A UB AND IFNDR ANNEX FOR C++

Shafik Yaghmour

Frontend Compiler Developer at Intel

The Problem

- The C++ Standard has a lot of Undefined Behavior and IFNDR
- They are hard to identify
  - The behavior of the program is undefined
  - Undefined behavior
  - The behavior is undefined
  - The behavior … is undefined
  - The effect of … is undefined
  - The result … is undefined
  - Has runtime-undefined behavior
- They are often hard to understand. Usually no examples. You are on your own.
- Sometimes UB is left implicit.

The Problem continued

- We could edit the standard and unify the wording across the whole document but…
- We normally don’t treat the standard as a tutorial so we still don’t have a good place for all of the examples
- We would not have all the UB and IFNDR cases in one place
- The solution for this is an Annex

What is an Annex?

- ISO/IEC Directives Part 2 says
  - Annexes are used to provide additional information to the main body of the document and are developed for several reasons, for example:
    - when the information or table is very long and including it in the main body of the document would distract the user;
    - set apart special types of information (e.g. software, example forms, results of interlaboratory tests, alternative test methods, tables, lists, data);
    - to present information regarding a particular application of the document.
  - They can be normative or Informative
    - Normative annexes provide additional normative text to the main body of the document.
    - Informative annexes provide additional information intended to assist the understanding or use of the document.

What are the Goals?

- Create an informative Annex for UB and an informative Annex for IFNDR with plain English explanations and code examples.
- To target
  - Developers and trainers
  - Tools developers
  - Security analysts
- Document all explicit UB
  - Documenting implicit UB could be error prone. We need to specify it.
- Adding/Removing UB has to be part of WG21 process for proposals

Why is it Important for C++?

- We want a “safer” language.
- UB is a trap for the unaware developer.
- Even “experts” can’t easily identify UB.
- A common set of examples to make UB more concrete.
- If we don’t document it, we don’t know if it is getting better or not.

Why is it Important for Clang?

- Our users want a “safer” language.
- We can communicate about UB to a users more effectively.
- It gives us a list of targets for: tools, diagnostics and extensions.
- We can move faster to implement solutions:
  - We already have “Safe” extensions e.g.
    - -fwrapv
    - -fno-strict-aliasing
    - -D_LIBCPP_HARDENING_MODE
    - RFC for Hardening mode
      - https://discourse.llvm.org/t/rfc-hardening-mode-for-the-compiler/87660

The Process

- Write a proposal(s) to create a UB and IFNDR Annex
  - http://wg21.link/p1705
- Present proposal to Evolution and Core Working groups
- Refine proposal and maybe write follow-up proposals
  - http://wg21.link/p3075
- Proposal gets accepted
- Iterate over Annex wording (We are here!)
  - https://github.com/cplusplus/draft/pulls?q=is%3Apr+is%3Aopen+label%3Aub-ifndr
- Final acceptance of Annex wording

Call To Action

- Annex could use more editors!
- Help identify implicit UB and write Defect Reports or even better papers
  - https://github.com/cplusplus/CWG or core reflector
- Let’s implement features in clang to diagnose/specify/eliminate UB