Study-Guide: Hardware und Systemarchitektur

In diesem Abschnitt geht es darum, verschiedene Einstellungen vorzunehmen, die als Grundvoraussetzung gelten, damit Linux überhaupt auf einem Computer arbeiten kann, bzw. damit eine bestimmte Hardware angesprochen werden kann. In der ersten Ausgabe der LPIC-Level 1 Prüfungen war dieses Thema zum zweiten Prüfungsteil zugeordnet, jetzt ist es beim ersten gelandet. In diesem Zusammenhang sind bestimmte Grundlagenkenntnisse von hardwarespezifischen Einstellungen notwendig, die sich sowohl auf die Einstellungen der Erweiterungskarten beziehen, als auch auf die Möglichkeiten, diese Einstellungen unter Linux zu überprüfen. Dazu folgen hier ein paar Seiten, die diese notwendigen Grundlagen vermitteln sollen. In den einzelnen späteren Kapiteln sind meist Verweise auf diese Grundlagen gegeben.Hardwareparameter Das /proc-Verzeichnis Plug and Play für Linux Konfiguration grundlegender BIOS-Einstellungen Konfiguration von Modem und Soundkarten Einrichten von SCSI-Geräten Einrichtung verschiedener PC-Erweiterungskarten Konfiguration von Kommunikationsgeräten Konfiguration von USB-Geräten

Seite:	-= LinuxLernSystem =- (http://www.lpi-test.de)
Kurs:	LPIC-1 [101]
Buch:	Study-Guide: Hardware und Systemarchitektur
Gedruckt von:	André Scholz
Datum:	Dienstag, 1 November 2005, 09:54 Uhr

Inhaltsverzeichnis

1.101 - Hardware und Systemarchitektur

In diesem Abschnitt geht es darum, verschiedene Einstellungen vorzunehmen, die als Grundvoraussetzung gelten, damit Linux überhaupt auf einem Computer arbeiten kann, bzw. damit eine bestimmte Hardware angesprochen werden kann. In der ersten Ausgabe der LPIC-Level 1 Prüfungen war dieses Thema zum zweiten Prüfungsteil zugeordnet, jetzt ist es beim ersten gelandet.

In diesem Zusammenhang sind bestimmte Grundlagenkenntnisse von hardwarespezifischen Einstellungen notwendig, die sich sowohl auf die Einstellungen der Erweiterungskarten beziehen, als auch auf die Möglichkeiten, diese Einstellungen unter Linux zu überprüfen. Dazu folgen hier ein paar Seiten, die diese notwendigen Grundlagen vermitteln sollen. In den einzelnen späteren Kapiteln sind meist Verweise auf diese Grundlagen gegeben.

- Hardwareparameter
- Das /proc-Verzeichnis
- Plug and Play für Linux

- Konfiguration grundlegender BIOS-Einstellungen
- Konfiguration von Modem und Soundkarten
- Einrichten von SCSI-Geräten
- Einrichtung verschiedener PC-Erweiterungskarten
- Konfiguration von Kommunikationsgeräten
- Konfiguration von USB-Geräten

Hardwareparameter

Jedes Stück Hardware, das wir installieren benötigt bestimmte Parameter, damit das Betriebssystem darauf zugreifen kann. Diese Parameter sind auf die verschiedensten Art und Weisen einstellbar, sowohl aus der Sicht des Betriebssystem, als auch aus der Sicht der Hardware selbst. Moderne Erweiterungskarten, die den PCI-Bus nutzen, kümmern sich selbst um die Einstellungen dieser Parameter, ältere Erweiterungskarten für den ISA-Bus müssen von Hand konfiguriert werden.

Damit unter Linux Hardwareerweiterungen korrekt installiert werden können, ist es notwendig etwas Basiswissen über die grundlegende Funktion solcher Karten zu besitzen. Diese Grundlageninformation soll hier auf die Schnelle vermittelt werden. Im Rahmen dieser Darstellung ist es natürlich nicht möglich, einen umfassenden Kurs über die verschiedenen Mechanismen der Erweiterungskarten abzuhalten, es soll versucht werden, sich auf die wesentlichen Grundlagen zu beschränken. Diese Darstellung bezieht sich auf die Architektur von IBM-kompatiblen Rechnern, die heute den sogenannten Industriestandard bestimmen. Andere Architekturen unterscheiden sich in Kleinigkeiten von den hier dargestellten Mechanismen.

Funktionsweise von Mainboards

Das Herz jedes Computers ist der Prozessor. Er verarbeitet die Befehle, die er aus Programmen erhält. Programme sind in diesem Zusammenhang natürlich nicht nur Anwenderprogramme, sondern auch das Betriebssystem und die von ihm verwendeten Gerätetreiber sowie das BIOS - die Sammlung grundlegender Ein- und Ausgaberoutinen (Basic Input Output System). Der Prozessor muß also in der Lage sein, sowohl die jeweiligen Befehle zu verstehen, als auch auf bestimmte Hardware zuzugreifen. Für beide dieser Handlungen greift er auf sogenannte Bus-Systeme zurück. Dabei handelt es sich zunächst einmal einfach um elektrische Verbindungen zwischen dem Prozessor und anderen Teilen des Computers.

Der wichtigste Teil des Computers - neben dem Prozessor - ist der Arbeitsspeicher. Er besteht aus einer großen Menge von Speicherbausteinen, die Daten während des laufenden Betriebs aufnehmen und speichern können. Der Zugriff auf diesen Arbeitsspeicher läuft wiederum über ein Bussystem. Damit der Prozessor genau bestimmen kann, auf welchen Teil des Speichers er zugreifen will, benötigt er ein Adressierungsschema. Jede Speicherzelle hat also eine Adresse. Will der Prozessor jetzt etwas in eine bestimmte Speicherzelle schreiben oder aus ihr lesen, so muß er zunächst einmal die Adresse dieser Speicherzelle auf den **Adressbus** schreiben. Dadurch öffnet sich ein Kanal auf dem Datenbus, vom Prozessor zu der angegebenen Speicherzelle. Erst jetzt können Daten auf diesem Weg übertragen werden.

Hardwareadressen

Aus der Sicht des Prozessors sind auch Zugriffe auf alle mögliche Hardware nur Speicherzugriffe. Das bedeutet, daß auch die andere Hardware des Computers entsprechende Adressen besitzen muß. Will der Prozessor etwa ein Byte auf die serielle

Schnittstelle schreiben, so ist das aus seiner Sicht kein Unterschied zu einem Schreibvorgang in den Speicher. Also benötigt er die genaue Adresse der seriellen Schnittstelle. Er kann diese Adresse auf den Adressbus legen, dadurch öffnet sich ein Kanal des Datenbusses vom Prozessor zu dem Baustein, der die serielle Schnittstelle steuert. Der Prozessor schreibt das Byte jetzt auf den Datenbus, auf der anderen Seite empfängt das entsprechende Bauteil jetzt das Byte und übernimmt die tatsächliche Ausgabe auf die Schnittstelle.

Jedes Stück Hardware, auf das der Prozessor zugreifen soll, muß also eine Adresse haben. Man spricht in diesem Zusammenhang von der IO-Adresse (IO - Input/Output - Ein/Ausgabe). Ein anderer Begriff für diese Adresse ist **ioport**.

Diese Hardware-Adresse ist sozusagen der Schlüssel für die Kommunikationsmöglichkeit zwischen Prozessor und Hardware. Es ist zwingend erforderlich, daß der Prozessor die richtige Adresse einer Hardware kennt, die er benutzen soll. Bei zusätzlich zu installierender Hardware (Erweiterungskarten) ist die Adresse meist einstellbar, so daß es zu keinerlei Doppelbelegung einzelner Adressen kommen kann. Denn das würde zwangsläufig zu Fehlfunktionen führen. Eine wichtige Aufgabe bei der Installation neuer Hardware ist es also, dafür zu sorgen, daß neue Hardware eine eindeutige Adresse zugewiesen bekommt, und der Prozessor diese Adresse kennt.

Das Interrupt-System

In einem Computersystem arbeitet der (oder die) Prozessor(en) jetzt also einzelne Programmanweisungen ab, die aus dem Speicher gelesen werden und die dann Ein- und Ausgaben auf Adressen vornehmen können. Es ist jetzt aber notwendig, daß die Abarbeitung dieser Programmanweisungen zu bestimmten Gelegenheiten unterbrochen werden müssen. In diesem Zusammenhang sprechen wir von Interrupts (Unterbrechungen). Ein Interrupt ist eine Unterbrechung des normalen Programmablaufs, um eine andere Aufgabe auszuführen. Sowohl das BIOS, als auch das Betriebssystem stellen jeweils viele hundert solcher Unterbrechungen zur Verfügung. Alle dieser Interrupts sind in einer sogenannten Interrupt-Tabelle zusammengefasst.

Wenn jetzt eine bestimmte Hardware eine Eingabe an den Computer schicken will, so muß sie den Prozessor benachrichtigen, daß auf ihrer io-Adresse neue Daten liegen, die zu lesen sind. Drückt der Anwender eines Computers beispielsweise eine Taste auf der Tastatur, so liegt der Wert dieser Taste jetzt an der IO-Adresse der Tastatur an. Nur leider weiß der Prozessor das noch nicht, hat also keinerlei Veranlassung, ein Byte von dieser Adresse zu lesen.

Aus diesem Grund muß jedes Gerät, das Eingaben an den Computer zulässt die Möglichkeit haben, eine Unterbrechungsanforderung (Interrupt-Request) an den Prozessor zu schicken. Wenn der Prozessor eine solche Anforderung erhält, unterbricht er seine Arbeit und führt den zu dieser Anforderung passenden Interrupt aus. Das heißt, er führt ein kleines Programm aus, und kehrt dann zum normalen Programmablauf zurück.

Eine zwingende Voraussetzung ist es dabei aber, daß der Prozessor wiederum genau weiß, welche Interrupt-Anforderung von welchem Gerät kommt. Nur so kann er das passende Interrupt-Programm starten, das auch zu der entsprechenden Hardware gehört.

Wenn also Hardware installiert werden soll, die nicht nur Ausgaben, sondern auch Eingaben an den Computer ermöglicht, dann muß dieser Hardware ein sogenannter Interrupt Request Channel oder kurz IRQ zugewiesen werden. Wie schon bei der IO-Adresse muß der Prozessor jetzt wieder genau wissen, welcher IRQ von welcher Hardware kommt.

Einige IRQs sind bereits standardmäßig von Systemhardware belegt, andere stehen für Erweiterungen zur Verfügung. Die folgende Tabelle zeigt die gängigste Belegung eines Standard-AT Systems:

IRQ	belegt durch
0	Timerbaustein
1	Tastatur
2	Kaskadierende Verbindung mit zweitem IRQ-Controller (dort 9)
3	zweite und vierte serielle Schnittstelle
4	erste und dritte serielle Schnittstelle
5	Frei (früher zweite parallele Schnittstelle)
6	Diskettenlaufwerk
7	Frei (früher erste parallele Schnittstelle)
8	RealTimeClock
9	Frei
10	Frei
11	Frei
12	Frei (meist PS/2-Maus)
13	Frei
14	Erster IDE-Controller
15	Zweiter IDE-Controller

Wenn zwei Geräte auf ein- und demselben IRQ liegen, so ist die Gefahr eines Konfliktes sehr groß. COM2 und COM4 liegen z.B. standardmäßig auf dem IRQ 3. Sollten Sie also versuchen, mit einem Modem an COM4 und einer Maus an COM2 gleichzeitig zu arbeiten, kann das nicht richtig funktionieren. Moderne Rechner sind - zumindestens in einigen Fällen etwa bei PCI-Erweiterungskarten - in der Lage sich IRQs zwischen mehreren Erweiterungen zu teilen (IRQ-sharing).

Das DMA-System

Unter dem Begriff DMA versteht man den direkten Speicherzugriff (*Direct Memory Access*) von Hardware auf den Arbeitsspeicher, ohne den Umweg über den Prozessor. Hardware, die schnellen und häufigen Zugriff auf den Arbeitsspeicher benötigt, kann über die Verwendung von DMA den Prozessor merklich entlasten.

Um dieses Feature zu ermöglichen sind sogenannte DMA-Kanäle vorgesehen, von denen

der Rechner wiederum nicht besonders viele anbietet. Ein Standard-PC bietet acht DMA-Kanäle an, von denen einer stets belegt ist (DMA4).

Auch hier haben wir aber wiederum die Notwendigkeit, daß die Hardware wissen muß, welchen DMA-Kanal sie benutzen darf, und der Prozessor auch weiß, welche Hardware welchen Kanal benutzt. Genauso wie bei IRQs ist es essentiell, daß jeder DMA-Kanal nur von einer Hardwareinstanz benutzt wird, damit es nicht zu schweren Konflikten kommt.

Einstellmöglichkeiten der Hardwareparameter

Die drei genannten Hardwareparameter **IO-Adresse**, **IRQ** und eventuell **DMA-Kanal** müssen für jede neu zu installierende Hardware eingestellt werden. Wichtig ist dabei, daß grundsätzlich jede Hardware eine IO-Adresse benötigt, Hardware, die Eingaben zulässt, dazu einen IRQ braucht und Hardware, die über direkten Speicherzugriff verfügt, zusätzlich noch einen DMA-Kanal benötigt.

Die Frage der Einstellung ist hier eine bilaterale Frage, denn die genannten Parameter müssen sowohl hardwareseitig eingestellt werden, als auch softwareseitig. Nur wenn diese beiden Einstellungen übereinstimmen, ist es möglich mit der Hardware zu arbeiten.

Die hardwareseitige Einstellung wird auf unterschiedliche Art und Weise vorgenommen, je nach verwendeter Erweiterungskartengeneration. Im Folgenden sollen kurz die üblichen Verfahren erläutert werden:

Feste Hardware auf dem Mainboard
: Fest auf dem Mainboard integrierte Hardware wie z.B. serielle Schnittstellen bekommen ihre Einstellungen über das BIOS-Setup Programm. Dort kann eingestellt werden, welche Adressen und welche IRQs diese Schnittstellen benutzen. In der Regel sind hier zwei oder drei feste Einstellungsmöglichkeiten vorgegeben.

Sehr alte ISA-Erweiterungskarten
: Auf sehr alten Karten müssen sämtliche Einstellungen hardwaremäßig vorgenommen werden. Das heißt auf diesen Karten müssen die Adressen mit Jumpern oder Dip-Schaltern eingestellt werden. In der Regel ist das nur mit einem passenden Handbuch möglich, es sei denn die Karten besitzen einen Siebdruck-Aufdruck, der die jeweiligen Einstellungen erklärt.

Etwas modernere ISA-Erweiterungskarten
: Etwas modernere Karten besitzen kleine Festspeicher (EEPROMS, Flash-EPROMS) die softwaremäßig eingestellt werden können. Dazu benötigt man ein spezielles kleines Programm, das der Hersteller solcher Karten mitliefert. Meist ist dieses Programm auf einer Diskette beigelegt und muß unter DOS gestartet werden. Es sind dann alle Einstellungen (IO-Adressen, IRQs und DMA-Kanäle) softwaremäßig einstellbar.

Die genannten drei Methoden bedingen grundsätzlich, daß auch das Betriebssystem bzw. damit der Prozessor erfährt, welche Einstellungen getroffen wurden. Wie oben schon gesagt, nur wenn beide Seiten (Hard- und Software) die gleiche Information haben, kann die Ansteuerung der Hardware funktionieren.

Modernere Systeme versuchen den Einbau von Erweiterungskarten dahingehend zu

vereinfachen, daß die Karte und das Betriebssystem sich gegenseitig einigen, welche Parameter sie benutzen. Damit sollen auch Nicht-Spezialisten in die Lage versetzt werden, neue Hardware einzubauen. In diesem Zusammenhang ist häufig das Schlagwort "*Plug And Play*" zu hören, als etwa *"einstecken und loslegen"*.

ISA Plug And Play Karten

Diese Karten haben genügend "Eigenintelligenz", um sich mit dem System die notwendigen Parameter auszutauschen, wenn das verwendete Betriebssystem Plug And Play-fähig ist. Im Abschnitt 1.101.5 werden wir uns damit auseinandersetzen, wie diese Art von Karten unter Linux zum Laufen gebracht werden können.

PCI Erweiterungskarten

PCI-Karten sind die modernen Erweiterungskarten, die in den letzten Jahren die ISA-Karten völlig vom Markt gedrängt haben. Der PCI-Bus ist ein intelligenter Bus, der bestimmte Mindestanforderungen an seine Erweiterungskarten stellt. PCI-Karten können grundsätzlich ihre Parameter mit dem Betriebssystem aushandeln, so daß hier keine manuellen Einstellungen mehr notwendig sind.

Speicherbereiche

Manche Erweiterungskarten besitzen einen eigenen Speicher auf der Karte, entweder einen wirklichen Arbeitsspeicher, wie etwa der Bildschirmspeicher auf der Graphikkarte oder einen EPROM mit eigenem BIOS, wie z.B. Graphikkarten und SCSI-Hostadapter. Auch diese Speicherbereiche haben eine Anfangsadresse, die dem System bekannt sein muß, damit es auf den entsprechenden Speicher zugreifen kann. Man spricht in diesem Zusammenhang von Memory-Base oder kurz MemBase (die Basisadresse des Kartenspeichers).

Das /proc-Dateisystem

Das `/proc`-Verzeichnis ist kein wirkliches Dateisystem, sondern eine Schnittstelle zum Kernel. Die Dateien, die in diesem Verzeichnis liegen, benutzen keinen Speicherplatz auf der Platte, sind aber trotzdem les- und in manchen Fällen auch beschreibbar.

Seinen Namen trägt dieses Verzeichnis daher, daß es für jeden laufenden Prozess ein Unterverzeichnis bereithält, das Informationen über diesen Prozess zur Verfügung stellt. Das Unterverzeichnis trägt als Namen die ProzeßID (PID) des jeweiligen Prozesses. Es enthält unter anderem folgende Dateien:

- `cmdline`
  Die Kommandozeile, mit der der Prozeß gestartet wurde, mit allen verwendeten Parametern.
- `cwd`
  (current working directory) Ein symbolischer Link auf das Verzeichnis, das beim Aufruf des Prozesses das aktuelle Arbeitsverzeichnis war.
- `environ`
  Die komplette Umgebung des Prozesses (Variablen, Funktionen usw) sofern er eine Umgebung hat.
- `exe`
  Ein symbolischer Link auf das aufgerufene Programm, daß den Prozeß ausmacht.
- `root`
  Ein symbolischer Link auf das Verzeichnis, das für den Prozeß das Wurzelverzeichnis darstellt.

Daneben finden sich weitere Informationen zu den verwendeten Ressourcen (Speicher, Libraries) und ein Unterverzeichnis **fd**, das die File-Descriptoren aller vom Prozeß verwendeten Dateien enthält.

Diese Information wird beispielsweise von den Programmen verwertet, die Prozeß-Informationen ausgeben.

Was aber für uns im Zusammenhang mit Hardware wesentlich wichtiger ist, ist die Möglichkeit, im /proc-Verzeichnis auf bestimmte Informationen zuzugreifen, die hardwarerelevant sind.

Die Hardwareparameter im /proc-Verzeichnis

Wie schon aus der Seite Hardwareparameter hervorgegangen ist, benötigen wir, um bestimmte Hardware anzusprechen, Einstellungen, die sowohl auf der Hardwareseite vorgenommen sein müssen, als auch dem Betriebssystem bekannt sein müssen. Um festzustellen, welche dieser Parameter Linux welcher Hardware zuweist bietet uns das `/proc`-Verzeichnis mehrere Dateien, die diese Information beinhalten. Nochmal zur Erinnerung: Es handelt sich hierbei nicht um reale Dateien, sondern um Schnittstellen zum Kernel. Es sind also keine statischen Informationen sondern die gegenwärtigen Informationen des Kernels, welche Parameter von welcher Hardware benutzt werden. Diese

Dateien sind hervorragende Diagnosewerkzeuge, die bei der Fehlersuche meist gute Dienste leisten. Ihr Inhalt ist mit Programmen wie cat oder **less** einsehbar.

Für die Hardware-Parameter spielen folgende Dateien im `/proc`-Verzeichnis eine Rolle:

`/proc/interrupts`
: Zeigt eine Liste der im Augenblick benötigten IRQ-Kanäle an, zusammen mit der Information, wer diese IRQs benutzt (wem sie zugewiesen sind) und wie oft die jeweiligen IRQs schon ausgelöst wurden.

`/proc/ioports`
: Zeigt eine Liste der IO-Adressen an, die gegenwärtig in Benutzung sind und die Information, wem sie zugeordnet sind. Die Adressen werden als Adressbereich angegeben, der zeigt, wie breit der Zugriff auf eine solche Adresse ist. Die Angabe

```
0170-0177 : ide1
```

zeigt etwa, daß auf die zweite IDE-Schnittstelle mit 8 Bit Breite (170-177=8 Bit) zugegriffen wird. (Es handelt sich hier um die Zugriffsbreite auf den Controller, nicht auf die Daten!).

`/proc/dma`
: Hier finden wir die Angabe der benutzten DMA-Kanäle zusammen mit der Information, wer sie benutzt.

`/proc/iomem`
: Hier werden uns die Speicherbereiche der verschiedenen Speicherarten angezeigt. Dazu zählen neben dem normalen Arbeitsspeicher (System RAM) auch die Bereiche Bildschirmspeicher der Graphikkarte (Video RAM) und die jeweiligen ROMs der Hauptplatine (System ROM) und der Erweiterungskarten (z.B. Video ROM), sowie der Speicher des PCI-Systems, in dem die Bus-Informationen abgelegt sind. Diese Angaben sind als Adressbereiche (Startadresse - Endadresse) angegeben.

Andere Hardwareinformationen im /proc-Verzeichnis

Neben den verwendeten Hardwareparametern finden sich im `/proc` Verzeichnis noch einige andere interessante Informationen über die verwendete Hardware, die für Diagnosezwecke sehr brauchbar sind. Dazu zählen die Dateien:

`/proc/cpuinfo`
: Alle Informationen, die der Kernel über den/die verwendeten Prozessor(en) hat, sind hier nachzulesen. Dazu zählen Prozessortyp und -modell, Taktrate, Cache-Größe und viele Angaben über mögliche Prozessorbugs.

`/proc/devices`
: Eine Liste der block- und zeichenorientierten Geräte, die der Kernel aktuell unterstützt. Neben der jeweiligen Angabe des Gerätenamens finden wir hier auch die verwendete Major-Number.

`/proc/partitions`
: Eine Liste aller dem System bekannten Plattenpartitionen, mitsamt Major- und Minornummern. Je nachdem, ob der Kernel das **devfs** Dateisystem unterstützt oder nicht, werden die Angaben in der Form

```
major minor  #blocks  name
 3     0   16617888 hda
 3     1    1028128 hda1
```

(alte Darstellung) oder

```
major minor  #blocks  name
 3     0   19925880 ide/host0/bus0/target0/lun0/disc
 3     1    4200966 ide/host0/bus0/target0/lun0/part1
```

(neue Darstellung bei Verwendung von **devfs**) angezeigt.

`/proc/pci`
: Informationen, die der Scan des PCI-Busses ergeben haben. Hier finden sich Informationen über alle am System angeschlossenen PCI-Geräte. Auch der AGP-Bus (der in Wahrheit nur ein extra PCI-Bus mit nur einem Steckplatz ist) ist hier angegeben zusammen mit allen gefundenen Karten.

Kernel- und Softwareinformationen im /proc-Verzeichnis

Neben der Hardware-Information hält das `/proc`-Verzeichnis auch noch einige Details über den Kernel selbst und seine Fähigkeiten zur Verfügung, sowie Informationen über aktuelle Systemzustände. Dazu zählen:

`/proc/cmdline`
: Die Kommandozeile, mit der der Kernel selbst gestartet wurde. Hier sind auch die entsprechenden Kernelparameter nachzulesen, die beim Booten mitgegeben wurden. Außerdem ist der Dateiname des aktuellen Kernels genannt.

`/proc/filesystems`
: Eine Liste aller Dateisystemtypen, die der Kernel im Augenblick kennt.

`/proc/meminfo`
: Informationen über die gegenwärtige Auslastung des Arbeitsspeichers.

`/proc/modules`
: Eine Liste der aktuell geladenen Kernel-Module.

`/proc/mounts`
: Eine Liste aller gemounteter Dateisysteme. Hier finden sich auch die Dateisysteme, die mit der Option **-n** gemountet wurden und so weder in der Datei `/etc/mtab` stehen, noch durch den Befehl df anzeigbar sind.

`/proc/version`
: Die aktuelle Kernelversion.

Weitere wichtige Unterverzeichnisse in /proc

Neben den besprochenen Dateien finden sich auch noch eine Menge Unterverzeichnisse im `/proc`-Verzeichnis, die weitere Informationen über angeschlossene Geräte und Kernelfeatures enthalten. Dazu zählen insbesondere:

`/proc/bus`
: Informationen über die gefundenen Bussysteme (PCI, USB, ...)

/proc/ide
: Informationen über IDE-Geräte und -Schnittstellen.

/proc/scsi
: Informationen über SCSI-Geräte und Schnittstellen.

/proc/net
: Informationen über netzwerk-relevante Einstellungen. Hier finden wir beispielsweise die Routing-Tabellen und die ARP-Informationen, die der Kernel im Augenblick besitzt.

Aktive Veränderungsmöglichkeiten unter /proc/sys

Bisher haben wir das `/proc` Verzeichnis nur benutzt, um uns Informationen über bestimmte Kerneleigenschaften geben zu lassen. Im Verzeichnis `/proc/sys` können wir aber auch die Einstellungen bestimmter Kernelfähigkeiten verändern. Das geschieht dadurch, daß wir die gewünschten Werte in die Dateien schreiben, statt sie nur zu lesen. In der Regel handelt es sich hierbei aber nur um Dateien, die einzelne Werte enthalten, meist eine 1 oder 0, die ein "wahr" oder "falsch" repräsentieren.

Ein einfaches Beispiel für diese Fähigkeit ist die Einstellung, ob unser Kernel in der Lage ist, als Router zu arbeiten oder nicht. Ein Router ist ein Rechner, der einzelne Netzwerkpakete von einer Netzschnittstelle zu einer anderen weitergibt. Ob diese Fähigkeit aktiviert ist oder nicht, erfahren wir, wenn wir uns den Inhalt der Datei `/proc/sys/net/ipv4/ip_forward` ansehen. Wir schreiben also

```
cat /proc/sys/net/ipv4/ip_forward
```

Als Ausgabe bekommen wir eine 0, was soviel bedeutet, wie "Nein". Jetzt wollen wir unseren Kernel routing-fähig machen. Dazu schreiben wir eine 1 in diese Datei hinein. Dazu bedienen wir uns nicht eines Editors, sondern einfach des Befehls **echo** Dieser Befehl gibt etwas auf die Standard-Ausgabe aus, was wir aber dann in die gewünschte Datei umleiten:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Und schon ist der Kernel in der Lage zu routen.

An diesem Beispiel wird klar, daß das `/proc`-Verzeichnis auch zur Veränderung der Kerneleigenschaften geeignet ist. Im Großen und Ganzen sind alle Einstellungen unter `/proc/sys` auf diese Weise variierbar.

Plug and Play für Linux

Wie in der Beschreibung über die Hardwareparameter schon gezeigt, benötigen ISA-Karten die korrekte Einstellung ihrer Parameter wie IRQ, IO-Port und DMA-Kanal. Das Betriebssystem muß sich im Klaren über diese Parameter sein, damit es die entsprechende Hardware ansprechen kann. Bei einer Installation einer solchen Karte müssen also normalerweise die entsprechenden Einstellungen sowohl auf der Harwareseite, als auch der Softwareseite vorgenommen werden, damit die Karte dann funktioniert.

Die moderneren ISA-Karten haben zur Vereinfachung dieser Einstellungen einen Standard entwickelt, der die Karte und das Betriebssystem in die Lage versetzen soll, miteinander zu kommunizieren. Mit Hilfe dieser Verständigung sollte es möglich werden, daß sich die Karte beim Betriebssystem erkundigt, welche Ressourcen (IO-Ports, IRQs, usw) noch frei sind und dann selbstständig eine entsprechende Einstellung vornimmt. Das Betriebssystem selbst kann diese Einstellung dann aus der Karte lesen und den Gerätetreiber entsprechend konfigurieren. Diese Technik stellt natürlich entsprechende Anforderungen an die Erweiterungskarte. Sie muß einen ROM-Speicher besitzen, der die denkbaren Parameter enthält und die entsprechende Verdrahtung oder Logik, die diese Einstellungen dann nutzt.

Dieser Standard wird *Plug and Play* genannt, also etwa *Einstecken und loslegen*. Voraussetzung für die korrekte Funktionsweise ist es aber, daß das Betriebssystem diese Technik beherrscht. Die schlechte Nachricht gleich zuerst, Linux ist nicht Plug and Play fähig.

Um mit Linux sogenannte ISA-PnP Karten anzusprechen, muß eine passende Einstellung an die Karte geschickt werden, die ihre Ressourcen fest zuweist. Wenn das geschehen ist, kann Linux die Karte genauso behandeln, als ob es eine Karte mit fester manueller oder softwaremäßiger Einstellung wäre. Das heißt, beim Laden der Gerätetreiber können die entsprechenden Parameter angegeben werden.

Um die entsprechenden Einstellungen vorzunehmen gibt es zwei Programme, die diese Aufgabe in zwei Schritten übernehmen. Allerdings ist dazu etwas Handarbeit nötig. Die beiden Programme sind:

- pnpdump
- isapnp

Das erste Programm, **pnpdump**, scant den ISA-Bus nach PnP-Karten ab und ließt von den gefundenen Karten die möglichen Einstellungen (Ressourcen) aus dem ROM. Diese Einstellungen werden dann in einem bestimmten (für Menschen lesbaren) Format auf die Standard-Ausgabe geschrieben. Die denkbaren Einstellungen sind angegeben, eine tatsächliche Konfiguration ist aber auskommentiert. Die Ausgabe von **pnpdump** wird üblicherweise in eine Datei umgeleitet und dann mit einem Editor bearbeitet um eine bestimmte Einstellung zu aktivieren. Die bearbeitete Datei wird dann zur Konfigurationsdatei des zweiten Programms, mit dem wir die Einstellungen dann an die Karten zurückgeben. Dieses zweite Programm ist **isapnp**

Wir können also die Konfiguration von PnP-Karten in drei Schritte gliedern:

- **Aufruf von pnpdump**
  pnpdump wird aufgerufen und seine Ausgabe am besten gleich in die Datei `/etc/isapnp.conf` umgeleitet. Das ist die voreingestellte Konfigurationsdatei von **isapnp**. Wir schreiben also

  ```
  pnpdump > /etc/isapnp.conf
  ```

- **Editieren der Datei `/etc/isapnp.conf`**
  Die gerade erstellte Datei wird mit einem Editor bearbeitet, um die gewünschten Einstellungen zu aktivieren. Es ist jetzt möglich, die Hardwareparameter auszusuchen, die auf dem System noch frei sind.
- **Aufruf von isapnp**
  Das Programm **isapnp** schreibt die Einstellungen aus seiner Konfigurationsdatei wieder zurück an die Karten. Dieser Vorgang muß nach jedem Booten wiederholt werden. Das Programm **isapnp** sollte also über ein Startscript bei jedem Systemstart aufgerufen werden.

Nachdem diese drei Schritte vollzogen wurden, sind die ISA-Karten soweit, als wären es normale Karten, die über Jumper oder Software konfiguriert worden sind. Jetzt erst können - mit den entsprechenden Parametern - die Kernelmodule für diese Karte geladen werden.

Eine Beispielkonfiguration

Um den Vorgang der manuellen Einstellung einmal genauer zu betrachten folgt hier eine Beispielkonfiguration. Ein Rechner ist mit einer Plug and Play-fähigen Ethernetkarte ausgestattet. Der Aufruf von

```
pnpdump > /etc/isapnp.conf
```

erzeugt die Datei `/etc/isapnp.conf` mit folgendem Inhalt (gekürzt). Wichtige Zeilen sind rot hervorgehoben.

```
# $Id: pnpdump.c,v 1.21 1999/12/09 22:28:33 fox Exp $
# Release isapnptools-1.21 (library isapnptools-1.21)
#
# This is free software, see the sources for details.
# This software has NO WARRANTY, use at your OWN RISK
#
# For details of the output file format, see isapnp.conf(5)
#
# For latest information and FAQ on isapnp and pnpdump see:
# http://www.roestock.demon.co.uk/isapnptools/
#
# Compiler flags:  -DREALTIME -DNEEDSETSCHEDULER -DABORT_ONRESERR
# (for   library:  -DREALTIME -DNEEDSETSCHEDULER -DABORT_ONRESERR)

# Card 1: (serial identifier 13 0e 1e 37 b4 19 01 89 14)
# EDI0119 Serial No 236861364 [checksum 13]
```

```
# Version 1.0, Vendor version 1.0
# ANSI string -->PLUG & PLAY ETHERNET CARD<--
# Logical device id EDI0119
#     Device support I/O range check register
        (CONFIGURE EDI0119/236861364 (LD 0
#     Compatible device id PNP80d6
#     Logical device decodes 10 bit IO address lines
#         Minimum IO base address 0x0240
#         Maximum IO base address 0x03e0
#         IO base alignment 32 bytes
#         Number of IO addresses required: 32
#         (IO 0 (BASE 0x0340))
#     IRQ 3, 4, 5, 9, 10, 11, 12 or 15.
#         High true, edge sensitive interrupt
#         (INT 0 (IRQ 10 (MODE +E)))
#     Memory is non-writeable (ROM)
#     Memory is non-cacheable
#     Memory decode supports high address
#     memory is 8-bit only
#     memory is shadowable
#     memory is an expansion ROM
#     Minimum memory base address 0x0c0000
#     Maximum memory base address 0x0dc000
#     Range base alignment mask 0xff4000 bytes
#     Range length 16384 bytes
# Choose UPPER = Range, or UPPER = Upper limit to suit hardware
# (MEM 0 (BASE 0x0c0000) (MODE bu) (UPPER 0x0c4000))
# (MEM 0 (BASE 0x0c0000) (MODE br) (UPPER 0x004000))
#         (ACT Y))
)
# End tag... Checksum 0x00 (OK)

(WAITFORKEY)
```

Aus den Angaben kann folgendes ersehen werden:

- Es wurde eine Plug and Play fähige Karte mit der Seriennummer 13 0e 1e 37 b4 19 01 89 14 gefunden.
- Es handelt sich um eine Ethernet-Karte.

Die eigentliche Konfigurationsarbeit findet zwischen den Zeilen

```
  (CONFIGURE EDI0119/236861364 (LD 0
```

und

```
  )
```

statt. Alle dort gemachten Angaben sind noch auskommentiert, also wirkungslos. Erst durch manuellen Eingriff werden sie aktiviert.

Die Karte benötigt eine IO-Adresse und einen IRQ. Für diese beiden Einstellungen sind jeweils die Informationen angegeben, welche Angaben gemacht werden können. Zunächst die IO-Adresse:

Aus dem Kommentar

```
#       Minimum IO base address 0x0240
#       Maximum IO base address 0x03e0
#       IO base alignment 32 bytes
```

können wir entnehmen, daß die kleinste Adresse die 0x240 ist, die größte Adresse 0x3e0. Die Adressen dazwischen können in Schrittweiten von 32 Byte angegeben werden. Gültige IO-Adressen wären also:

- `0x0240`
- `0x0260`
- `0x0280`
- `0x02a0`
- `0x02c0`
- `0x02e0`
- `0x0300`
- `0x0320`
- `0x0340`
- `0x0360`
- `0x0380`
- `0x03a0`
- `0x03c0`
- `0x03e0`

Jede dieser Adressen repräsentiert einen Adressbereich von dieser Adresse bis zur jeweils nächsten. Die Adresse `0x0240` meint also eigentlich den Bereich `0x0240-0x025f`.

Aus diesem Adressenpool muß nun eine Adresse ausgewählt werden, die noch nicht vom System oder einem anderen Gerät benutzt wird. Dazu wird die Datei `/proc/ioports` überprüft, die z.B. für den genannten Adressbereich folgende Angaben beinhalten könnte:

```
...
0170-0177 : ide1
01f0-01f7 : ide0
02f8-02ff : serial(set)
0376-0376 : ide1
0378-037a : parport0
03c0-03df : vga+
03f6-03f6 : ide0
03f8-03ff : serial(set)
...
```

Die Adressen `0x02e0-0x02ff` sind für unser Beispiel also nicht zu gebrauchen, weil in diesem Bereich bereits eine Adresse (serial) vergeben ist. Auch die Adressen `0x0360` bis `0x03e0` sind bereits durch andere Geräte (ide1, Parport0, vga, ide0 und serial) belegt. Alle anderen Adressen stehen uns zur freien Verfügung. Wir wählen beispielsweise die Adresse `0x0300`. Um diese Einstellung vorzunehmen ändern wir die Zeile

```
#       (IO 0 (BASE 0x0340))
```

entsprechend um, und entziehen ihr das Kommentarzeichen:

```
        (IO 0 (BASE 0x0300))
```

Ähnlich gehen wir jetzt mit dem IRQ um. Die Zeile

```
#    IRQ 3, 4, 5, 9, 10, 11, 12 or 15.
```

zeigt uns alle IRQs, die unsere Karte zur Verfügung stellt. Ein Blick nach `/proc/interrupts` zeigt uns, welche IRQs auf unserem System schon belegt sind:

```
 0:    882379        XT-PIC  timer
 1:     31607        XT-PIC  keyboard
 2:         0        XT-PIC  cascade
10:       126        XT-PIC  ES1938
11:     54397        XT-PIC  aic7xxx
12:    250763        XT-PIC  PS/2 Mouse
14:     41476        XT-PIC  ide0
15:     63616        XT-PIC  ide1
```

IRQ 5 wäre beispielsweise noch frei. Um diesen IRQ zu aktivieren, ändern wir die Zeile

```
#       (INT 0 (IRQ 10 (MODE +E)))
```

entsprechend um und entfernen wiederum das Kommentarzeichen:

```
        (INT 0 (IRQ 5 (MODE +E)))
```

Damit sind alle notwendigen Einstellungen erledigt, wir müssen die Karte jetzt nur noch aktivieren. Dazu wird der Zeile

```
#       (ACT Y))
```

das Kommentarzeichen entfernt. Damit sind die manuellen Einstellungen erledigt, die Datei kann abgespeichert werden.

Um die vorgenommenen Einstellungen jetzt tatsächlich der Karte bekannt zu machen, muß das Programm **isapnp** aufgerufen werden. Das Programm bekommt den Namen der Konfigurationsdatei als Parameter, der Aufruf lautet also:

```
/sbin/isapnp /etc/isapnp.conf
```

Diese Zeile muß bei jedem Systemstart erneut ausgeführt werden. Sie sollte also in einem der init-Scripts eingetragen sein. Meist wird bei der Installation von **isapnp** bereits ein entsprechendes init-Script angelegt.

Mit der hier besprochenen Einstellungsarbeit ist die ISA-Karte aber noch nicht ins System eingebunden. Das Einzige, was mit dieser Arbeit erreicht wurde ist, daß die Karte jetzt die entsprechenden Hardwareparameter benutzt. Die Arbeit entspricht also der manuellen Einstellung durch Jumper auf alten Karten. Das Laden der entsprechenden Module und die Übergabe der Hardwareparameter wird Thema der Vorbereitung auf die zweite Prüfung sein.

1.101.1 - Konfiguration grundlegender BIOS-Einstellungen

Beschreibung: Prüfungskandidaten sollten in der Lage sein, grundlegende Systemhardware durch korrekte Einstellungen im System-BIOS zu konfigurieren. Dieses Lernziel beinhaltet das richtige Verständnis von BIOS-Konfigurations-Fragen wie der Verwendung von LBA bei IDE-Festplatten mit mehr als 1024 Zylindern, das Aktivieren und Deaktivieren von integrierten Peripheriegeräten, sowie die Konfiguration von Systemen mit bzw. ohne externen Peripheriegeräten wie z.B. Tastaturen. Ebenfalls enthalten ist das korrekte Setzen von Interrupts, DMA- und I/O-Adressen für alle vom BIOS verwalteten Ports und Einstellungen für Fehlerbehandlung.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen

- `/proc/ioports`
- `/proc/interrupts`
- `/proc/dma`
- `/proc/pci`

Die hier geforderten Fähigkeiten sind eigentlich nicht Linux-spezifisch, haben jedoch eine Bedeutung für die Installation und die fehlerfreie Ausführung von Linux auf einem PC. Ein Systemverwalter sollte eigentlich keinerlei Probleme damit haben. Trotzdem gehe ich hier der Vollständigkeit halber die einzelnen Punkte noch einmal durch.

Die Funktion des Computer-BIOS

Das Basic Input Output System (BIOS) eines PC ist eine kleine Softwareschicht, die sich zwischen die Hardware und das Betriebssystem legt, um eine einheitliche Schnittstelle gegenüber der Hardware zur Verfügung zu stellen. Bei der Vielfältigkeit der Mainboards wäre ansonsten jedes Betriebssystem gezwungen, die verschiedensten Architekturen der Boards alle genau zu kennen.

Vereinfacht kann man sagen, daß das BIOS die wesentliche Funktionalität des Mainboards für das Betriebssystem zugänglich macht. Dazu gehören natürlich auch die verschiedenen Schnittstellen, die heutzutage selbstverständlich auf einem Mainboard integriert sind. Standardmäßig zählen dazu:

- Tastaturschnittstelle
- PS/2 Mausport (psaux)
- Zwei serielle Schnittstellen
- Eine parallele Schnittstelle
- Zwei E-IDE Schnittstellen (für bis zu 4 IDE-Geräte)
- Ein Floppy-Controller (für bis zu zwei Diskettenlaufwerke)

Optional sind womöglich noch andere Schnittstellen enthalten wie

- USB-Ports
- Sound
- SCSI
- Game-Port

Diese optionalen Schnittstellen werden in späteren Kapiteln noch genauer behandelt. In diesem Abschnitt interessieren uns die Standard-Schnittstellen des Boards.

Das BIOS ist also eine Sammlung von Software, die sich physikalisch in einem Speicherbaustein (ROM) auf dem Board selbst befindet. Die Software passt genau zu dem jeweiligen Board und ist entsprechend nicht einfach austauschbar. In der Praxis hat das BIOS ein paar unterschiedliche Aufgaben zu bewältigen, von denen Linux einige aus Geschwindigkeitsgründen umgeht. Von daher sind einige Einstellungen, die wir machen können für den täglichen Gebrauch mit Linux gar nicht wichtig, andere hingegen sind unumgänglich, da sie den Ladevorgang selbst betreffen.

Das CMOS-Setup Programm

Jedes BIOS hat die Möglichkeit, bestimmte Einstellungen vom User verändern zu lassen. Dazu stellt es ein kleines Setup-Programm zur Verfügung, das beim Systemstart geladen werden kann. Verschiedene BIOS-Hersteller gehen dabei unterschiedliche Wege, meist wird durch eine bestimmte Taste(nkombination) während des Startvorganges - vor dem Booten des Betriebssystems - dieses Programm aktiviert. Typische Tasten(kombinationen) sind

- DEL (Entf)
- F2
- Strg-Alt-Esc

In der Regel erscheint daraufhin ein Menü, das verschiedene Auswahlmöglichkeiten bietet. Die optische Aufmachung dieses Menüs variiert von Hersteller zu Hersteller und von Version zu Version. Gemeinsam sind diesen Menüs die Einstellmöglichkeiten für die verschiedenen Peripheriegeräte, die auf dem Mainboard liegen. Alle weiteren Peripheriegeräte, die etwa über Erweiterungskarten hinzugefügt werden, können nicht mit diesem Programm eingestellt werden!

Das Standard-Setup

Klassische Einstellmöglichkeiten finden wir meist in einem Untermenü mit Namen Standard-Setup. Dieses Menü war bei den ersten AT-Computern schon vorhanden und trägt daher diesen Namen. Hier können folgende Einstellungen vorgenommen werden:

- Einstellung des Datums und der Uhrzeit
- Einstellung, welche Diskettenlaufwerke angeschlossen sind.
- Einstellungen welche Festplatten angeschlossen sind. (Diese Einstellung bezieht sich wiederum nur auf die Platten, die an die Schnittstellen angeschlossen sind, die auf dem Mainboard integriert sind, nicht die an zusätzlichen Controllern.)

- Primäre Graphikkarte. Stammt aus einer Zeit, als es noch Graphikkarten mit unterschiedlichen Speicheradressen gab (Hercules, MDA). Steht heute praktisch grundsätzlich auf VGA/EGA.
- Fehlerbehandlung beim Booten

Wichtig sind hier die Einstellungen der Festplatten und die Fehlerbehandlung.

Festplatteneinstellungen

Die üblichen Mainboards haben zwei IDE Kanäle onboard, können also bis zu vier IDE Platten (oder CDROMS/DVD/Brenner) anschließen. Die Einstellungen für die Platten sind für Linux eigentlich unerheblich, da der Kernel die Controller direkt anspricht und nicht über das BIOS. Trotzdem sind ein paar Einstellungen hier von großer Bedeutung.

Festplatten werden traditionell über drei Größen klassifiziert. Die Anzahl der Zylinder (übereinanderliegende Spuren auf den Plattenoberflächen), die Anzahl der Sektoren pro Spur (die jeweils 512 Byte Speicher enthalten) und die Anzahl der Köpfe (und damit auch die Anzahl der beschreibbaren Oberflächen). Daraus errechnet sich die Kapazität einer Festplatte folgendermaßen:

Zylinder * *Köpfe* * *Sektoren_pro_Spur* * 512 Byte

Physikalisch sind die Platten zwar heute intern etwas anders aufgebaut (so haben etwa die inneren Spuren weniger Sektoren als die äußeren), aber logisch werden sie immer noch so verwaltet. Das Problem liegt jetzt darin, daß das BIOS sich schwertut, Betriebssysteme von Platten zu booten, die mehr als 1024 Zylinder besitzen. Aus diesem Grund bieten moderne BIOSe die Möglichkeit einer alternativen Rechnung an. Wenn wir etwa die Anzahl der Zylinder halbieren, aber dafür die Anzahl der Köpfe verdoppeln (rechnerisch, nicht physikalisch), dann bekommen wir das exakt gleiche Ergebnis der obigen Rechnung. Als Nebeneffekt haben wir aber jetzt weniger als 1024 Zylinder und können problemlos booten. Dieser Modus heißt *Large Block Architecture* oder einfach LBA-Modus.

Trotzdem Linux das BIOS bei Plattenzugriffen umgeht, sollten wir diesen Modus für Platten mit mehr als 1024 Zylindern wählen, weil wir sonst Probleme bekommen können mit

- Partitionierungssoftware (fdisk), die immer noch auf den BIOS-Einstellungen fusst
- Bootmanagern (Lilo, Grub), die vom BIOS geladen werden müssen.

Grundsätzlich ist also für Festplatten mit mehr als 1024 Zylindern der LBA-Modus einzustellen. Wir haben dadurch keinerlei Nachteile, jedoch auch keine Probleme mehr.

Fehlerbehandlung

Die Einstellungen für die Fehlerbehandlung sind relativ trivial. Hier kann eigentlich nur festgehalten werden, bei welchen Fehlern das System gar nicht erst bootet. In der Regel bietet ein Setup-Programm folgende Alternativen:

- Halt on all errors

- Halt on all errors but disk
- Halt on all errors but keyboard
- Halt on all errors but disk and keyboard

Normalerweise könnten wir davon ausgehen, daß eine Einstellung *Halt on all errors* eine gute Idee ist, aber bei Linux ist das womöglich anders. In vielen Fällen arbeiten Linux-Server ohne eigene Tastatur. Das kann daran liegen, daß diese Server in einem Rack hängen und softwaremäßig über eine Konsole verwaltet werden, oder daß der Server sich zusammen mit anderen eine Tastatur (und einen Monitor) über eine Umschaltbox teilt. In jedem Fall kann es vorkommen, daß ein solcher Rechner bootet und dabei keine Tastatur vorfindet. Es wäre aber ärgerlich, wenn dadurch der Bootvorgang abgebrochen werden würde. Für solche Rechner empfielt es sich also, die Einstellung *Halt on all errors but keyboard* vorzunehmen.

Peripheriegeräte einstellen

An einer anderen Stelle im CMOS-Setup-Programm können die Einstellungen für die weiteren Geräte vorgenommen werden, die auf dem Mainboard integriert sind. Dieser Menüpunkt heißt oft *integrated peripherals*. Hier können die integrierten Schnittstellen aktiviert bzw. deaktiviert werden und ihre Hardwareparameter können eingestellt werden.

Hierbei kann es naturgemäß zu Überschneidungen mit Hardwareparametern externer Erweiterungskarten kommen. Sollte eine Erweiterungskarte einen Adressbereich, einen IRQ oder DMA-Kanal benutzen wollen, der hier schon vergeben ist, so muß dafür gesorgt werden, daß alle Geräte eigene Werte bekommen. Um zu überprüfen, welche Geräte mit welchen Hardwareparametern arbeiten, stehen uns dafür unter Linux die Dateien im /proc-Verzeichnis zur Verfügung.

In manchen Fällen kann es auch sinnvoll sein, bestimmte Peripheriegeräte auf dem Mainboard ganz abzuschalten, wenn sie etwa nicht benutzt werden. Ein Rechner, der nur mit SCSI-Geräten arbeitet, muß keine IDE-Kanäle aktiviert haben. So sparen wir uns schon zwei IRQs (14 und 15). Oder ein Rechner mit integrierter Soundkarte kann diese bedenkenlos deaktivieren, wenn er als Webserver oder Internet-Router in einem 19 Zoll Schrank arbeitet und gar keine Boxen angeschlossen hat...

Auch wenn statt der integrierten Soundkarte eine andere (bessere) verwendet werden soll, bietet es sich an, die integrierte Soundkarte einfach zu deaktivieren.

PnP und PCI Setup

Auf der Hardwareparametern-Seite wurden bereits die unterschiedlichen Zuweisungen von IO-Adressen und IRQs bei den verschiedenen Bussystemen angesprochen. Unter diesem Menüpunkt sind in Systemen, die sowohl alte (nicht Plug and Play) ISA-Karten, als auch moderne Plug and Play oder PCI Karten benutzen, ein paar Einstellungen nötig.

Wie schon erwähnt, können Plug and Play Karten (und PCI-Karten) sich selbst die passenden Hardware-Parameter aussuchen. Dabei kann es aber zu Konflikten kommen, wenn andere ISA-Karten, die nicht Plug and Play-fähig sind, die selben Parameter fest eingestellt haben. Das BIOS kann diese feste Einstellung nicht überprüfen.

Aus diesem Grund bietet das Setup-Programm die Möglichkeit, bestimmte IRQs für ISA-Karten zu reservieren, und so zu verhindern, daß Plug and Play-Karten sich diese Parameter aneignen. Wenn wir also alte und moderne Karten gleichzeitig in einem Rechner betreiben, sollten die IRQs, die die alten Karten fest eingestellt haben, hier reserviert werden.

Weitere Einstellmöglichkeiten

Im Setup-Programm existieren noch viele weitere Möglichkeiten, die Hardware einzustellen. Dazu zählen unter anderem die Ansteuerung der Festplatten mit modernen Methoden wie PIO-Modi oder (U)DMA, das korrekte Management des Arbeitsspeichers (Zugriffsmodi, Burst, ...), die Bootreihenfolge oder der Schutz des Setup-Programms durch ein Passwort. Die Beschreibung all dieser Einstellmöglichkeiten würde den Rahmen dieser Darstellung sprengen und ist auch nicht relevant für die LPI 101 Prüfung.

1.101.3 - Konfiguration von Modem und Soundkarten

Beschreibung: Prüfungskandidaten sollten in der Lage sein sicherzustellen, daß die Geräte Kompatibilitätskriterien erfüllen (speziell, daß es sich bei einem Modem NICHT um ein Win-Modem handelt). Ebenfalls enthalten ist die Überprüfung, daß sowohl Modem und Soundkarte eigene und die richtigen Interrupts, I/O- und DMA-Adressen verwenden, die Installation und Ausführung von sndconfig und isapnp bei PnP-Soundkarten, die Konfiguration des Modems für DFÜ und PPP/SLIP/CSLIP-Verbindungen und das Setzen des seriellen Ports auf 115.2 kbps.

Die Konfiguration von Modem und Soundkarte enthält nichts, was hardwaremäßig sich von anderen Karten unterscheiden würde oder was nicht von anderen Themen dieser Prüfungsziele auch schon behandelt wäre. Insbesondere der Abschnitt 1.101.6 - Konfiguration von Kommunikationsgeräten deckt den Bereich Modem vollständig ab.

Kompatibilitätskriterien

Bemerkenswert sind die Kompatibilitätskriterien für diese beiden Gerätearten. Bei der Anschaffung eines Gerätes, das unter Linux betrieben werden soll, ist es natürlich einfach, entsprechend darauf zu achten, daß es sich um Geräte handelt, die auch unter Linux betrieben werden können. Es gibt sowohl Kompatibilitätslisten einzelner Distributoren, als auch unabhängige Aufzählungen, welche Hardware von Linux mit welchem Treiber angesprochen werden kann. Ein guter Startpunkt für die Suche ist hierbei das Hardware-HOWTO des Linux-Documentation-Project.

Schlimmer sieht es aus, wenn mit bestehender Hardware auf Linux umgerüstet werden soll. In diesem Fall kann es hin und wieder vorkommen, daß die entsprechenden Geräte nicht, oder nur eingeschränkt benutzbar sind. Ein typisches Beispiel für solche Geräte sind die sogenannten Winmodems, die hier noch einer genaueren Betrachtung unterworfen werden sollen.

Winmodems

Früher waren alle Modems eigenständige Geräte, die eine eigene Logik und Steuerung enthielten. Das einzige Problem am Anschluß eines Modems war (und ist - bei echten Modems - bis heute) die Einstellung der verwendeten seriellen Schnittstelle und ihrer Kommunikationsparameter. Wenn der Computer das Modem ansprechen konnte, dann konnte er es auch benutzen. Modems benötigen eigentlich keinerlei Treiber, sondern ein Programm, das ihren Befehlssatz kennt. Die normalen Modems arbeiten alle mit einem Befehlssatz, der kompatibel zum früheren Marktführer auf diesem Gebiet, der Firma Hayes, ist. Dieser Befehlssatz beginnt alle Befehle mit einem AT (Attention prefix) und antwortet mit entsprechenden Meldungen wie OK... Ein Programm zur Ansteuerung eines Modems musste also nur diese Befehle kennen und konnte mit dem Modem umgehen. In der Regel konnte

der Benutzer des Programms die entsprechenden Befehle (wählen, auflegen, ...) in eine Konfigurationsmaske eintragen und das Programm konnte dann damit arbeiten.

Moderne, sogenannte Winmodems, arbeiten leider nicht mehr mit diesem Prinzip. Aus Kostengründen ist den Modems die Eigenintelligenz gestrichen worden, das heißt, der Rechner muß den größten Teil der Arbeit leisten. Solche Modems halten sich an keinen Standard, sondern benötigen entsprechende Gerätetreiber. Diese Treiber werden in der Regel nur für Windows ausgeliefert, was es in einigen Fällen unmöglich macht, unter Linux mit diesen Modems zu arbeiten. Das Linux Documentation Project hat ein spezielles Winmodem-Howto zu diesem Thema zusammengestellt. Außerdem gibt es eine Webseite unter www.linmodem.org, auf der Versucht wird, auch Winmodems unter Linux zum Laufen zu bringen.

Architektur von Sound unter Linux

Es gibt unter Linux verschiedene Arten, wie eine Soundkarte zum Laufen gebracht werden kann. Die LPI-Examensziele nennen in diesem Zusammenhang nur die OSS-Technik, die hier also kurz angesprochen werden soll.

Bei OSS (Open Sound System) handelt es sich um eine Treiberschnittstelle zum Kernel, die neben der eigentlichen Schnittstelle noch sogenannte Low-Level Treiber für jede Soundkarte benötigt. Es gibt eine große Menge solcher Treiber für fast alle Soundkarten, die manchmal aber nur rudimentäre Unterstützung gewährleisten. Alle Treiber können als Module oder als fester Bestandteil des Kernels implementiert werden, heute wird sich aber auf die Verwendung von Modulen beschränkt.

Die meisten Soundkarten sind heute PCI-Karten, es existieren aber immer noch alte ISA und ISA-PnP Karten, für die all das gilt, was unter Hardwareparameter und isapnp schon gesagt wurde. Zur einfacheren Installation von Soundkarten hat der Distributor RedHat ein kleines Programm geschrieben, das inzwischen auch für andere Distributionen erhältlich ist, **sndconfig**. Dieses Programm scant die Bussysteme durch und sucht Soundkarten. Falls es sich um ISA-PnP Karten handelt, erledigt **sndconfig** gleich die Aufgabe, mit **isapnp** die entsprechenden Einstellungen menügeführt vorzunehmen.

Voraussetzung für die Verwendung von **sndconfig** ist die Verfügbarkeit von OSS-Treibern und der OSS-Schnittstelle im Kernel. **sndconfig** ist ein menügeführtes Programm, das im Normalfall keine Kommandozeilenparameter benötigt.

1.101.4 - Einrichten von SCSI-Geräten

Beschreibung: Prüfungskandidaten sollten in der Lage sein, SCSI-Geräte unter Verwendung des SCSI-BIOS und der notwendigen Linux-Werkzeuge zu konfigurieren. Sie sollten ebenso fähig sein, zwischen den verschiedenen SCSI-Typen zu unterscheiden. Dieses Lernziel beinhaltet die Handhabung des SCSI-BIOS zum Auffinden von verwendeten und freien SCSI-IDs und zum Setzen der korrekten ID-Nummer für verschiedene Geräte, im speziellen für das Boot-Device. Ebenso enthalten ist das Verwalten der Einstellungen im System-BIOS zur Bestimmung der gewünschten Bootreihenfolge, wenn sowohl SCSI- als auch IDE-Laufwerke verwendet werden.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen

- SCSI-ID
- `/proc/scsi`
- **scsi-info**

Das *Small Computer System Interface* (SCSI) ist jahrelang die Profi-Schnittstelle für Geräte wie Festplatten, CD-ROMs aber auch Scanner und anderer externer Geräte gewesen. Auch heute noch finden sich in den meisten Server-Installationen SCSI Geräte, anstelle der sonst üblichen IDE-Platten. SCSI-Geräte sind in der Regel um einiges teurer als ihre IDE-Pendants, bieten aber oft eine bessere Eignung für den Dauerbetrieb und haben auch sonst Features, die unter IDE nicht oder nur schwer zu realisieren sind (Hot-Pluging).

Grundsätzliche Architektur von SCSI

SCSI ist ein Bussystem, das bis zu 8 Geräte (16 Geräte bei *Wide-SCSI*) verwalten kann. Eines der 8 (oder 16) Geräte ist der SCSI-Hostadapter selbst. Der Hostadapter (oft fälschlicherweise als SCSI-Controller bezeichnet) ist das SCSI-Gerät, das den Bus mit dem Computer verbindet.

Grundsätzlich besteht ein SCSI System aus einem Bus, der in der Regel im Inneren des Computers durch ein 50poliges Flachbandkabel realisiert wird. Nach außen hin (für externe Geräte wie Scanner) wird ein abgeschirmtes Spezialkabel verwendet. An diesem Bus hängen die SCSI-Geräte, auch der Hostadapter.

Beim Aufbau des Busses müssen folgende Punkte beachtet werden:

- Der Bus darf nur zwei Enden haben, T-Stücke (Abzweigungen) sind verboten.
- Jedes Ende des Busses muß mit einem Terminator (Abschlußwiderstand) abgeschlossen sein. (Die Werte betragen 330 Ohm gegen Masse und 220 Ohm gegen +5V.) Heute werden diese Widerstände entweder softwaremäßig oder gar automatisch gesetzt. Bei älteren SCSI-Geräten war hier noch Steckarbeit erforderlich.
- Der Leitungsabstand zwischen zwei SCSI Geräten muß mindestens 10 cm betragen.
- Der SCSI Bus darf eine Gesamtlänge von drei Metern nicht überschreiten. (Bei mehr

als vier Geräten darf er bei klassischem SCSI sogar nur 1,5 Meter lang sein.)
- Bastlerverbindungen (Pfostenstecker o.ä.) sind nicht zulässig, das Flachbandkabel muß durchgängig sein.
- Der Bus darf nicht durch ein offenes Kabelende abgeschlossen sein, d.h., daß selbst wenn nur ein Gerät (außer dem Adapter) angeschlossen ist immer der letzte Anschluß des Kabels verwendet werden muß.

Diese Regeln sind unbedingt einzuhalten, sonst kommt es mit Sicherheit zu Fehlern, deren Ursache oft sehr schwer ermittelbar ist.

Unterschiedliche SCSI-Typen

SCSI ist ein gewachsenes System, das in verschiedenen Generationen auftritt. Die ursprünglichen Bezeichnungen SCSI, SCSI2 und SCSI3 haben mehr zur Verwirrung beigetragen, als zur Klärung, daher werden heute die folgenden Begriffe verwendet:

SCSI (oder SCSI1)
: Das klassische SCSI, bis zu 8 Geräte (7+Hostadapter) an einem 50poligen Kabel. Es wurde 1986 standardisiert, seitdem können alle SCSI-Hostadapter Geräte ansteuern, die diesem Standard folgen.

SCSI2
: Eine Erweiterung des SCSI-Befehlssatzes, die hauptsächlich auch die Ansteuerung anderer Geräte als Festplatten ermöglichte. Diese Form von SCSI wurde weiter aufgespalten in eine schnellere Variante (Fast SCSI2) und eine Variante mit der Möglichkeit mehrere Geräte (16 statt 8) anzuschließen (Wide-SCSI-2).

SCSI3 (oder Ultra SCSI)
: SCSI3 ist noch nicht vollständig fertig entwickelt, trotzdem bieten es einige Hersteller schon an. Es zeichnet sich durch eine noch höhere Geschwindigkeit aus. Für SCSI3 existieren auch völlig neue Kabelformen (Glasfaser, serielle Übertragung), die aber keine Bedeutung für die LPI-Prüfung haben.

Die Wide-SCSI Varianten benutzen statt einem 50poligen Kabel (8-Bit-Bus) ein 68poliges Kabel (16 Bit Bus), damit die einzelnen Geräte adressiert werden können. Die folgende Tabelle zeigt die verschiedenen Kombinationsmöglichkeiten, die auf dem Markt sind, samt ihren maximalen Transferraten:

Busbreite	Kabelbreite	Standard	Fast	Ultra
8-Bit	50-polig	5 MB/sec	10 MB/sec	20 MB/sec
16-Bit	68-polig	10 MB/sec	20 MB/sec	40 MB/sec

Durch entsprechende Kombinationen entstehen dadurch etwas verständlichere Namen wie Ultra-Wide-SCSI oder Fast-Wide-SCSI,...

Adressierung im SCSI-Bus

Jedes SCSI Gerät muß über eine SCSI-ID verfügen, also einer numerischen Adresse, über

die es eindeutig ansprechbar ist. Diese Eindeutigkeit bedeutet, daß kein SCSI Gerät in einem Bus die gleiche ID wie ein anderes haben darf. Die IDs beginnen mit der Null und enden mit der Sieben (Normales SCSI) oder der 15 (Wide-SCSI) Es gibt inzwischen auch schon Versuche mit 32 Geräten, die erfordern jedoch ganz andere Kabel und zum Teil auch andere Protokolle. (Es ist schlicht unmöglich auf einem 1,5 Meter langen Kabel 32 Geräte mit einem Mindestabstand von 10 cm anzuschließen.)

Der Hostadapter hat üblicherweise die ID 7. Die Vergabe der ID sollte nicht wahllos geschehen, den die höchste ID hat immer auch die höchste Priorität im System. Hingegen hat die ID nichts mit der physikalischen Position am Bus zu tun.

Die IDs müssen entweder softwaremäßig (Adapter) oder über Schalter (externe Geräte) und Jumper (interne Geräte) eingestellt werden. Falls eine reine SCSI-Installation ohne IDE Platten eingerichtet werden soll, muß die Bootfestplatte die ID 0 haben.

Neben der SCSI-ID gibt es für jedes Gerät noch eine sogenannte LUN (*Logical Unit Number*), die eventuell existierende Untergeräte addressiert. So kann etwa ein SCSI-Plattenschrank mit 5 Festplatten aus der Sicht des SCSI-Busses nur ein Gerät mit einer SCSI-ID sein. Um die einzelnen Platten trotzdem ansprechen zu können werden die eigentlichen SCSI-IDs der Platten jetzt zu den LUNs des Plattenschrankes.

Um auch mehrere SCSI-Hostadapter in einem Rechner betreiben zu können, bekommt auch der SCSI-Bus selbst eine Nummerierung. Auch diese Busnummer fängt mit 0 zu zählen an, der erste SCSI-Bus hat also die Busnummer 0, der zweite die 1 usw.

Aus der Kombination der drei Angaben (durch Komma getrennt) lässt sich so eine eindeutige Adressierung innerhalb eines Systems erreichen. Jedes SCSI-Gerät kann mit der Adresse

Busnummer,SCSI-ID,LUN

angesprochen werden. In den meisten Fällen werden Busnummer und LUN jeweils auf 0 gesetzt sein (wenn nicht mehrere SCSI-Adapter oder Geräte mit Untergeräten im Einsatz sind).

Das SCSI-BIOS

SCSI ist ein intelligentes Subsystem, von dem das Standard-BIOS des PC nichts weiß. Die Ansteuerung von SCSI-Geräten erfolgt durch entsprechende Gerätetreiber des Betriebssystems. Das ist weiter kein Problem, wenn das Betriebssystem von einem Medium gebootet werden kann, das vom BIOS erkannt wird. Auf einem System, das nur SCSI-Platten aufweist, ist das aber nicht möglich.

Auf einem solchen System muß ein Hostadapter mit eigenem BIOS installiert werden, damit auch von SCSI-Geräten gebootet werden kann. In diesem Punkt unterscheiden sich die Hostadapter stark voneinander. Billige Adapter, wie sie etwa beim Kauf eines SCSI-Scanners mitgeliefert werden, besitzen kein eigenes BIOS sondern dienen ausschließlich der Ansteuerung von SCSI-Geräten im laufenden Betrieb. Teurere Adapter weisen ein eigenes BIOS auf, das - analog zum BIOS des Computers selbst - auch ein Setup-Programm

beinhaltet.

Durch eine Tastenkombination beim Hochfahren des Rechners (etwa Strg-A bei Adaptec-Adaptern) wird das Setup-Programm des SCSI-BIOS gestartet. Hier können verschiedene Einstellungen vorgenommen werden, die den Adapter selbst und die angeschlossenen Geräte betreffen (etwa seine SCSI-ID) und Informationen über alle angeschlossenen Geräte ermittelt werden. Dazu zählen insbesondere

- Ermittlung aller vergebenen SCSI-IDs
- Einstellungen die Transferrate betreffend
- Einstellung, von welchem Gerät gebootet werden soll

Manche modernen SCSI-Geräte sind auch in der Lage, ihre SCSI-ID softwaremäßig einzustellen, was dann auch innerhalb dieses Programms vorgenommen werden kann.

Booten von SCSI-Geräten

Wenn von einem SCSI-Gerät gebootet werden soll, müssen die oben genannten Einstellungen im SCSI-BIOS vorgenommen werden. SCSI-Hostadapter ohne eigenes BIOS eignen sich nicht, um zu booten.

Im System-BIOS muß dann noch eingestellt werden, daß von einem SCSI-Gerät gebootet werden soll. Um welches Gerät es sich handelt, kann das System-BIOS nicht bestimmen, dazu hat der Hostadapter ja sein eigenes BIOS. Bei der Einstellung der Boot-Reihenfolge im System-BIOS (Bootsequence) wird einfach nur SCSI angegeben. In diesem Fall übergibt das System-BIOS die Aufgabe des Bootens an das SCSI-BIOS, das wiederum über die entsprechenden Informationen verfügt, von welchem Gerät gebootet werden soll.

SCSI im Linux-System

Im /proc-Verzeichnis findet sich ein Unterverzeichnis `scsi`, das alle gefundenen SCSI-Hostadapter wiederum als Unterverzeichnis enthält. Jedes dieser Unterverzeichnisse enthält wiederum eine Datei, die eine Nummer als Namen trägt. Das ist die Nummer, mit der der SCSI-Bus an diesem Adapter bezeichnet wird. Die Datei selbst enthält Informationen über den entsprechenden Adapter.

Ein Beispiel: Auf einem Linux-System sind ein Adaptec-SCSI Adapter AHA-2940 und ein Zip-Laufwerk am Parallelport installiert. Auch das Zip-Laufwerk ist ein SCSI-Gerät, der Parallelport dient hier als SCSI-Adapter (PPA-ParallelPort Adapter). Im Verzeichnis `/proc/scsi` findet sich folgender Inhalt:

```
dr-xr-xr-x    ...    aic7xxx
  |
  +   -rw-r--r--    ...  0

dr-xr-xr-x    ...    ppa
  |
  +   -rw-r--r--    ...  1
```

```
-r--r--r--    ...    scsi
```

Jedes der beiden Verzeichnisse enthält eine Datei. Das Verzeichnis `aic7xxx` enthält eine Datei mit Namen `0`, das Verzeichnis `ppa` eine mit Namen `1`. Der Bus am Adaptec-Adapter ist also Bus 0, der am Parallelport Bus 1. Die Datei `scsi` enthält Informationen über alle angeschlossene Geräte, egal auf welchem Bus sie hängen.

Die Namensgebung von SCSI-Geräten unter Linux

Die Namen der SCSI-Geräte unter Linux sind abhängig von der Reihenfolge, in der die Hostadapter beim Booten erkannt werden. Die einzelnen Platten werden anhand der Reihenfolge ihrer Erkennung durchnummeriert. So ist die erste erkannte Festplatte auf dem ersten erkannten SCSI-Adapter die Platte mit dem Namen `/dev/sda`. Die nächste Platte des selben Adapters bekäme den Namen `/dev/sdb`. Sind keine weiteren Platten mehr angeschlossen, beginnt das gleiche Spiel beim nächsten Adapter wieder. Die dortige Platte, die zuerst erkannt wurde wäre also in unserem Beispiel `/dev/sdc`...

CDROM-Laufwerke bekommen entsprechend die Namen `/dev/sr0, /dev/sr1`,... oder auch (gleichzeitig) `/dev/scd0, /dev/scd1`,... Das `sr` steht für SCSI-Removable also SCSI-Wechselplatte.

Eine andere Form der Adressierung wäre eindeutiger, die schon bekannte Form

Busnummer,SCSI-ID,LUN

Um die beiden Adressierungsformen zueinander in Verbindung zu bringen, existiert das kleine Programm **scsi_info**, dem als Parameter der Name der Gerätedatei eines beliebigen SCSI-Gerätes mitgegeben wird. Als Ausgabe gibt das Programm dann die Adresse in der Form `Busnummer,SCSI-ID,LUN` und weitere Informationen aus `/proc/scsi/scsi`. Ein Aufruf von

```
scsi_info /dev/sda
```

ergibt beispielsweise eine Ausgabe wie

```
SCSI_ID="0,2,0"
MODEL="IBM DFHSS1F       !c"
FW_REV="1717"
```

Mit diesem Hilfsmittel ist es also möglich, die tatsächlichen physikalischen Adressen zu ermitteln.

1.101.5 - Einrichtung verschiedener PC-Erweiterungskarten

Beschreibung: Prüfungskandidaten sollten in der Lage sein, verschiedene Karten für die unterschiedlichen Erweiterungssteckplätze zu konfigurieren. Sie sollten die Unterschiede zwischen ISA- und PCI-Karten in Hinsicht auf Konfigurationsfragen kennen. Dieses Lernziel beinhaltet das korrekte Setzen von Interrupts, DMAs und I/O-Ports der Karten, speziell um Konflikte zwischen Geräten zu vermeiden. Ebenfalls enthalten ist die Verwendung von isapnp, wenn es sich um eine ISA PnP-Karte handelt.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen

- `/proc/dma`
- `/proc/interrupts`
- `/proc/ioports`
- `/proc/pci`
- **pnpdump(8)**
- **isapnp(8)**
- **lspci(8)**

Alles, was für dieses Thema wichtig ist, wurde schon in den Abschnitten

- Hardwareparameter
- Das /proc-Verzeichnis
- Plug and Play mit Linux

besprochen. Hier also nur noch eine kurze Zusammenfassung:

Unterschied zwischen ISA und PCI

Die Unterschiede zwischen ISA-Karten und PCI Karten wurden im Groben bereits unter
- Hardwareparameter besprochen. Wichtig ist hier insbesondere, daß PCI ein System ist, das die bei ISA eher wackelige Funktionalität des Plug and Play wirklich standardisiert übernommen hat. Das heißt, es existieren weltweite Standards, wie sich eine PCI-Karte gegenüber dem System ausweist. Jeder Hersteller von PCI-Karten ist zentral registriert (ähnlich wie bei der Vergabe der Hardware-Adressen bei Netzwerkkarten) und hat eine eindeutige ID. Jedes Gerät, das der Hersteller produziert, bekommt wiederum eine eindeutige ID, die bei der Bestimmung der Funktionalität hilft. Zudem hat jede PCI-Karte die entsprechenden Informationen (IDs und ausgeschriebene Beschreibung) auf einem kleinen ROM-Speicherstein gespeichert und kann sich so gegenüber einem System ausweisen. Linux speichert die Informationen über diese IDs in der Datei `/usr/share/misc/pci.ids`.

Im Gegensatz zu ISA-Karten ist also bei PCI Karten niemals die manuelle Vergabe von Hardware-Parametern (IO-Ports, IRQs, DMA-Kanäle) notwendig. Linux unterstützt die PCI-

Technik vollständig.

Die ISA Karten benötigen unter Linux grundsätzlich immer diese Parameter, egal, ob sie alte Karten mit manueller Einstellung sind, oder Plug and Play Karten. Bei den letzteren wird das Programm isapnp benutzt, um die Einstellungen festzulegen, beim Laden der Module müssen die Einstellungen aber trotzdem vorgenommen werden.

Um von Vorneherein Mißverständnisse auszuschließen, hier noch eine kurze Beschreibung des AGP-Busses für Graphikkarten. Der AGP-Bus ist - technisch gesehen - ein eigenständiger PCI-Bus mit einer anderen Bauform der Steckverbinder. Dieser Bus hat nur einen Steckplatz und der ist für die Graphikkarte gedacht. Der Grund für diese Technik liegt in dem besonders großen Datentransfer-Volumen von Graphikkarten. Um einen ungebremsten Transfer zwischen Computer und Monitor sicherzustellen, wurde für diese Aufgabe ein eigenständiger Bus entwickelt, bei dem sich die Graphikkarte die Leistung des Bussystems nicht mit anderen Karten teilen muß.

PCI-Bussysteme werden intern ähnlich adressiert wie SCSI-Systeme. Auch hier gibt es eine dreigeteilte Adressierung in der Form

Busnummer:Steckplatznummer.Funktionsnummer

Die Busnummer für den normalen PCI-Bus ist 00, die für den AGP-Bus ist 01. Die beiden Bussysteme werden also tatsächlich als unterschiedliche Systeme behandelt. Die Funktionsnummer steht für das jeweilige Gerät, das die entsprechende Funktionalität zur Verfügung stellt.

Das Programm lspci

Damit Linux in die Lage versetzt wird, den PCI-Bus zu scannen, existiert das Programm **lspci**. **lspci** ist ein Hilfsmittel, um Informationen über alle PCI-Bussysteme des Systems und alle dort angeschlossenen Geräte darzustellen.

Das Programm **lspci** ermöglicht es, genau zu bestimmen, welche Geräte (Karten) am PCI-Bus angeschlossen sind und als was sie sich ausgeben. Das ist insbesondere wichtig, um herauszufinden, welche Chipsätze oder Kompatibilitätskriterien bestimmte Karten aufweisen, um mit dieser Information dann das entsprechende Modul zu laden. Moderne Linux-Distributionen, die bei der Installation selbstständig herausfinden, welche Karten angeschlossen sind (Hardwareerkennung) bedienen sich dieses Programms.

Die eigentlichen Informationen, welche Hersteller und welche Geräte die gefundenen Ergebnisse repräsentieren, wird die systemweite Datenbank `/usr/share/misc/pci.ids` benutzt. Durch den Parameter `-n` wird **lspci** dazu gezwungen, diese Datei nicht zu konsultieren, sondern die Ergebnisse numerisch auszugeben.

Linux Kernel, die neuer als die Version 2.1.82 sind, stellen zudem im Verzeichnis `/proc/pci` weitere Informationen (binär) zur Verfügung. Das Verzeichnis enthält für jeden PCI-Bus ein entsprechendes Unterverzeichnis, das wiederum für jede angeschlossene Karte eine Datei beinhaltet.

1.101.6 - Konfiguration von Kommunikationsgeräten

Beschreibung: Prüfungskandidaten sollten in der Lage sein, verschiedene interne und externe Kommunikationsgeräte wie Modems, ISDN-Adapter und DSL-Router zu installieren und zu konfigurieren. Dieses Lernziel beinhaltet die Prüfung von Kompatibilitätskriterien (besonders wichtig, wenn es sich um ein Winmodem handelt), notwendige Hardwareeinstellungen für interne Geräte (Interrupts, DMAs, I/O-Adressen) und das Laden und Konfigurieren von passenden Gerätetreibern. Ebenfalls enthalten sind Anforderungen an die Konfiguration von Kommunikationsgeräten und -schnittstellen, wie z.B. der korrekte serielle Port für 115.2 kbps und korrekte Modemeinstellungen für ausgehende PPP-Verbindungen.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen

- `/proc/dma`
- `/proc/interrupts`
- `/proc/ioports`
- **setserial(8)**

Die Installation von Kommunikationsgeräten unterscheidet sich nicht von der anderer Erweiterungsgeräte. Also gilt für diese Geräte wiederum all das, was unter

- Hardwareparameter
- Das /proc-Verzeichnis
- Plug and Play mit Linux

schon gesagt wurde. Auch die bereits gemachten Erklärungen zu Kompatibilitätskriterien insbesondere von WIN-Modems wurden in Konfiguration von Modem und Soundkarten bereits besprochen. Die konkrete Einrichtung von PPP wird in der Vorbereitung auf die Prüfung LPI102 noch Thema werden.

Setzen der seriellen Schnittstelle auf 115.2 kbps mit setserial

Das Programm zur Konfiguration serieller Schnittstellen heißt **setserial**. **setserial** wurde dazu entworfen, Konfigurationsinformation einer seriellen Schnittstelle zu setzen oder zu lesen. Diese Information beinhaltet welche IO-Ports und IRQs serielle Schnittstellen benutzen und mit welchem Multiplikator Hochgeschwindigkeitsverbindungen bearbeitet werden.

Normalerweise wird **setserial** über ein init-Script beim Systemstart für alle angeschlossenen seriellen Schnittstellen gestartet um sie zu konfigurieren. Ohne besondere Parameter gibt **setserial** die Informationen über eine bestimmte serielle Schnittstelle aus.

Die vier standardmäßigen seriellen Schnittstellen des PC werden als `/dev/ttyS0` bis `/dev/ttyS3` bezeichnet. Mit diesen Bezeichnungen kann **setserial** arbeiten. Der Aufruf von

```
setserial /dev/ttyS0
```

gibt uns beispielsweise folgende Ausgabe:

```
/dev/ttyS1, UART: 16550A, Port: 0x02f8, IRQ: 3
```

Um jetzt Einstellungen vorzunehmen, müssen dem Programm **setserial** mehr Parameter mitgegeben werden. Die Aufrufform ist dabei

```
setserial Gerätedatei Parameter
```

Wichtige solcher Einstellungsparameter sind

port *Portnummer*
: setzt die IO-Portadresse der angegebenen Schnittstelle

irq *IRQ-Nummer*
: setzt den IRQ der angegebenen Schnittstelle

uart *UART-Typ*
: setzt den verwendeten UART Baustein (Universal Assynchronous Receiver Transmitter - der Chip, der die ser. Schnittstelle ansteuert). Gültige Werte sind: none, 8250, 16450, 16550, 16550A, 16650, 16650V2, 16654, 16750, 16850, 16950, und 16954. Der Wert none schaltet die Schnittstelle aus.

Viele Anwendungen, die mit seriellen Schnittstellen und Modems arbeiten kennen nur Geschwindigkeitseinstellungen bis zu 38,4 kb. Moderne Modems erfordern aber wesentlich schnellere Verbindungen zwischen Rechner und Modem, da diese Modems selbst noch Datenkompression anbieten, also mehr Daten zwischen Rechner und Modem transportiert werden müssen, als zwischen Modem und Modem. Damit auch ältere Anwendungen mit diesen Hochgeschwindigkeitsschnittstellen arbeiten können, bietet **setserial** die Möglichkeit an, verschiedene Flags zu setzen, die eine höhere Geschwindigkeit anschalten, wenn die Anwendung 38,4 kb anfordert.

spd_normal
: Benutzt 38,4kb wenn die Anwendung 38,4kb einstellt.

spd_hi
: Benutzt 57,6kb wenn die Anwendung 38,4kb einstellt.

spd_vhi
: Benutzt 115kb wenn die Anwendung 38,4kb einstellt.

spd_shi
: Benutzt 230kb wenn die Anwendung 38,4kb einstellt.

spd_warp
: Benutzt 460kb wenn die Anwendung 38,4kb einstellt.

Natürlich muß der verwendete UART diese Geschwindigkeiten tatsächlich unterstützen. Um also die von LPI geforderte Einstellung von 115kb auf der Schnittstelle `/dev/ttyS0`

vorzunehmen schreiben wir:

```
setserial /dev/ttyS0 spd_vhi
```

ein erneuter Aufruf von

```
setserial /dev/ttyS0
```

bringt uns jetzt die Ausgabe:

```
/dev/ttyS1, UART: 16550A, Port: 0x02f8, IRQ: 3, Flags: spd_vhi
```

Wenn eine Anwendung jetzt 38,4kb anfordert, wird die Schnittstelle in Wahrheit mit 115kb angesteuert.

Konfiguration eines Modems

Ein Modem wird über einen seriellen Port angeschlossen. Entweder ist es ein externes Modem, dann wird die Verbindung direkt über einen existierenden Port und ein serielles Kabel hergestellt, oder das Modem ist eine interne Steckkarte, die einen eigenen - zusätzlichen - seriellen Port enthält. Auch dieser zusätzliche serielle Port muß entsprechend mit **setserial** konfiguriert werden.

Modems werden über einen Befehlssatz gesteuert, der nicht hundertprozentig standardisiert ist. Grundsätzlich beginnen Modembefehle mit dem Wort AT, dem weitere Zeichen folgen. Die genauen Befehle müssen dem jeweiligen Modemhandbuch entnommen werden. Hier ein kleiner Überblick über die Befehle, die in der Regel alle Modems kennen. Jeder der folgenden Befehle muß mit einem Enter abgeschlossen werden, damit ihn das Modem überhaupt erst ausführt:

AT	Das Modem antwortet mit OK. Tut es das nicht, so stimmt etwas mit der Kommunikation zwischen Modem und Rechner nicht.
ATD *Nummer*	Das Modem wählt die angegebene Nummer und baut eine Verbindung auf.
ATH0	Modem legt auf
ATH1	Modem nimmt ab
ATX*Ziffer*	Stellt das Modem-Verhalten beim Verbindungsaufbau ein.

- ATX0 - Modem wählt und meldet CONNECT bei erfolgreichem Verbindungsaufbau
- ATX1 - Modem wählt und meldet CONNECT (Geschwindigkeit)
- ATX2 - Modem wartet auf Freizeichen, wählt und meldet CONNECT (Geschwindigkeit).
- ATX3 - Modem wählt und meldet CONNECT (Geschwindigkeit) oder BUSY (belegt). X3 sollte bei Nebenstellenanlagen verwendet werden, um das Warten auf ein Freizeichen der Amtsleitung zu vermeiden.
- ATX4 - X4 Modem wartet auf Freizeichen, wählt und meldet CONNECT (Geschwindigkeit).

- ATX5 - Modem wählt und meldet CONNECT (Geschwindigkeit), Busy, Voice (Telefon anstatt eines Modems an der Gegenstelle).
- ATX6 - Modem wartet auf Freizeichen, wählt und meldet CONNECT (Geschwindigkeit), Busy (belegt), Voice.

ATS0=0	Das Modem nimmt nicht selbstständig Anrufe entgegen. ATS0=1 bedeutet das Modem nimmt nach einem Läuten ab, S0=2 wäre nach zweimal Klingeln abheben ...
AT&W	Das Modem speichert seine gegenwärtige Einstellung
ATZ	Das Modem liesst die gespeicherte Einstellung und aktiviert sie.

Jedes Datenfernübertragungsprogramm wie z.B. **minicom**, aber auch andere Programme, die mit Modems arbeiten, wie etwa die Programme, die PPP und SLIP Verbindungen aufbauen, müssen diese Befehle nutzen. Normalerweise wird entweder ein Modeminitialisierungsbefehl angegeben, der alle notwendigen Einstellungen enthält wie etwa

```
ATS0=0 X3
```

oder es ist vorher dafür gesorgt worden, daß das Modem seine Einstellungen gespeichert hat, dann genügt ein

```
ATZ
```

Nun sollte das Modem für ausgehende Verbindungen konfiguriert sein.

1.101.7 - Konfiguration von USB-Geräten

Beschreibung: Prüfungskandidaten sollten in der Lage sein, USB-Unterstützung zu aktivieren und verschiedene USB-Geräte zu verwenden und zu konfigurieren. Dieses Lernziel beinhaltet die korrekte Auswahl des USB-Chipsatzes und des dazugehörigen Moduls. Ebenfalls enthalten ist das Wissen über die allgemeine Architektur des USB-Schichtenmodells und die verschiedenen Module, die in den einzelnen Schichten verwendet werden.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen

- **lspci(8)**
- `usb-uhci.o`
- `usb-ohci.o`
- `/etc/usbmgr/`
- **usbmodules**
- `/etc/hotplug`

Grundsätzliche Unterstützung von USB

USB wird von Kerneln ab Version 2.2.7 unterstützt. Besser sind die Kernel ab 2.4.0. Die generelle Unterstützung von USB ist heute meist fest in einen Kernel hineingebaut, kann aber auch als Modul realisiert werden. Dieses Modul muß - falls mit USB gearbeitet werden soll - entsprechend geladen werden. Das Modul heißt `usbcore.o` und kann mit dem Befehl

```
modprobe usbcore
```

im laufenden Betrieb eingehängt werden. Üblich ist aber, daß diese grundsätzliche USB-Unterstützung fest integriert ist. Beim Booten sollte - falls das der Fall ist - eine Meldung wie die folgende angezeigt werden:

```
usb.c: registered new driver usbdevfs
usb.c: registered new driver hub
Initializing USB Mass Storage driver...
usb.c: registered new driver usb-storage
USB Mass Storage support registered.
```

Damit ist klar, daß die USB-Unterstützung grundsätzlich aktiviert ist.

UHCI oder OHCI

Die meisten modernen Mainboards stellen einen USB-Controller zur Verfügung. Ältere Maschinen können mit einem USB-Controller auf einer PCI-Karte nachgerüstet werden. USB-Controler sind entweder kompatibel zum *Open Host Controller Interface* (OHCI - Compaq)

oder zum *Universal Host Controller Interface* (UHCI - Intel). Beide Typen haben die selben Fähigkeiten und USB-Geräte arbeiten mit beiden Typen zusammen.

Die wesentliche Aufgabe bei der Konfiguration von USB unter Linux ist es, herauszufinden, welchen der beiden Controller auf dem Rechner verwendet wird, auf dem USB konfiguriert werden soll.

Mainboards der folgenden Firmen benutzen UHCI:

- Intel
- VIA (auch VIA basierte USB-Adapter)

Mainboards der folgenden Firmen benutzen OHCI:

- Compaq
- Ali
- SiS
- NEC
- Opti Chipsatz

Eine gute Methode, um den entsprechenden Chipsatz herauszufinden ist ein Aufruf von **lspci**. Eine typische Ausgabe wäre z.B.

```
...
00:07.1 IDE interface: VIA Technologies, Inc. Bus Master IDE (rev 06)
00:07.2 USB Controller: VIA Technologies, Inc. USB (rev 16)
00:07.3 USB Controller: VIA Technologies, Inc. USB (rev 16)
00:07.4 Bridge: VIA Technologies, Inc. VT82C686 [Apollo Super ACPI] (rev 40)
...
```

was eindeutig VIA - also UHCI bedeutet.

Falls das nicht weiterführt, gibt es die Möglichkeit, die Datei `/proc/pci` anzusehen. Dort findet sich ein Eintrag in der Art:

```
Bus  0, device   7, function  3:
  USB Controller: VIA Technologies, Inc. UHCI USB (#2) (rev 22).
    IRQ 10.
    Master Capable.  Latency=32.
    I/O at 0xa800 [0xa81f].
```

Wenn auch hier keine Angaben stehen, dann sollte die IO-Adresse angesehen werden. Ist sie in der Form `0xHHHH` (wobei HHHH für vier hexadezimale Ziffern steht), dann haben wir es mit UHCI zu tun, ansonsten, wenn die Form der Adresse `0xHH000000` ist, dann ist es OHCI.

Sobald herausgefunden wurde, welcher Controller benutzt wird, kann das Grundmodul für USB geladen werden. Das ist eines der beiden folgenden Module

- usb-ohci.o
- usb-uhci.o

Geladen werden diese Module entweder von Hand über den Befehl

```
modprobe usb-uhci
```

oder

```
modprobe usb-ohci
```

oder automatisiert über einen entsprechenden Eintrag in `/etc/modules.conf` etwa in der Art:

```
alias usb uhci
```

Jetzt kann in einer Systemstartdatei ein `modprobe usb` eingetragen werden und das entsprechende Modul wird automatisch beim Start geladen. Es ist sogar möglich dadurch alle gewünschten anderen USB-Module gleich mitzuladen, etwa indem dort eingetragen wird

```
alias usb uhci
post-install uhci modprobe printer
post-install printer modprobe joydev
post-install joydev modprobe hid
```

Das USBDevFS-Dateisystem

Das USB Geräte-Dateisystem ist ein dynamisch generiertes Dateisystem, ähnlich dem `/proc` Dateisystem. Es kann theoretisch an jede beliebige Stelle des Systems gemountet werden, es hat sich aber durchgesetzt, es nach `/proc/bus/usb` einzuhängen.

Wenn im Kernel die Unterstützung für dieses Dateisystem aktiviert ist (Preliminary USB Device Filesystem) dann sollte es - bei 2.4er Kerneln automatisch beim Start gemountet werden. Ältere Kernel erfordern entweder Handarbeit wie

```
mount -t usbdevfs none /proc/bus/usb
```

oder einen Eintrag nach `/etc/fstab` in der Art

```
none    /proc/bus/usb      usbdevfs   defaults   0   0
```

Nachdem das Dateisystem gemountet ist, sollte der Inhalt von `/proc/bus/usb` etwa folgendermaßen aussehen:

```
-r--r--r--   1 root     root            0 2002-08-20 00:02 devices
-r--r--r--   1 root     root            0 2002-08-20 00:02 drivers
```

Sobald das entsprechende Modul `usb-ohci.o` oder `usb-uhci.o` geladen ist, der USB-

Controller also ansprechbar ist, sollte mindestens noch ein (meist aber zwei) Unterverzeichnis zu finden sein, für jede USB-Schnittstelle eines:

```
dr-xr-xr-x   1 root     root            0 2002-08-20 00:11 001
dr-xr-xr-x   1 root     root            0 2002-08-20 00:11 002
-r--r--r--   1 root     root            0 2002-08-20 00:11 devices
-r--r--r--   1 root     root            0 2002-08-20 00:11 drivers
```

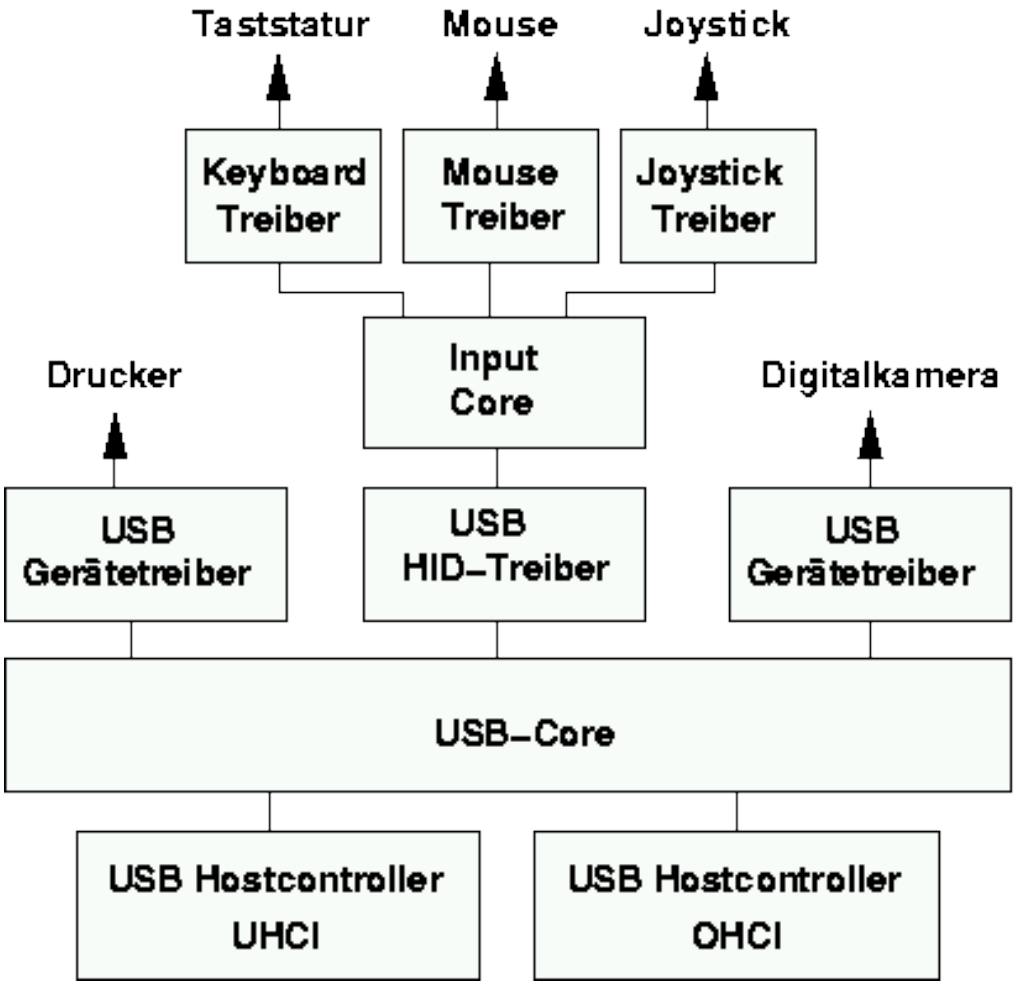
Die Einträge in diesen Verzeichnissen und Dateien sind binär und somit nicht für menschliche Augen gedacht. Es geht darum, im Fehlerfall Zugriff auf jedes einzelne Gerät zu haben und Diagnosen stellen zu können.

Das USB-Schichtenmodell von Linux

Um zu verstehen, welche Module wie voneinander abhängen, folgt hier eine Darstellung des USB-Schichtenmodells, wie es der Linux-Kernel benutzt. Daraus wird klar, daß der USB-Kern (usbcore) nicht die unterste Schicht des USB-Stapels ist, sondern in beide Richtungen erweiterbar ist.

Nach "unten" hin, also Richtung Controller-Hardware sind ihm die bereits besprochenen USB-Hostcontroller-Module (`usb-ohci` und `usb-uhci`) angeschlossen, mit denen er in die Lage versetzt wird, den entsprechenden USB-Controller anzusprechen. Nach "oben" hin folgen die Module, die für die einzelnen Geräte selbst notwendig sind.

Der Kern (usbcore) selbst stellt die Funktionalität von USB selbst zur Verfügung, also alles, was für alle Geräte gleich ist.



Die einzelnen Geräte, die über USB angeschlossen werden, können in sogenannte Klassen aufgeteilt werden. Als Klassen werden Geräte zusammengefasst, die eine bestimmte Menge gleichartiger Mechanismen benutzen. Unter Linux ist die Aufteilung nicht besonders intensiv, die einzige große Klasse, die sich ergibt ist die Klasse HID. (*Human Interface Device* - Gerät für eine menschliche Schnittstelle). Diese Klasse fasst alles zusammen, was als Eingabegerät des Benutzers klassifiziert werden kann. Also z.B. Tastatur, Mouse und Joystick. Das entsprechende Modul unter Linux heißt `hid.o`.

Diese Klasse wiederum benutzt ein weiteres Modul, das die Eingabe (*input*) von solchen Geräten regelt. Unter Linux wird diese Aufgabe vom Modul `input.o` gelöst. Erst darauf bauen dann die einzelnen Module für die jeweiligen Eingabegeräte wie etwa `usbkbd.o` oder `usbmouse.o` auf.

Durch diese Architektur ist es relativ wenig Aufwand, neue Treiber für Eingabegeräte zu schreiben. Die wesentliche Funktionalität ist ja schon durch die Module `hid.o` und `input.o` gegeben. Nur noch die spezifischen Besonderheiten des jeweiligen Gerätes müssen in den entsprechenden Treiber integriert werden.

Andere Geräte, die nicht zur HID-Klasse gehören, verwenden nur jeweils ein Modul, um die entsprechende Funktionalität zur Verfügung zu stellen. So wird z.B. für den Einsatz eines USB-Druckers nur das Modul `printer.o` benötigt. Dieses Modul erstellt dann eine (oder mehrere) Gerätedatei(en) im Verzeichnis `/dev/usb`, die wie parallele Schnittstellen zu benutzen sind (`/dev/usb/lp0`), um einen dort angeschlossenen Drucker entsprechend anzusprechen.

Wenn also an USB kein Eingabegerät angehängt ist, so kann auf den Einsatz des gesamten HID/Input Astes verzichtet werden.

Genauere Informationen, über die gefundenen (angeschlossenen) USB-Geräte sind mit dem Programm lsusb zu ermitteln.

Module für die USB-Geräte automatisch einbinden

USB ist von seinem ganzen Ansatz her ein sogenannter *Hotplugging Service*, also ein Dienst, der im laufenden Betrieb ein- und ausgehängt werden kann. Diese Fähigkeit wird von Linux unterstützt, es müssen dazu aber ein paar Voraussetzungen erfüllt sein.

Grundsätzlich existieren zwei unterschiedliche Ansätze, um diese Fähigkeit anzubieten:

- **hotplug**
  Ein Programm, das die Hotplugging-Fähigkeit für viele verschiedene Systeme anbietet, etwa USB oder PCMCIA.
- **usbmgr**
  Ein Programm, das die Hotplugging-Fähigkeit nur für USB anbietet.

Beide Ansätze werden hier kurz beschrieben:

Funktionsweise von hotplug

hotplug ist ein Programm, das vom Kernel benutzt werden kann, um User-Programme von bestimmten Ereignissen (zumeist Hardware-spezifisch) zu unterrichten. Ein solches Ereignis kann beispielsweise das Anstecken eines USB- oder PCMCIA Gerätes sein. Das kann brauchbar sein, um die entsprechenden Module für dieses Gerät automatisch zu laden.

hotplug benutzt für jedes zu überwachende System einen sogenannten Agenten, ein Shellscript, normalerweise unter `/etc/hotplug/`*Systemname*`.agent`. Für USB wäre das also das Script `/etc/hotplug/usb.agent`.

Informationen über solche Ereignisse werden den Agenten vom Kernel über Umgebungsvariablen übermittelt. Das Programm usbmodules wird beispielsweise vom usb-Agenten benutzt, um die erkannten Geräte mit den notwendigen Modulen auszustatten. Die Information über die Geräte erhält **hotplug** über die Umgebungsvariable `DEVICES`.

hotplug ist ein Daemon, der über ein init-Script (`/etc/init.d/hotplug`) gestartet wird.

Funktionsweise von usbmgr

usbmgr ist ein Daemon, der benötigte USB-Module etsprechend seiner Konfigurationsdateien läd und entläd. Zusätzlich können noch Scripts ausgeführt werden, sofern das notwendig sein sollte. **usbmgr** benutzt die offiziellen USB-Vendor-IDs und die USB-Device-IDs, die vom USB-Consortium offiziell vergeben werden, um herauszufinden, welche Geräte er gefunden hat. Die beiden Konfigurationsdateien des Programms sind:

- `/etc/usbmgr/usbmgr.conf`
  Eine Datei, die die offiziellen IDs enthält. Diese Datei kann aktuell über http://www.wondernetworkresources.com/staff/shuu/linux/usbmgr/ bezogen werden und muß nicht verändert werden. Sie dient nur der Information, welche Module für welche IDs geladen werden müssen.
- `/etc/usbmgr/preload.conf`
  Diese Datei enthält eine Liste der Module, die **usbmgr** laden soll, wenn er startet. Diese Datei kann vom Systemverwalter entsprechend verändert werden.
- `/etc/usbmgr/host`
  enthält den Modulnamen (ohne `.o`) des verwendeten USB-Hostcontrollers also entweder `usb-ohci` oder `usb-uhci`

Für eine korrekte Funktion von **usbmgr** müssen folgende Voraussetzungen erfüllt sein:

- Der Kernel muß USB-fähig sein (usbcore)
- Das USBDEVFS muß unterstützt werden
- Die benötigten Kernelmodule müssen existieren

Auch **usbmgr** ist ein Daemon, der über ein init-Script gestartet wird.