S. Lavirotte
Polytech Nice Sophia
SI4
2020-2021

TD n° 5
Production de Systèmes Embarqués : Buildroot

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/td05/sde-buildroot.7z
```

Ce disque virtuel doit être monté dans votre machine de travail dans le dossier `/work/td05`.

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 librairies nécessaires pour la production de `gcc`), une librairie C (`uClibc`), `busybox`, un noyau linux et des outils spécifiques pour des cibles particulières (firmware pour la Rapsberry 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 commence 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 `Builroot`. Nous avons utilisé la version `2020.02.11`

```
wget https://buildroot.uclibc.org/downloads/buildroot-2020.02.11.tar.bz2
tar xjf buildroot-2020.02.11.tar.bz2
cd buildroot-2020.02.11/
```

1.2 Configuration et compilation de Builroot pour la cible embarquée

Nous allons produire un système pour Raspberry Pi3. `Buildroot` intègre une configuration basique pour cette plate-forme 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 plate-forme moins connue, il serait nécessaire de faire la configuration manuelle des différents éléments nécessaires pour la plate-forme.

```
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 : 4.19.x
    C library (uClibc-ng)
```

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

TD n° 5
Production de Systèmes Embarqués : Buildroot

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
```

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.

2 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 formattée en VFAT et une partition formatée au format Linux (`ext3` ou `ext4`).

La compilation de Builroot a produit deux fichiers contenant le système de fichier de chacune des partitions : `boot.vfat` et `rootfs.ext2`. Vous pouvez montez 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 cartes SD. Mais il faudra tout d'abord partitionner la carte SD et formatter 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/td05`.

```
cd /work/td05
./imgmount.sh buildroot-2020.02.11/output/images/sdcard.img
```

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 répondre aux différentes questions qui se trouvent à l'adresse suivante :

```
https://lms.univ-cotedazur.fr/mod/quiz/view.php?id=26296
```

TD n° 5
Production de Systèmes Embarqués : Buildroot

3 Test avec un émulateur de Raspberry Pi 3

Pensez à démonter bien l'image de la carte SD avant de poursuivre (pour éviter un double montage du disque).

Une alternative au test sur le matériel est d'utiliser un émulateur de la plateforme. `qemu` peut être utilisé pour émuler une Raspberry Pi 3 s'il est correctement configuré.

Dans le dossier `/work/td05/qemu-rasp3`, vous trouverez un script qui utilise `qemu` pour émuler une Raspberry Pi 3 : `raspemu.sh`. Contrairement à vos utilisations précédentes de `qemu`, cette configuration lance l'équivalent de l'écran de la Raspberry Pi 3 dans le terminal où vous avez lancé la commande. Vous avez donc intérêt à lancer ce script depuis un nouveau terminal (mais l'avantage est que l'on peut le lancer depuis une console, sans interface graphique).

Pour l'utiliser, il suffit de lui passer en argument le nom de l'image de la carte SD.

```
cd /work/td05/qemu-rasp3
./raspermu.sh /work/td05/buildroot-2020.02.11/output/images/sdcard.img
```

4 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
```

4.1 Création de la table des utilisateurs

Nous avons rempli l'option `Path to the users tables` avec le nom d'un fichier (`users.cfg`) qui contient la liste des utilisateurs que notre nouveau système connaîtra. 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 `user`.
- `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 inclus dans Busybox.
- `groups` : ce champ contient la liste des groupes supplémentaires auxquels appartient l'utilisateur (`-1` si aucun).

TD n° 5
Production de Systèmes Embarqués : Buildroot

– `gecos` : des informations sur le compte, comme le nom en clair de l'utilisateur. Ce dernier champ peut contenir éventuellement des espaces.

Pour vous éviter de créer un tel fichier, vous pouvez récupérer un fichier `users.cfg` à l'adresse suivante :

http://trolen.polytech.unice.fr/cours/isle/td05/users.cfg

Voici le contenu du fichier `users.cfg` :

```
user -1 user -1 =user /home/user /bin/sh - Raspberry Pi 3 user
pi -1 pi -1 =raspberry /home/pi /bin/sh - Raspberry Pi 3 pi user
```

4.2 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).

4.3 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 distante (serveur `ssh`) et fournir du contenu (server `web`).

Ajoutez ces 3 serveurs à votre configuration. Le lancement d'`avahi` et de `dropbear` sera automatiquement configuré par l'ajout du paquetage. Cependant, pour l'ajout de `httpd` et son lancement, ce travail sera à votre charge. Vous devrez regarder la manière dont `rcS` est fait et ajoutez 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.

4.4 Test des nouvelles fonctionnalités ajoutées

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. (Attention aux redirections de ports faites dans le script `raspemu.ssh`)

TD n° 5
Production de Systèmes Embarqués : Buildroot

4.5 Utilisation de la chaîne de cross-compilation de Buildroot

Vous trouverez dans le dossier `output/host/bin` la chaine de cross-compilation pour la cible (la Raspberry Pi 3). Recompiler avec ce compilateur croisé votre programme `hello-world` d'un TD précédent pour pourvoir l'exécuter sur la Raspberry Pi 3. Ajouter votre programme `hello-world` à l'overlay pour regénérer l'image complète contenant ce « code métier ».

5 Installation sur une vraie Raspberry Pi 3

Il était tout de même plus facile de tester à l'aide de `qemu`. Mais maintenant que vous avez mis en point votre système, vous allez pouvoir le tester sur la plateforme matériel.

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[1] de la cible. Dans notre cas d'application, nous allons devoir installer le système sur la carte SD que nous insèrerons dans la plate-forme Raspberry Pi.

5.1 Enregistrement du système produit sur la carte SD

Nous devons enregistrer le contenu du fichier `sdcard.img` sur la carte SD. Cette opération de « flashage » 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. Vous copierez donc le fichier `sdcard.img` sur votre machine physique.

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

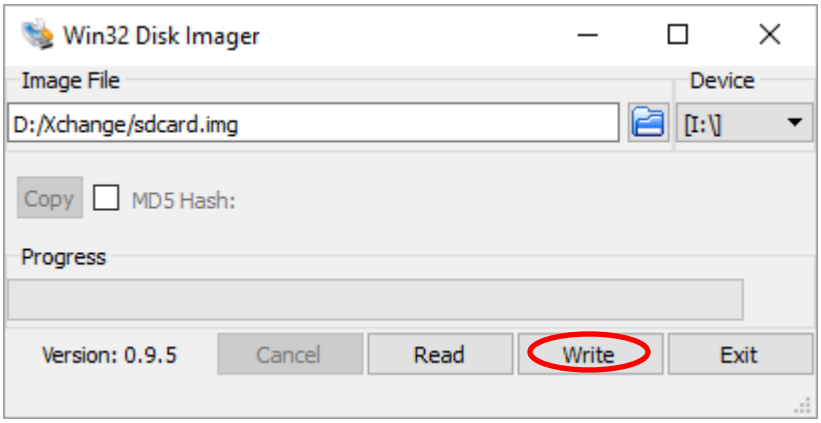
Sous Windows, une des applications à utiliser pour faire la copie de l'image est `win32diskimager`. Vous trouverez cette application à l'adresse suivante :

`http://trolen.polytech.unice.fr/cours/isle/td05/win32diskimager-1.0.0-install.exe`

Si vous rencontrez des problèmes avec cet utilitaire, vous pouvez utiliser à la place `balenaEtcher` :

`https://www.balena.io/etcher/`

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.



5.1.2 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 :

[1] https://fr.wikipedia.org/wiki/Erasable_Programmable_Read_Only_Memory

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

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.

5.1.3 Pour d'autres plateformes ou de plus amples informations.

Pour de plus amples informations sur un système particulier, vous pouvez consulter la page :

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

5.2 Test de votre configuration

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

Et c'est parti vous voyez la console de votre Raspberry Pi qui démarre.

6 Conclusion

L'utilisation du dossier `overlay` permet d'ajout 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.

En complément de ce TD, vous pouvez consulter la page suivante qui contient quelques informations supplémentaires (ajout d'une partition utilisateur, sauvegarde des paramètres réalisés, ...) :

https://www.blaess.fr/christophe/buildroot-lab/index.html