Compte rendu
des TPs de LO52

Sommaire

Killian Pujol, Léo Schwartz, Roméo Dontio Ngoula, Valentin Sourice

TP1: mise en place de l'environnement de développement

Paramétrage d'Android Studio

Modèle de tablettes :	Galaxy tab 10" 2018
API :	Nougat jusqu'à 10
SDK :	7

Le paramétrage d'Android Studio et l'ajout du SDK sur les machines de l'UTBM n'était pas aisé par rapport aux permissions allouées aux étudiants ; nous avons préféré faire la configuration sur nos propres machines.

TP2 : Manipulation d'un Kernel Linux

L'utilisation de la commande repo sur Windows était soumise à l'utilisation de logiciel tiers : nous avons préféré utiliser Ubuntu sur nos machines personnelles.

repo init -u https://android.googlesource.com/kernel/manifest -b android-hikey-linaro-4.4

repo sync
L'exécution de cette commande était (très) longue dû au téléchargement des sources : 6,9Go !

build/build.sh
Après le build, le dossier fait 10,9 Go soit 4 Go supplémentaire.

Le binaire du kernel, les modules et l'image associée localisée dans out/mirror-aosp-android-hikey-linaro-4.4/dist font bien un total de 4 Go.

Suite aux difficultés rencontrées le TP2 et son compte rendu ont été annulé

TP3 : Création d'un device Android

Objectif du TP

- Intégrer un composant externe sous Android
- Définir le fichier « Makefile .mk » correspondant
- Ecrire la définition d'un nouveau produit
- Utiliser les notions d'héritage

Intégration de la libusb

Le fichier Android.mk contenu dans le dossier libusb-1.0.3 contient les lignes suivantes :

```
LOCAL_PATH := $(call my-dir)
ubdirs := $(addprefix $(LOCAL_PATH)/,$(addsuffix /Android.mk, \
     libusb \
))
include $(subdirs)
```

La première ligne « *LOCAL_PATH := $(call my-dir)* » est indispensable au début d'un fichier Android.mk et correspond à la définition de la variable LOCAL_PATH. Cela sert à localiser les fichiers sources. Ici, « my-dir » correspond au chemin du répertoire actuel (contenant le fichier Android.mk lui-même).

La deuxième ligne correspond aux fichiers à inclure. La fonction « *addprefix* » permet d'ajouter un préfixe dans la variable des fichiers à ajouter (ici tous les fichiers doivent être situés dans le dossier défini dans la variable LOCAL_PATH). La fonction « *addsuffix* » permet d'insérer des suffixes dans les fichiers à ajouter.

Au final dans notre cas, cela ajoute tous les fichiers ayant pour chemin : « [LOCAL_PATH]/Android.mk » et « [LOCAL_PATH]/libusb/Android.mk », c'est-à-dire, le chemin vers les autres fichiers Android.mk à traiter.

On règle ensuite le fichier Android.mk situé dans le dossier « libusb ».

```
 LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_SRC_FILES := core.c descriptor.c io.c sync.c os/darwin_usb.c
os/linux_usbfs.c
LOCAL_C_INCLUDES += $(LOCAL_PATH) $(LOCAL_PATH)/os
LOCAL_MODULES := libusb
LOCAL_MODULE_TAGS := optional
include $(BUILD_SHARED_LIBRARY)
```

Nous n’avons malheureusement pas été plus loin dû au faite que nous avons des problèmes de compilation (la commande make que nous n’avons pas réussi à faire reconnaître par l’IDE)

Définition d’un nouveau produit

Nous aurions dû ensuite créer notre propre produit. Comme on souhaite que le vendeur soit « utbm » et que le nom du produit soit « lo52 », on crée l’arborescence : « */device/utbm/lo52* ».

On crée ensuite les fichiers de base pour la définition de notre produit. On crée donc les fichiers suivants :

- Android.mk
- AndroidProducts.mk
- BoardConfig.mk
- lo52.mk (après les modifications liées aux personnalisations suivantes)
- vendorsetup.sh

On aurait ensuite voulu personnaliser notre produit. Pour cela on réalise les opérations suivantes :

- On surcharge le fichier « sym_keyboard_delete.png »
- On personnalise les propriétés (dans le fichier lo52.mk) :
    - ro.hw avec la valeur lo52
    - net.dns1 avec la valeur 8.8.8.8
    - net.dns2 avec la valeur 4.4.4.4
- On inclut le composant libusb dans le fichier Android.mk

Tout est maintenant personnalisé et paramétré comme il le faut.

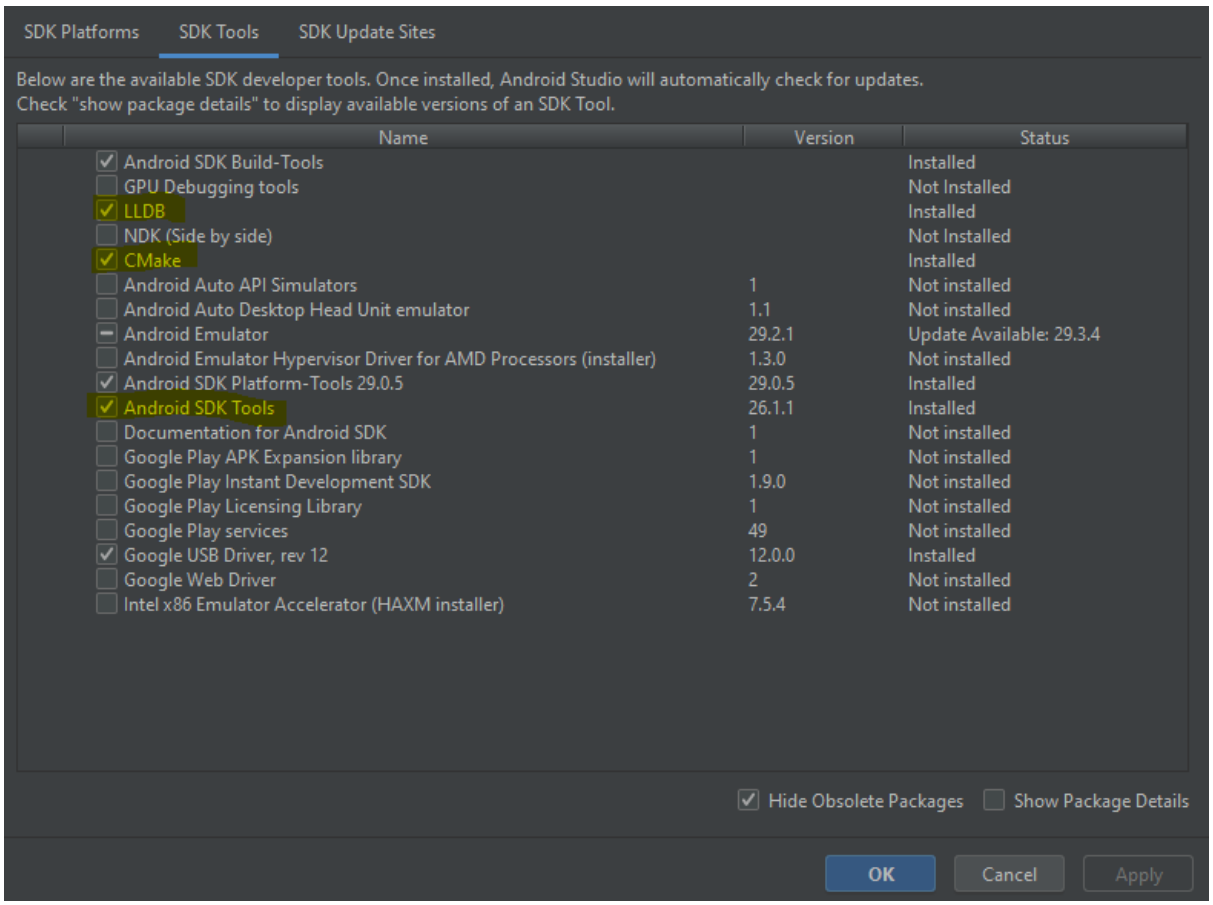
TP4 : Utilisation de JNI

Objectif du TP

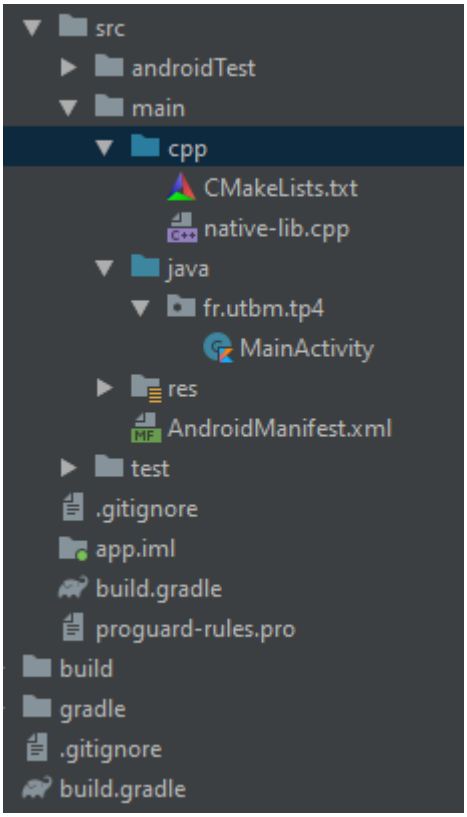
- Apprendre le codage en langage natif sous Android NDK

Installation de l'environnement

Il est nécessaire d'installer les composants requis pour JNI sur Android Studio. Pour se faire, il faut se rendre dans le SDK Manager (Tools>SDK Manager), Onglet "SDK Tools", puis installer les 3 composants suivants : *Cmake, NDK, LLDB*, comme sur la copie d'écran suivante :



Ensuite, il faudra créer un nouveau projet Android Studio de type *Native C++*.



Voici l'architecture du projet ainsi créé, possédant un "Hello World" généré automatiquement :

L'interface graphique

Voici l'interface graphique réalisé pour le TP4 :

On dispose d'un champ de saisie de texte (EditText) contenant par défaut la valeur "Def".
Les boutons READ, WRITE, UP, DOWN, LEFT et RIGHT.
Et enfin un TextView afin d'afficher le texte.