SPARC-bare-metal, CMake, LLVM/Clang

Paris Embedded – Meetup
13/04/2016

Patrick Boettcher

YAISE

Introduction

- German
- Kernel developer since 2004 – contributions to mainly v4l-dvb
- followed by years of activity in embedded systems
- Freelancing for my company – YAISE

- This presentation based on work I‘m currently doing for one of my clients
  - SDR solution, generic CPU, embedded development architecture, using LLVM/Clang

    WORK IN PROGRESS

Overview

- CMake in a nutshell
- CMake example
- The test-system
- Software components
- Example

CMake in a nutshell

- Is a “Makefile”-generator:
  - detects the environment (compilers, external deps (libraries, tools, includes))
  - evaluates dependencies between internal targets
  - generates the build-files
- Supports: GNU Makefiles, .ninja (ever tried ninja?), vcproj-files, works with QT Creator and what not
- Promotes out-of-source-builds
- Can do cross-compilation
- Syntax … one has to get used to it

CMake examples

- Simplest case

```
add_executable(program program.c)
```

- Library

```
add_library(coolstuffpp api.cpp core.cpp)
target_include_directories(coolstuffpp
                           PUBLIC
                           ${CMAKE_CURRENT_SOURCE_DIR})
```

- Dependencies

```
target_link_libraries(program coolstuffpp)
```

- Build for C++11 (compiler features)

```
target_compile_features(coolstuffpp
                        PUBLIC cxx_nonstatic_member_init)
```

The system

- Purely emulated
  - CPU – sparc-based Instruction Set Simulator:
    - simulation/emulation of each in inst. with a SystemC-model
  - RAM
  - UART
  - Communication bus

System’s basic software development components

- Cross-compiler
  - sparc-unknown-elf-gcc – generated with ct-ng (great stuff)
  - LLVM/Clang 3.8 – generated with CMake/Ninja
    - The same LLVM-binaries contain the compiler-backend for different CPUs
- Crt0 (C run-time – the very beginning)
  - initializes the system before calling main() - e.g.: memsetting the bss to ‘0’, resetting things
- LDScript – defining the memory-layout of the system;
  - where to put code, where to put the data
  - used by the linker
- CMake-toolchain-file – one per compiler (gcc vs. clang)

Example

- Hello World!
  - Host for Release
  - Host with DebugInfo
  - Sparc with clang
  - Sparc with gcc