Cross-platform file names in Rust

WTF-8: a wonderful and horrifying hack!

Simon Sapin, Mozilla Research

!!Con, 2015-05-16

1.0 released on Friday!

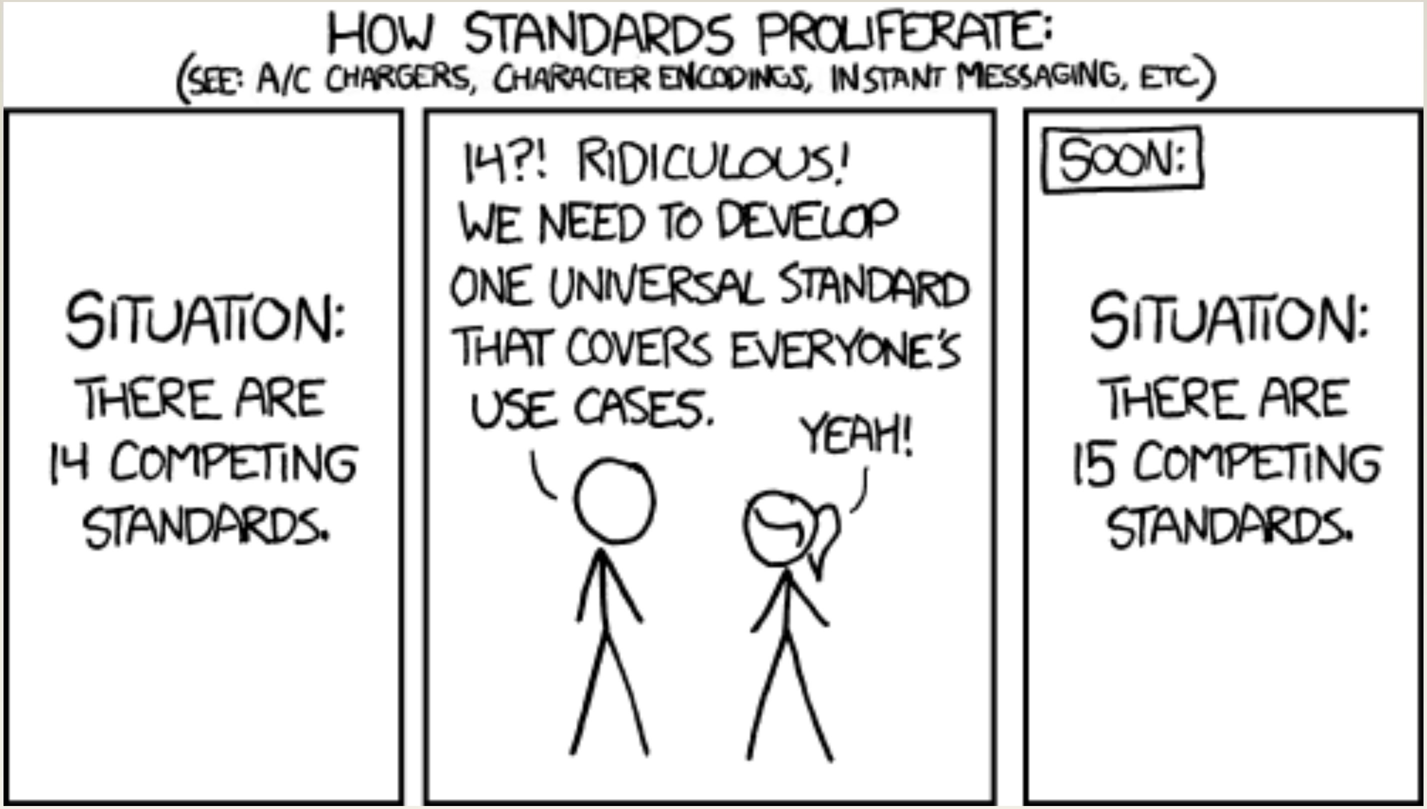
www.rust-lang.org

Character encodings

> “ The nice thing about standards is that you have so many to choose from. ”

- ISO-8859-15
- Windows-1251
- GBK
- EUC-JP
- Shift-JIS
- EUC-KR
- …

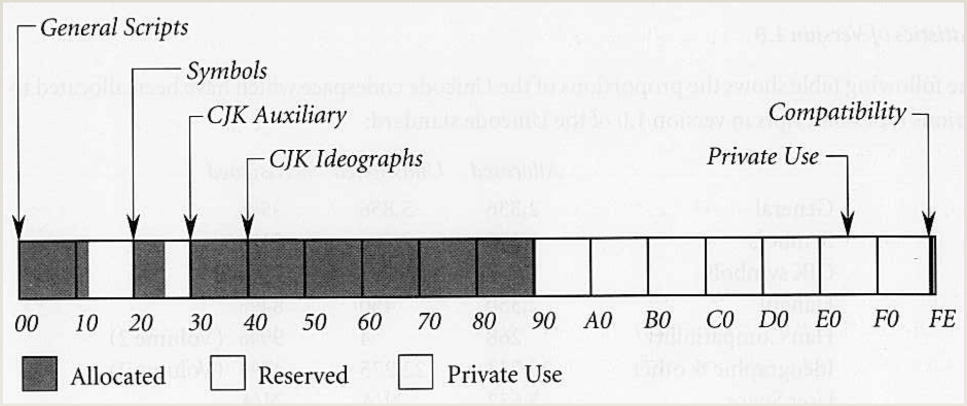
Standards


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

xkcd.com

Unicode 1.0.0 – 1989

16 bits → Up to 65 536 characters*



> “ With over 30,000 unallocated character positions, the Unicode character encoding provides **sufficient space for forseeable future expansion**. ”

Unicode/UCS-2 adoption

- Windows NT
- Java
- JavaScript
- Qt
- (.NET)
- (OS X)

UTF-8 – 1992

```
0.......
110..... 10......
1110.... 10...... 10......
11110... 10...... 10...... 10......
111110.. 10...... 10...... 10...... 10......
1111110. 10...... 10...... 10...... 10...... 10......
```

Up to 31 bits

UTF-16 – 1996 (Unicode 2.0.0)

Walks like UCS-2, swims like UCS-2, quacks like UCS-2

lead surrogate (0xD800 ~ 0xDBFF)
 · trail surrogate (0xDC00 ~ 0xDFFF)
= **surrogate pair**
→ 1 **supplementary** character.

Up to 1 112 064 characters

No supplementary character allocated

Surrogates not in a pair: ¯_(ツ)_/¯

Also in Unicode 2.0.0

Abstract characters:

- U+0000 ~ U+D7FF
- U+E000 ~ U+10FFFF

(exclude surrogates)

Multiple encodings: UTF-8, UTF-16, UTF-32, ...

Artificially restricted

E.g. 0xED 0xA0 0x80 → U+D800 is ill-formed in UTF-8

Rust strings

UTF-8 all the things!

```
pub struct String {
    vec: Vec<u8>,
}
```

API enforces UTF-8 well-formedness

OS strings

- File names
- Environment variables
- Command line parameters

Unix: arbitrary bytes, often UTF-8

Windows: supposedly UTF-16, not always well-formed.

can i haz cross-platform?

std::ffi::OsString

Encapsulate platform differences.

```
#[cfg(unix)]
pub struct OsString {
    data: Vec<u8>,
}

#[cfg(windows)]
pub struct OsString {
    data: // ...  not Vec<u16>!
}
```

WTF-8

UTF-8 superset with surrogates, but only if not in pairs

Same possible values as *potentially ill-formed UTF-16*

(and special concatenation)

Prior art: Scheme 48, Racket

Specification: simonsapin.github.io/wtf-8