Engineer

Бібліотека

код, оформлений так, щоб його викликали інші програми

Що це таке

Бібліотека — це набір готових підпрограм, які програма позичає, замість того щоб писати їх сама: хтось один раз розв'язав задачу, скажімо розбір дат, стиснення файлів чи побудову QR-коду, і виклав розв'язок разом із набором іменованих точок входу, якими може скористатися будь-яка програма. Повна назва — програмна бібліотека, англійською library. На Linux бібліотеки мовою C часто мають у назвах префікс lib, у Windows спільну бібліотеку називають DLL, а в Rust кажуть «крейт» (crate), і книга The Rust Programming Language зазначає, що це слово вживають як синонім бібліотеки. Більшість мов постачається ще й зі стандартною бібліотекою — модулями, доступними в кожній інсталяції без жодних додаткових кроків.

Бібліотека — різновид коду для повторного використання, і найпростіше пояснити її через сусідів. Застосунок працює сам по собі й має користувача; бібліотека не має ні того, ні іншого і працює лише всередині програми, яка її викликає. Ближче до неї стоїть фреймворк. Із бібліотекою керує код самої програми: він звертається до бібліотеки, коли щось потрібно. Фреймворк керує сам і викликає код програми в місцях, які для цього відвів. Нарешті, пакет — це коробка, у якій бібліотеку постачають, з назвою, версією й переліком того, що їй потрібно, а бібліотека — це код усередині коробки.

Придатною до використання бібліотеку робить її інтерфейс: функції, типи й константи, які вона обіцяє зберігати. На цій обіцянці тримається вся схема Semantic Versioning. Проєкт, що її дотримується, мусить оголосити публічний API, а його мажорна версія зростає щоразу, коли цей API змінюється несумісно з попереднім.

Program Library HOWTO поділяє бібліотеки на Linux за тим, коли вони потрапляють у програму. Статичну бібліотеку копіюють у виконуваний файл під час збирання. Спільну завантажують під час запуску програми, і нею користуються всі програми, яким вона потрібна, тож її можна оновити, не перезбираючи їх. Динамічно завантажувану бібліотеку відкривають уже під час роботи програми — так зазвичай працюють системи плагінів.

Застосуйте поняття до рішення

Застосуйте «Що таке бібліотека?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Структура системи, спершу продумана на папері, щоб частини пасували одна до одної, а вартість була відома до першого рядка коду.

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. Один інфраструктурний план для 14 підрозділів — 14 дослідницьких підрозділів, охоплених спільним інфраструктурним планом

Один інфраструктурний план для 14 підрозділів Зв'язатися з нами →

Чому це важливо

Кожна бібліотека в проєкті — це рішення з трьома можливими відповідями: взяти наявну, написати код самим або скопіювати невеликий, чітко окреслений фрагмент у репозиторій і відтоді володіти ним. Останній варіант називають вендорингом (vendoring).

Взяти наявну бібліотеку зазвичай правильно. Її автори вже натрапили на крайові випадки, а інші користувачі вже повідомили про помилки. Ціна приходить пізніше, і її легко не помітити: разом із бібліотекою приходять її власні залежності, її бюлетені безпеки стають вашими, її графік випусків задає темп ваших оновлень, а її ліцензія переходить у ваш продукт. Бібліотека, яку автор покинув, лишається у збиранні, доки хтось її не замінить.

Власний код дає контроль і коштує всього, що бібліотека заощадила б: крайових випадків, тестів і супроводу, поки код живе. Це правильний вибір, коли задача вузька й центральна, коли наявні бібліотеки значно більші за потребу або коли проєкт має причину тримати кількість залежностей низькою.

Важить і спосіб компонування. Програма, зібрана зі статичною бібліотекою, несе в собі фіксовану копію, тож виправлення безпеки в бібліотеці дійде до користувачів лише тоді, коли програму перезберуть і випустять знову. Спільну бібліотеку можна виправити один раз для всієї системи, але тоді програма покладається на ту версію, яку надає система. Ліцензіям ця різниця байдужа: FAQ щодо ліцензій GNU прямо каже, що статичне чи динамічне компонування твору під GPL з іншими модулями створює комбінований твір, і умови GPL поширюються на всю комбінацію.

Для замовника програмного забезпечення практичні питання такі: від яких бібліотек залежить результат, під якими ліцензіями і кому належать бібліотеки, які постачальник приносить із собою.

Як це виглядає на практиці

Генератор документів мав ставити на кожен документ QR-код із посиланням на його онлайн-версію. Із цим упоралася б будь-яка з наявних QR-бібліотек, але проєкт свідомо обмежився двома залежностями під час виконання, шаблонізатором і рендерером PDF, щоб генератор і через багато років можна було встановити без пригод. Тож проєкт отримав власний QR-енкодер на 522 рядки: кодування в байтовому режимі, корекцію помилок кодом Ріда-Соломона над GF(256), пошукові, синхронізаційні й вирівнювальні візерунки, відомості про формат і версію та всі вісім масок, оцінених за чотирма штрафними правилами. Окремо написаний декодер під час кожного запуску зчитує кожен згенерований код і перевіряє, що в результаті виходять ті самі вхідні дані. Сторінка обережна у твердженнях: енкодер підтримує лише ті режими й версії, які потрібні документам, а виправдання йому — бюджет залежностей, а не віра в те, що він кращий за зрілу бібліотеку.

Нативний клієнт для macOS викликає бібліотеку-ядро через її інтерфейс мовою C, і спершу цей інтерфейс описував написаний вручну заголовний файл. Виклики трьох функцій, яких у ядрі вже не було, компілювалися, компонувалися й падали під час виконання, бо компілятор і компонувальник бачили лише рукописну заміну, а не справжні оголошення ядра. Виправленням став парсер, що перевіряє кожне місце виклику за справжнім заголовним файлом на 7 308 рядків і 268 оголошень: збирання зупиняється на будь-якому виклику чи константі переліку, яких цей файл не оголошує. Парсер перевіряє оголошення, тож функція, яка зберегла сигнатуру, але змінила семантику, проходить перевірку.

Умови співпраці прямо визначають, кому що належить. Код, написаний для замовника, після повної оплати стає його власністю. Винятком є бібліотеки й інструменти загального призначення, які компанія написала раніше або для себе: вони лишаються за компанією, а замовник отримує безстрокову ліцензію, яку не можна відкликати, щоб користуватися ними в межах переданої роботи. Сторонній відкритий код і далі підпорядковується ліцензіям, які обрали його автори, і замовникові повідомляють, які саме з цих компонентів потрапили в результат.

Де виникають проблеми

Звичайне оновлення ламає збирання

Залежність переходить із 2.3 на 2.4, і код, який учора компілювався, падає або поводиться інакше. Або бібліотека змінила публічний інтерфейс, не піднявши мажорної версії, або проєкт спирався на те, чого бібліотека ніколи не обіцяла, як-от внутрішню функцію чи незадокументовану поведінку. Фіксуйте точні версії в lock-файлі, оновлюйтеся свідомо, з відкритим журналом змін, і викликайте лише те, що бібліотека документує як публічне.

Програма довіряє опису бібліотеки

Збирання проходить без жодної помилки, а програма падає, щойно хтось уперше вмикає певну функцію. Код компілювали з використанням заголовних файлів, заглушок типів чи прив’язок, написаних вручну, і вони розійшлися з бібліотекою, яку насправді компонують. Генеруйте прив’язки з опублікованого інтерфейсу самої бібліотеки або перевіряйте кожен виклик за ним під час збирання, щоб розбіжність зупиняла збирання, а не проявлялася перед користувачем.

Одна дрібна функція тягне за собою ціле дерево

Потрібна була одна функція, а бібліотека, яка її надає, принесла десятки власних залежностей, і кожна з них — це ще бюлетені безпеки, ліцензії та оновлення. Перш ніж додавати бібліотеку, подивіться, що вона тягне за собою, а не лише на її власний розмір. Коли потреба вузька й добре описана, невеликий фрагмент коду, включеного в репозиторій, з окремими тестами може виявитися дешевшим у володінні.

Ліцензія приходить разом із кодом

Наприкінці проєкту хтось помічає, що бібліотека всередині закритого продукту має ліцензію, умови якої поширюються на весь комбінований твір. Ліцензій ніхто не перевіряв, коли бібліотеку додавали, а динамічне компонування нічого не змінило. Перевіряйте ліцензію кожної бібліотеки в момент додавання, зберігайте перелік разом із випуском і просіть постачальників надати свій.

Що почитати далі

- Program Library HOWTO: Introduction — що таке програмна бібліотека в Linux і чим різняться статичні, спільні та динамічно завантажувані бібліотеки.
- The Rust Programming Language: Packages and Crates — бібліотечні й бінарні крейти та те, як їх містить пакет.
- Semantic Versioning 2.0.0 — правила нумерації версій, що перетворюють публічний API бібліотеки на обіцянку.
- The Python Standard Library — стандартна бібліотека в повному обсязі: усе, що входить до кожної інсталяції Python.
- Frequently Asked Questions about the GNU Licenses — що означає GPL для програм, які компонують бібліотеку під цією ліцензією, статично чи динамічно.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке бібліотека?» 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/library/