Document number: N3751
Date: 2013-09-03
Working groups: SG12, EWG, CWG
Reply to: gdr@cs.tamu.edu

Object Lifetime, Low-level Programming, and memcpy

Gabriel Dos Reis
Texas A&M University
http://www.axiomatics.org/~gdr/

Abstract

This document proposes to recognize a category of uses of `memcpy` — currently left undefined — as another form of object construction, thereby putting a class of existing practice in low-level system programming on firmer ground.

1 The Problem

The C++ standards currently recognizes the interaction between `memcpy` and object value interpretation only a limited number of cases as listed in paragraphs §3.9/2 and §3.9/3. It leaves undefined the semantics of the following program fragment:

```
const uint32_t bits = 0x9A99993F;
float x;
std::memcpy(&x, &bits, sizeof x); // #1
float y = x * x;              // #2
```

Here, we assume that `float` implements IEEE-754 32-bit single precision floating point arithmetic data type. There are at least two questions here:

1. The line numbered #1 is a `memcpy` between two different objects, and as such is not currently covered by the standards text. So, this line is potentially a source of undefined behavior.

2. The line numbered #2 reads the `float` object `x`, which appears to be initialized nowhere. So, a strict interpretation of the standards suggests that this is another source of undefined behavior.

This fragment is just illustration of a pattern widely used in several low-level C++ system programming communities. The main suggestion of this paper to formally recognize it. Note that while the example uses *declared object*, the core issue remains the same (or possibly worse) with objects with dynamically allocated storage.

2 Suggestions of resolution

The C++ standards is currently silent on whether the use of `memcpy` to copy object representation bytes is conceptually an assignment or an object construction. The difference does matter for semantics-based program analysis and transformation tools, as well as optimizers, tracking object lifetime.

This paper suggests that

1. uses of `memcpy` to copy the bytes of two distinct objects of two different trivial copyable tables (but otherwise of the same size) be allowed

2. such uses are recognized as initialization, or more generally as (conceptually) object construction.

Recognition as object construction will support binary IO, while still permitting lifetime-based analyses and optimizers.

3 Acknowledgment

This write-up is distilled from a larger discussion within the SG-12 Study Group on Undefined Behavior.