Avoiding ABI Breaks

Sutter's Mill

Herb Sutter on software development

My little New Year's Week project (and maybe one for you?)

Herb Sutter 2025-01-02 7 Minutes

[Updates: Clarified that an intrusive discriminator would be far beyond what most people mean by "C++ ABI break." Mentioned unique addresses and common initial sequences. Added "unknown" state for passing to opaque functions.]

Here is my little New Year's Week project: Trying to write a small library to enable compiler support for automatic raw `union` member access checking.

The problem, and what's needed

During 2024, I started thinking: **What would it take to make C/C++ `union` accesses type-checked?** Obviously, the ideal is to change naked `union` types to something safe.**(*)** But because it will take time and effort for the world to adopt any solution that requires making source code changes, I wondered how much of the safety we might be able to get, at what overhead cost, just by recompiling existing code in a way that instruments ordinary `union` objects?

Search ...

Follow by email

Email Address

Subscribe

I'm an author and speaker, and a programming language nerd whose focus is on enabling our program code to be both clean *and* fast. I've been writing about programming since 1993, usually about C++ or about concurrency and parallelism. I'm the designer

Avoiding ABI Breaks: Extrinsic Storage

```
//  That's it. Here's an example:
//    {
//      union Test { int a; double b; };
//      Test t = {42};                          union_registry<>::on_set_alternative(&u,0);
//      std::cout << t.a;                       union_registry<>::on_get_alternative(&u,0);
//      t.b = 3.14159;                          union_registry<>::on_set_alternative(&u,1);
//      std::cout << t.b;                       union_registry<>::on_get_alternative(&u,1);
//    }                                         union_registry<>::on_destroy(&u);
//
```