Cross-compilation environment setup.

Matthieu Vigne

May 27, 2018

Contents

1 Introduction

This guide explains how to setup a cross-compilation environment on a Linux PC (Fedora or Debian/Ubunu) to compile our code, i.e. the BBBEurobot library. This setup works for compiling code to a Beaglebone Black, and to a Raspberry Pi as well. Three things are needed to cross-compile this code:

- a C compiler targeting the *armhf* platform, called **gcc-arm-linux-gnueabihf**.

- a cross-compiled version of the glib, used by our code.

- a renamed *pkg-config* file to load the cross-compiled glib when compiling.

2 Debian setup

These instructions were tested on Debian 8 (Jessie) in 2015. The setup for Ubuntu is probably quite similar.

Setup on Debian is quite fast, as the official repo already have the compiler and a cross-compiled glib. The only step needed is to tell *apt* to add the arm platform, then install the packages as usual:

```
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
```

Now everything is installed, but there is one last setup step needed: link to the *pkg-config* file of the glib. Since it has the same name as the one installed for your system library version, we will rename it to prevent any conflict. To keep things clear, I advise you modify (freely) the *Name:* and *Description:* fields of this file, */usr/lib/arm-linux-gnueabihf/pkgconfig/glib-2.0.pc* though for instance the following command:

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

Then, simply copy this file to the default *pkg-config* search path, renaming it:

```
sudo mv /usr/lib/arm-linux-gnueabihf/pkgconfig/glib-2.0.pc /
    usr/share/pkgconfig/armglib-2.0.pc
```

To check that everything works correctly, type:

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

This should give you the *armglib-2.0* library, with the name and description you previously entered.

3 Fedora setup

These instructions were tested in Febuary 2018 on Fedora 27.

Contrary to Debian, the official Fedora repo do not contain either the compiler nor the cross-compiled glib. We will get the compiler from an unofficial Copr repo, and then cross-compile the glib from source.

First, install the compiler and cross-compiled glibc (for basic C functions: this is a dependency of glib) using:

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

In order to compile the glib, we must first install its dependencies.

The firs is libffi: get the latest version from https://sourceware.org/libffi/ (v 3.2.1 in Febuary 2018), decompress it, then run from the source folder:

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

The first line configures the build, telling the system to cross-compile the library using the cross-compiler instead of system gcc, and to install it in */usr/arm-linux-gnueabihf/sys-root/*, the path used on install of the compiler and glibc. Then we compile and install the library.

Note: for faster compilation, you can append -jN *to the make instruction, where N is the number of process being run in parallel.*

The second dependancy is the zlib (http://www.zlib.net/, v 1.2.11 in Febuary 2018), installed with the same process:

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

Finally, download the glib http://ftp.gnome.org/pub/GNOME/sources/glib/ (v 2.55.1 in Febuary 2018). Go to the parent of the source directory (i.e. where you uncompressed the archive), create a build folder, in which we place a config file *arm.cache*:

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

Copy the following content in *arm.cache*:

Listing 1: File 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"
```

Now, configure the build using the given *arm.cache*:

```
../glib-2.55.1/configure --host=arm-linux-gnueabihf --prefix
    =/usr/arm-linux-gnueabihf/sys-root/ --cache-file=./arm.
    cache --with-pcre --enable-libmount=no
make
sudo make install
```

Comments: the option –with-pcre tells the configuration script to use internally-package pcre library. –enable-libmount=no disables libmount, a library used for disk mounting that we won't be using on the Beaglebone, and hence that wasn't installed (it can otherwise be installed using the same process as before).

Now everything is installed, but there is one last setup step needed: link to the *pkg-config* file of the glib. Since it has the same name as the one installed for your system library version, we will rename it to prevent any conflict. To keep things clear, I advise you modify (freely) the

Name: and *Description:* fields of this file, */usr/arm-linux-gnueabihf/sys-root/lib/pkgconfig/glib-2.0.pc* though for instance the following command:

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

Then, simply copy this file to the default *pkg-config* search path, renaming it:

```
sudo mv /usr/arm-linux-gnueabihf/sys-root/lib/pkgconfig/glib
    -2.0.pc /usr/share/pkgconfig/armglib-2.0.pc
```

To check that everything works correctly, type:

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

This should give you the *armglib-2.0* library, with the name and description you previously entered.