oL10 - Embedded Modern C++ Programming

Objectives

- Discover the modern C++ features
- Learn the language changes in C++11, C++14, C++17 and C++20
- Discover the new functionalities added to the standard library
- Learn advanced modern C++ features like perfect forwarding
- Moving from traditional C++ to modern C++
- Emphasizing the essential modern C++ features used in embedded application

Labs are conducted QEMU ARM-based board

Prerequisite

- Basic C++ Language knowledge (see our oL3 - Embedded C++ Programmingcourse)

Course environment

- Theoretical course
  - PDF course material (in English)
  - Course dispensed using the Teams video-conferencing system
  - The trainer to answer trainees' questions during the training and provide technical and pedagogical assistance through the Teams video-conferencing system
- Practical activities
  - Practical activities represent from 40% to 50% of course duration
  - One Online Linux PC per trainee for the practical activities
  - The trainer has access to trainees' Online PCs for technical and pedagogical assistance.
  - Eclipse environment and GCC compiler
- Downloadable preconfigured virtual machine for post-course practical activities

Duration

- Total: 12 hours
- 2 sessions, 6 hours each (excluding break time)
- From 40% to 50% of training time is devoted to practical activities
- Some Labs may be completed between sessions and are checked by the trainer on the next session

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First Session

Introduction to modern C++

- Overview
- Storage class specifiers
- Uniform initialization
- C++ Named Requirements
- Automatic type deduction
  - The auto keyword
  - The auto keyword as a return type from a function
  - Using auto for declaring function signatures
  - Automatic constant references
  - Forwarding references
  - Advantages of using auto in embedded systems

Exercise: Using auto to declare variables

Keywords

- Enum class
- override and final
- Inline variables
- nullptr
- static_assert
- noexcept
- constexpr and if constexpr
- decltype
- Defaulted and deleted functions
  - Implementing a thread-safe singleton

Exercise: Using modern C++ keywords
Exercise: Create a singleton using modern C++

New functionalities

- Structured binding
- Range-based for loops
- Nested namespaces and namespace aliases
- Alignment
  - Alignas
  - Alignof
- Move semantics and r-value references
  - Copy-constructing and Move-constructing
  - r-value references
  - Perfect forwarding

Exercise: Using the new for loop syntax
Exercise: Using std::tuple
Exercise: Move semantics performance advantages on embedded systems

Modern C++ Standard Library

- Standard Library
  - std::optional

    - std::variant
    - std::any
    - std::byte
    - std::hash
    - Filesystem library
- Literals
    - Cooked literals
    - Standard literal operators
    - Raw literals
    - Raw string literals
- Random number generation
    - Random number generation engines
    - Random number generation distributors
- Containers
    - std::array
    - std::forward_list
    - Unordered associative containers

Exercise: Using the new elements added to the standard library
Exercise: Using std::optional

Second Session

String Manipulation

- New string Types
    - std::u16string
    - std::u32string
- Basic string view
- Converting between numeric and string types
- Elementary string conversions
- Input/output manipulators
    - std::get_money, std::put_money
    - std::get_time, std::put_time
    - std::quoted
- Regular expressions
    - Format of a string
    - Parsing the content of a string
    - Replacing the content of a string

Exercise: Using String class and String literals

Concurrency and Multithreading

- Introduction
- Thread
- Atomic operations
    - Atomic features
    - Non-class functions
    - Atomic flag
    - Memory order
- Mutex
    - Avoiding using recursive mutexes
- Sending notifications between threads
- Condition variables
- Future and Promise
- Task and Async
- Modern C++ and RTOS

Exercise: Blink synchronously 4 Leds

Lambda functions

- Syntax of lambdas
- Defining lambdas
- Using lambdas
  - Using lambdas with standard algorithms
  - Assigning lambdas to function pointers
  - Lambdas and std::function
  - Writing a function that accepts a lambda as parameter
- Polymorphic lambdas
- Recursive lambdas

Exercise: Understanding lambda
Exercise: Using lambda to modify and display a vector

Dynamic memory management

- Memory Management
- Memory Errors
- Smart Pointers
  - Raw Pointers
  - Automatic pointers
  - Unique Pointers
  - Shared Pointers
  - Weak Pointers

Exercise: Override new and delete
Exercise: Understanding unique and shared pointers