

e2 factory – the Next Generation Build System for Embedded Systems

An Overview of the Design and Performance Features

Dr. Cord Seele

Today, Linux is the operating system that is often preferred for embedded systems. It is being used with an increasing range of devices and for many of these, such as routers for example, it is already hard to imagine life without it. In some market analyses it has already been designated as the dominating operating system for particular fields of application. The expectation that its use in the embedded sector will increase is generally undisputed. Advantages such as technical transparency, independence from any individual manufacturer and lower costs are often the deciding factors for choosing Linux.

Once the decision to use Linux has been taken, the question soon arises as to where to get „the system“ Linux from. This is because Linux, unlike many of its operating system competitors, does not consist of a clearly defined software package. Instead, it can, indeed must, be pieced together from a multitude of individual software packages. The reason for this can be found in the special nature of open source software, which does not have an exclusive „manufacturer“. Open source software is derived from different development processes to those that are known in classical software development. This implies that in the process of adapting Linux for embedded systems, previous development strategies should be reconsidered and modified if necessary.

What complicates matters further for the beginner is the challenge of finding orientation and gaining an overview of the multitude of availa-

ble software packages and, in some cases, competing approaches - for example in the field of real time extensions. Furthermore, the development of almost every software package follows a schedule of its own and is largely independent of the others. Although this may seem complex and unmanageable at first sight, it offers enormous opportunities: An individual combination – including older software packages, if required – is readily possible. This development should be followed and exploited as profitably as possible for one’s own embedded product.

Linux only becomes useful for the majority of users after this important groundwork has been done.

In the desktop and server sector this is the task of the distributors, who sort through the multitude of necessary packages and put them together in functional combinations for the end user. Linux only becomes useful for the majority of users after this important groundwork has been done. In the embedded world, so-called board support packages (BSPs) have become established instead of ready-to-use distributions. The BSPs serve as a starting point for individual adaptation to the respective embedded system.

In the embedded sector, however, there are even more, very special requirements in addition to the mechanisms known from the server/

desktop industry. Among these, increased quality and maintenance requirements should be mentioned – especially due to the very long product life cycles, commonly ten to fifteen years or longer. Furthermore, many embedded systems are difficult and costly to access in the field. The procedure of loading any updates or fixes needs to be designed accordingly.

In the process of putting together „his“ embedded Linux system, the equipment manufacturer can rely either on the „Roll-your-own“ option, which implies falling back on freely available information and sources by means of one’s own know-how, or on support and/or products from service providers. The selection of the software packages that make sense for the respective product already justifies the consultation of a specialist.

Since each of the software packages needed for a BSP is maintained in an individual version administration system (for this purpose, git, subversion, monoton, CVS and others are used), there is the additional task of following all the changes in the different systems. During the development phase, coupling is very desirable, especially for the kernel, in order to avoid losing touch with the general development at an early stage.

In order to make this process manageable in terms of the workload, seamless coupling to the source code management system (SCM) of the respective package is indispensable

because every change of the source code management system always implies a discontinuity of versioning and thus a loss of information. In addition, this transition usually has to be effected manually, which consumes time and thus scarce developer resources.

A change of the source code management system always implies a discontinuity of versioning...

Not only during the development phase, i.e. prior to the product's market introduction, but also in the subsequent maintenance phase, continuous comparison with the developments in the community is often very useful. Without this comparison, the effort required to integrate new features into the next version of the product is much greater and riskier and is thus often refrained from, at the expense of product innovation.

Therefore, e2 factory developed by emlix provides the option to switch from one source code management system to another for each software package. Since not every package needs this flexibility in practice, of course, the software packages can also be managed and processed by means of tarball (mostly in compliance with the official releases) and patch sets. This variant will usually be chosen for all those packages that need little or no adaptation in the source code and make do with a configuration via the configure switch.

For a modern build system for embedded systems, however, it is insufficient simply to use whatever seems to be suitable among the resources of open source software. It is as important to know at any time what has been built for the target

system and in which way. In order to accomplish this, it is indispensable to combine the build system with a version management system for the build process. This saves „what“ has been built. For a reliable answer to the question „how“, another condition has to be fulfilled: independence from the computer used for development.

In order to ensure this independence for the developed result of each software package, the total build process has to go on under completely controlled conditions. For this purpose, e2 factory uses a chroot cage. Apart from the kernel of the running host system, this contains all the programs, libraries and tools (cross-compiler, make, etc.), and is entirely independent of the host system.

To prevent a package with an unrecognized faulty configuration (e.g. if in a „make install“ a header file of the target architecture is copied to the wrong place in the file system) from causing any side effects in subsequent build processes, the chroot environment is deleted and set up again from scratch before any build process.

The chroot cage itself is administered as an independent project in e2 factory. Thus, the assembly of the chroot cage is subject to the same high quality requirements as a customer BSP. Additionally, it can be equipped with various tools, depending on the package to be built. The build process for any package therefore consists of the following steps in e2 factory:

- Setting up the chroot cage with the specific tools required for the respective task
- Copying the sources needed into the chroot
- Building the package within the

chroot

- Copying the required results of building from the chroot
- Archiving the results
- Deleting the chroot

This independence is a necessary condition for being able to reconstruct the whole system exactly, even after many years. This reproducibility is necessary, for instance, if a previously unrecognized problem occurs after a long time in medical equipment with a long lifespan. In such a situation, it is crucial to be able to solve this problem by means of a clearly defined fix in order to reduce time-consuming and costly validation measures to a minimum.

...it is as important to know at any time what has been built for the target system and in which way.

In addition, reproducibility requires consistent versioning of the state of the entire project at the time of building in order to ensure that exactly this state can be reactivated if necessary. To do this, e2 factory is coupled to a version control system in every necessary phase. For this task, git is preferentially used, but other options are also possible.

From the perspective of e2 factory, the toolchain, i.e. the compiler with its tools, already belongs to the BSP. Of particular importance in this context are the difficult interdependencies of the kernel, the compiler and libstdc++. If they are not observed accurately, very strange effects during runtime can lead to arduous debugging sessions.

Because of this, the entire rootfs depends on the actual compiler in use and its implicit optimizations. In

order to avoid having to maintain these interdependences for each individual package, e2 factory provides the possibility to abstract the interdependence chain of the packages as required by defining „metapackages“ (e.g. toolchain, for binutils, gcc and glibc).

This provides clarity in the project and allows abstraction of the package structure...

This provides clarity in the project and allows abstraction of the package structure according to the needs and preferences of the developers. Of course, any intermediate step can be built specifically at any time. At the end of the interdependence chain is usually the step of creating one or several image files from the built programs. These image files can be written directly into the flash of the target system.

Due to this high degree of flexibility, the dependence tree can be designed very variably. This is of particular advantage if the equipment manufacturer pursues a platform strategy, i.e. if he develops different products on the basis of the same or only slightly modified hardware – a trend that can be observed increasingly at present. Frequently, large parts of the software, especially of the BSP, can be taken over unchanged. Any differences are mostly limited to the application itself as well as to a few hardware drivers in the case of hardware modifications. For this utilization scenario, e2 factory with its flexible package organization system offers the optimal conditions to allow all the products, if necessary, to profit immediately from any bug fixes or upgrades for shared packages.

Yet another feature of e2 factory that makes highly efficient development

processes with reproducible results possible is an ingenious, integrated caching mechanism. Unlike the traditional make, it does not consider the timestamps of files (which can quite possibly diverge on different computers), but the actual content of all the files that are relevant for the build process. These include not only the actual sources and configuration data for a package, but especially all the project-wide settings as well as the entire content of the chroot cage. By means of this clever mechanism, it can typically be decided within a few seconds whether, for example, the whole kernel has to be rebuilt or not. This saves precious time during development and allows work at different locations to be synchronized.

If the software of whole platforms is managed, only those packages that are actually affected by the changes or innovations need to be retranslated for each product. This ensures high productivity and thus savings in time and costs.

Because of the multitude of licences in this area, a proper licence management system is indispensable.

The high quality requirements on present-day embedded systems apply to the operating system as well as the subsequent (main) application. Therefore, it is an obvious and very reasonable measure not to separate the development of the BSP from the development of the application – even if the BSP has not been developed in-house, but by a service provider.

e2 factory provides the optimal conditions for this purpose, because it supports an independent source code management system for each

package. This makes it possible to use the established „in-house standard“ frequently in application development without being forced to switch to possibly unknown new tools. The connection from e2 factory to this source code management system is the only addition which may still have to be created.

Apart from the efficient administration and creation of software, it is equally important in the open source field to know which licences apply to the packages in use. Because of the multitude of licences in this area, a proper licence management system is indispensable. e2 factory supports this by means of clear allocation of the valid licences to each individual package.

With e2 factory, emlix has developed a tool that allows all the software components within an individual system to be built reproducibly and assembled. Furthermore, these features are also available for development processes in distributed teams. The tool is also used internally by emlix for the development and creation of all customer projects.

There are at least two different reasons to opt for e2 factory as a build system and to use it oneself as well. Firstly, the technical advantages alone can be the deciding factor for choosing this modern and innovative tool.

If, in addition, several partners are involved in the development, or if the team of developers is distributed over several locations, the potential of e2 factory in comparison with other build systems really comes into play. This can be the case if several development departments exist in different places, or if individual software components are developed by external service providers. e2 factory

offers these teams seamless cooperation on the development level and ensures the important quality of reproducibility at the same time. In many cases, the time-consuming generation of releases and their distribution to the partners is no longer necessary.

Those manufacturers of chips or components who typically generate reference designs for their products by means of a Linux BSP, on the basis of which their customers develop the end product, gain yet another benefit from using e2 factory: If the BSP is e2 factory-based, they can offer their customers seamless support through emlix for the entire process of product development.

The only thing necessary for this cooperation network to function, apart from e2 factory itself, is a source code management system with the ability to replicate, such as git...

Since the integration of one's own software packages is also possible, the chip manufacturer can also perform seamless co-development on his packages together with his customers if desired. At the same time, and independent of this, emlix can sup-

port these customers in the adaptation of the BSP to the hardware of their end product. The only thing necessary for this cooperation network to function, apart from e2 factory itself, is a source code management system with the ability to replicate, such as git, and a one-time configuration of the cooperation conditions.

The BSPs created using e2 factory are delivered in three variants: Normally, all the sources and configurations that the customer needs to reproduce the system by means of the make command are extracted. The chroot cage ensures reproducibility at the customer's location. Optionally, the „Standard Edition“ of e2 factory is available together with the BSP. This includes the tool in a binary form and access to upgrades. If complete independence from emlix is requested or required by any regulations, the „Enterprise Edition“ provides the tool in the form of its source code. This allows maintenance and further development to be performed independently of emlix.

At present, e2 factory is a tool that has been written and designed for use with the command prompt. For further development, it is planned to extend it by adding graphical front ends, which are intended to make it easier to use for the new or occasional user.