UNIVERSITÉ
PARIS 8

UNIVERSITE PARIS 8
UFR MITSIC
MASTER 2 INFORMATIQUE

Rapport

Création d'un système avec Buildroot

Date de dépot : 06 Décembre 2018

PARIS 8

2 Rue de la Liberté, 93526 Saint-Denis

Réalisé par :

M.ANIS TAIRI

M.NASSIM DJERROUD

2018 - 2019

Contents

1 Introduction

Le projet Buildroot nous fournit désormais une version de travail trimestrielle et une version annuelle maintenue sur le long terme. Buildroot permet de construire un système embarqué plus traditionnel qu'en utilisant une distribution pré-compilée, et d'ajuster plus finement son contenu. Nous allons l'utiliser pour construire un système personnalisé pour Raspberry Pi.

2 Environnement de travail

on crée une arborescence de travail comme suite :

```
[~]$ mkdir buildroot-anis
[~]$ cd buildroot-anis
[buildroot-anis]$
```

Au sein de cet environnement, nous allons essayer de respecter l'organisation des fichiers proposée par le projet Buildroot : une arborescence board/ contenant un sous-répertoire pour chaque carte que nous supporterons. Tous nos fichiers personnalisés se trouveront dans cette sous-arborescence.

```
[buildroot-anis]$ mkdir -p board/rpi-3/
```

à partir de la on télècharge la dérniere version de Buildroot

```
[buildroot-anis]$ wget http://www.buildroot.org/downloads/buildroot-2018.02.7.tar.bz
[buildroot-anis]$ tar xf buildroot-2018.02.7.tar.bz2
[buildroot-anis]$ cd buildroot-2018.02.7/
```

3 Toolchain

Pour produire la toolchain, nous demandons une configuration de Buildroot par défaut pour la cible choisie, et l'élaguons pour ne laisser que la production du compilateur et de ses outils. Les configurations par défaut disponibles sont visibles dans le sous-répertoire configs/. Nous choisissons celle pour Raspberry Pi 3.

```
[buildroot-2018.02.7]$ make raspberrypi3_defconfig
[buildroot-2018.02.7]$ make menuconfig
```

suite à ça on doit effectuer des modification au niveau du menu config

- Menu Build options :
  - Download dir : Nouvelle valeur : $(TOPDIR)/../dl.
  - Host dir : Nouveau chemin : $(TOPDIR)/../board/rpi-3/cross.
- Menu Toolchain :
  - C library : Nouvelle valeur : Linux 4.9.x kernel headers.
  - Build cross gdb for the host : Nouvelle valeur : [*].
- Menu System configuration :
  - Init system : Nouvelle valeur : None.

  - Custom scripts to run before creating filesystem images : Nouvelle valeur : ().
  - Custom scripts to run after creating filesystem images : Nouvelle valeur : ().

- Menu Kernel :
  - Linux Kernel : Nouvelle valeur : [ ]
- Menu Target packages :
  - BusyBox : Nouvelle valeur : [ ] .
- Menu Filesystem images :
  - ext2/3/4 root filesystem : Nouvelle valeur : [ ].

  On sauvgarde les modifications et on lance les commandes suivantes :

```
[buildroot-2018.02.7]$ cp .config ../board/rpi-3/buildroot-2018.02-toolchain.config
```

Après quelques minutes, la compilation se termine avec ces lignes :

```
-e 's#@@STAGING_SUBDIR@@#arm-buildroot-linux-gnueabihf/sysroot#' \
-e 's#@@TARGET_CFLAGS@@#-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 -Os#' \
-e 's#@@TARGET_CXXFLAGS@@#-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 -Os#' \
-e 's#@@TARGET_FCFLAGS@@#-Os#' \ -e 's#@@TARGET_LDFLAGS@@##' \
-e 's#@@TARGET_CC@@#bin/arm-buildroot-linux-gnueabihf-gcc#' \
-e 's#@@TARGET_CXX@@#bin/arm-buildroot-linux-gnueabihf-g++#' \
-e 's#@@TARGET_FC@@#bin/arm-buildroot-linux-gnueabihf-gfortran#' \
-e 's#@@CMAKE_SYSTEM_PROCESSOR@@#armv6l#' \
-e 's#@@TOOLCHAIN_HAS_FORTRAN@@#0#' \
-e 's#@@CMAKE_BUILD_TYPE@@#Release#' \ /home/testing/Build/br-tree/buildroot-2018.02.7/support/misc/too
```

La toolchain de cross-compilation regroupe tous les outils dont les noms sont préfixés par l'architecture (arm), l'outil de production (buildroot), le système d'exploitation de la cible (linux) et les conventions d'interfaçage binaire entre applications et système (gnueabi). Pour simplifier l'appel des outils, des liens symboliques existent raccourcissant le préfixe pour ne garder que l'architecture et le système d'exploitation. On invoquera donc arm-linux-gcc ou arm-linux-g++.

4 Système complet

Nous allons construire à présent une image d'un système complet, y compris le noyau, en utilisant la toolchain obtenue précédemment. Il nous faut effacer les fichiers objets, fichiers temporaires, etc. produits auparavant et l'on serait tenté de faire un make clean. Abstenons-nous en néanmoins car cela aurait pour effet d'effacer la toolchain compilée. La solution la plus simple pour éviter les erreurs de manipulation est de supprimer le répertoire de compilation de Buildroot et de décompresser à nouveau l'archive téléchargée.

```
[buildroot-anis]$ rm -rf buildroot-2018.02.7
[buildroot-anis]$ tar xf buildroot-2018.02.7.tar.bz2
[buildroot-anis]$ cd buildroot-2018.02.7/
[buildroot-2018.02.7]$
```

Puis nous préparons une nouvelle configuration, toujours, en partant de celle par défaut.

```
[buildroot-2018.02.7]$ make raspberrypi3_defconfig
[buildroot-2018.02.7]$ make menuconfig
```

- Build options :
  - Download dir : Nouvelle valeur : $(TOPDIR)/../dl.
- Toolchain : plusieurs modifications sont nécessaires pour retrouver la toolchain précédente.
  - Toolchain type : Nouvelle valeur : External toolchain
  - Toolchain : Valeur conservée : Pre-installed toolchain
  - Toolchain path : Nouvelle valeur : $(TOPDIR)/../board/RPI-3/cross/usr
  - External toolchain gcc version : Nouvelle valeur : 6.x
  - External toolchain kernel headers series : Nouvelle valeur : 4.9.x
  - External toolchain C library : Nouvelle valeur : glibc/eglibc
  - Toolchain has SSP support? : Nouvelle valeur : [*].
  - Toolchain has RPC support? : Nouvelle valeur : [*].
  - Toolchain has C++ support : Nouvelle valeur : [*]

On sauvragde les modification une nouvelle fois et on lance les commandes suivantes :

```
[buildroot-2015.11]$ cp .config ../board/rpi-3/buildroot-2018.02-system-01.config
[buildroot-2015.11]$ make
```

5 Installation et boot

1. On lance la commande lsblk afin de voir la liste des périphériques blocs et partitions présents.
2. on insérons une carte micro-SD sur le poste de développement (par exemple avec un adaptateur USB), puis relançons la même commande pour voir ce qui est apparu.
3. On lance les commandeq suivantes :

```
[buildroot-2018.02.7]$ umount /dev/sdb
[buildroot-2018.02.7]$ sudo cp output/images/sdcard.img /dev/sdb
```

La copie dure un petit moment, c'est normal, il faut remplir quelques centaines de méga-octets. Pas d'inquiétude sur la taille de la carte SD, Buildroot n'utilise que le minimum vital.

Une fois la copie terminée, j'insère la carte micro-SD sur un Raspberry Pi 3 auquel je suis relié par une liaison série.

Une fois termié on peut la booter sur la carte rasepberry.

6 Conclusion

Nous disposons ainsi d'un système Linux embarqué minimal assez personnalisé. Bien sûr il faudra ajouter de nombreux outils pour avoir un véritable environnement de travail complet, ca nous à permis de connaitre les configurations des systemes linux embarqué.