SEOC3 – TP 02 – Installation de Yocto sur un Raspberry Pi

Matériel : 1 lecteur de cartes microSD USB + 1 RPi 3B + 1 alim. microUSB + 1 carte microSD 16 GB + 1 câble HDMI/VGA ou HDMI/DVI

Il vous est demandé un compte-rendu (en PDF) précisant les commandes et les résultats (capture de texte principalement).

1. Création d'une image minimale pour raspberry pi :

a) Installer les paquets nécessaires à votre distribution (voir https://www.yoctoproject.org/docs/ + « Quick build »).

b) Vérifier que vous avez un espace disque d'au moins 40 Gbytes et que la « locale » est `en_US.UTF-8`. Si besoin taper :

```
locale -a
sudo dpkg-reconfigure locales
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
locale
```

c) Puis taper les commandes suivantes (il faut environ 94 minutes suivant les machines) :

```
umask 022
git clone https://git.yoctoproject.org/git/poky
cd poky
git clone https://git.yoctoproject.org/git/meta-raspberrypi
source oe-init-build-env rpi-build
bitbake-layers add-layer ../meta-raspberrypi
echo 'MACHINE="raspberrypi3"' >> conf/local.conf
time bitbake core-image-minimal
```

NB : Pour mesurer le temps de fabrication, on a utilisé la commande `time` *sur la dernière ligne mais elle n'est pas forcément nécessaire.*

d) Contrôler la taille du répertoire `rpi-build` à présent.

e) Pour gagner un peu de place disque, on peut ajouter la directive suivante à la fin du fichier `conf/local.conf` (cela prend environ 10 minutes):

```
INHERIT+="rm_work"
```

Quel espace disque économise-t-on ?

f) Quel est la taille du fichier image créé (Voir `tmp/deploy/images/raspberrypi3/core-image-minimal-raspberrypi3.rpi-sdimg`) ?

g) Copier se fichier sur une carte microSD et tester le démarrage du Rpi avec un câble HDMI/VGA.

h) On peut définir le nom de l'hôte avec la directive suivante à la fin du fichier `conf/local.conf` :

```
hostname_pn-base-files="yocto-XX"
```

Vérifier que cela fonctionne bien.

2. Création d'une couche, d'une recette et configuration IP :

La configuration par défaut intègre une version basique du paramétrage réseau voir `meta/recipes-core/init-ifupdown/init-ifupdown-1.0/interfaces`.

a) Vérifier que le fichier `/etc/network/interfaces` du Rpi est bien identique.

b) Afin de surcharger le contenu de ce fichier pour personnaliser le paramétrage réseau de l'image fabriquée, il faut créer une couche (`layer`) de personnalisation (remplacer XXX par le nom que l'on désire donner à la couche) puis ajouter celle-ci à la construction :

```
bitbake-layers create-layer ../meta-XXX
bitbake-layers add-layer ../meta-XXX
bitbake-layers show-layers
```

c) Éditer le fichier `../meta-XXX/conf/layer.conf` pour changer la priorité de la couche (Voir variable `BBFILE_PRIORITY_meta*`) puis ré-afficher la liste des couches utilisées.

d) Taper ensuite les commandes suivantes afin de créer une recette (`recipe`) de mise à jour du fichier `interfaces` (Adapter bien entendu les valeurs d'adresses IP de l'exemple) :

```
RCP=../meta-XXX/recipes-core/init-ifupdown
mkdir -p $RCP/files
cat << EOF > $RCP/init-ifupdown_1.0.bbappend
FILESEXTRAPATHS_prepend:="\${THISDIR}/files:"
EOF
cat << EOF > $RCP/files/interfaces
# /etc/network/interfaces -- See ifup(8), ifdown(8)
# The loopback interface
auto lo
iface lo inet loopback
# Wired interface
auto eth0
iface eth0 inet static
  address 192.168.2.2
  netmask 255.255.255.0
  gateway 192.168.2.254
EOF
```

e) Copier la nouvelle image sur la carte microSD et avant de la placer sur le Rpi, vérifier directement le contenu du fichier `/etc/network/interfaces` en affichant le contenu de la carte microSD sur votre machine Linux.

f) Vérifier ensuite que le Rpi possède bien la bonne adresse IP sur l'interface eth0.

g) Si on veut changer le mot de passe par défaut de l'utilisateur `root` et ajouter l'utilisateur `iot`, on peut indiquer dans le fichier `conf/local.conf` (noter le **`-P`** majuscule) :

```
INHERIT+="extrausers"
EXTRA_USERS_PARAMS="useradd iot; \
                    usermod -P abc123 iot; \
                    usermod -P secretPwd root; \
                   "
```

Vérifier le contenu du fichier `/etc/shadow` dans l'image créée. On pourra utiliser la commande suivante pour vérifier le mot de passe (en adaptant le « sel ») :

```
openssl passwd -6 -salt XXXXXX
```

h) Pour plus de sécurité au niveau du fichier de configuration, on peut utiliser l'option `-p` de `usermod` et pré-calculer à l'avance les valeurs hachées des mots de passe avec la commande `openssl`. Taper par exemple dans une console :

```
openssl passwd -6 torototo
```

i) Vérifier à nouveau le contenu du fichier `/etc/shadow` dans l'image créée.

3. Ajout de paquets :

a) Ajouter les directives de création d'un historique de fabrication au fichier `conf/local.conf` et étudier les contenu du répertoire `buildhistory` généré :

```
INHERIT+="buildhistory"
BUILDHISTORY_COMMIT="1"
```

b) Afficher ensuite la liste des recettes existantes et vérifier que `example` existe bien : `bitbake-layers show-recipes` ou `bitbake -s`

NB : La recette `example` a été créée automatiquement à l'étape §2b, mais comme elle ne permet pas actuellement de fabriquer un paquet, on l'adaptera par la suite pour s'exercer à la création de paquets personnalisés.

c) Afin d'ajouter un ou plusieurs paquets à notre distribution, il faut indiquer le(s) nom(s) de la recette dans la variable `IMAGE_INSTALL_append`. Ajouter par exemple à la fin du fichier `conf/local.conf` :

```
IMAGE_INSTALL_append="openssh"
```

Vérifier le contenu du répertoire `buildhistory` après avoir refabriqué l'image.

d) On désire maintenant changer le nom de la recette `example` ainsi que sa version. Pour cela, il faut renommer tous les noms depuis le sous répertoire `meta-XXX/recipes-example`. Changer le nom pour `rgb-sensor` version `1.0` par exemple.

e) Fixer alors le contenu du fichier `*.bb` de cette recette avec les instructions suivantes :

```
DESCRIPTION="RGB sensor application"
SECTION="Examples"
LICENSE="GPLv2"
LIC_FILES_CHKSUM="file://${COMMON_LICENSE_DIR}/GPL-2.0;md5=801f80980d171dd6425610833a22dbe6"
PR="r0"
SRC_URI="file://${PN}-${PV}.tar.gz"
SRC_URI[md5sum]="e574da4da3f63804ab3509888e516a52"
SRC_URI[sha256sum]="24a6a3cc96027ff52d3670ad867df7e05e09fdd22c9eac3c78961b1d7e61b58d"
EXTRA_OECMAKE=""
inherit cmake
```

NB : Il est aussi possible de fournir les sources du paquet à l'aide d'une URI http, https, ftp, cvs, svn, git, etc...

Puis copier le fichier `rgb-sensor-1.0.tar.gz` dans le sous répertoire `files` de la recette et lancer la fabrication manuellement :

```
bitbake rgb-sensor
```

f) Pour intégrer ce paquet à votre distribution, ajouter alors la directive suivante au fichier `conf/local.conf` (noter le **+=**) :

```
IMAGE_INSTALL_append+="rgb-sensor"
```

Vérifier que le fichier `/usr/bin/rgb-sensor` est alors bien disponible dans l'image créée.

g) Pour utiliser ce programme sur le Rpi, il faut activer l'I2C et ajouter le module correspondant du noyau Linux (on ajoute ici aussi les outils i2c pour vérifier que le système voit bien les capteurs, mais ce n'est pas forcément nécessaire). Ajouter alors les directives suivantes au fichier `conf/local.conf` :

```
ENABLE_I2C="1"
IMAGE_INSTALL_append+="i2c-tools kernel-modules"
KERNEL_MODULE_AUTOLOAD+="i2c-dev"
```

h) Tester alors l'image sur le Rpi et vérifier que le module i2c a bien été chargé. Tester le fonctionnement du programme `rgb-sensor` en branchant le capteur sur les broches (3.3v – 1, SDA.1 – 3, SCL.1 – 5, GND – 9).