Mise en place d'un environnement de cross-compilation.

Matthieu Vigne

21 novembre 2017

Table des matières

1 Objectif

Pour pouvoir cross-compiler le code du robot afin qu'il soit exécutable par la Beaglebone, trois choses sont nécessaires. Ces trois étapes sont communes quel que soit le système utilisé, mais la manière de procéder sera différente selon la distribution (ici, Debian (et probablement Ubuntu fonctionne de la même manière) ou Fedora, pour les autres distributions il faudra chercher vous-même ce qui est disponible sur les dépots, mais la méthode est la même).

Tout d'abord, il faut un compilateur C capable de créer du code pour arm, plus précisément la plateforme *armhf* (pour ARM v7 et supérieur). Le compilateur C sous Linux s'appelle gcc, le compilateur à installer s'appelle **gcc-arm-linux-gnueabihf**.

Ensuite, notre code utilise la glib. Il faut donc disposer d'une version de cette bibliothèque compilée pour la plateforme arm (et pas la bibliothèque utilisée par votre ordinateur, qui n'est pas exécutable par la Beaglebone).

Enfin, pour pouvoir compiler simplement votre code, il faut dire à *pkg-config* où se trouve la glib pour arm (celle-ci ne sera pas ajouté automatiquement à la liste de *pkg-config* comme c'est normalement le cas, tout simplement pour ne pas supprimer la bibliothèque de votre ordinateur). Pour cela, il faut copier le fichier *glib-2.0.pc* contenant la configuration qu'utilise *pkg-config* au bon endroit (après avoir renommé ce fichier pour qu'il ne remplace pas la bibliothèque de votre système).

2 Configuration pour Debian

Ces instructions ont été testées sur Débian 8 (Jessie).

L'installation Debian est assez rapide, puisque les dépôts officiels disposent déjà du compilateur et de la glib cross-compilée pour arm ! Il suffit de dire à votre système d'ajouter la plateforme arm, puis d'installer les paquets comme d'habitude : seule différence, le suffixe armhf est ajouté à l'installation de la glib pour obtenir la bonne version.

```
$ sudo dpkg --add-architecture armhf
$ sudo apt-get update
$ sudo apt-get install gcc-arm-linux-gnueabihf
$ sudo apt-get install libglib2.0-dev:armhf
```

Les deux premières étapes ont été effectuées, il ne reste plus que la troisième à faire, configurer pkg-config. Pour cela, commencez par éditer le fichier *glib-2.0.pc* pour modifier sa description (ce n'est pas obligatoire, mais je vous le conseil pour ne pas le confondre par la suite) :

```
$ cd /usr/lib/arm-linux-gnueabihf/pkgconfig/
$ sudo nano glib-2.0.pc
```

Modifiez librement les lignes commençant par *Name :* et *Description :* pour expliquer qu'il s'agit de la bibliothèque pour arm (c'est ces descriptions que *pkg-config* vous montrera par la suite).

Ensuite, copiez ce fichier dans le répertoire de recherche de pkg-config :

```
$ sudo mv armglib-2.0.pc /usr/share/pkgconfig/armglib
    -2.0.pc
```

Et voilà, c'est terminé ! Pour vérifier que cela à fonctionner, tapez :

```
$ pkg-config --list-all | grep arm
```

Ceci devrait faire apparaitre la bibliothèque *armglib-2.0*, avec la description que vous avez entrée (si ce n'est pas le cas c'est que *pkg-config* utilise un répertoire différent, cherchez où se trouve les autres fichiers *.pc* de votre système).

3 Configuration pour Fedora

Ces instructions ont été testées pour Fedora 23/24.

Les dépôts officiels de Fedora ne contiennent ni le compilateur ni la glib pour arm (en tout cas pas en Février 2016, mais vous pouvez tester pour voir, si ces paquets sont disponibles, dans ce cas installez-les directement). Par contre, le compilateur est disponible sur Copr, un autre dépôt de Fedora qu'il faut activer.

```
$ sudo dnf copr enable lantw44/arm-linux-gnueabihf-
    toolchain
$ sudo dnf install arm-linux-gnueabihf-gcc arm-linux-
    gnueabihf-glibc
```

Ceci a installé le compilateur, ainsi que *glibc*, une bibliothèque contenant les fonctions de base du c (tel que printf... Attention, glibc est entièrement différent de glib !).

L'installation de la glib est plus longue, car je n'ai pas trouvé de version déjà compilée : il faut donc le faire nous-même. Pour cela, il faut commencer par cross-compiler (à la main là aussi) les bibliothèques dont dépend la glib.

Tout d'abord, libffi : téléchargez la dernière version (https://sourceware.org/libffi/, v 3.2.1 en février 2016) et décompressez l'archive ; placez-vous dans le dossier créé puis faites :

```
$ ./configure --host=arm-linux-gnueabihf --prefix=/
    usr/arm-linux-gnueabihf/sys-root/
$ make
$ sudo make install
```

La première ligne configure la bibliothèque, ici vous lui dites qu'il s'agit de compiler pour arm, et que vous voulez l'installer dans */usr/arm-linux-gnueabihf/sys-root/* (ce répertoire a été créé à l'installation du compilateur, et est prévu pour que vous y installiez toutes les bibliothèques cross-compilées, afin de ne pas les mélanger avec votre système). Le fichier *configure* crée ensuite un Makefile, que vous exécutez pour compiler puis installer.

La méthode est absolument la même pour la zlib (http://www.zlib.net/, v 1.2.8 en février 2016)

```
$ CC=arm-linux-gnueabihf-gcc ./configure --prefix=/
    usr/arm-linux-gnueabihf/sys-root/
$ make
$ sudo make install
```

Le dernier prérequis est libpcre : cette petite bibliothèque est utilisée pour les expressions régulières, et n'a pas besoin d'être cross-compilée (je suppose qu'elle n'ajoute que des headers, en tout cas rien de compilé).

```
$ sudo dnf install pcre-devel
```

Enfin, pour installer la glib, commencez par la télécharger sur http://ftp.gnome.org/pub/GNOME/sources/glib/ (v 2.46.2 en février 2016), puis extraire le fichier et placez-vous dans le dossier correspondant.

Cette fois, pour configurer la glib, il ne suffit pas de donner des instructions à *configure* : un fichier supplémentaire de configuration est nécessaire. Commencez par créer ce fichier dans un répertoire *build* (c'est là que la glib va être compilée).

```
$ mkdir build
$ cd build
$ gedit arm.cache
```

Ceci crée le fichier *arm.cache* et l'ouvre avec *gedit* : copiez alors le contenu suivant dans ce fichier :

Listing 1 – Fichier arm.cache

```
glib_cv_long_long_format=ll
glib_cv_stack_grows=no
glib_cv_uscore=yes
ac_cv_func_posix_getpwuid_r=no
ac_cv_func_posix_getgrgid_r=no
LIBFFI_CFLAGS="-I/usr/arm-linux-gnueabihf/sys-root/
    lib/libffi-3.2.1/include"
LIBFFI_LIBS="-L/usr/arm-gnueabihf/sys-root/lib -lffi"
ZLIB_CFLAGS="-I/usr/arm-linux-gnueabihf/sys-root/
    include"
ZLIB_LIBS="-L/usr/arm-linux-gnueabihf/sys-root/lib -
    lz"
```

Une fois le fichier enregistré, compilez (en lui disant d'utiliser le fichier tout juste créé) puis installez la *glib* (ce qui peut prendre un peu de temps...) :

```
$ ../glib-2.46.2/configure --host=arm-linux-gnueabihf
    --prefix=/usr/arm-linux-gnueabihf/sys-root/ --
    cache-file=./arm.cache
$ make
$ sudo make install
```

Vous pouvez ensuite supprimer toutes les données téléchargées (y compris le contenu du répertoire *build*, puisque les fichiers ont été installés).

Pour finir, il faut configurer *pkg-config*. Pour cela, commencez par éditer le fichier *glib-2.0.pc* pour modifier sa description (ce n'est pas obligatoire, mais je vous le conseil pour ne pas le confondre par la suite) :

```
$ cd /usr/arm-linux-gnueabihf/sys-root/lib/pkgconfig
$ sudo gedit glib-2.0.pc
```

Modifiez librement les lignes commençant par *Name :* et *Description :* pour expliquer qu'il s'agit de la bibliothèque pour arm (c'est ces descriptions que *pkg-config* vous montrera par la suite).

Ensuite, copiez ce fichier dans le répertoire de recherche de pkg-config :

```
$ sudo mv armglib-2.0.pc /usr/share/pkgconfig/armglib
    -2.0.pc
```

Et voilà, c'est terminé ! Pour vérifier que cela à fonctionner, tapez :

```
$ pkg-config --list-all | grep arm
```

Ceci devrait faire apparaitre la bibliothèque *armglib-2.0*, avec la description que vous avez entrée (si ce n'est pas le cas c'est que *pkg-config* utilise un répertoire différent, cherchez où se trouve les autres fichiers *.pc* de votre système).