Engineer

Démon

programme en arrière-plan qui fonctionne sans surveillance

Ce que c’est

Un démon est un processus durable que personne ne lance à la main : le système d’exploitation ou un gestionnaire de services le démarre, il n’a pas de terminal et reste actif pour répondre aux requêtes ou surveiller une partie de la machine. Le pluriel est démons, et beaucoup portent ce mot dans leur nom sous la forme d’un d final : `sshd` accepte les connexions SSH, `crond` exécute les tâches planifiées et `systemd-journald` rassemble les journaux. Windows appelle ce type de programme un service, tout comme systemd, qui supervise aujourd’hui la plupart des démons sous Linux.

La recette Unix traditionnelle, que le manuel Linux `daemon(7)` décrit encore en quinze étapes numérotées, demande au programme de se détacher lui-même. Il ferme les descripteurs de fichiers hérités, réinitialise les gestionnaires de signaux, crée un processus fils, appelle `setsid()` pour quitter la session du terminal, puis crée un autre fils pour ne jamais pouvoir récupérer un terminal. Le premier fils se termine pour que PID 1 adopte le survivant. Celui-ci dirige ses flux standard vers `/dev/null`, passe dans `/`, écrit un fichier PID et abandonne ses privilèges. L’appel `daemon()` de la bibliothèque C en fait une partie, et le manuel le déconseille parce qu’il n’applique qu’un sous-ensemble des étapes.

Le démon moderne décrit par le même manuel, celui qu’attendent systemd et launchd d’Apple, inverse la recette. Le programme reste au premier plan, car le superviseur l’a déjà démarré détaché, avec un environnement propre, le bon utilisateur et le bon répertoire de travail. Il écrit ses journaux sur la sortie d’erreur standard, où le gestionnaire les recueille, annonce qu’il est prêt à servir, se termine proprement sur SIGTERM et recharge sa configuration sur SIGHUP. Le guide d’Apple sur les tâches launchd est net sur l’ancienne habitude : une tâche ne doit pas se transformer elle-même en démon ; si elle crée un fils puis se termine, launchd conclut que le processus est mort.

Deux voisins sont souvent confondus avec lui. Une tâche planifiée, lancée par cron ou un timer systemd, démarre, travaille puis se termine, tandis qu'un démon reste. Et macOS sépare le mot en deux : un launch daemon sert tout le système et se charge au démarrage ; un launch agent appartient à un utilisateur et ne fonctionne que pendant sa connexion. Linux trace la même limite avec les instances systemd par utilisateur.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'un démon ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. L'entretien quotidien des serveurs, de la messagerie, des comptes et des mises à jour, pour que les outils indispensables restent sans histoire.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 10 serveurs PostgreSQL et MS SQL sur Ubuntu — 10 serveurs de bases de données déployés et maintenus

10 serveurs PostgreSQL et MS SQL sur Ubuntu Contactez-nous →

Pourquoi c'est important

Chaque démon d'un serveur est un engagement permanent. Il occupe de la mémoire, qu'on l'utilise ou non, et quelqu'un doit lui appliquer un correctif chaque fois que le fournisseur en publie un. S'il écoute sur un port réseau, il fait partie de la surface d'attaque pendant toute sa durée de fonctionnement. La question utile consiste donc rarement à choisir quel démon installer, mais à déterminer si ce travail a vraiment besoin d'un processus résident.

Trois formes couvrent la plupart des travaux. Une activité planifiée, comme une sauvegarde, un rapport ou un renouvellement de certificat, n'a pas besoin d'un processus résident : un timer la démarre, elle s'exécute jusqu'au bout et ne coûte rien entre deux exécutions. Une activité répondant à des requêtes occasionnelles peut être activée par socket : le gestionnaire détient le socket d'écoute et lance le démon à la première connexion. Le socket reste associé à son adresse pendant les redémarrages ; les requêtes entrantes attendent donc en file plutôt que d'être refusées. Seul un travail devant réagir continuellement, serveur web, base de données ou agent de transfert de courrier, mérite un processus toujours actif. Sur une petite machine, ce choix détermine si tout tient en mémoire.

Une seconde décision accompagne chaque démon tiers destiné à la production, et quatre questions la tranchent. Reste-t-il au premier plan si on le lui demande ? Écrit-il sur la sortie d'erreur standard, ou impose-t-il ses propres fichiers journaux ? Peut-il dire au superviseur qu'il est prêt, pas simplement démarré ? A-t-

il besoin de root, ou le gestionnaire peut-il lui donner la seule ressource privilégiée nécessaire, comme un socket sur le port 443 ? Un démon qui répond bien aux quatre coûte peu à exploiter pendant des années ; celui qui échoue arrive avec un script d'encapsulation, sa propre rotation des journaux et une exception à la prochaine revue de sécurité.

À quoi cela ressemble en pratique

Servir le site en Gemini et Gopher a demandé deux processus résidents, pas une seconde construction. Lors de la création des miroirs sur les deux protocoles, la construction qui écrivait le HTML a aussi écrit 777 menus Gopher et autant de documents Gemini au stade d'implémentation consigné, et deux petits démons du même hôte les ont servis. Une comparaison de construction distincte a constaté zéro octet HTML modifié lors de l'ajout des formats de sortie ; ce résultat ne décrit pas les modifications éditoriales ou visuelles ultérieures.

Le miroir onion Tor montre le côté réseau : son démon construit des circuits sortants ; aucune connexion entrante n'est donc établie et aucun port du pare-feu n'est ouvert. Le rôle qui configure le miroir onion installe le démon depuis le paquet de la distribution, écrit une configuration désactivant le port du proxy sortant et refuse d'écraser une identité qu'il n'a pas générée. Une étape a enseigné une leçon générale : la clé était bien en place, la convergence ne signalait aucun changement et la nouvelle adresse n'était pas publiée, car rien dans la configuration ne nommait alors le nouveau répertoire. Le rôle échoue désormais dès qu'une identité est sur disque sans qu'aucun service ne la désigne.

Les rapports montrent ce qu'un démon silencieux peut cacher. Six collecteurs en lecture seule qui représentent l'hôte en Markdown ont été remaniés lorsqu'il est devenu clair qu'une collecte aveugle et un hôte sain produisaient la même page : un miroir dont le démon avait changé le format de ses journaux affichait le même zéro qu'un miroir sans visiteurs. Les collecteurs indiquent désormais, entre autres, le nombre de lignes du journal lues et celui des lignes reconnues. Un démon qui change ce qu'il écrit apparaît donc comme une lacune plutôt que comme du calme.

Où les problèmes surviennent

Le service démarre et s'arrête une seconde plus tard. Un démon traditionnel crée un fils et laisse son parent se terminer, conformément à la recette. Avec `Type=simple` , le réglage préconfiguré de systemd, le processus lancé par systemd

est le processus principal ; sa fin signifie donc l'arrêt du service. Avec le réglage préconfiguré `KillMode=control-group`, le fils censé poursuivre est tué avec tous les autres processus du groupe de contrôle de l'unité. Lancez le programme avec son option de premier plan. Pour les rares programmes impossibles à changer, déclarez `Type=forking` avec `PIDFile=`, un type déconseillé par le manuel pour du code nouveau.

Les clients se connectent trop tôt et sont refusés. Avec `Type=simple`, systemd juge le service démarré dès qu'il a créé le processus fils ; les unités dont le démarrage est ordonné après le sien démarrent donc alors que le démon lit encore sa configuration. Le manuel ajoute que `systemctl start` annonce un succès avec ce type même si le binaire manque ou si l'utilisateur configuré n'existe pas. Préférez `Type=exec`, qui attend l'exécution réelle du binaire. Pour tout service dont d'autres dépendent, laissez le démon annoncer lui-même qu'il est prêt sous `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 d'arrêt tue un inconnu. Un démon qui crée un fils écrit son identifiant de processus dans un fichier pour que des scripts puissent ensuite lui envoyer des signaux. Après un plantage, le fichier reste ; le noyau attribue le même nombre à un processus sans rapport et l'arrêt suivant envoie SIGTERM au mauvais programme. Le manuel conseille d'éviter entièrement les fichiers PID dans les projets modernes. Le superviseur qui a lancé le processus le suit par son groupe de contrôle et n'a aucun fichier susceptible de devenir périmé.

Un démon meurt au démarrage sans laisser de trace. L'ancienne recette dirige la sortie standard et la sortie d'erreur vers `/dev/null` ; une erreur survenant avant que le démon ouvre son propre fichier journal disparaît donc. Sous un superviseur, laissez les deux flux tels quels : tout ce que le démon écrit sur la sortie d'erreur standard va dans le journal, où `journalctl -u` le lit unité par unité.

Une seule faille exploitée livre toute la machine. Un démon exécuté comme root parce qu'il lui fallait le port 80 au démarrage reste root pour chaque requête ultérieure. Laissez le gestionnaire associer le socket privilégié à son adresse et le transmettre, puis confiez-lui l'abandon des privilèges, comme le recommande `daemon(7)` . Des réglages tels que `NoNewPrivileges=` et `ProtectSystem=` limitent ce qu'un processus compromis peut atteindre.

Pour aller plus loin

- daemon(7) — les quinze étapes de la recette classique et ce que fait à la place un démon moderne.
- daemon(3) — le raccourci de la bibliothèque C et le défaut lié au rôle de chef de session, décrit dans son propre manuel.
- sd_notify(3) — comment un démon dit à systemd qu'il est prêt, recharge sa configuration ou s'arrête.
- systemd.socket(5) — l'activation par socket, où le gestionnaire détient le port et démarre le démon à la demande.
- setsid — la définition POSIX de l'appel qui détache un processus de son terminal.
- Créer des Launch Daemons et Agents — les règles d'Apple pour les tâches launchd, dont les raisons de ne jamais créer un fils puis se terminer.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'un démon ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

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

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/daemon/