SERVICIOS LINUX

Descargar PDF

INTRODUCCIÓN

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 explicita 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 parar comprobar el estado el servicio
;;
*)
```

```
#Cualquier otra opción nuestra 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	`/etc/init.d/<servicio> status`	`service <servicio> status`
Iniciar	`/etc/init.d/<servicio> start`	`service <servicio> start`
Detener	`/etc/init.d/<servicio> stop`	`service <servicio> stop`
Reiniciar	`/etc/init.d/<servicio> restart`	`service <servicio> restart`

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
0	Parada del sistema
1	Inicio Monousuario. Mantenimiento del sistema
2	Inicio Multiusuario sin soporte de red.
3	Inicio Multiusuario con soporte de red.
4	Inicio no definido. Para definir runlevel personalizado
5	Inicio Multiusario modo gráfico. Login en modo gráfico (X-Windows)
6	Reinicio del sistema
S	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 (01,02,03,...)

- **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 alfabetico. 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.

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
```

Conocer el runlevel actual

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

Ejecutar o cambiar un runlevel

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

Para **generar los enlaces simbolicos** 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 lineas 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 vacio, 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 lineas de la salida
```

FICHEROS DE LOG

`System V` no utiliza un sistema centralizado de registro de logs (bitácora), depende de otros servicios del sistema para registar eventos, especialmente el sistema de logging tradicional de linux `syslog`. Esto quiere decir que los mensajes de los servicios de `/etc/init.d/` no se registran automáticamente a no ser que utilicen llamadas a `syslog`.

El sistema de logging `syslog` es gestionado tradicionalmente por el demonio `syslogd` o en distribuciones más modernas `rsyslogd`, `syslog-ng` o `busybox-syslog`.

Los registros de logging se escriben en archivos dentro del directorio `/var/log/`. Entre los archivos más comunes tenemos:

archivo log	descripción
`/var/log/messages`	Registro general del sistema
`/var/log/syslog`	Registro general en distribuciones Debian, Ubuntu,..
`/var/log/auth.log`	Registro de autenticaciones y sudo
`/var/log/dmesg`	Registro del kernel (hardware,arranque)
`/var/log/boot.log`	Registro del proceso de arranque
`/var/log/daemon.log`	Registro de servicios o demonios
`/var/log/<servicio>`	Registro particular de un servicio Ej. `/var/log/ssh`

Para consultar los mensajes de log podemos usar los siguientes comandos: `less`, `tail`, `grep` ...

consulta de syslog

```
#less consultar el fichero syslog
#para terminar pulsar q
less /var/log/syslog

#tail consultar ultimos registros de syslog
tail /var/log/syslog

#tail consultar ultimos 10 registros de syslog
tail --lines 10 /var/log/syslog

#tail consultar ultimos registros de syslog y continuar a la escucha
#para terminar pulsar CTRL+C
tail --follow /var/log/syslog

#tail consultar todos los registros de syslog
tail --lines +1 /var/log/syslog

#tail consultar ultimos registros de syslog solo para
#el servicio dhclient
tail /var/log/syslog | grep "dhclient"

#tail consultar los ultimos registros que hayan dado error
tail /var/log/syslog | grep "error"

#tail consultar todos los registros de syslog para
#el servicio dhclient en el mes de octubre que hayan dado error
tail --lines +1 /var/log/syslog | grep "dhclient" | grep "Oct" | grep "error"
```

De la misma manera que hemos consultado /var/log/syslog podriamos haber realizado la consulta sobre cualquier otro fichero de log.

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	`/etc/systemd/system/`	Unidades definidas por el usuario administrador del sistema. Se pueden modificar
2	`/run/systemd/system`	Unidades definidas temporalmente en tiempo de ejecución. Se pierden tras reiniciar
3	`/lib/systemd/system/`	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.
- **Categoria**
  Se utiliza para agrupar unidades según su funcionalidad.
  Es la extensión del archivo de configuración. Algunas de ellas son:

Categoria	Descripción
.service	Servicio en el sistema
.target	Conjunto de unidades relacionadas entre sí. Similar a un runlevel
.automount	Punto de montaje automático para un sistema de archivos
.mount	Punto de montaje para un sistema de archivos
.device	Dispositivo físico reconocido por el kernel

Categoria	Descripción
.path	Monitoriza cambios en rutas de archivo para el control de los servicios
.scope	Organiza y gestiona procesos que se han creado externamente
.slice	Agrupación de recursos de sistema como la CPU o la memoria
.socket	Socket de comunicación entre procesos
.swap	Dispositivo o archivo de intercambio
.timer	Temporizador. Activa o desactiva un servicio específico basandose en temporizador
.snapshot	Realiza una instantánea del estado actual de todas las unidades en ejecución. Se utiliza para copia y restauración del sistema
.busname	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 categoria 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 categoria de la unidad. Entre las **secciones** para la categoria **.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

Parametro	Proposito
`Wants`	Lista de unidades no críticas que se desean que esten activas para iniciar la unidad.
`Requires`	Lista de unidades críticas que se requiere que esten activas para iniciar la unidad.
`BindsTo`	Similar a `Requires`
`Before`	Indica que la unidad debe iniciarse antes 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 podra 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 `wants`, desde el punto de vista de la unidad dependiente. Se usa para crear enlaces simbólicos en `/etc/systemd/system/<target>.wants/`
`RequiredBy`	Igual que `WantedBy` pero con dependencia. Se usa para crear enlaces simbólicos en `/etc/systemd/system/<target>.requires`
`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: `simple`. El servicio ejecuta en primer plano un único proceso `forking`. El servicio ejecuta un proceso principal o padre que lanza un proceso hijo, normalmente en segundo plano (daemon), y termina. `oneshot`. El servicio ejecuta un proceso que termina inmediatamente al ejecutar su tarea. `notify`. El servicio ejecuta un proceso que envía una notificación a systemd cuando ha terminado de inicializarse y de esa manera pasar a estado `active`. `dbus`. El servicio lanza un proceso que se considera iniciado cuando se registra un nombre en el bus de D-Bus. `exec`. Similar a simple pero más estricto. `idle`. Como `simple` pero systemd espera a que el resto del arranque del sistema termine antes de ejecutar el servicio.
`ExecStart`	Comando que se ejecuta para iniciar el servicio.

Parametro	Proposito
`ExecStartPre`	Comando(s) que se ejecutan antes de `ExecStart`. Si fallan, el servicio no arranca.
`ExecStartPost`	Comando(s) que se ejecutan después de `ExecStart`, una vez que el proceso principal arranca correctamente.
`ExecReload`	Comando que se ejecuta cuando se llama a `systemctl reload`.
`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: `no`, `on-failure`, `always`, `on-abort`, etc.
`RestartSec`	Cuántos segundos esperar antes de reiniciar. Ej: `5`
`KillMode`	Define como se terminan los procesos asociados a la unidad cuando ésta se detiene. Pare ello se envía señales `SIGTERM` y/o `SIGKILL` a los procesos. Tenemos los siguientes valores: `control-group`. Termina el cgroup ,es decir, el proceso y todos los recursos y procesos hijos que dependen de el. `process`. Termina solo el proceso principal. `mixed`. Termina primero el proceso principal y si no termina en un tiempo termina el cgroup. `none`. No envia ninguna señal de terminación a los procesos.
`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: `Environment="PORT=3000"`
`EnvironmentFile`	Archivo desde el cual cargar variables de configuración del servicio. Ej: `/etc/default/mi-app`
`TimeoutStartSec`	Tiempo máximo para que el servicio arranque.
`TimeoutStopSec`	Tiempo máximo para que el servicio se detenga.
`RemainAfterExit`	Útil para servicios `oneshot`, si se establece en `yes`, el servicio se considera activo tras la ejecución.
`GuessMainPID`	Solo para `forking`. 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 `yes`, le da al servicio su propio `/tmp` 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	-	-
5	Inicio multiusuario gráfico	graphical.target	runlevel5.target
6	Reinicio	reboot.target	runlevel6.target
-	Emergencia	emergency.target	-

La manera más habitual de asociar un servicio a un determinado runlevel es añadir una propiedad WantedBy en la sección install del servicio.

servicio ssh asociado a runlevel multi-user

```
#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
....
# Al asignar la propiedad WantedBy a multi-user.target se crea un enlace simbólico
# al servicio en el  directorio /etc/system/systemd/multi-user.target.wants
# De forma similar sucedería si en lugar de WantedBy se utiliza RequieredBy.
```

OPERACIONES

Para realizar operaciones sobre las unidades y servicios en `systemd` utilizaremos el comando `systemctl`. Este comando tiene numerosas opciones que puedes consultar en man systemctl. Entre ellas tenemos:

Operación	Comando
Iniciar	`systemctl start <servicio>`
Parar	`systemctl stop <servicio>`
Reiniciar	`systemctl restart <servicio>`
Mostrar estado	`systemctl status <servicio>`
Recargar configuración	`systemctl reload <servicio>`
Habilitar	`systemctl enable <servicio>`
Deshabilitar	`systemctl disable <servicio>`
Enmascarar	`systemctl mask <servicio>`
Desenmascarar	`systemctl umask <servicio>`
Activo?	`systemctl is-active <servicio>`
Habilitado?	`systemctl is-enabled <servicio>`
Fallo?	`systemctl is-failed <servicio>`
Dependecias after	`systemctl list-dependecies --after <servicio>`
Dependecias before	`systemctl list-dependecies --before <servicio>`
Mostrar fichero configuración	`systemctl cat <servicio>.service`

Operación	Comando
Mostrar propiedades configuración	`systemctl show <servicio>.service`
Editar fichero configuración	`systemctl edit --full --force <servicio>.service`
Mostrar runlevel por defecto	`systemctl get-default`
Cambiar runlevel por defecto	`systemctl set-default <runlevel.target>`
Cambiar runlevel	`systemctl isolate <runlevel.target>`
Mostrar unidades sistema	`systemctl list-units` `systemctl list-units-files`
Mostrar servicios sistema	`systemctl list-units --type=service [--state={active|inactive|failed|dead|running}]`
Aplicar cambios sin reiniciar	`systemctl daemon-reload`

También podemos utilizar para iniciar, parar, pausar o mostrar el estado de un servicio en lugar de `systemctl` el comando `service`.

Mostrar estado de un servicio

```
systemctl status ssh

ssh.service - OpenBSD Secure Shell server
Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
Active: active (running) since Thu 2025-10-09 14:18:06 CEST; 6h ago
  Docs: man:sshd(8)
        man:sshd_config(5)
Process: 384 ExecStartPre=/usr/sbin/sshd -t
(code=exited, status=0/SUCCESS)
 Main PID: 395 (sshd)
 Tasks: 1 (limit: 528)
 Memory: 3.7M
 CPU: 34ms
 CGroup: /system.slice/ssh.service
         └─395 sshd: /usr/sbin/sshd -D [listener] 0 of
          10-100 startups
```

La propiedad `Main PID` nos indica el identificador del proceso asociado al servicio. La propiedad `cgroup` nos muestra los recursos y procesos asociados al servicio.

La propiedad `loaded` nos muestra el estado del servicio en cuanto al momento en el que se inicia ("carga") en el sistema:

valor	descripción
`enable`	Servicio que se ejecuta en el arranque del sistema automáticamente.
`static`	Servicio que se ejecuta manualmente o por otra unidad.

valor	descripción
`disabled`	Servicio que no se ejecuta en el arranque del sistema de manera automática. Se puede ejecutar manualmente.
`masked`	Servicio que esta deshabilitada y que no se puede ejecutar ni siquiera manualmente. Es como bloquear el servicio.

La propiedad `active` nos muestra el estado del servicio en cuanto a su ejecución:

valor	descripción
`active`	El servicio esta en ejecución.
`inactive`	El servicio no esta en ejecución.1
`failed`	El servicio ha fallado.
`activating`	El servicio se está iniciando.
`deactivating`	El servicio se está parando.

Runlevel

```
#Obtener el runlevel por defecto
systemctl get-default

#Cambiar el runlevel por defecto a multi-user
systemctl set-default multi-user.target

#Cambiar el runlevel actual para apagar la maquina
systemctl isolate poweroff.target
```

Mostrar servicios

```
#En la salida de los comandos se muestran solo algunas
#líneas a modo de ejemplo.

#Mostrar servicio inactivos
systemctl list-units --type=service --state=inactive

UNIT                          LOAD   ACTIVE    SUB DESCRIPTION
apt-daily-upgrade.service     loaded inactive  dead  Daily apt updgrade and clean
apt-daily.service             loaded inactive dead Daily apt download activities
.....
```

```
#Mostrar servicios activos
systemctl list-units --type=service --state=active

UNIT                          LOAD   ACTIVE    SUB DESCRIPTION
apparmor.service              loaded active    exited Load AppArmor profiles
cron.service                  loaded active    running Background processing daemon
networking.service            loaded active    exited Raise network interfaces
ssh.service                   loades active     running OpenBSD Secure Shell Server
.....
```

```
#Mostrar servicios en ejecución
systemctl list-units --type=service --state=running

UNIT                            LOAD   ACTIVE    SUB DESCRIPTION
cron.service                    loaded active    running Background processing daemon
ssh.service                     loades active     running OpenBSD Secure Shell Server
.....
```

FICHEROS DE LOG

`Systemd` utiliza un sistema centralizado de registro de logs (bitácora) denominado `journal` que gestiona y almacena eventos del sistema, servicios, kernel y usuarios. El sistema `journal` es gestionado por el servicio `systemd-journald`.

Los registros generados por `journal` pueden ser consultados con la herramienta `journalctl`. Puedes obtener información de este comando en man journalctl. Entre las opciones más comunes:

opción	descripción
`--unit <servicio>`	Registros para un determinado servicio
`--lines <n>`	Muestra n registros más recientes. También admite all.
`--follow`	Muestra los registros más recientes y continua a la escucha. Se termina la escucha con CTRL+C
`--since "<fecha>"`	Muestra los registros desde la fecha. La fecha en formato americano yyyy-mm-dd hh:mm:ss. También admite today,yesterday,now,ago,...
`--until "<fecha>"`	Muestra los registros hasta la fecha. La fecha en formato americano yyyy-mm-dd hh:mm:ss. También admite today,yesterday,now,ago...
`--priority <n>`	Muestra los registros según su prioridad. 0 emerg 1 alert 2 crit 3 err 4 warning 5 notice 6 info 7 debug

journalctl

```
#Todos los registros de ssh
journalctl --unit ssh

#Los ultimos 20 registros de ssh
journalctl --lines 20 --unit ssh

#Los ultimos registros de ssh y mantenerse
journalctl --follow --unit ssh
```

```
#Los registros de ssh desde hace 1 minuto
journalctl --since "1 minute ago" --unit ssh

#Los registros de ssh desde hace 2 dias
journalctl --since "2 days ago" --unit ssh

#Todos los registros de ssh de tipo error
journalctl --lines all --priority 3 --unit ssh

#Los ultimos 10 registros de ssh de tipo error
journalctl --lines 10 --priority 3 --unit ssh

#Todos los registros de error de hoy
journalctl --lines all --priority 3 --since "today"
```