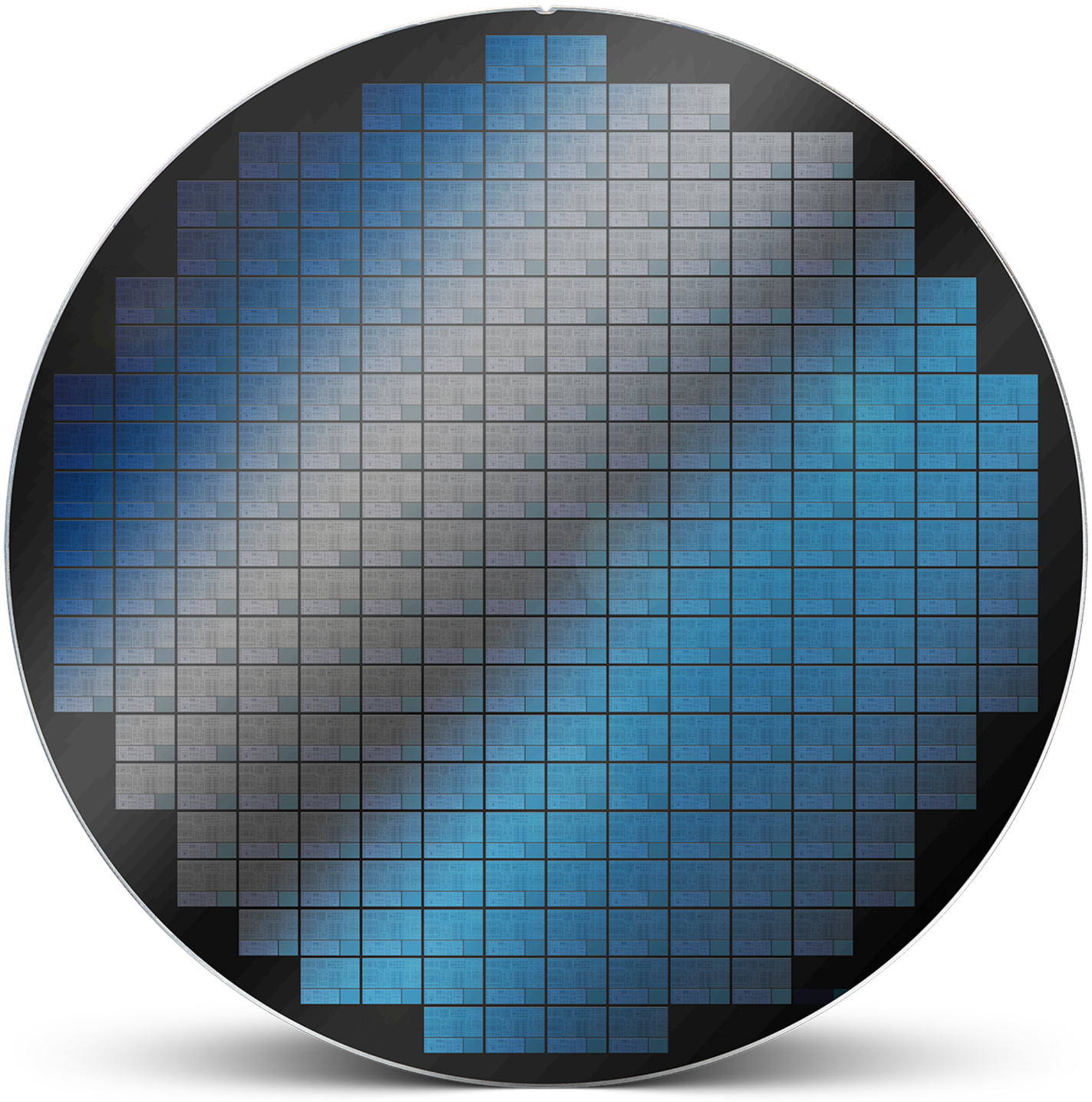
Sender Patterns to Wrangle Concurrency in Embedded Devices

Michael Caisse

michael.caisse@intel.com
@MichaelCaisse

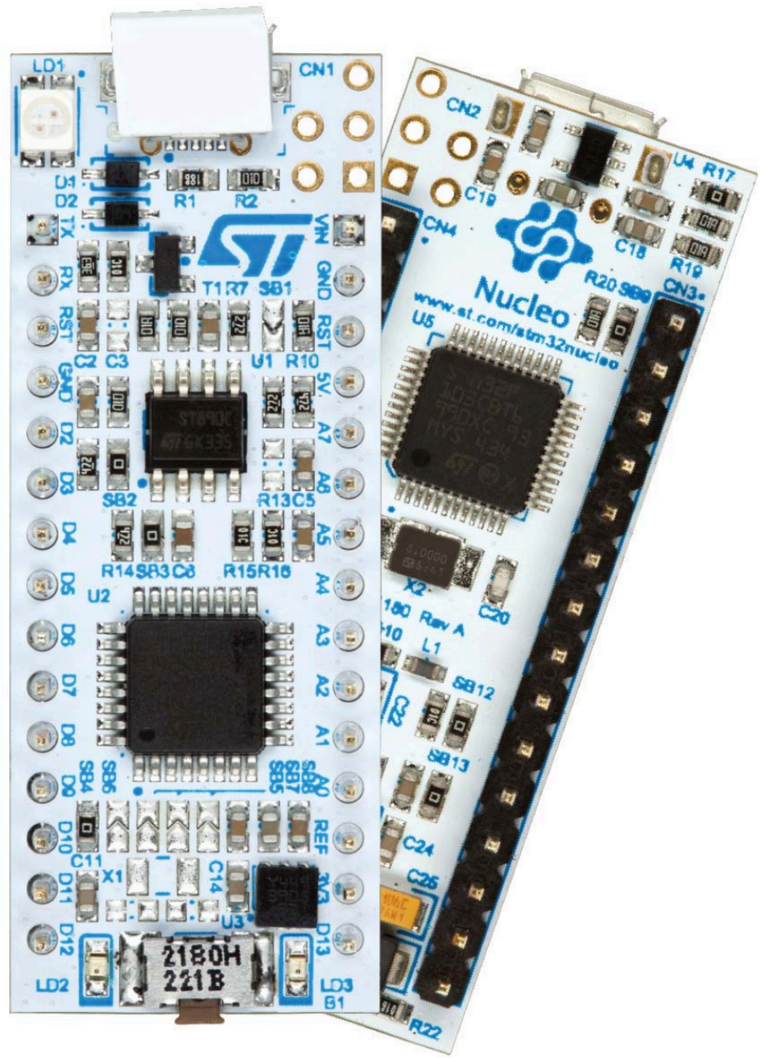


The Environment

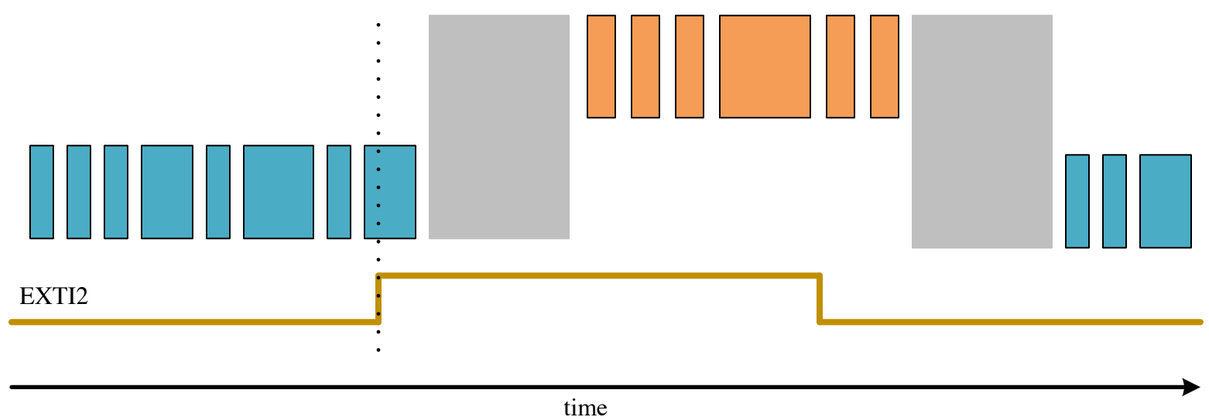
- our program doesn't end
- we don't use exceptions
- we don't have a dynamic allocator
- we don't have an rtos

STM32L432KC

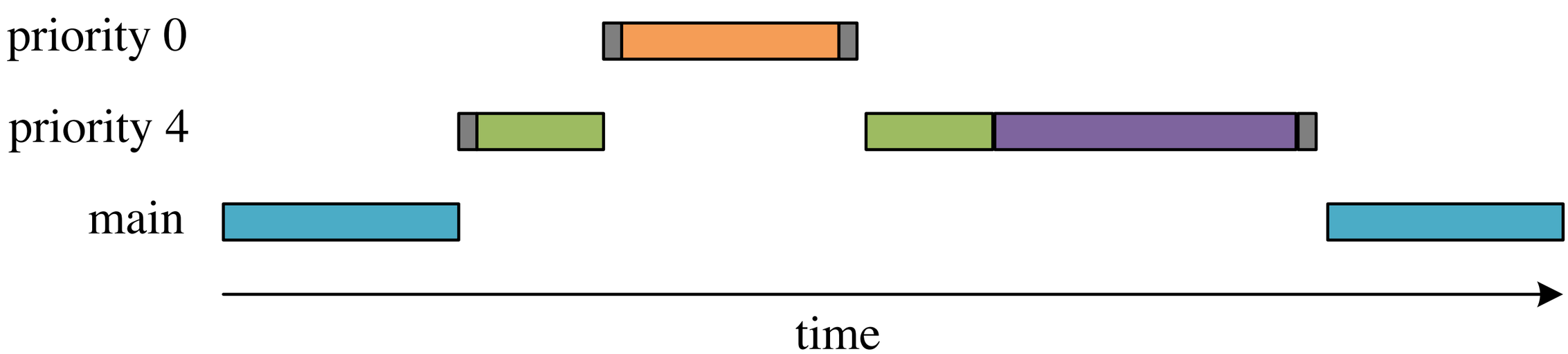
- 26 GPIO
- 14 communication interfaces (USB, SAI, I2C, CAN, …)
- Quad SPI memory interface
- 11 Timers
- 3 cap-sense channels
- 1 ADC
- 2 DAC



Interrupts


EXTI2
time

Interrupts



Exercise

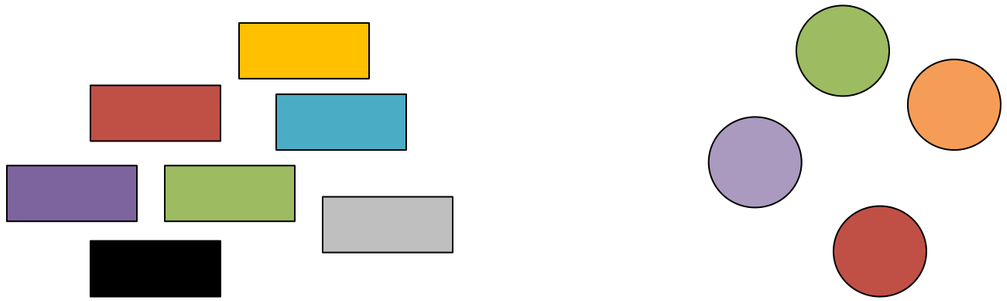
Producer ID 1

1. Think of a colour
2. Spell the colour name to Ben backwards one character at a time.
3. When done, yell "End of Line"

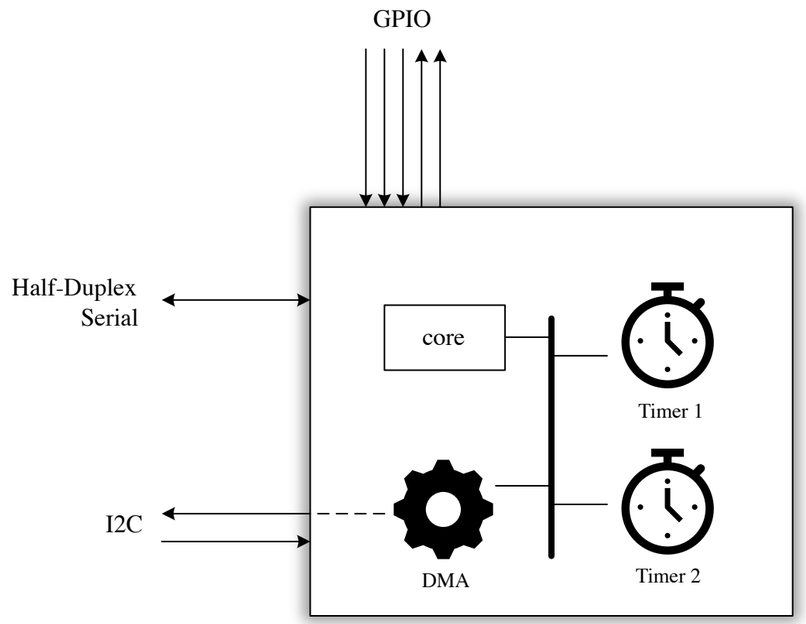
Producer ID 2

1. Count the number of people in your row.
2. Tell Ben the count one digit at a time.
3. When done, yell "End of Line"

async/concurrency

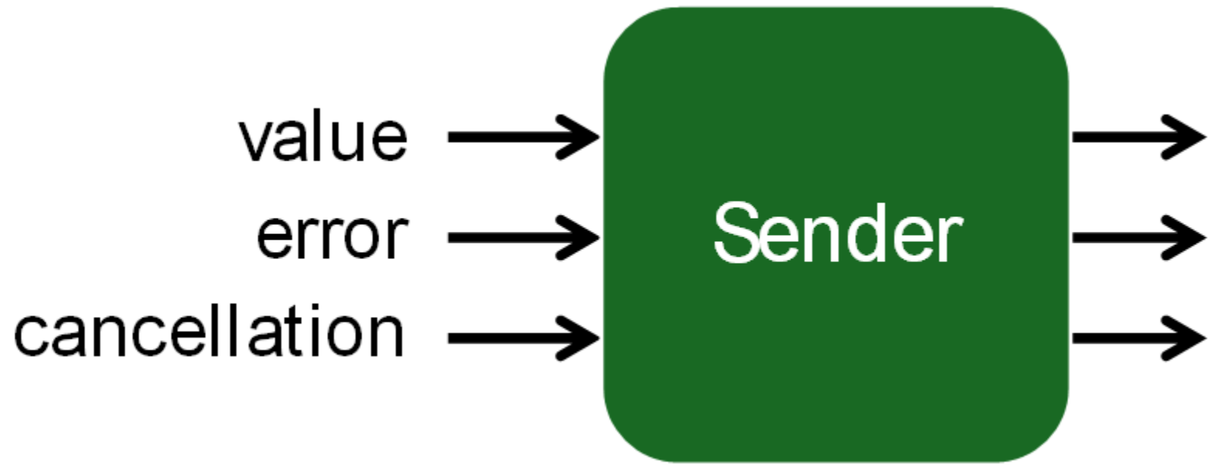


async/concurrency

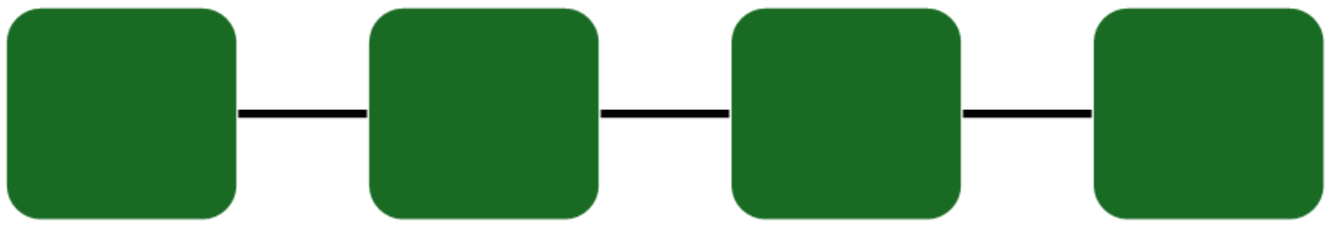


Sender World View

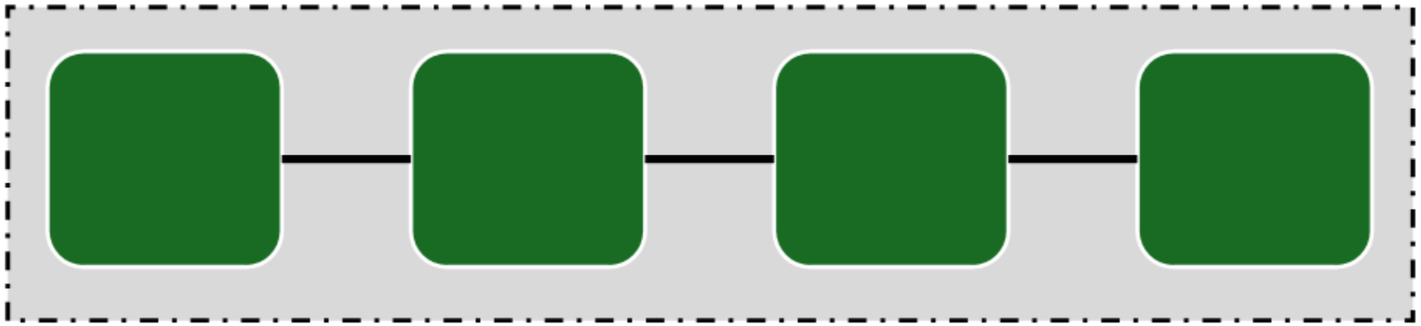
Sender



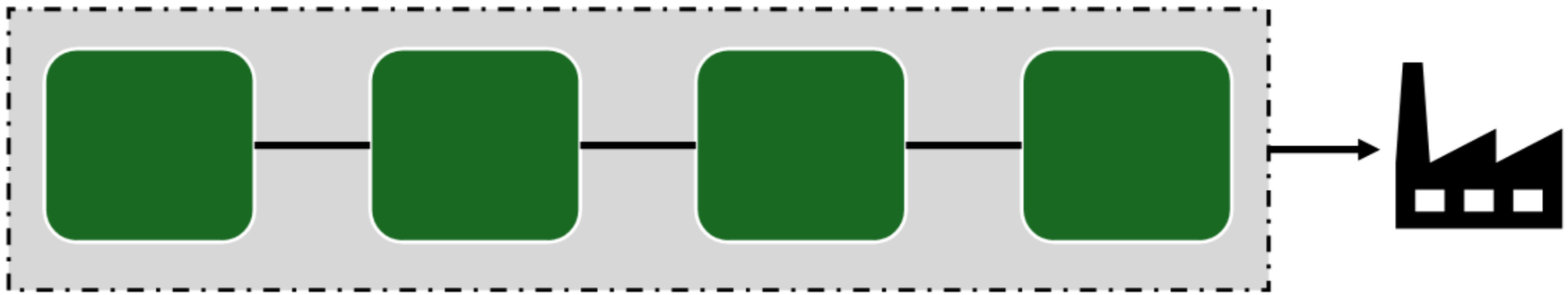
Composition



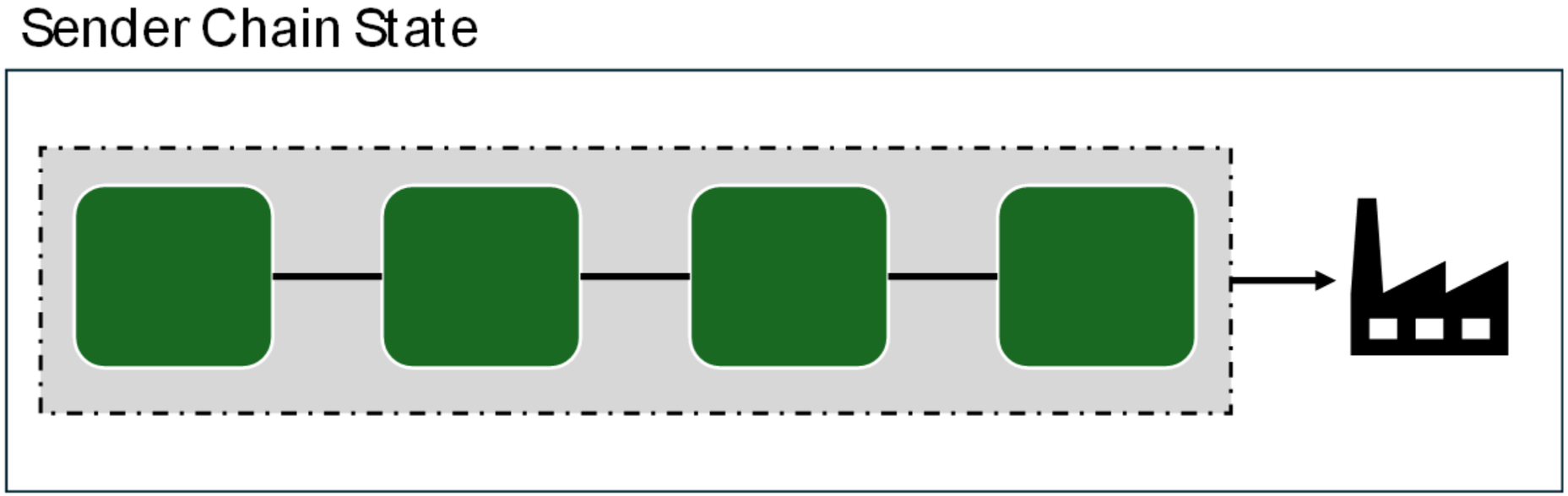
Potential Energy



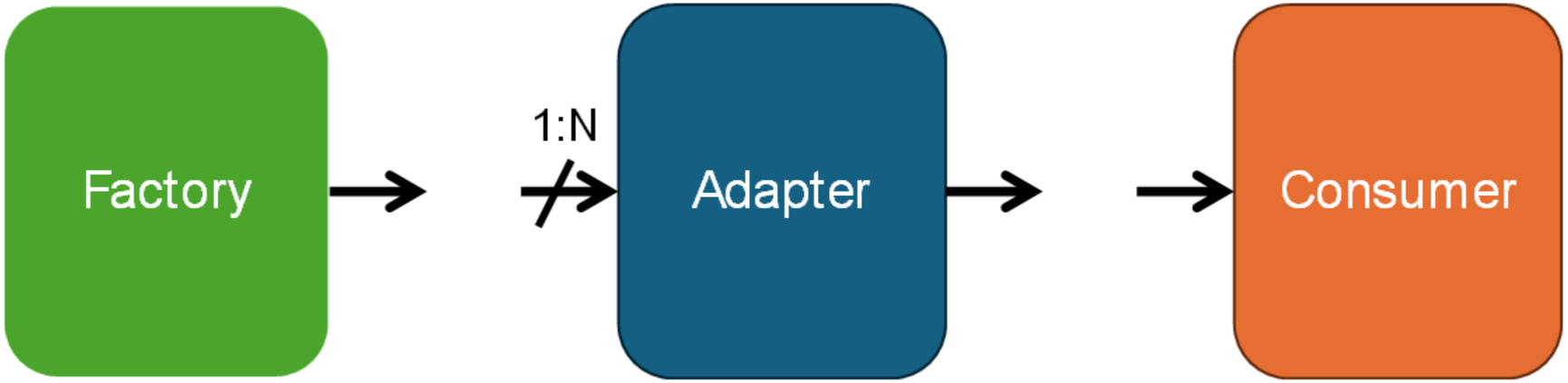
Do Something



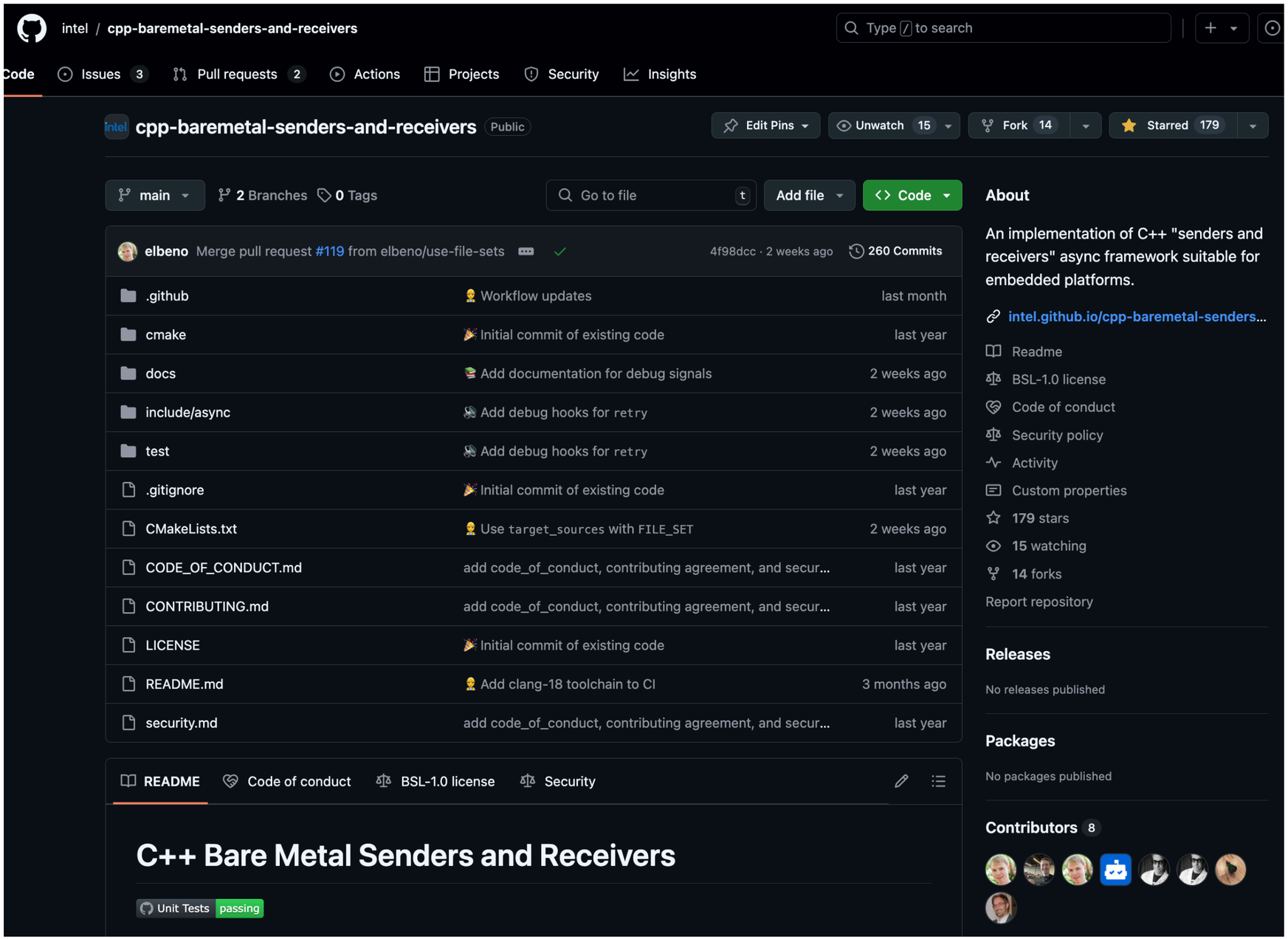
Sender Chain Contains State



The Players



Intel Baremetal Senders



https://github.com/intel/cpp-baremetal-senders-and-receivers

Sender Factories

Functions that return senders.

- just
- just_result_of
- just_error
- just_error_result_of
- just_stopped

```
1 auto sndr = async::just(42, 17);
```

Sender Factories

Functions that return senders.

- just
- just_result_of
- just_error
- just_error_result_of
- just_stopped

```
1 auto sndr = async::just_result_of(
2               [] { return 42; },
3               [] { do_something(); },
4               [] { return 17; });
```

Sender Factories

Functions that return senders.

- `schedule`

```
1 auto sndr = sched.schedule();
```

Sender Adapters

Take one or more senders and returns a sender that is the composition.

- `continue_on`
- `start_on`
- `let_value`
- `let_error`
- `let_stopped`
- `then`
- `sequence`
- `upon_error`
- `upon_stopped`
- `repeat`
- `repeat_n`
- `repeat_until`
- `retry`
- `retry_until`
- `split`
- `when_any`
- `when_all`
- `timeout_after`

Sender Consumers

Functions that take senders and start the work.

- `start_detached`
- `start_detached_unstoppable`
- `sync_wait`

Composition

```
auto comp =
    s1.schedule()
  | async::then([] { return 42; })
  | async::continue_on(s2)
  | async::then([] (int i) { return std::to_string(i); })
  ;

auto r = comp | async::sync_wait();

auto [str] = r.value_or(std::make_tuple(""_s));
```

Composition

```
auto comp =
    s1.schedule()
  | async::then([] { return 42; })
  | async::continue_on(s2)
  | async::then([] (int i) { return std::to_string(i); })
  ;

auto r = comp | async::sync_wait();

auto [str] = r.value_or(std::make_tuple(""_s));
```

Composition

```
auto comp =
    s1.schedule()
  | async::then([] { return 42; })
  | async::continue_on(s2)
  | async::then([] (int i) { return std::to_string(i); })
  ;

auto r = comp | async::sync_wait();

auto [str] = r.value_or(std::make_tuple(""_s));
```

Patterns

Timeout

- Request a thing
- Wait `N` units of time for thing
- Timeout if not received

Timeout

```
auto timout =
  async::start_on(time_scheduler{5ms},
                  async::just_error(error{42}));

auto thing = async::when_any(get_thing, timeout);

auto s =
   | thing
   | async::then( [](auto v) { /* ... */ } )
   | async::upon_error( [](auto e) { /* ... */ } )
   ;
```

Timeout

```
auto timout =
  async::start_on(time_scheduler{5ms},
                  async::just_error(error{42}));

auto thing = async::when_any(get_thing, timeout);

auto s =
   | thing
   | async::then( [](auto v) { /* ... */ } )
   | async::upon_error( [](auto e) { /* ... */ } )
   ;
```

Timeout

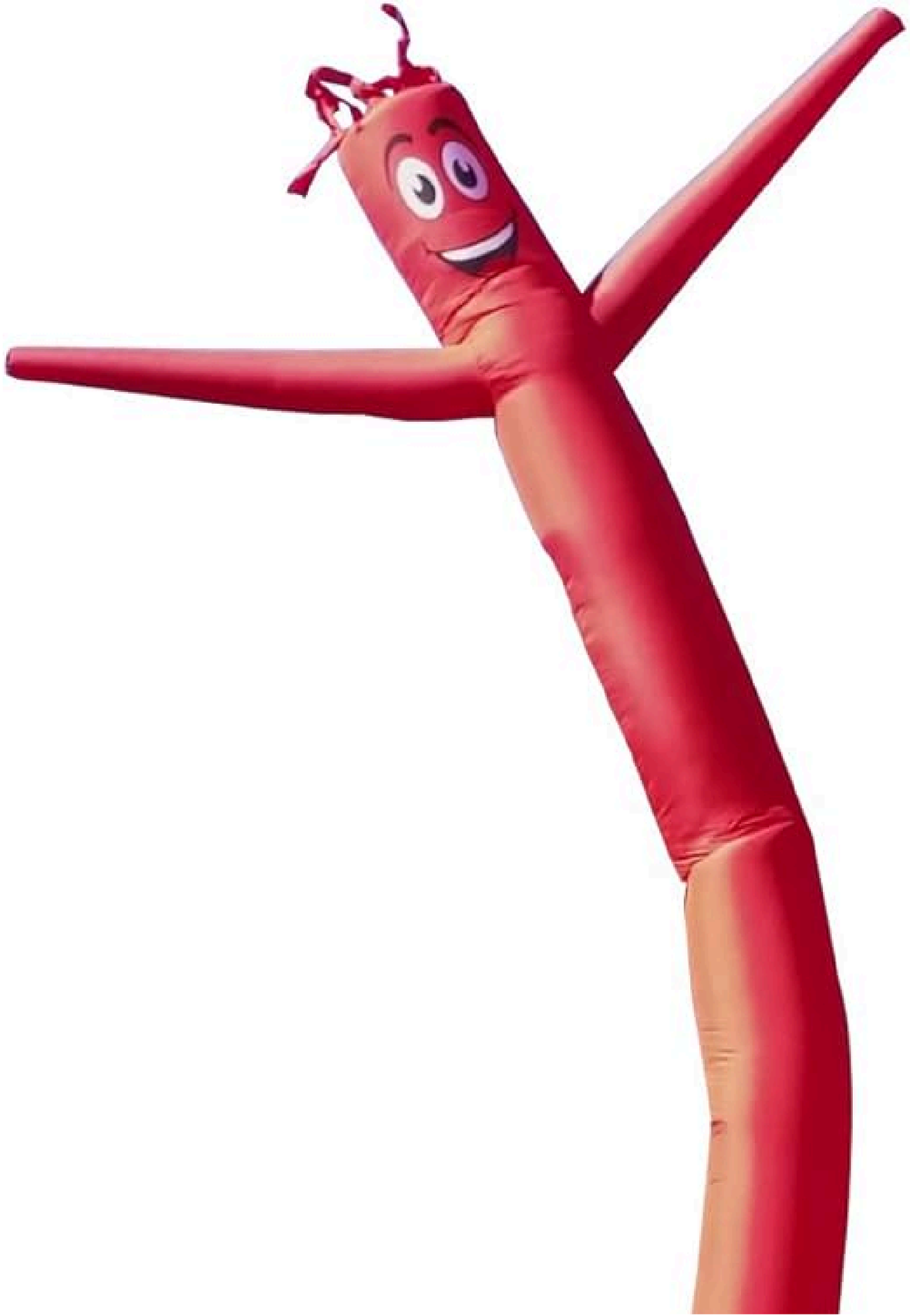
```
auto timout =
  async::start_on(time_scheduler{5ms},
                  async::just_error(error{42}));

auto thing = async::when_any(get_thing, timeout);

auto s =
   | thing
   | async::then( [](auto v) { /* ... */ } )
   | async::upon_error( [](auto e) { /* ... */ } )
   ;
```

Timeout

```
1 auto s =
2     | thing
3     | async::timeout_after(5ms, error{42})
4     | async::then( [](auto v) { /* ... */ } )
5     | async::upon_error( [](auto e) { /* ... */ } )
6     ;
```

Patterns

- `stop_detached`
- Schedulers
  - `fixed_priority_scheduler`
  - `time_scheduler`
  - `trigger_scheduler`



Robot Tubeman

LX-16A Communication

`0x55 0x55 <id> <length> <cmd> <params> <checksum>`

```
auto request_temperature =
    msg::send(
      [](std::uint_8 id) {
        serial.send(servo_temp_request_t{"id"_f = id});
      }, 0x01)
  | msg::then_receive<"recv_temp", servo_temp_read_t>(
      [&](auto msg) { return msg.get("temp"_f); }
      )
  ;

auto temp_cycle =
    request_temperature
  | async::continue_on( disp_sched )
  | async::then( [](auto t){ display(t); } )
  | delay(1s)
  | async::repeat()
  | async::start_detached()
  ;
```

LX-16A Communication

`0x55 0x55 <id> <length> <cmd> <params> <checksum>`

```
auto request_temperature =
    msg::send(
      [](std::uint_8 id) {
        serial.send(servo_temp_request_t{"id"_f = id});
      }, 0x01)
  | msg::then_receive<"recv_temp", servo_temp_read_t>(
      [&](auto msg) { return msg.get("temp"_f); }
      )
  ;

auto temp_cycle =
    request_temperature
  | async::continue_on( disp_sched )
  | async::then( [](auto t){ display(t); } )
  | delay(1s)
  | async::repeat()
  | async::start_detached()
  ;
```

LX-16A Communication

`0x55 0x55 <id> <length> <cmd> <params> <checksum>`

```
auto request_temperature =
    msg::send(
      [](std::uint_8 id) {
        serial.send(servo_temp_request_t{"id"_f = id});
      }, 0x01)
  | msg::then_receive<"recv_temp", servo_temp_read_t>(
      [&](auto msg) { return msg.get("temp"_f); }
      )
  ;

auto temp_cycle =
    request_temperature
  | async::continue_on( disp_sched )
  | async::then( [](auto t){ display(t); } )
  | delay(1s)
  | async::repeat()
  | async::start_detached()
  ;
```

LX-16A Communication

`0x55 0x55 <id> <length> <cmd> <params> <checksum>`

```
auto request_temperature =
    msg::send(
      [](std::uint_8 id) {
        serial.send(servo_temp_request_t{"id"_f = id});
      }, 0x01)
  | msg::then_receive<"recv_temp", servo_temp_read_t>(
      [&](auto msg) { return msg.get("temp"_f); }
      )
  ;

auto temp_cycle =
    request_temperature
  | async::continue_on( disp_sched )
  | async::then( [](auto t){ display(t); } )
  | delay(1s)
  | async::repeat()
  | async::start_detached()
  ;
```

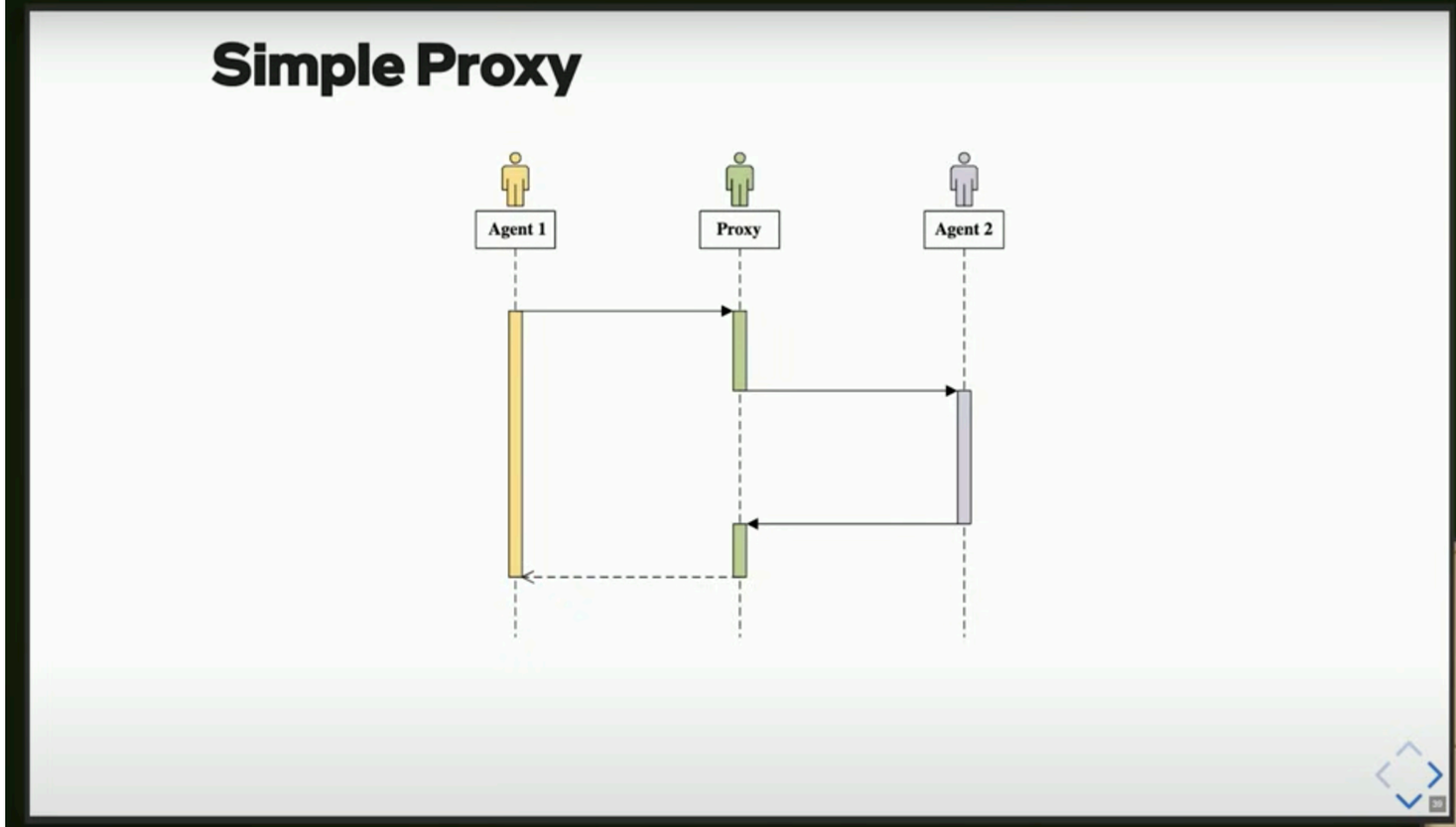
LX-16A Communication


```
0x55 0x55 <id> <length> <cmd> <params> <checksum>
```

```
auto request_temperature =
    msg::send(
      [](std::uint_8 id) {
        serial.send(servo_temp_request_t{"id"_f = id});
      }, 0x01)
  | msg::then_receive<"recv_temp", servo_temp_read_t>(
      [&](auto msg) { return msg.get("temp"_f); }
      )
  ;

auto temp_cycle =
    request_temperature
  | async::continue_on( disp_sched )
  | async::then( [](auto t){ display(t); } )
  | delay(1s)
  | async::repeat()
  | async::start_detached()
  ;
```

What is Going On?

Keynote: Employing Senders & Receivers to Tame Concurrency in C++ Embedded Systems - Michael Caisse



C++ now 2024
Aspen, Colorado

CppNow.org

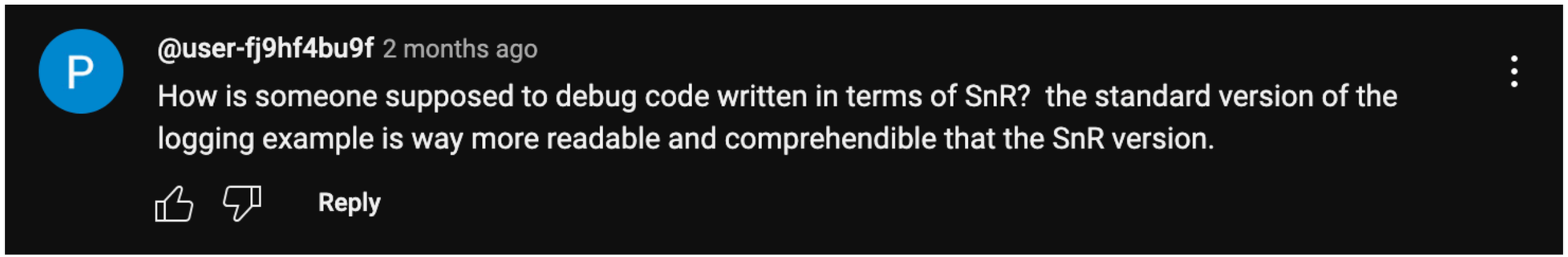
Video Sponsorship Provided By

millennium

think-cell

Employing Senders and Receivers
to Tame Concurrency in Embedded Systems

Michael Caisse


P
@user-fj9hf4bu9f 2 months ago
How is someone supposed to debug code written in terms of SnR? the standard version of the logging example is way more readable and comprehendible that the SnR version.
Reply


24
How Meta Made Debugging Async Code Easier with Coroutines and Senders
IAN PETERSEN & JESSICA WONG
Cppcon
The C++ Conference
2024
September 15 - 20

The intuitive mind is a sacred gift and the rational mind is a faithful servant. We have created a society that honors the servant and has forgotten the gift.

– Albert Einstein

Debugging

One way to debug a sender chain is to use a debugger and insert a breakpoint inside a suitable place where "real work" is being done: inside a function passed to `then` for example. This is certainly doable, but perhaps challenging for all the same reasons that debugging asynchronous code is usually challenging.

Another approach to debugging is to construct sender chains without deciding which scheduler they run on. Switching a sender chain to run on an `inline_scheduler` provides a way to debug — it is basically the same as debugging synchronous code.

Handling a debug signal

To debug code running asynchronously, this library provides a mechanism to inject a debug handler. This is done by defining a handler struct and specializing the `injected_debug_handler` variable template. The debug handler has one member function (template): `signal`.

```
#include <async/debug.hpp>

struct debug_handler {
    template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S,
              typename Ctx>
    constexpr auto signal(auto &&...) {
        fmt::print("{} {} {}", C, L, S));
    }
};

template <> inline auto async::injected_debug_handler<> = debug_handler{};
```

> **NOTE** The injection mechanism uses the same pattern as for other global concerns, like the timer manager or the priority task manager.

Michael Caisse @MichaelCaisse · Aug 29

Looking forward to this talk. I'm working hard to have a bonus on how this pattern enables debugging/inspection.

> **CppCon** @CppCon · Aug 29
>
> CppCon 2024 SESSION ANNOUNCEMENT: Sender Patterns to Wrangle Concurrency in Embedded Devices by @MichaelCaisse
>
> cppcon2024.sched.com/event/1gZfH/se...
> ...
> Show more

1 3 19 1.3K

Odin Holmes ✓
@odinthenerd

❤️

10:23 PM · Aug 29, 2024 · **123** Views

Add simple debug handler

```
namespace async_trace {

template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S, typename Ctx>
bool handled{};

struct debug_handler {
  template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S, typename Ctx>
  constexpr auto signal(auto ...) -> void {
    handled<C, L, S, Ctx> = true;
  }
};
}

template <>
inline auto async::injected_debug_handler<> = async_trace::debug_handler{};
```

Add simple debug handler

```
namespace async_trace {

template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S, typename Ctx>
bool handled{};

struct debug_handler {
  template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S, typename Ctx>
  constexpr auto signal(auto ...) -> void {
    handled<C, L, S, Ctx> = true;
  }
};
}

template <>
inline auto async::injected_debug_handler<> = async_trace::debug_handler{};
```

Add simple debug handler

```
namespace async_trace {

template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S, typename Ctx>
bool handled{};

struct debug_handler {
  template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S, typename Ctx>
  constexpr auto signal(auto ...) -> void {
    handled<C, L, S, Ctx> = true;
  }
};
}

template <>
inline auto async::injected_debug_handler<> = async_trace::debug_handler{};
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```

demo sender

```
namespace things {
  int av = 0;

  auto a0 = async::just<"just-a0">(0);
  auto a1 = async::just<"just-a1">(1);
  auto a2 = async::just<"just-a2">(2) | async::then([](auto v){ things::av = v; });
  auto w  = async::when_all(a0, a1, a2);

  int var = 0;
  auto s =
      async::just<"link-start">(42)
    | async::then<"before-wa-then">([](auto v){ things::var = v; })
    | async::seq(w)
    | async::then<"last-then">([](auto ...){ ++things::var; })
    | async::repeat();
}

int main() {
  auto d = async::start_detached<"my_chain">(things::s);

  while(true) {}
}
```


AISLE
1
AISLE
2
WEIRDSTUFF
WAREHOUSE
MISC CARDS
RACK HARDWARE
Priced AS Marked
OPEN TO THE PUBL
TEST BENCH
Effective Immediately, All Sales Are Final.
Last Day of Business is Sunday, April 8.
MOVING SERVICES

fire up gdb

```
mjcaisse@sisqi build % gdb example/demo1
GNU gdb (GDB) 15.1
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin23.4.0".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from example/demo1...
(gdb) b main
Breakpoint 1 at 0x100001e0f: file /Users/mjcaisse/sandbox/caisselabs/gdb-senders-py-utils/example/demo1.cpp, line 50.
(gdb) run
Starting program: /Users/mjcaisse/sandbox/caisselabs/gdb-senders-py-utils/build/example/demo1
[New Thread 0x2903 of process 2315]
[New Thread 0x2803 of process 2315]
[Thread 0x2903 of process 2315 exited]
warning: unhandled dyld version (17)

Thread 2 hit Breakpoint 1, main () at /Users/mjcaisse/sandbox/caisselabs/gdb-senders-py-utils/example/demo1.cpp:50
50        auto d = async::start_detached<"my_chain">(things::s);
(gdb)
```

```
(gdb) info variables
All defined variables:

File /Users/mjcaisse/sandbox/caisselabs/gdb-senders-py-utils/example/demo1.cpp:
28:     async_trace::debug_handler async::injected_debug_handler<>;
34:     async::_just::sender<{{"just-a0"}}, async::set_value_t, int> things::a0;
35:     async::_just::sender<{{"just-a1"}}, async::set_value_t, int> things::a1;
32:     int things::av;
39:     int things::var;
14:     static bool _ZN11async_trace7handledIXtlN4stdx2v19ct_stringILm9EEEtlNSt3__15arrayIcLm9EEEtlA9_cLc109ELc121ELc95ELc99ELc104ELc97ELc10
5ELc110EEEEEXtlNS3_ILm10EEEtlNS6_IcLm10EEEtlA10_cLc108ELc97ELc115ELc116ELc45ELc116ELc104ELc101ELc110EEEEEXtlS9_tlSA_tlSB_Lc115ELc101ELc116EL
c95ELc118ELc97ELc108ELc117ELc101EEEEEN5async5_then8receiverIXtlS9_tlSA_tlSB_Lc108ELc97ELc115ELc116ELc45ELc116ELc104ELc101ELc110EEEEENSC_11se
t_value_tENSC_7_repeat8receiverINSG_8op_stateIXtlNS3_ILm7EEEtlNS6_IcLm7EEEtlA7_cLc114ELc101ELc112ELc101ELc97ELc116EEEEENSD_6senderIXtlS9_tlS
A_tlSB_Lc108ELc97ELc115ELc116ELc45ELc116ELc104ELc101ELc110EEEEESF_NSC_9_sequence6senderIXtlNS3_ILm4EEEtlNS6_IcLm4EEEtlA4_cLc115ELc101ELc113E
EEEENSM_IXtlNS3_ILm15EEEtlNS6_IcLm15EEEtlA15_cLc98ELc101ELc102ELc111ELc114ELc101ELc45ELc119ELc97ELc45ELc116ELc104ELc101ELc110EEEEESF_NSC_5_j
ust6senderIXtlNS3_ILm11EEEtlNS6_IcLm11EEEtlA11_cLc108ELc105ELc110ELc107ELc45ELc115ELc116ELc97ELc114ELc116EEEEESF_JiEEEJN6things3$_1EEEENSN_6
detail7wrapperINSC_9_when_all6senderIXtlS4_tlS7_tlS8_Lc119ELc104ELc101ELc110ELc95ELc97ELc108ELc108EEEEEJNS16_10sub_senderINSW_IXtlNS3_ILm8EE
EtlNS6_IcLm8EEEtlA8_cLc106ELc117ELc115ELc116ELc45ELc97ELc48EEEEESF_JiEEELm0EEENS18_INSW_IXtlS19_tlS1A_tlS1B_Lc106ELc117ELc115ELc116ELc45ELc9
7ELc49EEEEESF_JiEEELm1EEENS18_INSM_IXtlNS3_ILm5EEEtlNS6_IcLm5EEEtlA5_cLc116ELc104ELc101ELc110EEEEESF_NSW_IXtlS19_tlS1A_tlS1B_Lc106ELc117ELc1
15ELc116ELc45ELc97ELc50EEEEESF_JiEEEJNS11_3$_0EEEELm2EEEEEEEEEEJNS11_3$_2EEEENSC_15_start_detached8receiverINS1S_8op_stateINS2_5cts_tIXtlS4_
tlS7_tlS8_Lc109ELc121ELc95ELc99ELc104ELc97ELc105ELc110EEEEEEENSG_6senderIXtlSJ_tlSK_tlSL_Lc114ELc101ELc112ELc101ELc97ELc116EEEEES1R_NSG_3$_3
EEENSC_15stack_allocatorENSC_19inplace_stop_sourceENSC_3envIJNSC_4propINSC_21get_debug_interface_tENSC_5debug15named_interfaceIXtlS4_tlS7_tl
S8_Lc109ELc121ELc95ELc99ELc104ELc97ELc105ELc110EEEEEJEEEEENS22_IJEEEEEEEEEES1Y_EES2C_EEJS1Q_EEEEE;
14:     static bool _ZN11async_trace7handledIXtlN4stdx2v19ct_stringILm9EEEtlNSt3__15arrayIcLm9EEEtlA9_cLc109ELc121ELc95ELc99ELc104ELc97ELc10
5ELc110EEEEEXtlNS3_ILm11EEEtlNS6_IcLm11EEEtlA11_cLc108ELc105ELc110ELc107ELc45ELc115ELc116ELc97ELc114ELc116EEEEEXtlNS3_ILm10EEEtlNS6_IcLm10EE
EtlA10_cLc115ELc101ELc116ELc95ELc118ELc97ELc108ELc117ELc101EEEEEN5async5_just8op_stateIXtlS9_tlSA_tlSB_Lc108ELc105ELc110ELc107ELc45ELc115ELc
116ELc97ELc114ELc116EEEEENSF_11set_value_tENSF_5_then8receiverIXtlNS3_ILm15EEEtlNS6_IcLm15EEEtlA15_cLc98ELc101ELc102ELc111ELc114ELc101ELc45E
Lc119ELc97ELc45ELc116ELc104ELc101ELc110EEEEESI_NSF_9_sequence8receiverINSO_8op_stateIXtlNS3_ILm4EEEtlNS6_IcLm4EEEtlA4_cLc115ELc101ELc113EEEE
ENSJ_6senderIXtlSL_tlSM_tlSN_Lc98ELc101ELc102ELc111ELc114ELc101ELc45ELc119ELc97ELc45ELc116ELc104ELc101ELc110EEEEESI_NSG_6senderIXtlS9_tlSA_t
lSB_Lc108ELc105ELc110ELc107ELc45ELc115ELc116ELc97ELc114ELc116EEEEESI_JiEEEJN6things3$_1EEEENSO_6detail7wrapperINSF_9_when_all6senderIXtlS4_t
lS7_tlS8_Lc119ELc104ELc101ELc110ELc95ELc97ELc108ELc108EEEEEJNS12_10sub_senderINSV_IXtlNS3_ILm8EEEtlNS6_IcLm8EEEtlA8_cLc106ELc117ELc115ELc116
ELc45ELc97ELc48EEEEESI_JiEEELm0EEENS14_INSV_IXtlS15_tlS16_tlS17_Lc106ELc117ELc115ELc116ELc45ELc97ELc49EEEEESI_JiEEELm1EEENS14_INSU_IXtlNS3_I
Lm5EEEtlNS6_IcLm5EEEtlA5_cLc116ELc104ELc101ELc110EEEEESI_NSV_IXtlS15_tlS16_tlS17_Lc106ELc117ELc115ELc116ELc45ELc97ELc50EEEEESI_JiEEEJNSX_3$_
```

```
_ZN11async_trace7handledIXtlN4stdx2v19ct_stringILm9EEEtlNSt3__15arrayIcLm9EEEtlA9_cLc109ELc121E
Lc95ELc99ELc104ELc97ELc105ELc110EEEEEXtlNS3_ILm10EEEtlNS6_IcLm10EEEtlA10_cLc108ELc97ELc115ELc11
6ELc45ELc116ELc104ELc101ELc110EEEEEXtlS9_tlSA_tlSB_Lc115ELc101ELc116ELc95ELc118ELc97ELc108ELc11
7ELc101EEEEEN5async5_then8receiverIXtlS9_tlSA_tlSB_Lc108ELc97ELc115ELc116ELc45ELc116ELc104ELc10
1ELc110EEEEENSC_11set_value_tENSC_7_repeat8receiverINSG_8op_stateIXtlNS3_ILm7EEEtlNS6_IcLm7EEEt
lA7_cLc114ELc101ELc112ELc101ELc97ELc116EEEEENSD_6senderIXtlS9_tlSA_tlSB_Lc108ELc97ELc115ELc116E
Lc45ELc116ELc104ELc101ELc110EEEEESF_NSC_9_sequence6senderIXtlNS3_ILm4EEEtlNS6_IcLm4EEEtlA4_cLc1
15ELc101ELc113EEEEENSM_IXtlNS3_ILm15EEEtlNS6_IcLm15EEEtlA15_cLc98ELc101ELc102ELc111ELc114ELc101
ELc45ELc119ELc97ELc45ELc116ELc104ELc101ELc110EEEEESF_NSC_5_just6senderIXtlNS3_ILm11EEEtlNS6_IcL
m11EEEtlA11_cLc108ELc105ELc110ELc107ELc45ELc115ELc116ELc97ELc114ELc116EEEEESF_JiEEEJN6things3$_
1EEEENSN_6detail7wrapperINSC_9_when_all6senderIXtlS4_tlS7_tlS8_Lc119ELc104ELc101ELc110ELc95ELc9
7ELc108ELc108EEEEEJNS16_10sub_senderINSW_IXtlNS3_ILm8EEEtlNS6_IcLm8EEEtlA8_cLc106ELc117ELc115EL
c116ELc45ELc97ELc48EEEEESF_JiEEELm0EEENS18_INSW_IXtlS19_tlS1A_tlS1B_Lc106ELc117ELc115ELc116ELc4
5ELc97ELc49EEEEESF_JiEEELm1EEENS18_INSM_IXtlNS3_ILm5EEEtlNS6_IcLm5EEEtlA5_cLc116ELc104ELc101ELc
110EEEEESF_NSW_IXtlS19_tlS1A_tlS1B_Lc106ELc117ELc115ELc116ELc45ELc97ELc50EEEEESF_JiEEEJNS11_3$_
0EEEELm2EEEEEEEEEEJNS11_3$_2EEEENSC_15_start_detached8receiverINS1S_8op_stateINS2_5cts_tIXtlS4_
tlS7_tlS8_Lc109ELc121ELc95ELc99ELc104ELc97ELc105ELc110EEEEEEENSG_6senderIXtlSJ_tlSK_tlSL_Lc114E
Lc101ELc112ELc101ELc97ELc116EEEEES1R_NSG_3$_3EEENSC_15stack_allocatorENSC_19inplace_stop_source
ENSC_3envIJNSC_4propINSC_21get_debug_interface_tENSC_5debug15named_interfaceIXtlS4_tlS7_tlS8_Lc
109ELc121ELc95ELc99ELc104ELc97ELc105ELc110EEEEEJEEEEENS22_IJEEEEEEEEEES1Y_EES2C_EEJS1Q_EEEEE
```

```
async_trace::handled<stdx::v1::ct_string<9ul>{std::__1::array<char, 9ul>{char [9]{(char)109,
(char)121, (char)95, (char)99, (char)104, (char)97, (char)105, (char)110}}},
stdx::v1::ct_string<10ul>{std::__1::array<char, 10ul>{char [10]{(char)108, (char)97,
(char)115, (char)116, (char)45, (char)116, (char)104, (char)101, (char)110}}},
stdx::v1::ct_string<10ul>{std::__1::array<char, 10ul>{char [10]{(char)115, (char)101,
(char)116, (char)95, (char)118, (char)97, (char)108, (char)117, (char)101}}},
async::_then::receiver<stdx::v1::ct_string<10ul>{std::__1::array<char, 10ul>{char
[10]{(char)108, (char)97, (char)115, (char)116, (char)45, (char)116, (char)104, (char)101,
(char)110}}}, async::set_value_t,
async::_repeat::receiver<async::_repeat::op_state<stdx::v1::ct_string<7ul>{std::__1::array<char
, 7ul>{char [7]{(char)114, (char)101, (char)112, (char)101, (char)97, (char)116}}},
async::_then::sender<stdx::v1::ct_string<10ul>{std::__1::array<char, 10ul>{char
[10]{(char)108, (char)97, (char)115, (char)116, (char)45, (char)116, (char)104, (char)101,
(char)110}}}, async::set_value_t,
async::_sequence::sender<stdx::v1::ct_string<4ul>{std::__1::array<char, 4ul>{char
[4]{(char)115, (char)101, (char)113}}},
async::_then::sender<stdx::v1::ct_string<15ul>{std::__1::array<char, 15ul>{char [15]{(char)98,
(char)101, (char)102, (char)111, (char)114, (char)101, (char)45, (char)119, (char)97,
(char)45, (char)116, (char)104, (char)101, (char)110}}}, async::set_value_t,
async::_just::sender<stdx::v1::ct_string<11ul>{std::__1::array<char, 11ul>{char
[11]{(char)108, (char)105, (char)110, (char)107, (char)45, (char)115, (char)116, (char)97,
(char)114, (char)116}}}, async::set_value_t, int>, things::$_1>,
async::_sequence::detail::wrapper<async::_when_all::sender<stdx::v1::ct_string<9ul>{std::__1::a
rray<char, 9ul>{char [9]{(char)119, (char)104, (char)101, (char)110, (char)95, (char)97,
(char)108, (char)108}}},
async::_when_all::sub_sender<async::_just::sender<stdx::v1::ct_string<8ul>{std::__1::array<char
, 8ul>{char [8]{(char)106, (char)117, (char)115, (char)116, (char)45, (char)97, (char)48}}},
async::set_value_t, int>, 0ul>,
async::_when_all::sub_sender<async::_just::sender<stdx::v1::ct_string<8ul>{std::__1::array<char
, 8ul>{char [8]{(char)106, (char)117, (char)115, (char)116, (char)45, (char)97, (char)49}}},
async::set_value_t, int>, 1ul>,
async::_when_all::sub_sender<async::_then::sender<stdx::v1::ct_string<5ul>{std::__1::array<char
, 5ul>{char [5]{(char)116, (char)104, (char)101, (char)110}}}, async::set_value_t,
async::_just::sender<stdx::v1::ct_string<8ul>{std::__1::array<char, 8ul>{char [8]{(char)106,
(char)117, (char)115, (char)116, (char)45, (char)97, (char)50}}}, async::set_value_t, int>,
things::$_0>, 2ul>>>>, things::$_2>,
async::_start_detached::receiver<async::_start_detached::op_state<stdx::v1::cts_t<stdx::v1::ct_
```

```
172 [9]{(char)109, (char)121, (char)95, (char)99, (char)104, (char)97, (char)105, (char)110}}}>>,
173 async::env<>>>>, async::_repeat::$_3>,
174 async::_start_detached::receiver<async::_start_detached::op_state<stdx::v1::cts_t<stdx::v1::ct_
175 string<9ul>{std::__1::array<char, 9ul>{char [9]{(char)109, (char)121, (char)95, (char)99,
176 (char)104, (char)97, (char)105, (char)110}}}>,
177 async::_repeat::sender<stdx::v1::ct_string<7ul>{std::__1::array<char, 7ul>{char [7]{(char)114,
178 (char)101, (char)112, (char)101, (char)97, (char)116}}},
179 async::_then::sender<stdx::v1::ct_string<10ul>{std::__1::array<char, 10ul>{char
180 [10]{(char)108, (char)97, (char)115, (char)116, (char)45, (char)116, (char)104, (char)101,
181 (char)110}}}, async::set_value_t,
182 async::_sequence::sender<stdx::v1::ct_string<4ul>{std::__1::array<char, 4ul>{char
183 [4]{(char)115, (char)101, (char)113}}},
184 async::_then::sender<stdx::v1::ct_string<15ul>{std::__1::array<char, 15ul>{char [15]{(char)98,
185 (char)101, (char)102, (char)111, (char)114, (char)101, (char)45, (char)119, (char)97,
186 (char)45, (char)116, (char)104, (char)101, (char)110}}}, async::set_value_t,
187 async::_just::sender<stdx::v1::ct_string<11ul>{std::__1::array<char, 11ul>{char
188 [11]{(char)108, (char)105, (char)110, (char)107, (char)45, (char)115, (char)116, (char)97,
189 (char)114, (char)116}}}, async::set_value_t, int>, things::$_1>,
190 async::_sequence::detail::wrapper<async::_when_all::sender<stdx::v1::ct_string<9ul>{std::__1::a
191 rray<char, 9ul>{char [9]{(char)119, (char)104, (char)101, (char)110, (char)95, (char)97,
192 (char)108, (char)108}}},
193 async::_when_all::sub_sender<async::_just::sender<stdx::v1::ct_string<8ul>{std::__1::array<char
194 , 8ul>{char [8]{(char)106, (char)117, (char)115, (char)116, (char)45, (char)97, (char)48}}},
195 async::set_value_t, int>, 0ul>,
196 async::_when_all::sub_sender<async::_just::sender<stdx::v1::ct_string<8ul>{std::__1::array<char
197 , 8ul>{char [8]{(char)106, (char)117, (char)115, (char)116, (char)45, (char)97, (char)49}}},
198 async::set_value_t, int>, 1ul>,
199 async::_when_all::sub_sender<async::_then::sender<stdx::v1::ct_string<5ul>{std::__1::array<char
200 , 5ul>{char [5]{(char)116, (char)104, (char)101, (char)110}}}, async::set_value_t,
201 async::_just::sender<stdx::v1::ct_string<8ul>{std::__1::array<char, 8ul>{char [8]{(char)106,
202 (char)117, (char)115, (char)116, (char)45, (char)97, (char)50}}}, async::set_value_t, int>,
203 things::$_0>, 2ul>>>>, things::$_2>, async::_repeat::$_3>, async::stack_allocator,
204 async::inplace_stop_source, async::env<async::prop<async::get_debug_interface_t,
205 async::debug::named_interface<stdx::v1::ct_string<9ul>{std::__1::array<char, 9ul>{char
206 [9]{(char)109, (char)121, (char)95, (char)99, (char)104, (char)97, (char)105, (char)110}}}>>,
207 async::env<>>>>>, things::$_2>>
208 async_trace::handled<stdx::v1::ct_string<9ul>{std::__1::array<char, 9ul>{char [9]{(char)109, (char)121, (char)95, (char)99, (char)104, (char)97, (char)105, (char)110}}}, stdx::v1::ct_str
209 , (char)107, (char)45, (char)115, (char)116, (char)97, (char)114, (char)116}}}, async::set_value_t, int>, things::$_1>, async::_sequence::detail::wrapper<async::_when_all::sender<stdx::v1
```

Pretty print the `ct_string`

```
async_trace::handled<"my_chain", "last-then", "set_value", async::_then::receiver<"last-then",
async::set_value_t, async::_repeat::receiver<async::_repeat::op_state<"repeat",
async::_then::sender<"last-then", async::set_value_t, async::_sequence::sender<"seq",
async::_then::sender<"before-wa-then", async::set_value_t, async::_just::sender<"link-start",
async::set_value_t, int>, things::$_1>,
async::_sequence::detail::wrapper<async::_when_all::sender<"when_all",
async::_when_all::sub_sender<async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
async::_when_all::sub_sender<async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
async::_when_all::sub_sender<async::_then::sender<"then", async::set_value_t,
async::_just::sender<"just-a2", async::set_value_t, int>, things::$_0>, 2ul>>>>, things::$_2>,
async::_start_detached::receiver<async::_start_detached::op_state<stdx::v1::cts_t<"my_chain">,
async::_repeat::sender<"repeat", async::_then::sender<"last-then", async::set_value_t,
async::_sequence::sender<"seq", async::_then::sender<"before-wa-then", async::set_value_t,
async::_just::sender<"link-start", async::set_value_t, int>, things::$_1>,
async::_sequence::detail::wrapper<async::_when_all::sender<"when_all",
async::_when_all::sub_sender<async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
async::_when_all::sub_sender<async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
async::_when_all::sub_sender<async::_then::sender<"then", async::set_value_t,
async::_just::sender<"just-a2", async::set_value_t, int>, things::$_0>, 2ul>>>>, things::$_2>,
async::_repeat::$_3>, async::stack_allocator, async::inplace_stop_source,
async::env<async::prop<async::get_debug_interface_t,
async::debug::named_interface<"my_chain">>, async::env<>>>>, async::_repeat::$_3>,
async::_start_detached::receiver<async::_start_detached::op_state<stdx::v1::cts_t<"my_chain">,
async::_repeat::sender<"repeat", async::_then::sender<"last-then", async::set_value_t,
async::_sequence::sender<"seq", async::_then::sender<"before-wa-then", async::set_value_t,
async::_just::sender<"link-start", async::set_value_t, int>, things::$_1>,
async::_sequence::detail::wrapper<async::_when_all::sender<"when_all",
async::_when_all::sub_sender<async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
async::_when_all::sub_sender<async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
async::_when_all::sub_sender<async::_then::sender<"then", async::set_value_t,
async::_just::sender<"just-a2", async::set_value_t, int>, things::$_0>, 2ul>>>>, things::$_2>,
async::_repeat::$_3>, async::stack_allocator, async::inplace_stop_source,
async::env<async::prop<async::get_debug_interface_t,
async::debug::named_interface<"my_chain">>, async::env<>>>>>, things::$_2>>
```

Find `start_detached`

```
async_trace::handled<"my_chain", "start_detached", "start",
async::_start_detached::op_state<stdx::v1::cts_t<"my_chain">, async::_repeat::sender<"repeat",
async::_then::sender<"last-then", async::set_value_t, async::_sequence::sender<"seq",
async::_then::sender<"before-wa-then", async::set_value_t, async::_just::sender<"link-start",
async::set_value_t, int>, things::$_1>,
async::_sequence::detail::wrapper<async::_when_all::sender<"when_all",
async::_when_all::sub_sender<async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
async::_when_all::sub_sender<async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
async::_when_all::sub_sender<async::_then::sender<"then", async::set_value_t,
async::_just::sender<"just-a2", async::set_value_t, int>, things::$_0>, 2ul>>>>, things::$_2>,
async::_repeat::$_3>, async::stack_allocator, async::inplace_stop_source,
async::env<async::prop<async::get_debug_interface_t,
async::debug::named_interface<"my_chain">>, async::env<>>>>
```

`start_detached` contained sender

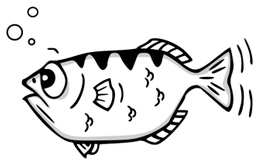
```
async::_repeat::sender<"repeat",
async::_then::sender<"last-then", async::set_value_t, async::_sequence::sender<"seq",
async::_then::sender<"before-wa-then", async::set_value_t, async::_just::sender<"link-start",
async::set_value_t, int>, things::$_1>,
async::_sequence::detail::wrapper<async::_when_all::sender<"when_all",
async::_when_all::sub_sender<async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
async::_when_all::sub_sender<async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
async::_when_all::sub_sender<async::_then::sender<"then", async::set_value_t,
async::_just::sender<"just-a2", async::set_value_t, int>, things::$_0>, 2ul>>>>, things::$_2>,
async::_repeat::$_3>
```

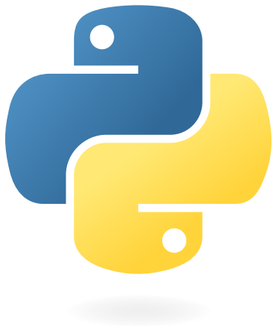
start_detached

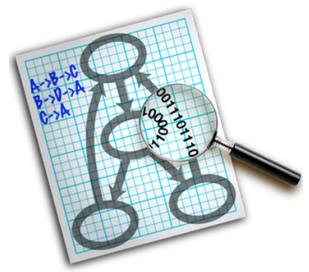
```
async::_repeat::sender<"repeat",
  async::_then::sender<"last-then", async::set_value_t,
    async::_sequence::sender<"seq",
      async::_then::sender<"before-wa-then", async::set_value_t,
        async::_just::sender<"link-start", async::set_value_t, int>, things::$_1>,
      async::_sequence::detail::wrapper<
        async::_when_all::sender<"when_all",
          async::_when_all::sub_sender<
            async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
          async::_when_all::sub_sender<
            async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
          async::_when_all::sub_sender<
            async::_then::sender<"then", async::set_value_t,
              async::_just::sender<"just-a2", async::set_value_t, int>,
              things::$_0>,
            2ul>>>>,
    things::$_2>,
  async::_repeat::$_3>
```

```
async::_repeat::sender<"repeat",
  async::_then::sender<"last-then", async::set_value_t,
    async::_sequence::sender<"seq",
      async::_then::sender<"before-wa-then", async::set_value_t,
        async::_just::sender<"link-start", async::set_value_t, int>, things::$_1>,
      async::_sequence::detail::wrapper<
        async::_when_all::sender<"when_all",
          async::_when_all::sub_sender<
            async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
          async::_when_all::sub_sender<
            async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
          async::_when_all::sub_sender<
            async::_then::sender<"then", async::set_value_t,
              async::_just::sender<"just-a2", async::set_value_t, int>,
              things::$_0>,
            2ul>>>>,
    things::$_2>,
  async::_repeat::$_3>
```


cppcon | 2018
THE C++ CONFERENCE • BELLEVUE, WASHINGTON
LIBERATING THE DEBUGGING EXPERIENCE
WITH THE GDB PYTHON API
JEFF TRULL
27 SEPTEMBER 2018
JEFF TRULL
Liberating the Debugging
Experience with
the GDB Python API
CppCon.org







Lark, Transformers, and Python

```
handled_type : "async_trace::handled" "<" string "," string "," string "," (op_type|context) ">"
op_type      : "async::_start_detached::op_state" "<" cts "," sender_type "," template_params ">"
context      : "async_trace::context" "<" namespace_type "," string "," sender_type ">"

cts          : "stdx::v1::cts_t<" string ">"

?sender_type : then_sender_type
             | seq_sender_type
             | just_sender_type
             | when_all_sender_type
             | repeat_sender_type
             | namespace_type

then_sender_type : "async::_then::sender" "<" string "," namespace_type "," sender_type "," namespace_type ">"
seq_sender_type  : "async::_sequence::sender" "<" string "," sender_type "," "async::_sequence::detail::wrapper" "<" sender_type ">" ">"
just_sender_type : "async::_just::sender" "<" string "," namespace_type "," namespace_type ">"

repeat_sender_type : "async::_repeat::sender" "<" string "," sender_type "," namespace_type ">"

when_all_sender_type      : "async::_when_all::sender" "<" string ("," _when_all_sub_sender_type)* ">"
_when_all_sub_sender_type : "async::_when_all::sub_sender" "<" sender_type "," integral ">"

namespace_type : ("::")? (identifier "::")* identifier template?

template : "<" template_params? ">"
template_param : string
               | namespace_type
               | integral

template_params: template_param ( "," template_param )*


string       : ESCAPED_STRING
identifier   : CPP_IDENTIFIER

CPP_IDENTIFIER: (("_"|"$"|ALPHA) ("_"|ALPHA|DIGIT)*)
ALPHA         : "a.."z" | "A..Z"
DIGIT         : "0".."9"

integral      : NUMBER ("ul")?
```

```
def async_debug():
    async_debug_re = re.compile(r"async_trace::handled")
    start_detached_op_state_re = re.compile(r"async::_start_detached::op_state")
    start_detached_context_re = re.compile(r"async_trace::context<async_trace::start_detached_t")

    frame = gdb.selected_frame()
    block = frame.block()
    debug_symbols = set()

    while block:
        for symbol in block:
            name = symbol.name
            demangled_name = cxxfilt.demangle(name)
            if async_debug_re.search(demangled_name):
                debug_symbols.add((symbol, demangled_name))
        block = block.superblock

    handled_states = []

    for (debug_symbol, demangled_name) in debug_symbols:
```

```
from .gdb_wedge import get_symbols

class DotColorMapper:
    """
    Signals have priority. The highest priority signal that is set
    will dictate the color
    """
    signal_priority_map = (
        ("set_error","orangered"),
        ("set_stopped","lightgrey"),
        ("set_value","dodgerblue"),
        ("start","mediumseagreen"),
        ("eval_predicate", "tan"))

    def __init__(self, chain_name, frame):
        self.chain_name = chain_name
        self.frame = frame

    def map(self, node_identifier):
        if not node_identifier or not self.frame:
            return {}

        syms = get_symbols(self.chain_name, **node_identifier)

        mapped_color = 'ghostwhite'
        for (sig, color) in self.signal_priority_map:
            sig_values = [s.get_value(self.frame) for s in syms if s.signal_name == sig]
            if any(sig_values):
                mapped_color = color
                break

        return {'style': 'filled', 'fillcolor': mapped_color}
```

```
import graphviz

class BaseColorMapper:
    def map(self, node_identifier):
        return {}

class DotGraphBuilder:
    def __init__(self, dot=None, color_mapper=None, **kwargs):
        if not dot:
            u_graph_attr = kwargs.get('graph_attr', {})
            graph_attr={'compound': 'true', **u_graph_attr}
            kwargs = {**kwargs, 'graph_attr': graph_attr}
            dot = graphviz.Digraph(**kwargs)

        self.dot = dot
        self.color_mapper = color_mapper if color_mapper else BaseColorMapper()

    def add_node(self, name, label, node_identifier=None, **kwargs):
        color_spec = self.color_mapper.map(node_identifier)
        extra_args = {**color_spec, **kwargs}
        self.dot.node(name, label, **extra_args)

    def add_edge(self, a, b, **kwargs):
        self.dot.edge(a, b, **kwargs)

        def subgraph(self, name, label, node_identifier=None, **kwargs):
        color_spec = self.color_mapper.map(node_identifier)
        style = kwargs.get('style', 'striped')
        sub = DotGraphBuilder(name=f"cluster_{name}", graph_attr={'label': f'{label}', 'style': style, **color_spe
        sub.add_node(name=name, label="", shape="point", style="invis")
        return sub
```

```
        pretty_name = stdx.prettify_ct_strings(demangled_name)
        add_symbol(debug_symbol)
        if start_detached_op_state_re.search(pretty_name) or start_detached_context_re.search(pretty_name):
            try:
                h = Handled(pretty_name)
                h.set_gdb_symbol(debug_symbol)
                handled_states.append(h)
            except:
                pass
    return handled_states
```

```
    def add_subgraph(self, g, node_identifier=None, **kwargs):
        color_spec = {} #self.color_mapper.map(node_identifier)
        extra_args = {**color_spec, **kwargs}
        self.dot.subgraph(g.dot, **extra_args)

    def display(self):
        self.dot.render("./foo.gv", view=True)
```

Demo Time

`repeat` doesn't clear

It is a simple matter of programming.

– Ben Deane

Templates are Pattern Matching

```
async::_repeat::sender<"repeat",
  async::_then::sender<"last-then", async::set_value_t,
    async::_sequence::sender<"seq",
      async::_then::sender<"before-wa-then", async::set_value_t,
        async::_just::sender<"link-start", async::set_value_t, int>, things::$_1>,
      async::_sequence::detail::wrapper<
        async::_when_all::sender<"when_all",
          async::_when_all::sub_sender<
            async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
          async::_when_all::sub_sender<
            async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
          async::_when_all::sub_sender<
            async::_then::sender<"then", async::set_value_t,
              async::_just::sender<"just-a2", async::set_value_t, int>,
              things::$_0>,
            2ul>>>>,
    things::$_2>,
  async::_repeat::$_3>
```

Rework the `debug_handler`

```
namespace async_trace {

template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S, typename Ctx>
bool handled{};

struct debug_handler {
  template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S,
            typename Ctx, typename... Args>
  constexpr auto signal(Args &&...) -> void {
    handled<C, L, S, Ctx> = true;
  }
};
}

template <>
inline auto async::injected_debug_handler<> = async_trace::debug_handler{};
```

Rework the `debug_handler`

```
struct debug_handler {
  template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S,
            typename Ctx, typename... Args>
  constexpr auto signal(Args &&...) -> void {
    using context_t = extract_context_t<Ctx>;
    using tag_t = typename context_t::tag_t;
    using sndrs_t = typename context_t::sndrs_t;

    if constexpr (S == stdx::ct_string("start") && contains_v<reset_recursively_t, tag_t>) {
      recursively_reset_handled<C, context_t, sndrs_t>();
    }

    handled<C, L, S, context_t> = true;
  }
};
```

Rework the `debug_handler`

```
struct debug_handler {
  template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S,
            typename Ctx, typename... Args>
  constexpr auto signal(Args &&...) -> void {
    using context_t = extract_context_t<Ctx>;
    using tag_t = typename context_t::tag_t;
    using sndrs_t = typename context_t::sndrs_t;

    if constexpr (S == stdx::ct_string("start") && contains_v<reset_recursively_t, tag_t>) {
      recursively_reset_handled<C, context_t, sndrs_t>();
    }

    handled<C, L, S, context_t> = true;
  }
};
```

Rework the `debug_handler`

```
struct debug_handler {
  template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S,
            typename Ctx, typename... Args>
  constexpr auto signal(Args &&...) -> void {
    using context_t = extract_context_t<Ctx>;
    using tag_t = typename context_t::tag_t;
    using sndrs_t = typename context_t::sndrs_t;

    if constexpr (S == stdx::ct_string("start") && contains_v<reset_recursively_t, tag_t>) {
      recursively_reset_handled<C, context_t, sndrs_t>();
    }

    handled<C, L, S, context_t> = true;
  }
};
```

Rework the `debug_handler`

```
1 struct debug_handler {
2   template <stdx::ct_string C, stdx::ct_string L, stdx::ct_string S,
3             typename Ctx, typename... Args>
4   constexpr auto signal(Args &&...) -> void {
5     using context_t = extract_context_t<Ctx>;
6     using tag_t = typename context_t::tag_t;
7     using sndrs_t = typename context_t::sndrs_t;
8
9     if constexpr (S == stdx::ct_string("start") && contains_v<reset_recursively_t, tag_t>) {
10      recursively_reset_handled<C, context_t, sndrs_t>();
11    }
12
13    handled<C, L, S, context_t> = true;
14  }
15 };
```

Recursively Walk the Type Onion

```
template <stdx::ct_string C, typename Ctx, typename Graph>
auto recursively_reset_handled = [](){

  reset_handled<C, Ctx>();

  using sub_nodes_t = sub_senders_t<Graph>;

  []<template <class...> class List, typename... Subs>(List<Subs...>){
    (..., recursively_reset_handled<C, extract_context_t<Subs>, Subs>());
  }(sub_nodes_t{});
};
```

Recursively Walk the Type Onion

```
1 template <stdx::ct_string C, typename Ctx, typename Graph>
2 auto recursively_reset_handled = [](){
3
4   reset_handled<C, Ctx>();
5
6   using sub_nodes_t = sub_senders_t<Graph>;
7
8   []<template <class...> class List, typename... Subs>(List<Subs...>){
9     (..., recursively_reset_handled<C, extract_context_t<Subs>, Subs>());
10  }(sub_nodes_t{});
11 };
```

Recursively Walk the Type Onion

```
1  template <stdx::ct_string C, typename Ctx, typename Graph>
2  auto recursively_reset_handled = [](){
3
4    reset_handled<C, Ctx>();
5
6    using sub_nodes_t = sub_senders_t<Graph>;
7
8    []<template <class...> class List, typename... Subs>(List<Subs...>){
9      (..., recursively_reset_handled<C, extract_context_t<Subs>, Subs>());
10   }(sub_nodes_t{});
11 };
```

Recursively Walk the Type Onion

```
template <stdx::ct_string C, typename Ctx, typename Graph>
auto recursively_reset_handled = [](){

  reset_handled<C, Ctx>();

  using sub_nodes_t = sub_senders_t<Graph>;

  []<template <class...> class List, typename... Subs>(List<Subs...>){
    (..., recursively_reset_handled<C, extract_context_t<Subs>, Subs>());
  }(sub_nodes_t{});
};
```

Recursively Walk the Type Onion

```
template <stdx::ct_string C, typename Ctx, typename Graph>
auto recursively_reset_handled = [](){

  reset_handled<C, Ctx>();

  using sub_nodes_t = sub_senders_t<Graph>;

  []<template <class...> class List, typename... Subs>(List<Subs...>){
    (..., recursively_reset_handled<C, extract_context_t<Subs>, Subs>());
  }(sub_nodes_t{});
};
```

Recursively Walk the Type Onion

```
1 template <typename ...T>
2 using sub_senders_t =
3   mp11::mp_flatten<
4     mp11::mp_list< typename sub_senders<T>::type... >>;
```

default

```
template <typename T>
struct sub_senders {
  using type = mp11::mp_list<>;
};
```

```
async::_repeat::sender<"repeat",
  async::_then::sender<"last-then", async::set_value_t,
    async::_sequence::sender<"seq",
      async::_then::sender<"before-wa-then", async::set_value_t,
        async::_just::sender<"link-start", async::set_value_t, int>, things::$_1>,
      async::_sequence::detail::wrapper<
        async::_when_all::sender<"when_all",
          async::_when_all::sub_sender<
            async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
          async::_when_all::sub_sender<
            async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
          async::_when_all::sub_sender<
            async::_then::sender<"then", async::set_value_t,
              async::_just::sender<"just-a2", async::set_value_t, int>,
              things::$_0>,
            2ul>>>>,
    things::$_2>,
  async::_repeat::$_3>
```

_then

```
template <stdx::ct_string Name, typename Tag, typename Sndr, typename... Ts>
struct sub_senders <async::_then::sender<Name, Tag, Sndr, Ts...>> {
  using type = mp11::mp_list<Sndr>;
};
```

```
async::_repeat::sender<"repeat",
  async::_then::sender<"last-then", async::set_value_t,
    async::_sequence::sender<"seq",
      async::_then::sender<"before-wa-then", async::set_value_t,
        async::_just::sender<"link-start", async::set_value_t, int>, things::$_1>,
      async::_sequence::detail::wrapper<
        async::_when_all::sender<"when_all",
          async::_when_all::sub_sender<
            async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
          async::_when_all::sub_sender<
            async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
          async::_when_all::sub_sender<
            async::_then::sender<"then", async::set_value_t,
              async::_just::sender<"just-a2", async::set_value_t, int>,
              things::$_0>,
            2ul>>>>,
    things::$_2>,
  async::_repeat::$_3>
```

_sequence

```
template <stdx::ct_string Name, typename SndrA, typename SndrB>
struct sub_senders <async::_sequence::sender<Name,
                                             SndrA,
                                             async::_sequence::detail::wrapper<SndrB>>>{
  using type = mp11::mp_list<SndrA, SndrB>;
};
```

```
async::_repeat::sender<"repeat",
  async::_then::sender<"last-then", async::set_value_t,
    async::_sequence::sender<"seq",
      async::_then::sender<"before-wa-then", async::set_value_t,
        async::_just::sender<"link-start", async::set_value_t, int>, things::$_1>,
      async::_sequence::detail::wrapper<
        async::_when_all::sender<"when_all",
          async::_when_all::sub_sender<
            async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
          async::_when_all::sub_sender<
            async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
          async::_when_all::sub_sender<
            async::_then::sender<"then", async::set_value_t,
              async::_just::sender<"just-a2", async::set_value_t, int>,
              things::$_0>,
            2ul>>>>,
    things::$_2>,
  async::_repeat::$_3>
```

_when_all

```
template <stdx::ct_string Name, typename... Sndrs, size_t ... N>
struct sub_senders <async::_when_all::sender<Name, async::_when_all::sub_sender<Sndrs, N>...>> {
  using type = mp11::mp_list<Sndrs...>;
};
```

```
async::_repeat::sender<"repeat",
  async::_then::sender<"last-then", async::set_value_t,
    async::_sequence::sender<"seq",
      async::_then::sender<"before-wa-then", async::set_value_t,
        async::_just::sender<"link-start", async::set_value_t, int>, things::$_1>,
      async::_sequence::detail::wrapper<
        async::_when_all::sender<"when_all",
          async::_when_all::sub_sender<
            async::_just::sender<"just-a0", async::set_value_t, int>, 0ul>,
          async::_when_all::sub_sender<
            async::_just::sender<"just-a1", async::set_value_t, int>, 1ul>,
          async::_when_all::sub_sender<
            async::_then::sender<"then", async::set_value_t,
              async::_just::sender<"just-a2", async::set_value_t, int>,
              things::$_0>,
            2ul>>>>,
    things::$_2>,
  async::_repeat::$_3>
```

demo with reset

demo with multiple threads

Will formalize the `Ctx` protocol

Summary

What

Declarative

Procedural

How

What

Easily
Composable

How

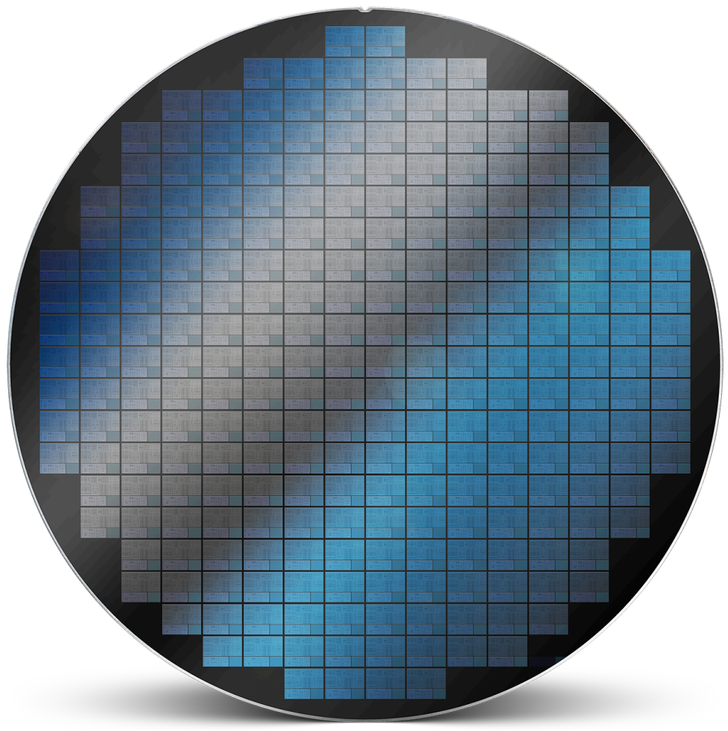
Sender Value Proposition

- Expressions not Statements
- Localized reasoning of concurrency
- Brings the events to the state

Raising the Level of Abstraction

Enabled:

- Structured concurrency
- "Structured" debugging
- "Structured" monitoring



Thank you

Questions?