

# SERVICIOS LINUX

 [Descargar PDF](#)

## ÍNDICE

### ▼ SERVICIOS LINUX

#### ▼ SYSTEM V

- SERVICIOS
- OPERACIONES
- RUNLEVELS
- INITTAB

#### ▼ SYSTEMD

- SERVICIOS
- RUNLEVELS
- OPERACIONES

# SERVICIOS LINUX

Los sistemas de inicio son el primer proceso del sistema (PID 1) y son los responsables de arrancar la máquina, iniciar servicios y administrar el resto de procesos del sistema. En las distribuciones Linux podemos encontrar dos sistemas inicio diferentes.

- **system V** distribuciones Linux antiguas (1983).
- **systemD** distribuciones Linux modernas (2010).



## ¿system V o systemd?

Comprobar proceso con PID 1

```
# Ejemplo sistema System V
ps -p 1
PID  USER   TIME  COMMAND
1    root    0:00  /sbin/init
# Ejemplo sistema systemd
ps -p 1
PID  USER   TIME  COMMAND
1    root    0:00  systemd
```

## SYSTEM V

Denominado también UNIX System V o SysVinit. El proceso inicial se llama `init` y ejecuta el comando `/sbin/init` con PID 1. La **ejecución** de los servicios se realiza **secuencialmente**, es decir, cuando lanza la ejecución de un servicio espera a que termine antes de lanzar el siguiente. No existe definición explícita de dependencias más allá del orden en la ejecución secuencial.

## SERVICIOS

Los scripts que definen los servicios están ubicados en la carpeta `/etc/init.d` o `/etc/rc.d/init.d` dependiendo de la distribución Linux. Un script de definición de servicio tiene normalmente la siguiente estructura:

```
#script definición servicio
#$1 parámetro de ejecución del script
case "$1" in
start)
#Instrucciones para iniciar el servicio
;;
stop)
#Instrucciones para detener el servicio
;;
restart)
#Instrucciones para reiniciar el servicio
;;
status)
#Instrucciones para comprobar el estado el servicio
;;
*)
#Cualquier otra opción muestra la ayuda de uso
echo "Usage: $0 {start|stop|restart|status}"
exit 1
;;
esac
exit 0
```

## OPERACIONES

Para ejecutar las operaciones básicas sobre un servicio se puede invocar el script del servicio con la opción deseada `/etc/init.d/<servicio> <opcion>` o bien ejecutar el comando `service <servicio> <opcion>`

Acción	script	comando
Mostrar	<code>/etc/init.d/&lt;servicio&gt; status</code>	<code>service &lt;servicio&gt; status</code>
Iniciar	<code>/etc/init.d/&lt;servicio&gt; start</code>	<code>service &lt;servicio&gt; start</code>
Detener	<code>/etc/init.d/&lt;servicio&gt; stop</code>	<code>service &lt;servicio&gt; stop</code>
Reiniciar	<code>/etc/init.d/&lt;servicio&gt; restart</code>	<code>service &lt;servicio&gt; restart</code>

## RUNLEVELS

El runlevel (nivel de ejecución) determina el conjunto de servicios que `init` va a ejecutar por defecto. Por ejemplo, en un runlevel se puede iniciar el sistema con interfaz gráfica mientras que en otro no.

runlevel	descripción
<b>0</b>	Parada del sistema
<b>1</b>	Inicio Monousuario. Mantenimiento del sistema
<b>2</b>	Inicio Multiusuario sin soporte de red.
<b>3</b>	Inicio Multiusuario con soporte de red.
<b>4</b>	Inicio no definido. Para definir runlevel personalizado
<b>5</b>	Inicio Multiusuario modo gráfico. Login en modo gráfico (X-Windows)
<b>6</b>	Reinicio del sistema

runlevel	descripción
<b>S</b>	Arranque Base. Se ejecuta antes que cualquier otro runlevel solo durante el arranque del sistema. Configura lo mínimo necesario para que el sistema funcione: monta particiones, verifica sistemas de archivos, inicializa dispositivos, configura la consola,...

Por cada runlevel existe un directorio `rcX.d` (X=runlevel) donde se define el conjunto de servicios que se ejecutan mediante **enlaces simbólicos** a los scripts del directorio `/etc/init.d`. La ubicación de los directorios `rcX.d` varía según la distribución de linux, normalmente: `/etc/rcX.d` o `/etc/rc.d/rcX.d`

Los nombres de los enlaces simbólicos siguen la siguiente estructura `[S|K]NNservicio`

- **S** ejecuta el servicio con la opción start
- **K** ejecuta el servicio con la opción stop
- **NN** nº de orden de ejecución (01 ejecuta antes que 02)
- **servicio** nombre del servicio en `/etc/init.d`

El proceso `init` utiliza el script `/etc/init.d/rc` para ejecutar los runlevels 0-6 y el script `/etc/init.d/rcS` para ejecutar S. Los servicios definidos se ejecutan secuencialmente en orden alfabético. Por tanto, primero se ejecutan los que comienzan por K según su número de orden y luego los que comienzan por S según su número de orden.

## 1 2 3 Enlace simbolico

Ejemplo de enlaces simbólicos para rc3.d. Observar como los servicios ftp o squid no son referenciados por rc3.d.

```
/etc/init.d/
├─ rc
├─ apache2
├─ ssh
├─ network
├─ ftp
├─ squid
/etc/rc3.d/
├─ S01network -> ../init.d/network
├─ S02ssh -> ../init.d/ssh
├─ K01apache2 -> ../init.d/apache2
```

## 1 2 3 Conocer el runlevel actual

Ejecutar comando `runlevel`. Este comando muestra en su salida el runlevel anterior (primer valor) y el runlevel actual (segundo valor).

```
runlevel
N 5
# N runlevel anterior no definido
# 5 runlevel actual multiusuario grafico
```

## 1 2 3 Ejecutar o cambiar un runlevel

Ejecutar comando `init <runlevel>` ó `telinit <runlevel>` .

```
# Ejecutar runlevel de reinicio de maquina
init 6
```

Para **generar los enlaces simbólicos** asociados a un runlevel utilizaremos el comando `update-rc.d` .

### **update-rc.d**

```
# Iniciar y parar un servicio en runlevel por defecto
# (start=2,3,4,5, stop=0,1,6) prioridad 30 start y 15 stop.
update-rc.d <servicio> defaults 30 15
# Iniciar un servicio en runlevel 3 y 5 prioridad 30
update-rc.d <servicio> start 30 3 5
# Detener un servicio en runlevel 3 y 5 prioridad 15
update-rc.d <servicio> stop 15 3 5
```

## INITTAB

El fichero de configuración del proceso `init` es `/etc/inittab`. Le indica al proceso `init` que acciones realizar en el arranque del sistema como por ejemplo: el runlevel predeterminado, el script de arranque base, etc... El fichero está formado por líneas de entrada con el siguiente formato: **id:runlevels:action:process**

- id  
Identificador único para la entrada
- runlevels  
Especifica los runlevels a los que se aplica la entrada. Cuando esta vacío, significa cualquier runlevel.
- action  
Especifica que acción debe realizar init, entre otras tenemos:

action	descripción
respawn	Reiniciar el proceso si está parado
wait	Esperar a que el proceso termine antes de continuar
once	Ejecutar el proceso solo una vez al entrar en runlevel
initdefault	Especifica el runlevel por defecto
sysinit	Ejecutar al inicio antes de cualquier runlevel

- process  
Comando o script a ejecutar

### inittab

```
# /etc/inittab
# runlevel por defecto 3
id:3:initdefault:
# proceso de arranque rcS
si::sysinit:/etc/init.d/rcS
# Niveles de ejecución
# Nivel 0 ejecutar rc 0
l0:0:wait:/etc/init.d/rc 0
# Nivel 1 ejecutar rc 1
l1:1:wait:/etc/init.d/rc 1
....
# Se omiten líneas de la salida
```

## SYSTEMD

El proceso inicial se llama `systemd` y ejecuta el comando `/sbin/systemd` con PID 1. La **ejecución** de los servicios se realiza **en paralelo**, es decir, cuando lanza la ejecución de un servicio no espera a su terminación antes de lanzar un nuevo servicio a no ser que existan dependencias entre ellos. El proceso `systemd` tiene asociado un fichero de configuración global que define su comportamiento `/etc/systemd/system.conf`

Por otro lado, el gestor `systemd` está diseñado para ser compatible con los scripts de inicio de `init` (System V) y por tanto se pueden definir los servicios al estilo de `/etc/init.d`.

## SERVICIOS

Para la definición de los servicios se utiliza el concepto de **unidad**. Una unidad es un objeto de `systemd` que realiza o controla una tarea o acción en particular, por ejemplo, gestionar procesos, organizar el arranque del sistema, crear sockets, montar sistemas de archivos, etc.. Para definir una unidad se utilizan los siguientes elementos:

- **Archivo de configuración**

Archivo que define la tarea que realiza la unidad. Se suelen utilizar los siguientes directorios:

Prioridad	Directorio	Proposito
1	<code>/etc/systemd/system/</code>	Unidades definidas por el usuario administrador del sistema. Se pueden modificar
2	<code>/run/systemd/system</code>	Unidades definidas temporalmente en tiempo de ejecución. Se pierden tras reiniciar
3	<code>/lib/systemd/system/</code>	Unidades definidas por la instalación de paquetes del sistema. No deben modificarse

Si una unidad está definida en más de un directorio el orden de prioridad será: 1->2->3.

- **Nombre**

Identificador único de la unidad dentro del sistema. Es el nombre del archivo de configuración.

- **Categoría**

Se utiliza para agrupar unidades según su funcionalidad. Es la extensión del archivo de configuración. Algunas de ellas son:

Categoría	Descripción
<code>.service</code>	Servicio en el sistema
<code>.target</code>	Conjunto de unidades relacionadas entre sí. Similar a un runlevel
<code>.automount</code>	Punto de montaje automático para un sistema de archivos
<code>.mount</code>	Punto de montaje para un sistema de archivos
<code>.device</code>	Dispositivo físico reconocido por el kernel
<code>.path</code>	Monitoriza cambios en rutas de archivo para el control de los servicios
<code>.scope</code>	Organiza y gestiona procesos que se han creado externamente
<code>.slice</code>	Agrupación de recursos de sistema como la CPU o la memoria
<code>.socket</code>	Socket de comunicación entre procesos
<code>.swap</code>	Dispositivo o archivo de intercambio
<code>.timer</code>	Temporizador. Activa o desactiva un servicio específico basandose en temporizador
<code>.snapshot</code>	Realiza una instantánea del estado actual de todas las unidades en ejecución. Se utiliza para copia y restauración del sistema
<code>.busname</code>	Controla un sistema DBUS, o sistema de comunicación entre procesos

Como podemos ver para definir un servicio se utilizaría la categoría **.service**

## Archivo de configuración de unidad

El archivo `/etc/systemd/system/mi-servicio.service` sería un archivo de configuración de la unidad mi-servicio que pertenece a la categoría service y que ha sido definida por el administrador del sistema.

## Estructura archivo de configuración

Los archivos de configuración de unidades definen los valores que tienen los parámetros de configuración de las unidades agrupados en diferentes secciones.

## Estructura definición unidad

```
[sección]
parámetro=valor
parámetro=valor
....
[sección]
parámetro=valor
....
```

Las secciones pueden variar en función de la categoría de la unidad. Entre las **secciones** para la categoría **.service** tenemos:

- [ Unit ]

Describe unidad y su relación con otras unidades. Algunos parámetros de configuración son:

Parametro	Proposito
Description	Descripción breve de la unidad
Documentation	Indica las URIs donde encontrar información sobre la unidad
Wants	Lista de unidades no críticas que se recomienda iniciar junto con la unidad.
Requires	Lista de unidades críticas que deben estar activas para iniciar la unidad.
BindsTo	Similar a Requires
Before	Indica que la unidad debe iniciarse despues de las unidades listadas
After	Indica que la unidad deben iniciarse despues de las unidades listadas
Conflicts	Lista de unidades con la unidad no debe ejecutarse al mismo tiempo
Condition<nombre>	Condicion que debe cumplir la unidad para iniciarse
Assert<nombre>	Igual que Condition pero se lanza un error sino se cumple

- [ Install ]

Contiene los parámetros relacionados con las acciones que se deben realizar para habilitar o deshabilitar una



unidad. Si la unidad no incluye esta sección se podrá iniciar manualmente pero no se podrá habilitar para el inicio automático. Algunos parámetros de configuración son:

Parametro	Proposito
WantedBy	Lista de unidades (target) que quieren que esta unidad esté activa. Es la contraparte de <code>wants</code> , desde el punto de vista de la unidad dependiente. Se usa para crear enlaces simbólicos en <code>/etc/systemd/system/&lt;target&gt;.wants/</code>
RequiredBy	Igual que <code>WantedBy</code> pero con dependencia. Se usa para crear enlaces simbólicos en <code>/etc/systemd/system/&lt;target&gt;.requires</code>
Alias	Nombre alternativo para la unidad

- [ Service ]

Contiene los parámetros relacionados con las unidades de la categoria servicio. Algunos parámetros de configuración son:

Parametro	Proposito
Type	Tipo de servicio: <code>simple</code> , <code>forking</code> , <code>oneshot</code> , <code>notify</code> , <code>dbus</code> , <code>exec</code>
ExecStart	Comando que se ejecuta para iniciar el servicio.
ExecStartPre	Comando(s) que se ejecutan antes de <code>ExecStart</code> . Si fallan, el servicio no arranca.
ExecStartPost	Comando(s) que se ejecutan después de <code>ExecStart</code> , una vez que el proceso principal arranca correctamente.
ExecReload	Comando que se ejecuta cuando se llama a <code>systemctl reload</code> .
ExecStop	Comando que se ejecuta para detener el servicio (si no se quiere usar SIGTERM).
ExecStopPost	Comando que se ejecuta después de detener el servicio.
Restart	Define si el servicio debe reiniciarse al fallar. Ej: <code>no</code> , <code>on-failure</code> , <code>always</code> , <code>on-abort</code> , etc.
RestartSec	Cuántos segundos esperar antes de reiniciar. Ej: <code>5</code>
KillMode	Define como se terminan los procesos asociados a la unidad cuando esta se detiene. Tenemos los siguientes valores: <code>control-group</code> , <code>process</code> , <code>mixed</code> , <code>none</code>
User	Usuario con el que se ejecuta el servicio.
Group	Grupo con el que se ejecuta.
WorkingDirectory	Directorio de trabajo antes de ejecutar el servicio.
Environment	Define variables de entorno. Ej: <code>Environment="PORT=3000"</code>
EnvironmentFile	Archivo desde el cual cargar variables de configuración del servicio. Ej: <code>/etc/default/mi-app</code>
TimeoutStartSec	Tiempo máximo para que el servicio arranque.

Parametro	Proposito
TimeoutStopSec	Tiempo máximo para que el servicio se detenga.
RemainAfterExit	Útil para servicios <code>oneshot</code> ; si se establece en <code>yes</code> , el servicio se considera activo tras la ejecución.
GuessMainPID	Solo para <code>forking</code> : indica si systemd debe intentar adivinar el PID principal.
CapabilityBoundingSet	Limita las capacidades Linux (capabilities) disponibles para el servicio.
ProtectSystem	Establece protección de solo lectura en partes del sistema de archivos.
NoNewPrivileges	Evita que el proceso y sus hijos ganen nuevos privilegios.
PrivateTmp	Si se pone <code>yes</code> , le da al servicio su propio <code>/tmp</code> aislado.



## Definición servicio ssh

```
#archivo de unidad en /lib/systemd/system/ssh.service

[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Install]
WantedBy=multi-user.target
Alias=sshd.service

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

# La variable SSHD_OPTS se encuentra definida en /etc/ default/ssh.
# La variable MAINPID se crea al cargarse la unidad
```

## RUNLEVELS

Systemd emula los niveles de ejecución de System V a través de unidades de tipo target. Las equivalencias son las siguientes:

RunLevel	Descripción	Unidad	Alias
0	Apagar sistema	poweroff.target	runlevel0.target
1	Inicio monousuario	rescue.target	runlevel1.target
2	Inicio multiusuario sin red	multi-user.target	runlevel2.target
3	Inicio multiusuario con red	multi-user.target	runlevel3.target
4	Inicio personalizado	-	-

RunLevel	Descripción	Unidad	Alias
5	Inicio multiusuario gráfico	graphical.target	runlevel5.target
6	Reinicio	reboot.target	runlevel6.target
-	Emergencia	emergency.target	-

## OPERACIONES

Para realizar operaciones sobre las unidades y servicios en `systemd` utilizaremos el comando `systemctl`. Este comando tiene numerosas opciones que puedes consultar en la siguiente [url](#). Entre ellas tenemos:

Operación	Comando
Iniciar	<code>systemctl start &lt;servicio&gt;</code>
Parar	<code>systemctl stop &lt;servicio&gt;</code>
Reiniciar	<code>systemctl restart &lt;servicio&gt;</code>
Mostrar estado	<code>systemctl status &lt;servicio&gt;</code> - enable Unidad habilitada automáticamente - static Unidad ejecutada manualmente o por otra unidad - disabled Unidad deshabilitada - masked Unidad deshabilitada y no se puede ejecutar ni siquiera manualmente
Recargar configuración	<code>systemctl reload &lt;servicio&gt;</code>
Habilitar	<code>systemctl enable &lt;servicio&gt;</code>
Deshabilitar	<code>systemctl disable &lt;servicio&gt;</code>
Enmascarar	<code>systemctl mask &lt;servicio&gt;</code>
Desenmascarar	<code>systemctl umask &lt;servicio&gt;</code>
Activo?	<code>systemctl is-active &lt;servicio&gt;</code>
Habilitado?	<code>systemctl is-enabled &lt;servicio&gt;</code>
Fallo?	<code>systemctl is-failed &lt;servicio&gt;</code>
Dependencias after	<code>systemctl list-dependencies --after &lt;servicio&gt;</code>
Dependencias before	<code>systemctl list-dependencies --before &lt;servicio&gt;</code>
Mostrar fichero configuración	<code>systemctl cat &lt;servicio&gt;.service</code>
Mostrar propiedades configuración	<code>systemctl show &lt;servicio&gt;.service</code>
Editar fichero configuración	<code>systemctl edit --full --force &lt;servicio&gt;.service</code>

Operación	Comando
Mostrar runlevel por defecto	<code>systemctl get-default</code>
Cambiar runlevel por defecto	<code>systemctl set-default &lt;runlevel.target&gt;</code> Ej. runlevel multiuser <code>systemctl set-default multi-user.target</code>
Cambiar runlevel	<code>systemctl set-default isolate &lt;runlevel.target&gt;</code>
Mostrar unidades del sistema	<code>systemctl list-units</code> <code>systemctl list-units-files</code>
Mostrar servicios del sistema	<code>systemctl list-units --type=service [--state={active inactive failed dead} ]</code> Ej. Servicios inactivos <code>systemctl list-units --type=service --state=active</code>
Aplicar cambios sin reiniciar	<code>systemctl daemon-reload</code>