Daniel Fewster n10459219

Youtube Demo Link: https://youtu.be/cik_Z-0CsbE

Statement of Completeness

All functionality across the simulator, manager, and fire alarm have been implemented. There are no known bugs.

Statement of Contribution

This assignment was done individually, so everything was completed by Daniel Fewster (myself).

Assessment of the Safety-Critical Fire Alarm System

Identification of Failed Safety-Critical Standards

When first inspecting the provided **firealarm.c**, I was immediately able to spot several MISRA C guideline violations. Allocation of memory on the heap due to a linked list implementation for storing temperature values, to using octal constants to create addresses for indexing values in the shared memory data. Using a goto statement for when "emergency mode" activates, which was in a busy waiting loop instead of using pthread wait with the unused mutex and condition variables declared, and more.

In order to give better coverage of the MISRA C guideline violations, I will list those detected when running **cppcheck** against **firealarm.c**:

- Octal constants shall not be used (MISRA-C 2012 7.1)
- A string literal shall not be assigned to an object unless the object's type is "pointer to const-qualified char". (MISRA-C 2012 7.4)
- Function types shall be in prototype form with named parameters. (MISRA-C 2012 8.2)
- A compatible declaration shall be visible when an object or function with external linkage is defined. (MISRA-C 2012 8.4)
- Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category. (MISRA-C 2012 10.4)
- A conversion should not be performed from pointer to void into pointer to object. (MISRA-C 2012 11.5)
- A cast shall not be performed between pointer to void and an arithmetic type. (MISRA-C 2012 11.6)
- MISRA violation 1201 with no text in the supplied rule-texts-file. (MISRA-C 2012 12.1)
- The comma operator should not be used. (MISRA-C 2012 12.3)
- A full expression containing an increment (++) or decrement (--) operator should have no other potential side effects other than that caused by the increment or decrement operator. (MISRA-C 2012 13.3)
- The controlling expression of an if statement and the controlling expression of an iteration-statement shall have essentially Boolean type. (MISRA-C 2012 14.4)
- The goto statement should not be used. (MISRA-C 2012 15.5)
- A function should have a single point of exit at the end. (MISRA-C 2012 15.5)

- The body of an iteration-statement or a selection-statement shall be a compound-statement. (MISRA-C 2012 15.6)
- Functions shall not call themselves, either directly or indirectly. (MISRA-C 2012 17.2)
- The value returned by a function having non-void return type shall be used. (MISRA-C 2012 17.7)
- The +, -, += and -= operators should not be applied to an expression of pointer type. (MISRA-C 2012 18.4)
- The memory allocation and deallocation functions of <stdlib.h> shall not be used. (MISRA-C 2012 21.3)
- The Standard Library input/output functions shall not be used. (MISRA-C 2012 21.6)
- The Standard Library functions bsearch and qsort of <stdlib.h> shall not be used. (MISRA C 2012 21.9)
- The Standard Library time and date functions shall not be used. (MISRA-C 2012 21.10)
- Functions and objects should not be defined with external linkage if they are referenced in only one translation unit. (MISRA-C 2012 8.7)

Description of Approach

As said above, I noticed several things wrong with **firealarm.c**, and these were spread across the whole program. I decided it would be easier to write my own program (**fire_alarm.c**) that followed MISRA C guidelines to my knowledge at the time, and then debug my violations from there using **cppcheck**.

Potential Safety-Critical Concerns and Reservations of New Implementation

After having debugged my fire alarm implementation, I was left with two violations which I cannot see myself overcoming due to limitations with pthread. To my knowledge, **pthread_create** expects function pointers to be of type **(void *(*)(void *))**, so no matter what, the routine given will need a parameter of type **(void *)**, so you have no choice but to cast from there if you want to parse anything other than that type, violating MISRA-C 2012 11.5.