Building Cicero with Rust

Cicero: A Unicode® Tool

A simple iOS app for checking Unicode Character Database (UCD)

More About Cicero

Cicero is my personal project

It was released on June, 2018

Some of users are from Apple and The Unicode Consortium

My first time to use Rust in a real project

Rust because libicu is too hard to use

Demo

Let's take a look at the app

Languages & Libraries Used

```
Cicero iOS
├── User Interface
│   └── Swift/ObjC (Apple Cocoa Touch)
└── Unicode Database
    └── libucd
        ├── C Headers
        ├── Unsafe and no_mangle Rust
        ├── Unicode Data
        │   └── Rust (open-i18n/rust-unic)
        └── Full-Text Search
            └── Rust (jgallagher/rusqlite)
```

Building A Static Library for iOS with Rust

ARMv7/ARMv7s/ARM64 (iOS Devices)

i386/x86_64 (iOS Simulators)

Debug/Release build configs

A cargo subcommand that makes these a lot easier to work with:

```
TimNN/cargo-lipo
```

Calling the Rust Library from Swift/ObjC

Rust ABI is not stable

Rust has good C FFI

Swift/ObjC has good interoperability with C

Use a C header and an unsafe no_mangled Rust layer to bridge the core and the UI

unicode_version.rs

```
pub const UNICODE_VERSION: UnicodeVersion = ...;
```

ucd.rs

```
#[no_mangle]
pub extern fn UCDGetUnicodeVersion(out_major: *mut u16,
                                   out_minor: *mut u16,
                                   out_micro: *mut u16) {
    unsafe {
        *out_major = UNICODE_VERSION.major;
        *out_minor = UNICODE_VERSION.minor;
```

ucd.rs

```rust
#[no_mangle]
pub extern fn UCDGetUnicodeVersion(out_major: *mut u16,
                                   out_minor: *mut u16,
                                   out_micro: *mut u16) {
    unsafe {
        *out_major = UNICODE_VERSION.major;
        *out_minor = UNICODE_VERSION.minor;
        *out_micro = UNICODE_VERSION.micro;
    }
}
```

```
nm libucd.a | rg UCDGetUnicodeVersion

0000000000000050 T _UCDGetUnicodeVersion
```

libucd.h

```header
UCD_EXPORT
void
UCDGetUnicodeVersion(uint16_t * outMajor,
                     uint16_t * outMinor,
                     uint16_t * outMicro);
```

```
0000000000000050 T _UCDGetUnicodeVersion
```

libucd.h

```
UCD_EXPORT
void
UCDGetUnicodeVersion(uint16_t * outMajor,
                     uint16_t * outMinor,
                     uint16_t * outMicro);
```

Swift Interface Generated by swiftc

```
public func UCDGetUnicodeVersion(_ outMajor: UnsafeMutablePointer<UInt16>,
                                 _ outMinor: UnsafeMutablePointer<UInt16>,
                                 _ outMicro: UnsafeMutablePointer<UInt16>)
```

UnicodeData.swift

```
var major: UInt16 = 0
var minor: UInt16 = 0
var micro: UInt16 = 0
UCDGetUnicodeVersion(&major, &minor, &micro)
```

Passing Around Objects

Passing around a Rust struct to Swift/ObjC

```
#[derive(Clone)]
pub struct UCDGraphemeCluster {
    character: String,
    scalars: Vec<UCDScalar>,
}
```

It's hard to map the internals of the Rust struct into a C struct

Opaque pointer

```
typedef struct UCDGraphemeCluster * UCDGraphemeClusterRef;
```

Clone the struct and by-pass the Rust memory management

Pass around the opaque pointer instead

```
#[no_mangle]
pub extern fn UCDStringCopyGraphemeClusterAtIndex(
    string: *const UCDString,
```

Pass around the opaque pointer instead

```
#[no_mangle]
pub extern fn UCDStringCopyGraphemeClusterAtIndex(
    string: *const UCDString,
    index: size_t) -> *mut UCDGraphemeCluster {

    let ucd_string = unsafe { &*string };
    let grapheme_cluster = ucd_string.grapheme_clusters[index].clone();
    Box::into_raw(Box::new(grapheme_cluster))
}
```

When finished with the opaque pointer, pass it back to deallocate the memory

```
#[no_mangle]
pub extern fn UCDGraphemeClusterRelease(
    grapheme_cluster: *mut UCDGraphemeCluster) {

    unsafe {
        Box::from_raw(grapheme_cluster);
    }
}
```

```
    Box::into_raw(Box::new(grapheme_cluster))
}
```

When finished with the opaque pointer, pass it back to deallocate the memory

```
#[no_mangle]
pub extern fn UCDGraphemeClusterRelease(
    grapheme_cluster: *mut UCDGraphemeCluster) {

    unsafe {
        Box::from_raw(grapheme_cluster);
    }
}
```

This mimics the CoreFoundation naming convention when used in Swift

- If you use a Get function, you cannot be certain of the returned object's life span
- If you use a Copy or Create function, you are responsible for releasing the object

```
let graphemeCluster = UCDStringCopyGraphemeClusterAtIndex(ucdString, index)
// do something with graphemeCluster
UCDGraphemeClusterRelease(graphemeCluster)
```

Debugging the Rust Library

One day a test user reported:

> Characters are ignored after U+0000
> "A,U+0000,B" → "A"

Debugging the UI code observed the string was correctly passed from the UI

The related Rust API in libucd:

libucd.h

```
UCD_EXPORT UCDStringRef UCDStringCreateWithCString(
    const char * cString);
```

ucd.rs

```
#[no_mangle]
pub extern fn UCDStringCreateWithCString(
    string: *const c_char) -> *mut UCDString {

    let c_str = unsafe { CStr::from_ptr(string) };
    let input = c_str.to_str().unwrap();
```

```rust
pub extern fn UCDStringCreateWithCString(
    string: *const c_char) -> *mut UCDString {

    let c_str = unsafe { CStr::from_ptr(string) };
    let input = c_str.to_str().unwrap();
    // ...
}
```

U+0000 → 0x0 → The Null Terminator '\0'

Use a counted byte buffer instead of a C string fixed it

ucd.rs

```rust
#[no_mangle]
pub extern fn UCDStringCreateWithUTF8Bytes(
    bytes: *mut u8,
    bytes_len: size_t) -> *mut UCDString {

    let input = unsafe {
        String::from_raw_parts(bytes, bytes_len, bytes_len)
    };
    // ...
}
```

Debugging libucd in Xcode

Cicero 〉 Yan's iPhone

Running Cicero on Yan's iPhone

ucd.rs 〉 No Selection

Cicero PID 12316

CPU	0%
Memory	33.8 MB
Energy Impact	Low
Disk	Zero KB/s
Network	Zero KB/s

- Thread 1 Queue: com.apple.main-thread (serial)
  - 0 UCDStringCreateWithUTF8Bytes
  - 1 static UnicodeData.splitStringIntoGraphemeClusters(string:)
  - 2 ScalarViewController.textFieldDidEndEditing(_:)
  - 9 -[UITextField resignFirstResponder]
  - 10 ScalarViewController.textFieldShouldReturn(_:)
  - 36 UIApplicationMain
  - 37 main
  - 38 start
- com.apple.uikit.eventfetch-thread (7)
- Thread 11
- Thread 12
- Thread 13
- Thread 14
- Thread 15

```
103 }
104
105 #[no_mangle]
106 pub extern fn UCDStringCreateWithUTF8Bytes(bytes: *mut u8, bytes_len: size_t) -> *mut UCDString {
107     let input = unsafe { String::from_raw_parts(bytes, bytes_len, bytes_len) };
108
109     let mut results = vec![];
110     {
111         let grapheme_clusters = Graphemes::new(&input);
112         grapheme_clusters.for_each(|cluster| {
113             let mut scalars = vec![];
114             {
115                 cluster.chars().for_each(|chr| {
116                     let scalar = UCDScalar::new(chr);
117                     scalars.push(scalar);
118                 });
119             }
120
121             let result = UCDGraphemeCluster {
122                 character: cluster.to_string(),
123                 scalars: scalars,
124             };
125             results.push(result);
126         });
127     }
128
129     let ucd_string = UCDString {
130         grapheme_clusters: results,
131     };
132     Box::into_raw(Box::new(ucd_string))
133 }
```

Thread 1: st

Cicero 〉 Thread 1 〉 0 UCDStringCreateWithUTF8Bytes

- **bytes** = (unsigned char *) "A"
- **bytes_len** = (unsigned long) 3

```
2018-09-10 20:06:50.192621+1200 Cicero[12316:9139729] Returning
object of class NSString
2018-09-10 20:06:53.877160+1200 Cicero[12316:9139729] [MC] Syst
```

Cicero PID 12316

CPU	0%
Memory	33.8 MB
Energy Impact	Low
Disk	Zero KB/s
Network	Zero KB/s

- Thread 1 Queue: com.apple.main-thread (serial)
  - 0 UCDStringCreateWithUTF8Bytes
  - 1 static UnicodeData.splitStringIntoGraphemeClusters(string:)
  - 2 ScalarViewController.textFieldDidEndEditing(_:)
  - 9 -[UITextField resignFirstResponder]
  - 10 ScalarViewController.textFieldShouldReturn(_:)
  - 36 UIApplicationMain
  - 37 main
  - 38 start
- com.apple.uikit.eventfetch-thread (7)
- Thread 11
- Thread 12
- Thread 13
- Thread 14
- Thread 15
- Memory
  - 0x1c4010c30

Cicero 〉 0x1c4010c30

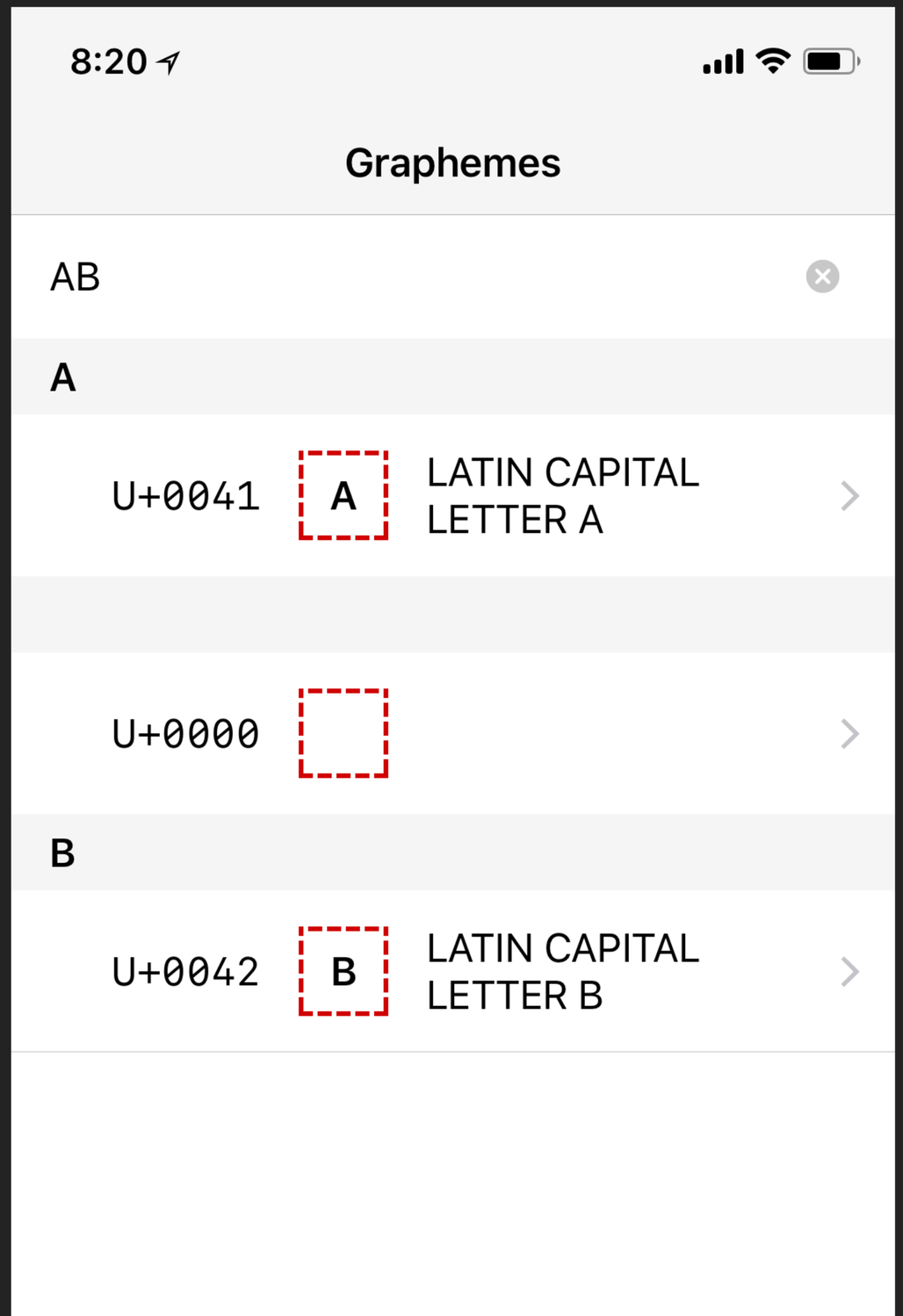
```
1C4010C30 41 00 42 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 A.B.............................
1C4010C4E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010C6C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010C8A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010CA8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010CC6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010CE4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010D02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010D20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010D3E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010D5C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010D7A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010D98 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010DB6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010DD4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010DF2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010E10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................................
1C4010E2E 00 00                                                                                ..
```

Address 0x1c4010c30 Page Lock Number of Bytes 512

Cicero 〉 Thread 1 〉 0 UCDStringCreateWithUTF8Bytes

- bytes = (unsigned char *) "A"
- bytes_len = (unsigned long) 3

```
2018-09-10 20:06:50.192621+1200 Cicero[12316:9139729] Returning local
object of class NSString
2018-09-10 20:06:53.877160+1200 Cicero[12316:9139729] [MC] System group
container for systemgroup.com.apple.configurationprofiles path is /
private/var/containers/Shared/SystemGroup/
systemgroup.com.apple.configurationprofiles
2018-09-10 20:06:53.877886+1200 Cicero[12316:9139729] [MC] Reading from
public effective user settings.
2018-09-10 20:06:55.153488+1200 Cicero[12316:9139729] [Common]
_BSMachError: port 8207; (os/kern) invalid capability (0x14) "Unable to
insert COPY_SEND"
Error! nul byte found in provided data at position: 0
2018-09-10 20:07:17.346781+1200 Cicero[12316:9139729] [Common]
_BSMachError: port 8f0b; (os/kern) invalid capability (0x14) "Unable to
insert COPY_SEND"
(lldb) p bytes
(unsigned char *) $0 = 0x00000001c4010c30 "A"
(lldb)
```


8:20
Graphemes
AB
A
U+0041
A
LATIN CAPITAL LETTER A
U+0000
B
U+0042
B
LATIN CAPITAL LETTER B

Rust Saved Me

```
let transaction = database.connection.transaction();
// continue to use database to do stuff
```

Rust Saved Me

```
error[E0507]: cannot move out of borrowed content
  --> src/search.rs:59:5
   |
59 |     ucd_database.connection;
   |     ^^^^^^^^^^^^ cannot move out of borrowed content
```

Need it -> Build it -> Share it

Lots and lots of awesome crates

- Well designed
- Well implemented
- Well documented

The community is young

You might have to do it by yourself

Don't be shy to contribute

`open-i18n/rust-unic`

- `unic/ucd/name`
- `unic/ucd/hangul`
- `unic/ucd/unihan`
- `unic/ucd/block`
- `Unicode 11.0`

Apps with a Native Look & Feel and a Rust Core

google/xi-editor

ImageOptim/gifski

Codezerker/Cicero

Platform native look & feel usually creates a superior UX

Reuse low-level logics across all platforms

Cargo build system ❤️

Future Plans for Cicero

fdehau/tui-rs + libucd = Cicero for Command Line

Cocoa + libucd = Cicero for macOS

UWP + libucd = Cicero for Windows

Thank you!