Personalización del arranque del sistema

El arranque del sistema Linux empieza después de la secuencia de inicio, una vez que el núcleo Linux ha sido cargado en memoria y activado. Este núcleo inicializará el material, cargará los módulos dinámicos necesarios para la gestión de los elementos materiales detectados e iniciará los distintos servicios, en función de su configuración.

Los dos modos principales de gestión de este arranque son `init System V`, modelo más antiguo, derivado de `Unix system V`, y `systemd`, más reciente y originario de Linux.

Para la certificación LPIC-2 será necesario dominar estos dos entornos de arranque, ya que los dos se utilizan en producción, aunque `systemd` tiende a ir reemplazando `init System V`. El arranque `systemd` está implementado por defecto en las versiones recientes de la mayoría de las distribuciones.

1. init System V

`init System V` es un mecanismo de arranque y de paro de diferentes servicios y procesos según el nivel de funcionalidades (nivel de ejecución, identificado por un número o una letra) deseado. Se apoya en el programa `init`, proceso lanzado como tarea en segundo plano, creado por el núcleo en el arranque del sistema. El proceso `init` inicia después un conjunto de demonios definidos como activos en el nivel de ejecución por defecto.

Originario de Unix, en su versión System V, este método de arranque ha sido el más utilizado en Linux durante mucho tiempo. Desde la creación de `systemd` por **Lennart Poettering** en 2010, las versiones recientes de las distribuciones (Red Hat, Debian, etc.) tienden a llevarlo por defecto, en detrimento del primero.

La versión 6 de la distribución CentOS y la versión Debian 7 Wheezy utilizan por defecto `init System V`, lo que permite comprobar los conocimientos adquiridos para la certificación LPIC-2.

2. El proceso init

El proceso que ejecuta el programa `/sbin/init` es creado por el núcleo, con PID número 1. Además del proceso de PID número 2 y de los procesos hijos, que se ejecutan en modo núcleo, este proceso es el que se encuentra en el origen de todos los otros procesos que se ejecutan en el sistema.

Como ha sido creado directamente por el núcleo, su proceso padre es un pseudoproceso con el PID número 0.

a. Procesos hijos de init

Extractos de la lista de procesos de un sistema Linux Debian 7 Wheezy:

```
ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 13:47 ?        00:00:00 init [5]
[...]
```

Proceso `init`, PID 1, PPID 0, creado directamente por el núcleo.

```
[...]
root       290     1  0 13:47 ?        00:00:00 udevd --daemon
root       349   290  0 13:47 ?        00:00:00 udevd --daemon
root       350   290  0 13:47 ?        00:00:00 udevd --daemon
[...]
```

El demonio `udevd` es ejecutado por `init` y crea dos procesos hijos.

```
[...]
root     1937     1  0 13:47 ?    00:00:00 /usr/sbin/rsyslogd -c5
104      2015     1  0 13:47 ?    00:00:00 /usr/bin/dbus-daemon --system
[...]
```

El demonio de gestión de los registros `rsyslogd` *y el demonio* `dbus` *también son ejecutados por* `init`.

```
[...]
root     2598     1  0 13:47 tty1     00:00:00 /bin/login --
root     2679  2598  0 13:47 tty1     00:00:00 -bash
```

Una conexión de usuario en la consola virtual `tty1`*, gestionada por el programa* `login` *ejecutado por* `init`.

b. Configuración del proceso init

Tradicionalmente, el archivo `/etc/inittab` es el archivo de configuración del demonio `init`. Permite especificar el nivel de ejecución por defecto, así como los programas que hay que lanzar en función de los niveles de ejecución.

Cada línea del archivo, excepto las líneas de comentarios que empiezan por el carácter #, tiene el formato siguiente:

Sintaxis

```
Id:Niveles:Modo:Comando
```

Con:

`Id`	Identificador único de la línea, con 1 o 2 caracteres.
`Niveles`	Números de los niveles de ejecución en los que se ejecutará la línea.
`Modo`	Tipo de acción (ver abajo).
`Comando`	Línea de comando que se tiene que ejecutar.

Tipo de acciones clásicas:

- `initdefault` : esta línea define el nivel de ejecución por defecto, especificado en el segundo campo de la línea.
- `sysinit` : ejecuta el comando durante la inicialización del sistema. El segundo campo debe estar vacío.
- `wait` : ejecuta el comando y espera el final de la ejecución antes de continuar con las líneas siguientes.
- `respawn` : ejecuta el comando del cuarto campo, sin esperar a que termine, y continúa con las líneas siguientes. Si el proceso creado por el comando se termina, `init` lo vuelve a lanzar automáticamente.
- `ctrlaltdel` : se ejecutará esta línea de comando cuando se pulsen las teclas [Ctrl][Alt][Del] en la consola física del sistema.

Ejemplo

Extractos del archivo `/etc/inittab` de un sistema Linux Debian 7 Wheezy:

```
# /etc/inittab: init(8) configuration.
# $Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $

# The default runlevel.
id:2:initdefault:
```

Esta línea fuerza el nivel de ejecución número 2 como el nivel por defecto.

```
# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS
```

Script de inicialización, ejecutado en el arranque del sistema.

```
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.

l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

Scripts que se ejecutarán cuando se cambie de nivel de ejecución (de 0 a 6), en modo espera al final del script.

```
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

En el caso de que se pulse [Ctrl][Alt][Supr], reinicio inmediato del sistema, para los niveles de ejecución de 1 a 5.

```
# /sbin/getty invocations for the runlevels.
#
# The "id" field MUST be the same as the last
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
```

```
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
```

Gestión de las consolas virtuales en la consola física del sistema, gestionadas por el programa `getty`. En este caso hay seis consolas virtuales (accesibles usando las teclas de función de [Ctrl][Alt][F1] a [Ctrl][Alt][F6]).

Algunas distribuciones usan una versión modificada de la configuración de `init system V`, `upstart`. En ese caso, el archivo `/etc/inittab` solo servirá para configurar el nivel de ejecución por defecto. Sin embargo, el principio de los niveles de ejecución es el mismo.

3. Los niveles de ejecución init System V

Los servicios y demonios activos en un sistema Linux que utilizan `init System V` dependen de su nivel de ejecución (*run level*). Estos niveles de ejecución son gestionados por el proceso `init`, que se ocupará de parar y arrancar los diferentes programas vinculados a un cambio de nivel de ejecución.

a. Los diferentes niveles de ejecución (run levels)

Los niveles de ejecución (*run levels*) se identifican gracias a un número o una letra. Aunque están normalizados en el marco de la LSB (*Linux Standard Base*), su significado puede cambiar según la distribución utilizada.

Los niveles comunes son los siguientes, con respecto al estándar LSB, para las distribuciones de tipo Red Hat y de tipo Debian (versiones que utilizan todavía `init System V`) :

- El nivel 0: (LSB, Red Hat, Debian).

  El sistema está parado. Cuando el sistema pasa a nivel 0, todos los programas configurados para ser parados en este nivel son parados correctamente.

- El nivel 1 (o S o s para algunas distribuciones): (LSB, Red Hat, Debian).

  Nivel monousuario de mantenimiento: solamente autoriza la conexión de la cuenta del superusuario. La mayoría de los servicios están parados en este nivel.

- El nivel 2:

  LSB: multiusuario, sin red ni interfaz gráfica.

  Red Hat: multiusuario, con red, sin NFS y sin interfaz gráfica.

  Debian: multiusuario, con red e interfaz gráfica.

- El nivel 3:

  LSB : multiusuario, con red y sin interfaz gráfica.

  Red Hat: multiusuario, con red y sin interfaz gráfica.

  Debian: multiusuario, con red e interfaz gráfica.

- El nivel 4:

  LSB: indefinido.

  Red Hat: como el nivel 3.

  Debian: multiusuario, con red e interfaz gráfica.

- El nivel 5:

  LSB: multiusuario, con red e interfaz gráfica.

  Red Hat: multiusuario, con red e interfaz gráfica.

  Debian: multiusuario, con red e interfaz gráfica.

- El nivel 6: (LSB, Red Hat, Debian).

  Reinicio del sistema. Todos los programas configurados para ser parados en este nivel lo serán correctamente.

b. Configuración de los diferentes niveles de ejecución

Las distribuciones proponen una configuración por defecto para los diferentes niveles de ejecución. Cuando el sistema cambia de nivel de ejecución, se ejecutan automáticamente scripts de paro y de reinicio de los servicios en función de esta configuración.

El administrador del sistema debe conocer los mecanismos de configuración de esos scripts, para poder intervenir en caso de problema o para personalizarlos si fuera necesario.

c. Scripts de gestión de los servicios

En un sistema Linux que use `init System V`, los servicios son generalmente ejecutados y parados por scripts normalizados.

Se encuentran, directamente o a través de un enlace, en el directorio `/etc/init.d`.

Tienen un primer parámetro que acepta los valores `start`, `stop` y, eventualmente, `status`, `reload` y `restart`.

Sintaxis

```
/etc/init.d/NombreScript acción
```

Donde `acción` puede tener alguno de los valores descritos anteriormente.

El comando `service` permite solicitar la ejecución de un script de gestión de servicio:

Sintaxis

```
service NombreScript acción
```

Los scripts de arranque `init System V` deberían seguir las especificaciones LSB (*Linux Standard Base*). Deberían, en particular, tener una zona de comentarios en la que se describieran sus características en formato LSB.

Ejemplo

Zona de descripción LSB del script de gestión del servicio `ssh`:

```
### BEGIN INIT INFO
# Provides:          sshd
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:
# Short-Description: OpenBSD Secure Shell server
### END INIT INFO
```

Ejemplo

Scripts de gestión de los servicios de un sistema Linux Debian 7 Wheezy:

```
ls /etc/init.d
acpid          bootmisc.sh          dbus          hwclock.sh     motd
mountnfs.sh    procps               reboot        single         udev-mtab
atd            checkfs.sh           exim4         kbd
mountall-bootclean.sh  mtab.sh      pulseaudio    rmnologin      skeleton       umountfs
avahi-daemon   checkroot-bootclean.sh  gdm3       keyboard-setup  mountall.sh
networking     rc                   rpcbind       speech-dispatcher  umountnfs.sh
binfmt-support  checkroot.sh        halt          killprocs      mountdevsubfs.sh
network-manager  rc.local           rsyslog       ssh            umountroot
bluetooth      console-setup        hdparm        kmod           mountkernfs.sh
nfs-common     rcS                  saned         sudo           urandom
bootlogs       cron                 hostname.sh   minissdpd      mountnfs-bootclean.sh
pppd-dns       README               sendsigs      udev           x11-common
```

Extracto del script de arranque de `sshd`:

```
vi /etc/init.d/ssh
[...]
case "$1" in
 start)
      check_privsep_dir
```

```
        check_for_no_start
        check_dev_null
        log_daemon_msg "Starting OpenBSD Secure Shell server" "sshd" || true
        if start-stop-daemon --start --quiet --oknodo --pidfile /var/run/
sshd.pid --exec /usr/sbin/sshd -- $SSHD_OPTS; then
            log_end_msg 0 || true
        else
            log_end_msg 1 || true
        fi
        ;;
    stop)
        log_daemon_msg "Stopping OpenBSD Secure Shell server" "sshd" || true
        if start-stop-daemon --stop --quiet --oknodo --pidfile /var/run/sshd.pid; the
            log_end_msg 0 || true
        else
            log_end_msg 1 || true
        fi
        ;;

    reload|force-reload)
        check_for_no_start
        check_config
        log_daemon_msg "Reloading OpenBSD Secure Shell server's configuration"
"sshd" || true
        if start-stop-daemon --stop --signal 1 --quiet --oknodo --pidfile
/var/run/sshd.pid --exec /usr/sbin/sshd; then
            log_end_msg 0 || true
        else
            log_end_msg 1 || true
        fi
        ;;

    restart)
[...]
    status)
        status_of_proc -p /var/run/sshd.pid /usr/sbin/sshd sshd && exit 0 || exit $
        ;;

    *)
        log_action_msg "Usage: /etc/init.d/ssh {start|stop|reload|force-reload|
```

```
restart|try-restart|status}" || true
    exit 1
    exit 1
esac
```

La estructura `case` *permite gestionar los diferentes valores del argumento recibido:* `start`, `stop`, `status`, `reload`, `restart`, *etc.*

Estatus de `sshd`*:*

```
service ssh status
[ ok ] sshd is running.
```

d. Niveles de ejecución y servicios

En el archivo `/etc/inittab`, por defecto, se encuentran las siete líneas siguientes:

```
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

Para cada uno de los niveles de ejecución, el script `/etc/init.d/rc` es ejecutado por `init`, en modo `wait`, usando como argumento el nivel de ejecución.

El script hace un listado del contenido del directorio `/etc/rcX.d`, donde `X` es el nivel de ejecución. Ejecuta todos los scripts cuyo nombre comienza por la letra `K` (de *Kill*), con el argumento `stop`, y después todos los scripts que comienzan por la letra `S` (de *Start*), con el argumento `start`.

Como cada uno de los archivos de `/etc/rcX.d` es un enlace simbólico hacia un script de gestión de servicio de

`/etc/init.d`, el script `rc` para y arranca los servicios según el nivel de ejecución.

Según las distribuciones este esquema general puede cambiar.

Ejemplo

Configuración de los servicios que se tienen que parar y arrancar en el nivel de ejecución 1 de un sistema Linux Debian 7 Wheezy:

Archivos de enlaces simbólicos en `/etc/rc1.d`:

```
ls -l /etc/rc1.d
total 1
lrwxrwxrwx 1 root root  13 junio  25 13:06 K01atd -> ../init.d/atd
lrwxrwxrwx 1 root root  19 junio  25 13:42 K01bluetooth -> ../init.d/bluetooth
lrwxrwxrwx 1 root root  15 junio  25 13:07 K01exim4 -> ../init.d/exim4
lrwxrwxrwx 1 root root  14 junio  25 13:44 K01gdm3 -> ../init.d/gdm3
lrwxrwxrwx 1 root root  19 junio  25 13:44 K01minissdpd -> ../init.d/minissdpd
lrwxrwxrwx 1 root root  20 junio  25 13:43 K01pulseaudio -> ../init.d/pulseaudio
lrwxrwxrwx 1 root root  15 junio  25 13:44 K01saned -> ../init.d/saned
lrwxrwxrwx 1 root root  27 junio  25 13:43 K01speech-dispatcher ->
../init.d/speech-dispatcher
lrwxrwxrwx 1 root root  22 junio 25 13:44 K02avahi-daemon -> ../init.d/avahi-daemon
lrwxrwxrwx 1 root root  25 junio 25 13:43 K02network-manager -> ../init.d/network-manager
lrwxrwxrwx 1 root root  17 junio 25 13:43 K04rsyslog -> ../init.d/rsyslog
lrwxrwxrwx 1 root root  20 junio 25 13:43 K06nfs-common -> ../init.d/nfs-common
lrwxrwxrwx 1 root root  17 junio 25 13:43 K06rpcbind -> ../init.d/rpcbind
-rw-r--r-- 1 root root 369 oct.  15  2012 README
lrwxrwxrwx 1 root root  19 junio 25 13:03 S01killprocs -> ../init.d/killprocs
lrwxrwxrwx 1 root root  14 junio 25 13:03 S01motd -> ../init.d/motd
lrwxrwxrwx 1 root root  18 junio 25 13:44 S20bootlogs -> ../init.d/bootlogs
lrwxrwxrwx 1 root root  16 junio 25 13:44 S21single -> ../init.d/single
```

Y la mayoría de los archivos de enlaces simbólicos son scripts de paro `(K*)`. Todos

apuntan a los scripts de gestión de servicios, en `/etc/init.d/` .

4. Gestión de los niveles de ejecución init system V

Existen diferentes comandos que permiten conocer el nivel de ejecución actual, y hay otros que permiten personalizar el contenido de los niveles de ejecución.

a. Nivel de ejecución actual

El comando `runlevel` indica el nivel de ejecución actual. No tiene ninguna opción. Nos muestra el antiguo nivel (N si no hay ninguno) y el nivel actual.

Ejemplo

runlevel
N 2

El sistema (Debian) está corriendo en el nivel de ejecución por defecto, `2`.

El comando `who -r` indica también el nivel de ejecución actual. No tiene ninguna opción. Muestra el antiguo nivel (`S` si no hay ninguno) y el nuevo.

Ejemplo

who -r
run level 2 2021-07-21 14:08 último=S

b. Cambiar el nivel de ejecución

Los comandos `init` y `telinit` permiten cambiar el nivel de ejecución del sistema.

Sintaxis

```
telinit|init Nivel
```

Donde `Nivel` representa el nuevo nivel de ejecución.

El comando `telinit` propone la opción `-t` para especificar un lapsus de tiempo antes del cambio de nivel.

En la mayoría de las distribuciones recientes, los dos comandos corresponden al mismo binario.

Ejemplo

```
runlevel
N 2
init 3
runlevel
2 3
```

c. Comandos de gestión del contenido de los niveles de ejecución

Los comandos `update-rc.d` (Debian) y `chkconfig` (Red Hat) permiten gestionar el estado de los servicios según los niveles de ejecución.

Pueden configurar el lanzamiento de un nuevo servicio en diferentes niveles de ejecución, o suprimir un servicio de diferentes niveles de ejecución.

Incorporación de un servicio

```
update-rc.d NombreScript defaults
chkconfig --add NombreScript
```

Donde `NombreScript` representa el nombre del script de gestión del servicio, en el directorio `/etc/init.d`.

El argumento `defaults` del comando `update-rc.d` especifica que el servicio será lanzado en los niveles funcionales por defecto (2, 3, 4, 5) y parado en los otros.

El comando `chkconfig --add` usa los datos LSB en los comentarios del script para determinar su estado en los diferentes niveles de ejecución.

Supresión de los enlaces de gestión de servicios

```
update-rc.d NombreScript remove
chkconfig --del NombreScript
```

Comprobación del estado de un servicio según los niveles

```
chkconfig --list service
```

d. Script independiente del nivel de ejecución: /etc/rc.local

Una vez que todos los scripts vinculados a un nivel de ejecución multiusuario (2, 3, 4, 5) han sido ejecutados, el script `/etc/rc.local`, si existe, también será ejecutado.

5. systemd

`systemd` tiene como principal función el control del arranque y del paro de los servicios, de manera más flexible, más completa y eficaz que `init System V`. Creado por **Lennart Poettering** en 2010, este programa tiende a reemplazar, en las distribuciones recientes, los otros mecanismos de inicialización de los servicios (`init System V`, `upstart`…).

`systemd` está compuesto por un conjunto de programas que aseguran diferentes funcionalidades (gestión de los servicios, montaje de los sistemas de archivos, registro de

los eventos, etc.). En este capítulo describiremos su papel de gestor de servicios vinculados con el arranque y el paro del sistema.

a. Arranque de systemd

Uno de los primeros objetivos de concepción de `systemd` fue el de permitir un arranque más rápido del sistema Linux, reemplazando el antiguo programa `init`, originario de Unix, y sus scripts de arranque de los servicios.

Ejemplo

En la versión 8 de la distribución CentOS, así como en la versión 10 de la distribución Debian, `systemd` reemplaza efectivamente a `init`; el archivo `/sbin/init` es un enlace simbólico hacia el ejecutable `systemd`:

```
ls -l /sbin/init
lrwxrwxrwx 1 root root 20 junio 29 19:07 /sbin/init -> /lib/systemd/systemd
```

Después de ser cargado por el programa de arranque, el núcleo es inicializado, carga los módulos necesarios usando para ello el archivo imagen `initramfs`, y después crea el proceso ejecutando `systemd`, de PID 1.

`systemd` justo después arrancará los servicios del sistema y estará en el origen de casi todos los procesos. También se ocupará del montaje de sistemas de archivos configurados en montaje automático así como del registro de todos los eventos del arranque (incluidos los que están vinculados al inicio del núcleo).

Una vez que los servicios hayan sido arrancados, `systemd` asegurará el control de los mismos. Podrá determinar para cada servicio el conjunto de procesos que dependen de él (a través de la noción de `Cgroups`), para poder así pararlos correctamente en el momento en que el servicio se tenga que parar. También puede conocer exactamente el consumo de recursos de cada servicio y asegurar que estos no sobrepasen los límites eventualmente configurados para dichos servicios.

Ejemplo

Proceso ejecutándose en un sistema Linux CentOS 8 con `systemd`:

```
pstree
systemd-+-ModemManager---2*[{ModemManager}]
        |-NetworkManager---2*[{NetworkManager}]
        |-accounts-daemon---2*[{accounts-daemon}]
        |-alsactl
        |-atd
        |-auditd-+-sedispatch
        |        `-2*[{auditd}]
        |-avahi-daemon---avahi-daemon
        |-bluetoothd
        |-chronyd
        |-colord---2*[{colord}]
        |-crond
        |-cupsd
        |-dbus-daemon---{dbus-daemon}
        |-firewalld---{firewalld}
        |-gssproxy---5*[{gssproxy}]
        |-ibus-x11---6*[{ibus-x11}]
        |-iio-sensor-prox---2*[{iio-sensor-prox}]
        |-irqbalance---{irqbalance}
        |-iscsid
        |-ksmtuned---sleep
        |-libvirtd---16*[{libvirtd}]
        |-lsmd
        |-mcelog
        |-named---4*[{named}]
        |-polkitd---5*[{polkitd}]
        |-rhsmcertd
        |-rngd---2*[{rngd}]
        |-rpc.idmapd
        |-rpc.mountd
        |-rpc.statd
        |-rpcbind
        |-rsyslogd---2*[{rsyslogd}]
        |-rtkit-daemon---2*[{rtkit-daemon}]
        |-smartd
        |-sshd---sshd---sshd---bash-+-more
        |                           `-pstree
```

```
|-systemd-+-(sd-pam)
|       |-dbus-daemon---{dbus-daemon}
|       `-pulseaudio---2*[{pulseaudio}]
|-systemd-+-(sd-pam)
|       |-at-spi-bus-laun-+-dbus-daemon---{dbus-daemon}
|       |                 `-3*[{at-spi-bus-laun}]
|       |-at-spi2-registr---2*[{at-spi2-registr}]
|       |-dbus-daemon---{dbus-daemon}
|       |-ibus-portal---2*[{ibus-portal}]
|       |-pulseaudio---{pulseaudio}
|       `-xdg-permission----2*[{xdg-permission-}]
|-systemd-journal
|-systemd-logind
|-systemd-machine
|-systemd-udevd
|-tuned---3*[{tuned}]
|-udisksd---4*[{udisksd}]
|-upowerd---2*[{upowerd}]
`-wpa_supplicant
```

b. Directorio de trabajo de systemd

`systemd` utiliza el directorio `/run` como directorio de trabajo, y particularmente `/run/systemd` . En el directorio `/run` se monta un sistema de archivos de tipo tmpfs, gestionado en memoria RAM y cuyo contenido se pierde al parar o reiniciar el sistema.

Ejemplo

Directorios `/run` y `/run/systemd` de una distribución Debian 10:

```
mount -v | grep /run
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=397572k,mode=755)
[...]

ls -l /run/systemd
total 0
drwxr-xr-x 2 root root  40 junio  30 14:56 ask-password
srw------- 1 root root   0 junio  30 14:56 fsck.progress
```

```
drwxr-xr-x 7 root            root            280 junio  30 14:55 generator
drwxr-xr-x 4 root            root            160 junio  30 14:56 generator.late
d--------- 3 root            root            160 junio  30 14:55 inaccessible
drwxr-xr-x 2 root            root            360 junio  30 14:57 inhibit
drwxr-xr-x 3 root            root            180 junio  30 14:56 journal
drwxr-xr-x 2 root            root             40 junio  30 14:56 machines
drwxr-xr-x 4 systemd-network systemd-network   80 junio  30 14:56 netif
srwxrwxrwx 1 root            root              0 junio  30 14:56 notify
srwxrwxrwx 1 root            root              0 junio  30 14:56 private
drwxr-xr-x 2 root            root             60 junio  30 14:57 seats
drwxr-xr-x 2 root            root            120 junio  30 09:40 sessions
drwxr-xr-x 2 root            root             40 junio  30 14:56 shutdown
drwxr-xr-x 2 root            root             40 junio  30 14:55 system
drwxr-xr-x 2 root            root             80 junio  30 09:40 transient
drwx------ 2 root            root             40 junio  30 14:56 unit-root
drwxr-xr-x 2 root            root           1900 junio  30 09:40 units
drwxr-xr-x 2 root            root             80 junio  30 09:40 users
```

c. Directorios de configuración de systemd

`systemd` utiliza algunos directorios jerarquizados para gestionar su configuración. Este mecanismo permite preservar los archivos de configuración personalizados, cuando se actualiza el sistema o se reinstala `systemd` o sus componentes:

- `/etc/systemd/system` : directorio que contiene los elementos de configuración personalizados para el sistema local. Su contenido es prioritario.
- `/run/systemd/system` : directorio que contiene los elementos de configuración actuales.
- `/usr/lib/systemd/system` : directorio que contiene los elementos de configuración estándar de los componentes de `systemd` . Su contenido es usado si el elemento correspondiente no existe en `/etc/systemd/system` .

Ejemplo

Definición del objetivo por defecto en los diferentes directorios de configuración de

`systemd`:

```
ls -l /usr/lib/systemd/system/default.target /etc/systemd/system/default.target
lrwxrwxrwx. 1 root root 41  3 junio 10:04 /etc/systemd/system/default.target ->
/usr/lib/systemd/system/multi-user.target
lrwxrwxrwx. 1 root root 16 11 mayo  18:18 /usr/lib/systemd/system/default.target
-> graphical.target
```

El objetivo por defecto ha sido personalizado para este sistema.

El comando `systemd-delta` muestra los archivos de configuración redirigidos, que extienden o cubren otros archivos de configuración.

Ejemplo

```
systemd-delta
[REDIRECTED] /etc/systemd/system/dbus-org.freedesktop.timedate1.service
/usr/lib/systemd/system/dbus-org.freedesktop.timedate1.servi>
[REDIRECTED] /etc/systemd/system/default.target
/usr/lib/systemd/system/default.target
[MASKED]  /etc/systemd/system/systemd-timedated.service
/usr/lib/systemd/system/systemd-timedated.service
[EXTENDED]  /usr/lib/systemd/system/systemd-tmpfiles-clean.service
/usr/lib/systemd/system/systemd-tmpfiles-clean.service.d/10-time>
[EXTENDED]  /usr/lib/systemd/system/systemd-udev-trigger.service
/usr/lib/systemd/system/systemd-udev-trigger.service.d/systemd-ude>
[EXTENDED]  /usr/lib/systemd/system/time-sync.target
/usr/lib/systemd/system/time-sync.target.d/10-time-set.conf

6 overridden configuration files found.
```

d. El comando systemctl

El comando `systemctl` permite interactuar con `systemd` gracias a sus numerosos subcomandos.

Este comando puede interactuar con el servicio `systemd` de un sistema remoto (opción `-H`).

Ejemplo

Versión de `systemd`:

```
systemctl --version
systemd 239
+PAM +AUDIT +SELINUX +IMA -APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP
+GCRYPT +GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD +IDN2 -IDN
+PCRE2 default-hierarchy=legacy
```

Ayuda de `systemd`:

```
systemctl --help
systemctl [OPTIONS...] {COMMAND} ...

Query or send control commands to the systemd manager.

 -h --help           Show this help
    --version        Show package version
    --system         Connect to system manager
    --user           Connect to user service manager
 -H --host=[USER@]HOST
                     Operate on remote host
 -M --machine=CONTAINER
[...]
```

6. Los objetivos systemd

Los objetivos `systemd` (*targets*) corresponden aproximadamente a los niveles de

ejecución de `init System V`. Un objetivo recoge un conjunto de funcionalidades activas. Su nombre lleva el sufijo `.target`.

Los objetivos de `systemd` gestionan las dependencias entre ellos. Un objetivo puede necesitar la implementación de uno o más objetivos.

Por defecto, `systemd` propone diferentes objetivos, los principales son los siguientes, listados en orden creciente de dependencia:

- `sysinit.target`: este objetivo corresponde a la activación de LVM y RAID, al montaje de los sistemas de archivos, la activación de zonas de swap, al arranque del servicio de registro, del servicio de detección de dispositivos (`udev`), etc.
- `basic.target`: este objetivo activa el servicio `firewalld`, `SELinux`, la gestión de los mensajes del núcleo, etc.
- `multi-user.target`: este objetivo inicia los servicios necesarios para activar el modo multiusuario (red de base y servicios de red, impresión, cron, etc.).
- `graphical.target`: este objetivo arranca el entorno gráfico.

`systemd` también proporciona dos objetivos que se pueden utilizar para el mantenimiento del sistema:

- `rescue.target`: este objetivo pone el sistema en modo mantenimiento monousuario, con los sistemas de archivos montados.
- `emergency.target`: este objetivo pone el sistema en modo mantenimiento monousuario, con el sistema de archivos en solo lectura.

a. Objetivo por defecto

En el momento del arranque del sistema, `systemd` activa el objetivo por defecto, `default.target`, definido en el archivo `default.target` (que se encuentra en `/etc/systemd/system` o `/usr/lib/systemd/system`), enlace simbólico hacia uno de los objetivos existentes.

Ejemplo

```
ls -l /etc/systemd/system/default.target
lrwxrwxrwx. 1 root root 36 29 junio 17:00
/etc/systemd/system/default.target
-> /lib/systemd/system/graphical.target
```

El sistema se iniciará en modo gráfico.

El comando `systemctl get-default` muestra el objetivo por defecto del sistema.

El comando `systemctl set-default NombreTarget` permite modificar el objetivo por defecto del sistema.

Ejemplo

Cambio de objetivo por defecto, de multiusuario en modo gráfico a multiusuario sin modo gráfico:

```
systemctl get-default
graphical.target
systemctl set-default multi-user.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target
/usr/lib/systemd/system/multi-user.target.
systemctl get-default
multi-user.target
```

b. Configuración de los objetivos

Un objetivo está compuesto por unidades (`units`). Un objetivo, él mismo, es un tipo particular de unidad.

Los principales tipos de unidades son:

- Servicios (`.service`): programas que se ejecutan como procesos.
- Dispositivos (`.device`): dispositivos detectados por `udev`, y cuya detección puede provocar acciones.

- Montajes de particiones (`.mount`): montaje de sistemas de archivos, desde dispositivos fijos, extraíbles o de red.
- Sockets (`.socket`): creación de sockets que permiten la comunicación entre programas.

Las diferentes unidades pueden depender unas de otras.

Para poner en marcha un objetivo, `systemd` activa las unidades de las que dependen, de manera simultánea o según sus dependencias o la detección de un evento.

La capacidad para arrancar en paralelo diferentes programas hace que el inicio sea mucho más rápido.

La capacidad para vincular el arranque de un programa a la detección de un evento (dispositivo conectado, montaje de un sistema de archivos, mensaje en un socket, etc.) hace que la gestión de los servicios sea más flexible y potente que la del mecanismo tradicional `init System V`.

c. Objetivos y niveles de ejecución

Incluso aunque los objetivos `systemd` tengan un perímetro de funcionalidades más importante que los niveles de ejecución `init system V`, podemos establecer la tabla de correspondencia siguiente:

Nivel de ejecución	Objetivo systemd
`0`	`poweroff.target`
`1`	`rescue.target`
`2`	`multi-user.target`
`3`	`multi-user.target`
`4`	`multi-user.target`
`5`	`graphical.target`
`6`	`reboot.target`

d. Modificar el objetivo durante la carga del núcleo

En caso de problema en el inicio del sistema, se puede especificar el objetivo `systemd` que se quiera activar, pasándolo como argumento del comando de carga del núcleo (ver la sección El gestor de arranque GRUB - Uso de GRUB en modo interactivo, de este capítulo).

Para ello, usando la funcionalidad de edición del menú de GRUB, hay que añadir el argumento siguiente en el comando de carga del núcleo:

```
systemd.unit=NombreObjetivo
```

Por ejemplo, para iniciar en modo de rescate, el argumento sería:

```
systemd.unit=rescue
```

7. Gestión de los servicios por systemd

Los servicios son gestionados por `systemd` usando archivos de configuración que vienen con los paquetes de software de esos mismos servicios. Estos archivos de configuración, que presentan el sufijo `.service`, se encuentran en uno de los directorios de configuración de `systemd` (`/etc/systemd/system` o `/usr/lib/systemd/system`).

El archivo de configuración es un archivo de texto, organizado en diferentes secciones. Éstas son las principales:

- `[Unit]`: esta sección describe las características generales del servicio: descripción, documentación (man), orden de carga y dependencias con respecto a los otros objetivos, etc.
- `[Service]`: esta sección especifica las características del servicio: tipo, archivos de entorno, comandos para ejecutarlo, para pararlo y para recargarlo.
- `[Install]`: esta sección especifica las características de activación/desactivación del servicio.

Ejemplo

Configuración por defecto del servicio `sshd`:

```
vi /usr/lib/systemd/system/sshd.service
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target sshd-keygen.target
Wants=sshd-keygen.target

[Service]
```

```
Type=notify
EnvironmentFile=-/etc/crypto-policies/back-ends/opensshserver.config
EnvironmentFile=-/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS $CRYPTO_POLICY
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

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

El comando `systemctl cat NombreService` muestra el archivo de configuración actual del servicio especificado.

Ejemplo

```
systemctl cat sshd
# /lib/systemd/system/ssh.service
[Unit]
Description=OpenBSD Secure Shell server
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify
RuntimeDirectory=sshd
RuntimeDirectoryMode=0755

[Install]
```

```
WantedBy=multi-user.target
Alias=sshd.service
```

a. Lista y estado de los servicios

El comando `systemctl list-unit-files --type=service --all` muestra la lista y el estado de todos los servicios configurados para `systemd`.

Ejemplo

```
systemctl list-unit-files --type=service --all
UNIT FILE                    STATE
accounts-daemon.service           enabled
alsa-restore.service           static
alsa-state.service             static
[...]
atd.service                  enabled
auditd.service                enabled
auth-rpcgss-module.service         static
autofs.service                disabled
[...]
sshd.service                  enabled
[...]
```

El comando `systemctl status NombreService` muestra el estado actual del servicio especificado.

Ejemplo

Estado del servicio `sshd`:

```
systemctl status sshd
 ssh.service - OpenBSD Secure Shell server
 Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
 Active: active (running) since Tue 2020-06-30 14:56:35 CEST; 2 days ago
  Docs: man:sshd(8)
```

```
        man:sshd_config(5)
 Process: 687 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
Main PID: 745 (sshd)
   Tasks: 1 (limit: 4558)
  Memory: 5.2M
  CGroup: /system.slice/ssh.service
             745 /usr/sbin/sshd -D

julio 03 09:40:15 debian10 sshd[13612]: Accepted password for root from
192.168.0.24 port 52074 ssh2
julio 03 09:40:15 debian10 sshd[13612]: pam_unix(sshd:session): session opened for
user root by (uid=0)
```

El servicio está activo (enabled), está ejecutándose (running) y ha aceptado una conexión por parte del superusuario.

b. Arranque y paro de los servicios

El comando `systemctl start|stop|restart|reload NombreService` arranca, para, reinicia o recarga la configuración del servicio especificado.

Ejemplo

Arranque y paro del servicio `httpd`:

```
systemctl start httpd
systemctl status httpd
  httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled;
vendor preset: disabled)
  Active: active (running) since Fri 2020-07-03 13:50:10 CEST; 6s ago
    Docs: man:httpd.service(8)
Main PID: 41081 (/usr/sbin/httpd)
  Status: "Started, listening on: port 80"
   Tasks: 213 (limit: 23370)
  Memory: 42.4M
  CGroup: /system.slice/httpd.service
```

```
41081 /usr/sbin/httpd -DFOREGROUND
41082 /usr/sbin/httpd -DFOREGROUND
41083 /usr/sbin/httpd -DFOREGROUND
41084 /usr/sbin/httpd -DFOREGROUND
41231 /usr/sbin/httpd -DFOREGROUND

julio 03 13:50:09 centos8 systemd[1]: Starting The Apache HTTP Server...
julio 03 13:50:10 centos8 systemd[1]: Started The Apache HTTP Server.
julio 03 13:50:11 centos8 httpd[41081]: Server configured, listening on: port 80
systemctl stop httpd
```

c. Activación y desactivación de los servicios

La activación por defecto de un servicio depende de la configuración efectuada en el programa. Si está activado, el servicio será arrancado por los objetivos que lo contengan. Sí está desactivado, el servicio no se arrancará.

El comando `systemctl enable|disable NombreService` activa o desactiva el servicio especifícado.

Ejemplo

Activación del servicio `httpd`:

```
systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service
/usr/lib/systemd/system/httpd.service.
```

El comando crea un enlace simbólico en el directorio de los elementos contenidos en el objetivo `multi-user`.

```
systemctl status httpd
  httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled;
vendor preset: disabled)
```

[...]

d. Registro de systemd

`systemd` utiliza su propio servicio de registro de eventos, `systemd-journald` , que administra un archivo en formato binario, en un subdirectorio del directorio `/run/log/` . Los mensajes emitidos por los servicios y el núcleo son transferidos hacia el servicio tradicional de gestión de registros (`syslogd` o `rsyslogd`).

Como el archivo de registros de `systemd` está en formato binario, no se puede acceder directamente a la información. El comando `journalctl` permite consultar su contenido. El comando puede recibir muchos argumentos y funciona en modo interactivo.

Ejemplo

journalctl

```
-- Logs begin at Tue 2020-06-30 14:51:56 CEST, end at Fri 2020-07-03 14:20:01 CEST. --
junio 30 14:51:56 centos8 kernel: microcode: microcode updated early to revision 0x2f,
date = 2019-02-17
junio 30 14:51:56 centos8 kernel: Linux version 4.18.0-193.6.3.el8_2.x86_64
(mockbuild@kbuilder.bsys.centos.org) (gcc version 8.3.1 201>
junio 30 14:51:56 centos8 kernel: Command line: BOOT_IMAGE=(hd0,msdos1)/
vmlinuz-4.18.0-193.6.3.el8_2.x86_64 root=/dev/mapper/cl-root ro>
junio 30 14:51:56 centos8 kernel: x86/fpu: Supporting XSAVE feature 0x001:
'x87 floating point registers'
junio 30 14:51:56 centos8 kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
junio 30 14:51:56 centos8 kernel: x86/fpu: Enabled xstate features 0x3, context size is
576 bytes, using 'standard' format.
junio 30 14:51:56 centos8 kernel: BIOS-provided physical RAM map:
junio 30 14:51:56 centos8 kernel: BIOS-e820:
[mem 0x0000000000000000-0x000000000009ebff] usable
[...]
```

La opción `-e` muestra solamente las últimas líneas del registro:

journalctl -e

```
julio 03 14:15:01 debian10 CRON[14475]: pam_unix(cron:session): session opened for
user root by (uid=0)
julio 03 14:15:01 debian10 CRON[14476]: (root) CMD (command -v debian-sa1 >
/dev/null && debian-sa1 1 1)
julio 03 14:15:01 debian10 CRON[14475]: pam_unix(cron:session): session closed for
user root
julio 03 14:17:01 debian10 CRON[14479]: pam_unix(cron:session): session opened for
user root by (uid=0)
julio 03 14:17:01 debian10 CRON[14480]: (root) CMD (   cd / && run-parts --report
/etc/cron.hourly)
julio 03 14:17:01 debian10 CRON[14479]: pam_unix(cron:session): session closed for
user root
julio 03 14:18:17 debian10 squid[11822]: Logfile: opening log stdio:
/var/spool/squid/netdb.state
julio 03 14:18:17 debian10 squid[11822]: Logfile: closing log stdio:
/var/spool/squid/netdb.state
julio 03 14:18:17 debian10 squid[11822]: NETDB state saved; 0 entries, 0 msec
julio 03 14:25:01 debian10 CRON[14490]: pam_unix(cron:session): session opened for
user root by (uid=0)
julio 03 14:25:01 debian10 CRON[14491]: (root) CMD (command -v debian-sa1 >
/dev/null && debian-sa1 1 1)
julio 03 14:25:01 debian10 CRON[14490]: pam_unix(cron:session): session closed for
user root
```

8. Paro y reinicio del sistema por systemd

El comando `systemctl reboot|suspend|hibernate|poweroff` reinicia, suspende, pone en modo hibernación o para el sistema.