Engineer

demonio

programa en segundo plano que funciona sin atención humana

Qué es

Un demonio es un proceso de larga duración que ninguna persona inicia a mano: lo pone en marcha el sistema operativo o un gestor de servicios, no tiene una terminal y permanece activo para responder a peticiones o vigilar alguna parte de la máquina. En inglés, el plural es daemons, y muchos llevan la palabra en el nombre como una letra d final: `sshd` acepta conexiones SSH, `crond` ejecuta tareas programadas y `systemd-journald` reúne registros. Windows llama servicio al mismo tipo de programa, al igual que systemd, que hoy supervisa la mayoría de los demonios en Linux.

La receta tradicional de Unix, que el manual `daemon(7)` de Linux aún expone en quince pasos numerados, hace que el programa se independice por sí mismo. Cierra los descriptores de archivo heredados, restablece los controladores de señales, crea un proceso hijo, llama a `setsid()` para salir de la sesión de la terminal y vuelve a crear un hijo para que nunca pueda recuperar una terminal. El primer hijo termina para que PID 1 adopte al superviviente, que dirige sus flujos estándar a `/dev/null`, cambia a `/`, escribe un archivo PID y reduce sus privilegios. La llamada `daemon()` de la biblioteca de C realiza parte de estos pasos, y el manual desaconseja utilizarla porque solo implementa un subconjunto de los pasos.

El demonio de estilo nuevo descrito en el mismo manual, el que esperan tanto systemd como launchd de Apple, invierte la receta. El programa permanece en primer plano porque el supervisor ya lo inició separado de la terminal, con un entorno limpio, el usuario correcto y el directorio de trabajo adecuado. Escribe los registros en la salida de error estándar, donde los recoge el gestor de servicios, avisa cuando está listo para atender, termina limpiamente con SIGTERM y recarga su configuración con SIGHUP. La guía de Apple para tareas de launchd

es tajante sobre la costumbre anterior: una tarea no debe convertirse por sí misma en demonio y, si crea un hijo y termina, launchd concluye que el proceso ha muerto.

Se confunde con dos conceptos cercanos. Una tarea programada, ejecutada por cron o un temporizador de systemd, empieza, hace su trabajo y termina, mientras que un demonio permanece. Además, macOS divide el término en dos: un launch daemon atiende a todo el sistema y se carga al arrancar, mientras que un launch agent pertenece a un usuario y solo se ejecuta mientras este tiene una sesión iniciada. Linux traza la misma separación con las instancias de systemd por usuario.

Utilice el concepto en una decisión

Aplique «¿Qué es un demonio?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. El cuidado diario de servidores, correo, cuentas y actualizaciones para que los servicios de los que depende todo el mundo sigan funcionando sin sorpresas.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 10 servidores PostgreSQL y MS SQL en Ubuntu — 10 servidores de bases de datos desplegados y mantenidos

10 servidores PostgreSQL y MS SQL en Ubuntu Póngase en contacto →

Por qué importa

Cada demonio de un servidor es un compromiso permanente. Ocupa memoria lo utilice alguien o no, y alguien tiene que aplicarle los parches cuando el proveedor publica una corrección. Si escucha en un puerto de red, forma parte de la superficie de ataque durante cada hora que se ejecuta. Por eso la pregunta útil rara vez es qué demonio instalar, sino si el trabajo necesita un proceso residente.

Tres formas cubren la mayoría del trabajo. Algo que ocurre según una programación, como una copia de seguridad, un informe o la renovación de un certificado, no necesita un proceso residente: un temporizador lo inicia, se ejecuta hasta terminar y entre ejecuciones no consume recursos. Algo que responde a peticiones ocasionales puede activarse por socket: el gestor de servicios posee el socket de escucha e inicia el demonio en la primera conexión; el socket sigue vinculado al puerto durante los reinicios, así que las peticiones entrantes esperan en una cola en vez de ser rechazadas. Solo el trabajo que debe

reaccionar continuamente, como un servidor web, una base de datos o un agente de transferencia de correo, justifica un proceso siempre activo. En una máquina pequeña, esa elección decide si todo cabe en memoria.

Cada demonio de terceros que va a producción trae una segunda decisión, y cuatro preguntas la resuelven. ¿Permanecerá en primer plano cuando se le pida? ¿Escribe en la salida de error estándar o exige archivos de registro propios? ¿Puede comunicar al supervisor que está listo, en lugar de que solo ha arrancado? ¿Necesita root o puede entregarle el gestor de servicios el único recurso privilegiado que requiere, como un socket en el puerto 443? Uno que responda bien a las cuatro se puede mantener en funcionamiento durante años con poco coste; uno que no las supere llega con un script envoltorio, su propia rotación de registros y una excepción en la siguiente revisión de seguridad.

Cómo se aplica en la práctica

Servir el sitio por Gemini y Gopher costó dos procesos residentes, no una segunda compilación. Cuando se crearon réplicas del sitio para ambos protocolos, la compilación que escribe HTML también escribió 777 menús Gopher y el mismo número de documentos Gemini en la etapa de implementación documentada, y dos pequeños demonios en el mismo host se encargaron de servirlos. Una comparación separada de compilaciones registró cero bytes HTML cambiados al añadir los formatos de salida; ese resultado no describe los cambios editoriales o visuales posteriores.

La réplica onion de Tor muestra la parte de red: su demonio construye circuitos salientes, de modo que nada inicia conexiones hacia dentro y no se abre ningún puerto del cortafuegos. El rol que configura la réplica onion instala el demonio desde el paquete de la propia distribución, escribe una configuración con el puerto del proxy de salida desactivado y se niega a sobrescribir una identidad que no generó. Un paso enseñó una lección general: la clave llegó correctamente, la ejecución de convergencia no notificó cambios y la dirección nueva quedó sin publicar porque, en ese momento, nada en la configuración mencionaba el directorio nuevo. El rol falla ahora siempre que hay una identidad en disco sin ningún servicio que la mencione.

Los informes muestran lo que puede ocultar un demonio silencioso. Seis recopiladores de solo lectura que representan el host como Markdown se revisaron cuando quedó claro que una recopilación a ciegas y un host sano imprimían la misma página: una réplica cuyo demonio cambió el formato del

registro notificaba el mismo cero que otra que nadie visitaba. Los recopiladores informan ahora, entre otras cosas, de las líneas del diario leídas frente a las reconocidas, de modo que un demonio que cambia lo que escribe aparece como una carencia de observación, no como tranquilidad.

Dónde surgen los problemas

El servicio arranca y se detiene un segundo después. Un demonio de estilo antiguo crea un hijo y deja que el padre termine, como indica la receta. Con el `Type=simple` predeterminado de systemd, el proceso que este inició es el principal, por lo que su salida significa que el servicio se ha detenido y, con el `KillMode=control-group` predeterminado, se mata al hijo que debía continuar, junto con todos los demás procesos del grupo de control de la unidad. Ejecute el programa con su opción de primer plano; para los pocos que no pueden cambiarse, declare `Type=forking` con `PIDFile=` , un tipo que el manual desaconseja para código nuevo.

Los clientes se conectan demasiado pronto y son rechazados. Con `Type=simple` , systemd considera iniciado el servicio en cuanto ha creado el proceso, así que las unidades ordenadas después empiezan mientras el demonio todavía lee su configuración. El manual de servicios añade que `systemctl start` comunica éxito con ese tipo incluso si falta el binario o no existe el usuario configurado. Prefiera `Type=exec` , que espera a que el binario se haya ejecutado realmente, y, para cualquier servicio del que dependan otros, deje que el demonio anuncie por sí mismo que está listo mediante `Type=notify` :

```
#include <systemd/sd-daemon.h>

int main(void) {
    load_config();
    open_listeners();
    sd_notify(0, "READY=1"); /* units ordered after this one start now */
    return serve_forever();
}
```

Un script de parada mata un proceso ajeno. Un demonio que crea un hijo escribe su ID de proceso en un archivo para que los scripts puedan enviarle señales más tarde. Tras un fallo, el archivo permanece, el kernel entrega el mismo número a un proceso sin relación y la siguiente parada envía SIGTERM al programa equivocado. El manual de servicios aconseja evitar por completo

los archivos PID en proyectos modernos. Un supervisor que inició el proceso lo sigue mediante su grupo de control y no tiene ningún archivo que pueda quedar obsoleto.

Un demonio muere al arrancar y no deja rastro. La receta antigua dirige la salida estándar y la de error a `/dev/null`, de modo que desaparece un error producido antes de que el demonio abra su propio archivo de registro. Bajo un supervisor, deje ambos flujos sin redirigir: todo lo que escriba el demonio en la salida de error estándar llega al diario, donde `journalctl -u` lo lee unidad por unidad.

Explotar un solo fallo entrega toda la máquina. Un demonio que se ejecuta como root porque necesitaba el puerto 80 al arrancar conserva root en todas las peticiones que atiende después. Deje que el gestor de servicios vincule el socket privilegiado y se lo entregue, y confíe al supervisor la reducción de privilegios, como recomienda `daemon(7)`; opciones como `NoNewPrivileges=` y `ProtectSystem=` limitan lo que puede tocar un proceso comprometido.

Lecturas adicionales

- daemon(7) — los quince pasos de la receta clásica y lo que hace en su lugar un demonio de estilo nuevo.
- daemon(3) — el atajo de la biblioteca de C y el defecto relacionado con el líder de sesión que documenta su propio manual.
- sd_notify(3) — cómo comunica un demonio a systemd que está listo, recargando o deteniéndose.
- systemd.socket(5) — activación por socket: el gestor conserva el puerto e inicia el demonio cuando hace falta.
- setsid — la definición POSIX de la llamada que separa un proceso de su terminal.
- Crear demonios y agentes de lanzamiento — las reglas de Apple para las tareas de launchd, incluido por qué nunca deben crear un hijo y terminar.



Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es un demonio?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por **D.**

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/daemon/