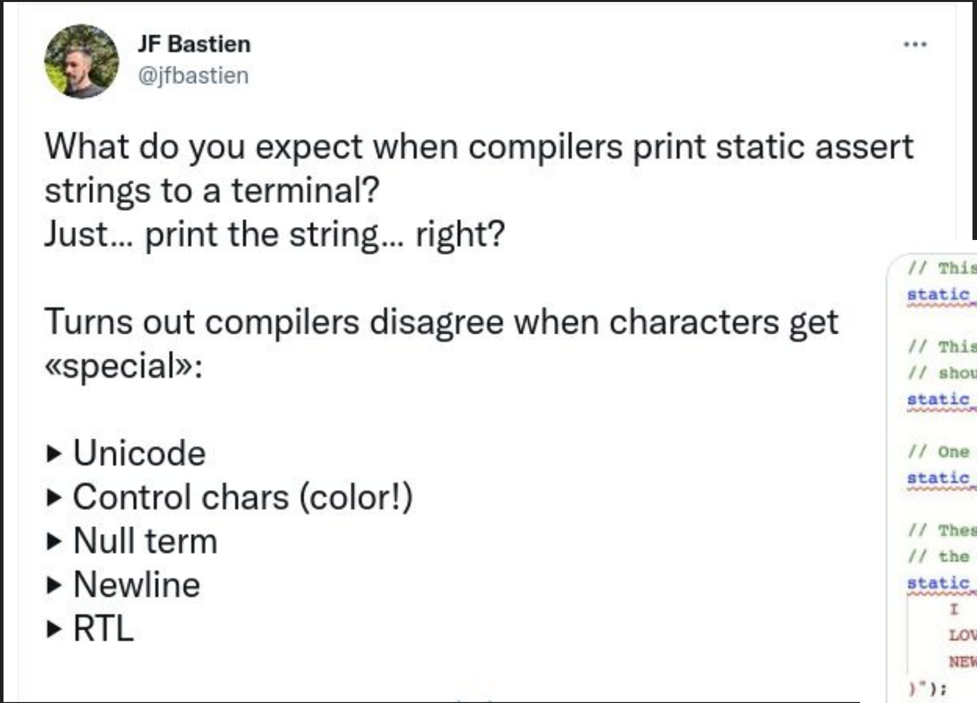
Unevaluated string literals

https://wg21.link/p2361

Corentin Jabot, Aaron Ballman

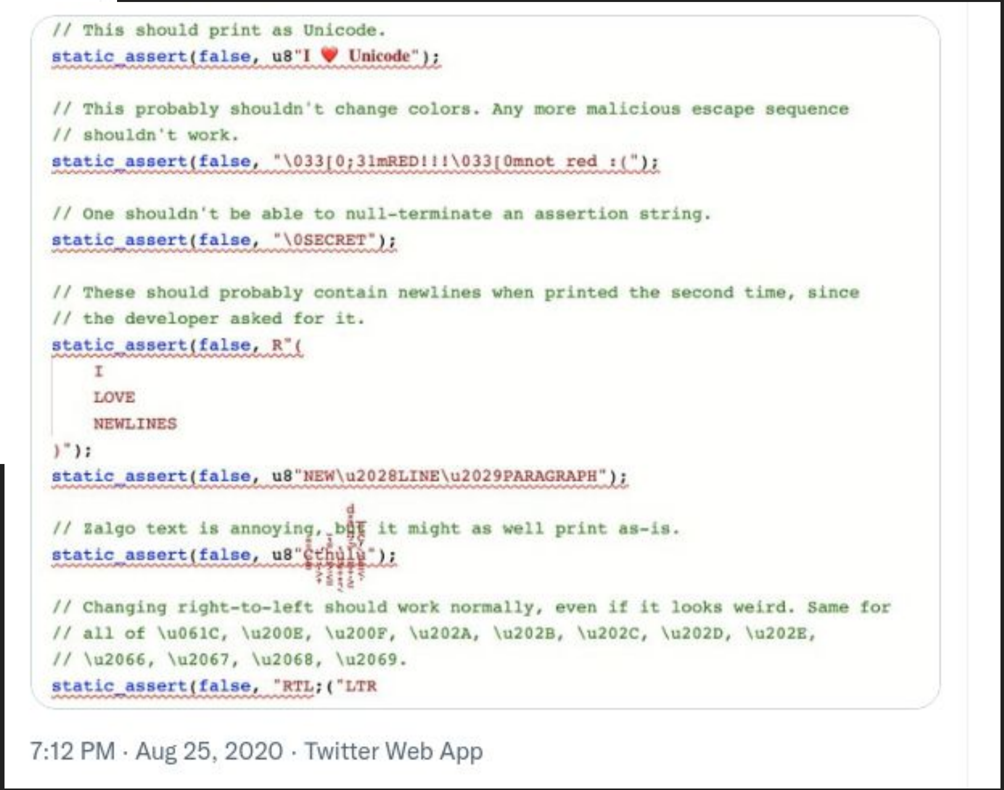


JF Bastien
@jfbastien

What do you expect when compilers print static assert strings to a terminal?
Just… print the string… right?

Turns out compilers disagree when characters get «special»:

- ▸ Unicode
- ▸ Control chars (color!)
- ▸ Null term
- ▸ Newline
- ▸ RTL



```
// This should print as Unicode.
static_assert(false, u8"I ❤️ Unicode");

// This probably shouldn't change colors. Any more malicious escape sequence
// shouldn't work.
static_assert(false, "\033[0;31mRED!!!\033[0mnot red :(");

// One shouldn't be able to null-terminate an assertion string.
static_assert(false, "\0SECRET");

// These should probably contain newlines when printed the second time, since
// the developer asked for it.
static_assert(false, R"(
    I
    LOVE
    NEWLINES
)");
static_assert(false, u8"NEW\u2028LINE\u2029PARAGRAPH");

// Zalgo text is annoying, but it might as well print as-is.
static_assert(false, u8"Cthulu");

// Changing right-to-left should work normally, even if it looks weird. Same for
// all of \u061C, \u200E, \u200F, \u202A, \u202B, \u202C, \u202D, \u202E,
// \u2066, \u2067, \u2068, \u2069.
static_assert(false, "RTL;("LTR
```

7:12 PM · Aug 25, 2020 · Twitter Web App

Goals

- Identifying where string-literals are unevaluated
- Have consistent restrictions on these literals
- Have consistent rules for the use of prefixes

Previously

- Character encoding of diagnostic text https://wg21.link/p2246 (Aaron Ballman)
- The wording already state that all the phase 5 transformation happens when strings are initialized

Unevaluated string Literals

- `_Pragma`
- `#line directives`
- `[[nodiscard]] and [[deprecated]] attributes`
- `extern linkage specifications`
- `asm statements`
- `static_assert`

Restrictions

- ❌ `numeric escape sequences (including \0)`
- ❌ `conditional escape sequences`
- ✅ `UCNs`
- ✅ `Normal escape sequences`

Encoding prefixes

	Standard	Implementations
`_Pragma`	L allowed Everything allowed in C	Clang supports everything MSVC supports nothing GCC supports L
`static_assert`	Allowed, presumably	All compilers allow a prefix MSVC converts to the associated encoding
Attributes	Allowed, presumably	Clang reject prefixes Other compiler reject prefixes
extern&asm	Allowed, presumably	All compilers reject prefixes
#line	Disallowed, maybe?	All compilers except MSVC reject prefixes

What about users?

Users don't use prefixes in any of these cases

Number of strings with encoding prefix in `_Pragma`: 3/3383 (all in test suits)

Number of strings with encoding prefix in `deprecated/nodiscard` attributes: 0/845

Number of strings with encoding prefix in `static_assert`: 62/92800 (all in in test suits)

Number of strings with encoding prefix in `extern` : 3/39829 (all in in test suits)

Proposal

Never allow prefixes

- Simpler
- More consistent model (not encoded, no encoding prefix)

Future work

static_assert-declaration:

static_assert (constant-expression) ;

static_assert (constant-expression , unevaluated-string) ;   // Unicode

static_assert (constant-expression , constant-expression) ; // literal encoding

Implementation

- 2 PRs in clang
- That work proved necessary to support EBCDIC in clang, as clang would eagerly convert `static_assert` messages to the literal encoding, which would break when more encodings are added