IUT Informatique Bordeaux I
Licence SIL

Pierre RAMET
2012–2013

Android C++

Ce document est basé sur l'article "Android C++" du numéro 154 de Linux Magazine France par Y. Bailly.

1 Configuration

▶ Kits de développement

Pour pouvoir développer une application en C/C++ pour la plateforme Android, il est nécessaire d'installer le NDK (kit natif) ainsi que le SDK (kit standard). Au département, ces environnements de développement ont été installés dans `/info-nfs/opt/Android-sdk/`.

Il vous faut cependant positionner les variables d'environnement suivantes, par exemple :

```
export ANDROID_SDK=/Users/ramet/Work/android/android-sdk-macosx
export PATH=$ANDROID_SDK/tools:$ANDROID_SDK/platform-tools:$PATH
export ANDROID_NDK=/Users/ramet/Work/android/android-ndk-r8c
export PATH=$ANDROID_NDK:$PATH
```

▶ Périphérique virtuel

Pour tester le code que nous allons écrire, il est nécessaire de disposer d'un périphérique virtuel ou physique. Pour configurer un périphérique virtuel, utilisez la commande :

```
android avd
```

puis démarrez le périphérique. Les options par défaut sont suffisantes, notez cependant le niveau d'API disponible (`Android 4.2 - API Level 17`, par exemple). Pour vérifier que le périphérique est bien reconnu, exécutez la commande :

```
adb devices
```

2 Création d'un projet

Il faut commencer par créer une arborescence initiale :

```
mkdir mandel && cd mandel
mkdir -p jni res/values
```

A la racine du projet, il faut créer un fichier `"Android-Manifest.xml"` :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.mandel"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="9" />
    <application android:label="@string/app_name"
```

```
                android:hasCode="false" android:debuggable="true">
        <activity android:name="android.app.NativeActivity"
                  android:label="@string/app_name"
                  android:configChanges="keyboard|keyboardHidden|orientation">
            <meta-data android:name="android.app.lib_name"
                       android:value="mandel" />
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

ainsi qu'un fichier "`default.properies`" contenant le niveau API de la cible :

```
target=android-17
```

Dans le répertoire `res/values`, il faut nommer l'application avec un fichier "`strings.xml`" contenant :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Mandel</string>
</resources>
```

Enfin, créez deux fichiers dans le répertoire `jni`, le premier "`Application.mk`" :

```
APP_ABI := armeabi
APP_PLATFORM := android-10
```

et le second "`Android.mk`" :

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE    := mandel
LOCAL_SRC_FILES := mandel.cpp
LOCAL_LDLIBS    := -lm -llog -landroid
LOCAL_STATIC_LIBRARIES := android_native_app_glue
include $(BUILD_SHARED_LIBRARY)
```

3 Le code

Les sources du projet sont disponibles dans le répertoire `/info-nfs/users/pramet/mandel_android.tar.gz`.

La partie concernant l'affichage de l'ensemble de MandelBrot se trouve dans la fonction `draw_frame()`.

La gestion des évènements est traitée dans la fonction `engine_handle_cmd()`.

La mise en place de l'application débute avec la fonction `android_main()`.

4 Compiler et déployer

Pour compiler le projet :

```
android update project -p . -s
ndk-build -B V=1
ant debug
```

Il ne reste plus qu'à envoyer le paquetage sur le périphérique virtuel :

```
adb -e install bin/NativeActivity-debug.apk
```

ou physique :

```
adb -d install bin/NativeActivity-debug.apk
```

Pour visualiser la sortie "journal" de l'application (cf la macro `LOGW`) :

```
adb -e logcat
```

Pour désinstaller l'application :

```
adb -e uninstall com.example.mandel
```