ozzmaker.com

Linux, Raspberry Pi

HOW TO CROSS COMPILE THE KERNEL FOR THE RASPBERRY PI

November 26, 2012 | Mark Williams | 1 Comment

August 2015 ## This guide has been updated to the latest version of Raspbian ##

Compiling the kernel on the Raspberry Pi can take some time, Im not sure how long it takes as I have never waited long enough, I gave up after 4 hours.
Below is a guide on how to compile the kernel on a faster PC and then transfer the new kernel and modules over to the Raspberry Pi.

Cross compiling is when you compile on a different platform then what the kernel will be used for. This is mostly done when the host receiving the kernel is on a slow or legacy device.

In this guide, I will use Ubuntu 12.10 running within VMWare on a Windows 7 Quad core PC. Kernel compile time is 15 mins.

Network connectivity is needed between the VMWare host and the Raspberry Pi.

Please take note of what host the cammands are run on.
This is the Raspberry Pi;

```
pi@raspberrypi ~ $ pi@raspberrypi~$
```

This is the Ubunti VMWare host;

```
mark@ubuntu~$
```

Prepare the Raspberry Pi

-Download and install Raspbian "wheezy".
For this guide, I used 2015-05-05-wheezy-raspbian.img

-Configure your Pi. E.g. Memory split, overclocking, etc..

```
pi@raspberrypi ~ $ sudo raspi-config
```

-Update firmware and reboot.

```
pi@raspberrypi ~ $ sudo rpi-update
```

Prepare the sources and tools

-Install packages used for cross compiling on the Ubuntu box.

```
mark@ubuntu~$ sudo apt-get install gcc-arm-linux-gnueabi make git-core ncurses-dev
```

-Install toolchain.

```
mark@ubuntu~$ git clone https://github.com/raspberrypi/tools
```

-Add the toolchain to your path, this helps further on.

```
mark@ubuntu~$ PATH=$PATH:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian:~/tools/arm-
bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin
```

-Download the current source for your kernel onto the Raspberry. We will use 'rpi-source'.
More info here https://github.com/notro/rpi-source/wiki

```
pi@raspberrypi ~ $ sudo wget https://raw.githubusercontent.com/notro/rpi-source/master/rpi-source
-O /usr/bin/rpi-source && sudo chmod +x /usr/bin/rpi-source && /usr/bin/rpi-source -q -tag-update
pi@raspberrypi ~ $ rpi-source – -skip-gcc
```

Once finished, you will end up with a new folder and tar file in your home directory, these contain the source.
E.g **linux-df068289972288ecd46a02906df91358e1eca56b.tar.gz**

-We need to copy the tar file over to the Ubuntu box. Replace 'raspberrypi' below with the IP address of your Raspberry Pi if hostname lookup fails.

```
mark@ubuntu~$ sudo scp pi@raspberrypi:linux-df068289972288ecd46a02906df91358e1eca56b.tar.gz .
```

-Now extract the contents.

```
mark@ubuntu~$ tar xfv linux-df068289972288ecd46a02906df91358e1eca56b.tar.gz
```

-Grab the current config off the Raspberry Pi.

```
pi@raspberrypi ~ $ sudo zcat /proc/config.gz > config
```

-Copy the .config from the Raspberry Pi to the Ubuntu box using SCP. CD into the Linux source folder first.

```
mark@ubuntu~$ cd linux-df068289972288ecd46a02906df91358e1eca56b
mark@ubuntu~$ sudo scp pi@raspberrypi:config .
mark@ubuntu~$ mv config .config
```

Build Source and Compile Kernel

-CD into the Linux source on the Ubuntu box. And inter the following commands;
For P1 and compute module

```
mark@ubuntu~$ cd linux
mark@ubuntu~$ KERNEL=kernel
mark@ubuntu~$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcmrpi_defconfig
```

For P2

```
mark@ubuntu~$ cd linux
mark@ubuntu~$ KERNEL=kernel7
mark@ubuntu~$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig
```

-Now compile the kernel.
For all versions of Pi, use.

```
pi@ubuntu:~$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs
```

To enable parallel processing for a faster compile. If you have a dual core processor add **-j 3** to the end of the command below. If you have quad core, add **-j 6**

-Create modules on the Ubuntu box which we will copy over later.

```
pi@ubuntu:~$ mkdir ../modules
pi@ubuntu:~$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- INSTALL_MOD_PATH=../modules/
modules_install
```

-Create the new kernel image and copy it to your home folder ~

```
pi@ubuntu:~$../tools/mkimage/mkknlimg arch/arm/boot/zImage ~/kernel.img
```

-Copy the new kernel over to the Raspberry Pi and place it into the /boot directory.

```
pi@raspberrypi ~ $ sudo scp pi@ubuntu:kernel.img /boot/kernel.img
```

-Now copy over the new modules and device tree info. First remove the unneeded directories, otherwise you will be copying over 1.9GB

```
pi@ubuntu:$  rm -f -r ../modules/lib/modules/4.1.4/build
pi@ubuntu:$  rm -f -r ../modules/lib/modules/4.1.4/source
pi@raspberrypi ~ $ sudo rm -f -r /lib/modules
pi@raspberrypi ~ $ sudo rm -f -r /lib/firmware
pi@raspberrypi ~ $ sudo scp -r pi@ubuntu:/home/pi/modules/lib /
pi@raspberrypi ~ $ sudo scp -r pi@ubuntu:/home/pi/linux-
df068289972288ecd46a02906df91358e1eca56b/arch/arm/boot/dts/*.dtb /boot/
pi@raspberrypi ~ $ sudo scp -r pi@ubuntu:/home/pi/linux-
df068289972288ecd46a02906df91358e1eca56b/arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/
```

-Reboot.

COMPILE KERNEL LINUX RASPBERRY PI

ONE THOUGHT ON “HOW TO CROSS COMPILE THE KERNEL FOR THE RASPBERRY PI”

Pingback: noise is good | Howto: easy i2s audio with your Raspberry Pi