KNOPPER.NET

Dipl.-Ing. Klaus Knopper

Linux-Administration

(LINUX4ADMINS, 03/2006)

Eine Kurzeinführung von

Klaus Knopper (Knopper.Net)

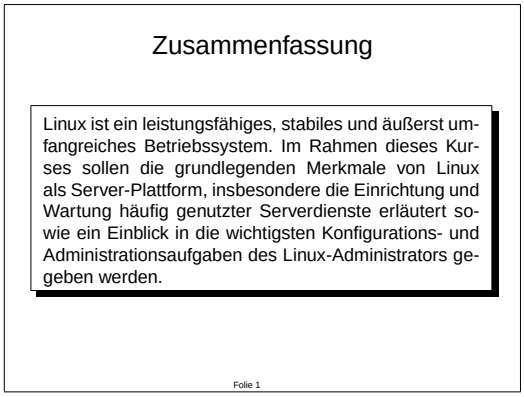
 Erstellt für: FH Kaiserslautern/Zweibrücken

Inhaltsverzeichnis

1 Einführung

In der Einführung lernen Sie zunächst Linux in seinem historischen, politischen und technischen Umfeld kennen.

1.1 Zusammenfassung



Notizen:

1.2 Ein wenig Geschichte

um 1970: In den Bell Labs wird von Ken Thompson ein Dokumentenverwaltungssystem (MULTICS) auf einer PDP-7 für kleine, modulare Aufgaben umgeschrieben, ☞ Unix

Ende der 70er: Diverse innovative Universitäten benutzen und erweitern das System. An der University of California in Berkeley (UCB) entsteht die Berkeley System Distribution (BSD). Sie enthält u. a. Ansätze zur Netzwerkfähigkeit und zum virtuellen Speicher.

70er-80er: Zwischen den beiden Hauptderivaten System V Unix (aus dem AT&T-Zweig hervorgegangen) und der Berkeley-Version BSD zieht sich der Streit über die Vorherrschaft während der gesamten 80er und Anfang der 90er Jahre hin.

1984 Richard Stallman gründet die Free Software Foundation, eine Gesellschaft, die freie Software (mit offenen Quelltexten) fördert, und mit Hilfe einer speziellen Lizenz, der GNU General Public License, die Offenheit und freie Verteilbarkeit der Software garantiert. Langfristiges Ziel ist es, ein Betriebssystem und eine Suite von Anwendungen zur Verfügung zu stellen, die vollständig frei sind von proprietärem oder nutzungslizenzpflichtigem Material.

1988: POSIX 1003.1 wird verabschiedet, ein Standard, der die Mindestanforderungen beider Lager vereint. Fast alle modernen Unices sind POSIX-compliant.

1993- Der finnische Student Linus Torvalds schreibt eine virtuelle Speicherverwaltung für i386-basierte Rechner. Er entscheidet sich dafür, den Quelltext seiner Arbeit zu veröffentlichen, was eine zuvor selten gekannte Kooperation zwischen Entwicklern über das Internet weltweit auslöst, und schafft damit die Grundlage für das heute populärste freie Unix-artige Betriebssystem für kostengünstige Desktop-PCs und andere. Auf Anwendungsebene wird die bereits für andere Unix-Systeme vorhandene GNU-Software portiert und verwendet, so dass nach kurzer Entwicklungszeit ein vollständiges Set an Anwendersoftware inclusive Entwicklungsumgebungen zur Verfügung steht.

1995- Das Linux-Betriebssystem und auf Unix-Betriebssystemen basierende Anwendungen wie SAMBA und Apache verbreiten sich vor allem als preisgünstige Server-Systeme, zunächst nur als Geheimtipp unter Technikern, später im regulären Ersatz als File, Print- und Informationsserver sowie Gateways in heterogenen Netzwerkumgebungen.

1998-heute Neben dem zunehmenden Einsatz als Desktop/Client-System mit KDE oder GNOME hält Linux auch Einzug als embedded Betriebssystem in Handhelds, Kameras, MP3-Spielern und anderen Geräten der Unterhaltungs- und Kommunikationsindustrie.

Ein wenig Geschichte

1970- Erste Unix-Betriebssysteme auf Großrechnern, vor allem an Universitäten.

1988- POSIX 1003.1 als Standard verabschiedet.

1993- Linus Torvalds gibt den Quelltext für Kernel 0.1 frei. Das System wird zusammen mit der GNU-Software der Free Software Foundation von vielen Entwicklern weltweit über das Internet weiterentwickelt.

heute- "Linux World Domination"? (Im Einsatz als kleines und mittleres Server-System auf kostengünstigen PCs im Intranet/Internet-Bereich bereits erreicht.)

Folie 2

Notizen:

1.3 Die GNU General Public License

Die GNU General Public License

Gibt den Empfängern der Software das Recht, ohne Lizenzgebühren

- den Quelltext zur Software zu erhalten, um diese analysieren und nach eigenen Wünschen modifizieren zu können,
- die Software in beliebiger Anzahl zu kopieren,
- die Software zu modifizieren (s.o.),
- die Software im Original oder in einer modifizierten Version weiterzugeben oder zu verkaufen, auch kommerziell, wobei die Empfänger der Software diese ebenfalls unter den Konditionen der GPL erhalten.

`http://www.gnu.org/`

Folie 3

Notizen:

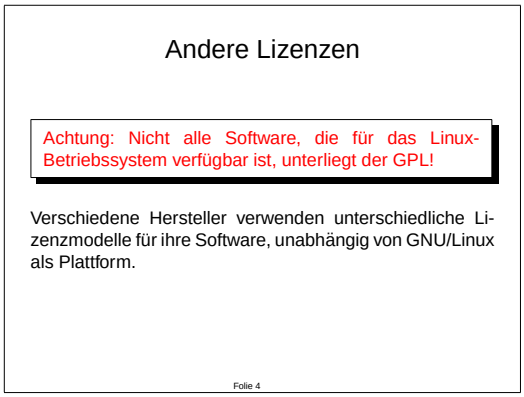
Lizenzmodelle kennen Sie im Umfeld proprietärer Software als mehr oder weniger genaue Festlegungen, die bestimmen, was Sie mit der Software tun dürfen und was nicht, wobei i.d.R. Gebühren mit der Benutzung der Software verbunden sind. Die GNU General Public License

gehört zu den sogenannten „Open Source“-Lizenzen (`http://www.opensource.org/`), die dem Empfänger der Software das Recht zum Kopieren, Modifizieren und zur Weitergabe/zum Verkauf der Software explizit zuspricht, ohne die Nutzung der Software dabei an einen Preis oder Zweck zu binden. Die einzige Einschränkung hierbei ist die nicht-aufhebbare Vererbung dieser Lizenz: Nach Modifikation und Weitergabe der Software erhalten die neuen Empfänger stets die gleichen Rechte, die der ursprüngliche Empfänger hatte. Dies bedeutet im Regelfall, dass der neue Empfänger/Käufer der Software Zugriff auf alle Quelltexte der Software erhalten muss, um Modifikationen durchführen zu können (sofern er dies möchte). Das Urheberrecht bleibt hiervon unberührt.

1.4 Andere Lizenzen

GNU/Linux als Plattform (d.h. das Linux-Betriebssystem und die Suite von Anwender- und Entwicklungsprogrammen der Free Software Foundation) unterliegt zwar der GPL, individuelle Anwenderprogramme und Softwarepakete können jedoch herstellerspezifisch anderen Lizenzen unterliegen.

Die Mischung aus GPL und proprietären Lizenzen bei Rechner-Installationen auf dem gleichen Medium ist durchaus üblich, und mit der GPL verträglich, solange keine GPL-Komponenten in proprietäre Binärformate fest integriert werden (und umgekehrt).



Notizen:

1.5 Eigenschaften von Unix

Unix ist grundsätzlich nicht „einfacher“ oder „schwerer“ in der Anwendung und Administration als andere Betriebssysteme. Es gibt jedoch Unterschiede in der Funktions- und „Denkweise“ sowie in Aufbau und Komplexität.

Eigenschaften von Unix

- Mehrere Aufgaben gleichzeitig (Multitasking)
- Mehrbenutzerfähig (Multiuser)
- Auf vielen Hardware-Plattformen lauffähig (portabel)
- Effiziente Ausnutzung der Resourcen (nicht proprietär)
- hierarchisches Dateisystem
- Stabilität durch eigenen Speicherbereich für jedes Programm (Virtual Memory, Speicherschutz)
- strikte Trennung zwischen Betriebssystem („Kernel“) und Anwendersoftware (Desktop-, Server-Suiten)

Folie 5

Notizen:

1.6 Verschiedene Unix-Betriebssysteme

Neben dem erst im letzten Jahrzehnt sehr populär gewordenen GNU/Linux gibt es eine Reihe weiterer Unix-Derivate, die seit langer Zeit vor allem im Server- und Mainframe-Bereich etabliert sind.

Durch die weitgehende Standardisierung von Programmiersprachen (hauptsächlich C, C++) und Schnittstellen, sog. APIs (definiert durch POSIX) besteht weitgehende Sourcecodekompatibilität.

Den Standards entsprechender Code kann durch vergleichsweise wenig Aufwand von einem Unix-Derivat auf ein anderes portiert werden (i.d.R. erneute Übersetzung aus dem Quelltext erforderlich).

Verschiedene Unix-Betriebssysteme

Name/TM	**Hersteller**
SunOS/Solaris	SUN Microsystems
HPUX	Hewlett Packard
Aix	IBM
Sinix	Siemens/Nixdorf
Ultrix/DEC Unix(OSF/1)	Digital Equipment
Linux	Community (Entwickler)
FreeBSD/NetBSD	Community (Entwickler)
...	

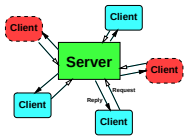
Folie 6

Notizen:

1.7 Client/Server-Prinzip

Im Umfeld von Unix hat sich ein allgemeines, leistungsfähiges Konzept zur Kommunikation entwickelt und durchgesetzt: Das Client/Server-Prinzip, das nicht nur die Netzwerkfähigkeit fast aller Anwendungen unter Unix garantiert, sondern auch die Interoperabilität von z.B. von Datenbank-Anwendungen. Client und Server können auf unterschiedlichen Plattformen realisiert werden.

Client/Server-Prinzip



- Client: Dienstanforderer (z.B. Browser)
- Server: Diensterbringer (z.B. WWW-Server)
- Abwicklungsprotokoll, das beide Partner verstehen, z.B. **`http`**.

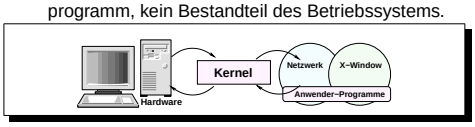
Folie 7

Notizen:

1.8 Linux-Internals

Linux-Internals

- Der **Kernel** ist die Schnittstelle zwischen Hardware und Anwendersoftware.
- Die Netzwerkfähigkeit ist im Kernel integriert, daher sind prinzipiell **alle** Programme netzwerkfähig.
- Graphische Benutzeroberfläche ist ein Anwenderprogramm, kein Bestandteil des Betriebssystems.



Folie 8

Notizen:

Der **Kernel** (=„Betriebssystem-Kern“) enthält u.a. das, was Sie in anderen Betriebssystemen als „Treiber“ für Geräte kennen und steuert die Vorgänge des Betriebes, insbesondere Zugriff auf die Hardware (Speichermedien, RAM). Das Internet in der heutigen Form geht auf die Netzwerkfähigkeit von Unix zurück (erste Netzwerktechnologie, die sich für mehrere Hardware-Plattformen durchgesetzt hat). Durch die Netzwerkfähigkeit des Systemkerns sind hier sämtliche Programme von vornherein „netzwerkfähig“, können also sowohl lokal als auch über ein Netzwerk bedient werden. Dies ist ein enormer Vorteil für die Administration des Systems. Die graphische, netzwerkfähige Benutzeroberfläche, das **X-Window System** mit seinen wählbaren Desktop-Aufsätzen (z.B. KDE, GNOME) ist nicht Bestandteil des Betriebssystems, sondern ein Anwenderprogramm, und daher austauschbar (wichtig vor allem für die Benutzerführung bei embedded Systemen).

Für Administrator und Benutzer stehen **Programme** und **Kommandos** für eine Vielzahl von Einsatzzwecken und angeschlossene Peripherie zur Verfügung.

1.9 Die Shell

Der Befehlsinterpreter, Shell genannt, ist eins der zentralsten Programme in jedem Unix-System, da jeder Benutzer ihn dazu gebraucht, um mit der Tastatur Befehle einzugeben. Wer die Shell gewohnt ist, kann hiermit um Größenordnungen schneller arbeiten, als dies mit graphischen Utilities möglich ist, da Aufgaben automatisiert und mehrfach ausgeführt werden können. Beispielsweise können Sie in der Shell als Administrator mit einer einzigen, kurzen Kommandozeile 5000 Benutzer anlegen. Mit einer graphischen Administrationsoberfläche (die es unter Unix natürlich auch gibt und die Sie im Laufe des Kurses noch näher kennenlernen werden) würden Sie hierfür den ganzen Tag brauchen.

Es gibt unter Unix mehrere Shells, die mit unterschiedlicher Syntax interner Kommandos arbeiten. Zum interaktiven Betrieb werden häufig die `bash`, `(t)csh`, oder `ksh` verwendet.

Die Shell ist damit das Gegenstück zu `COMMAND.COM` unter DOS, aber wesentlich leistungsfähiger, da beispielsweise Jokerzeichen (`*`) in Dateinamen schon durch die Shell und nicht erst durch das Anwendungsprogramm aufgelöst werden.

Mit der Shell ist es auch möglich, komplexe Programme zu schreiben, sogenannte „Shellskripte“.

Unter Unix gibt es eine sehr große Anzahl von Kommandos für eine Vielzahl von Aufgaben. Die Aufrufkonvention von Befehlen ist jedoch immer gleich:

1. Kommandoname
2. Optionen
3. Argumente

Zur Kombination von Programmen verwendet man **Ein- und Ausgabeumlenkung** bzw. die sogenannten **Pipes**.

Den Programmen, die zum GNU/Linux-System (und Unix im allgemeinen) gehören, liegt eine allgemeine „Philosophie“ zugrunde:

Kommandos in der Shell

- Viele, kleine Programme für jeweils nur eine Aufgabe,
- extrem kurze, „selbsterklärende“ Kommandonamen,
- leichte Kombinationsmöglichkeit dieser kleinen Programme.

Folie 9

Notizen:

1.10 Unterschiede zu DOS/Windows

Unterschiede zu DOS/Windows

Einige augenfällige Unterschiede zu DOS:

- Groß-/Kleinschreibung wird beachtet.
- Foreslash „`/`“ statt Backslash. „`\`“ als Trenner bei Pfaden.
- Es gibt keine „Laufwerksbuchstaben“.
- Ausgabe-Umleitungen werden nicht über Hilfsdateien „simuliert“.
- Mehrere Programme können gleichzeitig im Hintergrund laufen („&“).

Folie 10

Notizen:

Kommando-Syntax

In der Shell eingegebene Kommandos haben im Allgemeinen das folgende Format:

```
Programmname Optionen Argumente Umleitung
```

Die Umlenkung von Ein- und Ausgabe funktioniert, anders als bei DOS, wo Temporärdateien geschrieben werden, auf direktem Weg auch zwischen Programmen.

> Datei	Umlenkung der Ausgabe in Datei
< Datei	Umlenkung der Eingabe von Datei
| Kommando	Umlenkung der Ausgabe in die Eingabe eines anderen Programms.

Folie 11

Notizen:

1.11 Navigieren im Dateisystem mit der Shell

NB: Es wird hier vorausgesetzt, dass Sie mit der Strukturierung eines hierarchischen Dateisystems vertraut sind, und wissen, was Dateien und Verzeichnisse/Unterverzeichnisse sind. Die nachfolgende Folie fasst die Kommandos, die Sie hierfür kennen sollten, zusammen.

Navigieren im Dateisystem mit der Shell

`pwd`	Ausgabe aktuelles Arbeitsverzeichnis
`cd Verzeichnisname`	Wechsel des aktuellen Verzeichnisses
`ls -l [wildcards]`	Ausführliches Auflisten von Dateien*)
`mkdir [-p] Verz.`	Lege [Mit Unterverz.] Verzeichnis an.
`cp [-a] Alt Neu`	Kopiere [Klone Alles] von Alt nach Neu
`mv Alt Neu`	Benenne Alt nach Neu um
`rm [-rf] wildcard`	Lösche unwiderruflich [rekursiv forciert] †)

*) Die angezeigten Dateirechte werden noch ausführlich besprochen.
†) Tipp: Überlegen Sie zweimal, bevor Sie mit der Option **-rf** etwas löschen, gerade, wenn Sie momentan Administratorstatus haben!

Folie 12

Notizen:

1.12 Linux Benutzer und Administrator

Nach erfolgreicher Anmeldung am System erhalten alle vom Benutzer gestarteten Programme seine **User-ID**. Der Benutzer hat nur Zugriff auf Dateien und Prozesse, die dieser User-ID zugeordnet sind, oder die **global** freigegeben sind. Dies ist zum Arbeiten auf einem Unix-System vollkommen ausreichend, wenn der Administrator die Zugriffsrechte für alle Systemkomponenten so verteilt hat, dass die Benutzer alle für ihre Arbeit notwendigen Rechte an Systemresourcen besitzen (z.B. Zugriff auf Geräte wie CD-Brenner, Diskettenlaufwerk usw.). Da ein gewöhnlicher Benutzer keinerlei Rechte an systemkritischen Dateien (wie Laufzeitbibliotheken, globalen Konfigurationsdateien etc.) hat, kann er im System auch kaum Schaden durch versehentliches Fehlverhalten oder gar Absicht anrichten. Daraus folgt auch die systembedingte Resistenz eines Unix-Systems gegen sog. „Computerviren“: Da ein Benutzer Systemdateien nicht verändern kann, können sich solche Programme gar nicht installieren, bzw. unbemerkt „festsetzen“.

Der Administrator eines Unix-Systems besitzt, unabhängig vom Namen, mit dem er sich am System anmeldet, die User-ID **0** und hat, unabhängig von irgendwelchen Einstellungen in Konfigurationsdateien, alle Rechte an allen Systemresourcen. Er ist normalerweise der einzige Benutzer, der auf einem Unix-System wirklich Schaden anrichten kann. Daher soll und darf die Administrator-Kennung ausschließlich zu Zwecken der systemweiten Installation von Programmen sowie Änderungen an der Systemkonfiguration verwendet werden.

Die Benutzer mit den für das System relevanten, numerischen User-IDs, sind in der Konfigurationsdatei **`/etc/passwd`** vermerkt, die, entgegen ihrem Namen, eigentlich gar keie Passwörter enthält. Diese wiederum sind in der – für normale Benutzer ungleich User-ID 0 – nicht lesbar in der Datei **`/etc/shadow`** untergebracht.

Passwörter werden nach einem Verfahren verschlüsselt gespeichert, das keine Rückabbildung hat (Checksummen-Verfahren, **md5** oder **crypt** mit unvollständiger Ausgabe). Daher können unbekannte Passwörter unter Unix allenfalls mit Brute-Force-Methoden „ausprobiert“, oder mit Hilfe von Wörterbüchern und Permutationen „geraten“ aber nicht aus einem bekannten verschlüsselten Passwort zurückgerechnet werden.

Der Administrator hat auf den meisten Unix-Systemen eine sehr restriktive Arbeitsumgebung: i.d.R. ist das Anmelden des Administrators mit graphischer Oberfläche nicht erlaubt, und bestimmte „gefährliche“ Komponenten des Kommando-Suchpfades (z.B. das aktuelle Verzeichnis) sind ausgeschlossen.

Linux Benutzer und Administrator

Merkregel: Benutzen Sie den Administratoraccount (User-ID 0) **ausschließlich** zur systemweiten Installation von Programmpaketen und für Konfigurationsarbeiten, **niemals** jedoch zum regulären Arbeiten!
Nur dann kann Unix/Linux Ihnen die vielgepriesene Sicherheit und Stabilität bieten.
Zum regulären Arbeiten in einer komfortablen Umgebung (z.B. graphische Benutzeroberfläche) ist ein normaler Benutzer-Account vollkommen ausreichend, und in vieler Hinsicht im Arbeitskomfort dem Administratoraccount sogar überlegen.

Folie 13

Notizen:

1.13 Benutzer anlegen und Passwort setzen

Neben GUIs (graphical User Interfaces) zur Benutzerverwaltung existiert auf jedem Unix-System auch ein textorientiertes Kommando für den Administrator, mit dem neue Benutzer im System eingerichtet werden können. Unter GNU/Linux heißt dies **useradd**, und unterstützt eine Reihe von Optionen, welche Sie mit

```
useradd --help
```

erfahren. Generell unterstützen fast alle Kommandos unter Linux die `--help`-Option, um eine Kurzhilfe auszugeben. Die ausführliche Online-Hilfe erhalten Sie mit **man Kommandoname**.

Nach dem Einrichten einer neuen Benutzerkennung (engl. „Account“) ist aus Sicherheitsgründen zunächst kein Login mit der neuen Kennung möglich. Hierzu muss der Administrator erst für diese neue Kennung ein Passwort setzen, und dies dem neuen Benutzer auf sicherem Weg mitteilen.

```
passwd Kennung
```

Benutzer dürfen nur ihr eigenes Passwort ändern und werden aus Sicherheitsgründen bei dieser Aktion stets zunächst nach dem alten Passwort gefragt.

Übung: Benutzer einrichten

Melden Sie sich auf der Textkonsole Ihres Rechners[1] als **root** an und richten Sie sich mit dem Kommando

```
useradd -c "Mein Name" -m benutzer
```

eine Benutzerkennung ein. Setzen Sie dem neuen Benutzer mit dem Kommando

```
passwd benutzer
```

ein Passwort.[2]

Folie 14

Notizen:

[1] Mit der Tastenkombination `Steuerung-Alt-F1` gelangen Sie auf die erste Textkonsole, falls Ihr Rechner Ihnen stattdessen ein graphisches Login präsentiert.
[2] Wenn das Passwort bei der unter Linux üblichen Sicherheitsüberprüfung als „zu einfach“ durchfällt, wird es dennoch akzeptiert, wenn Sie es als Administrator einrichten.

Tipp: Die Option **-m** sollten Sie beim Einrichten von Benutzern mit `useradd` stets angeben, auch wenn sie bei neueren Linux-Distributionen bereits Default ist. Diese Option sorgt dafür, dass der neu angelegte Benutzer einige nützliche, gebrauchsfertig vorkonfigurierte Einstellungsdateien aus dem Systemverzeichnis **/etc/skel** in sein Heimverzeichnis kopiert bekommt.

Übung: Anmelden als Benutzer

Wechseln Sie wieder zum graphischen Login (`Steuerung-Alt-F7`) oder, falls Ihr Rechner zunächst nur für den Textmodus vorbereitet wurde, geben Sie als **root** das Kommando **xdm** ein.
Melden Sie sich nun mit Ihrer neuen Benutzerkennung an. Starten Sie ein Terminal-Fenster (Menüleiste), denn wir wollen nun mit der Shell weiterarbeiten, ohne auf den Komfort der graphischen Oberfläche verzichten zu müssen.

Folie 15

Notizen:

1.14 Wechsel des Benutzerstatus

Wenn Sie wirklich eine Systemadministrationsaufgabe erledigen müssen, müssen Sie sich auf einem Unix-System **nicht** abmelden und neu als Administrator anmelden, wie Sie es vielleicht von anderen Systemen gewohnt sind. Vielmehr können Sie als Benutzer mit Hilfe von sogenannten SUID-Programmen („Set User ID“) temporär und nur für diesen einen Prozeß Administratorstatus erlangen. Im Falle von **su** zum „Umschalten“ auf den Administratorstatus in der aktiven Shell ist hierzu (eigentlich selbstverständlich) die Eingabe eines Passwortes erforderlich.

Ausnahme: Das Kommando **sudo** erlaubt, konfigurierbar durch die Konfigurationsdatei **/etc/sudoers**, bestimmten Benutzern bestimmte Kommandos mit einer anderen User-ID (z.B. **0**) aufzurufen, mit oder ohne spezielle Authentifizierung (vergl. `/etc/sudoers` unter Knoppix).

Wechsel des Benutzerstatus

Mit dem Kommando **su** wechseln Sie in der aktiven Shell zum Status des Systemadministrators. Hierbei werden Sie nach dem Passwort des Administrators gefragt, das Sie (unsichtbar) eingeben müssen.
Bei Erfolg verändert sich Ihr Eingabeprompt, und Sie haben in dieser Shell alle Rechte des Administratoraccounts (welcher auf den meisten Systemen die Benutzerkennung **root** hat).

Vorsicht: ab diesem Zeitpunkt können falsch eingegebene Kommandos, die als normaler Benutzer harmlos sind, in dieser Shell zerstörerische Wirkung auf Ihr System haben!

Folie 16

Notizen:

Mit dem Kommando **`id`** können Sie Ihren momentanen Status, Kennung und Gruppenzugehörigkeit feststellen.

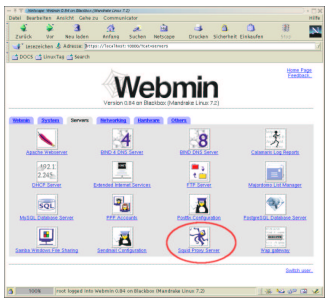
Mit Benutzer- und Gruppenrechten auf Dateisystemebene werden wir uns noch ausführlich im Abschnitt 7 „Dateirechte“ beschäftigen.

1.15 Graphische (Remote-)Administration mit Webmin

Für Linux (und andere Unix-Derivate) gibt es eine ganze Zahl graphischer Konfigurationsfrontends, einerseits, um den Bedienungskomfort zu erhöhen, und andererseits, um eine schnelle Hilfe zur Hand zu haben, um auch mit noch ungewohnten Diensten und Konfigurationsmöglichkeiten einen ersten Einstieg zu finden.

Ein für die Fernwartung gut geeignetes Tool ist der komplett in PERL geschriebene **Webmin**. Dieser stellt in Form eines vollkommen autarken, SSL-fähigen Webservers eine Möglichkeit dar, auch über das Internet Systemadministration betreiben zu können, bis hin zu der Variante, sich per Java-Applet auf Kommandozeilenebene einzuloggen.

Remote-Administration mit **Webmin**



Folie 17

Notizen:

Der professionelle Administrator muss dennoch die „manuelle“ Variante der zugrundeliegenden Konfigurationsaufgaben kennen, um auch dann eingreifen zu können, wenn das GUI seiner Wahl einmal nicht zur Verfügung steht. Auch die internen Zusammenhänge zwischen Diensten unter Unix müssen ihm klar sein, um Fehler schnell erkennen und beheben zu können.

Übung: Benutzerattribute mit Webmin

Verbinden Sie sich zum auf Ihrem Rechner laufenden Webmin. Starten Sie hierzu

```
netscape https://localhost:10000/
```

Auch hier müssen Sie sich zunächst als **root** anmelden. Finden Sie den Konfigurationspunkt „Benutzer und Gruppen“ und lassen Sie sich die Einstellungen für Ihre neu eingerichtete Kennung anzeigen.
Beenden Sie netscape wieder.

Folie 18

Notizen:

1.16 vi – Seitenorientierter Texteditor

Der `vi` ist ein unscheinbarer, aber dennoch sehr mächtiger Fullscreen-Editor, der (im Gegensatz zum ebenfalls sehr beliebten Emacs) zum Standard-Equipment jedes Unix-Systems gehört. Er ist eines der Hauptwerkzeuge des Systemadministrators zur Bearbeitung von Konfigurationsdateien.

Der vi kennt zwei Modi: Den **Kommando-Modus**, in dem die Bearbeitung des Textes mit Hilfe von Kommandos und Makros geschieht, und den **Direkteingabe-Modus**, in dem die Tastatureingaben direkt in den Text übernommen werden.

Im Kommandomodus sind komplexe Operationen (interaktives Suchen und Ersetzen, automatisches Formatieren von Textstellen, Record und Replay, Speichern, Laden, Anfügen, ...) mit wenigen Eingaben möglich, während im Direkteingabemodus der getippte Text unverändert von der Tastatur übernommen wird.

Direkt nach dem Start befindet sich der `vi` normalerweise im Kommandomodus!

vi – Seitenorientierter Texteditor

- Kompakter, schneller Texteditor,
- gehört zum Standard-Equipment auf jedem Unix-System,
- kennt *Kommandomodus* (Befehlseingabe) und *Insert*-Modus (Texteingabe),

Vorsicht: Direkt nach dem Start befindet sich der vi im *Kommandomodus*, d.h. jede Tastatureingabe wird als **Kommando** interpretiert, nicht als Eingabetext!

Folie 19

Notizen:

Tabelle der wichtigsten vi-Kommandos im Kommandomodus

<**Escape**>	Rückkehr vom Insert- in den Kommandomodus
i	Wechsel in den **I**nsert-Modus (Direkteingabe)
o	**O**pen: Neue Zeile anfügen ☞ Insert-Modus
dd	**D**elete: Aktuelle Zeile löschen
p	**P**aste: [Gelöschten] Text einfügen
x	Zeichen, auf dem der Cursor steht, löschen
:r Datei	**R**ead: Datei ab Cursor einfügen
:w	**W**rite: Datei speichern
:q	**Q**uit: `vi` beenden
:wq	**W**rite and **Q**uit: Speichern und Beenden
:q!	Beenden ohne Speichern
:%s/alt/neu/gc	Interaktiv `alt` durch `neu` ersetzen

Fast alle Kommandos lassen sich gruppieren (aneinanderreihen) oder mit einer vorangestellten Zahl mehrfach ausführen.

Übung

1. Legen Sie in Ihrem Benutzer-Heimverzeichnis eine neue Datei *uebung1.txt* mit **vi** an und geben Sie einen (beliebigen) Text ein. Speichern Sie die Datei und verlassen Sie vi.
2. Legen Sie mit **`vi`** in Ihrem Benutzer-Heimverzeichnis eine neue Datei *uebung2.txt* an. Laden Sie hier die zuvor erstellte Datei, fügen Sie die Datei */etc/passwd* am Ende an und ersetzen Sie alle „:“ durch Leerzeichen. Speichern Sie das Ergebnis unter dem Dateinamen *uebung2b.txt* ab.
3. Verlassen Sie vi und listen Sie mit **ls** Ihr Heimverzeichnis. Fällt Ihnen etwas auf?

Folie 20

Notizen:

2 Bootvorgang, Zustandsmanagement mit `init`

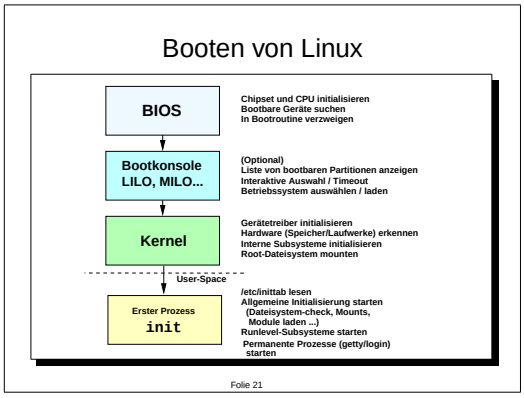
In diesem Kapitel wird der Systemstart sowie der Shutdown („Herunterfahren“) behandelt, also die Vorgänge vom Einschalten des Rechners bis zum arbeitsfähigen System, sowie das Beenden der Systemsoftware bis zum Ausschalten des Rechners.

2.1 Booten von Linux

Das Booten von Linux ist, bis zum Start von **`init`**, kaum unterschiedlich zu anderen Betriebssystemen. Zunächst müssen hardwareseitige Initialisierungsroutinen, die vom BIOS des Rechners gesteuert werden, durchlaufen werden. Anschließend wird der Systembus nach bootbaren Komponenten durchsucht. Schließlich wird, sofern vorhanden, ein *Bootlader* gestartet, dessen einzige Aufgabe es ist, Kernel-Komponenten zu laden und zu aktivieren.

Nachdem der Linux-Kernel die Kontrolle über das System übernommen hat, muss das Linux-System sich selbst in einen benutzbaren Zustand bringen. Oft wird dies durch Module und Programme auf einer **`initial ramdisk`** gelöst, teilweise durch Komponenten, die auf einem mehr oder weniger fest installierten Datenträger nach dessen Aktivierung zur Verfügung stehen.

Beispiel: Booten von KNOPPIX im **`debug`**-Modus.



Notizen:

2.2 Aufgaben von `init`

Das Programm **`init`** hat eine zentrale Bedeutung für den Betrieb eines Unix-Systems. Hauptsächlich kommt unter Linux das an System V angelehnte Paket zu Einsatz. **`init`** ist das erste Programm, welches nach dem Booten des Kernels die Aufgabe des Zustandsmanagements übernimmt, wobei der System-Zustand als sogenannter „Runlevel“ modelliert wird. **`init`** bestimmt über seine Konfigurationsdatei **`/etc/inittab`**, welche Programme und Dämonen in welchem Systemzustand aktiv sind.

Beispiele hierfür sind der *administrative Zustand*, in dem ausschließlich der Systemadministrator arbeitet, oder der *Mehrbenutzer-Zustand*, der als der „Normalzustand“ eines Unix-Systems gilt.

Aufgaben von **init**

- System-Initialisierungen (`rc.sysinit` bzw. Skripte in `rcS.d`),
- dauerhaft zu kontrollierende Prozesse (Logins, USV-Kontrolle),
- Subsysteme in bestimmten Systemzuständen (Netzwerk, Serverdienste, Remote-Dateisystem, ...)

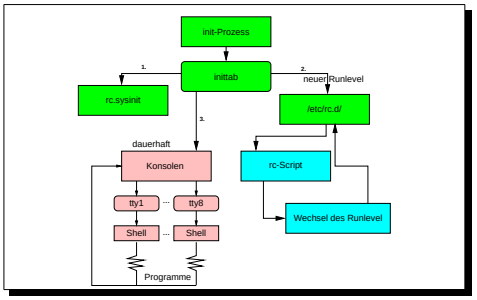
Initialiserungen und *dauerhafte Einstellungen* ändern sich selten, *Subsysteme* werden jedoch häufiger, auch im laufenden Betrieb, umkonfiguriert ☞ Runlevel.

Folie 22

Notizen:

2.3 Ablauf von `init`

Ablauf von **init**



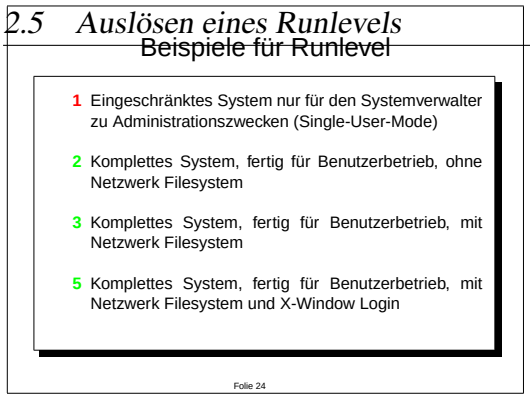
Folie 23

Notizen:

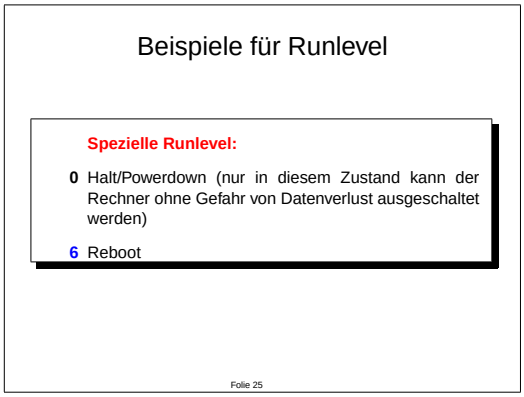
2.4 Was sind Runlevel?

Runlevel

- Modellieren den „*Gesamtzustand*“ des Systems
- *Gruppieren* die zur Verfügung stehenden Services und Subsysteme
- *Aktivieren* dynamische Systemkomponenten, z. B. Netzwerkdämonen oder NFS
- werden durch eine *Ziffer* (oder einige Buchstaben) benannt.



Notizen:



Notizen:

2.5 Auslösen eines Runlevels

- Auslösen des Default-Runlevels (aus `/etc/inittab`)
- Übergabe als Parameter beim Booten durch die Boot-Konsole (LILO, MILO, SILO, ...)

```
lilo boot: linux 1
```

- Expliziter Aufruf durch Systemverwalter bei laufendem Betrieb:

```
# telinit 4
```

- Shutdown durch Befehl oder Hotkeys (Ctrl-Alt-Del)
- Sicheres Herunterfahren durch `powerd` wegen Stromausfall etc.

2.6 Vorgänge beim Wechsel

- `rc` wird aufgerufen (Parameter: neuer (und alter) Runlevel).
- Im neuen Runlevel-Verzeichnis werden *Stoppskripte* aufgerufen.
- Im neuen Runlevel-Verzeichnis werden *Startskripte* aufgerufen.
- Skripte liegen in `/etc/rcN.d/...`
- Die Skripte sind symbolische Links nach `/etc/init.d/...`, wo **alle** Skripte für **alle** Runlevel gesammelt werden.

- Jedes Skript startet bzw. stoppt genau *ein* Subsystem, je nachdem, ob es mit Paramter `start` oder `stop` aufgerufen wird.

Runlevel-Skripte

Installation: Beim Installieren eines Programmpaketes, das einen Dienst zur Verfügung stellt, wird häufig ein Init-Skript automatisch nach **`/etc/init.d/dienstname`** installiert.
Mit distributionsabhängigen Tools werden dann, automatisch oder manuell, die entsprechenden Start- und Stopp-Links in die Runlevel-Verzeichnisse eingetragen (was sich natürlich genausogut manuell erledigen lässt).

Folie 26

Notizen:

Runlevel-Skripte

- RPM-basierte Systeme: **`chkconfig`**
- DEB-basierte Systeme: **`update-rc.d`**

Folie 27

Notizen:

Übung

- Sehen Sie in **`/etc/inittab`** nach, in welchem Runevel Ihr System normalerweise startet. Falls nicht per Default, eingetragen, ändern Sie den Default-Runlevel zu **`5`** (Multiuserbetrieb incl. graphischem Login) und teilen Sie **`init`** die Änderung mit, indem Sie ein **`HUP`**-Signal an den Prozess mit der Prozess-ID **`1`** schicken (**`init`** hat immer die erste Prozess-ID im System).
- Sorgen Sie dafür, dass in allen Runlevels automatisch der Apache-Webserver gestartet wird. Beobachten Sie, was sich durch **`update-rc.d`** im Verzeichnis **`/etc/rc5.d`** und **`/etc/rc0.d`** geändert hat!

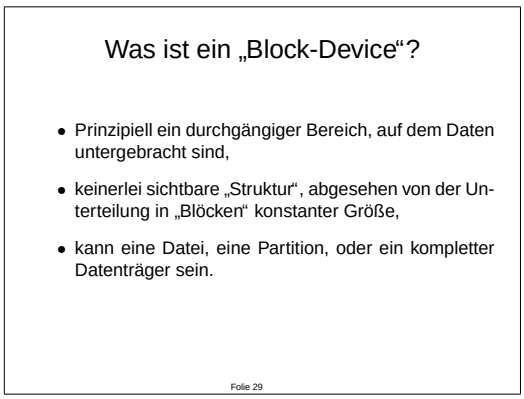
Folie 28

Notizen:

3 Allgemeines

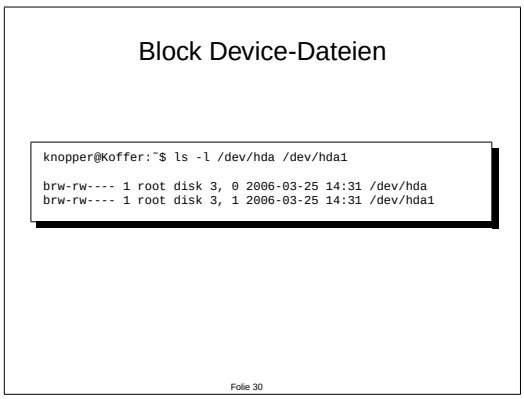
Dateisysteme unter Unix sind traditionell eines der komplexesten Gebiete, mit denen sich der Administrator beschäftigen muss, und deren Eigenschaften und Besonderheiten ihm vertraut sein sollten.

Speziell unter Linux existieren viele auch in anderen Betriebssystemen übliche Dateisysteme, teils mit Herstellerunterstützung implementiert, teils aufgrund restriktiver Handhabung von Quelltexten und Dokumentation durch den Hersteller reverse engineered, sofern möglich.



Notizen:

Ein Blockdevice ist eine sehr einfache technische Modellierung von tatsächlichen Datenträgern bzw. Datenbereichen. Der Kernel verwaltet keine Informationen über Struktur oder Art der Daten, die auf einem Blockdevice gespeichert sind. Es wird jedoch ein dynamischer „Block-Cache“ verwendet, der einmal gelesene Daten im RAM des Rechners vorrätig hält, so dass diese beim erneuten Lesen nicht wieder physikalisch ausgelesen werden müssen. Daher erscheinen unter Linux auch „langsame“ Datenträger wie Disketten oder Flash-Memory beim mehrfachen Lesen, sowie beim gepufferten Schreiben, sehr schnell zu sein.

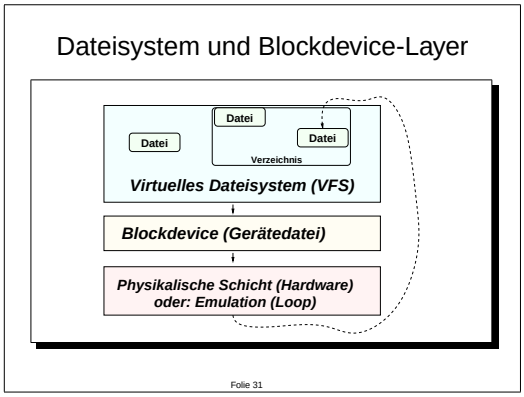


Notizen:

Der Kernel verwaltet bestimmte „Geräte“ in Form von Block Device Dateien, die die physikalischen Gegebenheiten abstrahieren. So steht das Block Device mit der *Major ID* 3 für die erste Festplatte am ersten IDE-Conroller, wobei die *Minor ID* festlegt, ob es sich um das komplette Gerät oder um eine Partition darauf handelt.

Dabei ist die Zuordnung zwischen Minor und Major ID für das jeweilige Gerätemodul zuständig, nicht etwa der (durchaus bei vielen Distributionen standardisierte) Dateiname der Gerätedatei in **`/dev`**.

Durch die Zugriffsrechte der Blockgerätedatei kann der Administrator festlegen, wer Zugang zu den „Rohdaten“ auf dem Gerät bekommt, z.B. um CDs auslesen oder brennen zu können.



Notizen:

Das *Virtuelle Dateisystem* bildet scheinbar „unstrukturierte“ Daten auf Block Devices in einer für Menschen „leichter handhabbaren“ Form ab, nämlich in Form von Dateien, Verzeichnissen und Unterverzeichnissen.

Die Art des gewählen Dateisystems bestimmt, in welcher Struktur die Daten im Block Device erwartet und abgelegt werden.

Dateisysteme unter Unix setzen immer auf einen – mehr oder weniger – physikalischen Layer auf, und sind nicht mit bestimmten Anwendungen oder Diensten unmittelbar verkoppelt. Auf Blockebene werden stets auf einem physikalischen (oder virtuellen) Datenträger Datenblöcke gelesen oder geschrieben. Ein Löschen oder Neuanlegen von Blöcken in dieser Schicht ist nicht vorgesehen. Das virtuelle Dateisystem (VFS) sorgt für die Präsentation der in der Blockebene gelagerten Daten in Form von Dateien und Verzeichnissen mit entsprechenden Attributen und Meta-Informationen. Sowohl Dateisysteme (die Präsentationsschicht) als auch Blockdevices (“Gerätetreiber“) werden als Modul oder fest in den Kernel eincompiliert vom Betriebssystem angesprochen, sobald ein Zugriff auf der logischen Ebene (z.B. “Datei öffnen“) in irgendeiner durch den Benutzer gestarteten Anwendung erfolgt.

Die Blockgerätedateien (z.B. **`/dev/hda1`**) und die zugehörigen Kernel-Module können mit hardwarespezifischen Modulen (IDE, SCSI, ...) oder ein Gerät “simulierenden“ Modulen verbunden sein. Beispiele für letzteren Fall sind **`loop.o`** (Loopback-Blockdevice, nicht zu verwechseln mit der Loopback-Netzwerkadresse des Rechners!) und **`cloop.o`** (Compressed Loopback). Diese Module stellen Dateien als Blockdevices bzw. virtuelle Festplatten-Partitionen dar, und sorgen dafür, dass beliebige Dateisysteme darin “verpackt“ und gemountet werden können. Auf Blockebene kann somit eine transparente Verschlüsselung (`loop.o` mit AES-Support) oder Kompression (`cloop.o`) stattfinden, unabhängig vom in der Präsentationsschicht gewählten Dateisystem.

4 Dateisysteme

Das (Client-)Program, das die Abbildung eines Block Device auf die „logische“ Ebene eines Dateisystems mit Verzeichnissen und Dateien etc. durch den Kernel initiiert, heißt **`mount`**.

`mount`

Syntax: **`mount -t dateisystemtyp \`**
`-o optionen,... \`
`blockdevice \`
`zielverzeichnis`

Aufgabe: Abbilden der „unstrukturierten“ Daten eines Block Device in eine Verzeichnisstruktur.
Hierbei sind die Optionen und die Wirkung hochgradig Dateisystemtyp-spezifisch!

Folie 32

Notizen:

Linux unterstützt eine große Anzahl verschiedener Dateisysteme...

Unterstützte Dateisysteme (1)

Native (blockdevice-basiert)	
ext2	Standard
ext3	ext2 mit Journal-File
reiserfs	Journaling, b-tree
jfs	Journaling, b-tree (IBM)
xfs	Journaling, b-tree (SGI)
iso9660	ISO/Rockridge/Joliet/zISO
minix	heute eher akademisches FS

Folie 33

Notizen:

Der Zugriff auf den Inhalt einer Datei (d.h. das Auffinden einer Inode aufgrund des Dateinamens/Pfades) lässt sich per hash oder (b-)tree-Verfahren beschleunigen. Bei ReiserFS ist der Dateizugriff bei komplexen Dateisystemstrukturen i.d.R. wesentlich schneller als bei ext2, da die Dateisystem-internen Suchpfade kürzer sind. Jedoch sind b-tree Suchbäume empfindlicher gegen Fehler (v.a. physikalische) in der Dateiorganisation.

Ein Journal ist ein Transaktions-Protokoll, in dem jede Änderung am Dateisystem zunächst angemeldet, und nach der Durchführung bestätigt wird. Beispiel: Erst wenn eine Änderung an einer Datei physikalisch geschrieben wurde, darf der Plattenplatz für die ursprüngliche Version der Datei freigegeben werden. Ansonsten bleibt die alte Version der Datei als gültig vermerkt, was beim “Replay“ des Journals nach einem Reboot dazu führt, dass ansonsten durch Stromausfall “verlorene“ Dateien wieder verfügbar sind.

Achtung: Stromausfall ist allgemein ein schlechtes Beispiel. Wenn ein physikalischer Schreibvorgang durch äußere Einwirkung manipuliert wird (z.B. Headcrash durch Erschütterung, Stromausfall während des Schreibens interner Datenstrukturen), kann es durchaus passieren, dass das Dateisystem auch durch das Transaktionslog nicht mehr zu retten ist. In diesem Fall müssen

manuelle Verfahren zur Datenrettung angewandt werden (z.B. fsck bei ext2/ext3), welche jedoch besonders bei b-tree orientierten Dateisystemen sehr schwierig sind, da ohne die verlorengegangenen Verwaltungsinformationen keine Zuordnung von Dateninhalten zu Dateien mehr möglich sind.

Unterstützte Dateisysteme (2)

RAM/Flash/Packet

jffs2	Journaling Flash Filesystem
romfs	ROM-FS
cramfs	komprimiertes ROM-FS
ramfs	experimentelles Ramdisk-FS
tmpfs	skalierendes Ram-FS
udf	DVD/CDRW-Packetmode FS (u.U. RW)

Folie 34

Notizen:

Unterstützte Dateisysteme (3)

Compatibility

adfs	Acorn (Archimedes) Disc Filing System
affs	Commodore Amiga Fast File System
hfs	"Hierarchical File System" (MAC)
bfs	"Boot File System" (SCO Unix)
efs	Altes IRIX (SGI) Dateisystem
vxfs	VERITAS VxFS (SCO UnixWare)
ntfs	NTFS/WinFS (vorwiegend ro)
hpfs	"High Performance FS" (OS/2)
qnx	QNX FS (vorwiegend RO)
sysv	XENIX/SCO/Coherent+PDP11
ufs	BSD/SunOS/NeXTstep FS
vfat	Altes und neues MSDOS Format

Folie 35

Notizen:

Achtung: Einige Dateisystemtypen unterstützen nicht die für Unix-Systeme essentiell notwendigen Dateiattribute (Permissions, Owner, Groups, Devices, Symlinks/Hardlinks). **umsdos** ist eine Aufsatz auf das msdos/vfat Dateisystem, das die Unix-Features in Form von in Dateien vermerkten Attributen "emuliert".

Unterstützte Dateisysteme (4)

Virtuelle

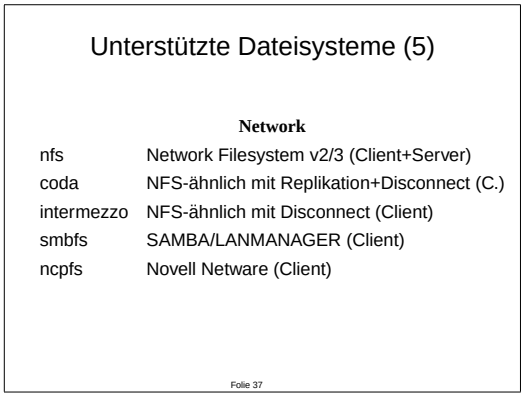
proc	virt. Kernel-Filesystem (wichtig!)
devfs	automatische Device-Generierung (RIP†)
devpts	virt. Terminals
usbfs	USB-spezifisch
capifs	ISDN-Karten

Folie 36

Notizen:

Die angegebenen, Linux-spezifischen, virtuellen Dateisysteme dienen der automatischen (on-demand) Generierung von speziellen Dateien und Verzeichnissen. **`/proc`** ist ein wichtiges

Verzeichnis zum Setzen und Erfragen von Systemparametern, das von einigen Programmen (z.B. `ps`) verwendet wird.



Notizen:

NFS (siehe Abschnitt über NFS) ist das nach wie vor am weitesten verbreitete Netzwerkdateisystem. *Coda* und *Intermezzo* sind Netzwerkdateisysteme, die (zukünftig) eine Resynchronisation von Clients nach nicht-vernetztem Betrieb ermöglichen sollen. Beide befinden sich seit längerer Zeit im Experimentalstudium, werden aber aktiv entwickelt. Serverseitig ist ein zusätzlicher Daemon erforderlich (wie bei Samba).

Smbfs erlaubt das Mounten (ggf. zahlreiche Optionen beim *mount*-Kommando erforderlich) von SMB-Shares auf einen Linux-Rechner, wie bei NFS. Leider unterstützt SMB nicht alle Unix-spezifischen Features, ist aber vergleichbar performant wie NFS (wenn nicht sogar, wegen TCP, etwas schneller im Erkennen von Timeouts).

Ncpfs wird zum Einbinden bestehender Novell-Netware-Shares verwendet, die von einem echten Novell-Server oder vom MARS-NWE (Linux) exportiert werden können. Die unterstützten Dateisystemfeatures und die Stabilität ist ähnlich wie bei smbfs.

5 Datenträgerverwaltung

In diesem Abschnitt lernen Sie das Handling von festen und wechselbaren Datenträgermedien unter Linux kennen.

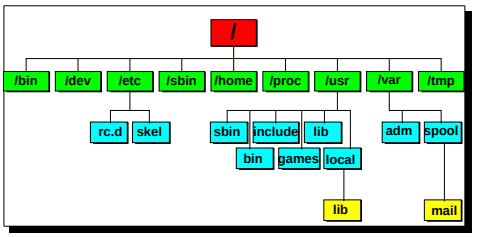
5.1 Eine Reise durch das Unix-Dateisystem

Einige Dateipfade sind Standard unter Unix, und für den Administrator wichtig, um schnell an essentielle Subsysteme und Einstellungen heranzukommen.

`/`	Wurzelverzeichnis und oberste Verzeichnisebene.
`/bin`	Wichtige Programme, die immer zur Verfügung stehen müssen, auch während des Systemstarts.
`/usr/bin`	Global installierte (ausführbare) Programmdateien, die von allen Benutzern genutzt werden können.
`/sbin`	Programme, die beim Systemstart und für den Administrator gebraucht werden.
`/usr/sbin`	Programme, die nach dem Systemstart vorwiegend vom Systemadministrator gebraucht werden.
`/lib`	Laufzeit-Bibliotheken, die auch während des Systemstarts gebraucht werden.
`/usr/lib`	Systembibliotheken, die vorwiegend von Programmen aus `/usr/bin` benötigt werden.
`/dev`	Geräte-Dateien (Abbildung der Peripherie auf Dateien).
`/etc`	Systemweite Konfigurationsdateien.
`/home`	enthält die Benutzer-Unterverzeichnisse, die für den jeweiligen Benutzer schreibbar sind.
`/tmp`	Verzeichnis für Temporärdateien, für alle Benutzer schreibbar.
`/var/*`	Log- und Spooldateien/Verzeichnisse.
`/usr/local/*`	enthält Programmpakete, die nicht zur Standard-Distribution gehören. Manchmal per NFS geteiltes, clusterweites Verzeichnis.
`/proc`	Linux-spezifisches, virtuelles Verzeichnis, das Informationen über das System und zur Laufzeit konfigurierbare Kernel-Parameter in Form von virtuellen Dateien enthält.

Jeder Benutzer (User) hat ein eigenes Verzeichnis, in dem er sich befindet, wenn er sich am System anmeldet (einloggt). Dieses Verzeichnis nennt man **Homedirectory**. Die Heimverzeichnisse der Benutzer befinden sich meist im Dateisystem unter dem Pfad **`/home/benutzerkennung`**. In seinem eigenen Homedirectory kann jeder Benutzer seine Dateien nach eigenen Anforderungen selbst strukturieren und anlegen.

Eine Reise durch das Unix-Dateisystem



Folie 38

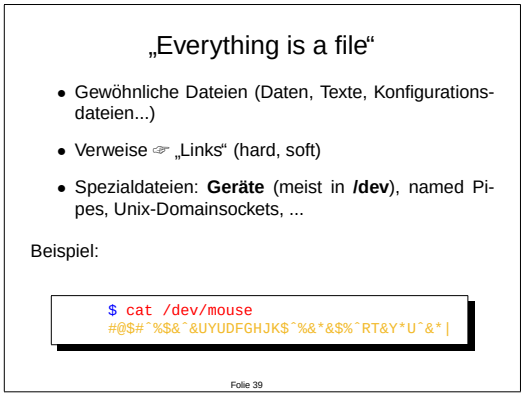
Notizen:

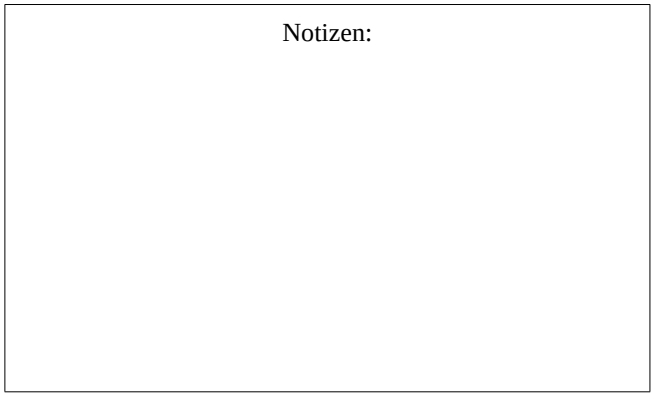
5.2 „Everything is a file“

Viele Systemresourcen werden unter Unix auf **Dateien** abgebildet:

- gewöhnliche Dateien (Files):
  - Texte, Daten
  - Konfigurationsdateien
  - ausführbare Programme
- Verweise (Links):
  - symbolische Links über Dateinamen („Softlinks“)
  - inhaltliche Links („Hardlinks“)
- Devices („Gerätedateien“)

Dateinamen können beliebig lang werden und aus beliebigen Zeichen bestehen (außer dem als Trenner gebrauchten „/“. Einige Dateiendungen sind zwar üblich, aber selten verbindlich oder an den Dateityp gebunden.





Die **Gerätedateien** bilden Hardware auf das Dateisystem ab, und machen damit einen „direkten“, wahlfreien Zugriff auf Datenträgermedien aus der Kommandozeile möglich. Daher ist es unter Unix sehr einfach, z.B. defekte Bereiche auf Datenträgern zu überspringen (Datenrettung!), oder 1:1-Kopien von Disketten, CD-Roms oder Festplattenpartitionen anzulegen.

Eine 1:1-Kopie einer Daten-CD als sogenanntes **Image** erzeugen Sie beispielsweise so:

```
dd if=/dev/cdrom of=/tmp/kopie.iso
```

Das dadurch erzeugte CD-Image `/tmp/kopie.iso` enthält ein vollständiges, bitweises Abbild des ursprünglichen Datenträgers, und läßt sich beispielsweise mit

```
cdrecord /tmp/kopie.iso
```

auf einen neuen Rohling brennen.

5.3 `fdisk` – Partitionieren von Medien

fdisk erinnert vom Namen her an das unter DOS übliche Tool zum Partitionieren von Festplatten. Unter Linux stehen mehrere Programme zur Verfügung, um diese Aufgabe zu erledigen, neben **fdisk** sind dies **parted**, **cfdisk**, und **sfdisk**, die mit unterschiedlichem Bedienkomfort ausgestattet sind.

Es folgt eine Beispielsitzung mit **`fdisk`**, in der die Hilfe abgerufen und anschließend die aktuelle Partitionstabelle der ersten Festplatte am ersten IDE-Controller, **/dev/hda**, aufgelistet wird:

```
$ fdisk /dev/hda
Command (m for help): m

Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatiblity flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

$ fdisk /dev/hda
Command (m for help): p

Disk /dev/hda: 64 heads, 63 sectors, 827 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot Begin  Start    End   Blocks   Id  System
/dev/hda1          1      1     21    42304+   6  DOS 16-bit >=32M
/dev/hda2         22     22     42    42336   83  Linux native
/dev/hda3         43     43     53    22176   82  Linux swap
/dev/hda4         54     54    827  1560384    5  Extended
/dev/hda5         54     54    354   606784+  83  Linux native
/dev/hda6        355    355    385    62464+  83  Linux native
/dev/hda7        386    386    416    62464+  83  Linux native
/dev/hda8        417    417    827   828544+  83  Linux native

Command (m for help):
```

5.4 mkfs – Anlegen von Dateisystemen

mkfs dient dazu, einen Datenträger bzw. eine Partition für das Anlegen von Dateien und Verzeichnissen vorzubereiten, ihn also für das System nutzbar zu machen. Linux unterstützt eine Anzahl auch für andere Betriebssysteme gebräuchliche Dateisysteme, unter anderem **vfat**, das erweiterte DOS-Dateisystem, welches vorwiegend zum Datenaustausch per Diskette verwendet wird, jedoch wegen mangelnder Unterstützung von Unix-spezifischen Features wie Benutzerattribute, Dateirechte, Gerätedateien usw. als Grundlage für ein Unix-Dateisystemlayout unbrauchbar ist.

Für Festplatten wird unter Linux häufig das **ext2**-Dateisystem eingesetzt. Neuere Installationen verwenden transaktionsorientierte Dateisysteme wie **reiserfs** oder **ext3**, welche die Fehlererkennung und -korrektur beim Wiederanlauf des Systems nach Stromausfällen oder anderen außerplanmäßigen Unterbrechungen des Betriebes enorm beschleunigen.

mkfs unterstützt eine große Anzahl von Optionen, die teilweise Dateisystem-spezifisch sind.

Zur Formatierung von DOS-kompatiblen Disketten stellt GNU/Linux die Kommandos **mkdosfs** und **mformat** zur Verfügung. Hierbei sollte, ebenso wie beim Lowlevel-Formatieren mit **fdformat**, die für die gewählte Schreibdichte und Kapazität zuständige *Gerätedatei* als Parameter angegeben werden!

```
$ fdformat /dev/fd0H1440
$ mkdosfs /dev/fd0H1440


$ fdformat /dev/fd0u1440
$ mkdosfs /dev/fd0u1440
```

5.5 fsck – Überprüfen von Dateisystemen

Beim Hochfahren des Rechners werden die in der Dateisystem-Konfigurationsdatei **/etc/fstab** angegebenen Dateisysteme überprüft, und leichte Fehler automatisch korrigiert. Dieser Check ist auch vom Administrator mit dem Kommando **fsck** manuell durchführbar.

```
$ fsck -t ext2 /dev/fd0
Parallelizing fsck version 1.04 (16-May-96)
e2fsck 1.04, 16-May-96 for EXT2 FS 0.5b, 95/08/09
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/fd0: 91/360 files (1.1% non-contiguous), 1083/1440
blocks
```

5.6 mount – Einbinden von Dateisystemen

Mit dem **mount**-Kommando hängen Sie neue Datenträger, auch per NFS freigegebene Verzeichnisse anderer Rechner, an einer wählbaren Stelle im Unix-Dateisystembaum ein.

Mit **umount** lässt sich unter Angabe des zuvor bei `mount` angegebenen Einhängepunktes ein Datenträger wieder aus dem System abmelden. Hierbei darf jedoch kein laufender Prozess mehr auf den Datenträger zugreifen, sonst schlägt das `umount`-Kommando fehl.

Welche Prozesse noch auf den Datenträger (oder eine bestimmte Datei) zugreifen, erfahren Sie mit

```
fuser -v Verzeichnis
```

mount – Einbinden von Dateisystemen

```
$ mount -t ext2 /dev/hda2 /usr

$ mount -t iso9660 -o ro /dev/hdc /mnt/cdrom

$ mount -t vfat /dev/fd0 /mnt/floppy

$ mount /mnt/floppy

$ mount pizza:/mnt/cdrom /mnt/pizza/cdrom
```

Folie 40

Notizen:

5.7 quota und edquota – Einschränkung des zulässigen Plattenplatzverbrauches

Die Kontingente von Plattenplatz für einzelne Benutzer werden vom **Linux-Kernel** verwaltet. Der Benutzer kann mit Hilfe des **quota**-Kommandos die aktuelle Belegung erfragen, der Administrator kann für jeden Benutzer die temporäre („Softquota“) sowie dauerhafte („Hardquota“) Maximalbelegung festlegen.

Quota

Auf Mehrbenutzerrechnern wird den Benutzern oft nur ein bestimmter Plattenplatzverbrauch zugestanden. Dieser kann mit dem Befehl `quota` überprüft und vom Administrator mit `edquota benutzer` für jeden Benutzer festgelegt werden.

```
$ quota -v

Disk quotas for knopper (uid 26001):
 Filesystem   usage   quota     limit   timeleft   files   quota   limit   time
 /home        46451   50000   2997944        ---    3068   50000   90000     --
```

Folie 41

Notizen:

Bei Linux ist die Überprüfung und Beachtung der Diskquotas bereits im Kernel eingebaut, und erfordert keinen speziellen, als Daemon im Hintergrund laufenden Dienst. Einige Tools zur Administration sind dennoch hilfreich:

quotaoff partition	Deaktiviert Quota
quotacheck partition	Generiert/repariert Quota-Informationen
quotaon partition	Aktiviert Quota
repquota partition	Erzeugt einen Report

5.8 /etc/fstab – Die Dateisystem-Konfigurationsdatei

In der Datei **/etc/fstab** befinden sich für diverse Kommandos (u.a. das **mount**-Kommando) die wichtigsten Einstellungen und Konfigurationen des Dateisystembaums unter Linux wie *Gerätedatei, Einhängepunkt im Dateisystem, Dateisystemtyp, Mount-Optionen* und die *Frequenz des automatischen Backups* sowie die *Reihenfolge beim automatischen Dateisystemcheck.*

```
# <device>    <mountpoint>   <type>  <options> <dump> <fsck>

/dev/hda2     /              ext2    defaults  1      1
/dev/hda3     /usr           ext2    defaults  1      2
/dev/hda5     /home          ext2    defaults  1      3

pizza:/www    /mnt/pizza/www nfs     rw,hard,intr,bg 0 0
//pizza/pub   /mnt/pizza/pub smbfs   username=gast,\
                                          password=gast 0 0

/dev/hda6     none           swap    sw
none          /proc          proc    defaults

/dev/fd0      /floppy    vfat    noauto    0      0
/dev/hdc4     /cdrom     iso9660 noauto,ro,user  0 0
```

5.9 Einbinden von Datenträgern – Schritte

Anlegen einer ext3-Partition – Schritte

```
1   $ fdisk /dev/hdb
2   $ mke2fs -N 10000000 -b 1024 /dev/hdb1
3   $ e2fsck /dev/hdb1
4   $ tune2fs -m 0 -j /dev/hdb1
5   $ mount -t ext3 /dev/hdb1 /tmpbig
```

Folie 42

Notizen:

Im vorliegenden Beispiel wird eine Partition für 10 Millionen Dateieinträge, eine Blockgröße von 1kB und 0% (`-m 0`) reservierten Platz für den Administrator vorbereitet.[1] Mit **tune2fs -j** wird nachträglich ein Journal für ext3 angelegt.[2]

[1] **ext2** verwendet eine statische Tabelle für Dateisystemeinträge, daher muss mei **mke2fs** die Anzahl der Dateien/Verzeichnisse bereits beim Formatieren festgelegt werden.

[2] So lassen sich auch vorhandene **ext2**-Partitionen nach **ext3** konvertieren.

Sinngemäß gilt die folgende Übung allgemein zum Einbinden neuer Datenträger, wobei Sie im Falle von Festplatten anstelle der Formatierung partitionieren müssen, und nach erfolgreichem Anlegen der Dateisysteme einen dauerhaften Eintrag in die Konfigurationsdatei **`/etc/fstab`** vornehmen.

Übung

(Überlegen oder testen Sie, für welche dieser Aktionen Sie Administratorrechte benötigen!)

1. Formatieren Sie eine 1.44MB-Diskette mit **`fdformat /dev/fd0H1440`** und legen Sie ein Dateisystem auf der Diskette an mit **`mkdosfs /dev/fd0H1440`**
2. Melden Sie die Diskette an mit **`mount -t vfat /dev/fd0H1440 /mnt/floppy`**
3. Kopieren Sie ein paar der in den vorigen Übungen erzeugten Textdateien aus Ihrem Heimverzeichnis auf die Diskette.
4. Melden Sie die Diskette wieder ab und überprüfen Sie mit Hilfe von **`mdir`**, ob das Kopieren geklappt hat.

Folie 43

Notizen:

5.10 Loopback-Device (Disk)

Mit Hilfe des Loopback-Moduls (nicht zu verwechseln mit der Loopback-Adresse der Netzwerkkarte) ist es möglich, auch Dateien anstelle von Festplattenpartitionen mit **`mount`** einzubinden.

Loopback-Device (Disk)

Direktes Einbinden von Partitions-Images als Dateisystem:

```
losetup /dev/loop2 cdrom.iso
mount -r /dev/loop2 /media/cdrom
```

oder:

```
mount -o loop,ro cdrom.iso /media/cdrom
```

Folie 44

Notizen:

Loopback-Device (Swap)

Swappen auf Dateien wird auch direkt von **`mkswap`** und **`swapon`** unterstützt, geht aber auch mit

```
dd if=/dev/zero of=/media/hda1/datei.swp \
   bs=1000k count=100
mkswap /media/hda1/datei.swp
losetup /dev/loop3 /media/hda1/datei.swp
```

Folie 45

Notizen:

Bei modernen Linux-Distributionen werden als „Plugin“ für **`loop.ko`** auch Verschlüsselungsmodule unterstützt, was das Verschlüsseln von Daten (Dateisysteme, Swap, komplette Partitionen...) auf Blockebene ermöglicht.

Loopback-Device (Crypto)

Voraussetzungen:

1. Kernel mit crypto-lopback Support,
2. crypto-loopback-fähiges **`losetup`** und **`mount`**-Kommando.

Folie 46

Notizen:

Loopback-Device (Crypto)

Einrichten:

```
dd if=/dev/zero of=geheim.img bs=1000k \
   count=100
losetup -e AES256 /dev/loop4 geheim.img
mke2fs /dev/loop4
```

Anschließend kann **`/dev/loop4`** direkt gemountet werden.

Folie 47

Notizen:

Loopback-Device (Crypto)

Mounten (abgekürzt):

```
mount -o loop,encryption=AES256 \
  geheim.img geheim
```

Folie 48

Notizen:

Mit den Loopback-Devices in **`/dev/loop`** kann nach erfolgreicher Assiziierung mit Dateien (fast) alles durchgeführt werden, was auch mit „echten“ Partitionen möglich ist.

Übung

1. Erzeugen Sie sich eine „virtuelle Partition“ als Image-Datei, die ein verschlüsseltes ext2-Dateisystem enthalten soll, und mounten Sie diese in ein Unterverzeichnis Ihres Heimverzeichnisses. Sorgen Sie (ggf. als Administrator) dafür, dass der Benutzer (einer) dies auch ohne Zuhilfenahme von root tun kann.
2. Werden Sie paranoid: Verschlüsseln Sie Ihre Swap-Partition (oder legen Sie eine verschlüsselte Swap-Datei an und binden Sie diese ins System als zusätzlichen Speicher ein).

Folie 49

Notizen:

6 Arten von Dateien im Unix-Dateisystem

Unix/Linux kennt, neben „einfachen“ Dateien und Verzeichnissen, auch diverse „Spezialdateien“, welche z.T. Netzwerk- und Kernel-Funktionen in den Anwenderbereich abbilden.

Dateitypen unter Unix/Linux

In diesem Beispiel wurde ein Verzeichnis namens **`test`** angelegt, in dem sich verschiedene Dateiarten befinden.

```
knopper@Koffer:~$ ls -l test/
insgesamt 8
brw-rw---- 1 root    floppy 2, 0 2006-04-04 12:06 blockdevice
crw-rw---- 1 root    root  10, 1 2006-04-04 12:06 chardevice
-rw-r--r-- 2 knopper users     5 2006-04-04 13:14 datei.txt
drwxr-xr-x 2 knopper users    48 2006-04-04 13:17 directory
prw-r--r-- 1 knopper users     0 2006-04-04 13:16 fifo
-rw-r--r-- 2 knopper users     5 2006-04-04 13:14 hardlink
srwxrwxrwx 1 knopper users     0 2006-04-04 10:06 socket
lrwxrwxrwx 1 knopper users     9 2006-04-04 13:14 symlink
                                                  -> datei.txt
```

Der erste von **`ls -l`** angezeigte Buchstabe in den Dateirechten kennzeichnet die Art der Datei.

Folie 50

Notizen:

6.1 Einfache Dateien

Einfache Dateien

Einfache Dateien können Dokumente, Programme, Bibliotheken oder Daten jedweder Art sein.

```
-rw-r--r-- 2 knopper users     5 2006-04-04 13:14 datei.txt
```

Folie 51

Notizen:

6.2 Verzeichnisse

Verzeichnisse

Verzeichnisse und Unterverzeichnisse sind ein Ordnungsmittel, um Dateien zu kategorisieren und leichter wiederauffindbar zu machen.

```
drwxr-xr-x 2 knopper users    48 2006-04-04 13:17 directory
```

Folie 52

Notizen:

6.3 Symlinks

Symlinks

Symbolische Links, also „Namensverknüpfungen“, sind Zeiger auf Datei- oder Verzeichnisnamen, die den Zugriff auf die entsprechenden Daten unter einem anderen Namen ermöglichen. Sie werden mit dem Kommando
`ln -s datei verknüpfung`
angelegt. Der symbolische Link und sein Ziel ist bei **`ls -l`** deutlich identifizierbar.

```
-rw-r--r-- 2 knopper users 5 2006-04-04 13:14 datei.txt
lrwxrwxrwx 1 knopper users 9 2006-04-04 13:14 symlink -> datei.txt
```

Achtung: Wird die Originaldatei gelöscht, so ist der Inhalt auch nicht mehr über den symbolischen Link verfügbar, obwohl dieser vorhanden bleibt.

Folie 53

Notizen:

6.4 Hardlinks

Hardlinks

Ähnlich wie beim Symbolischen Link auf den NAMEN einer Datei verwiesen wird, wird beim Hardlinks auf den Datei-INHALT verwiesen. D.h. es wird mit
`ln datei hardlink`
eine neue Datei angelegt, deren Inhalt aber mit dem der ersten Datei immer identisch ist. Aus dem Verzeichnislisting ist die Tatsache, dass es sich um einen Hardlink handelt, allerdings nicht ohne weiteres erkennbar.

```
-rw-r--r-- 2 knopper users     5 2006-04-04 13:14 datei.txt
-rw-r--r-- 2 knopper users     5 2006-04-04 13:14 hardlink
```

Wird eine der beiden Dateien gelöscht, so ist der Inhalt nach wie vor unter dem anderen Dateinamen verfügbar. Originaldatei und Hardlink sind also “gleichberechtigt“.

Folie 54

Notizen:

6.5 Character und Block Devices

Character und Block Devices

Character Devices erlauben sequentielles, zeichenweises Schreiben und Lesen von Daten auf die damit verbundenen Geräte (vergl. voriges Kapitel über Dateisysteme und Blockdevices).
Block Devices erlauben wahlfreien Zugriff auf beliebige „Blöcke“ eines Gerätes, ohne dass ein „Vor-“ oder „Zurückspulen“ nötig ist.

```
brw-rw---- 1 root    floppy 2, 0 2006-04-04 12:06 blockdevice
crw-rw---- 1 root    root  10, 1 2006-04-04 12:06 chardevice
```

Block Devices bilden üblicherweise Festplatten ab, während Char Devices Mäuse oder Bandlaufwerke und Modems abbilden.

Folie 55

Notizen:

6.6 Fifos und Sockets

Fifos und Sockets

Während Sockets mit Netzwerkoperationen (z.B. Kommunikation mit dem X-Server) verknüpft sind, sind Fifos ein Mittel, um die Ein- und Ausgabe von Programmen miteinander zu verknüpfen, ähnlich wie mit dem Pipe-Symbol |.

```
prw-r--r-- 1 knopper users     0 2006-04-04 13:16 fifo
srwxrwxrwx 1 knopper users     0 2006-04-04 10:06 socket
```

Folie 56

Notizen:

7 Dateirechte

Die Attribute von Dateien, *Besitzer*, *Gruppe* und die *Dateirechte* sind für die korrekte Funktion eines Unix-Systems essentiell. Sowohl zu restriktive als auch zu freizügige Rechtevergabe können die Stabilität und lokale Sicherheit des Systems ernsthaft gefährden! Daher sollten Sie als Administrator mit den Dateirechten bestens vertraut sein, und diese mit Bedacht wählen!

7.1 chown – Setzen des Dateibesitzers

chown – Setzen des Dateibesitzers

```
chown [Optionen] Benutzer Datei(en)...
```

chown ändert das Besitzer-Attribut von Dateien und Verzeichnissen. Der chown-Befehl kann auf POSIX-konformen Unix-Systemen nur vom Systemadministrator ausgeführt werden.
Der ursprüngliche Besitzer der Datei verliert mit sofortiger Wirkung die Besitzer-Rechte an dieser Datei und kann nur noch aufgrund gesetzter Gruppen- oder globaler Rechte auf die Datei oder das Verzeichnis zugreifen.

```
chown -R demo /home/demo
```

Mit der Option -R kann rekursiv das Besitzerattribut ganzer Verzeichnisbäume geändert werden.

Folie 57

Notizen:

7.2 chgrp – Ändern der Gruppenzugehörigkeit

chgrp – Ändern der Gruppenzugehörigkeit

```
chgrp [Optionen] Gruppe Dateien...
```

chgrp ändert die Unix-Gruppe von Dateien und Verzeichnissen. Der Befehl kann vom Besitzer einer Datei ausgeführt werden, wenn er selbst Mitglied der angegebenen Unix-Gruppe ist (POSIX).

```
$ ls -l helloworld.c
-rw-r--r-- 1 knopper users     29 Aug 5 22:39 helloworld.c
$ groups
users developer
$ chgrp developer helloworld.c
$ ls -l helloworld.c
-rw-r--r-- 1 knopper developer 29 Aug 5 22:39 helloworld.c
```

Folie 58

Notizen:

7.3 chmod – Ändern von Rechten

chmod – Ändern von Rechten

`chmod [Optionen] Änderungen Dateien`

`chmod` ändert die Zugriffsrechte von Dateien und Verzeichnissen. Man kann die **Rechte**

r = read lesen	w = write schreiben	x = execute ausführen	s = suid set ID

an bestimmte **Personenkreise** vergeben

u = user Besitzer	g = group Gruppe	o = others Andere

Mit der Option `-R` werden die Änderungen auch für Unterverzeichnisse durchgeführt.

Folie 59

Notizen:

7.4 Beispiele zu chmod

Beispiele zu **chmod**

```
$ ls -l
total 11
-rw-r--r--  1 knopper  users  7185 Nov 20 23:17 auswertung.sh
-rw-------  1 knopper  users   938 Nov 20 23:17 juli.dat
-rw-------  1 knopper  users   469 Nov 20 23:17 juni.dat
-rw-------  1 knopper  users    54 Nov 20 23:17 mai.dat
$ chmod u+x auswertung.sh
```

Das Script `auswertung.sh` wird zum Ausführen freigegeben.

Folie 60

Notizen:

Beispiele zu **chmod**

```
$ chmod og+r *.dat
$ ls -l
total 11
-rwxr--r--  1 knopper  users  7185 Nov 20 23:17 auswertung.sh
-rw-r--r--  1 knopper  users   938 Nov 20 23:17 juli.dat
-rw-r--r--  1 knopper  users   469 Nov 20 23:17 juni.dat
-rw-r--r--  1 knopper  users    54 Nov 20 23:17 mai.dat
```

Alle dürfen ab jetzt die „`.dat`“-Dateien lesen.

Folie 61

Notizen:

7.5 Spezielle Dateiattribute

Spezielle Dateiattribute

Neben den Standard-Rechten Lesen, Schreiben und Ausführen existieren noch weitere Dateiattribute, die vom Besitzer einer Datei oder vom Systemadministrator gesetzt werden können.

```
$ chmod u+s /usr/bin/cdrecord
$ ls -l /usr/bin/cdrecord
-rwsr-xr-x 1 root root 13956 May 10 17:31 /usr/bin/cdrecord
```

Durch das Setzen des s-Attributes ("s-Bit") für den Besitzer bzw. die Gruppe einer Datei wird beim Ausführen der Datei der Besitzer bzw. die Gruppe des neuen Prozesses auf den Besitzer bzw. die Gruppe der Datei gesetzt.

Folie 62

Notizen:

Eines der wichtigsten Programme, die diese Eigenschaft benutzen, ist das **`su`**-Kommando, das beim Start zunächst (wegen des s-Bit für den Dateibesitzer **root**) mit Systemadministrator-Rechten läuft. Anders wäre das Wechseln vom Benutzer- zum Systemadministratorstatus nicht möglich.

Das **s**-Attribut hat bei Verzeichnissen die Bedeutung, dass neu angelegte Dateien und unterverzeichnisse innerhalb dieses Verzeichnisses automatich die Gruppen- bzw. Besitzerzugehörigkeit desVerzeichnisses erhalten.

Das hier **t**-Attribut sorgt bei Verzeichnissen dafür, dass nur der Besitzer einer Datei diese innerhalb eines Verzeichnisses, das das t-Attribut gesetzt hat, die Datei wieder entfernen kann, obwohl das Verzeichnis durchaus für alle schreibbar sein kann. Häufig wird dieses Feature für das **`/tmp`**-Verzeichnis eingesetzt, in dem alle Benutzer Dateien und Verzeichnisse anlegen dürfen.

Übung

1. Kontrollieren Sie mit **`ls -l`** die gesetzten Dateiattribute der Disketten-Gerätedateien **`/dev/fd0*`**.
2. Geben Sie die Disketten-Gerätedateien für alle Benutzer zum Lesen und Schreiben frei.
3. Sorgen Sie dafür, dass Ihr selbst eingerichteter Normalbenutzer CDs brennen kann, indem Sie ihn in die Benutzergruppe aufnehmen, welche mit dem Programm **`cdrecord`** auf das CD-Brenner-Gerät schreiben kann.

Folie 63

Notizen:

Beachten Sie beim letzten Punkt der Übung, dass jeder Benutzerprozess nur beim Starten seine Gruppenattribute überprüft. Der Benutzer muss sich daher, um seine neuen „Freiheiten" in der CD-Brenner-Gruppe genießen zu können, neu anmelden.

8 Netzwerk-Konfiguration

8.1 Hardware

Wenn nicht bereits bei der Installation des Systems die Netzwerkkarte erfoglreich erkannt und das zugehörige Modul in der Modul-Einstellungsdatei **`/etc/modules.conf`** konfiguriert wurde, kann der erfahrene Administrator dies manuell nachholen. Die Netzwerkkarten erhalten unter Linux virtuelle Gerätenamen (d.h. es existiert kein Gerät im **`/dev`**-Verzeichnis, und sind über diese Namen von diversen Programmen aus ansprechbar.

`/etc/modules.conf` konfigurieren

```
### /etc/modules.conf
alias eth0 wd
options wd io=0x360 irq=5
```

Mit **`modprobe eth0`** oder beim ersten Zugriff auf die erste Netzwerkkarte wird das passende Modul mit den angegebenen Optionen geladen.

Folie 64

Notizen:

Vorsicht: Bei einigen Linux-Distributionen wird **`/etc/modules.conf`** automatisch aus Eingabe-Dateien wie **`/etc/modutils/*`** generiert. Debian: **`update-modules`**

Im System-Log (**`/var/log/syslog`** oder **`/var/log/messages`**) meldet der Kernel das erfolgreiche Laden des Moduls.

```
Nov  7 00:14:58 kernel: loading device 'eth0'...
Nov  7 00:14:58 kernel: wd.c:v1.10 9/23/94 Donald Becker (becker@cesdis.gsfc.nasa.gov)
Nov  7 00:14:58 kernel: eth0: WD80x3 at 0x300,  00 00 C0 68 FB 29 WD8003, IRQ 5,
             i                          shared memory at 0xca000-0xcbfff.
```

WLAN-Karten (Hardware)

PCI-Karten: Wie Netzwerkkarten (LAN)

PCMCIA-Adapter: Über `cardmgr` (**`/etc/pcmcia/wireless.conf`**) oder `hotplug/udev` automatisch, sofern Module und Firmware vorhanden. Notfalls mit **`ndiswrapper`** den Windows-Treiber laden.

USB-Adapter: Über `hotplug/udev` automatisch, sofern Module und Firmware vorhanden. Notfalls mit **`ndiswrapper`** den Windows-Treiber laden.

Folie 65

Notizen:

Einige Hersteller liefern Quelltexte für Kernel-Module mit, die mit den eigenen Kernel-Quellen zusammen gebaut werden müssen. Dies ist jedoch ein für die meisten Endanwender nicht ge-

eignetes Verfahren, und oft sind die Lizenzen, unter denen Teile der angebotenen Module stehen, unklar.

Mit dem **`ndiswrapper`** (Kernel-Modul + Utility) können viele eigentlich für Windows gedachte **`.inf`**-Dateien und die dazugehörenden binären Treiber in die Kernel-Schnittstellen integriert werden, was es auch erlaubt, viele Karten zu nutzen, die eigentlich aus technischen (fehlende Spezifikation) oder lizenztechnischen (keine Verteilungslizenz) Gründen nicht von GNU/Linux unterstützt werden.

8.2 Layer-2 (Ethernet) Parameter einstellen

Ethernet-Parameter einstellen

Bringt viele Administratoren MAC-basierter Firewalls zur Verzweiflung:

```
ifconfig eth0 down
ifconfig eth0 hw ether 00:04:23:44:22:11
```

Hiermit wird die „Hardware-Adresse“ von Netzwerkkarten eingestellt.

Folie 66

Notizen:

Ethernet-Parameter (WLAN) einstellen

```
iwconfig interface [essid {NN|on|off}]
                   [mode {managed|ad-hoc|...}
                   [channel N]
                   [ap {N|off|auto}]
                   [key {NNNN-NNNN|off}]
```

Z.B.: `iwconfig eth1 essid "fhzw" key 123456789ABCDEF01234567890`

Folie 67

Notizen:

Ethernet-Parameter (WLAN) einstellen

Einige Karten (v.a. Prism2) werden nur von „wlan-ng“ (WLAN Next Generation) unterstützt, das eine wesentlich kompliziertere Syntax hat.

Wer WPA statt WEP zur Authentifizierung/Verschlüsselung einsetzen will oder muss, kann auf den **`wpasupplicant`** zurückgreifen.

Folie 68

Notizen:

8.3 IP-Adresse, Netzmaske und Default-Gateway setzen

ifconfig – IP-Adresse und Netzmaske

```
ifconfig eth0 192.168.0.1
       netmask 255.255.255.0
       broadcast 192.168.0.255
ifconfig eth0
```

```
eth0  Link encap:10Mbps Ethernet  HWaddr 00:00:C0:68:FB:29
      inet addr:192.168.0.1 Bcast:192.168.0.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0
      TX packets:0 errors:0 dropped:0 overruns:0
      Interrupt:5 Base address:0x310 Memory:ca000-cc000
```

Folie 69

Notizen:

route – Netzwerkrouten und Gateway(s)

route add -net 192.168.1.0
netmask 255.255.255.0 dev eth0
Setzt eine Route zum Netzwerk 192.168.1.0 auf die gleiche Netzwerkkarte wie vorher 192.168.0.0. Es muß allerdings ggf. vorher eine zweite lokale IP-Adresse auf dem Interface **eth0:1** gesetzt werden, die diesem Netz entspricht.

route add default gw 192.168.0.254
Setzt das „Tor zur Welt“ über den Rechner mit der IP-Adresse 192.168.0.254.

Folie 70

Notizen:

/etc/resolv.conf – Nameserver

In der Datei **/etc/resolv.conf** werden mit dem vorangestellten Schlüsselwort **nameserver** die IP-Adressen der Nameserver angegeben, die befragt werden sollen, wenn Ihr Rechner versucht, einen DNS-Namen aufzulösen (DNS = „Domain Name System“ oder „Service“). Fehlt dieser Eintrag, so kann lediglich über die numerische Adresse, nicht aber über Rechnernamen, auf andere Rechner im Internet zugegriffen werden.
Dynamische Dienste wie **DHCP** (**pump**, **dhclient**, **dhcpcd**) oder **PPP** setzen bei erfolgreichem Verbindungsaufbau automatisch einen gültigen Nameserver in **/etc/resolv.conf** ein. Ein Programm, mit der Sie überprüfen können, ob Ihr Nameserver korrekt arbeitet, ist **nslookup** (interaktiv) oder **host rechnername** (nicht-interaktiv).

Folie 71

Notizen:

Übung

1. Setzen Sie mit **ifconfig** eine zweite IP-Adresse auf das virtuelle Netzwerkkarten-Interface **eth0:1**, welche eine um 1 erhöhte Netzwerkadresse (nicht Host-Adresse!) besitzen soll.
2. Sehen Sie mit **route** nach, ob der Kernel automatisch eine Netzwerkroute für dieses Interface angelegt hat.
3. Versuchen Sie mit **ping**, das zweite, virtuelle Interface Ihres Tischnachbarn “anzupingen“.
4. Sehen Sie mit dem Kommando **traceroute IP-Adresse** nach, welchen Weg Ihre IP-Pakete nehmen, wenn Sie einen bestimmten Rechner außerhalb des lokalen Netzes zu erreichen versuchen.

Folie 72

Notizen:

Um die Netzwerkkonfiguration permanent zu machen, bzw. dafür zu sorgen, dass `ifconfig` und `route` beim nächsten Reboot automatisch wieder mit den richtigen Einstellungen aufgerufen werden, verwenden unterschiedliche Distributionen unterschiedliche Einstellungsdateien in `/etc`. Unter RedHat ist dies beispielsweise `/etc/sysconfig/network` sowie `/etc/sysconfig/network-scripts/ifcfg-eth[0123...]`, unter Debian werden die Netzwerk-Einstellungen unter `/etc/network/interfaces` eingetragen.

8.4 TCP/IP und andere

Wenn Sie sich in der Kernel-Konfguration unter Punkt „Netzwerk-Optionen“ etwas umgesehen haben, werden Sie festgestellt haben, dass Linux nicht nur das Standard-Internetprotokoll **TCP/IP** sondern auch eine ganze Reihe anderer, teilweise älterer Netzwerkprotokolle unterstützt. IPV6, die nächste Generation von TCP/IP ist ebenfalls in einer voll funktionsfähigen Implementation bereits vorhanden.

Um die Netzdienste, die in diesem Abschnitt konfiguriert werden sollen, besser verstehen zu können, soll noch einmal auf die grundlegenden Eigenschaften jeglicher Netzwerkverbindung unter TCP/IP eingegangen werden.

Jedes Datenpaket, das über das Internet verschickt wird, ist durch 5 wichtige Parameter bestimmt, die dafür sorgen, dass die Daten sicher von Punkt A nach Punkt B kommen.

Eigenschaften eines TCP/IP-Paketes

- SOURCE (Herkunfts-) **Adresse**,
- DESTINATION (Ziel-) **Adresse**,
- SOURCE (Herkunfts-) **Port**,
- DESTINATION (Ziel-) **Port**,
- Protokolltyp (**TCP** oder **UDP**).

Der SOURCE-Port auf einem Server kennzeichnet i.d.R. den angesprochenen **Dienst** (s.a. **`/etc/services`**).

Folie 73

Notizen:

8.5 Netzdienste

Um Server-Dienste auf einen bestimmten Port zu binden, gibt es zwei Möglichkeiten.

Netzdienste starten

- Einen Server-Dienst starten, der sich (gemäß seiner Einstellungen) auf einen bestimmten **Port** bindet, oder
- Mit Hilfe des Internet-Metadämons **`inetd`** einen Dienst oder ein Programm mit einem wählbaren **Port** verbinden.

Folie 74

Notizen:

8.6 Der Internet-Metadämon `inetd`

Einige Serverdienste werden bereits beim Hochfahren des Systems durch **`init`**-Skripte gestartet, und binden sich selbständig an spezifische Ports. Auf diese Methode des automatischen oder manuellen Startes von Diensten wird im Abschnitt 2 *init* noch eingegangen. Sie ist, da die Server die ganze Zeit „idle“ mitlaufen, aus Sicht der Clients die performanteste Lösung. Einige Dienste werden jedoch bevorzugt „on demand“ gestartet, da sie entweder selten gebraucht werden, eine spezielle Zugriffskontrolle benötigen oder nicht standalone laufen können.

Zum automatischen Start solcher Netzwerk-Serverdienste beim ersten Zugriff auf einen bestimmten Port, von außen oder lokal, dient der Internet-Metadämon **`inetd`** mit seiner Konfigurationsdatei **`/etc/inetd.conf`**.

`/etc/inetd.conf`

```
# inetd.conf
#       This file describes the services that will be available
#       through the INETD TCP/IP super server.  To re-configure
#       the running INETD process, edit this file, then send the
#       INETD process a SIGHUP signal.
#
# <service_name> <sock_type> <proto> <flags> <user> <ppath> <args>
#
finger  stream  tcp    nowait  root  /usr/sbin/in.fingerd fingerd
ftp     stream  tcp    nowait  root  /usr/sbin/tcpd  in.ftpd -l -a
telnet  stream  tcp    nowait  root  /usr/sbin/tcpd  in.telnetd
shell   stream  tcp    nowait  root  /usr/sbin/tcpd  in.rshd
talk    dgram   udp    wait    root  /usr/sbin/tcpd  in.talkd
```

Folie 75

Notizen:

8.7 Zugriffskontrolle auf Dienste mit tcpd

Im Design des **inetd** war ursprünglich keine Möglichkeit vorgesehen, an zentraler Stelle eine Zugriffskontrolle einzurichten. Vielmehr wurde es als Aufgabe jedes individuellen Serverdienstes angesehen, eine IP-basierte (oder durch Authentifizierung durchgeführte) Zugriffsbeschränkung zu implementieren.

Daher wurde als Sicherheitserweiterung der **tcpd** als ein weiterer Meta-Dämon eingeführt, der mit Hilfe der Dateien **/etc/hosts.allow** und **/etc/hosts.deny** sowohl eine Zugangskontrolle auf IP-Ebene, als auch ein rudimentäres Logging von Zugriffen über den System-Logger **syslogd** erlaubt. Format und Syntax seiner Konfigurationsdateien sind in der Online-Hilfe mit **man 5 hosts_access** angegeben.

Wenn Sie in **/etc/inetd.conf** anstelle eines Serverdienstes manchmal **/usr/sbin/tcpd** angegeben finden, so bedeutet dies, dass die Zugriffskontrolle über diesen Dämon aktiviert ist.

Übung

1. Aktivieren Sie den **pop3**- und den **ftp**-Server in **/etc/inetd.conf**, indem Sie die ggf. vor den entsprechenden Zeilen stehenden Kommentarzeichen # entfernen, **/etc/inetd.conf** abspeichern und dem **inetd**-Prozess ein **HUP**-Signal senden (z.B. mit **killall -HUP inetd**, wenn Sie nicht die Prozess-ID des inetd-Servers nachsehen wollen).
2. Testen Sie das Vorhandensein der nun freigeschalteten Dienste, indem Sie mit dem Universalclient **telnet localhost portnummer** eine Verbindung zu Ihrem eigenen Rechner aufbauen.

Folie 76

Notizen:

Wie Sie anhand dieser Übung sehen, sind einige Dienste (wie der Mail-Abholdienst **pop3**) erstaunlich einfach aufgebaut, und benötigen keine zusätzlichen Konfigurationsdateien. Die Sicherheit solcher Dienste beruht alleine auf der korrekten Einstellung von **inetd** und **tcpd**!

8.8 sendmail als Standalone-Server

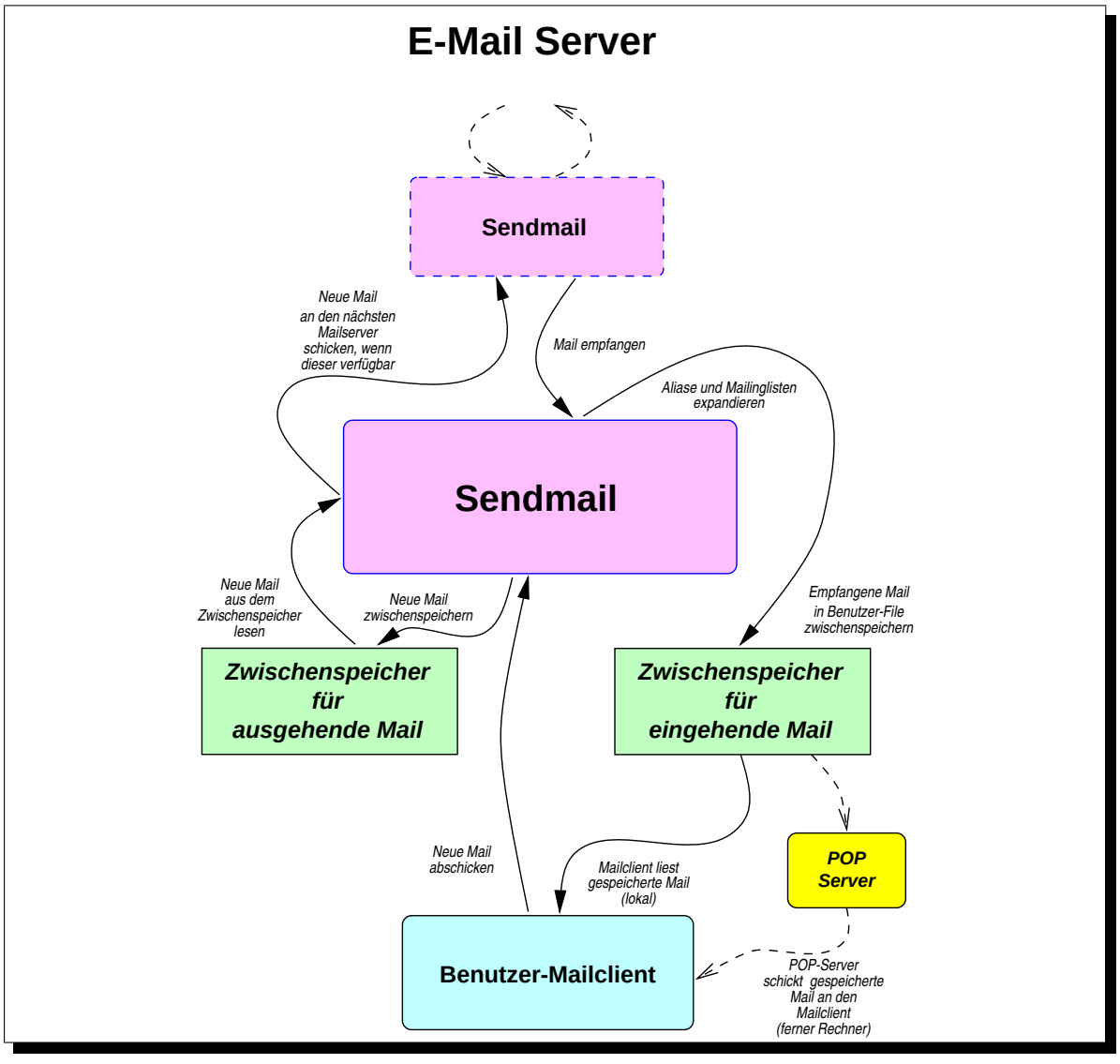
Übung: sendmail aktivieren

1. Konfigurieren Sie mit **yast** oder **webmin** den Mail-Versanddienst **sendmail** für die direkte Zustellung im lokaen Netzwerk, und starten Sie den Dienst, indem Sie das Skript **/etc/init.d/sendmail** mit dem Parameter **start** aufrufen.
2. Testen Sie sendmail, indem Sie **telnet localhost smtp** starten, und bei erfolgreicher Verbindung, das SMTP-**HELP** und **QUIT**-Kommando eingeben.

Folie 77

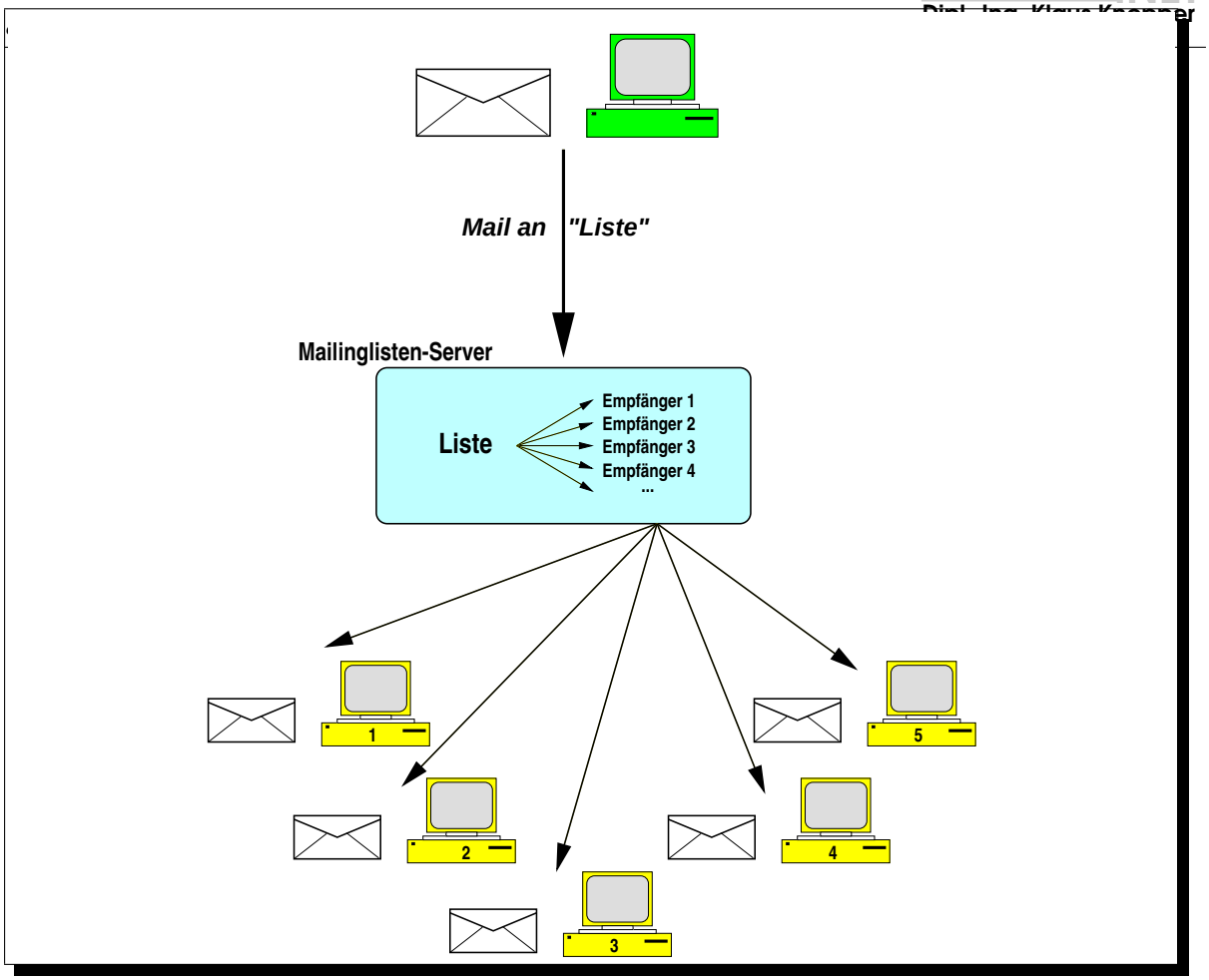
Notizen:

Dass die **`sendmail`**-Konfiguration im Beispiel so schnell vonstatten ging, ist der Ausnahmefall. Wenn Sie einen Mailserver im Internet betreiben möchten, ist i.d.R. eine eingehende Lektüre zum Thema Mail-Delivery und -routing erforderlich. Mehr hierzu erfahren Sie in der Fachliteratur des O'Reilly-Verlages zu den verschiedenen Serverdiensten.



8.9 Mailinglisten

Als Mailserver-Aufsatz gibt es mehrere *Mailinglistenverwalter*-Programme. Eines der beliebtesten davon ist **`majordomo`**, ein (weiteres) Perl-Paket, das die Verwaltung und Wartung von Adresslisten und -gruppen halbautomatisch erledigt. Auch ein Selbsteintrag von neuen Benutzern ist, sofern vom Administrator zugelassen, möglich („subscribe", „approve").



Übung: Majordomo

Wenn Majordomo erfolgreich auf Ihrem System eingerichtet wurde, finden Sie die (Test-)Mailinglisten im Verzeichnis **`/var/lib/majordomo/lists`**. Dies sind gewöhnliche Textdateien, die Sie z.B. mit **`vi`**, oder mit **webmin** verändern können.
In **`/etc/mail/aliases`** finden Sie die Verweise auf Majordomo-Listen. Diese Datei ist Bestandteil des **`sendmail`**-Systems, und ermöglicht eine Zuordnung zwischen Mailadressen und Benutzern (oder, wie im Fall von Majordomo, von Programmen).

Folie 78

Notizen:

9 Das Netzwerk Dateisystem NFS

NFS ist ein typischer RPC („Remote Procedure Call“)-Dienst, und mit erstaunlich wenig Aufwand einzurichten.

NFS – Serverdienste

Um einen NFS-Server zu betreiben, ist neben dem RPC-Portmapper **`portmap`** mindestens der Start der folgenden Server notwendig:

`rpc.mountd` Kontrolliert und autorisiert **`mount`**-Zugriffe auf die freigegebenen Verzeichnisse aufgrund der Einstellungen in **`/etc/exports`**.

`rpc.nfsd` Ist in einer oder mehreren Instanzen für die tatsächliche Datenübertragung verantwortlich.

Debian: `/etc/init.d/portmap start ; /etc/init.d/nfs-kernel-server start`
Auf der Client-Seite ist nur ein NFS-fähiger Kernel und das **`mount`**-Kommando erforderlich!

Folie 79

Notizen:

Hinweis: Auf Client-Seite kann noch ein laufender Portmapper erforderlich sein: das **`mount`**-Kommando versucht nämlich u.U., einen lokalen RPC-Dienst zur Verwaltung von Locks zu kontaktieren, was, wie bei UDP-Diensten üblich, zu langen Timeout-Wartezeiten führen kann.

`/etc/exports` – Dateisysteme exportieren

Die einzige Konfigurationsdatei auf NFS-Server-Seite ist **`/etc/exports`**, welche die *Namen* der exportierten Verzeichnisse sowie die erlaubten *IP*- oder *Netzwerkadressen* der Clients enthält.

```
/mnt/cdrom pizza(ro,nosuid,nodev)
/usr       lasagne(ro)        sushi(ro)
/tmp       lasagne(rw,no_root_squash)
```

Folie 80

Notizen:

Die Optionen im vorigen Beispiel bedeuten:

ro Das Verzeichnis wird **read-only** exportiert.

nosuid Das s-Attribut bei Dateien wird ignoriert (Security!).

nodev Device-Dateien werden ignoriert (Security!).

no_root_squash Der clientseitige Administrator wird serverseitig NICHT auf die anonyme “Gast-Benutzer-ID“ gesetzt, d.h. es dürfen wie bei lokalen Datenträgern alle Dateien von `root` gelesen werden.

mounten von NFS-Dateisystemen

Das **mount**-Kommando kennt einige Erweiterungsoptionen für NFS-Dateisysteme, die gerade beim automatischen Mounten von NFS-Verzeichnissen beim Systemstart nützlich sein können.

`mount -o bg,hard,intr lasagne:/tmp /mnt/tempnfs`

bg „Background“, wenn das Remote-System momentan nicht erreichbar ist, legt sich das **mount**-Kommando nach kurzem Timeout in den Hintergrund und versucht weiter, das NFS-Verzeichnis einzubinden.

hard Netzausfälle führen nicht zu Schreibfehlern, sondern es wird so lange gewartet, bis das NFS-Dateisystem wieder zur Verfügung steht. Notfalls unendlich lange.

intr Schreib-/Lesezugriffe auf NFS dürfen durch Steuerung-C oder Signale unterbrochen werden.

Folie 81

Notizen:

NFS ist ein traditionell auf dem UDP-Protokoll basierender Dienst. Daher lässt sich der Ausfall eines NFS-Servers nur schwer feststellen, und es kann zu langen Timeouts kommen. Neuere Versionen von NFS (unter Linux client- wie serverseitig im Kernel) unterstützen auch NFS über TCP, jedoch hat dieses Feature momentan - zumindest offiziell - nur experimentellen Status.

Inzwischen wird anstelle von NFS auch hin und wieder das SMB-Protokoll für File & Printsharing unter Linux eingesetzt, obwohl dies eigentlich eher unter Windows üblich ist. SMB fehlt allerdings der Support für viele unter Unix notwendige Attribute, um als vollständiger Ersatz für NFS herhalten zu können.

Weitere Netzwerkdateisysteme in der Entwicklung sind das CODA-Dateisystem, INTERMEZZO und AFS. NFS erfreut sich weiterhin großer Verbreitung, da es durch seine Einfachheit in der Konfiguration und Anwendung und seine Stabilität besticht, wenngleich es auf Seite der Performance doch einige Schwächen aufweist.

Einige weitere Mount- und Export-Optionen sind in der Manpage zu nfs beschrieben. Bei Mounts über VPN-Router kann beispielsweise die **insecure**-Option wichtig sein, um das strikte Vergleichen von Ports und IP-Adressen der Portmapper zu unterbinden. Ist der Portmapper mit der **libtcpwrap** übersetzt, so muss den Clients mittels korrekter Einträge in **/etc/hosts.allow** noch der Zugriff auf den portmapper-Dienst gestattet werden.

Für die Client-Seite von NFS sind ebenfalls eine Reihe von Timeout-Optionen in der Manpage zu nfs beschrieben, die jedoch nur selten eingesetzt werden (zumal sie bei ungeschickter Wahl u.U. zu Datenverlusten führen können).

Übung

1. Exportieren Sie Ihr **/tmp**-Verzeichnis read-only an ALLE.
2. Teilen Sie diese Änderung den NFS-Diensten durch Aufruf von **exportfs -a** mit. Falls diese noch nicht gestartet wurden, starten Sie sie manuell durch Aufruf von **/etc/init.d/nfs-server start**.
3. Überprüfen Sie die exportierten Verzeicnisse mit **showmount -e**.
4. Versuchen Sie, die von Ihrem Tischnachbarn freigegebenen Verzeichnisse auf Ihren eigenen Rechner unter dem Pfad **/mnt/nfs** zu mounten (Sie müssen dieses Verzeichnis vorher anlegen).

Folie 82

Notizen:

10 Das Common Internet File System (CIFS)

Übung – Windows-Heimverzeichnis mounten

1. Versuchen Sie, Ihr Windows-Heimverzeichnis auf Ihren Linux-Client einzubinden! Im folgenden Beispiel ist **`benutzername`** Ihr Windows-Login-Name. Nach dem zugehörigen Passwort werden Sie beim Mounten gefragt.

```
cd
mkdir fh-home
su
mount -t cifs -o user=benutzername,domain=DS \
  //zwo222-fs1.ds.fh-kl.de/benutzername\$ fh-home
```

2. Bauen Sie eine „Abkürzung“ in /etc/fstab ein, so dass der Benutzer in Zukunft nur noch **`mount fh-home`** in seinem Heimverzeichnis tippen muss, ohne root zu werden.

Folie 83

Notizen:

Übung – SAMBA-Freigaben

1. Überprüfen Sie, ob die Benutzer-Heimverzeichnisse (Samba-Sharename **`[homes]`**) mit oder ohne Anmeldung freigegeben werden in der **`/etc/samba/smb.conf`**.
2. Geben Sie Ihr CDROM-Verzeichnis für ALLE, OHNE PASSWORT unter dem Sharenamen **`[public]`** frei.
3. Überprüfen Sie die SAMBA-Konfiguration mit **`testparm`**.
4. Starten Sie den SAMBA-Server über sein Init-Skript, und überprüfen Sie mit Hilfe von konqueror (**`smb:/`**), ob Ihr Samba-Server im Netz sichtbar ist. Falls er nicht öffentlich freigegeben ist (**`browsable`**-Direktive), versuchen Sie die Adresse **`smb://localhost`**.

Folie 84

Notizen:

11 SSH

ssh kann mehr als nur eine „telnet-ähnliche Session verschlüsseln“.

SSH

- kann eine Verbindung zwischen zwei Rechnern verschlüsseln,
- kann über **Public Key Verfahren** (56) authentifizieren/einloggen,
- kann beliebige Ports und Dienste über die verschlüsselte Verbindung tunneln (**`-L`** und **`-R`** Optionen),
- kann auch graphische Programme (X11) über eine authentifizierte, verschlüsselte Verbindung tunneln,
- kann einige Dienste (FTP, rsync, rsh) in einer verschlüsselten Variante zur Verfügung stellen.

Folie 85

Notizen:

Passwortlose Authentifizierung per SSH

1. Keypaar (einmalig!) erzeugen:

   ```
   ssh-keygen -t dsa
   ```

2. Public Key (**`.ssh/id_dsa.pub`**) auf Remote-System in **`.ssh/authorized_keys`** eintragen.
3. Host-Keys in Hin- und Rückrichtung austauschen (ssh-connect).

Folie 86

Notizen:

Graphische Programme remote starten

```
ssh -X user@remotehost gimp
```

Man beachte auch den durch **`-X`** auf dem **`remotehost`** geöffneten Tunnel-Port, auf den die Variable **`DISPLAY`** verweist.

Folie 87

Notizen:

Advanced: Verschlüsselen Proxy-Zugang einrichten

```
ssh -C -L 3128:localhost:3128 user@proxy-host
```

`-L localport:remotehostip:remoteport` öffnet einen lokalen Port auf dem Client, der über eine verschlüsselte SSH-Session mit dem **`remoteport`** auf der **`remotehostip`**-Adresse verbunden wird. Die Datenüberragung ist wegen **`-C`** zusätzlich komprimiert.

Eine Verbindung auf dem SSH-Clientrechner zu dessen Port **`3128`** nach dem o.a. Kommando würde also mit dem Port **`3128`** auf dem entfernten Rechner Kontakt aufnehmen, wodurch man z.B. einen Proxy im Außennetz ansprechen kann, wenn man im lokalen Browser **`localhost:3128`** als Proxy-Adresse einträgt.

Folie 88

Notizen:

Übung zur SSH

1. Erzeugen Sie sich ein Public/Secret-Key Paar für SSH, und konfigurieren Sie den passwortlosen Remote-Login damit (testen können Sie dies in der nächsten Aufgabe).
2. Starten Sie als root **`sshd`** und loggen Sie sich (als Normalbenutzer) auf dem eigenen Rechner per SSH ein.
3. Wiederholen Sie die letzte Aufgabe mit einer neuen Shell, tunneln Sie diesmal alle X11-Verbindungen, sowie Port 25 des „Remote-Rechners“ auf Port 2500 Ihres „Client-Rechners“. Starten Sie den sendmail-Daemon und testen Sie, ob Sie ein connect zum Remote-Sendmail über Port 2500 bekommen (**`telnet localhost 25`**).

Folie 89

Notizen:

11.1 SSL - Secure Socket Layer

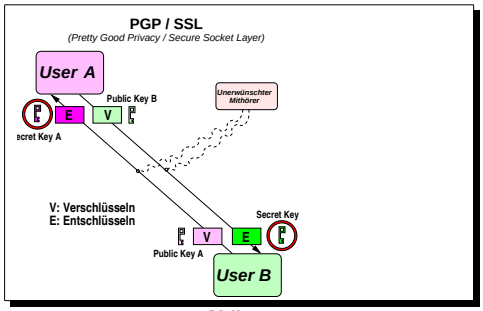
Wenn Webseiten von einem Webserver abgerufen werden, wird Information üblicherweise im Klartext übertragen. Weder das TCP/IP-Protokoll, noch das Protokoll `http` sehen einen Mechanismus zum Schutz der übertragenen Daten vor Ausspähen vor. Dies ist natürlich gerade dann fatal, wenn sensitive Daten wie personenbezogene Informationen, persönliche Briefe, Kreditkartennummern oder Passwörter übertragen werden.

SSL (Secure Socket Layer) ist ein 1993 von Netscape Inc. entwickeltes Verfahren zur transparent verschlüsselten Übertragung von Daten und zur Verifikation des Absenders mittels elektronischer Signatur. Das im E-Mail Umfeld bekanntere PGP (Pretty Good Privacy) ist ein nach dem gleichen Prinzip funktionierendes Verschlüsselungs- und Verifikationsverfahren, das aber vorwiegend zum Verschlüsseln und Signieren von Dateien (z.B. E-Mail Inhalte und Attachments) verwendet wird und nicht zur eigentlichen Übertragung von Daten (wie SSL).

Beiden Verfahren gemeinsam ist, dass asymmetrisch verschlüsselt wird. Vereinfacht dargestellt wird zunächst bei der Einrichtung der Software ein Schlüsselpaar erzeugt. Der sogenannte *öffentliche Schlüssel* (”Public Key”, PK) dient zum Verschlüsseln von Daten (z.B. der vertraulichen Inhalt einer E-Mail). Im Gegensatz zu symmetrischen Verschlüsselungsverfahren kann der verschlüsselte Datensatz jedoch mit dem gleichen Schlüssel nicht wieder dechiffriert bzw. in Klartext übersetzt werden. Nur mit dem zum öffentlichen Schlüssel passenden, sogenannten *geheimen Schlüssel* (”Secret Key”, SK) kann der verschlüsselte Datensatz wieder in die ursprüngliche Form gebracht werden.

11.1 SSL - Secure Socket Layer

SSL - Secure Socket Layer



Folie 90

Notizen:

Übung

Mit

```
openssl req -new -x509 -nodes \
        -out cert.crt -keyout cert.key
```

erzeugen Sie ein SSL-Zertifikat für Apache. Wichtig ist hierbei der **Common Name**, der den nach außen gültigen DNS-Namen (oder die IP-Adresse) des WWW-Servers enthalten muss. Kopieren Sie anschließend das neue Zertifikat an die Stelle, die in den Direktiven **`SSLCertificateFile`** und **`SSLCertificateKeyFile`** in der Apache-Konfiguration angegeben ist. Starten Sie Apache neu und testen Sie, ob Sie über **`https:`** nun einen neuen Public-Key in Netscape von Ihrem Apache-Server erhalten.

Folie 91

Notizen:

12 Prozess- und Jobverwaltung

Der sogenannte **Scheduler** im Linux-Kernel verwaltet quasi-parallel (oder tatsächlich parallel bei Multiprozessor-Systemen) ablaufende Programme im Multitasking-Betrieb. Sowohl dem Benutzer, als auch dem Administrator stehen die Befehle **`ps`** und **`kill`** zur Verfügung, um den Prozess-Status zu erfragen und einem Prozess, der durch eine im System eindeutige Nummer, die **Prozess-ID** gekennzeichnet ist, **Signale** zu schicken. Insofern ist der Name **`kill`**, der lediglich ein Signal verschickt, etwas missverständlich, da er keineswegs in jedem Fall den angegebenen Prozess beendet.

12.1 `ps` – Prozessinformationen anzeigen

`ps` – Prozessinformationen anzeigen

`ps` `[Optionen]`

`ps` zeigt die Liste der laufenden Prozesse (=Programme) an. Das Kommando ist insbesondere im Zusammenspiel mit `kill` sehr praktisch, um die Prozess-ID „amoklaufender" Programme zu erfahren und diese „gewaltsam" zu beenden.

```
$ ps aux
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
root         1  0.0  0.0   796   128  ?  S    Jun 12   0:58 init
root         2  0.0  0.0     0     0  ?  SW   Jun 12   3:45 (kflushd)
root         3  0.0  0.0     0     0  ?  SW   Jun 12   0:00 (kpiod)
root         4  0.0  0.0     0     0  ?  SW   Jun 12   6:53 (kswapd)
root       296  0.2  0.0   820   188  ?  S    Jun 12 101:09 syslogd
knopper   4832  0.1  0.8  3044  2148  pf S    23:09   0:30 gv unixkurs.ps
knopper   4936  0.0  0.3  1436   996  q3 S    23:15   0:01 vi unixkurs.tex
```

Folie 92

Notizen:

12.2 `ps` – Prozessinformationen anzeigen

`kill` – Signal an Prozess schicken

`kill` `[Signal] Prozeßnummer`

`kill` versendet Signale an einen laufenden Prozess. Wenn `kill` ohne die Signal-Option ausgeführt wird, wird der angeführte Prozeß mit dem Signal `TERM` beendet. Die Signal-Option `-HUP` (Hang Up) gibt dem Programm die Möglichkeit, sich „sauber" zu beenden, und dient bei einigen Systemprozessen (daemons) dazu, Konfigurationsdateien neu einzulesen. Bei hartnäckigeren Fällen hilft das Signal `-KILL`, gegen das sich kein Prozess wehren kann.

$ kill 1234	beendet den Prozeß 1234
$ kill –KILL 1233	beendet den Prozeß, falls der obere Versuch erfolglos bleibt.

Folie 93

Notizen:

12.3 Shell-interne Jobverwaltung

Die Shell „merkt" sich in einer Art internen Prozessliste, zusätzlich zur systemweiten Prozessverwatung, welche Programme mit nachgestelltem „**&**" als Hintergrundprozesse („Jobs") ge-

startet wurden, und bietet weitergehende Möglichkeiten zu deren Verwaltung. Eine Übersicht hierzu finden Sie in Anhang C „Tipps und Tricks zur Shell“.

Übung

1. Starten Sie das Programm **`xclock`** in der Shell als *Hintergrundprozess*.
2. Kontrollieren Sie mit dem Shell-internen Kommando **`jobs`** Ihre Hintergrundprozesse.
3. Finden Sie die Prozess-ID des laufenden `xclock` mit Hilfe von **`ps aux | grep xclock`**.
4. Halten Sie den Prozess kurzfristig an, indem Sie ihm mit **`kill -STOP Prozess-ID`** ein STOP-Signal schicken. Beobachten Sie die Reaktion des Programms.
5. Lassen Sie den Prozess weiterlaufen, indem Sie ihm das Signal **`CONT`** schicken.
6. Terminieren Sie den Prozess mit dem Signal **`TERM`**.

Folie 94

Notizen:

13 Kernel

Der **Kern** eines Unix-Betriebssystems ist der Kernel. Er steuert den Ablauf der Programme im Multitasking-Betrieb (**Scheduler**) und ermöglicht ihnen den Zugriff auf die Hardware (Speicher, Dateisystem, Grafikkarte usw.) über hardwarenahe **Module** oder eingebaute **Gerätetreiber**. Dabei sorgt die **Speicherverwaltung** als Bestandteil des Kernels dafür, dass jeder Prozeß einzig und allein den ihm zugewiesenen Speicherbereich „sieht“ und benutzen kann.

Der Kernel ist die Schnittstelle zwischen der Hardware und jeglicher Anwendersoftware, und kontrolliert als oberste Instanz den Zugriff von Prozessen auf alle Hardware-Resourcen. „Linux“ ist eigentlich wie in im Abschnitt 1.2 bereits erwähnt, nur der Betriebssystemkern, die Anwender- und Systemsoftware hingegen setzt sich vorwiegend aus GNU-Programmen der Free Software Foundation zusammen.

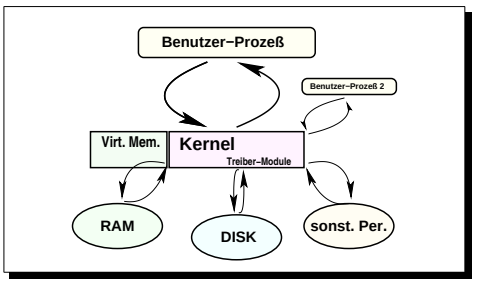
Der Linux-Kernel

- kontrolliert und steuert alle Zugriffe von Anwendersoftware auf Hardware und Peripheriegeräte,
- enthält den Prozess-Scheduler (multitasking),
- verwaltet alle Datenträger und bildet deren Inhalt auf Verzeichnisse ab,
- verwaltet „echtes“ und virtuelles RAM,
- regelt die Kommunikation zwischen Prozessen,
- enthält den Netzwerk-Stack und beherrscht die zugrundeliegenden Netzwerk-Protokolle,
- bestimmt die Stabilität des Gesamtsystems.

Folie 95

Notizen:

Kernel-Schema



Folie 96

Notizen:

13.1 Kernel-Konfiguration

Einige Systemparameter lassen sich zur Laufzeit durch virtuelle Dateien im **/proc**-Dateisystem tunen, andere durch Optionen von **Modulen**, dynamisch ladbaren Bestandteilen des Kernels, verändern.

Um einen auf das eigene System optimierten Kernel zu erhalten, muss das Kernel-Binary und die dynamischen Module aus den Quelltexten, welche sich traditionell unter **`/usr/src/linux`**

befinden, neu übersetzt werden. Hierfür gibt es Konfigurationshilfen, z.B. das textorientierte **`make menuconfig`** oder das graphische **`make xconfig`**.

Die Konfiguration des Kernels setzt ein gewisses Maß an Kenntnissen über die verwendete Hardware und aktuelle Standards voraus, denn nicht alle Optionen sind in der Online-Hilfe im Detail erklärt.

Eine etwas detailliertere Anleitung zum Kernel-Neubau und zur Installation finden Sie im Anhang D „Kernel neu konfigurieren und installieren“.

Übung

1. Ändern Sie (als Administrator) rekursiv (d.h. inclusive Unterverzeichnissen) die Besitzrechte des Verzeichnisses **`/usr/src/linux/.`** auf die Benutzerkennung Ihres normalen Benutzeraccounts.
2. Wechseln Sie (als Normalbenutzer) in das Verzeichnis **`/usr/src/linux/.`** und führen Sie **`make xconfig`** aus.
3. Sehen Sie sich ein wenig in den Kerneloptionen um. Sie können auch Änderungen durchführen, brauchen diese jedoch nicht zu speichern.

Folie 97

Notizen:

13.2 Kernel-Update

Außer bei signifikant neuen Features, die Sie unbedingt verweden möchten, oder sicherheitsrelevanten Updates besteht selten die Notwendigkeit, einen neuen Kernel zu installieren. Bei Umbauten an der Hardware oder Einfügen neuer Karten, für die es noch kein fertiges Kernel-Modul gibt, kann es jedoch hin und wieder notwendig sein, einen neuen Kernel zu installieren oder zumindest ein bisher nicht aktiviertes Modul zu übersetzen und zu installieren.

Am Beispiel einer neu installierten Netzwerkkarte soll dies exemplarisch nachvollzogen werden.

Netzwerkkarten-Modul aktivieren

1. Starten Sie (als Normalbenutzer) die Kernel-Konfiguration im Textmodus mit **`cd /usr/src/linux`** und **`make menuconfig`**.
2. Suchen Sie nach dem Modul für eine ältere **WD800x**-Netzwerkkarte. Dieses verbirgt sich unter dem normalerweise inaktiven Punkt „other ISA cards“ im Netzwerkkarten-Konfigurationsmenü.
3. Aktivieren Sie die Unterstützung für diese Karte als **Modul**, speichern Sie die Einstellung und starten Sie die Kernel-Übersetzung mit **`make bzImage modules`**.

Folie 98

Notizen:

Da die Übersetzung des Kernels auch auf schnellen Rechnern doch eine gewisse Zeit in Anspruch nimmt, und Sie mit Sicherheit nicht ALLE Konfigurationsoptionen auf Anwendbarkeit

für Ihr System hin überprüft haben, verzichten wir auf die in Anhang D beschriebene Installation von Kernel und Modulen sowie Neuschreiben des Master-Boot-Records Ihrer Festplatte. Sie können einen neuen Kernel jedoch auch auf Diskette kopieren und testen (wobei Sie allerdings vorher die dyamischen Module mit **`make modules_install`** als Administrator installiert haben sollten, damit diese vom neuen Kernel auch gefunden und benutzt werden können).

Bootfähige Diskette mit Kernel-Image

```
cd /usr/src/linux
dd if=arch/i386/boot/bzImage of=/dev/fd0
```

erzeugt eine bootfähige Diskette für i386-basierte PCs. Mit

```
rdev /dev/fd0 /dev/hda1
```

können Sie nachträglich einstellen, dass das Wurzelverzeichnis Ihres Linux-Systems sich auf der ersten Partition der ersten Festplatte am ersten IDE-Controller befindet.

Folie 99

Notizen:

14 Der System-Logger Dämon `syslogd`

Die meisten Programme unter Unix schreiben Status- oder Fehlermeldungen auf die Standardausgabe (die Shell, aus der heraus sie gestartet wurden). Viele Systemdienste beherrschen jedoch eine weitere, sehr praktische Methode, regelmäßige Statusberichte abzuliefern: das Logging per **`syslogd`**.

Der **`syslogd`** ist ein lokaler Systemdienst, dessen einzige Aufgabe es ist, Statusmeldungen diverser Programme entgegenzunehmen, und in definierbare Reportdateien („Logs“) zu schreiben, welche sich standardgemäß im Verzeichnis **`/var/log`** und Unterverzeichnissen befinden.

`syslogd` kann jedoch nicht nur in Dateien schreiben, sondern auch über das Netzwerk Nachrichten an auf anderen Rechnern laufende *syslogd*-Prozesse schicken, was einer Manipulation von lokalen Logdateien durch potentielle Eindringlinge entgegenwirkt. Weiterhin kann *syslogd* Nachrichten auf virtuelle Terminals („Konsolen“) senden oder per Mail verschicken.

Die Konfigurationsdatei des *syslogd* heißt **`/etc/syslog.conf`** und ist in der Online-Hilfe unter **`man 5 syslog.conf`** im Detail beschrieben.

Der Logger-Dämon **`syslogd`**

- nimmt Statusmeldungen von verschiedenen Systemdiensten sowie Kernel-Meldungen entgegen,
- schreibt diese in Logdateien unter **`/var/log`**, per Mail an Benutzer, auf lokale Konsolen oder schickt sie übers das Netz an weitere *syslogd*-Prozesse,
- kann über die Einstellungen in **`/etc/syslog.conf`** entscheiden, **welche** (subsystem, loglevel) Meldungen **wohin** geschickt werden.

Folie 100

Notizen:

Übung

Stellen Sie mit dem Kommando

```
grep pop /var/log/* | less
```

fest, in welcher Logdatei pop-3 Verbindungen (also Versuche, Mail von Ihrem Rechner abzurufen) mitprotokolliert wurden, und wann zuletzt eine solche pop-3 Verbindung stattgefunden hat.

Folie 101

Notizen:

Übung

Sehen Sie sich die Datei **`/var/log/messages`** mit dem Kommando **`less`** etwas genauer an. Können Sie feststellen, wann der Rechner das letzte Mal neu gebootet wurde, und was dabei im Detail ablief?
Vergleichen Sie die entsprechenden Meldungen mit der Ausgabe des Kernel-Logbuffers, die Sie mit dem Kommando **`dmesg`** erhalten.

Folie 102

Notizen:

Übung

Stellen Sie in **`/etc/syslog.conf`** ein, dass zusätzlich ALLE, auch die „unwichtigen“ Systemmeldungen (**`*.*`**) auf die Textkonsole 12 (auf diese können Sie, wie Sie sich vielleicht erinnern, mit Steuerung-Alt-F12 umschalten, zurück zu X11 finden Sie mit Steuerung-F7) ausgegeben werden. Auch der **`syslogd`** verwendet das Signal **`HUP`**, um von der Änderung informiert zu werden und diese zu beachten.

Folie 103

Notizen:

15 Der Timer-Dämon crond

crond (manchmal auch einfach als das *cron*-System bezeichnet) verwaltet eine Liste von zeitgesteuert wiederkehrenden Tasks, die automatisch zu bestimmten Zeiten unter der User-ID des Benutzers, welcher die entsprechenden Einstellungen mit **crontab** vorgenommen hat, ausgeführt werden.

15.1 crond und atd

Ein seltener eingesetztes Gegenstück zum **crond** ist der **atd**, welcher über das Kommando **at** gesteuert wird, und ein Kommando nur *einmalig* zu einer bestimmten Zeit ausführt. Ein Beispiel für eine solche zeitgesteuerte Ausführung ist das **shutdown**-Kommando: Der Administrator kann eine Wartezeit angeben, nach der das System heruntergefahren wird, um allen Benutzern noch Gelegenheit zu geben, ihre Arbeit zu beenden.

15.2 crontab

Einige Systemtasks sind unter GNU/Linux in **/etc/crontab** vordefiniert und werden, unabhängig vom *crontab*-Kommando (s.u.) in Form von Shell-Skripten unterhalb der Verzeichnisse **/etc/cron.d/***, **/etc/cron.weekly**, **/etc/cron.daily**, **/etc/cron.hourly** in den jeweils spezifizierten Zeiträumen ausgeführt. Es handelt sich hierbei meist um „Aufräumarbeiten“ wie das Löschen von Temporärdateien aus dem **/tmp**-Verzeichnis, Sortieren, Komprimieren und Rotieren von Logdateien oder ähnliches.

Das Format der *crontab*-Einstellungen ist in **man 5 crontab** beschrieben, die Hilfe zum **crontab**-Kommando selbst hingegen in **man 1 crontab**.

crontab

richtet eine **cron**-**Tab**elle ein, in der die Zeiten und Kommandos angegeben sind, die von **crond** automatisch abgearbeitet werden sollen. Das am Ende jeder Zeile angegebene Kommando wird mit der Benutzer-ID ausgeführt, unter der **crontab** aufgerufen wurde.
Vorsicht: Wird **crontab** ohne Argumente aufgerufen, so überschreibt es eine eventuell existierende Einstellung mit den Eingaben von Tastatur, auf die es beharrlich bis zur Eingabe von Steuerung-C (Abbruch) oder Steuerung-D wartet!
Der komfortabelste Aufruf ist **crontab -e**, wodurch die aktuellen Einstellungen, sofern existent, in den **vi** geladen und bearbeitet werden können.

Folie 104

Notizen:

15.2 crontab

Übung

Richten Sie sich mit **crontab** eine Zeittabelle ein, in der zu jeder vollen Stunde einmal die Logdatei **/var/log/auth.log** auf fehlgeschlagene Login-Versuche hin untersucht wird. Da diese Logdatei i.A. nur vom Administrator lesbar ist, müssen Sie dies ausnahmsweise als **root** tun. Tragen Sie mit dem Kommando **crontab -e** folgende Tabelle ein:

```
0 * * * * grep fail /var/log/auth.log
```

und speichern Sie sie mit dem vi wie gewohnt mit **:wq** . Sie (bzw. **root**) sollten nun zu jeder vollen Stunde eine E-Mail mit einer entsprechenden Ausgabe (oder Fehlermeldung) erhalten.

Folie 105

Notizen:

Übung

In der vorangegangenen Aufgabe hat die eingetragene Zeile den Nachteil, dass auch bei unverändertem Stand der Dinge immer eine Meldung erzeugt wird. Das folgende Shellskript, das Sie unter dem Namen **/usr/local/bin/checklogins** speichern und in der *crontab* anstelle von *grep* aufrufen können, schafft hier Abhilfe. Vergessen Sie nicht, das Skript nach dem Anlegen mit **chmod +x** ausführbar zu machen!

```
#!/bin/sh
grep fail /var/log/auth.log > /var/log/failed.new
touch /var/log/failed.old
if test -s /var/log/failed.new; then
 diff -N /var/log/failed.old /var/log/failed.new
fi
mv /var/log/failed.new /var/log/failed.old
exit 0
```

Folie 106

Notizen:

A Datenträgerverwaltung und Dateisysteme

Unter Linux (Unix im allgemeinen) sind alle Dateien und Verzeichnisse in einem hierachisch strukturierten Dateisystem untergebracht. Dieses beginnt mit dem Wurzelverzeichnis **/** und wird durch Unterverzeichnisse, Unter-unterverzeichnisse etc. sortiert.

Das Trennzeichen zwischen Verzeichnissen und Dateien ist (wie auch im World-Wide-Web üblich) der "normale" Schrägstrich **/** (im Gegensatz zum unter DOS/Windows gebräuchlichen "Backslash" \).

Groß und Kleinschreibung müssen beachtet werden, d.h. "DATEI.TXT" ist nicht dasselbe wie "datei.txt".

Beispiel 1: **/home/knopper/Meine_Datei.txt**
Die Datei `Meine_Datei.txt` liegt im Heimatverzeichnis des Benutzers **knopper**, welches ein Unterverzeichnis von **home** ist, welches wiederum im Hauptverzeichnis des Linux-Dateisystems (**/**) liegt.

Beispiel 2: **/floppy/Backup.zip**
Die Datei **Backup.zip** liegt im Verzeichnis **floppy**, welches wiederum ein Unterverzeichnis von **mnt** im Hauptverzeichnis ist.

Um Datenträger wie Festplatten und Wechselmedien wie Disketten und CD-Roms in das hierarchische Unix-Dateisystem einzubinden, ist das **An-** (`mount`) und **Abmelden** (`umount`) des jeweiligen Dateisystems erforderlich.

Hintergrund: Im Unix-Dateisystem können die Inhalte von Datenträgern an **beliebiger Stelle** im Verzeichnisbaum `montiert` werden. Daher ist es z.B. möglich, größere Massenspeicher (Festplatten) für die Heimverzeichnisse der Benutzer zu montieren, ohne dass die Benutzer neue Programmpfade lernen müssen oder sich Dateien plötzlich an anderer Stelle befinden. **Laufwerksbuchstaben** wie unter DOS **gibt es nicht**.

Da dies als Konsequenz hat, dass das Unix-Dateisystem durch das **mount**-Kommando "umgebaut" werden kann, ist dieser Befehl (mit Ausnahmen, siehe Konfigurationsdatei **/etc/fstab**) nur vom Systemadministrator direkt ausführbar.

Beispiel: Mit **mount -t vfat /dev/fd0 /floppy** wird als Systemadministrator (als normaler Benutzer bei entsprechendem Eintrag in `/etc/fstab` einfach **mount /floppy**) der Inhalt einer Win95-formatierten Floppy-Diskette unter dem Verzeichnis `/floppy` sichtbar, und es können Dateien durch Kopieren in dieses Verzeichnis auf die Diskette übertragen werden.

WICHTIG: Erst nach dem **Abmelden** der Diskette mit **umount /floppy** werden alle noch auszuführenden physikalischen Schreibvorgänge auf der Diskette abgeschlossen, und die Diskette darf erst dann aus dem Laufwerk entfernt werden.

Bei CD-Roms ist es sogar unmöglich, die CD aus dem Laufwerk zu nehmen, solange sie noch angemeldet ("gemountet") ist. Solange noch Programme auf die CD zugreifen, kann diese auch nicht mit `umount` abgemeldet werden.

Zusatzinfo:

Welche Prozesse noch auf ein Dateisystem zugreifen, kann man mit dem Kommando **`/usr/sbin/fuser -v Verzeichnisname`** herausfinden, wobei das angegebene Verzeichnis dem **Mountpoint** entspricht, unter dem das Dateisystem mit `mount` zuvor montiert wurde.

Mit dem Kommando **`df`** (”Disk Filling”) ist eine ”Füllstandsanzeige” aller angemeldeten Dateisysteme abrufbar.

In der Konfigurationsdatei `/etc/fstab` werden die Dateisysteme und ihre **Einhängepunkte** (Mountpoint) sowie Optionen und Hinweise für den automatischen Überprüfungs und Anmeldevorgang beim Booten festgelegt.

Beispiel (`/etc/fstab`):

```
/dev/fd0   /floppy  vfat  noauto,user 0 0
/mnt/hdb1  /home        ext2  defaults    1 1
```

In der ersten Zeile wird das Diskettenlaufwerk `/dev/fd0` für normale Benutzer (`user`-Option) zum Montieren unter dem Verzeichnis `/floppy` freigegeben. Durch die `noauto`-Option wird jedoch verhindert, dass bereits beim Booten des Systems versucht wird, eine Diskette zu mounten. Diese Zeile macht es unter Linux möglich, dass mit einem einfachen **`mount /floppy`** die Diskette auch ohne Administratorrechte angemeldet werden kann. Dies wird z.B. auch vom KDE-Dateimanager genutzt, um durch Mausklick die für Benutzer freigegebenenMounts ausführen zu können.

In der zweiten Zeile wird dafür gesorgt, dass die erste Partition der zweiten Platte am ersten IDE-Kontroller, auf der sich ein Linux `ext2`-Dateisystem befindet, bereits beim Booten des Systems unter dem Verzeichnis `/home` eingebunden wird. Außerdem wird die Backup-Reihenfolge (die erste von den beiden Einsen am Ende der Zeile) festgelegt und die automatische Dateisystemprüfung (die zweite von den beiden Einsen am Ende der Zeile) aktiviert.

```
man 5 fstab
man 8 mount
```

Vorgehensweise beim Einbau von neuen Festplatten am Beispiel einer zweiten Platte am ersten IDE-Kontroller (**`/dev/hdb`**):

1.	**fdisk /dev/hdb**	Platte partitionieren
2.	**mkfs -t ext2 /dev/hdb1**	EXT2-Dateisystem auf erster Partition anlegen.
3.	**fsck -t ext2 /dev/hdb1**	Dateisystem überprüfen (optional)
4.	**mount -t ext2 /dev/hdb1 /mnt/neu**	Neues Dateisystem unter Verzeichnis `/mnt/neu` (muss vorher angelegt werden) montieren.
5.	**chmod a=rwx /mnt/neu**	Dateisystem für alle Benutzer schreibbar machen (ist dies gewünscht?).
6.	Partition `/dev/hdb1`, Mountpoint `/mnt/neu` und Dateisystemtyp sowie Optionen in `/etc/fstab` eintragen, damit auch nach einem Neustart das neue Dateisystem automatisch eingebunden wird.	

Tipp: Als "Umstiegserleichterung" für DOS-Benutzer gibt es ein Programmpaket mit dem Namen **mtools**, das ohne Zuhilfename des **mount**-Kommandos auf MSDOS-formatierte Disketten schreiben und von ihnen lesen kann. Die Aufrufkonventionen wurden hier absichtlich ähnlich den DOS-Kommandos **copy**, **del**, **format**, **ren** usw. gestaltet, insbesondere gibt es hier eine Option **a:**, die das Floppylaufwerk **/dev/fd0** unter dem von DOS her bekannten Namen anspricht.

Beispiele (man beachte v.a. die Notwendigkeit, bei der `a:`-Option Shell-Jokerzeichen zu deaktivieren):

mdir a:	Listet den Inhalt des MSDOS-Dateisystems auf /dev/fd0.
mcopy *.txt a:	Kopiert alle `*.txt`-Dateien auf die Floppy `/dev/fd0`.
mcopy a:*.txt .	Kopiert alle `*.txt`-Dateien von der Floppy ins aktuelle Verzeichnis.
mdel a:*	Löscht alle Dateien im Hauptverzeichnis der Floppy.
mformat a:	Legt ein neues MSDOS-Dateisystem auf `/dev/fd0` an.

Hinweis: Wenn man sich die Speicherauslastung unter Linux ansieht (**xosview** oder Kommando **free**) stellt man fest, dass Linux ungenutzten Speicher immer fast komplett als Dateisystempuffer im RAM zu nutzen versucht, was die Arbeitsgeschwindigkeit beim Lesen und Schreiben auf Datenträger drastisch erhöht und die mechanische Belastung (z.B. durch Kopfbewegungen) senkt. Dieser dynamische Festplattencache wird sofort freigegeben, wenn eine Anwendung Hauptspeicher benötigt. Mit **mount** eingebundene Datenträger werden automatisch mit in dieses dynamische Caching übernommen, außer, wenn die Option **-o sync** bei `mount` verwendet wird.

Beim Mounten von Netzwerk-Dateisystemen (NFS) wird anstelle der Gerätedatei die Rechneradresse des Servers und das dort freigegebene Dateisystem angegeben, z.B. mit:

mount 192.168.140.10:/usr/local /usr/local

(**man nfs**)

B Shell – Umgebungsvariablen, Aliase & Co.

Den meisten Shells unter Unix gemeinsam sind die folgenden Umgebungsvariablen. Diese sind i.d.R. nach der Anmeldung (Einloggen) schon auf sinnvolle Werte gesetzt und bestimmen das Verhalten vieler Programme und Shell-interner Kommandos.

Variable	Bedeutung
`$HOME`	Heimatverzeichnis des Benutzers, in das bei `cd` (ohne Angabe von Parametern) und beim Einloggen gewechselt wird.
`$PATH`	Suchpfad für Programme; durch **`:`** getrennte Liste von Verzeichnissen.
`$TERM`	Typ der aktuellen Textkonsole, wird beispielsweise von **`vi`** und bei der Farbdarstellung des GNU-**`ls`**-Kommandos ausgewertet.
`$SHELL`	Name der aktuellen Shell.
`$USER`	Benutzerkennung.

Der Benutzer kann diese und alle anderen Umgebungsvariablen selbst setzen oder verändern. Hierbei unterscheiden sich jedoch die Bourne-Shell-kompatiblen (**`sh`**, **`ksh`**, **`bash`**) von den C-Shell-kompatiblen (**`csh`**, **`tcsh`**) Shells in der Syntax.

Beispiel 1: Setzen des Kommandosuchpfades auf
`/bin:/sbin:/usr/bin:/usr/sbin:/usr/X11R6/bin:/opt/kde/bin:.`

`bash`:	`export PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/X11R6/bin:."`
`tcsh`:	`setenv PATH "/bin:/sbin:/usr/bin:/usr/sbin:/usr/X11R6/bin:."`

Beispiel 2: Setzen des Prompt auf **User@Rechner[aktuelles Verzeichnis]**.

`bash`:	`export PS1="\u@\h[\w]\\$ "`
`tcsh`:	`set prompt="%n@%m[%~] "`

Merke: Beim Zugriff auf eine Variable (`echo $PATH`) wird der Variablen ein **$** vorangestellt, beim Setzen der Variablen jedoch nicht!

Welche speziellen Umgebungsvariablen welche Wirkung haben, und die genaue Definition der Inhalte von Shell-Variablen, ist in den sehr ausführlichen **`man`**-Pages der entsprechenden Shell nachzulesen.

Um sich für häufig benutzte Kommandosequenzen oder Optionen Tipparbeit zu sparen, kann mit Hilfe von **Aliasen** eine Abkürzung für ein Kommando oder eine Kommandosequenz definiert werden.

Beispiele:

bash:	`alias dir='ls -la'`
tcsh:	`alias dir 'ls -la'`
bash:	`alias del='rm -i'`
tcsh:	`alias del 'rm -i'`
bash:	`alias suche='find / -name'`
tcsh:	`alias suche 'find / -name'`

bash und **(t)csh** lesen vor dem Start traditionell (per Option oder beim Übersetzen aus dem Quelltext konfigurierbar) folgende Dateien ein:

bash	
/etc/profile	systemweit, beim Login auf der Textkonsole
/etc/bashrc	systemweit, bei jedem Aufruf der Shell
$HOME/.profile	User-spezifisch, beim Login auf der Textkonsole
$HOME/.bashrc	User-spezifisch, bei jedem Aufruf der Shell
tcsh	
/etc/csh.login	systemweit, beim Login auf der Textkonsole
/etc/csh.cshrc	systemweit, bei jedem Aufruf der Shell
$HOME/.login	User-spezifisch, beim Login auf der Textkonsole
$HOME/.cshrc	User-spezifisch, bei jedem Aufruf der Shell

Bei den meisten Unix-Derivaten kann sich der Benutzer seine Standard-Shell (Login-Shell) frei aussuchen, und mit Hilfe des **chsh**-Kommandos selber setzen.

Mit Hilfe der Shell als Programmiersprache in sog. **Shell-Scripten** lassen sich auch komplexe, häufig wiederkehrende Aufgaben vereinfachen.

Beispiel: Mehrmaliges Kopieren einer Daten-CD.

```
#!/bin/sh

antwort="j"
while [ "$antwort" != "n" ]
do
 cdrecord -v dev=0,1,0 speed=4 -isosize /dev/cdrom
 echo "Bitte n^^e4chste CD in das CD-Rom Laufwerk, einen neuen Rohling"
 echo "in den CD-Brenner einlegen und <ENTER> dr^^fccken."
 echo -n "Eingabe ('n' f^^fcr Abbruch) > "
 read antwort
done
```

Die meisten Subsysteme (Netzwerk, X-Window Sessions, NFS, ...) unter Unix werden mit Hilfe von Shell-Scripten gestartet.[3]

[3] Siehe Abschnitt 2 **init**.

C Tipps & Tricks für die Shell

Die folgenden Kontrollsequenzen sind Standard auf den meisten Unix-Installationen und können mit dem Kommando `stty` verändert werden.

Beispiele: `stty -a` (zeigt aktuelle Terminal-Einstellungen an)
`stty erase '^?'` (ändert die Zeichen-Löschsequenz auf DEL)

Wichtige Kontrollsequenzen (^ steht für <Steuerung> bzw. <Strg>)	
TAB	Automatische Kommando/Dateinamenergänzung
^C	Abbruch des laufenden Programms (cancel)
^Z	Anhalten des laufenden Programms (suspend)
^U	Löschen der Zeile
^H	Löschen des Zeichens vor dem Cursor (erase)
^D	1. Löschen des Zeichens unter dem Cursor (delete) 2. Beenden der Shell oder der Eingabe (end of file)
^L	Löschen oder Auffrischen des Bildschirms
^S	Anhalten der Ausgabe (scroll lock)
^Q	Wiedereinschalten der Ausgabe

Job Control	
^Z	Anhalten des laufenden Programms (suspend)
&	Bewirkt, wenn es hinter einem Kommando steht, daß das Kommando als Hintergrundprozeß gestartet wird.
jobs	Anzeige der aus der aktuellen Shell gestarteten Jobs
bg %1	Läßt Job Nr. 1 im Hintergrund weiterlaufen (background)
fg %1	Holt Job Nr. 1 wieder in den Vordergrund (foreground)
stop %1	Hält Job Nr. 1 an (suspend)
kill %1	Terminiert (beendet) Job Nr. 1

History	
history	Gibt eine nummerierte Liste der zuletzt eingegebenen Befehle aus.
!komman	Führt diejenige Kommandozeile erneut aus, die mit `komman...` begann.
!10	Führt das Kommando Nr. 10 aus der `history`-Liste erneut aus.
!10 -l /etc	Führt das Kommando Nr. 10 aus der `history`-Liste mit neuen Parametern aus.

Hinweise: Wird die aktuelle Shell beendet, laufen nur die im Hintergrund gestarteten Jobs weiter, die nicht im Zustand „suspended“ sind, also keine Aus- und Eingaben mehr erwarten. Alle anderen, von dieser Shell abhängigen Jobs werden terminiert.

Ein häufiger Fehler ist es, ein Programm ohne die vorgesehenen Argumente (z.B. Dateinamen oder Optionen) zu starten.

Beispiel: `grep -i /etc/passwd`
In diesem Beispiel würde `grep` von der Standardeingabe lesen und alle eingetippten Zeilen, die den String `/etc/passwd` enthalten, ausgeben (was sicher nicht im Sinne des Aufrufers war, der vermutlich eher nach einem Namen in der Datei `/etc/passwd` suchen wollte, und diesen Namen hinter dem `-i`

anzugeben vergaß). Beendet werden solche interaktiv arbeitenden Fehlläufer mit `Control-C` oder `Control-D` am Zeilenanfang.

Shell-interne Kommandos (bash)	
Beispiel	**Bedeutung**
`alias ll='ls -l'`	Erzeugt die Abkürzung `ll` für das Kommando `ls -l`
`set`	Gibt aktuelle Einstellungen der Shell aus
`env`	Zeigt vererbbare Umgebungsvariablen an (environment)
`export VARIABLE="Wert"`	Setzt eine vererbbare Variable
`export PATH="$HOME/bin:$PATH"`	Stellt dem Suchpfad das zusätzliche Verzeichnis `Benutzerhomeverzeichnis/bin` voran
`which Kommando`	Zeigt an, unter welchem Verzeichnis im Suchpfad das Kommando installiert ist.

Quotes und Wildcards	
`\`	Verhindert das Auswerten des folgenden Zeichens (beispielsweise ein &, welches ansonsten Hintergrundprozesse starten würde) durch die Shell.
`" "`	Angabe von Namen, die Leer- oder Sonderzeichen enthalten. Shell-Variablen wie `$PATH` werden aber durch ihren Inhalt ersetzt.
`' '`	Angabe von Namen, die Leer- und Sonderzeichen inclusive $ enthalten.
`` ` ` ``	Das Kommando innerhalb der Backquotes wird ausgeführt und durch die Kommandoausgabe ersetzt.
`*`	Ersetzt ein oder mehrere beliebige Zeichen.
`?`	Ersetzt genau ein Zeichen.
`*.{tex,daten}`	Paßt auf alle Dateien mit der Endung `.tex` oder `.daten`
`[a-h]`	Ersetzt alle Buchstaben von `a` bis `h`

Trickreiche Beispiele und hilfreiche Programme	
`cd -`	wechselt zurück in das Verzeichnis, aus dem man beim letzten `cd`-Aufruf kam.
`rm '*'`	Löscht eine Datei mit dem Namen * aus dem aktuellen Verzeichnis.
`ls -l 'which passwd'`	Zeigt die Dateiberechtigungen des Programms `passwd` an, welches irgendwo im Pfad liegen kann.
`rm ./-i`	Löscht eine Datei mit dem Namen `-i` im aktuellen Verzeichnis.
`rm -i *`	Löscht alle Dateien im aktuellen Verzeichnis, fragt bei jeder Datei vorher nach.
`echo *`	erzeugt gleiche Ausgabe wie `ls .`, so können Sie das `ls`-Kommando ersetzen, wenn es nach einem Crash fehlt.
`ps aux | grep ^knopper`	Erzeugt eine ausführliche Prozeßliste aller Prozesse, die vom Benutzer `knopper` gestartet wurden.
`top`	Erzeugt eine sich selbst aktualisierende Liste der Prozesse im System.
`w`	Eines der kürzesten Kommandos, zeigt an, welche Benutzer eingeloggt sind und was sie gerade tun.

`cat /proc/{ioports,interrupts}`	Zeigt die Hardware-Adressen aller an das System angeschlossenen Karten und deren Interruptbelegung an.
`free`	Zeigt die Speicherauslastung an (Linux-spezifisch).
`killall -TERM xterm`	Alle laufenden `xterm`s werden terminiert. (Linux-spezifische Kommandosyntax)

D Kernel konfigurieren und installieren

Der **Kernel** ist die Schnittstelle zwischen Hardware und Anwenderprogramm unter Unix. Hierin befinden sich die meisten **Gerätetreiber**, die **virtuelle Speicherverwaltung**, der **Prozess-Scheduler**, die **Dateisysteme** und vieles andere mehr. Benutzerprozesse haben ausschließlich über den Kernel Zugriff auf Systemresourcen wie am Rechner angeschlossene Peripherie.

Unter **Linux** ist der Systemkern im Quelltext verfügbar, und kann vom Anwender verändert, neu konfiguriert, optimiert und (vom Administrator) installiert werden. Viele Bestandteile des Kernels, die nicht permanent gebraucht werden, können als zur Laufzeit nachladbare **Module** gebaut werden, und werden vom Kernel-Modullader bei Bedarf initialisiert oder entfernt. Bereits beim Systemstart benötigte Module wie Festplattencontroller-Treiber (IDE oder SCSI) sowie das Dateisystem der Root-Partition (meist ext2) sollten jedoch unbedingt fest in den Kernel eincompiliert sein.

Die Kernel-Versionsnummer setzt sich immer aus drei Zahlen zusammen:

1. Die Haupt-Versionsnummer.
2. Die Releasenummer. Ist diese geradzahlig, so handelt es sich um ein stabiles Kernelrelease. Ist sie ungeradzahlig, so handelt es sich um einen experimentellen Entwicklerkernel, in dem zwar viele neue Features gegenüber der stabilen Version implementiert sind, jedoch noch keine ausreichenden Tests zur Verifikation der Stabilität stattgefunden haben. Für den Produktionseinsatz sollten stets die stabilen Kernel (gerade Releasenummer) eingesetzt werden.
3. Der Patchlevel. In unregelmäßigen Abständen gibt es sowohl für die stabilen Anwenderkernel als auch für die Entwicklerkernel Updates, die neue Funktionalitäten oder Fehlerbereinigungen enthalten.

Die folgende Tabelle listet die Schritte auf, die zur Neukonfiguration und Installation des Linux-Kernels erforderlich sind. Die meisten sind durch das Haupt-**Makefile** des Kernel-Sourcebaums automatisiert, einige jedoch erfordern manuellen Eingriff, da z.B. das Bootmenü nach den Wünschen des Anwenders konfiguriert wird.

1.	Aktuelle Kernel-Quelltexte (**tar**-Archiv, **rpm**/**deb**-Paket o.ä.) beschaffen und in das Verzeichnis `/usr/src/linux` entpacken.
2.	In das Verzeichnis `/usr/src/linux` wechseln.
3.	Kernel-Konfigurationsmenü aufrufen mit `make menuconfig` (textorientiert) oder `make xconfig` (graphisch).
4.	Einstellen der gewünschten Konfigurationen.
5.	Abhängigkeiten generieren mit `make depend`.
6.	Komprimiertes Kernelimage generieren und Module übersetzen mit `make bzImage modules`.
7.	Module installieren mit `make modules_install`.
8.	Kernel in das Verzeichnis `/boot` kopieren, z.B. `cp arch/i386/boot/bzImage /boot/vmlinuz-neu`.
9.	Eintrag in `/etc/lilo.conf` z.B. mit `image=/boot/vmlinuz-neu` und `label=linux-neu`.
10.	`lilo` aufrufen, um den Master-Boot-Record neu zu schreiben.
11.	Reboot
12.	Falls das neue Kernel-Image nicht als Default in `/etc/lilo.conf` eingetragen wurde, muss es beim LILO-Bootprompt angegeben werden (hier: `linux-neu`).

E RPM – Der RedHat Package Manager

Die graphische Variante des Redhat Package Managers, die beim KDE verwendet wird, heißt `kpackage`. Um Pakete im System zu installieren oder zu entfernen, sollte `kdesu -c kpackage` aus der Shell oder dem KDE-Menü aufgerufen werden. Um sich lediglich die auf dem System installierten Softwarepakete in der RPM-Datenbank anzusehen, reichtg es jedoch, `kpackage` als normaler Benutzer aufzurufen. RPM-Paketdateien können mit Drag & Drop in das Hauptfenster von `kpackage` gezogen, dort angeschaut und installiert werden.

Hin und wieder werden Sie auch die originale, textorientierte Variante von RPM benötigen, vor allem, wenn Ihnen die graphische Version keine brauchbare Fehlerbeschreibung liefert, falls eine Installation einmal fehlschlagen sollte.

Beispiele	
`rpm -qiv paketname`	Gibt Informationen über das installierte Paket `paketname` aus (query information).
`rpm -qivp datei.rpm`	Gibt Informationen über den Inhalt der RPM-Datei `datei.rpm` aus.
`rpm -qivlp datei.rpm`	Gibt zusätzlich auch eine Liste der im Paket enthaltenen Dateien aus.
`rpm -qivf programm`	Gibt Informationen über das Paket aus, zu welchem das Programm `programm` gehört. (file)
`rpm -ivh datei.rpm`	Installiert ein neues Paket aus `datei.rpm`.
`rpm -Uvh datei.rpm`	Aktualisiert auf eine neue Version (Update).
`rpm -Fvh datei.rpm`	Aktualisiert auf eine neue Version, jedoch nur dann, wenn das angegebene Paket bereits in einer älteren Version installiert war.
`rpm -ev paketname`	Entfernt das angegebene Paket und alle dazugehörenden Dateien vollständig aus dem System.
`rpm --help`	Gibt eine kurze Hilfestellung zu allen Optionen von `rpm` aus.

F dpkg und apt – Softwareverwaltung unter Debian

Von **kpackage** gibt es ebenfalls eine Version speziell für Debian-Pakete. Um mit diesem GUI Pakete im System zu installieren oder zu entfernen, kann, wie bei der RPM-Version, `kdesu -c kpackage` aus der Shell oder dem KDE-Menü aufgerufen werden.

Die eigentlichen Programme zur Verwaltung von Softwarepaketen und Debian sind **dpkg** und **apt-get** bzw. **apt-cache**.

Ziel der Debian-Paketverwaltung ist es, Software-Installation/Update/Entfernen unter Berücksichtigung von Abhängigkeiten zwischen verschiedenen Programmpaketen zu vereinfachen, und so immer einen konsistenten Zustand der Software zu gewährleisten.

Installation von Paket-Dateien:
Redhat: **`rpm -i paketname.rpm`**
Debian: **`dpkg -i paketname.deb`**

Deinstallation installierter Softwarepakete:
Redhat: **`rpm -e paketname`** (paketname aus der Datenbank)
Debian: **`dpkg --purge paketname`** (paketname aus der Datenbank)
`dpkg --purge` ☞ Entfernen des Paketes inclusive aller Komponenten und Entfernen aller Konfigdateien+Doku.
`dpkg -r` ☞ Entfernen der Software, aber Behalten der (geänderten) Konfigdateien.

Was ist installiert auf meinem System?
Redhat: **`rpm -qa`** (Übersicht)
Debian: **`dpkg -l`** (Übersicht)

Welche Dateien sind in dem Paket vorhanden?
Redhat: **`rpm -qivlp paketname.rpm`**
Debian: **`dpkg --contents paketname.deb`**

Welche Dateien sind auf meinem Rechner installiert, aus diesem Paket?
Redhat: **`rpm -qivl paketname`** (Name aus der Datenbank)
Debian: **`dpkg -L paketname`** (Name aus der Datenbank)

Aus welchem Softwarepaket kommt die Datei *datei*?
Redhat: **`rpm -qivf datei`** (Voller Pfadname von *datei*)
Debian: **`dpkg -S datei`** (Voller Pfadname von *datei*)

Informationen über installiertes Paket
Redhat: **`rpm -qiv paketname`**
Debian: **`dpkg -s paketname`**

Debian-spezifisch: **`apt-get`** und **`apt-cache`**, die Netzwerk-Variante von **dpkg**

`apt-get`	Kann Software installieren (aus dem Netz oder von CD) mit Auflösung aller Abhängigkeiten, Software deinstallieren/updaten/downgraden.
`apt-cache`	Informationsabfrage mit Datenbank aller in den unter `/etc/apt/sources.list` aufgeführten Repositories vorhandenen Debian-Pakete

Beispiele:

1. Software-Datenbank anlegen
   (a) In `/etc/apt/sources.list` sind die verfügbaren Medien (auch FTP/http-Server) angegeben. Neue können mittels **`vi`** oder **`apt-cdrom add`** hinzugefügt werden.
   (b) Die Datenbank aller *verfügbaren* Pakete wird neu erzeugt mit **`apt-get update`**.
2. Software suchen, z.B. backup-Software:
   `apt-cache search backup`
3. Informationen über alle Versionen eines Pakets anzeigen, z.B. amanda.client:
   `apt-cache show amanda-client`
4. Software installieren:
   `apt-get install [-t unstable] paketname`
5. Software updaten:
   `apt-get install [-t unstable] paketname` (wie oben.)
6. Software löschen:
   `apt-get [--purge] remove paketname`
7. Erweiterte Benutzung:
   (a) **`apt-get install paketname/unstable`**
   ☞ installiert das Paket aus der ”unstable”-Sektion von Debian.
   (b) **`apt-get install paketname=1.0.0-1`**
   ☞ up-/downgrade auf die angegebene Version.
   (c) **`apt-get install --reinstall paketname`** ☞ Führt die Installation des Paketes erneut durch (z.B. dann sinnvoll, wenn man versehentlich eine Komponente des Paketes gelöscht hat, das Paket sich aber nicht komplett deinstallieren und neuinstallieren lässt).
   (d) **`apt-get [--download-only] dist-upgrade`**
   ☞ Installiert die neuen Versionen ALLER bereits installierten Pakete. Mit **`--download-only`** werden zunächst nur die neuen Pakete nach `/var/cache/apt/archives/` kopiert zwecks späterer Installation.
   (e) **`apt-get -s dist-upgrade`**
   ☞ Zeigt an, was beim dist-Upgrade passieren *würde*.

`dselect` ist ein (Text-)GUI für **`dpkg`**; **`aptitude`** hingegen eins für **`apt-get`**. Beide unterstützen mit Menüs eine Paketauswahl und Installation/Deinstallation.

G Kurz-HOWTO: dhcpd einrichten

1. DHCP-Paket installieren (yast2 bei SuSE, `apt-get install dhcp` bei Debian)
2. Die Beispiel-dhcpd.conf aus `/usr/share/doc/packages/dhcp/dhcpd.conf` (SuSE) bzw. `/usr/share/doc/dhcp*/examples/dhcpd.conf` (Debian) nach `/etc/` kopieren und auf das eigene Netz anpassen, Beispiel:

```
ddns-update-style ad-hoc;
option domain-name "mydomain.net";
option domain-name-servers 192.168.96.101, 194.25.2.129, 194.162.162.194;

default-lease-time 600;
max-lease-time 7200;

# Dynamische Adressvergabe 200-254:
subnet 192.168.96.0 netmask 255.255.255.0 {
  range 192.168.96.200 192.168.96.254;
  option routers 192.168.96.1;
}

# Feste Adresse f^^fcr einen bestimmten Rechner (MAC-Adresse):
host pizza {
 hardware ethernet 0:0:c0:5d:bd:95;
 fixed-address 192.168.96.199;
 option routers 192.168.96.1;
}
```

3. `/etc/init.d/dhcpd (re)start`

H Kurz-HOWTO: DNS einrichten

1. `bind` (8 oder 9) installieren mit yast2 (SuSE) bzw. `apt-get install bind9` (Debian)
2. `/etc/named.conf` anpassen. Falls eigene DNS-Zonen gewünscht, ist die empfohlene Methode:
   (a) Mit webmin 2 neue Master-Zonefiles anlegen (eins für Forward, eins für Reverse-Lookup)
   (b) Mit webmin ein paar Hosts eintragen
   (c) Mit vi Zeilen in `/var/named/masterzonefile.hosts` bzw. `masterzonefile.rev` editieren/kopieren für die gewünschte Rechnerzahl
3. `/etc/init.d/named (re)start`
4. Testen mit `dig test.rechner.name @127.0.0.1`
5. Falls Fehler: In `/var/log/messages` nachschauen, was beim Start von `named` schiefgegangen ist, Fehler korrigieren.
6. Falls alles OK: Eintragen von

   ```
   nameserver 127.0.0.1
   ```

   in die `/etc/resolv.conf` des eigenen Rechners, die Clients im Intranet tragen entsprechend die IP-Adresse des neuen Nameservers in ihre `/etc/resolv.conf` ein.

I Kurz-HOWTO: Einrichten von `/etc/hosts.allow`

Beispiele für `/etc/hosts.allow`:

```
# SSH erlauben f^^fcr alle
sshd ssh : ALL : ALLOW

# POP3 und IMAP erlauben f^^fcr localhost
pop3 ipop3d imap imap2d : LOCAL 127.0.0.1 : ALLOW

# PORTMAP (NFS und andere) Zugriff von au^^dfen sperren
portmap : 127.0.0.1 LOCAL : ALLOW
portmap : ALL@ALL : DENY

# Und alle anderen Versuche ergeben eine Fehlermeldung, mit 10 Sekunden
# Wartezeit und Protokollierung in /var/log/messages (Portscannerfalle)
ALL : ALL@ALL : twist echo "%c\: You are not allowed to connect. \
                                                Go away!^M" ; sleep 10
```

J Kurz-HOWTO: LDAP-Authentifikationsdienst einrichten

1. LDAP-Datenbank installieren (auf dem Server): `rpm -Uvh openldap*rpm` (oder unter Debian mit `apt-get install openldap2`)
   SuSE Special: In `/etc/rc.config` `START_LDAP=yes`
   Dann `/etc/init.d/ldap` start.

2. LDAP-Server- (`/etc/openldap/slapd.conf`) und Client-Konfiguration (`/etc/openldap/ldap.conf`) einrichten.

3. Das schwierigste: Benutzerdaten von `/etc/passwd`+`/etc/shadow`+`/etc/group` ins LDIF-Format migrieren (diverse Skripte hierfür gibt es im Netz, aber das Ausgabeformat muss meistens überarbeitet werden). Das generierte .LDIF-File wird mit `ldapadd` in die LDAP-Datenbank übernommen.

   Alternativ: LDAP-Benutzerdaten komplett neu anlegen mit graphischem Frontend (z.B. `gq`)

   Die Directory-Struktur der LDAP-Daten für die Benutzerauthentifizierung ist durch den verwendeten Client festgelegt.

   Beispiel: Konvertierte Daten liegen in etcpasswd.ldif vor.

   ```
   ldapadd -x -D "cn=Manager,dc=my-domain,dc=com" -w geheimes_passwort
           -f etcpasswd.ldif
   ```

4. `pam_ldap` oder/und `nss_ldap` installieren. `pam_ldap` authentifiziert über das Pluggable Authentication Module System, das auf den meisten Distributionen Standard ist (Konfigurations-Dateien in `/etc/pam.d/*`), `nss_ldap` hingegen ist ein Plugin für die C-Systembibliothek, das per
   `/etc/nsswitch.conf` eine für alle mit der `libc` gebundenen Programme funktionierende Abfrage der entsprechenden Dienste (lokale Dateien, NIS Server, LDAP Server) ermöglicht.

Eine gute Beschreibung von LDAP, auch im Zusammenhang mit SAMBA als LDAP-authentifizierender PDC für Windows-Clients, ist unter
`http://www.skills-1st.co.uk/papers/security-with-ldap-jan-2002/security-with-ldap.html`
zu finden. Unter `http://www.padl.com/OSS/MigrationTools.html` sind einige Konvertierungs-Tools zu finden, die die Migration von Benutzerdaten ins LDAP-Format erleichtern sollen.

K Kurz-HOWTO: Forwarding/Masquerading einrichten

1. `iptables` (Frontend zur Verwaltung von Kernel-Filterregeln) installieren.
2. Netzwerkkarte(n) konfigurieren, Routen setzen (`ifconfig`, `route` und Eintrag in `/etc/network/interfaces` (Debian) bzw. `/etc/rc.config`, `/etc/route.conf` bei SuSE).
3. Forwarding einschalten: `{echo 1 > /proc/sys/net/ipv4/ip_forward`
   ACHTUNG: Ab diesem Moment wird IP-Forwarding in alle Richtungen auf allen Netzwerkkarten erlaubt, wenn nicht mit `iptables` Filterregeln gesetzt sind!
4. Masquerading einschalten (IP-Address-Translation in einer Richtung):
   ```
   iptables -A POSTROUTING -t nat -j MASQUERADE
   ```
   (besser: Masquerading nur für bestimmte Netze und Netzwerkkarten erlauben, s.a. iptables-HOWTO!)

Ab jetzt kann die ins interne Netz zeigende IP-Adresse des Rechners als Defaultgateway bei den Clients verwendet werden.

L Kurz-HOWTO: SAMBA einrichten (Server & Client

1. In der `/etc/samba/smb.conf` finden Sie einen Abschnitt `[homes]`, der für alle Benutzerverzeichnisse zuständig ist. Die Benutzerverzeichnisse werden dann als Windows-Share `\\servername\benutzername` exportiert.

2. Damit auch NT4 und Win2K die entsprechenden Benutzer authentifizieren kann, muss für jeden Benutzer auf dem Server noch ein NT-Passwort gesetzt werden:

   ```
   smbpasswd -a demo
   ```
   (`-a` zum erstmaligen Einrichten, ohne `-a` zum Ändern)

3. Unter Linux können per Samba oder Windows-Filesharing freigegebene Verzeichnisse mit

   ```
   smbmount //rechnername/benutzername /Zielverzeichnis \
                  -o username=Benutzer,password=Passwort
   ```

   gemountet werden. Soll für einen bestimmten Benutzer der Zugriff möglich sein, so kann noch `uid=Kennung` in den Optionen angegeben werden.

Samba-FS bzw. SMB unterstützt nicht die Unix-spezifischen Attribute wie erweiterte Dateirechte (`s`, `t`, Gruppenrechte), Devices, Pipes und ähnliches, und ist daher nicht als ”echtes” Unix-Dateisystem brauchbar, sondern vorwiegend zum einfachen Datenaustausch nützlich.

Im Gegensatz zu NFS authentifiziert Samba benutzerspezifisch und kann, sofern von Client und Server unterstützt, SSL/TLS zur verschlüsselten Übertragung von Passwörtern und Daten verwenden.

M Kurz-HOWTO: Swap-FILE einrichten

Problem: Es wird mehr Swap benötigt, es steht aber keine separate Partition mehr für mkswap zur Verfügung.

Verfahren (screen-logs):

1. Anlegen einer Datei mit 100MB Größe auf /

```
server1:/home/knopper # dd if=/dev/zero of=/swap bs=1000k count=100
100+0 Records ein
100+0 Records aus
```

2. Datei mit Swapsignatur versehen

```
server1:/home/knopper # mkswap /swap
Swapbereich Version 1 mit der Grˆˆf6ˆˆdfe 102395904 Bytes wird angelegt
```

3. Kernel in Datei swappen lassen

```
server1:/home/knopper # swapon /swap
```

4. Infos abrufen

```
server1:/home/knopper # cat /proc/swaps
Filename                Type            Size    Used    Priority
/dev/hda2               partition       1028152 58100   42
/swap                   file            99992   0       -1
```

Eine Zeile der Form

```
/swap   none   swap  defaults 0 0
```

in `/etc/fstab` sorgt schließlich dafür, dass die Swap-Datei nach dem nächsten Reboot automatisch wieder verwendet wird.

N Links

http://www.linuxportal.de/ Linux- und OpenSource-bezogenes Newsdiensteportal, konfigurierbar.

http://freshmeat.net/ Die Informationsquelle für Neuigkeiten im Linux-Softwarebereich, stündlich aktualisiert.

http://www.rpmfind.org/ Datenbank für RPM-Softwarepakete mit Kurzbeschreibungen.

http://packages.debian.org/ Datenbank für Debian-Softwarepakete mit Kurzbeschreibungen.

http://www.kernelnotes.org/ Kernel-Aktualisierungen, Patches, Treiber.

http://www.webmin.com/ WEBMIN Homepage.

http://www.samba.org/ SAMBA Homepage.