A freestanding Carol

Paul M. Bendixen

March 2024

Agenda

whoami

- Paul M. Bendixen
- Electronics Engineer by training
- Firmware pilot @ Trifork by day
- Part of SG14 almost since its inception

Story Time

For those who prefer doodling with crayons:

```
docker pull paulbendixen/freestanding-libstdcxx:14.0.1
```

Embedded C++ was dead as a door nail...

The ghost of emBO++ past

Freestanding

From the C++ 17 standard:

> *A freestanding implementation is one in which execution may take place without the benefit of an operating system*

Freestanding

From the C++ 17 standard:

> *A freestanding implementation is one in which execution may take place without the benefit of an operating system*

From the introduction of P0829:

> *This proposal seeks to refine the list of headers and declarations to include all parts of the hosted implementation that don't use problematic features.*

An implementation

From the motivation of P0829:

> *Systems programmers want to sort things. They want to use move semantics. They may even want to bundle the arguments of a variadic template function into a tuple.*

An implementation

From the motivation of P0829:

> *Systems programmers want to sort things. They want to use move semantics. They may even want to bundle the arguments of a variadic template function into a tuple.*

Show me the code!

An implementation

From the motivation of P0829:

> *Systems programmers want to sort things. They want to use move semantics. They may even want to bundle the arguments of a variadic template function into a tuple.*

Show me the code!

emBO++ 2019 – Paul M. Bendixen

Freestanding on the shoulders of giants

`https://gitlab.com/avr-libstdcxx`

An implementation

From the motivation of P0829:

> *Systems programmers want to sort things. They want to use move semantics. They may even want to bundle the arguments of a variadic template function into a tuple.*

Show me the code!
emBO++ 2019 – Paul M. Bendixen
Freestanding on the shoulders of giants
`https://gitlab.com/avr-libstdcxx`

Standardizing

The timeline for P0829:

October 2017 P0829R0 is submitted

Standardizing

The timeline for P0829:

October 2017	P0829R0 is submitted
March 2019	P0829R4 Goes to Hawaii Needs revision

Standardizing

The timeline for P0829:

October 2017	P0829R0 is submitted
March 2019	P0829R4 Goes to Hawaii Needs revision
May 2020	P0829 Presented to LEWG

Standardizing

The timeline for P0829:

October 2017	P0829R0 is submitted
March 2019	P0829R4 Goes to Hawaii Needs revision
May 2020	P0829 Presented to LEWG
February 2021	P0829 declared dead, to be split up

The proposals today

Paper	Title	status
P2268	Freestanding Roadmap	Open
P2013	F... Optional ::operator new	Merged
P1642	F... Library: Easy [utilities], [ranges], and [iterators]	Merged
P2198	F... Feature-Test Macros and Implementation-Defined Extensions	Merged
P2338	F... Library: Character primitives and the C library	Merged
P2407	F... Library: Partial Classes	Merged
P2833	F... Library: inout expected span	LWG
P2937	F...: Remove strtok	Merged
P2976	F... Library: algorithm, numeric, and random	LWG

Any pulished paper can be found using: `http://wg21.link/{}`

Select papers

Paper	Title	status
P1642	Freestanding Library: Easy [utilities], [ranges], and [iterators]	Merged

Select papers

Paper	Title	status
P1642	Freestanding Library: Easy [utilities], [ranges], and [iterators]	Merged
P2338	Freestanding Library: Character primitives and the C library	Merged

Select papers

Paper	Title	status
P1642	Freestanding Library: Easy [utilities], [ranges], and [iterators]	Merged
P2338	Freestanding Library: Character primitives and the C library	Merged
P2407	Freestanding Library: Partial Classes	Merged

Select papers

Paper	Title	status
P1642	Freestanding Library: Easy [utilities], [ranges], and [iterators]	Merged
P2338	Freestanding Library: Character primitives and the C library	Merged
P2407	Freestanding Library: Partial Classes	Merged
P2976	Freestanding Library: algorithm, numeric, and random	LWG

Select papers

Paper	Title	status
P1642	Freestanding Library: Easy [utilities], [ranges], and [iterators]	Merged
P2338	Freestanding Library: Character primitives and the C library	Merged
P2407	Freestanding Library: Partial Classes	Merged
P2976	Freestanding Library: algorithm, numeric, and random	LWG
P2833	Freestanding Library: inout expected span	LWG

Select papers

Paper	Title	status
P1642	Freestanding Library: Easy [utilities], [ranges], and [iterators]	Merged
P2338	Freestanding Library: Character primitives and the C library	Merged
P2407	Freestanding Library: Partial Classes	Merged
P2976	Freestanding Library: algorithm, numeric, and random	LWG
P2833	Freestanding Library: inout expected span	LWG

Would have been nice to have the freestanding features done in fewer papers to save time for LEWG.
— Weakly Favor

Todays implementation

Implemented in the Freestanding library `avr-libstdc++`:

Paper	Title	status
P2013	F. . . Optional ::operator new	Merged
P1642	F. . . Library: Easy [utilities], [ranges], and [iterators]	Merged
P2338	F. . . Library: Character primitives and the C library	Merged
P2407	F. . . Library: Partial Classes	Merged

Todays implementation

Implemented in the Freestanding library `avr-libstdc++`:

Paper	Title	status
P2013	F... Optional ::operator new	Merged
P1642	F... Library: Easy [utilities], [ranges], and [iterators]	Merged
P2338	F... Library: Character primitives and the C library	Merged
P2407	F... Library: Partial Classes	Merged
P2198	F... Feature-Test Macros and Implementation-Defined Extensions	Merged
P2338	F... Library: Character primitives and the C library	Merged
P2833	F... Library: inout expected span*	LWG
P2976	F... Library: algorithm, numeric, and random	LWG

* `mdspan` is not implemented in libstdc++ so it is also missing here

Using it today

```
docker pull paulbendixen/freestanding-libstdcxx:14.0.1
```

Contains:

- avr-g++
- arm-none-eabi-g++
- Ninja
- CMake
- CMake Toolchain files

What is yet to come

- `<chrono>`
- `<random>`
- Things coming down the line

emBO++30

Possible titles:

- No more raw loops (except `while(1)`)

emBO++30

Possible titles:

- No more raw loops (except `while(1)`)
- Embed some signal processing in standard C++

emBO++30

Possible titles:

- No more raw loops (except `while(1)`)
- Embed some signal processing in standard C++
- Using senders and receivers with interrupts

emBO++30

Possible titles:

- No more raw loops (except `while(1)`)
- Embed some signal processing in standard C++
- Using senders and receivers with interrupts
- Microcontroller modules

Modules

```
import std.freestanding;
int fib = 0;

int main(void)
{
    std::pair<int, int> variable{1, 0};

    while( variable.first )
    {
        std::swap( variable.first, variable.second );
        variable.first += variable.second;
        *const_cast<volatile int*>(&fib) = variable.first;
    }

}
```

Some people laughed to see the alteration in him, but he let them laugh, and little heeded them;
for he was wise enough to know that nothing ever happened on this globe, for good, at which some people did not have their fill of laughter in the outset.

Questions?