Émuler le Raspberry Pi sous Debian avec Qemu

Jérémie Decock
http://www.jdhp.org

Janvier 2015

Table des matières

Introduction

Qemu est un logiciel libre de virtualisation, capable de simuler un grand nombre d'architectures matérielles. Il est notamment capable de simuler le processeur ARM 1176JZFS contenu dans le SoC [1] Broadcom BCM2835 du Raspberry Pi. Avec Qemu, il est ainsi possible de manipuler un ou plusieurs Raspberry Pi virtuels dans un ordinateur hôte. Cette virtualisation du Raspberry Pi est très utile pour configurer et tester rapidement des environnements logiciels. Elle permet entre autre de simplifier la création de sauvegardes lors de la mise au point de ces environnements. Elle permet également d'épargner les longs transferts de données avec les cartes SD qui sont utilisées comme support de stockage principale du Raspberry Pi.

Dans ce tutoriel, nous verrons comment émuler un Raspberry Pi avec Qemu sur un système hôte Gnu/Linux Debian Jessie. L'émulation sur Mac OS X et Microsoft Windows est également possible mais n'est pas abordée ici.

1 Installer Qemu

Commençons par installer Qemu sur le système hôte. Pour Debian Jessie, il suffit d'installer le paquet *qemu-system-arm* :

1. Système sur une puce, un système complet embarqué sur un seul circuit intégré, pouvant comprendre un ou plusieurs processeurs, de la mémoire, des contrôleurs et des périphériques.

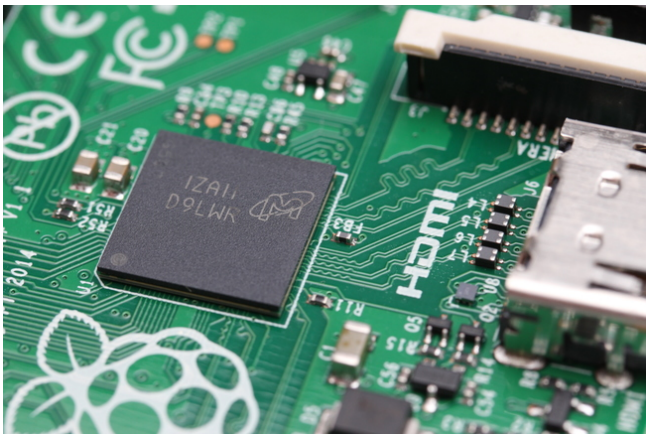

FIGURE 1 – Le SoC Broadcom BCM2835 d'un Raspberry Pi A+.

```
# apt-get install qemu-system-arm
```

Si vous utilisez un autre système d'exploitation que Debian Jessie ou que vous compilez vous-même Qemu, vérifiez que le processeur ARM1176 du Raspberry Pi est bien supporté avec la commande

```
$ qemu-arm -cpu help
```

Cette commande liste les processeurs ARM pris en charge. Assurez-vous que l'arm1176 y figure.

2 Récupérer un système et un noyau pour le Raspberry Pi

L'étape suivante consiste à télécharger l'image du système à émuler. Les images officiellement supportées par la fondation Raspberry Pi sont accessibles sur la page `http://www.raspberrypi.org/downloads/` et de nombreuses autres sont disponibles sur le web. Dans ce tutoriel, nous utilisons le système *Raspbian 2014-12-24* téléchargeable sur `http://downloads.raspberrypi.org/raspbian/images/raspbian-2014-12-25/2014-12-24-wheezy-raspbian.zip` (ce fichier doit être décompressé pour obtenir l'image système « 2014-12-24-wheezy-raspbian.img »).

Nous allons également avoir besoin d'un noyau Linux particulier, compilé pour être utilisable avec Qemu. La page http://xecdesign.com/compiling-a-kernel/ explique comment compiler un tel noyau. Vous pouvez également télécharger un noyau tout prêt à l'adresse suivante : http://xecdesign.com/downloads/linux-qemu/kernel-qemu.

3 Préparation de l'image du système

Dans le cas de Raspbian, l'image système n'est pas directement utilisable par Qemu et nécessite quelques modifications. Les images système du Raspberry Pi contiennent un disque complet avec un MBR et plusieurs partitions (deux dans le cas de Raspbian) ce qui complique un peu leur manipulation [2]. Néanmoins, plusieurs techniques existent pour modifier ce type d'images système [3] et nous utiliserons ici la commande *kpartx* pour effectuer ces modifications. Sur Debian, il faut préalablement installer le paquet *kpartx* :

```
# apt-get install kpartx
```

Les périphériques correspondant aux partitions peuvent ensuite être générés avec la commande

```
# kpartx -av 2014-12-24-wheezy-raspbian.img
```

Cette commande crée les périphériques /dev/mapper/loop0p1 et /dev/mapper/loop0p2 correspondant aux deux partitions contenues dans l'image *2014-12-24-wheezy-raspbian.img*. Seule la deuxième partition nous intéresse ici (celle qui contient la *racine* du système) et nous pouvons la monter normalement avec les commandes :

```
# mkdir /mnt/loop0p2
# mount /dev/mapper/loop0p2 /mnt/loop0p2
```

L'arborescence de la Raspbian est alors accessible et modifiable dans le répertoire « /mnt/loop0p2 ».

La première modification à apporter consiste à désactiver l'utilisation de la bibliothèque *cofi_rpi*, propre au Raspberry Pi et inutilisable dans Qemu [4]. Pour ce faire, éditez le fichier « /mnt/loop0p2/etc/ld.so.preload » et commentez son contenu en mettant un # au tout début de la ligne suivante :

```
/usr/lib/arm-linux-gnueabihf/libcofi_rpi.so
```

2. Voir par exemple le commentaire de Gordon Hollingworth sur `http://www.raspberrypi.org/forums/viewtopic.php?f=29&t=48811&p=380844&hilit=kpartx#p380844`.

3. Certains tutoriels utilisent par exemple la commande *mount* avec l'option « -o loop,offset=XXX », où XXX est l'index du premier bit de la partition à monter (récupérable avec les commandes *fdisk* et *file*). D'autres effectuent les modifications via une instance minimale du système émulé dans Qemu, en désignant un shell comme processus initial, via l'option « init=/bin/bash » du noyau (ce que de nombreux systèmes appellent le « mode de maintenance »). Mais je trouve plus propre et plus souple d'effectuer les modifications depuis le système hôte et de générer automatiquement les périphériques correspondant aux partitions de l'image système via la commande *kpartx*.

4. La bibliothèque cofi_rpi (libcofi_rpi.so) propose une implémentation propre au Raspberry Pi des fonctions *memcpy* et *memset* de la librairie standard. Elle améliore la performance des programmes en exploitant l'accélération matérielle du Raspberry Pi (cf. `http://www.raspberrypi.org/forums/viewtopic.php?f=63&t=9260`).

Sans cette modification, Raspbian plantera au démarrage dans Qemu.

La seconde modification est utile mais pas indispensable. Elle consiste à corriger une différence de nommage des périphériques de stockage entre le noyau utilisé pour Qemu et le noyau de Raspbian. Cette différence est susceptible de générer quelques erreurs. Pour corriger ce problème, il faut éditer le fichier « /mnt/loop0p2/etc/udev/rules.d/90-qemu.rules » avec le contenu suivant :

```
KERNEL=="sda", SYMLINK+="mmcblk0"
KERNEL=="sda?", SYMLINK+="mmcblk0p%n"
KERNEL=="sda2", SYMLINK+="root"
```

Le noyau utilisé par Qemu voit le disque « /dev/sda », alors que le « vrai » noyau de la Raspbian voit « /dev/mmcblk0 ». Notre modification permet de créer des liens symboliques pour être plus proche du vrai Raspberry Pi.

Une fois ces ajustements opérés, nous pouvons démonter la partition *loop0p2* et supprimer les périphériques précédemment générés :

```
# umount /mnt/loop0p2
# kpartx -d 2014-12-24-wheezy-raspbian.img
```

Le script suivant résume toutes les commandes que nous avons utilisées pour configurer l'image du système[5] :

```
#!/bin/sh

RASPBIAN_IMAGE_FILE=2014-12-24-wheezy-raspbian.img
MOUNT_PATH=/mnt/loop0p2

# Mount the partition
kpartx -av ${RASPBIAN_IMAGE_FILE}
mkdir -v ${MOUNT_PATH} 2> /dev/null
mount /dev/mapper/loop0p2 ${MOUNT_PATH}

# Disable libcofi_rpi.so
# Comment "/usr/lib/arm-linux-gnueabihf/libcofi_rpi.so"
# in "${MOUNT_PATH}/etc/ld.so.preload"
sed -ri "s/^.*libcofi_rpi.so/#\0/g" ${MOUNT_PATH}/etc/ld.so.preload

# Fix partition names
cat << 'EOF' >> ${MOUNT_PATH}/etc/udev/rules.d/90-qemu.rules
KERNEL=="sda", SYMLINK+="mmcblk0"
KERNEL=="sda?", SYMLINK+="mmcblk0p%n"
KERNEL=="sda2", SYMLINK+="root"
EOF

# Umount the partition
umount ${MOUNT_PATH}
kpartx -d ${RASPBIAN_IMAGE_FILE}
```

5. Ce script peut être téléchargé à l'adresse suivante : `https://github.com/jdhp-docs/rpi_qemu_part1/tree/master/listings`

4 Premier démarrage

Maintenant que l'image du système est prête, nous allons pouvoir démarrer l'émulation du Raspberry Pi avec la commande suivante :

```
$ qemu-system-arm \
    -kernel kernel-qemu \
    -cpu arm1176 \
    -m 256 \
    -M versatilepb \
    -no-reboot \
    -serial stdio \
    -append "root=/dev/sda2 panic=1 rootfstype=ext4 rw" \
    -hda 2014-12-24-wheezy-raspbian.img
```

On suppose ici que le répertoire courant du terminal contient :

— le noyau Linux adapté pour Qemu dans le fichier « kernel-qemu » ;

— l'image du système Raspbian (aménagée pour Qemu) dans le fichier « 2014-12-24-wheezy-raspbian.img ».

La manpage de Qemu détaille les nombreuses options disponibles :

```
$ man qemu-system-arm
```

L'option « root=/dev/sda2 » de la commande précédente est pertinente pour le système Raspbian mais ne l'est par forcement si vous utilisez un autre système. Si c'est le cas, indiquez le nom du périphérique contenant la partition root.

N'essayez pas d'indiquer plus de 256 Mo de RAM avec l'option « -m », l'émulation du Raspberry Pi ne fonctionnerait pas correctement.

Notez que les options « -no-reboot » et « -serial stdio » sont facultatives. Elles indiquent respectivement de quitter qemu à l'extinction du système émulé et de rediriger le port série virtuel du système émulé vers le terminal courant du système hôte.

L'option « -M versatilepb » indique quant à elle le type de machine ARM à émuler. La page suivante donne plus d'informations à ce sujet : http://www.arm.com/products/tools/development-boards/versatile/index.php.

Ne perdez pas de vue non plus que des alertes peuvent apparaître au démarrage du système émulé car Qemu ne prend pas en charge tous les composants du Raspberry Pi. Les principaux composants sont néanmoins correctement émulés et ces alertes sont dans la plupart des cas sans grande importance.

Conclusion

Bien que l'émulation ne soit malheureusement pas complète (il n'est par exemple pas possible d'exploiter les ports GPIO et SPI du Raspberry Pi depuis une machine virtuelle), vous verrez qu'elle peut néanmoins être très utile

pour configurer et tester rapidement des environnements logiciels nécessitant une configuration spécifique (serveurs dédiés, appliances, robots, etc.).

Le manuel wikibook de Qemu est une bonne source d'informations (en anglais) pour se familiariser avec les nombreuses possibilités offertes par outil : `http://en.wikibooks.org/wiki/QEMU`. La documentation officielle est également disponible (toujours en anglais) sur la page suivante : `http://wiki.qemu.org/download/qemu-doc.html`.

Références

[1] Aurélien Jarno. Debian on an emulated arm machine. `http://www.aurel32.net/info/debian_arm_qemu.php`.

[2] elinux.org. Virtual development board. `http://www.elinux.org/Virtual_Development_Board`.

[3] xecdesign.com. Compiling an arm1176 kernel for qemu. http://xecdesign.com/compiling-a-kernel/.

[4] xecdesign.com. Qemu – emulating raspberry pi the easy way (linux or windows!). http://xecdesign.com/qemu-emulating-raspberry-pi-the-easy-way/.

[5] xecdesign.com. Working with qemu. http://xecdesign.com/working-with-qemu/.

Creative Commons BY-SA