

## TD n° 6

# Un système sur une cible matérielle

Après avoir largement utilisé la virtualisation et l'émulation pour mettre en œuvre nos systèmes embarqués, le but de ce TD est de produire et d'installer un système pour une cible matérielle. Ce sera donc l'occasion de mettre en œuvre toutes les briques nécessaires à un système embarqué avec système d'exploitation.

## 1 Installation d'un outil de production d'un système complet : Buildroot

Commencez par récupérer l'image disque suivante :

```
http://trolen.polytech.unice.fr/cours/isle/td06/sdd-buildroot.7z
```

Ce disque virtuel doit être monté dans votre machine de travail dans le dossier `/work/td06`.

Cette image intègre les sources de l'outil Buildroot ainsi que les packages de code source nécessaires à la construction du système complet (afin d'éviter les trop longs temps de téléchargement). De plus, afin de vous éviter des temps de compilation trop longs pour ce TD, une première compilation a déjà été réalisée. Donc si vous apportez des modifications au système déjà configuré, la compilation ne sera effectuée que pour les éléments modifiés (donc pas de recompilation de chaîne de cross-compilation, du noyau, ... si ces éléments n'ont pas été modifiés).

Buildroot est un outil d'aide à la construction d'un système complet. Il récupère les archives des codes sources des différents outils nécessaires à la production d'un système embarqué complet : une chaîne de compilation croisée (compilateur et bibliothèques nécessaires pour la production de `gcc`), une bibliothèque C (`uClibc`), `busybox`, un noyau linux et des outils spécifiques pour des cibles particulières (firmware pour la Raspberry Pi). Tous les paquetages déjà téléchargés sont dans le répertoire `dl` dans les sources de Buildroot.

Voici les différentes étapes que nous avons suivies pour obtenir ce qui se trouve dans le disque virtuel que vous avez récupéré. **Les opérations suivantes (section 1) ne sont donc pas à réaliser. Le travail à réaliser se trouve à partir de la section 2. Mais cela ne doit pas vous empêcher de savoir comment nous sommes arrivés à ce résultat que vous pourrez reproduire.**

### 1.1 Téléchargement et installation des sources de Buildroot

Il suffit de télécharger et d'extraire l'archive des sources de Buildroot. Nous avons utilisé la version 2017.11.2.

```
wget http://www.buildroot.org/downloads/buildroot-2017.11.2.tar.bz2
tar xjf buildroot-2017.11.2.tar.bz2
cd buildroot-2017.11.2/
```

### 1.2 Configuration et compilation de Buildroot pour la cible embarquée

Nous allons produire un système pour Raspberry Pi 3. Buildroot intègre une configuration basique pour cette plateforme car celle-ci est très répandue. Nous allons donc repartir de cette configuration comme point de départ pour notre système. Dans le cas d'une plateforme moins connue, il serait nécessaire de faire la configuration manuelle des différents éléments nécessaires pour la plateforme.

```
make raspberrypi3_defconfig
make menuconfig
```

Il est alors possible de modifier la configuration de base pour produire le type de système que vous souhaitez.

Nous avons modifié la configuration de base du système afin d'inclure les options suivantes :

```
Toolchain
Kernel series : 4.9.x
C library (uClibc-ng)
```

Vous pouvez aussi explorer les autres configurations (le type de processeur de la plateforme, ...)

## TD n° 6

# Un système sur une cible matérielle

Le système de compilation de Buildroot intègre automatiquement la compilation en tirant parti de la puissance de calcul disponible (`Build options / Number of jobs to run simultaneously (0 for auto)`). Donc il n'est pas nécessaire d'ajouter l'option `-j` à `make`.

Une fois la configuration effectuée, vous pouvez alors lancer la compilation de tous les éléments nécessaires à la production de votre système complet (la commande suivante vous permettra de sauvegarder toute la trace de la compilation dans le fichier `build.log`).

```
make 2>&1 | tee build.log
```

## 2 Installation du système construit sur la cible

Le résultat de la compilation se trouve dans le dossier `output` et en particulier dans le dossier `output/images` qui contient le ou les systèmes de fichiers à copier sur la cible. Il ne nous reste alors plus qu'à copier ce ou ces systèmes de fichiers sur la cible pour pouvoir démarrer notre système embarqué.

La procédure d'installation du système sur la cible est spécifique à celle-ci, mais le principe général reste le même : il faut copier le ou les systèmes de fichiers à l'aide d'un utilitaire sur la mémoire Flash ou l'Eprom<sup>1</sup> de la cible. Dans notre cas d'application, nous allons devoir installer le système sur la carte SD que nous insérerons dans la plate-forme Raspberry Pi.

### 2.1 Analyse du résultat de la compilation avec Buildroot

Vous pouvez vous reporter au cours sur le démarrage de la Raspberry Pi. Mais pour rappel, la carte SD doit contenir deux partitions : une formatée en VFAT et une partition formatée au format Linux (Ext3 ou Ext4).

La compilation de Buildroot a produit deux fichiers contenant le système de fichiers de chacune des partitions : `boot.vfat` et `rootfs.ext2`. Vous pouvez monter respectivement le contenu de ces deux fichiers dans les dossiers `/mnt/vfat` et `/mnt/ext` afin d'examiner les fichiers qui seront déployés dans chacune des partitions. Une fois le contenu inspecté, vous pouvez démonter ces deux fichiers.

Vous pourriez alors vous lancer dans la copie du contenu de ces deux fichiers sur la carte SD. Mais il faudra tout d'abord partitionner la carte SD et formater chacune des partitions avant d'y copier le contenu de chacun des fichiers. Voici les commandes qui vous permettraient de réaliser ces opérations, **mais que vous ne ferez pas, lisez la suite !**

```
fdisk /dev/sdX
# faire une partition 1 de 10Mo de type VFAT et une partition 2 de 60Mo de type ext
mkfs.vfat -n BOOT /dev/sdX1
mkfs.ext2 -L ROOT /dev/sdX2
mount /dev/sdX1 /mnt/raspberry/boot
mount /dev/sdX2 /mnt/raspberry/root
cp -ar /mnt/raspberry/vfat/* /mnt/raspberry/boot/
cp -ar /mnt/raspberry/ext/* /mnt/raspberry/root/
umount /mnt/raspberry/boot /mnt/raspberry/root
```

En fait le contenu de ces deux fichiers a été regroupé dans un fichier `sdcard.img` qui est l'image contenant les deux partitions. Cette image peut être montée et démontée respectivement par les scripts `imgmount.sh` et `imgumount.sh` qui se trouvent dans le dossier `/work/td06`.

```
cd /work/td06
./imgmount.sh buildroot-2017.11.2/output/images/sdcard.img
```

<sup>1</sup> [https://fr.wikipedia.org/wiki/Erasable\\_Programmable\\_Read\\_Only\\_Memory](https://fr.wikipedia.org/wiki/Erasable_Programmable_Read_Only_Memory)

## TD n° 6

# Un système sur une cible matérielle

L'appel à ce script vous affiche en retour le `/dev/loopX` qui a été configuré pour vous donner accès au contenu de ce fichier (cela remplace le `mount -o loop` que l'on a fait jusqu'à présent). Notez le bien car c'est lui qu'il faudra donner en paramètre à l'autre script (`imgumount`) que vous utiliserez pour assurer le démontage des partitions.

```
./imgumount /dev/loopX
```

Le script monte automatiquement les partitions définies dans l'image dans les dossiers `/mnt/raspberry/loopXpY`. Rendez-vous dans ces dossiers pour voir ce qu'ils contiennent.

En inspectant le contenu de ces partitions, vous devez pouvoir répondre aux différentes questions suivantes :

- Quels sont les fichiers contenus dans la première partition ? A l'aide la commande `mount` vérifiez le type de formatage de cette première partition ? Cela correspond-t-il à ce que vous attendiez ?
- A quoi servent chacun des fichiers sur cette première partition ? En particulier, quelle est l'utilité des informations dans le fichier `config.txt` et `cmdline.txt` ?
- Quels sont les fichiers présents sur la seconde partition ? A quoi correspondent-ils ? Quel est le formatage de la deuxième partition ?
- Quel système a été utilisé pour vous fournir les différentes commandes de base Unix ?
- Quel système a été utilisé pour le démarrage automatique des services ?

## 2.2 Enregistrement du système produit sur la carte SD

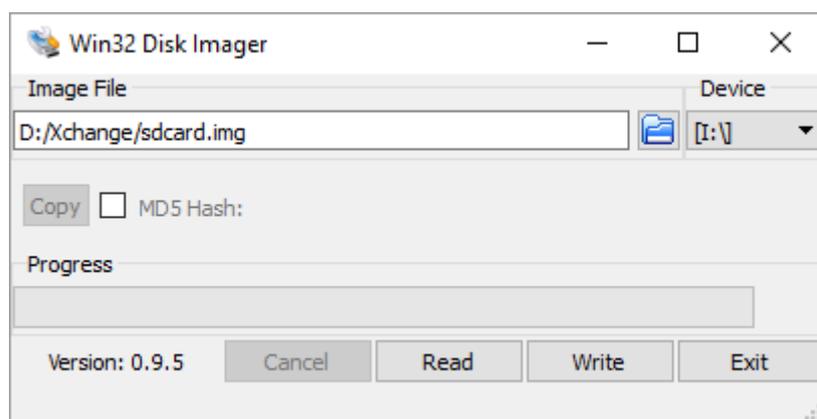
Démontez le fichier `sdcard.img` et rendez le accessible à votre machine physique pour copier le contenu du fichier sur la carte SD. Cette copie demande d'avoir un accès direct à la carte et ne peut être réalisé facilement depuis la machine virtuelle de travail qui fonctionne sous VirtualBox.

### 2.2.1 Copier l'image sur la carte SD depuis une machine Windows

Sous Windows, l'application à utiliser pour faire la copie de l'image est `win32diskimager`. Vous trouverez cette application à l'adresse suivante :

<http://trolen.polytech.unice.fr/cours/isle/td06/Win32DiskImager-0.9.5-install.exe>

Après avoir chargé l'image `sdcard.img`, il vous suffit de lancer l'écriture de cette image sur la carte SD. Ce même outil vous permet aussi de faire une sauvegarde (dans le même format `.img`) des données présentes sur la carte SD.



### 2.2.2 Copier l'image sur la carte SD depuis une machine Unix

Depuis un système Unix, pensez à tout d'abord démonter la carte si celle-ci a été montée automatiquement, puis :

```
dd bs=1M if=sdcard.img of=/dev/sdX
sync
```

## TD n° 6

# Un système sur une cible matérielle

La commande `sync` permet d'assurer que les caches systèmes ont bien été vidés donc que les écritures ont bien toutes été effectuées sur le support. Vous pouvez alors démonter la carte de votre système et l'insérer dans la Raspberry Pi.

### 2.2.3 Pour d'autres plateformes ou de plus amples informations.

Pour de plus ample information sur un système particulier, vous pouvez consulter la page :

<https://www.raspberrypi.org/documentation/installation/installing-images/README.md>

## 2.3 Test de votre configuration

Pour tester la configuration par défaut qui a été obtenu, démarrez une plateforme Raspberry Pi 3 avec la carte SD sur laquelle vous venez d'inscrire la distribution que vous avez obtenue.

## 3 Modifications/Aménagement du système produit

Maintenant que nous disposons d'un système de base, nous allons apporter des modifications à celui-ci pour l'adapter très précisément à nos besoins. Nous pourrions faire des modifications directement sur la plateforme Raspberry Pi en nous connectant sur le système. Mais toutes les modifications effectuées manuellement devraient être reproduites sur chacune des cibles dans le cas du déploiement de l'image sur de nombreux matériels.

Il est donc préférable de modifier le paramétrage de l'outil de construction du système (à l'aide de `make menuconfig`) pour qu'il intègre les modifications souhaitées sur la cible et ainsi produire un système reproductible au besoin sur `n` cibles.

```
System configuration
  Hostname: votre nom
  System Banner: Welcome to My Custom Raspberry Pi 3 System
  Root password: choisissez un mot de passe pour accéder à votre système
  Network interface to configure through DHCP: eth0
  Path to the users table: $(TOPDIR)/../raspberrypi3/users.cfg
  Root filesystem overlay directories: $(TOPDIR)/../raspberrypi3/overlay
```

### 3.1 Création de la table des utilisateurs

Nous avons rempli l'option `Path to the users tables` avec le nom d'un fichier qui contient la liste des utilisateurs. Il doit y avoir un compte par ligne. Les champs, séparés par un espace, sont les suivants :

- `login` : identifiant de connexion du compte (sauf `root`).
- `uid` : numéro d'utilisateur. `-1` pour que l'attribution soit automatique.
- `group` : groupe principal de l'utilisateur. Généralement le même nom que le `login`, ou alors un groupe global pour tous les comptes, comme `users`.
- `gid` : numéro du groupe. `-1` pour une attribution automatique.
- `password` : le mot de passe, en clair si précédé d'un '=', crypté sinon. Si le mot de passe est '!', pas de connexion possible (compte utilisé pour un démon système par exemple).
- `home` : répertoire personnel (aucun si '-').
- `shell` : le shell de connexion. Sur notre système minimal, `/bin/sh` est un shell `ash` inclus dans `Busybox`.
- `groups` : ce champ contient la liste des groupes supplémentaires auxquels appartient l'utilisateur (`-1` si aucun).
- `gecos` : des informations sur le compte, comme le nom en clair de l'utilisateur. Ce dernier champ peut contenir éventuellement des espaces.

Voici le contenu du fichier `users.cfg` à créer :

## TD n° 6

# Un système sur une cible matérielle

```
user -l user -l =user /home/user /bin/sh - Raspberry Pi 3 user
pi -l pi -l =raspberrypi /home/pi /bin/sh - Raspberry Pi 3 pi user
```

### 3.2 Création du dossier overlay

Le répertoire indiqué dans l'option `Root filesystem overlay directories` est l'origine d'une arborescence qui sera appliquée « par-dessus » le système de fichiers obtenu à l'issue des compilations et installations de `Buildroot`, avant de préparer l'image de sortie. Autrement dit notre arborescence va venir se superposer (remplaçant éventuellement des fichiers) à celle se trouvant dans `output/images` et donc dans les fichiers images résultants (`rootfs.ext2` ou `sdcard.img`).

Vous devez donc ajouter dans cette arborescence tout fichier (fichier de configuration, programme, script) qui sera ajouté à l'arborescence finale sur le système de fichier de la cible.

Vous avez pu constater que la configuration du clavier était en anglais. Ajouter donc dans l'overlay les fichiers nécessaires pour bénéficier automatiquement d'un clavier en français (cf TDs précédents).

### 3.3 Ajout d'applications au système

De très nombreux paquetages sont disponibles et peuvent être directement intégrés à l'image de notre système. Ajoutez les applications suivantes à votre image :

- Le serveur `avahi` (service de découverte sur le réseau local) : ajoutez toutes les options
- Le serveur `dropbear` (ssh) : ne pas ajouter d'options supplémentaires
- Le serveur `httpd` de `busybox` (web) : modifier la configuration de `busybox`

Ces différents serveurs vous permettront d'étendre les fonctionnalités de votre système embarqué en particulier pour trouver celui-ci plus facilement sur le réseau (`avahi`), s'y connecter à distance (serveur `ssh`) et fournir du contenu (serveur `web`).

Ajoutez ces 3 serveurs à votre configuration. Le lancement d'`avahi` et de `dropbear` sera automatiquement configuré par l'ajout du paquetage. Par contre, pour l'ajout de `httpd` et son lancement, ce travail sera à votre charge. Vous devrez regarder la manière dont `rcS` est fait et ajouter les fichiers nécessaires pour le chargement de la bonne configuration du clavier et pour le lancement du serveur web en ne cassant pas la configuration existante du système.

### 3.4 Test

Vous commencerez par tester que vous pouvez bien vous connecter à la plateforme Raspberry Pi avec les identifiants que vous avez ajoutés (et pas avec le compte `root` du système).

Vous essayerez ensuite de trouver l'adresse IP prise par votre plateforme grâce à la découverte de service (serveur `avahi`), puis, vous vous connecterez à distance grâce au serveur `ssh`. Enfin, vous tenterez d'accéder au contenu de la page Web que vous avez mis en place.

## 4 Conclusion

L'utilisation du dossier `overlay` permet d'ajouter tout fichier supplémentaire de configuration ou tout programme à votre distribution. Ceci permet la reproductibilité de la création de votre système complet sans devoir effectuer des opérations manuelles sur la cible.