Study Group 14: The Story So Far

John McFarlane

Happy Birthday Jimmy!

John McFarlane, Jaguar Land Rover, Shannon, County Clare, Ireland

jaguarlandrovercareers.com/ireland Ireland

4 Years Ago

- Entire industries of professional C++ users missing from committee.
- With poor awareness of the state of modern C++
- Lack of recognition of where the zero-overhead principle is not followed.
- Implementers providing leaner language subsets to games, embedded etc..

CppCon 2014: "Grill the Committee"

10,315 views

89 2 SHARE

Study Group 14

unofficial-real-time-cxx ›

Welcome to SG14

1 post by 1 author

Michael Wong 12/06/2015

Other recipients: sg...@isocpp.org

Hi all, my name is Michael Wong. I am the current chair of SG14 for C++ Standard charged with improving Games Development and Low latency for C++.

I like to invite everyone to begin participating in this group as the official ISO C++ SG14 group. This group is now linked from the ISOCPP forum lists which contains links to all the other Evolution WG Study groups (for instance I also chair SG5 Transactional Memory)

https://isocpp.org/forums

I am not saying we should stop talking on unofficial-real-time-cxx as it has become a real hit. But as time goes by, I think all the relevant conversations should occur here but we should maintain references to unofficial-real-time-cxx

Please sign up. Thank you.

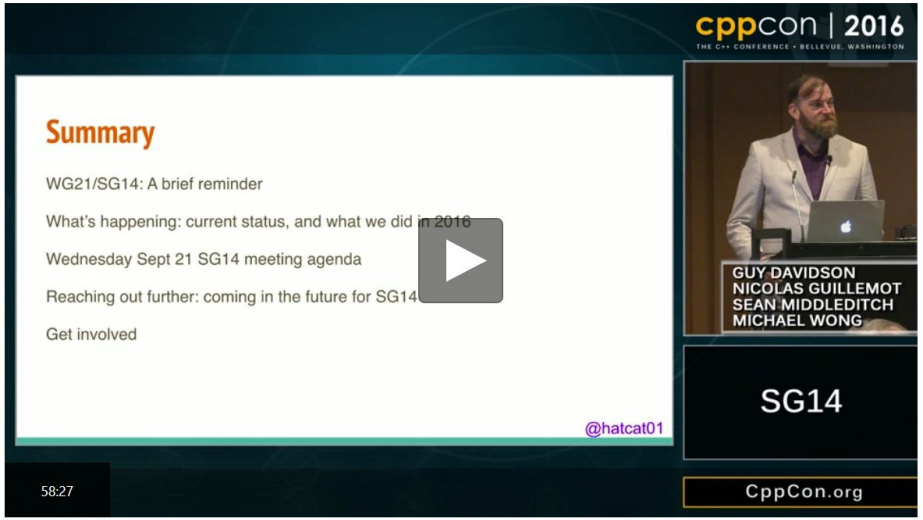
Click here to Reply

CppCon 2016

CppCon 2016: "WG21-SG14 – Making C++ better for games, embedded and financial developers"

Nov 02, 2016 at 5:00PM by Michael Wong, Nicolas Guillemot, Sean Middleditch, Guy Davidson

★★★★★ 0 ratings



SG14 - Game Development & Low Latency

- Game Development
  - Maintaining consistent framerate,
  - reducing HID latency
- High-Frequency Trading (HTF)
  - Rapid turnaround of incoming information
- Embedded Systems
  - Limited resources,
  - hard real-time constraints

Some Objectives

- Do study group stuff
  - centralize low-latency expertise
  - collaborate on committee work with a shared aim
  - be a sorting house for proposals
- Represent missing C++ users
  - industries who don’t traditionally interact with the community
  - developers who have difficulty attending committee meetings
- Get the modern C++ message to those users
- Relay their grievances to the committee

Some Objectives

- Do study group stuff
  - centralize low-latency expertise
  - collaborate on committee work with a shared aim
  - be a sorting house for proposals
- Represent missing C++ users
  - industries who don’t traditionally interact with the community
  - developers who have difficulty attending committee meetings
- **Get the modern C++ message to those users**
- **Relay their grievances to the committee**

Topics of Interest

- Build systems & package management
- Containers
- Error reporting
- Feature tests
- Free-standing implementation
- Graphics
- Low-level APIs
- Numerics
- Parallelism
- Performance hints

How We Collaborate

- Forum
- Telecons
- Face-to-face Meetings
- GitHub

SG14

Chair: **Michael Wong**

Co-chair: **John McFarlane**

Embedded: **Ben Craig, Wooter Van Ooijen and Odin Holmes**

Games: **John McFarlane, Guy Davidson and Paul Hampson**

Finance: **Carl Cooke, Neal Horlock, Mateusz Pusz and Clay Trychta**

SG14

Chair: **Michael Wong**

Co-chair: **John McFarlane**

Embedded: **Ben Craig, Wooter Van Ooijen and Odin Holmes**

Games: **John McFarlane, Guy Davidson and Paul Hampson**

Finance: **Carl Cooke, Neal Horlock, Mateusz Pusz and Clay Trychta**

and ...

SG14

Chair: **Michael Wong**

Co-chair: **John McFarlane**

Embedded: **Ben Craig, Wooter Van Ooijen and Odin Holmes**

Games: **John McFarlane, Guy Davidson and Paul Hampson**

Finance: **Carl Cooke, Neal Horlock, Mateusz Pusz and Clay Trychta**

and ...

Get Involved!

- Learn more about the committee: https://isocpp.org/std
- Subscribe to the forum: https://groups.google.com/a/isocpp.org/forum/#!forum/sg14
  - Air your grievances with ISO C++
  - Suggest new features or fixes
  - Join the telecon
- Clone the repos: https://github.com/WG21-SG14/SG14
  - Test the code and suggest improvements
- Meet up
  - Marriott Courtyard, 8:30am Wednesday (Tomorrow)
  - Observe a real live ISO C++ study group in the wild

[[likely]], [[unlikely]]

Document number: P0479R0
Date: 2016-10-16
Audience: Evolution Working Group, SG14
Reply-to: Clay Trychta <clay.trychta@gmail.com>

Attributes for Likely and Unlikely Branches

I. Table of Contents

II. Introduction

Two new attributes `[[likely]]` and `[[unlikely]]` are proposed. These attributes will serve as hints on the likelihood that a subsequent branch is taken. Compilers may use these hints to improve the code they generate in various ways.

III. Motivation and Scope

Compiler's optimizers often have no information relating to branch probability which can lead to suboptimal code generation. In many cases the excellent dynamic branch predictors on modern processors can make up for this lack of information. However, in some cases code may execute more slowly than necessary even though the programmer knew the probability of particular branches being executed because they did not have an easy way to communicate this to the compiler.

Several existing compilers including GCC and Clang implement `__builtin_expect` which

9 Declarations [dcl.dcl]

9.11 Attributes [dcl.attr]

9.11.7 Likelihood attributes [dcl.attr.likelihood]

1 The *attribute-token*s `likely` and `unlikely` may be applied to labels or statements. The *attribute-token*s `likely` and `unlikely` shall appear at most once in each *attribute-list* and no *attribute-argument-clause* shall be present. The *attribute-token* `likely` shall not appear in an *attribute-specifier-seq* that contains the *attribute-token* `unlikely`.

2 [*Note:* The use of the `likely` attribute is intended to allow implementations to optimize for the case where paths of execution including it are arbitrarily more likely than any alternative path of execution that does not include such an attribute on a statement or label. The use of the `unlikely` attribute is intended to allow implementations to optimize for the case where paths of execution including it are arbitrarily more unlikely than any alternative path of execution that does not include such an attribute on a statement or label. A path of execution includes a label if and only if it contains a jump to that label. Excessive usage of either of these attributes is liable to result in performance degradation. — *end note*]

3 [*Example:*

```
void g(int);
int f(int n) {
  if (n > 5) [[unlikely]] {      // n > 5 is considered to be arbitrarily unlikely
    g(0);
    return n * 2 + 1;
  }

  switch (n) {
  case 1:
    g(1);
    [[fallthrough]];

  [[likely]] case 2:             // n == 2 is considered to be arbitrarily more
    g(2);                        // likely than any other value of n
    break;
  }
  return 3;
}
```

— *end example*]

SG14

Forum: https://groups.google.com/a/isocpp.org/forum/#!forum/sg14

Thank You!

@JSAMcFarlane