Sistemas Operativos

Práctica 4A - cgroups & namespaces

Parte 1: Conceptos teóricos

1. Defina virtualización. Investigue cuál fue la primera implementación que se realizó.

Es una técnica que permite ejecutar múltiples sistemas operativos (SO) o entornos virtuales en una misma máquina física, utilizando recursos compartidos.
Se usa para abstraerse de los recursos de la computadora, desacoplar el hardware físico del sistema

Primera implementación:
IBM en los años 60, con el sistema CP-40, seguido por VM/370, permitiendo múltiples entornos virtuales en mainframes.

2. ¿Qué diferencia existe entre virtualización y emulación?

Virtualización: Ejecuta SOs sobre hardware real, aprovechando instrucciones nativas. La arquitectura tiene que ser la misma

Emulación: Simula un hardware distinto del real, permitiendo correr software incompatible, pero con menor rendimiento.

3. Investigue el concepto de hypervisor y responda:

(a) ¿Qué es un hypervisor?

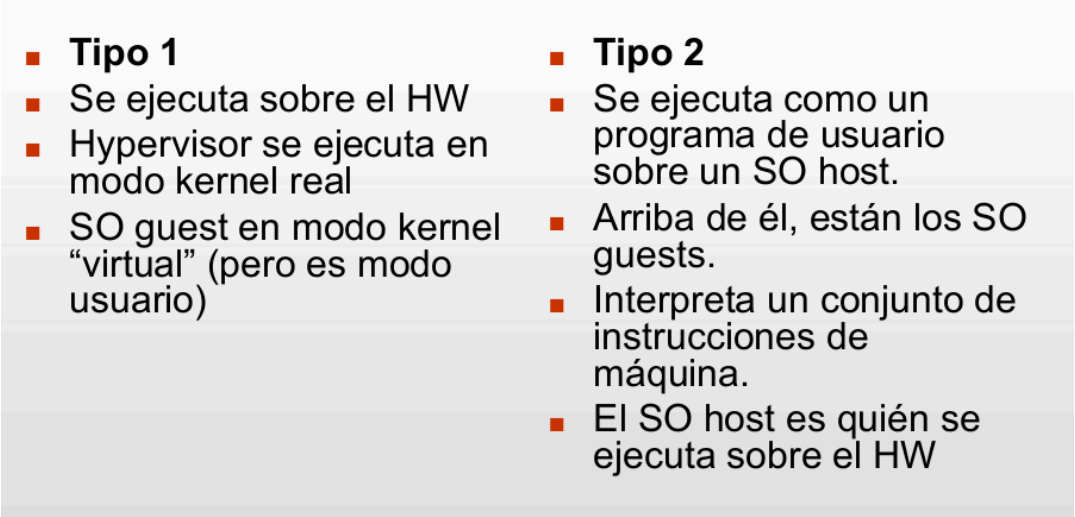
Es una capa de software que permite crear y gestionar máquinas virtuales (VMs) compartiendo los recursos físicos. separa las “aplicaciones/SO” del hardware subyacente

(b) ¿Qué beneficios traen los hypervisors? ¿Cómo se clasifican?

Beneficios: Aislamiento, eficiencia en uso de recursos, facilidad de administración.

Tipo 1 (bare-metal): Corre directamente sobre el hardware (ej. VMware ESXi, Xen).

Tipo 2 (hosted): Corre sobre un SO (ej. VirtualBox, VMware Workstation).



4. ¿Qué es la full virtualization? ¿Y la virtualización asistida por hardware?

Full virtualization: Emula completamente el hardware para el SO invitado, sin necesidad de modificaciones. particionar un procesador físico en distintos contextos, donde cada uno de ellos corre sobre el mismo procesador. Los SO guest deben ejecutar la misma arquitectura de hardware sobre la que corren

Virtualización asistida por hardware: Utiliza características específicas del procesador (Intel VT-x, AMD-V) para mejorar el rendimiento de la virtualización. permite que el hypervisor delegue directamente al hardware ciertas operaciones sensibles (como acceso a privilegios).
Esto elimina la necesidad de usar binary translation, ya que el procesador puede manejar "traps" de forma más eficiente.

5. ¿Qué implica la técnica binary translation? ¿Y trap-and-emulate?
Binary translation:
Es una técnica activa y dinámica.
El hypervisor analiza y traduce instrucciones “sensibles” del sistema operativo invitado antes de que se ejecuten.
En lugar de dejar que se ejecuten directamente, las reemplaza por otras instrucciones que llaman explícitamente al hypervisor.
Esta traducción se hace en tiempo de ejecución, y se guarda en caché para mayor rendimiento.
Ventaja: Permite ejecutar sistemas operativos sin modificar, incluso en hardware que no soporta virtualización.

Trap-and-emulate:
Es una técnica reactiva.
El SO se ejecuta directamente sobre el hardware hasta que intenta ejecutar una instrucción privilegiada no permitida.
Esa instrucción genera un trap (interrupción o excepción), que el hypervisor intercepta.
Luego, el hypervisor emula lo que esa instrucción haría, pero de forma segura.

Diferencia principal:
Binary Translation: Reescribe instrucciones antes de ejecutarlas.

Trap-and-Emulate: Deja ejecutar, pero si se intenta algo prohibido, lo atrapa y lo emula.

6. Investigue el concepto de paravirtualización y responda:
(a) ¿Qué es la paravirtualización?
Es una técnica donde el SO invitado es modificado para colaborar con el hypervisor, evitando instrucciones problemáticas.En lugar de intentar ejecutar directamente instrucciones privilegiadas del hardware, el SO modificado utiliza llamadas especiales al hypervisor (llamadas hypercalls) para solicitar esos servicios.

(b) Mencione algún sistema que implemente paravirtualización.
ue par
(c) ¿Qué beneficios trae con respecto al resto de los modos de virtualización?
Mejor rendimiento que la full virtualization en sistemas sin soporte de hardware.

Menor sobrecarga al no requerir emulación completa.

7. Investigue sobre containers y responda:
(a) ¿Qué son?
Son entornos aislados que comparten el mismo kernel del SO pero ejecutan aplicaciones como si fueran independientes.
Una forma de empaquetar aplicaciones
Una máquina virtual liviana
Un conjunto de procesos aislados del resto
Autocontenidos: tiene todo lo que necesita para funcionar
Aislados: mínima influencia en el nodo y otros contenedores
Independientes: administraci´on de un contenedor no afecta al resto
Portables: desacoplados del entorno en el que ejecutan. Pueden ejecutar de igual manera en diferentes entornos.

(b) ¿Dependen del hardware subyacente?
No directamente. Dependen del kernel del SO, no del hardware como tal.

(c) ¿Qué lo diferencia por sobre el resto de las tecnologías estudiadas?
Son más ligeros y rápidos que las máquinas virtuales.

No necesitan hypervisor ni SO completo por VM.

(d) Investigue qué funcionalidades son necesarias para poder implementar containers.
Namespaces:aislamiento entre procesos, cada contenedor con entorno separado. Lo que pasa en un contenedor no afecta al resto

cgroups (control de recursos): limitar y monitorizar el uso de recursos, permite que un contenedor no consuma más de lo permitido

UnionFS (sistema de archivos): es en capas, los contenedores comparten partes comunes (imagen) y solamente almacenan los cambios realizados. La imagen no se modifica, lo que se modifica hay que guardarlo explícitamente

soporte del kernel para estas tecnologías (ej. Docker usa estas funciones en Linux).

Parte 2: chroot, Control Groups y Namespaces

Debido a que para la realización de la práctica es necesario tener más de una terminal abierta simultáneamente tenga en cuenta la posibilidad de lograr esto mediante alguna alternativa (ssh, terminales gráficas, etc.)

Chroot

En algunos casos suele ser conveniente restringir la cantidad de información a la que un

proceso puede acceder. Uno de los métodos más simples para aislar servicios es chroot, que consiste simplemente en cambiar lo que un proceso, junto con sus hijos, consideran que es el directorio raíz, limitando de esta forma lo que pueden ver en el sistema de archivos. En esta sección de la práctica se preparará un árbol de directorios que sirva como directorio raíz para la ejecución de una shell.

1. ¿Qué es el comando chroot? ¿Cuál es su finalidad?

Hace creer a un proceso que / es una carpeta, permite ejecutar procesos de forma “aislada”

2. Crear un subdirectorio llamado sobash dentro del directorio root. Intente ejecutar el comando chroot /root/sobash. ¿Cuál es el resultado? ¿Por qué se obtiene ese resultado?

```
capi@capi-Aspire-S5-371T:~$ sudo chroot /root/sobash
chroot: failed to run command ‘/bin/bash’: No such file or directory
```

El comando chroot cambia el directorio raíz del entorno de ejecución de un proceso. Sin embargo, al hacerlo, espera encontrar un sistema mínimo funcional dentro del nuevo directorio raíz. En particular, busca un intérprete de comandos (/bin/sh) para ejecutarlo por defecto. Como /root/sobash está vacío, no contiene ni /bin/sh ni las bibliotecas necesarias para que se ejecute, por eso falla.

3. Cree la siguiente jerarquía de directorios dentro de sobash:

sobash/
├── bin
├── lib
│
└── x86_64-linux-gnu
└── lib64

```
capi@capi-Aspire-S5-371T:~$ sudo mkdir -p /root/sobash/bin
capi@capi-Aspire-S5-371T:~$ sudo mkdir -p /root/sobash/lib
capi@capi-Aspire-S5-371T:~$ sudo mkdir -p /root/sobash/lib/x86_64-linux-gnu
capi@capi-Aspire-S5-371T:~$ sudo mkdir -p /root/sobash/lib64
```

4. Verifique qué bibliotecas compartidas utiliza el binario /bin/bash usando el comando ldd /bin/bash.¿En qué directorio se encuentra linux-vdso.so.1? ¿Por qué?

```
capi@capi-Aspire-S5-371T:~$ ldd /bin/bash
        linux-vdso.so.1 (0x00007ffd6b8d6000)
        libtinfo.so.6 => /lib/x86_64-linux-gnu/libtinfo.so.6 (0x00007dc373f55000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007dc373c00000)
        /lib64/ld-linux-x86-64.so.2 (0x00007dc37410d000)
```

no tiene una ruta de archivo, solo una dirección de memoria: (0x00007ffd6b8d6000).

Esto confirma que no está en el sistema de archivos físico, sino que es mapeado directamente por el kernel en el espacio de direcciones del proceso.

5. Copie en /root/sobash el programa /bin/bash y todas las librerías utilizadas por el programa bash en los directorios correspondientes. Ejecute nuevamente el comando chroot ¿Qué sucede ahora?

```
capi@capi-Aspire-S5-371T:~$ sudo cp /bin/bash /root/sobash/bin/
capi@capi-Aspire-S5-371T:~$ sudo cp /lib/x86_64-linux-gnu/libtinfo.so.6 /root/sobash/lib/x86_64-linux-gnu/
sudo cp /lib/x86_64-linux-gnu/libc.so.6 /root/sobash/lib/x86_64-linux-gnu/
sudo mkdir -p /root/sobash/lib64
sudo cp /lib64/ld-linux-x86-64.so.2 /root/sobash/lib64/
capi@capi-Aspire-S5-371T:~$ sudo chroot /root/sobash /bin/bash
bash-5.2#
```

Copie /bin/bash y las 3 librerías que el comando de antes me dijo que necesitaba. AHora estoy parada en un entorno simulado que empieza en /root/sobash y no tiene acceso al resto del sistema.

6. ¿Puede ejecutar los comandos cd "directorio" o echo? ¿Y el comando ls? ¿A qué se debe esto?

cd y echo si que andan, ls no. Los primeros dos son comandos internos de la shell (builtin). Los tiene bash por default. ls no, es un programa que suele estar en /bin/ls. nosotros no lo copiamos así que no anda.

7. ¿Qué muestra el comando pwd? ¿A qué se debe esto?

```
bash-5.2# pwd
/
```

EL tema es que con chroot /root/sobash le dijimos al sistema que esa ruta se comporte como si fuera /. Entonces a partir de ahora la raiz del sistema de archivos es la nueva.

8. Salir del entorno chroot usando exit

ok

9. Usando el repositorio de la cátedra acceda a los materiales en practica4/02-chroot:

a. Verifique que tiene instalado busybox en /bin/busybox

ok

b. Cree un chroot con busybox usando /buildbusyboxroot.sh

```
capi@capi-Aspire-S5-371T:~/codigo-para-practicas/practica4/02-chroot$ sudo ./buildbusyboxroot.sh
        linux-vdso.so.1 (0x00007ffcf356a000)
        libresolv.so.2 => /lib/x86_64-linux-gnu/libresolv.so.2 (0x000079a421f38000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x000079a421c00000)
        /lib64/ld-linux-x86-64.so.2 (0x000079a422032000)
BusyBox root filesystem created in /home/capi/codigo-para-practicas/practica4/02-chroot/busyboxroot
You can now chroot into it with:
chroot /home/capi/codigo-para-practicas/practica4/02-chroot/busyboxroot /bin/sh
```

c. Entre en el chroot

```
capi@capi-Aspire-S5-371T:~/codigo-para-practicas/practica4/02-chroot$ sudo chroot /home/capi/codigo-para-practicas/practica
4/02-chroot/busyboxroot /bin/sh


BusyBox v1.36.1 (Ubuntu 1:1.36.1-6ubuntu3.1) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ #
```

d. Busque el directorio /home/so ¿Qué sucede? ¿Por qué?

No ta, porque no lo cargamos en el chroot. Para tenerlo deberíamos haberlo copiado a mano

e. Ejecute el comando “ps aux” ¿Qué procesos ve? ¿Por qué (pista: ver el contenido de /proc)?

ps aux se usa para ver los procesos en ejecución en el sistema.

No veo nada porque /proc no tiene nada /proc es un sistema de archivos virtua que da info sobre los procesos en ejecución. SIn montar /proc no veo los procesos del sistema real. Igual en teoría debería ver algo, tipo lo que hay dentro del entorno este pero no me muestra nada

f. Monte /proc con “mount -t proc proc /proc” y vuelva a ejecutar “ps aux” ¿Qué procesos ve? ¿Por qué?

Se ven un montonazo de procesos, todos los del sistema posta.

Montar /proc dentro de un entorno chroot implica hacer que el sistema de archivos virtual /proc sea accesible dentro del entorno aislado del chroot. /proc es un sistema de archivos especial en Linux que contiene información sobre los procesos en ejecución, la configuración del sistema, el hardware, y muchos otros aspectos del sistema operativo.

g. Acceda a /proc/1/root/home/so ¿Qué sucede?

Bueno no tengo /home/so, pero con el /proc montado pedo entrar al luar porque /proc/1/root apuntará a la raíz del sistema real, y entonces llego a donde quiera./proc/1/root es un enlace simbólico que refleja el sistema de archivos "root" del proceso con PID 1.

h. ¿Qué conclusiones puede sacar sobre el nivel de aislamiento provisto por chroot?

Es medio una verga jaja salu2

En resumen, **`chroot`** crea un entorno aislado al cambiar la raíz del sistema de archivos, pero **no ofrece aislamiento completo**. Aquí están las conclusiones clave:

1. **Aislamiento parcial**: Cambia la raíz del sistema de archivos, pero no aísla completamente procesos, redes o dispositivos.
2. **Dependencia de recursos del sistema real**: Algunas rutas, como `/proc/1/root`, siguen apuntando al sistema real, lo que limita el aislamiento.
3. **No es seguro por sí solo**: Un usuario con privilegios puede escapar fácilmente del `chroot`.
4. **Útil para pruebas o entornos controlados**: Es útil para simular un sistema de archivos separado, pero no es una solución robusta para seguridad o aislamiento total.

Conclusión: `chroot` proporciona un aislamiento básico, pero no es adecuado para entornos que requieren seguridad o un aislamiento completo. Para eso, se prefieren contenedores o máquinas virtuales.

Control Groups

Se aconseja realizar esta parte de la práctica en una máquina virtual (por ejemplo en la provista por la práctica) ya que es necesario cambiar la configuración de CGroups.

Preparación: Actualmente Debian y la mayoría de las distribuciones usan CGroups 2 por defecto, pero para esta práctica usaremos CGroups 1. Para esto es necesario cambiar un parámetro de arranque del sistema en grub:

1. Editar /etc/default/grub:

Cambiar:

GRUB_CMDLINE_LINUX_DEFAULT="quiet"

Por:

GRUB_CMDLINE_LINUX_DEFAULT="quiet systemd.unified_cgroup_hierarchy=0"

ok

2. Actualizar la configuración de GRUB:

sudo update-grub
ok
3. Reiniciar la máquina.
ok (no vuelve a abrir la practica)
4. Verificar que se esté usando CGroups 1. Para esto basta con hacer “ls /sys/fs/cgroup/” Se deberían ver varios subdirectorios como cpu, memory, blkio, etc. (en vez de todo montado de forma unificada).

```
so@so:~$ ls /sys/fs/cgroup/
blkio  cpu  cpuacct  cpu,cpuacct  devices  freezer  hugetlb  misc  net_cls  net_cls,net_prio  net_prio  perf_event  pids
so@so:~$
```

A continuación se probará el uso de cgroups. Para eso se crearán dos procesos que compartirán una misma CPU y cada uno la tendrá asignada un tiempo determinado.
Nota: es posible que para ejecutar xterm tenga que instalar un gestor de ventanas. Esto puede hacer con apt-get install xterm.
1. ¿Dónde se encuentran montados los cgroups? ¿Qué versiones están disponibles?
están montados en /sys/fs/cgroup/

```
Procesando disparadores para mailcap (3.70+nmu1) ...
root@so:/home/so# mount | grep cgroup
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,size=4096k,nr_inodes=1024,mode=755,inode64)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/misc type cgroup (rw,nosuid,nodev,noexec,relatime,misc)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
root@so:/home/so# _
```

2. ¿Existe algún controlador disponible en cgroups v2? ¿Cómo puede determinarlo?
mount | grep cgroup
cgroup on /sys/fs/cgroup/cpu type cgroup (rw,relatime,cpu)

Creo que estamos usando los dos pq en la salida de arriba nos marca tanto cgroup1 como 2
3. Analice qué sucede si se remueve un controlador de cgroups v1 (por ej. Umount /sys/fs/cgroup/rdma).
Al desmontar, el subsistema rdma desaparece de /sys/fs/cgroup,queda un directorio vacio /sys/fs/cgroup, no tiene archivos virtuales y el kernel no lo reconoce como un controlador y ya no podés asignar recursos relacionados a rdma. No afecta otros controladores como cpu o memory. Es una forma de "desactivar" ese controlador temporalmente.
4. Crear dos cgroups dentro del subsistema cpu llamados cpualta y cpubaja. Controlar que se hayan creado tales directorios y ver si tienen algún contenido
mkdir /sys/fs/cgroup/cpu/"nombre_cgroup"

```
root@so:/sys/fs/cgroup/rdma# cd /sys/fs/cgroup/cpu/cpubaja
root@so:/sys/fs/cgroup/cpu/cpubaja# ls
cgroup.clone_children  cpuacct.usage         cpuacct.usage_percpu_sys   cpuacct.usage_user  cpu.cfs_quota_us  cpu.stat
cgroup.procs           cpuacct.usage_all     cpuacct.usage_percpu_user  cpu.cfs_burst_us    cpu.idle          cpu.stat.l
cpuacct.stat           cpuacct.usage_percpu  cpuacct.usage_sys          cpu.cfs_period_us   cpu.shares        notify_on_
root@so:/sys/fs/cgroup/cpu/cpubaja# _
```

5. En base a lo realizado, ¿qué versión de cgroup se está utilizando?
Pareciera ser v1 pq en v2 no hay subdirectorios por controlador sino que está todo junto en /unified

6. Indicar a cada uno de los cgroups creados en el paso anterior el porcentaje máximo de CPU que cada uno puede utilizar. El valor de cpu.shares en cada cgroup es 1024. El cgroup cpualta recibirá el 70 % de CPU y cpubaja el 30 %.
echo 717 > /sys/fs/cgroup/cpu/cpualta/cpu.shares
echo 307 > /sys/fs/cgroup/cpu/cpubaja/cpu.shares

```
root@so:/home/so# echo 717 > /sys/fs/cgroup/cpu/cpualta/cpu.shares
root@so:/home/so# echo 307 > /sys/fs/cgroup/cpu/cpubaja/cpu.shares
```

Bue

7. Iniciar dos sesiones por ssh a la VM.(Se necesitan dos terminales, por lo cual, también podría ser realizado con dos terminales en un entorno gráfico). Referenciaremos a una terminal como termalta y a la otra, termbaja.

Bue

8. Usando el comando taskset, que permite ligar un proceso a un core en particular, se iniciará el siguiente proceso en background. Uno en cada terminal. Observar el PID asignado al proceso que es el valor de la columna 2 de la salida del comando.
taskset -c 0 md5sum /dev/urandom &

```
Last login: Sun May 11 15:11:32 2025
so@so:~$ taskset -c 0 md5sum /dev/urandom &
[1] 1140
so@so:~$ _
```

```
root@so:/sys/fs/cgroup/cpu/cpubaja# taskset -c 1 md5sum /dev/urandom &
[1] 1153
root@so:/sys/fs/cgroup/cpu/cpubaja# _
```

9. Observar el uso de la CPU por cada uno de los procesos generados (con el comando top en otra terminal). ¿Qué porcentaje de CPU obtiene cada uno aproximadamente?

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1153 root      20   0    5476   1824   1696 R  99,7   0,1   1:15.33 md5sum
 1140 so        20   0    5476   1844   1716 R  99,3   0,1   3:02.71 md5sum
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  854 root      20   0    5476   1820   1692 R  50,2   0,1   2:59.92 md5sum
  857 so        20   0    5476   1900   1772 R  49,8   0,1   1:35.48 md5sum
```

10. En cada una de las terminales agregar el proceso generado en el paso anterior a uno de los cgroup (termalta agregarla en el cgroup cpualta, termbaja en cpubaja. El process_pid es el que obtuvieron después de ejecutar el comando taskset)
echo "process_pid" > /sys/fs/cgroup/cpu/cpualta/cgroup.procs

Bue

```
root@so:/home/so# echo 854 > /sys/fs/cgroup/cpu/cpualta/cpu.shares
root@so:/home/so# echo 857 > /sys/fs/cgroup/cpu/cpubaja/cpu.shares
```

11. Desde otra terminal observar cómo se comporta el uso de la CPU. ¿Qué porcentaje de CPU recibe cada uno de los procesos?

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  854 root      20   0    5476   1820   1692 R  70,0   0,1   9:21.49 md5sum
  857 so        20   0    5476   1900   1772 R  30,0   0,1   7:53.49 md5sum
```

12. En termalta, eliminar el job creado (con el comando jobs ven los trabajos, con kill %1 lo eliminan. No se olviden del %.). ¿Qué sucede con el uso de la CPU?

se lo adueña la cpubaja

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  857 so        20   0    5476   1900   1772 R  99,7   0,1   9:49.14 md5sum
```

13. Finalizar el otro proceso md5sum.

Bue

14. En este paso se agregarán a los cgroups creados los PIDs de las terminales (Importante: si se tienen que agregar los PID desde afuera de la terminal ejecute el comando echo $$ dentro de la terminal para conocer el PID a agregar. Se debe agregar el PID del shell ejecutando en la terminal).

echo $$ > /sys/fs/cgroup/cpu/cpualta/cgroup.procs (termalta)

echo $$ > /sys/fs/cgroup/cpu/cpubaja/cgroup.procs (termbaja)

ok

15. Ejecutar nuevamente el comando taskset -c 0 md5sum /dev/urandom & en cada una de las terminales. ¿Qué sucede con el uso de la CPU? ¿Por qué?

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  933 root      20   0    5476   1824   1696 R  58,7   0,1   0:44.66 md5sum
  932 root      20   0    5476   1820   1692 R  41,1   0,1   1:09.21 md5sum
```

Los procesos fueron creados desde terminales que ya estaban afectadas por un cgroup, produciendo que los procesos creados en ellos conservan el cgroup de la terminal

16. Si en termbaja ejecuta el comando: taskset -c 0md5sum /dev/urandom & (deben quedar 3 comandos md5 ejecutando a la vez, 2 en el termbaja). ¿Qué sucede con el uso de la CPU? ¿Por qué?

```
  933 root      20   0    5476   1824   1696 R  61,3   0,1   3:21.05 md5sum
  950 root      20   0    5476   1908   1780 R  19,7   0,1   0:08.97 md5sum
  932 root      20   0    5476   1820   1692 R  19,3   0,1   2:50.21 md5sum
```

Como fueron creados dos procesos dentro de una misma terminal, se comparte la utilizacion dentro del limite asignado por el cgroup, es decir agarran el limite asignado por el cgroup y se divide por la cantidad de procesos que comparten ese cgroup

Namespaces

1. Explique el concepto de namespaces.

Permite abstraer un recurso global del sistema para que los procesos dentro de ese “namespace” piensen que tienen su propia instancia aislada de ese recurso global
Limitan lo que un proceso puede ver y, en consecuencia, lo que puede usar.

2. ¿Cuáles son los posibles namespaces disponibles?

IPC,Network,Mount,PID,User,UTS,Cgroup,Time,DEMO

3. ¿Cuáles son los namespaces de tipo Net, IPC y UTS una vez que inicie el sistema (los que se iniciaron la ejecutar la VM de la cátedra)?
sudo lsns | grep -E "net|ipc|uts"

```
root@so:/home/so# sudo lsns | grep -E "net|ipc|uts"
4026531838 uts         118     1 root             /sbin/init
4026531839 ipc         121     1 root             /sbin/init
4026531840 net         121     1 root             /sbin/init
4026532164 uts           1   309 root             ├─/lib/systemd/systemd-udevd
4026532239 uts           1   333 systemd-timesync ├─/lib/systemd/systemd-timesyncd
4026532295 uts           1   616 root             └─/lib/systemd/systemd-logind
root@so:/home/so#
```

4. ¿Cuáles son los namespaces del proceso cron? Compare los namespaces net, ipc y uts con los del punto anterior, ¿son iguales o diferentes?

```
root@so:/home/so# ls -l /proc/588/ns/ | grep -E "net|ipc|uts"
lrwxrwxrwx 1 root root 0 may 11 16:18 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 root root 0 may 11 16:18 net -> net:[4026531840]
lrwxrwxrwx 1 root root 0 may 11 16:18 uts -> uts:[4026531838]
```

El proceso cron en sistemas basados en Linux es el encargado de ejecutar tareas programadas.
ES IGUAL

5. Usando el comando unshare crear un nuevo namespace de tipo UTS.
a. unshare –uts sh (son dos (- -) guiones juntos antes de uts)
adas.
b. ¿Cuál es el nombre del host en el nuevo namespace? (comando hostname)

el mismo que teníamos antes (so en este caso)

c. Ejecutar el comando lsns. ¿Qué puede ver con respecto a los namespace?.

```
4026532240 uts         5  1546 root             sh
# lsns | grep -E "net|ipc|uts"
4026531838 uts       114     1 root             /sbin/init
4026531839 ipc       120     1 root             /sbin/init
4026531840 net       120     1 root             /sbin/init
4026532164 uts         1   309 root             ├─/lib/systemd/systemd-udevd
4026532239 uts         1   333 systemd-timesync ├─/lib/systemd/systemd-timesyncd
4026532295 uts         1   616 root             └─/lib/systemd/systemd-logind
4026532240 uts         3  1546 root             sh
#
```

ES IGUAL– hay un uts nuevo que es el que creamos recién c:

d. Modificar el nombre del host en el nuevo hostname.

```
# hostname pepe
# hostname
pepe
```

e. Abrir otra sesión, ¿cuál es el nombre del host anfitrión?

```
so@so:~$ hostname
so
```

f. Salir del namespace (exit). ¿Qué sucedió con el nombre del host anfitrión?

```
# hostname
pepe
# exit
root@so:/home/so# hostname
so
```

6. Usando el comando unshare crear un nuevo namespace de tipo Net.

a. unshare –pid sh

ok

b. ¿Cuál es el PID del proceso sh en el namespace? ¿Y en el host anfitrión?

son el mismo (69)

c. Ayuda: los PIDs son iguales. Esto se debe a que en el nuevo namespace se sigue viendo el comando ps sigue viendo el /proc del host anfitrión. Para evitar esto (y lograr un comportamiento como los contenedores), ejecutar:

unshare --pid --fork --mount-proc

DEMENCIAAAA

d. En el nuevo namespace ejecutar ps -ef. ¿Qué sucede ahora?

DEMENCIAAAA

e. Salir del namespace

ok