Lab

Exercise 04 „Platforms for Embedded Systems“ Prof. Cochlovius

- Aspects of Programminng Embedded Platforms -

Goal:
This exercise deepens your experience with using Eclipse and you will dive into some aspects of programming embedded platforms.

Exercise 4.1: Handling of signals
Create a new Managed-Make project within Eclipse. Copy the hello-world example with loop from the very first exercise into your new project (or you can quickly write it new). Now you extend the program, register signal handlers to capture the signals INT, USR1, KILL and STOP. You decide whether you have a separate handler for each signal or a single handler combined for all signals). Compile and execute. What do you notice during signal registration?

In the handler(s), pls. implement the following functionality: On processing signal USR1 the loop index gets decreased by 20 (min. value 0), INT the loop index gest increased by 10 (max. value 100). Recompile and verify the intended behavior on both host and target platform.

Exercise 4.2: Memory requirements of *structs* and sequence of declaration
Create a C++-Template project in Eclipse (managed make). Declare the structs Bad and Better, instantiate one variable of both and print out the size of both variables (**`cout << sizeof(<varname>);`**). Execute the program in Eclipse on both host and target platform.

Note:
In case, Eclipse complains about the symbol „EXIT_SUCCESS“: this is related tot he C++-indexer (i.e. intelligent editor) and is NOT related tot he compilation process. You can substitute the symbol by “0”.

Exercise 4.3: Memory requirements of a simple object
Explore the memory requirements and the memory layout of a simple *object* on the heap. Explore the impact of virtual methods. For this, create a new C++-template project mem_size.

Then, pls. define a **`class A`** including:

- 2 private attributes of type **`int`** (mDummy, mValue), and
- 1 function void f(), which prints the address of these attributes: (**`cout << &(this->mDummy)`**);

Now create one instance of A on the heap, and run the program. Note the addresses. Now add a virtual function (e.g. a destructor) and run again. What has changed? What can we learn about the memory layout of an object in C++?

Exercise 4.4: Memory requirements of objects of a derived class with virtual method
Now find out about the memory requirements and layout of an object of a class with virtual method, which is derived from a base class.

Create another C++-template project mem_size2 in Eclipse and copy the code **`mem_size2.cpp`** into your project. Try to understand the code. Then compile and run it. What is the memory layout?

Exercise 4.5: Problems with multiple inheritance
Explore the problems related to cross-casting across different branches in the inheritance graph. Create a new C++-Template project mem_size3 and copy the code **`mem_size3.cpp`** into your project. Try to understand the code. Then compile and run it. Where does it get dangerous, i.e. wrong? Why? How can you fix the issue safely?