



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

**C:\>man capsh.1**

CAPSH(1) User Commands CAPSH(1)

### NAME

capsh - capability shell wrapper

### SYNOPSIS

capsh [OPTION]...

### DESCRIPTION

Linux capability support and use can be explored and constrained with this tool.

This tool provides a handy wrapper for certain types of capability testing and environment creation. It also provides some debugging features useful for summarizing capability state.

### OPTIONS

The tool takes a number of optional arguments, acting on them in the order they are provided. They are as follows:

- help Display the list of commands supported by capsh.
- print Display prevailing capability and related state.
- [args] Execute /bin/bash with trailing arguments. Note, you can use -c 'command to execute' for specific commands.
- == Execute capsh again with remaining arguments. Useful for testing exec() behavior.
- caps=cap-set Set the prevailing process capabilities to those specified by cap-set. Where cap-set is a text-representation of capability state as per cap\_from\_text(3).

`--drop=cap-list` Remove the listed capabilities from the prevailing bounding set. The capabilities are a comma separated list of capabilities as recognized by the `cap_from_name(3)` function. Use of this feature requires that the `capsh` program is operating with `CAP_SETPCAP` in its effective set.

`--inh=cap-list` Set the inheritable set of capabilities for the current process to equal those provided in the comma separated list. For this action to succeed, the prevailing process should already have each of these capabilities in the union of the current inheritable and permitted capability sets, or the `capsh` program is operating with `CAP_SETPCAP` in its effective set.

`--user=username` Assume the identity of the named user. That is, look up the user's uid and gid with `getpwuid(3)` and their group memberships with `getgrouplist(3)` and set them all using `cap_setuid(3)` and `cap_setgroups(3)`. Following this command, the effective capabilities will be cleared, but the permitted set will not be so the running program is still privileged.

`--modes` Lists all of the libcap modes supported by `--mode`.

`--mode=<mode>` Force the program into a `cap_set_mode(3)` security mode. This is a set of securebits and prevailing capability arrangement recommended for its pre-determined security stance.

`--inmode=<mode>` Confirm that the prevailing mode is so named, or exit with a status 1.

`--uid=id` Force all uid values to equal `id` using the `setuid(2)` system call. This argument may require explicit preparation of the effective set.

`--cap-uid=<uid>` use the `cap_setuid(3)` function to set the uid of the current process. This performs all preparations for setting the uid without dropping capabilities in the process. Following this command the prevailing effective capabilities will be lowered.

`--is-uid=<id>` Exit with status 1 unless the current uid equals `<id>`.

`--gid=<id>` Force all gid values to equal id using the `setgid(2)` system call.

`--is-gid=<id>` Exit with status 1 unless the current gid equals `<id>`.

`--groups=<gid-list>` Set the supplementary groups to the numerical list provided. The groups are set with the `setgroups(2)` system call. See `--user` for a more convenient way of doing this.

`--keep=<0|1>` In a non-pure capability mode, the kernel provides liberal privilege to the super-user. However, it is normally the case that when the super-user changes uid to some lesser user, then capabilities are dropped. For these situations, the kernel can permit the process to retain its capabilities after a `setuid(2)` system call. This feature is known as `keep-caps` support. The way to activate it using this program is with this argument. Setting the value to 1 will cause `keep-caps` to be active. Setting it to 0 will cause `keep-caps` to deactivate for the current process. In all cases, `keep-caps` is deactivated when an `exec()` is performed. See `--secbits` for ways to disable this feature.

`--secbits=N` Set the security-bits for the program, this is via `prctl(2)`, `PR_SET_SECUREBITS` API, and the list of supported bits and their meaning can be found in the `<sys/secbits.h>` header file. The program will list these bits via the `--print` command.

`--chroot=path` Execute the `chroot(2)` system call with the new root-directory (`/`) equal to `path`. This operation requires `CAP_SYS_CHROOT` to be in effect.

`--forkfor=sec` This command causes the program to fork a child process for so many seconds. The child will sleep that long and then exit with status 0. The purpose of this command is to support exploring the way processes are killable in the face of capability changes. See the `--killit` command. Only one fork can be active at a time.

`--killit=sig` This command causes a `--forkfor` child to be `kill(2)`d with

the specified signal. The command then waits for the child to exit. If the exit status does not match the signal being used to kill it, the capsh program exits with status 1.

`--decode=N` This is a convenience feature. If you look at `/proc/1/status`

there are some capability related fields of the following

form:

CapInh: 0000000000000000

CapPrm: ffffffff

CapEff: ffffffff

CapBnd: ffffffff

This option provides a quick way to decode a capability vec?

tor represented in this form. For example, the missing capa?

bility from this effective set is 0x0100. By running:

```
capsh --decode=0x0100
```

we observe that the missing capability is: `cap_setpcap`.

`--supports=xxx` As the kernel evolves, more capabilities are added. This op?

tion can be used to verify the existence of a capability on

the system. For example, `--supports=cap_syslog` will cause

capsh to promptly exit with a status of 1 when run on kernel

2.6.27. However, when run on kernel 2.6.38 it will silently

succeed.

`--has-p=xxx` Exit with status 1 unless the permitted vector has capability

xxx raised.

`--has-ambient` Performs a check to see if the running kernel supports ambi?

ent capabilities. If not, the capsh command exits with status

1.

`--has-a=xxx` Exit with status 1 unless the ambient vector has capability

xxx raised.

`--addamb=xxx` Adds the specified ambient capability to the running

process.

`--delamb=xxx` Removes the specified ambient capability from the running

process.

`--noamb` Drops all ambient capabilities from the running process.

## EXIT STATUS

Following successful execution the tool exits with status 0. Following an error, the tool immediately exits with status 1.

## AUTHOR

Written by Andrew G. Morgan <morgan@kernel.org>.

## REPORTING BUGS

Please report bugs via:

[https://bugzilla.kernel.org/buglist.cgi?component=libcap&list\\_id=1047723&prod?uct=Tools&resolution=---](https://bugzilla.kernel.org/buglist.cgi?component=libcap&list_id=1047723&product=Tools&resolution=---)

## SEE ALSO

libcap(3), getcap(8), setcap(8) and capabilities(7).

libcap 2

2020-01-07

CAPSH(1)