TP5 - NDK

Paul TREHIOU <paul.trahiou@utbm.fr>

Victor SENE <victor.sene@utbm.fr>

LO52

December 22, 2017

List of Figures

List of Tables

Contents

1 Création de l'interface

Nous n'avons pas rencontré de problème particulier lors de la définition de l'interface. Il a fallut se rappeler comment faire un en sorte que le bouton soit cliquable. Nous avons utilisé le code suivant pour cela :

```
final Button button = (Button) findViewById(R.id.read);
button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                Toast.makeText(v.getContext(), "Coucou", Toast.LENGTH_SHORT).show();
        }
});
```

2 JNI

Dans ce TP, le but était de manipuler des fonctions JNI et d'être confronté à des erreurs parfois aberrantes. La première difficulté rencontré a été de comprendre comment $AndroidStudio$ gère les fichier cpp. Ainsi nos fonctions ont été codés dans `app/src/main/cpp/native-lib.cpp`. On remarque qu'il aurait été possible de créer un autre fichier *.cpp*. Ensuite dans la $MainActivity$ il faut penser à mettre les prototypes de nos fonctions :

```
public native String stringFromJNI();
public native String read (String s);
public native String stop (int s);
public native String reset ();
```

$AndroidStudio$ détecte quand les fonctions sont implémentés dans le `native-lib.cpp`. Cependant après plusieurs essais nous avons pu voir que les fonctions n'étaient pas détecté comme implémentées par l'application Android provoquant ainsi un crash de celle-ci lors d'un clic. Il s'est avéré qu'il nous suffisait d'ajouter `extern "C"` avant la déclaration de notre fonction. Le pourquoi ceci est obligatoire est en cours d'exploration.

Afin de pouvoir utiliser les $string$ passé en paramètre de fonction, il a fallu utiliser des fonction de conversion de type `env->GetStringUTFChars` pour pouvoir convertir un `jstring` en `string`. Pour la conversion de chiffre en `string`, nous avons utilisé des `ostringstream` afin de pouvoir afficher le chiffre tel quel.

Finalement les difficultés rencontrées sont liées au typage des variables et fonction. Une fois ses données au bon format, la façon de coder est relativement transparente.

3 Conclusion

Le NDK est une fonctionnalité intéressante que d'exécuter du code natif au sein d'un projet JAVA mais il nous semble vraiment complexe le passage d'un langage à un autre avec le changement de type et les difficultés rencontrés dû au spécificités de chaque langage. Peut être que nous ne voyons que des aspect négatif car nous n'utilisons pas réellement des applications qui sont plus rapide en natif. La maîtrise du $C++$ est indispensable et doit nous manquer pour une meilleur efficacité. C'est donc une idée pas encore abouti mais prometteuse très certainement.

4 Sources

`https://github.com/gxfab/LO52_A2017/tree/star_lord`