Engineer

ライブラリー

ほかのプログラムから呼べるように、まとめたコード

どのようなものか

ライブラリーは、プログラムが、自分で書く代わりに使う、既製の処理を集めたものです。誰かが、日付の解析、ファイルの圧縮、QRコードの描画などの問題を一度解き、どのプログラムも使える、名前付きの呼び出し口を用意して公開します。英語の複数形はlibrariesです。LinuxのCライブラリーは、lib-という接頭辞を持つことが多く、Windowsは、共有する種類をDLLと呼びます。Rustではcrateといい、Rust Bookは、ライブラリーという言葉と置き換えて使うと説明します。大半の言語には、何も加えず、どのインストールにもあるモジュールをまとめた、標準ライブラリーも付属します。

ライブラリーは、再利用できるコードの一種です。近い概念と比べると分かりやすくなります。アプリケーションは、自分で動き、利用者を持ちます。ライブラリーには、そのどちらもなく、呼ぶプログラムの中だけで動きます。より近い比較対象は、フレームワークです。ライブラリーなら、利用側のコードが処理を主導し、必要なときに呼びます。フレームワークなら、処理の主導権はフレームワークにあり、それが定めた箇所で、利用側のコードが呼ばれます。最後に、パッケージは、名前、版、必要とするものの一覧を持つ、ライブラリーの配布用の箱です。ライブラリーは、その中のコードです。

ライブラリーを使えるものにするのは、そのインターフェースです。維持すると約束した関数、型、定数を指します。Semantic Versioningの規則全体は、この約束を土台にしています。従うプロジェクトは、公開APIを宣言しなければならず、そのAPIに互換性のない変更をするたび、メジャーバージョンを上げます。

Program Library HOWTOは、プログラムに加わる時点で、Linuxのライブラリーを分けます。静的ライブラリーは、ビルド時に実行ファイルへコピーされます。共有ライブラリーは、プログラムの起動時に読み込まれ、必要なすべてのプログ

ラムが使うため、各プログラムを再ビルドせずに更新できます。動的に読み込むライブラリーは、プログラムの実行中に開かれ、プラグインは、通常、この方式で動きます。

概念を判断に生かす

「ライブラリーとは？」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。 まず全体像を設計し、各部分のつながりと費用を、コードを書き始める前に明らかにします。

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。 14部門を支える共通インフラ計画 — 14 共通インフラ計画の対象となった研究部門

14部門を支える共通インフラ計画 お問い合わせ →

なぜ重要なのか

プロジェクトでライブラリーの利用を検討するたびに、選択肢は三つあります。既存のものを使うか、プロジェクト内でコードを書くか、小さく目的を絞った部分をリポジトリーへコピーし、その後は自分で責任を持つかです。最後を、コードの取り込み（vendoring）と呼びます。

通常は、既存のライブラリーを使うのが正解です。作成者は、境界事例に出会っており、ほかの利用者も、不具合を報告しています。代償は後から来るため、見落としやすいものです。ライブラリー自身の依存も加わり、そのセキュリティー情報は自分の問題となり、リリース予定に更新の速さを合わせ、ライセンスも自分の製品に入ります。保守担当者が放棄したライブラリーも、誰かが置き換えるまで、ビルドに残ります。

独自に書けば、制御できますが、ライブラリーが省いてくれたはずの、境界事例、テスト、保守を、期限なく引き受けます。問題の範囲が狭く、中核である場合、既存のライブラリーが必要以上に大きい場合、依存数を少なくする理由がある場合に、正しい選択です。

リンク方法も重要です。静的ライブラリーを使って作ったプログラムは、固定のコピーを持つため、ライブラリーのセキュリティー修正を届けるには、プログラムを再ビルドして、再配布しなければなりません。共有ライブラリーは、システム全体で一度修正できますが、プログラムは、システムが渡す版を信頼します。ライセンスでは、この違いは消えます。GNUのライセンスFAQは、GPLの対象となる著作物を、静的でも動的でも、ほかのモジュールとリンクすると、一体の著作物になり、GPLの条件が、その全体に及ぶと述べます。

ソフトウェアを発注する人が確かめるのは、成果物が、どのライブラリーに、どのライセンスで依存し、供給者が持ち込むライブラリーを誰が所有するかです。

実際の活用例

ある文書生成器には、各文書のオンライン版へ案内するQRコードが必要でした。既存のQRライブラリーなら、どれでも足りましたが、公開から長い時間が経っても、問題なくインストールできるよう、実行時の依存を、テンプレートエンジンとPDFレンダラーの、二つに絞っていました。解決策は、522行のQRエンコーダーを、リポジトリーに取り込むことでした。バイトモードの符号化、GF(256)上のReed-Solomon誤り訂正、位置検出、タイミング、位置合わせの各パターン、形式と版の情報、四つのペナルティー規則で採点する、八つすべてのマスクパターンを実装しました。別に書いたデコーダーが、毎回、生成したすべてのコードを読み戻し、復号結果が入力と一致することを確認します。ページは、主張を慎重に区切ります。対応するのは、その文書が使うモードと版だけです。理由も、成熟したライブラリーに勝つという考えでなく、依存数の上限です。

macOSのネイティブクライアントは、Cインターフェースを通して、中核のライブラリーを呼びます。初期は、そのインターフェースを、手書きのヘッダーで表していました。その後、中核から外れた三つの関数への呼び出しは、コンパイルもリンクも通り、実行時に異常終了しました。コンパイラーとリンカーは、手書きの代用品しか見ておらず、中核の実際の宣言を、一度も見ていなかったためです。修正は、すべての呼び出し箇所を検証するパーサーです。実際の7,308行のヘッダーと、その268の宣言を照合し、宣言のない呼び出しや列挙定数が、一つでもあれば、ビルドを失敗させます。調べるのは宣言なので、シグネチャーを保ったまま、意味を変えた関数は、通ります。

各案件の契約条件は、所有者を定めます。顧客のために書いたコードは、全額の支払い後、顧客に属します。会社が以前に書いたか、自社のために書いた、ライブラリーや汎用ツールは例外です。所有権は会社に残り、顧客には、納品物の中で使うための、期限がなく、取り消せないライセンスを与えます。第三者のオープンソースコードは、作成者が選んだライセンスに従い、成果物に入った部品を、顧客へ知らせます。

問題が起きるところ

通常の更新で、ビルドが壊れます

依存が2.3から2.4へ変わり、昨日コンパイルできたコードが失敗するか、違う動きをします。メジャーバージョンを上げずに公開インターフェースを変えたか、内部関数や文書化されていない動作など、約束していなかったものに、プロジェ

クトが依存したためです。ロックファイルで版を厳密に固定し、変更履歴を見ながら、意図して更新します。公開APIとして文書化されたものだけを呼びます。

ライブラリーの説明だけを、プログラムが信じます

ビルドに成功したプログラムが、最初に機能を使ったときに、異常終了します。手書きのヘッダー、型スタブ、バインディングを使ってコンパイルしましたが、それらが、実際にリンクしたライブラリーとずれていました。ライブラリー自身の公開インターフェースからバインディングを生成するか、ビルド時に、すべての呼び出しを照合します。不一致があれば、利用者が使う前の、その段階でビルドを失敗させます。

小さな機能が、大きな依存ツリーを連れてきます

一つの関数が必要だっただけなのに、提供するライブラリーには、何十もの依存が付いてきました。それぞれに、セキュリティー情報、ライセンス、更新があります。加える前に、ライブラリー自身の大きさだけでなく、何を引き込むかを調べます。必要な範囲が狭く、明確なら、自身のテストを備えた、小さな取り込み済みコードの方が、自前で管理する負担を小さくできる場合があります。

コードとともに、ライセンスが入ります

プロジェクトの終わりに近づいて、ソースコードを公開していない製品に含むライブラリーの条件が、一体になった著作物全体へ及ぶと、誰かが気付きます。追加時に誰もライセンスを確認せず、動的にリンクしても、何も変わりませんでした。追加する時点で、すべてのライブラリーのライセンスを確認し、公開する版とともに一覧を残し、供給者にも、その一覧を求めます。

参考資料

- Program Library HOWTO：導入 — Linuxのプログラムライブラリーとは何か、静的、共有、動的読み込みの違いです。
- The Rust Programming Language：パッケージとcrate — ライブラリーcrateとバイナリーcrate、パッケージが、それらを保持する仕組みです。
- Semantic Versioning 2.0.0 — ライブラリーの公開APIを、約束にする、版番号の規則です。
- Python標準ライブラリー — 標準ライブラリーの全体像です。どのPythonのインストールにも付属する、すべてを示します。
- GNUライセンスFAQ — 対象ライブラリーを、静的または動的にリンクするプログラムにとって、GPLが意味することです。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「ライブラリーとは？」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/library/