Open *Source* | Open *Possibilities*

lld, a linker framework

Presented by: Shankar Easwaran
Qualcomm Innovation Center, Inc

QuIC
QUALCOMM INNOVATION CENTER, INC.

Motivation

- Provide a necessary set of reusable libraries to create target specific Linkers.
- Clang will not have to rely on the system linker on each OS.
- Increase in optimization opportunities.
- Provide necessary tools to diagnose linker behavior.

Terms Used

- The term ***"flavor"*** is associated with different form of command line styles that lld supports.
  - GNU, Darwin(ld64), Windows(Link) are examples of supported flavors.
- ***File Formats*** specifies types of inputs/outputs supported by lld
  - ELF, Mach-O, COFF, YAML, IR are examples of supported file formats.
- The term ***"target"*** is used as a way to support a particular architecture/ABI in the lld linker.
  - Often associated with a flavor and a file format.

Agenda

- lld features
- Atom Model
- Driver Model

- Lots of details are not in the presentation in the interest of time.

lld Features

- A modular design
- A set of re-usable libraries that can be used to construct a linker
- There are Readers/Writers to support each file format
  - ELF, PE/COFF, Mach-O are examples
- Linking is performed on lld IR which is an internal and format-independent representation.
- lld is structured around a PassManager that runs passes over the IR
- The IR is in-memory but can be translated to/from textual (YAML) or binary files aka, Native Files.
- Features a number of tests, testing linker behavior
- lld is also asan/tsan clean!
- Provide a framework to perform experiments!

lld – Atom Model

- The basic model which achieves format-independent linking.
- The Reader associated with each Input file format converts Input files into Atoms.
- Atoms contain generic attributes and target specific attributes
- Generic attributes
  - Name
  - Scope
  - Ordinal
- Target specific attributes
  - Type
  - Relocations
  - Atom Content
- Readers can control the layout of atoms
  - Layout-After, Layout-Before references

lld – Driver Model

- The Driver models each flavor.
- Uses a TableGen based parser to parse command line arguments.
- The Driver creates a
  - LinkingContext
    - Encapsulates “what and how” to link
    - Readers/Writers subclass the LinkingContext to contain additional options
  - InputGraph
    - Converts command line arguments to a Graph.
    - Nodes in the InputGraph represented as InputElements.
    - InputElements are associates with either FileNodes/ControlNodes.
      - » FileNodes represent Input Files and its associated command line attributes
      - » ControlNodes represent a way to control Linker behavior
    - The InputGraph also controls how input files are resolved and typically controlled by each Flavor.
    - Contains various utility / diagnostic functions to control/dump information.

Final Notes

- Extend LLVM with a Linker framework!
- Current Status
  - ~300 tests across Gnu, Darwin, Link flavors
    - Tested on buildbots too.
  - Address sanitizer and Thread sanitizer Clean!
  - Supports a lot of options on each flavor
  - Highly integrated with LLVM
  - Supports a wide range of options with Gnu/Darwin/Link flavors
  - Yes, a linker that permits you to do a lot of experiments!
  - More status in the next page … ☺
- Future
  - Debug executables linked with lld across all flavors.
  - Add dynamic linking support on all flavors
  - Add LTO support
  - Update documentation at lld.llvm.org, really behind ☺

More Status ….

- With GNU flavor
  - Link lld using itself (self-host) and run tests
    - 55% fails (153 failures, 126 passes)
  - Link clang using lld and run tests
    - 14% fail (935 failures, 5756 passes)
  - There might be a lot of assumptions with the current GNU linkers which needs to be still understood ☺

- Credits to lld team on making this happen!
- Patches Welcome ☺
- Thank you!
- Questions …. ??

QuIC
QUALCOMM INNOVATION CENTER, INC.