Linux in embedded systems – Realtime-Linux

Karim Andreas Siebenrok

Proseminar – Linux

Abteilung für Verteilte-Systeme
Universität Ulm

ABSTRACT

In den letzten Jahren haben embedded systems sowohl im industriellen Bereich, als auch bei privaten Anwendungen immer mehr an Bedeutung gewonnen. Aufgrund der günstigen Lizenzbedingungen und insbesondere wegen der Open Source Idee wird Linux, das in Netzwerken überhaupt nicht mehr wegzudenken ist, mit zunehmender Verbreitung auch für den embedded Markt immer interessanter. Im industriellen Bereich spielt die Echtzeitfähigkeit der embedded systems eine große Rolle. Auch hier bietet Linux sehr komfortable und leistungsfähige Lösungen. Besonders hervorgetan hat sich hier Realtime-Linux, das im folgenden näher beschrieben werden soll. Aber auch konkrete Beispiele zur aktuellen Entwicklung werden nicht zu kurz kommen.

1. Einleitung

Diese Arbeit, ist eine Ausarbeitung eines Vortrages, den ich anlässlich des Proseminars – Linux - im Sommersemester 2000 gehalten habe.

Was ist eigentlich ein eingebettetes System? Je nach Anwendungsgebiet gibt es auf diese Frage verschiedene Antworten. Deshalb werde ich hier nur den Versuch einer Definition wagen. „Ein eingebettetes System interagiert mit seiner meist elektromechanischen Umwelt. Dabei ist kein menschlicher Benutzer vonnöten, oder jener muss keinerlei Kenntnisse über die technischen Innereien des eingebetteten Systems mit sich bringen um das Gesamtsystem bedienen zu können“ [1]. In den letzten Jahren ist der Markt für eingebettete Systeme explosionsartig gewachsen. Man findet Sie in Autos, Waschmaschinen, Videorekordern, Mobilfunk-Telefonen und natürlich auch im industriellen Bereich. Nachdem Linux seine Tauglichkeit im Netzwerkbereich bewiesen hat, hält es nun auch Einzug auf dem expandierenden embedded Markt. Eine wichtige Rolle für die Verbreitung spielt die einfach zu realisierende Echtzeitfähigkeit unter Linux. Im industriellen Bereich hat sich hier Realtime-Linux bereits mehrfach bewährt.

In der folgenden Ausarbeitung werde ich mich zunächst mit Linux in embedded systems allgemein beschäftigen. Danach wird die Betriebssystemerweiterung Realtime-Linux genauer beschrieben. Und zum Schluss werde ich anhand konkreter Beispiele zeigen, in wie weit die eingebetteten Systeme schon in unser Leben vorgedrungen sind

2. Linux in embedded systems

2.1 Warum Linux?

Zunächst stellt sich die Frage, warum gerade Linux dazu prädestiniert sein soll in eingebetteten Systemen eingesetzt zu werden. Natürlich spielen die Kosten eine große Rolle. Da embedded systems in sehr großen Stückzahlen produziert werden, sind schon geringe Lizenzkosten ein großer Finanzfaktor für den Hersteller. Ein weiterer Punkt ist die Zuverlässigkeit des Systems, da auch hier Ausfallzeiten schnell sehr teuer werden können. Die Systeme sind viele Jahre im Einsatz und auftretende Bugs sollten ohne größere Schwierigkeiten beseitigt werden können. Da ist es natürlich besonders vorteilhaft, wenn der gesamte Source zu Verfügung steht. Linux ist wohl im Moment eines der stabilsten und zuverlässigsten Betriebssysteme auf dem Markt. Durch die open source Verbreitung ist es auch besonders flexibel, wenn es darum geht, ein bestehendes System auf ein anderes zu portieren. Es würde viel Zeit und Geld kosten, sich sämtliche Funktionen mühsam zu erschließen und selbst zu programmieren. Weil embedded systems meist klein und billig sein sollen, werden häufig nicht mehr ganz aktuelle PC-Komponenten zu ihrer Herstellung verwendet. Das Betriebssystem sollte dann alles aus dieser „low-power“ Hardware herausholen. Auch hier sticht Linux, im Gegensatz zu anderen Betriebsystemen, durch seinen geringen Sourcenverbrauch hervor.

2.2 Die Grenzen

Selbst Linux ist nicht auf jeder Hardware lauffähig. Gewisse Mindestanforderungen sollten schon gegeben sein, damit das System einwandfrei funktioniert. Man spricht in der Regel von einem Bedarf von 2 MB RAM, 2 MB Flash und einem 32 Bit Mikroprozessor. Natürlich gibt es keine Regel ohne Ausnahme, es gibt bereits Linux Kernel die in 80 KB Speicher Platz finden und damit theoretisch nur 256 KB RAM benötigen würden.

Linux kann selbstverständlich selbst bei sehr kleinen Zielsystemen als Entwicklungsumgebung eingesetzt werden, hierfür stehen sehr viele, frei verfügbare, Softwarewerkzeuge zur Verfügung. Oft kann sogar das Testen der Software auf dem Zielsystem ausbleiben, da Linux die Hardware sehr gut emulieren kann. Das spart Entwicklungszeit und damit viel Geld. Eine weitere Schwäche des Standardlinux ist seine nicht vorhandene harte Echtzeitfähigkeit. Zwar übertrifft die implementierte weiche Echtzeitunterstützung die anderer Betriebssysteme bei Weitem, aber für den industriellen Einsatz müssen Interrupt Antwortzeiten von wenigen Mikrosekunden gewährleistet werden. Auch für dieses Problem gibt es unter Linux eine Lösung, die sog. Betriebssystemerweiterungen. Hier hat Realtime-Linux (RT-Linux) bereits seine Industrietauglichkeit bewiesen. Im Folgenden werde ich RT-Linux ausführlich behandeln.

3. Realtime-Linux

3.1 Allgemeines

Das Ziel eines normalen Betriebssystems ist es, die zur Verfügung stehende Rechenzeit möglichst gleichmäßig auf alle Prozesse zu verteilen. Dagegen müssen Echtzeitbetriebssysteme in der Lage sein, Anweisungen zu genau definierten Zeitpunkten auszuführen. Anwendungsbereiche sind zum Beispiel die Steuerungstechnik, Automatisierung und der Multi Media Bereich.

Linux genügt, wie die meisten Unix-Systeme, dem POSIX-Realtime Standard. POSIX, das "Portable Operating System Interface", ist ein Dokument, das von der IEEE erarbeitet und von ANSI und ISO standardisiert wurde. Aufgabe des POSIX Standards ist die Portabilität von Applikationen auf Ebene des Source Codes. Es werden also eindeutige Schnittstellen in Form von Funktionen definiert, die ein Betriebssystem bereitstellen muss, um diesen Standards zu genügen, so dass ein POSIX konform geschriebener Code auf allen POSIX Betriebssystemen kompilierbar ist. Allerdings definiert dieser nur die weiche Echtzeit und garantiert den geforderten zeitliche Determinismus nicht.

Wenn das Einhalten der Echtzeit statisch definiert ist, spricht man von weicher Echtzeit. Ein gutes Beispiel hierfür ist ein Videokonferenzsystem. Fehlt hier ein Bild, stört das zwar die Performance, aber weitere Konsequenzen sind nicht zu befürchten. Bei der harten Echtzeit muss das Einhalten von Zeitpunkten garantiert werden und einzelne Zeitpunkte dürfen keinesfalls verpasst werden.

Mit dem Kernel-Patch RT-Linux können diese harten Echtzeitbedingungen eingehalten werden. Es wurde ein eigenes, dediziertes Echtzeitbetriebssystem, das unter den normalen Linux Kernel gesetzt wird, geschaffen. Dieser Betriebssystemunterbau lässt Linux als einen sogenannten Realtime Task mit niedrigster Priorität laufen. An dieser Stelle wird schon deutlich, dass parallel weitere RT-Tasks laufen können, die von einem eigenen Scheduler verwaltet werden. Das Betriebssystem Linux läuft nur noch im Hintergrund und bearbeitet die Applikationen, denen die weichen Echtzeitbedingung genügen, wohingegen RT-Linux die Anteile übernimmt, die auf harte Echtzeit setzen. RT-Linux läuft auch auf Multiprozessormaschinen, steht aber nur für die x386er Architektur bereit.

3.2 Architektur

Der Aufbau von RT-Linux ist Modular, es besteht aus folgenden Modulen:

?? RT-Linux Kern Modul
?? Scheduling Modul
?? Kommunikations- Modul
?? Modul mit Synchronisations- und Interprozess-kommunikations-primitiven
?? Modul, durch das ausgewählte Parameter des Echtzeitsystems aus dem Gastbetriebssystem heraus verändert werden können

Die Module werden in Form von Objektdateien, die zur Laufzeit an den Betriebssystemkern gebunden werden können, realisiert. Eine Ausnahme macht hier das Kern Modul, dieses Modul wird in Form eines Kernel-Patches geliefert. Dieser verlangt das erneute kompilieren des Kernels, deshalb wird er bereits beim Hochfahren aktiviert. Ansonsten kann RT-Linux zur Laufzeit in beliebiger Weise skaliert werden.

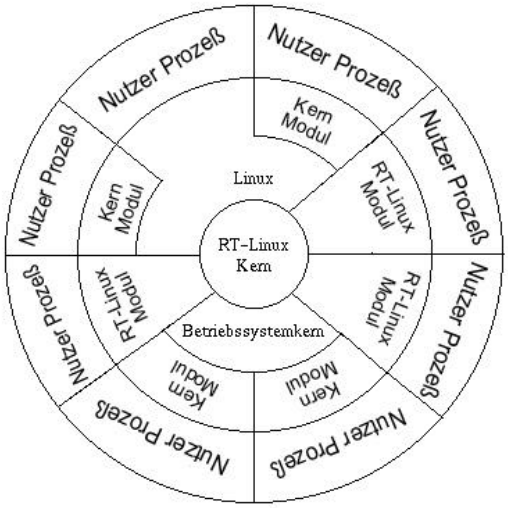

Abb. 1: Architektur des Echtzeitbetriebssystems RT-Linux

Die Architektur erläutert an Abbildung 1:

Der in der Mitte liegende RT-Linux Kern stellt die grundlegenden Funktionen zur Realisierung des RT-Betriebssystems zur Verfügung. Er kontrolliert den um ihn liegenden Kern des Gastbetriebssystems Linux. Die nächste Schale enthält die Module, durch die beide Systeme erweitert werden können. Hierbei kann man zwei Arten von dynamisch ladbaren Modulen unterscheiden. Zunächst die Kern Module, das sind normale Module des Gastbetriebssystems, diese interagieren nicht mit dem Echtzeitkern, wie zum Beispiel die Netzwerktreiber. Die zweite Art sind die RT-Linux Module, diese interagieren mit dem Echtzeitkern. Hier werden Erweiterungen des RT-Systems und dessen Anwendungen realisiert. In der äußersten Schale befinden sich die Nutzerprozesse, hier sind die nicht zeitkritischen Komponenten des Gesamtsystems realisiert.

3.3 Task-Typen

Bei der Implementierung von RT-Linux werden zwei verschiedene Task-Typen verwendet. Einerseits die sporadischen Tasks, sie werden durch externe Interrupts ausgelöst, entsprechen also den herkömmlichen Interruptroutinen. Sie haben alle dieselbe höchste Priorität im Gesamtsystem. Die Interruptvektor-Tabelle von Linux wird im RT-Kern emuliert, das hat den Vorteil, dass die Interruptroutinen des Betriebssystems unverändert bleiben und eine Anpassung der vorhandenen Treiber nicht nötig ist. Tritt nun ein Interrupt auf, wird geprüft, ob er durch einen sporadischen Task zu bearbeiten ist. Falls die der Fall ist, wird dieser ausgeführt, andernfalls wird der Interrupt in der Emulation als ausstehend gekennzeichnet und die Interruptverarbeitung des Gastbetriebssystems ausgelöst.

Andererseits die zyklischen Tasks, diese werden periodisch ausgeführt und sind mit einer Priorität versehen. Gleichwertige Interrupts werden nach dem Fifo Algorithmus zur Ausführung gebracht. Die Ausführung eines zyklischen Tasks wird durch einen Zeitgeber-Interrupt ausgelöst. Das Gastbetriebssystem wird als Task mit niedrigster Priorität verwaltet. Erst wenn alle Tasks höherer Priorität abgearbeitet wurden, wird Linux der Prozessor zugeteilt. Das Gastbetriebssystem bleibt unverändert, da es als Task des Echtzeitsystems betrachtet wird.

3.4 Kommunikation

3.4.1 Inter-Task Kommunikation

Zur Kommunikation und Synchronisation zwischen RT-Tasks stehen ein Botschafts-System, globale Variablen und Semaphoren zur Verfügung. Diese sind in einem Kernel Modul implementiert, welches man unter /EPP/ findet

3.4.2 Kommunikation mit Linux Prozessen

Zur Kommunikation mit Linux Prozessen stehen sogenannte RT-Fifo's zur Verfügung, über die binäre Datenströme ausgetauscht werden. Für den normalen Linux Prozess repräsentiert sich ein RT-Fifo als Charakter Device. Das bedeutet er kann synchron und asynchron in den RT-Fifo schreiben und synchron, asynchron und zeitgesteuert daraus lesen. RT-Linux greift asynchron auf ein RT-Fifo zu, da Echtzeitaufgaben nicht durch unzeitkritische Aufgaben blockiert werden dürfen. Schreibt ein Prozess in einen RT-Fifo, wird eine Prozedur abgearbeitet. Durch diese Prozedur können die Daten auf Realtime Seite gelesen werden. Es handelt sich also auf Seiten des Echtzeitsystems um eine ereignisorientierte Benachrichtigung über das Vorhandensein von zu lesenden Daten.

3.4.3 Export von ausgewählten Parametern

Die Parameter und Zustände von RT-Linux sind nicht im Benutzeradressraum sichtbar, da es sich um ein eigenständiges System handelt. Das heißt, dass die Daten nicht so ohne weiteres verändert werden können. Unter Linux gibt es das sog. /proc – Dateisystem, das Informationen über den Betriebssystemkern und Prozesse bereitstellt. Auf dieser Basis wurde auch in RT-Linux eine Schnittstelle implementiert, mit deren Hilfe die Parameter des Realtime-Betriebssystem exportiert und verändert werden können. Eine weitere Aufgabe dieser Schnittstelle ist es die interne Zeitrechnung der zwei Kerne ineinander umzurechnen. RT-Linux benutzt Ticks, mit einer Auflösung von 1193180 RT-Ticks pro Sekunde. Dagegen verwendet Linux die seit dem 1.1.1970 vergangenen Sekunden und Mikrosekunden zur Zeitberechnung.

Das sollte es eigentlich zu RT-Linux gewesen sein, mir ist klar, das meine Ausarbeitung zu diesem Thema nur einen kleinen Einblick in das komplexe System liefert. Auch sollte man erwähnen das jegliche Software die RT-Linux genügen soll, dazu eigens programmiert werden muß.

4. Speicher in embedded systems

In den nun folgenden Teilen meiner Ausarbeitung will ich mich nicht mehr auf die Softwaregrundlagen beschränken, sondern konkrete aktuelle Projekte und Entwicklungen vorstellen.

Auch embedded systems benötigen einen gewissen Speicherplatz für das Betriebssystem und die darauf laufenden

Programme. Diskettenlaufwerke und Festplatten sind nicht unbedingt für den harten Industrieeinsatz geeignet, da sie sehr anfällig für Erschütterungen, Staub und Temperaturschwankungen sind. Der Ausweg für die embedded systems sind sogenannte Flashdisks. Dabei handelt es sich um Permanentspeicher ohne verschleißende bewegliche Teile, mit logischen oder physikalischen Festplattenschnittstellen. Allerdings haben diese Bauteile auch einen entscheidenden Nachteil, sie vertragen selten mehr als 10000 bis 100000 Schreibzyklen. Es empfiehlt sich deshalb nicht Logdateien oder Swap-Partitionen darauf anzulegen. Zwei Systeme haben sich im Bereich embedded Linux durchgesetzt. Der erste Hersteller heißt SanDisk. Er produziert Compaktflashkarten, Flashchips und einen Festplattenersatz mit dazugehörigen Mikrokontrollern in verschieden Größen von 2 - 340 MB. Der Vorteil besteht darin, dass ein Zielsystem auf einer gewöhnlichen Festplatte erstellt werden kann, um dann bei der Serienproduktion durch eine SanDisk mit IDE-Schnittstelle ersetzt zu werden. Der Entwickler muss sich nicht um die Treiber kümmern, dafür setzt das System einen unter Linux nutzbaren IDE Kontroller voraus.

Der zweite Hersteller ist M-Systems, dessen Flashspeicher besitzen einen JEDEC-Kompatiblen Anschluss, diese lassen sich über ein Prozessor Interface, wie einen RAM Baustein, adressieren. Diese Chips enthalten ein Festplattenkontroller-BIOS für x86-kompatible Systeme. Da der Zugriff über die Biosfunktionen unter Linux nicht funktioniert, benötigt man einen speziellen Kernel-Treiber.

Da beide Hersteller auch für private Anwendungen wie Speicher für Digitalkameras und ähnliches produzieren, werden sich die Preise auch in Zukunft im Rahmen halten.

5. Aktuelle Beispiele

5.1 uClinux

Bei uClinux handelt es sich um ein Embedded Linux/Microcontroller Project. uClinux ist ein Abkömmling von Linux 2.0. Es wurde speziell für Mikrocontroller ohne Memory Management Unit entwickelt. Der Code wurde verändert oder neu geschrieben, um den Kernel möglichst klein zu halten. Auf der Strecke blieb deshalb die Multitasking Fähigkeit, aber für die meisten Anwendungen wird diese nicht benötigt. Erhalten blieben die wichtigsten Grundeigenschaften von Linux, wie Stabilität, Netzwerkfähigkeit und der Dateisystem-Support. Unterstütz werden TCP/IP und weitere Netzwerkprotokolle. Der Hersteller schreibt: „uClinux is an internet-ready OS, perfect for embedded system“. An Dateisystemen können u.a. NFS, ext2, MS-DOS, und FAT 16/32 verwendet werden. Das alles findet in einem Kernel der Größe < 512 KB Platz, mit Tools wächst das System auf 900 KB an.

Abb. 2: Microcontroller Project: uCsimm [4]

Es gibt auch erste Hardware für dieses System. Der sog. uCsimm wurde speziell für uClinux entwickelt. Er hat die Größe und Form eines Standard SIMM mit 30 Pins und wurde bereits vom Webserver bis zum programmierbaren Logik Controller eingesetzt.

In Abbildung 2 kann man sehr schön die Größe des uCsimms sehen.

Hier finden ein
? ? Dragon Ball Mikrocontroller mit 16 MHz,
? ? 2 MB Flash ROM,
? ? 8 MB DRAM,
? ? ein LCD Panel Driver (320x240),
? ? eine RS 232 Unterstützung und
? ? ein 10 Base-T Ethernet Controller Chip Platz.

5.2 Touchphone

Das Linux auch in erfolgreichen Konsumerprodukten eingesetzt werden kann, zeigt ein italienischer Elektronikhersteller. Er präsentiert mit Touchphone ein komplett über Touchscreen bedienbares Telefon. Mit diesem kann man Nummern speichern, Faxe und eMails erstellen, senden und empfangen, sowie Notizen speichern. In Planung ist ein Version mit einem integrierten Webbrowser.

Hardwaremäßig basiert das System auf
? ? einem 386er Prozessor,
? ? 8 MB Speicher,
? ? einer kleinen Festplatte,
? ? einer LCD-Anzeige,
? ? einem Touchscreen, das über eine serielle Schnittstelle mit dem System verbunden ist.

Abb. 3: Touchphone [5]

Das Telefon läuft mit einer modifizierten, sehr reduzierten Version von Linux Red Hat 4.2 . Die Faxfunktionen sind von den open source Systemem efax und vgetty übernommen worden, wobei ein paar Änderungen für das embedded system vorgenommen werden mussten. Die Grafik wurde mit der Bibliothek SVGAlib realisiert. Der Grafikeditor, für die Notizen und das Fax, wurde speziell für das Telefon geschrieben. Der Rest des Systems wurde mit einfachen Shell Skripts realisiert, die auf touchscreen-events reagieren.

5.3 empeg car

empeg car ist ein mp3-Player fürs Auto und zu Hause. Der Upload von mp3-files ist über den PC per USB, Seriell oder Ethernet möglich.

Die eingebaute Hardware besteht aus:

? ? einem Intel StrongARM Prozessor mit 220 MHz,
? ? 1 MB Flash memory,
? ? 12 MB RAM,
? ? zwei Festplatten mit je bis zu 18 GB Kapazität,
? ? Serieller, USB und Ethernet Anschluss.

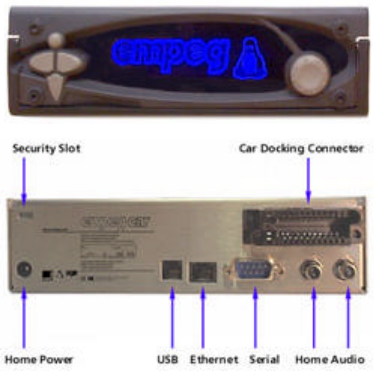

Abb. 4: empeg car (Vorder- / Hinteransicht) [6]

ARM [7] ist ein amerikanischer RISC Prozessor Entwickler. Er entwirft Prozessoren für alle großen Hersteller, wie Intel oder AMD, die speziell für die mobile Kommunikation, Handhelds, Multi Media und embedded Lösungen geeignet sind.

Als Software wurde eine modifizierte Version von ARM Linux [8] verwendet, die Playersoftware wurde neu geschrieben und die Visualisierung wurde in Assembler programmiert. ARM Linux ist eine Portierung von Linux speziell für die ARM Prozessoren. Es läuft zur Zeit auf 30 Verschiedenen Maschinen Varianten, wie komplette Computer, Netzwerkcomputern und evaluation boards.

5.4 DIMM PC / 486-I

Linux läuft mittlerweile auf fast allen 32-Bit Prozessoren, die eine Speicherverwaltungseinheit aufweisen. Embedded Linux wird aber bevorzugt auf x86 – kompatiblen Systemen eingesetzt, da eigene Platinenentwicklungen viel Know How, Zeit und Geld voraussetzen. Dagegen gibt es PC Komponenten und die dazugehörige Peripherie von der Stange. Des weiteren ist die PC-Linux Variante, die am weitesten entwickelte. Man kann hier auf sog. Mini Linux Distributionen, die auf eine Diskette passen, als Ausgangsbasis für embedded systems aufbauen.

Der DIMM-PC/486-I der Firma JUMPTEC [9] ist nur 40 mm x 68 mm groß und dennoch ein kompletter PC.

Abb. 5: DIMM-PC / 486-I (40 x 68 mm) [9]

Hardware:

? ? CPU: 486SX – 66 MHz
? ? System Bios
? ? Bis zu 16 MB RAM
? ? 16 MB IDE kompatiblen Flash Disk
? ? 2x serielle Ports, 1x paralleler Port
? ? Tastatur Controller
? ? Betrieb: 0° - 65°C
? ? Max Luftfeuchtigkeit 10 % - 90%

Für diesen PC gibt es speziell entwickelte Software. Zunächst EMJs White Dwarf Linux, das auf dem Linux Kernel 2.2.14 basiert. Es ist klein genug um in 8 MB Flash geladen zu werden und enthält u.a. eine grafische Installation, TCP/IP Tools (ftp, telnet, ping, hostname, ftpd, telnetd), mailx zum empfangen/senden von eMails, ein terminal emulation Packet, den lynx web Browser und einen Apache 1.3.11 Webserver.

BFAD-Erix ist die embedded-realtime-Linux Version für die Jumptec Hardware. Mit ihr kann der 486 DIMM-PC eine Reaktionszeit von bis zu 35 Mikrosekunden garantieren. Es basiert auf dem Linux Kernel 2.2.13 und die Realtime Erweiterung auf NMT-RT-Linux 2.0 . Die enthaltenen Hauptmodule sind außerdem, ein Apache Webserver, TCP/IP, PPP, Modem, Serial-Treiber und ein Editor. Das alles ist optimiert für den DIMM PC und findet auf 10 MB Flash platz.

5.5 Weitere Linux Systeme

5.5.1 V-MOD 201

Abb. 5: Digitaler Videorekorder V-MOD 201

Es gibt bereits Digitale Videorekorder, die auf Linux basieren und hauptsächlich aus Standardkomponenten bestehen. Hier findet sich die Linux-Installation auf einer SanDisk, zur Videoaufzeichnung dienen SCSI – Platten bzw. MO-Laufwerke. Die zeitkritische Einzelbildansteuerung professioneller externer Bildsignalquellen, werden durch RT-Linux realisiert. Damit war Steenback Digital [10] die erste Firma, welche die freie Echtzeiterweiterung für ein kommerzielles Produkt nutzte.

5.5.2 Flugsimulatoren

Abb. 7: Großraumflugsimulator

Die Großraumflugsimulatoren zur Pilotenausbildung der Firma CAE-Electronics [11] aus Kanada kosten pro Stück 22 Millionen Mark. Hier spielen ein paar Mark für das Betriebssystem natürlich keine Rolle, dafür aber dessen Zuverlässigkeit, da jede ausgefallene Trainingsstunde sehr hohe Kosten verursacht. Da dieselbe Software in einer abgespeckten Version selbst in Notebooks läuft, ist auch hier x86-Linux der kleinste gemeinsame Nenner. Die Echtzeiterweiterung für das System wurde von der Firma von einem früheren System auf Linux portiert.

5.5.3 Motorola Surfstation

Abb. 9: MobilCom Surfstation

Für Internet-Einsteiger ohne Computerkenntnisse gibt es die Motorola Surfstation [12], die an einen Fernseher angeschlossen werden kann. In der Box arbeitet eine auf FLASH-Linux basierende JAVA-Laufzeitumgebung von Infomatec. Allerdings merkt der Benutzer nichts von Linux, es handelt sich also um ein waschechtes embedded system. Auch hier besteht das System wieder größtenteils aus x86-Komponenten:

? ? Cyrix MediaGX 180 MHz Prozessor
? ? 64 MB SD-RAM, 10 ns
? ? SoundPort AD1819AJST
? ? IGS CyberPro 2010 Grafikboard
? ? M-Systems Disk On Chip

Die Surfstation wird über einen normalen Telefonanschluss an das Internet angebunden. Gesurft wird mit einem Netscape Browser.

5.5.4 Ben Hur

Abb. 10: Ben-hur bietet etliche Kommunikationsdienste

Ben-hur [13] ist ein auf Linux basierender Kommunikationsserver. Er bietet Dienste wie Routing, HTTP-Proxy, eMail, Fax-Versand/Empfang, Voice-Mail, ISDN Remote Access und Firewall an. In dem speziellen Gehäuse arbeitet ein handelsübliches ATX-Mainboard mit entsprechender Standardperipherie.

6. Zusammenfassung

Linux folgt der GNU Bewegung, die durch die Leistung mehrerer tausend Programmierer, qualitativ sehr hochwertige Software produziert. GNU wurde, genau wie Linux, am Anfang als Hobby von Software Anarchisten mit zu viel Zeit belächelt. Nachdem nun aber so manche Welt-Klasse Software auf dieser Basis entstanden ist, wird es ernst genommen. Was mit den Betriebssystemen passiert ist, wird auch mit den Realtime-Betriebssystemen passieren. RT-BS, die auf Linux basieren werden immer mehr in den Vordergrund rücken und sich nicht hinter high-end Produkten verstecken müssen. Letztendlich wird daraus eine Goldgrube für Entwickler von high-end embedded systems resultieren, die selbst mit ihrem billigen Realtime-Betriebssystem mehr Fähigkeiten haben werden, als die meisten teuren Produkte.

In den meisten Produkten die wir in den nächsten Jahren benutzen werden, werden nach und nach embedded systems Einzug halten. Oder wer hätte noch vor wenigen Jahren gedacht, das moderne Autos von Mikrochips gesteuert werden. Die gezeigten Beispiele sind natürlich nur ein sehr kleiner Ausschnitt von dem was schon heute möglich ist. In Zukunft werden wir wohl solche Systeme mit der gleichen Selbstverständlichkeit benutzen, wie heutige Geräte.

Die Zukunft von Linux in eingebetteten System sieht sehr rosig aus. Noch aber können sich nicht alle Firmen an den freien Support gewöhnen. Schließlich gibt es keine Garantie auf Hilfe. Wegen der gewaltig steigenden Nachfrage formiert sich nun aber auch kommerzieller Support. Zumeist mittelständische Unternehmen sehen in Systemen auf Linux Basis ihre Chancen. Auch wenn sie das nicht an die große Glocke hängen, wahrscheinlich um sich den Wettbewerbsvorteil zu erhalten.

7. Literaturverzeichnis

Die Idee zu meinem Referat und dieser Ausarbeitung, sowie die allgemeinen Informationen zu Linux in embedded systems stammen aus diesem Bericht:

[1] Embedded Linux von Bernhard Kuhn, Linux Magazin 01/2000 (S. 34 – 41)

Informationen zu Realtime Linux habe ich aus den folgenden Quellen erhalten:

[2] Einführung in die Echtzeiterweiterung RT-Linux – Zeitkritisch – von Till Christian Siering, Linux Magazin 03/1998

[3] Diplomarbeit von Sven Heursch: Latenzzeitmessungen des Betriebssystems Linux (Realtime Linux/KURT Linux) 07/1999

Informationen zu den einzelnen vorgestellten Produkten findet man unter:

[4] uClinux und uCsimm: http://www.uclinux.org

[5] Touchphone: http://lwn.net/1999/features/TouchPhone/

[6] empeg car: www.empeg.com

[7] ARM: http://www.arm.com

[8] ARM-Linux: http://www.arm.uk.linux.org

[9] JUMPTEC DIMM-PC: http://www.jumptec.de

[10] Digitalvideosysteme: http://www.steenbeckdigital.de

[11] CAE-Electronics: http://www.cae.ca

[12] Motorola: http://www.motorola.de

[13] Pyramid Computer: http://www.ben-hur.de