Engineer

デーモン

利用者の操作なしに、バックグラウンドで動き続けるプログラム

どのようなものか

デーモンは、人が手作業で起動するのではなく、OSやサービスマネージャーが起動する、長く動き続けるプロセスです。端末を持たず、リクエストに応じたり、マシンの一部を見守ったりするために常駐します。英語の複数形はdaemonsで、名前の末尾にdを付けるものも多くあります。`sshd` はSSH接続を受け付け、`crond` は予定したジョブを実行し、`systemd-journald` はログを集めます。Windowsは、同じ種類のプログラムをサービスと呼びます。現在、Linuxのデーモンの大半を監督するsystemdでも、同じ呼び方をします。

Linuxの `daemon(7)` マニュアルが今も十五の番号付き手順で説明する、従来のUnixの方法では、プログラムが自分自身を端末から切り離します。継承したファイル記述子を閉じ、シグナルハンドラーを初期化し、forkし、`setsid()` で端末のセッションを離れます。さらにforkして、二度と端末を取得できないようにします。最初の子プロセスが終了すると、残ったプロセスをPID 1が引き取ります。そのプロセスは、標準ストリームを `/dev/null` へ向け、作業ディレクトリーを `/` に移し、PIDファイルを書き、権限を落とします。Cライブラリーの `daemon()` は、この一部を行いますが、一部の手順しか実装しないため、マニュアルは使用を勧めていません。

同じマニュアルにある新方式のデーモン、つまりsystemdやAppleのlaunchdが期待する方式は、この手順を逆転させます。プログラムは、フォアグラウンドにとどまります。監督する側が既に、端末から切り離し、クリーンな実行環境、適切な利用者、適切な作業ディレクトリーで起動しているためです。ログは標準エラー出力へ書き、サービスマネージャーが収集します。対応の準備ができたことを通知し、SIGTERMで正常に終了し、SIGHUPで設定を読み直します。Appleのlaunchdジョブのガイドは、以前の習慣を明確に禁じています。ジョブは自らデーモン化してはならず、forkして終了すると、launchdはプロセスが死んだと判断します。

混同しやすい、近い概念が二つあります。cronやsystemdタイマーで実行する定期ジョブは、起動し、仕事を終えて終了しますが、デーモンは常駐します。また、macOSでは役割を二つに分けます。launch daemonはシステム全体へサービスを提供し、起動時に読み込まれます。launch agentは一人の利用者に属し、その人のログイン中だけ動きます。Linuxも、systemdの利用者別インスタンスで、同じ区別を設けます。

概念を判断に生かす

「デーモンとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 サーバー、メール、アカウント、更新を日々管理し、誰もが頼る仕組みを平穏に動かし続けます。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 UbuntuでPostgreSQL・MS SQLサーバー10台を運用 — 10 導入し正常に保ったDBサーバー

UbuntuでPostgreSQL・MS SQLサーバー10台を運用 お問い合わせ →

なぜ重要なのか

サーバーへデーモンを一つ置くたび、継続的な負担が生まれます。誰も使っていなくてもメモリーを占め、ベンダーが修正を出すたび、誰かがパッチを適用する必要があります。ネットワークポートで待ち受けるなら、稼働するすべての時間にわたり、攻撃対象の一部になります。そのため、有用なのは、多くの場合、どのデーモンを入れるかではなく、そもそもその仕事に常駐プロセスが必要かという問いです。

多くの仕事は、三つの形で捉えられます。バックアップ、レポート、証明書更新など、予定した時刻に行うものは、常駐プロセスが不要です。タイマーが起動し、完了まで動き、実行と実行の間はコストがかかりません。ときどき来るリクエストに応じるものは、ソケットで起動できます。サービスマネージャーが待ち受けソケットを持ち、最初の接続でデーモンを起動します。再起動をまたいでもソケットをバインドしたまま保持するため、到着したリクエストは拒否されず、キューで待ちます。常に起動するプロセスが必要なのは、ウェブサーバー、データベース、メール転送エージェントのように、継続して応答する仕事だけです。小さなマシンでは、この選択が、すべてをメモリーに収められるかを決めます。

本番へ導入する第三者製デーモンには、もう一つ判断が必要で、四つの問いで決められます。指定すればフォアグラウンドにとどまるか。標準エラー出力へログを書くか、それとも独自のログファイルを要求するか。単に起動しただけでなく、準備ができたことを監督側へ知らせられるか。rootが必要か、それとも443番ポートのソケットのような、必要な特権リソースだけをサービスマネージャー

から受け取れるか。四つに適切に対応するものは、何年も低いコストで運用できます。できないものは、ラッパースクリプト、独自のログローテーション、次のセキュリティーレビューでの例外を伴います。

実際の活用例

GeminiとGopherでサイトを配信するために必要だったのは、二つの常駐プロセスで、二つ目のビルドではありません。両プロトコルへサイトを複製したとき、実装段階の記録では、HTMLを書くビルドが777件のGopherメニューと、同数のGemini文書も書き出し、同じホスト上の小さなデーモン二つが配信を受け持ちました。別途行ったビルド比較では、出力形式を追加した際のHTMLの変更はゼロバイトでした。この結果は、その後の文章や見た目の変更を示すものではありません。

Torのonionミラーでは、ネットワーク側の性質が分かります。デーモンは外向きの回路を作るため、外から接続を受けず、ファイアウォールのポートも開きません。onionミラーを設定するロールは、ディストリビューション自身のパッケージからデーモンを入れ、外向きプロキシポートを無効にした設定を書き、自分で生成していない識別情報の上書きを拒否します。ある手順は、一般化できる教訓を残しました。鍵は正しく配置され、構成の適用は変更なしと報告したのに、新しいアドレスは公開されませんでした。その時点では、新しいディレクトリーを設定のどこからも参照していなかったためです。現在のロールは、ディスク上の識別情報を参照するサービスがなければ、必ず失敗します。

レポートは、静かなデーモンが何を隠せるかも示します。ホストをMarkdownで表す六つの読み取り専用コレクターは、観測できていない収集結果と、正常なホストが、同じページを出すと判明してから作り直しました。デーモンのログ形式が変わったミラーが、誰も訪問していないミラーと同じゼロを報告していたのです。現在は、たとえばジャーナルから読んだ行数と、認識できた行数を併記します。デーモンが出力を変えたら、平穏ではなく、観測の欠落として現れます。

問題が起きるところ

サービスが起動して、一秒後に止まります。 旧方式のデーモンは、従来の手順どおり、forkして親を終了させます。systemdの既定の `Type=simple` では、systemdが起動したプロセスがメインプロセスです。その終了は、サービスの停止を意味します。既定の `KillMode=control-group` では、動き続けるはずの子も、ユニットの制御グループ内のほかのプロセスとともに終了させられます。プログラ

ムにフォアグラウンドで動くフラグを指定します。変更できない少数のものには、`Type=forking` と `PIDFile=` を指定します。ただし、この型は、新しいコードには使わないよう、マニュアルが勧めています。

クライアントが早過ぎるタイミングで接続し、拒否されます。 `Type=simple` では、systemdはforkした時点でサービスを起動済みと見なします。そのため、後続に指定したユニットは、デーモンがまだ設定を読んでいる間に起動します。サービスのマニュアルは、この型では、バイナリーが存在しない場合や、設定した利用者が存在しない場合にも、`systemctl start` が成功を報告すると補足しています。バイナリーが実際に実行されるまで待つ `Type=exec` を優先します。ほかが依存するサービスでは、`Type=notify` を使い、デーモン自身に準備完了を通知させます。

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

int main(void) {
    load_config();
    open_listeners();
    /* 後続のユニットが起動可能になる */
    sd_notify(0, "READY=1");
    return serve_forever();
}
```

停止スクリプトが、無関係なプロセスを止めます。 forkするデーモンは、後からスクリプトでシグナルを送れるよう、プロセスIDをファイルに書きます。クラッシュ後もファイルが残り、カーネルが同じ番号を無関係なプロセスへ渡すと、次の停止処理が、別のプログラムへSIGTERMを送ります。サービスのマニュアルは、現代のプロジェクトでは、PIDファイル自体を避けるよう勧めています。起動した監督側は、制御グループを通してプロセスを追跡するため、古くなってしまうファイルがありません。

起動時にデーモンが死に、痕跡が残りません。 古い方法は、標準出力と標準エラー出力を `/dev/null` へ向けます。そのため、デーモンが独自のログファイルを開く前のエラーは消えてしまいます。監督する仕組みの下では、両ストリームをそのままにします。標準エラー出力へ書いたものはジャーナルに入り、`journalctl -u` でユニットごとに読めます。

一つの不具合を悪用されるだけで、マシン全体を渡してしまいます。 起動時に80番ポートが必要だったためrootで動くデーモンは、その後のすべてのリクエストでもrootのままです。サービスマネージャーに特権ソケットをバインドさせ、

デーモンへ渡させます。`daemon(7)` の推奨どおり、権限を落とすのも監督側へ任せます。`NoNewPrivileges=` や `ProtectSystem=` などの設定で、侵害されたプロセスが触れられる範囲を狭めます。

参考資料

- daemon(7) — 古典的な十五の手順と、新方式のデーモンが代わりに行うことを説明します。
- daemon(3) — Cライブラリーの簡略化した呼び出しと、そのマニュアル自身が記すセッションリーダーの問題を説明します。
- sd_notify(3) — デーモンが、準備完了、再読み込み中、停止中をsystemdへ知らせる方法です。
- systemd.socket(5) — マネージャーがポートを保持し、必要に応じてデーモンを起動する、ソケットによる起動を説明します。
- setsid — プロセスを端末から切り離す呼び出しについての、POSIXの定義です。
- 起動デーモンとエージェントの作成 — launchdジョブについてのAppleの規則です。forkして終了してはならない理由も説明します。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「デーモンとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/daemon/