1 Servicios en GNU/Linux

Un **demonio o servicio** es un proceso que se ejecuta en **segundo plano**, fuera del control interactivo de los usuarios del sistema, ya que carecen de interfaz con estos (se auto-gestionan). Se encargan de supervisar el sistema, proporcionar funcionalidad a otros procesos y atender sus peticiones, y responder ante cierta actividad del hardware.

El término demonio se usa fundamentalmente en sistemas basados en UNIX, como GNU/Linux o Mac OS. En Windows y otros sistemas operativos se denominan servicios, porque fundamentalmente son programas que ofrecen servicios al resto del sistema.

Como ya sabe, el sistema generalmente inicia los demonios durante el arranque, una vez que se carga el kernel. Así, por ejemplo, *acpid* es un demonio que maneja el apagado del sistema cuando el usuario pulsa el botón de encendido del equipo, *y cron* es otro demonio que se ejecuta para lanzar tareas planificadas. Tradicionalmente en sistemas UNIX y derivados, los nombres de los demonios terminaban con la letra "*d*", por lo que podemos encontrar nombres de demonios como *syslogd,* que es el demonio que implementa el registro de eventos del sistema, o *sshd,* que es el demonio que sirve a las conexiones SSH entrantes (acceso a una máquina mediante conexión remota).

Puesto que un demonio se ejecuta en segundo plano, **no debe estar conectado a ninguna terminal o *shell***, ya que cualquier proceso que se ejecute desde el *shell* o interprete de comandos es hijo de este (la *shell* lanza un *fork()* y posteriormente un *exec(comando)*), y si se cierra la terminal se cerraría el demonio. Por lo tanto los procesos que quieren convertirse en un demonio deben asegurarse que siempre se ejecutan en segundo plano, desvinculándose del proceso de la *shell* que los invocó. Esto plantea la cuestión de cómo indicar condiciones de error, advertencias u otro tipo de sucesos del servicio. Algunos demonios almacenan estos mensajes en archivos específicos o en su propia base de datos de sucesos. Sin embargo en muchos sistemas existe un servicio especifico (otro demonio) para registrar estos eventos, s*yslogd*, al que otros procesos pueden enviar mensajes a través de la función *syslog()*.

```
// Abrir una conexión al demonio syslog. Puede consultar Posix para una completa descripción
openlog(argv[0], LOG_NOWAIT | LOG_PID, LOG_USER);
...
// Enviar un mensaje al demonio syslog
syslog(LOG_NOTICE, "Demonio iniciado con éxito\n");
...
// Cuando el demonio termine, cerrar la conexión con el servicio syslog
closelog();
```

Todas las distribuciones de Linux utilizan archivos de registro para registrar los eventos del sistema, incluyendo conectar dispositivos, sesiones nuevas y otros mensajes. La mayoría de las distribuciones guardan estos archivos en el directorio **/var/log**. Algunos de los archivos están protegidos, hace falta iniciar una sesión como usuario ***root*** o usar ***sudo*** para leerlos.

Los archivos de registro varían por distribución. No obstante, la mayoría de las distribuciones GNU/Linux suelen tener los siguientes archivos:

- **auth.log**: Información sobre eventos de autenticación de usuarios.
- **boot.log**: Muestra eventos y servicios empezados cuando se inicia el sistema.
- **crond.log**: Tareas de *cron.*
- **daemon.log**: Alertas de servicios como *ntfs-3g* o *dhcpd*.
- **dmesg.log**: en este archivo se almacena la información que genera el kernel durante el arranque del sistema. Podemos ver su contenido con el comando ***dmesg***.
- **mysqld.log**: Archivo de MySQL.
- **syslog.log**: Registro del sistema de registro.

1.1 Arranque de servicios con la versión clásica *SysV init*

Los demonios se pueden lanzar a través del proceso *init* (*/sbin/init*), que es la versión clásica al estilo de Unix System-V, también conocida como ***SysV init.***

El proceso *init* es llamado por el núcleo de GNU/Linux para iniciar el espacio de usuario durante el proceso de arranque y gestionar posteriormente todos los demás procesos a nivel de usuario. Por tanto *init* es el primer proceso demonio que se inicia a nivel de usuario (se ejecuta en el espacio de usuario), y su función principal es lanzar el resto de demonios necesarios a partir de *scripts* escritos en *bash.*

El demonio *init* de muchos sistemas operativos opera indicando un nivel de ejecución, que define un estado de la máquina después del arranque, estableciendo qué otros demonios deben ser iniciados por *init* a través del lanzamiento de los citados *scripts*. Por lo general existen 7 niveles (del 0 al 6), en términos prácticos cuando el computador entra al *runlevel* 0 está apagado, y cuando entra al *runlevel* 6 se reinicia. Los *runlevels* intermedios (1 a 5) difieren en relación a qué unidades de disco se montan y qué servicios son iniciados. Los niveles más bajos se utilizan para el mantenimiento o la recuperación de emergencia, ya que por lo general no ofrecen ningún servicio de red.

Los *scripts* de *bash* que puede lanzar *init*, que a su vez ejecutarán ficheros binarios, por ejemplo de /sbin, /bin, /usr/sbin, /usr/bin, etc, e incluso cargarán módulos del kernel, se localizan en el directorio ***/etc/init.d/.***

Por otro lado, los *scripts* de ***/etc/init.d/*** se encuentran enlazados mediante enlaces simbólicos (*symlink*) desde directorios con nombres como ***/etc/rc0.d/*** a ***/etc/rc6.d/***, uno para cada nivel. Por ejemplo, en ***/etc/rc3.d*** se encuentran los enlaces simbólicos a los *scripts* del directorio ***/etc/init.d/*** que se lanzarán en el nivel 3.

Los detalles particulares de configuración del *runlevel* varía entre sistemas operativos. LSB o *Linux System Standard* es una iniciativa de la *Linux Foundation* para reducir las diferencias entre las distintas distribuciones de Linux. La mayoría de las distribuciones Linux, definen los niveles de ejecución mostrados en la siguiente tabla:

Nivel de ejecución	Nombre o denominación	Descripción
0	Alto	Alto o cierre del sistema (apagado), mediante el comando *halt*. No se debe poner este nivel como predeterminado.
1	Modo monousuario	No configura la interfaz de red o los demonios de inicio, ni permite que ingresen otros usuarios que no sean el usuario *root*. Este nivel de ejecución permite reparar problemas, o hacer pruebas en el sistema.
2	Multiusuario	Multiusuario sin soporte de red (sin NFS o xinetd - demonio de servicios extendidos de Internet-)
3	Multiusuario con soporte de red	Inicia el sistema normalmente. GNU/Linux completamente funcional con soporte multiusuario y acceso a la red. La interfaz de usuario es en modo texto.
4	Multiusuario con soporte de red.	Similar al 3 o sin uso, reservado para personalización para administradores de sistemas.
5	Multiusuario gráfico (X11)	Nivel de ejecución 3 + display manager (entorno gráfico usando gestores de pantalla como *gdm - GNOME Display Manager*-). Normalmente siempre se arranca en este nivel.
6	Reinicio	Se reinicia el sistema. No se debe poner este nivel como predeterminado.

Niveles de ejecución generales en GNU/Linux

El archivo de configuración de más alto nivel para el proceso *init* es ***/etc/init/rc-sysinit.conf,*** aunque en otros sistemas el primer fichero que lee *init* es ***/etc/inittab.*** Independientemente de eso ambos ficheros de configuración tienen el mismo fin para *init*, saber qué directorio ***rc*** con enlaces simbólicos a *scripts* almacenados en ***/etc/init.d/*** debe utilizar. Por tanto, durante el arranque del sistema se verifica si existe un nivel de ejecución predeterminado en uno de los ficheros citados, y después se procede a ejecutar todos los enlaces simbólicos del directorio ***/etc/rc?.d/*** relativos al nivel de ejecución especificado.

Después de que se han lanzado todos los *scripts*, el proceso *init* se aletarga y espera a que uno de estos tres eventos sucedan:

- Que procesos comenzados finalicen o mueran.
- Un fallo de la señal de potencia (energía).
- Una petición a través del proceso ***/sbin/telinit*** para cambiar el nivel de ejecución.

En el apagado, *init* es llamado a cerrar toda las funcionalidades del espacio de usuario de una manera controlada, de nuevo a través de secuencias de comandos o *scripts*, tras lo cual *init* termina y el núcleo ejecuta el apagado.

Aparte de poder editar los archivos de configuración de más alto nivel para el proceso *init* para ejecutar por defecto un nivel determinado en el arranque del sistema, el administrador puede cambiar en cualquier momento el valor del nivel de ejecución con el comando ***telinit***. Una instrucción como la siguiente:

telinit 3

cambia a nivel de ejecución 3. Si se hace esto se dejará de tener una pantalla gráfica y aparecerá una terminal, y es probable que se pierda la información que no se tenga guardada. Para regresar al modo gráfico es necesario establecer de nuevo a 5. El comando ***telinit*** no altera el contenido de los archivos de configuración de más alto nivel (***/etc/init/rc-sysinit.conf*** o ***/etc/inittab***).

¿Cómo podemos saber cuál es el nivel de ejecución actual del sistema?

- Con el comando ***who*** y la opción -***r***
- Con el comando ***runlevel***.

Para añadir nuevos demonios en un nivel de arranque solo hay que introducir su correspondiente s*cript* en la carpeta ***/etc/init.d/*** y actualizar los enlaces simbólicos de ***/etc/rc0.d*** a ***/etc/rc6.d***, según el nivel**.** Antes de hacer esto, ya sea a partir de demonios ya implementados o si usted quiere hacer uno desde cero, debería consultar cuales son las recomendaciones para añadir y construir nuevos demonios y no obtener comportamientos inesperados.

Una vez que que se tiene un *script* para usarlo como demonio puede automatizar el proceso de enlazado simbólico mediante el comando ***update-rc.d.*** Si por ejemplo deseamos que el *script* de servicio de nombre *servicio* se ejecute en el arranque:

1. Se pone *servicio* en el directorio ***/etc/init.d/***
2. Después se crean los enlaces simbólicos mediante el comando
   # ***update-rc.d servicio defaults 35***

   Al pasar el parámetro *defaults* se fuerza a que el servicio se cree para los niveles de ejecución que van del 2 al 5. Con el 35 se obliga a que *servicio* se arranque antes de cualquier *script* que contenga un número mayor de 36 (es una manera de gestionar el orden).

Por otro lado, también puede consultar los servicios activos en su máquina con el comando ***service****,* por ejemplo con la opción ***–status-all****:*

service –status-all

No debe confundir este comando y su salida con el comando ***pstree -p*** (visualizar el árbol de procesos, mostrando la relación padre hijo y sus PID), el cual muestra todos los procesos a nivel de usuario, los cuales son todos hijos del proceso *init*. No todos los procesos de usuario son servicios o demonios, por lo que habrá procesos al ejecutar ***# service –status-all*** que no verá reflejados al ejecutar ***# pstree -p.***

Puede ver qué parámetros admite un determinado servicio pasando al comando ***service*** el nombre del servicio (por ejemplo *bluetooth*):

 # service bluetooth

Usage: /etc/init.d/bluetooth {start|stop|restart|force-reload|status}

En el caso anterior *bluetooth* se podría parar indicándolo al comando ***service*** así:

 # service bluetooth stop

Actualmente el proceso de arranque con *init* se está sustituyendo por el nuevo sistema de inicio *systemd,* pero este último se ha hecho compatible hacia atrás con *init,* con lo que aún se puede gestionar el arranque del espacio de usuario y sus servicios de la manera tradicional.

Algunos de los servicios o demonios más usuales lanzados por *init* o *systemd* en el arranque del sistema son los siguientes:

- **Servicio de red:** Demonio necesario para que funcione la interfaz de red, independientemente del protocolo utilizado.
- **Servidor web Apache:** Demonio necesario para poder acceder a Internet mediante conexiones *http*.
- **Servidor Samba:** Demonio para poder acceder a carpetas, archivos e impresoras en red, independientemente de que sean de Windows o GNU/Linux.
- **Servidor de correo:** Demonio para poder aceptar conexiones de correo electrónico
- **Manejador de base de datos MySQL y PostgreSQL:** Demonio que permite el acceso a las bases de datos de nuestro sistema.
- **Programador de tareas:** Demonio para lanzar tareas programadas en el tiempo.
- **Servicio de impresión:** Demonio necesario para poder enviar y recibir información de las impresoras.
- **Servicio de descubrimiento de hardware:** Demonio necesario para la detección de hardware.
- **Servicio de gestión de energía:** Demonio necesario para el control de la energía y acciones a llevar a cabo al pulsar determinados botones en nuestro ordenador.
- **Servidor de sonido:** Demonio necesario para poder activar y controlar el sistema de sonido.
- **Servicios de automontaje del sistema de archivos:** Demonios para montar en el arranque determinados sistemas de ficheros (ntfs, fat, ext).
- **Servicios de monitoreo**: Demonios para hacer auditorias y sensores, como por ejemplo la frecuencia de trabajo de la CPU.
- **Servicios para bluetooth e infrarojos:** Demonios encargados de las conexiones hardware por Bluetooth e infrarrojos (irDA).
- *Un largo etc.*

1.2 Arranque de servicios con el sistema systemd

Systemd fue diseñado para reemplazar al mencionado *init* como primer y último proceso que se ejecuta en el espacio de usuario y del que dependen los demás procesos a ese nivel de ejecución.

El principal motivo de la aparición de *systemd* es unificar para todas las distribuciones de GNU/Linux las configuraciones básicas de la administración de servicios y la conexión entre las aplicaciones y el núcleo, y desde 2015 la mayoría de las distribuciones ya lo usan.

Systemd no es un simple reemplazo para *init* puesto que integra más partes del sistema, es un "**super servicio**", que además de permitir en el arranque la ejecución paralela de procesos (el demonio *init* tradicional es estrictamente síncrono, bloqueando futuras tareas hasta que la actual se haya completado), intenta reemplazar varios servicios del sistema, como por ejemplo *journal*, que es el sistema de registros o *log* de *systemd* (reemplaza o convive con *syslog*). En enero de 2013, Poettering, su autor, describió a *systemd* como una *suite* o conjunto de aplicaciones que incluía 69 binarios individuales.

Esto ha generado gran controversia dentro de la comunidad del software libre. Los críticos se argumentan en al menos dos niveles:

1. **Arquitectura interna:** Critican que *systemd* ya es demasiado complejo y que sufre de una continua invasión de nuevas características, absorbiendo cada vez más funciones, procesos y demonios, además de ser más complicado de administrar que *SysV init*.
2. **Implementación futura:** También existe la preocupación de que se torne en un sistema de dependencias entrelazadas, obligando así a los mantenedores de distribuciones a no tener más remedio que depender de *systemd* a medida que más piezas de software del espacio de usuario sigan dependiendo de sus componentes.

En *systemd* los demonios se definen y configuran en los llamados **archivos de unidad** o ***unit files***, que son ficheros con un tipo de extensión definida y que se alojan en varios directorios, esto sería parecido a los ficheros de *scripts* en *bash* situados en ***/etc/init.d/***

- ***/etc/systemd/system/***: Unidades instaladas por el administrador del sistema. Aquí se deberían colocar unidades de nueva creación.
- ***/usr/lib/systemd/system/:*** Otras unidades instaladas por paquetes instalados.

Para definir un demonio solo hay que crear una unidad o ***unit file*** con la forma ***unit_name.type_extension***, y editarlo siguiendo una serie de reglas y nomenclatura, entre ellas indicar la ruta del verdadero fichero binario que actuará como demonio o incluso otro *script* que se ha de ejecutar. Por tanto en las unidades diferenciamos por una parte el nombre del demonio o unidad con el *unit_name* y por otro lado el tipo de extensión *type_extension* (hay 12 extensiones según la funcionalidad del demonio). La extensión *.service* es la que se puede usar para definir demonios propios. Además de esto, pueden existir carpetas que acompañan a los *unit files* para añadir configuraciones adicionales.

En *systemd* ya no existen diferentes niveles de arranque como en *init*, eso se ha reemplazado por los denominados *target*, que son ficheros de unidad con extensión *.target* en los que se indican

conjuntos de ficheros de unidad *.service* e incluso otros ficheros *.target*. Es decir, los ficheros *.target* agrupan en cadenas de dependencias a otras unidades de ***systemd***. El nivel o *target* por defecto, que suele ser *graphical.target* se indica en el fichero ***/lib/systemd/system/default.target***.

En la siguiente tabla se muestran algunas equivalencias entre los niveles de *SysV init* y Syst*emd.* Recuerde que lo que se estime arrancar en cada nivel difiere entre distribuciones, por tanto tómelo como algo genérico y orientativo*:*

Nivel SysV	**Unidades *.target***	**Descripción**
0	runlevel0.target, poweroff.target	Alto o cierre del sistema (apagado), mediante el comando *halt*. No se debe poner este nivel como predeterminado.
1	runlevel1.target, rescue.target	No configura la interfaz de red o los demonios de inicio, ni permite que ingresen otros usuarios que no sean el usuario *root*. Este nivel de ejecución permite reparar problemas, o hacer pruebas en el sistema.
2	runlevel2.target, multi-user.target	Multiusuario sin soporte de red (sin NFS o xinetd - demonio de servicios extendidos de Internet-)
3	runlevel3.target, multi-user.target	Inicia el sistema normalmente. GNU/Linux completamente funcional con soporte multiusuario y acceso a la red. La interfaz de usuario es en modo texto.
4	runlevel4.target, multi-user.target	Similar al 3 o sin uso, reservado para personalización para administradores de sistemas.
5	runlevel5.target, graphical.target	Nivel de ejecución 3 + display manager (entorno gráfico usando gestores de pantalla como *gdm - GNOME Display Manager*-). Normalmente siempre se arranca en este nivel.
6	runlevel6.target, reboot.target	Se reinicia el sistema. No se debe poner este nivel como predeterminado.

En cuanto a los 12 tipos de unidades pueden ser los siguientes (si desea saber en profundidad para lo que sirve y cómo gestionar cada tipo de unidad consulte la Web y bibliografía relacionada al respecto):

1. **Service**: Una unidad ***.service*** controla demonios y los procesos relativos a ellos. Es el tipo de unidad que interesa para crear un nuevo servicio.
2. **Target**: Las unidades *.target* codifican información respecto a agrupar unidades y lograr una buena sincronización entre ellas. Se encargan de las dependencias entre unidades y sus relaciones entre si.
3. **Socket**: Codifica información respecto a un *IPC* (*inter-process communication) network socket.* Establece un canal de comunicación entre diferentes unidades.
4. **Device**: Las unidades *.device* codifican información respecto a dispositivos físicos incluidos en *sysfs/udev*.
5. **Mount**: Las unidades *.mount* codifican información respecto a un punto de montaje. Permiten montar o desmontar un sistema de archivos.
6. **Automount**: Las unidades *.automount* encapsulan las funcionalidades de montaje de *.mount*, pero de manera automática en el arranque.

7. **Snapshot**: Las unidades *.snapshot* hacen referencia a una instantánea dinámica del estado de ejecución del sistema *systemd*. Son útiles para hacer retroceder a un estado definido después de iniciar/detener temporalmente servicios.
8. **Timer**: Las unidades *.timer* codifican la información acerca de eventos al estilo *cron*.
9. **Swap**: Las unidades *.swap* codifican la información acerca de espacios de intercambio.
10. **Path**: Utilizadas para activación de unidades *.service* basándose en eventos del sistema de archivos referenciado por el *path.*
11. **Scope**: Las unidades *.scope* gestionan conjuntos de procesos creados externamente.
12. **Slice**: Las unidades *.slice* gestionan conjuntos de procesos con la posibilidad de configurarse ciertos limites a los recursos que se aplican a esos procesos.

Un ejemplo abstracto de un fichero *unit_name.service*:

```
[Unit]
Description=servicio de ejemplo
After=network.target mysql.service

[Service]
User=www-data
ExecStart=/ruta/script/demonio
Restart=on-failure

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

- [Unit]: En este apartado se establecen las opciones genéricas que no dependen del tipo de extensión.
- [Service]: En este apartado se establecen las opciones específicas para el tipo de extensión elegido.
- [Install]: Contiene información sobre la instalación del servicio que afecta a la ejecución de los comandos ***systemctl enable*** y ***systemctl disable*** (activar y desactivar servicio o unidad). Aquí se indicará si la unidad se cargará o no en el arranque.

El comando ***service*** de *init* se sustituye o convive con el comando ***systemctl*** de *systemd.* Una vez que se crea un demonio mediante la creación de una unidad, para activarlos es necesario ejecutar el comando ***# systemctl enable*** seguido del *unit_name* (en modo *root*), por ejemplo:

sudo systemctl enable unit_name.service

También se dispone del comando ***# systemctl disable unit_name.service*** para eliminar el servicio de la secuencia de arranque.

Otros comandos que pueden ser útiles para trabajar con servicios *systemd:*

- ***systemctl get-default:*** Indica la unidad *.target* o nivel actual del sistema.

- ***systemctl list-units --type target***: Indica los *.target* cargados en el sistema. Como se ha dicho, los *.target* cargados implicarán que están cargadas una serie de unidades de tipo *.service* que dependen de las *.target*.
- ***sudo systemctl set-default unit_name.target***: Establece a *unit_name.target* el nivel *.target* por defecto en el arranque del sistema.
- ***systemctl list-units --type service:*** Permite obtener un listado de la totalidad de servicios que actualmente están activos:
  - *UNIT*: El nombre de la unidad.
  - *LOAD*: Si la información de la configuración de la unida está cargada en memoria.
  - *ACTIVE*: Informa de si la unidad está o no activa.
  - *SUB*: Muestra más información sobre la unidad.
  - *DESCRIPTION*: Una descripción breve de lo que hace la unidad.
- ***systemctl status unit_name.service:*** Permite ver la información de estado sobre el servicio en cuestión:
  - *Loaded* - Información sobre si la unidad servicio ha sido cargada y la ruta absoluta.
  - *Active* - Información sobre si la unidad de servicio está ejecutándose, con *timestamp* incluido.
  - *Main PID* - El PID del servicio, seguido de su nombre.
  - *Process* - Información adicional sobre procesos relacionados.
  - *CGroup* - Información adicional sobre si ese servicio pertenece a un determinado grupo de servicios.
- ***systemctl list-units --type service –all:*** Permite obtener un listado de la totalidad de servicios, independientemente de si están activos o inactivos.
- ***sudo systemctl stop unit_name.service:*** Permite detener el servicio.
- ***sudo systemctl restart unit_name.service:*** Permite reiniciar el servicio.
- ***sudo systemctl daemon-reload:*** Permite recargar todos los servicios de nuevo, útil si modificamos uno de ellos.
- ***journalctl -u unit_name.service:*** Permite ver el registro, o *log*, generado por el servicio.