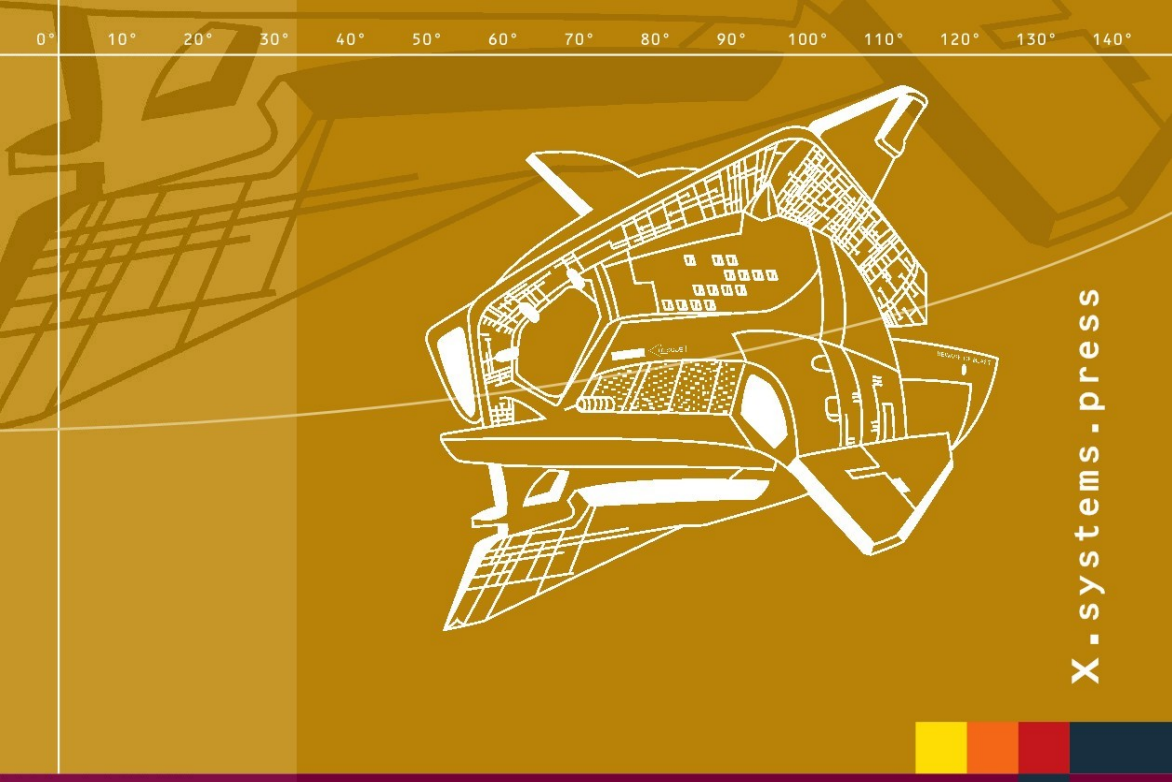

X.systems.press

Christoph Czernohous

Pervasive Linux

Springer

X.systems.press

X.systems.press ist eine praxisorientierte
Reihe zur Entwicklung und Administration von
Betriebssystemen, Netzwerken und Datenbanken.

Christoph Czernohous

Pervasive Linux

Basistechnologien, Softwareentwicklung, Werkzeuge

Springer

Christoph Czernohous
Silcherplatz 5
71106 Magstadt
Deutschland
cc@de.ibm.com

ISSN 1611-8618
ISBN 978-3-540-20940-9 e-ISBN 978-3-540-68426-8
DOI 10.1007/978-3-540-68426-8
Springer Heidelberg Dordrecht London New York

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Einbandentwurf: KünkelLopka GmbH, Heidelberg

Gedruckt auf säurefreiem Papier

Springer ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.com)

Für Bettina

Vorwort

Pervasive heißt auf Deutsch so viel wie durchdringend, allgegenwärtig, sich überall ausbreitend. Und genau das geschieht zur Zeit mit der Rechenleistung von Computern aller Art. Es gibt kaum noch technische Alltagsgegenstände, die nicht irgendeine Form von Mikroprozessor beinhalten, die diese Geräte steuern und zusätzliche Funktionalität bereitstellen, von der Waschmaschine über das Auto zum Mobiltelefon. Wir werden jeden Tag fast überall von Rechenleistung umgeben, sie ist allgegenwärtig und durchdringt unser Leben, auch wenn wir es manchmal gar nicht merken.

Zur gleichen Zeit gibt es einen weiteren Trend mit starkem Wachstumspotenzial: Open Source. Die Verbreitung von Software, deren Quellen für jedermann einsehbar sind, hat in den letzten Jahren stark zugenommen, allem voran das offene Betriebssystem Linux. Ausgehend von einem Betriebssystem für Personal Computer, hat sich Linux in der Zwischenzeit ein festes Standbein im Server-Umfeld geschaffen und wird auch in eingebetteten Systemen, der Grundlage für Pervasive Computing, immer beliebter.

Der Kombination dieser beiden Wachstumsperspektiven, Pervasive Computing und Linux, gilt das vorliegende Buch. Es richtet sich an Produkt- und Softwareentwickler, die Linux in eingebetteten Systemen einsetzen möchten, setzt dabei aber Grundkenntnisse über die Benutzung und Administration von Linux, sowie Programmierkenntnisse voraus. Auch die Vor- und Nachteile von Open Source im allgemeinen und Linux im besonderen werden nicht erörtert. Dieses Buch nähert sich dem Thema Pervasive Linux von der Anwendungs- bzw. Softwareentwicklungsseite. Es geht hauptsächlich darum, den Einstieg in die Entwicklung für Pervasive Linux zu erleichtern. Angesichts der Vielfalt an verfügbarer Hard- und Software in diesem Bereich sei für das vertiefende Studium einzelner Themen auf die jeweils spezialisierte Literatur verwiesen. Nicht Gegenstand des Buches ist die Portierung von Linux selbst auf zusätzliche Hardware-Plattformen. Aus den selben Gründen werden auch nicht sämtliche Parameter diverser Kommandozeilenbefehle und Konfigurationsdateien aufgelistet.

Oft ist die grösste Hürde, überhaupt den Einstieg in Pervasive Linux zu finden. Der Aufbau einer funktionsfähigen Entwicklungsumgebung kann sehr frustrierend sein. Oftmals sind Informationen auf den relevanten Webseiten nicht sofort auffindbar. Dieses Buch soll den Einstieg in die Entwicklung von Linux-Anwendungen für

das Pervasive Computing erleichtern, indem es jeweils mit Beispielen die Grundlagen für die benötigten Schritte erläutert. Dabei werden auch Werkzeuge vorgestellt, die nicht spezifisch für Pervasive Linux sind, aber bei der Benutzung und Entwicklung häufig verwendet werden.

Es wird erklärt, welche Besonderheiten beim Einsatz von Linux in eingebetteten Systemen zu beachten sind, welche Code- und Informationsquellen es gibt, welche Aufgaben ein eingebettetes Linuxsystem zu erfüllen hat und aus welchen Bestandteilen es besteht. Die Softwareentwicklung wird mit Hilfe gängiger Werkzeuge beispielhaft dargestellt.

Die Beispiele orientieren sich an Linux-Systemen, die auf tragbaren digitalen persönlichen Organisationshilfen (engl.: Personal Digital Assistants – PDA, bzw. Handhelds), mobilen Internetgeräten (engl.: Mobile Internet Device – MID), Tablett-Computern und Mobiltelefonen bzw. so genannten Smartphones einsetzbar sind, da diese Art Hardware-Grundlage als Massenware relativ unkompliziert und günstig für jedermann erhältlich ist und es für diese Systeme oft eine breite Linux-Unterstützung gibt. Die Konzepte lassen sich aber auch auf andere eingebettete Systeme übertragen.

Das Motto des Buches lässt ich also wie folgt formulieren: *Linux macht Spass. Pervasive Linux macht den Spass allgegenwärtig*.

Danksagung

Mein Dank gilt meiner Frau Bettina und meinem Redakteur Hermann Engesser vom Springer Verlag für die schier endlose Geduld und Unterstützung bei diesem Buchprojekt.

Ausserdem möchte ich mich bei Silke und Clemens van Dinther, Jan Burchhardt und Jens Kretzschmar fürs Korrekturlesen des Manuskripts und die konstruktiven Anregungen bedanken.

Magstadt, Deutschland Christoph Czernohous
Juni 2011

Inhaltsverzeichnis

Teil I
Systembestandteile

Kapitel 1
Pervasive Computing

Bei *Pervasive Computing* geht es hauptsächlich um kleine, verteilte, auf unterschiedlichster Hardware basierende Computer, die auf diverse Weise mit anderen Geräten oder dem Menschen kommunizieren. Oft ist auch von eingebetteten Systemen (engl.: Embedded Systems) oder so genannter Rechenallgegenwart (engl.: Ubiquitous Computing) die Rede.

Systeme des Pervasive Computing besitzen besondere Eigenschaften, die es bei deren Einsatz und insbesondere bei der Software-Entwicklung für diese Systeme zu berücksichtigen gilt. Besonders für Anwender und Entwickler, die aus dem Bereich des Desktop Computing kommen, gibt es einige zusätzliche Aspekte zu beachten.

1.1 Ressourcenbeschränkung

Wie in allen Bereichen der Informationstechnik steigt auch die Leistungsfähigkeit von eingebetteten Geräten immer weiter, während gleichzeitig die physikalischen Bauteile kleiner und integrierter werden. Nichtsdestotrotz sind die in diesem Umfeld zur Verfügung stehenden Betriebsmittel und Ressourcen im Vergleich zu gängigen Arbeitsplatzrechnern deutlich beschränkt. Das betrifft in erster Linie die Leistungsfähigkeit der verwendeten Prozessoren und den verfügbaren Speicher (sowohl Arbeits- als auch Ablagespeicher), aber auch Kommunikations- und Netzwerkverbindungen sowie die Energieversorgung.

All diese Aspekte spielen im Pervasive Computing eine zentrale Rolle und müssen bei der Entwicklung solcher Systeme bedacht werden.

1.1.1 Speicherverwaltung

Für die Verwaltung des Zugriffs auf Arbeitsspeicher besitzen Prozessoren üblicherweise eine Speicherverwaltungseinheit (engl.: Memory Management Unit – MMU). Die Aufgabe einer MMU ist es, laufenden Prozessen Arbeitsspeicher zur Verfügung zu stellen, der für einen konkreten Prozess exklusiv reserviert ist, so dass andere Prozesse den Inhalt des Speicherbereiches nicht auslesen oder verändern können.

C. Czernohous, *Pervasive Linux*, X.systems.press,
DOI 10.1007/978-3-540-68426-8_1, © Springer-Verlag Berlin Heidelberg 2012

Für das in einem Prozess aufgeführte Programm erscheint der so bereitgestellte Speicherbereich als ein kontinuierlicher Block. Tatsächlich kann sich der Speicherbereich aber über unterschiedlichste unzusammenhängende Speicheradressen und auch ausgelagerten Speicher, z. B. auf einer Festplatte, erstrecken.

Aus Platz- und Energiegesichtspunkten wird bei manchen Prozessoren auf eine MMU verzichtet. Wenn auf einem solchen System ein Betriebssystem betrieben werden soll, das nebenläufigen Prozessen exklusiven Speicher zur Verfügung stellt, wie das bei Linux der Fall ist, muss die Speicherverwaltungseinheit im Betriebssystem implementiert sein. Abschn. 3.2.3 beschäftigt sich mit der Unterstützung von Linux für solche Systeme.

1.1.2 Grafische Benutzerschnittstellen

Während man im Umfeld von Desktop-Computern viele Jahre lang von immer grösseren Bildschirmen mit entsprechender Auflösung und einheitlichem Seitenverhältnis ausging, erfordert die Entwicklung von Software für Geräte des Pervasive Computing ein Umdenken und die Berücksichtigung einiger Besonderheiten, wie z. B. variable Bildschirmgrössen und unterschiedliche Interaktionsmöglichkeiten mit dem Benutzer.

Für Computerprogramme, die sich in grafische Benutzeroberflächen (engl.: Graphical User Interface – GUI) integrieren, sollte es selbstverständlich sein, auf unterschiedliche und sich potenziell ändernde Umgebungen flexibel reagieren zu können. Diese Regel verschärft sich für das Pervasive Computing insofern, als der verfügbare sichtbare Bereich meistens sehr beschränkt ist, und durch weitere eingeblendete Bereiche, wie z. B. eine virtuelle Tastatur, zusätzlich eingeschränkt wird.

Ein weiterer wichtiger Punkt, den es zu berücksichtigen gilt, sind die unterschiedlichen Wortlängen, die Texte in verschiedenen Sprachen aufweisen. Daher sollte bei der Gestaltung von Oberflächen nicht von den Dimensionen der primär unterstützten Sprache ausgegangen werden. Das gilt natürlich nicht nur für das Pervasive Computing, aufgrund der eingeschränkten Platzverhältnisse verschärft sich hier aber die Problematik. In Abschn. 6.8 wird auf die technischen Möglichkeiten eingegangen, Anwendungen für unterschiedliche Sprach- und Kulturkreise zu entwickeln.

1.1.2.1 Variable Bildschirmgrössen

Die Anzeigefläche, die für Programme zur Verfügung steht, variiert zwischen unterschiedlichen Geräten stark. Nicht nur das Platzangebot selbst, also die Grösse des Bildschirms, sondern auch das Seitenverhältnis in Kombination mit der Ausrichtung muss bei der Entwicklung beachtet werden. Moderne Geräte reagieren mit einem Bewegungssensor darauf, wie das Gerät momentan eingesetzt wird, also ob es waagerecht oder senkrecht gehalten wird und welche Seite oben ist. Entsprechend wird das Seitenverhältnis angepasst und Anwendungen müssen darauf reagieren. Anwendungen sollten möglichst wenig Annahmen über den verfügbaren Raum machen und flexibel auf Änderungen reagieren.

Da sich die Grösse von Fenstern und Dialogen durch Modifikation des Benutzers jederzeit ändern kann, und auch Anzeigeflächen unterschiedliche gross sein können, bedarf dies bei der Entwicklung besonderer Aufmerksamkeit. Insofern sollten Steuerelemente wie Auswahlknöpfe und Eingabefelder möglichst dynamisch innerhalb eines Fensters angeordnet werden.

1.1.2.2 Interaktion und Eingabemöglichkeiten

Mobile Geräte des Pervasive Computing sind häufig mit einem berührungsempfindlichen Bildschirm (engl.: Touch Screen) ausgestattet. In vielen Fällen wird dann auf eine separate Tastatur verzichtet und die Eingabe komplett über den Bildschirm gesteuert, oft mit Hilfe eines speziellen Stiftes, mit welchem sehr genaue Eingabepunkte möglich sind. Allerdings werden für die Eingabe im mobilen Umfeld häufig auch die Finger verwendet. Bei Anwendungen, die mit den Fingern bedient werden sollen, muss darauf geachtet werden, dass die angezeigten Steuerelemente ausreichend grossflächig sind, da die Positionierung mit den Fingern deutlich ungenauer als mit einem Spezialstift ist.

1.2 Grafik

Für die grafische Darstellung von Informationen gibt es die Möglichkeit der punktgenauen Wiedergabe (Bitmap-Grafik) oder der beschreibenden Visualisierung (Vektorgrafik).

1.2.1 Vektorgrafik

Bei Abbildungen auf Bildschirmen unterschiedlicher Grösse und Auflösung kann durch den Einsatz von Vektorgrafiken eine einheitliche Darstellung erreicht werden. Dabei wird die darzustellende Grafik abhängig von den Anzeigemöglichkeiten berechnet. Das hat zwar eine erhöhte Rechenzeit zur Folge, sorgt aber für gleichmässigere Visualisierung auf unterschiedlichen Geräten, was gerade bei den unterschiedlichen Anzeigeformaten mobiler Geräte wichtig ist. Ein weiterer Vorteil von Vektorgrafiken ist der in der Regel geringe Speicherbedarf der Bildinformationen im Vergleich zu pixelbasierten Lösungen.

1.2.2 Bitmap-Grafik

Bei Bitmap-Grafiken wird die Anzeigeinformation Punkt für Punkt in der Bilddatei abgelegt. Bitmap-Grafiken könnten daher sehr schnell geladen und angezeigt werden, besitzen aber immer die gleichen physikalischen Ausmasse und variieren daher in Grösse und Lesbarkeit in Abhängigkeit des eingesetzten Anzeigegerätes.

Wie in Abschn. 1.1.2.1 bereits beschrieben, ist das eine besondere Herausforderung für Systeme mit unterschiedlichen Darstellungsgrössen, der sich Projekte wie z. B. Enlightenment (Abschn. 3.6.6) angenommen haben. Da die Bildinformation für jeden Bildpunkt eindeutig abgelegt wird, steigt der Speicherbedarf proportional zur Bildgrösse.

1.3 Mobilität

Mobiltelefone, digitale Organisationshilfen für Adress- und Terminverwaltung (so genannte *Personal Digital Assistants – PDA*), MP3-Spieler etc. sind dem Pervasive Computing zuzurechnen. Derartige Geräte bleiben aufgrund ihres für sie vorgesehenen Einsatzzwecks nicht fest an einem Ort, sondern werden in sich laufend verändernden Umgebungen eingesetzt. Die Software auf solchen Geräten muss deutlich flexibler und fehlertoleranter z. B. mit einer unterbrochenen Netzwerkverbindung oder Stromversorgung oder sich ändernden Uhrzeiten und Zeitzonen umgehen können.

Dadurch erhöhen sich die Anforderungen bzgl. der Datensicherheit dieser Geräte.

Ein weiterer Aspekt der Mobilität dieser Geräte ist die Möglichkeit, den aktuellen Standort auf der Erde zu bestimmen. Auf diverse Weise ist es heute möglich, die aktuelle Position sinnvoll in einer Anwendung zu verarbeiten.

1.3.1 Kontextverarbeitung

Die zunehmende Vernetzung und die modernen Möglichkeiten im Bereich des Pervasive Computing erfordern eine stärkere Berücksichtigung des Anwendungskontextes, als es bei herkömmlichen Anwendungen der Fall ist.

Bei einem MP3-Spieler ist es gleichgültig, in welcher Zeitzone er betrieben wird. Bei einem PDA ist diese Information schon wichtiger, damit z. B. Termine der momentanen Zeitzone entsprechend angezeigt werden. Mit aktuellen Mitteln wie z. B. der so genannten Geolocation, also der Standortbestimmung, nimmt die Bedeutung des Anwendungskontextes noch zu. Beispielsweise kann über entsprechende Dienste der aktuelle Standort an bekannte Personen verteilt werden, und ein mobiles Gerät so zusätzlichen Nutzen bieten. Weitere Beispiele sind das Erkennen von Systemen in der Umgebung, mit denen potenziell kommuniziert werden kann. Z. B. kann sich ein mobiles Gerät automatisch mit so genannten WLAN-Hotspots (Abschn. 4.4) eines bestimmten Betreibers verbinden, um eine Internetverbindung aufzubauen oder über Bluetooth (Abschn. 4.6) zusätzliche Funktionalität bereitstellen.

Je mehr Informationen ein Gerät über die aktuelle Umgebung liefern kann, desto anspruchsvoller wird die Auswertung dieser Daten, aber desto höher ist auch der potenzielle zusätzliche Nutzen für den Anwender.

1.3.2 Datensicherheit

Die im Vergleich zur herkömmlichen Datenverarbeitung deutlich erhöhte Mobilität im Pervasive Computing erfordert es, ein besonderes Augenmerk auf Datensicherheit und der Vermeidung von Datenverlust zu legen. Datensicherheit hat diverse Aspekte. Einserseits geht es um Datenverlust im Allgemeinen, andererseits um die Sicherheit der gespeicherten Daten vor unbefugtem Zugriff.

1.3.2.1 Datenabgleich und Synchronisierung

Da bei mobilen Geräten der unterbrechungsfreie Betrieb oft nicht garantiert werden kann, z. B. aufgrund mangelnder Energieversorgung, sollten geänderte Nutzdaten so schnell wie möglich vom Hauptspeicher auf ein persistentes Speichermedium gesichert werden, um Datenverlust zu vermeiden.

Mobilgeräte sind aufgrund ihrer Grösse und ihres Einsatzzwecks in erhöhtem Masse verlustgefährdet, sei es durch Unachtsamkeit, Diebstal oder Zerstörung. Meistens können Hard- und Software in solchen Fällen relativ einfach ersetzt werden, das eigentlich Wertvolle, nämlich die lokal gespeicherten Daten, sind dabei deutlich aufwändiger, wenn überhaupt, wieder herzustellen. Es ist daher wichtig, eine tragfähige Strategie zur Erstellung von Sicherheitskopien (engl.: Backup) z. B. mittels Synchronisation zu etablieren.

Insbesondere mobile Geräte, selbst wenn sie wie PDAs oder Smartphones nicht auf den Abgleich ihrer Daten mit anderen, zentralen Systemen angewiesen sind, sollten ihren Datenbestand regelmässig mit einem Sicherungssystem synchronisieren. Wenn, wie im beim Pervasive Computing häufig der Fall, ein ständiger Datenaustausch nicht möglich ist, weil keine Netzwerkverbindung besteht, müssen bestimmte Daten zwischengespeichert und, sobald eine Netzverbindung vorhanden ist, mit der Bestandsdatenbank abgeglichen werden.

All das erfordert die Möglichkeit, Daten lokal persistent zu speichern, und bei der Verfügbarkeit einer Netzverbindung, diese mit dem zentralen Datenbestand abzugleichen. Gerade für moderne Web-Anwendungen (Abschn. 6.7.2) ist das eine besondere Herausforderung.

1.3.2.2 Zugriffskontrolle und Verschlüsselung

Ein weiterer Aspekt ist der unbefugte Zugriff auf Daten und Funktionalität.

Das Ausspähen von Daten, die über eine Netzwerkverbindung übertragen werden, oder der Zugriff auf Daten bei Verlust des Gerätes konfrontieren Anwender und Entwickler von Software für Pervasive Computing mit erhöhten Sicherheitsanforderungen.

Passwortschutz und geeignete Verschlüsselungsmechanismen können die Datensicherheit erhöhen.

Kapitel 2
Hardware

Die Hardware, die zusammen mit Linux zum Einsatz kommt – bzw. kommen kann – reicht von Grossrechnern, bis zu teilweise winzigen Geräten, dem Thema dieses Buches. Je kleiner die Systeme und je günstiger die Produktionskosten sind, desto grösser ist auch die Anzahl an Varianten, die von Linux unterstützt werden. Von Seiten der Hardware sind dem Einsatz von Linux fast keine Grenzen gesetzt.

Daher ist es wichtig, mit der breiten Palette der Hardware-Bestandteile vertraut zu sein, um deren Auswahl, Einsatz und Eigenschaften beurteilen zu können.

2.1 Prozessor

Im Umfeld des Pervasive Computing kommen häufig Prozessoren auf Grundlage der so genannten *Reduced Instruction Set Computer* (*RISC*) Architektur zum Einsatz. RISC Prozessoren haben im Gegensatz zu Prozessoren auf Basis eines komplexen Befehlsumfangs (engl.: *Complex Instruction Set Computer* (*CISC*)) einen stark reduzierten Befehlssatz, der wiederum sehr effizient implementiert ist. Dadurch wird die Prozessorstruktur einfach gehalten. Ein RISC Prozessor benötigt in der Regel deutlich weniger Energie als ein CISC Prozessor, was besonders bei mobilen Einheiten wie PDAs und Mobiltelefonen wichtig ist, da diese nicht über eine ständige Stromversorgung verfügen und in den meisten Fällen auf Batterie- oder Akkustrom angewiesen sind, mit dem es möglichst effizient umzugehen gilt.

Allerdings drängen inzwischen auch Prozessoren mit mehreren Rechenkernen in den Bereich des Pervasive Computing vor. Damit wird es möglich, Programme echt parallel auszuführen, was den Entwickler insbesondere dann vor neue Herausforderungen stellt, wenn diese Parallelität Einfluss auf das Programmverhalten zum Ausführungszeitpunkt hat.

Für RISC Prozessoren übersetzte Programme sind in Binärform bis auf die einfachsten Fälle immer größer als der für CISC Prozessoren entsprechend übersetzte Code, da die fehlenden Prozessorbefehle durch Software abgebildet werden müssen. Dies geht natürlich zu Lasten der ebenfalls eingeschränkten und daher kostbaren Ressource Speicherplatz. Der durch die RISC Architektur eingesparte Energie- und teilweise auch physikalische Platzverbrauch des Prozessors wiegt den etwas erhöhten Speicherbedarf der RISC Programme in der Regel auf.

C. Czernohous, *Pervasive Linux*, X.systems.press,
DOI 10.1007/978-3-540-68426-8_2, © Springer-Verlag Berlin Heidelberg 2012

Die für Pervasive Computing momentan am häufigsten eingesetzten Prozessoren basieren auf den RISC Architekturen PowerPC, Advanced RISC Machines (ARM) und MIPS, sowie der CISC-Architektur M68000.

2.2 Speicher

Pervasive Computing stellt auch an die einsetzbaren Speicheralternativen besondere Anforderungen.

2.2.1 Arbeitsspeicher

Arbeitsspeicher ist die Systemkomponente, aus der der Prozessor Daten und Befehle zu deren Verarbeitung erhält und in den die Ergebnisse nach Ausführung der Befehle abgelegt werden.

Der Hauptspeicher besteht in den meisten Fällen aus *Dynamic Random Access Memory* (*DRAM*) Speicherbausteinen. Die Speicherverwaltungseinheit des Prozessors kann auf den DRAM Speicherbereich beliebig lesend und schreibend zugreifen.

DRAM ist sehr schnell und eignet sich daher für den Einsatz als Hauptspeicherkomponente, die für eine effiziente Kommunikation mit der Prozessoreinheit über schnelle Antwortzeiten verfügen muss. DRAM Speicherbausteine benötigen eine regelmäßige Stromzufuhr in kurzen Intervallen, um den gespeicherten Datenbestand aufrechtzuerhalten. Sobald die Energiezufuhr unterbrochen wird, gehen die gespeicherten Daten verloren.

2.2.2 Permanenter Speicher

Um Daten über einen längeren Zeitraum und mögliche Unterbrechungen der Energiezufuhr hinweg zu speichern, werden permanente, so genannte nicht flüchtige Speicherformen benötigt. Für den Einsatz in mobilen Geräten haben sich diverse Varianten sogenannten Flash-Speichers etabliert. Aber auch besonders kleine, stromsparende und robuste Festplatten kommen zum Einsatz.

Um den Zugriff auf Speicher, insbesondere Flash-Speicher zu vereinheitlichen entwickelt das Projekt *Memory Technology Device* (*MTD*) *Subsystem for Linux* eine Abstraktionsschicht, so dass Gerätetreiber sich auf die Basiszugriffsfunktionen beschränken können. Im Internet ist das Projekt zu finden unter http://www.linux-mtd.infradead.org.

2.2.2.1 Flash

Flash Speicher ist die am häufigsten anzutreffende Speicherart für die dauerhafte Ablage von Daten in mobilen Geräten. Bei Flash handelt es sich um

Speicherbausteine, die ihren Inhalt auch ohne permanente Stromversorgung behalten. Sie beinhalten keine mechanischen Komponenten und verfügen über eine Zugriffsgeschwindigkeit, die mit gängigen Festplatten vergleichbar ist.

Die Speicherbereiche von Flash Speicher können nur begrenzt oft beschrieben werden, in der Regel einige tausend Mal. Das macht sie zu einer besonders wertvollen Komponente, die mit Bedacht eingesetzt werden muss. Insbesondere ist darauf zu achten, dass die Speichereinheiten, in die der verfügbare Gesamtspeicher eingeteilt ist, möglichst gleichmäßig beschrieben werden, um die Lebensdauer zu maximieren.

Aus diesem Grund ist es wichtig, dass die für Flash Speicher verwendeten Dateisysteme über sogenanntes *Wear Leveling* verfügen, das eine Verschleiss-Gleichverteilung der Schreibzugriffe über den vorhandenen Speicherbereich sicherstellt. Speziell für diesen Einsatzzweck wurden geeignete Dateisysteme entwickelt, die in in Abschn. 3.4 vorgestellt werden.

Flash Speicher wird in so genannte Löschblöcke (engl.: *Erase Blocks*) aufgeteilt, die üblicherweise deutlich grösser sind als die von Festplatten bekannte Blockgrösse. Eine Besonderheit von Flash Speicher ist es, dass der zu beschreibende Bereich zuerst gelöscht werden muss, bevor eine Schreiboperation möglich ist. Das Löschen eines Blocks erfolgt, indem allen Bits des Blocks der Wert Eins zugewiesen wird. Der technische Hintergrund dafür ist, dass die einzig mögliche Schreiboperation bei Flash-Speicher ist, den Wert einzelner Bits von eins auf null zu setzen. Das bedeutet, dass ein Block, damit er neu beschrieben werden kann, zuerst gelöscht werden muss, sobald mindestens ein Bit des Blocks, das nach der Schreiboperation den Wert eins annehmen soll, vor der Schreiboperation eine Null enthält.

Zur Zeit gibt es zwei Typen von Flash Speichern, die unterschiedliche technische und ökonomische Eigenschaften aufweisen, so genannte *NOR*- und *NAND* Speicherbausteine.

NAND Speicher ist sequenziell organisiert, d.h. ein Speicherblock wird nach dem anderen gelesen, ohne direkt auf einen bestimmten Speicherblock zugreifen zu können. Die Schreiboperationen für NAND Speicher sind sehr schnell. Im Gegensatz dazu kann auf NOR Speicher wahlfrei (engl.: random access) zugegriffen werden, allerdings sind Schreiboperationen im Vergleich zu NAND Speicher relativ langsam. NOR Speicherchips sind physikalisch grösser als entsprechender NAND Speicher und können nicht so oft wiederbeschrieben werden.

CompactFlash

Bei Speicher vom Typ *CompactFlash* (*CF*) handelt es sich um Flash-Speicher, der bereits mit einer Steuereinheit versehen ist und als Einheit mit dieser geliefert wird. Diese Steuereinheit sorgt unter anderem für das bereits angesprochene Wear Leveling, also die möglichst gleichmäßige Verteilung der Speicherzugriffe auf den bereitgestellen Gesamtspeicher der Einheit. Die entsprechenden Spezifikationen werden von der *CompactFlash Association* (*CFA*) entwickelt und auf der Seite http://www.compactflash.org zur Verfügung gestellt.