description* key for the en\_GB locale.

```
description[fr]=French translation
```

This will localise the *description* key for all French locales.

### CHROOT NAMES

A number of characters or words are not permitted in a chroot name, session name or configuration filename. The name may not contain a leading period ('.'). The characters ':' (colon), ',' (comma) and '/' (forward slash) are not permitted anywhere in the name. The name may also not contain a trailing tilde ('~'). The rationale for these restrictions is given below.

- '.' A leading period could be used to create a name with a relative path in it, in combination with '/', and this could allow overwriting of files on the host filesystem. Not allowing this character also means hidden files cannot be created. It also means some editor backups are automatically ignored. Periods are allowed anywhere else in the name.
- ':' A colon is used as a namespace delimiter, and so is not permitted as part of a chroot or session name. LVM snapshot names may also not contain this character due to a naming restriction by **lvcreate(8)**.
- '/' Names containing this character are not valid filenames. A forward slash would potentially allow creation of files in subdirectories.
- ',' Commas are used to separate items in lists. Aliases are separated by commas and hence can't contain commas in their name.
- '~' Filenames containing trailing tildes are used for editor backup files, which are ignored. Tildes are allowed anywhere else in the name.

'dpkg-old'

'dpkg-dist'

'dpkg-new'

'dpkg-tmp'

These names may not appear at the end of a name. These are saved copies of conffiles used by the dpkg package manager, and will be ignored.

### SECURITY

#### Untrusted users

Note that giving untrusted users root access to chroots is a **serious security risk!** Although the untrusted user will only have root access to files inside the chroot, in practice there are many obvious ways of breaking out of the chroot and of disrupting services on the host system. As always, this boils down to *trust*.

**Do not give chroot root access to users you would not trust with root access to the host system.**

#### Profiles

Depending upon which profile you have configured with the *script-config* option, different filesystems will be mounted inside the chroot, and different files will be copied into the chroot from the host. Some profiles will mount the host's */dev*, while others will not. Some profiles also bind mount additional parts of the host filesystem in order to allow use of certain features, including user's home directories and specific parts of */var*. Check the profile's *fstab* file to be certain of what will be mounted, and the other

profile files to see which files and system databases will be copied into the chroot. Choose a different profile or edit the files to further restrict what is made available inside the chroot.

There is a tradeoff between security (keeping the chroot as minimal as possible) and usability (which sometimes requires access to parts of the host filesystem). The different profiles make different tradeoffs, and it is important that you assess which meets the security/usability tradeoff you require.

### EXAMPLE

```
# Sample configuration

[sid]
type=plain
description=Debian unstable
description[fr_FR]=Debian instable
directory=/srv/chroot/sid
priority=3
users=jim
groups=sbuild
root-users=rleigh
aliases=unstable,default

[etch]
type=block-device
description=Debian testing (32-bit)
priority=2
groups=users
#groups=sbuild-security
aliases=testing
device=/dev/hda_vg/etch_chroot
mount-options=-o atime
personality=linux32

[sid-file]
type=file
description=Debian sid file-based chroot
priority=3
groups=sbuild
file=/srv/chroots/sid.tar.gz

[sid-snapshot]
type=lvm-snapshot
description=Debian unstable LVM snapshot
priority=3
groups=sbuild
users=rleigh
source-root-users=rleigh
source-root-groups=admin
device=/dev/hda_vg/sid_chroot
mount-options=-o atime, sync, user_xattr
lvm-snapshot-options=--size 2G
```

### FILES

#### Chroot definitions

**@SCHROOT\_CONF@**

The system-wide chroot definition file. This file must be owned by the root user, and not be writable by other.

**@SCHROOT\_CONF\_CHROOT\_D@**

Additional chroot definitions may be placed in files under this directory. They are treated in exactly that same manner as @SCHROOT\_CONF@. Each file may contain one or more chroot definitions.

**Setup script configuration**

The directory @SCHROOT\_SYSCONF\_DIR@/default contains the default settings used by setup scripts.

**config** Main configuration file read by setup scripts. The format of this file is described in **schroot-script-config(5)**. This is the default value for the *script-config* key. Note that this was formerly named @SCHROOT\_SYSCONF\_DIR@/script-defaults. The following files are referenced by default:

**copyfiles**

A list of files to copy into the chroot from the host system. Note that this was formerly named @SCHROOT\_SYSCONF\_DIR@/copyfiles-defaults.

**fstab**

A file in the format described in **fstab(5)**, used to mount filesystems inside the chroot. The mount location is relative to the root of the chroot. Note that this was formerly named @SCHROOT\_SYSCONF\_DIR@/mount-defaults.

**nssdatabases**

System databases (as described in */etc/nsswitch.conf* on GNU/Linux systems) to copy into the chroot from the host. Note that this was formerly named @SCHROOT\_SYSCONF\_DIR@/nss-databases-defaults.

**AUTHORS**

Roger Leigh.

**COPYRIGHT**

Copyright © 2005–2011 Roger Leigh <rleigh@debian.org>

**schroot** is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

**SEE ALSO**

**sbuid(1)**, **schroot(1)**, **schroot-script-config(5)**, **schroot-faq(7)**, **mount(8)**.

## NAME

schroot-setup – schroot chroot setup scripts

## DESCRIPTION

**schroot** uses scripts to set up and then clean up the chroot environment. The directory */usr/local/etc/schroot/setup.d* contains scripts run when a chroot is created and destroyed. Several environment variables are set while the scripts are being run, which allows their behaviour to be customised, depending upon, for example, the type of chroot in use.

The scripts are run in name order, like those run by **init**(8), by using the same style of execution as **run-parts**(8).

The setup scripts are all invoked with two options:

- 1 The action to perform.

When a session is first started, the chroot is set up by running the scripts in */usr/local/etc/schroot/setup.d* with the ‘setup-start’ option. When the session is ended, the scripts in */usr/local/etc/schroot/setup.d* are run in reverse order with the ‘setup-stop’ option.

- 2 The chroot status.

This is either ‘ok’ if there are no problems, or ‘fail’ if something went wrong. For example, particular actions may be skipped on failure.

Note that the scripts should be *idempotent*. They **must** be idempotent during the ‘setup-stop’ phase, because they may be run more than once, for example on failure.

## ENVIRONMENT

### General variables

**AUTH\_USER**

The username of the user the command in the chroot will run as.

**CHROOT\_NAME**

The chroot name. Note that this is the name of the original chroot before session creation; you probably want **SESSION\_ID**.

**HOST**

**HOST\_OS**

**HOST\_VENDOR**

**HOST\_CPU**

The host system architecture schroot is running upon. This may be used to introduce architecture-specific behaviour into the setup scripts where required. **HOST** is the GNU triplet for the architecture, while **HOST\_OS**, **HOST\_VENDOR** and **HOST\_CPU** are the component parts of the triplet.

**LIBEXEC\_DIR**

The directory under which helper programs are located.

**MOUNT\_DIR**

The directory under which non-filesystem chroots are mounted (e.g. block devices and LVM snapshots).

**PID** The process ID of the schroot process.

**PLATFORM**

The operating system platform schroot is running upon. This may be used to introduce platform-specific behaviour into the setup scripts where required. Note that the **HOST** variables are probably what are required. In the context of schroot, the platform is the supported configuration and behaviour for a given architecture, and may be identical between different architectures.

**SESSION\_ID**

The session identifier.

**VERBOSE**

Set to 'quiet' if only error messages should be printed, 'normal' if other messages may be printed as well, and 'verbose' if all messages may be printed. Previously called AUTH\_VERBOSITY.

**CHROOT\_SESSION\_CREATE**

Set to 'true' if a session will be created, otherwise 'false'.

**CHROOT\_SESSION\_CLONE**

Set to 'true' if a session will be cloned, otherwise 'false'.

**CHROOT\_SESSION\_PURGE**

Set to 'true' if a session will be purged, otherwise 'false'.

**CHROOT\_TYPE**

The type of the chroot. This is useful for restricting a setup task to particular types of chroot (e.g. only block devices or LVM snapshots).

**CHROOT\_NAME**

The name of the chroot. This is useful for restricting a setup task to a particular chroot, or set of chroots.

**CHROOT\_DESCRIPTION**

The description of the chroot.

**CHROOT\_MOUNT\_LOCATION**

The location to mount the chroot. It is used for mount point creation and mounting.

**CHROOT\_LOCATION**

The location of the chroot inside the mount point. This is to allow multiple chroots on a single filesystem. Set for all mountable chroot types.

**CHROOT\_PATH**

The absolute path to the chroot. This is typically CHROOT\_MOUNT\_LOCATION and CHROOT\_LOCATION concatenated together. This is the path which should be used to access the chroots.

**Plain and directory chroot variables**

These chroot types use only general variables.

**File variables****CHROOT\_FILE**

The file containing the chroot files.

**CHROOT\_FILE\_REPACK**

Set to 'true' to repack the chroot into an archive file on ending a session, otherwise 'false'.

**Mountable chroot variables**

These variables are only set for directly mountable chroot types.

**CHROOT\_MOUNT\_DEVICE**

The device to mount containing the chroot. mounting.

**CHROOT\_MOUNT\_OPTIONS**

Options to pass to **mount(8)**.

**CHROOT\_LOCATION**

The location of the chroot inside the mount point. This allows the existence of multiple chroots on a single filesystem.

**Filesystem union variables****CHROOT\_UNION\_TYPE**

Union filesystem type.

**CHROOT\_UNION\_MOUNT\_OPTIONS**

Union filesystem mount options.

**CHROOT\_UNION\_OVERLAY\_DIRECTORY**

Union filesystem overlay directory (writable).

**CHROOT\_UNION\_UNDERLAY\_DIRECTORY**

Union filesystem underlay directory (read-only).

**Block device variables****CHROOT\_DEVICE**

The device containing the chroot root filesystem. This is usually, but not necessarily, the device which will be mounted. For example, an LVM snapshot this will be the original logical volume.

**LVM snapshot variables****CHROOT\_LVM\_SNAPSHOT\_NAME**

Snapshot name to pass to **lvcreate(8)**.

**CHROOT\_LVM\_SNAPSHOT\_DEVICE**

The name of the LVM snapshot device.

**CHROOT\_LVM\_SNAPSHOT\_OPTIONS**

Options to pass to **lvcreate(8)**.

**FILES****Setup script configuration**

The directory */usr/local/etc/schroot/default* contains the default settings used by setup scripts.

**config** Main configuration file read by setup scripts. The format of this file is described in **schroot-script-config(5)**. This is the default value for the *script-config* key. Note that this was formerly named */usr/local/etc/schroot/script-defaults*. The following files are referenced by default:

**copyfiles**

A list of files to copy into the chroot from the host system. Note that this was formerly named */usr/local/etc/schroot/copyfiles-defaults*.

**fstab**

A file in the format described in **fstab(5)**, used to mount filesystems inside the chroot. The mount location is relative to the root of the chroot. Note that this was formerly named */usr/local/etc/schroot/mount-defaults*.

**nssdatabases**

System databases (as described in */etc/nsswitch.conf* on GNU/Linux systems) to copy into the chroot from the host. Note that this was formerly named */usr/local/etc/schroot/nss-databases-defaults*.

**Setup scripts**

The directory */usr/local/etc/schroot/setup.d* contains the chroot setup scripts.

**00check**

Print debugging diagnostics and perform basic sanity checking.

**05file**

Unpack, clean up, and repack file-based chroots.

**05fsunion**

Create and remove union filesystems.

**05lvm**

Create and remove LVM snapshots.

**10mount**

Mount and unmount filesystems.

**15killprocs**

Kill processes still running inside the chroot when ending a session, which would prevent unmounting of filesystems and cleanup of any other resources.

**20copyfiles**

Copy files from the host system into the chroot. Configure networking by copying *hosts* and *resolv.conf*, for example.

**20nssdatabases**

Configure system databases by copying *passwd*, *shadow*, *group* etc. into the chroot.

**50chrootname**

Set the chroot name (*/etc/debian\_chroot*) in the chroot. This may be used by the shell prompt to display the current chroot.

**AUTHORS**

Roger Leigh.

**COPYRIGHT**

Copyright © 2005–2011 Roger Leigh <rleigh@debian.org>

**schroot** is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

**SEE ALSO**

**schroot(1)**, **fstab(5)**, **schroot.conf(5)**, **schroot-script-config(5)**, **run-parts(8)**.

**NAME**

schroot-script-config – schroot chroot setup script configuration

**DESCRIPTION**

**schroot** uses scripts to set up and then clean up the chroot environment. These scripts may be customised using the *script-config* key in */usr/local/etc/schroot/schroot.conf*. This key specifies a file which the setup scripts will source when they are run. The file is a Bourne shell script, and in consequence may contain any valid shell code, in addition to simple variable assignments. This will, for example, allow behaviour to be customised according to the specific chroot type or name.

**ENVIRONMENT**

The environment is the same as for all setup scripts, described in **schroot-setup(5)**.

**VARIABLES**

The following variables may be set to configure setup script behaviour. Note that new variables may be added in future releases. Third-party extensions to schroot which add their own setup scripts may add additional variables which are not documented here; consult the extension documentation for further details.

**COPYFILES**

A file containing a list of files to copy into the chroot (one file per line). The file will have the same absolute location inside the chroot.

**FSTAB** The filesystem table file to be used to mount filesystems within the chroot. The format of this file is the same as for */etc/fstab*, documented in **fstab(5)**. The only difference is that the mountpoint path *fs\_dir* is relative to the chroot, rather than the root.

**NSSDATABASES**

A file listing the system databases to copy into the chroot. The default databases are 'passwd', 'shadow', 'group', 'services', 'protocols', 'networks', and 'hosts'. 'gshadow' is not yet copied by default, due to not being supported by all but the most recent version of the GNU C library. The databases are copied using **getent(1)** so all database sources listed in */etc/nsswitch.conf* will be used for each database.

**AUTHORS**

Roger Leigh.

**COPYRIGHT**

Copyright © 2005–2011 Roger Leigh <rleigh@debian.org>

**schroot** is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

**SEE ALSO**

**sbuid(1)**, **schroot(1)**, **sh(1)**, **schroot.conf(5)**, **schroot-setup(5)**.

## NAME

schroot – frequently asked questions

## DESCRIPTION

This manual page covers various frequently asked questions about configuration and usage of schroot.

## CONFIGURATION

### Why is schroot overwriting configuration files in the chroot?

By default, schroot copies over the system NSS databases ('passwd', 'shadow', 'group', 'gshadow', 'services', 'protocols', 'networks', and 'hosts', etc.) into the chroot. The reason for this is that the chroot environment is not a completely separate system, and it copying them over keeps them synchronised. However, this is not always desirable, particularly if installing a package in the chroot creates system users and groups which are not present on the host, since these will disappear next time the databases are copied over.

The suggested workaround here is to disable the copying by removing or commenting out the databases in the NSSDATABASES file in the script-config file for the chroot. These are `/usr/local/etc/schroot/default/nssdatabases` and `/usr/local/etc/schroot/default/config` by default.

In the future, we will be working on a better scheme for keeping the host and chroot databases in sync which can merge entries rather than overwriting the entire database, which would preserve chroot-specific changes.

### Should I use the plain or directory chroot type?

These two chroot types are basically equivalent, since they are both just directories in the filesystem. `plain` is very simple and does not perform any setup tasks; the only reason you would want to use it is if you're upgrading from a program such as `dchroot(1)` or `chroot(8)` which don't do anything other than running a command or shell in a directory. On the other hand, directory chroots do run setup scripts, which can mount additional filesystems and do other setup tasks.

## ADVANCED CONFIGURATION

### What are snapshots and unions?

Some chroot types support *cloning*. This means when you start a session, you get a *copy* of the chroot which lasts just for the lifetime of the session. This is useful when you want a temporary clean copy of a system for a single task, which is then automatically deleted when you're done with it. For example, the Debian package build daemons run `sbuid(1)` to build Debian packages, and this program uses schroot to create a clean build environment for each package. Without snapshotting, the chroot would need to be reset to its initial state at the end of each build to make it ready for the next one, and any debris left over from package removals or earlier builds could interfere with the next build.

The most commonly-used snapshotting method is to use LVM snapshots (chroot type 'lvm-snapshot'). In this case the chroot must exist on an LVM logical volume (LV); snapshots of an LV may then be made with `lvcreate(8)` during chroot session setup. However, these use up a lot of disk space. A newer method is to use Btrfs snapshots which use up much less disk space (chroot type 'btrfs-snapshot'), and may be more reliable than LVM snapshots. Btrfs is however still experimental, but it is hoped that it will become the recommended method as it matures.

Unions are an alternative to snapshots. In this situation, instead of creating a copy of the chroot filesystem, we overlay a read-write temporary filesystem on top of the chroot filesystem so that any modifications are stored in the overlay, leaving the original chroot filesystem untouched. The Linux kernel has yet to integrate support for union filesystems such as aufs and unionfs, so LVM snapshots are still the recommended method at present.

## USAGE

### Can I run a daemons in a chroot?

A common problem is trying to run a daemon in a chroot, and finding that this doesn't work. Typically, the daemon is killed shortly after it starts up.

When schroot runs, it begins a session, runs the specified command or shell, waits for the command or shell to exit, and then it ends the session. For a normal command or shell, this works just fine. However, daemons normally start up by running in the background and detaching from the controlling terminal. They

do this by forking twice and letting the parent processes exit. Unfortunately, this means schroot detects that the program exited (the daemon is a orphaned grandchild of this process) and it then ends the session. Part of ending the session is killing all processes running inside the chroot, which means the daemon is killed as the session ends.

In consequence, it's not possible to run a daemon *directly* with schroot. You can however do it if you create a session with `--begin-session` and then run the daemon with `--run-session`. It's your responsibility to end the session with `--end-session` when the daemon has terminated or you no longer need it.

### How do I manually cleaning up a broken session?

Occasionally, it may be necessary to manually clean up sessions. If something changes on your system which causes the setup scripts to fail when ending a session, for example removal of a needed file or directory, it may not be possible for schroot to clean everything up automatically. For each of the session directories listed in the “*Session directories*” section in **schroot(1)**, any files with the name of the session ID need deleting, and any directories with the name of the session ID need unmounting (if there are any filesystems mounted under it), and then also removing.

For example, to remove a session named *my-session* by hand:

- Remove the session configuration file  

```
% rm /usr/local/var/lib/schroot/session/my-session.␣
```
- Check for mounted filesystems  

```
% /usr/local/libexec/schroot/schroot-list-mounts -m /usr/local/var/lib/schroot/mount/my-session.␣
```
- Unmount any mounted filesystems
- Remove `/usr/local/var/lib/schroot/mount/my-session`
- Repeat for the other directories such as `/usr/local/var/lib/schroot/union/underlay`, `/usr/local/var/lib/schroot/union/overlay` and `/usr/local/var/lib/schroot/unpack`

**NOTE:** Do not remove any directories without checking if there are any filesystems mounted below them, since filesystems such as `/home` could still be bind mounted. Doing so could cause irretrievable data loss!

## ADVANCED USAGE

### How do I use sessions?

In normal use, running a command might look like this:

```
% schroot -c squeeze -- command.␣
```

which would run the command *command* in the *squeeze* chroot. While it's not apparent that a session is being used here, schroot is actually doing the following steps:

- Creating a session using the *squeeze* chroot. This will be automatically given a unique name, such as `squeeze-57a69547-e014-4f5d-a98b-f4f35a005307`, though you don't usually need to know about this
- Setup scripts are run to create the session chroot and configure it for you
- The command *command* is run inside the session chroot
- Setup scripts are run to clean up the session chroot
- The session is deleted

Now, if you wanted to run more than one command, you could run a shell and run them interactively, or you could put them into shell script and run that instead. But you might want to do something in between, such as running arbitrary commands from a program or script where you don't know which commands to run in advance. You might also want to preserve the chroot state in between commands, where the normal automatic session creation would reset the state in between each command. This is what sessions are for: once created, the session is persistent and won't be automatically removed. With a session, you can run as many commands as you like, but you need to create and delete the session by hand since schroot can't know by itself when you're done with it unlike in the single command case above. This is quite easy:

```
% schroot --begin-session -c squeeze.␣
```

```
squeeze-57a69547-e014-4f5d-a98b-f4f35a005307
```

This created a new session based upon the *squeeze* chroot. The unique name for the session, the session ID, was printed to standard output, so we could also save it as a shell variable at the same time like so:

```
% SESSION=$(schroot --begin-session -c squeeze)↵
% echo $SESSION↵
squeeze-57a69547-e014-4f5d-a98b-f4f35a005307
```

Now we have created the session and got the session ID, we can run commands in it using the session ID:

```
% schroot --run-session -c squeeze-57a69547-e014-4f5d-a98b-f4f35a005307
-- command1↵
```

or

```
% schroot --run-session -c "$SESSION" -- command1↵
```

and then as many more commands as we like

```
% schroot --run-session -c "$SESSION" -- command2↵
% schroot --run-session -c "$SESSION" -- command3↵
% schroot --run-session -c "$SESSION" -- command4↵
```

etc.

When we are done with the session, we can remove it with *--end-session*:

```
% schroot --end-session -c
squeeze-57a69547-e014-4f5d-a98b-f4f35a005307↵
```

or

```
% schroot --end-session -c $SESSION↵
```

Since the automatically generated session names can be long and unwieldy, the *--session-name* option allows you to provide you own name:

```
% schroot --begin-session -c squeeze --session-name my-name↵
my-name
```

## CONTRIBUTING

### Getting help and getting involved

The mailing list <build-tools-devel@lists.alioth.debian.org> is used for both user support and development discussion. The list may be subscribed to from the project website at <https://alioth.debian.org/projects/build-tools/> or the Mailman list interface at <http://lists.alioth.debian.org/mailman/listinfo/build-tools-devel>.

### Reporting bugs

On Debian systems, bugs may be reported using the **reportbug(1)** tool, or alternatively by mailing <submit@bugs.debian.org> (see <http://bugs.debian.org> for details on how to do that).

### Getting the latest sources

schroot is maintained in the git version control system. You can get the latest sources from [git://git.debian.org/git/build-tools/schroot](https://git.debian.org/git/build-tools/schroot).

```
% git clone git://git.debian.org/git/build-tools/schroot↵
```

The master branch contains the current development release. Stable releases are found on branches, for example the 1.4 series of releases are on the schroot-1.4 branch.

## AUTHORS

Roger Leigh.

## COPYRIGHT

Copyright © 2005–2011 Roger Leigh <rleigh@debian.org>

**schroot** is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

**SEE ALSO**

**dchroot(1), schroot(1), sbuild(1), schroot-setup(5), schroot.conf(5).**

**NAME**

dchroot – enter a chroot environment

**SYNOPSIS**

**dchroot** [**-h**|**--help** | **-V**|**--version** | **-l**|**--list** | **-i**|**--info** | **--config** | **--location**] [**--directory=directory**] [**-d**|**--preserve-environment**] [**-q**|**--quiet** | **-v**|**--verbose**] [**-c chroot**|**--chroot=chroot** | **--all**] [**COMMAND** [ **ARG1** [ **ARG2** [ **ARGn**]]]]

**DESCRIPTION**

**dchroot** allows the user to run a command or a login shell in a chroot environment. If no command is specified, a login shell will be started in the user's home directory inside the chroot.

The command is one or more arguments which will be run in the user's default shell using its **-c** option. As a result, shell code may be embedded in this argument. If multiple command options are used, they are concatenated together, separated by spaces. Users should be aware of the shell quoting issues this presents, and should use **schroot** if necessary, which does not have any quoting issues.

The directory the command or login shell is run in depends upon the context. See **--directory** option below for a complete description.

This version of dchroot is a compatibility wrapper around the **schroot(1)** program. It is provided for backward compatibility with the dchroot command-line options, but schroot is recommended for future use. See the section "*Migration*" below for help migrating an existing dchroot configuration to schroot. See the section "*Incompatibilities*" below for known incompatibilities with older versions of dchroot.

If no chroot is specified, the chroot name or alias 'default' will be used as a fallback. If using the configuration in */usr/local/etc/dchroot.conf*, the first chroot in the file is the default.

**OPTIONS**

**dchroot** accepts the following options:

**Basic options**

**-h, --help**

Show help summary.

**-a, --all**

Select all chroots.

**-c, --chroot=chroot**

Specify a chroot to use. This option may be used multiple times to specify more than one chroot, in which case its effect is similar to **--all**.

**-l, --list**

List all available chroots.

**-i, --info**

Print detailed information about the specified chroots. Note that earlier versions of dchroot did not include this option.

**-p, --path**

Print location (path) of the specified chroots.

**--config**

Print configuration of the specified chroots. This is useful for testing that the configuration in use is the same as the configuration file. Any comments in the original file will be missing. Note that earlier versions of dchroot did not include this option.

**--directory=directory**

Change to *directory* inside the chroot before running the command or login shell. If *directory* is not available, dchroot will exit with an error status.

The default behaviour is as follows (all directory paths are inside the chroot). Unless the **--preserve-environment** option is used to preserve the environment, the login shell or command will run in the user's home directory, or / if the home directory is not available. When the

`--preserve-environment` option is used, it will attempt to use the current working directory, again falling back to `/` if it is not accessible. If none of the directories are available, `dchroot` will exit with an error status.

**-d, --preserve-environment**

Preserve the user's environment inside the `chroot` environment. The default is to use a clean environment; this option copies the entire user environment and sets it in the session.

**-q, --quiet**

Print only essential messages.

**-v, --verbose**

Print all messages. Note that earlier versions of `dchroot` did not include this option.

**-V, --version**

Print version information.

Note that earlier versions of `dchroot` did not provide long options.

## CONFIGURATION

The `dchroot` configuration file, `/usr/local/etc/dchroot.conf`, used by earlier versions of `dchroot`, has the following format:

- `#` starts a comment line.
- Blank lines are ignored.
- `Chroot` definitions are a single line containing an *identifier*, *path*, and an optional *personality* separated by whitespace.
- The first `chroot` is also the default `chroot`.

An example file:

```
# Example comment

sarge /srv/chroot/sarge
sid /srv/chroot/sid linux32
```

This file defines a `chroot` called 'sarge', located at `/srv/chroot/sarge`, and a second `chroot` called 'sid', located at `/srv/chroot/sid`. The second `chroot` uses the 'linux32' personality, which allows a 32-bit `chroot` to be used on a 64-bit system. 'sarge' is the default `chroot`, because it was listed first, which means if the `-c` option is omitted this `chroot` will be used.

## INCOMPATIBILITIES

### Debian `dchroot` prior to version 0.99.0

- Log messages are worded and formatted differently.
- The parsing of `/usr/local/etc/dchroot.conf` uses a smaller list of allowed whitespace characters (space and tab), which may cause a parse error during tokenising if the file contains odd characters as separators, such as carriage returns, vertical tabs and form feeds.
- `su(1)` is no longer used to run commands in the `chroot`; this is done by `dchroot` internally. This change may cause subtle differences. If you find an incompatibility, please report it so it may be corrected.
- `dchroot` provides a restricted subset of the functionality implemented by `schroot`, but is still `schroot` underneath. Thus `dchroot` is still subject to `schroot` security checking, including PAM authentication and authorisation, and session management, for example, and hence may behave slightly differently to older `dchroot` versions in some circumstances.

### DSA `dchroot`

Machines run by the Debian System Administrators for the Debian Project have a `dchroot-dsa` package which provides an alternate `dchroot` implementation.

- All the above incompatibilities apply.
- This version of dchroot has incompatible command-line options, and while some of those options are supported or have equivalent options by a different name, the `-c` option is not required to specify a chroot, and this version of dchroot cannot implement this behaviour in a backward-compatible manner (because if `-c` is omitted, the default chroot is used). DSA dchroot uses the first non-option as the chroot to use, only allowing one chroot to be used at once.
- This version of dchroot has an incompatible format for `dchroot.conf`. While the first two fields are the same, the remaining fields are an optional **users**, a list of users permitted to access the chroot, instead of the *personality* field allowed by this version. If access restrictions are needed, please use `/usr/local/etc/schroot/schroot.conf` and add the allowed users there, as shown in “Migration” below.

## MIGRATION

To migrate an existing **dchroot** configuration to **schroot**, perform the following steps:

- 1 Dump the dchroot configuration in schroot keyfile format to `/usr/local/etc/schroot/schroot.conf`.  

```
# dchroot --config >> /usr/local/etc/schroot/schroot.conf
```
- 2 Edit `/usr/local/etc/schroot/schroot.conf` to add access to the users and/or groups which are to be allowed to access the chroots, and make any other desired changes to the configuration. See **schroot.conf(5)**.
- 3 Remove `/usr/local/etc/dchroot.conf`, so that dchroot will subsequently use `/usr/local/etc/schroot/schroot.conf` for its configuration.

## EXAMPLES

```
$ dchroot -l
Available chroots: sarge [default], sid

$ dchroot -p sid
/srv/chroot/sid

$ dchroot -q -c sid -- uname -smr
Linux 2.6.16.17 ppc
$ dchroot -q -c sid -- "uname -smr"
Linux 2.6.16.17 ppc

$ dchroot -q -c sid "ls -l / | tac | head -n 4"
var
usr
tmp
sys

$ dchroot -c sid
I: [sid chroot] Running login shell: "/bin/bash"
$
```

Use `--` to allow options beginning with `-` or `---` in the command to run in the chroot. This prevents them being interpreted as options for dchroot itself. Note that the top line was echoed to standard error, and the remaining lines to standard output. This is intentional, so that program output from commands run in the chroot may be piped and redirected as required; the data will be the same as if the command was run directly on the host system.

## TROUBLESHOOTING

If something is not working, and it's not clear from the error messages what is wrong, try using the `--debug=level` option to turn on debugging messages. This gives a great deal more information. Valid debug levels are `'none'`, and `'notice'`, `'info'`, `'warning'` and `'critical'` in order of increasing severity. The

lower the severity level, the more output.

If you are still having trouble, the developers may be contacted on the mailing list:

Debian build-tools Developers  
<build-tools-devel@lists.alioth.debian.org>

## BUGS

On the **mips** and **mipsel** architectures, Linux kernels up to and including at least version 2.6.17 have broken **personality(2)** support, which results in a failure to set the personality. This will be seen as an “Operation not permitted” (EPERM) error. To work around this problem, set *personality* to ‘undefined’, or upgrade to a more recent kernel.

## FILES

### */usr/local/etc/dchroot.conf*

The system-wide **dchroot** chroot definition file. This file must be owned by the root user, and not be writable by other. If present, this file will be used in preference to */usr/local/etc/schroot/schroot.conf*.

### */usr/local/etc/schroot/schroot.conf*

The system-wide **schroot** definition file. This file must be owned by the root user, and not be writable by other. It is recommended that this file be used in preference to */usr/local/etc/dchroot.conf*, because the chroots can be used interchangeably with schroot, and the user and group security policies provided by schroot are also enforced.

## AUTHORS

Roger Leigh.

This implementation of dchroot uses the same command-line options as the original **dchroot** by David Kimdon <dwhedon@debian.org>, but is an independent implementation.

## COPYRIGHT

Copyright © 2005–2011 Roger Leigh <rleigh@debian.org>

**dchroot** is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

## SEE ALSO

**schroot(1)**, **sbuid(1)**, **chroot(2)**, **schroot-setup(5)**, **schroot.conf(5)**.

**NAME**

`dchroot-dsa` – enter a chroot environment

**SYNOPSIS**

`dchroot-dsa` [`-h`|`--help` | `-V`|`--version` | `-l`|`--list` | `-i`|`--info` | `--config` | `-p`|`--listpaths`] [`-d` *directory*|`--directory=directory`] [`-q`|`--quiet` | `-v`|`--verbose`] [`-c` *chroot*|`--chroot=chroot` | `--all` | **CHROOT**] [**COMMAND**]

**DESCRIPTION**

`dchroot-dsa` allows the user to run a command or a login shell in a chroot environment. If no command is specified, a login shell will be started in the user's home directory inside the chroot.

The user's environment will be preserved inside the chroot.

The command is a single argument which must be an absolute path to the program. Additional options are not permitted.

The directory the command or login shell is run in depends upon the context. See `--directory` option below for a complete description.

This version of `dchroot-dsa` is a compatibility wrapper around the `schroot(1)` program. It is provided for backward compatibility with the `dchroot-dsa` command-line options, but `schroot` is recommended for future use. See the section "*Migration*" below for help migrating your existing `dchroot-dsa` configuration to `schroot`. See the section "*Incompatibilities*" below for known incompatibilities with older versions of `dchroot-dsa`.

**OPTIONS**

`dchroot-dsa` accepts the following options:

**Basic options**

`-h, --help`

Show help summary.

`-a, --all`

Select all chroots. Note that earlier versions of `dchroot-dsa` did not include this option.

`-c, --chroot=chroot`

Specify a chroot to use. This option may be used multiple times to specify more than one chroot, in which case its effect is similar to `--all`. If this option is not used, the first non-option argument specified the chroot to use. Note that earlier versions of `dchroot-dsa` did not include this option.

`-l, --list`

List all available chroots.

`-i, --info`

Print detailed information about the available chroots. Note that earlier versions of `dchroot-dsa` did not include this option.

`-p, --listpaths`

Print absolute locations (paths) of the available chroots.

`--config`

Print configuration of the available chroots. This is useful for testing that the configuration in use is the same as the configuration file. Any comments in the original file will be missing. Note that earlier versions of `dchroot-dsa` did not include this option.

`-d, --directory=directory`

Change to *directory* inside the chroot before running the command or login shell. If *directory* is not available, `dchroot-dsa` will exit with an error status.

The default behaviour (all directory paths are inside the chroot) is to run the login shell or command in the user's home directory, or `/` if the home directory is not available. If none of the directories are available, `dchroot-dsa` will exit with an error status.

**-q, --quiet**

Print only essential messages. Note that earlier versions of `dchroot-dsa` did not include this option.

**-v, --verbose**

Print all messages. Note that earlier versions of `dchroot-dsa` did not include this option.

**-V, --version**

Print version information.

**CONFIGURATION**

The original **dchroot-dsa** configuration file, `/usr/local/etc/dchroot.conf`, used by earlier versions of `dchroot-dsa`, has the following format:

- `#` starts a comment line.
- Blank lines are ignored.
- Chroot definitions are a single line containing an *identifier*, *path*, and *users*, an optional user list separated by whitespace (space and tab), or a colon (`:`), semicolon (`;`), or comma (`,`).

An example file:

```
# Example comment

sarge /srv/chroot/sarge
sid /srv/chroot/sid rleigh,fred
```

This file defines a chroot called ‘sarge’, located at `/srv/chroot/sarge`, and a second chroot called ‘sid’, located at `/srv/chroot/sid`. The second chroot specifies that it may only be used by the users “rleigh” and “fred”.

**INCOMPATIBILITIES****DSA dchroot**

- Log messages are worded and formatted differently.
- `dchroot-dsa` provides a restricted subset of the functionality implemented by **schroot**, but is still **schroot** underneath. Thus `dchroot-dsa` is still subject to **schroot** security checking, including PAM authentication and authorisation, and session management, for example, and hence may behave slightly differently to earlier versions of `dchroot-dsa` in some circumstances.

**Debian dchroot**

A **dchroot** package provides an alternative `dchroot` implementation.

- All the above incompatibilities apply.
- This version of `dchroot` has incompatible command-line options, and while some of those options are supported or have equivalent options by a different name, the `-c` option is required to specify a chroot. It also allows a shell script to be used as the option instead of a single absolute path, and allows multiple command options instead of a single option.
- This version of `dchroot` has an incompatible format for `dchroot.conf`. While the first two fields are the same, the third field is an optional *personality*, instead of the *users* user list permitted to access the chroot allowed by this version. If personality support is needed, please use `schroot.conf` and add the allowed users there, as shown in “*Migration*” below.

**MIGRATION**

To migrate an existing **dchroot-dsa** configuration to **schroot**, perform the following steps:

- 1 Dump the `dchroot-dsa` configuration in **schroot** keyfile format to `/usr/local/etc/schroot/schroot.conf`.

```
# dchroot-dsa --config >> /usr/local/etc/schroot/schroot.conf
```
- 2 Edit `/usr/local/etc/schroot/schroot.conf` to add access to the users and/or groups which are to be allowed to access the chroots, and make any other desired changes to the configuration. See

**schroot.conf(5).**

- Remove `/usr/local/etc/dchroot.conf`, so that `dchroot-dsa` will subsequently use `/usr/local/etc/schroot/schroot.conf` for its configuration.

## EXAMPLES

```
$ dchroot-dsa -l
Available chroots: sarge, sid

$ dchroot-dsa --listpaths
/srv/chroot/sarge
/srv/chroot/sid

$ dchroot-dsa -q sid -- /bin/uname
Linux

$ dchroot-dsa sid
I: [sid chroot] Running login shell: "/bin/bash"
$
```

Note that the top line was echoed to standard error, and the remaining lines to standard output. This is intentional, so that program output from commands run in the chroot may be piped and redirected as required; the data will be the same as if the command was run directly on the host system.

## TROUBLESHOOTING

If something is not working, and it's not clear from the error messages what is wrong, try using the `--debug=level` option to turn on debugging messages. This gives a great deal more information. Valid debug levels are 'none', and 'notice', 'info', 'warning' and 'critical' in order of increasing severity. The lower the severity level, the more output.

If you are still having trouble, the developers may be contacted on the mailing list:

```
Debian buildd-tools Developers
<buildd-tools-devel@lists.alioth.debian.org>
```

## BUGS

None known at this time.

## FILES

***/usr/local/etc/dchroot.conf***

The system-wide **dchroot-dsa** chroot definition file. This file must be owned by the root user, and not be writable by other. If present, this file will be used in preference to `/usr/local/etc/schroot/schroot.conf`.

***/usr/local/etc/schroot/schroot.conf***

The system-wide **schroot** definition file. This file must be owned by the root user, and not be writable by other. It is recommended that this file be used in preference to `/usr/local/etc/dchroot.conf`, because the chroots can be used interchangeably with `schroot`, and the user and group security policies provided by `schroot` are also enforced.

## AUTHORS

Roger Leigh.

This implementation of `dchroot-dsa` uses the same command-line options as the **dchroot** found on machines run by the Debian System Administrators for the Debian Project. These machines have a **dchroot-dsa** source package which provides a **dchroot-dsa** package, written by Ben Collins <bcollins@debian.org> and Martin Schulze <joeey@debian.org>.

## COPYRIGHT

Copyright © 2005–2011 Roger Leigh <rleigh@debian.org>

**dchroot-dsa** is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your

option) any later version.

**SEE ALSO**

**schroot(1), sbuild(1), chroot(2), schroot-setup(5), schroot.conf(5).**