HERTAVILLE

Welcome to Hertaville!

Development Environment for the Raspberry Pi using a Cross Compiling Toolchain and Eclipse (../../../../2012/09/28/development-environment-raspberry-pi-cross-compiler/)

Date | 📅 Fri 28 September 2012 | Tags | Raspberry Pi (../../../../tag/raspberry-pi/) / GCC (../../../../tag/gcc/) / Cross compilation (../../../../tag/cross-compilation/) / C++ (../../../../tag/c/) / Eclipse (../../../../tag/eclipse/)

UPDATED September 7th 2014.

In this blog entry the setup of a cross-compiling development environment for the Raspberry Pi will be demonstrated. This will include the

- Official Raspbian (armhf) cross compiling toolchain (https://github.com/raspberrypi/tools)(available from github)
- Eclipse for C/C++ Developers (Linux) (http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplerr)

We will finally write a simple Hello World program on our development PC, compile it using the cross compiler and then deploy it onto our Raspberry Pi board to run it.

I'm going to assume that you have already installed a Raspbian Wheezy image on your RPi board (../../../../2012/09/27/raspbian-raspberry-pi/) and that you have Linux installed on your desktop PC. For this tutorial I am using the Crunchbang 11 Linux OS (http://crunchbang.org/) (64-bit) on my PC. The instructions provided should work on most Debian/Ubuntu based Linux distributions running directly on a PC or as a a guest operating system via VMWare/ VirtualBox (../../../../2012/09/27/kubuntu-vmware-player/) .

A remote debugging tutorial; which I consider to be the continuation of this tutorial, can be found here (../../../../2013/01/11/remote-debugging/).

Finally, Derek Molloy has a great video (http://www.youtube.com/watch?v=vFv_-ykLppo&feature=share&list=PLF4A1A7E09E5E260A) tutorial on setting up a similar environment for the Beaglebone. Watching this video was incredibly informative and helped me set up this tutorial.

So what is a cross compiling toolchain and why use one ?

A native compiler such as the default gcc tool on the PC is a compiler that runs on an Intel machine, as well as creates binaries intended to be run on an Intel machine. i.e it creates binaries for the same type of machine that it runs on. Similarly the GCC tool in the RPi's Raspbian Linux OS is intended to run on an ARM machine as well as creates binaries for an ARM machine.

A cross compiler such as the "arm-linux-gnueabihf-gcc" that we will use, is able to run on an Intel machine but creates binaries for an ARM machine. In other words, it runs on one architecture and creates binaries for another. This allows us to develop and compile our programs/libraries on our Desktop PC, but when it comes to deploying the binaries/libraries we deploy them and run them on the Raspberry Pi.

So why use a cross-compiler instead of developing our code and compiling it natively on the Raspberry Pi itself? After all, the Raspberry Pi has a native GCC compiler. We can also use code editors such as nano or vi from the command line (remotely over SSH) or GUI programs such as Geany (remotely over VNC).

🏠 Categories
(../../../../categories.html)

↗ Links

The main reason to use cross-compilation over native compilation (develop and compile on the RPi itself) is to speed up compile/build time. Remember the RPi board may be fast compared to a microcontroller...but its still has limited RAM resources and is pretty slow compared to an average desktop computer....

Also you have a myriad of development tools that you can use on your desktop PC that you simply can't use on the Raspberry Pi; such as the Eclipse IDE.

Cross compilation of simple applications that only require the use of the standard C/C++ libraries included with the cross compiling toolchain is relatively straightforward. Cross compilation of more complicated applications that require external libraries however (such as libjpeg, GTK+, Qt4/5 e.t.c), is typically more complicated to setup and is out of the scope of this tutorial.

- To learn more about cross-compiling GTK+ applications for the Raspberry Pi, take a look at this blog entry (../../../../2013/07/19/cross-compiling-gtk-applications-for-the-raspberry-pi/).
- To learn more about cross-compiling QT4 applications for the Raspberry Pi, take a look at this blog entry (../../../../2014/04/12/cross-compiling-qt4-app/).

Downloading and Setting Up the Cross Compiling Toolchain
- Open a terminal window on your desktop Linux OS and type the following command:

```
sudo apt-get install build-essential git
```

This will install the native build tools on your PC along with the GIT tool which will be used to download / clone the cross compiling toolchain from GitHub.com.

- Create a "rpi" directory in your home directory by typing "

```
mkdir rpi
```

while in your home directory.

- Go to the rpi directory with:

```
cd rpi
```

- and then type:

```
git clone git://github.com/raspberrypi/tools.git
```

- This command will download (clone) Raspbian's official cross compiling toolchain from Github. The command will take a few minutes to complete its task.
- When the previous command completes, navigate to "/home/halherta/rpi/tools/arm-bcm2708":

```
cd ~/rpi/tools/arm-bcm2708
```

on your deskop Linux OS. Note that My username is "halherta" and therefore my home folder is "/home/halherta". Please substitute your username with mine where necessary.

- In the arm-bcm2708 folder you'll see four other folders, each containing a separate toolchain:

1. arm-bcm2708-linux-gnueabi
2. arm-bcm2708hardfp-linux-gnueabi
3. gcc-linaro-arm-linux-gnueabihf-raspbian
4. gcc-linaro-arm-linux-gnueabihf-raspbian-x64

If you are running a 32-bit Linux operating system, you'll want to use the third toolchain 'gcc-linaro-arm-linux-gnueabihf-raspbian'. If however you are running a 64-bit Linux operating system, you'll want to use the fourth toolchain 'gcc-linaro-arm-linux-gnueabihf-raspbian-x64. If you don't know whether you are running 32-bit or a 64-bit version of Linux type in the command line :

```
uname -a
```

If you see something like 'i386'/'386' in the output string, then you're running a 32-bit version of Linux. If however you see 'x86_64' / 'amd64' in the output string, then you're running a 64-bit version of Linux.

The next step is to add the directory containing the binary files of the toolchain to the PATH environment variable in Linux. This way we can access the toolchain's binary files/tools from the command line. This step is not necessary if you want to using the toolchain strictly from Eclipse. I still do highly recommend that you do it, in case you decide to compile/debug arm applications directly from the command line later on. We will do this by adding an "export PATH" command to the bottom of the .bashrc files in the home directory.

- In a terminal window (on your Desktop Linux OS) type:

```
cd ~/
```

to point to the home directory, then type:

```
nano .bashrc
```

This will open the .bashrc file in a command line based editor called nano. Go to the bottom of the .bashrc file using the down arrow key. Then if you're running 32-bit Linux, type the following command:

```
export PATH=$PATH:$HOME/rpi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin
```

Alternatively if you're running 64-bit Linux, type the following command instead:

```
export PATH=$PATH:$HOME/rpi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin
```

- Then hit Ctrl+x to exit. You will be prompted to save changes. Say yes and hit "Enter".
- Then type in the command line:

```
source .bashrc
```

This allows the current open terminal to see the updated PATH variable. Alternatively one can close the terminal and open a new one.

- To test if you can access the cross compiling toolchain from the command line type:

```
arm-linux-gnueabihf-gcc -v
```

If you were successful, you should see output similar to that in Figure 1. Congratulations! you just installed Raspbian's official cross compiling toolchain on your Desktop PC / Virtual machine!

Figure 1. Cross Compiling Toolchain successfully installed and accessible from anywhere within your Desktop side Linux OS!!!

(files/uploads/2012/09/cc1.png)

Figure 1. Cross Compiling Toolchain successfully installed and accessible from anywhere within your Desktop side Linux OS!!!

Downloading and Setting Up Eclipse

The Next step is to download and install Eclipse. Unfortunately as of this writing the apt-get repositories rarely contain the latest Eclipse build.. So we will not be using the apt-get package manager. Instead we will download the latest Eclipse IDE from the web using a web browser.

- Before we can run Eclipse, we need to ensure that we have a Java runtime environment installed; since Eclipse relies heavily on Java. We can do this with the following command:

```
sudo apt-get install openjdk-7-jre
```

- Go to the link provided below and download the latest linux version of the Eclipse IDE for C/C++. Download the 32-bit version of the IDE if you're running a 32-bit Linux OS or download the 64-bit version of the IDE if you're running a 64-bit Linux OS.

Link: http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplerr (http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplerr)

- The next step is to extract the "eclipse" folder into our home directory from the eclipse-cpp-kepler-R-linux-gtk-x86_64.tar.gz compressed file. This can be easily done using File Manager and Archive Manager. Alternatively to perform the extraction from the command line, open a new console and navigate to the directory containing the .gz file (typically "~/Downloads/")then type:

```
tar -xvzf eclipse-cpp-kepler-R-linux-gtk-x86_64.tar.gz -C ~/
```

Either way you will end up having the eclipse directory created in your home directory.

- To start eclipse, go to the home directory:

```
cd ~/
```

then 'cd' your way into the eclipse folder

```
cd eclipse
```

then type.

```
./eclipse &
```

Alternatively you could simply type in the terminal:

```
~/eclipse/eclipse &
```

Creating a New Project in Eclipse

When Eclipse runs, it will first launch the workspace launcher. You will be prompted to choose a workspace directory. Accept the default (/home/halherta/workspace). Feel free to check the "Use this as the default and do not ask again" option if you feel inclined to do so. (You can always switch/change the workspace directory by going to "File->Switch Workspace").

- If you see a Welcome tab, click on the 'x' on the Welcome tab to close it.
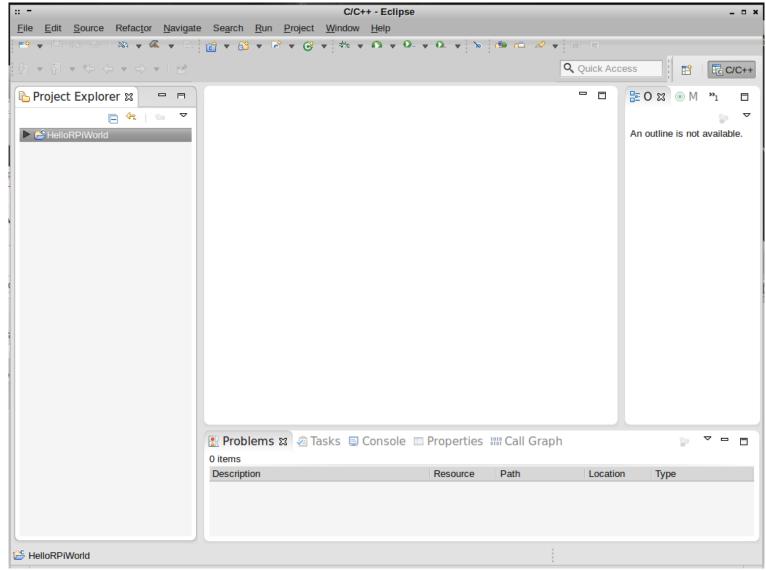- Now go to File->New->C++ Project. This will open the "C++ Project" window shown in Figure 2.

Figure 3. C++ Project Window **(files/uploads/2012/09/cc3.png)**

Figure 2. C++ Project Window

- Type "HelloRpiWorld" in the "Project Name" field. Under Project type select "Executable->Empty Project" Under "Toolchains" select "Cross GCC". Then click on "Next" This should take you to the "Select Configurations" Window. Accept the defaults and click "Next".
- This should then take you to the "Cross GCC command" Window (Figure 3).
  - In the "Cross compiler prefix" field type: "arm-linux-gnueabihf-".
  - In the "Cross compiler path" field type the full path to the toolchain: "/home/halherta/rpi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin/"
  - Click Finish. This step informs Eclipse of the location of the cross compiling toolchain and its standard C/C++ libraries and headers.

Figure 4 Configure the Cross GCC path and prefix **(files/uploads/2012/09/cc4.png)**

Figure 3. Configure the Cross GCC path and prefix

(files/uploads/2012/09/cc5.png)

Figure 4. New Hello World Project initial setup[/caption]

- Notice how this project does not have any source files. So lets add one. Click on "File->New Source File". This should open the "New Source File" window.
- The "Source folder" field should already have the project name "HelloRPiWorld" displayed in it. In the "Source file" field type "HelloRPiWorld.cpp" and click Finish. This will add a HelloRPiWorld.cpp source file to our project. In

the source file copy the following C++ code:

```
1   /*
2    * HelloRPiWorld.cpp
3    */
4
5   #include <iostream>
6
7   using namespace std;
8
9   int main (void)
10  {
11      cout << "Hello RPi Development World !"<< endl;
12      return 0;
13  }
```

- Save your project by clicking on "File->Save".

To build the project click on "Project->Build Project". To clean the project click on "Project->Clean". Take note of the various options under the "Clean dialog box" that enable multiple projects to be clean at once as well as automatic rebuild. The output of the compilation/build process should be available for viewing in the "console" tab. Figure 5 shows the Eclipse workspace with the HelloWorld program successfully cross compiled for the Raspberry Pi!!!

Figure 6. HelloWorld program successfully compiled for Raspberry Pi.!!! **(files/uploads/2012/09/cc6.png)**

Figure 5. HelloWorld program successfully compiled for Raspberry Pi.!!!

The binary file for our HelloWorld application resides in the "~/workspace/HelloRPiWorld/Debug". Lets go there via a console window with cd:

```
cd ~/workspace/HelloRPiWorld/Debug/
```

Type

```
ls -al
```

The "HelloRPiWorld" Binary file is highlighted in green. The binary can be executed as follows

```
./HelloRPiWorld
```

but should output the followingerror message "**cannot execute binary file**". Now type:

```
file HelloRPiWorld
```

You should get the following output:

```
HelloRPiWorld: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses sha
```

This indicates that the HelloRPiWorld Binary File was compiled for a 32-bit ARM machine and cannot run on an Intel Machine.

Deploying the Binary file onto the Raspberry Pi

To deploy the binary on the Raspberry Pi, make sure that the RPi board is powered and connected to your network. The next step is click on "Window->Show View->Other". This will open a show view window. In this window select the "Remote Systems" folder and then select "Remote Systems Details" and press OK (Figure 6).

Figure 7. Remote system view (files/uploads/2012/09/cc7.png)

Figure 6. Remote system view

- If the "Remote Systems Details" shows up in the bottom of the workspace, drag it such that its tab shares the same region as the project explorer tab as shown in Figure 7.

Figure 8. Creating a New Connection (files/uploads/2012/09/cc8.png)

Figure 7. Creating a New Connection

- Now right click in the "Remote Systems Detail" area and click on "New Connection". In the "Select Remote System Type" window (Figure 8), select "SSH only" then click "Next"

Figure 9. Select remote system type (files/uploads/2012/09/cc9.png)

Figure 8. Select remote system type

Figure 10. Remote SSH only system connection (files/uploads/2012/09/cc10.png)

Figure 9. Remote SSH only system connection[/caption]

- At this point a second device (Raspberry Pi) shows up in the "Remote Systems Detail" tab (Figure 10). Right click on the Raspberry Pi resource and click connect. You will be prompted to enter the username and password. Make sure that you utilize "username:pi" and "password:raspberry" (if you haven't changed the default password). You may get a warning windows asking you if you are sure that you want to accept this connection. Affirm that you want to proceed with the connection. At this point you should be connected to your RPi from Eclipse.

Figure 11. Starting an SSH terminal in Eclipse (files/uploads/2012/09/cc11.png)

Figure 10. Starting an SSH terminal in Eclipse

- Right click on the Raspberry Pi resource again in the "Remote Systems Detail" tab and click on the "Show in Remote Systems View". This will open a "Remote Systems" tab in the same region of the workspace. In this tab, one can navigate the root file systems of both the Local Linux OS and that of the Raspbian/Raspberry Pi.
- Locate the HelloWorld Binary file on our local desktop PC and drag it to the home directory on the Raspberry Pi (You could also right click on the binary and click on "Copy" and then right click on the home folder in the Raspberry Pi and click "Paste". This effectively copies the binary file to the home directory of the Raspberry Pi.

Figure 12. Launch Terminal (files/uploads/2012/09/cc12.png)

Figure 11. Launch Terminal

- Now right click on the SSH terminal icon under the Raspberry Pi icon (Figure 11) in the "Remote Systems" and select "Launch Terminal" . This should launch a terminal window in the bottom of the Eclipse workspace (Figure 12). This is basically a console / terminal window connected to the Raspberry Pi via SSH.

Figure 13. Running the HelloRPiWorld binary on the Raspberry Pi via an eclipse's SSH terminal

(files/uploads/2012/09/cc13.png) Figure 12. Running the HelloRPiWorld binary on the Raspberry Pi via an eclipse's SSH terminal

- In the terminal window type

```
ls -al
```

to confirm that the "HelloWorld" binary indeed got copied into the home directory of the Raspberry Pi's root file system. Then change the "HelloWorld" binary's permission to make it executable:

```
chmod +x HelloWorld
```

- Finally type

```
./HelloWorld
```

You should see the output shown in Figure 12 :**"Hello RPi Development World !"**

- Congratulations!!! You just developed and ran your first cross compiled application for the Raspberry Pi.!!!!
- To remotely debug your applications on the Raspberry Pi from Eclipse, check out this blog entry! (../../../../2013/01/11/remote-debugging/)