Apuntes SinCara Extras systemd

Consideraciones iniciales

- Se presenta el 30 de abril de 2010, por Lennart Poetterig
- SystemD es un sistema de inicio y administración de servicios.
- Reemplaza los sistemas de inicio tradicionales como SysV init o Upstart (de Canonical / Ubuntu)
- Se enfoca en la eficiencia, la rapidez y la robustez.
- Es ampliamente usado en la comunidad de Linux, aunque ha generado bastante controversia debido a su enfoque radical, a su reemplazo de sistemas de inicio tradicionales y al cambio de paradigma histórico de UNIX.
- Es el estándar de facto para la gestión de servicios en Linux.

Ventajas de SystemD

- Arranque Paralelo: SystemD permite iniciar servicios de manera paralela, lo que reduce significativamente el tiempo de arranque del sistema.
- Dependencias Explícitas: Los servicios en SystemD pueden especificar sus dependencias de manera explícita, lo que facilita la gestión de la secuencia de arranque y evita problemas de dependencia circular.
- Control Dinámico de Servicios: SystemD proporciona herramientas para controlar dinámicamente los servicios en ejecución, como iniciar, detener, reiniciar o consultar el estado de un servicio.
- Integración con cgroups: SystemD integra el control de grupos de procesos (cgroups) para una gestión más eficiente de recursos y aislamiento de procesos.
- Compatibilidad con SysV init: SystemD es compatible con los scripts de inicio de SysV init, lo que facilita la transición para las distribuciones que migran a SystemD.
- □□ https://without-systemd.org/wiki/index_php/Arguments_against_systemd/ - Argumentos contra SystemD

Paquetes RPM

- `rpm -qa | grep systemd` # Instalados en Fedora 42
  - SystemD 256 x86_64
    - systemd-256
    - systemd-container-256
    - systemd-libs-256
    - systemd-networkd-256
    - systemd-pam-256
    - systemd-resolved-256
    - systemd-udev-256
  - Otros
    - rpm-plugin-systemd-inhibit-4.20.0
    - python3-systemd-235-11
    - libreport-plugin-systemd-journal-2.17.15
  - noarch
    - systemd-oomd-defaults-256
- `yum list systemd` # Paquetes no instalados
  - x86_64
    - systemd-boot-unsigned
    - systemd-bootchart

        - systemd-devel
        - systemd-journal-remote
        - systemd-standalone-repart
        - systemd-standalone-shutdown
        - systemd-standalone-sysusers
        - systemd-standalone-tmpfiles
        - systemd-tests
        - uwsgi-logger-systemd.x86_64
    - Otros
        - python-systemd-doc-235-11
    - noarch
        - systemd-networkd-defaults
        - systemd-rpm-macros
        - systemd-swap
        - systemd-ukify
        - wine-systemd-10.0

Comandos de SystemD

- `relacionados.sh hostnamectl` # Para ver todos los comandos
- `apropos ctl | \grep «ctl ([18])»` # Comandos que terminan en ctl instalados, sean de SystemD o no
- Comandos de SystemD
    - `rpm -ql systemd-container systemd-networkd systemd-udev systemd-resolved systemd | grep bin/`
    - `rpm -ql systemd-container systemd-networkd systemd-udev systemd-resolved systemd | grep bin/ | xargs whatis` # Para ver que hace cada uno
- https://www.linuxfromscratch.org/lfs/view/systemd/chapter08/systemd.html - Listado de comandos
- Alias de Comandos:
    - `alias sd_servicios='systemctl --no-pager --state=running --type=service'`
    - `alias sd_sockets='systemctl --no-pager --state=running --type=socket'`
    - `alias sd_timers='systemctl --no-pager --state=running --type=timer'`

Componentes principales

- **systemd**: Es el proceso inicial de SystemD y actúa como el proceso principal del sistema. Es responsable de inicializar el sistema y coordinar el arranque de otros servicios y procesos. Es el proceso con el PID número 1.
- **systemd-journald**: Es el servicio de registro del sistema de SystemD. Reemplaza al tradicional syslog y proporciona un registro centralizado y estructurado de eventos del sistema y de los servicios.
- **systemd-udev**: Es el administrador de dispositivos de SystemD. Se encarga de la detección y gestión dinámica de dispositivos en el sistema, incluyendo la asignación de nombres de dispositivos y la configuración de udev.
- **systemd-networkd**: Es el administrador de red de SystemD. Proporciona una manera de configurar y gestionar la red del sistema, incluyendo la configuración de interfaces de red, enrutamiento y resolución DNS. No suele estar activo, ya que normalmente se utilizan otros servicios:
    - **NetworkManager.service** en Red Hat, Fedora y derivados
    - **networking.service** en Debian, Ubuntu y derivados
- **systemd-resolved**: Es el servicio de resolución de DNS de SystemD. Proporciona resolución de nombres de dominio y reemplaza a las soluciones tradicionales como resolv.conf.
- **systemd-logind**: Es el servicio de gestión de sesiones de usuario de SystemD. Se encarga de gestionar

las sesiones de usuario, el control de acceso y la suspensión/hibernación del sistema.
- **systemd-userdbd**: Este servicio gestiona bases de datos de usuarios y grupos para la sesión del usuario. Permite almacenar y recuperar información sobre usuarios y grupos dentro del contexto de una sesión de usuario.
- **systemd-timedated**: Es el servicio de gestión de fecha y hora de SystemD. Se encarga de mantener y sincronizar el reloj del sistema con fuentes de tiempo externas.
- **systemd-oom**d: Este servicio, conocido como «Out-Of-Memory Daemon» (Demonio de Falta de Memoria), supervisa la memoria del sistema y gestiona situaciones de falta de memoria. Cuando el sistema se queda sin memoria, systemd-oomd intenta identificar y terminar procesos no esenciales para liberar recursos y evitar el agotamiento total de la memoria.
- **systemd-machined**: Este servicio es responsable de gestionar máquinas virtuales y contenedores en el sistema.
- **systemd-ask-password-wall**: Este servicio se utiliza para mostrar solicitudes de contraseña en la pantalla del sistema. Cuando un servicio necesita autenticación (por ejemplo, al iniciar una unidad cifrada durante el arranque), systemd-ask-password-wall muestra una ventana emergente en el entorno de escritorio para que el usuario introduzca la contraseña.

Información general

- `systemctl --version` # Para ver la versión de SystemD instalada
- El - y el + indican las características habilitadas de SystemD. Cada Distro elige cuales activar o no, pero no se puede cambiar sin reinstalar el sistema.
    - PAM: La integración con PAM (Pluggable Authentication Modules) permite a SystemD interactuar con el sistema de autenticación del sistema operativo. Esto significa que SystemD puede utilizar los mecanismos de autenticación configurados en PAM para servicios que requieren autenticación.
    - AUDIT: La compatibilidad con AUDIT permite a SystemD interactuar con el subsistema de auditoría del kernel Linux (Linux Audit). Esto proporciona capacidades de auditoría avanzadas para registrar eventos del sistema, como accesos a archivos, cambios de configuración, etc.
    - SELINUX: SystemD tiene integración con SELinux (Security-Enhanced Linux), un sistema de seguridad para el control de acceso obligatorio (MAC). Cuando esta característica está habilitada, SystemD puede interactuar de manera adecuada con las políticas de SELinux para aplicar restricciones de seguridad adicionales en el sistema.
    - APPARMOR: SystemD tiene integración con AppArmor, un marco de seguridad para el control de acceso obligatorio (MAC) similar a SELinux. Cuando esta característica está habilitada, SystemD puede interactuar con las políticas de AppArmor para aplicar restricciones de seguridad adicionales en el sistema.
    - IMA (Integrity Measurement Architecture): La compatibilidad con IMA permite a SystemD interactuar con la arquitectura de medición de integridad del kernel Linux. Esto facilita la realización de mediciones de integridad del sistema para detectar cambios no autorizados en los archivos del sistema.
    - SMACK (Simplified Mandatory Access Control Kernel): La compatibilidad con SMACK permite a SystemD interactuar con el marco de control de acceso obligatorio (MAC) Simplified Mandatory Access Control Kernel. Esto proporciona capacidades de control de acceso a nivel de archivo basadas en etiquetas en el sistema.
    - SECCOMP (Secure Computing Mode): La compatibilidad con SECCOMP permite a SystemD utilizar el modo de computación segura del kernel Linux para restringir las llamadas al sistema disponibles para un proceso. Esto ayuda a mejorar la seguridad limitando las capacidades de los procesos.
    - GCRYPT: SystemD puede interactuar con GnuPG Cryptography (GCRYPT), una biblioteca para cifrado y descifrado de datos. Esto puede ser útil para servicios que requieren funciones criptográficas en el sistema.
    - GNUTLS: SystemD tiene integración con GNU Transport Layer Security (GNUTLS), una biblioteca de criptografía y seguridad de red. Esto puede ser útil para servicios que requieren capacidades de cifrado y autenticación segura.
    - OPENSSL: SystemD puede integrarse con OpenSSL, una biblioteca de cifrado y seguridad ampliamente utilizada en sistemas Linux. Esta integración permite que SystemD utilice funciones

criptográficas proporcionadas por OpenSSL para tareas como cifrado, descifrado, generación de claves, etc.
- ACL (Access Control List): La compatibilidad con ACL permite a SystemD interactuar con listas de control de acceso (ACL) en sistemas de archivos. Esto amplía las capacidades de control de acceso a nivel de archivo en el sistema.
- BLKID: SystemD puede interactuar con blkid, una utilidad para identificar dispositivos de bloques (como discos duros, particiones, etc.). Esto puede ser útil para la gestión de dispositivos de almacenamiento en el sistema.
- CURL: SystemD puede integrarse con cURL, una herramienta de línea de comandos y una biblioteca para transferencias de datos con URLs. Esta integración permite que SystemD realice solicitudes HTTP, HTTPS, FTP y otras solicitudes de red utilizando cURL para tareas como descargar archivos, acceder a servicios web, etc.
- ELFUTILS: SystemD tiene integración con ELF Utils, una colección de herramientas para trabajar con archivos ejecutables y objetos ELF (Formato de Archivo Ejecutable y Enlace). Esto puede ser útil para la depuración y análisis de ejecutables en el sistema.
- FIDO2: FIDO2 es un estándar de autenticación en dos factores (2FA) basado en claves públicas que proporciona una forma segura y conveniente de autenticar usuarios en servicios en línea. La compatibilidad con FIDO2 en SystemD significa que puede interactuar con dispositivos de seguridad compatibles con FIDO2 para autenticación de usuario.
- IDN2, IDN (Internationalized Domain Names): Estas características indican el soporte para el manejo de nombres de dominio internacionalizados (IDN) en el sistema. Esto puede ser útil para la interoperabilidad con nombres de dominio que contienen caracteres no ASCII.
- IPTC: IPTC (International Press Telecommunications Council) es un estándar para el intercambio de información entre sistemas de gestión de contenido y aplicaciones relacionadas con la fotografía y el periodismo. La compatibilidad con IPTC en SystemD puede implicar la capacidad de leer, escribir o manipular metadatos IPTC en archivos de medios.
- KMOD: La compatibilidad con KMOD permite a SystemD interactuar con el sistema de carga de módulos del kernel Linux. Esto puede ser útil para la gestión de módulos del kernel y la carga de controladores de dispositivos en el sistema.
- LIBCRYPTSETUP: SystemD tiene integración con libcryptsetup, una biblioteca para configurar y gestionar volúmenes cifrados en el sistema. Esto permite que SystemD maneje servicios y unidades que dependen de volúmenes cifrados de manera adecuada.
- LIBCRYPTSETUP_PLUGINS:
- LIBFDISK: Libfdisk es una biblioteca de utilidades para trabajar con tablas de particiones de disco y manipular particiones en sistemas Linux. La compatibilidad con Libfdisk en SystemD significa que puede utilizar las funciones proporcionadas por esta biblioteca para realizar operaciones relacionadas con el particionado de discos.
- PCRE2 (Perl Compatible Regular Expressions): La compatibilidad con PCRE2 indica que SystemD tiene soporte para expresiones regulares compatibles con Perl. Esto puede ser útil para la búsqueda y manipulación de cadenas de texto en archivos de configuración y otros contextos.
- PWQUALITY: PWQUALITY es un módulo de SystemD que proporciona políticas de calidad de contraseña para el sistema de autenticación. Esto implica que SystemD puede aplicar reglas y restricciones a la creación y el cambio de contraseñas de usuario para garantizar contraseñas seguras y robustas.
- P11KIT: P11Kit es una biblioteca para interactuar con dispositivos de seguridad que admiten el estándar PKCS #11. La compatibilidad con P11Kit en SystemD significa que puede utilizar esta biblioteca para acceder a dispositivos de hardware de seguridad, como tokens USB criptográficos, para tareas de autenticación y cifrado.
- QRENCODE: QREncode es una biblioteca para generar códigos QR, que son códigos de barras bidimensionales que almacenan información en una matriz de puntos. La compatibilidad con QREncode en SystemD podría implicar la capacidad de generar códigos QR para usar en aplicaciones relacionadas con la identificación o la autenticación.
- TPM2 (Trusted Platform Module 2): TPM2 es un estándar para chips de seguridad integrados en placas base que proporcionan funcionalidades como el almacenamiento seguro de claves, la

generación de claves criptográficas y la realización de operaciones criptográficas seguras. La compatibilidad con TPM2 en SystemD podría implicar la capacidad de interactuar con chips TPM2 para tareas de seguridad y autenticación.
    - BZIP2, LZ4, XZ, ZLIB, ZSTD: Estas características indican que SystemD tiene soporte para diferentes algoritmos de compresión. Esto puede ser útil para la compresión y descompresión de datos en el sistema, por ejemplo, en archivos de registro.
    - BPF_FRAMEWORK (Berkeley Packet Filter Framework): BPF es un marco para programación de filtros de paquetes de red en sistemas Unix. La compatibilidad con BPF en SystemD puede implicar la capacidad de utilizar esta tecnología para filtrar y analizar el tráfico de red en el sistema.
    - XKBCOMMON: XKBCOMMON es una biblioteca para manejar la configuración del teclado en sistemas X Window. La compatibilidad con XKBCOMMON en SystemD puede implicar la capacidad de interactuar con esta biblioteca para configurar y gestionar el comportamiento del teclado en sistemas Linux.
    - UTMP: SystemD puede interactuar con el archivo utmp (registro de usuarios en el sistema) para registrar la actividad de inicio de sesión y otros eventos relacionados con los usuarios en el sistema.
    - SYSVINIT: Esta característica indica que SystemD tiene compatibilidad con los scripts de inicio tradicionales de SysV init. Esto permite que SystemD inicie y controle servicios que aún utilizan el sistema de inicio basado en SysV init.
    - default-hierarchy=hybrid: Esta opción indica el tipo de jerarquía de montaje predeterminada utilizada por SystemD para CGroups. En este caso, «hybrid» significa que se utiliza una combinación de la antigua jerarquía de montaje SysV y la nueva jerarquía de montaje de SystemD.
    - LIBARCHIVE:
- https://github.com/systemd/systemd - Repositorio de SystemD
- `man systemd.index` # Listado de todas las páginas del manual de SystemD

Unidades

- `systemctl -t help` # Para ver los tipos de unidades disponibles
- `man 7 systemd.special` # Para ver las unidades especiales que no se pueden renombrar
- Units o unidades: es como SystemD llama a los diferentes recursos que maneja. Mientras en SysV solo había servicios, en SystemD las unidades se pueden catalogar como:
    - .service: son los servicios, equivalentes a los de SysV. Se pueden crear servicios personalizados.
    - .socket: se utilizan para configurar y controlar sockets de red o UNIX en el sistema.
    - .device: descripción de dispositivos necesarios, como udev, sysfs, etc. Es decir, los necesarios para montar particiones, acceder a dispositivos, etc.
    - .mount: unidad para definir los puntos de montaje del sistema.
    - .automount: unidades montadas automáticamente bajo demanda, al acceder al directorio de montaje. Deben tener una unidad .mount coincidente para definir los detalles del montaje
    - .swap: unidad que describe el espacio SWAP.
    - .target: las correspondientes a los niveles de ejecución, para poder sincronizar una serie de sistemas para determinar el estado.
    - .path: Las unidades de ruta se utilizan para monitorear rutas de archivos o directorios en el sistema de archivos. Se utilizan para activar servicios o acciones cuando se producen cambios en archivos o directorios específicos, como la generación de eventos cuando se crea un nuevo archivo en un directorio determinado.
    - .timer: es similar a las tareas de cron, un temporizador o planificador de tareas.
    - .snapshot: una unidad creada automáticamente por systemctl con la que poder reconstruir el estado del sistema tras realizar cambios. Pero es temporal, no sobrevive entre sesiones.
    - .slice: asociada con nodos Linux Control Group, para asignar o restringir recursos a un proceso dentro del slice.
    - .scope: también creada automáticamente por systemd a partir de información recibida de las interfaces de bus. Se emplea para la gestión de conjuntos de procesos que se crean externamente.
- https://es.linux-console.net/?p=5705 - Comprender las unidades y los archivos de unidades de Systemd

- https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units-es - Manejo de las unidades

Capacidades de las Unidades

- Activación basada en sockets:
  - Los sockets asociados con un servicio se separan mejor del mismo daemon para poder manejarlos por separado.
  - Esto proporciona una serie de ventajas, como retrasar el inicio de un servicio hasta que se acceda por primera vez al socket asociado.
  - Esto también permite que el sistema cree todos los sockets al principio del proceso de inicio, lo que permite iniciar los servicios asociados en paralelo.
- Activación basada en bus (dbus):
  - Las unidades también se pueden activar en la interfaz de bus proporcionada por D-Bus.
  - Se puede iniciar una unidad cuando se publica un bus asociado.
- Activación basada en ruta (path):
  - Una unidad se puede iniciar en función de la actividad o la disponibilidad de ciertas rutas del sistema de archivos.
  - Esto utiliza inotify.
- Activación basada en dispositivos (udev):
  - Las unidades también se pueden iniciar en la primera disponibilidad de hardware asociado aprovechando los eventos de udev.
- Mapeo de dependencias implícitas:
  - La mayor parte del árbol de dependencias entre unidades lo crea SystemD.
  - Se pueden agregar dependencias, pero la mayor parte del trabajo pesado la realiza por debajo SystemD.
- Instancias y plantillas:
  - Los archivos de unidades de plantilla se pueden usar para crear varias instancias de la misma unidad general.
  - Esto permite ligeras variaciones o unidades hermanas que brindan la misma función general.
- Reforzamiento de la seguridad fácil:
  - Las unidades pueden implementar algunas características de seguridad bastante buenas al agregar directivas simples.
  - Por ejemplo, se puede especificar acceso nulo o de solo lectura a una parte del sistema de archivos, limitar las capacidades del kernel y asignar /tmp y acceso a la red privado.
- drop-ins y fragmentos:
  - Las unidades se pueden ampliar fácilmente al proporcionar fragmentos que anularán partes del archivo de unidad del sistema.
  - Esto hace que sea fácil cambiar entre implementaciones de unidades estándar y personalizadas.

Estados de las Unidades

- Hay cinco estados en los que puede estar una unidad: **inactive**, **activating**, **activate**, **deactivating** o **failed**.
  - Por defecto, las unidades son **inactive**.
  - Cuando systemd inicia una unidad, el estado cambia a **activating**
  - y una vez que finaliza el inicio se marca como **active**.
  - Luego permanece en este estado hasta que se detiene, ya sea por sí mismo (por ejemplo, si el ejecutable de un servicio finaliza) o si systemd le indica que se detenga. Luego, el estado cambia a **deactivating**, seguido de **inactive** una vez que finaliza el apagado.
  - Y si algo sale mal, el estado es **failed**.

- https://seb.jambor.dev/posts/systemd-by-example-part-2-dependencies/ - Estados de las unidades

Archivos de configuración

- Orden de preferencia de directorios, de menor a mayor prioridad:
  - **/usr/lib/systemd/system/<unit>** - Directorio original donde se instalan por defecto las units. No debemos tocar nada en este directorio, porque se machaca en cada actualización.
  - **/run/systemd/system/<unit>** - Es donde están las runtime units, es decir, unidades durante el tiempo de ejecución. Si queremos cambiar algo del fichero de configuración de forma temporal, hasta que se reinicie el sistema, lo copiamos a esta localización y lo editamos. Pero al estar en /run se pierde al rebotar. Si lo quiero hacer persistente, lo meto en /etc
  - **/etc/systemd/system/<unit>** - Directorio con mayor preferencia, pero puede existir o no. Copio aquí el fichero de configuración de la unit, y de esta forma es persistente lo que yo cambie en el.
  - **Comandos relacionados**
    - `tree /etc/systemd/system` # Para ver el árbol de ficheros y directorios
    - `systemd-analyze unit-paths` # Vemos todos los directorios implicados
    - `systemctl edit --full <unit>` # Editamos el fichero de configuración, creándolo automáticamente en /etc/systemd/system/<unit>
    - `systemctl cat <unit>` # Vemos todos los ficheros que aplican a la configuración
    - `systemd-delta [<unit>]` # se puede utilizar para identificar y comparar ficheros de configuración que se anulan unos a otros. Es decir, es una utilidad interesante para detectar conflictos con la configuración de systemd, y el orden de precedencia entre los distintos directorios anteriormente nombrados.
  - **Drop-in files**: Si queremos añadir un fichero que solo modifique algún parámetro en concreto, lo podemos hacer con el siguiente comando:
    - `systemctl edit <unit> --drop-in=<nombre>` # Esto crea un fichero llamado /etc/systemd/system/<unit>/<nombre>.conf. Si no se especifica nombre, se crea con el nombre por defecto **override.conf**
    - `systemctl set-property <unit> CPUShares=1024` # Con esto me crea un Drop-in file automáticamente

Creación de tus propias unidades

- Puedes agregar tus propios servicios para ejecutar acciones o scripts creando ficheros con extensión **.service**.
- Todas las opciones dentro de las unidades, se explican en `man systemd.exec`
- `cat /usr/lib/systemd/system/cups.service` # Vemos el fichero principal de configuración

```
[Unit]
Description=CUPS Scheduler
Documentation=man:cupsd(8)
After=network.target nss-user-lookup.target nslcd.service
Requires=cups.socket

[Service]
ExecStart=/usr/sbin/cupsd -l
Type=notify
Restart=on-failure

[Install]
Also=cups.socket cups.path
WantedBy=printer.target multi-user.target
```

Secciones dentro de los ficheros de configuración

- **[Unit]**: Define directivas específicas de la definición de la unit
- **[Install]**: Define directivas de gestión de units estableciendo relaciones en aspectos como, por ejemplo, la relación de la unit con targets asociados
- **[Service]**: Define directivas específicas de un service
    - Type
        - **Type=simple** (default): El servicio se inicia inmediatamente. El proceso no debe bifurcarse. No utilice este tipo si otros servicios necesitan ser ordenados en este servicio, a menos que esté activado por socket.
        - **Type=idle**: SystemD retrasará la ejecución del binario del servicio hasta que todos los trabajos hayan terminado. Aparte de eso el comportamiento es muy similar a Type=simple.
        - **Type=notify**: Idéntico a Type=simple, pero con la estipulación de que el demonio enviará una señal a SystemD cuando esté listo. La implementación de referencia para esta notificación la proporciona libsystemd-daemon.so.
        - **Type=oneshot**: Esto es útil para scripts que hacen un solo trabajo y luego salen. Es posible que desee establecer RemainAfterExit=yes también para que SystemD siga considerando el servicio como activo después de que el proceso haya salido. Establecer RemainAfterExit=yes es apropiado para las unidades que cambian el estado del sistema (por ejemplo, montar alguna partición).
        - **Type=forking**: SystemD considera el servicio iniciado una vez que el proceso se bifurca y el padre ha finalizado. Para demonios clásicos, utilice este tipo a menos que sepa que no es necesario. Debe especificar también PIDFile= para que SystemD pueda realizar un seguimiento del proceso principal.
        - **Type=dbus**: El servicio se considera listo cuando el BusName especificado aparece en el bus de sistema de DBus.
- **[Socket]**: Define directivas de configuración de sockets asociados a services
- **[Mount]** y **[Automount]**: Define directivas para puntos de montaje
- **[Swap]**: Directivas que definen y habilitan espacios de intercambio para páginas de memoria virtual anónimas
- **[Timer]**: Definición y gestión de eventos temporales
- **[Path]**: Monitorización del sistema de archivos
- **[Slice]**: Gestión de asignación de recursos a los procesos (CGroups)

Particularidades de configuración

- Si en la sección **ExecStart** el binario empieza con un «-», significa que **SystemD** ignorará el código de salida del comando. Este prefijo, y otros, aparece en la Tabla 2 dentro de: `man systemd.service`
- Si en la sección **EnvironmentFile** el fichero empieza con un «-», significa que si no existe dicho fichero, no se intenta leer y no da error. Esto aparece en `man systemd.exec`

Ficheros y entorno

- `systemd-delta cups.service` # Vemos si hay algún fichero de configuración que modifique el principal
- `systemctl cat cups.service` # Vemos todos los ficheros que aplican a la configuración
- `systemctl edit --full cups.service` # Editamos el fichero de configuración, creándolo automáticamente en **/etc/systemd/system/cups.service**
    - `export SYSTEMD_EDITOR=/usr/bin/vim` # Para usar el vim en vez del nano que usa por defecto
- `systemctl show cups.service` # Mostramos todas las variable que afectan la unidad
- `systemctl daemon-reload` # Comando para indicarle a **SystemD** que hemos tocado un fichero de

configuración **por nuestra cuenta** y que relea y aplique los cambios de todos los ficheros de configuración.

Dependencias

- `systemctl list-dependencies cups.service` # Vemos el árbol de dependencias de esta unidad
- `systemd-analyze critical-chain cups.service` # Vemos en el árbol de dependencias de esta unidad, cuanto tarda el elemento más lento de cada nivel

Entorno

- `systemctl show-environment` # Ver las variables de entorno que usa SystemD
- `localectl` # Para ver el idioma local, y el teclado
- `localectl set-locale LANG=es_US.UTF-8` # Para cambiar el idioma local
- https://systemd.io/ENVIRONMENT/ - Variables de entorno

Targets

- **Target = runlevel**: SystemD llama targets a los runlevels de SysV, y existen equivalencias entre ambos sistemas. Se pueden ver referenciadas en los ficheros de configuración de unidades en el campo WantedBy=.
  - 0 = **shutdown.target**, **poweroff.target**
  - **halt.target** para el sistema, pero no apaga el hardware
  - S = **emergency.target**
  - 1 = **rescue.target**
  - 3 = **multi-user.target**
  - 5 = **graphical.target**
  - 6 = **reboot.target**
  - **default.target** es un link a **multi-user.target** o **graphical.target**
    - `ls -l /etc/systemd/system/default.target` # Este link es usado por SystemD como target por defecto
  - También existen desde **runlevel0.target** a **runlevel6.target**
- `systemctl list-units -t target --all` # Para ver todos los targets disponibles
- `systemctl list-dependencies graphical.target` | grep target # Para ver las dependencias de las unidades tipo target
- `systemctl get-default` # Ver el target por defecto actual
- `systemctl set-default multi-user.target` # Configurar el modo multi-usuario por defecto
- `systemctl isolate multi-user.target` # Cambiar al modo multi-usuario (3)

Sockets

- Para cada unidad de tipo socket, debe existir una unidad de tipo servicio correspondiente
- Opciones
  - **ListenStream**: dirección de escucha para el stream (puede ser un puerto, ruta del socket o una dirección IPv4 o IPv6 junto con el puerto).
  - **Accept**: si es true, la instancia del servicio se lanza por cada conexión; si es false, la encargada de gestionar todas las conexiones es una instancia.
  - **MaxConnections**: es el número máximo de conexiones para un servicio
  - **Service**: servicio que activa el socket (por defecto es el servicio que tiene el mismo nombre).
- `man systemd.socket` #
  - https://www.man7.org/linux/man-pages/man5/systemd.socket.5.html

- `systemctl list-sockets` # Lista todos los sockets
- `alias sockets='systemctl --no-pager -t socket --state=running'` # Lista los activos
- https://www.linux.com/training-tutorials/end-road-systemds-socket-units/

Timers

- Para cada archivo **.timer**, existe un archivo **.service** coincidente (por ejemplo, foo.timer y foo.service).
  - El archivo **.timer** activa y controla el archivo **.service**.
  - El archivo **.service** no requiere una sección **[Install]** ya que son las unidades de temporizador las que se activan.
  - Si es necesario, es posible controlar una unidad con un nombre diferente usando la opción Unit= en la sección [Timer] del archivo temporizador.
- `systemctl list-timers` Para listar los «timers» de SystemD
- `journalctl -b -u .timer` # Para ver los timers ejecutados desde el arranque del sistema
- Si un temporizador se **desincroniza**, puede ayudar eliminar su archivo **stamp-*** en **/var/lib/systemd/timers** (o **~/.local/share/systemd/** en caso de temporizadores de usuario). Estos son archivos vacios que marcan la última vez que se ejecutó cada temporizador. Si se eliminan, se reconstruirán en el próximo inicio de su temporizador.
- Frecuencias con el parámetro OnCalendar (funciona igual que el cron)
  - **Realtime timers** (a.k.a. wallclock timers) **Temporizadores en tiempo real** (también conocido como «wallclock timers») se activan con un evento del calendario, de la misma manera que lo hacen los cronjobs. La opción OnCalendar= se utiliza para definirlos.
    - DayOfWeek Year-Month-Day Hour:Minute:Second
      - Con el DayOfWeek siendo opcional. Los operadores *, `/` y `,` tienen el mismo significado que los usados para los trabajos de cron, mientras que puede usar `..` entre dos valores para indicar un rango
      - ~ indica el último día del mes
      - OnCalendar=Mon -*-10..27 05:30:00
    - minutely # Cada minuto
    - hourly # cada hora al comienzo de la hora.
    - daily # una vez al día a medianoche.
    - weekly # una vez a la semana a medianoche del lunes.
    - monthly # una vez al mes a la medianoche del primer día del mes.
    - yearly # una vez al año a medianoche del primer día de enero.
    - quarterly # trimestral
    - semiannually # semianual
    - `man 7 systemd.time`
    - `systemd-analyze calendar weekly`
    - `systemd-analyze calendar «Mon,Tue -*-01..04 12:00:00»`
  - Monotonic timers - Temporizadores monotónicos se activan después de un intervalo de tiempo condicionado a un punto de inicio variable. Se detienen si el equipo está temporalmente suspendido o apagado. Hay varios temporizadores monotónicos diferentes
    - **OnActiveSec**= Defines a timer relative to the moment the timer unit itself is activated.
    - **OnBootSec**= Defines a timer relative to when the machine was booted up. In containers, for the system manager instance, this is mapped to OnStartupSec=, making both equivalent.
    - **OnStartupSec**= Defines a timer relative to when the service manager was first started. For system timer units this is very similar to OnBootSec= as the system service manager is generally started very early at boot. It's primarily useful when configured in units running in the per-user service manager, as the user service manager is generally started on first login only, not already during boot.
    - **OnUnitActiveSec**= Defines a timer relative to when the unit the timer unit is activating was last activated.

        - **nUnitInactiveSec**= Defines a timer relative to when the unit the timer unit is activating was last deactivated.
- `systemctl enable my.timer`
- `systemctl start my.timer`
- `systemctl status my.timer`
- Puede cambiar la frecuencia de su trabajo programado, modificando el valor OnCalendar y luego escribiendo el comando systemctl daemon-reload
- Ejemplos
    - /usr/lib/systemd/system/plocate-updatedb.timer

```
[Unit]
Description=Update the plocate database daily

[Timer]
OnCalendar=daily
RandomizedDelaySec=1h
AccuracySec=6h
Persistent=true

[Install]
WantedBy=timers.target
```

    - /usr/lib/systemd/system/plocate-updatedb.service

```
[Unit]
Description=Update the plocate database
ConditionACPower=true

[Service]
Type=oneshot
ExecStart=/usr/sbin/updatedb
LimitNOFILE=131072
IOSchedulingClass=idle

PrivateTmp=true
PrivateDevices=true
PrivateNetwork=true
```

- Unidades .timer transitorias (equivalentes a at)
    - `systemd-run --on-calendar='2019-10-06 11:30' date`
    - `systemd-run --on-active="2m" ./foo.sh`
    - `man 1 systemd-run`
- Información sobre los timers
    - https://wiki.archlinux.org/title/Systemd_(Espa%C3%B1ol)/Timers_(Espa%C3%B1ol%29 - Timers
    - `man 5 systemd.timer`

Mount

- **/etc/fstab**
    - El script systemd-fstab-generator(8) traduce todas las entradas presentes en /etc/fstab en

unidades de systemd, esto se realiza en el momento del arranque y cada vez que se vuelve a cargar la configuración del gestor del sistema.
    - En un disco particionado con GPT, el script systemd-gpt-auto-generator(8) montará particiones siguiendo la especificación de particiones detectables , por lo tanto, las mismas se pueden omitir de fstab.
- El nombre del unit file guarda relación con el punto de montaje del sistema de archivos. Si el punto de montaje es **/mnt/backups**, el nombre del unit file debe de ser mnt-backups.mount.
- `man 5 systemd.mount` #
- `man 1 systemd-mount` #
- `systemctl -t mount` #
- https://manuais.iessanclemente.net/index.php/Gesti%C3%B3n_de_Puntos_de_Montaje - Gestión de Puntos de Montaje
- https://wiki.archlinux.org/title/Systemd_(Espa%C3%B1ol)#Montaje - Mount Units

Automount

- Los automounts en systemd son un tipo de unidad que permite montar automáticamente sistemas de archivos en un directorio determinado cuando este es accedido por primera vez, y desmontarlos cuando ya no son necesarios.
- Para cada automount en systemd es necesario tener un mount correspondiente.
    - `man 5 systemd.automount` #
    - `systemctl -t automount` #
    - `systemctl list-automounts` #
    - `systemctl cat proc-sys-fs-binfmt_misc.automount` #

MATE EXTRA: https://community.hetzner.com/tutorials/automount-filesystems-with-systemd

PATH

- https://www.linux.com/topic/desktop/systemd-services-monitoring-files-and-directories/ - Systemd Services: Monitoring Files and Directories
- systemctl list-paths #

Temporales

- Hay varios servicios que crean o borran ficheros o directorios:
    - systemd-tmpfiles-clean.service
    - systemd-tmpfiles-setup-dev.service
    - systemd-tmpfiles-setup.service
        - systemctl cat systemd-tmpfiles-setup.service # Para su configuración

Configuración

- Los archivos de configuración se almacenan en los siguientes directorios:
    - /usr/lib/tmpfiles.d/
    - /run/tmpfiles.d/
    - /etc/tmpfiles.d/
- Los archivos de configuración se proporcionan normalmente junto con los archivos de servicio, y reciben

su nombre en el estilo /usr/lib/tmpfiles.d/**programa**.conf
- `cat /usr/lib/tmpfiles.d/tmp.conf` # Vemos el contenido de tmp.conf
  - q /tmp 1777 root root 10d
  - q /var/tmp 1777 root root 30d
- `systemd-tmpfiles --cat-config` # Vemos la configuración de TODOS los ficheros
- Para ver la sintaxis de la creación, borrado o gestión de los temporales:
  - https://www.freedesktop.org/software/systemd/man/latest/tmpfiles.d.html - Página man
  - man 5 tmpfiles.d # Para verlo en local
- Si hacemos un cambio en algún fichero, no lo va a ver hasta que rebotemos el sistema, o hasta que ejecutemos los siguientes comandos:
  - `systemd-tmpfiles --create` # Crea los temporales especificados
  - `systemd-tmpfiles --remove` # Borra los temporales especificados
  - `systemd-tmpfiles --clean` # Vacía el contenido de los temporales especificados
  - Es posible especificar las tres opciones juntas en una sola orden. En ese caso, `--create` siempre es lo que se ejecuta lo último.
- Los usuarios pueden tener sus propios ficheros temporales, definidos en el siguiente directorio:
  - **~/.config/user-tmpfiles.d/**
  - `systemd-tmpfiles --user --create` # Para gestionar sus propios temporales
  - `systemd-tmpfiles --user --cat-config` # Para verlos
- Uso
  - `df -t tmpfs -h` # Para ver los sistemas de ficheros de tipo **tmpfs** montados en memoria
  - `journalctl -b -u systemd-tmpfiles` # Para ver los logs
  - `systemctl cat systemd-tmpfiles-clean.timer` # Se ejecuta una vez al día, y 15 minutos después de arrancar
- Información general
  - https://wiki.archlinux.org/title/Systemd_(Espa%C3%B1ol)#Archivos_temporales
  - https://wiki.archlinux.org/title/Tmpfs_(Espa%C3%B1ol)
  - https://mastering-linux.com/engineer/systemd#managing-temporary-files
  - https://www.baeldung.com/linux/systemd-tmpfiles-configure-temporary-files

Seguridad, monitorización, análisis

- https://www.linuxjournal.com/content/systemd-service-strengthening - Systemd Service Hardening con el comando systemd-analyze security
- https://0pointer.de/blog/projects/security.html - Explicación de algunas de las características de seguridad
- https://github.com/tim-seoss/rest-server/commit/76d995edfc45b7ed913c1125c3a95f626841276e - Ejemplo de mejora de la seguridad de un comando

Arranque

- `systemd-analyze plot > graph1.svg ; eog graph1.svg` # Crea un gráfico de todo el arranque del sistema, y lo visualiza con la app gráfica eog
- `systemd-analyze blame | less` # Lo mismo, pero en modo texto, aunque no muestra dependencias ni paralelismo. Ordena la salida por tiempo que tarda cada componente en estar disponible, para poder ver a quien culpar (blame) en caso de retardo
- `systemd-analyze dot 'avahi-daemon.'` | dot -Tsvg > avahi.svg ; eog avahi.svg # Para ver las dependencias de un componente concreto en modo gráfico
- `systemd-analyze critical-chain` # Muestra el árbol de dependencias de targets y de los servicios que bloquean la activación de cada target
- https://www.thegeekdiary.com/systemd-analyze-command-examples-in-linux/ - Info

Monitorización

- `systemctl status` # Estado general de SystemD, mostrando todos los procesos lanzados con sus PIDs
- `systemctl --no-page --state=failed` # Para ver si alguna unidad ha fallado
- `systemctl --no-page` # Para ver todas las unidades lanzadas, ordenadas por tipo

Logs

journalctl

- https://atareao.es/tutorial/trabajando-con-systemd/journalctl-y-logs-en-systemd/ - Buen resumen
- https://www.solvetic.com/tutoriales/article/4051-administrar-logs-eventos-systemd-journalctl-linux/ - Cómo hacer persistentes los logs con journalctl, si no están almacenando
- Ejemplos de comandos:
    - journalctl -u ruiz-tapiador.service # Mensajes generados por el servicio y por SystemD
        - journalctl UNIT=ruiz-tapiador.service # Mensajes generados por SystemD (arranque del servicio, parada, etc...)
        - journalctl _SYSTEMD_UNIT=ruiz-tapiador.service # Mensajes generados por el propio servicio (problemas al cargar algo, errores..)
    - journalctl -p emerg..err # Mensajes con prioridad entre emerg y err
        - journalctl -p 0..4 # Mensajes con prioridad entre emerg y err
        - journalctl -p 4 # Mensajes con prioridad entre emerg y err
        - journalctl -p 4..4 # Mensajes con prioridad err
        - Colores de las prioridades:
            - 3-0 - rojo
            - 4 - amarillo
            - 5 - blanco en negrita
            - 6 - blanco
            - 7 - gris

From:
https://matebcn.eu/ - **miguel angel torres egea**

Permanent link:
https://matebcn.eu/info:cursos:pue:ethical-hacker:extras:sincara-systemd

Last update: **28/02/2025 12:51**