How to Create a “Hello World” Program for the Hammer

There are five basic steps to creating a “hello world” program for the Hammer:

1) Build a cross-compiler tool chain that will generate ARM output
2) Create the “hello world” c source file
3) Compile the “hello world” program
4) Transfer the “hello world” program to the Hammer
5) Execute the “hello world” program

Step 1 : Create An ARM Cross-Compiler Toolchain

First you will need a compiler to generate the “hello world” program. The Hammer has limited memory resources so it is not advisable[1] to run a full size compiler and linker directly on the Hammer. Therefore, you will use a cross-compiler toolchain. This cross-compiler toolchain is a set of programs (e.g., compiler, linker, libraries) that run on a Linux PC and generate ARM compatible output files that run on the Hammer target platform.

To create an ARM cross-compiler tool chain use the GNU Compiler Collection (GCC) compiler tools and Buildroot (http://buildroot.uclibc.org/). Buildroot is a set of scripts that allow you to easily generate both the cross-compiler tool chain and a root file system[2] for the Hammer target platform. The cross compiler tool chain uses uClibc (http://www.uclibc.org/), an alternative library for C applications. It is much smaller than the GNU C Library (http://www.gnu.org/software/libc/libc.html)[3], but nearly all applications supported by glibc can also work perfectly with uClibc.

This tutorial uses Ubuntu 8.04 (Hardy Heron)[4]. All commands are executed from the command line.[5][6]

[1] While it is certainly possible for you to set up the Hammer as a native build environment, doing so requires an in depth knowledge of Linux, and is still extremely experimental.
[2] A root file system is a collection of files and folders that the Linux system (in this case the Hammer) needs to boot.
[3] The GNU C Library is a basic set of functions and definitions for programs written in C. This library (or a derivative like uClibc) is found on all Linux systems.
[4] Linux comes in many different distributions. Ubuntu is one of those many different versions.
[5] The text in between the dashed lines are terminal input. You might have to consult your distribution’s manual for how to open a terminal or command line. Type the commands one line at a time, followed by the <Enter> key.
[6] The commands listed assume that you are running an account that is in the “wheel” group. This special group is needed to allow users to change system level settings. If you followed you distribution’s install instructions, it is likely that you are already part of this group.

Install the compilers and tools by entering the following commands:

```
------------------------------------------------------------------------------------------------------------
sudo apt-get[7] install build-essential libncurses-dev bison flex texinfo zlib1g-dev gettext libssl-dev wget
------------------------------------------------------------------------------------------------------------
```

Create a directory called hammer in your home directory.

```
------------------------------------------------------------------------------------------------------------
cd ~/
mkdir hammer
cd hammer
------------------------------------------------------------------------------------------------------------
```

Download[8] the latest files from the Hammer_Board_Software page located on the www.elinux.org wiki.
This assumes that you're still in the hammer directory.

```
------------------------------------------------------------------------------------------------------------
wget http://www.elinux.org/upload/b/b6/Buildroot-01082008.tar.gz
wget http://www.elinux.org/images/f/ff/Hammer-linux-2.6.22-09122007.diff.gz
wget http://www.elinux.org/images/e/ef/Hammer-kernel-config
wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.22.tar.gz
------------------------------------------------------------------------------------------------------------
```

Start the buildroot process. Be patient, the build process can take 30 – 45 minutes or more depending on the speed of your PC and your internet connection.

```
------------------------------------------------------------------------------------------------------------
tar xvzf Buildroot-01082008.tar.gz
cd buildroot
cp Hammer-config .config
make clean
make oldconfig
make
cd ..
------------------------------------------------------------------------------------------------------------
```

You will probably encounter the following (similar) compile error. The error message is listed below:

[7] apt-get is a distribution specific package manager. If this command does not work, then reference your particular distribution's manual for installing software.
[8] The following files are also found on the CD shipped with your Hammer_Board under the hammer/source directory. If you are not connected to the internet, then copy the files from the Hammer CD to the ~/hammer directory on your computer. Keep in mind that the file name is everything after the last forward slash.

```
make[1]: Leaving directory `/home/hammer/buildroot/project_build_arm/Hammer/busybox-1.6.1'
# Just in case
chmod a+x /home/hammer/buildroot/project_build_arm/Hammer/root/usr/share/udhcpc/default.script
wget --passive-ftp  -P /home/hammer/buildroot/dl
http://ftp.debian.org/debian/pool/main/f/fakeroot/fakeroot_1.7.1.tar.gz
--20:33:12--  http://ftp.debian.org/debian/pool/main/f/fakeroot/fakeroot_1.7.1.tar.gz
           => `/home/hammer/buildroot/dl/fakeroot_1.7.1.tar.gz'
Resolving ftp.debian.org... 128.101.240.212
Connecting to ftp.debian.org|128.101.240.212|:80... connected.
HTTP request sent, awaiting response... 404 Not Found
20:33:13 ERROR 404: Not Found.

make: *** [/home//hammer/buildroot/dl/fakeroot_1.7.1.tar.gz] Error 1
```

The error listed above is generated because the file: fakeroot_1.7.1.tar.gz cannot be found. The version number on the fakeroot_1.7.1.tar.gz file has changed. For example, if the current version of the fakeroot file is 1.9.5[9] (i.e. the filename is fakeroot_1.9.5.tar.gz) you simply do the following:

Open a browser and go to the download location: http://ftp.debian.org/debian/pool/main/f/fakeroot .
Get the current version number of the file: fakeroot_x.x.x.tar.gz.[10]

Edit the file fakeroot.mk located at ~/hammer/buildroot/package/fakeroot . Change the version number from 1.7.1 to 1.9.5. as shown below. Be sure to check for the **current** version number because the fakeroot file is always being updated.

```
gedit ~/hammer/buildroot/package/fakeroot.mk
```

```
#############################################################
#
# fakeroot
#
#############################################################
FAKEROOT_VERSION:=1.9.5
FAKEROOT_SOURCE:=fakeroot_$(FAKEROOT_VERSION).tar.gz
FAKEROOT_SITE:=http://ftp.debian.org/debian/pool/main/f/fakeroot
[…]
```

[9] 1.9.5 will be the version referenced for the rest of this example. If your version is higher, substitute the correct version number.
[10] Replace the x's with the largest numbers you can find on the webpage, starting with the leftmost x and working right.

Save the updated file, and exit the editor.
Then continue with the make command…

```
cd ~/hammer/buildroot
make
```

Once the command has completed (with no errors), go to the following directory: ~/hammer/buildroot/binaries/Hammer.

```
cd ~/hammer/buildroot/binaries/Hammer
```

There should be two files in this directory[11]:
rootfs.arm.ext2
rootfs.arm.ext2.gz

If both of these files are present, it is a good indication that the buildroot process has been successful!

Add the cross compiler tools into your path. The best way to do this is by modifying the file: ~/.bashrc.
Run the following command to open a text editor:

```
gedit ~/.bashrc
```

Add the following line to the very end of the ./bashrc file:

```
export PATH=$PATH:/home/hammer/buildroot/build_arm/staging_dir/bin
```

Save the file, and exit the text editor. The PATH environment variable will not be updated until the next time you log in. To ensure the PATH gets updated, log off the terminal you are using, and open a new one.

Type the following command:

```
exit
```

[11] Run the command `ls` and the two file names listed should be shown on the screen.

This will close the current terminal window. Now open another terminal window. The PATH variable should now include the path to the cross compiler tools.

To verify this, run the following command:

```
-----------------------------------------------------------------------------
echo $PATH
-----------------------------------------------------------------------------
```

Linux will display a list of all of the paths included in the PATH environmental variable. The path to the cross compiler tools:

/home/hammer/buildroot/build_arm/staging_dir/bin

should also be included in the list.

Step 2 : Create the “Hello world” C Source File

Use a text editor to make the hello.c program:

```
----------------------------------------------------------------------------------------------------
cd ~/hammer
gedit hello.c
----------------------------------------------------------------------------------------------------
```

Create the hello.c file as shown below:

```
----------------------------------------------------------------------------------------------------
#include <stdio.h>
main ()
{
  printf("Hello world\n");
}
----------------------------------------------------------------------------------------------------
```

Save the file, and exit the text editor.

Step 3 : Compile the "Hello world" program

To execute the cross-compiler, issue the following command:

```
----------------------------------------------------------------------------------------
arm-linux-uclibc-gcc hello.c -o hello
----------------------------------------------------------------------------------------
```

FYI: The structure of the command is:

```
---- designates the platform
|
|    ---- designates the OS
|    |
|    |     ---- designates which libc library should be used
|    |     |    (uclibc is a small embedded version of libc - saves memory space)
|    |     |
|    |     |    ---- designates the compiler
|    |     |    |
|    |     |    |    ---- source file(s)
|    |     |    |    |
arm-linux-uclibc-gcc hello.c -o hello
                              |   |
                              |   ---- output filename = hello
                              |
                              ---- use the -o option to create an output file with a user chosen name
```

To check the output file and make sure it is an ARM executable, type the following command:

```
----------------------------------------------------------------------------------------
file hello
----------------------------------------------------------------------------------------
```

The file command will return the following message:

```
hello: ELF 32-bit LSB executable, ARM, version 1 (ARM), dynamically linked (uses shared libs), not stripped
```

This confirms that the hello file is an ARM executable file and it indicates it is compatible with the Hammer.

Step 4 : Transfer the "Hello world" program to the Hammer

There are two methods to transfer the "hello" program from the Linux PC to the Hammer:
- Use a USB Flash Drive (USB Thumbdrive)

or

- Use the Hammer's RS-232 serial port and transfer the file using Xmodem protocol.

<u>USB Flash Drive</u>
Follow the steps below to use a USB Flash Drive to transfer files from the PC to the Hammer:

1) Plug a USB Flash Drive into the Linux PC.
2) Type the following commands on the Linux PC command line. This will copy the hello program to the USB Flash Drive.[12]

```
-------------------------------------------------------------------------------------------------------------
cd ~/hammer
cp hello /media/usbdisk
-------------------------------------------------------------------------------------------------------------
```

3) Eject and remove the USB Flash Drive from the Linux PC.
4) Plug the USB Flash Drive into the Hammer Carrier Board (USB host port - J4).[13]
5) Type the following commands on the **Hammer** command line:

```
-------------------------------------------------------------------------------------------------------------
mkdir /mnt/thumb
mount /dev/sda1 /mnt/thumb
cp /mnt/thumb/hello ~/
-------------------------------------------------------------------------------------------------------------
```

The "hello" program should now be located in the `~/` directory.

Now skip to the section labeled: **Step 5 : Execute the "Hello world" program**

[12] Different distributions handle USB media differently. If these commands do not work, please consult your distribution's manual for how to use USB flash media.
[13] The Hammer will delay two seconds before scanning the USB Flash Drive to let the Flash Drive hardware settle.

RS-232 Transfer via Xmodem protocol

Follow the steps below to use the Hammer's RS-232 serial port to transfer the hello program using Xmodem protocol:

1) Connect the RS-232 serial cable from the Hammer Carrier's DB9 connector (J5) to the Linux PC's serial port
2) We will install minicom (Linux terminal emulation software).
3) Type the following commands to install minicom and lrzsz (xmodem protocol software)

```
------------------------------------------------------------------------------------------
sudo apt-get install minicom
sudo apt-get install lrzsz
------------------------------------------------------------------------------------------
```

4) Configure minicom by typing the following command at the Linux PC command line:

```
------------------------------------------------------------------------------------------
sudo minicom -s
------------------------------------------------------------------------------------------
```

5) Select the **Serial Port Setup** from the minicom menu.
6) Configure the settings as shown below:

```
 A -    Serial Device      : /dev/ttyS0
 B - Lockfile Location     : /var/lock
 C -   Callin Program      :
 D -  Callout Program      :
 E -    Bps/Par/Bits       : 115200 8N1
 F - Hardware Flow Control : No
 G - Software Flow Control : No
```

7) Then select **SAVE as dfl** to save these settings as the default.
8) Select **Exit** and you will enter the minicom console.
9) From the Hammer command prompt, start the xmodem receive process by entering the following:

```
------------------------------------------------------------------------------------------
cd ~/
rx hello
------------------------------------------------------------------------------------------
```

The Hammer will display a "C" on the command line. This indicates that the rx receive program is waiting for the xmodem protocol from the Linux PC. The "C" will repeat while the Hammer is waiting for the proper xmodem protocol command.

10) Then type Ctrl+A followed by an S (w/o the Ctrl key). This selects the "Send files ..S" option from the minicom menu.[14]
11) Select **xmodem** from the pop-up menu.
12) Then arrow down and select the file you want to send. Select the file by putting the cursor in front of the filename and then pressing the "space bar". This highlights the filename. Then press "enter/return".
13) The file transfer begins, using the xmodem protocol. Minicom will display the status of the file transfer. You must press any key to return to the basic terminal mode within minicom.
14) The hello program file should have been successfully received by the Hammer. Check this by entering the following at the Hammer command line:

```
--------------------------------------------------------------------------------
ls
--------------------------------------------------------------------------------
```

The program file **hello** should be listed.

[14] Type Ctrl+A followed by Z to see all of the command options.

Step 5 : Execute the “Hello world” program

Before executing the hello program, you must change the file permissions to allow the program to execute.

1) From the Hammer command prompt, type the following:

```
chmod 750 hello
```

2) Now execute the program by entering the following at the Hammer command prompt:

```
./hello
```

The dot-slash (./) means ‘current directory’. Therefore, preceding a file name with the dot-slash (./) tells Linux that the application indicated resides in the current directory.

The hello program will run and display “Hello world”.

Congratulations!   You have completed your first Hammer program.

Up next, “Hello World” the embedded way: How to blink an LED on the Hammer module!

www.TinCanTools.com