The LLVM compiler framework

Writing a pass: Quick Start

Daniele Cattaneo

Politecnico di Milano

2024-04-03

These slides were originally written by Stefano Cherubin for the “Code Transformation and Optimization” course.

Contents

Understanding LLVM

LLVM is **not** a compiler.

Understanding LLVM

LLVM is **not** a compiler.

LLVM is a
collection of components
which is **useful**
to build a compiler.

Getting LLVM

All work on LLVM goes to the **monorepo** on GitHub

- It contains LLVM + major subprojects handled by the LLVM project
- `git clone -b release/10.x --single-branch git@github.com:llvm/llvm-project.git`

A few years ago they used a private SVN repository, the switch to GitHub is recent

What LLVM is made of

- C++ libraries
  - `llvm/include/llvm/...`
  - `llvm/lib/...`
- small application (tools)
  - `llvm/tools/...`
  - `llvm/utils/...`

Binaries installed under `bin/...`

Commands

llvm-as	LLVM assembler
llvm-dis	LLVM disassembler
opt	LLVM optimizer
llc	LLVM static compiler
lli	directly execute programs from LLVM bitcode
llvm-link	LLVM bitcode linker
llvm-mca	LLVM machine code analyzer
llvm-nm	list LLVM bitcode and object file's symbol table
llvm-stress	generate random .ll files
llvm-config	prints out install configuration parameters
llvm-dwarfdump	print contents of DWARF sections

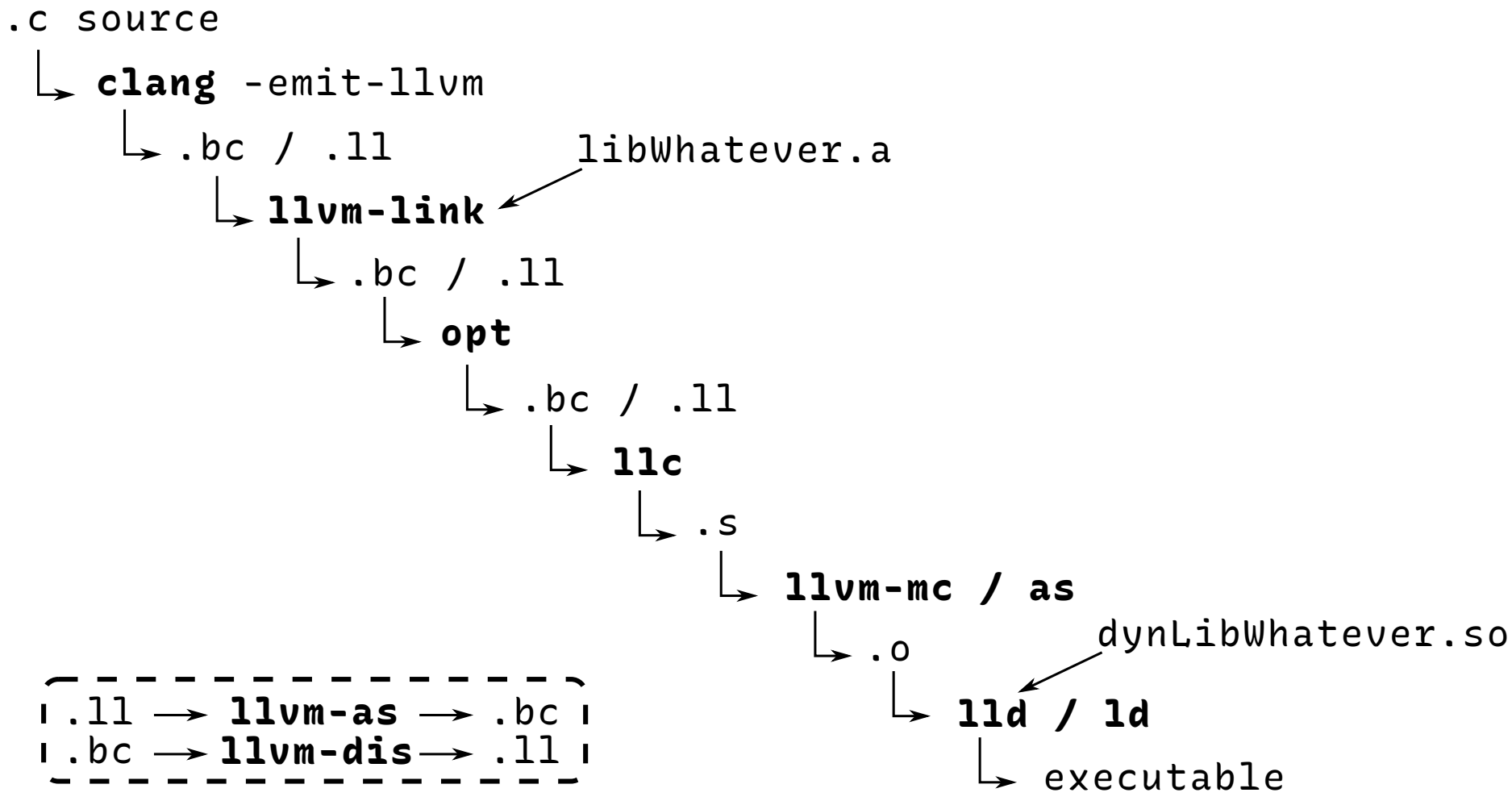
For a complete reference, see the LLVM command guide*

---

*`http://llvm.org/docs/CommandGuide/index.html`

Contents

Simulating a LLVM driver manually



Writing a LLVM pass

There are a lot of tutorials available:

- Official developer guide
  `llvm.org/docs/WritingAnLLVMPass`

- Out-of-source pass
  `github.com/quarkslab/llvm-dev-meeting-tutorial-2015`

We will follow the first one, with a few adjustments.

Building LLVM

To test your pass you need a **Debug+Assertions** build of LLVM.

This build needs to be **kept separated** from normal Release builds
(it's very slow!)

The best way to get such a LLVM build is to **make it yourself**!

Building LLVM

- Detailed instructions:
  `https://llvm.org/docs/GettingStarted.html`

Problem 1 With the **default options**, a finished build takes **25 GB of disk space**

Problem 2 A standard build with the GNU toolchain uses **a lot of RAM** (≈16 GB or more with a modern 4 core CPU!) especially when linking

We need to customize the build process a bit...

Building LLVM

- The build flags I like to use:

```
-GNinja
-DLLVM_ENABLE_PROJECTS='clang'
-DLLVM_INSTALL_UTILS=ON
-DLLVM_BUILD_LLVM_DYLIB=ON
-DLLVM_LINK_LLVM_DYLIB=ON
-DLLVM_OPTIMIZED_TABLEGEN=ON
-DLLVM_INCLUDE_EXAMPLES=OFF
-DCMAKE_INSTALL_PREFIX=/opt/llvm-10.0-d
-DLLVM_USE_LINKER=lld
-DCMAKE_C_COMPILER=clang-10
-DCMAKE_CXX_COMPILER=clang++-10
```

- Building with LLVM itself solves the RAM usage problem!
  - Not required on macOS or *BSD: they already ship LLVM as default
- Using **shared libraries** drops the disk usage to **10 GB**.

  The build products alone will still take 20 GB of disk space...

Last notes on building

You can add other projects to the LLVM build by modifying the value of the `LLVM_ENABLE_PROJECTS` flag

Good practice: **always include clang**

- You can easily see on a production quality compiler the impact of changes you made on your local copy of LLVM

To install cutting-edge release LLVM if your Linux distribution does not provide it:

- `https://apt.llvm.org`

Testing

LLVM has an internal testing infrastructure*. Please use it.

llvm-lit LLVM Integrated Tester

1. Forge a proper LLVM-IR input file (.ll) for your test case
2. Instrument it with `lit` script comments
3. Run `lit` on your test
   - `llvm-lit /llvm/test/myTests/singleTest.ll`
     run a single test
   - `llvm-lit /llvm/test/myTests`
     run the test suite (folder)
4. Run `lit` on the LLVM test suite (regression testing)

To submit a bug report to LLVM developers you will be asked to write a `lit` test case that highlights the bug.

---

*`http://llvm.org/docs/TestingGuide.html`

Thank You!

Questions?