

Halt and Catch Fire: formatting your hard drive with UB, Clang
and modest for loops

Adam Galiński

C++ **FFFE**, May 2021

Today's talk is inspired by this tweet



StarBrilliant @m13253 · Mar 16

...

With Clang 13, you can literally format your hard drive, using C++ undefined behavior.

godbolt.org/z/1q1bjn

The screenshot shows a Godbolt compiler explorer interface. On the left, the C++ source code is displayed with syntax highlighting. It includes `<stdio.h>`, defines `f1()` and `f2()` functions, and a `main` function that calls them. The code is annotated with comments like `// Underlined behavior`. On the right, the assembly output for Clang 13.0.0 is shown. The assembly includes instructions like `push rax`, `mov edi, offset .L.str`, `call puts`, and `ret`. It also shows the generation of a string literal `"formatting /dev/sda1..."` and the call to `puts` for the string `.L.str:1`.



28



337



962



<https://twitter.com/m13253/status/1371615680068526081>

<https://godbolt.org/z/1q1bjn>

So what is an undefined behavior, anyway?

undefined behavior - there are *no restrictions* on the behavior of the program. Examples of undefined behavior are data races, memory accesses outside of array bounds, *signed integer overflow*, null pointer dereference, more than one modifications of the same scalar in an expression: without any intermediate sequence point (until C++11) / that are unsequenced (since C++11), access to an object through a pointer of a different type, etc. Compilers are not required to diagnose undefined behavior (although many simple situations are diagnosed), and the compiled program is *not required to do anything meaningful*.

Source: <https://en.cppreference.com/w/cpp/language/ub>, emphasis mine.

<https://godbolt.org/z/jKh9GaMna>

undefined behavior - there are *no restrictions* on the behavior of the program.

- the compiler *really* is allowed to emit *any assembly*...
 - ...one that just crashes or *ceases to function*
 - ...one that looks good, but has Jumps without corresponding RETs
 - ...one that corrupts memory of another program, but only on friday evenings
- ...and *you, the programmer* are to blame.

Key takeaways

- be aware of the common root causes of UB
- use `volatile` judiciously
- use sanitizers ([UBSan](#)) and other fuzzing tools (e.g. [Valgrind+memcheck](#)) to detect the triggering of UB
- prefer modern C++ syntax over old C-style syntax — lambdas over naked pointers to functions.

Thank you!