NAME

bwrap − container setup utility

SYNOPSIS

bwrap [*OPTION...*] [*COMMAND*]

DESCRIPTION

bwrap is a unprivileged low−level sandboxing tool (optionally setuid on older distributions). You are unlikely to use it directly from the commandline, although that is possible.

It works by creating a new, completely empty, filesystem namespace where the root is on a tmpfs that is invisible from the host, and which will be automatically cleaned up when the last process exits. You can then use commandline options to construct the root filesystem and process environment for the command to run in the namespace.

By default, **bwrap** creates a new mount namespace for the sandbox. Optionally it also sets up new user, ipc, pid, network and uts namespaces (but note the user namespace is required if bwrap is not installed setuid root). The application in the sandbox can be made to run with a different UID and GID.

If needed (e.g. when using a PID namespace) **bwrap** is running a minimal pid 1 process in the sandbox that is responsible for reaping zombies. It also detects when the initial application process (pid 2) dies and reports its exit status back to the original spawner. The pid 1 process exits to clean up the sandbox when there are no other processes in the sandbox left.

OPTIONS

When options are used multiple times, the last option wins, unless otherwise specified.

General options:

−−help
Print help and exit

−−version
Print version

−−args FD
Parse nul−separated arguments from the given file descriptor. This option can be used multiple times to parse options from multiple sources.

−−argv0 VALUE
Set argv[0] to the value VALUE before running the program

−−level−prefix
Prefix each line of diagnostic output with a numeric severity level enclosed in angle brackets. The severity levels used are based on the constants used by **syslog**(3): for example, <4> indicates a warning, because LOG_WARNING has numeric value 4. Numbers smaller than 4 indicate fatal errors, and numbers larger than 4 indicate informational messages. These prefixes can be parsed by tools compatible with logger −−prio−prefix (see **logger**(1)) or systemd−cat −−level−prefix=1 (see **systemd-cat**(1)).

Options related to kernel namespaces:

−−unshare−user
Create a new user namespace

−−unshare−user−try
Create a new user namespace if possible else skip it

−−unshare−ipc
Create a new ipc namespace

−−unshare−pid
Create a new pid namespace

−−unshare−net
Create a new network namespace

−−unshare−uts
Create a new uts namespace

−−unshare−cgroup
Create a new cgroup namespace

−−unshare−cgroup−try
Create a new cgroup namespace if possible else skip it

−−unshare−all
Unshare all possible namespaces. Currently equivalent with: **−−unshare−user−try −−unshare−ipc −−unshare−pid −−unshare−net −−unshare−uts −−unshare−cgroup−try**

−−share−net
Retain the network namespace, overriding an earlier **−−unshare−all** or **−−unshare−net**

−−userns FD
Use an existing user namespace instead of creating a new one. The namespace must fulfil the permission requirements for setns(), which generally means that it must be a descendant of the currently active user namespace, owned by the same user.

This is incompatible with −−unshare−user, and doesn't work in the setuid version of bubblewrap.

−−userns2 FD
After setting up the new namespace, switch into the specified namespace. For this to work the specified namespace must be a descendant of the user namespace used for the setup, so this is only useful in combination with −−userns.

This is useful because sometimes bubblewrap itself creates nested user namespaces (to work around some kernel issues) and −−userns2 can be used to enter these.

−−disable−userns
Prevent the process in the sandbox from creating further user namespaces, so that it cannot rearrange the filesystem namespace or do other more complex namespace modification. This is currently implemented by setting the user.max_user_namespaces sysctl to 1, and then entering a nested user namespace which is unable to raise that limit in the outer namespace. This option requires **−−unshare−user**, and doesn't work in the setuid version of bubblewrap.

−−assert−userns−disabled
Confirm that the process in the sandbox has been prevented from creating further user namespaces, but without taking any particular action to prevent that. For example, this can be combined with **−−userns** to check that the given user namespace has already been set up to prevent the creation of further user namespaces.

−−pidns FD
Use an existing pid namespace instead of creating one. This is often used with −−userns, because the pid namespace must be owned by the same user namespace that bwrap uses.

Note that this can be combined with −−unshare−pid, and in that case it means that the sandbox will be in its own pid namespace, which is a child of the passed in one.

−−uid UID
Use a custom user id in the sandbox (requires **−−unshare−user**)

−−gid GID
Use a custom group id in the sandbox (requires **−−unshare−user**)

−−hostname HOSTNAME
Use a custom hostname in the sandbox (requires **−−unshare−uts**)

Options about environment setup:

−−chdir DIR
Change directory to DIR

−−setenv VAR VALUE
Set an environment variable

−−unsetenv VAR
Unset an environment variable

−−clearenv
Unset all environment variables, except for **PWD** and any that are subsequently set by **−−setenv**

Options for monitoring the sandbox from the outside:

−−lock−file DEST
Take a lock on DEST while the sandbox is running. This option can be used multiple times to take locks on multiple files.

−−sync−fd FD
Keep this file descriptor open while the sandbox is running

Filesystem related options. These are all operations that modify the filesystem directly, or mounts stuff in the filesystem. These are applied in the order they are given as arguments.

Any missing parent directories that are required to create a specified destination are automatically created as needed. Their permissions are normally set to 0755 (rwxr−xr−x). However, if a **−−perms** option is in effect, and it sets the permissions for group or other to zero, then newly−created parent directories will also have their corresponding permission set to zero. **−−size** modifies the size of the created mount when preceding a **−−tmpfs** action; **−−perms** and **−−size** can be combined.

−−perms OCTAL
This option does nothing on its own, and must be followed by one of the options that it affects. It sets the permissions for the next operation to OCTAL. Subsequent operations are not affected: for example, −−perms 0700 −−tmpfs /a −−tmpfs /b will mount /a with permissions 0700, then return to the default permissions for /b. Note that **−−perms** and **−−size** can be combined: −−perms 0700 −−size 10485760 −−tmpfs /s will apply permissions as well as a maximum size to the created tmpfs.

−−size BYTES
This option does nothing on its own, and must be followed by −−tmpfs. It sets the size in bytes for the next tmpfs. For example, −−size 10485760 −−tmpfs /tmp will create a tmpfs at /tmp of size 10MiB. Subsequent operations are not affected: for example, −−size 10485760 −−tmpfs /a −−tmpfs /b will mount /a with size 10MiB, then return to the default size for /b. Note that **−−perms** and **−−size** can be combined: −−size 10485760 −−perms 0700 −−tmpfs /s will apply permissions as well as a maximum size to the created tmpfs.

−−bind SRC DEST
Bind mount the host path SRC on DEST

−−bind−try SRC DEST
Equal to **−−bind** but ignores non−existent SRC

−−dev−bind SRC DEST
Bind mount the host path SRC on DEST, allowing device access

−−dev−bind−try SRC DEST
Equal to **−−dev−bind** but ignores non−existent SRC

−−ro−bind SRC DEST
Bind mount the host path SRC readonly on DEST

−−ro−bind−try SRC DEST
Equal to **−−ro−bind** but ignores non−existent SRC

−−remount−ro DEST
Remount the path DEST as readonly. It works only on the specified mount point, without changing any other mount point under the specified path

−−overlay−src SRC
This option does nothing on its own, and must be followed by one of the other overlay options. It specifies a host path from which files should be read if they aren't present in a higher layer.

This option can be used multiple times to provide multiple sources. The sources are overlaid in the order given, with the first source on the command line at the bottom of the stack: if a given path to be read exists in more than one source, the file is read from the last such source specified.

(For readers familiar with overlayfs, note that this is the reverse of the order used by the kernel's lowerdir mount option.)

−−overlay RWSRC WORKDIR DEST

−−tmp−overlay DEST

−−ro−overlay DEST
Use overlayfs to mount the host paths specified by RWSRC and all immediately preceding **−−overlay−src** on DEST. DEST will contain the union of all the files in all the layers.

With −−overlay all writes will go to RWSRC. Reads will come preferentially from RWSRC, and then from any **−−overlay−src** paths. WORKDIR must be an empty directory on the same filesystem as RWSRC, and is used internally by the kernel.

With −−tmp−overlay all writes will go to the tmpfs that hosts the sandbox root, in a location not accessible from either the host or the child process. Writes will therefore not be persisted across multiple runs.

With −−ro−overlay the filesystem will be mounted read−only. This option requires at least two **−−overlay−src** to precede it.

None of these options are available in the setuid version of bubblewrap. Using −−ro−overlay or providing more than one **−−overlay−src** requires a Linux kernel version of 4.0 or later.

Due to limitations of overlayfs, no host directory given via −−overlay−src or −−overlay may be an ancestor of another, after resolving symlinks. Depending on version, the Linux kernel may or may not enforce this, but if not then overlayfs's behavior is undefined.

For more information see the Overlay Filesystem documentation in the Linux kernel at https://www.kernel.org/doc/Documentation/filesystems/overlayfs.txt

−−proc DEST
Mount procfs on DEST

−−dev DEST
Mount new devtmpfs on DEST

−−tmpfs DEST
Mount new tmpfs on DEST. If the previous option was **−−perms**, it sets the mode of the tmpfs. Otherwise, the tmpfs has mode 0755. If the previous option was **−−size**, it sets the size in bytes of the tmpfs. Otherwise, the tmpfs has the default size.

−−mqueue DEST
Mount new mqueue on DEST

−−dir DEST
Create a directory at DEST. If the directory already exists, its permissions are unmodified, ignoring

−−perms (use **−−chmod** if the permissions of an existing directory need to be changed). If the directory is newly created and the previous option was **−−perms**, it sets the mode of the directory. Otherwise, newly−created directories have mode 0755.

−−file FD DEST
Copy from the file descriptor FD to DEST. If the previous option was **−−perms**, it sets the mode of the new file. Otherwise, the file has mode 0666 (note that this is not the same as **−−bind−data**).

−−bind−data FD DEST
Copy from the file descriptor FD to a file which is bind−mounted on DEST. If the previous option was **−−perms**, it sets the mode of the new file. Otherwise, the file has mode 0600 (note that this is not the same as **−−file**).

−−ro−bind−data FD DEST
Copy from the file descriptor FD to a file which is bind−mounted read−only on DEST. If the previous option was **−−perms**, it sets the mode of the new file. Otherwise, the file has mode 0600 (note that this is not the same as **−−file**).

−−symlink SRC DEST
Create a symlink at DEST with target SRC.

Since version 0.9.0, it is not considered to be an error if DEST already exists as a symbolic link and its target is exactly SRC.

Before version 0.9.0, if DEST already existed, this would be treated as an error (even if its target was identical to SRC).

−−chmod OCTAL PATH
Set the permissions of PATH, which must already exist, to OCTAL.

Lockdown options:

−−seccomp FD
Load and use seccomp rules from FD. The rules need to be in the form of a compiled cBPF program, as generated by seccomp_export_bpf. If this option is given more than once, only the last one is used. Use **−−add−seccomp−fd** if multiple seccomp programs are needed.

−−add−seccomp−fd FD
Load and use seccomp rules from FD. The rules need to be in the form of a compiled cBPF program, as generated by seccomp_export_bpf. This option can be repeated, in which case all the seccomp programs will be loaded in the order given (note that the kernel will evaluate them in reverse order, so the last program on the bwrap command−line is evaluated first). All of them, except possibly the last, must allow use of the PR_SET_SECCOMP prctl. This option cannot be combined with **−−seccomp**.

−−exec−label LABEL
Exec Label from the sandbox. On an SELinux system you can specify the SELinux context for the sandbox process(s).

−−file−label LABEL
File label for temporary sandbox content. On an SELinux system you can specify the SELinux context for the sandbox content.

−−block−fd FD
Block the sandbox on reading from FD until some data is available.

−−userns−block−fd FD
Do not initialize the user namespace but wait on FD until it is ready. This allow external processes (like newuidmap/newgidmap) to setup the user namespace before it is used by the sandbox process.

−−info−fd FD
Write information in JSON format about the sandbox to FD.

−−json−status−fd FD

Multiple JSON documents are written to FD, one per line (**"JSON lines" format**[1]). Each line is a single JSON object. After **bwrap** has started the child process inside the sandbox, it writes an object with a child−pid member to the **−−json−status−fd** (this duplicates the older **−−info−fd**). The corresponding value is the process ID of the child process in the pid namespace from which **bwrap** was run. If available, the namespace IDs are also included in the object with the child−pid; again, this duplicates the older **−−info−fd**. When the child process inside the sandbox exits, **bwrap** writes an object with an exit−code member, and then closes the **−−json−status−fd**. The value corresponding to exit−code is the exit status of the child, in the usual shell encoding (n if it exited normally with status n, or 128+n if it was killed by signal n). Other members may be added to those objects in future versions of **bwrap**, and other JSON objects may be added before or after the current objects, so readers must ignore members and objects that they do not understand.

−−new−session
Create a new terminal session for the sandbox (calls setsid()). This disconnects the sandbox from the controlling terminal which means the sandbox can't for instance inject input into the terminal.

Note: In a general sandbox, if you don't use −−new−session, it is recommended to use seccomp to disallow the TIOCSTI ioctl, otherwise the application can feed keyboard input to the terminal which can e.g. lead to out−of−sandbox command execution (see CVE−2017−5226).

−−die−with−parent
Ensures child process (COMMAND) dies when bwrap's parent dies. Kills (SIGKILL) all bwrap sandbox processes in sequence from parent to child including COMMAND process when bwrap or bwrap's parent dies. See prctl, PR_SET_PDEATHSIG.

−−as−pid−1
Do not create a process with PID=1 in the sandbox to reap child processes.

−−cap−add CAP
Add the specified capability CAP, e.g. CAP_DAC_READ_SEARCH, when running as privileged user. It accepts the special value ALL to add all the permitted caps.

−−cap−drop CAP
Drop the specified capability when running as privileged user. It accepts the special value ALL to drop all the caps. By default no caps are left in the sandboxed process. The **−−cap−add** and **−−cap−drop** options are processed in the order they are specified on the command line. Please be careful to the order they are specified.

ENVIRONMENT

HOME
Used as the cwd in the sandbox if **−−chdir** has not been explicitly specified and the current cwd is not present inside the sandbox. The **−−setenv** option can be used to override the value that is used here.

EXIT STATUS

The **bwrap** command returns the exit status of the initial application process (pid 2 in the sandbox).

NOTES

1. "JSON lines" format
   https://jsonlines.org/