Port C++ to Rust

jserv 許佑庭

COSCUP 17 - Rust Community

Kan-Ru Chen

Why Rust?

- Fun 好玩
- Great community 友善社群
- Fearless programming 內建安全帶
- Fast 就是快
- Great development environment 開發環境
- High quality libraries 高品質函式庫

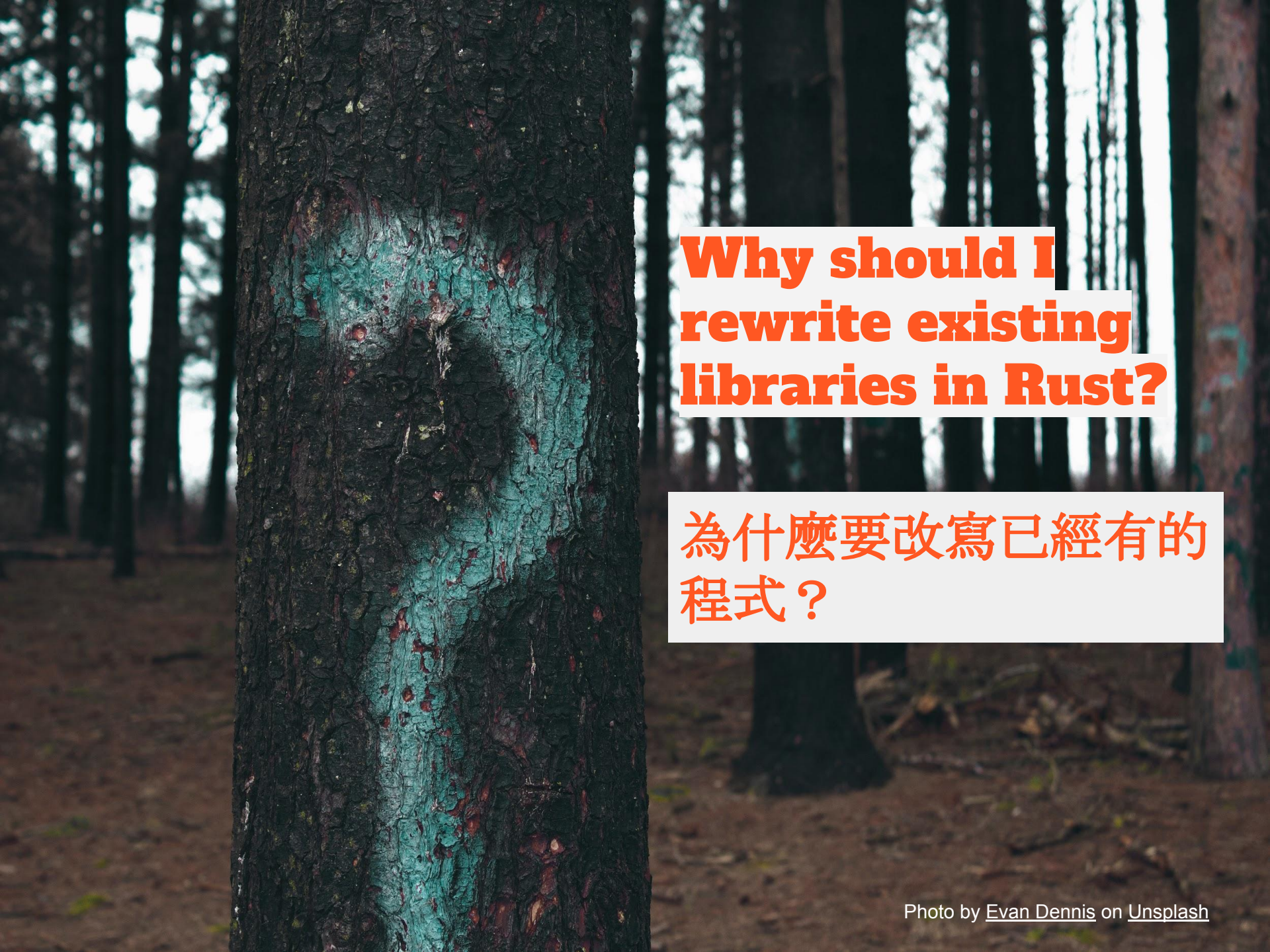

Photo by Evan Dennis on Unsplash

What library can be port to Rust?

- 挑你熟悉的部份下手
  Pick something familiar
- 最好已經有現成完整的測試
  Best with existing test suite
- 挑選 Rust 擅長處理的問題，例如跟記憶體管理相關的
  Dealing with bugs that Rust is especially good at, like memory safety.

Step by step

Install rust with rustup

Go to here: https://rustup.rs/

```
curl https://sh.rustup.rs -sSf | sh
```

Connect existing build system and cargo

- Use cargo to create static library archive.
- Link C code and Rust code together.

Autotools + libtool

- Check if rustc and cargo are installed.
- Automatic dependency handling.
- Create libtool library archive files with libtool-rs.

Translate source from C to Rust

- By hand
- Automatically
  - Rust-bindgen (C ⇨ Rust)
  - Rusty-cheddar (Rust ⇨ C)

Tests

- Cargo test to cover new rust code.
- Existing tests to cover integration issues.
  - Can often find undocumented assumptions.

Let’s do it

Why do I choose libchewing?

- Relatively familiar to me.
  - Being the maintainer for several years.
- Modularized design.
- Track record of memory bugs.
- Great test coverage.

The Chewing (酷音) is an intelligent phonetic (Zhuyin/Bopomofo) input method, one of the most popular choices for Traditional Chinese users. -- https://github.com/chewing/libchewing

First, decide which part to replace first

- Start with a module with smaller surface (self-contained).
- Straightforward to rewrite, no external dependency.
- No complicated algorithms.

User Phrases

- `userphrase-private.h`
- For storing phrases learned automatically.
- Originally in some ondisk hash file format.
- Rewritten to use sqlite3 backend.

Check if rustc and cargo is installed

configure.ac:

```
AC_CHECK_PROG(CARGO, [cargo], [yes], [no])
AS_IF(test x$CARGO = xno,
      AC_MSG_ERROR([cargo is required])
)
AC_CHECK_PROG(RUSTC, [rustc], [yes], [no])
AS_IF(test x$RUSTC = xno,
      AC_MSG_ERROR([rustc is required])
)

CARGO_TARGET_DIR=release
AC_SUBST(CARGO_TARGET_DIR)
```

Link C and Rust code together

src/Makefile.am:

```
libchewing_la_LIBADD = \
        $(top_builddir)/src/common/libcommon.la \
        $(top_builddir)/src/porting_layer/src/libporting_layer.la \
        $(top_builddir)/src/userphrase/target/release/libuserphrase.la \
        $(NULL)


.PHONY: $(top_builddir)/src/userphrase/target/release/libuserphrase.la
$(top_builddir)/src/userphrase/target/release/libuserphrase.la:
    cd userphrase && cargo build --release

clean-local:
    -cd userphrase && cargo clean
```

Generate libtool archive automatically

src/userphrase/Cargo.toml:

```
[package]
name = "userphrase"
version = "0.1.0"
authors = ["Kan-Ru Chen <kanru@kanru.info>"]
build = "build.rs"

[build-dependencies]
libtool = "0.1"

[lib]
crate-type = ["staticlib"]
```

src/userphrase/build.rs:

```
extern crate libtool;

fn main() {
    libtool::generate_convenience_lib("libuserphrase").unwrap();
}
```

Declare Interfaces

Some Original C interface:

```
int UserUpdatePhrase(struct ChewingData *pgdata,
                     const uint16_t phoneSeq[],
                     const char wordSeq[]);

UserPhraseData* UserGetPhraseNext(struct ChewingData *pgdata,
                                  const uint16_t phoneSeq[]);
```

src/userphrase/build.rs:

```
use libc;

#[no_mangle]
pub extern "C" fn UserUpdatePhrase(
    pgdata: *mut ChewingData,
    phoneSeq: *const libc::uint16_t,
    wordSeq: *const libc::c_schar,
) -> libc::c_int {
    unimplemented!();
}
```

Use libc crate to get C types aliases

- `libc::c_char`
- `libc::uint16_t`

Or use std::ffi

- `ffi::CStr`
- `ffi::CString`

Opaque struct pattern

C interface (create, using, free):

```
struct UserPhraseIter;

struct UserPhraseIter* userphrase_iter_new(struct ChewingData *pgdata);
struct UserPhrase* userphrase_iter_next(struct UserPhraseIter *iter);
void userphrase_iter_free(struct UserPhraseIter *iter);
```

Rust:

```
/// Iterator to enumerate all user phrases.
#[no_mangle]
pub extern "C" fn userphrase_iter_new<'a>(
    pgdata: *mut ChewingData
) -> *mut AllUserPhraseIter<'a> {
    unsafe {
    let db = chewing_internal_userphrase(pgdata);
    Box::into_raw(Box::new(AllUserPhraseIter((&*db).iter_all())))
    }
}
```

Opaque struct pattern (cont.)

Rust:

```
#[no_mangle]
pub extern "C" fn userphrase_iter_next(
    iter: *mut AllUserPhraseIter
) -> *const UserPhrase {
    let iter = unsafe {
    assert!(!iter.is_null());
    &mut *iter
    };
    // ...
    ptr::null()
}

#[no_mangle]
pub extern "C" fn userphrase_iter_free(iter: *mut AllUserPhraseIter) {
    if iter.is_null() { return; }
    unsafe {
    Box::from_raw(iter);
    }
}
```

Passing C strings to Rust

Rust:

```
/// Update or add a new user phrase.
#[no_mangle]
pub extern "C" fn UserUpdatePhrase(
    pgdata: *mut ChewingData,
    phoneSeq: *const libc::uint16_t,
    wordSeq: *const libc::c_schar,
) -> libc::c_int {
    let db = unsafe { &mut *chewing_internal_userphrase(pgdata) };
    let phoneSeq = unsafe { phoneseq_from_raw(phoneSeq) };
    let word = unsafe {
        CStr::from_ptr(wordSeq).to_string_lossy().into_owned()
    };
    db.update_or_add(phoneSeq, word) as libc::c_int
}
```

Passing raw pointers to Rust

Rust:

```
use std::ptr;

fn phoneseq_from_raw(ptr: *const libc::uint16_t) -> Vec<libc::uint16_t> {
    unsafe {
    let mut len = 0;
    let mut phoneSeq = Vec::new();
    loop {
        let v = *ptr.offset(len);
        phoneSeq.push(v);
        len += 1;
        if v == 0 {
            break;
        }
    }
    phoneSeq
    }
}
```

Common FFI patterns

- into_raw
- from_raw
- from_ptr
- as_ptr
- mem::forget

Does it work?

```
==================================
Testsuite summary for libchewing 0.5.1
==================================
# TOTAL: 18
# PASS:  16
# SKIP:  0
# XFAIL: 0
# FAIL:  2
# XPASS: 0
# ERROR: 0
```

JOIN US TODAY

Rust Taiwan Community

http://discuss.rust-lang.tw

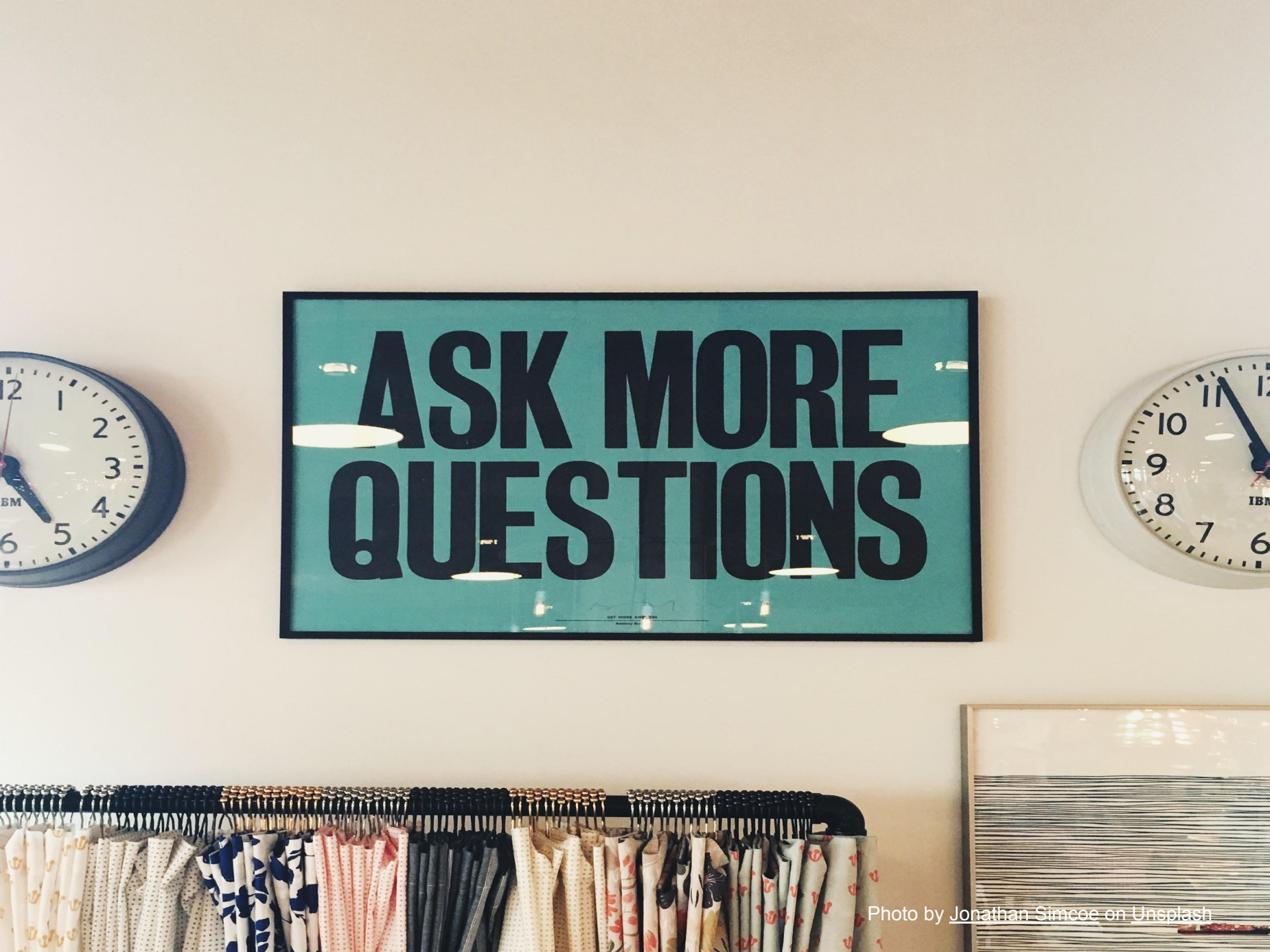

Photo by Jonathan Simcoe on Unsplash