Core библиотеки: идеи и примеры

Александр Еналдиев

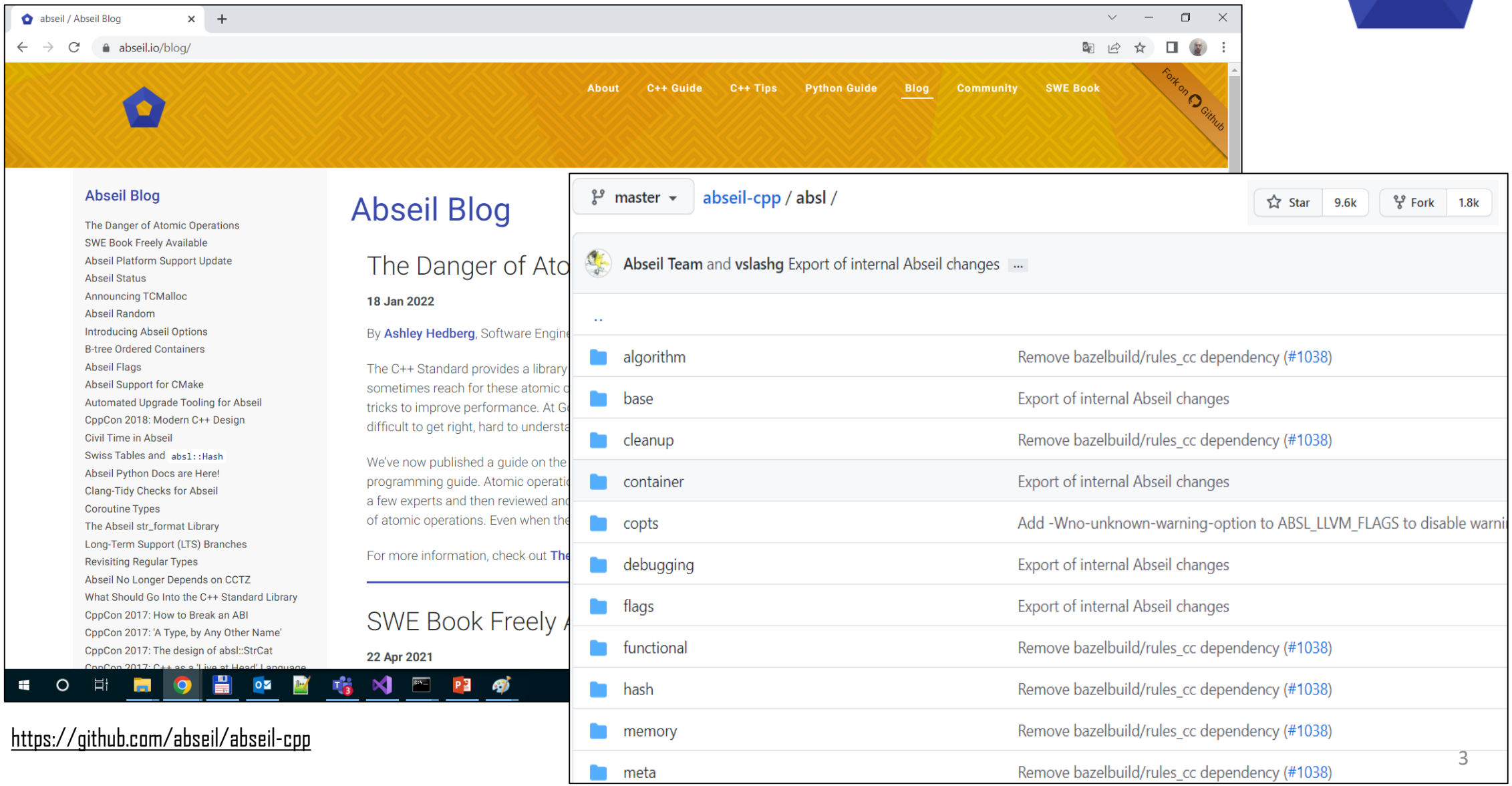
Kaspersky

folly:: absl:: ...

Содержание

- ABI (Application Binary Interface) и текущие подходы к решению проблем с ним ;
- Идеи при построении библиотек:
  - Сборка библиотеки ;
  - “Live at head” Principle ;
  - Решаемые задачи ;

Core библиотеки: Abseil



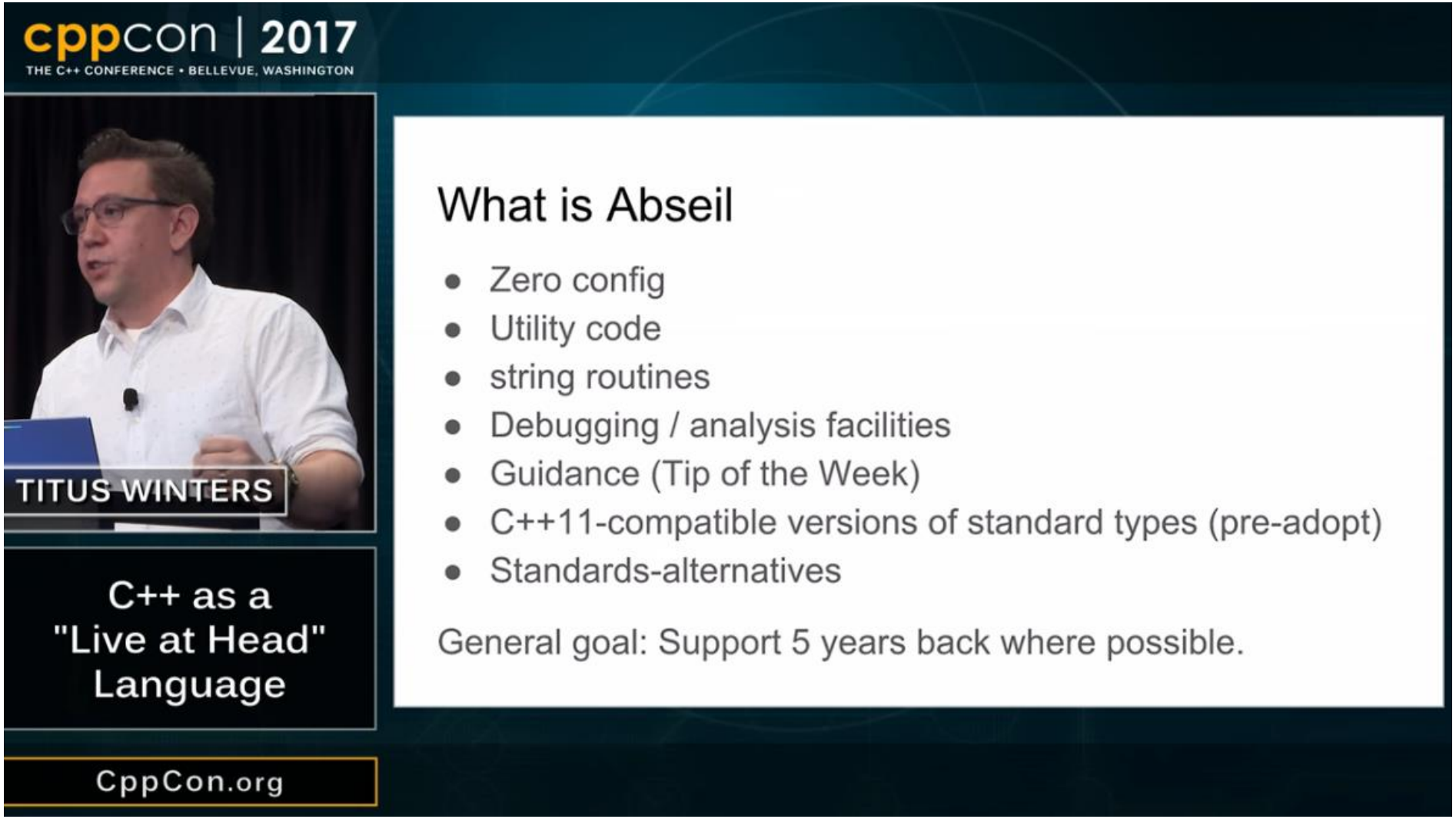
https://github.com/abseil/abseil-cpp

Core библиотеки: Abseil

Доклады:

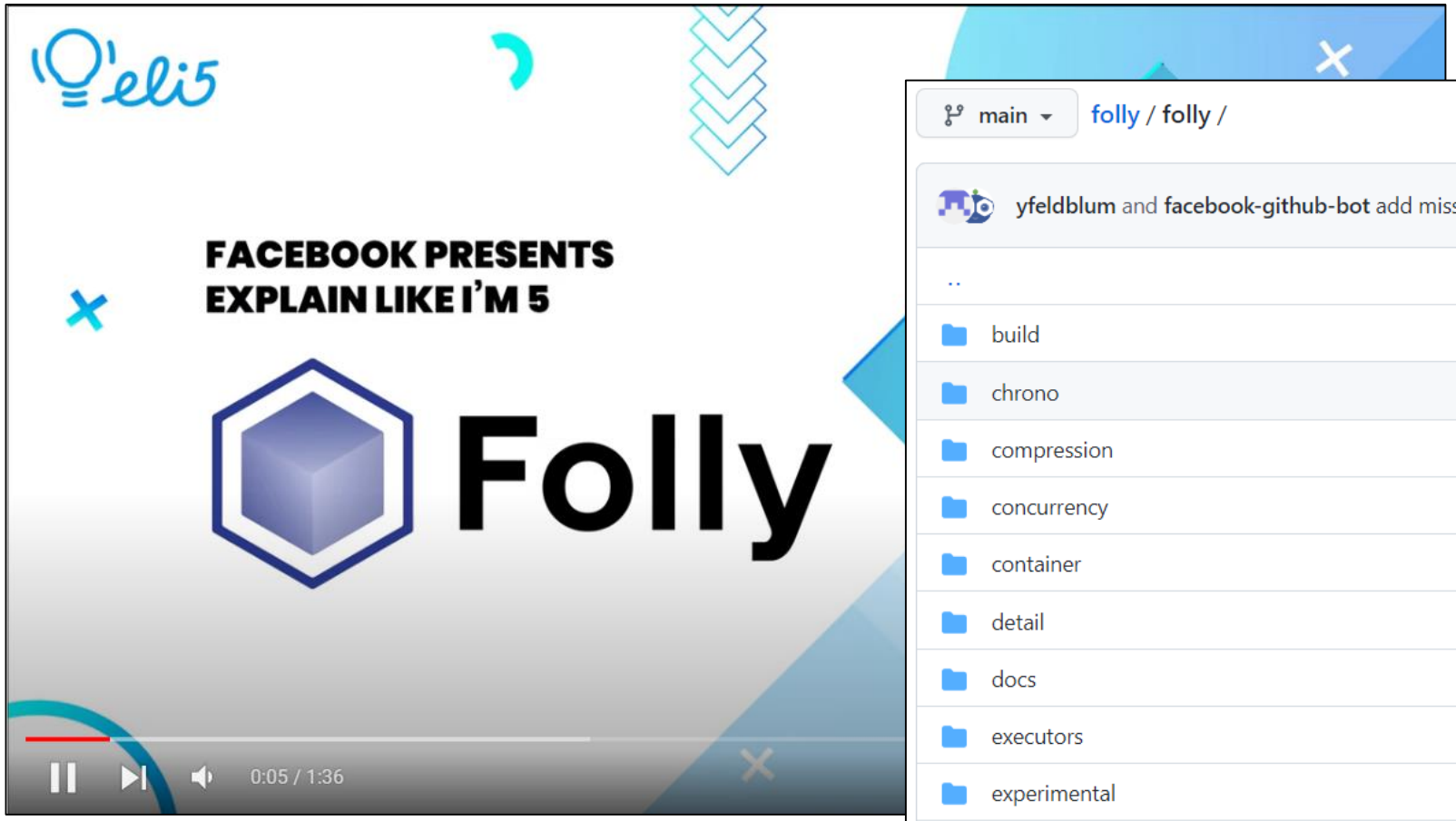
- CppCon 2017: Titus Winters “Hands-On With Abseil”
- CppCon 2017: Matt Kulukundis “Designing a Fast, Efficient, Cache-friendly Hash Table, Step by Step”
- CppCon 2018: Titus Winters “Modern C++ Design”
- CppCon 2019: Titus Winters “Maintainability and Refactoring Impact of Higher-Level Design Features”
- CppCon 2019: Matt Kulukundis “Abseil's Open Source Hashtables: 2 Years In”
- …

Core библиотеки: Abseil



https://www.youtube.com/watch?v=tlSy7EJQPzI

Core библиотеки: Folly



main folly / folly / Fork 4.6k Star 21.9k

yfeldblum and facebook-github-bot add missing sanitizer suppression in MicroLock ...

..	
build	use gmock 1.10 instead of 1.8
chrono	Update copyright headers from Facebook to Meta
compression	Expose Method to Clear Compression Context Pools
concurrency	folly::ConcurrentHashMap assign_if
container	let small_vector handle MaxInline = 0 under MSVC (#1775)
detail	Remove -inl.h include cycle from folly/detail
docs	Add an Rcu.md documentation page for folly RCU
executors	Implement step() in folly Manual Executor
experimental	Provide a Default Initializer for the Huge Page Allocator
ext	folly::ext::test_find_resource
external	update farmhash.cpp to avoid an unused function error on an

https://github.com/facebook/folly

Core библиотеки: Folly

What is `folly`?

Folly (acronymed loosely after Facebook Open Sourc
designed with practicality and efficiency in mind. **Foll
components used extensively at Facebook**. In partic
other open source C++ efforts and place where thos

It complements (as opposed to competing against) c
In fact, we embark on defining our own component c
available, or does not meet the needed performance
`std` or Boost obsoletes them.

Performance concerns permeate much of Folly, some
would otherwise be (see e.g. `PackedSyncPtr.h`, `Smal`
in all of Folly.

main ▾ folly / folly /

Fork 4.6k Star 21.9k

yfeldblum and facebook-github-bot add missing sanitizer suppression in MicroLock ...

..

build	use gmock 1.10 instead of 1.8
chrono	Update copyright headers from Facebook to Meta
compression	Expose Method to Clear Compression Context Pools
concurrency	folly::ConcurrentHashMap assign_if
container	let small_vector handle MaxInline = 0 under MSVC (#1775)
detail	Remove -inl.h include cycle from folly/detail
docs	Add an Rcu.md documentation page for folly RCU
executors	Implement step() in folly Manual Executor
experimental	Provide a Default Initializer for the Huge Page Allocator
ext	folly::ext::test_find_resource
external	update farmhash.cpp to avoid an unused function error on an

https://github.com/facebook/folly

Core библиотеки: Folly

Доклады:

- CppCon 2016: Nicholas Ormrod “The strange details of std::string at Facebook“

- CppCon 2016: David Watson “Experiences with Facebook's C++ library“

Core библиотеки: The ABI

```
struct IntAndString
{
    int i;
    std::string s;
};
```

```
template<>
struct std::hash<std::string> {
    std::size_t operator()(std::string const& s) const noexcept
    {
        return _Fnv1a_append_bytes(std::_FNV_offset_basis,
                                   &*s.begin(), s.size());
    }
};
```



```
template<>
struct std::hash<std::string> {
    std::size_t operator()(std::string const& s) const noexcept
    {
        boost::crc_32_type result;
        result.process_bytes(s.data(), s.size());
        return static_cast<std::size_t>(result.checksum());
    }
};
```

Core библиотеки: The ABI

- N4028 Defining a Portable C++ ABI ; (Herb Sutter)

  "…The programmer needs to be able to distinguish between code that uses "the usual std:: library" (whatever that is today, which usually is different for different C++ implementations on the same platform and could even be a user-provided standard library if they are not using the one that came with their compiler) and code that uses the target platform's C++ standard library ABI. C++ already has a way to designate the C++ standard library types and functions: namespace std.

  For symmetry with extern "abi", we propose the namespace std::abi ... This gives us two distinct namespaces for two distinct thing:

  - ***std*** contains "the C++ standard library implementation that can change/evolve." This is provided by each C++ implementer, exactly as today.
  - ***std::abi*** contains "the C++ standard library implementation that is binary stable." This is provided by each OS platform, however it wants, and is shared by all compilers that target that platform. A likely choice is to make this the release build of a snapshot of the OS platform owner's own C++ product's implementation of std, taken at the time they support the C++ ABI.

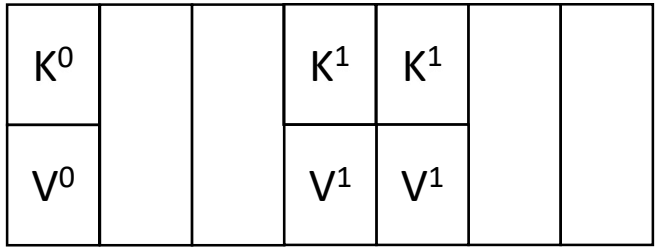
  For convenience and safety, in an extern "abi" block std:: means std::abi:: …"

Core библиотеки: The ABI

- N4028 Defining a Portable C++ ABI ; (Herb Sutter)
- P1863 ABI - Now Or Never ;
- P2028 What is ABI, and What Should WG21 Do About It? ; (Titus Winters)
  - std::hash, std::unordered_map impl;
  - std::regex impl ;
  - lock_guard vs scoped_lock ;
  - push_back vs emplace_back ;
  - …

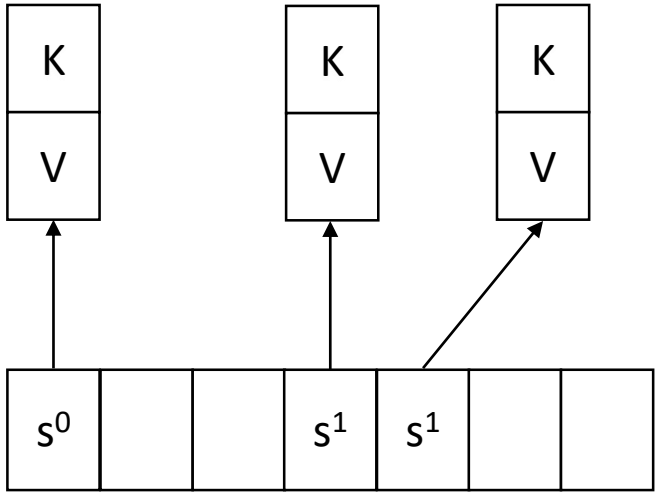
Core библиотеки: The ABI

- absl::Hash, absl::HashOf ;
- absl::flash_hash_set / absl::flat_hash_map ;

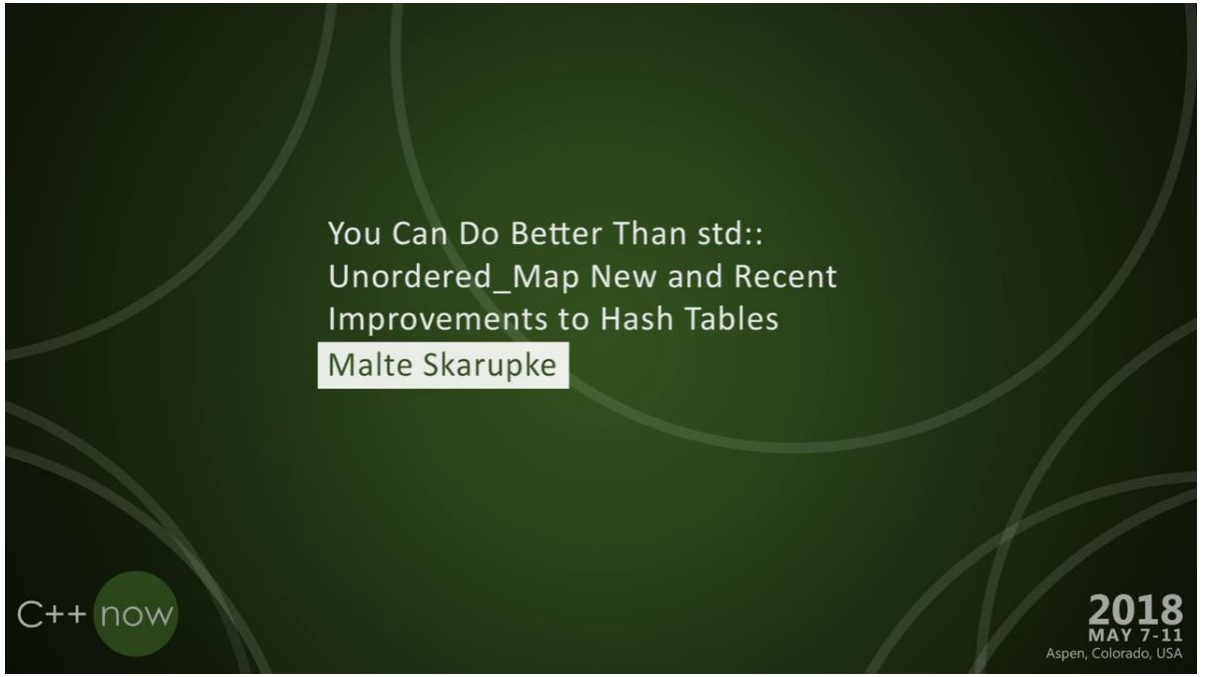


- absl::node_hash_set

  absl::node_hash_map ;



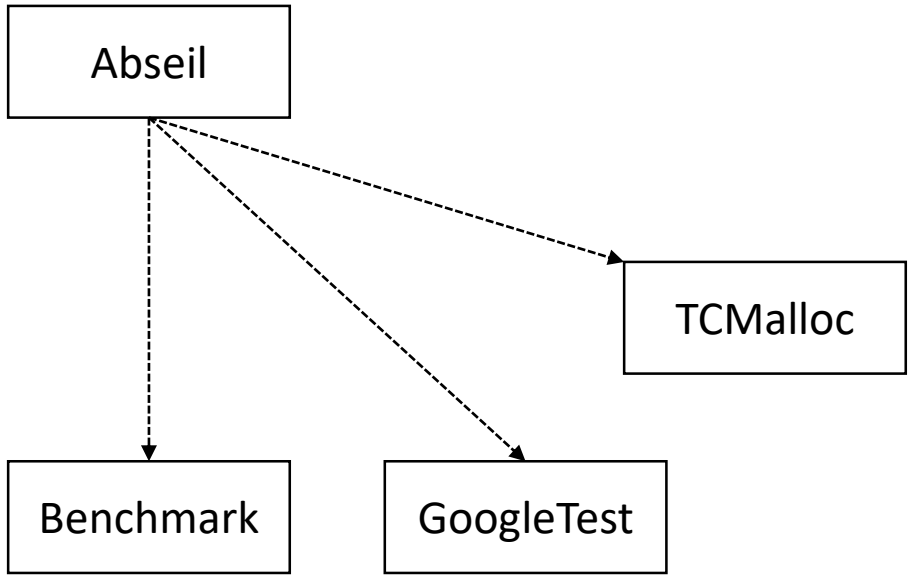

```
struct A {
  int m;

  template <typename H>
  friend H AbslHashValue(H state, const A& v) {
    return H::combine(std::move(state), v.m);
  }
};
```



https://www.youtube.com/watch?v=M2fKMP47slQ

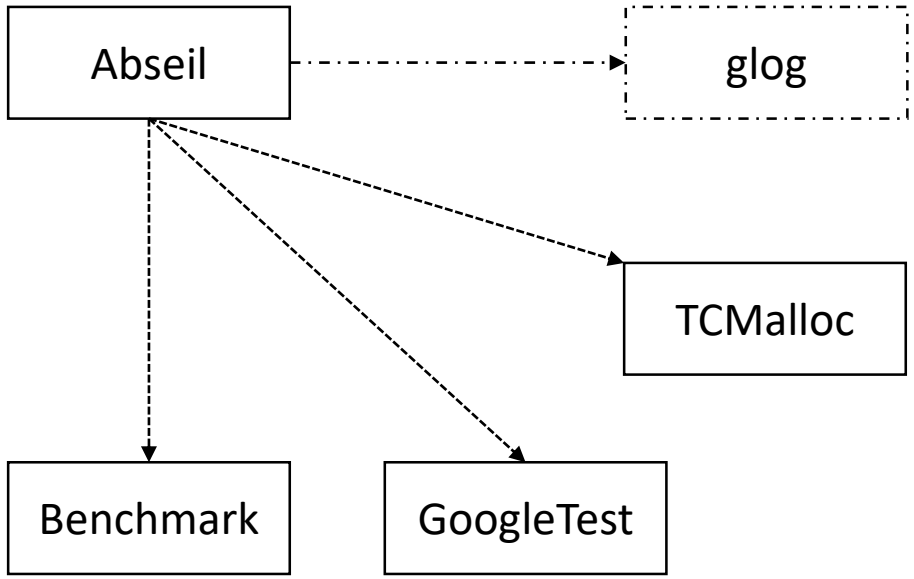
Core библиотеки: Идеи

- Модульность



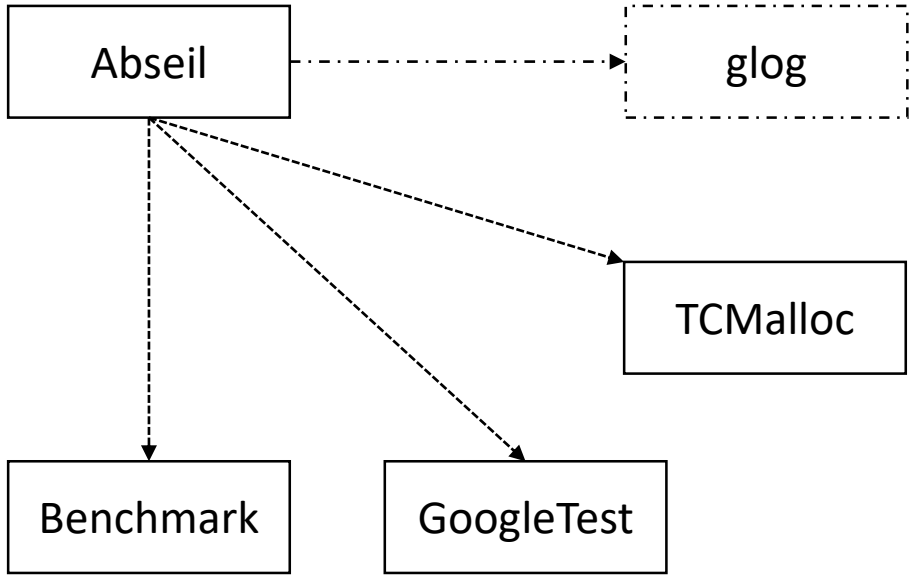
Core библиотеки: Идеи

- Модульность



Core библиотеки: Идеи

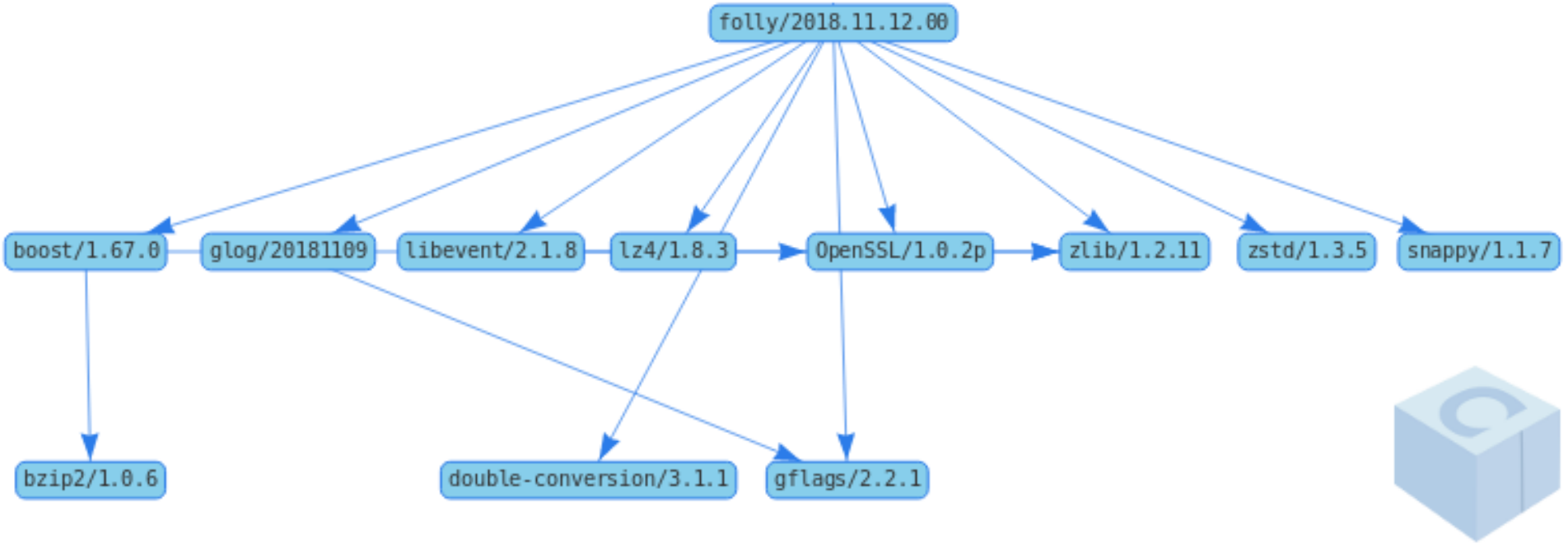
- Модульность



- Core lib
  - Allocator ;
  - Logging ;
  - Container ;
  - Util ;
- Tiny lib
  - Has logging?!;

Core библиотеки: Идеи

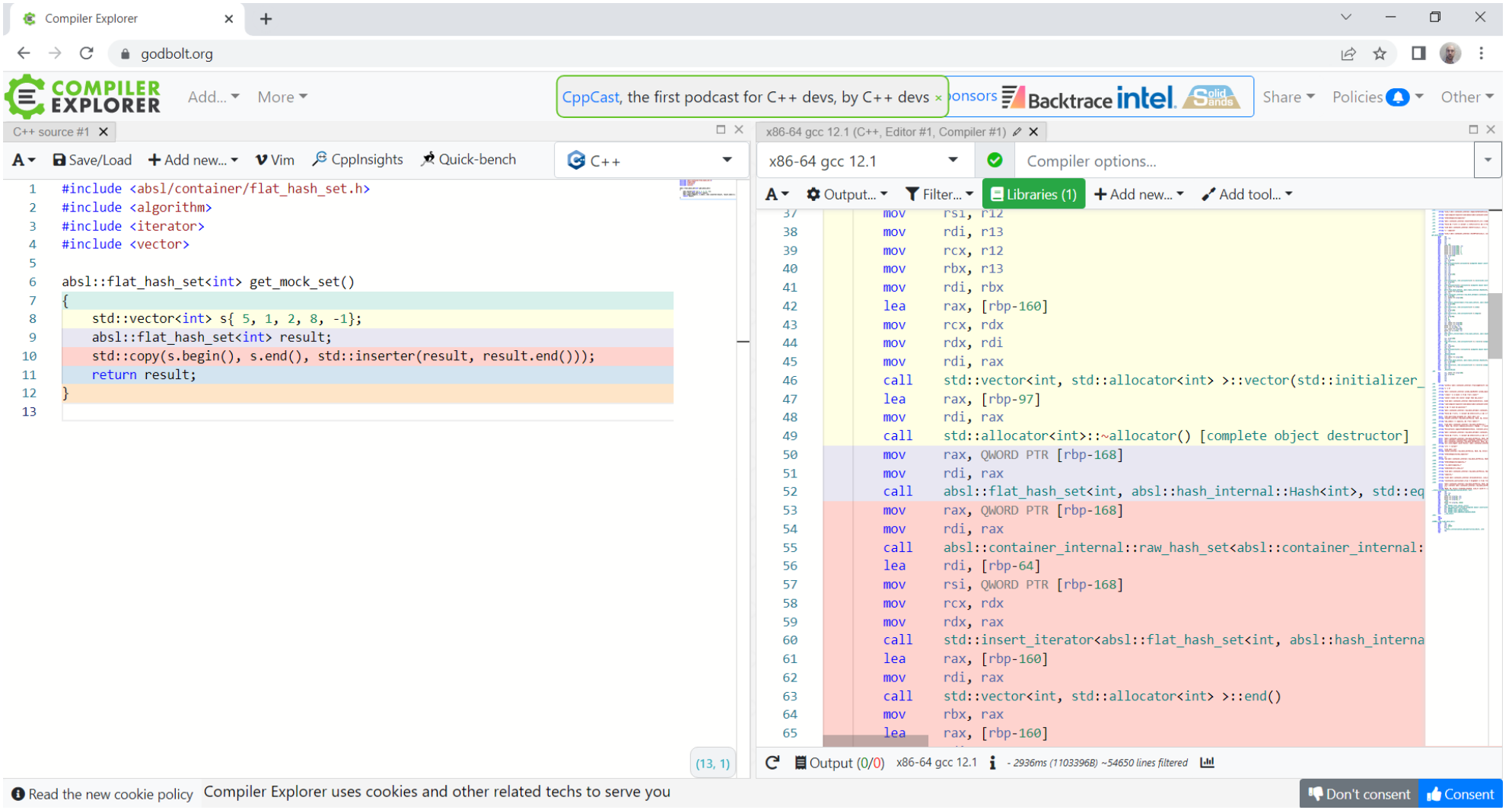
- Модульность



https://blog.conan.io/2018/12/03/Using-Facebook-Folly-with-Conan.html

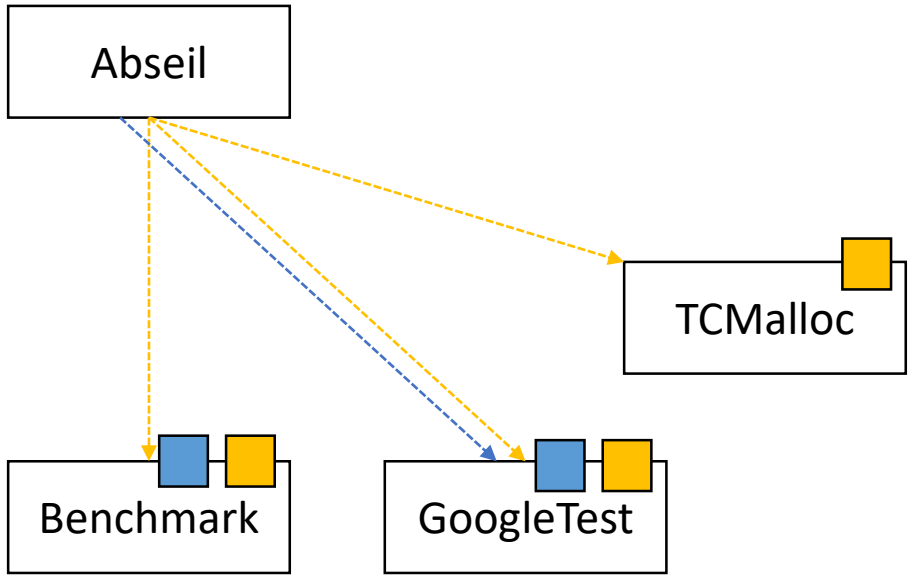
Core библиотеки: Идеи

- Модульность



Core библиотеки: Идеи

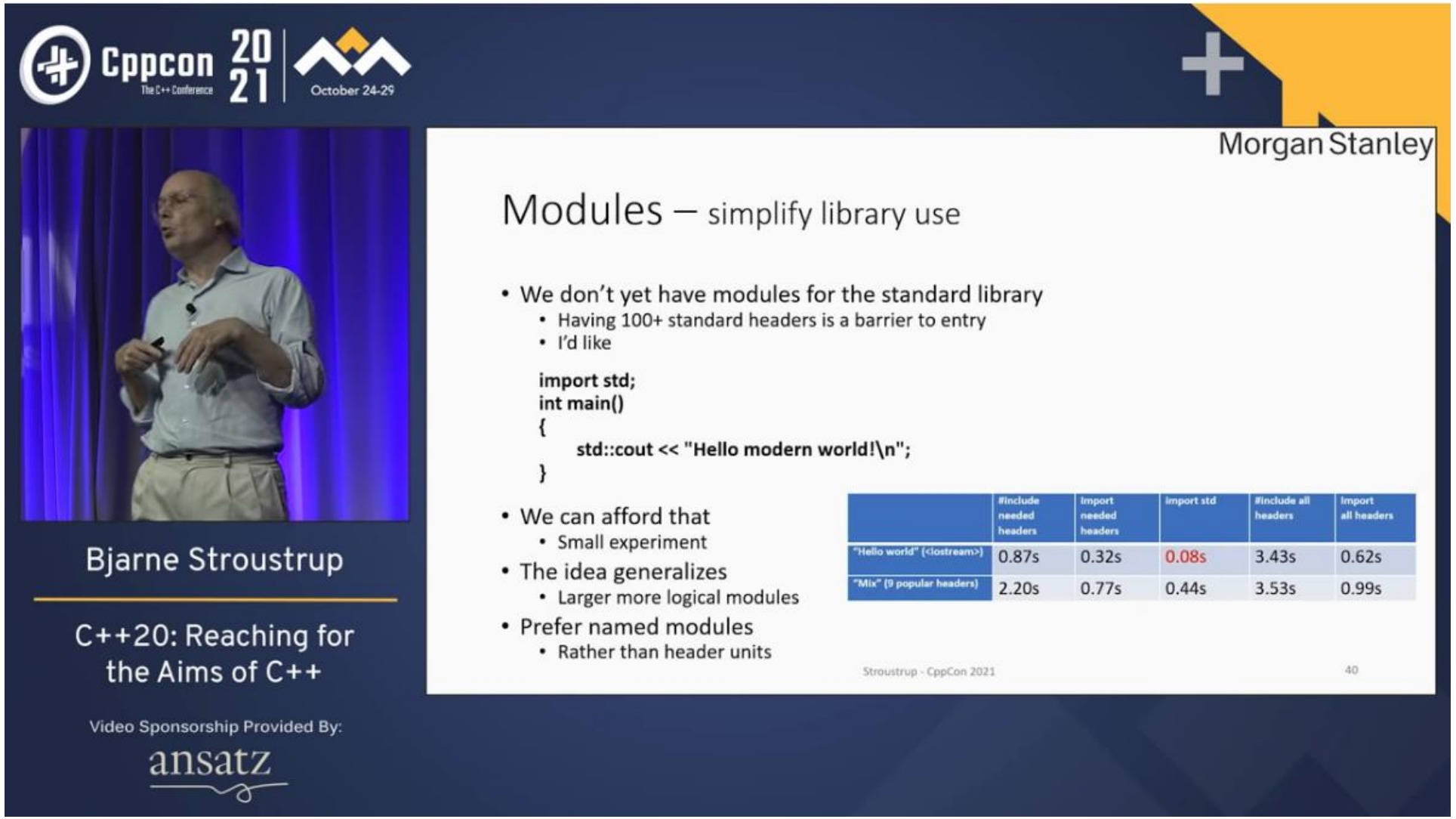
- Сборка из сорцов



CMake

Bazel

Core библиотеки: Идеи



https://www.youtube.com/watch?v=15QF2q66NhU

Core библиотеки: Идеи

- Менеджеры пакетов

vcpkg

CONAN
C/C++ Package Manager

HomeBrew

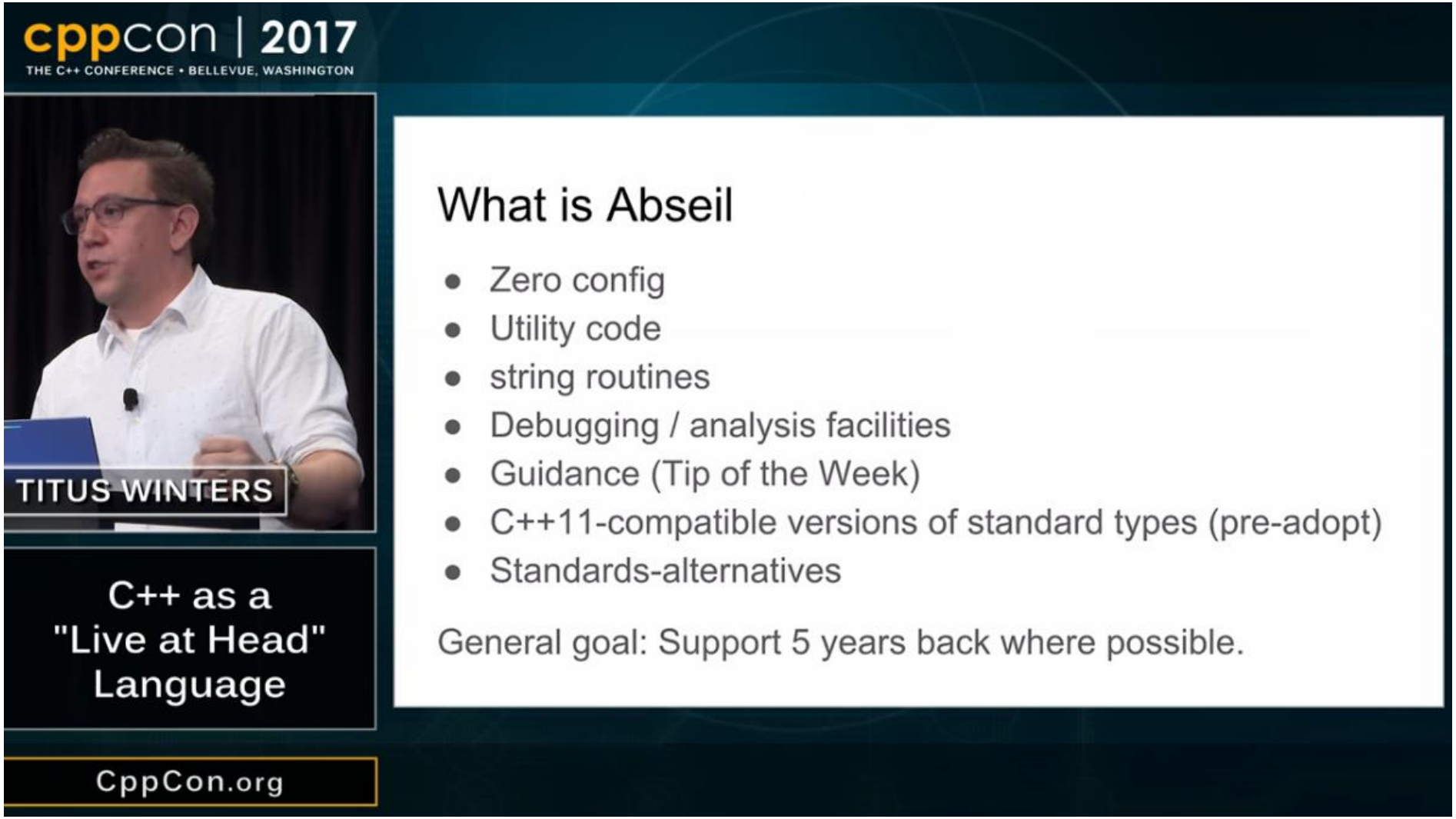
MacPorts

Core библиотеки: Идеи

Опрос: Как быстро вы начинаете использовать новинки в боевом коде?

- Еще до попадания в стандарт ;
- Как только попало в стандарт ;
- Как только в следующем стандарте пофиксят баги ;

Core библиотеки: Идеи



https://www.youtube.com/watch?v=tISy7EJQPzI

Core библиотеки: Идеи

- Live at Head Principle

Long-Term Support (LTS) Branches

By Tom Manshreck, Abseil Tech Writer

Abseil encourages developers to “live at head” but we understand that philosophy may not work for everyone. We are therefore providing snapshots of the Abseil codebase. These snapshots are available as “Long Term Support” (LTS) branches of Abseil, and **we intend to provide a new snapshot every 6 months or so**.

We pledge to support these LTS snapshots **for at least 2 years**. If critical bug fixes, such as security issues, require us to change Abseil, we will also change them within any supported LTS snapshot.

NOTE: we don’t want you to think of these snapshots as “versions.” They are simply a snapshot of the codebase at a specific point in time. If you cannot build from source or otherwise live at head, prefer to use the latest LTS branch of Abseil instead.

https://abseil.io/blog/20180618-lts-branches

Core библиотеки: Идеи

- Live at Head Principle

Releases | Tags

Find a release

03 Nov 2021

derekmauro

20211102.0

2151058

Compare

Abseil LTS branch, Nov 2021 Latest

Abseil LTS 20211102

What's New:

- `absl::Cord` is now implemented as a b-tree. The new implementation offers improved performance in most workloads.
- `absl::SimpleHexAtoi()` has been added to `strings` library for parsing hexadecimal strings.

Breaking Changes:

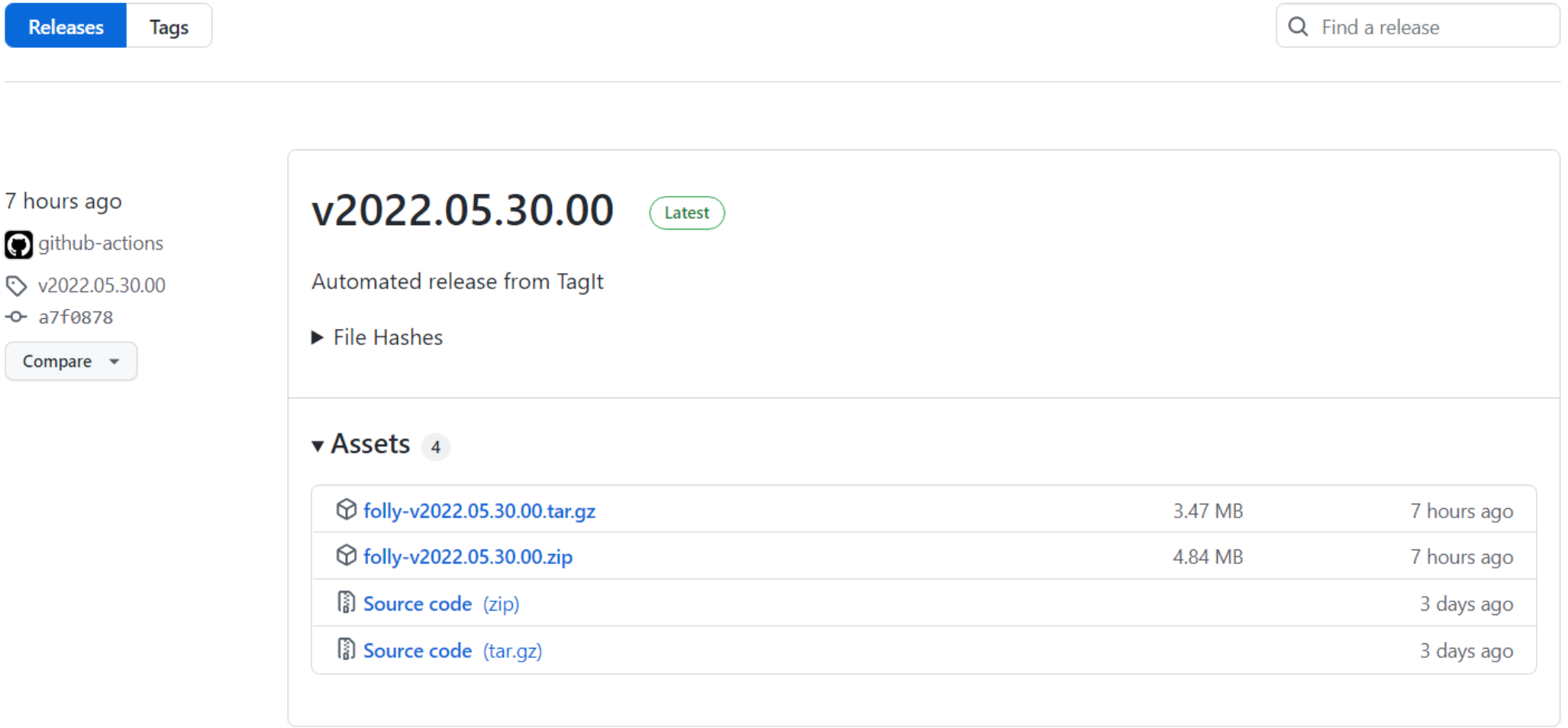
- Bazel builds now depend on the bazelbuild/platforms repository. See Abseil's WORKSPACE file for an example of how to add this dependency.

Baseline: `2151058`

https://github.com/abseil/abseil-cpp/releases

Core библиотеки: Идеи

- Live at Head Principle

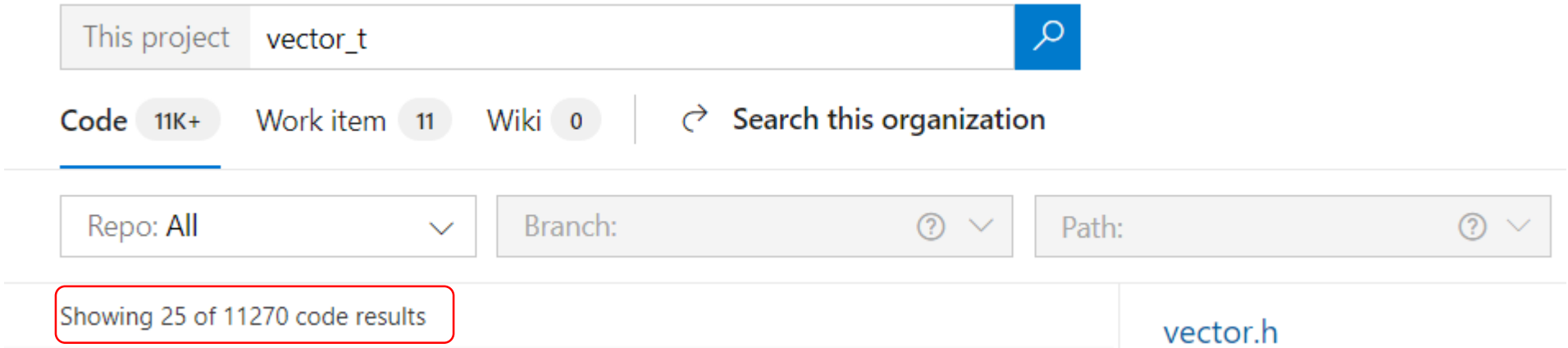


https://github.com/facebook/folly/releases

А что с опросом?

Core библиотеки: Идеи

- Live at Head Principle



Core библиотеки: Идеи

- Live at Head Principle

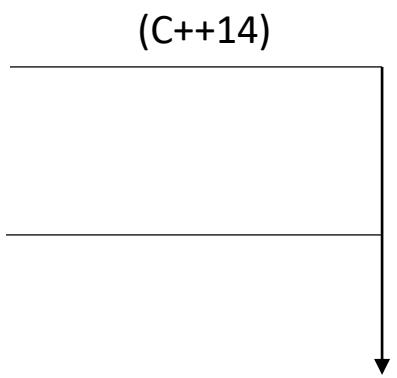
```
int http_get(std::string const& uri);
```

```
int http_get(std::string const& uri, std::vector<std::uint8_t>& data);
```

Core библиотеки: Идеи

- Live at Head Principle

```
[[deprecated]]
int http_get(std::string const& uri);

// deprecated
int http_get(std::string const& uri);
```




```
int http_get(std::string const& uri, std::vector<std::uint8_t>& data);
```

Core библиотеки: Идеи

- Live at Head Principle

  Clang Tidy

```
// Original - Comparison in the integer domain
int x;
absl::Time t;
if (x < absl::ToUnixSeconds(t)) ...

// Suggested - Compare in the absl::Time domain instead
if (absl::FromUnixSeconds(x) < t) ...
```

Core библиотеки: Идеи

- Live at Head Principle

  Clang Tidy

```
// Original - Comparison in the integer domain
int x;
absl::Time t;
if (x < absl::ToUnixSeconds(t)) ...

// Suggested - Compare in the absl::Time domain
instead
if (absl::FromUnixSeconds(x) < t) ...
```

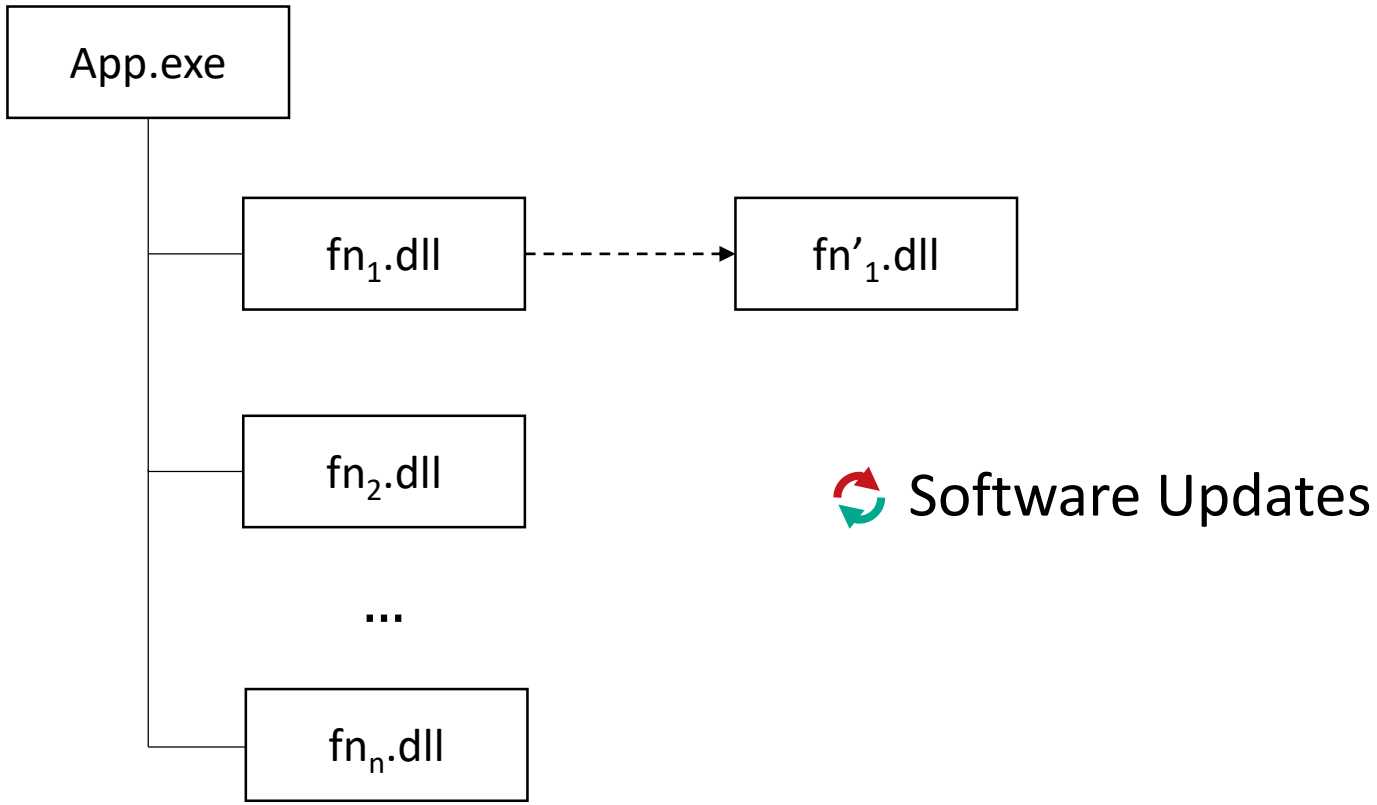
https://clang.llvm.org/extra/clang-tidy/

Clang-Tidy Checks

Name	Offers fixes
abseil-cleanup-ctad	Yes
abseil-duration-addition	Yes
abseil-duration-comparison	Yes
abseil-duration-conversion-cast	Yes
abseil-duration-division	Yes
abseil-duration-factory-float	Yes
abseil-duration-factory-scale	Yes
abseil-duration-subtraction	Yes
abseil-duration-unnecessary-conversion	Yes
abseil-faster-strsplit-delimiter	Yes
abseil-no-internal-dependencies	
abseil-no-namespace	
abseil-redundant-strcat-calls	Yes
abseil-str-cat-append	Yes
abseil-string-find-startswith	Yes
abseil-string-find-str-contains	Yes
abseil-time-comparison	Yes
abseil-time-subtraction	Yes
abseil-upgrade-duration-conversions	Yes
altera-id-dependent-backward-branch	
altera-kernel-name-restriction	

Core библиотеки: Идеи

- Цели создания: Boost vs Abseil vs ?



Core библиотеки: Источник вдохновения

...

Спасибо

Core библиотеки: Ещё

folly/fbvector

"...The initial HP implementation by Stepanov used a growth factor of 2; i.e., whenever you'd push_back into a vector without there being room, it would double the current capacity. This was not a good choice: it can be mathematically proven that a growth factor of 2 is ***rigorously the worst possible*** because it never allows the vector to reuse any of its previously other compilers ***reducing the growth factor to 1.5***, maintained its factor of 2. This makes std::vector cac manager unfriendly..." © github folly

folly::fbVector – улучшенный std::vector от Facebook (https://habr.com/ru/company/infopulse/blog/238131/)

```cpp
#include <iostream>
#include <iterator>
#include <algorithm>

#include <folly/FBVector.h>

int main(int argc, char** argv)
{
    folly::fbvector<int> vec;
    for (int ix = 1; ix <= 4096; ++ix)
    {
        vec.emplace_back(ix); // hello, jemalloc
    }
    copy(vec.begin(), vec.end(), std::ostream_iterator<int>(std::cout, " "));

    return 0;
}
```

Core библиотеки: Ещё

folly/Thread pools & Executors ?

How do I use the thread pools?
Wangle provides two concrete thread pools (IOThreadPoolExecutor, CPUThreadPoolExecutor) as well as building them in as part of a complete async framework…or maybe you need to construct a thrift/memcache client, and need an event base:

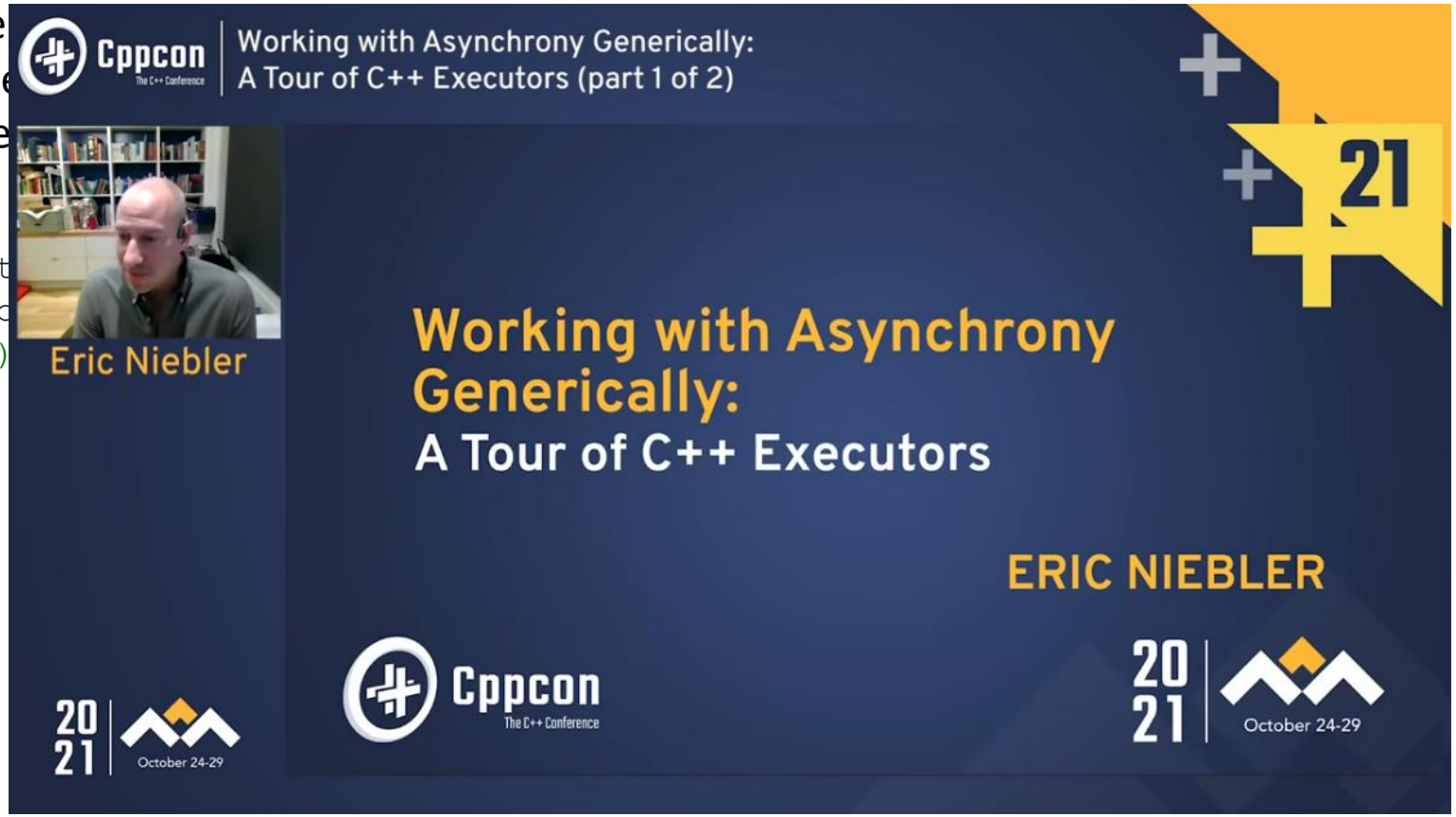
```
auto f = getClient(getIOExecutor()->getEventBase())->callSomeFunction(args...)
    .via(getCPUExecutor())
    .then([](Result r){ .... do something with result});
```

Core библиотеки: Ещё

folly/Thread pools & Executors ?

How do I use the thread pools?
Wangle provides two concrete
CPUThreadPoolExecutor) as we
framework…or maybe you nee
base:

```
auto f = getClient(get
    .via(getCPUExecuto
    .then([](Result r)
```



https://www.youtube.com/watch?v=xLboNlf7BTg