S. Lavirotte
Polytech'Nice – Sophia
SI5 – Master 2 IFI
2015-2016

TD n° 10
Construire son Système Embarqué

Le but de ce TD est de faire la synthèse sur la création d'un système embarqué en incluant toutes les phases nécessaires à l'obtention d'un système fonctionnant sur une architecture différente de celle de votre station de travail et de créer un système embarqué fonctionnel.

1 Quel système embarqué ?

Afin de construire une chaîne de compilation croisée, il faut déterminer le type de matériel pour lequel nous souhaitons mettre en place cet outil. Nous allons travailler avec `qemu` pour émuler un système ARM. Vous pourrez voir l'ensemble des plates-formes que `qemu` est capable d'émuler à l'aide de la commande :

```
qemu-system-arm -M \?
```

Nous allons travailler avec la plate-forme émulée par défaut par `qemu` à savoir :

```
integratorcp926 ARM Integrator/CP (ARM926EJ-S)
```

Nous configurerons donc l'ensemble des outils pour les caractéristiques de cette machine.

2 Construire une chaîne de cross-compilation

Notre but aujourd'hui est de fabriquer cet environnement de compilation croisée afin d'obtenir un système fonctionnel. Nous utiliserons pour cela `BuildRoot`. Vous devrez disposer d'environ 2Go d'espace libre sur la partition sur laquelle vous allez installer `BuildRoot`.

2.1 Installation de BuildRoot

Commencez par récupérer l'image disque contenant les sources de `BuildRoot` et les paquetages nécessaires (afin d'éviter des téléchargements trop longs pendant le TD) :

```
http://trolen.polytech.unice.fr/cours/sae/td10/sdc-buildroot.7z
```

Cette image intègre l'application `BuildRoot` ainsi que les packages nécessaires à sa configuration et son installation déjà préchargés dans cette image (afin d'éviter les trop longs temps de téléchargement).

2.2 Configuration de BuildRoot

Configurez les sources de `BuildRoot` (`make menuconfig`) avec les options suivantes. Vous vous renseignerez par vous-même, au fur et à mesure, sur l'utilité de telle ou telle option.
N'oubliez pas qu'une chaîne de compilation croisée n'est pas relocalisable sur votre système de fichier. Choisissez donc de manière appropriée l'endroit où sera installée votre chaîne de compilation croisée (dans notre cas `/work/toolchain` par exemple). Si, pour une raison ou une autre, vous aviez besoin de relocaliser votre chaîne de compilation croisée, il faudra recréer un lien depuis l'ancien endroit (celui spécifié lors de la compilation) vers la nouvelle localisation.

```
Target Architecture: arm
Target Architecture variant: arm926t
Target ABI: EABI
Build options :
  Host dir: /work/toolchain
  Number of jobs to run simultaneously: 4
  strip: sstrip
Toolchain
  Kernel Headers 2.6.36.x
  Enable RPC
Package selection for the target: unselect all (désélectionner busybox)
Filesystem images: unselect all
```

```
Kernel: none
```

Vous noterez que `BuildRoot` permet aussi d'inclure et de générer la compilation d'un noyau ainsi que l'image d'un système de fichiers pour la plate-forme cible. Nous n'utiliserons pas dans ce TD ces fonctionnalités afin de préserver du temps et en particulier d'éviter d'avoir le temps de compilation du noyau qui vienne s'ajouter à ce processus déjà long. Et nous produirons nous même un système de fichier avec `BusyBox`. Mais sachez que le processus peut s'automatiser grâce à `BuildRoot`.

2.3 Compilation de BuildRoot et option de la chaîne de compilation croisée

Il ne vous reste plus qu'à lancer la compilation grâce à la simple commande `make` (Attention à ne pas utiliser `make -j X` car le système de `BuildRoot` intègre déjà la gestion de la parallélisation des compilations : paramètre *Number of jobs*... que vous avez configuré lors de l'étape précédente).

Suivant la vitesse de votre machine (et de votre connexion Internet car les paquetages nécessaires sont téléchargés au fur et à mesure des besoins), cette opération pourra prendre entre 20 et 60 minutes. Au fur et à mesure de la compilation, vous pourrez regarder ce qui est créé dans le répertoire `/work/toolchain`.

3 Utilisation de la chaîne de compilation croisée

3.1 Utilisation d'un noyau pour booter sur la cible ARM

Vous disposez dans le dossier `td10` d'un fichier contenant un noyau linux compilé pour ARM. Vous pourrez ainsi lancer `qemu` et le faire émuler une architecture ARM.

```
qemu-system-arm –kernel zImage.arm
```

Après le démarrage du noyau, vous obtiendrez une erreur vous indiquant qu'il n'y a pas de système de fichier racine. Nous allons donc en réaliser un comme nous avons pu le faire lors du TD précédent.

3.2 Compilation de BusyBox pour ARM

Le but dans la suite de ce TD est de reprendre le TD précédent n°8 avec la contrainte supplémentaire de produire un système de fichier avec des programmes pour architecture ARM à l'aide de la chaîne de cross-compilation que nous venons de mettre en place. N'oubliez pas de positionner les variables d'environnement du Shell dans lequel vous lancerez la cross-compilation pour ARM (modification des variables ARCH et CROSS_COMPILE de votre Shell ou passage en paramètre à `make`).

Vous vérifierez grâce à la commande `file` que l'exécutable `busybox` que vous avez créé est bien généré avec les bonnes propriétés.

3.3 Création d'un système de fichier pour la cible

Reprenez la première question du TD n°9 pour créer un système de fichier pour notre cible ARM, voir si vous avez terminé le résultat du TD n°9.

Une fois votre système de fichier rempli avec le résultat de la compilation de `BusyBox` pour votre nouvelle cible embarquée ARM, vous pourrez tester que votre système est fonctionnel grâce à `qemu` à l'aide d'une ligne de commande du type :

```
qemu-system-arm –kernel zImage.arm –initrd rootarm.img –append "root=/dev/ram0"
```

TD n° 10
Construire son Système Embarqué

4 Rendu de ce TD

4.1 Un programme Hello World pour la cible

Créez un programme « Hello World ! » que vous compilerez pour x86 et pour ARM avec chargement dynamique des librairies et en tant qu'exécutable statique. Vous veillerez à l'ajouter à votre système ce programme « Hello World » pour vérifier qu'il fonctionne bien sur la cible (vous pourrez le vérifier la cible x86 et ARM).

Notez toutes les valeurs obtenues pour les différentes configurations sur un papier pour l'évaluation à venir.

4.2 Evaluation de ce TD

A la fin des 4 heures, avant de quitter la salle, vous devrez appeler votre encadrant pour lui faire la démonstration que vous avez réalisé ce TD et que vous avez obtenu un système complet pour architecture ARM et répondre à quelques questions supplémentaires.