Document	P1337R0
Date	2019-04-01
Author	CJ Johnson <johnsoncj@google.com>
Audience	Library Evolution Working Group (LEWG)

Aliasing the standard library as a means to save C++

Background

With the advent of high-performance and scalable development tools such as Node.js and Ruby on Rails, C++ has been losing out in the market. The best developers, referred to as 10Xers, no longer use C++ when building out their Web 3.0 applications and systems. C++ used to be the go-to language but in recent years has fallen behind. If something doesn't change, C++ as a language may cease to exist. The Abseil team at Google is committed to course-correcting C++ so that all developers, even the best of the best, may use it fully.

Proposal 1337

We, the Abseil team, propose that, as of C++2a and forevermore, a second namespace be added to the standard library. The namespace shall be `57d::` and all symbols in namespace `57d::` shall be aliases of names in namespace `std::`. All aliases in `57d::` shall be formed from their `std::` counterpart and transformed via a standard 1337-speak mapping. No symbol shall exist in `std::` without a properly mapped `57d::` alias.

What is 1337-speak?

1337-speak is a form of character substitution where certain ASCII [1] letters are replaced with corresponding numbers. It's commonly used by internet hackers and genius programmers to speak in code in a way that feels natural to them.

Examples

1. `std::aligned_storage` aliased as `57d::4116n3d_570r463`
2. `std::index_sequence_for` aliased as `57d::1nd3x_539u3nc3_f0r`
3. `std::uninitialized_default_construct_n` aliased as `57d::un1n17141123d_d3f4u17_c0n57ruc7_n`

Precedent

- namespace `std2::` [2]

Justification

It is a known issue in C++ that importing `<windows.h>` can leak the `min(...)` and `max(...)` function-style macros into your program. `-DNOMINMAX` has been the standard solution to this for a while. With the adoption of this proposal, build flags will no longer be required as C++ developers everywhere will be able to use `57d::m1n(...)` and `57d::m4x(...)` in their code without fear.

0DR and U8

To prevent One Definition Rule (0DR) violations, all names in namespace `57d::` must be type aliases of names in `std::`. However, it shall be Undefined Behavior (U8) to convert, implicitly or explicitly, instances of types in one namespace to types in the other. That is to say `std::is_same<std::in_place_type_t, 57d::1n_p14c3_7yp3_7>::value` shall be define as `true` but `std::in_place_type_t x = 57d::1n_p14c3_7yp3_7{};` may never appear in a valid, standards-compliant C++ program. We expect this to cause limited programmer error in practice as 10Xers are very careful and only 10Xers will feel the need to use types in namespace `57d::`.

Leading numeric characters in symbols

As of C++17, symbols with leading numeric characters are invalid. This will be fixed in C++2a with the addition of modules.

Proposed mapping

It is the opinion of the Abseil team that the Standard Library's `57d::` 1337-speak symbol aliases should be formed using the below table as a mapping. For all names in the standard library, for all letters in each name, substitute the letters in the left column with the characters in the right column. Further, the mapping shall be applied to all public member functions, fields and type aliases of everything in namespace `57d::` recursively.

Original Letters	Corresponding 1337-speak Alias Characters
a	**4** *
A	**4** *
b	**8** *
B	**8** *
c	c
C	C
d	d
D	D
e	**3** *
E	**3** *
f	f
F	F
g	**6** *
G	**6** *
h	h
H	H
i	**1** *
I	**1** *
j	j
J	J
k	k
K	K
l	**1** *

Original Letters	Corresponding 1337-speak Alias Characters
L	**1** *
m	m
M	M
n	n
N	N
o	**0** *
O	**0** *
p	p
P	P
q	**9** *
Q	**9** *
r	r
R	R
s	**5** *
S	**5** *
t	**7** *
T	**7** *
u	u
U	U
v	v
V	V
w	w
W	W
x	x
X	X
y	y
Y	Y
z	**2** *
Z	**2** *

* Denotes mappings that differ from the identity function

Future Additions

Once adopted and implemented, if users are happy with namespace `57d::`, 1337-speak aliases could be expanded to the language keywords. Why worry about `co_*` when coroutines could safely use `4w417` (await), `y131d` (yield) and `r37urn` (return) keywords?

Open Questions

- Should name mangling now be done in base64?
- Should unqualified P051X (POSIX) names be given unqualified 1337-speak aliases for C-programming 10Xers?

References

[1] HTML ASCII Reference at w3schools.com

[2] P0180: Reserve a New Library Namespace Future Standardization

[3] C++ Logo ASCII Art