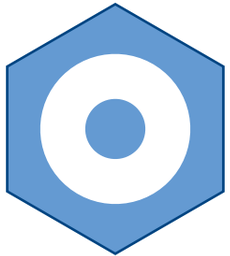


Sync & Send in Circle

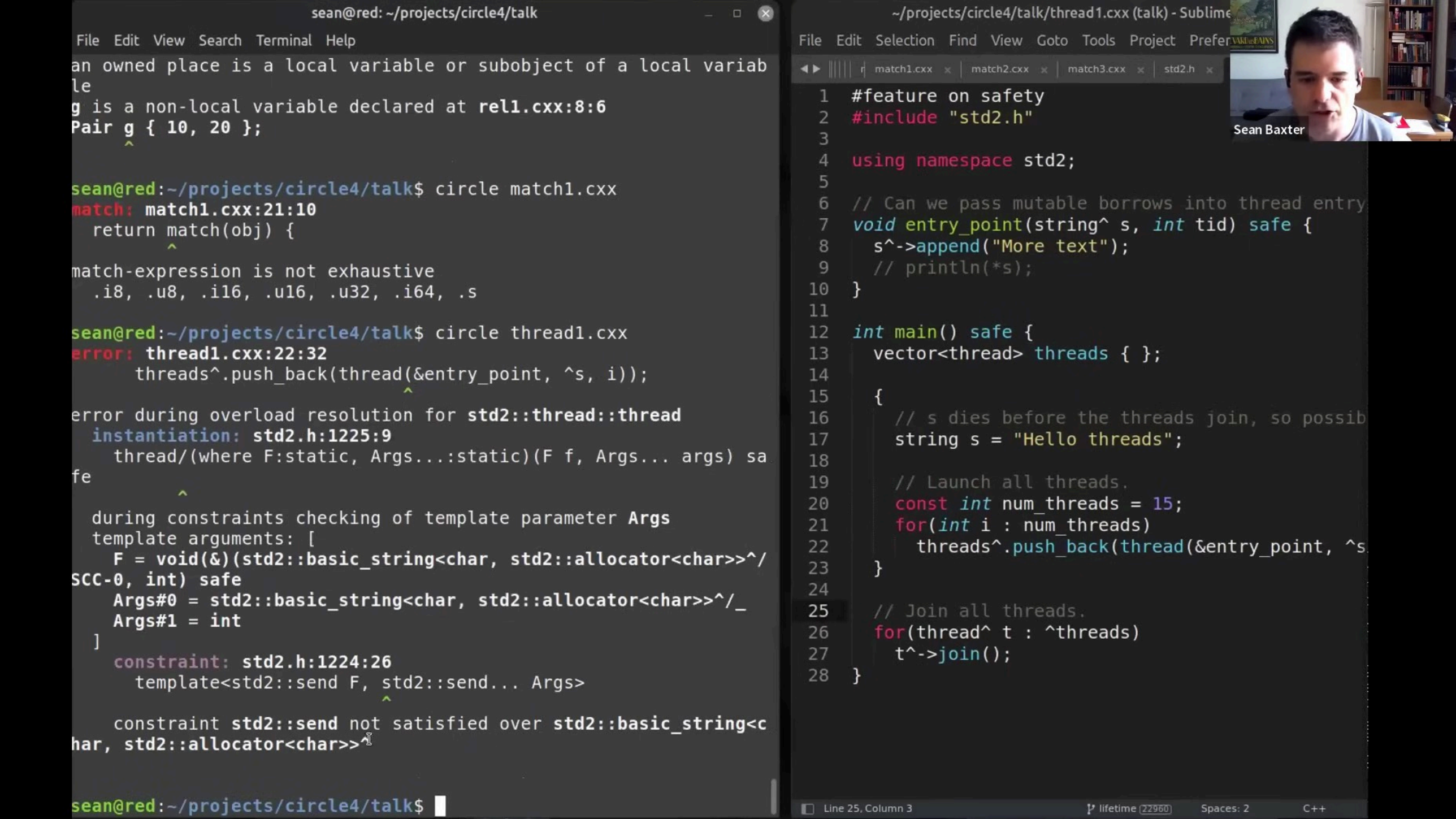
- **`send`** is a “marker interface” (in Rust a “marker trait”)
  - Similar to a C++ “type trait”

- Inferred if:
  - A copy can be made (value semantics)
  - A borrow can shared (`const T^`)
  - **NOT** mutable borrow (`T^`)

85

Sync & Send in Circle

- `send` is a “marker interface” (in Rust a “marker trait”)
  - Similar to a C++ “type trait”
- Inferred if:
  - A copy can be made (value semantics)
  - A borrow can shared (`const T^`)
  - **NOT** mutable borrow (`T^`)


sean@red: ~/projects/circle4/talk
File Edit View Search Terminal Help
an owned place is a local variable or subobject of a local variable
g is a non-local variable declared at rel1.cxx:8:6
Pair g { 10, 20 };
sean@red:~/projects/circle4/talk$ circle match1.cxx
match: match1.cxx:21:10
return match(obj) {
match-expression is not exhaustive
.i8, .u8, .i16, .u16, .u32, .i64, .s
sean@red:~/projects/circle4/talk$ circle thread1.cxx
error: thread1.cxx:22:32
threads^.push_back(thread(&entry_point, ^s, i));
error during overload resolution for std2::thread::thread
instantiation: std2.h:1225:9
thread/(where F:static, Args...:static)(F f, Args... args) safe
during constraints checking of template parameter Args
template arguments: [
F = void(&)(std2::basic_string<char, std2::allocator<char>>^/SCC-0, int) safe
Args#0 = std2::basic_string<char, std2::allocator<char>>^/_
Args#1 = int
]
constraint: std2.h:1224:26
template<std2::send F, std2::send... Args>
constraint std2::send not satisfied over std2::basic_string<char, std2::allocator<char>>^
sean@red:~/projects/circle4/talk$
~/projects/circle4/talk/thread1.cxx (talk) - Sublime
File Edit Selection Find View Goto Tools Project Prefer
match1.cxx
match2.cxx
match3.cxx
std2.h
1 #feature on safety
2 #include "std2.h"
3
4 using namespace std2;
5
6 // Can we pass mutable borrows into thread entry
7 void entry_point(string^ s, int tid) safe {
8 s^->append("More text");
9 // println(*s);
10 }
11
12 int main() safe {
13 vector<thread> threads { };
14
15 {
16 // s dies before the threads join, so possib
17 string s = "Hello threads";
18
19 // Launch all threads.
20 const int num_threads = 15;
21 for(int i : num_threads)
22 threads^.push_back(thread(&entry_point, ^s
23 }
24
25 // Join all threads.
26 for(thread^ t : ^threads)
27 t^->join();
28 }
Line 25, Column 3
lifetime 22960
Spaces: 2
C++
Sean Baxter