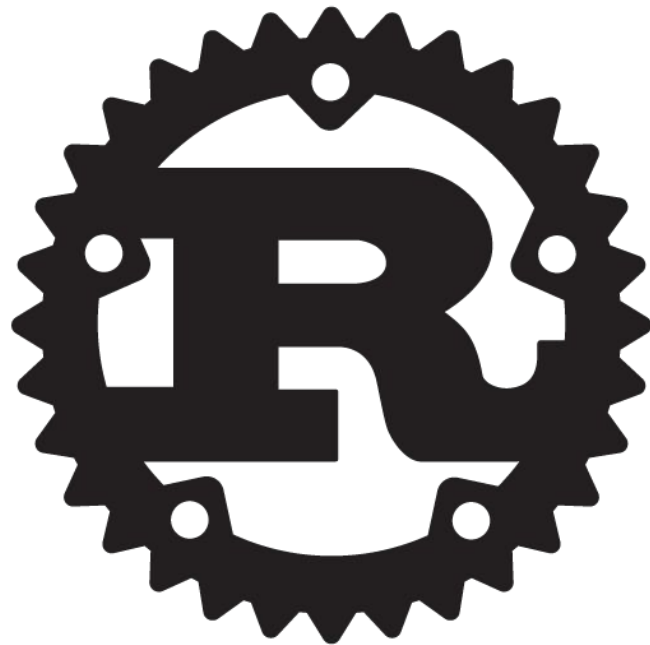

Rust in Production (basics)

By Anshuman Tiwari

anshumantiwari.com

What is Rust?

Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.





Who am I?

I am an entrepreneur who works on real life problems to change the world using data and automation. I do

→ **Freelancing**

Freelancer, Upwork, Fiverr

→ **Open Source Contributions**

Garbo, Elixir and Kalam

→ **Build Products**

Drubbr, Lootore.com, Lead-generation tools, automations tools, marketing automation

Why am I here today?

Rust! And Rust....

I am here to tell you how you can get started with a systems programming language and do interesting stuff because...

“Hello World” is boring



What are we doing today?

By the end of this session, you all will get enough knowledge about::

- **How to execute Rust programs in production**
- **Using Rust with C**
- **The Rust Package Manager (Cargo)**
- **Using the Rust Package Manager**
- **Win32 API**

Let's create a cargo project

```
cargo new --bin windows_dialog
```

**What are we doing
in this project?**

—

**We are doing a simple
implementation of Rust in
production by creating a
dialog in Windows.**

Cargo.toml

Tells us about the author of a project,
dependencies we need to use for the project
etc.

In this project we are using two
dependencies,

`winapi = "0.2.7"`

`user32-sys = "0.2.0"`

MessageBox function

Displays a modal dialog box that contains a system icon, a set of buttons, and a brief application-specific message, such as status or error information. The message box returns an integer value that indicates which button the user clicked.

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms645505\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms645505(v=vs.85).aspx)

—

First we will tell the compiler what we are using

Add these two lines to your rust code

```
extern crate user32;  
extern crate winapi;
```

```
int MessageBox(  
    HWND    hWnd,  
    LPCTSTR lpText,  
    LPCTSTR lpCaption,  
    UINT    uType  
);
```

HWND is a handle (pointer) to a window,
LPCTSTR is a const string, and UINT is, of
course, an unsigned integer.

— MessageBoxA (ASCII)

MessageBoxW (Unicode)

We are using ASCII for now

Looks like this in pseudocode

```
MessageBoxA(  
    NULL,  
    "Hello, world!",  
    "MessageBox Example",  
    MB_OK | MB_ICONINFORMATION  
);
```

Three things we need to know right now

- Where are MessageBoxA, MB_OK, and MB_ICONINFORMATION declared? So we can import them
- What is the equivalent of NULL?
Idiomatically this would be None but we actually need a raw pointer.
- How can we pass Rust string into a C function

—

1. We will visit this link of the documentation of win32 API

```
use user32::MessageBoxA;  
use winapi::winuser::{MB_OK, MB_ICONINFORMATION};
```

<https://retep998.github.io/doc/winapi/index.html>

2. We should use the `std::ptr` module to get a NULL pointer. Since the `winapi` crate defines `MessageBoxA` as taking a `HWND` value which is a mut pointer, we'll use the `std::ptr::null_mut()` function.

3. The standard library's ffi module already has a CString type for precisely this purpose. We simply need to create a new value of this type and use the `.as_ptr()` function to get a raw pointer to pass to the C API.

—

This what our imports look like:

```
use std::ffi::CString;  
use user32::MessageBoxA;  
use winapi::winuser::{MB_OK, MB_ICONINFORMATION};
```

The main function

```
fn main() {  
    let lp_text = CString::new("Hello, world!").unwrap();  
    let lp_caption = CString::new("MessageBox  
Example").unwrap();  
  
    unsafe {  
        MessageBoxA(  
            std::ptr::null_mut(),  
            lp_text.as_ptr(),  
            lp_caption.as_ptr(),  
            MB_OK | MB_ICONINFORMATION  
        );  
    }  
}
```

—
**Just in case you need
something from me in
Python**

—
**Here's a Win32 API
implementation to click at a
given point in Python**

```
import win32api, win32con
def click(x,y):
    win32api.SetCursorPos((x,y))

    win32api.mouse_event(win32con.MOUSEEVENTF_
NTF_LEFTDOWN,x,y,0,0)

    win32api.mouse_event(win32con.MOUSEEVENTF_
NTF_LEFTUP,x,y,0,0)
click(10,10)
```


Thank You

