Answers to exercises – mismatching new/delete

devblogs.microsoft.com/oldnewthing/20040204-00

February 4, 2004

Raymond Chen

Answers to yesterday's exercises:

What happens if you allocate with scalar "new" and free with vector "delete[]"?
The scalar "new" will allocate a single object with no hidden counter. The vector "delete[]" will look for the hidden counter, which isn't there, so it will either crash (accessing nonexistent memory) or grab a random number and attempt to destruct that many items. If the random number is greater than one, you will start corrupting memory after the object. If the random number is zero, you fail to destruct anything. If the random number is exactly one, then the one object is destructed.

Next, the vector "delete[]" will attempt to free the memory block starting one size_t in front of the actual memory block. Depending on how the heap feels today, this may be detected as an invalid parameter and ignored, or this can result in heap corruption.

Final result: not good.

What happens if you allocate with vector "new[]" and free with scalar "delete"?
The vector "new[]" allocates several objects and stores the "howmany" in the hidden counter. The scalar "delete" destructs the first object in the vector. If it was a vector of zero objects, you corrupted memory. If it was a vector of two or more objects, then objects 2 an onward will not be destructed. (Result: Memory or other leak.)

Next, the scalar "delete" will free the memory block directly, which will fail because the memory block actually starts at the hidden size_t in front of the vector. This again corrupts the heap since you are freeing memory that is not a valid heap pointer.

Final result: also not good.

What optimizations can be performed if the destructor MyClass::~MyClass() is removed from the class definition?
If the class does not have a destructor, then no special work needs to be done when the vector is freed aside from freeing the memory. In this case, no hidden counter is necessary; the block can be allocated directly with no overhead and freed with no overhead.

More specifically, if the class has a trivial destructor (none of its base classes or sub-objects – if any – have a destructor), then the scalar and vector new/delete allocate and free the memory the same way, and mixing them does not generate a runtime error. You got lucky.

Of course, somebody might add a destructor to your class tomorrow, and then you won't be so lucky any more.

Note of course that all of this discussion assumes compiler behavior as described yesterday. That behavior is implementation-dependent so you should not rely on it. You may be lucky today, but the next version of the compiler may change the way it manages vectors and your luck will have run out.

Raymond Chen

Follow