init System V

1. init System V en 2020

init System V ha sido remplazado masivamente por systemd en todas las distribuciones recientes. Ninguna distribución reciente utiliza init System V por defecto. Podría tener la tentación de saltarse esta parte.

Esto sería un error, y aquí las razones:

- La primera, y pragmática, es que muchos editores todavía no han hecho el esfuerzo de adaptar sus scripts de arranque, los cuales están ubicados en /etc/init.d/.
- La segunda, es que en entornos empresariales no actualizamos los servidores tan rápido como aparecen las nuevas versiones. Encontramos muchas versiones antiguas de distribuciones que funcionan con init (esperemos que estas empresas ofrezcan un soporte técnico extendido).
- La tercera, es que systemd y upstart mantienen une compatibilidad con la noción de runlevels y scripts init. Es evidente que esto es por la primera razón.
- La cuarta, es que los sucesores de init en Linux no son POSIX. Init es POSIX. Si comprende su funcionamiento usted estará más cómodo con otros Unix, libres y propietarios.

Finalmente, estas distribuciones profesionales, están o han estado soportadas hasta hace muy poco, y utilizan init System V:

- El soporte de ELS de Red Hat 5 terminó en noviembre de 2020.
- El soporte LTSS de Suse Linux Enterprise Server terminará en marzo de 2022.

Sin embargo, si su caso no entra en esta categoría, vaya al punto siguiente y sáltese la sección que no le interese.

2. Funciones

El programa init es el primer proceso que se inicia y el último que se para dentro del sistema, y su misión es ejecutar las demás tareas. La función de init consiste en iniciar y parar todos los servicios. init ejecutará las diferentes tareas iniciales necesarias para el buen funcionamiento de Linux mediante la ejecución de varios comandos y scripts.

Una vez iniciado el sistema y ejecutados los servicios, init sigue activo para gestionar los cambios de estado de los procesos que controla y de los niveles de ejecución.

El programa init puede cambiar de una distribución a otra. Ya que se trata del primer programa ejecutado, podría ser:

- init System V
- upstart
- systemd
- u otro ejecutable

El proceso **init** es el padre de todos los procesos y tiene siempre el PID 1. Es arrancado por el kernel, que lleva el PID "virtual" 0.

La sección siguiente solo concierne init System V.

Su configuración está en el archivo `/etc/inittab`. Si este archivo está corrompido o es inutilizable, habrá que arrancar el sistema en modo single (S, s, 1 Single) y arreglarlo, o en el peor de los casos arrancar desde un soporte externo o un disco de emergencia. Es un archivo central del sistema operativo.

3. Niveles de ejecución

Un nivel de ejecución, o runlevel, corresponde al estado en el cual está Unix/Linux. Init controla dicho estado. Cada estado dispone de su propia configuración (o por inittab, o por scripts llamados initscripts). Por ejemplo, se puede utilizar un nivel de ejecución para arrancar Unix en modo monousuario, en multiusuarios, con o sin red, con o sin modo gráfico. El administrador puede personalizar todos los niveles. Por convención, en las distribuciones Red Hat/Fedora, Mandriva, OpenSUSE y asociadas, se suelen definir estos niveles de la siguiente manera:

Nivel	Efecto
0	Halt: detiene el sistema operativo, apaga la máquina.
1	Modo monousuario utilizado para el mantenimiento, modo consola.
2	Multiusuario, sin red, consola.
3	Multiusuario, con red, consola.
4	Igual que el 3, a conveniencia del administrador.
5	Multiusuario, con red, con entorno gráfico X Window.
6	Reboot: reinicio de la máquina.
S,s	Single user mode, el modo más bajo en caso de problema.

Los niveles 7 a 9 son perfectamente válidos, pero no se utilizan por defecto. El nivel de ejecución se sitúa por defecto en `/etc/inittab` en la línea initdefault.

```
id:5:initdefault:
```

Sustituya 5 por el nivel deseado en el momento del arranque.

Un reto en ciertos ejercicios eliminatorios de certificaciones consiste en crear una situación de avería en la cual el valor por defecto está en 0 o 6. En el primer caso, la máquina se apaga en cuanto se ejecuta init. En el otro, arranca en bucle...

La distribución Debian (y las distribuciones que derivan de ella) considera también los niveles 2 a 5 como multiusuario, pero no establece diferencias entre estos niveles. Por defecto, se arranca en el nivel 2, donde se inicia todo, incluso en su caso la interfaz gráfica.

Como se puede modificar y volver a configurar completamente cada nivel, es posible volver a definirlo todo, y por lo tanto, hacer que una Debian se inicie como una Red Hat, y viceversa. Por motivos de conformidad y asistencia, tenga en cuenta que es importante seguir el "estándar" de la distribución que utiliza.

4. /etc/inittab

Se define el comportamiento del proceso init y de los runlevels en el archivo **`/etc/inittab`**. La sintaxis de una línea es la siguiente:

```
Id:[niveles]:acción:comando
```

Campo	Descripción
Id	Identificador de línea sobre cuatro caracteres (en Linux con getty/mingetty: número de terminal).
Niveles	Indica si se debe tener en cuenta el comando para el nivel requerido. Corresponde a la lista de los niveles sin separador.
Acción	Tipo de acción que efectuar según las circunstancias para esta línea.
Comando	El comando que ejecutar con sus parámetros y redirecciones.

La acción es muy importante, ya que define las actividades de init durante el arranque y cambio de nivel. Las principales se presentan a continuación:

Acción	Significado
initdefault	Define el nivel por defecto durante el boot y el inicio de init.
sysinit	Se ejecuta una única vez durante el arranque del sistema.
boot	Igual, pero después de sysinit.
bootwait	Igual, pero init espera el final de la ejecución del comando antes de seguir leyendo el archivo inittab.
off	Se ignora la línea.
once	Se ejecuta el comando a cada cambio de nivel para los niveles especificados.
wait	Igual, pero init espera que finalice la ejecución antes de proseguir.
respawn	El comando se ejecuta para los niveles correspondientes. Si el proceso termina, vuelve a arrancarse de nuevo automáticamente. Es lo que ocurre con los terminales si un usuario se desconecta.

Damos un ejemplo procedente de una antigua instalación OpenSUSE 10.3:

```
# Nivel de ejecución a 5 (multiusuario gráfico)
id:5:initdefault:

# Primer script ejecuta el arranque
si::bootwait:/etc/init.d/boot

# Gestión de los servicios por nivel de ejecución
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

# Caso del modo single, consola de emergencia para root
ls:S:wait:/etc/init.d/rc S
~~:S:respawn:/sbin/sulogin

# Acción en Alt+Ctrl+Del
ca::ctrlaltdel:/sbin/shutdown -r -t 4 now

# Qué hacer en caso de corte de la corriente
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop

# Arranque de las consolas virtuales Alt+Fx
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

5. Cambio de nivel

También puede cambiar de nivel después de arrancar la máquina con el comando **/sbin/init** o **/sbin/telinit**, al ser este último un simple vínculo simbólico a init. El comando siguiente pasa al nivel 5.

```
# telinit 5
```

Se pueden especificar los valores `q`, `u` o `-t`:

- `Q` o `q`: init vuelve a leer el archivo **`/etc/inittab`**, si se ha modificado, y corrige sus tablas internas.
- `U` o `u`: init vuelve a iniciarse sin leer inittab y sin cambiar de nivel. Si se han añadido o suprimido servicios del nivel en curso, init tiene en cuenta la modificación.
- `-t`: cuando init termina la parada de los servicios (o más bien cuando lo hace el script rc; ver más adelante), init manda la señal SIGTERM a todos los procesos restantes, les pide que se terminen correctamente, espera el número de segundos especificado (5 por defecto), luego manda SIGKILL.

El nivel de ejecución es visible con el comando **/sbin/runlevel**. El primer valor devuelto corresponde al nivel que precede el nivel actual. Una N significa que no hay un nivel anterior. El segundo valor es el nivel actual.

```
# runlevel
N 5
```

6. Configuración del sistema básico

Sea cual sea el nivel de ejecución especificado por defecto, init inicia siempre el comando asociado a las acciones sysinit, bootwait o boot de **`/etc/inittab`** en el momento de arrancar el sistema. La acción sysinit es la primera.

- En Red Hat: `si::sysinit:` **`/etc/rc.d/rc.sysinit`**

- En OpenSUSE: `si::bootwait:` **`/etc/init.d/boot`**
- En Debian: `si::sysinit:` **`/etc/init.d/rcS`**

En Red Hat, es un script único monolítico el que se encarga de toda la configuración básica. En Debian, el script llama a todos los scripts del nivel S (single). En OpenSUSE, el script instala lo estrictamente necesario y luego ejecuta el contenido de **`/etc/rc.d/boot.d`**, que establece el resto de la configuración básica.

En todos los casos, las tareas siguientes se ejecutan aproximadamente en este orden:

- Configuración de los parámetros del núcleo presentes en **`/etc/sysctl.conf`** (p. ej.: IP Forwarding).
- Instalación de los archivos periféricos (/dev mediante udev, por ejemplo).
- Configuración del reloj del sistema.
- Carga de las tablas de caracteres del teclado.
- Activación de las particiones de intercambio SWAP.
- Definición del nombre de anfitrión.
- Control y montaje del sistema de archivos raíz (en lectura-escritura esta vez).
- Añadido de los periféricos RAID, LVM o ambos. Esto ya se puede instalar durante la carga de inittab.
- Activación de las cuotas de disco.
- Control y montaje de los demás sistemas de archivos.
- Limpieza de los bloqueos (stale locks) y de los archivos PID en caso de parada brusca.

En algunas distribuciones es posible interactual con init. Al principio del boot, después del arranque de init, le pueden pedir que pulse la letra i y que conteste sí o no a las diferentes acciones.

7. Nivel de ejecución

El script **`/etc/init.d/rc`** coge como parámetro el nivel de ejecución por defecto

según la línea **initdefault** de `/etc/inittab` o el parámetro especificado durante la llamada manual de los comandos **init** o **telinit**. El script rc inicializa el nivel de ejecución deseado y es responsable del inicio y de la parada de los servicios asociados cuando el nivel de ejecución cambia.

```
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
```

Se analizan los servicios en cada nivel de ejecución. Durante el paso de un nivel a otro, y sea cual sea el orden (del 2 al 5, del 5 al 3, etc.) el script rc compara entre el antiguo y el nuevo nivel los servicios que se deben detener o iniciar. Si dos niveles tienen el mismo servicio, lo mantienen. Si se debe iniciar un nuevo servicio en el nuevo nivel, lo hace. Si se debe parar un servicio porque no está en el nuevo nivel, lo para.

Este funcionamiento, estándar a todas las distribuciones Linux de tipo System V, no es común en todos los Unix. HP-UX (un Unix de HP) considera que debe haber un progreso constante en los niveles, pasando sucesivamente del 1 al 3 (1 luego 2 luego 3) y cargando de manera sucesiva los servicios. Parado, baja hasta el nivel 0 y termina los servicios sucesivamente. La diferencia es importante: no compara los niveles y no efectúa parada/reinicio entre cada nivel...

8. Gestión de los niveles y de los servicios

a. Servicios en init.d

El nivel de ejecución define los servicios que se deben iniciar para este nivel. Le

corresponde al script rc cargar los servicios. Se controlan los servicios (inicio, parada, reinicio, estatus, etc.) mediante scripts presentes en el directorio **`/etc/init.d`**.

```
# cd /etc/init.d
# ls -l
-rwxr-xr-x  1 root root  1128 ago  9  2004 acpid
-rwxr-xr-x  1 root root   834 sep 28  2004 anacron
-rwxr-xr-x  1 root root  1429 jun 22  2004 apmd
-rwxr-xr-x  1 root root  1176 abr 14  2006 atd
-rwxr-xr-x  1 root root  2781 mar  5  2007 auditd
-rwxr-xr-x  1 root root 17058 sep  5  2007 autofs
-rwxr-xr-x  1 root root  1368 feb  2  2007 bluetooth
-rwxr-xr-x  1 root root  1355 may  2  2006 cpuspeed
-rwxr-xr-x  1 root root  1904 jul 16  2007 crond
-rwxr-xr-x  1 root root  2312 oct 30 13:46 cups
...
```

Para cada nivel de ejecución n, existe un directorio rcn.d que contiene vínculos simbólicos (atajos) hacia los servicios presentes en **`/etc/init.d`** que se quieren iniciar o parar. Este directorio está en diferentes lugares según la distribución:

- Red Hat: **`/etc/rc.d/rcn.d`** con vínculos **`en/etc/rcn.d`**
- OpenSUSE: **`/etc/init.d/rcn.d`**, sabiendo que **`/etc/rc`**.d apunta a **`/etc/init.d`**
- Debian: **`/etc/rcn.d`**

El prefijo del nombre de cada vínculo define su orden de ejecución o de parada. El nombre está con la forma siguiente:

```
[SK]nnservicio
```

- **S**: start.
- **K**: kill (stop).
- **nn**: orden numérico de ejecución o parada (00=primero, 99=último).
- **servicio**: nombre del servicio.

Por ejemplo, el vínculo S10network indica que se iniciará el servicio network, responsable de la instalación de la red, en orden 10, después de los S01, S05, etc., pero antes de los S11, S15, S20, etc.

```
# ls -1 S*
S00microcode_ctl
S01sysstat
S02lvm2-monitor
S05kudzu
S06cpuspeed
S08iptables
S09isdn
S09pcmcia
S10network
S12syslog
S13irqbalance
S13portmap
S14nfslock
S15mdmonitor
S18rpcidmapd
...
```

Cuando se ejecuta rc, primero lista todos los vínculos que comienzan por K* usando un bucle for. Luego hace lo mismo para S*, y esta vez inicia los servicios. Presentamos una parte del archivo rc para entender mejor la secuencia de inicio:

```
# prueba de la existencia de /etc/rcn.d
if [ -d /etc/rc${level}.d ]
then
  # Lista todos los scripts que empiezan por S en este directorio
  for i in /etc/rc${level}.d/S*
  do
     # el script existe y no está vacío: se ejecuta
     if [ -s ${i} ]
        then
           sh ${i} start
     fi
  done
```

```
fi
```

b. Control manual de los servicios

Mediante scripts

Se pueden iniciar los servicios en todos los casos individualmente o con la ayuda de herramientas según la distribución. El primer método es el único por defecto en Debian.

Cada servicio presente en /etc/init.d acepta al menos dos parámetros:

- **start**: el servicio se inicia.
- **stop**: el servicio se para.

Si desea iniciar o parar el servicio sshd (servidor ssh) manualmente:

```
# /etc/init.d/sshd start
Starting SSH daemon                    done
# /etc/init.d/sshd stop
Shutting down SSH daemon                    done
```

Algunos servicios pueden aceptar otros parámetros:

```
# /etc/init.d/sshd
Usage: /etc/init.d/sshd {start|stop|status|try-restart|restart|
force-reload|reload|probe}
```

- **status**: facilita el estado del servicio (iniciado o no). Según los servicios, se puede facilitar información adicional.
- **probe**: indica si es necesario cargar la configuración; si, por ejemplo, se han modificado archivos de configuración.
- **reload / forcereload**: indica al servicio que vuelva a leer su configuración (mediante una señal 1).
- **restart**: espera y vuelve a iniciar el servicio, sea cual sea el final de la parada.

- **try-restart**: para y vuelve a iniciar el servicio solamente en caso de parada.

La distribución OpenSUSE crea vínculos simbólicos **rc<servicio>** que permiten insertar **rcsshd**, por ejemplo para el control manual de los servicios.

Mediante el comando service

El comando **service** está disponible en Red Hat y OpenSUSE. Permite simplemente prescindir de la ruta hacia el script de inicio del servicio y utilizar simplemente su nombre:

```
# service sshd stop
Shutting down SSH daemon                    done
# service sshd start
Starting SSH daemon                         done
```

Para controlar la configuración de los servicios de System V lanzados por init, no se aconseja hacerlo todo a mano, sino más bien utilizar las herramientas del sistema correspondiente cuando existan, tanto en modo texto o como gráfico.

c. Modificación de los niveles de ejecución

Red Hat y OpenSUSE

En Red Hat/Fedora y OpenSUSE, el comando **chkconfig** permite añadir, suprimir, activar o desactivar scripts, por nivel de ejecución. Este comando es muy práctico para configurar los servicios porque sabe gestionar tanto los servicios System V como los servicios xinetd.

```
chkconfig [opción] [servicio]
```

Aunque la sintaxis sea idéntica en las dos distribuciones, chkconfig no funciona de la misma manera. En Red Hat, se inserta una línea especial al principio del script que indica a chkconfig los parámetros por defecto (runlevels, posiciones de inicio y parada). En OpenSUSE, chkconfig es un frontend para el comando **insserv**. Este último aprovecha también el encabezamiento de los scripts, pero de manera más compleja (gestiona la ordenación y el paralelismo, por ejemplo).

Presentamos las primeras líneas del script de inicio de servicio sshd en Red Hat. El script se inicia y se para en los niveles 2, 3, 4 y 5. Se ejecuta en posición 55 (S55sshd) y se para en posición 25 (K25sshd).

```
# chkconfig: 2345 55 25
# description: OpenSSH server daemon
```

Veamos lo mismo en OpenSUSE. chkconfig e insserv gestionan ellos mismos el orden de inicio y parada gracias a los campos Required-Start y Required-Stop. Se deben iniciar los servicios de red y remote_fs antes de sshd. El servicio se inicia en los niveles 3 y 5 y se detiene en los niveles 0 (parada), 1 (single user), 2 (sin red) y 6 (reboot).

```
### BEGIN INIT INFO
# Provides: sshd
# Required-Start: $network $remote_fs
# Required-Stop: $network $remote_fs
# Default-Start: 3 5
# Default-Stop: 0 1 2 6
# Description: Start the sshd daemon
### END INIT INFO
```

Damos la lista de las opciones de chkconfig:

- **--list**: lista del conjunto de la configuración.
- **--list** servicio: la configuración de un servicio dado.

- **--add** servicio: añade el servicio indicado en la configuración System V.
- **--del** servicio: suprime el servicio de la configuración System V.
- **--level xxx** servicio **on/off**: activa o desactiva el servicio para los niveles de ejecución indicados.

```
# chkconfig --list
rwhod    0:parada 1:parada   2:parada   3:parada   4:parada   5:parada   6:parada
atd      0:parada 1:parada   2:parada   3:funciona 4:funciona 5:funciona 6:parada
snmpd    0:parada 1:parada   2:funciona 3:funciona 4:funciona 5:funciona 6:parada
ntpd     0:parada 1:parada   2:funciona 3:funciona 4:funciona 5:funciona 6:parada
keytable 0:parada 1:funciona 2:funciona 3:funciona 4:funciona 5:funciona 6:parada
syslog   0:parada 1:parada   2:funciona 3:funciona 4:funciona 5:funciona 6:parada
...

# chkconfig --list smb
smb      0:parada 1:parada   2:parada   3:parada   4:parada   5:parada   6:parada

# chkconfig --level 35 smb on
# chkconfig --list smb
smb      0:parada 1:parada   2:parada   3:funciona 4:parada   5:funciona 6:parada

# /sbin/chkconfig --add httpd
```

chkconfig no inicia ningún servicio. Sólo configura los niveles de ejecución. Para iniciar un servicio, se utilizará el script asociado o el comando servicio.

En Debian y Ubuntu

El comando **update-rc.d** crea los vínculos necesarios en los distintos directorios rcn.d, un poco como chkconfig, pero de manera más "bruta": los scripts no tienen la información particular en su encabezamiento y el comando sólo funciona a nivel del sistema de archivos, por lo que instala los diferentes vínculos según sus indicaciones.

Veamos dos ejemplos de sintaxis. El primero inscribe un servicio con parámetros por defecto. En este caso, se configura el servicio para ejecutarse del nivel 2 a 5 y pararse en los niveles 0, 1 y 6. La posición de parada/relanzamiento está en 20.

```
# update-rc.d ssh defaults
```

En el segundo ejemplo, se inserta el servicio con opciones completas. Al iniciarse, el servicio está en posición 10 en los niveles 3, 4 y 5. Parado, el servicio está en la posición 5 en los niveles 0, 1 y 6. No olvide los puntos.

```
# update-rc.d ssh start 10 3 4 5 . stop 05 0 1 6 .
```

El parámetro remove suprime los vínculos de los diversos directorios. Sin embargo, el propio script **`/etc/init.d/xxx`** asociado ya no debe existir. En caso contrario, utilice el parámetro `-f` para forzar la supresión de los vínculos (el propio script sigue en su sitio).

```
# update-rc.d -f ssh remove
```

9. Consolas virtuales

Las consolas virtuales permiten obtener terminales virtuales en una máquina. Se definen en **`/etc/inittab`**. Están disponibles mediante los periféricos **`/dev/ttyn`**, donde n es el número de consola.

No se suele instalar ni iniciar la capa gráfica en los servidores de empresa. Se accede al servidor o bien directamente (o por kvm), o bien por la red (ssh por ejemplo) o mediante soluciones integradas (puertos de administración del servidor, como ILO en las máquinas HP).

- Pase de una consola a la otra con la secuencia de teclas [Alt][Fn] (p. ej.: [Alt][F2]) desde la consola o [Ctrl][Alt][Fn] desde X Window.
- Utilice las teclas [Alt][Flecha derecha] y [Alt][Flecha izquierda] para pasar a la

consola siguiente o a la anterior.

- /dev/ttyn representa la consola virtual n.
- /dev/tty0 representa la consola actual.
- Como hay 12 teclas de función, puede haber por defecto 12 terminales virtuales.
- Sin embargo, se activan “solamente” 6 por defecto.
- Se inicia X Window por defecto en la primera consola disponible, en general la 7.

Se inician las consolas con inittab y los procesos **getty** o **mingetty**. Son las únicas entradas de inittab donde la etiqueta tiene su importancia: corresponde al número de la consola. Observe la utilización de respawn. Como la consola establece la petición de login, luego el shell se sustituye y, cuando se termina, se inicia un proceso mingetty automáticamente para aceptar una nueva conexión.

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

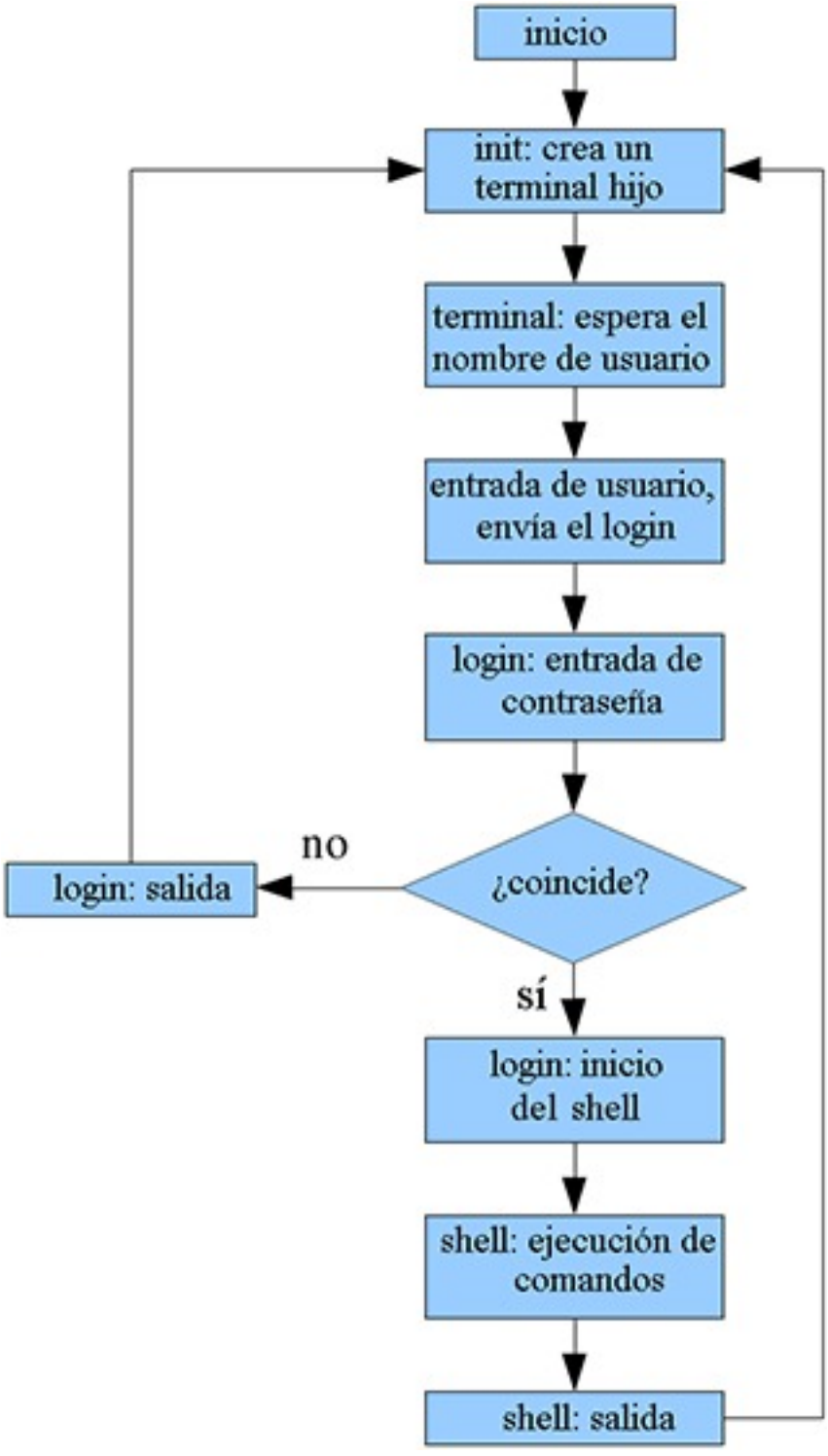
10. Los logins

Una vez que init ha iniciado los terminales (getty), un prompt espera la inserción del nombre (el login) del usuario. Antes de este prompt, se visualiza el contenido del archivo `/etc/issue`. Con el nombre validado, el terminal ejecuta el comando **login**, que requiere la inserción de la contraseña. Si la contraseña es correcta (verificación en `/etc/passwd` y `/etc/shadow` o utilización de los módulos PAM), entonces login muestra el contenido de `/etc/motd` y ejecuta un shell (siempre definido en `/etc/passwd`).

Observe que getty y login no efectúan un fork: los procesos iniciados no son hijos, pero solapan el proceso actual (API exec). No hay relaciones hijos-padres entre los procesos (getty->login->shell) sino que cada uno sustituye al anterior, y el proceso guarda el mismo PID. De esta manera init sabe cuándo se termina una conexión.

Una vez terminada la sesión (fin del shell), init vuelve a iniciar un terminal para una nueva conexión (comando **respawn**).

getty va a permitir un buen funcionamiento del terminal del usuario: para ello se va a adaptar a sus diferentes parámetros (VT100, VT220, XTERM, CONSOLE...). Además, getty puede escuchar perfectamente un puerto serie y soportar una conexión con módem (e iniciar luego una sesión ppp, por ejemplo).



Secuencia de inicio de sesión empleando login

11. Parada

Varios métodos permiten parar correctamente una máquina en Linux. Antes de nada y a título orientativo, init gestiona las paradas con los niveles 0 y 6. En la práctica, los dos son casi idénticos salvo para la última acción.

- Runlevel 0: el ordenador se apaga automáticamente.
- Runlevel 6: el ordenador se apaga y se reinicia de nuevo.

De esta manera, el comando siguiente apaga el ordenador:

```
# init 0
```

Este comando vuelve a iniciarlo:

```
# init 6
```

Sin embargo, el comando más correcto, más limpio y más seguro para parar el sistema es **shutdown**. shutdown llama a init, pero acepta parámetros adicionales. Su sintaxis básica es: `shutdown <param> <plazo> <mensaje>`.

Los parámetros son:

Parámetro	Acción
`-k`	No apaga el sistema, sino que manda el mensaje de apagado a todo el mundo.
`-r`	Reinicio.
`-h`	(halt) Parada.
`-f`	Impide la ejecución de fsck en el boot.
`-F`	Fuerza la ejecución de fsck en el boot.
`-c`	Cancela el shutdown sin demora, pero puede enviar un mensaje.

Se puede especificar el plazo de diferentes maneras:

- **hh:mm**: una hora concreta.
- **+m**: en m minutos.
- **now**: un alias para +0, es decir, ahora mismo.

El ejemplo siguiente programa un reinicio para dentro de 10 minutos con un mensaje de aviso.

```
# shutdown -r +10 "Reinicio para mantenimiento en 10 minutos"

Broadcast message from root (pts/2) (Fri Apr 4 15:00:34 2008):

Reinicio para mantenimiento en 10 minutos
The system is going DOWN for reboot in 10 minutes!
```

El ejemplo siguiente cancela el reinicio.

```
# shutdown -c "Mantenimiento cancelado"

Shutdown cancelled.

Broadcast message from root (pts/2) (Fri Apr  4 15:02:21 2008):

Mantenimiento cancelado
```

Se llama a los comandos **reinicio** y **halt** al final de init 6 y 0, respectivamente. Si se llaman a un nivel distinto de 6 o 0, equivale a una llamada de shutdown:

- **halt**: shutdown -h
- **reboot**: shutdown -r