Running Android Application with Rust

Shuhei Tsuriya[1], Min Hong Yun[2], Lin Zhong[2]
[1]Department of Engineering Science, Osaka University
[2]Department of Electrical and Computer Engineering, Rice University

About Rust in Android apps

Android applications are mainly written in Java, but it often calls C functions through JNI in order to gain high performance.

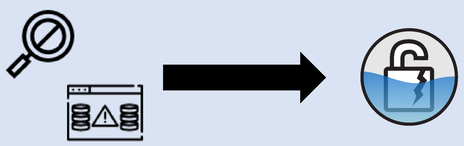
In my research, I make an Android application which uses **Rust** functions from Java because Rust functions make applications **safer.**

Safety of Rust

Calling C programs is unsafe.
This is because

- C compiler does not check data type
- C can access incorrect memory address

These are connected to unsafety.



In order to enhance safety, I replace C functions with Rust functions.

Why is Rust safe?

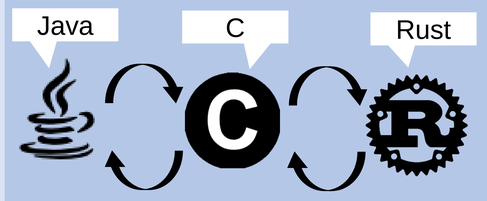
The ownership model confirms safety of Rust

- **Ownership** : limiting access to resources
- **Borrowing** : borrowing ownerships to data and getting temporary accessibilities to certain resources
- **Lifetimes** : giving an expiration of borrowed ownerships

Methods

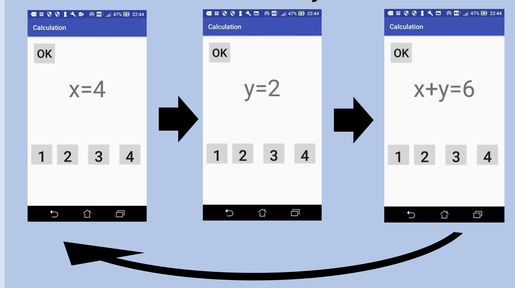
Algorithm of the app

I make a simple calculator application which calls a Rust function from Java through C.

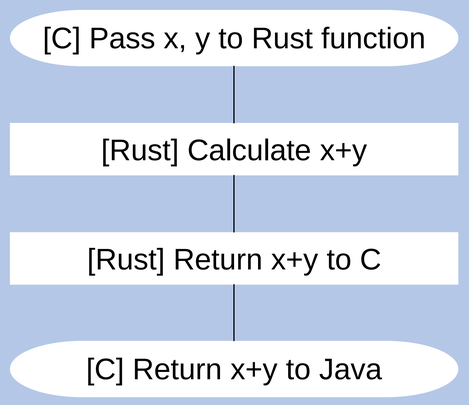


This application follows 3 steps below.

1.Choose x 2.Choose y 3.Show result

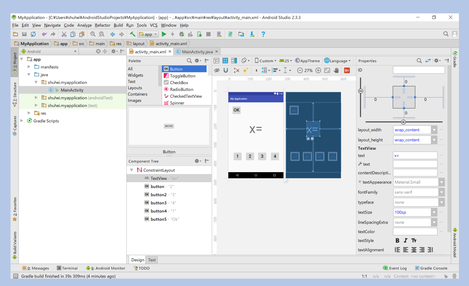


C calls a Rust function in step 3. The flowchart below shows how the Rust function works.



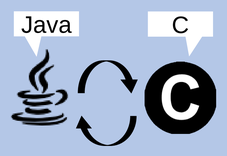
Android Studio

I use **Android Studio** to build the calculator application. Android Studio is an official IDE to build Android applications.



JNI(Java Native Interface)

JNI(Java Native Interface) is a framework to call other language (C or C++) from Java.



Java code

```
public class MainActivity extends AppCompatActivity{
  ..↓
  ..↓
static {↓
    System.loadLibrary( libname: "Hello_JNI");↓
  }↓
  ..↓
  ..↓
  ..↓
  ..↓
textView.setText(GetString(x,y));↓
  ..↓
  ..↓
  ..↓
public native String GetString(int x, int y);↓
}[EOF]
```

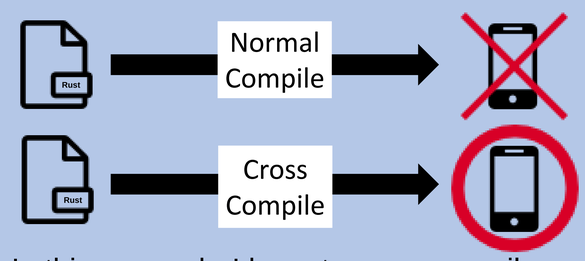
C code

```
  ..↓
  ..↓
JNIEXPORT jstring JNICALL ↓
  Java_com_example_shuhei_hello↓
  _jni_MainActivity_GetString↓
  (JNIEnv *env, jobject instance,↓
  jint x, jint y)↓
{↓
  char *str;↓
  ..↓
  ..↓
  ..↓
  return (*env)->NewStringUTF(env,str);↓
}↓
[EOF]
```

Methods

Cross compile

Cross compile is a way to compile source codes for other devices, not for one on which the compiler is running, and programs compiled by a usual compiler cannot be used in other devices.



In this research, I have to cross compile the Rust codes because Rust functions are used in Android smartphone, not in my laptop.

Future Research

- Measuring the latency caused by calling Rust function
- Figuring out when we should use Rust instead of C
- Figuring out what kind of errors Rust can prevent

Acknowledgements

This research project was conducted as a part of the Nakatani Foundation's 2018 Nakatani RIES Fellowship for Japanese Students. Special thanks to the members of the Zhong Group for their research mentorship and support. I would also like to than Prof. Junichiro Kono, Sarah Phillips, Kenji Ogawa, Aki Shimada, Natsumi Komatsu and other members of Rice University for making this program possible.

References

[1] Nicholas, M. and Aaron, T. *The Rust Programming Language* Retrieved from https://doc.rust-lang.org/book/second-edition/foreword.html
[2] (2018, Apr 25) *Add C and C++ code* Retrieved from https://developer.android.com/studio/projects/add-native-code?hl=ja