Correction Docker Namespace

Exercice 1 :

L'objectif de cet exercice est de démontrer l'utilité des namespaces dans l'architecture des containers Docker. Comme vu en cours, les namespaces sont un des principes d'isolation des processus qui sont à l'origine des containers. Chaque namespaces (UTS,USER,PID etc...) permettent d'isoler des ressources (Utilisateurs, PID, hostname, point de montage etc..) entre différents processus sur une machine Linux.

Le flag `--pid=host` permet de partager le namespace **pid** entre l'hôte et le container.

```
{9:21}/home/allta ➲ docker run --rm --pid=host -ti ubuntu bash
root@a91f725b30b5:/#
```

Le container ubuntu et notre hôte partagent le même namespace donc depuis le container nous pouvons voir et interargir avec les processus de l'hote.

Nous savons que le **PID 1** est important pour un container. Il s'agit du process lancé lors de l'appel d'une image via l'instruction CMD/ENTRYPOINT.

Cependant la commande ps nous montre que le **PID 1** est le processus **d'initialisation** de la machine Linux et non le programme **bash** que nous avions spécifié lors du lancement du container. Cela montre bien que depuis le container nous ayons bien accés aux processus de l'hôte.

```
root@a91f725b30b5:/# ps aux | head
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.1 164584 10880 ?        Ss   08:04   0:01 /sbin/init
root           2  0.0  0.0      0     0 ?        S    08:04   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   08:04   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   08:04   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   08:04   0:00 [kworker/0:0H-events_highpri]
root           7  0.0  0.0      0     0 ?        I    08:04   0:00 [kworker/0:1-rcu_gp]
root           9  0.0  0.0      0     0 ?        I<   08:04   0:00 [mm_percpu_wq]
root          10  0.0  0.0      0     0 ?        S    08:04   0:00 [rcu_tasks_rude_]
root          11  0.0  0.0      0     0 ?        S    08:04   0:00 [rcu_tasks_trace]
root@a91f725b30b5:/#
```

Pour confirmer cette hypothèse nous pouvons vérifier les ID des namespaces liés à chaque processus. Toutes les informations liées aux processus sont retrouvable dans le dossier `/proc/<PID>/`.

```
root@a91f725b30b5:/# ll /proc/self/ns/
total 0
dr-x--x--x 2 root root 0 Nov 10 08:24 ./
dr-xr-xr-x 9 root root 0 Nov 10 08:24 ../
lrwxrwxrwx 1 root root 0 Nov 10 08:24 cgroup -> 'cgroup:[4026533264]'
lrwxrwxrwx 1 root root 0 Nov 10 08:24 ipc -> 'ipc:[4026533196]'
lrwxrwxrwx 1 root root 0 Nov 10 08:24 mnt -> 'mnt:[4026533194]'
lrwxrwxrwx 1 root root 0 Nov 10 08:24 net -> 'net:[4026533198]'
lrwxrwxrwx 1 root root 0 Nov 10 08:24 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 Nov 10 08:24 pid_for_children -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 Nov 10 08:24 time -> 'time:[4026531834]'
lrwxrwxrwx 1 root root 0 Nov 10 08:24 time_for_children -> 'time:[4026531834]'
lrwxrwxrwx 1 root root 0 Nov 10 08:24 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0 Nov 10 08:24 uts -> 'uts:[4026533195]'
root@a91f725b30b5:/#
```

```
{9:24}/home/allta ➲ ll /proc/self/ns
total 0
lrwxrwxrwx 1 root root 0 10 nov.  09:24 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 root root 0 10 nov.  09:24 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx 1 root root 0 10 nov.  09:24 mnt -> 'mnt:[4026531840]'
lrwxrwxrwx 1 root root 0 10 nov.  09:24 net -> 'net:[4026531992]'
lrwxrwxrwx 1 root root 0 10 nov.  09:24 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 10 nov.  09:24 pid_for_children -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 10 nov.  09:24 time -> 'time:[4026531834]'
lrwxrwxrwx 1 root root 0 10 nov.  09:24 time_for_children -> 'time:[4026531834]'
lrwxrwxrwx 1 root root 0 10 nov.  09:24 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0 10 nov.  09:24 uts -> 'uts:[4026531838]'
{9:24}/home/allta ➲
```

Ci-dessus : Le terminal supérieur présente les namespaces du container et celui du bas ceux de l'hôte. On constate bien que l'ID du namespace PID est le même pour les 2 processus. Le namespace UTS est lui différent et nous pouvons le constater facilement en regardant l'hostname du container et celui de l'hôte, ce ne sont pas les mêmes donc le namespace UTS ne sont pas partagés

Note : `/proc/self` équivaut au process actuellement en cours qui appele le symlink /proc/self

Source : https://unix.stackexchange.com/questions/333225/which-process-is-proc-self-for

Code source du kernel Linux : https://elixir.bootlin.com/linux/latest/source/fs/proc/self.c https://elixir.bootlin.com/linux/latest/source/fs/proc/self.c

Dans le container, nous créons une variable d'environnement `FOO` ainsi qu'un processus qui va tourner non stop. Le processus ici sera `sleep`.

Le `echo $$` permet d'afficher le **PID** du process en cours. Ici il s'agit du process `bash` lancé lors de la création du container.

Puisque le namespace pid est partagé nous devrions pouvoir le retrouvé depuis notre hôte.

```
root@a91f725b30b5:/# export FOO=BAR
root@a91f725b30b5:/# while true; do echo $$ && sleep 100;done
5484
5484
```

Le proccess lancé dans notre container doit forcément être un enfant du processus *dockerd* (Docker Daemon). On recherche donc le process `dockerd` et on affiche les quelques lignes après pour retrouver le process `sleep` de notre container.

```
{9:42}/home/allta ➲ ps auxf | grep -A 5 "[d]ockerd"
root        3611  0.1  1.3 3596928 111408 ?      Ssl  09:18   0:01 /usr/sbin/dockerd -H fd://
root        3626  0.2  0.6 1928144 51964 ?       Ssl  09:18   0:03  \_ containerd --config /var/run/docker/containerd/containerd.toml --log-level info
root        5464  0.0  0.2 1452896 21648 ?       Sl   09:21   0:00 /usr/bin/containerd-shim-runc-v2 -namespace moby -id a91f725b30b593f0ef65755d67aa1601c064d59927ce6adb293977057e
root        5484  0.0  0.0   4240  3380 pts/0    Ss   09:21   0:00  \_ bash
root        7705  0.0  0.0   2516   580 pts/0    S+   09:41   0:00      \_ sleep 100
{9:42}/home/allta ➲
```

Nous voyons bien que le processus **5484** est `bash` comme vu dans notre container grâce à la commande `echo $$`.

Une fois le PID retrouvé nous pouvons afficher les informations relative à ce processus. Toutes les infos seront dans `/proc/7965/` ce qui nous intéresse est la variable d'environnement créée dans le container.

```
{9:44}/home/allta ➲ cat /proc/7965/environ
HOSTNAME=a91f725b30b5PWD=/FOO=BARHOME=/root
```