Beware the C++ implicit conversion

devblogs.microsoft.com/oldnewthing/20060524-12

May 24, 2006

Raymond Chen

Today's topic was inspired by a question from a customer:

> I am working on a stack overflow bug. To reduce the size of the stack frame, I removed as many local variables as I could, but there's still a a lot of stack space that I can't account for. What else lives on the stack aside from local variables, parameters, saved registers, and the return address?

Well, there's also structured exception handling information, but that's typically not too much and therefore wouldn't be the source of "a lot" of mysterious stack usage.

My guess is that the code is generating lots of large C++ temporaries. Consider the following program fragment:

```
class BigBuffer
{
public:
 BigBuffer(int initialValue)
   { memset(buffer, initialValue, sizeof(buffer)); }
private:
 char buffer[65536];
};
extern void Foo(const BigBuffer& o);
void oops()
{
 Foo(3);
}
```

"How does this code even compile? The function `Foo` wants a `BigBuffer`, not an integer!" Yet compile it does.

That's because the compiler is using the `BigBuffer` constructor as a converter. In other words, the compiler inserted the following temporary variable:

```
void oops()
{
 BigBuffer temp(3);
 Foo(temp);
}
```

It did this because a constructor that takes exactly one argument serves two purposes: It can be used as a traditional constructor (as we saw with `BigBuffer temp(3)`) or it can be used to provide an implicit conversion from the argument type to the constructed type. In this case, the `BigBuffer(int)` constructor is being used as a conversion from `int` to `BigBuffer`.

To prevent this from happening, use the `explicit` keyword:

```
class BigBuffer
{
public:
 explicit BigBuffer(int initialValue)
   { memset(buffer, initialValue, sizeof(buffer)); }
private:
 char buffer[65536];
};
```

With this change, the call to `Foo(3)` raises a compiler error:

```
sample.cpp: error C2664: 'Foo' : cannot convert parameter 1 from
     'int' to 'const BigBuffer &'
     Reason: cannot convert from 'int' to 'const BigBuffer'
     Constructor for class 'BigBuffer' is declared 'explicit'
```

Raymond Chen

Follow