Práctica: Iniciando LKM con el sistema

Introducción

Una de las principales características de los LKM es que pueden ser cargados y descargados del kernel bajo demanda. Pero eso implica que, se deben estar ejecutando los comandos definidos para dichas funciones. Existe la posibilidad de configurar el sistema operativo Linux para que, de manera automatizada, se carguen los LKM en el kernel y puedan ser utilizados sin necesidad de hacerlo de forma manual.

Resultados de aprendizaje.

- Entender cómo el sistema operativo Linux maneja los archivos de arranque
- Crear scripts para cargar el LKM al kernel
- Configurar la ejecución de servicios en el inicio del sistema
- Entender la diferencia entre init y systemd

Marco Teórico.

En la actualidad, el inicio Linux se puede encontrar en 3 variantes:

INIT

En los sistemas tipo Unix, init (abreviatura de initialization) es el primer proceso en ejecución tras la carga del kernel y el que a su vez genera todos los demás procesos. Se ejecuta como demonio y por lo general tiene PID 1.

Tradicionalmente, esta funcionalidad se ha implementado de forma distinta en los dos grandes universos Unix: System V y BSD. En el Unix original, el proceso init arrancaba los servicios de mediante un único script denominado /etc/rc. Posteriormente, la versión System V del Unix de AT&T introdujo un nuevo esquema de directorios en /etc/rc.d/ que contenía scripts de arranque/parada de servicios.

Linux adoptó el esquema System V, aunque algunas distribuciones, como Slackware, usan el estilo BSD y otros, como Gentoo, tienen su propia versión personalizada. Ubuntu y algunas otras distribuciones de Linux utilizan ahora upstart como reemplazo para el proceso de inicialización tradicionales y el motivo son las desventajas que posee Init en comparación con Upstart:

El demonio init tradicional es estrictamente síncrono, bloqueando futuras tareas hasta que la actual se haya completado. Sus tareas deben ser definidas por adelantado, y solo pueden ser ejecutadas cuando el demonio init cambia de estado (cuando la máquina se arranca o se apaga). Esto hace que no sea capaz de manejar de forma elegante varias tareas en computadoras de escritorio modernas, incluyendo:

- La conexión o desconexión de una memoria USB y otros medios de almacenamiento portables / dispositivos de red mientras la máquina está arrancada.

- El descubrimiento y exploración de nuevos dispositivos de almacenamiento, sin bloquear el sistema, especialmente cuando un disco puede no encenderse hasta que este es explorado.
- La carga de firmware para un dispositivo, lo cual podría tener que realizarse después de que sea detectado, pero antes de que sea utilizable.

UPSTART

En informática, Upstart es un reemplazo basado en eventos para el demonio init, el método utilizado por varios sistemas operativos Unix-like para realizar tareas durante el arranque del sistema. Fue programado por Scott James Remnant, un antiguo trabajador de Canonical Ltd.

Mientras que Init presentaba las desventajas mencionadas anteriormente, Upstart trabaja de forma asíncrona supervisando las tareas mientras el sistema esta arrancado. También gestiona las tareas y servicios de inicio cuando el sistema arranca y los detiene cuando el sistema se apaga.

La fácil transición y la perfecta retrocompatibilidad con sysvinit fueron objetivos explícitos en el diseño. Por lo tanto, Upstart es capaz de ejecutar scripts de sysvinit sin modificaciones. De esta manera se diferencia de la mayoría de los reemplazos de init, que normalmente requieren una transición completa para funcionar correctamente y no son compatibles con un entorno mixto formado por métodos de arranque tradicionales y nuevos.

A medida que Upstart madura, se pretende que sus funciones se extiendan a las tareas gestionadas por cron, anacron, el demonio del comando at (atd) y posiblemente (pero mucho menos probable) inetd.

SYSTEMD

Systemd es un sustituto para el Init de Linux. Está hecho para proveer un mejor framework para expresar las dependencias del servicio, permite hacer más trabajo paralelamente al inicio del sistema y reducir la sobrecarga del shell.
El nombre viene del sufijo system daemon (procesos en segundo plano) con la letra "d".
Systemd fue escrito por Lennart Poettering y publicado como software libre bajo los términos de la GNU Lesser General Public License versión 2.1 ó posterior.

Comparado con System V init, que es utilizado por la mayoría de las distribuciones anteriores, systemd puede tomar ventaja de nuevas técnicas:

Los servicios de activación de sockets y la activación de buses, que conduce a una mejor paralelización de servicios independientes.
cgroups se utilizan para realizar un seguimiento de los procesos de servicio, en lugar de PIDs. Esto significa que los demonios no pueden "escapar" de systemd aunque estén doblemente-bifurcados.
systemd es sólo para Linux por diseño, ya que depende de características como cgroups y fanotify.

systemd ha sido propuesto por Poettering como dependencia externa para GNOME 3.2. Esto esencialmente requeriría que todas las distribuciones que usan GNOME utilicen systemd, o al menos lo incluyan como opción configurable.

Distribuciones en las que systemd está habilitado de forma predeterminada:

- Fedora 15 y superior, Frugalware 1.5 y superior, Mageia desde la versión 2, Mandriva 2011, openSUSE 12.1 y superior, Arch Linux desde octubre de 2012 y Siduction desde diciembre de 2013.

Distribuciones en las que systemd está disponible:

- Debian GNU/Linux tiene paquetes para systemd.
- Gentoo ofrece paquetes systemd como elementos no soportados oficialmente.
- Debian GNU/Linux y Ubuntu han decidido utilizar systemd de forma predeterminada en sus próximas versiones.

Desarrollo.

En esta práctica se cubrirán dos formas en las que se puede configurar el sistema para que cargue el LKM de forma automática al kernel.

1era forma – Modificando el archivo `/etc/modules`

El archivo modules, que se encuentra en el directorio etc, contiene los nombres de los LKM que van a ser cargados al momento del inicio del sistema, están detallados uno por cada línea; si algún módulo necesita argumentos, se establecen en la misma línea. Las líneas que comienzan con numeral (#) no se ejecutan, se toman como comentarios.

1. Crear un LKM. Pueden usar el "Hola Mundo".

2. Después de crear el LKM, copiarlo en la carpeta **`misc`** de la siguiente ruta **`/lib/modules/(uname -r)/kernel/drivers/misc`**

3. Al hacer la copia, es necesario ejecutar el comando **`depmod -a`** para actualizar el mapa de módulos de sistema que se encuentra en **`/lib/modules/(uname -r)`**

4. Probar que se cargue el LKM con el comando **`modprobe`**.

5. Una vez se verifique que aparece el módulo cargado, descargarlo con el comando **`modprobe`**

6. Ahora que, el LKM se encuentra registrado de forma global en el sistema, es necesario editar el archivo **`/etc/modules`**. Al final del archivo, basta con agregar el nombre del módulo que se quiere cargar, sin colocar la extensión ko

7. Reiniciar al sistema y verificar que el módulo se haya cargado correctamente.

8. Entrar al archivo **`/etc/modules`** y comentar la última línea para que quede sin efecto y se pueda trabajar con las otras opciones.

2da forma – Utilizando Scripts (1era variante)

Es necesario conocer el concepto de los runlevels. El runlevel (del inglés, nivel de ejecución) es cada uno de los estados de ejecución en que se puede encontrar el sistema Linux. Primero

vamos a explicar un poco cómo funcionan los runlevel del sistema y donde están los ficheros de configuración de estos.

El sistema se carga según la configuración que tenga establecida, y según el runlevel que tenga, se cargará el sistema de una manera u otra.

Para saber en qué nivel o runlevel estamos, deberemos de escribir el comando **`runlevel`**.

Existen 6 niveles de ejecución o runlevel:

– Nivel de ejecución 0: Apagado.
– Nivel de ejecución 1: Monousuario (sólo usuario root; no es necesaria la contraseña). Se suele usar para analizar y reparar problemas.
– Nivel de ejecución 2: Multiusuario sin soporte de red.
– Nivel de ejecución 3: Multiusuario con soporte de red.
– Nivel de ejecución 4: Como el runlevel 3, pero no se suele usar
– Nivel de ejecución 5: Multiusuario en modo gráfico
– Nivel de ejecución 6: Reinicio.

Para ejecutar un runlevel especifico deberemos de escribir:
`#sudo init 6 (reiniciará el sistema)`

`#sudo init 5 (reiniciará el sistema en modo gráfico, en el caso en que estuviéramos en modo consola init 3)`

Los archivos de configuración de los runlevels, los podemos encontrar en /etc/, tenemos 7 tipos:

- rc0.d -> scripts de carga cuando se ejecuta el runlevel 0
- rc1.d -> scripts de carga cuando se ejecuta el runlevel 1
- rc2.d -> scripts de carga cuando se ejecuta el runlevel 2
- rc3.d -> scripts de carga cuando se ejecuta el runlevel 3
- rc4.d -> scripts de carga cuando se ejecuta el runlevel 4
- rc5.d -> scripts de carga cuando se ejecuta el runlevel 5
- rc6.d -> scripts de carga cuando se ejecuta el runlevel 6

Una vez ya sabemos cómo funciona un poco el sistema, si hacemos un **`ls`** por ejemplo a **`/etc/rc2.d/`** encontraremos allí una lista de todos los scripts que se ejecutan al arrancar el sistema. Si nos fijamos bien todos son enlaces simbólicos a archivos que se encuentran en **`/etc/init.d/`** que es donde están todos los daemon.
También se puede observar que todos los archivos comienzan con una S (start) mayúscula y seguidos de un numero entre el 00 y el 99. Estos números determinan el orden en que se ejecutan cada uno de los procesos en este archivo.

1. Crear un script que cargue el LKM al kernel y asignar permisos de ejecución.

2. Copiar el script a la carpeta **`/etc/init.d`**

3. Es necesario convertir el script en servicio o Daemon, se hace con la siguiente línea: **`update-rc.d miscript.sh defaults`**

4. Esto creará los enlaces simbólicos automáticos para todos los runlevels, así dependiendo del tipo de inicio, arrancará nuestro script, y ya estará hecho

5. Listamos todo lo que hay en el directorio por ejemplo **`/etc/rc2.d/,`** y veremos como ahora sale nuestro script

6. Ahora hay que reiniciar el equipo, y verificar que el módulo se haya cargado en el kernel

7. Si no se cargó el LKM, es porque la versión de sistema operativo ya no utiliza Init como demonio principal para el inicio del sistema. Habrá que probar la siguiente variante.

2da forma – Utilizando Scripts (2da variante)

En las versiones de SO más recientes, es necesario crear los servicios para que puedan ejecutarse las funciones como Scripts.

1. Con un editor de texto, crear el servicio "**`[nombre].service`**". Para el ejemplo, se creará como "**`startup.service`**" en la ruta **`/etc/systemd/system`**
2. El contenido del archivo **`startup.service`** deberá contener al menos lo siguiente:

```
[Unit]
Description=startup

[Service]
ExecStart=[ruta del script]

[Install]
WantedBy=default.target
```

3. Asignar los permisos 664 al archivo **`startup.service`**

4. Crear un script en la ruta que definimos dentro del archivo **`startup.service`**, que permita cargar el LKM. La ruta del script será la siguiente: **`/usr/local/bin/;`** el script debe tener asignado el mismo nombre que el servicio creado, por lo tanto, para el ejemplo, el script se llamará **`startup.sh`**

5. Asignar los permisos 744 al script.

6. Actualizar la lista de demonios o servicios con el comando **`systemctl daemon-reload`**

7. Habilitar el servicio que creamos con el comando **`systemctl enable startup.service`**

8. Ahora, hay que probar que el script funcione de forma correcta ejecutando el comando **`systemctl enable startup.service`**

9. Verificar que el LKM se haya cargado en el kernel

10. Descargar el LKM del kernel y reiniciar la computadora y verificar que el LKM se cargue de forma automática al cargar el SO.

ASIGNACIÓN

Utilizando las dos formas vistas en la práctica, hacer que el LKM creado para la evaluación 3 se cargue automáticamente al iniciar el sistema operativo.