

e2 factory – Bausystem der nächsten Generation für Embedded Systeme

Ein Überblick über Konzeption und Leistungsmerkmale

von Dr. Cord Seele

Linux ist heute ein gern gewähltes Betriebssystem für Embedded Systeme. Es erschließt sich ein wachsendes Feld an Geräten und ist in vielen Bereichen wie etwa Routern bereits nicht mehr wegzudenken. In einzelnen Marktanalysen wird es für bestimmte Bereiche bereits heute als dominierendes Betriebssystem bezeichnet. Seine weitere Ausbreitung im Embedded-Bereich ist allgemein unbestritten und erwartet. Die Vorteile wie technische Transparenz, Unabhängigkeit von einem spezifischen Hersteller und geringere Kosten sind häufig die ausschlaggebenden Kriterien für die Wahl von Linux.

Wenn Linux zum Einsatz kommen soll, stellt sich schnell die Frage, woher „das“ Linux genommen wird. Denn Linux besteht nicht wie viele seiner Betriebssystem-Wettbewerber aus einem klar definierten Software-Paket, sondern kann beziehungsweise muss aus einer Vielzahl von einzelnen Software-Paketen zusammengestellt werden. Dies ist in der besonderen Natur von Open Source Software begründet, die keinen alleinigen „Hersteller“ kennt. Open Source Software unterliegt anderen Entwicklungsprozessen als diese bisher aus der klassischen Software-Entwicklung bekannt sind. Dies führt dazu, dass bei der Adaption von Linux für Embedded Systeme bisherige Entwicklungsstrategien überdacht und gegebenenfalls angepasst werden sollten.

Für den Neuling kommt erschwerend hinzu, sich in der Vielzahl der zur Verfügung stehenden Software-

Pakete und zum Teil konkurrierenden Ansätze wie etwa im Bereich Realtime-Erweiterungen zurechtzufinden und den Überblick zu gewinnen. Außerdem unterliegt die Entwicklung von fast jedem Software-Paket einem eigenen Zeitplan und ist weitgehend unabhängig von den übrigen. Was im ersten Augenblick als Undurchsichtigkeit und Unbeherrschbarkeit empfunden werden könnte, bietet enorme Chancen: Eine individuelle Zusammenstellung – bei Bedarf auch von älteren Software-Paketen – ist ohne Weiteres möglich. Diese Entwicklung gilt es, zu verfolgen und möglichst Gewinn bringend für das eigene Embedded-Produkt umzusetzen.

Erst durch diese Vorarbeit ist Linux für die Mehrzahl der Anwender nutzbar geworden.

Im Desktop- und Serverbereich ist dies die Aufgabe der Distributionen, die die Vielzahl der benötigten Pakete sichten und funktionsfähig für den Endbenutzer zusammenstellen. Erst durch diese wichtige Vorarbeit ist Linux für die Mehrzahl der Anwender benutzbar geworden. Für die Embedded-Welt haben sich anstelle fertiger Distributionen so genannte Board Support Packages (BSPs) etabliert, die als Ausgangsbasis für die individuelle Anpassung an das jeweilige Embedded System dienen.

Für den Embedded-Bereich kommen jedoch neben den aus dem Server/Desktop-Bereich bekannten Mechanismen noch weitere, sehr

spezielle Anforderungen hinzu. Hierzu zählen – insbesondere wegen der häufig sehr langen Produktlebenszyklen von zehn bis fünfzehn Jahren oder mehr – gesteigerte Qualitäts- und Wartungsanforderungen. Außerdem sind viele der Embedded Systeme im Feld nur noch schwer – verbunden mit hohen Kosten – erreichbar. Das Einspielen von Updates oder Fixes ist dementsprechend zu gestalten.

Bei der Zusammenstellung „seines“ Embedded Linux Systems kann der Gerätehersteller neben der „Roll-your-own“-Variante, bei der durch eigenes Know-how auf frei verfügbare Informationen und Sourcen zurückgegriffen wird, auch auf Support und/oder Produkte von Dienstleistern zurückgreifen. Schon die Selektion der für das jeweilige Produkt sinnvollen Software-Pakete in der geeigneten Version rechtfertigt es, das Know-how eines Spezialisten hinzuzuziehen.

Da jedes der für ein BSP benötigten Software-Pakete von seinem Maintainer in einem eigenen Versionsverwaltungssystem gepflegt wird (hier kommen beispielsweise *git*, *subversion*, *monoton*, *CVS* und andere zum Einsatz), besteht zusätzlich die Aufgabe, die Veränderungen in den verschiedenen Systemen zu verfolgen. Eine Kopplung ist während der Entwicklung speziell für den Kernel sehr wünschenswert, um sich nicht frühzeitig von der allgemeinen Entwicklung abzukoppeln.

Damit dieser Prozess vom Aufwand her überhaupt leistbar ist, ist eine nahtlose Kopplung an das Source

Code Management System (SCM) des jeweiligen Paketes nötig. Denn ein Wechsel des Source Code Management Systems bedeutet immer einen Bruch in der Kontinuität der Versionierung und damit einen Informationsverlust. Außerdem ist dieser Übergang meist manuell zu leisten, was Zeit und damit knappe Entwicklerressourcen beansprucht.

Ein Wechsel des Source Code Management Systems bedeutet immer einen Bruch in der Kontinuität der Versionierung...

Aber nicht nur während der Entwicklung, also vor der Markteinführung des Produktes, sondern auch in der Wartungsphase danach ist ein kontinuierlicher Abgleich mit den Weiterentwicklungen in der Community häufig sehr sinnvoll. Denn ohne diesen Abgleich ist der Aufwand, neue Features in die nächste Produktversion zu integrieren, ungleich höher und risikoreicher. Er wird dann gerne zu Lasten der Produktinnovation unterlassen.

Das von emlix entwickelte *e2 factory* bietet daher die Möglichkeit, für jedes Software-Paket das verwendete Source Code Management zu wechseln. Da natürlich nicht für jedes Paket diese Flexibilität in der Praxis benötigt wird, können Software-Pakete natürlich auch per tar-Ball (meist entsprechend der offiziellen Releases) und Patchsets verwaltet und bearbeitet werden. In der Regel wird man diese Variante für all jene Pakete wählen, die keine oder sehr wenig Anpassungen im Sourcecode benötigen und mit Konfiguration per Configure-Schalter auskommen.

Für ein modernes Bausystem für Embedded Systeme reicht es aber

nicht aus, sich nur geeignet aus den Ressourcen der Open Source Software zu bedienen. Es ist ebenso wichtig, jederzeit zu wissen, was wie für das Zielsystem gebaut wurde. Um dies zu erreichen, ist eine Integration des Bausystems mit einer Versionsverwaltung für den Bauprozess zwingend notwendig. Dies sichert „Was“ gebaut wurde. Für die sichere Beantwortung der Frage nach dem „Wie“ ist eine weitere Maßnahme erforderlich: die Unabhängigkeit vom Baurechner.

Um diese für das Bauergebnis eines jeden Software-Paketes zu gewährleisten, muss der gesamte Bauvorgang unter vollständig kontrollierten Bedingungen ablaufen. *e2 factory* verwendet dafür einen chroot-Käfig. Neben dem Kernel des laufenden Host-Systems sind darin sämtliche Programme, Bibliotheken und Tools (Cross-Compiler, make, etc.) vollständig unabhängig vom Hostsystem enthalten.

Damit auch ein unerkannt fehlerhaft konfiguriertes Paket (z.B. wenn bei einem „*make install*“ ein Headerfile der Targetarchitektur an die falsche Stelle im Dateisystem kopiert wird) nicht zu Seiteneffekten in nachfolgenden Bauvorgängen führt, wird die chroot-Umgebung vor jedem Bauvorgang gelöscht und neu aufgesetzt.

Der chroot-Käfig wird selbst als eigenständiges Projekt in *e2 factory* verwaltet. Damit unterliegt auch seine Zusammenstellung den gleich hohen Qualitätsanforderungen wie ein Kunden-BSP. Zusätzlich kann er je nach zu bauendem Paket mit verschiedenen Tools ausgestattet sein.

Der Bauvorgang für ein beliebiges Paket besteht in *e2 factory* daher aus den folgenden Schritten:

- Aufsetzen des chroot-Käfiges mit den spezifisch benötigten Tools

- Kopieren der benötigten Quellen in die chroot
- Bauen des Paketes innerhalb der chroot
- Kopieren der benötigten Bauergebnisse aus der chroot
- Archivierung der Ergebnisse
- Löschen der chroot.

Diese Unabhängigkeit ist notwendige Voraussetzung dafür, dass das Gesamtsystem auch in vielen Jahren noch exakt rekonstruiert werden kann. Diese Reproduzierbarkeit ist beispielsweise dann notwendig, wenn in einem langlebigen medizintechnischen Produkt erst spät ein bis dato unerkanntes Problem auftaucht.

In einer solchen Situation ist es sehr wichtig, dieses mit einem klar definierten Fix beheben zu können, um zeitraubende und kostenintensive Validierungsmaßnahmen auf ein Minimum zu reduzieren.

Die Reproduzierbarkeit erfordert außerdem eine konsequente Versionierung des gesamten Projektzustandes zum Bauzeitpunkt...

Die Reproduzierbarkeit erfordert außerdem eine konsequente Versionierung des gesamten Projektzustandes zum Bauzeitpunkt, um genau diesen Stand bei Bedarf wieder aktivieren zu können. Dafür ist *e2 factory* in jeder nötigen Phase mit einem Versionskontrollsystem gekoppelt. Hierfür wird vorzugsweise *git* verwendet, andere sind aber ebenso denkbar.

Aus Sicht von *e2 factory* gehört die Toolchain, also der Compiler mit seinen Tools bereits zum BSP. Insbesondere gibt es hier diffizile Abhängigkeiten von Kernel, Compiler und libstdc++. Werden diese nicht sauber eingehalten, können merkwürdigste

Effekte zur Laufzeit zu mühseligen Debuggingssessions führen.

Dadurch ist das gesamte rootfs abhängig vom konkreten Compiler und seinen impliziten Optimierungen. Um diese Abhängigkeiten aber nicht für jedes Paket einzeln pflegen zu müssen, bietet *e2 factory* die Möglichkeit, durch Definition von ‚Metapaketen‘ (z.B. toolchain, für binutils, gcc und glibc) die Abhängigkeitskette der Pakete beliebig zu abstrahieren.

Dies schafft Übersichtlichkeit im Projekt und ermöglicht eine Abstraktion der Paketstruktur...

Dies schafft Übersichtlichkeit im Projekt und ermöglicht eine Abstraktion der Paketstruktur entsprechend der Bedürfnissen und Wünschen der Entwickler. Natürlich kann jederzeit jeder beliebige Zwischenschritt gezielt gebaut werden. Am Ende der Abhängigkeitskette steht typischerweise der Schritt, aus den gebauten Programmen eine oder mehrere Imagedateien zu erstellen, die direkt ins Flash des Zielsystems geschrieben werden können.

Durch diese hohe Flexibilität lässt sich der Abhängigkeitsbaum sehr variabel gestalten. Dies ist insbesondere dann von großem Vorteil, wenn der Gerätehersteller – wie heute zunehmend festzustellen – eine Plattformstrategie verfolgt, also mit der gleichen oder einer gering modifizierten Hardware mehrere Endgeräte entwickelt. Häufig können weite Teile der Software, insbesondere des BSPs, unverändert übernommen werden. Unterschiede sind meist auf die Anwendung selbst sowie auf einige Gerätetreiber bei Hardware-Modifikationen beschränkt. Für dieses Einsatzszenario bietet *e2 factory*

mit seiner flexiblen Paketorganisation die optimalen Voraussetzungen, damit alle Produkte bei Bedarf sofort von Bugfixes oder Upgrades gemeinsamer Pakete profitieren.

Als ein weiteres Feature für die sehr effiziente Entwicklung mit reproduzierbaren Ergebnissen ist in *e2 factory* ein ausgeklügelter Caching-Mechanismus integriert. Dieser berücksichtigt nicht wie das traditionelle make die Zeitstempel von Dateien (die auf unterschiedlichen Rechnern durchaus divergieren können), sondern den tatsächlichen Inhalt sämtlicher für den Bauprozess relevanten Dateien. Dies sind neben den eigentlichen Sourcen und Konfigurationsdaten für ein Paket insbesondere sämtliche projektweiten Einstellungen sowie der gesamte Inhalt des chroot-Käfigs. Durch den ausgeklügelten Mechanismus kann typischerweise in wenigen Sekunden entschieden werden, ob beispielsweise der gesamte Kernel neu gebaut werden muss oder nicht. Dies spart wertvolle Zeit während der Entwicklung und ermöglicht eine Synchronisation über Arbeitsplätze hinweg.

Wird die Software für ganze Plattformen verwaltet, sind für jedes Produkt nur jene Pakete neu zu übersetzen, die wirklich von den Änderungen oder Neuerungen betroffen sind. Dies gewährleistet eine hohe Produktivität und damit Zeit- und Kostenersparnis.

Durch die Vielzahl der in diesem Umfeld anzutreffenden Lizenzen ist ein sauberes Lizenzmanagement unverzichtbar.

Die hohen Qualitätsanforderungen an heutige Embedded Systeme gelten sowohl für das Betriebssystem als auch die spätere (Haupt-)Anwen-

dung. Daher ist es naheliegend und sehr sinnvoll, die Entwicklung des BSPs nicht künstlich von der der Anwendungen zu trennen – auch wenn das BSP nicht im eigenen Haus, sondern von einem Dienstleister entwickelt wurde.

e2 factory bietet dafür die besten Voraussetzungen, da es pro Paket ein eigenes Source Code Management System unterstützt. Damit kann für die Entwicklung der Anwendung häufig auch der etablierte „Hausstandard“ zum Einsatz kommen, ohne den Zwang auf eventuell unbekannte, neue Tools zu wechseln. Lediglich die Anbindung von *e2 factory* an dieses Source Code Management System ist gegebenenfalls noch zu erstellen.

Neben der effizienten Verwaltung und Erstellung von Software ist es im Umfeld von Open Source Software ebenso wichtig zu wissen, welchen Lizenzen die verwendeten Pakete unterliegen. Durch die Vielzahl der in diesem Umfeld anzutreffenden Lizenzen ist ein sauberes Lizenzmanagement unverzichtbar. *e2 factory* unterstützt dies durch eine klare Zuordnung der gültigen Lizenzen für jedes einzelne Paket.

Mit *e2 factory* hat emlix ein Tool entwickelt, das den reproduzierbaren Bau und die Zusammenstellung aller Software-Komponenten in einem System ermöglicht. Darüber hinaus stehen diese Eigenschaften auch bei der Arbeit in verteilten Teams zur Verfügung. Das Werkzeug wird auch intern von emlix für die Entwicklung und die Erstellung aller Kundenprojekte verwendet.

Es gibt mindestens zwei unterschiedliche Motivationen, sich für *e2 factory* als Bausystem zu entscheiden und es auch selbst zu verwenden. Zunächst können bereits die reinen technischen Vorteile ausschlagge-

bend sein, auf dieses moderne und innovative Tool zu setzen.

Werden darüber hinaus aber zum Beispiel mehrere Partner an der Entwicklung beteiligt oder ist das Entwicklerteam auf verschiedene Standorte verteilt, kann *e2 factory* seine Stärke gegenüber anderen Bausystemen erst richtig ausspielen. Dabei kann es sich um mehrere Entwicklungsabteilungen an verschiedenen Standorten handeln oder auch um externe Dienstleister für einzelne Software-Komponenten.

e2 factory bietet diesen Teams eine nahtlose Zusammenarbeit auf Entwicklerebene, ohne die wichtige Eigenschaft der Reproduzierbarkeit zu verlieren. Die aufwändige Generierung von Releases und ihre Verteilung an die Partner ist häufig nicht mehr nötig.

Für diese netzartige Zusammenarbeit ist neben e2 factory lediglich ein replikationsfähiges SCM wie etwa git notwendig...

Herstellern von Chips oder Komponenten, die für ihre Produkte

typischerweise Referenzdesigns mit einem Linux BSP erstellen, auf dessen Basis ihre Kunden dann das Endprodukt entwickeln, bietet *e2 factory* einen weiteren Bonus: Ist das BSP *e2 factory*-basiert, können sie ihren Kunden den nahtlosen Support durch emlix für die gesamte Produkt-Entwicklung anbieten. Da auch die Integration eigener Software-Pakete möglich ist, kann der Chip-Hersteller mit seinen Kunden auf Wunsch ebenfalls die nahtlose Ko-Entwicklung auf seinen Paketen durchführen.

Gleichzeitig und davon unabhängig kann emlix diese Kunden bei der Anpassung des BSPs an die Hardware ihres Endproduktes unterstützen. Für diese netzartige Zusammenarbeit ist neben *e2 factory* lediglich ein replikationsfähiges Source Code Management System wie etwa *git* notwendig sowie die einmalige Konfiguration der Kollaborationsverhältnisse.

Mit *e2 factory* erstellte BSPs werden in drei Varianten ausgeliefert: Im Normalfall werden alle Sourcen und Konfigurationen extrahiert, die der Kunde zum Nachbau des Systems per *make*-Befehl benötigt. Der chroot-Käfig gewährleistet dabei die Reproduzierbarkeit beim Kunden. Optional ist mit dem BSP

die „*Standard Edition*“ von *e2 factory* erhältlich: das Tool in binärer Form und Zugang zu Weiterentwicklungen. Ist eine komplette Unabhängigkeit von emlix gewünscht oder durch Regularien gefordert, bietet die „*Enterprise Edition*“ das Tool im Sourcecode. Damit ist eine von emlix unabhängige Pflege und Weiterentwicklung möglich.

Zur Zeit ist *e2 factory* ein Werkzeug, das für die Bedienung auf der Kommandozeile geschrieben und konzipiert worden ist. Für die Weiterentwicklung ist vorgesehen, es um graphische Frontends zu erweitern, die dem neuen oder gelegentlichen Benutzer mehr Sicherheit in der Bedienung geben sollen.