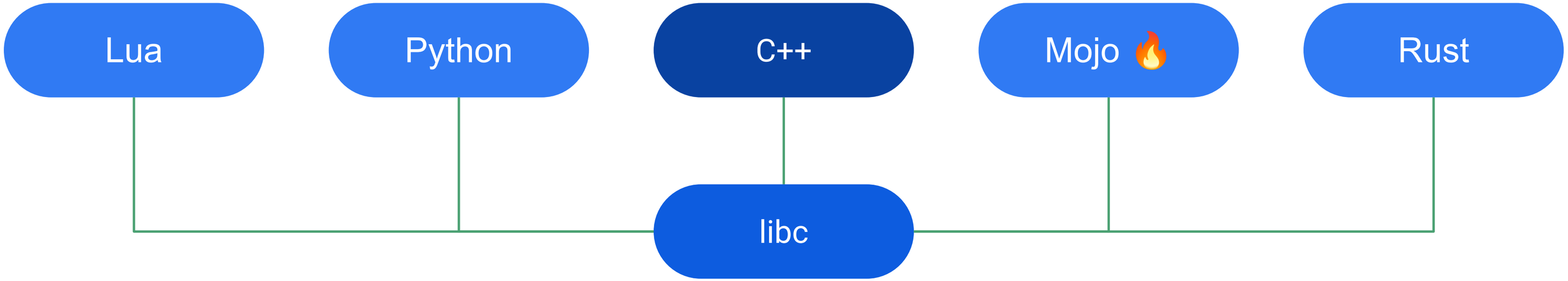
Project Hand-in-Hand

The beginning of a beautiful friendship

Why have a libc in LLVM?

What is the purpose of a libc?

High-level languages eventually make calls to libc



libc++ often defers to libc

Example: <cmath>

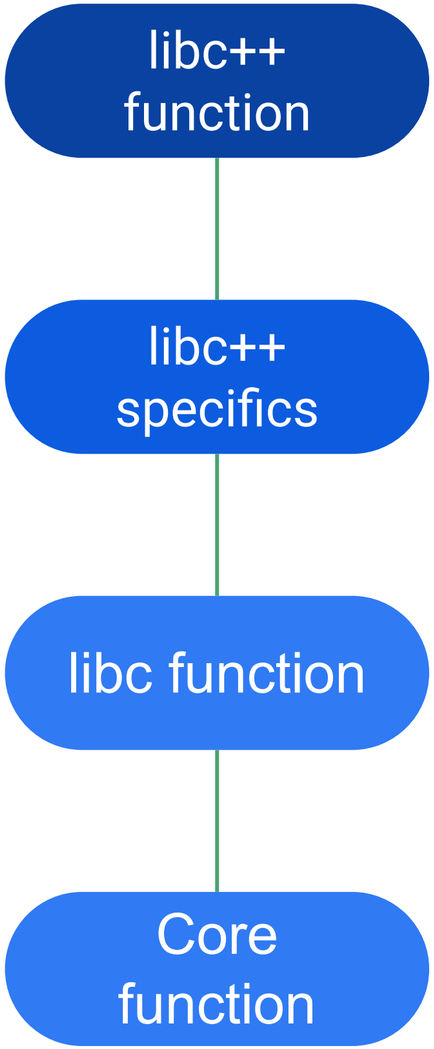
```
#include <math.h>

namespace std {
  using ::sinf;

  constexpr float sin(float x)
  {
    return sinf(x);
  }

  // ...
}
```

A normal pattern



LLVM-libc is implemented in freestanding C++

- Better syntax abstractions
- Cleaner function interface
- Designed to be modular

libc++'s missing feature

Floating-point numbers are hard

C++17 added `from_chars`

```
from_chars_result from_chars(const char* first, const char* last,
                             floating-point-type& value,
                             chars_format fmt = chars_format::general)
```

5 *Preconditions:* `fmt` has the value of one of the enumerators of `chars_format`.

6 *Effects*: The pattern is the expected form of the subject sequence in the `"C"` locale, as described for `strtod`, except that

(6.1) — the sign `'+'` may only appear in the exponent part;

(6.2) — if `fmt` has `chars_format::scientific` set but not `chars_format::fixed`, the otherwise optional exponent part shall appear;

(6.3) — if `fmt` has `chars_format::fixed` set but not `chars_format::scientific`, the optional exponent part shall not appear; and

(6.4) — if `fmt` is `chars_format::hex`, the prefix `"0x"` or `"0X"` is assumed.

[*Example 1:* The string `0x123` is parsed to have the value `0` with remaining characters `x123`.— *end example*]

In any case, the resulting `value` is one of at most two floating-point values closest to the value of the string matching the pattern.

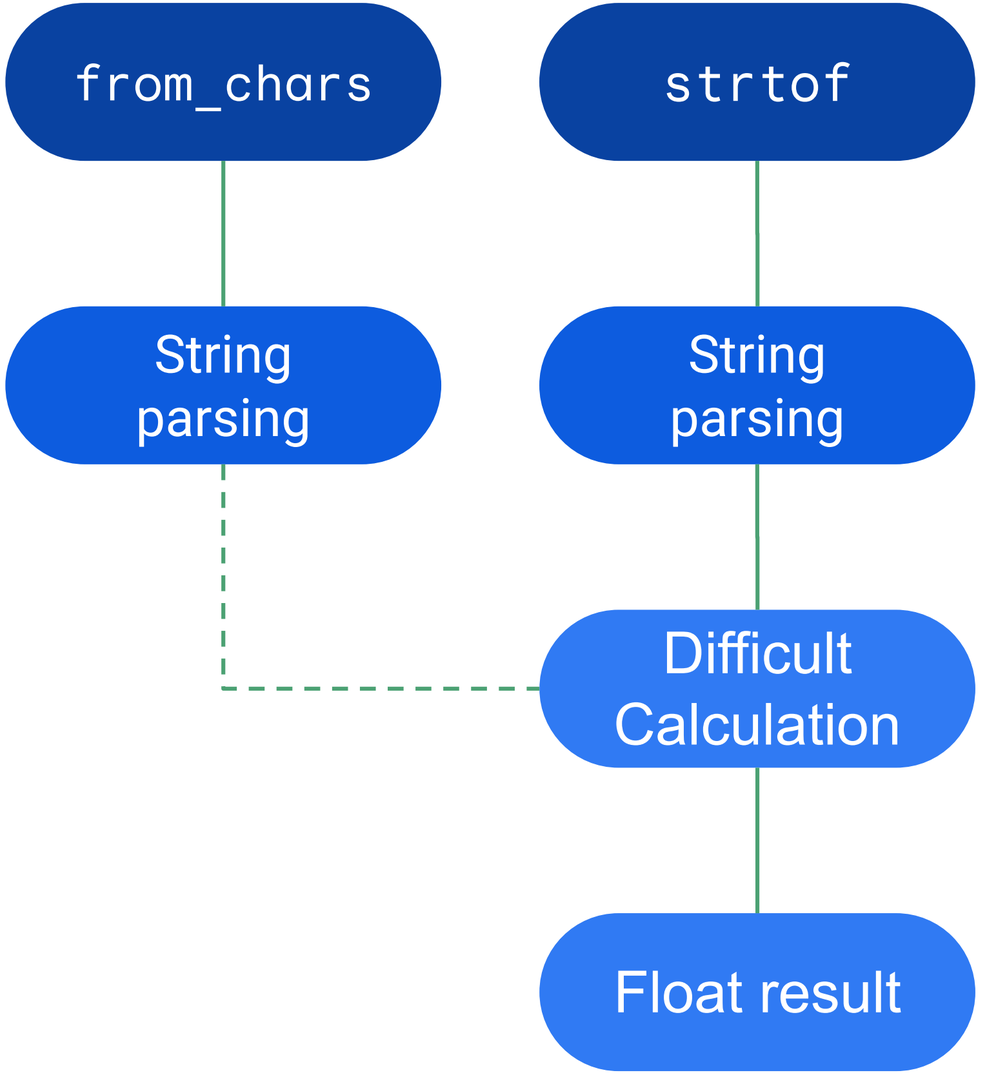
7 *Throws:* Nothing

SEE ALSO: ISO/IEC 9899:2018, 7.22.1.3, 7.22.1.4

But they have very different interfaces?

```
from_chars_result from_chars(
  const char* first,
  const char* last,
  floating-point-type& value,
  chars_format fmt);
```

```
double strtod(
  const char *restrict str,
  char **restrict end);
```



Why can't libc++ roll its own thing?

- String-to-float parsing is hard (like, really hard)[1] [2]
- The core functionality is the same (why rewrite ~2300 LoC?)
- If only libc had a different interface...

1. 'Approximating at Scale: How strto float in LLVM's libc is faster' by Michael Jones
2. 'Floating-Point ＜charconv＞: Making Your Code 10x Faster With C++17's Final Boss' by Stephan T. Lavavej

LLVM-libc already has that!

- String-to-float functions call a function template
- All implemented in headers
- No OS-specific dependencies

Extending a hand 

What if we worked together?

Can libc++ use LLVM-libc's code for `std::from_chars`?

From the libc++ side...

Needs to be user-transparent

⚠️ Caution! Unstable APIs! ⚠️

- Explicit and narrow interface
- Need a plan for API changes[3]
- If you randomly include LLVM-libc internals, Michael will scream 😱

3. 'What Does It Take to Implement the C++ Standard Library? (C++Now Edition)' by Christopher Di Bella

From the LLVM-libc side...

Code is written in standalone C++

- Similar API to what libc++ wants
- Well optimised[2]

Header-only: easy to include

```
1  #ifndef LLVM_LIBC_SHARED_STR_TO_INTEGER_H
2  #define LLVM_LIBC_SHARED_STR_TO_INTEGER_H
3
4  #include "src/__support/str_to_integer.h"
5
6  namespace LIBC_NAMESPACE_DECL {
7  namespace shared {
8
9  using LIBC_NAMESPACE::StrToNumResult;
10
11 using internal::strtointeger;
12
13 } // namespace shared
14 } // namespace LIBC_NAMESPACE_DECL
15
16 #endif // LLVM_LIBC_SHARED_STR_TO_INTEGER_H
```

But what if it doesn't work?

Coupling APIs limits evolution

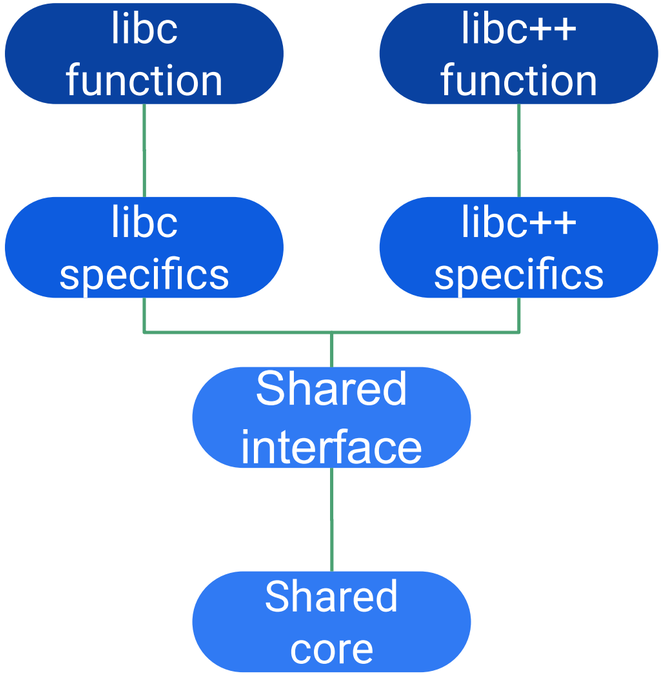
- Public dependencies are difficult to change
- glibc and libstdc++ tried interop with files and streams:

> *“There are additional benefits to this approach:*
>
> - *The old libstdc++ documentation calls this **the cool way**.*
>
> ...
>
> *The downside is a strong coupling between glibc and libstdc++, and the loss of optimization opportunities within the glibc stdio implementation....”*

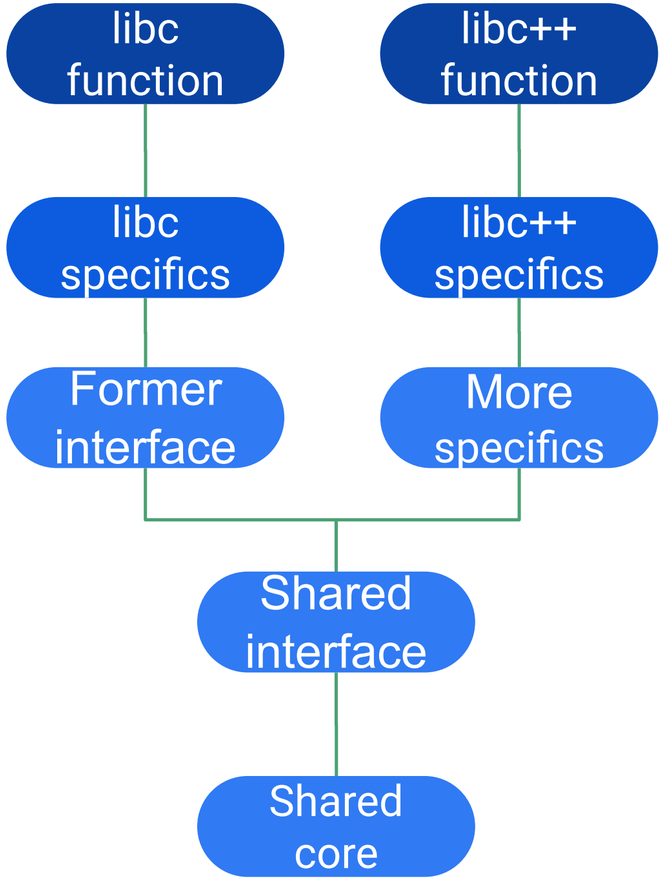
—libio vtables

What if libc and libc++ diverge?

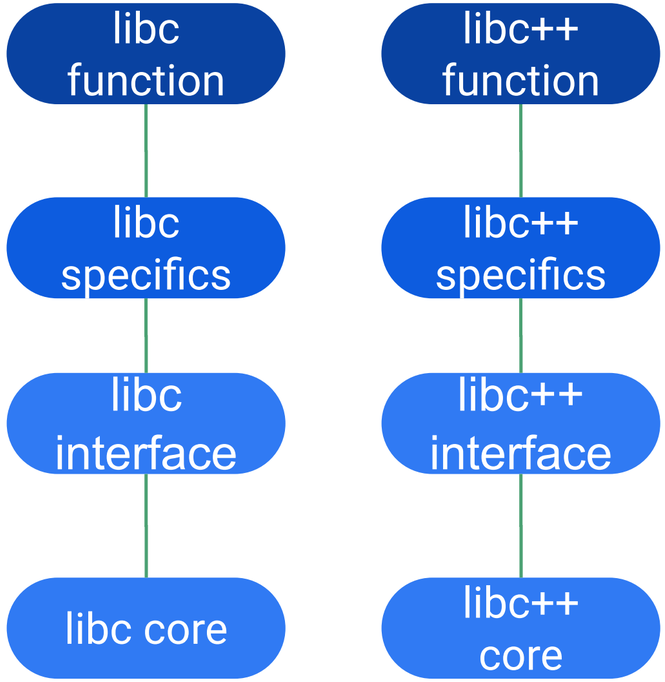
Existing



Sink common ancestor



Completely diverge



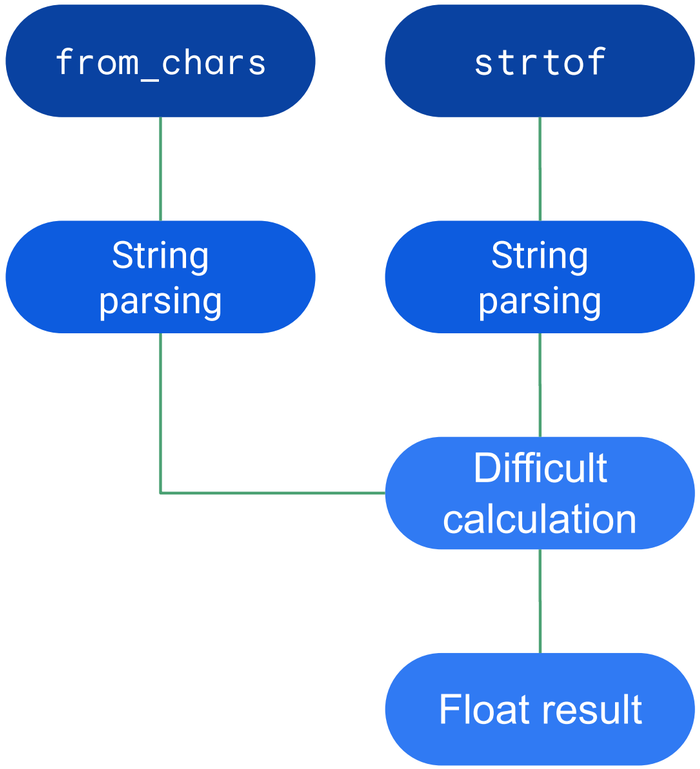
libc	[libc] Fix long double is double double const (#113258)
libclc	[libclc] Give a helpful error when an unknown target is req...
libcxx	[libc++] Refactor vector constructors to eliminate code d...
libcxxabi	[runtimes] Improve the documentation for LIBCXX_ADDIT...
libunwind	[libunwind][AIX] Call dlclose only when dlsym() fails (#112...
lld	[lld][WebAssembly] Improve -v/-V/--version flag compat (...
lldb	[lldb] Log errors to the system log if they would otherwise...
llvm-libgcc	[runtimes] Correctly apply libdir subdir for multilib (#933...
llvm	[DebugInfo] Emit linkage name into DWARF for types for ...

It seems like it'll work?

Only one thing left to do

🤝 So we did it! 🤝

- LLVM 20 ships floating-point `std::from_chars`
  - llvm/llvm-project#91651
- The project has been a success for both libc++ and LLVM-libc



Special thanks

- **Louis Dionne**: organisation, code review
- **Mark de Weaver**: wrote the libc++ parts
- Everyone who participated in the RFC and code reviews

Future work

- Find other places libc++ would benefit
- Support other parts of LLVM
- Build LLVM-libc/libc++/libunwind as one library

🤝 Project Hand-in-Hand 🤝