CPH.RS

+

Hello Android, using Rust

Tutorials

https://users.rust-lang.org/t/rust-on-android-today/17884

https://medium.com/@authmane512/how-to-build-an-apk-from-command-line-without-ide-7260e1e22676

Bionic Beaver 18.04 LTS

Android

```
sudo apt-get install \
python \
build-essential \
android-sdk \
google-android-ndk-installer \
google-android-build-tools-24-installer \
google-android-platform-24-installer \
apksigner
```

Rustup

wget https://sh.rustup.rs/ --output-document - | sh

Rust

```
for TARGET in aarch64-linux-android armv7-linux-androideabi i686-linux-android
do
  rustup target add ${TARGET}
done
```

NDK

```
mkdir NDK
NDK_HOME="/usr/lib/android-ndk/"
API_LEVEL=24
for ARCH in arm64 arm x86
do
  ${NDK_HOME}/build/tools/make_standalone_toolchain.py \
      --api ${API_LEVEL} --arch ${ARCH} --install-dir NDK/${ARCH}
done
```

df -h /

/dev/nvme0n1p1 7.7G 7.6G 152M 99% /

Cargo Config

(~/.cargo/config)

```
[target.armv7-linux-androideabi]
ar = "`pwd`/NDK/arm/bin/arm-linux-androideabi-ar"
linker = "`pwd`/NDK/arm/bin/arm-linux-androideabi-clang"

[target.aarch64-linux-android ...]

[target.i686-linux-android ...]
```

Boilerplate

AndroidManifest.xml

Boilerplate

AndroidManifest.xml
res/layout/activity_main.xml

Boilerplate

AndroidManifest.xml
res/layout/activity_main.xml
res/values/strings.xml

Boilerplate

AndroidManifest.xml
res/layout/activity_main.xml
res/values/strings.xml
res/drawable

Boilerplate

AndroidManifest.xml
res/layout/activity_main.xml
res/values/strings.xml
res/drawable
src/rs/cph/hellorust

Boilerplate

AndroidManifest.xml
res/layout/activity_main.xml
res/values/strings.xml
res/drawable
src/rs/cph/hellorust
jniLibs/arm64
jniLibs/armeabi
jniLibs/x86

Boilerplate

AndroidManifest.xml
res/layout/activity_main.xml
res/values/strings.xml
res/drawable
src/rs/cph/hellorust
jniLibs/arm64
jniLibs/armeabi
jniLibs/x86
libs

Boilerplate

AndroidManifest.xml
res/layout/activity_main.xml
res/values/strings.xml
res/drawable
src/rs/cph/hellorust
jniLibs/arm64
jniLibs/armeabi
jniLibs/x86
libs

build_java.sh
build_rust.sh
build_apk.sh

Java

src/rs/cph/hellorust/MainActivity.java
src/rs/cph/hellorust/FFI.java

MainActivity.java (src/rs/cph/hellorust/)

```
package rs.cph.hellorust;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends Activity {

    static {
        System.loadLibrary("rustcode");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        FFI rustcode = new FFI();
        String s = rustcode.runFunc("Sundkaj 7");
        ((TextView)findViewById(R.id.greetingField)).setText(s);
    }
}
```

FFI.java (src/rs/cph/hellorust/)

```
package rs.cph.hellorust;

public class FFI {

    private static native String func(final String pattern);

    public String runFunc(String to) {
        return func(to);
    }
}
```

Cargo.toml (rustcode/)

```
[...]

[target.'cfg(target_os="android")'.dependencies]
jni = { version = "0.5", default-features = false }

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

lib.rs, top (rustcode/lib/)

```
use std::os::raw::{c_char};
use std::ffi::{CString, CStr};

#[no_mangle]
pub extern fn rust_func(to: *const c_char) -> *mut c_char {
    let c_str = unsafe { CStr::from_ptr(to) };
    let recipient = match c_str.to_str() {
        Err(_) => "there",
        Ok(string) => string,
    };

    CString::new("Hello ".to_owned() + recipient).unwrap().into_raw()
}
```

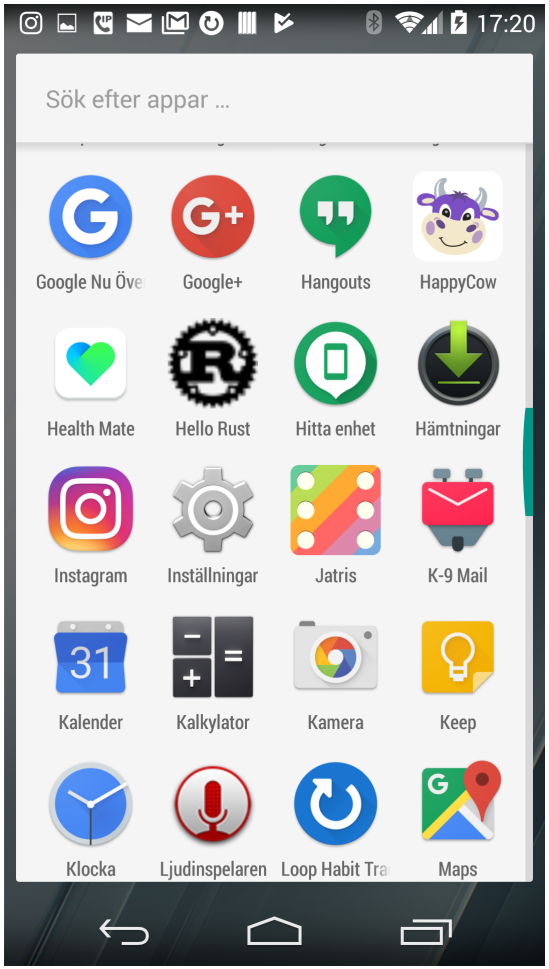
lib.rs, end (rustcode/lib/)

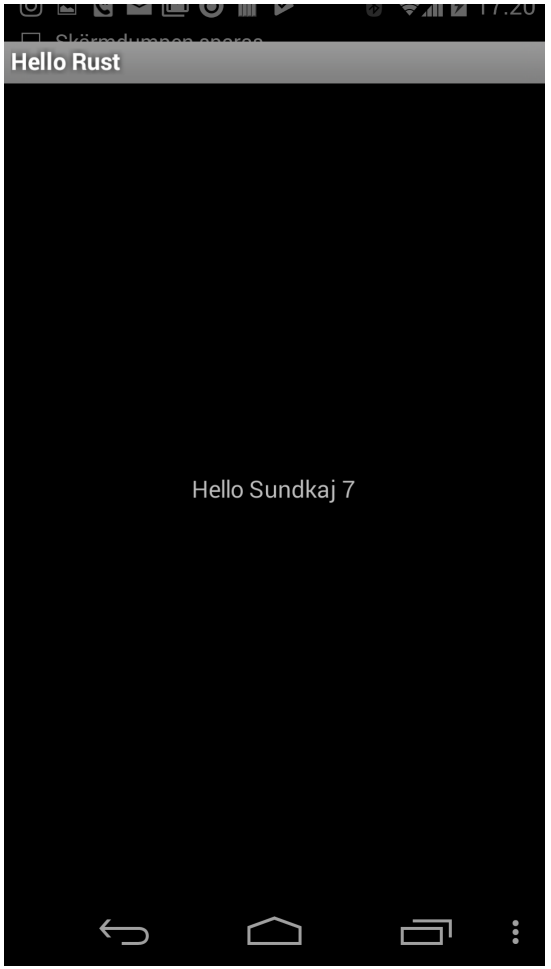
```
#[cfg(target_os="android")]
#[allow(non_snake_case)]
pub mod android {
    extern crate jni;

    use super::*;
    use self::jni::JNIEnv;
    use self::jni::objects::{JClass, JString};
    use self::jni::sys::{jstring};

    #[no_mangle]
    pub unsafe extern fn Java_rs_cph_hellorust_FFI_func(env: JNIEnv, _: JClass, java_pattern: JString)
        -> jstring {
        let world = rust_func(env.get_string(java_pattern).expect("invalid pattern string").as_ptr());
        // Retake pointer, to use it below and allow memory to be freed when it goes out of scope.
        let world_ptr = CString::from_raw(world);
        let output = env.new_string(world_ptr.to_str().unwrap()).expect("Couldn't create java string!");

        output.into_inner()
    }
}
```





Hello Android, using Rust

Martin Samuelsson

 cph.rs@nibufri.netizen.se

#irc |cos| @ #rust-cph

 https://mastodon.social/@brillious