Engineer Software

Analizador que verifica cada llamada FFI

Escribí un analizador que lee la cabecera C real de 7.308 líneas y verifica cada punto de llamada, cada constante de enumeración y que toda clase propietaria de un puntero sea final, después de que una cabecera provisional escrita a mano permitiera compilar y enlazar llamadas a tres funciones eliminadas que causaban fallos al ejecutarse.

septiembre 2026

7,308

líneas de cabecera C leídas en
vez de darse por supuestas

Situación. Al principio del proyecto, la interfaz C se representaba mediante una cabecera escrita a mano que describía las funciones que la aplicación esperaba. Esa cabecera compilaba, la aplicación enlazaba y las llamadas a tres funciones que ya no existían en el núcleo llegaban a ejecutarse y provocaban fallos abruptos. Tanto el compilador como el enlazador habían quedado satisfechos con una descripción de la biblioteca en lugar de con la biblioteca, y la diferencia solo aparecía en tiempo de ejecución.

Tarea. La cabecera real — 7.308 líneas y 268 declaraciones — tenía que convertirse en la autoridad, y cada uso de ella en el código Swift debía verificarse automáticamente frente a ella, no mediante revisión.

Acción. La comprobación es un analizador que lee la cabecera real del proyecto de origen y construye el conjunto de funciones, constantes de enumeración y tipos que declara; después lee el código fuente de Swift y resuelve cada punto de llamada y cada referencia a una constante contra ese conjunto. Una llamada a una función que la cabecera no declara hace fallar la compilación. Una referencia a una constante de enumeración cuyo nombre ha cambiado hace fallar la compilación. El recuento actual es de 132 de las 268 declaraciones referenciadas, y saber cuáles son las 136 no utilizadas es útil por sí mismo, porque indica exactamente cuánto del núcleo no ha alcanzado el cliente.

El analizador también impone una regla que el compilador no puede: cada clase de Swift propietaria de un puntero al núcleo debe ser final. Una clase no final propietaria de un puntero permite subclases, y una subclase que sobrescriba la desinicialización o añada su propio ciclo de vida cambia cuándo se libera el puntero: un uso después de liberar sin ninguna palabra clave unsafe cerca. La regla se comprueba por nombre en todo el árbol.

Resultado. La clase de defecto que produjo el fallo original no puede repetirse, porque una referencia obsoleta es ahora un fallo de compilación y no de ejecución. La limitación es que el analizador entiende las declaraciones de la cabecera y no su semántica: demuestra que una función existe con un nombre coincidente, y una función cuyo significado o regla de propiedad haya cambiado en origen conservando la firma pasa sin comentarios.

Aplique esta experiencia a su proyecto

Utilice «Analizador que verifica cada llamada FFI» como punto de partida para un reto similar en su organización. Los servicios que respaldan este trabajo están disponibles para su proyecto. La parte que el usuario no ve: dónde se guardan los datos, qué reglas se aplican y cómo pueden comunicarse otros sistemas con ella.

Explore nuestros servicios Póngase en contacto →

Mire más allá del resultado

Lea la situación, las decisiones y el resultado de este ejemplo. Podemos hablar de qué partes se adaptan a sus restricciones y cuáles necesitan otro enfoque. Analizador que verifica cada llamada FFI — 7,308 líneas de cabecera C leídas en vez de darse por supuestas

Explore nuestros servicios Póngase en contacto →

¿Detectaría su compilación una llamada a una función eliminada?

Converse sobre su proyecto con la ingeniera responsable de la entrega. Acordamos por escrito el alcance, la base del precio y el traspaso antes de empezar a trabajar.

Contexto de la función: Ingeniera de aplicaciones macOS. Se apoya en Ingeniería de backend, API e integración, Automatización y CI/CD, Seguridad, Pruebas y control de calidad y Documentación. Y ya que está aquí, también se nos da bien Desarrollo de backend y API, DevOps y automatización CI/CD y Documentación técnica.

Póngase en contacto Explore nuestros servicios

Una parte de **una trayectoria más amplia** — y toda ella está en este sitio.

Cada entrada sigue la misma estructura: la situación, la tarea, lo que hicimos y lo que cambió.

Vea la trayectoria completa

Defina la mejora que necesita

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

¿Detectaría su compilación una llamada a una función eliminada?

Pregunte por las decisiones detrás de «Analizador que verifica cada llamada FFI». Hablará con las personas que pueden explicar el trabajo y valorar su propio proyecto.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://software.engineer.company/es/portfolio/wrote-a-parser-verifying-every-ffi-call/