Engineer

systemd

те, що запускає все інше в Linux

Що це таке

Коли ядро Linux завершує власний старт, воно передає керування одному процесові з ідентифікатором 1, і майже в усіх поширених дистрибутивах цим процесом є systemd. Це менеджер системи та служб: він піднімає машину під час завантаження, запускає служби й наглядає за ними, поки вона працює, а під час вимкнення зупиняє все в правильному порядку. Проєкт пише свою назву малими літерами, без варіантів на кшталт SystemD чи System D, а кінцева d — давня позначка демона в Unix. Він належить до родини систем ініціалізації, поруч зі скриптами SysV init та Upstart, які він замінив у більшості дистрибутивів, з OpenRC і runit, які досі обирають деякі менші дистрибутиви, і з launchd, що виконує ту саму роботу в macOS.

Усе, чим керує systemd, — це юніти, описані в коротких текстових файлах із секціями та парами «ключ — значення». Посібник налічує 11 типів: service, socket, target, device, mount, automount, timer, swap, path, slice і scope. Юніт типу service запускає програму, а timer за розкладом запускає інший юніт. Socket тримає порт, що слухає з'єднання, і запускає свою службу, коли надходить перше з них, а target об'єднує юніти в стан, якого машина може досягти, наприклад багатокористувацьку систему з мережею. Вимоги й порядок задають окремі параметри, тож «цьому потрібне те» і «це стартує після того» — два різні рядки, а менеджер збирає їх в одну транзакцію завдань запуску й зупинки, яку можна перевірити ще до виконання.

Від скриптових систем ініціалізації, що були до нього, його відрізняють два рішення. Кожна служба працює у власній контрольній групі ядра, тож менеджер відстежує процеси в цій групі й може зупинити їх разом, обмежити їхню пам'ять і врахувати витрати процесора. А ще проєкт — це цілий набір компонентів, а не лише менеджер: його власний сайт називає демон журналювання, journal, і компоненти для входу користувачів,

налаштування мережі, синхронізації часу та розв’язання імен. Які з них увімкнути, дистрибутиви вирішують самі; менеджер на PID 1 — та частина, яка є майже скрізь.

Застосуйте поняття до рішення

Застосуйте «Що таке systemd?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Щоденний догляд за серверами, поштою, обліковими записами й оновленнями, щоб те, на що всі покладаються, працювало тихо й без сюрпризів.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Підтримувала 10 серверів PostgreSQL і MS SQL — 10 серверів баз даних розгорнуто й підтримувано в справному стані

Підтримувала 10 серверів PostgreSQL і MS SQL Зв'язатися з нами →

Чому це важливо

На сервері з Linux питання рідко звучить так, чи запускати systemd: дистрибутив уже зробив цей вибір. Справжнє рішення — чи використовувати його як супервізор і планувальник, чи нашаровувати поверх нього інший менеджер процесів, політику перезапуску контейнерного демона або cron.

Якщо використовувати його напряму, порядок запуску, політика перезапуску, журнали, обмеження ресурсів і пісочниця опиняються в одному файлі на службу, який читає вже запущений менеджер, без другого процесу-супервізора. Юніт може обмежити пам’ять через `MemoryMax=`, зробити системні каталоги доступними службі лише для читання через `ProtectSystem=` і запускати її від тимчасового користувача через `DynamicUser=`, а `systemd-analyze security` оцінює, наскільки відкрита кожна служба, за шкалою від 0,0 до 10,0. Завдання, яке запускає таймер, на відміну від рядка в crontab, пише в журнал, отримує ту саму пісочницю, що й будь-яка служба, а `systemctl list-timers` показує його останній і наступний запуск. Навичка ще й переноситься: інженер, який уміє читати юніт-файл, може обслуговувати більшість серверів Linux незалежно від дистрибутива.

Ціна теж реальна. Юніти працюють лише в Linux, тож команда, яка постачає ще й на macOS, пише для launchd файли property list, а та, що постачає на BSD, пише rc-скрипти. У більшості контейнерів systemd немає взагалі: PID 1 там — сама програма, а політика перезапуску належить тому, що запускає контейнер. Там, де контейнери й звичайні служби ділять один хост,

Quadlet від Podman скорочує цю відстань: він перетворює опис контейнера на звичайний юніт служби, тож обидва види навантаження стоять під одним супервізором.

Як це виглядає на практиці

Власна платформа компанії спирається на systemd у трьох місцях.

У записі про реалізацію за вересень 2026 року виробничий хост, описаний у роботі про публічні сервіси компанії на одному хості з 512 МБ, мав одне ядро й близько 464 МБ придатної до використання пам’яті. На ньому постійно працювали вебсервер, що обслуговує TLS для семи доменів, git-фордж, onion-сервіс, по одному серверу Gemini і Gopher та блокування спроб вторгнення. Автоматизацію резервного копіювання реалізовано окремо; це не підтверджує наявності активних зашифрованих резервних копій. За такого розміру кожен постійний процес мав виправдати використання пам’яті, тому супервізором був systemd, без другого процесу-супервізора над ним, а контейнери працювали як юніти Quadlet під тим самим менеджером, а не під власним контейнерним демоном. Коли постало питання, чи може хост також обслуговувати onion-сервіс, вибірки, зафіксовані протягом доби вимірювань, показали, що доступна пам’ять жодного разу не опускалася нижче приблизно 310 МБ із 464.

Реалізація резервного копіювання та тестового відновлення використовує юніти systemd для дампа бази даних, якщо його налаштовано, надсилання зашифрованого знімка restic із дедуплікацією до сховища іншого провайдера через SFTP, очищення за політикою зберігання та перевірки цілісності. Окремий щомісячний процес відновлює відомий файл і порівнює його байти, а резервне копіювання й відновлення мають окремі сигнали про успіх. Таймери залежать від налаштованого репозиторію; у записі про реалізацію від 2026-10-27 зазначено, що за його відсутності вони були вимкнені, і не зафіксовано жодного завершеного реального відновлення. Під час роботи також зіткнулися з однією з підступних особливостей systemd, описаною нижче: парольна фраза зберігається у захищеному файлі, який читають юніти, бо у значенні, переданому через середовище юніта, обробляється екранування, і зворотна скісна риска могла б непомітно змінити парольну фразу.

Сторінка технологій проводить загальну межу: звичайні юніти systemd як основний вибір, а Docker і Kubernetes — для навантажень, які на них заслуговують.

Де виникають проблеми

Відредагований юніт працює зі старими налаштуваннями. Менеджер тримає юніт-файли в пам'яті, тож зміна на диску не діє, доки `systemctl daemon-reload` не перебудує його уявлення про них, а `systemctl cat` може показати файл, який відрізняється від того, чим користується запущений менеджер. Вносьте зміни через `systemctl edit` , який сам перечитує конфігурацію, або перечитайте її вручну й потім перезапустіть службу.

Секрет доходить до служби без одного символу. Значення в `Environment=` проходять через власні правила лапок systemd: екранування в стилі C обробляється, а буквальну зворотну скісну риску треба подвоювати. Документація також попереджає, що змінні середовища видимі непривілейованим клієнтам через D-Bus. Кладіть секрети у файл і передавайте їх через `LoadCredential=` : служба отримає копію лише для читання, доступну тільки її користувачеві та root.

Оновлення пакета тихо скасовує локальне виправлення. Зміни у файлі постачальника в `/usr/lib/systemd/system/` наступне оновлення пакета перезапише. Натомість кладіть їх у drop-in-файл у `/etc/systemd/system/<unit>.d/` , який накладається поверх оригіналу й переживає оновлення. Списки команд доповнюються, а не замінюються, тож drop-in, що змінює команду запуску, спершу очищає її порожнім присвоєнням:

```
# /etc/systemd/system/app.service.d/override.conf
[Service]
ExecStart=
ExecStart=/usr/local/bin/app --config /etc/app/app.toml
Restart=on-failure
```

Нічне завдання пропускає ніч, коли сервер був вимкнений. Календарний таймер, чий час настав, поки машина стояла, просто чекає наступного слоту, бо `Persistent=` типово має значення false. Задайте `Persistent=true` для

резервних копій та інших обов'язкових завдань, і пропущений запуск відбудеться, коли таймер знову стане активним, з урахуванням випадкової затримки, якщо її налаштовано.

Служба, що впала, так і лишається лежати. `Restart=` типово має значення `no`, тож служба, яка аварійно завершилася, лишається мертвою, доки хтось це помітить. Задайте `Restart=on-failure` для служб, які мають працювати постійно, і пам'ятайте: коли юніт досягає ліміту запусків, менеджер відмовляється від нього, а ручний запуск доведеться відкласти, доки мине інтервал або хтось виконає `systemctl reset-failed`.

Що почитати далі

- systemd(1) — сам менеджер: типи юнітів, послідовність завантаження і сигнали, на які він реагує.
- systemd.unit(5) — юніт-файли, залежності, ліміти запусків і каталоги drop-in, що їх перевизначають.
- systemd.service(5) — типи служб, політика перезапуску і те, як скидаються списки команд.
- systemd.exec(5) — середовище виконання, пісочниця і передавання облікових даних службам.
- systemd.timer(5) — календарні та монотонні таймери, зокрема `Persistent=`.
- systemctl(1) — командний рядок для запуску, редагування, перечитування й перегляду юнітів.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке systemd?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/systemd/