Engineer Logiciels

Parseur vérifiant chaque appel FFI

Écrit un parseur lisant le véritable en-tête C de 7 308 lignes et vérifiant chaque site d'appel, chaque constante d'énumération et le caractère final de toute classe propriétaire de pointeurs, après qu'un en-tête de substitution manuscrit a laissé compiler et lier des appels à trois fonctions supprimées, puis provoqué un plantage.

septembre 2026

7,308

lignes d'en-tête C réellement lues, sans supposition

Situation. Au début du projet, l’interface C était représentée par un en-tête écrit à la main décrivant les fonctions attendues par l’application. Cet en-tête compilait, l’application était liée, et les appels à trois fonctions qui n’existaient plus dans le cœur arrivaient jusqu’à leur invocation puis plantaient. Le compilateur et l’éditeur de liens avaient tous deux été satisfaits par une description de la bibliothèque plutôt que par la bibliothèque elle-même, et l’écart n’apparaissait qu’à l’exécution.

Tâche. Le véritable en-tête — 7 308 lignes et 268 déclarations — devait devenir la référence, et chaque utilisation dans le code Swift devait être vérifiée automatiquement contre lui plutôt que par relecture.

Action. Le contrôle est un analyseur qui lit le véritable en-tête amont et construit l’ensemble des fonctions, constantes d’énumération et types qu’il déclare, puis lit le source Swift et résout chaque site d’appel et chaque référence de constante contre cet ensemble. Un appel à une fonction non déclarée par l’en-tête fait échouer la compilation. Une référence à une constante d’énumération renommée fait échouer la compilation. Le compte actuel est de 132 déclarations référencées sur les 268, et savoir quelles sont les 136 inutilisées est utile en soi, car cela dit exactement quelle part du cœur le client n’a pas encore atteinte. L’analyseur impose aussi une règle que le compilateur ne peut pas imposer : chaque classe Swift possédant un pointeur vers le cœur doit être

final. Une classe propriétaire de pointeur non final peut être sous-classée, et une sous-classe qui redéfinit la désinitialisation ou ajoute sa propre durée de vie change le moment où le pointeur est libéré — une utilisation après libération sans mot-clé unsafe à proximité. La règle est vérifiée par nom sur toute l'arborescence.

Résultat. La catégorie de défaut ayant produit le plantage initial ne peut plus se reproduire, car une référence périmée provoque désormais un échec de compilation plutôt que d'exécution. La limite est que l'analyseur comprend les déclarations de l'en-tête, pas leur sémantique : il prouve qu'une fonction existe sous un nom correspondant, et une fonction dont le sens ou la règle de propriété a changé en amont tout en gardant sa signature passe sans commentaire.

Mettez cette expérience au service de votre projet

Prenez « Parseur vérifiant chaque appel FFI » comme point de départ pour un défi similaire dans votre organisation. Les prestations mobilisées pour cette réalisation sont disponibles pour votre projet. La partie invisible pour les utilisateurs : où résident les données, quelles règles s'appliquent et comment les autres systèmes peuvent y accéder.

Découvrir nos prestations Contactez-nous →

Examinez ce qui a conduit au résultat

Découvrez la situation, les décisions et le résultat de cet exemple. Nous pouvons discuter des éléments transposables à vos contraintes et de ceux qui nécessitent une autre démarche. Parseur vérifiant chaque appel FFI — 7,308 lignes d'en-tête C réellement lues, sans supposition

Découvrir nos prestations Contactez-nous →

Votre compilation détecterait-elle l'appel d'une fonction supprimée ?

Discutez de votre projet avec l'ingénieure responsable de sa réalisation. Nous convenons par écrit du périmètre, de la base tarifaire et de la transmission avant le début des travaux.

Rôle exercé dans ce contexte : Ingénieure applications macOS. Cette réalisation fait appel à Ingénierie backend, API et intégration, Automatisation et CI/CD, Sécurité, Tests et assurance qualité et Documentation. Et puisque vous êtes ici, nous sommes aussi plutôt doués en Développement backend et API, DevOps et automatisation CI/CD et Documentation technique.

Contactez-nous Découvrir nos prestations

Un élément **d'un parcours plus long** —
entièrement présenté sur ce site.

Chaque entrée suit la même structure : la situation, la tâche, ce que nous avons fait et ce qui a changé.

Voir l'ensemble du parcours

Définissez l'amélioration nécessaire

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 →

Votre compilation détecterait-elle l'appel d'une fonction supprimée ?

Interrogez-nous sur les décisions qui sous-tendent « Parseur vérifiant chaque appel FFI ». Vous parlerez aux personnes capables d'expliquer le travail et d'évaluer votre propre projet.

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://software.engineer.company/fr/portfolio/wrote-a-parser-verifying-every-ffi-call/