# Bake your Own Raspberry Pi Filling

In the previous issue we learnt a little about what it takes to build a custom Linux version for the Raspberry Pi and also configured our Ubuntu Linux desktop ready to compile software for the Raspberry Pi.

With that done we can now proceed to get the necessary build tools and start down the path of compiling a small Linux operating system. In this article we will create a bootable Linux SD card for our Raspberry Pi that will boot in around five to ten seconds.

The tool that we will use to help us make our software that will be copied onto the SD card is called BuildRoot. This tool not only looks after compilation of the various source code elements but also provides the ability to customise the software to our needs.

Before we begin, ensure that we are in the home directory:

**cd ~**

and then download the BuildRoot tool:

**wget**[**http://buildroot.uclibc.org/downloads/buildroot-2013.08.1.tar.bz2**](http://buildroot.uclibc.org/downloads/buildroot-2013.08.1.tar.bz2)

Next we need to unpack the download and change to the newly created directory that contains the Buildroot system. We do this with the following two commands.

**tar -xjf buildroot-2013.08.1.tar.bz2**

**cd buildroot-2013.08.1**

BuildRoot is capable of building Linux for many different types of hardware; we however are only interested in the Raspberry Pi, so load the default configuration for our board using the following command.

**make rpi\_defconfig**

With the default configuration for the Raspberry Pi done we also have the ability to further customise the system we are building. This step can be simple or complex depending on what additional pieces of software we want to be installed on the new system along with the level of other configuration each piece of software needs in order to work correctly.

Fortunately we have access to a menu system that makes the management of this process a little easier. This is started with the following command

**make menuconfig**

This brings up a menu that allows for control of almost every aspect of the new operating system. We will begin by customizing the message that is displayed with the login prompt.

Select ‘System configuration’ from the initial menu then select ‘System banner’. Delete the existing text and add in the text you would like to see, in my example I used “The Raspberry Pi Rocks”.

Next save this configuration by pressing TAB to highlight the ‘save’ option and press enter to accept all the defaults.

With the additional configuration completed, we are all set up to actually download and build the software ready for the Raspberry Pi. The BuildRoot tool does this by first working out the order of which the software is to be built, known as dependency checking. Then for each of the software elements it does the following:

1. Download the source code from the Internet
2. Download any patches (fixes)
3. Unzip the source code
4. Apply any patches found
5. Compile the software and once complete copy it to the correct location ready for use with the new operating system.

<<Insert Build.png Here>>

Note that the download process is done once, all subsequent builds that are done will skip the download as we have already downloaded the required packages. To start this process, use the following command.

**make**

This will take some time to complete as there is much to be done, depending on the speed of the PC you are using and the Internet connection, this could even take overnight to finish. Once complete, the final task is to copy the brand new system across to a SD Card.

The process for this begins by un-mounting the existing SD card partitions as Ubuntu always mounts them. Next we fill the very first part of the card with zeros wiping out the boot and partition information. Then the new partition table needs to be created and formatted. The Raspberry Pi requires that the first partition, commonly known as the boot partition, is a FAT disk format. Finally the newly created binary software that was just built is copied onto the card and the card is un-mounted.

A script provided with this edition of the magazine called ‘FlashRaspberryPiSDCard.sh’ makes this task easier. It is called with the name of the device where your SD card is plugged in. To use it, first copy it to the buildroot-2013.8 directory. Then make sure that the correct permissions are set on the file to allow execution as follows:

**chmod 0755 FlashRaspberryPiSDCard.sh**

Before we use this script, we first must discover the name of the SD card device. This is a really important step, since entering in the wrong disk device can lead to you destroying your Ubuntu Linux installation!

Before inserting the SD Card into your computer type the following

**df**

This will give a list of devices that are currently mounted on your computer. The result should look something like this.

**Filesystem 1K-blocks Used Available Use% Mounted on**

**/dev/sda1 236179500 3842064 220333532 2% /**

**none 4 0 4 0% /sys/fs/cgroup**

**udev 1949312 4 1949308 1% /dev**

**tmpfs 391772 792 390980 1% /run**

**none 5120 0 5120 0% /run/lock**

**none 1958852 152 1958700 1% /run/shm**

**none 102400 56 102344 1% /run/user**

**/dev/sdb1 240176648 4638336 223331364 3% /home**

Note down the devices starting with ***/dev*** in the first column of the list, in this case we have ***/dev/sda1*** and ***/dev/sdb1***. Un no circumstances should we use these names with this script; this is where your Linux system is running from.

Next, insert the sd card into the card reader and type the following.

**dmesg | tail**

You should see something like this on the screen

**[ 522.669182] sd 2:0:0:0: [sdc] 15523840 512-byte logical blocks: (7.94 GB/7.40 GiB)**

**[ 522.671422] sd 2:0:0:0: [sdc] No Caching mode page present**

**[ 522.671427] sd 2:0:0:0: [sdc] Assuming drive cache: write through**

**[ 522.675174] sd 2:0:0:0: [sdc] No Caching mode page present**

**[ 522.675178] sd 2:0:0:0: [sdc] Assuming drive cache: write through**

**[ 522.684320] sdc: sdc1 sdc2**

What we are looking for is the name of the device between the square brackets; in this case it is ***sdc***. This is the name of the device for which you will be writing the newly created Linux installation with the script file.

Now we have the name of the SD card we create the bootable Linux system we have created, I am using ***/dev/sdc*** as my SD card location, as this is what we discovered when we plugged the card in. Notice that we have prepended the name ***/dev/*** to the beginning of the device name as the script requires the full path name to the SD card.

**sudo ./FlashRaspberryPiSDCard.sh /dev/sdc**

Upon running the script, it will ask you if you really want to run this script. If you enter ‘y’ then the process continues and generates the card ready for use.

All that is left is to remove the card, plug it into your Raspberry Pi and plug in the power. When it has started, use the username of ‘root’ to log in.