David Greni

Report 2

CS 332

Inlining Optimization

Inlining is a compiler optimization that takes a function that would normally be a function call and places that code into the function that is calling it. An example being in a programs main function a call to a function named foo would normally be compiled as a separate function and require a separate stack frame, arguments, and jumps that would have to be undone after that function has ended [1]. This is ultimately slow and creates issues with other compiler optimization techniques. In contrast when the compiler inlines the code it essentially brings the function into main and runs the code as if it was written in main. This speeds up the program but more importantly it allows other compilation optimization techniques to be applied to foo. Inlining should be used when a function is relatively small and used several times in a program. There are also many instances where functions cannot be inlined, such as when a function contains a loop or is recursive, the function uses a return statement with no return value, or when the function contains any switch or go to statements[2]. Overall inlining is a very useful function because it speeds up a program and allows for other compiler optimizations.

Reference:

[1] *Inlining and compiler optimizations*. Inlining and Compiler Optimizations. (2021, January 23). <https://wolchok.org/posts/inlining-and-compiler-optimizations/>

[2] *Inlining and compiler optimizations*. Inlining and Compiler Optimizations. (2021, January 23). https://wolchok.org/posts/inlining-and-compiler-optimizations/