Introduction to Build Systems and CMake

DAY 4 : Designing your Build System

Sponsored by

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.

01

Review: The Problem

The Problem

There are several problems that teams are facing:

- Managing multiple build configurations
- Slow builds
- Software quality issues
- Inability to use modern techniques like DevOps, Simulation, TDD, etc, effectively
- Productivity issues (time to market, product quality)

The Solution

A carefully designed CMake build system will:

- Simplify build configurations with better dependency management
- Allow for faster, cross-platform builds
- Enable consistency across different development environments
- Unlock modern development processes and tools like DevOps, Simulation, and TDD
- Increase productivity

THE SPEAKER

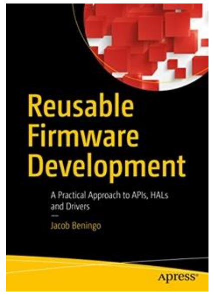
Jacob Beningo

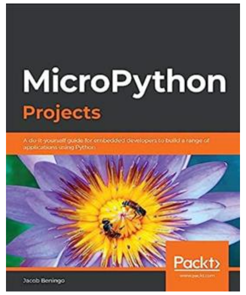
Jacob@beningo.com

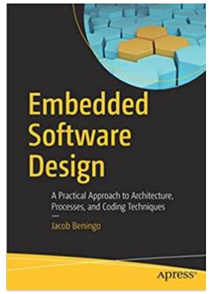
Beningo Embedded Group – CEO / Founder

Focus: Embedded Software Consulting and Training

Help teams deliver higher-quality embedded software faster. We specialize in creating and promoting embedded software excellence in businesses around the world.


Reusable Firmware Development
A Practical Approach to APIs, HALs and Drivers
Jacob Beningo
Apress




Embedded Software Design
A Practical Approach to Architecture, Processes, and Coding Techniques
Jacob Beningo
Apress

Blogs for:

- DesignNews.com
- Embedded.com
- EmbeddedRelated.com
- MLRelated.com

Visit **www.beningo.com** to learn more

The Plan

Transform Your Build Process: Streamline, Modernize, and Boost Productivity with CMake

Step 1

Learn the Technology

Step 2

Design the Solution

Step 3

Adopt Modern Practices

02

Your Ideal Build System

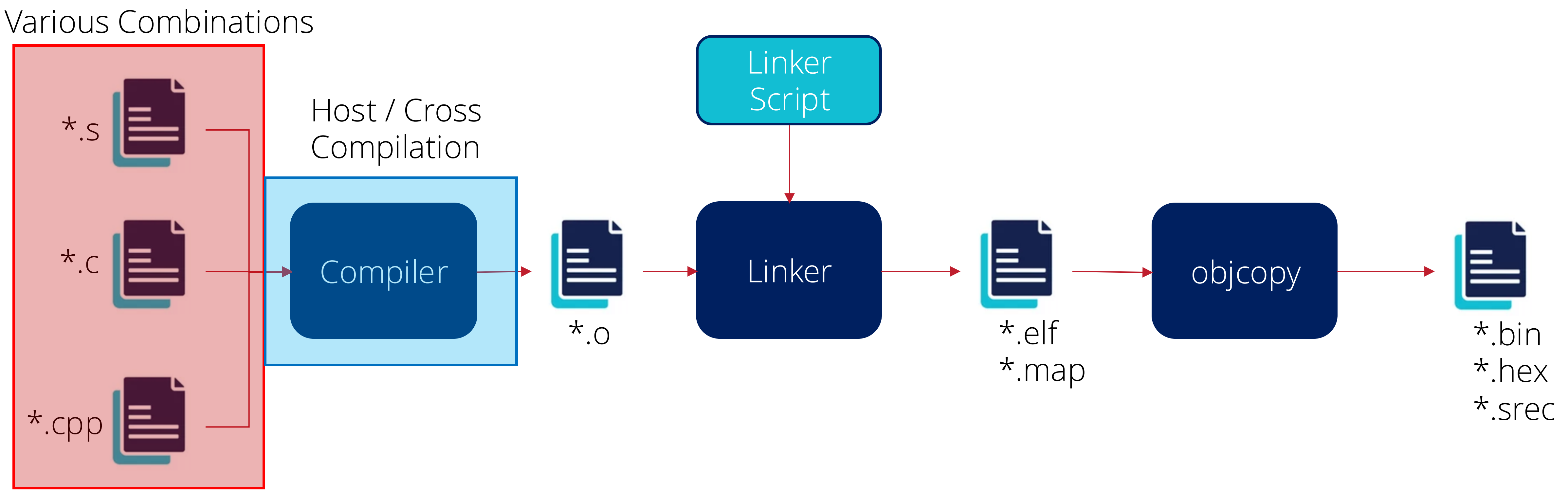
What your build system should get you . . .

An **ideal build system** for embedded developers should

- streamline the development workflow
- improve productivity
- ensure reliability
- support the unique challenges of embedded systems

It should also be **adaptable** to **various** microcontroller architectures and vendor-specific SDKs.

What your build system does



Characteristics

- Cross Compilation Support
- Toolchain Management
- Configuration Management
- Build System
- Build Management
- Dependency Management
- Debugging and Debugging Symbols
- Flashing and Deployment
- Build Artifact Management
- Testing and Test Automation
- CI/CD Integration
- Documentation Integration
- Version Control Management
- Reporting
- Extensibility and Customization
- Scalable and Portable
- Flexible

Audience POLL Question

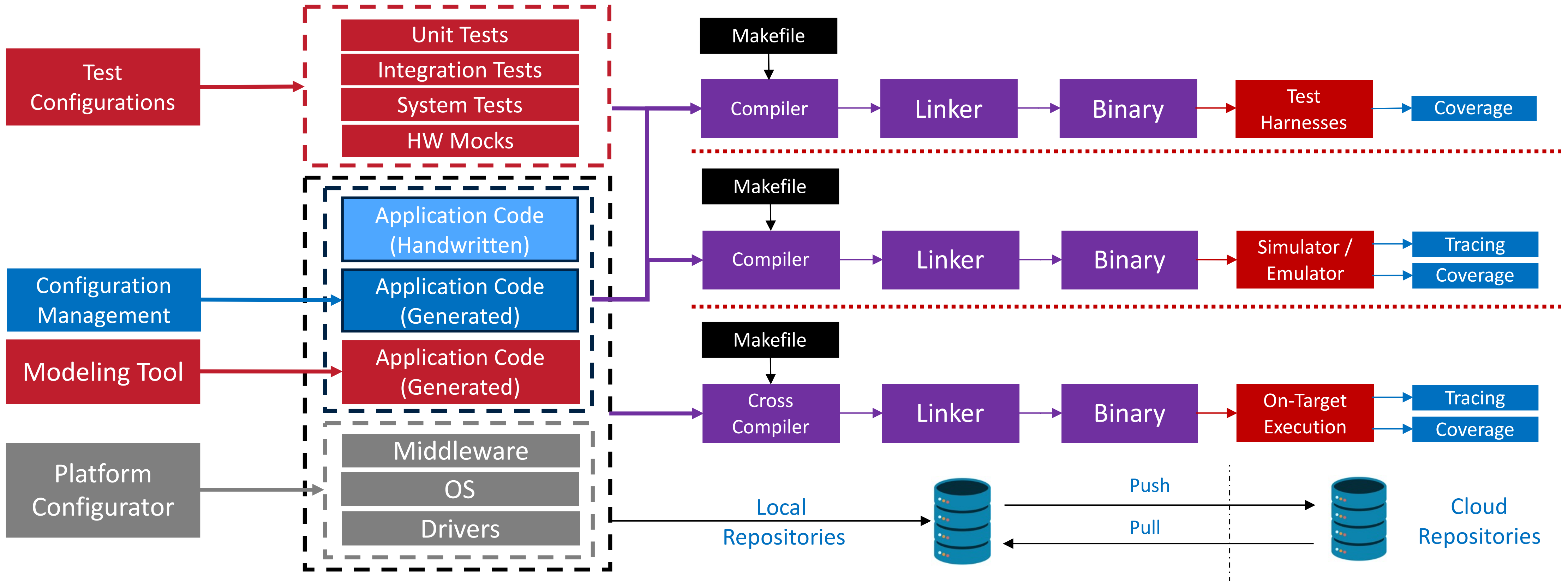
What should your ideal build system do?

a) streamline the development workflow
b) improve productivity
c) ensure reliability
d) support the unique challenges of embedded systems
e) All the above

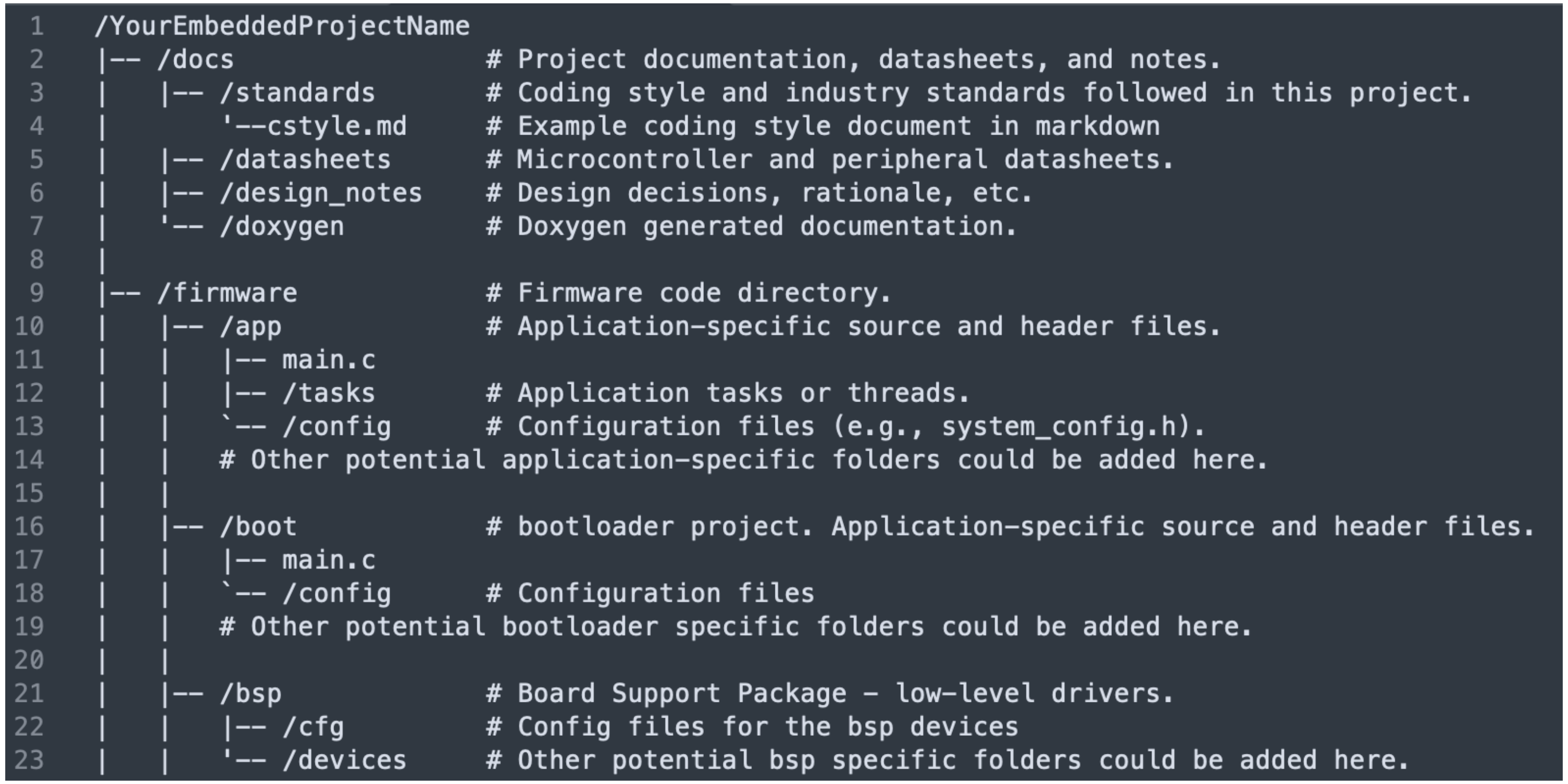
03

Project Organization

The Build System Diagram

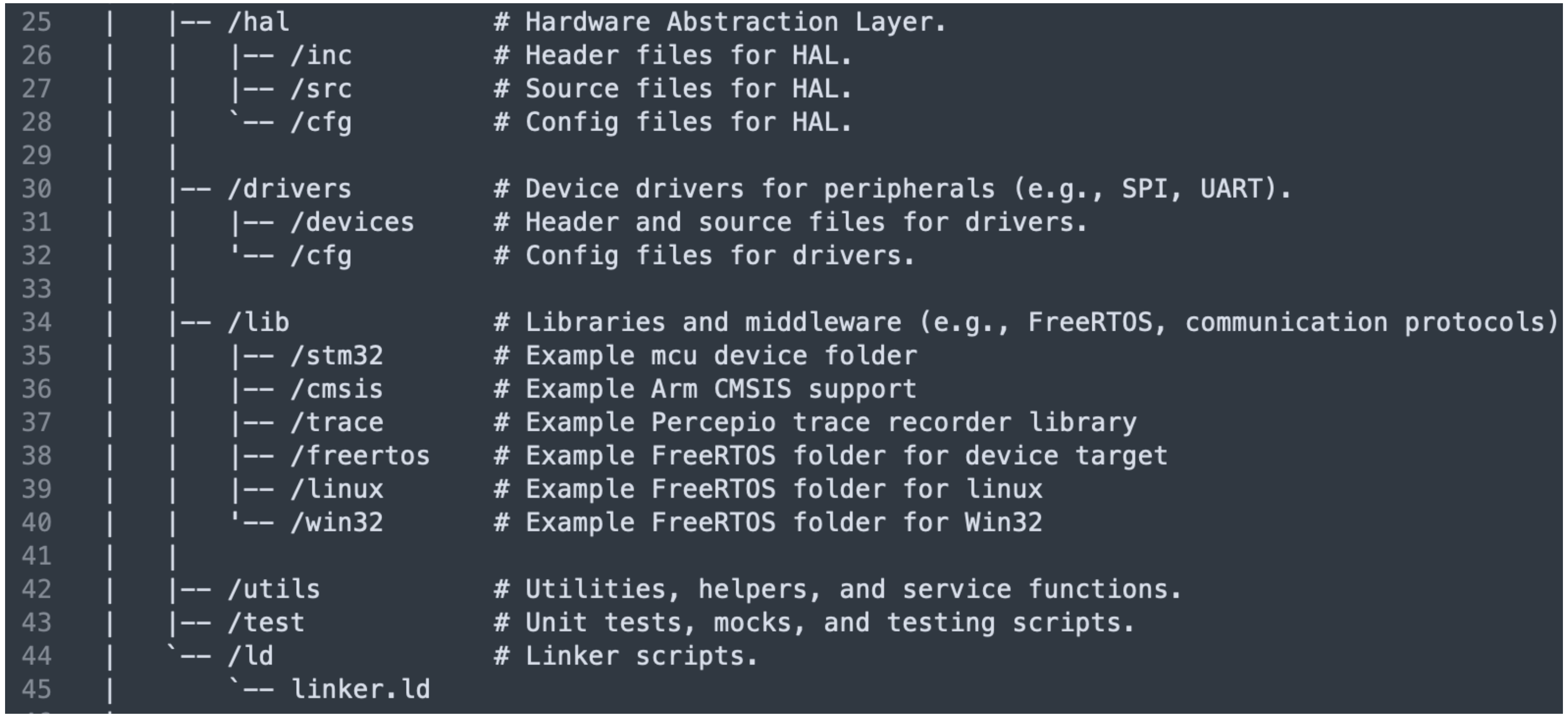


Project Organization



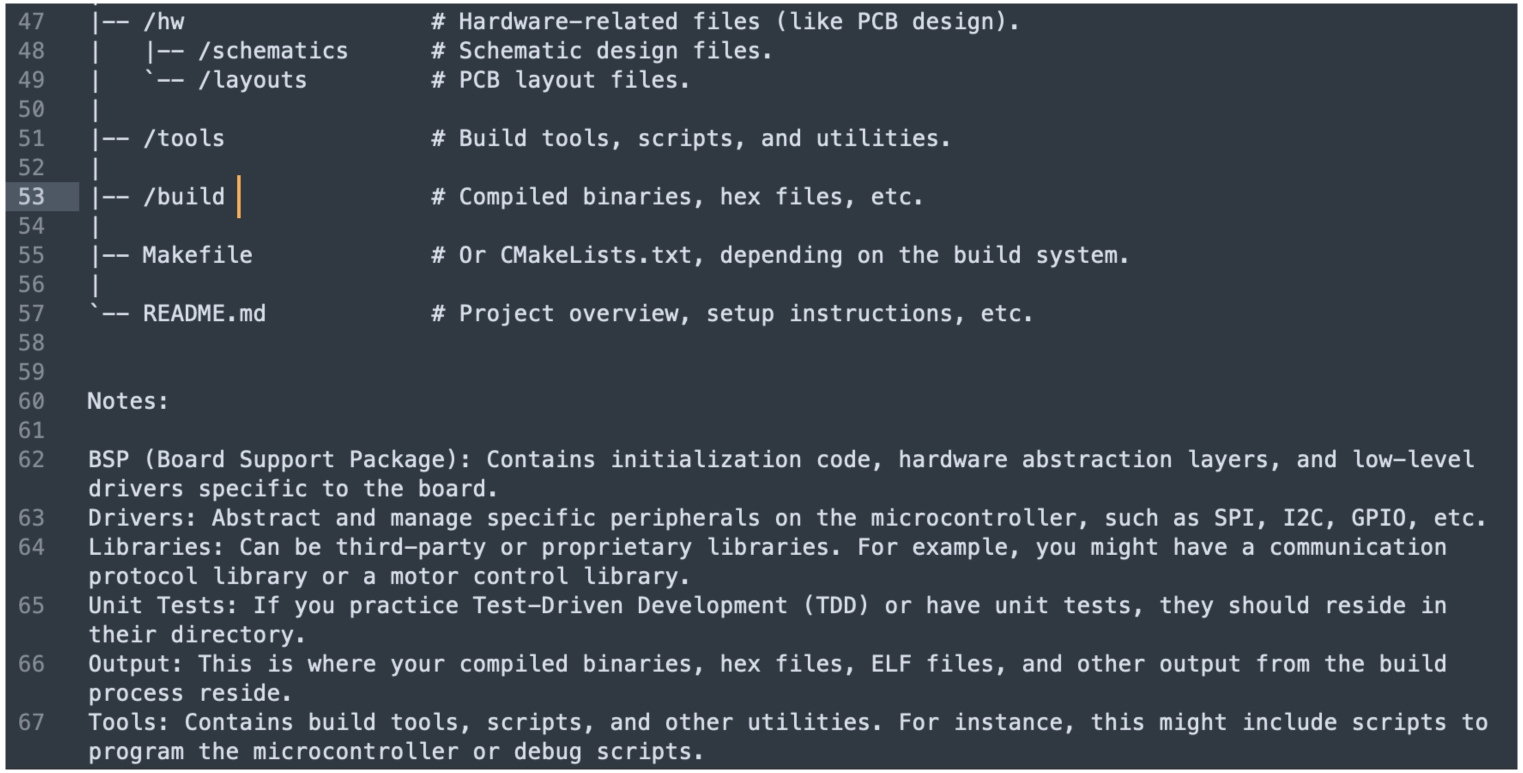
```
/YourEmbeddedProjectName
|-- /docs                # Project documentation, datasheets, and notes.
|   |-- /standards       # Coding style and industry standards followed in this project.
|       '--cstyle.md     # Example coding style document in markdown
|   |-- /datasheets      # Microcontroller and peripheral datasheets.
|   |-- /design_notes    # Design decisions, rationale, etc.
|   '-- /doxygen         # Doxygen generated documentation.
|
|-- /firmware            # Firmware code directory.
|   |-- /app             # Application-specific source and header files.
|   |   |-- main.c
|   |   |-- /tasks       # Application tasks or threads.
|   |   `-- /config      # Configuration files (e.g., system_config.h).
|   |   # Other potential application-specific folders could be added here.
|   |
|   |-- /boot            # bootloader project. Application-specific source and header files.
|   |   |-- main.c
|   |   `-- /config      # Configuration files
|   |   # Other potential bootloader specific folders could be added here.
|   |
|   |-- /bsp             # Board Support Package - low-level drivers.
|   |   |-- /cfg         # Config files for the bsp devices
|   |   '-- /devices     # Other potential bsp specific folders could be added here.
```

Project Organization



```
|   |-- /hal            # Hardware Abstraction Layer.
|   |   |-- /inc        # Header files for HAL.
|   |   |-- /src        # Source files for HAL.
|   |   `-- /cfg        # Config files for HAL.
|   |
|   |-- /drivers        # Device drivers for peripherals (e.g., SPI, UART).
|   |   |-- /devices    # Header and source files for drivers.
|   |   '-- /cfg        # Config files for drivers.
|   |
|   |-- /lib            # Libraries and middleware (e.g., FreeRTOS, communication protocols)
|   |   |-- /stm32      # Example mcu device folder
|   |   |-- /cmsis      # Example Arm CMSIS support
|   |   |-- /trace      # Example Percepio trace recorder library
|   |   |-- /freertos   # Example FreeRTOS folder for device target
|   |   |-- /linux      # Example FreeRTOS folder for linux
|   |   '-- /win32      # Example FreeRTOS folder for Win32
|   |
|   |-- /utils          # Utilities, helpers, and service functions.
|   |-- /test           # Unit tests, mocks, and testing scripts.
|   `-- /ld             # Linker scripts.
|       `-- linker.ld
```

Project Organization



```
|-- /hw                          # Hardware-related files (like PCB design).
|    |-- /schematics             # Schematic design files.
|    `-- /layouts                # PCB layout files.
|
|-- /tools                       # Build tools, scripts, and utilities.
|
|-- /build                       # Compiled binaries, hex files, etc.
|
|-- Makefile                     # Or CMakeLists.txt, depending on the build system.
|
`-- README.md                    # Project overview, setup instructions, etc.


Notes:

BSP (Board Support Package): Contains initialization code, hardware abstraction layers, and low-level drivers specific to the board.
Drivers: Abstract and manage specific peripherals on the microcontroller, such as SPI, I2C, GPIO, etc.
Libraries: Can be third-party or proprietary libraries. For example, you might have a communication protocol library or a motor control library.
Unit Tests: If you practice Test-Driven Development (TDD) or have unit tests, they should reside in their directory.
Output: This is where your compiled binaries, hex files, ELF files, and other output from the build process reside.
Tools: Contains build tools, scripts, and other utilities. For instance, this might include scripts to program the microcontroller or debug scripts.
```

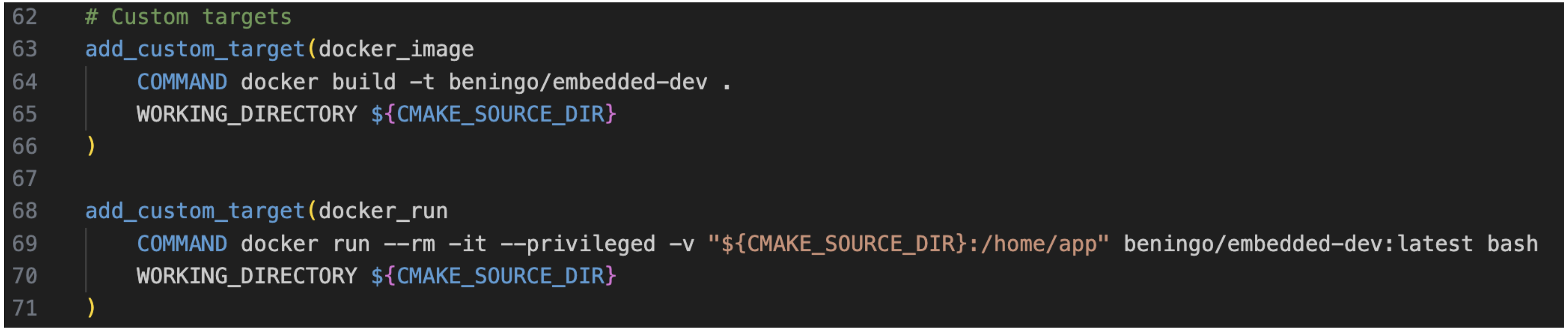
Audience POLL Question

How important is configuration management to your development?
a) Not important
b) Somewhat important
c) Important
d) Critically important

04

Custom Build Commands

Custom Docker Commands



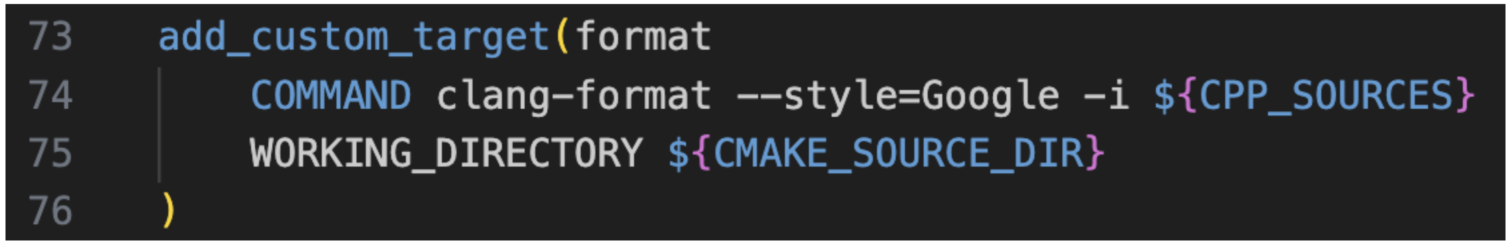
```
# Custom targets
add_custom_target(docker_image
    COMMAND docker build -t beningo/embedded-dev .
    WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}
)

add_custom_target(docker_run
    COMMAND docker run --rm -it --privileged -v "${CMAKE_SOURCE_DIR}:/home/app" beningo/embedded-dev:latest bash
    WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}
)
```

```
cmake --build build --target docker_image
```

```
cmake --build build --target docker_run
```

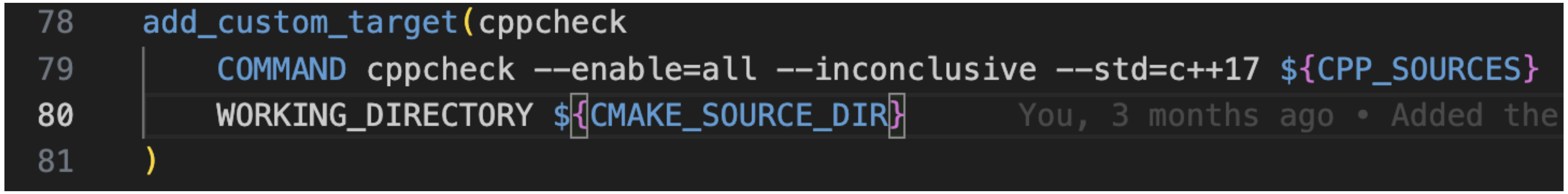
Running clang-format



```
add_custom_target(format
    COMMAND clang-format --style=Google -i ${CPP_SOURCES}
    WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}
)
```

```
cmake --build build --target format
```

Running Unit Tests


```
add_custom_target(cppcheck
    COMMAND cppcheck --enable=all --inconclusive --std=c++17 ${CPP_SOURCES}
    WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}      You, 3 months ago • Added the
)
```

cmake --build build --target cppcheck

Audience POLL Question

What might you use custom commands for?

a) Unit Testing
b) Linting
c) All the above and more
d) Nothing

Next Steps

05

Embedded Build System

Transform your build system with the free Beningo Embedded Build System example:

- Docker container build system
- Makefile-based
- CMake with Ninja Example
- Compilation scripts
- Integrated tools like cpputest

https://mailchi.mp/beningo/beningo-devops

Additional Resources

Please consider the resources below:

- Jacob's Blogs
- Jacob's CEC courses
- Embedded Software Academy

- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

www.beningo.com

Consulting

Coaching

Training

Next Steps

- ✓ Introduction to Embedded Build Systems
- ✓ CMake Fundamentals
- ✓ CMake for Embedded Systems
- ✓ Designing your Build System
- Adopting Modern Practices

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

BENINGO EMBEDDED GROUP

informa markets