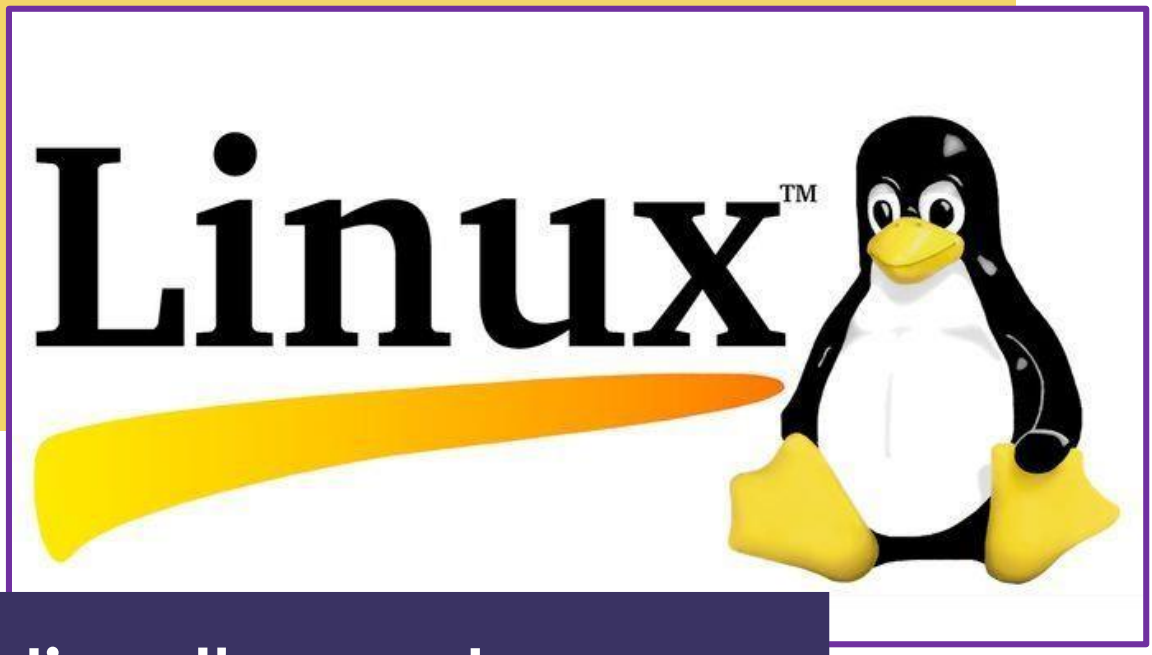

Création d'un conteneur via les Namespaces et les Cgroups

Module: Système exploitation

Encadré par:

Pr.Airaj Mohammed

Réalisé par:

EL OUADI Abdelati

&

KHADIM Meriem

Filière :

Deuxième année cycle ingénieur en réseau informatique et système d'information (IRISI2)

INTRODUCTION

Les conteneurs sont une forme de virtualisation du système d'exploitation. Un seul conteneur peut être utilisé pour exécuter n'importe quoi, d'un petit micro service ou processus logiciel à une application plus grande. À l'intérieur d'un conteneur se trouvent tous les exécutables, le code binaire, les bibliothèques et les fichiers de configuration nécessaires.

Pour le créer nous allons besoin des namespaces et les cgroups.

Clone

Nous utilisons clone pour créer un processus enfant qui partage des ressources avec son parent. Une utilisation du clone consiste à implémenter le multithreading, une autre utilisation consiste à implémenter des espaces de noms

Namespaces

Les espaces de noms Linux sont l'un des blocs de construction pour implémenter les conteneurs. Les espaces de noms contrôlent ce qu'un processus peut voir. Il peut s'agir des ID de processus, des points de montage, etc.

Pour utiliser les espaces de noms, nous appelons l'appel système clone.

•Pour créer un processus enfant dans un nouvel espace de noms et des ressources isolées, nous devons utiliser un ou plusieurs des indicateurs suivants:

- CLONE_NEWNET : isoler les périphériques réseau.
- CLONE_NEWUTS : noms d'hôte et de domaine (système de partage de temps UNIX)

•CLONE_NEWIPC : Si CLONE_NEWIPC est défini, clone va créer le processus dans un nouveau espace de noms IPC (interconnexion process communication).

- CLONE_NEWPID : PIDs
- CLONE_NEWNS : points de montage (systèmes de fichiers) :

Lors du premier démarrage du système, il existe un seul espace de noms de montage, appelé «espace de noms initial». Les nouveaux espaces de noms de montage sont créés à l'aide de l'indicateur CLONE_NEWNS avec l'appel système clone (), en effet l'enfant obtient une copie des données du système de fichiers monté de son parent.

- CLONE_NEWUSER : utilisateurs et groupes Voici ci-dessus l'appel du clone dans notre projet:

```
/*
in this case clone function will create a new isolate process that will execute child function:
The stack argument specifies the location of the stack used by the child process.
CLONE_NEWNET  - isolate network devices
CLONE_NEWUTS - host and domain names (UNIX Timesharing System)
CLONE_NEWIPC - IPC objects
CLONE_NEWPID - PIDs
CLONE_NEWNS - mount points (file systems)
CLONE_NEWUSER - users and groups
SIGCHLD (anciennement SIGCLD) is a signal used to wake up a process where one of the children has just died
*/
pid_t pid = clone(child, stack+STACK_SIZE,
CLONE_NEWNET | CLONE_NEWUTS | CLONE_NEWIPC | CLONE_NEWPID | CLONE_NEWNS | SIGCHLD, NULL);
```

Après l'exécution de clone, un processus fils va être créé et il va exécuter la fonction child.

```c
int child(void* arg)
{
  char c;
  //sethostname() sets the hostname to ELOUADI_KHADIM
  sethostname("ELOUADI_KHADIM",14);

  // changing the root directory of the child process to Host directory.
  chroot("./Host");

  //changing the current working directory of the child process to the root directory
  chdir("/");

  //Attaching the filesystem "proc " to "/proc" ,the third argument of mount refers to the type of the filesystem
  mount("proc", "/proc", "proc", 0, NULL);

  //this function execute the shell bash in the context of the child process
  execlp("/bin/bash", "/bin/bash" , NULL);

  return 1;
}
```

Les modifications apportées à la liste des points de montage ne sont (par défaut) visibles que pour les processus dans l'espace de noms de montage où réside le processus; les modifications ne sont pas visibles dans les autres espaces de noms de montage.

Cgroups

Les cgroups sont essentiellement des gestionnaires de ressources qui peuvent contrôler les ressources physiques suivantes dans le système par groupe de processus:

Configurer l'allocation des ressources:

•Consommation CPU

•Consommation de mémoire

•...

Limitation de la mémoire :

```
//cgcreate will create a new cgroup ELOUADI_KHADIM in memory controller with default values
system("sudo cgcreate -g memory:ELOUADI_KHADIM");

//Here we want to limit the memory that will be used by ELOUADI_KHADIM
system(" echo 20M > /sys/fs/cgroup/memory/ELOUADI_KHADIM/memory.limit_in_bytes");

//We add the child process to the cgroup ELOUADI_KHADIM
sprintf(bufMemory,"sudo echo %d  > /sys/fs/cgroup/memory/ELOUADI_KHADIM/cgroup.procs",pid);
system(bufMemory);
```

Limitation des nombres de processus à exécuter:

```
//cgcreate will create a new cgroup ELOUADI_KHADIM in PIDs controller with default values
system("sudo cgcreate -g pids:ELOUADI_KHADIM");

/* Here we want to limit the number of processes that can be launched from the shell bash
already executed in the context of isolated process  */

system("echo 2 > /sys/fs/cgroup/pids/ELOUADI_KHADIM/pids.max");

//finally we add the child process to the cgroup ELOUADI_KHADIM
sprintf(bufPid,"sudo echo %d  > /sys/fs/cgroup/pids/ELOUADI_KHADIM/cgroup.procs",pid);
system(bufPid);
```