systemd (Español)

From ArchWiki

De la página web del proyecto (http://freedesktop.org/wiki/Software/systemd):

> «***systemd*** *es un gestor del sistema y de los servicios para Linux, compatible con los initscript SysV y LSB.* systemd *proporciona una notable capacidad de paralelización, utiliza la activación de socket y D-Bus para iniciar los servicios, permite el inicio de los demonios bajo demanda, realiza un seguimiento de los procesos con el uso de los grupos de control de Linux, apoya snapshotting y la restauración del estado del sistema, mantiene los puntos montaje y servicios de montaje automático e implementa un elaborado sistema de gestión de dependencias basado en un control lógico de los servicios.»*

Artículos relacionados

- systemd/User (Español)
- systemd/Timers
- systemd FAQ (Español)
- init Rosetta (Español)
- Daemons List
- udev (Español)
- Improve boot performance

Nota: Para conocer una explicación detallada del motivo por el cual Arch está cambiando a systemd, consulte este post (https://bbs.archlinux.org/viewtopic.php?pid=1149530#p1149530).

Contents

Uso básico de systemctl

La principal orden para controlar *systemd* es `systemctl`. Algunos de los posibles usos son el examen del estado del sistema, y la gestión del sistema y de los servicios. Consulte `man 1 systemctl` para conocer más detalles.

Sugerencia: Puede utilizar las siguientes órdenes `systemctl` con el parámetro `-H usario@host` para controlar una instancia de systemd en una máquina remota. Esto utilizará SSH para conectarse a la instancia systemd remota.

Nota: `systemadm` es el frontend gráfico oficial para `systemctl`. Proporcionado por el paquete `systemd-ui-git (https://aur.archlinux.org/packages/systemd-ui-git/)`[AUR][broken link: archived in aur-mirror (https://github.com/felixonmars/aur3-mirror/tree/master/systemd-ui-git)] disponible en AUR.

Analizar el estado del sistema

Listado de unidades activas:

```
$ systemctl
```

o bien:

```
$ systemctl list-units
```

Listado de unidades que han tenido problemas:

```
$ systemctl --failed
```

Los archivos de las unidades disponibles se pueden ver en `/usr/lib/systemd/system/` y `/etc/systemd/system/` (este último tiene prioridad). Puede ver un listado de las unidades instaladas con:

```
$ systemctl list-unit-files
```

Usar las unidades

Las unidades pueden ser, por ejemplo, servicios (`.service`), puntos de montaje (`.mount`), dispositivos (`.device`) o sockets (`.socket`).

Cuando se usa `systemctl` , por lo general, tiene que especificar el nombre completo de la unidad, incluyendo el sufijo, por ejemplo, `sshd.socket` . Sin embargo, hay unos pocos atajos cuando se especifica la unidad en las siguientes órdenes `systemctl` :

- Si no se especifica el sufijo, systemctl asumirá que es `.service` . Por ejemplo, `netcfg` y `netcfg.service` se consideran equivalentes.
- Los puntos de montaje se traducirán automáticamente en la correspondiente unidad `.mount` . Por ejemplo, si especifica `/home` será equivalente a `home.mount` .
- Similar a los puntos de montaje, los dispositivos se traducen automáticamente en la correspondiente unidad `.device` , por lo tanto, la especificación `/dev/sda2` es equivalente a `dev-sda2.device` .

Consulte `man systemd.unit` para más detalles.

> **Sugerencia:** La mayoría de las siguientes órdenes también funcionan si se especifican varias unidades, vea `man systemctl` para más información.

Activa una unidad de inmediato:

```
# systemctl start unidad
```

Desactiva una unidad de inmediato:

```
# systemctl stop unidad
```

Reinicia la unidad:

```
# systemctl restart unidad
```

Hace que una unidad recargue su configuración:

```
# systemctl reload unidad
```

Muestra el estado de una unidad, incluso si se está ejecutando o no:

```
$ systemctl status unidad
```

Comprueba si la unidad ya está habilitada o no:

```
$ systemctl is-enabled unidad
```

Activa el inicio automático en el arranque:

```
# systemctl enable unidad
```

Nota: Si los servicios no tienen una sección `[Install]` significa, por lo general, que se les llama de forma automática por otros servicios. Pero si necesita instalarlos manualmente, utilice la orden siguiente, reemplazando `foo` con el nombre del servicio.

```
# ln -s /usr/lib/systemd/system/foo.service /etc/systemd/system/graphical.target.wants/
```

Desactiva el inicio automático durante el arranque:

```
# systemctl disable unidad
```

Muestra la página del manual asociada con una unidad (esto tiene que ser apoyado por el archivo .unit):

```
$ systemctl help unidad
```

Recarga *systemd*, escaneando en busca de unidades nuevas o modificadas:

```
# systemctl daemon-reload
```

Gestionar la energía

polkit es necesario para gestionar la energía. Si se encuentra en una sesión local de `systemd-logind` y ninguna otra sesión está activa, las órdenes siguientes funcionarán sin requerir privilegios de root. Si no es así (por ejemplo, debido a que otro usuario ha iniciado otra sesión tty), *systemd* automáticamente le requerirá la contraseña de root.

Apagado y reinicio del sistema:

```
$ systemctl reboot
```

Apagado del sistema:

```
$ systemctl poweroff
```

Suspensión del sistema:

```
$ systemctl suspend
```

Poner el sistema en hibernación:

```
$ systemctl hibernate
```

Poner el sistema en estado de reposo híbrido —*«hybrid-sleep»* — (o suspensión combinada —*«suspend-to-both»*—):

```
$ systemctl hybrid-sleep
```

Escribir archivos .service personalizados

La sintaxis de los archivos de unidad de *systemd* se inspira en los archivos .desktop de XDG Desktop Entry Specification, que, a su vez, están inspirados en los archivos .ini de Microsoft Windows.

Manejar las dependencias

Con *systemd* las dependencias pueden ser resueltas planificando la unidad correctamente. El caso más típico es que la unidad `A` requiere la unidad `B` para poder funcionar, por lo que esta última debe iniciarse antes que `A`. En ese caso, agregue `Requires=B` y `After=B` a la sección `[Unit]` de `A`. Si la dependencia es opcional agregue, en su lugar, `Wants=B` y `After=B`. Tenga en cuenta que `Wants=` y `Requires=` no incluyen `After=`, lo que significa que si `After=` no esté especificado, las dos unidades se iniciarán en paralelo.

Las dependencias se colocan normalmente en los archivos .service y no en los .target. Por ejemplo, `network.target` es llamado por cualquiera que sea el servicio que configure las interfaces de red, por lo tanto, la solicitud que hace después la propia unidad personalizada es suficiente, ya que `network.target` se inicia de todos modos.

Type

Existen diferentes tipos de arranque a tener en cuenta cuando se escribe un archivo de servicio personalizado. Esto se configura mediante el parámetro

`Type=` en la sección `[Service]`. Consulte `man systemd.service` para una explicación más detallada.

- `Type=simple`: *systemd* considera que el servicio debe iniciarse inmediatamente. El proceso no debe romperse. No utilice este tipo si otros servicios tienen que ser llamados por ese servicio, a menos que no sea activado por el socket.
- `Type=forking`: *systemd* considera que el servicio debe ser iniciado antes que el proceso se rompa y el antecesor se haya terminado. Para los demonios clásicos use este tipo a menos que sepa que no es necesario, ya que la mayoría de los demonios usan doble bifurcación para indicar que están listos. Debe especificar también `PIDFile=` para que *systemd* puede realizar un seguimiento del proceso principal.
- `Type=oneshot`: Esto es útil para los scripts que hacen un solo trabajo y luego concluyen. Es posible que desee también establecer `RemainAfterExit=yes` de modo que *systemd* sigue considerando el servicio como activo después de que el proceso haya terminado.
- `Type=notify`: Igual que `Type=simple`, pero con la condición de que el demonio va a enviar una señal a *systemd* cuando esté listo. Esto requiere del código específico proporcionado por `libsystemd-daemon.so`.
- `Type=dbus`: El servicio se considera listo cuando el `BusName` especificado aparece en el bus del sistema DBus.

Modificar los archivos de unidad suministrados

Para editar un archivo de unidad proporcionado por un paquete, podemos crear un directorio llamado `/etc/systemd/system/`*`unit`*`.d/` por ejemplo `/etc/systemd/system/httpd.service.d/` y colocar los archivos `*.conf` en dicho directorio para reemplazarlos o añadir nuevas opciones. *systemd* analizará estos archivos `*.conf` y los aplicará antes que los de la unidad original. Por ejemplo, si deseamos simplemente agregar una dependencia adicional a una unidad, podemos crear el siguiente archivo:

```
/etc/systemd/system/unit.d/customdependency.conf
```

```
[Unit]
Requires=dependencia nueva
After=dependencia nueva
```

Siguiendo otro ejemplo, con el fin de reemplazar la directiva `ExecStart` para una unidad que no es del tipo `oneshot`, crearemos el siguiente archivo:

```
/etc/systemd/system/unit.d/customexec.conf
```

```
[Service]
ExecStart=
ExecStart=orden nueva
```

Otro último ejemplo, para reiniciar automáticamente un servicio:

```
/etc/systemd/system/unit.d/restart.conf
```

```
[Service]
Restart=always
RestartSec=30
```

A continuación, ejecutaremos lo que sigue para que los cambios surtan efecto:

```
# systemctl daemon-reload
# systemctl restart unidad
```

Por otro lado, podemos copiar el archivo de la antigua unidad desde `/usr/lib/systemd/system/` a `/etc/systemd/system/` y realizar los cambios allí. Un archivo de unidad ubicado en `/etc/systemd/system/` siempre tiene preferencia sobre la misma unidad localizada en `/usr/lib/systemd/system/`. Debemos tener en cuenta que cuando la unidad original localizada en `/usr/lib/` ha cambiado debido a una actualización del paquete que lo suministra, estos cambios no se aplicarán automáticamente al archivo de unidad personalizada ubicado en `/etc/`. De este modo, tendremos que volver a activar manualmente la unidad con la orden `systemctl reenable unidad`. Por consiguiente, se recomienda utilizar el método `*.conf` descrito anteriormente.

Sugerencia: Podemos utilizar la orden **systemd-delta** para ver qué archivos de la unidad han sido invalidados y cuáles han cambiado.

Como los archivos de unidad suministrados se actualizarán de vez en cuando, es conveniente utilizar systemd-delta para tareas de mantenimiento del sistema.

Resaltar la sintaxis de las unidades de systemd con Vim

El resaltado de sintaxis para las unidades de *systemd* con Vim se puede activar mediante la instalación de `vim-systemd (https://www.archlinux.org/packages/?name=vim-systemd)` desde los repositorios oficiales.

Targets

systemd utiliza *targets* («*objetivos*») que sirven a un propósito similar a los *runlevels* («*niveles de ejecución*»), pero que tienen un comportamiento un poco diferente. Cada *target* se nomina, en lugar de numerarse, y está destinado a servir a un propósito específico con la posibilidad de realizar más de una acción al mismo tiempo. Algunos *targets* son activados heredando todos los servicios de otro *target* e implementando servicios adicionales. Como hay *targets* de *systemd* que imitan los runlevels de SystemVinit, es, por tanto, posible pasar de un *target* a otro utilizando la orden `telinit RUNLEVEL`.

Conocer los targets presentes

La siguiente orden debe ser utilizada bajo *systemd*, en lugar de `runlevel`:

```
# systemctl list-units --type=target
```

Crear un target personalizado

Los niveles de ejecución («*runlevels*») son asignados a un fin específico de la instalación vanilla de Fedora; 0, 1, 3, 5, y 6; tienen una correlación de 1:1 con un específico *target* de systemd. Desafortunadamente, no hay una buena manera de hacer lo mismo para los niveles de ejecución definidos por el usuario como son el 2 y el 4. Si se hace uso de estos últimos, se sugiere dar un nuevo nombre al *target* de systemd como `/etc/systemd/system/su target` que tome como base uno de los runlevels existentes (vea `/usr/lib/systemd/system/graphical.target` como ejemplo), cree un directorio `/etc/systemd/system/su target.wants`, y haga un enlace a los servicios adicionales de `/usr/lib/systemd/system/` que desea habilitar.

Tabla de targets

Runlevel de SysV	Target de systemd	Notas
0	runlevel0.target, poweroff.target	Detiene el sistema.
1, s, single	runlevel1.target, rescue.target	Modalidad de usuario único.
2, 4	runlevel2.target, runlevel4.target, multi-user.target	Definidos por el usuario. Preconfigurados a 3.
3	runlevel3.target, multi-user.target	Multiusuario, no gráfica. Los usuarios, por lo general, pueden acceder a través de múltiples consolas o a través de la red.
5	runlevel5.target, graphical.target	Multiusuario, gráfica. Por lo general, tiene todos los servicios del nivel de ejecución 3, además de un inicio de sesión gráfica.
6	runlevel6.target, reboot.target	Reinicia el sistema.
emergency	emergency.target	Consola de emergencia.

Cambiar el target vigente

En *systemd* los targets quedan expuestos a través de «target units». Se pueden cambiar de esta manera:

```
# systemctl isolate graphical.target
```

Esto solo cambiará el target actual, y no tendrá ningún efecto sobre el siguiente arranque. Esto es equivalente a las órdenes `telinit 3` o `telinit 5` en Sysvinit.

Cambiar el target predeterminado para arrancar

El target estándar es *default.target*, que es un alias predefinido para *graphical.target* (que corresponde al antiguo nivel de ejecución 5). Para cambiar el target predeterminado en el arranque, añada uno de los siguientes parámetros del kernel al gestor de arranque:

Sugerencia: La extensión *.target* puede omitirse.

- `systemd.unit=multi-user.target` (que corresponde con el antiguo nivel de ejecución 3),
- `systemd.unit=rescue.target` (que corresponde con el antiguo nivel de ejecución 1).

Como alternativa, se puede dejar el gestor de arranque inalterado y cambiar *default.target*. Esto puede hacerse usando *systemctl*:

```
# systemctl enable multi-user.target
```

El efecto de esta orden se puede ver en la salida de *systemctl*; se crea un enlace simbólico al nuevo target prefedinido en `/etc/systemd/system/default.target`. Esto funciona solo si:

```
[Install]
Alias=default.target
```

reside en el archivo de configuración del target. En la actualidad, tanto *multi-user.target* como *graphical.target* lo tienen.

Archivos temporales

«**systemd-tmpfiles** crea, elimina y limpia archivos y directorios volátiles y temporales.» Lee los archivos de configuración en `/etc/tmpfiles.d/` y `/usr/lib/tmpfiles.d/` para descubrir qué acciones realizar. Los archivos de configuración del primer directorio tienen prioridad sobre los del último directorio.

Los archivos de configuración son proveidos normalmente junto con los archivos de servicio, y reciben su nombre en el estilo `/usr/lib/tmpfiles.d/`*`programa`*`.conf`. Por ejemplo, el demonio Samba espera que el

directorio `/run/samba` exista para obtener los permisos adecuados. Por tanto, el paquete `samba (https://www.archlinux.org/packages/?name=samba)` viene con esta configuración:

```
/usr/lib/tmpfiles.d/samba.conf
```

```
D /var/run/samba 0755 root root
```

Los archivos de configuración también pueden ser usados para escribir en el arranque valores en ciertos archivos. Por ejemplo, si usa `/etc/rc.local` para dehabilitar la reactivación del sistema («*wakeup*») a través de dispositivos USB con la orden `echo USBE > /proc/acpi/wakeup` , se puede utilizar, en su lugar, el siguiente tmpfile:

```
/etc/tmpfiles.d/disable-usb-wake.conf
```

```
w /proc/acpi/wakeup - - - - USBE
```

Consulte `systemd-tmpfiles` y `tmpfiles.d(5)` para obtener más detalles.

Nota: Este método puede no funcionar ajustando las opciones en `/sys` desde el momento en que el servicio `systemd-tmpfiles-setup` puede ejecutarse antes de que los módulos de los dispositivos adecuados se carguen. En este caso, se puede comprobar si el módulo tiene un parámetro para la opción que desea ajustar con `modinfo` *`modulo`* y establecer esta opción con un archivo de configuración en /etc/modprobe.d[broken link: invalid section]. De lo contrario, tendrá que escribir una regla udev para establecer el atributo apropiado tan pronto como el dispositivo lo reclame.

Temporizadores

systemd puede reemplazar la funcionalidad cron en gran medida. Para más información, consulte systemd/Timers.

Journal

Desde la versión 38, *systemd* tiene un sistema de registro («*log*») propio llamado journal. Por tanto, ya no es necesario hacer funcionar el demonio syslog. Para leer el registro, utilice:

```
# journalctl
```

Por defecto, (cuando `Storage=` está definido como `auto` en `/etc/systemd/journald.conf`), journal escribe en `/var/log/journal/` . Si el directorio `/var/log/journal/` no existe (por ejemplo, si lo ha eliminado usted o algún programa), systemd **no** lo crea de forma automática, sino que escribe los

registros en `/run/systemd/journal`. Esto significa que los registros se perderán al reiniciar.

> **Sugerencia:** Si `/var/log/journal/` reside en un sistema de archivos btrfs debería considerar la opción de desactivar Copy-on-Write para el directorio:
>
> ```
> # chattr +C /var/log/journal
> ```

Filtrar la salida

`journalctl` le permite filtrar los resultados por campos específicos. Tenga en cuenta que si hay muchos mensajes para mostrar o el filtrado que hay que hacer abarca mucho tiempo, la salida de esta orden puede retrasarse durante bastante tiempo.

Ejemplos:

Mostrar todos los mensajes del arranque:

```
# journalctl -b
```

Sin embargo, a veces a uno le interesan no los mensajes actuales, sino los mensajes desde el arranque anterior (por ejemplo, si ocurrió un fallo del sistema irrecuperable). Esto es posible pasando el parámetro `-b`: `journalctl -b -0` muestra los mensajes del arranque actual, `journalctl -b -1` muestra los mensajes del arranque anterior, `journalctl -b -2` muestra los mensajes desde los dos últimos arranques y así sucesivamente. Véase `man 1 journalctl` para una descripción completa, dado que los argumentos que se pueden pasar a la orden hacen que el filtrado pueda ser mucho más potente.

Seguir los mensajes nuevos:

```
# journalctl -f
```

Mostrar todos los mensajes de un ejecutable específico:

```
# journalctl /usr/lib/systemd/systemd
```

Mostrar todos los mensajes de un proceso específico:

```
# journalctl _PID=1
```

Mostrar todos los mensajes por una unidad específica:

```
# journalctl -u netcfg
```

Mostrar búfer circular del kernel:

```
# journalctl _TRANSPORT=kernel
```

Véase `man 1 journalctl`, `man 7 systemd.journal-fields` o esta entrada del blog (http://0pointer.de/blog/projects/journalctl.html) de Lennert para obtener más detalles.

Límite del tamaño de journal

Si journal se ha creado como permanente (no volátil), el límite de su tamaño se establece con un valor predeterminado correspondiente al 10% del tamaño del sistema de archivos. Por ejemplo, con `/var/log/journal` alojado en una partición raíz de 50 GiB, esto permitiría almacenar hasta 5 GiB de datos en journal. El tamaño máximo del journal permanente puede ser controlado por `SystemMaxUse` en `/etc/systemd/journald.conf`, por lo que, para limitarlo, por ejemplo, a 50 MiB, descomente y modifique la correspondiente línea a:

```
SystemMaxUse=50M
```

Consulte `man journald.conf` para más información.

Journald coexistiendo con syslog

La compatibilidad con las implementaciones del clásico syslog se proporciona a través de un socket: `/run/systemd/journal/syslog`, por donde pasan todos los mensajes. Para hacer que el demonio syslog funcione con journal, tiene que asociarlo a este socket en vez de a `/dev/log` (anuncio oficial (http://lwn.net/Articles/474968/)). El paquete `syslog-ng (https://www.archlinux.org/packages/?name=syslog-ng)` de los repositorios proporciona automáticamente la configuración necesaria.

```
# systemctl enable syslog-ng
```

Podemos encontrar un buen tutorial de `journalctl` aquí (http://0pointer.de/blog/projects/journalctl.html).

Reenviar journald a /dev/tty12

En `/etc/systemd/journald.conf` active lo siguiente:

```
ForwardToConsole=yes
TTYPath=/dev/tty12
```

```
MaxLevelConsole=info
```

Reinicie journald con `sudo systemctl restart systemd-journald`.

Solución de problemas

Investigar errores de systemd

Como ejemplo, vamos a investigar un error con el servicio `systemd-modules-load`:

1. Vamos a determinar los servicios de *systemd* que fallan al inicio:

```
$ systemctl --state=failed
```

```
systemd-modules-load.service   loaded failed failed  Load Kernel Modules
```

2. Encontramos un problema con el servicio `systemd-modules-load`. Indaguemos un poco más:

```
$ systemctl status systemd-modules-load
```

```
systemd-modules-load.service - Load Kernel Modules
   Loaded: loaded (/usr/lib/systemd/system/systemd-modules-load.service; static)
   Active: failed (Result: exit-code) since So 2013-08-25 11:48:13 CEST; 32s ago
     Docs: man:systemd-modules-load.service(8).
           man:modules-load.d(5)
  Process: 15630 ExecStart=/usr/lib/systemd/systemd-modules-load (code=exited, status=1/FAILURE)
```

3. Ahora tenemos el identificador del proceso (PID) para investigar este error en profundidad. Escribimos la siguiente orden con el `Process ID` (en este caso: 15630):

```
$ journalctl -b _PID=15630
```

```
-- Logs begin at Sa 2013-05-25 10:31:12 CEST, end at So 2013-08-25 11:51:17 CEST. --
Aug 25 11:48:13 mypc systemd-modules-load[15630]: Failed to find module 'blacklist usblp'
Aug 25 11:48:13 mypc systemd-modules-load[15630]: Failed to find module 'install usblp /bin/false'
```

4. Vemos que algunos de los ajustes del módulo del kernel tienen valores erróneos. Por lo tanto, echemos un vistazo a estos valores en `/etc/modules-load.d/`:

```
$ ls -Al /etc/modules-load.d/
```

```
...
-rw-r--r--   1 root root    79  1. Dez 2012  blacklist.conf
-rw-r--r--   1 root root     1  2. Mär 14:30 encrypt.conf
-rw-r--r--   1 root root     3  5. Dez 2012  printing.conf
-rw-r--r--   1 root root     6 14. Jul 11:01 realtek.conf
-rw-r--r--   1 root root    65  2. Jun 23:01 virtualbox.conf
...
```

5. El mensaje del error `Failed to find module 'blacklist usblp'` puede estar relacionado con un mal ajuste de `blacklist.conf`. Podemos desactivarlo insertando un signo **#** delante de cada opción que hemos descubierto que falla por medio del paso 3:

```
/etc/modules-load.d/blacklist.conf
```

```
# blacklist usblp
# install usblp /bin/false
```

6. Ahora, intente iniciar `systemd-modules-load`:

```
$ systemctl start systemd-modules-load.service
```

Si ha tenido éxito, no debe mostrarse ningún prompt. Si ve algún error, volveremos al paso 3 y utilizaremos el nuevo PID para solucionar los errores que aparecen en la izquierda.

Si todo está bien, se puede verificar que el servicio se ha iniciado satisfactoriamente con:

```
$ systemctl status systemd-modules-load
```

```
systemd-modules-load.service - Load Kernel Modules
   Loaded: loaded (/usr/lib/systemd/system/systemd-modules-load.service; static)
   Active: active (exited) since So 2013-08-25 12:22:31 CEST; 34s ago
     Docs: man:systemd-modules-load.service(8)
           man:modules-load.d(5)
 Process: 19005 ExecStart=/usr/lib/systemd/systemd-modules-load (code=exited, status=0/SUCCESS)
Aug 25 12:22:31 mypc systemd[1]: Started Load Kernel Modules.
```

A menudo se puede resolver este tipo de problemas como se ha descrito arriba. Para indagar más, mire el epígrafe siguiente: «**Diagnosticar problemas de arranque**».

Diagnosticar problemas de arranque

Arranque con esos parámetros en la línea de órdenes del kernel:

```
systemd.log_level=debug systemd.log_target=kmsg log_buf_len=1M
```

Más información sobre depuración de errores (http://freedesktop.org/wiki/Software/systemd/Debugging)

Apagar/reiniciar se hace terriblemente largo

Si el proceso de apagado tarda un tiempo muy largo (o parece congelarse) lo más probable es que un servicio no existente tenga la culpa. *systemd* espera un tiempo para iniciar cada servicio antes de tratar de acabar con él. Para averiguar si este es su caso, consulte este artículo (http://freedesktop.org/wiki/Software/systemd/Debugging#Shutdown_Completes_Eventually).

Los procesos de corta duración parecen no registrar ninguna salida

Si `systemctl -u foounit.service` no muestra ninguna salida para un servicio de breve duración, compruebe el PID. Por ejemplo, si `systemd-modules-load.service` falla, y `systemctl status systemd-modules-load` muestra que es seguido con PID 123, entonces es posible ver la salida de journal para dicho PID, por ejemplo `journalctl -b _PID=123`. Los campos con metadatos para journal, como _SYSTEMD_UNIT y _COMM, se recogen en modo asíncrono y se basan en la carpeta `/proc` para el proceso existente. La reparación de este proceso requiere la reparación del kernel para proporcionar estos datos por medio de una conexión socket, de forma similar a SCM_CREDENTIALS.

Desactivar el volcado de sucesos de journal respecto de las aplicaciones

Ejecute lo siguiente para sobrescribir la configuración de `/lib/sysctl.d/`:

```
# ln -s /dev/null /etc/sysctl.d/50-coredump.conf
# sysctl kernel.core_pattern=core
```

Esto desactivará el registro de coredumps en journal.

Tenga en cuenta que el RLIMIT_CORE por defecto es 0, lo que significa que tampoco hay archivos básicos que escribir. Si quiere que dichos archivos existan, necesita añadir el valor «unlimit» para el tamaño del archivo básico con la siguiente orden:

```
$ ulimit -c unlimited
```

Véase sysctl.d (http://www.freedesktop.org/software/systemd/man/sysctl.d.html) y the documentation for /proc/sys/kernel (https://www.kernel.org/doc/Documentation/sysctl/kernel.txt) para obtener más información.

Mensaje de error al reiniciar o apagar

cgroup : option or name mismatch, new: 0x0 "", old: 0x4 "systemd"

Véase este hilo (https://bbs.archlinux.org/viewtopic.php?pid=1372562#p1372562) para mayor explicación.

watchdog watchdog0: watchdog did not stop!

Véase este hilo (https://bbs.archlinux.org/viewtopic.php?pid=1372562#p1372562) para mayor explicación.

Véase también

- Sitio Web Oficial (http://www.freedesktop.org/wiki/Software/systemd)
- Artículo de Wikipedia
- Páginas del manual (http://0pointer.de/public/systemd-man/)
- Optimizar systemd (http://freedesktop.org/wiki/Software/systemd/Optimizations)
- FAQ (http://www.freedesktop.org/wiki/Software/systemd/FrequentlyAskedQuestions)
- Consejos y trucos (http://www.freedesktop.org/wiki/Software/systemd/TipsAndTricks)
- systemd para Administradores (PDF) (http://0pointer.de/public/systemd-ebook-psankar.pdf)
- Acerca de systemd en Fedora Project (http://fedoraproject.org/wiki/Systemd)
- Cómo depurar problemas en systemd (http://fedoraproject.org/wiki/How_to_debug_Systemd_problems)
- Two (http://www.h-online.com/open/features/Control-Centre-The-systemd-Linux-init-system-1565543.html) part (http://www.h-online.com/open/features/Booting-up-Tools-and-tips-for-systemd-1570630.html) artículo introductorio de la revista *The H Open*.
- Historia del blog de Lennart (http://0pointer.de/blog/projects/systemd.html)
- Status update (http://0pointer.de/blog/projects/systemd-update.html)
- Status update2 (http://0pointer.de/blog/projects/systemd-update-2.html)
- Status update3 (http://0pointer.de/blog/projects/systemd-update-3.html)
- Resumen más reciente (http://0pointer.de/blog/projects/why.html)
- Fedora's SysVinit to systemd cheatsheet (http://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet)
- Configurar systemd para permitir apagar a los usuarios normales