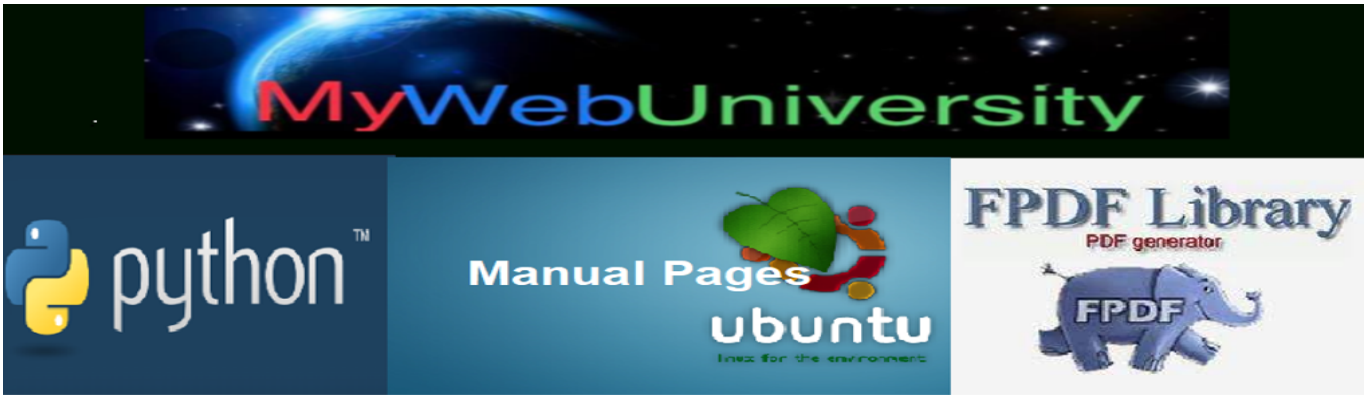

Rocky Enterprise Linux 9.2 Manual Pages on command 'cgroup_namespaces.7'

$ man cgroup_namespaces.7

CGROUP_NAMESPACES(7) Linux Programmer's Manual CGROUP_NAMESPACES(7)

NAME

cgroup_namespaces - overview of Linux cgroup namespaces

DESCRIPTION

For an overview of namespaces, see namespaces(7).

Cgroup namespaces virtualize the view of a process's cgroups (see cgroups(7)) as seen via /proc/[pid]/cgroup and /proc/[pid]/mountinfo.

Each cgroup namespace has its own set of cgroup root directories. These root directories are the base points for the relative locations displayed in the corresponding records in the /proc/[pid]/cgroup file. When a process creates a new cgroup namespace using clone(2) or unshare(2) with the CLONE_NEWCGROUP flag, its current cgroups directories become the cgroup root directories of the new namespace. (This applies both for the cgroups version 1 hierarchies and the cgroups version 2 unified hierarchy.)

When reading the cgroup memberships of a "target" process from /proc/[pid]/cgroup, the pathname shown in the third field of each record will be relative to the reading process's root directory for the corresponding cgroup hierarchy. If the cgroup directory of the target process lies outside the root directory of the reading process's cgroup namespace, then the pathname will show ../ entries for each ancestor level in the cgroup hierarchy.

The following shell session demonstrates the effect of creating a new cgroup namespace. First, (as superuser) in a shell in the initial cgroup namespace, we create a child cgroup in the freezer hierarchy, and place a process in that cgroup that we will use as part of the demonstration below:

```
# mkdir -p /sys/fs/cgroup/freezer/sub2
```

```
# sleep 10000 &     # Create a process that lives for a while
[1] 20124
# echo 20124 > /sys/fs/cgroup/freezer/sub2/cgroup.procs
```

We then create another child cgroup in the freezer hierarchy and put the shell into that cgroup:

```
# mkdir -p /sys/fs/cgroup/freezer/sub
# echo $$                    # Show PID of this shell
30655
# echo 30655 > /sys/fs/cgroup/freezer/sub/cgroup.procs
# cat /proc/self/cgroup | grep freezer
7:freezer:/sub
```

Next, we use unshare(1) to create a process running a new shell in new cgroup and mount namespaces:

```
# PS1="sh2# " unshare -Cm bash
```

From the new shell started by unshare(1), we then inspect the /proc/[pid]/cgroup files of, respectively, the new shell, a process that is in the initial cgroup namespace (init, with PID 1), and the process in the sibling cgroup (sub2):

```
sh2# cat /proc/self/cgroup | grep freezer
7:freezer:/
sh2# cat /proc/1/cgroup | grep freezer
7:freezer:/..
sh2# cat /proc/20124/cgroup | grep freezer
7:freezer:/../sub2
```

From the output of the first command, we see that the freezer cgroup membership of the new shell (which is in the same cgroup as the initial shell) is shown defined relative to the freezer cgroup root directory that was established when the new cgroup namespace was cre? ated. (In absolute terms, the new shell is in the /sub freezer cgroup, and the root di? rectory of the freezer cgroup hierarchy in the new cgroup namespace is also /sub. Thus, the new shell's cgroup membership is displayed as '/'.)

However, when we look in /proc/self/mountinfo we see the following anomaly:

```
sh2# cat /proc/self/mountinfo | grep freezer
155 145 0:32 /.. /sys/fs/cgroup/freezer ...
```

The fourth field of this line (/..) should show the directory in the cgroup filesystem

which forms the root of this mount. Since by the definition of cgroup namespaces, the process's current freezer cgroup directory became its root freezer cgroup directory, we should see '/' in this field. The problem here is that we are seeing a mount entry for the cgroup filesystem corresponding to the initial cgroup namespace (whose cgroup filesys? tem is indeed rooted at the parent directory of sub). To fix this problem, we must re? mount the freezer cgroup filesystem from the new shell (i.e., perform the mount from a process that is in the new cgroup namespace), after which we see the expected results:

```
sh2# mount --make-rslave /     # Don't propagate mount events
                               # to other namespaces
sh2# umount /sys/fs/cgroup/freezer
sh2# mount -t cgroup -o freezer freezer /sys/fs/cgroup/freezer
sh2# cat /proc/self/mountinfo | grep freezer
155 145 0:32 / /sys/fs/cgroup/freezer rw,relatime ...
```

CONFORMING TO

Namespaces are a Linux-specific feature.

NOTES

Use of cgroup namespaces requires a kernel that is configured with the CONFIG_CGROUPS op? tion.

The virtualization provided by cgroup namespaces serves a number of purposes:

* It prevents information leaks whereby cgroup directory paths outside of a container would otherwise be visible to processes in the container. Such leakages could, for ex? ample, reveal information about the container framework to containerized applications.
* It eases tasks such as container migration. The virtualization provided by cgroup name? spaces allows containers to be isolated from knowledge of the pathnames of ancestor cgroups. Without such isolation, the full cgroup pathnames (displayed in /proc/self/cgroups) would need to be replicated on the target system when migrating a container; those pathnames would also need to be unique, so that they don't conflict with other pathnames on the target system.
* It allows better confinement of containerized processes, because it is possible to mount the container's cgroup filesystems such that the container processes can't gain access to ancestor cgroup directories. Consider, for example, the following scenario:
  ? We have a cgroup directory, /cg/1, that is owned by user ID 9000.
  ? We have a process, X, also owned by user ID 9000, that is namespaced under the

cgroup /cg/1/2 (i.e., X was placed in a new cgroup namespace via clone(2) or un? share(2) with the CLONE_NEWCGROUP flag).

In the absence of cgroup namespacing, because the cgroup directory /cg/1 is owned (and writable) by UID 9000 and process X is also owned by user ID 9000, process X would be able to modify the contents of cgroups files (i.e., change cgroup settings) not only in /cg/1/2 but also in the ancestor cgroup directory /cg/1. Namespacing process X under the cgroup directory /cg/1/2, in combination with suitable mount operations for the cgroup filesystem (as shown above), prevents it modifying files in /cg/1, since it can? not even see the contents of that directory (or of further removed cgroup ancestor di? rectories). Combined with correct enforcement of hierarchical limits, this prevents process X from escaping the limits imposed by ancestor cgroups.

SEE ALSO

unshare(1), clone(2), setns(2), unshare(2), proc(5), cgroups(7), credentials(7), name? spaces(7), user_namespaces(7)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.