Non GCC toolchain for Embedded Linux

Khem Raj, Comcast
Introduction

- Embedded Linux Toolchains (Non Android)
- Cross compiling
- Clang/musl based toolchains for Embedded Linux
- What’s missing still
- Discussions
Current Norm

• Embedded Linux is primarily cross-compiled
• GCC cross toolchains build systems
  • Crosstool-NG
  • Buildroot
  • OpenWRT
  • OpenEmbedded
  • OpenADK
  • Crossdev (Gentoo)
  • Debian cross-tools
  • EmbToolkit
  • many more…. 
Current Norm

- GNU toolchain basic ingredients
  - Binutils
    - Provides linker, assembler and post processing tools
  - C/C++/Java/ADA/Fortran/golang gcc compiler
    - Cross Compilers
      - C/C++ runtime (libgcc, libstdc++, libfortan …)
  - System C Standard Library
    - glibc
    - uclibc
    - newlib
  - Debugger
    - gdb
Current Norm

• Supports many Architectures/machines

• As many as gcc backend can support

• [https://gcc.gnu.org/backends.html](https://gcc.gnu.org/backends.html)

• arc,arm,aarch64,mips,mips64,powerpc,powerpc64, x86, x86_64, tile, nios2, microblaze, and many more ……
Toolchain Build Sequence

Pre-requisites e.g gmp, mpfr, mpc

- cross binutils
- bootstrap cross gcc stage 1
  - linux kernel headers
  - libc headers/startupfiles/dummy libc.so

- cross gdb
- gcc-runtime
  - Full libc
  - full cross gcc stage 2
Toolchain Build Sequence

- Cross building GNU toolchain is a bit convoluted
  - GCC and glibc has catch-22
  - various features poked by configure are faked.
A non-gcc toolchain

• Use clang/llvm for cross compiler

• Use musl for C library

• Replace C/C++ runtime by compiler-rt and libc+

• Use native assembler, linker ( limited support )
Toolchain Build Sequence

Pre-requisites e.g. gmp, mpfr, mpc

cross binutils

stage 1 cross gcc

linux kernel headers

Clang/llvm universal driver

clang-cross stub

libc headers/startupfiles/dummy libc.so

cross gdb

gcc-runtime

Full libc

full cross gcc stage 2
Non-GCC Cross toolchain

Pre-requisites e.g. gmp, mpfr, mpc

cross binutils

cross gdb

cross gcc

stage 1 cross gcc

gcc-runtime

full cross gcc stage 2

Full libc

linux kernel headers

libc headers/
startupfiles/
dummy libc.so
Non-GCC Cross toolchain
Non-GCC Cross toolchain

Pre-requisites e.g gmp, mpfr, mpc

cross binutils

Clang universal driver

Linux kernel headers

musl headers/startupfiles/dummy

cross gdb

gcc-runtime

Full libc

Full cross gcc stage 2
Non-GCC Cross toolchain

- Pre-requisites e.g. gmp, mpfr, mpc
- cross binutils
- Clang universal driver
- linux kernel headers
- musl headers/startupfiles/dummy
- cross gdb
- gcc-runtime
- Full libc
Non-GCC Cross toolchain

Pre-requisites e.g gmp, mpfr, mpc

cross binutils

Clang universal driver

linux kernel headers

musl headers/startupfiles/dummy

musl libc

gcc-runtime

cross gdb

Full musl libc

cross gdb
Non-GCC Cross toolchain

Pre-requisites e.g. gmp, mpfr, mpc → cross binutils → Clang universal driver → linux kernel headers → musl headers/startupfiles/dummy

cross gdb → Compiler-rt/libc++ → Full musl libc
Clang-musl - Now

- Still uses GCC runtime
- Replace libgcc with compiler-rt
- Replace libstdc++ with libc++
- What about other runtimes?
Limitations

- Currently targeting armv7a+, aarch64 and x86/x86_64 architectures
- There are other prevalent architectures e.g. mips, PowerPC in Embedded Linux
Limitations

• Not all software is yet clang compile-ready
• Add GCC and Clang cross compilers to SDK
• musl based SDKs don't have full set of tools yet
OE Work

- meta-musl layer (https://github.com/kraj/meta-musl)
- meta-clang layer (https://github.com/kraj/meta-clang)
Thank you