Engineer Софтвер

Написала парсер, що перевіряє кожен виклик FFI

Написала парсер, який читає справжній C-заголовок на 7 308 рядків і перевіряє кожне місце виклику, кожну константу переліку й те, що кожен клас-власник вказівника є final, після того як через написаний вручну заголовок-заглушку виклики трьох вилучених функцій компілювалися, лінкувалися й аварійно завершувалися.

вересень 2026

7,308

рядків заголовка C — прочитано, а не вгадано

Ситуація. На початку проєкту C-інтерфейс описував написаний вручну заголовний файл з оголошеннями функцій, на які розраховував застосунок. Цей заголовний файл компілювався, застосунок компонувався, а виклики трьох функцій, яких у ядрі вже не існувало, доходили до моменту виклику й завершувалися аварійно. І компілятор, і компонувальник задовольнилися описом бібліотеки замість самої бібліотеки, тож розбіжність проявилася лише під час виконання.

Завдання. Справжній заголовний файл — 7 308 рядків і 268 оголошень — мав стати еталоном, а кожне використання оголошень із нього в коді Swift треба було звіряти з ним автоматично, а не вручну під час рев'ю.

Дія. Перевірка — це парсер, який читає справжній заголовний файл з основного репозиторію й будує множину функцій, констант переліків і типів, оголошених у ньому, а потім читає код Swift і звіряє з цією множиною кожне місце виклику та кожне посилання на константу. Виклик функції, якої немає в заголовному файлі, призводить до помилки збірки. Посилання на перейменовану константу переліку теж призводить до помилки збірки. Наразі використовуються 132 з 268 оголошень, і знати, які саме 136 не використовуються, корисно саме по собі: це точно показує, до якої частини ядра клієнт ще не дістався. Парсер також перевіряє правило,

якого не може забезпечити компілятор: кожен клас Swift, що володіє вказівником на дані ядра, має бути оголошений як final. Від класу без final, який володіє вказівником, можна успадкуватися, а підклас, що перевизначає деініціалізацію або додає власне керування часом життя, змінює момент звільнення вказівника — і виникає використання після звільнення, хоча поблизу немає жодного ключового слова unsafe. Правило перевіряють за назвами класів по всьому дереву коду.

Результат. Клас дефектів, через який стався початковий збій, більше не може повторитися, бо застаріле посилання тепер спричиняє помилку збірки, а не помилку під час виконання. Обмеження в тому, що парсер розуміє оголошення в заголовному файлі, але не їхню семантику: він доводить, що функція з відповідною назвою існує, а якщо в основному репозиторії змінилося призначення функції чи правило володіння, але сигнатура лишилася тією самою, перевірка пропустить її без зауважень.

Застосуйте цей досвід у своєму проєкті

Використайте «Написала парсер, що перевіряє кожен виклик FFI» як відправну точку для схожого завдання у вашій організації. Послуги, що стоять за цією роботою, доступні й для вашого проєкту. Та частина, якої користувач не бачить: де зберігаються дані, які діють правила і як інші системи можуть із нею взаємодіяти.

Переглянути наші послуги Зв'язатися з нами →

Дізнайтеся, що привело до результату

Прочитайте про ситуацію, рішення та результат цього прикладу. Ми можемо обговорити, що відповідає вашим умовам, а що потребує іншого підходу. Написала парсер, що перевіряє кожен виклик FFI — 7,308 рядків заголовка C — прочитано, а не вгадано

Переглянути наші послуги Зв'язатися з нами →

Чи спіймає ваша збірка виклик функції, якої вже немає?

Обговоріть свій проєкт з інженером, відповідальним за виконання. Обсяг, основу розрахунку вартості та передачу результатів узгоджуємо письмово до початку робіт.

Роль у цій роботі: Інженер застосунків macOS. Тут задіяні Backend-розробка, API та інтеграції, Автоматизація та CI/CD, Безпека, Тестування та QA та Документація. І, до речі, ми добре вміємо Backend- та API-розробка, DevOps та автоматизація CI/CD та Технічна документація.

Зв'язатися з нами Ознайомтеся з нашими послугами

Одна частина **довшого переліку** — і він увесь на цьому сайті.

Кожен запис написано однаково: ситуація, завдання, що ми зробили і що змінилося.

Переглянути весь перелік

Визначте потрібне покращення

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 →

Чи спіймає ваша збірка виклик функції, якої вже немає?

Запитайте про рішення в роботі «Написала парсер, що перевіряє кожен виклик FFI». Ви говоритимете з людьми, які можуть пояснити роботу й оцінити ваш проєкт.

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