CONCEPTES AVANÇATS DE SISTEMES OPERATIUS (CASO)

Facultat d'Informàtica de Barcelona, Dept. d'Arquitectura de Computadors, curs 2023/2024 – 2Q

Pràctiques de laboratori Contenidors (vers 1.01)

Material

La vostra instal·lació de Linux.

Els paquets de Linux Containers (lxc) i Docker (docker.io)

Algunes lectures interessants:

* ”Introducing runC: a lightweight universal container runtime”
  https://www.docker.com/blog/runc/
* ”Introducing Linux Network Namespaces”
  http://blog.scottlowe.org/2013/09/04/introducing-linux-network-namespaces/

Virtualització basada en contenidors

Com el seu nom indica, un contenidor (*container*) és una abstracció del sistema operatiu que engloba, protegeix i aïlla un conjunt de processos, de la resta del sistema. Com que en Linux els processos formen una jerarquia en la que els processos fills hereden algunes característiques del seu procés pare, la manera habitual d'implementar un contenidor és fent que un primer procés passi a formar part del nou contenidor i a partir d'ell, els seus descendents també en formen part. Aquesta és una manera d'implementar un entorn virtualitzat, on totes les màquines virtuals usen el mateix kernel de Linux.

La base de Linux necessària per tenir la implementació de contenidors és Linux Containers. Està basada en extensions a la crida *clone*, la que serveix per crear processos i fluxos. Aquestes extensions inclouen la possibilitat de tenir espais de noms diferents per:

- Jerarquia de Pids. Un procés pot començar un contenidor, creant un procés fill que inicïi una jerarquia de processos nova, amb pid=1. Aquesta característica és clau. Com que el seu pid és l'1, pot executar el procés `/sbin/init`, de forma compatible amb l'init original
- Directori arrel (chroot) i directori de treball (chdir)
- Punts de muntatge específics per cada contenidor. Un contenidor por tenir muntat el directori `/proc`, i un altre no
- Dispositius de xarxa, ports, filtres... és a dir, dispositius ethx particulars per cada contenidor
- Nom del host i del domini
- Usuaris i grups d’usuaris
- Comunicació entre processos de System V: semàfors, memòria compartida i missatges
- Comunicació entre processos de POSIX: cues de missatges
- ... i altres característiques...

Per obtenir més informació, veure la pàgina de manual de *namespaces* (*man namespaces*).

Docker (http://www.docker.com) és una eina basada en les mateixes extensions, que també permet gestionar contenidors. Docker també està disponible en macOS X i Windows 10, tot i que en aquests sistemes probablement no està implementat sobre contenidors, sinó sobre virtualització.

Linux Containers

Assegureu-vos que teniu els Linux Containers instal.lats en la vostra distribució de Linux:

```
$ lxc-checkconfig
```

Si no el teniu instal.lat, el podeu instal.lar fent un **`apt-get install lxc`**, o usant el gestor de paquets que tingui el vostre sistema.

En Ubuntu, lxc-checkconfig us hauria de donar com a resultat que el suport de Linux Containers és correcte.
Aquestes comandes de gestió dels contenidors és més còmode que les executeu com a **root (o amb sudo)**:

```
$ sudo su -
```

Creem el contenidor més senzill, amb només un intèrpret de comandes basat en busybox. És possible que hagueu d'instal.lar el busybox-static per tal que funcioni:

```
$ lxc-create -n busybox -t /usr/share/lxc/templates/lxc-busybox
```

Comprovem que el contenidor s'ha creat:

```
$ lxc-ls
```

L'engeguem:

```
$ lxc-start -n busybox
```

Amb aquesta comanda, l'hem posat en marxa en background i ara podem obrir un procés a dins, que executarà la comanda /bin/sh del busybox:

```
$ lxc-attach -n busybox

BusyBox...
Enter 'help'...

~#
```

Comproveu que aquest intèrpret de comandes corre en un entorn restringit i té unes característiques específiques:

- No té accés al sistema de fitxers arrel (root, /) del host. Comproveu que el conjunt de fitxers i directoris disponibles a l'arrel (/) és diferent.

- Té accés amb **chroot** a un subdirectori del sistema de fitxers (quin? Veure pregunta 7 més avall).

- Els punts de muntatge (donats per **mount**) són diferents.

- La llista de processos (**ps**) és diferent. En particular, dins el contenidor hi ha una (molt) petita jerarquia que comença amb un procés `init`, amb pid 1.

- Les llistes d'usuaris i grups són diferents (vegeu /etc/passwd i /etc/group).

Linux Containers també permet fer instal.lacions bàsiques de diverses distribucions. Podeu veure a `/usr/share/lxc/templates/lxc-*` algunes de les distribucions suportades en el vostre sistema.

Responeu a aquestes preguntes:

1) Podeu identificar els processos que corren a dins del contenidor i dir a quins corresponen dels de fora del contenidor? Proveu-ho amb la comanda "**ps**" executada a dins i a fora.

2) Quins pids de dins corresponen a quins pid's de fora?

3) Quins processos estan reflexats en el directori /proc/ de dins el contenidor i quins en el de fora?

Podem depurar des de fora un procés de dins? proveu-ho...

4) Proveu a restringir el conjunt de processadors en els quals poden córrer els processos del vostre contenidor. Pista: `man lxc-cgroup`

Quina comanda faríeu servir per permetre que els processos del contenidor només poguessin executar-se en els cores 0, 2, 4 i 6?

5) Quina comanda podeu usar des de fora del contenidor per comprovar que els processos només s'executen en els processadors que els heu assignat?

6) I quina comanda podeu usar des de dins del contenidor per veure la mateixa informació?

7) On té el directori arrel el contenidor busybox? (pista: /var/lib/lxc/...)

8) Quina comanda és en realitat el fitxer /sbin/init en aquest directori que heu trobat? (pista, buscar altres comandes que tinguin el mateix tamany dins del contenidor, o també a fora!!, o trobar la comanda apuntada pel *symbolic link*)

9) Quins usuaris i grups tenim dins el contenidor? (pista: /etc/passwd i /etc/group)

10) Si afegiu un nou usuari, podeu fer `su - usuari` i comprovar que funciona? Expliqueu-ho (pista: editeu el fitxer de passwd directament :)

Infrastructura bàsica per implementar els contenidors

La crida a sistema clone, vista a classe, permet iniciar jerarquies de processos que tornen a començar amb un pid=1, de forma que permeten l'execució de processos amb el binari de /sbin/init com a pare, que iniciarà una nova execució dels processos d'usuari per a la gestió d'una nova imatge del sistema. Aquest procés serà l'origen d'un nou contenidor.

Per a veure una demostració de la creació d'un contenidor, en el vostre Linux, i des de fora del contenidor, descarregueu l'exemple associat a aquesta pràctica: **containers.tar.xz**. Desempaqueteu el fitxer, entreu en el directori *containers* i compileu l'exemple `container-test`.

Primer mireu el llistat del fitxer container-test.c i enteneu el què fa. Després, executeu:

```
  $ ./container-test     # hauríeu de veure la següent sortida
 -------------   fork test   --------------
Parent: parent 4089, child 4090, waiting...
child: parent pid 4089, child 4090            # jerarquia normal
Parent: finishing
 -------------  clone test   --------------
Parent: parent 4089, child 4091, waiting...
child: parent pid 0, child 1    # el fill es pensa que és l'init (1)!
Parent: finishing               # I el seu pare és el sistema(0)!!!
```

A continuació mireu el codi del programa `container.c` . Fa el següent:

- crea un nou contenidor, seguint els paràmetres que li podeu passar a la línia de comandes:
  1- l'arrel del sistema de fitxers, pel qual usarem el contenidor que hem creat abans
     `/var/lib/lxc/busybox/rootfs`
  2- el procés a executar com a "init" del nou contenidor, podrem usar
     `/bin/sh`
     o altres comandes que hi hagi a dins del contenidor

11) En el fitxer "container.c" hi ha diversos comentaris en català, indicant les parts del codi que heu de completar. Feu-ho, abans de poder provar els següents exemples.

Per exemple, un cop l'hagueu completat, si l'executem així, des de l'usuari root:

```
$ ./container /var/lib/lxc/busybox/rootfs /bin/busybox
```

Ens ha d'imprimir l'ajuda del busybox. I si ho fem així:

```
$ ./container /var/lib/lxc/busybox/rootfs /bin/sh
```

Ens ha d'obrir un intèrpret de comandes (/bin/sh) en el contenidor creat.

Proveu:

```
$ ./container /var/lib/lxc/busybox/rootfs /sbin/init
```

12) Comproveu si la instal·lació del container s'engega correctament o no.

No feu cas a l'error d'udhcp. Busybox engega per defecte dhcp, sense cap xarxa configurada. Podeu comentar la línia al fitxer rootfs/etc/init.d/rcS

Docker

Docker és una infrastructura que permet distribuir contenidors, amb la instal.lació de Linux i aplicacions que es vulgui, de forma independent del kernel de Linux que s'estigui usant. Hi ha el requeriment que el contenidor podrà executar-se en un sistema, si aquest té el suport de Linux Containers activat (si lxc-checkconf dóna resultats positius).

Instal.leu Docker, segurament us farà falta afegir el paquet docker.io:

```
$ sudo apt-get install docker.io
```

Podeu baixar-vos una imatge bàsica d'ubuntu, fent:

```
$ docker run -i -t ubuntu
```

Aquesta comanda buscarà un repositori de docker d'on baixar la imatge, la instal.larà i l'executarà (executarà el seu `/bin/bash`) i us donarà aquest intèrpret de comandes.

Podeu trobar les imatges que ha fet altra gent, donant-li un patró a buscar a la subcomanda search del docker:

```
$ docker search <pattern>
```

D'aquesta manera podeu provar les instal.lacions bàsiques d'altres distribucions, com Fedora:

```
$ docker run -i -t fedora /bin/bash
```

Entregueu

Les respostes a les preguntes 1 a 10 i la 12 i el fitxer "container.c" que heu completat a la pregunta 11.