C++ IDENTIFIERS USING UAX 31

STEVE DOWNEY

Created: 2020-09-24 Thu 22:49

TABLE OF CONTENTS

- C++ Identifier Syntax using Unicode Standard Annex 31
- The Emoji Problem
- Script Issues
- Other adopters

C++ IDENTIFIER SYNTAX USING UNICODE STANDARD ANNEX 31

- That C++ identifiers match the pattern

  (XID_Start + _) + XID_Continue.*

- That portable source is required to be normalized as NFC.
- That using unassigned code points be ill-formed.

PROBLEM THIS FIXES : NL 029

Allowed characters include those from U+200b until U+206x; these are zero-width and control characters that lead to impossible to type names, indistinguishable names and unusable code & compile errors (such as those accidentally including RTL modifiers).

STATUS QUO: WE ALLOW OTHER "WEIRD IDENTIFIER CODE POINTS"

- The middle dot · which looks like an operator.
- Many non-combining "modifiers" and accent marks, such as ´ and ¨ and ., which don't really make sense on their own.
- "Tone marks" from various languages, including ꜒ (similar to a box-drawing character ╞ which is an operator).
- The "Greek question mark" ; (see below)
- Symbols which are simply not linguistic, such as ⯎ and ⸿.

https://gist.github.com/jtbandes/c0b0c072181dcd22c3147802025d0b59#weird-identifier-code-points

UAX 31 - UNICODE IDENTIFIER AND PATTERN SYNTAX

- Follows the same principles as originally used for C++
- Actively maintained
- Stable

XID_START AND XID_CONTINUE

- Unicode database defined properties
- Closed under normalization for all four forms
- Once a code point has the property it is never removed
- Roughly:
  - Start == letters
  - Continue == Start + numbers + some punctuation

THE EMOJI PROBLEM

- The emoji-like code points that we knew about were excluded
- We included all unassigned code points
- Status Quo Emoji 'support' is an accident, incomplete, and broken

STATUS QUO IS BROKEN

SOME STATUS QUO EXAMPLES

Not Valid	Valid
int 🐻 = 0;	int 🕐 = 0;
int ☠️ = 0;	int 💀 = 0;
int ✋ = 0;	int ✊ = 0;
int 🛩 = 0;	int 🚀 = 0;
int ☹ = 0;	int 😀 = 0;

When the character was added to Unicode controls validity

STATUS QUO: ♀ AND ♂ ARE DISALLOWED

Gendered variants of emoji are selected by using a zero width joiner together with the male and female sign.

```
// Valid
    bool 👷 = true; //  Construction Worker
// Not valid
    bool 👷 = false; // Woman Construction Worker ({Construction Worker}{ZWJ}{Female Sign})
```

PROBLEMS ADDING EMOJI AS IDENTIFIERS

EMOJI ARE COMPLEX

- Not just code points
- Need grapheme cluster analysis
- May incur costs even for code not using emoji

EMOJI ARE NOT "STABLE" IN UNICODE

From the emoji spec

> *isEmoji(🕴)=false for Emoji Version 5.0, but true for Version 11.0.*

It is possible that the emoji property could be removed.

IDENTIFYING EMOJI IS DIFFICULT

The unicode standard provides a regex that will reject non-emoji, but does not guarantee a valid emoji sequence.

```
\p{RI} \p{RI}
| \p{Emoji}
    ( \p{EMod}
    | \x{FE0F} \x{20E3}?
    | [\x{E0020}-\x{E007E}]+ \x{E007F} )?
    (\x{200D} \p{Emoji}
      ( \p{EMod}
      | \x{FE0F} \x{20E3}?
      | [\x{E0020}-\x{E007E}]+ \x{E007F} )?
    )*
```

It's not clear how much of the unicode database would be required for complete support.

UNICODE EMOJI

SOME SURPRISING THINGS ARE EMOJI

```
002A          ; Emoji                # E0.0   [1] (*)       asterisk
0030..0039    ; Emoji                # E0.0  [10] (0..9)    digit zero..digit nine
```

```
  {DIGIT ONE}{VARIATION SELECTOR-16}{COMBINING ENCLOSING KEYCAP} 1⃣

  {ASTERISK}{VARIATION SELECTOR-16}{COMBINING ENCLOSING KEYCAP} *⃣
```

```
/// would this be valid?
int 1⃣ = 1;
```

FIXING THE EMOJI PROBLEM WOULD MEAN BEING INVENTIVE

Being inventive in an area outside our expertise is HARD

Adopting UAX31 as a base to move forward is conservative

UAX 31 is a known good state

SCRIPT ISSUES

Some scripts require characters to control display or require punctuation that are not in the identifier set.

THIS INCLUDES ENGLISH

- Apostrophe and dash
  - `won't`
  - `can't`
  - `mustn't`
  - `mother-in-law`
- Programmers are used to this and do not notice

ZERO WIDTH CHARACTERS ARE EXCLUDED BY UAX 31

Status quo allows these invisible characters

```
int tmp = 0;
int tmp = 0;
```

- clang 10 warns

  <source>:2:6: warning: identifier contains Unicode character <U+200D> that is invisible in some environments [-Wunicode-zero-width]

int t<U+200D><U+200D>mp = 0;

ZWJ AND ZWNJ

However zero width joiner and non joiner are used in some scripts

Farsi word "names"
نامهای
NOON + ALEF + MEEM + HEH + ALEF + FARSI YEH
نامهای

Farsi word "a letter"
نامه‌ای
NOON + ALEF + MEEM + HEH + **ZWNJ** + ALEF + FARSI YEH
نامه‌ای

Anecdotally, these issues are understood and worked around

UAX 31 HAS AN EXPENSIVE SOLUTION

Identifiers can be checked for what script the code points in the identifier are used, and the rules for allowed characters can be tailored. This requires a Unicode database and would require extensive analysis during lexing.

SG 16 does not recommend this.

OTHER ADOPTERS

- Java (https://docs.oracle.com/javase/specs/jls/se15/html/jls-3.html#jls-3.8)
- Python 3 https://www.python.org/dev/peps/pep-3131/
- Erlang https://www.erlang.org/erlang-enhancement-proposals/eep-0040.html
- Rust https://rust-lang.github.io/rfcs/2457-non-ascii-idents.html
- JS https://tc39.es/ecma262/