Literal suffixes for `size_t` and `ptrdiff_t`

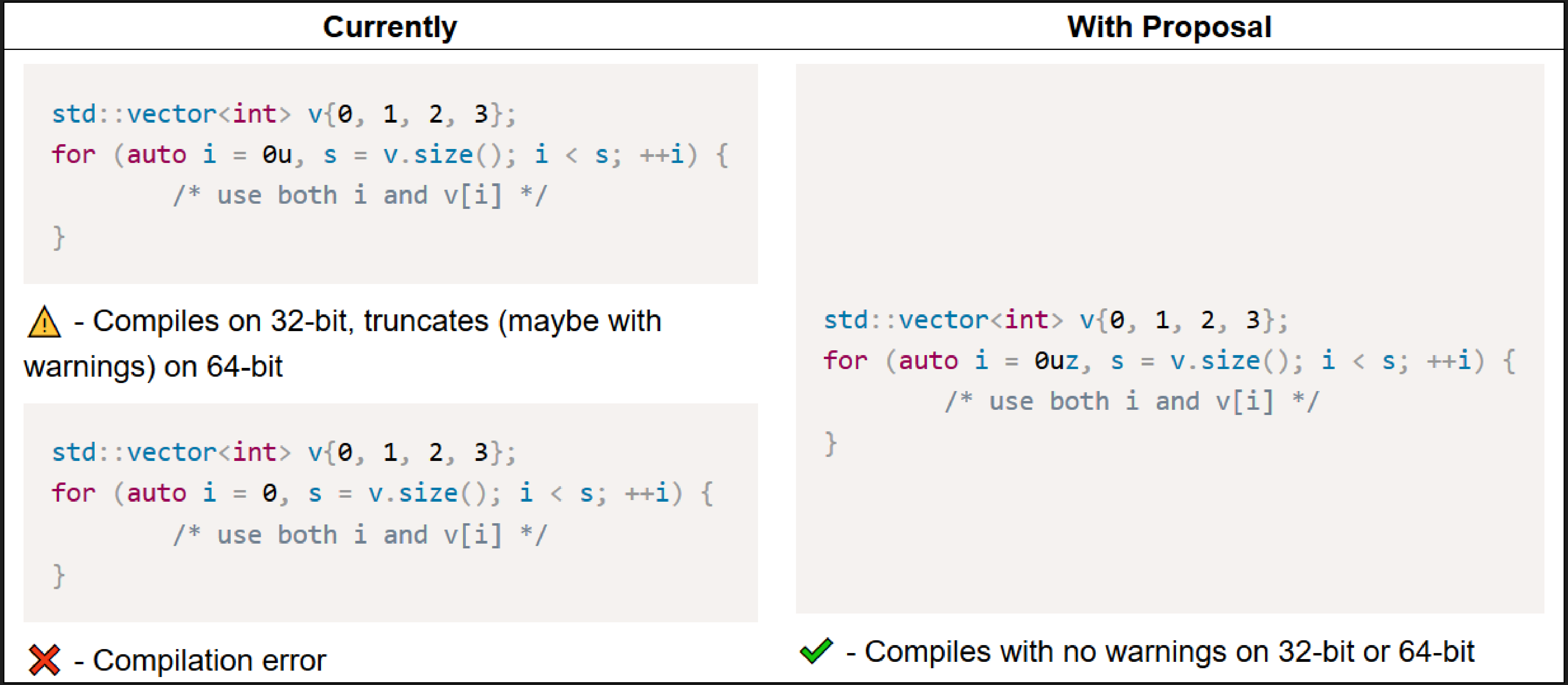
P0330 – A C++20 Paper

JeanHeyd Meneide

phdofthehouse@gmail.com – Twitter: @thephantomderp

https://thephd.github.io/vendor/future_cxx/papers/d0330.html

Before/After



Currently

```
std::vector<int> v{0, 1, 2, 3};
for (auto i = 0u, s = v.size(); i < s; ++i) {
        /* use both i and v[i] */
}
```

⚠ - Compiles on 32-bit, truncates (maybe with warnings) on 64-bit

```
std::vector<int> v{0, 1, 2, 3};
for (auto i = 0, s = v.size(); i < s; ++i) {
        /* use both i and v[i] */
}
```

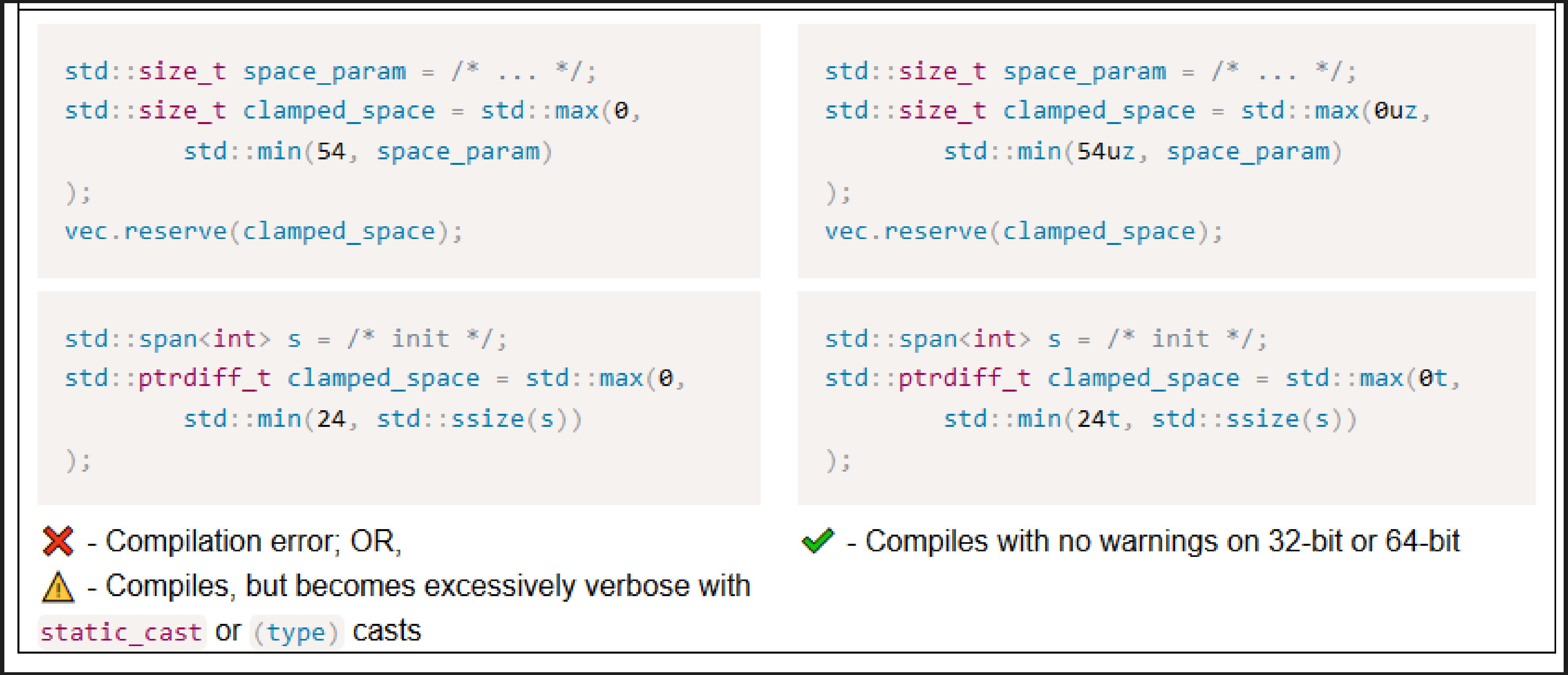
❌ - Compilation error

With Proposal

```
std::vector<int> v{0, 1, 2, 3};
for (auto i = 0uz, s = v.size(); i < s; ++i) {
        /* use both i and v[i] */
}
```

✓ - Compiles with no warnings on 32-bit or 64-bit

Before/After



```cpp
std::size_t space_param = /* ... */;
std::size_t clamped_space = std::max(0,
        std::min(54, space_param)
);
vec.reserve(clamped_space);
```

```cpp
std::size_t space_param = /* ... */;
std::size_t clamped_space = std::max(0uz,
        std::min(54uz, space_param)
);
vec.reserve(clamped_space);
```

```cpp
std::span<int> s = /* init */;
std::ptrdiff_t clamped_space = std::max(0,
        std::min(24, std::ssize(s))
);
```

```cpp
std::span<int> s = /* init */;
std::ptrdiff_t clamped_space = std::max(0t,
        std::min(24t, std::ssize(s))
);
```

❌ - Compilation error; OR,

⚠ - Compiles, but becomes excessively verbose with `static_cast` or `(type)` casts

✓ - Compiles with no warnings on 32-bit or 64-bit

Literals are Sensitive to Implementations

- 32 bit vs. 64 bit
- Signed vs. Unsigned
- Accidentally degrading math results by using wrong-sized value (VC++ streams…)
- Size issues from one compiler/platform to another (long on GCC versus VC++)
- Worked around with…
  - Macros
  - static_casts for template arguments, or providing explicit template arguments
  - Explicitly forbidding use of regular types and only use std:: types

History

- Originally proposed as Library User-Defined Literals in C++17 timeframe
  - Written by Rein Halbersma, presented by Walter E. Brown
  - EWG: "We want to solve this problem with a language extension"
    - Strongly favored (2 | 15 | 0 | 2 | 2), Albuquerque 2017
- Last-minute course-change from LWG, just before it was actually accepted:
  - *"This is not a library concern (any more); EWG has taken ownership. The types in question (size_t and ptrdiff_t) are not library types, but rather core language types."* – LWG, 2018

Design

- Uses the suffixes `z (signed size_t), t (ptrdiff_t)`
  - Cannot use `s`: is in clash with upcoming `sf` – `short float` paper
  - Avoid `sz`: clash with short float that makes `sz` harder to understand
  - Core language allows `u` to appear on either side of a literal

- `size_t (zu)`
  - Equivalent to type of `decltype(sizeof(0))`

- `ptrdiff_t (t)`
  - Equivalent to type of `decltype((char*)nullptr - (char*)nullptr)`

Design II

- "Two Suffixes? Aren't size_t and ptrdiff_t duals of one another?"
- Not necessarily: *arm7-apple-darwin* has
  - using size_t = unsigned long;
  - using ptrdiff_t = int;

Wording Complete

- Paper is Core-ready
  - Reviewed by Tim Song prior to meeting
  - Reviewed by Walter E. Brown prior to meeting
  - Despite not having a standard library typedef for "corresponding signed/unsigned type of size_t/ptrdiff_t", Hubert Tong and Jens Maurer donated wording