TP MI11 Prise en main de Linux embarqué
Partie 1

Semestre printemps 2017

Guillaume Sanahuja – guillaume.sanahuja@hds.utc.fr

Table des matières

Travail préalable à la séance de TP

Relire le cours sur Linux embarqué disponible sur le moodle MI11. Télécharger les ressources techniques sur le site MI11 et prendre connaissance du manuel utilisateur de la carte DevKit8600 (DevKit8600 User Manual.pdf). Les réponses de certaines questions se trouvent dans le cours et ce manuel.

Un compte rendu de TP au format PDF est à rendre sur le moodle de l'UV. Il vous est demandé d'y écrire les réponses aux questions figurant dans les différents exercices. Également, vous pouvez compléter le compte rendu avec toute information que vous jugerez utile, par exemple les manipulations que vous avez réalisées pendant la séance, les résultats que vous avez observés, …

Pour l'ensemble de ce TP, s'assurer qu'aucune carte SD n'est présente dans la cible.

Exercice 1 : Prise en main de la carte DevKit8600

La première étape consiste à installer la carte sur le PC. Il vous est demandé de brancher la carte via un port série et sur le réseau (brancher le câble Ethernet sur le switch). L'ensemble de ce TP s'effectuera avec une machine virtuelle sous Linux Mint 17.3. Le login et le mot de passe sont mi11/mi11.
Commencez donc par ouvrir *VirtualBox* et faire un **clone lié** de la machine *mi11_devkit8600*. Vous travaillerez sur ce clone.

Question 1.1 : Décrivez les caractéristiques de la carte ? Quelle est la référence du processeur de la carte, quelle est sa fréquence et son architecture ? Quelle est la quantité de mémoire vive sur la carte ?

Question 1.2 : Quelles sont les différentes possibilités pour stocker le noyau et le système de fichiers nécessaires pour démarrer la carte ?

Comme nous l'avons vu en cours, les cibles embarquées offrent dans la majorité des cas une interface de communication disponible dès le démarrage. Utilisez cette interface pour vous connecter à la cible. Sur le PC de développement il est conseillé d'utiliser l'application *cutecom*.

Question 1.3 : Quelle interface de communication avez-vous utilisé pour vous connecter à la cible ? Donnez les paramètres de connexion.

Allumez la carte et constatez le déroulement de la séquence de démarrage sur l'interface de communication.

Question 1.4 : Décrivez la séquence de démarrage? Que se passe-t-il et pourquoi (analysez la configuration sur le PC)? Quel fichier est manquant et où faudrait-il le mettre ? A quoi sert ce fichier ?

Question 1.5 : Si le fichier manquant était présent, que se passerait-t-il ensuite ? Quel serait le problème suivant ?

Exercice 2 : Prise en main de l'environnement Yocto

Dans cet exercice, nous allons prendre en main l'environnement *Yocto* et recompiler l'ensemble des binaires nécessaires à démarrer le linux embarqué sur la cible devkit8600.

Les dossiers suivants sont déjà présents dans la machine virtuelle :

- */home/mi11/poky-dizzy-12.0.3 :* sources de poky version 12, obtenu sur https://www.yoctoproject.org/
- */home/mi11/devkit8600/meta-devkit8600* : couche spécifique à la cible pour poky
- */home/mi11/poky :* répertoire de compilation de poky. Afin de gagner du temps, tout a déjà été précompilé.
- */home/mi11/poky/build/conf* : répertoire de configuration de poky.
- */home/mi11/poky/build/tmp/deploy* : répertoire contenant les fichiers générés par le système de compilation. Ce répertoire est vide pour le moment.

Question 2.1 : Analysez brièvement le contenu des dossiers /home/mi11/poky/build/conf et /home/mi11/devkit8600/meta-devkit8600.

Afin de supporter la devkit8600, il faut ajouter sa couche spécifique à poky. Modifiez pour cela le ficher *bblayers.conf.* Modifiez également la configuration de poky pour sélectionner la bonne cible de compilation.

Question 2.2 : Qu'avez vous changé dans les fichiers de configuration ?

Nous allons maintenant lancer la compilation du noyau et du système de fichier. Pour cela exécuter les commandes suivantes dans un terminal :

cd /home/mi11/poky
source ../poky-dizzy-12.0.3/oe-init-build-env

Gardez ce terminal ouvert pour les futures compilations de poky. Pour lancer une compilation, il faut utiliser la commande *bitbake* suivie de l'image à créer. Afin de faire un système de fichiers basique, lancez :

bitbake core-image-base

Cette commande va prendre du temps mais est grandement accélérée par le fait que tout est déjà précompilé.

Question 2.3 : Analysez le résultat de la compilation se trouvant dans le répertoire /home/mi11/poky/build/tmp/deploy/images. Quels sont les différents fichiers, à quoi servent-ils ? Quelle est la taille des fichiers générés ? Comparez avec ceux de votre VM sous Linux Mint. Pourquoi y a t-il une différence ?

Lancez maintenant la commande suivante afin de compiler la chaîne de compilation croisée :

bitbake meta-toolchain

Ouvrez un nouveau terminal et installez la chaîne de compilation croisée avec les commandes suivantes :
cd /home/mi11/poky/build/tmp/deploy/sdk
sudo ./poky-glibc-x86_64-meta-toolchain-armv7a-vfp-neon-toolchain-1.7.3.sh

Utilisez les réponses par défaut en appuyant sur entrée à chaque question.

Exercice 3 : Démarrer sur le noyau et le système de fichiers générés

Question 3.1 : Compte tenu de l'exercice 1, que faut-t-il faire pour démarrer sur le noyau et le système de fichiers que vous avez générés ?

Effectuez ces opérations et constatez le bon fonctionnement. Le login de la cible est *root* avec un mot de passe vide.

Question 3.2 : Décrivez le processus de boot complet de la cible. Quelles sont les applications démarrées et dans quel ordre ? Fournissez dans le rapport de TP l'ensemble des messages de sortie du terminal entre l'allumage de la cible et le prompt de login.

Pour plus de confort, connectez vous à la cible par ssh, avec la commande suivante :

ssh root@ip_cible

Où *ip_cible* est à adapter à votre situation.

Question 3.3 : Quelle est l'ip de la cible ? Comment l'avez vous trouvée ?

Affichez dans la console le contenu du fichier /proc/devices.

Question 3.4 : Que contient le fichier /proc/devices ? Quelle est la différence entre les 2 sections que l'on trouve dans ce fichier ? Identifiez le périphérique correspondant au port série, mettez le en évidence dans le fichier /proc/devices et affichez la(es) ligne(s) correspondantes dans le dossier /dev. A quoi correspond le numéro que l'on trouve en début de ligne dans le fichier /proc/devices ?

Question 3.5 : Analysez le contenu de /bin, /sbin, /usr/bin et /usr/sbin. Qu'y a t-il de particulier ? Quel est l'avantage ?

Exercice 4 : Ajout de paquets

Le système de fichier généré est basique et contient peu de paquets. Nous allons voir comment l'étoffer. Utilisez bitbake pour compiler le paquet libxml2. Il s'agit d'une bibliothèque pour gérer les xml.

Question 4.1 : Analysez le contenu du dossier /home/mi11/poky/build/tmp/deploy/ipk. Comment est il organisé ? Que contiennent les sous dossiers ? Où se trouve le noyau ? Où se trouve libxml2 ?

Question 4.2 :Quels sont les différents paquets relatifs à libxml2 ?

Copiez le fichier *libxml2_2.9.1-r0_armv7a-vfp-neon.ipk* dans le système de fichiers de la cible et installez le avec le gestionnaire de paquets sur la cible:

opkg install libxml2_2.9.1-r0_armv7a-vfp-neon.ipk

Question 4.3 : Donnez au moins deux méthodes pour copier le fichier dans le système de fichiers de la cible.

Question 4.4 : Où se trouvent les fichiers de libxml2 installés par le gestionnaire de paquets. Quels sont ces fichiers ?

Exercice 5 : Compilation manuelle du noyau

Le but de la seconde séance de TP sera de faire clignoter les LEDs de la carte. Nous devons donc nous assurer aujourd’hui que celles-ci sont opérationnelles.

Question 5.1: A l'aide de la documentation de la carte, comment peut on accéder aux LEDs en mode utilisateur ? Vérifiez cette fonctionnalité sur la cible.

Afin d'ajouter une fonctionnalité au noyau, nous allons le compiler manuellement, comme vu en cours. Les sources du noyau se trouvent dans le dossier : */home/mi11/devkit8600/linux-3.1.0-psp04.06.00.03.sdk*

Il va falloir utiliser la chaîne de compilation croisée installée précédemment. Pour cela utilisez le

script fourni avec la chaîne de compilation croisée :

source /opt/poky/1.7.3/environment-setup-armv7a-vfp-neon-poky-linux-gnueabi
unset LDFLAGS

Question 5.2 : A quoi sert ce fichier ? Quel est le préfixe de la chaîne de compilation croisée ?

Question 5.3 : Comment obtenir la liste des configurations pour défaut du noyau pour une architecture ARM ? Quelles sont les configurations par défaut possibles pour la devkit8600 ? Par la suite on ne gardera que la première.

Effectuez la configuration par défaut ; puis lancez la personnalisation du noyau (*menuconfig*). Nous avons besoin d'y ajouter un driver pour les LEDs connectées par GPIO.

Question 5.4 : Quelle est l'option à activer dans le noyau ? Quelles sont les différentes possibilités pour l'activer ?

Lancez la compilation du noyau en utilisant la cible *uImage* du Makefile afin de créer une image pour *u-boot* appelée *uImage*. Pensez à ajouter l'option *-j* pour lancer plusieurs compilations en parallèle et profiter de tous les threads disponibles sur votre machine. Par exemple *-j2*.

Question 5.5 : Où se trouve le résultat de la compilation ?

Copiez le fichier *uImage* pour qu'il soit utilisé par la cible puis démarrez la.

Question 5.6 : Comment vérifier dans les logs de démarrage que le noyau utilisé est bien celui qui vient d'être compilé ?

Question 5.7 : Comment vérifier dans les logs de démarrage que la fonctionnalité ajoutée est bien présente ?

Exercice 6 : Ajout de paquets, seconde méthode (facultatif)

Nous avons vu comment installer un paquet sur le système de fichiers. Le but de cet exercice est d'inclure directement dans le système de fichiers généré la *libxml2*.

Question 6.1 : En analysant le répertoire /home/mi11/poky-dizzy-12.0.3, où se trouve le fichier responsable de construire la core-image-base ?

Question 6.2 : Analysez les autres images disponibles dans le même répertoire. Que faut-il faire pour ajouter libxml2 à core-image-base ?

Faites les changements nécessaires, recompilez l'image et constatez le résultat.

Exercice 7 : Compilation manuelle des modules noyau (facultatif)

Lors de l'exercice 5, nous ne nous sommes pas occupés des modules noyau, car ceux-ci étaient déjà

présents sur le systèmes de fichiers.

Question 7.1 : Donnez la marche à suivre pour compiler les modules et les installer dans le système de fichiers.