Cross Compilation d’application Qt pour Processeur ARM

Après plusieurs essais infructueux, j’ai reussi à cross compiler Qt pour processeur ARM et à faire tourner une application compilée depuis mon ordinateur sur une BananaPi. Etant débutant en la matière, je n’ai pas trouvé d’explications claires à tous les problèmes que j’ai pu rencontrer.
Je vais donc vous expliquer la procédure que j’ai effectué pour cross compiler Qt comme il se doit.

Configurer et Compiler Qt

Pour commencer, j’ai utilisé la dernière version du compilateur arm-linux-gnueabihf:

```
apt-get install gcc-arm-linux-gnueabihf g++-arm-linux-gnueabihf
```

J’utilise la version de Qt 5.8 everywhere que vous trouverez facilement sur internet.

Une première étape afin de cross compiler Qt est de le configurer correctement. Pour cela, vous devez créer un sysroot de votre target.

Créer le sysroot:

Pour rendre Qt portable, il faudra que Qt génére un fichier libqxcb.so lors de sa compilation. Pour générer ce fichier, il faut installer les libxcb sur votre target:

```
apt-get install "^libxcb.*" libx11-xcb-dev libglu1-mesa-dev libxrender-dev
```

Une fois les librairies installées, vous pouvez copier le /usr et /lib et /etc/ld.so.conf.d de votre target sur votre ordinateur en les plaçant dans un dossier sysroot

Configurer Qt:

Pour configurer Qt j'ai dans un premier temps modifier le fichier [path/to/qtbase/dir/]mkspecs/linux-arm-gnueabi-g++/qmake.conf:

```
#
# qmake configuration for building with arm-linux-gnueabi-g++
#

MAKEFILE_GENERATOR      = UNIX
CONFIG                 += incremental
QMAKE_INCREMENTAL_STYLE = sublib
#QMAKE_LFLAGS = -static -static-libgcc

include(../common/linux.conf)
include(../common/gcc-base-unix.conf)
include(../common/g++-unix.conf)

# modifications to g++.conf
QMAKE_CC                = arm-linux-gnueabihf-gcc
QMAKE_CXX               = arm-linux-gnueabihf-g++
QMAKE_LINK              = arm-linux-gnueabihf-g++
QMAKE_LINK_SHLIB        = arm-linux-gnueabihf-g++

# modifications to linux.conf
QMAKE_AR                = arm-linux-gnueabihf-ar cqs
QMAKE_OBJCOPY           = arm-linux-gnueabihf-objcopy
QMAKE_NM                = arm-linux-gnueabihf-nm -P
QMAKE_STRIP             = arm-linux-gnueabihf-strip
load(qt_config)
```

puis j'utilise la ligne de commande suivante pour configurer qt:

> *./configure -opensource -confirm-license -prefix /usr/local/Qt-5.8-arm -xplatform linux-arm-gnueabi-g++ -nomake examples -nomake tests -sysroot /home/valentin/sysroot -xcb -no-opengl -no-pch*

-xplatform: indique le compilateur utilisé
-sysroot: chemin vers votre sysroot
-prefix: chemin où s'installe la bibliothèque
-xcb: impose la création de la lib qxcb
-no-opengl: ignorer la lib opengl
-no-pch: ignorer les headers précompilé (précompilé pour processeur x86 donc à ignorer)
-nomake: ne pas compiler les fichiers exemples et tests pour gagner du temps
- opensource: licence opensource
-confirm-license: confirmer la licence

A ce moment, vous pouvez avoir une erreur liée au linker ld. En effet pour une raison inconnue, mon linker ne prenait pas les paramètres du sysroot pour sa config. Il prenait la configuration de mon linker natif. Ne trouvant pas comment modifier ce défaut, j'ai ajouté à la config de mon linker deux lignes:

vim /etc/ld.so.conf.d/arm-linux-gnueabihf.conf

/home/valentin/sysroot/lib/arm-linux-gnueabihf
/home/valentin/sysroot/usr/lib/arm-linux-gnueabihf

Une fois ces lignes ajoutées, plus de problème de linker.

Compiler Qt

Qt se configure correctement, pour compiler Qt faite un:

make -j4
make install

Qt s'installera dans le dossier indiqué par la commande -prefix de la configuration.

Déploiement d'application

Une fois la bibliothèque Qt compilée, vous pouvez l'installer sur votre BananaPi, pour programmer des applications directement dessus.

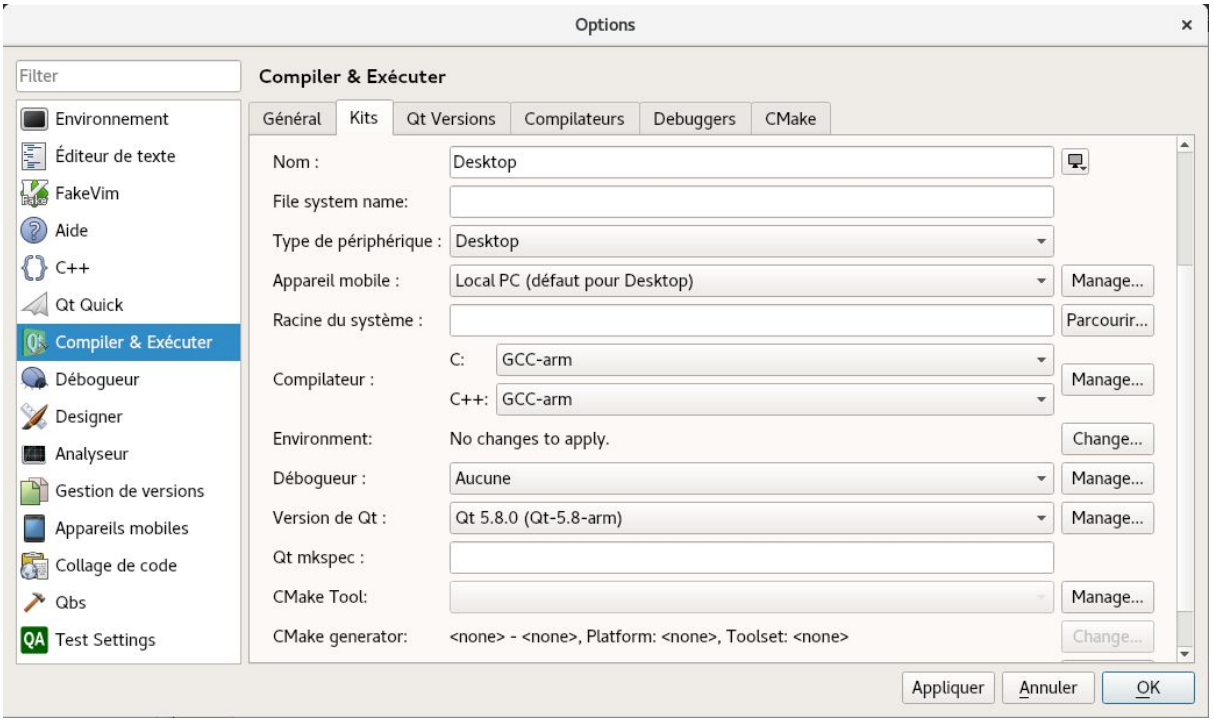
Si vous souhaitez compiler une application sur votre ordinateur et la lancer sur votre Banana, il vous faudra configurer votre IDE pour cela, puis créer un package pour votre application.

Personnellement j'utilise l'IDE de base de Qt, Qt creator.

Configuration de Qt Creator

Pour compiler une application ARM sous Qt creator, il faut indiquer l'emplacement de votre compilateur ainsi que la bibliothèque Qt ARM à votre IDE.

Dans l'onglet, outils>options de Qt creator rendez-vous dans l'onglet "Qt version" pour ajouter votre version de Qt qui se trouve dans votre dossier sysroot. Puis ajoutez les compilateurs GCC et G++ pour ARM dans l'onglet "compilateurs".
Une fois la version de Qt et les compilateurs ajoutés, modifier la configuration de Qt ou créez en une nouvelle dans l'onglet "kit" en indiquant à Qt creator d'utiliser votre version ARM de Qt et le compilateur ARM.



Un fois cette étape finie, il ne reste plus qu'à aller dans l'onglet "projet" de Qt Creator pour ajouter

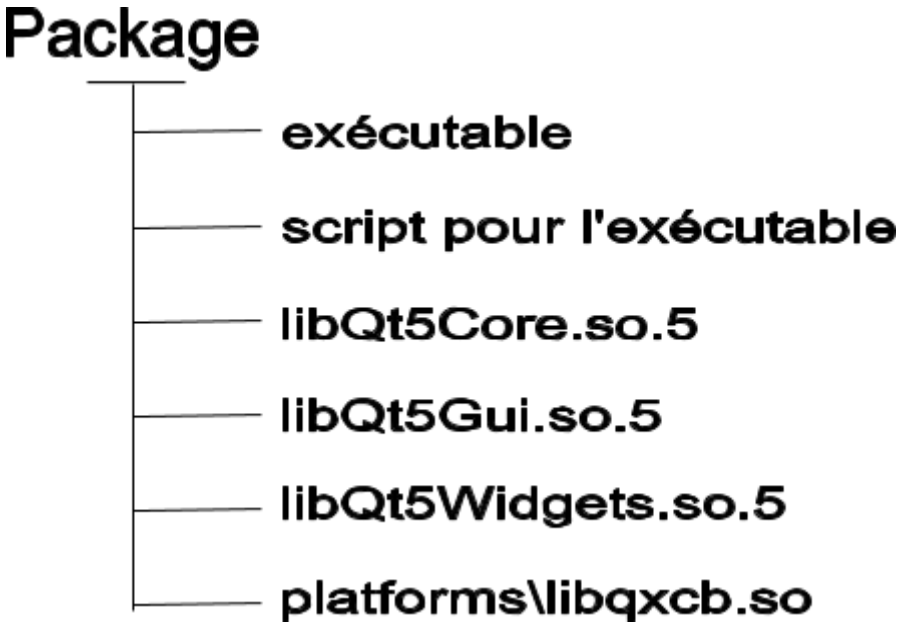
-spec linux-arm-gnueabi-g++ : en argument supplémentaires au Qmake

ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- : en argument du Make

Vous pouvez maintenant compiler votre application, il ne reste plus qu'a créer un package pour que votre BananaPi puis exécuter votre projet.

Créer un package

Pour créer votre package, je vous conseille de créer un nouveau dossier où vous placerez vos fichiers selon cette arborescence:



Le script pour lancer une application nous est donné par Qt:

```
#!/bin/sh
appname=`basename $0 | sed s,\.sh$,,`

dirname=`dirname $0`
tmp="${dirname#?}"

if [ "${dirname%$tmp}" != "/" ]; then
dirname=$PWD/$dirname
fi
LD_LIBRARY_PATH=$dirname
export LD_LIBRARY_PATH
$dirname/$appname "$@"
```

Le script doit s’appeler [nom_de_votre_application].sh

Une fois votre package créé, vous pouvez le transférer sur votre bananaPi et exécuter le script pour démarrer votre application.

Dépendance pour l’application

Si vous n'installez pas votre version de Qt ARM sur votre BananaPi, il se peut qu’il vous manque des dépendances pour lancer votre application.
Vous pouvez corriger ce problème en utilisant la commande “ldd” sur les .so de votre package pour voir si des liens sont brisés. Vous pouvez ajouter les librairies manquantes à votre package et les liés par un lien symbolique avec la commande “ln” pour résoudre le problème.