Random in C++

Posted on **December 8, 2014**

Recently, I posted an article about my work with random number generators. That work was all done in the Ruby programming language. Now Ruby is a fun language to program in, but I got to thinking that it is not the first choice for embedded work. (Yes, there is mruby, but it is still not well known) So it became clear to me that C or C++ would be good choices to explore.

So I set about porting my Fibonacci pseudorandom number generator to C++. It was not entirely a pleasant experience. I used Visual Studio 2013 (free edition) to do my development. My motivation for picking that tool was simple convenience. I supposed I could have set up Eclipse with GCC, but the tool setup time would have been excessive for my simple goal of evaluating a small bit of code. Further there can be no doubt that many alternative development communities provide little to no support for the majority of users who own Windows based computers. No VS is not perfect either. Most project templates start users off with code littered with non-standard, non-portable language extensions. To the best of my ability, I avoided and removed these. If I missed any, please tell me, so I can make the needed corrections.

It has been a few years since I've programmed in C/C++ in a professional capacity. In the beginning it seemed that everything I did was wrong. I was writing incorrect code in so many ways that the compiler was perplexed as to what I was trying to say. Over the course of a few hours though, things got better and soon, the IDE was starting to feel comfortable and productive.

So how did the two languages compare? Well the C++ code had a lot more "pomp and circumstance" to it. To give an example, the spin function that updates the internal random state looks like this in Ruby:

```
#Cycle through the PRNG once.
def do_spin
  @buffer[-2] = @buffer[0]
  @buffer[-1] = @buffer[1]
  (0...@depth).each do |idx|
    @buffer[idx] = (@buffer[idx+1] + (@buffer[idx+2] >> 1)) & CHOP
```

```
  end
end
```

The C++ version requires an entry in the class declaration in the header file:

```
// much omitted...
void spin(void);
// more omitted...
```

as well as the actual code:

```
// Scramble the eggs some more.
void FibRng::spin(void)
{
    // Copy over the 'end' values.
    ring[depth] = ring[0];
    ring[depth + 1] = ring[1];
    // Spin the wheel!
    for (int i = 0; i < depth; i++)
    {
        ring[i] = (ring[i+1] + (ring[i+2] >> 1)) & CHOP;
    }
}
```

In fact, most of the entities in the class had to be specified twice, once in the header file and once in the source file. On the plus side, this extra detail made it possible for the IDE and the compiler to scrutinize the code to an extent impossible in Ruby. On the other hand this is a lot of extra drudge typing and ignores the fact the C++ developer is responsible for dealing with memory management, which is simple in this case; not so simple in normal, larger programs.

So what about the bottom line? How did the code perform? Well, running from the command line, on the same hardware, with the same algorithm, the C++ version was about *one hundred* times faster. The executable was a mere 7K in size. While these figures are impressive, they are of a trivial example. In real applications, things are much more complex. For now, this is but one data point, and one data point it shall remain. The C++ and Ruby code in question may be found on GitHub.

As always, comments and suggestions are most welcome.

Best regards;

Peter Camilleri (aka Squidly Jones)

This entry was posted in **Announcements** by **Peter Camilleri**. Bookmark the **permalink [http://teuthida-technologies.com/?p=1606]** .