Respect vector::reserve(request) Relative to Reallocation

Document number: N4524
Date: 2015-05-22
Project: Programming Language C++, Library Evolution Working Group
Reply-to: Nevin “☺” Liber, nliber@drw.com

Table of Contents

Introduction......1
Motivation and Scope......1
Impact On the Standard......2
Design Decisions......2
Technical Specifications......2
[vector.capacity]......2
[vector.bool]......2
Acknowledgements......3
References......3

Introduction

If `vector::reserve()` reallocates, respect the user’s capacity request.

Motivation and Scope

As part of the discussion around N4416, a question about `vector::reserve()` arose. If `vector::reserve()` has to reallocate, it may ask for more space than requested by the user. Because `vector` tracks its capacity independent of the allocator, there is no mechanism for `vector` to communicate with its allocator to take advantage of this freedom (such as to negotiate an optimal allocation size). It is also a non-expert user expectation that `vector` won’t waste memory when a specific capacity is requested.

libstdc++ (gcc 5.1.0), libc++ (Apple clang 3.4) and Visual Studio 2013 all currently just pass the requested size down to the allocator, so this is just standardizing existing practice.

Impact On the Standard

A two word deletion in the wording for `vector::reserve()` as well as introducing a section describing the current behavior for `vector<bool>::reserve()`.

Design Decisions

Because `vector<bool>` would have to do additional bookkeeping to accomplish this, so no behavior change to `vector<bool>` is being proposed.

While `std::basic_string` also has a `reserve()` member function, it has sufficiently different semantics that such changes are beyond the scope of this proposal.

If there are other implementations of `vector` that somehow know what allocator they are expected to be used with, they would no longer have the freedom to adjust the size requested from the user to their tightly coupled allocator.

Technical Specifications

Changes are relative to N4431:

[vector.capacity]

void reserve(size_type n);

Requires: T shall be MoveInsertable into *this.

Effects: A directive that informs a vector of a planned change in size, so that it can manage the storage allocation accordingly. After reserve(), capacity() is ~~greater or~~ equal to the argument of reserve if reallocation happens; and equal to the previous value of capacity() otherwise. Reallocation happens at this point if and only if the current capacity is less than the argument of reserve(). If an exception is thrown other than by the move constructor of a non-CopyInsertable type, there are no effects.

Complexity: It does not change the size of the sequence and takes at most linear time in the size of the sequence.

Throws: length_error if n > max_size().

Remarks: Reallocation invalidates all the references, pointers, and iterators referring to the elements in the sequence. No reallocation shall take place during insertions that happen after a call to reserve() until the time when an insertion would make the size of the vector greater than the value of capacity().

[vector.bool]

void flip() noexcept;

Effects: Replaces each element in the container with its complement.

void reserve(size_type n);

Effects: A directive that informs a vector of a planned change in size, so that it can manage the storage allocation accordingly. After reserve(), capacity() is greater or equal to the argument of reserve if reallocation happens; and equal to the previous value of capacity() otherwise. Reallocation happens at this point if and only if the current capacity is less than the argument of reserve(). If an exception is thrown, there are no effects. Complexity: It does not change the size of the sequence and takes at most linear time in the size of the sequence.

Throws: length_error if n > max_size().

Remarks: Reallocation invalidates all the references, pointers, and iterators referring to the elements in the sequence. No reallocation shall take place during insertions that happen after a call to reserve() until the time when an insertion would make the size of the vector greater than the value of capacity().

Acknowledgements

Thanks to Howard Hinnant for testing out the various production standard libraries, Rob Douglas for looking over the wording as well as Marshall Clow for helping with the title.

References

N4416 - Don't Move: Vector Can Have Your Non-Moveable Types Covered
N4431 - Working Draft, Standard for Programming Language C++