RAPPORT DE TRAVAUX PRATIQUES n°4

LO52 – A2017

Étudiants : Pierre ROMET – Guillaume PROST

Sommaire

Table des matières

Le présent rapport a pour but de restituer le travail qui a été réalisé lors du TP n°4 de LO52 durant le semestre d'automne 2017 à L'UTBM, et qui portait sur la définition de fichiers makefile pour différents composants et produits destiné à un système Android.

Intégration de la *libusb*

La première partie de ce TP portait sur une librairie C nommée libusb. Les sources pour cette partie se trouvent toutes dans le répertoire *libusb-1.0.3/* et ont été fournie par l'enseignant via *Git* : nous avons dû réaliser un merge de deux commit de la branche *SnakeTeacher.*

Fichier *Android.mk* du répertoire racine

Le premier travail sur ces sources a été d'analyser le contenu du fichier *libusb-1.0.3/Android.mk*, contenu présent sur la ***Fig 1***.

```
1  LOCAL_PATH := $(call my-dir)
2  subdirs := $(addprefix $(LOCAL_PATH)/,$(addsuffix /Android.mk, \
3          libusb \
4  ))
5  include $(subdirs)
```

Illustration 1: Contenu du fichier libusb-1.0.3/Android.mk

La première ligne de ce fichier est une directive que l'on retrouve dans la une grande majorité des makefile que nous avons vu ce semestre en LO52, et qui attribue à la variable locale *LOCAL_PATH* une valeur égale au dossier courant du makefile.

Les lignes 2 à 4 correspondent également à une attribution de valeur à une variable locale. Dans ce cas-ci, la variable est *subdirs* et la valeur vaut *$(LOCAL_PATH)/libusb/Android.mk*. En effet, on observe deux appels de macro (addprefix et addsuffix), qui vont effectuer les actions suivantes :

1. Ajout du suffixe « */Android.mk* » à « *libsub* »
2. Ajout du préfixe « *$(LOCAL_PATH)* » au résultat de 1.
3. Attribution du résultat de 2. à la variable locale « *subdirs* »

La variable *subdirs* pointe donc sur le fichier *libusb-1.0.3/libusb/Android.mk*.

Enfin, la dernière ligne de ce fichier makefile est une directive permettant d'inclure le fichier makefile sur lequel pointe la variable *subdirs*.

En somme, ce fichier *Android.mk* permet de faire appel à un autre fichier makefile présent dans un sous-répertoire, ici *$(LOCAL_PATH)/libusb/Android.mk*.

Fichier *Android.mk* de la *libusb*

Deuxième étape de ce TP, nous devions écrire le contenu du fichier makefile inclus précédemment. Nous avons utilisé la même structure de makefile que celle vu en TD tout le long du semestre :

1. Le fichier commence par les directives « LOCAL_PATH:= $(call my-dir) » et « include $ (CLEAR_VARS) », qui permettent de reconfigurer les variables locales dont nous allons nous servir dans le makefile.
2. Ensuite, la directive « LOCAL_SRC_FILES » qui nous permet de spécifier tous les fichiers *.c que nous voulons ajouter à la librairie *libusb*. Nous avons ajouté tous les fichiers *.c dans le dossier *libusb-1.0.3/libusb* ainsi que le fichier *linus_usbfs.c* présent dans *libusb-1.0.3/libusb/os*.
   Ce dossier *os* contient également un fichier *darwin_usb.c* que nous n'incluons pas, car il est destiné aux librairies pour des plate-formes Apple, ce qui n'est pas le cas de notre librairie, celle-ci étant destinée à des plate-formes Android.
3. Après cela, nous ajoutons les directives LOCAL_MODULE et LOCAL_MODULE_TAGS, qui nous permettent de définir le nom de notre librairie ainsi que son tag, qui permet de définir si la librairie est obligatoire pour un certain scope de produit (*user, userdebug* ou *eng*), ou si elle est optionnelle. C'est ce dernier cas qui nous intéresse, pour des raisons de simplicité.
4. Nous utilisons ensuite la directive LOCAL_C_INCLUDES qui nous permet de spécifier les dossiers contenant les fichiers *.h que nous souhaitons utiliser. Cette directive ne cherchant pas les fichiers *.h présent dans les sous-répertoires, nous devons donc spécifier ces derniers.
   Dans notre cas, nous inscrivons donc dans les valeurs le chemin vers le dossier courant ainsi que le dossier *os*. L'utilisation de la variable locale LOCAL_PATH au lieu de *libusb-1.0.3/libusb* permet de rendre le makefile un peu plus modulable et plus lisible.

5. Enfin, nous ajoutons « include $(BUILD_SHARED_LIBRARY) » en tant que dernière ligne pour indiquer que ce makefile doit générer une librairie partagée.

Erreurs de compilation

Une fois ce fichier makefile écrit, nous passons à la compilation. Celle-ci étant relativement longue, le TP ne nous demandait pas de la réaliser, mais nous donnais les résultats de celle-ci. Il nous donnait notamment deux erreurs qui seraient survenues durant la compilation.

Erreur de macro « TIMESPEC_TO_TIMEVAL »

La première erreur concerne une macro TIMESPEC_TO_TIMEVAL qui n'est pas définie, ou en tout cas pas trouvée par le compilateur. La solution pour résoudre cette erreur est d'ajouter sa définition dans nos sources.

Pour cela, on peut par exemple rajouter les lignes figurant sur la **Fig 2** dans le fichier *libusb-1.0.3/libusb/io.c file*.

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

Illustration 2: Code C définissant la macro TIMESPEC_TO_TIMEVAL

Erreur « not in prelink map »

La deuxième erreur apparaît car la librairie *libsub.so*, notre librairie, n'est pas présente dans la « prelink map ». L'une des solutions serait donc de l'ajouter dans celle-ci. Notre librairie étant une librairie USB, nous pensons que cette *prelink map* se trouve dans des dossiers liés au kernel.

Une rapide recherche sur internet concernant cette erreur et nous trouvons le fichier à modifier : *./build/core/prelink-linux-arm.map*.

Ce fichier n'étant pas présent dans nos sources, nous ne pouvons pas vérifier l'existence de ce dernier.

Fichier *Android.mk* de l'utilitaire *lsusb*

Une fois ces erreurs corrigées, nous pouvons enfin passer à la suite, qui consiste là encore en un fichier makefile : nous devons écrire le fichier *Android.mk* permettant de compiler l'outil *lsusb*, qui permet de tester la librairie *libsub* écrite précédemment.

Les sources de cet outil ainsi que le fichier *Android.mk* (initialement vierge) se trouvent sous *libsub-1.0.3/examples*.

On retrouve dans ce fichier makefile la même structure que notre premier makefile (re-définitions des variables, définition du nom et du tag, inclusion des fichiers *.c, etc...), à quelques différences près :

```
1  LOCAL_PATH := $(call my-dir)
2  include $(CLEAR_VARS)
3
4  LOCAL_MODULE := lsusb
5  LOCAL_MODULE_TAGS := optional
6
7  LOCAL_SRC_FILES := lsusb.c
8
9  LOCAL_C_INCLUDES += $(LOCAL_PATH) \
10                      $(LOCAL_PATH)/../libusb \
11                      $(LOCAL_PATH)/../libusb/os \
12
13 LOCAL_SHARED_LIBRARIES := libusb
14
15 include $(BUILD_EXECUTABLE)
```

Illustration 3: Contenu du fichier libusb-1.0.3/examples/Android.mk

- Premièrement, nous devons ajouter la directive LOCAL_SHARED_LIBRARIES pour inclure notre librairie *libsub*.
- Ensuite, la directive LOCAL_C_INCLUDES doit contenir non seulement les dossiers contenant les fichiers *.h pour l'utilitaire *lsusb*, mais également ceux de la librairie *libusb*, car sinon les headers de notre librairie ne seront pas connu du compilateur, et la compilation échouera, car les fonctions ne seront pas définies.
- Enfin, la dernière ligne ne doit pas inclure $(BUILD_SHARED_LIBRARY), mais $(BUILD_EXECUTABLE) car nous voulons un exécutable et non un fichier librairie en .so comme pour *libusb*.

Une fois ce fichier écrit, nous pouvons compiler et exécuter notre exécutable, et nous avons donc terminé la première partie de ce TP. Tous les fichiers décrits ici sont alors mis dans un *commit* comme décrit dans le sujet, et sont présent sur le repo Git, branche LO52_Tauntaun.

Définition d'un nouveau produit

La seconde partie du TP concernait quant à elle la définition d'un nouveau produit.

Arborescence pour le nouveau produit

Concernant l'arborescence de dossiers pour ce nouveau produit, nous avons choisi de respecter la structure vue en TD au cours du semestre : *device/<nom_vendeur>/<nom_produit>*, ce qui donne dans notre cas *device/utbm/lo52*.

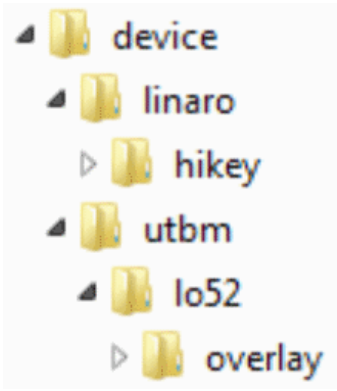

Illustration 4: Arborescence mise en place pour la partie 5 du TP

Nous placerons dans ce dossier *lo52* tous les fichiers décrit par la suite, sauf indication contraire. Nous ajoutons à ce dernier un sous-répertoire nommé *overlay*, qui contient toutes les ressources à surcharger via le méchanisme d'overlay d'Android. Nous y plaçons le fichier *sym_keyboard_delete.png* dans l'arborescence demandée dans le sujet.

Enfin, nous ajoutons l'arborescence *device/linaro/hikey*, dans laquelle nous plaçons le contenu du repository qu'il nous a été demandé de cloner au tout début du TP (*https://android.googlesource.com/device/linaro/hikey*).

Fichiers de base

Passons maintenant aux fichiers de base qui vont nous servir à définir notre produit. Ils sont au nombre de 4.

Vendorsetup.sh

Ce premier fichier va nous permettre d'ajouter des choix à l'outil *lunch*, choix qui vont permettre de build notre produit. Il est requis pour tout produit destiné à une plateforme Android, car sa présence permet de rendre possible le build via l'outil de compilation d'Android (nommé *lunch*).

```
1   add_lunch_combo lo52-user
2   add_lunch_combo lo52-eng
```
Illustration 5: Contenu du fichier vendorsetup.sh

Le sujet nous précise que nous devons définir deux cibles de build, l'une destinée aux utilisateurs (*user*) et l'autre destiné aux développeurs pour le debug (*eng*). Nous avons donc défini deux cibles via la directive *add-lunch-combo*, en faisant varier à chaque fois le suffixe de la valeur pour différencier les builds, le radical correspondant à notre nom de produit.

AndroidProduct.mk

Le deuxième fichier que nous ajoutons est le fichier makefile principal. Lui aussi est requis par Android, car un script va automatiquement le détecter et l'ajouter lors du build.

Cependant, il ne contiendra pas les directives définissant notre produit, mais seulement une directive d'inclusion d'un autre makefile. C'est cet autre fichier makefile qui contiendra la définition.

```
1   PRODUCT_MAKEFILES := $(LOCAL_DIR)/lo52.mk
```

Illustration 6: Contenu du fichier AndroidProduct.mk

BoardConfig.mk

Le troisième fichier est un makefile qui nous permet de définir la configuration hardware du produit.

```
1   include device/linaro/hikey/hikey/BoardConfig.mk
```

Illustration 7: Contenu du fichier BoardConfig.mk

Notre produit héritant de la carte hikey, ce fichier makefile contient une directive d'inclusion du fichier BoardConfig.mk de la carte, qui est présent dans le dossier *device/linaro/hikey/hikey*.

Nous aurions également pu utiliser un autre fichier BoardConfig.mk pour faire cet héritage, puisque le dossier *device/hikey* contient deux autres fichiers de ce type dans ses sous-dossiers, respectivement sous *device/linaro/hikey/hikey32* et *device/linaro/hikey/hikey960*. Nous avons choisi notre fichier pour des raisons de simplicité.

lo52.mk

Enfin, dernier fichier à ajouter pour conclure la définition initiale de notre produit, *lo52.mk* est le fichier makefile qui a été inclus dans *AndroidProduct.mk* précédemment.

```
1  $(call inherit-product, device/linaro/hikey/hikey.mk)
2
3  PRODUCT_PACKAGES += \
4                      libusb
5  PRODUCT_PROPERTY_OVERRIDES += \
6                                ro.hw=lo52 \
7                                net.dns1=8.8.8.8 \
8                                net.dns2=4.4.4.4
9
10 DEVICE_PACKAGE_OVERLAYS := device/utbm/lo52/overlay
11
12 PRODUCT_NAME := lo52
13 PRODUCT_DEVICE := lo52
14 PRODUCT_BRAND := lo52
15 PRODUCT_MODEL := lo52
```

Illustration 8: Contenu du fichier lo52.mk

Il commence tout d'abord par un appel vers la macro *inherit-product*, à laquelle on spécifie le chemin vers le fichier makefile du produit dont on hérite, dans notre cas la carte *hikey*.

Ensuite, on spécifie via une directive PRODUCT_PACKAGES le nom de la librairie faite dans la première partie du TP.

Les lignes 5 à 8 nous permettent de redéfinir les valeurs des propriétés, telles que demandé dans le sujet du TP.

On assigne également la valeur *device/utbm/lo52/overlay* à la variable DEVICE_PACKAGE_OVERLAYS pour activer le mécanisme d'overlay.

Enfin, dans les lignes 12 à 15, on défini différentes caractéristiques de notre produit, respectivement le nom du produit, le nom du device, le nom de la marque du produit et le nom du modèle. Il est important que PRODUCT_NAME et PRODUCT_MODEL aient une valeur égale au nom du répertoire, mais les deux autres variables sont sans contraintes au niveau de la valeur.

Ces quatre fichiers ont été ajouté au repository LO52_Tauntaun dans un commit, comme demandé dans le sujet.