Linux para administradores (intermedio)

Manuel Domínguez

Bienvenidos!

Esta sección corresponde con la **Gestión de procesos y servicios.**

Y en esta clase, vamos a explicar cómo se gestionan **los servicios.**

1.- Introducción

Actualmente, para administrar y configurar el sistema e incluso interactuar con el núcleo se utiliza **systemd**. Ha reemplazado a clásico **SysV.**

Systemd trabaja con **unidades.** Una **unidad es un recurso** que el sistema puede manejar. Una de esas unidades son los servicios.

Servicio	**.service**
Socket (Ip+puerto)	.socket
Punto de montaje	.mount
Estado del sistemas (Targets)	.targets

#systemctl list-unit-files → Podemos ver las unidades instaladas
 # systemctl list-unit-files -t service → Unidades tipo servicio

1.- Introducción

Systemd está formado por un conjunto de demonios (procesos en 2º plano). Utilizaremos **systemctl** para controlar esos demonios.

Para el trabajo con **los servicios**, utilizaremos:

systemctl **status** servicio: Muestra el estado de un servicio.
systemctl **start** servicio: Inicia un servicio.
systemctl **stop** servicio: Detiene el servicio.
systemctl **restart** servicio: Detiene el servicio y lo inicia a continuación.
systemctl **reload** servicio: Recarga la configuración de un servicio sin detenerlo**.**

1.- Introducción

Algunos servicios:

Servicio Red: **networking**

Servicio WEB: **apache2**

Servicio SSH: **ssh**

Impresora: cups

Tareas programadas: **cron**

3.- Trabajar con servicios: ssh

Práctica: ssh

1.- Instalamos el servicio ssh: **#apt install ssh**

2.- Visualizamos su estado: **#systemctl status ssh**

3.- Detenemos el servicio ssh: **#systemctl stop ssh**

4.- Reiniciamos el servicio: **#systemctl restart ssh**

4.- Trabajar con servicios: Apache (Servidor web)

Práctica: Apache

1.- Instalamos el servicio ssh: **#apt install apache2**

2.- Visualizamos su estado: **#systemctl status apache2**

3.- Comprobación si desde la máquina cliente, podemos acceder: http://192.168.0.158 (Esta IP es la Debian)

4.- Trabajar con servicios: Apache (servidor web)

Práctica: Apache

4.- Personalizamos la página que lanza:

#cp /var/www/html/index.html /var/www/html/index.html.ORIGINAL

Nos descargamos: index.html y admin.jpg y lo copiamos en:

#cp admin.jpg index.html /var/www/html/

5.- Comprobamos que se ha actualizado.

6.- Detenemos el servicio de apache: **#systemctl stop apache2**

7.- Vemos el estado:**#systemctl status apache2**

8.- Reiniciamos el servicio: **#systemctl restart apache2**

5.- Activar/Desactivar servicios

Puede ser que necesitemos que un servicio esté activo al inicio.

#systemctl enable Servicio: Activa el servicio en el próximo inicio del sistema.

#systemctl disable Servicio: Desactiva el servicio en el próximo inicio del sistema.

#systemctl is-enabled Servicio: Verifica si el servicio está activo.

5.- Activar/Desactivar servicios

Práctica:

1.- Visualiza el estado del servicio ssh: **#systemctl status ssh**

2.- Verifica si el servicio está habilitado:**# systemctl is-enabled ssh**

3.- Deshabilítalo: **# systemctl disable ssh**

4.- Ver el estado. ¿Te extraña el resultado? **#systemctl status ssh**

5.- Vuelve habilitarlo: **# systemctl enable ssh**

6.- Targets (Runlevels)

Targets (Runlevels):

Indica el **estado actual** del sistema operativo y eso conlleva implícito **qué servicios** del sistema se están ejecutando.

En systemd los niveles de ejecución se denominan **targets**.

 # systemctl list-units -t target → Podemos ver los targets presentes

6.- Targets (Runlevels)

Targets (Runlevels):

Tabla de targets

Runlevel de SysV	Target de systemd	Notas
0	runlevel0.target, poweroff.target	Detiene el sistema.
1, s, single	runlevel1.target, rescue.target	Modalidad de usuario único.
2, 4	runlevel2.target, runlevel4.target, multi-user.target	Definidos por el usuario. Preconfigurados a 3.
3	runlevel3.target, multi-user.target	Multiusuario, no gráfica. Los usuarios, por lo general, pueden acceder a través de múltiples consolas o a través de la red.
5	runlevel5.target, graphical.target	Multiusuario, gráfica. Por lo general, tiene todos los servicios del nivel de ejecución 3, además de un inicio de sesión gráfica.
6	runlevel6.target, reboot.target	Reinicia el sistema.
emergency	emergency.target	Consola de emergencia.

6.- Targets (Runlevels)

Práctica: Targets (Runlevels):

1.- Conocer el target por defecto: # **systemctl get-default** → graphical.target

2.- Conocer el target actual: **#runlevel**

3.- Imaginemos que tenemos que hacer tareas de mantenimiento, podríamos cambiar al target **rescue** (Monousuario): **# systemctl isolate rescue.target**

Nota: Hay que observar los mensajes que nos muestra en la parte superior. Introducimos la contraseña y pulsamos exit para salir.

7.- Creación de servicios

Crear un servicio:

Podemos convertir una aplicación (un script en bash o Python) en un servicio.

Y podemos hacer que ese servicio se inicie cuando arranque el ordenador.

7.- Creación de servicios

Práctica: Crear un servicio

1.- Creamos un script: **/root/conexiones** y le damos permiso de ejecución.

Nos da una lista de usuarios conectados y manda la información a

/root/UsuariosConectados.tmp, poniendo la fecha.

2.- Creamos un script: **/root/borradconexiones** y le damos permiso de

ejecución. Nos borra el archivo /rootUsuariosConectados.tmp

7.- Creación de servicios

Práctica: Crear un servicio

/root/conexiones /root/borradconexiones

```
#!/bin/bash
Fecha=$(date +%d-%m-%Y-%H:%M:%S)
Informe="/root/UsuariosConectados.tmp"
echo "**************************" >$Informe
echo "Usuarios conectados">>$Informe
echo "**************************">>$Informe
while true
do
sleep 30 #Esperamos 30 segundo
echo "$Fecha" >>$Informe
who >>$Informe
done
```

```
#!/bin/bash
rm -r /root/UsuariosConectados.tmp
```

7.- Creación de servicios

Práctica: Crear un servicio

3.- Comprobamos que ambos scripts funcionan, antes de crear el servicio.

4.- creamos un archivo llamado: **miservicio.service,** en **/etc/systemd/system,** de forma que cuando ponga **start**, se ejecute **conexiones,** y cuando ponga **stop**, se ejecute **borradconexiones.**

7.- Creación de servicios

Práctica: Crear un servicio.

5.- #nano /etc/systemd/system/miservicio.service
(Dejamos los permisos por defecto)

Puede tener más bloques.

Type=**simple** → Ejecuta el proceso en segundo plano y continua la inicialización de servicios.

RemainAfterExit → Tras ejcutarse el script, el servicio se marca como activo.

WantedBy → Indicamos el objetivo que cumple el servicio.

```
[unit]
Description=Mi servicio de prueba

[Service]
Type=simple
ExecStart=/root/conexiones
ExecStop=/root/borradconexiones
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

7.- Creación de servicios

Práctica: Crear un servicio.

6.- Se lee los nuevos servicios.**#systemctl daemon-reload**
7.- Vemos si está disponible: **# systemctl list-unit-files –t service**
8.- Comprobación:
#systemctl start miservicio → Ejecutará el archivo conexiones y creará en /root/UsuariosConectados.tmp
#systemctl status miservicio → Comprobamos el estado.
#systemctl stop miservicio → Ejecutará el archivo borradconexiones y borrará el archivo /root/UsuariosConectados.tmp

7.- Creación de servicios

Práctica: Crear un servicio.

9.- Si además queremos que ese servicio esté activo cuando se inicie la máquina nos falta escribir:

 # systemctl enable miservicio.service

10.- Podemos ver que se ha incorporado:

 # systemctl list-unit-files -t service → miservicio.service enabled

RETO

Te propongo que crees un script, llamado **ComprobarApache.sh**, de forma que compruebe si el servicio apache2 está running, cada 60 segundos.
Si No está running, entonces:
1.- Introduce una linea: “Error-Apache: Fecha y hora actual” en /root/ApacheError.tmp
2.- Reinicia el servicio apache2
Pista:
1.- Crea un bucle infinito: while do …….done
2.- Utiliza sleep 60, para esperar 60s
3.- Comprueba el estado del servicio con un grep.

Nota: También se podría crear una tarea periódica cada minuto, que ejecuta ese script, pero eso ya lo veremos en la clase correspondiente.

Linux para administradores (intermedio)

Manuel Domínguez

Despedida

Hemos llegado al final de este vídeo.

Nos vemos en el siguiente.