Übersicht: Was Linux-Distributionsbaukästen können

Montiert

Tam Hanna

Eigene Linux-Distributionen kann man mit diversen Baukastensystemen generieren. Viele nutzen das Build-System BitBake, aber es gibt auch Alternativen. Ein Überblick.

Mit dem Baukasten Yocto können Embedded-Entwickler eigene Linux-Distributionen erstellen, die auf die Bedürfnisse der Geräte abgestimmt sind, wie der Artikel „Linux nach Maß“ ab Seite 44 in diesem Heft zeigt. Trotz des beachtlichen Erfolgs des Projekts gibt es eine Vielzahl weiterer Anbieter, die ihre Marktposition teilweise seit langer Zeit verteidigen.

Es wäre grob fahrlässig, diesen Zustand mit einer fehlenden Flexibilität des durchschnittlichen Embedded-Programmierers zu begründen. Es steht außer Frage, dass sie nicht jedem Trend nachlaufen – das jahrelange Überleben offensichtlich unterlegener Plattformen wäre ein nach den Gesetzen des Marktes nur schwer erklärbarer Sonderfall. Aufgrund der enormen Vielfalt an weiteren Baukästen kann dieser Artikel nur einige populäre Vertreter ihrer Zunft vorstellen. Mit diesem Wissen kann man sich dann an die Auswahl der passenden Implementierung heranwagen.

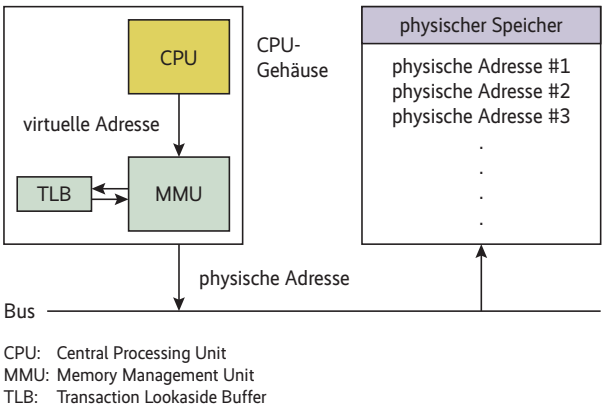
uClinux

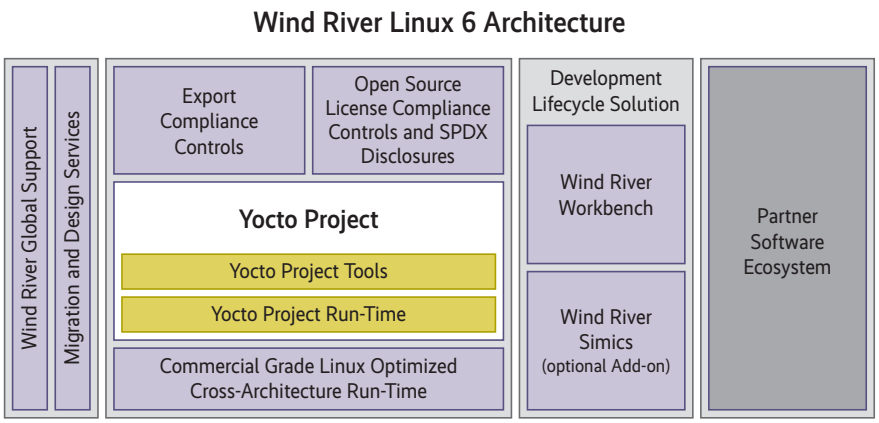
Linux war von Anfang an für Controller-Systeme vorgesehen, die mit einer Memory Management Unit (MMU) ausgestattet sind, also mit einem Bauteil, das Speicheranforderungen übersetzt und den Speicher mit einer als Paging bezeichneten Technik aufteilt (siehe Abbildung 1). Als angenehmer Nebeneffekt fangen MMUs zudem Zugriffe auf nicht berechtigte Teile des Speichers mit einer Exception ab, sodass sich mehrere Prozesse nicht so ohne Weiteres in die Haare geraten können.

David Wood, Mitgründer des Betriebssystems Symbian, schildert die Ergebnisse des Versuchs, das für Geräte mit MMU vorgesehene Symbian auf einem Telefon ohne Memory Management Unit (MMU) zum Laufen zu bringen (siehe „Onlinequellen“, [a]). Das Gerät wurde ob der erheblichen Verspätung nie ausgeliefert. Der enorme Entwicklungsmehraufwand lässt sich laut Wood auf das Fehlen der MMU zurückführen.

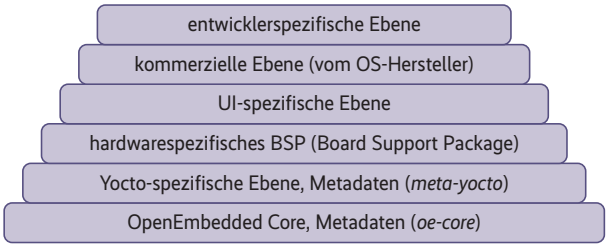
Aber es gibt auch Distributionen für Mikrocontroller ohne MMU, wie den im Jahr 1999 als uClinux bezeichneten Port, um eine Linux-Version für klassische 68k-Palms zu bekommen. Im Laufe der Jahre wurde das auf den Projektseiten zum Download bereitstehende System-Image an verschiedene Controller-Architekturen angepasst. Die grundlegende Vorgehensweise ist immer gleich: Ein Cross-Compiler erstellt Betriebssystem und Nutzerapplikation. Die beiden Prozesse lassen sich durch Anpassen des

Linux war von Anfang an für Controller-Systeme vorgesehen, die mit einer Memory Management Unit (MMU) ausgestattet sind (Abb. 1).





Wind Rivers echtzeitfähiges Linux basiert auf Yocto (Abb. 2).



Poky ist eine Embedded-Distribution, die die in *meta-oe* enthaltenen Layer nutzt (Abb. 3).

Makefiles vereinen: Es kann mitunter sinnvoller sein, größere Embedded-Systeme per TCP/IP mit neuen Versionen der Betriebssoftware zu versorgen.

Der Begriff „uClinux“ führt insofern in die Irre, als sich die Plattform als mit „normalen“ Linuxen nicht vollständig kompatibel erweist. Die englischsprachige FAQ [b] benennt drei relevante Unterschiede:

– Die Einschränkungen bedeuten nicht, dass uClinux gar kein Multitasking beherrscht. Der Eltern-Prozess bleibt so lange stehen, bis das Kind *exec()* oder *exit()* aufruft.
– uClinux kann seinen Stack nicht automatisch vergrößern (kein *autogrow*) und implementiert die Funktion *brk()* nicht. Entwickler müssen zum Allozieren von Speicher *mmap()* aufrufen. Zudem gibt es eine Compiler-Option, die die Größe des Stacks festlegt.
– Es gibt keinen Speicherschutz. Jedes Programm kann im Speicherbereich des anderen herumfuhrwerken und es so zum Absturz bringen. Das ist allerdings kein Problem, wenn Programmierer ihr Programm unter Berücksichtigung dieser Tatsache entwerfen.

Ein Einsatz von uClinux ergibt Sinn, wenn eine Applikation von einigen lokal begrenzten Unix-Funktionen profitiert. Wer existierenden Linux-Code recyceln möchte, ist bei uClinux ebenfalls gut aufgehoben. Größere Systeme lassen sich aber schwer realisieren, da das Verpacken der diversen Bibliotheken und Frameworks viel Aufwand erfordert.

OpenWrt

uClinux kommt unter anderem in Routern zum Einsatz und benötigt wenig Ressourcen. Im Gegensatz dazu kann die aktuelle Version des ebenfalls als Router-Betriebssystem zu Weltruhm gekommenen OpenWrt auf Geräten mit weniger als 16 Megabyte nicht überleben – eine MMU braucht die Distribution mehr oder weniger standardmäßig.

Der wichtigste Unterschied zu uClinux liegt darin, dass man das Betriebssystem-Image nicht mehr direkt über ein Makefile generiert. Stattdessen kommt ein als Buildroot bezeichnetes Kompilierungsmanagementsystem zum Einsatz, das die durch einzelne Makefiles beschriebenen Softwarepakete erzeugt. In seinem auf der Embedded Linux Conference 2012 gehaltenen Vortrag erwähnte Thomas Petazzoni [c], dass man den in Yocto und OpenEmbedded [d] verwendeten BitBake-Builder ab etwa 50 Komponenten sinnvoll nutzen kann. Das dynamische Verändern des Inhalts eines Embedded-Systems ist nicht vorgesehen.

OpenWrt geht insofern über Buildroot hinaus, als es mit *opkg* einen hauseigenen Paketmanager mitbringt, der sich an ausgewachsene Programme wie *apt-get* anlehnt. Entstanden ist *opkg* durch einen Fork des *ipkg* (Itsy Package Management System). Die angebotene Software wuchs rasch: Mittlerweile existieren fast 3000 verschiedene Pakete für OpenWrt – zwecks Fernsteuerung gehört ein LuCI genanntes Webinterface dazu.

Wer sein Embedded-Gerät mit Grafikausgabe ausstatten möchte, ist hier allerdings schlecht aufgehoben. Die Unterstützung von DirectFB (Direct Frame Buffer) und diversen Mini-X-Servern mag für das Anzeigen von Text genügen. Aufgrund

iX-TRACT

- Es existiert eine beträchtliche Anzahl von Baukästen, mit denen man eigene, auf die Bedürfnisse der Geräte abgestimmte Embedded-Distributionen erzeugen kann.
- Das aus dem OpenEmbedded-Projekt stammende Build-System BitBake ist sehr verbreitet.
- Im Laufe der letzten Jahre haben alle wichtigen Anbieter von Embedded Linux ihre hauseigenen Build-Systeme durch BitBake ersetzt.

Ohne Unix

Wer Yocto sehr sparsam konfiguriert, bekommt ein wenige Megabyte großes Image. Im Vergleich zum durchschnittlichen Desktop-Betriebssystem erscheint dies winzig – aber im Embedded-Bereich müssen viele weitverbreitete Prozessoren mit wenigen Kilobyte Codespeicher auskommen.

Bei kleinen oder zeitkritischen Steuerungen empfiehlt es sich gelegentlich, auf den Komfort eines unixoiden Betriebssystems zu verzichten. Kleinere Embedded-Betriebssysteme bieten Entwicklern eine – je nach Controllertyp mehr oder weniger – robuste Threading-Infrastruktur an, in manchen Fällen gibt es primitives Speichermanagement und Erweiterungen für TCP/IP, Dateisysteme und andere Hardware.

Allerdings nutzen Hersteller proprietärer Embedded-Betriebssysteme oft ein Lizenzmodell, das an die Aufpreisliste deutscher Autos erinnert. Die meist geringen Kosten für den Basiskernel steigern sich zu geradezu astronomischen Summen, wenn Zusatzfunktionen wie TCP/IP hinzukommen.

Die Programmierung erfolgt normalerweise in C oder C++. Bei der Auswahl eines Wunschsystems sollte man natürlich die Zielhardware berücksichtigen: Hierfür gibt es klassische und für gut befundene Kombinationen aus Compiler, Mikrocontroller-Unit (MCU) und Betriebssystem.

Manche Anbieter offerieren ganze GUI-Stacks für kleinere Echtzeitbetriebssysteme. Vor ihrem Einsatz ist es ratsam, die durch Unix entstehenden Mehrkosten sorgsam abzuwägen: Es ist am Ende oft günstiger, nicht auf proprietäre und wenig verbreitete Techniken zu setzen.

der radikal gestiegenen Ansprüche der Endkunden dürften sich die Grenzen der Plattform recht bald zeigen.

Yoctos Brüder

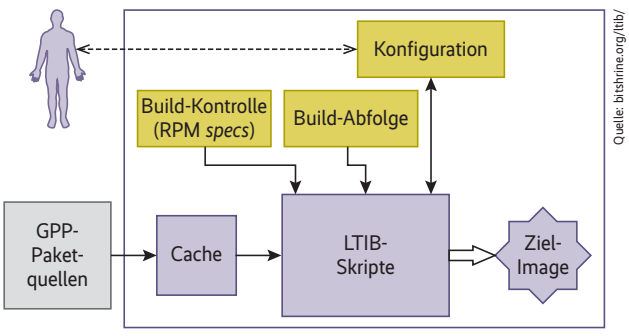
OpenWrt erweist sich als eine durchaus brauchbare Methode zum Erzeugen eines auf die eigenen Bedürfnisse zugeschnittenen Unix-Betriebssystems. Die Einfachheit des Buildroot-Systems entpuppt sich gleichzeitig als seine größte Schwäche: Komplexes Abhängigkeits-Tracking ist nicht (oder nur sehr leidlich) implementiert. Für ein kleines und schnell rekompilierbares Projekt bildet das kein Hindernis – ab einer gewissen Größe der Codebasis ist Dependency Tracking jedoch mehr als hilfreich.

Das komplett in Codeform vorliegende Gentoo Linux löste dieses Problem mit einem als Portage bezeichneten Paketmanager. Er diente als Basis für BitBake, das sich als Standard in Sachen Embedded-Build-Systeme etabliert hat. So treten Distributoren als Value Added Retailer auf, die ihren Kunden auf BitBake basierende Zusatzdienste offerieren. Das in Abbildung 2 gezeigte Übersichtsdiagramm zeigt Wind River Linux.

Im Laufe der letzten Jahre haben alle wichtigen Embedded-Linux-Anbieter ihre hauseigenen Build-Systeme durch BitBake ersetzt. Daraus folgt, dass sich die Produkte bis zu einem gewissen Grad ähneln: Unterschiede bestehen im Preis, in der Lizenzierung, dem Support und in der für das Erstellen von Zusatzprogrammen vorgesehenen Entwicklungsumgebung.

Auf eine an Wind River erinnernde Strategie setzt das Unternehmen MontaVista. Bekannt ist das Linux des Hauses für seine Echtzeitfähigkeiten: Der Hersteller betont, dass sein Betriebssystem auch für „harte Echtzeitsituationen“ geeignet sei, auch wenn sie im Artikel „Sofort reagieren“ ab Seite 50 hierfür ihr proprietäres Pendant VxWorks empfehlen.

Freunde von Qt finden in Qt Enterprise Embedded eine attraktive Alternative zu Yocto und Co. Das nur für kommerzielle Qt-Lizenznehmer zugängliche System erweist sich als optimale Runtime für auf dem Framework basierende Applikationen: Die Wartung durch das finnische Softwareunternehmen Digia soll optimale Performance gewährleisten.



Aufbau des Projekts Linux Target Image Builder (LTIB) – einer Alternative zu BitBake (Abb. 4)

BitBake-Distributionen verwischen die Grenze zwischen Embedded- und Desktop-Betriebssystem. So gut wie alle Produkte lassen sich auf Wunsch mit einem X-Server ausstatten, der das Ausführen mehr oder weniger beliebiger grafischer Anwendungen gestattet. Ein meist optionaler Paketmanager erlaubt das flexible Nachinstallieren benötigter Programme.

Der Unterschied zu Klassikern wie Debian, Red Hat und Ubuntu liegt in der geringeren Verbreitung – ein insofern relevanter Aspekt, als viele Entwickler ihre Programme nicht für wenig populäre Paketmanager anbieten. Dass von Hand kompilierte Binärdateien im Zweifelsfall laufen, darf ebenfalls als fraglich gelten.

Poke the Poky

Das OpenEmbedded-Framework bietet eine Sammlung von im BitBake-Format vorliegenden Softwareschichten an, die für das Generieren von Embedded-Systemen vorgesehen sind. Wer mit Yocto arbeitet, setzt zwangsweise auf *meta-oe*, weil die als Referenzsystem dienende Distribution „Poky“ auf den in *meta-oe* enthaltenen Rezepten basiert (siehe Abbildung 3).

Während es sich bei Poky um ein vergleichsweise neues System handelt, blickt die nach einer Längeneinheit benannte Distribution Angstrom auf eine längere Geschichte zurück: Sie entstand aus der Zusammenführung diverser im Mobilbereich angesiedelter Produkte.

Beide Plattformen nutzen das BitBake-Build-System. Der für Entwickler wohl wichtigste Unterschied liegt darin, dass das Poky-Projekt seine Software „nur“ in Form von Quellcodedateien anbietet. Angstrom liegt auch in Form von fertigen Images vor, deren Deployment Rechenzeit spart. Zudem verfügt es von Haus aus über *opkg*, was das Erstellen benutzerfreundlicher Produkte erleichtert.

Es geht auch anders

Die enorme Dominanz von Buildroot-basierten Distributionen lässt andere Build-Systeme in den Hintergrund treten. Ein paar Beispiele seien hier genannt. So steht LFS (Linux from Scratch) für ein vergleichsweise komplexes System zum Generieren abgespeckter Linux-Distributionen.

Im Allgemeinen nutzen Hardwarehersteller LFS aufgrund seiner vergleichsweise geringen Popularität nicht. Zum Zeitpunkt der Drucklegung dieses Hefts

war die Webseite der kreuzkompilierbaren Variante von LFS zeitweilig offline: Wer nicht auf ein x86-Zielsystem setzen möchte, hat gegebenenfalls Pech gehabt.

Auf der Habenseite steht die exzellente Dokumentation, die angehende Programmierer Schritt für Schritt zum fertigen System führt. Wer mehr über den inneren Aufbau von Linux lernen möchte, ist an dieser Stelle bestens aufgehoben.

Hinter dem Akronym LTIB verbirgt sich der „Linux Target Image Builder". Es handelt sich dabei um ein Alternativprojekt zu BitBake, das ähnliche Ziele verfolgt. Das in Abbildung 4 gezeigte Übersichtsdiagramm zeigt den inneren Aufbau des Produkts.

Der größte Nachteil von LTIB liegt in der vergleichsweise langsamen Weiterentwicklung. Laut der offiziellen Projektwebseite gibt es seit 2010 kein neues Release mehr, die rund zweihundert „User Space Packages" enthalten größtenteils veraltete Software.

In Japan genießt Scratchbox besondere Popularität. Das ehemals von Nokia für das Betriebssystem Maemo entwickelte Produkt eignet sich nicht zum Herstellen eigener Distributionen – es handelt sich dabei vielmehr um einen klassischen Cross-Compiler, der Applikationen und Bibliotheken übersetzt.

Desktop-Distro auf Diät

Viele Embedded-Systeme verhalten sich gemäß dem von PalmSource vor vielen Jahren festgestellten Zusammenhang: Das Hinzufügen weiterer Funktionen trägt ab einem gewissen Grad nicht mehr zur Steigerung der Nutzerzufriedenheit bei.

Als klassische Negativbeispiele seien die mit Angry Birds kompatible CNC-Maschine und das Qt Creator ausführende In-Vehicle-Entertainment-System angeführt. Insbesondere bei technikaffinen Kunden kann es sinnvoll sein, trotz dieser Risiken auf eine mehr oder weniger abgespeckte Version von Android, Debian oder Ubuntu zu setzen.

Oft ist der Hersteller des Chips beziehungsweise Einplatinencomputers an dieser Stelle der beste Ansprechpartner. Häufig bieten Chip-Häuser eine oder mehrere Referenzdistributionen an, im Fall des Raspberry Pi wäre dies beispielsweise Raspbian. Wer ein vorgegebenes System als Basis erwählt, muss sich nicht mit dem Konfigurieren der Hardware befassen.

Reduktionsvorgänge beginnen normalerweise mit einer mehr oder weniger standardisierten Distribution, die man im nächsten Schritt um einzelne Pakete erleichtert. Ein Unit-Test sollte bei dieser an Differential Debugging erinnernden Vorgehensweise sicherstellen, dass das Entfernen nicht versehentlich ein oder mehrere wichtige Module über den Jordan schickt.

Ein so erzeugtes System hält im Bezug auf den Speicherverbrauch mit den vorher genannten Alternativen auf keinen Fall mit. Der Vorteil dieser Vorgehensweise ist, dass ihre Steuerung auf eine Vielzahl von etablierten Infrastrukturdiensten zurückgreifen kann – das Stichwort Play Store sollte zur Illustration dieses Sachverhalts ausreichen.

Angebote von ungewohnter Stelle

Bei Embedded-Programmierern führt die Nennung des Namens Microsoft oft zu spontanem Gelächter: Die bei Webseiten wie Daily WTF verteilten Bilder von abgestürzten Windows-Betriebssystemen haben sich in das kollektive Gedächtnis dieser ohnehin abergläubischen Kundenschicht eingebrannt.

Freunde von C# sollten bei nicht zeitkritischen Systemen eventuell das .NET Micro Framework berücksichtigen. Das unter einer vergleichsweise liberalen Lizenz angebotene Produkt enthält einen GUI-Stack und diverse Netzwerkfunktionen; dank der Überschneidungen mit dem normalen .NET Framework lässt sich Desktop-Code rasch weiterverwenden.

Natürlich bietet Microsoft auch diverse kommerzielle Embedded-Betriebssysteme an. Neben dem hinreichend bekannten Windows CE liefern die Redmonder auch eingebettete Varianten der Desktop-Plattformen: Sie unterscheiden sich oft nur minimal vom Funktionsumfang des normalen PC-Systems.

Fazit

Embedded-Entwicklung bleibt nicht zuletzt deshalb interessant, weil es keine universell gültige Lösung gibt. Jedes der hier vorgestellten Betriebssysteme hat seine Lebensberechtigung: Die Bedürfnisse der automatischen Abstandsregelung eines mit 200 km/h fahrenden Autos unterscheiden sich radikal von denen eines smarten Kühlschranks.

Als zusätzliche Erschwernis gilt, dass ein suboptimaler Ansatz seine „Zähne" oft erst nach einiger Zeit zeigt. Primitive Echtzeitbetriebssysteme laufen schnell: Der Aufwand für die Rückportierung benötigter Funktionen nimmt aber sehr viel Zeit in Anspruch.

Ein Werbefachmann postulierte vor vielen Jahren, dass universelles Interesse die effizienteste Vorbereitung für das Erstellen kreativer Anzeigensujets sei. Wer die Welt gesehen hat, kann beim Lösen seiner Probleme auf einen reichen Erfahrungsschatz zurückgreifen.

Schon aus diesem Grund lohnt sich die Beschäftigung mit den hier vorgestellten Ansätzen – man weiß nie, wann sich einmal erworbenes Wissen auszahlt. (avr)

Tam Hanna

ist Gründer der Tamoggemon Holding k.s. und beschäftigt sich seit 2004 mit der Entwicklung und Anwendung mobiler Geräte.

Onlinequellen

[a] „Smartphones and Beyond" von David Wood
smartphonesandbeyond.com/

[b] FAQ von uClinux
www.uclinux.org/pub/uClinux/FAQ.shtml#2-5

[c] Vortrag von Thomas Petazzoni
elinux.org/images/9/9e/Buildroot2.pdf

[d] OpenEmbedded
www.openembedded.org/wiki/Main_Page

Distributionen

Angstrom
www.angstrom-distribution.org

Gentoo Linux
www.gentoo.de

LFS – Linux from Scratch
www.linuxfromscratch.org

LTIB – Linux Target Image Builder
ltib.org

OpenWrt
openwrt.org

Poky
www.pokylinux.org

Qt Enterprise Embedded
qt-project.org

Scratchbox
www.scratchbox.org

uClinux
www.uclinux.org

Yocto
www.yoctoproject.org

Alle Links: www.ix.de/ix1502036