

---

## 2 USB und Linux hotplug

### In diesem Kapitel lernen Sie

- ▶ das USB-Schichtenmodell kennen.
- ▶ die Kernelmodule für USB-Treiber kennen.
- ▶ wie Sie USB-Geräte unter Linux verwenden.
- ▶ das **hotplug**-System von Linux kennen.

### 2.1 Eigenschaften von USB

*USB* steht für *Universal Serial Bus*. Dieser wurde entwickelt, um eine einheitliche Schnittstelle für Peripherie-Geräte mit folgenden Eigenschaften bereitzustellen:

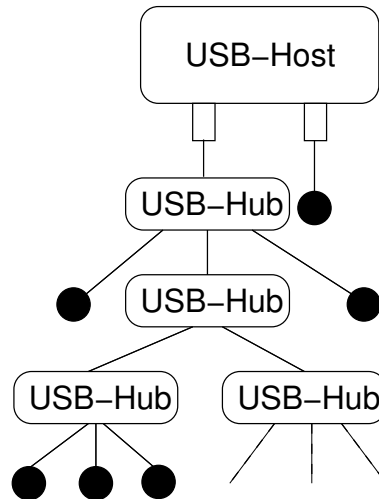
**Serielle Datenübertragung** Daten werden auf dem USB-Bus seriell übertragen.

**Baumhierarchie** Ein Host steuert die gesamte Kommunikation. Er kann bis zu 127 angeschlossene Geräte steuern. Diese sind in einer baumartigen Hierarchie mit dem Host-Controller verbunden. Dabei kann an einem USB-Anschluss entweder ein Gerät angeschlossen sein, oder ein USB-Hub, der weitere Anschlüsse (bis zu den maximal möglichen 127) zur Verfügung stellt.

**Hot-Plug-fähig** USB-Geräte sollen im laufenden Betrieb des Rechners angeschlossen oder entfernt werden können. Dabei soll das zuständige Betriebssystem in der Lage sein, ein Gerät beim Anschluss automatisch zu erkennen, zu konfigurieren, und in Betrieb zu nehmen.

**Stromversorgung** Über den USB-Anschluss wird eine Stromversorgung mit einer Spannung von 5 V und einer Stromstärke von 5 mA (gemäß Spezifikation) zur Verfügung gestellt. Geräte, die nicht besonders leistungshungrig sind, benötigen so keine zusätzliche Stromquelle.

Die USB-Baumhierarchie:



USB steht bisher mit folgenden Bandbreiten zur Verfügung:

- 1,5 MBit/s (USB 1.3)
- 12 MBit/s (USB 1.3)
- 480 MBit/s (USB 2.0)

Ein USB-Anschluss muss vier Verbindungen bereitstellen, je zwei für die Stromversorgung und zwei für die Kommunikation. Die Stecker für den Anschluss auf der Host-Seite sehen wie folgt aus:

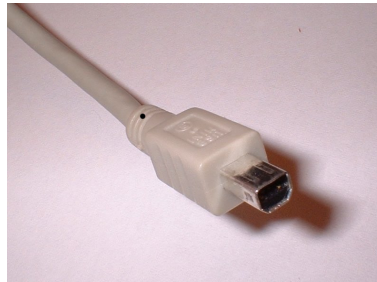


Auf der Geräteseite gibt es (neben einigen Sonderformaten) die folgenden beiden Steckerformen:

USB-Geräteanschluss:



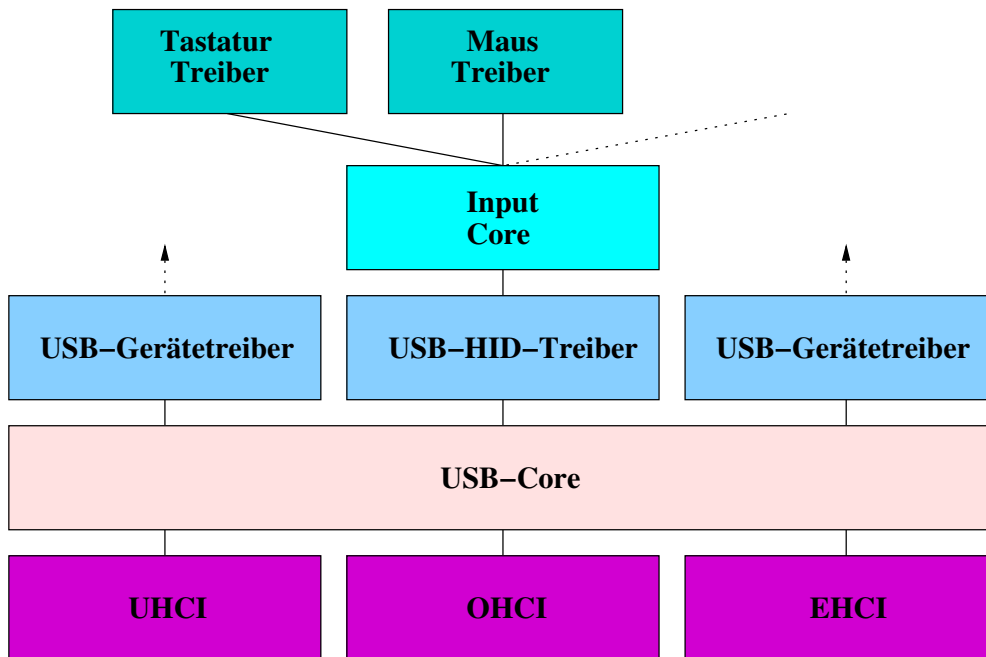
USB-Geräteanschluss klein:



Dabei ist die Kabellänge auf 5 m begrenzt, mit USB-Hubs ist eine Gesamtlänge von bis zu 30 m möglich.

## 2.2 Das Schichtenmodell der USB-Architektur

Die Treiberhierarchie von USB beruht auf einem Schichtenmodell:



**Die Hostcontroller** Es gibt derzeit folgende Arten von Hostcontrollern:

**UHCI** Universal Host Controller Interface, USB 1.3 (auf Mainboards von Intel, VIA)

**OHCI** Open Host Controller Interface, USB 1.3 (auf Mainboards von ALI, Compaq, NEC, Opti, SiS)

**EHCI** Enhanced Host Controller Interface, USB 2.0

## 2.3 Die USB Kernelmodule laden

*Den verwendeten Hostcontroller ermitteln:*

1. Mit `lspci`:



```
$ lspci | grep -i usb
00:1d.0 USB Controller: ... (ICH6 Family) USB UHCI #1 (rev 03)
00:1d.1 USB Controller: ... (ICH6 Family) USB UHCI #2 (rev 03)
00:1d.2 USB Controller: ... (ICH6 Family) USB UHCI #3 (rev 03)
00:1d.3 USB Controller: ... (ICH6 Family) USB UHCI #4 (rev 03)
00:1d.7 USB Controller: ... mily) USB2 EHCI Controller (rev 03)
```

Auf diesem System ist also UHCI für USB 1.3 und EHCI für USB 2 vorhanden.

Kennt man die verwendeten Hostcontroller, kann man die entsprechenden Treibermodule laden. Seit Kernel 2.6 heißen diese: **uhci-hcd.o** (für UHCI), **ohci-hcd.o** (für OHCI) und **ehci-hcd.o** (für EHCD). Auf älteren Systemen mit Kernel 2.4 gibt es für UHCI das Modul **usb-uhci.o** und für OHCI **usb-ohci.o**.

Zuerst muss aber das USB-Core Modul **usbcore.o** geladen werden.

Entweder lädt man die Module von Hand oder man lädt sie automatisch beim Systemstart, z.B. mittels `/etc/modules.conf`.

Das USB-Core Modul enthält dabei die USB-Grundfunktionalität. Die Ebenen darunter bauen auf dem USB-Core Modul auf und implementieren zusätzlich alle Funktionen die Für den Betrieb der Hostcontroller nötig sind.

Die Ebenen darüber bauen ebenfalls auf das USB-Core Modul auf. Die Gerätetreiber können als die Core-Funktionen verwenden. Es lassen sich nach oben auch Geräteklassen zusammenfassen, z.B. die *HID (Human Interface Device)* Geräte. Dadurch können abstrakte Treiber geschrieben werden, die alle für HID-Geräte typischen Funktionen implementieren. Für die eigentlichen Gerätetreiber ist dann nicht mehr soviel Programmierarbeit nötig.



Welche USB-Geräte am Hostcontroller angeschlossen sind, kann man mit dem Programm **lsusb** betrachten (LPI 2: 213.2). Die Option `-v` erhöht die Anzahl von angezeigten Informationen, die Option `-t` zeigt die Baumstruktur der angeschlossenen Geräte an. Ein grafisches Programm dafür ist **usbview**.

Der Zugriff auf Geräte und ihre Informationen erfolgt dabei über ein dynamisch erzeugtes Dateisystem, das **usbdevfs**. Es wird typischerweise unter `/proc/bus/usb` eingehängt. Das kann z.B. automatisch über einen Eintrag in `/etc/fstab` erfolgen:

```
usbfs      /proc/bus/usb      usbfs      noauto      0 0
```

## 2.4 Hot-Plug

Damit Linux bei Anschluss eines Hot-Plug fähigem Gerätes den passenden Treiber automatisch laden kann, gibt es das Programm **hotplug**. Es kann nicht nur USB-, sondern z.B. auch Firewire- und PCMCIA-Geräte verwalten.

## USB und Linux hotplug

---

Die Konfiguration findet sich im Verzeichnis `/etc/hotplug`.

Dabei wird jedes System wie z.B. USB oder PCMCIA von einem Agenten überwacht. Wird ein neues Gerät angeschlossen, wird anhand dessen ID versucht den richtigen Treiber zu ermitteln und zu laden. Dazu kann es im Fall von USB-Geräten z.B. das Programm **usbmodules** verwenden, das zu einem angeschlossenen Gerät alle verfügbaren Kernelmodule anzeigt.

Ein alternatives Programm zu Hotplug ist **usbmgr**. Es kann aber nur USB überwachen und wird in `/etc/usbmgr/usbmgr.conf` konfiguriert. Eine Liste beim Start von **usbmgr** zu ladender Module findet sich in `/etc/usbmgr/preload.conf`, der Modulname (ohne `.o` oder `.ko`) für den Hostcontroller steht in `/etc/usbmgr/host`.

Linux-Distributoren haben mitunter eigene Varianten zur Hotplug-Umsetzung.

## 2.5 udev

In neueren Linux-Distributionen (ab Kernel 2.6) findet ein neuartiges Geräteverwaltungssystem namens *udev* Einsatz (LPI 2: 203.4). Es ersetzt das ältere *devfs* und übernimmt auch Funktionalitäten, die bis dahin von Hotplug bereitgestellt wurden. **udev** läuft völlig im Userspace und ist damit einfacher wartbar und erweiterbar als alte Mechanismen. Ⓛ

**udev** wird ausschließlich über die Dateien in `/etc/udev` konfiguriert. Wichtig sind die im Verzeichnis `/etc/udev/rules.d/` abgelegten *Regeldateien*, die bestimmen, wie das **udev**-Programm auf vom Kernel generierte Ereignisse reagiert.

Ein Ausschnitt aus der mitgelieferten Konfigurationsdatei sieht folgendermaßen aus:

### Ausschnitt aus `/etc/udev/rules.d/50-udev.rules`

---

```
# floppy devices
KERNEL=="fd[0-9]*",      NAME="floppy/%n", SYMLINK+="%k", GROUP="floppy"

...

# usb devices
KERNEL=="hiddev*",NAME="usb/%k"
KERNEL=="auer*",NAME="usb/%k"
KERNEL=="legousbtower*",NAME="usb/%k", GROUP="usb"
KERNEL=="dabusb*",NAME="usb/%k"
BUS=="usb", KERNEL=="lp[0-9]*",NAME="usb/%k", GROUP="lp"
```

---

Hier wird z.B. festgelegt, dass für Diskettenlaufwerke Gerätedateien mit den Namen `/dev/floppy/<nummer>` und `/dev/fd<nummer>` mit der Gruppe *floppy* angelegt wer-

---

den. Für USB-Geräte gibt es ebenfalls verschiedene Anweisungen. Die letzte Zeile besagt z.B., dass für USB-Drucker eine Gerätedatei unter `/dev/usb` angelegt wird, die der Gruppe `lp` gehören soll.

Die genauen Bedeutungen der Direktiven sind in der Man-Page zu **udev** aufgelistet.



Mit dem Kommando **udevmonitor** (LPI 2: 203.4) können Sie die Events, die der Kernel und **udev** generieren, auf der Konsole mitschneiden. Diese Ausgaben dienen gewöhnlicherweise als Basis für eigene Regeln oder zu Diagnosezwecken.

## 2.6 Weiterführende Literatur

- <http://www.linux-usb.org>

### 2.7 Wissensfragen

1. Geben Sie vier Eigenschaften von USB an.

---

---

2. Welche Arten von USB-Hostcontrollern gibt es?

---

3. Nennen Sie zwei Hot-Plug Programme unter Linux.

---

4. Mit welchem Programm können Sie angeschlossene USB-Geräte betrachten?

---

5. Welches Programm können Sie benutzen um herauszufinden, welchen Hostcontroller ein System verwendet?

---

## 2.8 Lösungen

1. Hotplug-fähig, Baumhierarchie, serielle Datenübertragung, Stromversorgung.
2. UHCI, OHCI, EHCI
3. z.B. **hotplug**, **usbmgr**
4. z.B. **lsusb** oder **usbview**
5. z.B. **lspci**

