# Bake Your Own Pi Filling

By now you’ve probably tried at least once operating system version on your Raspberry Pi. After issue 12’s article on the many variations of operating systems, perhaps you have tried many more. But what if you needed some customizations that are not available in the various images out there? Perhaps you want to build your very own version for a custom purpose such as a DHCP server. Or maybe you just want a greater understanding of what goes into creating one of the many software images out on the web.

This is the first in a series of articles where we start with the raw ingredients and finish with various customised Raspberry Pi images. Along the way we will explore the various options and what is happening under the covers.

Building Linux and all of the other components from source code is a big job and can take many hours; on a small computer like the Raspberry Pi this could turn into days. To reduce the amount of time this takes we will be using a desktop PC running Ubuntu Linux for the job. This machine has an Intel CPU instead of an ARM CPU and the compilers that it uses will only produce software that will run on an Intel system.

To get around this problem we need to use a compiler that is capable of creating software that runs on the ARM CPU. This compiler is known as a cross-compiler and is capable of creating executable code for a CPU architecture that is different to the one the compiler is running on. There are a number of pre-built options available, some free and others that attract a fee. As we are interested in doing as much as we can from the source code we will be building our own.

The next hurdle is how do we actually do the compilation and ensure that not only do we have all the components needed for running Linux but that they are also compiled in the correct order and any fixes to each source package are applied where appropriate. To make this easier there are a number of tools available both open source and commercial that performs this function and makes it easier to ensure that after the hours of building the software the result will run on the Pi. We will be using the open source *Buildroot* tool for compiling our custom images in this series of articles.

Before we can begin with creating our images, we need to install a distribution on Linux on our Desktop PC and install the compilers and other tools that are used during the build process to make our images.

We start with a system that already has Ubuntu Desktop installed. It does not matter if it is a 32-bit or 64-bit installation however it must be version 12.04.2, 12.10 or 13.04. If we are installing a fresh copy of Ubuntu onto a PC then version 13.04 is recommended. All the commands that we need to configure our desktop system will be done from a terminal window so open up a terminal window. Let’s get started.

It’s always a good idea to make sure we have a system that is up-to-date prior to making any of these changes so we will do this first with the following command.

**sudo apt-get upgrade**

Next we install the compilers and other associated tools. These will be used for the compilation of the tool chain used to create the executable binaries that the Raspberry Pi can run.

**sudo apt-get install build-essential**

Next we need to install a few of support libraries and components that will be used by the cross compilation tool.

**sudo apt-get install libncurses5-dev**

**sudo apt-get install bison flex texinfo gawk**

Finally we need components that provide the functionality to get the source code used to create our final operating system.

**sudo apt-get install git mercurial subversion**

In the next issue we will be using our newly configured Ubuntu system to compile and install a basic Linux system onto a SD card that we can use on our Raspberry Pi.