CLANGD: LSP USING CLANG

MARC-ANDRÉ LAPERLE, ERICSSON
AGENDA/TOPICS

1. Introductions
2. Goals and scope of Clangd
3. Existing language server implementations
4. Challenges
5. Proposed architecture
6. Collaborations and planning
INTRODUCTIONS

› Marc-André Laperle
  − Software Developer at Ericsson since 2013
  − Eclipse committer for CDT (C/C++) and several other projects.
  − New LLVM/Clang contributor
  − Enthusiastic about C/C++, IDEs, and tooling in general (Not a compiler expert!)

Your turn!
GOALS AND SCOPE

› Tool in Clang “Extras”
› Implements the Language Server Protocol
› Should it offer other services??
› Compiling and linking?
EXISTING IMPLEMENTATIONS

› C/C++ for Visual Studio Code (Microsoft). Not open source.
› C/C++ Clang Command Adapter (Yasuaki MITANI, Github). Parses Clang command output.
› Others?
CHALLENGES

› Refactoring and code generation
› Speed?
› Persisted database/index (Find references, Go to Definition, Call Hierarchy, etc)
ARCHITECTURE

› Should other clang-tools be invoked directly? Clang-format, clang-tify, clang-rename, etc.
  - Can they all be used as libraries?
PLANNING

› Persisted database
› Use a JSon library (jsoncpp?)
› Improvements to code completion
› Open Declaration/Definition
› Find references (functions, classes, fields, variables, with read/write information)
› Call hierarchy. Callers and callees of a specific function or field.
› Type hierarchy
PLANNING

› Formatting (all done?)
› Syntax/Semantic highlighting
› Source hover
› Code Lens
› Signature Help
› Code folding
› Organize includes
PLANNING

› Implement Method (Source generation)
› Generate Getters and Setters (Source Generation)
› Rename (Refactoring)
› Extract Local Variable (Refactoring)
› Extract Function (Refactoring)
› Hide Method (Refactoring)
› Quick Assits (local code transformations)