bazelbuild / **bazel**

About the CROSSTOOL

Edit New Page

Marcel Hlopko edited this page on 12 Apr 2017 · 2 revisions

In order to be able to invoke the compiler with the right options, Bazel needs some knowledge about the compiler internals, such as include directories and important flags. In effect, Bazel requires a simplified model of the compiler. Parts of that model are necessarily still hard-coded into Bazel. Most importantly, Bazel expects the compiler toolchain to be based on (or at least be very similar to) GCC, and have most of the associated tools, like ld, ar, and objcopy.

The external information about the compiler is encoded into an ASCII protobuf, which is read by Bazel whenever a build operation is requested. The file must be named CROSSTOOL and must be present in the top-level toolchain directory (the value given to the `--crosstool_top` flag). At present, if the file is not available, Bazel falls back to its previous hard-coded implementation.

What does Bazel need to know?

Bazel needs to know:

- which platform the compiler runs on
- which platform the compiler compiles for
- whether it supports the gold linker, thin archives, and a special ar flag for archive normalization
- whether the target platform needs PIC (position independent code)
- the paths to the different tools, like gcc, ld, ar, objcopy, etc.
- the built-in system include directories - this is especially important if the compiler is supposed to run on a different machine

Pages 9

Wiki

- Who is using Bazel?
- Project Ideas

Website

- Docs
- Contribute

Connect

- Blog
- Stack Overflow
- Twitter

Clone this wiki locally

https://github.com/bazelb

- the default sysroot
- gcc include directories, if available
- the correct flags to use for compilation, linking, and object embedding
- the correct flags to use for the different compilation modes supported by Bazel (opt, dbg, fastbuild)
- the correct flags to use for the different linking modes (fully static, mostly static, and dynamic)
- additional make variables that should be available when this compiler is used, especially CC_FLAGS
- if the compiler has support for multiple architectures, it needs to know all these things for all supported architectures

Crosstool configuration file format

As the file format, we have decided to use the ASCII protobuf format. This has the advantage of being human-readable, so ad-hoc changes to the file can be made to test what effect certain changes have. The definition is stored in src/main/protobuf/crosstool_config.proto, and contains some documentation for each of the fields.

Create a configuration file

The easiest way to create a file is to use an existing configuration and adapt it to the new compiler.