

Немного о Boost

Antony Polukhin
Полухин Антон

Boost libraries maintainer (DLL, LexicalCast, Any, TypeIndex, Conversion)
+ Boost.CircularBuffer, Boost.Variant

Содержание

- * Вести с полей разработки
- * Из Boost в C++ Standard
- * Проблемы
- * Полезности
- * Boost онлайн

Новости

- * AFIO — на доработке
- * Fiber — на доработке
- * Compute — в ближайшем релизе
- * DLL — в ближайшем релизе
- * Hana — в ближайшем релизе

Compute

```
// create a vector on the device
compute::vector<float> device_vector(host_vector.size(), context);

// transfer data from the host to the device
compute::copy(
    host_vector.begin(), host_vector.end(), device_vector.begin(),
queue
);

// calculate the square-root of each element in-place
compute::transform(
    device_vector.begin(),
    device_vector.end(),
    device_vector.begin(),
    compute::sqrt<float>(),
queue
);
```

DLL

```
#include <boost/dll.hpp>

auto cpp11_func = boost::dll::import<int(std::string&&)>(
    path_to_shared_library, "i_am_a_cpp11_function"
);

cpp11_func("Hello");
```

DLL

```
#include <boost/dll/smart_library.hpp>

dll::smart_library sm("libcpp.so");
auto ovl1 = sm.get_function<void(int)> ("overloaded");
auto ovl2 = sm.get_function<void(double)>("overloaded");
```

Hana

```
auto animals = hana::make_tuple(  
    Fish{"Nemo"}, Cat{"Garfield"}, Dog{"Snoopy"}  
);
```

```
// Access tuple elements with operator[] instead of std::get.  
Cat garfield = animals[1_c];
```

```
// Perform high level algorithms on tuples (this is like std::transform)  
auto names = hana::transform(animals, [](auto a) {  
    return a.name;  
});
```

В стандарт C++!



- * Filesystem
- * Thread
- * Variant
- * ASIO
- * optional
- * any
- * string_ref → string_view
- * and_/or_ →
conjunction/disjunction

Проблемы в Boost

- * Долго компилируется
- * Большой объём бинарных файлов

Долго компилируется

Мы исправляемся:

- * Уменьшение зависимостей
- * Variadic templates
- * Убрали поддержку старых компиляторов

Советы:

- * Современный компилятор
- * Spirit, GIL, Fusion в .cpp файлах



Большой объём бинарных файлов

С каждым годом лучше:

- * noexcept
- * variadic templates
- * SCARY iterators
- * -fvisibility=hidden

Советы:

RTTI on *
-fvisibility=hidden *
современный компилятор *

Полезности



Полезности

```
class task_type;

class work_queue {
    std::deque<task_type>    tasks_;
    boost::mutex                tasks_mutex_;
    boost::condition_variable    condition_variable_;

public:
    void push_task(const task_type& task) {
        boost::unique_lock<boost::mutex> lock(tasks_mutex_);
        tasks_.push_back(task);
        condition_variable_.notify_one();
    }

    // ...
    task_type pop_task();
};
```

Полезности

```
class task_type;

class work_queue {
    std::deque<task_type> tasks_;
    boost::mutex tasks_mutex_;
    boost::condition_variable condition_variable_;

public:
    void push_task(const task_type& task) {
        boost::unique_lock<boost::mutex> lock(tasks_mutex_);
        tasks_.push_back(task);
        lock.unlock(); // up to 2 times faster
        condition_variable_.notify_one();
    }

    // ...
    task_type pop_task();
};
```



Полезности

```
task_type pop_task() {
    boost::unique_lock<boost::mutex> lock(tasks_mutex_);
    while (tasks_.empty()) {
        condition_variable_.wait(lock);
    }

    task_type ret = tasks_.front();
    tasks_.pop_front();
    return ret;
}

void push_task(const task_type& task) {
    boost::unique_lock<boost::mutex> lock(tasks_mutex_);
    tasks_.push_back(task);
    lock.unlock();
    condition_variable_.notify_one();
}
```

Полезности

```
#include <vector>
#include <boost/shared_ptr.hpp>

auto foo() {
    std::vector< boost::shared_ptr<int> > res;

    for (unsigned i = 0; i < 1000; ++i)
        res.push_back(
            boost::shared_ptr<int>(new int)
        );
}

return res;
}
```

Полезности

```
#include <vector>
#include <boost/shared_ptr.hpp>

auto foo() {
    std::vector< boost::shared_ptr<int> > res;

    res.reserve(1000);
    for (unsigned i = 0; i < 1000; ++i)
        res.push_back(
            boost::shared_ptr<int>(new int)
        );
}

return res;
}
```



Полезности

```
#include <vector>
#include <boost/shared_ptr.hpp>

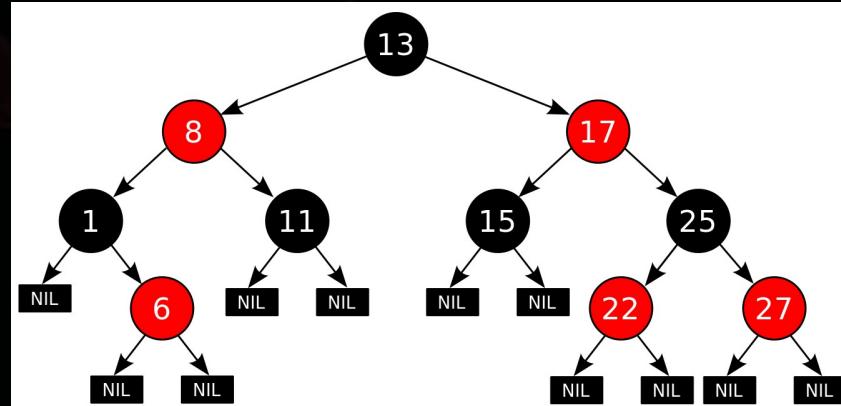
auto foo() {
    std::vector< boost::shared_ptr<int> > res;

    res.reserve(1000);
    for (unsigned i = 0; i < 1000; ++i)
        res.push_back(
            boost::make_shared<int>()
        );
}

return res;
}
```



Полезности: set<int>

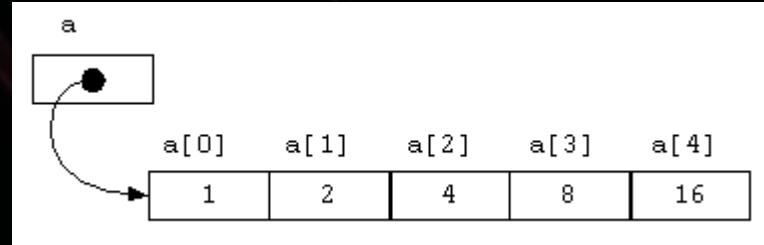


Выделенная под 1 нод память 64 Byte:



- | - вспомогательные данные аллокатора
- | - node's parent
- | - data
- | - неиспользованные байты кеша
- | - node's left / right

container::flat_set<int>



В одной кеш линии x86 64 Byte:



- | - вспомогательные данные аллокатора
- | - data

14 int в одной кеш линии

container::small_vector

```
#include <vector>
```

```
std::vector<int> data;
```

```
// Usually we get less than 16 ints
data.reserve(16);
get_some_data(data);
```

container::small_vector

```
#include <boost/container/small_vector.hpp>

// Usually we get less than 16 ints
boost::small_vector<int, 16> data;
get_some_data(data);
```

Полезности: Predef

```
#if !defined(__clang__)\n    && !defined(__ICC)\n    && !defined(__INTEL_COMPILER)\n    && (defined(__GNUC__)\n        || defined(__GNUG__))\n\n// GCC specific\n#endif
```

Полезности: Predef

```
#include <boost/predef/compiler.h>
```

```
#if BOOST_COMP_GNUC
```

```
// GCC specific
```

```
#endif
```



Boost онлайн

The screenshot shows a web browser window with the following details:

- Tab bar: Boost Application, std::shared_ptr, boost thread vers, Using and buildin.
- Address bar: en.cppreference.com/w/cpp/memory/shared_ptr/get
- Content area:
 - Section title: Example
 - Code editor:
 - Compiler: Run, Share, Exit, GCC 5.2 (C++14)
 - Powered by Coliru online compiler
 - Code listing (lines 1-24):

```
1 #include <deque>
2 #include <boost/function.hpp>
3 #include <boost/thread/mutex.hpp>
4 #include <boost/thread/locks.hpp>
5 #include <boost/thread/condition_variable.hpp>
6
7 class work_queue {
8 public:
9     typedef boost::function<void()> task_type;
10
11 private:
12     std::deque<task_type> tasks_;
13     boost::mutex tasks_mutex_;
14     boost::condition_variable cond_;
15
16 public:
17     void push_task(const task_type& task) {
18         boost::unique_lock<boost::mutex> lock(tasks_mutex_);
19         tasks_.push_back(task);
20         lock.unlock();
21         cond_.notify_one();
22     }
23
24     task_type try_pop_task() {
```
 - Compiler messages:
 - /tmp/ccRlXpT.o: In function `popper_sync':
 - main.cpp:(.text+0x2c5): undefined reference to `boost::detail::get_current_thread_data()'
 - main.cpp:(.text+0x45b): undefined reference to `boost::this_thread::interruption_point()'
 - main.cpp:(.text+0x468): undefined reference to `boost::system::system_category()'
 - /tmp/ccRlXpT.o: In function `boost::detail::thread_data<void (*)()>::~thread_data()':
 - main.cpp:(.text._ZN5boost6detail11thread_dataIPFvvEED2Ev[_ZN5boost6detail11thread_dataIPFvv
 - /tmp/ccRlXpT.o: In function `boost::detail::thread_data<void (*)()>::~thread_data()':

Compiler messages:

```
/tmp/ccRlXpT.o: In function `popper_sync':
main.cpp:(.text+0x2c5): undefined reference to `boost::detail::get_current_thread_data()'
main.cpp:(.text+0x45b): undefined reference to `boost::this_thread::interruption_point()'
main.cpp:(.text+0x468): undefined reference to `boost::system::system_category()'
/tmp/ccRlXpT.o: In function `boost::detail::thread_data<void (*)()>::~thread_data()':
main.cpp:(.text._ZN5boost6detail11thread_dataIPFvvEED2Ev[_ZN5boost6detail11thread_dataIPFvv
/tmp/ccRlXpT.o: In function `boost::detail::thread_data<void (*)()>::~thread_data()':
```

Boost онлайн

The screenshot shows a web browser window with the title bar "Boost Application" and several tabs open, including "std::shared_ptr<g>", "boost thread vers", and "Using and buildin". The main content area displays the URL "apolukhin.github.io/Boost-Cookbook-4880OS/#Chapter05-recipe3-part1". Below the URL, there is a snippet of C++ code and some explanatory text about tasks and mutexes.

an empty task if no tasks remain), and a method to post tasks.

Compile & Run

Run

Compile

Program arguments:

Compilation command:

```
g++ -Wall -DBOOST_THREAD_VERSION=4 main.cpp -lboost_thread -lboost_system
```

Output:

Compilation: SUCCESS. Program output:

Exit code: 0

Code (editable):

```
1 #include <deque>
2 #include <boost/function.hpp>
3 #include <boost/thread/mutex.hpp>
4 #include <boost/thread/locks.hpp>
5 #include <boost/thread/condition_variable.hpp>
6
7 class work_queue {
8 public:
9     typedef boost::function<void()> task_type;
10
11 private:
12     std::deque<task_type> tasks_;
13     boost::mutex tasks_mutex_;
14     boost::condition_variable cond_;
15
16 public:
17     void push_task(const task_type& task) {
18         boost::unique_lock<boost::mutex> lock(tasks_mutex_);
19         tasks_.push_back(task);
20         lock.unlock();
21         cond.notify_one();
22     }
23 }
```

The sidebar contains the following sections:

- Online Examples
- Boost Application Development Cookbook
- Boost C++ Application Development Cookbook
- Heading
- Chapters
- Recipe's Intro
- Compile & Run
- About

Boost онлайн

<http://apolukhin.github.io/Boost-Cookbook-4880OS/>



The image shows the front cover of the book 'Boost C++ Application Development Cookbook'. The cover has a dark background with red swirling smoke patterns. A large red rectangular stamp with the word 'ONLINE' in white, diagonally oriented, is placed across the center. Below the stamp, a thin green horizontal bar contains the text 'Quick answers to common problems'. The title 'Boost C++ Application Development Cookbook' is displayed prominently in white text at the bottom. At the very bottom, a smaller line of text reads 'Over 80 practical, task-based recipes to create applications using Boost libraries'. The bottom portion of the cover is a solid green color. In the bottom right corner of the slide, there is a small logo for 'PACKT PUBLISHING' with the tagline 'open source*'.

Antony Polukhin

[PACKT] open source*