Engineer

Bibliothèque

Code regroupé pour être appelé par d'autres programmes

Ce que c’est

Une bibliothèque est un ensemble de routines prêtes à l’emploi qu’un programme emprunte au lieu de les écrire : quelqu’un a résolu un problème, comme analyser des dates, compresser des fichiers ou dessiner un code QR, puis a publié la solution avec des points d’entrée nommés utilisables par tout programme. Le pluriel est bibliothèques. Les bibliothèques C sous Linux portent souvent le préfixe lib-, Windows appelle DLL les bibliothèques partagées, et les programmeurs Rust parlent de crate, mot que le livre Rust présente comme interchangeable avec bibliothèque. La plupart des langages fournissent aussi une bibliothèque standard, dont les modules sont disponibles dans chaque installation sans rien ajouter.

Une bibliothèque est du code réutilisable, plus facile à définir par comparaison avec ses voisins. Une application s’exécute seule et possède un utilisateur ; une bibliothèque n’a pas d’utilisateur propre et ne s’exécute qu’au sein d’un programme qui l’appelle. Le framework est le voisin le plus proche. Avec une bibliothèque, votre code dirige et l’appelle au besoin ; un framework dirige et appelle votre code aux points qu’il définit. Enfin, un paquet est la boîte dans laquelle une bibliothèque est distribuée, avec un nom, une version et la liste de ses besoins ; la bibliothèque est le code dans cette boîte.

Ce qui rend une bibliothèque utilisable est son interface : les fonctions, types et constantes qu’elle promet de conserver. Le versionnage sémantique fonde tout son système sur cette promesse. Un projet qui le suit doit déclarer une API publique et augmenter sa version majeure à chaque changement incompatible de cette API.

Le Program Library HOWTO classe les bibliothèques Linux selon le moment où elles rejoignent le programme. Une bibliothèque statique est copiée dans l'exécutable pendant sa construction. Une bibliothèque partagée est chargée au démarrage du programme et utilisée par tout programme qui en a besoin : elle peut donc être mise à jour sans les reconstruire. Une bibliothèque chargée dynamiquement est ouverte pendant l'exécution du programme ; les systèmes de plugins fonctionnent généralement ainsi.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'une bibliothèque ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Définir d'abord le système sur le papier pour que ses éléments s'articulent et que son coût soit connu avant d'écrire du code.

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. Un plan d'infrastructure pour 14 départements — 14 départements de recherche sur un même plan d'infrastructure

Un plan d'infrastructure pour 14 départements Contactez-nous →

Pourquoi c'est important

Chaque bibliothèque adoptée par un projet est une décision à trois réponses possibles : en prendre une existante, écrire le code en interne ou copier dans le dépôt un petit morceau ciblé pour en assumer ensuite la responsabilité, ce qu'on appelle vendoring.

Adopter une bibliothèque est généralement le bon choix. Ses auteurs ont rencontré les cas limites, et d'autres utilisateurs ont déjà signalé les bogues. Le coût arrive plus tard et passe facilement inaperçu : ses propres dépendances l'accompagnent, ses avis de sécurité deviennent votre affaire, son calendrier de publication rythme vos mises à jour et sa licence entre dans votre produit. Une bibliothèque abandonnée par son responsable reste incluse dans votre logiciel jusqu'à ce que quelqu'un la remplace.

Écrire le code vous-même apporte du contrôle et coûte tout ce que la bibliothèque aurait épargné : les cas limites, les tests et la maintenance, sans limite de durée. C'est le bon choix lorsque le problème est étroit et central, lorsque les bibliothèques disponibles dépassent largement le besoin ou lorsqu'un projet a une raison de limiter ses dépendances.

La façon dont une bibliothèque est liée compte aussi. Un programme construit avec une bibliothèque statique emporte une copie fixe : un correctif de sécurité de la bibliothèque ne parvient aux utilisateurs qu'après reconstruction et nouvelle livraison du programme. Une bibliothèque partagée peut être corrigée une seule fois pour tout le système, au prix de la confiance du programme dans la version fournie par celui-ci. Pour la GPL, la distinction ne change rien : la FAQ des licences GNU indique qu'une œuvre couverte par la GPL et liée à d'autres modules, statiquement ou dynamiquement, forme une œuvre combinée, et que les conditions de la GPL couvrent alors l'ensemble.

Pour qui commande un logiciel, les questions concrètes sont de savoir de quelles bibliothèques le résultat dépend, sous quelles licences et à qui appartiennent celles apportées par le fournisseur.

À quoi cela ressemble en pratique

Un générateur de documents avait besoin d'un code QR pointant vers la version en ligne de chaque document. Toute bibliothèque QR existante aurait convenu, mais le projet s'était volontairement limité à deux dépendances d'exécution, un moteur de modèles et un moteur de rendu PDF, pour continuer à s'installer proprement longtemps après sa sortie. La réponse a été d'intégrer au dépôt un encodeur QR de 522 lignes : encodage en mode octet, correction d'erreurs Reed-Solomon sur GF(256), motifs de repérage, de synchronisation et d'alignement, informations de format et de version, et les huit masques évalués selon les quatre règles de pénalité. Un décodeur écrit séparément relit chaque code généré à chaque exécution et vérifie qu'il restitue l'entrée. La page délimite soigneusement sa revendication : l'encodeur ne couvre que les modes et versions utilisés par ces documents, et sa raison d'être est le budget de dépendances, pas la conviction de surpasser une bibliothèque éprouvée.

Un client macOS natif appelle une bibliothèque centrale par son interface C, représentée au départ par un en-tête écrit à la main. Les appels à trois fonctions retirées depuis du cœur compilaient, étaient liés puis plantaient à l'exécution, parce que le compilateur et l'éditeur de liens n'avaient vu que le substitut écrit à la main, jamais les déclarations réelles du cœur. La correction a été un analyseur qui vérifie chaque site d'appel par rapport au véritable en-tête de 7 308 lignes et à ses 268 déclarations, faisant échouer la construction pour tout appel ou constante d'énumération non déclarés dans l'en-tête. Il vérifie les déclarations : une fonction qui conserve sa signature tout en changeant de sens réussit donc encore le contrôle.

Les conditions de chaque mission fixent qui possède quoi. Le code écrit pour un client devient le sien après paiement intégral. Les bibliothèques et outils génériques écrits auparavant par l'entreprise, ou pour ses propres besoins, font exception : elle en conserve la propriété et le client reçoit une licence pour les utiliser dans le travail livré, sans expiration et irrévocable. Le code open source tiers reste soumis à la licence choisie par ses auteurs, et le client est informé des composants inclus dans le résultat.

Où les problèmes surviennent

Une mise à jour ordinaire casse la construction

Une dépendance passe de 2.3 à 2.4, et le code qui compilait hier échoue ou se comporte autrement. Soit la bibliothèque a modifié son interface publique sans augmenter sa version majeure, soit le projet s'appuyait sur une promesse jamais faite, comme une fonction interne ou un comportement non documenté. Figez les versions précises dans un fichier de verrouillage, mettez à jour délibérément avec le journal des changements sous les yeux et n'appelez que ce que la bibliothèque documente comme public.

Le programme fait confiance à une description de la bibliothèque

Un programme construit sans erreur plante au premier usage d'une fonctionnalité. Le code a été compilé avec des en-têtes, des fichiers de types ou des liaisons écrits à la main, qui se sont écartés de la bibliothèque réellement liée. Générez les liaisons depuis l'interface publiée par la bibliothèque elle-même ou vérifiez chaque appel par rapport à elle pendant la construction : une divergence fera ainsi échouer la construction plutôt que de se manifester chez l'utilisateur.

Une petite fonctionnalité entraîne un grand arbre de dépendances

Il fallait une seule fonction, et la bibliothèque qui la fournit est arrivée avec des dizaines de dépendances, chacune apportant ses avis, licences et mises à jour. Avant d'ajouter une bibliothèque, examinez ce qu'elle entraîne, pas seulement sa taille. Pour un besoin étroit et bien spécifié, un petit morceau de code intégré au dépôt avec ses propres tests peut être moins coûteux à prendre en charge.

La licence arrive avec le code

Tard dans le projet, quelqu'un remarque qu'une bibliothèque d'un produit dont le code source n'est pas publié est soumise à une licence dont les conditions s'étendent à toute l'œuvre combinée. Personne n'a vérifié les licences à son ajout, et la liaison dynamique n'a rien changé. Vérifiez la licence de chaque bibliothèque au moment de l'ajouter, conservez la liste avec la version livrée et demandez aux fournisseurs leur propre liste.

Pour aller plus loin

- Program Library HOWTO : introduction — ce qu'est une bibliothèque de programmes sous Linux et les différences entre bibliothèques statiques, partagées et chargées dynamiquement.
- Le langage Rust : paquets et crates — crates bibliothèques ou binaires, et la façon dont un paquet les contient.
- Versionnage sémantique 2.0.0 — les règles de version qui transforment l'API publique d'une bibliothèque en promesse.
- La bibliothèque standard de Python — une bibliothèque standard dans toute son étendue : tout ce qui est fourni avec chaque installation de Python.
- Foire aux questions sur les licences GNU — ce que la GPL implique pour les programmes liés à une bibliothèque couverte, statiquement ou dynamiquement.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'une bibliothèque ? » 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/library/