There's a new kid on the block

Yocto is the new kid in the embedded Linux world; it challenges traditional thinking and solves many embedded Linux problems. By **Joe Nicholson**.

Embedded Linux should, in theory, be a great platform for a smart device; it's powerful, feature rich and the source is not only available, it's also free! Historically, though, developing embedded Linux based products has often turned out to be surprisingly traumatic.

One reason for this trauma is for the sheer variety of choices and options: different development boards, modules and CPU platforms have resulted in a range of distributions that differ wildly in their capabilities. They use different build systems, kernel versions and library versions. Some aren't well maintained; worse still, some cannot even be extended.

Frequently, a developer choosing embedded Linux is faced with two choices: either bring in a Linux consultant to create a custom Linux distribution for their new product; or use the distribution that's provided by their chosen module or SoC platform.

But either choice can be problematic:

- A fully custom Linux distribution, once developed, will never update over time. You may need to get the consultant back in to apply security updates, add new libraries and packages and so on.
- Distributions supplied with development boards or modules can be of surprisingly low quality. In fact, some seem to be intended more as a demonstration than as a stable, robust platform for your next product.

Yocto is open source

Enter Yocto. Yocto is an open source 'standardised custom distribution builder' which aims to provide a stable, tested core Linux framework for multiple CPU platforms.

Yocto is based on Open Embedded, which is a mature embedded Linux build system first developed in 2003. Open Embedded offers a wide variety of 'recipes' that allow thousands of Linux packages to be cross built.

Development of Open Embedded, however, has been uncoordinated, resulting in a constantly changing pool of package recipes that differ wildly in both quality and maintenance. This means it has been difficult to use Open Embedded to create stable, polished Linux distributions.

UK Linux consultancy OpenedHand tried addressing this issue by creating 'Poky Linux', a cut down subset of Open Embedded. It has a regular release cycle (half yearly, similar to desktop Linux distributions such as Ubuntu) and each release of Poky is heavily tested and validated on a number of target hardware platforms.

In 2008, Intel purchased OpenedHand and took Poky Linux into the Linux Foundation as a collaborative project: the Yocto Project. Although Intel did much of the work to create the Yocto Project, Intel's approach has been to foster collaboration. Yocto works just as well on a Freescale ARM SoC as it does on an Intel SoC.

Current members of the Yocto Project include the major CPU vendors, such as Freescale, Intel, Texas Instruments, AMD and Broadcom, consumer electronics companies like LG, Dell and Huawei, software tools vendors such as Mentor Graphics, Wind River and Timesys, and distributors like Silica. Yocto's charter focuses on creating a single build system infrastructure to address the needs of those using Poky – or Open Embedded.

At last; some much needed stability and standardisation in the 'wild west' of embedded Linux!

Layering offers key advantages

Have you heard of layering before? Other Linux distribution build systems have claimed the 'single build infrastructure' throne, but Yocto offers a key advantage over the build systems that have gone before it – layering.

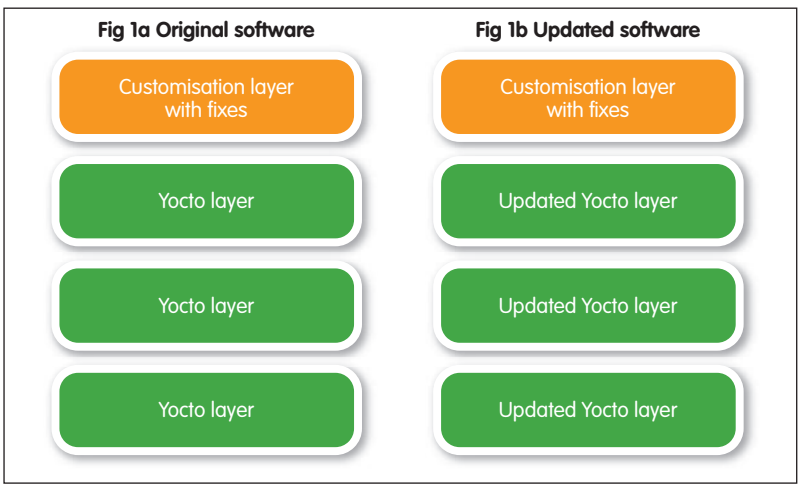
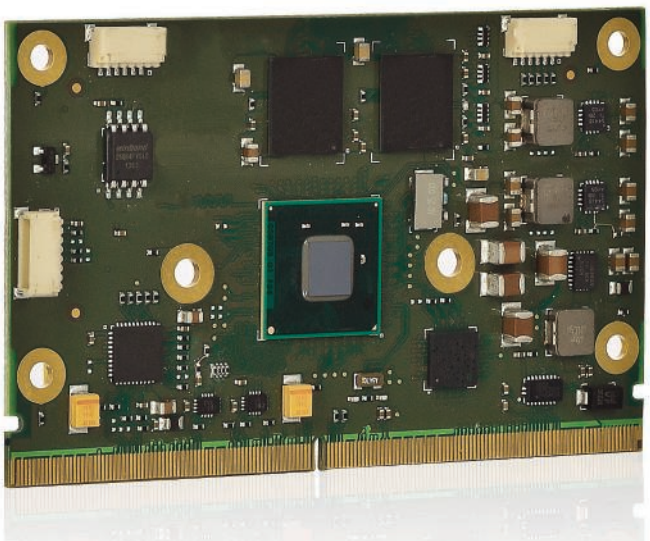


Fig 1: How layers help Yocto users to maintain their software



- *Without layers*

Let's look at the traditional approach to creating a custom embedded Linux product using a build system, such as buildroot or Open Embedded Classic.

First, you start with the basic build system, which builds a 'vanilla' distro. Next, you add in customisations, extensions and fixes for your particular device. Finally, after some work, you arrive at your fully customised Linux distribution.

However, as mentioned above, Linux isn't static. Security patches come out; sometimes these are important, such as fixes for the Shellshock Bash bug or OpenSSH. Meanwhile, bugs are corrected, libraries updated, new features added and new packages created.

While you need to keep your product up to date with these enhancements, a traditional build system means all your customisations and fixes have been made within the base build system itself. Updating to a new version requires making these changes afresh on a new version of the build system. Without significant work, you're effectively stuck.

- *With layers*

The Yocto system comprises of layers (see fig 1a). The core subset of Open Embedded sits in the lowest layer. Higher layers append or alter the package recipes within the lower layers, or add new software packages. To customise Yocto, you don't need to change the information in the layers themselves; instead you overlay your customisations in a fresh higher level layer.

This approach has huge advantages. When a new release of Yocto/Poky comes out, you no longer need to recreate your changes and modifications: you simply update the lower layers, then check over your customisation layer for validity post-update (see fig 1b).

"At last; some much needed stability and standardisation in the 'wild west' of embedded Linux!"
Joe Nicholson

Layering offers other advantages too. For example, layering makes it easier to move between CPU platforms, modules or development boards.

It's even possible to build recipes for your product's application in higher level layers. Your entire product software can therefore be built to a single output image by a single build system (see fig 2). Just like RTOS or bare metal development, one tool can build an entire Linux based product to a single output image that can be used as the master image.

Yocto has many more tricks up its sleeve. Here's a quick glimpse: Yocto allows a product to be extended or updated from a package feed (in the same manner as desktop Linux); Yocto also offers extensive facilities for monitoring and controlling the software licences used within your product.

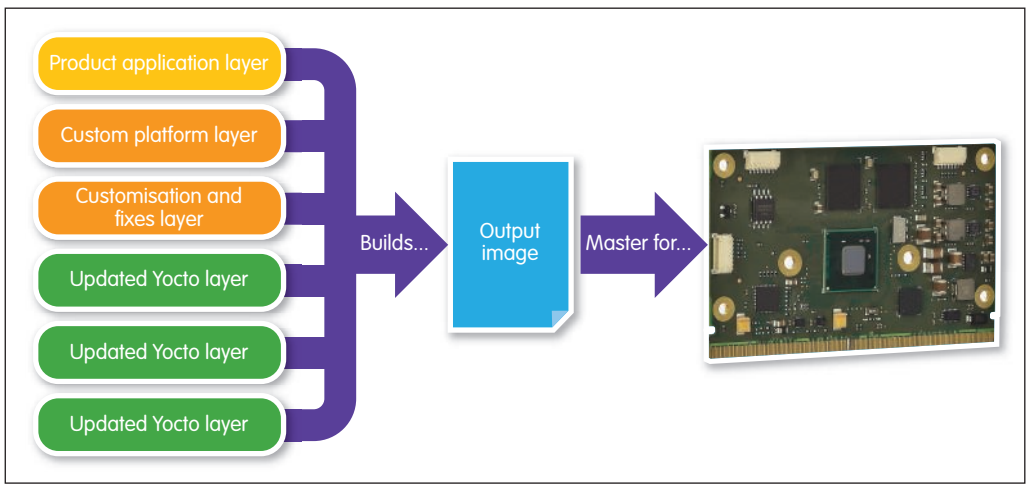


Fig 2: A single output image from a single build system

Disadvantages of Yocto

There's a price to pay for this power and the Yocto garden isn't entirely rosy.

Yocto can be hard to learn initially and builds can take many hours to complete on a mid priced PC, generating more than 50Gbyte of work data on the host system, rising to more than 100Gbyte for a fairly complex graphical Linux build. Lastly, module vendors moving to Yocto aren't always doing so in a way that allows products to be easily updated over time, thus reducing Yocto's usefulness.

However, Yocto is here to stay. CPU vendors, module vendors and tools vendors are increasingly adopting Yocto as a standardised base for their products, while Intel and Samsung have announced IoT platforms that are based around Yocto.

Thankfully, the Yocto Project is working to address the learning curve by developing web based visual configuration software. At last, embedded Linux has a credible standard base!

Author profile:
Joe Nicholson is managing director of Rufilla (**www.rufilla.com**).