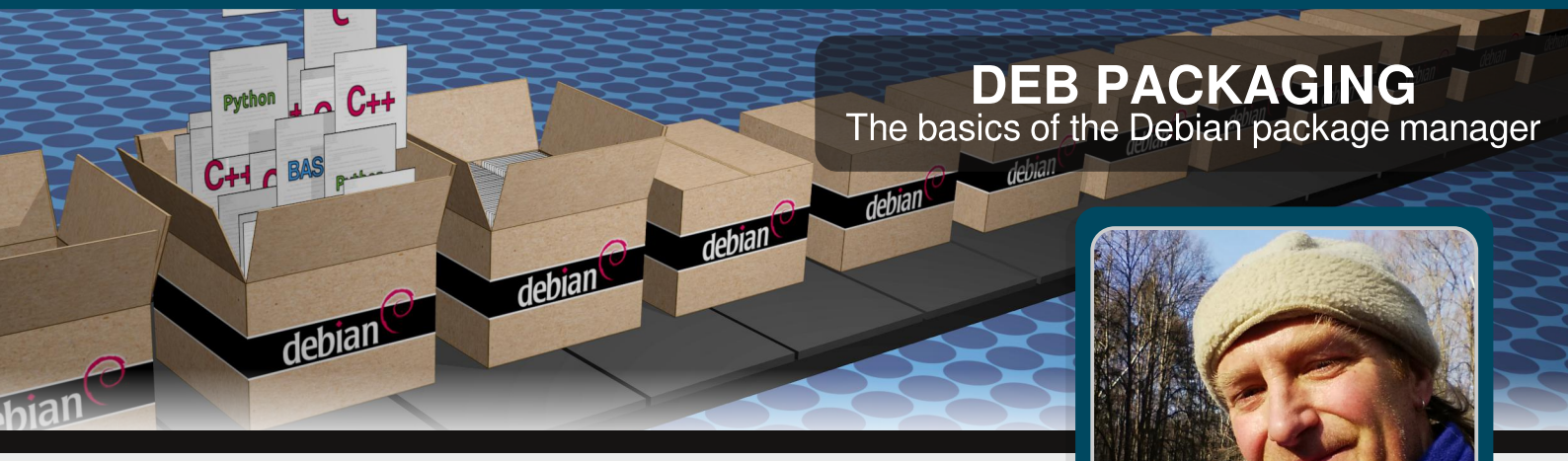

Packaging and distributing software for the Raspberry Pi

Hamish Cunningham
Guest Writer

SKILL LEVEL : ADVANCED

Packaging software is a vital part of software development. Packaging software correctly means that it is easier to install and configure, upgrade or remove. In this tutorial the Debian package manager is introduced through some selected examples.

Motivation

Software packaging and distribution has to deal with versions, dependencies and access.

Different versions of programs can incorporate lots of changes, including critical security fixes that need to be applied in a hurry. On older computer systems and when available network bandwidth is low, minimising the amount of new files that have to be downloaded is important.

Dependencies between programs are often tied to particular versions — if a facility that one program relies on changes, then the program's package data must reflect its need for the old version.

Access to packages is relatively easy now — most software is distributed via the Internet. However, what happens if the netword connection drops half-way through? Or if a million people all want updates on the same day?

These issues have been addressed in various ways; one of the most succesful and wide-reaching is the Debian package management system, and this is the one that is described in this tutorial. Debian packaging is used by the Raspbian operating system. The Raspbian operating system is an optimised rebuild of Debian Wheezy, which is optimised for use on the Raspberry Pi.

Debian packaging

This tutorial provides a quick-start set of instructions. For further reading, try consulting the official Debian documentation: https://wiki.debian.org/Packaging.

The examples given in this tutorial cover:
• scripts and other architecture-independent packages (including daemons, which are scripts run at system startup)
• binaries from compiled languages like C (including shared libraries)

In each case the key to the process is to:
1. create a `debian` directory in the top-level directory of your software tree, containing data to control the package construction process
2. set up a `Makefile` in the `debian` directory that follows certain conventions for installation targets