Rapport de projet LO52

Groupe “Bonsoir d’Abord”

Nicolas BARBOTIN, Quentin COUDERT, Amaury EINHOLTZ, Julien MEDICI

TP1

La première étape consiste à cloner le repository git. Celui ci étant déjà créée, un simple git clone à suffit. En ce qui concerne notre branche de travail, elle aussi était déjà créée ; ainsi, « git checkout BonsoirDAbord » a suffit pour la sélectionner.

La seconde étape consistait à installer Android Studio et à créer un projet. Ces étapes sont plutôt bien guidées par les assistants d’installation/de création de projet. Les seuls problèmes auxquels j’ai eu à faire face concernaient l’émulateur (que je n’ai d’ailleurs toujours pas réussi à faire fonctionner : il crash dès le lancement de l’application...) ainsi que la détection de mon téléphone. Cependant, après avoir installé les drivers USB Google, redémarré ADB, et après avoir activé et désactivé plusieurs fois le mode débogage USB, la connexion s’est établie et j’ai pu tester mon travail.

Une fois le projet crée, je disposais déjà d’une activité MainActivity avec un TextView contenant « HelloWorld ». J’ai supprimé cet élément et y ai mis à la place un bouton. Dans le fichier de ressource string.xml, j’ai ajouté ma propre entrée « Hello World ». J’ai ensuite assigné ce string à la propriété « text » du bouton de MainActivity. Finalement, j’ai contraint ce bouton de manière à ce qu’il soit au milieu de l’écran.

Ensuite, j’ai créé une autre activité dans laquelle j’ai ajouté un TextView avec le même texte et les mêmes contraintes (pour qu’il soit au milieu). Mes deux activités étant maintenant crées, j’ai ouvert le code de MainActivity et y ai ajouté une méthode « onClick » ne retournant rien et prenant en paramètre un objet de type « View ». Cette méthode est alors apparue dans le layout de mon MainActivity en tant que propriété « onclick » du bouton. J’ai alors sélectionné cette option de manière à ce que cette méthode soit exécutée dès que le bouton est appuyé. Le code de cette méthode est simple et se résume à appeler « startActivity » de la manière suivante :

```
public void onClick(View v) {
    startActivity(new Intent(this, JaajActivity.class));
}
```

Après avoir lancé l’application sur mon téléphone, j’ai pu constater que l’activité avec le texte était bien lancée lors d’un appui sur le bouton.

TP3

Réponse aux questions:

Déterminez les fichiers sources et les headers pour la compilation d'une libusb sur Android

- Fichier sources:
- core.c
- descriptor.c
- io.c
- darwin_usb.c
- linux_usbfs.c
- sync.c
- Headers:
- libusb.h
- libusbi.h
- darwin_usb.h
- linux_usbfs.h

Une erreur se produit sur la macro TIMESPEC_TO_TIMEVAL. Celle-ci n'est pasdéfinie. Corrigez.
Pour corriger cette erreur, nous définissions la macro suivante:

```
#define TIMESPEC_TO_TIMEVAL(tv, ts)                                     \
        do {                                                            \
                (tv)->tv_sec = (ts)->tv_sec;                            \
                (tv)->tv_usec = (ts)->tv_nsec / 1000;                   \
        } while (0)
```

Deuxième erreur
Pour la corriger, il faut mapper libusb.so à une addresse dans 'build/core/prelink-linux-arm.map'

TP4

Pour ce TP, j'ai créé un nouveau projet "C++" à partir d'Android Studio. Celui-ci à configurer pour moi le build afin d'inclure dans l'application une bibliothèque JNI écrite en C++.
Pour interfacer avec Java, j'ai défini trois fonctions, comme demandé par le sujet :

```
public native void callNativeRead(int nbr);
public native void callNativeWrite(int nbr);
public native void callNativeButton(String str);
```

Comme vous pouvez le constater, toutes les valeurs de retour son "void". En fait, le résultat de chacune de ces fonctions est directement inséré dans le label "TextView" situé dans l'activité. Pour cela, nous déclarons un TextView privé nommé "label" dont nous assignons la référence obtenue avec "findViewById". Ensuite, la bibliothèque C++ utilise JNI pour remplir ce label avec la fonction utilitaire suivante :

```
static void setLabelText(JNIEnv *env, jobject dis, const char *str)
{
    jclass clazz = env->GetObjectClass(dis);
    jfieldID labelField = env->GetFieldID(clazz, "label",
"Landroid/widget/TextView;");
    jobject label = env->GetObjectField(dis, labelField);

    clazz = env->GetObjectClass(label);
    jmethodID setTextMethod = env->GetMethodID(clazz, "setText",
"(Ljava/lang/CharSequence;)V");

    jstring jstr = env->NewStringUTF(str);
    env->CallVoidMethod(label, setTextMethod, jstr);
}
```

Le fonctionnement est le suivant:

- On récupère la référence vers le TextView (via notre champ label)
- On construit un string java à partir du string C
- On appelle setText avec ce string

Le reste est relativement simple et se trouve dans le code.