PREVENTING RUN-TIME BUGS AT COMPILE TIME USING C++*

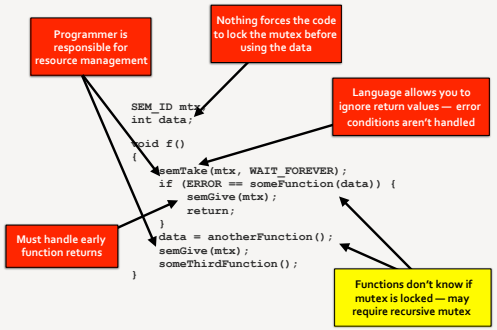
R. Neswold
FNAL, Batavia, IL USA

Abstract

In order for a system to be reliable, its software needs to be carefully designed. Despite our best efforts, however, errors occur and we end up having to debug them. Unfortunately, debugging an embedded system changes its dynamics, making it difficult to find and fix concurrency issues. This paper describes techniques, using C++, making it impossible to write code susceptible to certain run-time bugs. A concurrency library, developed at Fermilab, is used in the examples illustrating these techniques.

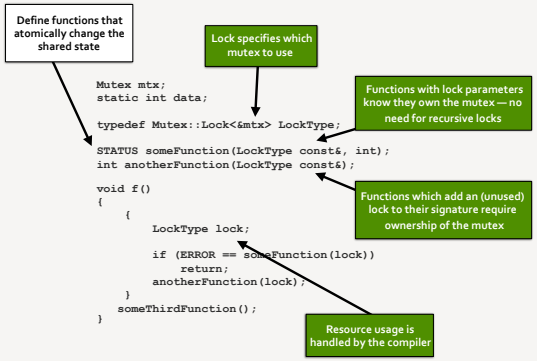
Problems with C APIs:

With C, there are many mundane details with which to contend that make it hard to focus on the overall design.



How can C++ Help?

C++ objects have a strictly defined lifetime which can be leveraged to manage resources.



Using Templates to Generate Code and Specialize Interfaces:

A templated class is defined to validate data requests from the network to a driver handler. It's job has two responsibilities. 1) make sure the length and offset parameters match the size of the driver's data and 2) write the data to a buffer destined for the network in the correct byte-order.

We define the class to check the length and offset in the constructor. If they are invalid, throw an exception. If the request is good, the object is created. Methods of the class handle transforming native data to network data.

This first handler only returns 16-bit integers:

```
ReadingProxy<uint16_t> reading(req);
```

Its template generates...

... the constructor, containing some simple tests to make sure the request is for a single 16-bit value

```
if (req->length != sizeof(uint16_t))
    return ERR_BADLEN;
if (req->offset != 0)
    return ERR_BADOFF;
```

... an assignment operator which takes a 16-bit value and writes it to the request's output buffer

```
uint16_t operator=(uint16_t const&);
```

This next handler returns any subset of a 16-element array of 32-bit integers:

```
ReadingProxy<uint32_t[16]> reading(req);
```

Its template generates...

... the constructor, containing more complicated tests (length and offset are multiple of size, etc.)

```
if (req->length % sizeof(uint32_t) != 0)
    return ERR_BADLEN;
if (req->offset % sizeof(uint32_t) != 0)
    return ERR_BADOFF;
if (req->length + req->offset > sizeof(uint32_t) * 16)
    return ERR_BADLENOFF;
```

... an array-like interface for writing to the buffer

```
void assignAt(size_t, uint32_t const&) const;
size_t total() const;
size_t offset()const;
```

* FNAL is operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.