Build Distribution for Maintaining the Famous GCC 4.7

Oliver Reiche <oliver.reiche@huawei.com>
Intelligent Cloud Technologies Lab, Huawei Munich Research Center
www.huaweicloud.com

FOSDEM 2024

February 4, 2024

Agenda

Famous GCC 4.7

“Build Distribution”

Patching GCC 4.7.4

Bootstrap Process

Wrap-up

Famous GCC 4.7

Famous GCC 4.7

The Bootstrappable Builds movement strives for

- building all software from source
- with only a minimal *binary seed*

But how to build a C++ compiler without a C++ compiler?

Why is GCC 4.7 famous?

It plays a key role for bootstrappable builds:

- last GCC that can be build with only a C compiler
- stepping stone for C++ and beyond (Rust, Java, etc.)

Software preservation

GCC 4.7 is from 2012–2014

- build on modern systems (glibc, musl)
- build with modern compilers
- build with modern C11 standard
- build reproducibly

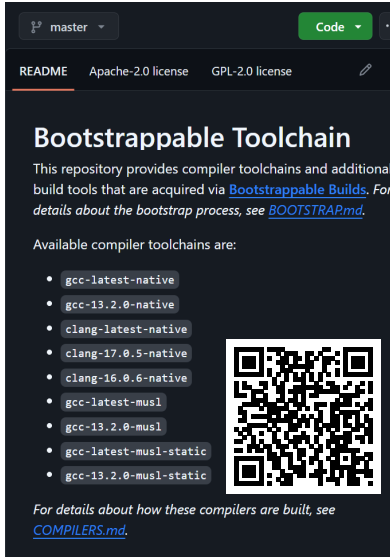
“Build Distribution”

"Build Distribution"

Project: Bootstrappable Toolchain

Using the open source build system JustBuild

- bootstraps latest compilers (GCC13/Clang17)
- bootstraps latest build tools (make/cmake/python)
- requires only a reduced binary seed
  - Coreutils
  - POSIX shell
  - C compiler (e.g., TinyCC)
- all toolchains are built from source
- uses existing build descriptions (make/cmake)
- on-demand tool bootstrap (minimal Linux distribution)
  ⇝ aka the *"Build Distribution"* for toolchains



Patching GCC 4.7.4

Patching GCC 4.7.4

Maintenance patches and backports

- building with C11 [Mike Frysinger]
- musl support in `config.sub` [4.8.0]
- **general musl support [6.1.0]**
- musl linker support [6.1.0]
- libstdc++ support for musl [6.1.0]
- use new type name `context_t` [8.1.0]
- musl detection in `config.guess` [9.1.0,11.1.0,11.1.0]

```
--- a/gcc/config/linux.h
+++ b/gcc/config/linux.h
@@ -33,10 +33,12 @@ see the files COPYING3 and COPYING.
 #define OPTION_GLIBC  (DEFAULT_LIBC == LIBC_GLIBC)
 #define OPTION_UCLIBC (DEFAULT_LIBC == LIBC_UCLIBC)
 #define OPTION_BIONIC (DEFAULT_LIBC == LIBC_BIONIC)
+#define OPTION_MUSL   (DEFAULT_LIBC == LIBC_MUSL)
 #else
 #define OPTION_GLIBC  (linux_libc == LIBC_GLIBC)
 #define OPTION_UCLIBC (linux_libc == LIBC_UCLIBC)
 #define OPTION_BIONIC (linux_libc == LIBC_BIONIC)
+#define OPTION_MUSL   (linux_libc == LIBC_MUSL)
 #endif
```

Patching GCC 4.7.4

Maintenance patches and backports

- building with C11 [Mike Frysinger]
- musl support in `config.sub` [4.8.0]
- general musl support [6.1.0]
- **musl linker support [6.1.0]**
- libstdc++ support for musl [6.1.0]
- use new type name `context_t` [8.1.0]
- musl detection in `config.guess` [9.1.0,11.1.0,11.1.0]

```
--- a/gcc/config/i386/linux64.h
+++ b/gcc/config/i386/linux64.h
@@ -31,3 +31,10 @@ see the files COPYING3 and COPYING.R
 #define GLIBC_DYNAMIC_LINKER32 "/lib/ld-linux.so.2"
 #define GLIBC_DYNAMIC_LINKER64 "/lib64/ld-linux-x86-64
 #define GLIBC_DYNAMIC_LINKERX32 "/libx32/ld-linux-x32.
+
+#undef MUSL_DYNAMIC_LINKER32
+#define MUSL_DYNAMIC_LINKER32 "/lib/ld-musl-i386.so.1"
+#undef MUSL_DYNAMIC_LINKER64
+#define MUSL_DYNAMIC_LINKER64 "/lib/ld-musl-x86_64.so.
+#undef MUSL_DYNAMIC_LINKERX32
+#define MUSL_DYNAMIC_LINKERX32 "/lib/ld-musl-x32.so.1"
```

Patching Reproducibility of GCC 4.7.4

JustBuild builds in isolation (i.e., using temp directory, possibly in the user's home).

Problem: **cc1**/**cc1plus** contain checksums, computed from:

1. the path of the linker that was used
2. the relevant object files

... both contain the build directory.

Compute Reproducible Checksums!

independently of the build directory:

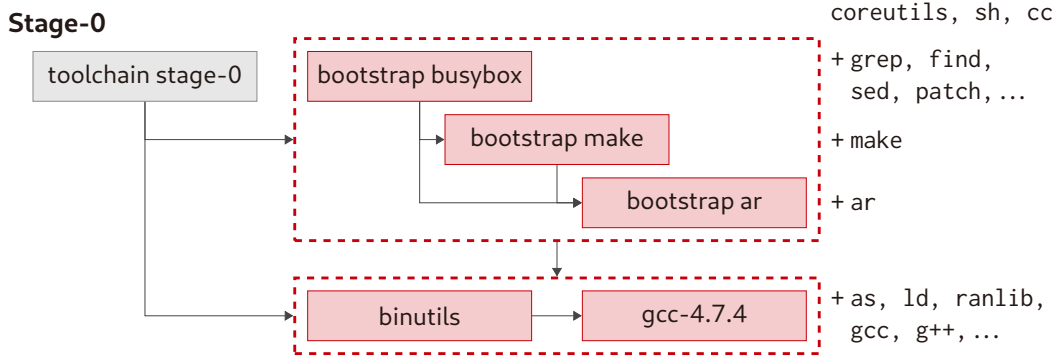
- strip linker path
- strip debug symbols from objects

```
--- a/gcc/Makefile.in
+++ b/gcc/Makefile.in
@@ -1804,12 +1804,17 @@ gcc-cross$(exeext): xgcc$(exeext)

 checksum-options:
     echo "$(LINKER) $(ALL_LINKERFLAGS) $(LDFLAGS)" > checks
+    && sed -i 's|'$${BUILD_ROOT_DIR:-/nonexistent}'|/build|
     && $(srcdir)/../move-if-change checksum-options.tmp che

 # compute checksum over all object files and the options
 cc1-checksum.c : build/genchecksum$(build_exeext) checksum
     $(C_OBJS) $(BACKEND) $(LIBDEPS)
-    build/genchecksum$(build_exeext) $(C_OBJS) $(BACKEND) $
+    rm -rf stripped_c_checksum_inputs; \
+    mkdir stripped_c_checksum_inputs; \
+    cp $(C_OBJS) $(BACKEND) $(LIBDEPS) stripped_c_checksum_
+    $(STRIP_FOR_TARGET) -g stripped_c_checksum_inputs/*; \
+    build/genchecksum$(build_exeext) $$(ls stripped_c_check
                   checksum-options > cc1-checksum.c.tmp &&
     $(srcdir)/../move-if-change cc1-checksum.c.tmp cc1-chec
```

Note that all patches are applied automatically as part of the *bootstrap process*.

Bootstrap Process

Bootstrap Stage-0: *GCC-4.7.4*



Run bootstrap and install with JustBuild:

```
just-mr --main stage-0/gcc install toolchain -o /opt/gcc-4.7.4
```

Wrap-up

Wrap-up

Works on any x86_64 Linux system

Tested on ArchLinux, Debian, Fedora, and VoidLinux, but also very different systems:

NixOS	only GCC/Clang + glibc
prologic/ulinux	TinyCC + musl libc (≈6 MB on DockerHub)

Use as explicit dependency

Make the toolchain a committed project dependency

- easier project setup
- `git bisect` support
- "predict" binary hashes! (see our demo application)

Install JustBuild and try it yourself:

```
git clone https://github.com/just-buildsystem/bootstrappable-toolchain.git
cd bootstrappable-toolchain
just-mr --main gcc-13.2.0-musl install -D'{"ARCH":"x86_64"}' -o /opt/gcc
# grab some coffee and wait...
```

Thank you.

Special thanks to the community!

- Rich Felker https://github.com/richfelker/musl-cross-make
- https://bootstrappable.org
- https://reproducible-builds.org
- https://bootstrapping.miraheze.org/wiki/C_compilers