Zurück

LPIC Level1 Study-Guides

Das Linux Professional Institiute (LPI) hat es sich zur Aufgabe gemacht, eine distributionunabhängige, aussagekräftige Zertifizierung zum Thema Linux anzubieten. Diese Zertifizierung findet in mehreren Stufen statt. Die erste Stufe dieser Zertifizierung wird durch Ablegen von zwei Prüfungen erreicht, die aufeinander aufbauend die Grundlagen des Umgangs mit, und der Verwaltung von Linux behandeln.

Wenn diese beiden Prüfungen (LPI 101 und LPI 102) abgelegt wurden, erhält man den Titel LPIC-Level 1. Später können dann weitere Levels erreicht werden, die aber eine wesentlich höhere Wissensanforderung stellen.

Diese Seiten sollen eine Vorbereitung auf die beiden Prüfungen für die Zertifizierung Level 1 zur Verfügung stellen und werden Schritt für Schritt jedes geforderte Thema durchsprechen und die dort notwendigen Techniken behandeln. Allerdings sollte vor der Durcharbeitung dieser Scripts bereits ein Basiswissen von Linux existieren, denn schon die erste Prüfung dringt in Tiefen vor, die nicht trivial sind.

Im Frühjahr 2002 wurden die Prüfungen der ersten Stufe einer Revision unterzogen, um sie der aktuellen Entwicklung von Linux anzupassen. Dabei haben sich nicht nur inhaltlich einige Themen geändert, sondern auch die Reihenfolge der Themenblöcke wurde verändert. So finden sich heute einige Themenblöcke, die früher in der LPI102 Prüfung abgefragt wurden in der ersten Prüfung (LPI101) und umgekehrt. Ich habe diese Umstellung zum Anlaß genommen, diese Seiten komplett zu überarbeiten und an die neuen Gegebenheiten anzupassen. Ab Mitte August 2002 sind die neuen Prüfungen gültig, ab November werden nur noch die neuen Prüfungen angeboten. Wer also jetzt neu anfängt zu lernen, sollte sich gleich an diese neue Struktur halten.

- Die Vorbereitung auf die Prüfung LPI 101
- Die Vorbereitung auf die Prüfung LPI 102

[Zurück zur Startseite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---


Are you certifiable?
Linux
Certification
LPI
The Linux Professional Institute

Zurück

LPI 101 Study-Guide

Das LPI 101 Examen ist die erste von zwei Prüfungen, die abgelegt werden müssen, um das LPIC Level 1 Zertifikat zu erhalten. Diese Seiten sollen die einzelnen Bereiche dieser Prüfung der Reihe nach durchgehen und so dabei helfen, die Prüfung zu bestehen. Sie sind keine Sammlung von Fragen, denn die Fragen sind nicht öffentlich zugänglich. Das Ziel der Zertifizierung ist ja auch nicht die Förderung auswendig gelernter Antworten, sondern die Überprüfung von Wissen!

Analog dazu verstehe ich diesen Kurs auch nicht als stures Training auf die Prüfung hin, sondern als den Versuch, Wissen zu vermitteln. Allein das Durcharbeiten der hier vorliegenden Seiten ist sicherlich nicht ausreichend, um die LPIC 101 Prüfung zu bestehen. Ich rate dazu, wirklich mit Linux zu arbeiten und dann diese Seiten durchzuarbeiten. Dann sollte die Zertifizierung kein wesentliches Problem darstellen.

Eine der besten Informationsquellen für die Prüfungsvorbereitung ist die Seite von LPI selbst, wo es auch deutsche Versionen der Informationen gibt. Der Banner unten auf allen Seiten führt Sie automatisch dort hin.

Der folgende Kurs richtet sich genau nach den Prüfungszielen, die LPI für dieses Examen definiert hat. Seit Mitte August 2002 gibt es die zweite Revision der LPIC Level 1 Prüfungen. Ich habe mich in diesem Kurs an die zweite Version gehalten, er sollte also einigermaßen aktuell sein. Schritt für Schritt werden alle notwendigen Bereiche durchgesprochen um so die Sicherheit für die Prüfung zu bekommen.

Die Zielsetzung der LPI 101 Prüfung besteht aus fünf Hauptgruppen, die jeweils mehrere fest definierte Ziele beinhalten. Jedes dieser Ziele besitzt eine Bewertung, die in zwar nicht aussagt, wieviele Fragen aus dieser Gruppe in der Prüfung vorkommen, aber Rückschlüsse auf die Häufigkeit zulassen. Niedrige Bewertung meint wenig Fragen, hohe Bewertung meint viele... Im Augenblick sind die Bewertungen noch nicht festgelegt, da sich die Prüfungen noch im Beta-Status befinden. Sobald sie zur Verfügung stehen, werden sie hier auch angegeben.

Der ganze Study-Guide steht auf der Download-Seite auch als tgz-Datei zum Download zur Verfügung.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

1.101

Hardware und Systemarchitektur

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

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

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

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

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

1.102

Installation von Linux und Paketmanagement

Auch dieses Thema war in der letzten Version der Level-1 Zertifizierung im zweiten Teil gelegen. Es geht prinzipiell um die Fragen, die sich vor, während und nach der Installation von Linux stellen, sowohl was die Fragen von Partitionsaufteilungen betrifft, als auch Einrichtung von Bootmanagern, Verwaltung der Libraries und Paketverwaltung.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.103

GNU und Unix Kommandos

Dieses Thema enthält acht definierte Ausbildungsziele, die sich alle um die grundlegenden Unix/Linux Kommandos drehen. Wichtige Elemente sind hier also sowohl die BASH, als auch die Kommandos zum Kopieren, Verschieben, Löschen von Dateien und Verzeichnissen. Dazu kommen die grundlegenden Unix-Texttools. Neu hinzugekommen ist das Thema vi, das früher im zweiten Teil behandelt wurde.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

Zurück

1.104

Gerätedateien, Linux Dateisysteme, Filesystem Hierarchy Standard

---

In diesem Abschnitt geht es um alles, was mit dem Umgang mit Dateisystemen zu tun hat. Dazu zählen die verschiedenen Befehle zum Anlegen, Reparieren und Überprüfen von Dateisystemen selbst genauso, wie die Befehle zum Modifizieren der Dateien.

Viele wichtige Fragen, die Dateisysteme betreffen, können nur vernünftig beantwortet werden, wenn die interne Architektur von Dateisystemen klar ist. Aus diesem Grund habe ich hier noch ein paar Zusatzinformationen zu diesem Thema erstellt, die nicht direkten Bezug auf einzelne Prüfungsziele von LPI nehmen:

- Das INode Prinzip von Unix Dateisystemen
- Details zum EXT2 Dateisystem
- Journaling Dateisysteme

- Erzeugen von Partitionen und Dateisystemen
- Erhaltung der Dateisystemintegrität
- Kontrolle des Ein- und Aushängen von Dateisystemen
- Verwalten von Diskquotas
- Zugriffskontrolle auf Dateien mittels Zugriffsrechten
- Verwaltung von Dateieigentum
- Erzeugen und Ändern von harten und symbolischen Links
- Auffinden von Systemdateien und Platzieren von Dateien an den korrekten Ort

---

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.110

X

Alles was mit der graphischen Oberfläche (dem X-Window-System) zu tun hat fällt unter dieses Kapitel.

- Installation und Konfiguration von XFree86
- Einrichten eines Display Managers
- Installation und Anpassung einer Window Manager Umgebung

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

LPI 102 Study-Guide

Das LPI 102 Examen ist die zweite Prüfung, die abgelegt werden muß, um das LPIC Level 1 Zertifikat zu erhalten. Erst wenn diese Prüfung erfolgreich abgelegt wurde, kann man sich mit dem Titel LPIC-1 schmücken.

Diese Seiten sollen die einzelnen Bereiche dieser Prüfung der Reihe nach durchgehen und so dabei helfen, die Prüfung zu bestehen. Sie sind keine Sammlung von Fragen, denn die Fragen sind nicht öffentlich zugänglich. Das Ziel der Zertifizierung ist ja auch nicht die Förderung auswendig gelernter Antworten, sondern die Überprüfung von Wissen!

Wie schon beim ersten Teil (101) ist die Praxis der wesentliche Faktor um die Prüfung zu schaffen. Der hier besprochene zweite Teil ist um einiges umfangreicher und schwieriger als der erste Teil, aber darin liegt ja auch die Herausforderung...

Der folgende Kurs richtet sich genau nach den Prüfungszielen, die LPI für dieses Examen definiert hat. Seit Mitte August 2002 gibt es die zweite Revision der LPIC Level 1 Prüfungen. Ich habe mich in diesem Kurs an die zweite Version gehalten, er sollte also aktuell sein. Schritt für Schritt werden alle notwendigen Bereiche durchgesprochen um so die Sicherheit für die Prüfung zu bekommen.

Die Zielsetzung der LPI 102 Prüfung besteht aus neun Hauptgruppen, die jeweils mehrere fest definierte Ziele beinhalten. Jedes dieser Ziele besitzt eine Bewertung, die in zwar nicht aussagt, wieviele Fragen aus dieser Gruppe in der Prüfung vorkommen, aber Rückschlüsse auf die Häufigkeit zulassen. Niedrige Bewertung meint wenig Fragen, hohe Bewertung meint viele... Im Augenblick sind die Bewertungen noch nicht festgelegt, da sich die Prüfungen noch im Beta-Status befinden. Sobald sie zur Verfügung stehen, werden sie hier auch angegeben.

Der ganze Study-Guide steht auf der Download-Seite auch als tgz-Datei zum Download zur Verfügung.

- 1.112 - Netzwerk-Grundlagen
- 1.113 - Netzwerkdienste
- 1.114 - Sicherheit

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.105

Kernel

Dieses Thema enthält zwei Kapitel, die sich um den Umgang mit dem Kernel selbst drehen. Dabei ist der Umgang mit Kernelmodulen während der Laufzeit und das Erstellen eines eigenen Kernels das Lernziel. Weitergehende Techniken, wie etwa das Patchen eines Kernels wird erst für LPIC Level 2 interessant.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.106

Booten, Initialisierung, Shutdown, Runlevels

Alles, was mit dem Booten und dem kontrollierten Herunterfahren des Systems zu tun hat, wird in diesen zwei Kapiteln besprochen.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.107

Drucken

Das Drucken unter Linux ist einerseits eine nicht ganz einfache Technik, weil es - im Gegensatz zu Windows - nicht für jeden Drucker geeignete Treiber gibt, aber andererseits eine Technik, die sehr symptomatisch für andere Unix/Linux Spoolsysteme ist. Die drei Kapitel dieses Abschnitts drehen sich genau darum.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.108

Dokumentation

Linux ist ein sehr komplexes System, das noch dazu nicht aus einer Hand stammt. Ein Umgang mit den verschiedenen Dokumentationen für System und Anwendungen ist daher eine fundamentale Notwendigkeit. In der Frühzeit von Linux wurde es einmal als *größtes Textadventure der Welt* bezeichnet, was in mancherlei Hinsicht noch immer den Nagel auf den Kopf trifft. In der Regel sind aber tatsächlich alle notwendigen Dokumentationen entweder lokal oder über das Internet erreichbar. Um den Umgang mit den verschiedenen Dokumentationsarten drehen sich die folgenden drei Kapitel dieses Abschnitts.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

Zurück

1.109

Shells, Scripting, Programmierung und Compilieren

Die folgenden zwei Kapitel behandeln die grundlegende Fähigkeit der Shell, programmiert zu werden. Die wirklich tiefergehenden Fähigkeiten zu diesem Thema werden erst in Level 2 (lpi201 und lpi202) besprochen und notwendig.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.111

Administrative Tätigkeiten

In diesem Abschnitt werden verschiedene Tätigkeiten der Systemverwaltung behandelt, die ansonsten nicht in einen anderen Abschnitt passen.

- Verwalten von Benutzer- und Gruppenkonten und verwandte Systemdateien
- Einstellen von Benutzer- und Systemumgebungsvariablen
- Konfigurieren und Nutzen der Systemlogs im Rahmen der administrativen und Sicherheitsanforderungen
- Automatisieren von Systemadministrationsaufgaben durch Planen von zukünftig laufenden Jobs
- Aufrechterhaltung einer effektiven Datensicherungsstrategie
- Verwalten der Systemzeit

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.112

Netzwerk-Grundlagen

Die drei folgenden Kapitel gehen auf Netzwerk-Grundlagen ein. Als Unix-System arbeitet Linux grundsätzlich mit TCP/IP, ein fundiertes Wissen über dieses System ist also von Nöten, wenn Linux ans Netz gehängt werden soll.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.113

Netzwerkdienste

Linux wird in der Praxis hauptsächlich als Serversystem eingesetzt, also als System, das Netzwerkdienste zur Verfügung stellen soll. Grundlegende Dienste werden in den folgenden sechs Kapiteln behandelt. Auch für diesen Abschnitt gilt, dass eine weiterführende Behandlung im nächsten Prüfungslevel erfolgt.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---


Are you certifiable?
Linux
Certification
LPI
The Linux Professional Institute

1.114

Sicherheit

Die letzten drei Kapitel der LPI102 Vorbereitung drehen sich um Fragen der Sicherheit.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

Hardwareparameter

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

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

Funktionsweise von Mainboards

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

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

Hardwareadressen

Aus der Sicht des Prozessors sind auch Zugriffe auf alle mögliche Hardware nur Speicherzugriffe. Das bedeutet, daß auch die andere Hardware des Computers entsprechende Adressen besitzen muß. Will der Prozessor etwa ein Byte auf die serielle Schnittstelle schreiben, so ist das aus seiner Sicht kein Unterschied zu einem Schreibvorgang in den Speicher. Also benötigt er die genaue Adresse der seriellen Schnittstelle. Er kann diese Adresse auf den Adressbus legen, dadurch öffnet sich ein Kanal des Datenbusses vom Prozessor zu dem Baustein, der die serielle Schnittstelle steuert. Der Prozessor schreibt das Byte jetzt auf den Datenbus, auf der anderen Seite empfängt das entsprechende Bauteil jetzt das Byte und übernimmt die tatsächliche Ausgabe auf die Schnittstelle.

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

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

Das Interrupt-System

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

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

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

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

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

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

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

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

Das DMA-System

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

Um dieses Feature zu ermöglichen sind sogenannte DMA-Kanäle vorgesehen, von denen der Rechner wiederum nicht besonders viele anbietet. Ein Standard-PC bietet acht DMA-Kanäle an, von denen einer stets belegt ist (DMA4).

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

Einstellmöglichkeiten der Hardwareparameter

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

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

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

Feste Hardware auf dem Mainboard

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

Sehr alte ISA-Erweiterungskarten

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

Etwas modernere ISA-Erweiterungskarten

Etwas modernere Karten besitzen kleine Festspeicher (EEPROMS, Flash-EPROMS) die softwaremäßig eingestellt werden können. Dazu benötigt man ein spezielles kleines Programm, das der Hersteller solcher Karten mitliefert. Meist ist dieses Programm auf einer

Diskette beigelegt und muß unter DOS gestartet werden. Es sind dann alle Einstellungen (IO-Adressen, IRQs und DMA-Kanäle) softwaremäßig einstellbar.

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

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

ISA Plug And Play Karten

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

PCI Erweiterungskarten

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

Speicherbereiche

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

Das /proc-Dateisystem

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

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

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

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

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

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

Die Hardwareparameter im /proc-Verzeichnis

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

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

`/proc/interrupts`

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

`/proc/ioports`

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

```
0170-0177 : ide1
```

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

`/proc/dma`

Hier finden wir die Angabe der benutzten DMA-Kanäle zusammen mit der Information, wer sie benutzt.

`/proc/iomem`

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

Andere Hardwareinformationen im /proc-Verzeichnis

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

`/proc/cpuinfo`

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

`/proc/devices`

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

`/proc/partitions`

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

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

(alte Darstellung) oder

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

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

`/proc/pci`

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

Kernel- und Softwareinformationen im /proc-Verzeichnis

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

`/proc/cmdline`

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

`/proc/filesystems`

Eine Liste aller Dateisystemtypen, die der Kernel im Augenblick kennt.

`/proc/meminfo`

Informationen über die gegenwärtige Auslastung des Arbeitsspeichers.

`/proc/modules`

Eine Liste der aktuell geladenen Kernel-Module.

`/proc/mounts`

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

`/proc/version`

Die aktuelle Kernelversion.

Weitere wichtige Unterverzeichnisse in /proc

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

`/proc/bus`

Informationen über die gefundenen Bussysteme (PCI, USB, ...)

`/proc/ide`

Informationen über IDE-Geräte und -Schnittstellen.

`/proc/scsi`

Informationen über SCSI-Geräte und Schnittstellen.

`/proc/net`

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

Aktive Veränderungsmöglichkeiten unter /proc/sys

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

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

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

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

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

Und schon ist der Kernel in der Lage zu routen.

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

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

Plug and Play für Linux

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

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

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

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

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

- pnpdump
- isapnp

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

Karten zurückgeben. Dieses zweite Programm ist **isapnp**

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

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

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

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

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

Eine Beispielkonfiguration

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

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

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

```
# $Id: pnpdump.c,v 1.21 1999/12/09 22:28:33 fox Exp $
# Release isapnptools-1.21 (library isapnptools-1.21)
#
# This is free software, see the sources for details.
# This software has NO WARRANTY, use at your OWN RISK
#
```

```
# For details of the output file format, see isapnp.conf(5)
#
# For latest information and FAQ on isapnp and pnpdump see:
# http://www.roestock.demon.co.uk/isapnptools/
#
# Compiler flags:  -DREALTIME -DNEEDSETSCHEDULER -DABORT_ONRESERR
# (for   library:  -DREALTIME -DNEEDSETSCHEDULER -DABORT_ONRESERR)

# Card 1: (serial identifier 13 0e 1e 37 b4 19 01 89 14)
# EDI0119 Serial No 236861364 [checksum 13]
# Version 1.0, Vendor version 1.0
# ANSI string -->PLUG & PLAY ETHERNET CARD<--
# Logical device id EDI0119
#     Device support I/O range check register
        (CONFIGURE EDI0119/236861364 (LD 0
#     Compatible device id PNP80d6
#     Logical device decodes 10 bit IO address lines
#         Minimum IO base address 0x0240
#         Maximum IO base address 0x03e0
#         IO base alignment 32 bytes
#         Number of IO addresses required: 32
#         (IO 0 (BASE 0x0340))
#     IRQ 3, 4, 5, 9, 10, 11, 12 or 15.
#         High true, edge sensitive interrupt
#         (INT 0 (IRQ 10 (MODE +E)))
#     Memory is non-writeable (ROM)
#     Memory is non-cacheable
#     Memory decode supports high address
#     memory is 8-bit only
#     memory is shadowable
#     memory is an expansion ROM
#     Minimum memory base address 0x0c0000
#     Maximum memory base address 0x0dc000
#     Range base alignment mask 0xff4000 bytes
#     Range length 16384 bytes
# Choose UPPER = Range, or UPPER = Upper limit to suit hardware
```

```
# (MEM 0 (BASE 0x0c0000) (MODE bu) (UPPER 0x0c4000))
# (MEM 0 (BASE 0x0c0000) (MODE br) (UPPER 0x004000))
#         (ACT Y))
)
# End tag... Checksum 0x00 (OK)

(WAITFORKEY)
```

Aus den Angaben kann folgendes ersehen werden:

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

Die eigentliche Konfigurationsarbeit findet zwischen den Zeilen

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

und

```
  )
```

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

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

Aus dem Kommentar

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

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

- `0x0240`
- `0x0260`
- `0x0280`

- 0x02a0
- 0x02c0
- 0x02e0
- 0x0300
- 0x0320
- 0x0340
- 0x0360
- 0x0380
- 0x03a0
- 0x03c0
- 0x03e0

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

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

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

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

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

entsprechend um, und entziehen ihr das Kommentarzeichen:

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

Ähnlich gehen wir jetzt mit dem IRQ um. Die Zeile

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

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

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

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

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

entsprechend um und entfernen wiederum das Kommentarzeichen:

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

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

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

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

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

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

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

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

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.101.1

Konfiguration grundlegender BIOS-Einstellungen

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

Die wichtigsten Dateien, Bezeichnungen und Anwendungen

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

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

Die Funktion des Computer-BIOS

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

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

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

Optional sind womöglich noch andere Schnittstellen enthalten wie

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

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

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

Das CMOS-Setup Programm

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

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

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

hinzugefügt werden, können nicht mit diesem Programm eingestellt werden!

Das Standard-Setup

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

- Einstellung des Datums und der Uhrzeit
- Einstellung, welche Diskettenlaufwerke angeschlossen sind.
- Einstellungen welche Festplatten angeschlossen sind. (Diese Einstellung bezieht sich wiederum nur auf die Platten, die an die Schnittstellen angeschlossen sind, die auf dem Mainboard integriert sind, nicht die an zusätzlichen Controllern.)
- Primäre Graphikkarte. Stammt aus einer Zeit, als es noch Graphikkarten mit unterschiedlichen Speicheradressen gab (Hercules, MDA). Steht heute praktisch grundsätzlich auf VGA/EGA.
- Fehlerbehandlung beim Booten

Wichtig sind hier die Einstellungen der Festplatten und die Fehlerbehandlung.

Festplatteneinstellungen

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

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

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

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

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

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

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

Fehlerbehandlung

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

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

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

Peripheriegeräte einstellen

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

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

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

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

PnP und PCI Setup

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

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

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

Weitere Einstellmöglichkeiten

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

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.101.3

Konfiguration von Modem und Soundkarten

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

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

Kompatibilitätskriterien

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

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

Winmodems

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

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

Architektur von Sound unter Linux

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

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

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

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

1.101.6

Konfiguration von Kommunikationsgeräten

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

Die wichtigsten Dateien, Bezeichnungen und Anwendungen

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

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

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

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

Setzen der seriellen Schnittstelle auf 115.2 kbps mit setserial

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

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

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

```
setserial /dev/ttyS0
```

gibt uns beispielsweise folgende Ausgabe:

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

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

```
setserial Gerätedatei Parameter
```

Wichtige solcher Einstellungsparameter sind

port *Portnummer*

: setzt die IO-Portadresse der angegebenen Schnittstelle

irq *IRQ-Nummer*

: setzt den IRQ der angegebenen Schnittstelle

uart *UART-Typ*

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

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

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

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

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

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

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

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

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

ein erneuter Aufruf von

```
setserial /dev/ttyS0
```

bringt uns jetzt die Ausgabe:

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

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

Konfiguration eines Modems

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

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

werden, damit ihn das Modem überhaupt erst ausführt:

AT	Das Modem antwortet mit `OK`. Tut es das nicht, so stimmt etwas mit der Kommunikation zwischen Modem und Rechner nicht.
ATD *Nummer*	Das Modem wählt die angegebene Nummer und baut eine Verbindung auf.
ATH0	Modem legt auf
ATH1	Modem nimmt ab
ATX*Ziffer*	Stellt das Modem-Verhalten beim Verbindungsaufbau ein. ● ATX0 - Modem wählt und meldet CONNECT bei erfolgreichem Verbindungsaufbau ● ATX1 - Modem wählt und meldet CONNECT (Geschwindigkeit) ● ATX2 - Modem wartet auf Freizeichen, wählt und meldet CONNECT (Geschwindigkeit). ● ATX3 - Modem wählt und meldet CONNECT (Geschwindigkeit) oder BUSY (belegt). X3 sollte bei Nebenstellenanlagen verwendet werden, um das Warten auf ein Freizeichen der Amtsleitung zu vermeiden. ● ATX4 - X4 Modem wartet auf Freizeichen, wählt und meldet CONNECT (Geschwindigkeit). ● ATX5 - Modem wählt und meldet CONNECT (Geschwindigkeit), Busy, Voice (Telefon anstatt eines Modems an der Gegenstelle). ● ATX6 - Modem wartet auf Freizeichen, wählt und meldet CONNECT (Geschwindigkeit), Busy (belegt), Voice.
ATS0=0	Das Modem nimmt nicht selbstständig Anrufe entgegen. ATS0=1 bedeutet das Modem nimmt nach einem Läuten ab, S0=2 wäre nach zweimal Klingeln abheben ...
AT&W	Das Modem speichert seine gegenwärtige Einstellung
ATZ	Das Modem liesst die gespeicherte Einstellung und aktiviert sie.

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

```
ATS0=0 X3
```

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

```
ATZ
```

Nun sollte das Modem für ausgehende Verbindungen konfiguriert sein.

1.101.4

Einrichten von SCSI-Geräten

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

Die wichtigsten Dateien, Bezeichnungen und Anwendungen

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

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

Grundsätzliche Architektur von SCSI

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

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

Beim Aufbau des Busses müssen folgende Punkte beachtet werden:

- Der Bus darf nur zwei Enden haben, T-Stücke (Abzweigungen) sind verboten.
- Jedes Ende des Busses muß mit einem Terminator (Abschlußwiderstand) abgeschlossen sein. (Die Werte betragen 330 Ohm gegen Masse und 220 Ohm gegen +5V.) Heute werden diese Widerstände entweder softwaremäßig oder gar automatisch gesetzt. Bei älteren SCSI-Geräten war hier noch Steckarbeit erforderlich.
- Der Leitungsabstand zwischen zwei SCSI Geräten muß mindestens 10 cm betragen.
- Der SCSI Bus darf eine Gesamtlänge von drei Metern nicht überschreiten. (Bei mehr als vier Geräten darf er bei klassischem SCSI sogar nur 1,5 Meter lang sein.)
- Bastlerverbindungen (Pfostenstecker o.ä.) sind nicht zulässig, das Flachbandkabel muß durchgängig sein.
- Der Bus darf nicht durch ein offenes Kabelende abgeschlossen sein, d.h., daß selbst wenn nur ein Gerät (außer dem Adapter) angeschlossen ist immer der letzte Anschluß des Kabels verwendet werden muß.

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

Unterschiedliche SCSI-Typen

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

SCSI (oder SCSI1)

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

SCSI2

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

SCSI3 (oder Ultra SCSI)

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

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

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

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

Adressierung im SCSI-Bus

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

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

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

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

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

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

Busnummer,SCSI-ID,LUN

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

Das SCSI-BIOS

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

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

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

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

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

Booten von SCSI-Geräten

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

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

SCSI im Linux-System

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

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

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

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

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

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

Die Namensgebung von SCSI-Geräten unter Linux

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

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

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

Busnummer,SCSI-ID,LUN

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

```
scsi_info /dev/sda
```

ergibt beispielsweise eine Ausgabe wie

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

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

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.101.5

Einrichtung verschiedener PC-Erweiterungskarten

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

Die wichtigsten Dateien, Bezeichnungen und Anwendungen

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

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

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

besprochen. Hier also nur noch eine kurze Zusammenfassung:

Unterschied zwischen ISA und PCI

Die Unterschiede zwischen ISA-Karten und PCI Karten wurden im Groben bereits unter

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

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

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

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

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

Busnummer:Steckplatznummer.Funktionsnummer

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

Das Programm lspci

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

Das Programm **lspci** ermöglicht es, genau zu bestimmen, welche Geräte (Karten) am PCI-Bus angeschlossen sind und als was sie sich

ausgeben. Das ist insbesondere wichtig, um herauszufinden, welche Chipsätze oder Kompatibilitätskriterien bestimmte Karten aufweisen, um mit dieser Information dann das entsprechende Modul zu laden. Moderne Linux-Distributionen, die bei der Installation selbstständig herausfinden, welche Karten angeschlossen sind (Hardwareerkennung) bedienen sich dieses Programms.

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

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

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.101.7

Konfiguration von USB-Geräten

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

Die wichtigsten Dateien, Bezeichnungen und Anwendungen

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

Grundsätzliche Unterstützung von USB

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

```
modprobe usbcore
```

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

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

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

UHCI oder OHCI

Die meisten modernen Mainboards stellen einen USB-Controller zur Verfügung. Ältere Maschinen können mit einem USB-Controller auf einer PCI-Karte nachgerüstet werden. USB-Controler sind entweder kompatibel zum *Open Host Controller Interface* (OHCI - Compaq) oder zum *Universal Host Controller Interface* (UHCI - Intel). Beide Typen haben die selben Fähigkeiten und USB-Geräte arbeiten mit beiden Typen zusammen.

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

Mainboards der folgenden Firmen benutzen UHCI:

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

Mainboards der folgenden Firmen benutzen OHCI:

- Compaq
- Ali
- SiS
- NEC
- Opti Chipsatz

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

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

```
...
```
was eindeutig VIA - also UHCI bedeutet.

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

```
Bus  0, device   7, function  3:
  USB Controller: VIA Technologies, Inc. UHCI USB (#2) (rev 22).
    IRQ 10.
    Master Capable.  Latency=32.
    I/O at 0xa800 [0xa81f].
```
Wenn auch hier keine Angaben stehen, dann sollte die IO-Adresse angesehen werden. Ist sie in der Form `0xHHHH` (wobei HHHH für vier hexadezimale Ziffern steht), dann haben wir es mit UHCI zu tun, ansonsten, wenn die Form der Adresse `0xHH000000` ist, dann ist es OHCI.

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

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

Geladen werden diese Module entweder von Hand über den Befehl

```
modprobe usb-uhci
```
oder

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

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

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

```
post-install joydev modprobe hid
```

Das USBDevFS-Dateisystem

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

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

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

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

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

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

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

Sobald das entsprechende Modul `usb-ohci.o` oder `usb-uhci.o` geladen ist, der USB-Controller also ansprechbar ist, sollte mindestens noch ein (meist aber zwei) Unterverzeichnis zu finden sein, für jede USB-Schnittstelle eines:

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

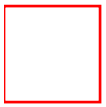
Die Einträge in diesen Verzeichnissen und Dateien sind binär und somit nicht für menschliche Augen gedacht. Es geht darum, im Fehlerfall Zugriff auf jedes einzelne Gerät zu haben und Diagnosen stellen zu können.

Das USB-Schichtenmodell von Linux

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

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

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



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

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

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

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

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

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

Module für die USB-Geräte automatisch einbinden

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

Grundsätzlich existieren zwei unterschiedliche Ansätze, um diese Fähigkeit anzubieten:

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

Beide Ansätze werden hier kurz beschrieben:

Funktionsweise von hotplug

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

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

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

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

Funktionsweise von usbmgr

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

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

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

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

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

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

1.102.1

Entwerfen einer Festplattenaufteilung

Beschreibung: Prüfungskandidaten sollten in der Lage sein, ein Partitionsschema für ein Linux-System zu entwerfen. Dieses Lernziel beinhaltet das Erzeugen von Dateisystemen und Swap-Bereichen auf separaten Partitionen oder Festplatten und das Maßschneidern des Systems für die beabsichtigte Verwendung des Systems. Ebenfalls enthalten ist das Platzieren von `/boot` auf einer Partition, die den BIOS-Anforderungen für den Systemstart genügt.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/ (root)` Dateisystem
- `/var` Dateisystem
- `/home` Dateisystem
- Swap-Bereiche
- Mount-Points
- Partitionen
- Zylinder 1024

Linux arbeitet - im Gegensatz zu anderen Systemen - nicht mit Laufwerksbuchstaben, sondern hängt alle zu benutzenden Partitionen in einen Dateibaum ein. Diese Technik des Mountens wird im Abschnitt 1.104 - Gerätedateien, Linux Dateisysteme, Filesystem Hierarchy Standard noch genauer behandelt. Hier geht es zunächst einmal um die grundlegenden Aufteilungen der Partitionen.

Es gibt verschiedene Gründe für die Benutzung von mehreren Partitionen unter Linux. Die wichtigsten sind:

- Mehrere physikalische Festplatten werden benutzt.
- Backups werden leichter planbar, wenn klar ist, daß veränderbare Daten nur auf bestimmten Partitionen vorkommen können.
- Userquotas (bestimmte Einschränkungen, wieviel Platz pro User zur Verfügung steht) beziehen sich auf Partitionen.
- Daten, die sich grundsätzlich nicht verändern (statische Systembereiche) können - wenn sie auf einer eigenen Partition liegen -

ReadOnly gemountet werden.

- Bootmanager können manchmal nur auf Partitionen unterhalb des 1024 Zylinders zugreifen.
- Bestimmte Teile des Systems sollen im Netz auch anderen Rechnern zur Verfügung stehen, andere aber nicht.

Andererseits erfordert die Aufteilung eines Linux-Systems in verschiedene Partitionen auch eine wesentlich genauere Planung, weil schon bei der Installation festgelegt werden muß, wieviel Platz auf welchem Bereich benötigt werden wird.

Jede Partition (Dateisystem) wird in ein Verzeichnis eingehängt (gemountet). Der ursprüngliche Inhalt dieses Verzeichnisses ist solange unsichtbar, solange die Partition in dieses Verzeichnis eingehängt ist. Eine Liste der wichtigen Verzeichnisse auf der Wurzelpartition ist im Abschnitt 1.104.8 nachlesbar.

Das Wurzeldateisystem

Das wichtigste Dateisystem ist das Wurzeldateisystem. Es ist die Partition, die beim Booten als erstes gemountet wird und von der aus alle weiteren Mount-Vorgänge erledigt werden. Jeder Bootmanager muß wissen, welche Partition des Systems die Wurzelpartition ist. Diese Partition enthält Verzeichnisse, die entweder Daten enthalten oder in die im weiteren Verlauf des Bootvorganges andere Partitionen gemountet werden.

Viele dieser Verzeichnisse können auf solche anderen Partitionen ausgelagert werden, andere dürfen unter keinen Umständen außerhalb der Wurzelpartition liegen. Diese Aufteilung ist nicht willkürlich, sondern hat ihre Gründe. Wie so oft, ist es besser diese Gründe zu verstehen, als einfach eine Liste von Verzeichnissen auswendig zu lernen. Daher folgt hier eine Liste all der Verzeichnisse, die niemals ausgelagert werden dürfen, zusammen mit der Begründung warum:

- **/bin**
  Das Verzeichnis `/bin` enthält die grundlegenden Systemprogramme, die zum Starten des Systems benötigt werden. Zum Beispiel das **mount**-Programm. Ohne dieses Programm können keine weiteren Dateisysteme gemountet werden, es ist also zwingend notwendig, daß `/bin` auf der Wurzelpartition liegt, wo es gleich nach dem Laden des Kernels zur Verfügung steht.
- **/dev**
  Das Verzeichnis `/dev` enthält die Gerätedateien, mit denen der Kernel physikalische Geräte (unter anderem auch Partitionen und Platten) ansprechen kann. Diese Gerätedateien werden benötigt, um Partitionen zu mounten. Also muß auch dieses Verzeichnis zwingend auf der Wurzelpartition liegen.
- **/etc**
  Das Verzeichnis `/etc` enthält - neben vielen anderen Informationen - die Information, wohin welche Partition gemountet werden soll. Ohne diese Information wäre es gar nicht möglich, andere Partitionen automatisch zu mounten.
- **/lib**
  Das Verzeichnis `/lib` enthält die grundlegenden Libraries, die von Programmen wie **mount** benötigt werden um zu funktionieren.

Außerdem liegt in diesem Verzeichnis das Unterverzeichnis `modules`, das die Kernelmodule enthält. Es können hier auch Kernelmodule liegen, die für die Erkennung von Dateisystemtypen notwendig sind.

- **/sbin**
  Das Verzeichnis `/sbin` enthält Programme, die während des Startvorganges notwendig sind, bevor gemountet wurde. So sind hier beispielsweise die Programme zur Überprüfung der Konsistenz von Dateisystemen enthalten. Diese Konsistenzüberprüfung muß vor dem Einhängen des jeweiligen Dateisystems stattfinden.

Diese Verzeichnisse müssen also zwingend auf der Wurzelpartition liegen, alle anderen Verzeichnisse können theoretisch auf andere Partitionen ausgelagert werden. Zumindestens das Verzeichnis `/root`, das Heimatverzeichnis des Systemverwalters sollte (muß aber nicht) noch auf dem Wurzelverzeichnis gelegen sein, da es vorkommen kann, daß der Systemverwalter in einem Wartungsmodus mit dem System arbeiten muß, in dem nur das Wurzelverzeichnis eingehängt ist. Wenn `/root` auf der Wurzelpartition liegt, ist so sichergestellt, daß er auch in diesem Wartungsmodus Zugriff auf seine Dateien hat.

Das /usr- und das /var-Dateisystem

Alle statischen Daten des Betriebssystems, die nicht während des Startvorganges benötigt werden, liegen im `/usr`-Verzeichnis (usr bedeutet nicht User sondern Unix System Resources). Dieses Dateisystem ist in aller Regel auf einer separaten Partition untergebracht, die wesentlich größer als die Wurzelpartition ist. Dieses Verzeichnis kann (und soll) Read-Only gemountet werden, so daß keine Veränderungen darin im laufenden Betrieb möglich sind. Das schafft eine wesentlich höhere Stabilität des Gesamtsystems, weil nur durch ein bewußtes Remounten eine Veränderung am System möglich wird.

Entsprechend werden alle Dateien, die im laufenden Betrieb veränderbar sein müssen, in ein anderes Verzeichnis ausgelagert, das die sogenannten variablen Dateien enthält und aus diesem Grund den Namen `/var` trägt. Hier liegen mindestens die folgenden Unterverzeichnisse:

- **/var/lib**
  Ein Verzeichnis für modifizierbare Einstellungen, die traditionell unter `/usr/lib` liegen würden.
- **/var/lock**
  Hier liegen sogenannte Lock-Dateien. Das sind Dateien, die von Programmen angelegt werden, um zu zeigen, daß diese Programme laufen. Damit kann verhindert werden, daß bestimmte Programme mehrfach gestartet werden.
- **/var/log**
  Hier liegen die Logbücher des Systems, die permanent erweitert werden müssen.
- **/var/run**
  Ähnlich wie `/var/lock` liegen hier Dateien, in die Programme ihre PID eintragen, damit diese auch ohne Aufruf von **ps** herausfindbar sind.
- **/var/spool**

- **/var/tmp**
  Temporäre Dateien.

Das Verzeichnis `/var` wird auch gerne und oft auf eine andere Partition als die Wurzelpartition ausgelagert. Es ist klar, daß in diesem Verzeichnis ständig Dateien angelegt, verändert und vergrößert werden. Die Gefahr, auf der Wurzelpartition keinen Platz mehr zu haben kann also speziell durch die Auslagerung von `/var` verringert werden.

Das /home-Dateisystem

Im Verzeichnis `/home` liegen die Verzeichnisse der Benutzer/innen. Je nach Anspruch bzw. je nach Aufgabe des Rechners wird dieses Verzeichnis viel oder wenig Platz brauchen. Soll der Rechner z.B. nur ein Router ins Internet werden, oder ein alleinstehender Webserver, so benötigen wir für ihn praktisch keinen Speicherplatz für Benutzer, da es auf einem solchen Rechner keine Benutzer gibt. Ein Rechner hingegen, der als Fileserver dient oder eine normale Arbeitsstation ist, wird hier - je nach verfügbarem Plattenplatz - möglichst viel Speicher anbieten.

Im zweiten Fall ist es unbedingt ratsam, das `/home` Verzeichnis auf eine andere Platte auszulagern. Ein bösartiger Benutzer könnte ansonsten ein Programm schreiben, das die Festplatte füllt, und damit die Wurzelpartition bis obenhin voll machen, was dazu führen würde, daß Linux nicht mehr arbeiten kann. Es gibt dagegen zwar verschiedene Hilfsmechanismen wie Disk-Quota (siehe Abschnitt 1.104.4 - Verwalten von Diskquotas) oder der Reservierung von Plattenplatz auf EXT2-Dateisystemen, trotzdem ist eine Auslagerung auf eine andere Partition in einem solchen Fall immer ratsam.

Ein weiterer Aspekt der Auslagerung sind die Backups. Das `/home` Verzeichnis sollte grundsätzlich immer in ein Backup aufgenommen werden, weil hier ja die Dateien der User liegen, die einer ständigen Veränderung unterworfen sind.

Das /tmp-Verzeichnis

Dieses Verzeichnis ist - neben dem `/home`-Verzeichnis - das einzige auf dem System, in dem Normaluser Schreibrechte besitzen. Also ist es auch hier ratsam, dafür zu sorgen, daß - wie oben schon erwähnt - die User nicht die Systemplatte füllen können. Auch dieses Verzeichnis kann und soll auf eine andere Partition ausgelagert werden, wenn es ein System ist, von dem wir erwarten, daß dort User arbeiten. Auf statischen Servern wie reinen Internet-Routern kann dieses Verzeichnis aber problemlos auch auf der Wurzelplatte verbleiben.

Das /boot Dateisystem und das Problem der 1024 Zylinder

Auf älteren Systemen gibt es ein Problem mit Festplatten, die mehr als 1024 Zylinder haben. Dieses Problem bezieht sich nicht auf den Zugriff auf die Platte durch Linux, sondern auf den Zugriff durch Bootmanagersoftware.

Wenn ein Bootmanager wie LILO beim Systemstart auf Festplatten zugreift, etwa um den entsprechenden Kernel zu laden, so muß er ohne Betriebssystem auf die Platte zugreifen. Er muß also mit BIOS-Routinen arbeiten, die eben das Problem haben können, nicht mehr als 1024 Zylinder direkt ansprechen zu können. Ist die Partition, die die Dateien des Bootmanagers enthält jetzt auf einer Partition, deren Grenzen überhalb der 1024 Zylinder liegen, dann kann der Bootvorgang nicht funktionieren.

Aus diesem Grund gibt es die Möglichkeit, eine sehr kleine Partition (ungefähr 20 MB) anzulegen, die am Anfang der Platte liegt, also unterhalb der 1024 Zylinder. Diese Partition wird ins Verzeichnis `/boot` gemountet und enthält alle Informationen, die der Bootmanager benötigt (Kerneldateien, Initrd-Images, `System.map, boot.b, chain.b`, ...).

Durch diesen Trick ist der Bootmanager während des Bootvorgangs in der Lage, auf seine Dateien zuzugreifen, ohne die 1024er Grenze zu überschreiten.

Moderne BIOSe und moderne Bootmanager umgehen dieses Problem über die Verwendung der LBA-Methode. Dann ist die Auslagerung nicht mehr nötig.

Swap-Partitionen

Ein Linux-System kann, wenn der physikalische Arbeitsspeicher zur Neige geht, einen bestimmten Partitionstyp als Speicherersatz für den Notfall benutzen. Diese Technik des Auslagerns (Paging) ist eine der ältesten Eigenschaften von Linux und hat es gerade auf Privatrechnern mit wenig Speicher so beliebt gemacht.

Eine Partition, die als Speichererweiterung dienen soll, wird als Swap-Partition bezeichnet und benötigt zwei grundlegende Vorbereitungstechniken:

- Sie muß beim Anlegen mit **fdisk** bereits den Partitionstyp *Linux Swap* (Typ 82) bekommen.
- Sie bekommt zwar kein "echtes" Dateisystem, muß jedoch mit dem Programm **mkswap** vorbereitet werden.

Während des Bootens wird eine solche Partition mit Hilfe des Kommandos **swapon** aktiviert. Sie wird also nicht wirklich gemountet, bekommt jedoch auch einen Eintrag in `/etc/fstab` in der Art:

```
/dev/hdc2       none            swap    sw      0       0
```

Eine schwierige Frage ist die richtige Größe einer solchen Swap-Partition. Früher gab es die Daumenregel, daß diese Partition doppelt so groß wie der physikalische Arbeitsspeicher sein sollte. Das war in den Zeiten verständlich, in denen Arbeitsspeichergrößen im Bereich

16-32 MB üblich waren. Da konnte es vorkommen, daß ein Programm geladen werden sollte, daß alleine 13 MB erfordert hatte (etwa Netscape). Damit wäre dieses Programm zwar langsam, aber eben doch gelaufen.

Heute sind wesentlich größere Speichergrößen üblich, die diese Daumenregel ad absurdum führen. Wenn ein Rechner 256 MB Ram besitzt, so macht es wenig Sinn, seine Swap-Partition 512 MB groß zu erstellen. Denn eine tatsächliche Arbeit mit einem System, das 512 MB Speicher auslagert ist nicht mehr möglich. Arbeitsspeicher ist einfach sehr viel schneller als Plattenzugriffe...

Eine praxistaugliche Größe der Swap-Partition ist etwas um 64 MB bis 128 MB.

Zusammenfassung

Es gibt keine wirklich allgemeingültige Aussage, wie die Partitionen unter Linux aufgeteilt werden sollen. Es ist zu sehr abhängig von der geplanten Verwendung des Systems. Ein kleiner Router stellt andere Anforderungen, als ein großer Fileserver, eine Arbeitsstation andere als ein Server. Prinzipiell sollten die oben gemachten Angaben über die verschiedenen auszulagernden Dateisysteme immer auf die jeweilige Verwendung des Rechners bezogen überdacht werden. Ein kleiner Router mit einer Festplatte mit weniger als 1024 Zylindern kann einfach eine Partition bekommen, die alles enthält, ein großer Server sollte eine durchdachte Aufteilung der Partitionen erhalten.

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.102.2

Installation eines Bootmanagers

Beschreibung: Prüfungskandidaten sollten in der Lage sein, einen Bootmanager auszuwählen, zu installieren und zu konfigurieren. Dieses Lernziel beinhaltet das Bereitstellen alternativer und Sicherungsbootmöglichkeiten (z.B. mittels Bootdiskette).

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

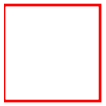
- `/etc/lilo.conf`
- `/boot/grub/grub.conf`
- **lilo**
- **grub-install**
- MBR
- Superblock
- Bootloader der ersten Stufe

Prinzipielles zur Funktion von Bootmanagern

Auf der Hardware-Ebene von industriestandardkompatiblen Computern (und um die geht es hier) ist der Bootvorgang von Betriebssystemen klar vorgegeben. Das BIOS (Basic Input Output System) im ROM jedes Computers sucht auf der Platte, von der es booten soll/will nach dem äußersten Zylinder (Nr.0). Dieser Zylinder ist selbst nicht Teil von Partitionen, sondern er enthält z.B. die Partitionstabelle, die erstmal festlegt, wo denn überhaupt welche Partitionen beginnen und enden. Neben dieser Information befindet sich auf der Spur 0 jedes physikalischen Laufwerks der sogenannte **Master Boot Record** (MBR). Dieser MBR enthält gewöhnlich einfach einen Verweis auf die erste Partition der Platte und der dort abgelegten Bootinformation.

Jede Partition der Platte besitzt nämlich wiederum auf ihrem ersten Zylinder einen lokalen Bootsektor, der Informationen enthalten kann (bei DOS/Win-Systemen) wie das auf dieser Partition befindliche Betriebssystem gebootet werden soll. Diese Information besteht aus

einem kleinen Programm, dem sogenannten Bootstrap-Loader.



Wenn auf einem System mehrere Betriebssysteme vorhanden sind, dann kann im Master Boot Record ein kleines Programm installiert werden, das den User entscheiden lässt, von welcher Partition welches System gebootet werden soll. Dieses Programm wird als **Bootmanager** bezeichnet.

Auf dem MBR ist aber nur sehr wenig Platz (weniger als 512 Byte). Das dort abgelegte Programm ist also zwangsläufig sehr klein. Meist wird das Programm nichts anderes tun, als ein zweites Programm zu laden, das das eigentliche Menü des Bootmanagers enthält. Dieses zweite Programm befindet sich zwangsläufig schon auf einer Festplattenpartition. Der Bootmanager muß also - unter Umgehung des Betriebssystems, das ja noch gar nicht geladen ist - Zugriff auf das Dateisystem dieser Partition haben und das gewünschte Programm dort laden. Dieser Vorgang wird als erste Stufe (Stage 1) des Bootloaders bezeichnet.

Das Programm, das das eigentliche Menü enthält, ist also die zweite Stufe (Stage 2) des Ladevorganges. Hier kann der/die BenutzerIn also aussuchen, welches Betriebssystem von welcher Partition gebootet werden soll. Erst nach dieser Auswahl wird das eigentliche System geladen (Stage 3).

Der klassische Bootmanager unter Linux war jahrelang das Programm **lilo** (Linux Loader). Neuerdings existiert eine zweite Möglichkeit, der GRand Unified Bootloader **grub**. Verschiedene Distributionen bedienen sich inzwischen dieser zweiten Methode, daher werden beide im weiteren Verlauf des Kapitels besprochen.

Konfiguration von lilo

Wenn der Bootmanager **lilo** benutzt werden soll, so werden alle Konfigurationsinformationen in der Datei `/etc/lilo.conf` abgelegt. Das Programm **lilo** liesst diese Datei und führt die folgenden Schritte aus:

- Aus den Einträgen der Konfigurationsdateien werden Menüeinträge erstellt, die in eine Datei (standardmäßig `/boot/boot.b`) abgelegt werden. Weitere Menüelemente können - je nach Konfiguration - in den Dateien `/boot/boot-menu.b`, `/boot/message` und `/boot/boot-bmp.b` abgelegt werden.
- Die Adressen und Größen der zu ladenden Kernel-Dateien werden in die Datei `/boot/map` geschrieben. Damit ist lilo später in der Lage, auf diese Dateien zuzugreifen, ohne ein Betriebssystem zu benutzen.
- Ein kleines Programm wird in den Master Boot Record geschrieben, das die Adressen der oben genannten Dateien in `/boot` enthält und die Aufgabe hat diese zu laden.

Das bedeutet, daß jedesmal, wenn eine Veränderung an den zu bootenden Kerneldateien gemacht wird, das Programm **lilo** neu aufgerufen werden muß. Selbst wenn alle Dateinamen gleich bleiben, so verändern sich eben doch die Adressen und Größen auf der Platte und die

Informationen in `/boot/map` würden nicht mehr stimmen.

Normalerweise genügt der Aufruf von lilo ohne weitere Parameter.

Wird lilo mit der Option `-u` oder `-U` aufgerufen, so löscht es die Einträge auf dem entsprechenden Bootsektor und stellt ihn wieder so her, wie er war, bevor lilo installiert wurde. Damit kann eine Platte also wieder vom Bootmanager befreit werden.

Wenn beim Booten mit lilo ein Fehler auftritt, so kann dieser Fehler anhand des Bootprompts eingekreist werden. Der Bootprompt ist standardmäßig einfach das Wort:

```
LILO boot:
```

Das Wort LILO wird Zeichen für Zeichen auf den Schirm geschrieben. Falls es zu einem Fehler während des Bootmanager-Vorgangs kommt, so kann dieser Fehler anhand der Tatsache eingekreist werden, wieviele Buchstaben schon geschrieben wurden. Die folgenden Schritte werden jeweils durchgeführt, wenn ein Buchstabe geschrieben wurde:

nichts

Kein Teil von Lilo wurde geladen. Entweder ist lilo nicht installiert oder er sitzt auf einem Bootsektor einer Partition, die nicht aktiviert ist.

L*Fehlernummer*

Der erste Teil des Bootloaders wurde lokalisiert und geladen, aber er kann den zweiten Teil nicht laden. Die zweistellige Fehlernummer gibt genauere Hinweise auf den Grund an. In der Regel deutet dieser Fehler auf ein Problem mit der Plattenoberfläche oder falschen Plattenparametern im BIOS hin.

LI

Der erste Teil des Bootloaders hat den zweiten Teil geladen, kann ihn aber nicht ausführen. Das kann daran liegen, daß es entweder eine Inkompatibilität mit der Plattengeometrie gibt, oder die Datei /boot/boot.b wurde von der Stelle bewegt, an der sie lag, als lilo installiert (aufgerufen) wurde.

LIL

Der zweite Teil des Bootloaders wurde gestartet, kann aber die Beschreibungstabelle des map-files nicht laden. Typischerweise ein Medien-Fehler (Oberflächenbeschädigung) oder unpassende Plattengeometrie.

LIL?

Der zweite Teil des Bootloaders wurde an eine falsche Adresse geladen. Auch hier ist die wahrscheinlichste Ursache, daß die Datei /boot/boot.b verändert oder bewegt wurde.

LIL-

Die Beschreibngstabelle ist beschädigt. Entweder ein Oberflächenfehler, oder die Datei /boot/map wurde verändert oder bewegt.

LILO

Alle Teile von lilo wurden ordnungsgemäß geladen.

In jedem Fall ist bei einem Fehler immer der erste Versuch, lilo nochmal aufzurufen, nachdem das System über eine andere Technik (Bootdiskette, BootCD) gebootet wurde. War es der erste Versuch, der gescheitert ist, also hat lilo noch nie vorher funktioniert, so sollte davor aber nochmal die Datei /etc/lilo.conf überprüft werden.

Die Datei /etc/lilo.conf

Die Datei /etc/lilo.conf enthält die Angaben, die lilo braucht um den Bootloader in einen Bootsektor zu schreiben. Sie besteht im Wesentlichen aus zwei Teilen. Der erste Teil beschreibt globale Einstellungen, der zweite beschreibt für jedes zu bootende System separate Einstellungen.

Ein typischer erster Teil einer lilo.conf-Datei könnte folgendermaßen aussehen:

```
append="reboot=warm"
boot=/dev/hda
lba32
message=/boot/message
prompt
timeout=300
```

Die Zeile `append=...` ermöglicht es, jedem zu bootenden Kernel bestimmte Parameter mitzugeben. Hier ist es der Parameter `reboot=warm`, der dazu führt, daß beim Reboot nicht ein Kaltstart (mit Speicherprüfung) sondern ein Warmstart ausgeführt wird.

Die Zeile `boot=...` gibt den Ort an, auf den lilo den Bootloader schreiben soll. Wird hier eine ganze physikalische Platte angegeben, also eine Gerätedatei ohne Partitionsnummer wie /dev/hda, so wird der Master-Boot-Record dieser Platte beschrieben. Stünde hier die Angabe einer Partition, also etwa /dev/hda2, so würde der Bootsektor der Partition beschrieben.

Die Angabe von `lba32` versetzt lilo in die Lage, auch mit Platten mit mehr als 1024 Zylindern umzugehen, wenn im BIOS der LBA-Modus eingestellt wurde.

Die Zeile `message=...` gibt an, aus welcher Datei die Bildschirmmeldung des Bootloaders stammen soll. Ohne diese Zeile wird keine Bildschirmmeldung benutzt. Der Inhalt der angegebenen Datei wird mit auf den Bootsektor geschrieben, wenn lilo aufgerufen wird. Sollten also Veränderungen vorgenommen werden sollen, so reicht es nicht, diese in die angegebene Datei zu schreiben. Es muß das Programm lilo nochmal aufgerufen werden, damit diese Veränderungen auch tatsächlich auf den Bootsektor kommen.

Die Angabe `prompt` veranlasst lilo, auf eine Benutzereingabe zu warten. Das ist insbesondere dann nötig, wenn mehr als ein

Betriebssystem bootbar sein soll.

Die Zeile `timeout=...` gibt schließlich die Zehntelsekunden an, die auf Benutzereingaben gewartet werden soll, bevor das voreingestellte System gebootet wird. Die Angabe 300 steht also für 30 Sekunden.

Die Handbuchseite über lilo.conf(5) zeigt alle möglichen globalen Parameter auf.

Als zweiter Teil der Datei `/etc/lilo.conf` stehen die Einstellungen für die zu bootenden Systeme. Jede dieser Angaben beginnt mit einem `image=...` für Linux-Systeme oder einem `other=...` für alles andere und enthält dann noch mindestens die Angabe, auf welcher Partition sich die Wurzel des zu bootenden Systems befindet und welches label sie hat, mit dem lilo dann mitgeteilt werden kann, daß diese Partition gebootet werden soll. Ein Beispiel:

```
image = /boot/zImage
  root = /dev/hda2
  initrd = /boot/initrd
  label = linux

image = /boot/vmlinuz
  root = /dev/hda2
  label = linuxalt

other = /dev/hda1
  label = windows
  alias = win
```

Der erste Block ist der voreingestellte. Dieses System wird also gebootet, wenn das Timeout erreicht ist oder LILO einfach mit der Enter-Taste beantwortet wird.

Dieser erste Block beschreibt also, daß der Kernel, der in /boot/zImage abgelegt wurde, gebootet werden soll. Das Wurzeldateisystem ist /dev/hda2. Die Zeile `initrd = /boot/initrd` bezeichnet die zu ladende initiale Ramdisk.

Der zweite Block beschreibt eine andere Kerneldatei (/boot/vmlinuz), die aber auch die Partition /dev/hda2 als Wurzelverzeichnis nützt. Um diesen Kernel zu booten muß auf dem Bootprompt das Wort "linuxalt" eingegeben werden.

Der dritte Block bezieht sich auf ein Nicht-Linux System (other). Statt der Kerneldatei wird hier einfach die Angabe der Partition gemacht, auf der dieses andere System liegt. Lilo kann ja davon ausgehen, daß jedes andere System auf dem Bootsektor seiner Partition seinen Bootstrap-Loader selbst abgespeichert hat. Das einzige, was wir noch an Information brauchen ist das Label, also das Wort, was wir auf dem Bootprompt angeben müssen, um dieses System zu laden. In unserem Beispiel ist es "windows". Weil das sehr lang ist, kann dazu

noch ein Alias vergeben werden, in unserem Fall das Wort "win".

Konfiguration von grub

Im Gegensatz zu **lilo** schreibt **grub** kein statisches Menü in eine der Dateien unter `/boot`. **grub** schreibt einfach nur einen Eintrag in den Master Boot Record der seinerseits wieder ein Programm unter `/boot/grub` läd. Dieses Programm ließt dann die Konfigurationsdatei `/boot/grub/grub.conf` und erstellt aus der darin gefundenen Information das Bootmenü.

Um mit **grub** zu arbeiten genügt es also einmal den Befehl

```
/sbin/grub-install /dev/hda
```

einzugeben, wobei die angegebene Gerätedatei `/dev/hda` den Master Boot Record der ersten IDE-Platte meint. Soll von einer SCSI-Platte gebootet werden (in einem System ganz ohne IDE-Platten), so muß diese Angabe entsprechend modifiziert werden. Diese Angabe bezieht sich **nicht** auf die Partitionen der zu bootenden Systeme sondern auf den Ort, wohin der Bootmanager geschrieben werden soll. Das BIOS der PCs sucht den Bootmanager immer zuerst auf dem Master Boot Record der ersten IDE-Platte, also ist `/dev/hda` in der Regel richtig.

Durch diese Architektur muß **grub** jetzt nicht mehr jedesmal aufgerufen werden, wenn sich an seiner Konfiguration etwas ändert. Da die zweite Stufe des Bootmanagers selbst die Konfigurationsdatei liesst, entfällt diese Notwendigkeit. Das ist natürlich auch der Grund, warum die Konfigurationsdatei von **grub** nicht im Verzeichnis `/etc` liegt, sondern auf dem `/boot` Dateisystem.

Die Datei /boot/grub/grub.conf

Diese Datei enthält die eigentliche Information über die zur Verfügung stehenden Bootmöglichkeiten. Damit geht **grub** einen anderen Weg als **lilo**. Diese Datei kann jederzeit auch im Nachhinein verändert werden und **grub** muß deshalb nicht erneut aufgerufen werden. Die Datei hat ein ähnliches Format wie die Konfigurationsdatei von **lilo**, aber sie unterscheidet sich im Detail erheblich.

Ein Beispiel für eine `grub.conf` Datei erfordert es, daß wir uns ein imaginäres Linux-System aufbauen. Nehmen wir folgendes Partitionsaufteilung:

```
/dev/hda1          /boot-Partition von Linux
/dev/hda2          Windows98-Partition
/dev/hda3          Wurzelpartition von Linux
```

Eine einfache Konfigurationsdatei von **grub** könnte dann folgendermaßen aussehen:

```
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz

title Linux (2.4.18)
      root (hd0,0)
      kernel /bzImage-2.4.18 ro root=/dev/hda3
      initrd /initrd-2.4.18.img

title Linuxalt (2.2.14)
      root (hd0,0)
      kernel /bzImage-2.2.14 ro root=/dev/hda3
      initrd /initrd-2.2.14.img

title Windows 98
      map (hd0,0) (hd0,1)
      map (hd0,1) (hd0,0)
      rootnoverify (hd0,1)
      chainloader +1
```

Die ersten drei Zeilen sind - ähnlich wie bei `lilo.conf` die globalen Einstellungen. `default=0` bedeutet, daß der erste Eintrag gewählt wird, wenn der Timeout überschritten wurde. Die zweite Zeile definiert diesen Timeout Wert in Sekunden. Die dritte Zeile definiert das verwendete Hintergrundbild. Hier finden wir bereits die erste Anwendung der grub-spezifischen Adressierung, die gleich näher erläutert wird.

Die folgenden Einträge definieren jeweils die zu bootenden Systeme. Die einzelnen Zeilen bedeuten jeweils folgendes:

title *Titelzeile*

Diese Zeile leitet einen Abschnitt eines zu bootenden Systems ein. Der unter *Titelzeile* gemachte Eintrag erscheint später im Menü.

root hd*n,m*

Dieser Eintrag ist **nicht** das Wurzelverzeichnis des Linuxsystems. Er bezeichnet die Partition, auf der die Kerneldateien und die **grub**-Konfigurationsdateien liegen. In unserem Beispiel ist das `/dev/hda1`, also die `/boot`-Partition von Linux. Die angegebenen Zahlen bezeichnen folgendes:

- *n* - Nummer der Platte (0 ist die erste Platte, 1 die zweite, ...)
- *m* - Nummer der Partition auf der Platte (0 ist die erste Partition, 1 die zweite, ...)

Unser Beispiel bezeichnet also mit `(hd0,0)` die erste Partition der ersten Platte also im Linux-Jagon `/dev/hda1`.

Alle weiteren Angaben beziehen sich auf diese Platte.

kernel *Kerneldatei Parameter*

Mit dieser Angabe wird die Kerneldatei angegeben und mit Aufrufparametern versehen. Der Pfad der Kerneldatei bezieht sich hier **nicht** auf die Wurzel des Linux-Systems sondern auf die Wurzel des Systems, das wir im letzten Punkt (root) angegeben hatten. In unserem Beispiel liegt die Kerneldatei also unter `/boot/bzImage-2.4.18`.

Die eigentliche Wurzel des zu bootenden Linux-Systems wird als Kernelparameter angegeben (hier also `/dev/hda3`.

initrd *Ramdiskimage*

Hier wird der Name der Ramdiskdatei angegeben, die als initiale Ramdisk geladen werden soll. Benutzt der Kernel keine initiale Ramdisk, so wird diese Zeile komplett weggelassen. Auch hier bezieht sich der Pfad auf die Wurzel des unter `root` angegebenen Devices.

Der zweite Eintrag definiert einen zweite Kernel auf der selben Partition. Die Angaben entsprechen denen des ersten Eintrags.

Im dritten Eintrag wird eine Windows-Partition definiert, die Win98 booten soll. Dieser Eintrag enthält einige Besonderheiten, die hier noch kurz dargestellt werden sollen.

map (*Partition1*) (*Partition2*)

Windows98 bootet normalerweise nur von der ersten Partition der ersten Festplatte. Durch den Befehl `map (hd0,0) (hd0,1)` erklären wir die Partition `hd0,0` zur zweiten Partition (`hd0,1`) und mit der nächsten Zeile wird entsprechend die zweite Partition zur ersten. Aus der Sicht des Windows-Systems ist also die zweite Partition jetzt die erste und umgekehrt. Damit ist für Windows die Welt wieder in Ordnung und es glaubt, von der ersten Partition zu booten.

rootnoverify (*Partition*)

Diese Angabe entspricht der Angabe `root` bei Linux-Einträgen, nur das diesesmal die entsprechende Partition nicht gemountet wird. (Das ist unter Windows nicht nötig, da ja nicht eine bestimmte Datei geladen werden soll, sondern der Bootsektor der Partition)

chainloader +1

Diese Angabe ist für Windows erforderlich. Windows wird über einen **chainloader** gebootet, der auf einem betimmten Sektor der Windows-Partition liegt. Die Angabe +1 bezeichnet hier die Sektornummer 1, also den Bootsektor der Windows-Partition (die unter `rootnoverify` angegeben wurde).

Es existieren natürlich noch viele weitere Möglichkeiten für die Konfigurationsdatei, aber die hier genannten sollten einen ausreichenden Überblick über den Einsatz von **grub** als Bootmanager geben.

Sicherheitsbootmöglichkeiten

Wenn aus irgendeinem Grund der Master Boot Record zerstört wurde, dann wird weder **lilo** noch **grub** ein Bootmenü anbieten. In einem solchen Fall ist es notwendig, über eine alternative Möglichkeit booten zu können um dann mit den entsprechenden Befehlen **lilo** (`lilo`) oder **grub** (`grub-install`) erneut in den MBR zu installieren.

Praktisch jedes Installationsmedium von Linux (Boot-CDs, Bootdisketten) bietet die Möglichkeit, ein bereits installiertes Linux-System zu booten. Dazu muß nur eine Angabe bekannt sein, nämlich die Partition, die für dieses zu bootende System die Wurzelpartition ist. Das kann einfach dadurch geschehen, daß beim Booten des Installationsmediums der Kernelparameter

```
root=Partition
```

angegeben wird. Die Partition wird hier in einer für Linux verständlichen Form angegeben, also beispielsweise `/dev/hda3`. Damit wird dann der auf dem Installationsmedium befindliche Kernel gebootet, seine Wurzelpartition ist aber dann die angegebene Partition. Das heißt, wir können dann entsprechend auf dem System arbeiten, um den richtigen Bootmanager wieder zu installieren.

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.102.3

Erstellen und Installieren von im Sourcecode vorliegenden Programmen

---

Beschreibung: Prüfungskandidaten sollten in der Lage sein, ein ausführbares Programm aus dem Quellcode zu erstellen und zu installieren. Dieses Lernziel beinhaltet die Fähigkeit, ein Source-Paket aus einem Archiv zu extrahieren. Kandidaten sollten in der Lage sein, einfache Anpassungen im Makefile vorzunehmen, wie z.B. Pfade zu ändern oder zusätzliche Verzeichnisse einzubinden.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **gunzip**
- **gzip**
- **bzip2**
- **tar**
- **configure**
- **make**

Grundsätzlicher Vorgang

Ein Programm wird von seinem Programmierer (oder seinen Programmierern) in vielen einzelnen Quellcode-Dateien geschrieben. Diese Dateien enthalten den Quellcode in der Programmiersprache, die vom Programmierer benutzt wurde, also beispielsweise C.

Ein Compiler muß jetzt jede dieser einzelnen Dateien in sogenannte Objekt-Code Dateien compilieren. Diese Objekt-Dateien enthalten dann den Maschinencode, der vom Compiler erzeugt wurde, sind aber noch nicht selbst lauffähig. Für jede einzelne Quellcode-Datei wird eine entsprechende Objektdatei erzeugt. In der Regel tragen diese Dateien die Namensendung `.o`.

Erst im nächsten Schritt werden die verschiedenen erzeugten Objektdateien zusammen mit eventuellen weiteren Libraries zu einer

ausführbaren Datei verbunden. Dieser Vorgang wird Linken (engl. to link - verbinden) genannt. Das Ergebnis dieses Vorgangs ist die ausführbare Datei, die dann verwendet werden kann.

Der Compilier- und Linkvorgang besteht also aus vielen Einzelschritten. Damit nicht bei jedem Compiliervorgang alle diese Einzelschritte angegeben werden müssen, gibt es das Programm **make**. Dieses Programm erhält seine Informationen welche Dateien mit welchem Compiler wie übersetzt werden müssen und welche Objektdateien dann wie zu welcher ausführbaren Datei zusammengefügt werden sollen aus einer Datei mit Namen **Makefile**.

Ein Programmierer kann also neben seinem Programm ein dazu passendes Makefile erstellen, so daß alle Schritte, die für die Compilierung notwendig sind, vom Programm **make** ausgeführt werden können. Dazu erstellt er im Makefile bestimmte Regeln, die dann die entsprechenden Befehle enthalten, die notwendig sind um eine Regel auszuführen.

Der Name einer Regel wird dann dem Programm **make** als Parameter mitgegeben. Das Programm sucht dann das Makefile nach der entsprechenden Regel ab und führt die dort gefundenen Befehle aus.

make arbeitet außerdem mit den Zeitmarken der Dateien. Bei einem zweiten Compiliervorgang wird für jede Objektdatei überprüft, ob die Zeitmarke der dazu passenden Quellcode-Datei älter als die der Objektdatei ist. Nur wenn die Zeitmarke der Quellcodedatei jünger als die der Objektdatei ist, wird die Quellcodedatei erneut compiliert.

Mit Hilfe dieser Technik ist es auch für einen Nicht-Programmierer einfach möglich, komplexe Programme, die im Quellcode vorliegen, selbst zu übersetzen und zu installieren. Im Linux-Bereich haben sich dazu bestimmte Standards herausgebildet, die im weiteren Verlauf besprochen werden sollen.

Format eines Quellcode-Paketes

Quellcode-Pakete werden im Linux-Bereich immer als Tar-Archive ausgeliefert. In der Regel sind diese Archive komprimiert, damit sie schneller übertragen werden können. Ein unkomprimiertes tar-Paket trägt die Endung `.tar`, ein mit **gzip** komprimiertes Paket hat entweder die Endung `.tar.gz` oder `.tgz`. Ist das Archiv mit **bzip2** komprimiert worden, so lautet die Endung `.tar.bz2`.

Das tar-Programm von Linux ist in der Lage mit Archiven direkt umzugehen, die mit gzip komprimiert wurden, Archive, die mit **bzip2** komprimiert wurden, müssen vorher entpackt werden.

Die Befehle zum Entpacken eines Tar-Archivs lauten also

```
tar -xzvf Archivname.tar.gz
```

für ein mit **gzip** gepacktes Archiv und

```
bzcat Archivname.tar.bz2 | tar -xvf -
```

für ein mit **bzip2** komprimiertes Archiv. Dabei ist zu beachten, daß zwischen dem Parameter f und dem folgenden Bindestrich ein Leerzeichen steht. Eine genauere Beschreibung des **tar**-Programms findet sich in der Vorbereitung auf die LPI102 Prüfung im Abschnitt 1.111.5 - Aufrechterhaltung einer effektiven Datensicherungsstrategie.

Moderne Versionen von **tar** können auch `.bz2`-Archive direkt entpacken. Dazu muß statt dem `z` ein `j` angegeben werden.

Als Ergebnis dieses Vorganges wird jetzt ein Verzeichnis erstellt, das den Quellcode des zu compilierenden Programmes enthält.

Erstellen des Makefiles

Ein sehr einfaches Programm enthält jetzt bereits das passende Makefile, das für die Compilierung notwendig ist. Nachdem aber heute die meisten Programme von der Existenz bestimmter Libraries abhängig sind, ist in den meisten Fällen zunächst eine genaue Prüfung der in diesem System vorliegenden Besonderheiten nötig. Dazu dient das GNU-Autoconf Paket, das einen entsprechenden Test des Systems ermöglicht und aus den entsprechenden Ergebnissen dieses Tests das Makefile erstellt.

Wenn das Verzeichnis bereits eine Datei enthält, die den Namen `Makefile` trägt, so ist dieser Schritt nicht mehr nötig. Fehlt diese Datei jedoch aber es existieren die Dateie `Makefile.in` und `configure` und die Datei `configure` ist ausführbar, dann muß das Makefile erst erstellt werden.

Dazu wird einfach das Script **configure** in diesem Verzeichnis aufgerufen. Der Befehl erfordert die Eingabe eines Pfades (`./`) um zu verhindern, daß ein anderes Programm diesen Namens benutzt wird. Er lautet also einfach

```
./configure
```

Jetzt wird das System nach allen möglichen Dingen durchsucht, die notwendig sind, um das Programm zu compilieren. Fehlen entsprechende Elemente, so wird darauf hingewiesen und kein Makefile erstellt. Sind aber alle Voraussetzungen erfüllt, so endet das **configure**-Script mit der Meldung

```
Creating Makefile
```

und erstellt unser Makefile. Jetzt erst kann die wirkliche Compilierarbeit beginnen.

Manuelles Veärndern des Makefiles

In manchen Fällen sind noch manuelle Veränderungen am Makefile notwendig. Diese Veränderungen beziehen sich in der Regel auf Pfadangaben zu bestimmten Programmen oder Angaben, wohin das Programm nach erfolgter Compilierung installiert werden soll. Diese Variablen können zumeist in den ersten paar Zeilen des Makefiles verändert werden.

Variablen werden in Makefiles ähnlich definiert, wie in Shellscripts. Eine einfache Definition sieht beispielsweise folgendermaßen aus:

```
installprefix=/usr/local
```

Der Zugriff auf Variableninhalte erfolgt aber über eine Konstruktion mit Dollarzeichen und geschweiften Klammern, also etwa `${`*`Variablenname`*`}`. Es könnte also folgendermaßen weitergehen:

```
installprefix=/usr/local
installdir=${installprefix}/foo
mandir=${installprefix}/man
confdir=${installdir}/etc
```

Das würde zu folgenden realen Verzeichnisnamen führen:

- installprefix=`/usr/local`
- installdir=`/usr/local/foo`
- mandir=`/usr/local/man`
- confdir=`/usr/local/foo/etc`

Wenn wir jetzt daran Veränderungen vornehmen wollen, so können wir das durch entsprechende Anpassungen an unser System tun. Um beispielsweise dafür zu sorgen, daß das Programm seine Konfigurationsdateien nicht unter `/usr/local/foo/etc` sucht, sondern unter `/etc/foo`, so verändern wir die entsprechende Zeile in

```
confdir=/etc/foo
```

Manchmal finden sich in solchen Variablen auch ganze Listen von Verzeichnissen, die z.B. nach Include-Dateien durchsucht werden sollen. Auch solche Listen können natürlich entsprechend angepasst werden, um beispielsweise mehr Verzeichnisse aufzunehmen. In der Regel werden Listenelemente durch Leerzeichen voneinander getrennt.

Aufruf von make

Wenn alle Veränderungen am Makefile abgeschlossen wurden, so kann das Programm jetzt compiliert werden. Dazu wird einfach der Befehl

```
make
```

eingegeben. Er compiliert jetzt das Programm nach der Regel, die im Makefile `all` genannt wurde. Der Aufruf ist also einfach eine Abkürzung für

```
make all
```

Ist dieser Befehl fehlerfrei abgearbeitet worden, dann ist das Programm jetzt compiliert. Das heißt, es liegen jetzt für jede Quellcodedatei eine Objektdatei vor und es existiert das fertige ausführbare Programm. Damit das Programm auch noch in die vorgesehenen Verzeichnisse kopiert (installiert) wird, haben die meisten Makefiles noch eine Regel, die `install` heisst und entsprechend mit

```
make install
```

aufgerufen wird. Dadurch werden die notwendigen Dateien in die vorgesehenen Zielverzeichnisse kopiert. Um alle Objektdateien wieder zu löschen kann der Befehl

```
make clean
```

benutzt werden, um sicherzustellen, daß bei einem zweiten Compiliervorgang alles erneut übersetzt wird.

Zusammenfassung

Nachdem ein Quellcode-Paket ausgepackt wurde müssen praktisch immer die drei einfachen Befehle

```
./configure
make
make install
```

in dem Verzeichnis ausgeführt werden, in dem der Quellcode liegt. In der Regel ist mindestens für `make install` das root-Recht erforderlich, denn dieser Befehl installiert das Programm ja an Orte im System, an denen niemand außer root Schreibrecht hat.

1.102.4

Verwaltung von Shared Libraries

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Shared Libraries, die von ausführbaren Programmen benötigt werden, zu bestimmen und nötigenfalls zu installieren. Sie sollten ebenfalls fähig sein, anzugeben wo sich die Systembibliotheken befinden.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **ldd**
- **ldconfig**
- `/etc/ld.so.conf`
- `LD_LIBRARY_PATH`

Die Technik der Shared Libraries

Shared Libraries sind Funktionsbibliotheken, die von mehreren Programmen benutzt werden können und die so nur einmal in den Speicher geladen werden müssen, anstatt für jedes Programm extra Platz zu verschwenden. Normalerweise würde ein Programm beim Linkvorgang (dem Vorgang, bei dem alle Objektdateien und die Systemlibraries zur ausführbaren Datei zusammengefügt werden) mit allen Libraries, aus denen das Programm Funktionen benötigt, statisch gelinkt. Das würde aber bedeuten, daß Programme, die gleichzeitig laufen, diese Funktionsbibliotheken - jedes Programm für sich - in den Speicher laden müsste. Die shared library Technik verhindert das. Ein Programm, das geladen wird, überprüft, ob die notwendigen Libraries schon im Arbeitsspeicher liegen. Zun sie das, so läd sich das Programm in den Arbeitsspeicher und benutzt einfach die schon geladenen Libraries. Liegen die benötigten Libraries noch nicht im Arbeitsspeicher, so werden sie zunächst geladen und erst dann läd sich das Programm.

So wird jede benötigte Librarie nur einmal in den Speicher geladen, was gerade in einem System, das viele Programme gleichzeitig geladen hält eine enorme Speichereinsparung bewirkt.

Wenn ein Programm geladen wird, dann muß es eine Instanz geben, die den geschilderten Vorgang startet. Denn das Programm selbst kann das nicht tun, es wird ja erst geladen, wenn die nötigen Libraries auch schon geladen sind. Diese Instanz ist sozusagen der Programmlader

oder der sogenannte *dynamische Linker und Lader*. Der Name dieses Programms ist **ld.so**.

ld.so läd die shared libraries, die ein Programm benötigt, bereitet dann das Programm entsprechend vor und läd es selbst. Die shared libraries, die das zu ladende Programm benötigt, werden in der folgenden Reihenfolge gesucht:

- Alle Pfade, die in der Umgebungsvariable `LD_LIBRARY_PATH` eingetragen sind, werden nach den Libraries durchsucht. `LD_LIBRARY_PATH` ist eine Shellvariable, die eine durch Doppelpunkte getrennte Liste von Verzeichnissen enthält.
- Die Datei `/etc/ld.so.cache` enthält eine binäre Liste aller Librarie-Kandidaten, die schon vorher in den genannten Verzeichnissen gefunden wurden. (siehe weiter unten bei **ldconfig**.)
- Die Verzeichnisse `/usr/lib` und `/lib` werden durchsucht.

Ein Linux-Programm wird also von einem speziellen Programmlader (**ld.so**) geladen, der die notwendigen Libraries gleich mitläd. Aus diesem Grund ist es nötig, daß jedes Programm, das installiert wird, auch die entsprechenden Libraries mitinstalliert bzw. überprüft, ob sie schon installiert sind. Die Verwaltung dieser Libraries obliegt der Systemverwaltung.

Welches Programm braucht welche Library?

Um herauszufinden, welche Shared Libraries ein bestimmtes Programm benutzt, gibt es das kleine Programm ldd. Dieses Programm gibt eine Liste aller Libraries zurück, die das Programm benötigt, daß als Parameter **ldd** mitgegeben wurde. Um z.B. herauszubekommen, welche Libraries das Programm **ls** benötigt, schreiben wir

```
ldd /bin/ls
```

Das zu untersuchende Programm muß mit vollem Pfad angegeben werden. Die Ausgabe wäre für unser Beispiel dann etwas in der Art:

```
librt.so.1 => /lib/librt.so.1 (0x40022000)
libc.so.6 => /lib/libc.so.6 (0x40033000)
libpthread.so.0 => /lib/libpthread.so.0 (0x4014e000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Daraus geht hervor, daß das Programm **ls** vom Linker/Loader `/lib/ld-linux.so.2` gestartet werden will und die Libraries `librt.so.1, libc.so.6` und `libpthread.so.0` benötigt. In der rechten Spalte sind die Fundorte der Libraries im Dateisystem zu lesen.

Wenn ein Programm neu installiert wird, so kann mit Hilfe dieses Befehls herausgefunden werden, ob alle Libraries für das Programm existieren oder ob einige nachinstalliert werden müssen.

Libraries installieren und aktualisieren

Wenn neue Libraries installiert werden sollen, dann stellt sich die Frage, wohin. Zunächst einmal bieten sich die Verzeichnisse `/lib`, `/usr/lib` und `/usr/local/lib` an. Sollen aber neue Verzeichnisse angelegt werden, die Libraries enthalten, dann muß das der internen Verwaltung der Libraries mitgeteilt werden.

Die interne Verwaltung der Libraries wird durch den Aufruf des Programms **ldconfig** vollzogen. Dieses Programm wartet den Library Cache und erstellt automatisch die notwendigen symbolischen Links auf Libraries. Der Librarie-Cache liegt in der Datei `/etc/ld.so.cache` und enthält eine binär codierte Liste aller dem System bekannten Libraries.

Damit **ldconfig** diese Datei erstellen kann und auch neu hinzugekommene Libraries dort aufgenommen werden, muß **ldconfig** wissen, welche Verzeichnisse nach Libraries durchsucht werden sollen. **ldconfig** durchsucht zunächst die beiden Verzeichnisse `/usr/lib` und `/lib`, danach alle Verzeichnisse, die in der Datei `/etc/ld.so.conf` aufgelistet sind.

Wenn also ein neues Programm **foo** installiert wird, das seine Shared Libraries im Verzeichnis `/usr/local/foo/lib` ablegt, so müssen wir nur dieses Verzeichnis in die Datei `/etc/ld.so.conf` aufnehmen. Nach der Installation neuer Libraries und/oder der Neuaufnahme von Pfaden in der Datei `/etc/ld.so.conf` muß das Programm **ldconfig** ausgeführt werden, bevor die neuen Libraries verwendet werden können. Erst nach dem Aufruf von **ldconfig** stehen sie ja in der Datei `/etc/ld.so.cache` und sind somit dem Linker/Loader bekannt.

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.102.5

Verwendung des Debian Paketmanagements

Beschreibung: Prüfungskandidaten sollten in der Lage sein, mit dem Debian Paketmanagement umzugehen. Dieses Lernziel beinhaltet das Benutzen von Kommandozeilen- und interaktiver Werkzeuge zum Installieren, Updaten oder Deinstallieren von Paketen sowie das Auffinden von Paketen, die spezifische Dateien oder Software enthalten (installierte bzw. nicht installierte Pakete). Ebenfalls enthalten ist das Abfragen von Informationen wie Version, Inhalt, Abhängigkeiten, Paketintegrität und Installationsstatus (ob installiert oder nicht) von Paketen.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **unpack**
- **configure**
- `/etc/dpkg/dpkg.cfg`
- `/var/lib/dpkg/*`
- `/etc/apt/apt.conf`
- `/etc/apt/sources.list`
- **dpkg**
- **dselect**
- **dpkg-reconfigure**
- **apt-get**
- **alien**

Es gibt im Linux Bereich zwei große Modelle einer Paketverwaltung für zu installierende Programmpakete. Nachdem sich das alte `tgz`-Format für die Verwendung bei binären (vorcompilierten) Paketen als unzulänglich erwiesen hatte, weil es keinerlei Mechanismen zur Deinstallation aufwies, begannen zwei verschiedene Distributionen, eigene Paketverwaltungen aufzubauen. Debian und RedHat.

Inzwischen haben nahezu alle Distributionen das RedHat-Paketmanagement übernommen, nur Debian beharrt auf sein eigenes Modell. Und das nicht zu Unrecht, ist es doch gerade die Paketverwaltung von Debian, die schlichtweg als genial bezeichnet werden muß.

Als distributionsübergreifende Zertifizierung fordert LPI das Wissen um beide Formate. Beide Formate werden auch tatsächlich abgefragt und das nicht zu knapp. In diesem Kapitel wird die Debian-Verwaltung besprochen, im nächsten kommt dann die RedHat-Technik zur Sprache.

Das grundlegende Prinzip der Debian Paketverwaltung

Debian Pakete enden mit der Endung `.deb`. Sie haben ein einheitliches Namensschema, das sich folgendermaßen zusammensetzt:

> *Programmname_Versionsnummer_Architektur*`.deb`

Wobei *Programmname* der Name des Programmes (oder des Paketes) ist, unter dem es später dann auch angesprochen werden kann, *Versionsnummer* ist die Version des Programms (oder Paketes) und *Architektur* bezeichnet die Hardwarearchitektur, für die dieses Paket gedacht ist, also etwa `i386` für die PC-Architektur.

Ein Debian Paket besteht aus einem `ar`-Archiv, das wiederum zwei komprimierte `tar`-Archive und eine Versionsdatei beinhaltet. Das erste `tar`-Archiv (`data.tar.gz`) enthält die zu installierenden Dateien, das zweite (`control.tar.gz`) enthält die Metainformationen über das Paket, die Scripts, die zum Installieren und Deinstallieren benötigt werden sowie eine Prüfsumme.

Jedes Debian-Paket enthält vier Scripte, je eines das vor und nach der Installation bzw. vor und nach der Deinstallation abgearbeitet wird.

Wird ein Debian-Paket installiert, so werden die Informationen über dieses installierte Paket in mehrere Dateien im Verzeichnis `/var/lib/dpkg` abgelegt, so daß jederzeit eine Überprüfung der bereits installierten Pakete und Dateien stattfinden kann. Die Informationen werden folgendermaßen aufgeteilt:

- In der Datei `/var/lib/dpkg/available` werden die Paketinformationen aller installierten Pakete abgelegt.
- In der Datei `/var/lib/dpkg/status` werden die Informationen über den Status der Installation abgelegt, so daß unterschieden werden kann zwischen korrekt oder nur teilweise installierten Paketen.
- Im Verzeichnis `/var/lib/dpkg/info` liegen zu jedem installierten Paket die vier Scripte (`*.preinst, *.postinst, *.prerm, *.postrm`), eine Liste aller enthaltener Dateien (`*.list`), die md5-Prüfsummendatei (`*.md5sums`) und evt. noch andere Informationen wie die zur Verfügung gestellten Libraries (`*.shlibs`).

Mit Hilfe dieser Informationen sind sowohl Installation, als auch Deinstallation von Paketen sehr sicher und umfassend möglich. Man kann das Verzeichnis `/var/lib/dpkg` also als Systemdatenbank der installierten Pakete betrachten. In der Regel sind keine manuellen Zugriffe auf dieses Verzeichnis notwendig, alle Zugriffe werden mit den Hilfsprogrammen erledigt, die im Folgenden näher besprochen werden.

Das dpkg-Programm

Das Programm **dpkg** ist sozusagen das Low-Level Paketverwaltungsprogramm. Mit ihm ist es möglich, direkt bestimmte Pakete zu installieren uder zu deinstallieren. Noch eine Stufe weiter unten liegt das Programm **dpkg-deb**, das die reine Archivverwaltung (Packen/Entpacken) erledigt, aber niemals manuell aufgerufen wird.

Das **dpkg** Programm muß nur dann manuell aufgerufen werden, wenn direkt vorliegende `.deb`-Pakete installiert werden sollen. Später werden wir das Programm **apt-get** kennenlernen, das sich auch um den Bezug der Pakete kümmert, dann aber selbstständig **dpkg** aufruft, um Pakete zu installieren bzw. zu deinstallieren. Ist jedoch ein Paket manuell z.B. aus dem Internet besorgt worden, dann benötigen wir **dpkg**.

Pakete installieren mit dpkg

Um ein Paket mit **dpkg** zu installieren wird folgender Befehl ausgeführt:

```
dpkg [Optionen] -i | --install Paketdatei.deb
```

(-i | --install bedeutet entweder -i oder --install). Wenn als Option `-R` oder `--recursive` angegeben wurde, so steht statt eines Paketnamens ein Verzeichnisname. Alle Debianpakete des genannten Verzeichnisses werden dann installiert.

Die Installation besteht aus folgenden Einzelschritten:

1. Die Kontrolldateien des neuen Paketes werden entpackt.
2. Wenn eine ältere Version des selben Paketes bereits installiert war, wird das PreRemove-Script der älteren Version ausgeführt.
3. Das PreInst-Script des neuen Paketes wird ausgeführt.
4. Die Dateien des neuen Paketes werden entpackt und gleichzeitig werden die Dateien einer eventuell existierenden älteren Version gesichert, so daß im Falle eines Fehlers die Installation des neuen Paketes rückgängig gemacht werden kann.
5. Wenn eine ältere Version existierte, dann wird das PostRemove-Script der älteren Version ausgeführt.
6. Das neue Paket wird durch das Abarbeiten des PostInstall-Scripts konfiguriert.

Pakete deinstallieren mit dpkg

Wenn ein installiertes Paket entfernt werden soll. dann werden zwei Möglichkeiten angeboten. Entweder werden zwar alle Dateien entfernt, aber die Konfigurationsdateien des Paketes bleiben installiert, oder es werden wirklich alle Dateien, auch die Konfigurationsdateien entfernt.

Zum normalen Deinstallieren (ohne Entfernung der Konfigurationsdateien) wird folgender Befehl benutzt:

```
dpkg [Optionen] -r | --remove Paketname
```

Um wirklich alles zu entfernen benutzt man

```
dpkg [Optionen] -P | --purge Paketname
```

Paketname ist hier der Name des Paketes, nicht der Name der Paketdatei. Wurde die Datei `foo_1.0.23_i386.deb` installiert, so ist der Paketname einfach nur `foo`.

Das Entfernen der Pakete geht in folgenden Schritten vor sich:

1. Das PreRemove-Script des Paketes wird abgearbeitet.
2. Die zu entfernenden Dateien werden gelöscht.
3. Das PostRemove-Script wird abgearbeitet.

Informationen über installierte Pakete abfragen

Um Informationen über ein bestimmtes installiertes Paket zu erhalten, wird der Befehl

```
dpkg  -p|--print-avail Paketname
```

ausgeführt. **dpkg** liesst daraufhin die Informationen aus der Datei `/var/lib/dpkg/available` und gibt sie aus.

Auflisten der installierten Pakete

Um eine Liste aller oder bestimmter installierter Pakete zu bekommen, wird der Befehl

```
dpkg -l | --list [Paketnamensmuster]
```

ausgeführt. Wird kein Namensmuster angegeben, so werden alle installierten Paketnamen ausgegeben, ansonsten nur die Pakete, deren Namen auf das angegebene Muster passen. Als Muster werden die üblichen Shell-Wildcards benutzt. Zu beachten ist, daß die Muster in Anführungszeichen gesetzt werden sollten, um die Shell daran zu hindern, sie zu interpretieren.

Auflisten der Dateien installierter Pakete

Um alle Dateien eines bestimmten Paketes aufzulisten, existiert der Befehl

```
dpkg -L | --listfiles Paketname
```

Die Dateien, die von Installationsscripts angelegt wurden, werden hier nicht angezeigt.

Abfrage des Installationsstatus

Debian-Pakete, die installiert werden, bekommen einen bestimmten Status, der angibt, wie weit die Installation fortgeschritten ist. Wenn eine Installation aus welchen Gründen auch immer, nicht vollständig abgeschlossen wurde, so kann das später aus dem Status entnommen werden. Folgende Stati sind gültig:

installed
: Das Paket ist vollständig installiert

half-installed
: Die Installation wurde nicht vollständig ausgeführt

not-installed
: Das Paket ist nicht installiert

unpacked
: Das Paket ist zwar ausgepackt, aber nicht konfiguriert

half-configured
: Das Paket ist ausgepackt und die Konfiguration wurde begonnen, aber nicht korrekt abgeschlossen

config-files
: Nur die Konfigurationsdateien des Paketes existieren auf dem System (typischerweise nach einer Deinstallation mit -R)

Um den Status eines Paketes abzufragen wird der Befehl

```
dpkg -s | --status Paketname
```

ausgeführt. Die Informationen werden der Datei `/var/lib/dpkg/status` entnommen.

Aus welchem Paket stammt eine bestimmte Datei?

Mit dem Befehl

```
dpkg -S | --search Dateinamesmuster
```

werden die installierten Dateien aus `var/lib/dpkg/info` durchsucht und alle Paketnamen zurückgegeben, die entsprechende Dateien enthalten. Zur Musterbildung stehen alle Shell-Wildcards zur Verfügung.

Optionen für dpkg

Vor jedem der besprochenen Befehle können verschiedene Optionen gesetzt werden, die den Ablauf des Befehls modifizieren können. Diese Optionen beziehen sich z.B. auf den Umgang mit Abhängigkeiten, oder ab wievielen Fehlern sich **dpkg** weigert, weiterzuarbeiten. Diese Optionen können entweder direkt an der Kommandozeile eingegeben werden, oder in die Datei `/etc/dpkg/dpkg.cfg` eingetragen werden. In letzterem Fall, werden die Optionen für jeden **dpkg** Befehl angewandt. Wichtige Optionen sind:

--abort-after=*Zahl*

Gibt an, nach wievielen Fehlern **dpkg** beendet werden muß. Voreingestellt ist 50.

-B|--auto-deconfigure

Wenn ein Paket deinstalliert wird, so ist es möglich, daß ein anderes installiertes Paket dieses Paket gebraucht hätte. Mit dieser Option werden automatisch alle anderen Pakete auch deinstalliert, die das zu installierende Paket benötigt hätten.

--ignore-depends=*Paket*

Abhängigkeiten für das angegebene Paket werden ignoriert. Es werden solche Abhängigkeiten zwar überprüft, aber es werden nur Warnungen statt Fehlermeldungen angezeigt, wenn nicht erfüllte Abhängigkeiten auftreten.

--refuse-downgrade | -G

Ein Paket, von dem eine neuere Version bereits installiert ist, wird nicht installiert

Eine vollständige Liste aller denkbaren Optionen sind der Handbuchseite zu entnehmen.

dpkg-reconfigure

Wenn ein bereits installiertes Paket erneut konfiguriert werden soll, so kann das durch den Aufruf von

```
dpkg-reconfigure Paketname
```

vorgenommen werden. Das hat aber nur dann eine tatsächliche Bedeutung, wenn ein Paket ein `debconf`-Script enthält.

dselect

Das Programm **dselect** ist ein menügeführtes Frontend für **dpkg**. Es übernimmt aber nicht nur die Aufgabe, Pakete zu installieren oder zu deinstallieren, sondern auch die Verwaltung der zur Verügung stehenen Pakete und der gegenseitigen Abhängigkeiten.

dselect bietet verschiedene Methoden an, um die zur Verfügung stehenden Pakete zu managen. Dabei geht es hauptsächlich darum, woher die Debian-Pakete bezogen werden sollen. Folgende Methoden werden angeboten:

cdrom

Installiert von einer Debian-CDROM. Die CDROM kann, muß aber nicht gemountet sein und sollte ein ISO9660 Dateisystem beinhalten.

nfs

Installiert über einen (noch nicht gemounteten) NFS-Server. Der Server muß die Paketbeschreibungsdateien (`Packages.gz`) jeder Distributionsabteilung (stable, contrib und non-free) zur Verfügung stellen und natürlich die entsprechenden `.deb` Dateien.

harddisk

Installiert von einer noch nicht gemounteten Festplattenpartition. Diese Partition muß die selben Elemente wie ein NFS-Server enthalten.

mounted

Installation von einem bereits gemounteten Dateisystem. Dabei kann es sich entweder um eine gemountete Festplattenpartition oder um eine gemounteten NFS-Freigabe handeln. Zum Inhalt gilt das selbe wie bei NFS-Server und Festplatte.

floppy

Installation über einen Stapel Disketten, von dem die erste Disk die Paketinformationen enthalten sollte. Veraltert.

apt

Benutzt die neue Methode, die weiter unten bei **apt-get** beschrieben wird. Hier kann die Installationsquelle entweder über ein Netz (ftp/http) oder über das Dateisystem (file) angesprochen werden.

Das **dselect** Programm wird normalerweise ohne weitere Parameter aufgerufen. Es bietet menügeführt folgende Möglichkeiten:

Zugriff (access)

Auswahl der Zugriffsmethode (siehe oben).

Erneuern (update)

Erneuert die Liste der verfügbaren Pakete, wenn möglich.

Auswählen (select)

Menügeführte Auswahl aller zu installierender oder zu entfernender Pakete.

Install

Installiert die Pakete, die angewählt wurden.

Konfig

Konfiguriert Pakete, die bei der letzten Installation nicht vollständig konfiguriert wurden.

Löschen (remove)

Entfernt die zum Löschen markierten Pakete.

Als menügeführtes Programm ist die Bedienung von **dselect** zwar manchmal gewöhnungsbedürftig, jedoch nicht weiter schwierig.

apt-get

apt-get ist das Kommandozeilenprogramm, das die modernste der Zugriffsmethoden auf Debian-Pakete steuert. Damit ist es möglich, die Installation und verschiedene andere Aufgaben der Paketverwaltung über ein einfaches Kommando auszuführen. Das Programm speichert seine Konfiguration in der Datei `/etc/apt/apt.conf` oder bei den moderneren Versionen in mehreren Verzeichnissen unter `/etc/apt/apt.conf.d`.

Die Quellen, von denen **apt-get** seine Debian-Pakete bezieht, werden in der Datei `/etc/apt/sources.list` angegeben. Diese Datei kann entweder von Hand erstellt/manipuliert werden oder über das Programm **dselect** im Menüpunkt *Zugriff*.

apt-get wird immer in einer der folgenden Formen aufgerufen:

apt-get upgrade

Holt die neuesten Paketbeschreibungen der in `/etc/apt/sources.list` angegebenen Installationsquellen.

apt-get install *Paketname*

Installiert das angegebene Paket von den eingestellten Installationsquellen. Das Paket und alle weiteren notwendigen Pakete die von ihm erfordert werden, wird vollständig installiert (mit dpkg -i).

apt-get remove *Paketname*

Das angegebene Paket wird deinstalliert.

apt-get source *Paketname*

Das Quellcode-Paket des angegebenen Paketes wird installiert.

Seine wahre Stärke spielt **apt-get** aus, wenn als Quellen offizielle Debian-Server im Internet angegeben wurden und der Paketname des zu installierenden Paketes bekannt ist. Wird z.B. festgestellt, daß auf einem System das Programm **foo** nicht installiert ist, so kann durch die Angabe des Befehls

```
apt-get install foo
```

die Installation des Programms ausgeführt werden, ohne lange ein menügeführtes Programm aufzurufen. Es existieren inzwischen auch mehrere Frontends für **apt-get**. Für Textterminals gibt es **aptitude** und für graphische Terminals kann **gnome-apt** verwendet werden.

alien

Debian ist die Distribution, die die meisten Pakete von allen bekannten Distributionen anbietet. Falls trotzdem einmal ein Paket nicht im `.deb` Format vorliegen sollte, man also gezwungen ist, auf ein `.rpm`-Paket zurückzugreifen, gibt es speziell dafür das Programm **alien**.

alien konvertiert verschiedene Paketformate in jeweils andere um. Das Programm kann mit folgenden Formaten umgehen:

- RedHat (rpm)
- Debian (deb)
- Stampede (slp)
- Slackware (tgz)
- Solaris (pkg)

Die Aufrufform ist sehr einfach. **alien** erkennt das Format einer angegebenen Paketdatei automatisch und konvertiert es dann in das angegebene Format. Das gewünschte Ausgabeformat wird durch die folgenden Parameter angegeben:

--to-deb
: Aus dem angegebenen Paket wird ein Debian-Paket erstellt

--to-rpm
: Aus dem angegebenen Paket wird ein RedHat-Paket erstellt

--to-tgz
: Aus dem angegebenen Paket wird ein Slackware-Paket erstellt

--to-slp
: Aus dem angegebenen Paket wird ein Stampede-Paket erstellt

Wenn also das Paket `foo-1.2.34.i386.rpm` existiert, und daraus ein Debian-Paket erstellt werden soll, so genügt der Befehl

```
alien --to-deb foo-1.2.34.i386.rpm
```

und **alien** erzeugt daraus das Paket `foo_1.2.34_i386.deb`

Jetzt kann dieses neu erstellte Paket mit **dpkg -i** installiert werden und wird so nahtlos in die Debian-Paketverwaltung integriert.

1.102.6

Verwendung des Red Hat Package Managers (RPM)

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Paketmanagement auf Linuxdistributionen, die RPM-Pakete für die Paketverwaltung verwenden, durchzuführen. Dieses Lernziel beinhaltet die (Neu-)Installation, das Update und das Entfernen von Paketen sowie das Abfragen von Status- und Versionsinformationen. Ebenfalls enthalten ist das Abfragen von Paketinformationen wie Abhängigkeiten, Integrität und Signaturen. Kandidaten sollten auch in der Lage sein, zu bestimmen, welche Dateien von einem Paket zur Verfügung gestellt werden, und das Paket zu finden, dem eine bestimmte Datei entstammt.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/rpmrc`
- `/usr/lib/rpm/*`
- **rpm**
- **grep**

Die meisten heutigen Linux-Distributionen haben das Paketformat von RedHat übernommen, weil es eine sehr weitreichende Verwaltung der installierten Software ermöglicht. Dazu zählen unter anderem die korrekte Installation und Deinstallation, die Informationen über installierte Programme und die Verifikation, daß bestimmte Pakete tatsächlich das enthalten, was sie vorgeben. Die gesamte Verwaltung der Red-Hat Pakete wird durch ein Programm realisiert, das den Namen **rpm** trägt. Auch Nicht-RedHat Distributionen unterstützen dieses Programm.

Das Prinzip der RPM-Verwaltung

Jedes RedHat Paket hat einen Dateinamen, der auf `.rpm` endet. Die Dateinamensstruktur ähnelt der von Debian:

Programmname-*Versionsnummer*.*Architektur*.`rpm`

Wobei *Programmname* der Name des Programmes (oder des Paketes) ist, unter dem es später dann auch angesprochen werden kann, *Versionsnummer* ist die Version des Programms (oder Paketes) und *Architektur* bezeichnet die Hardwarearchitektur, für die dieses Paket

gedacht ist, also etwa i386 für die PC-Architektur.

Jedes RPM-Paket enthält neben den zu installierenden Dateien wie Debian-Pakete vier Scripts, die jeweils vor und nach der Installation und Deinstallation ausgeführt werden. Zusätzlich existieren Informationen über das Paket und die Abhängigkeiten, die die darin enthaltenen Programme besitzen.

Im Verzeichnis `/usr/lib/rpm` liegen die verschiedenen Hilfsmittel, die das **rpm**-Programm benötigt und unter `/var/lib/rpm` finden sich die verschiedenen Datenbanken über die installierten Pakete. Im Gegensatz zu Debian werden diese Datenbanken in einem proprietären Format abgespeichert, das nur mit dem **rpm**-Programm zugänglich ist.

Installieren und deinstallieren von Paketen

Eine der Hauptaufgaben des **rpm**-Programmes ist das Installieren und Deinstallieren von Paketen. Dazu dienen die folgenden Befehle.

Installieren eines Paketes

Um eine bestehende `.rpm`-Datei zu installieren, wird folgender Befehl verwendet:

```
rpm -i | --install [Optionen] Paketdatei
```

das `-i | --install` bedeutet, daß entweder ein `-i` oder ein `--install` angegeben wird. Mögliche Optionen sind:

--nodeps
: Die Überprüfung von Abhängigkeiten wird abgeschalten.

--noscripts
: Es werden keine pre- oder postinstall Scripts abgearbeitet.

--test
: Die Installation wird nur simuliert, keine Dateien werden installiert.

--excludedocs
: Die im Paket enthaltenen Dokumentationen werden nicht mitinstalliert.

--replacepkgs
: Die Installation wird durchgeführt, auch wenn schon Teile installiert sind.

--replacefiles
: Die Installation wird auch durchgeführt, auch wenn dabei Dateien überschrieben werden, die von anderen Paketen stammen.

--oldpackage

Erlaubt ein Upgrade von älteren Versionen als die bereits installierten.

--force

Entspricht der Kombination von `--replacepkgs --replacefiles --oldpackage`.

Vorsicht ist geboten, wenn die Überprüfung der Abhängigkeiten abgeschalten wird, oder das zwingende Installieren von Paketen aktiviert wird. Das kann zu Inkonsistenzen des Systems führen und sollte nur verwendet werden, wenn man genau weiß, was man tut.

Deinstallieren eines Paketes

Ein Paket wird mit folgendem Befehl deinstalliert:

```
rpm -e | --uninstall [Optionen] Paketname
```

Auch für die Deinstallation gibt es einige Wichtige Optionen:

--nodeps

Die Überprüfung von Abhängigkeiten wird abgeschalten.

--noscripts

Es werden keine pre- oder postuninstall Scripts abgearbeitet.

--test

Die Deinstallation wird nur simuliert, keine Dateien werden gelöscht.

--allmatches

Deinstalliert alle Pakete, auf die das Namensmuster zutrifft. In diesem Fall ist *Paketname* also ein Muster und nicht ein Paketnamen.

Upgrade eines Paketes

Wenn eine neue Version eines bereits installierten Programmes aufgespielt werden soll, so kommen zwei verschiedene Techniken zum Einsatz:

```
rpm -U|--upgrade [Optionen] Paketdatei
```

Als Optionen kommen exakt die selben Optionen wie beim Installieren zum Einsatz. Der Befehl entspricht im Übrigen genau dem, der zum Installieren benutzt wird, nur daß eventuell vorher existierende Versionen des Paketes vor der Installation entfernt werden.

```
rpm -F|--freshen [Optionen] Paketdatei
```

Dieser Befehl, der auch wieder die selben Optionen wie beim Installieren kennt, installiert die angegebene Paketdatei nur, wenn eine ältere Version des selben Programms vorher schon installiert war. Dann verhält er sich wie der vorige (-U)

Abfage von Paketinformationen

Die generelle Form, wie Paketinformationen abgefragt werden, beginnt immer mit einem `-q` oder `--query`. Dem folgen bestimmte andere Befehle, die dann die eigentliche Abfrage stellen.

Wenn ein normaler Querry-Befehl angegeben wird, so bezieht er sich auf schon installierte Pakete. **rpm** durchsucht also nicht eine bestimmte Paketdatei, sondern die Datenbanken, in denen die Informationen über installierte Pakete liegen. Soll aber Information über eine Paketdatei erfragt werden, die noch nicht installiert ist, so muß die Option `-p` angegeben werden. Statt der Angabe eines Paketnamens ist dann die Angabe einer Paketdatei notwendig.

Abfrage von Versionsinformationen

Um die Versionsnummer eines installierten Paketes abzufragen wird einer der folgende Befehle benutzt:

```
rpm -qi Paketname
rpm --query --info Paketname
```

Soll stattdessen die Versionsinformation eines nicht installierten RPM-Paketes erfragt werden (was eigentlich unnötig ist, da diese Information im Dateinamen schon vorliegt) kommt einer der folgenden Befehle zum Einsatz

```
rpm -qi -p Paketdatei
rpm --query --info --package Paketdatei
```

Natürlich können die langen und kurzen Optionen auch durcheinander benutzt werden wie `--query -i` oder `-q --info`.

Neben der reinen Versionsinformation werden auch noch andere Paketinformationen ausgegeben. Wenn tatsächlich nur die Information über die Version abgefragt werden soll, kann auch der eingebaute Variablensubstitutionsmechanismus benutzt werden:

```
rpm -q Paketname --queryformat "%{VERSION}\n"
```

Auf diese Art und Weise können beliebige Informationen in einem beliebigen Format ausgegeben werden. Eine komplette Liste aller so benutzbaren Variablensubstitutionen ist mit dem Befehl

```
rpm --querytags
```

einsehbar. In der gegenwärtigen Version sind es 98 verschiedene Typen wie etwa NAME, VERSION, RELEASE, DESCRIPTION, SIZE, VENDOR, FILENAMES, ...

Auflisten aller Dateien eines Paketes

Oft soll überprüft werden, welche Dateien ein bestimmtes Paket geliefert hat. Mit dem Befehl

```
rpm -ql Paketname
```

wird eine Liste aller Dateien des angegebenen Paketes ausgegeben. Statt `-l` kann auch `--list` verwendet werden. Soll stattdessen wieder eine noch nicht installierte Paketdatei untersucht werden, so gilt wieder der Parameter `-p`

```
rpm -ql -p Paketdatei
```

Die Ausgabe erfolgt auf die Standard-Ausgabe, kann also problemlos mit Tools wie grep weiterverarbeitet werden.

Auflisten der Dokumentationsdateien eines Paketes

Um nur die Dateien eines Paketes aufzulisten, die reine Dokumentationsdateien (Handbuchseiten, READMEs, ...) sind, existiert der Befehl

```
rpm -qd Paketname
```

oder wieder mit langen Optionen `--docfiles` statt dem `d`. Auch dieser Befehl ist mit `-p` auf eine Paketdatei anwendbar.

Auflisten der Konfigurationsdateien eines Paketes

Analog zur Ausgabe der Dokumentationsdateien gibt es eine Möglichkeit nur die Konfigurationsdateien auszugeben:

```
rpm -qc Paketname
```

oder wieder mit langen Optionen `--configfiles` statt dem `c`. Auch dieser Befehl ist mit `-p` auf eine Paketdatei anwendbar.

Auflisten der Scriptdateien eines Paketes

Manchmal ist es auch interessant, herauszufinden, was die Installatiopsscripts eines Paketes eigentlich tun. Hierfür gibt es den Befehl

```
rpm -q --scripts Paketname
```

Er listet hintereinander alle vier Scripts auf die Standard-Ausgabe, sollte also entweder in eine Datei umgeleitet oder an less weitergepiped

werden.

Gerade dieser Befehl ist natürlich wichtig für noch nicht installierte Paketdateien, um vorher sicherzustellen, daß das Script nichts unerwünschtes macht. Also steht uns auch hier wieder die Option `-p` zur Verfügung.

Abfragen über installierte Pakete

Die nächsten zwei Befehle beziehen sich nur auf bereits installierte Pakete, sind aber auch Query-Formen des **rpm** Programms.

Auflisten aller installierten Pakete

Wenn einmal nicht klar ist, welche Pakete bereits installiert sind, so kann **rpm** auch eine Liste aller installierten Pakete ausgeben. Das geschieht mit dem Befehl

```
rpm -qa
```

oder

```
rpm -q --all
```

rpm verwaltet auch bestimmte Gruppen, so daß auch Abfragen möglich sind, die alle installierten Pakete einer Gruppe ausgeben:

```
rpm -qg Gruppe
```

oder

```
rpm -q --group Gruppe
```

Diese Ausgaben können natürlich sehr umfangreich sein, können aber wiederum z.B. mit **grep** durchsucht werden. Wenn z.B. erforscht werden soll, welche Pakete installiert sind, die etwas mit dem Editor **joe** zu tun haben, können wir einfach schreiben

```
rpm -qa | grep joe
```

Die Ausgabe wäre dann z.B.:

```
joe-2.8-154
```

Daraus können wir jetzt weitere Informationen gewinnen, mit einem der oben genannten Abfragemöglichkeiten.

Aus welchem Paket stammt eine bestimmte Datei?

Oft ist einfach die Frage, aus welchem Paket eine bestimmte Datei stammt. Auch diese Frage lässt sich mit **rpm** lösen.

```
rpm -qf Dateiname
```

oder

```
rpm -q --file Dateiname
```

rpm durchsucht jetzt seine Datenbanken nach diesem Dateinamen und gibt uns das Paket zurück, aus dem diese Datei stammt.

Abhängigkeiten

Oft benötigen bestimmte Pakete andere Pakete, um zu funktionieren. Diese Abhängigkeiten sind auch mit **rpm** herausfindbar.

Welche installierten Pakete benützt ein bestimmtes Paket?

Wenn ein Paket entfernt werden soll oder eine Datei gelöscht werden soll, dann stellt sich die Frage, ob das Paket bzw. die Datei nicht noch von einem anderen Paket benötigt werden würde. Mit dem Befehl

```
rpm -q --whatrequires Paket
```

oder

```
rpm -q --whatrequires Datei
```

kann herausgefunden werden, welches installierte Paket das angegebene Paket oder die angegebene Datei benötigt.

Welche Pakete werden von einem bestimmten Paket benötigt?

Umgekehrt kann es auch interessant sein, welche Libraries und andere Pakete von einem bestimmten Paket benötigt werden. Mit

```
rpm -qR Paketname
```

oder

```
rpm -q --requires Paketname
```

wird eine Liste aller Libraries und Pakete ausgegeben, die von dem angegebenen Paket benötigt werden.

Sicherheitsabfragen

Wenn Pakete installiert werden sollen, dann kann es interessant sein, ob diese Pakete wirklich aus der Quelle kommen, aus der sie vorgeben zu kommen. Es wäre ja auch denkbar, daß jemand ein Paket mit einem Trojanischen Pferd einschmuggeln will, und dazu ein echtes Paket mit seinem Paket austauscht. Auf der anderen Seite kann der Inhalt eines Paketes auch durch Übertragungsfehler unbeabsichtigt verfälscht werden. Gegen beide dieser Gefahren kann mit **rpm** vorgegangen werden.

Verifizierung der Paketintegrität eines Paketes

Eine Paketverifizierung vergleicht die Informationen, über die installierten Dateien eines Paketes mit den Informationen über diese Dateien, die in den Metadaten des Paketes in der Paketdatenbank gespeichert sind. Neben anderen Dingen vergleicht die Verifizierung die Größe, dei MD5-Prüfsumme, Zugriffsrechte, Typ, Eigentümer und Zugriffsrechte jeder Datei. Jede Diskrepanz wird dargestellt.

```
rpm  -V |--verify Paketname
```

Das Format der Ausgabe ist eine acht Zeichen lange Zeichenkette, ein mögliches "**c**" (für Konfigurationsdatei) und der Dateiname. Jedes der acht Zeichen bezeichnet das Ergebnis eines Vergleichs zwischen den Attributen einer Datei und denen, die in der Paketdatenbank gespeichert sind. Ein einfacher Punkt (.) bedeutet, daß der Test durchgelaufen ist, während ein Fragezeichen (?) anzeigt, daß der Test nicht durchgeführt werden konnte (z.B. weil Zugriffsrechte zum Lesen gefehlt hatten). In jedem anderen Fall repräsentieren die Buchstaben Fehler des jeweiligen Verifizierungstests:

S
: (Size) Die Dateigröße stimmt nicht

M
: (Mode) Der Zugriffsmodus stimmt nicht

5
: Die MD5 Prüfsumme stimmt nicht

D
: Die Major/Minor Numer der Gerätedatei stimmt nicht

L
: ReadLink(2) Systemaufruffehler

U

Eigentümer stimmt nicht

G

Gruppenzugehörigkeit stimmt nicht

T

Die Zeitmarke der mtime stimmt nicht

Abfrage der PGP/GPG Signatur eines Paketes

Abschließend bietet **rpm** noch die Möglichkeit, daß ein Paket von seinem Autor mit einer PGP oder GPG Signatur (einer Art elektronischer Unterschrift) versehen wird. Diese Signatur kann mit folgendem Befehl überprüft werden:

```
rpm --checksig Paketdatei
```

Damit diese Überprüfung funktioniert muß eine korrekte Installation von PGP oder GPG vorliegen und die Konfiguration von **rpm** muß dies berücksichtigen. Die Darstellung dieser Technik würde den Rahmen dieser Dokumentation sprengen.

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

Bezeichnung

tar - (tape archiver) verwaltet Dateiarchive

Syntax

tar [**-Acdrtux**] [**+delete**] [**-b** *N*] [**-BgGhiklmMoOpPPsSvwWz**] [**-C** *Verzeichnis*] [**-f** *Datei*] [**-F** *Datei*] [**-K** *Datei*] [**-L** *Länge*] [**-N** *Datum*] [**-T** *Datei*] [**-V** *Name*] [**-X** *Datei*] [**0**-**7**] [{**lmh**}]

Beschreibung

tar ist ursprünglich ein Tool zur Verwaltung von Bandarchiven. Das **GNU tar** kann aber auch auf rohen Disketten oder in normalen Dateien Archive im **tar** Format anlegen und verwalten. Normalerweise werden Archive mit **tar** nicht komprimiert. Das GNU **tar** kann aber die Ein- und Ausgabe durch einen Kompressor leiten. Die neuen Versionen (ab 1.11.2) unterstützen sowohl **compress** als auch **gzip**.

Linux kann alle SCSI Streamer und mit Kernelpatches auch QIC-02 Streamer betreiben. Wenn keine Datei und kein Gerät angegeben ist, versucht **tar** auf einen Streamer zuzugreifen. Je nach Konfiguration ist das meist /dev/tape oder /dev/rmt0.

Optionen

-A

hängt ein komplettes Archiv an ein anderes Archiv an

-c

erzeugt ein neues Archiv

-d

vergleicht das Archiv mit dem Dateisystem

-D *Datei*

löscht die *Datei* aus dem Archiv (für Bandarchive nicht empfehlenswert)

-r

hängt Dateien an das Archiv an

-t

zeigt den Inhalt des Archivs

-u

ersetzt Dateien, die neuer als eine bereits archivierte Version sind. Ist eine Datei noch nicht archiviert, so wird sie eingefügt

-x

kopiert *Datei* oder alle Dateien aus dem Archiv

Weitere Optionen

-b *N*

setzt die Blockgröße auf *N*=x512 Bytes (Voreinstellung ist N=20)

-B

unterdrückt den Abbruch von **tar** beim Lesen unvollständiger Blöcke; zum Lesen von 4.2BSD Pipes

-C *Verzeichnis*

wechselt während der Ausführung in das *Verzeichnis*, um von dort weitere Dateien zu archivieren

-f *Datei*

benutzt *Datei* oder das damit verbundenen Gerät als Archiv

-F *Datei*

bei mehrteiligen Archiven (Option -M) wird das Shellscript *Datei* ausgeführt, wenn das Medium voll ist

-G

erzeugt einen speziellen Eintrag für jedes archivierte Verzeichnis; spezielles GNU Format (alt)

-g

erzeugt auch einen Eintrag für jedes Verzeichnis, auch spezielles GNU Format (neu)

-h

archiviert die referenzierten Dateien anstelle der Links

-i

ignoriert Blöcke mit Nullbytes im Archiv

-k

existierende Dateien werden nicht überschrieben

-K *Datei*

beginnt eine Aktion bei *Datei* im Archiv

-l

verhindert Archivierung von Dateien aus anderen Dateisystemen

-L *Länge*

wartet auf Medienwechsel nach *Länge*Bytes

-m

das Datum der letzten Änderung wird nicht mit archiviert

-M

das Archiv ist auf mehrere Medien verteilt (Multi-Volume)

-N *Datum*

archiviert nur Dateien, die neuer sind als *Datum*

-o

benutzt das alte V7 tar-Format anstelle des ANSI Formates

-O

schreibt die Dateien in die Standardausgabe

-p

erhält die Zugriffsrechte der Dateien

-P

archiviert mit absoluten Dateinamen

-R

gibt zu jeder Meldung die Blocknummer des Archivblocks aus, von dem die Meldung verursacht wurde

-s

zeigt an, daß die Liste von Dateien im Argument die gleiche Reihenfolge hat, wie die Dateien im Archiv

-T *Datei*

holt die Namen der zu archivierenden Dateien aus *Datei*

-v

meldet jede Aktion

-V *Name*

erzeugt ein Archiv mit dem (internen) Label *Name*

-w

erwartet interaktiv Bestätigung jeder Aktion

-W

verifiziert die geschriebenen Daten im Archiv

-X *Datei*

liest aus der *Datei* Namen oder reguläre Ausdrücke von bzw. für Dateien, die nicht archiviert werden sollen

-z

erzeugt ein mit **gzip** komprimiertes Archiv

-Z

erzeugt ein mit **compress** komprimiertes Archiv

-{0..7}{lmh}

spezifiziert das Gerät und die Dichte des Speichermediums (für Diskettenarchive ohne Bedeutung); 0 ist der erste Streamer, 1 der zweite und so weiter; die Dichte bestimmt den Bandtyp

Siehe Auch

cpio(1) , **mt**(1) , **shar**(1) und das LunetIX Linuxhandbuch

Autor

John Gilmore

1.103.1

Arbeiten auf der Kommandozeile

Beschreibung: Prüfungskandidaten sollten in der Lage sein, mit Shell und Kommandos auf der Kommandozeile umzugehen. Das beinhaltet das Schreiben gültiger Kommandos und Kommandoabfolgen, das Definieren, Referenzieren und Exportieren von Umgebungsvariablen, die Benutzung der Kommando-History und Eingabemöglichkeiten, das Aufrufen von Kommandos im und außerhalb des Suchpfades, das Benutzen von Kommandosubstitution, das rekursive Anwenden von Kommandos über einen Verzeichnisbaum und das Verwenden von man, um Informationen über Kommandos zu erhalten.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **.**
- **bash**
- **echo**
- **env**
- **exec**
- **export**
- **man**
- **pwd**
- **set**
- **unset**
- `~/.bash_history`
- `~/.profile`

Kommandos

An dieser Stelle werden noch nicht viele Kommandos vorausgesetzt. Trotzdem sollten grundlegende Befehle wie `less, more, cat, echo, ...` klar sein und problemlos verwendet werden können.

Grundsätzlich ist ein Unix-Kommando immer gleich aufgebaut. Zuerst wird das Programm angegeben, dann optionale Parameter und dann die Dateien, auf die das Programm angewandt werden soll.

Parameter liegen meist in zwei verschiedenen Formen vor, einmal als kurze Form, eingeleitet durch einen Bindestrich gefolgt von einem Buchstaben (wie etwa `-r`) und andererseits als lange Form, eingeleitet durch zwei Bindestriche gefolgt von einem Wort (`--recursive`). Kurze Parameter können dabei meist zusammengefasst werden, so daß statt `-z -v -f` einfach `-zvf` geschrieben werden kann. Das erklärt auch, warum lange Parameter mit zwei Bindestrichen beginnen. Sonst könnte ja ein `--recursive` als `-r -e -c -u -r -s -i -v -e` missinterpretiert werden.

Kommandoeingabe

Grundsätzlich ist die Eingabe von Kommandos eine Eingabe an die Shell (bash). Die Shell interpretiert das eingegebene Kommando, ersetzt bestimmte Muster durch Dateinamen oder Variableninhalte und führt erst dann das Kommando aus.

Linux-Kommandos werden entweder einfach eingegeben, oder zusammen mit dem Pfad, der die genaue Position des aufzurufenden Programmes im Dateisystem festlegt. Normalerweise reicht der Aufruf eines Kommandos ohne Pfad, wenn das Kommando im Suchpfad liegt.

Der Suchpfad wird in einer Shellvariable `PATH` definiert, die in der Regel in den Konfigurationsdateien der Shell mit vernünftigen Werten belegt wird. Diese Variable enthält eine durch Doppelpunkte getrennte Liste von Verzeichnissen, die von der Shell nach einem Befehl durchsucht werden sollen. Das aktuelle Verzeichnis ist nicht automatisch Teil des Suchpfades (wie bei DOS/Windows) sondern wird nur durchsucht, wenn es explizit durch die Angabe eines einzelnen Punktes im Suchpfad angegeben wurde.

Welches Verzeichnis gerade das aktuelle Arbeitsverzeichnis ist, kann mit dem Befehl **pwd** herausgefunden werden. Dieser Befehl gibt den kompletten absoluten Pfad des aktuellen Arbeitsverzeichnisses auf der Standard-Ausgabe aus.

Die Shell durchsucht den Suchpfad in der angegebenen Reihenfolge. Existieren auf einem System jetzt zwei Programme gleichen Namens, so wird das Programm ausgeführt, das sich in dem zuerst im Suchpfad genannten Verzeichnis befindet. Dieser Mechanismus kann umgangen werden, wenn nicht nur der Programmname, sondern der komplette Pfad als Befehl eingegeben wird. Auch Programme, die sich in Verzeichnissen befinden, die nicht im Suchpfad sind, können auf diese Weise aufgerufen werden.

Die Eingabe gültiger Kommandos und Kommandogruppen beinhaltet natürlich auch das Wissen um die verschiedenen Möglichkeiten, mehrere Befehle in eine Befehlszeile zu schreiben. Zur Erinnerung sind hier nochmal die verschiedenen Möglichkeiten der bash aufgelistet:

`Kommando1 ; Kommando2`

Kommando2 wird nach Kommando1 ausgeführt.

`Kommando1 & Kommando2`

Kommando1 und Kommando2 werden gleichzeitig ausgeführt (K1 im Hintergrund, K2 im Vordergrund)

`Kommando1 && Kommando2`

Kommando2 wird nur dann ausgeführt, wenn Kommando1 fehlerfrei abgeschlossen wurde (Rückgabewert 0)

`Kommando1 || Kommando2`

Kommando2 wird nur dann ausgeführt, wenn Kommando1 nicht fehlerfrei abgeschlossen wurde (Rückgabewert ungleich 0)

Wenn ein Kommando durch die Enter-Taste abgeschlossen wurde, dann wird es ausgeführt, sofern es syntaktisch fertig war. Fehlt aber noch etwas, für den syntaktischen Abschluß, war also etwa ein Anführungszeichen, das noch nicht durch ein zweites abgeschlossen wurde, so erscheint ein Folgeprompt, der eine weitere Eingabe ermöglicht. Das wird solange fortgesetzt, bis die Eingabe syntaktisch vollständig ist.

Vor der eigentlichen Ausführung des Kommandos wird der gesamte eingegebene Befehl in einen Kommandospeicher abgespeichert, aus dem er jederzeit (durch Navigieren mit den Pfeiltasten nach oben und unten) wiedergeholt und erneut ausgeführt werden kann. Wird die Shell beendet (etwa durch das Ausloggen), so werden alle im Kommandospeicher gespeicherten Befehle in die Datei `~/.bash_history` im Heimatverzeichnis des jeweiligen Users abgelegt, so daß die Befehle auch beim erneuten Einloggen wieder zur Verfügung stehen.

Ein eingegebenes Kommando erzeugt in der Regel einen neuen Prozeß. Wenn der Prozeß abgeschlossen wurde kehrt die Shell wieder zur Eingabeaufforderung zurück. Ein spezieller Befehl erlaubt es aber, einen Befehl einzugeben, der die Shell ersetzt. Der Befehl **exec** ***Kommando*** ersetzt den Shellprozeß durch den Prozeß des eingegebenen Kommandos. Das macht Sinn, wenn eine andere Shell aufgerufen werden soll oder der Befehl innerhalb eines Scripts ausgeführt wird. Wird ein durch **exec** ausgeführter Befehl beendet, ist aber logischerweise die Shell nicht mehr da, der Shellprozeß wurde ja durch das Kommando ersetzt.

Umgebungsvariablen

Variablen sind kleine Stücke des Arbeitsspeichers - genauer gesagt des Umgebungsspeichers der jeweiligen Shell - die einen Namen haben und einen Wert speichern können. Jede Instanz einer Shell hat einen eigenen Umgebungsspeicher, so daß verschiedene Shell-Instanzen gleichnamige Variablen mit unterschiedlichen Werten besitzen können.

Umgebungsvariablen werden in einer Shell definiert, indem sie einfach mit der Anweisung

Variablenname=Wert

angelegt werden. Existierte die Variable mit dem entsprechenden Namen noch nicht, so wird sie neu angelegt, gab es sie schon, so wird ihr Wert verändert.

Um auf den Inhalt (Wert) einer Variable zuzugreifen (refernzieren der Variable), wird dem Variablennamen ein Dollarzeichen ($) vorangestellt. Wenn die Shell auf ein Dollarzeichen stößt, versucht sie, den darauf folgenden Begriff als Variablennamen zu interpretieren und das ganze Konstrukt mit dem Wert der entsprechenden Variable zu ersetzen.

```
NAME=Huber
VORNAME=Hans
echo Hallo $VORNAME $NAME

  Hallo Hans Huber
```

Alle definierten Variablen sind mit dem Befehl **set** anzeigbar, mit **unset** können Variablen aus dem Umgebungsspeicher entfernt werden:

```
unset NAME
unset VORNAME
```

Wenn aus einer Shell heraus eine neue Shell (Subshell) aufgerufen wird, dann werden nicht automatisch alle Variablen der aufrufenden Shell für die Subshell übernommen. Um eine bestimmte Variable an spätere Subshells weiterzugeben, muß diese Variable exportiert werden. Das geschieht mit der Shell-Anweisung `export`. Dabei kann entweder eine bestehende Variable exportiert werden wie mit

`export` *`Variablenname`*

oder eine Variable gleich so definiert werden, daß sie später exportiert wird mit

`export` *`Variablenname=Wert`*

Eine exportierte Variable ist eine Kopie der Orginalvariablen. Das heißt, daß die Orginalvariable der aufrufenden Shell unverändert bleibt, wenn die exportierte Variable der Subshell verändert wird!

Variablen, die immer wieder benötigt werden, können in der Datei `~/.profile` im Homeverzeichnis jedes Users definiert werden. Diese Datei wird bei jedem Start einer Login-Shell abgearbeitet. So stehen einmal dort definierte Variablen jederzeit wieder zur Verfügung.

Soll ein Kommando mit bestimmten Variablen zusammen aufgerufen werden (also in einer bestimmten Umgebung), dann kann das mit dem Befehl env erledigt werden. Dabei kann entweder das Programm in einer völlig leeren Umgebung gestartet werden, oder zusätzliche Variablen werden speziell für diesen Programmaufruf definiert.

Kommandosubstitution

Bei der Kommandosubstitution handelt es sich um ein Konstrukt, das eine Subshell öffnet, einen bestimmten Teil einer Kommandozeile in dieser Subshell ausführt und das, was diese Ausführung auf die Standard-Ausgabe schreiben würde, an die Stelle der ursprünglichen Kommandozeile einfügt, an der das Substitutionskonstrukt stand. Erst jetzt wird die gesamte Zeile ausgeführt.

Es gibt zwei Möglichkeiten, die Kommandosubstitution einzuschalten. Zum einen die Klammerung mit dem Grave-Zeichen (`) und zum anderen das Konstrukt `$(...)`

Das klingt schlimmer als es ist, ein einfaches Beispiel soll zeigen, worum es geht:

Der Befehl `pwd` gibt an, in welchem Verzeichnis wir uns gerade befinden (Print WorkingDirectory). Wir werden ihn jetzt in einer Kommandosubstitution benutzen. Das Kommando

```
echo Wir befinden uns im Verzeichnis $(pwd)
```

wird eingegeben. Statt dessen wäre auch

```
echo Wir befinden uns im Verzeichnis `pwd`
```

gegangen. Was passiert nun mit dieser Befehlszeile? Die Shell ließt diese Zeile und stößt auf das Konstrukt `$(pwd)`. Sie erkennt dieses Konstrukt als Aufruf einer Kommandosubstitution und startet jetzt eine Subshell. Diese Subshell führt den Befehl `pwd` aus. Dieser Befehl schreibt das aktuelle Arbeitsverzeichnis auf die Standard-Ausgabe. Nehmen wir mal an, wir befänden uns im Verzeichnis `/home/hans`. Die Ausgabe des Befehls wäre also `/home/hans`. Jetzt ersetzt die ursprüngliche Shell das Konstrukt `$(pwd)` mit der Ausgabe der Subshell, also mit `/home/hans`. Die dadurch entstehende Kommandozeile lautet jetzt also

```
echo Wir befinden uns im Verzeichnis /home/hans
```

Und erst jetzt führt die Shell dieses Kommando aus.

Rekursives Anwenden von Kommandos

Viele Unix Kommandos erlauben es, daß sie nicht nur auf eine Datei oder ein Verzeichnis angewandt werden können, sondern auf ganze Verzeichnis-Hierarchien. Dazu steht meist ein Kommandozeilen-Parameter wie -r, -R oder --recursive zur Verfügung. Unglücklicherweise unterscheiden sich die Kommandos dabei ob -r oder -R verstanden wird. Der lange Parameter --recursive funktioniert aber überall, wo Rekursion überhaupt zur Verfügung steht. Ein paar Beispiele:

Befehl	Kurze Form	Lange Form

ls	-R	--recursive
chown	-R	--recursive
chmod	-R	--recursive
chgrp	-R	--recursive
grep	-r	--recursive
cp	-r und -R	--recursive
rm	-r und -R	--recursive

Informationen über Befehle mit man

Bei der Vielzahl von Linux-Befehlen, kann es sehr schnell passieren, daß man vergisst, welche Parameter von welchem Programm verstanden werden. Linux bietet eine Online-Hilfe an, die diese Informationen bereitstellt. Dabei handelt es sich um ein komplettes Handbuchsystem, das sowohl ausgedruckt, als auch auf dem Bildschirm dargestellt werden kann.

In der Vorbereitung auf die LPI102 Prüfung wird dieses Handbuchsystem noch genauer erläutert. Für den einfachen Umgang damit reichen die folgenden Informationen:

Das Programm man bietet Zugriff auf die Handbuchseiten (manual-pages). Es erwartet mindestens den Namen des Programmes als Parameter, über das Informationen dargestellt werden sollen. Um also Informationen über den Befehl **cp** (copy) zu erhalten, genügt der Befehl

```
man cp
```

Das Programm **man** stellt jetzt eine Handbuchseite zum Thema cp zusammen und gibt sie über den systemeigenen Pager (meist das Programm **less** auf dem Bildschirm aus.

1.103.2

Texte mittels Filterprogrammen bearbeiten

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Filter auf Textströme anzuwenden. Dieses Lernziel beinhaltet das Senden von Textdateien und -ausgaben durch Textfilterprogramme, um die Ausgabe zu modifizieren, und die Verwendung von Standard-Unix-Kommandos, die im GNU textutils Paket enthalten sind.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **cat**
- **cut**
- **expand**
- **fmt**
- **head**
- **join**
- **nl**
- **od**
- **paste**
- **pr**
- **sed**
- **sort**
- **split**
- **tac**
- **tail**
- **tr**
- **unexpand**

- **uniq**
- **wc**

Textfilter

Unter Unix werden sehr viele kleine Programme angeboten, die als Filter für Textdateien zur Anwendung kommen und die in der Regel nur eine kleine Aufgabe beherrschen. Durch die Kombination mehrerer solcher Programme mit Pipes und Umleitungen (siehe Abschnitt 1.103.4 - Benutzen von Unix Streams, Pipes und Umleitungen) können dadurch vielfältige Aufgaben gelöst werden. Die in dieser Zielsetzung genannten Textfilter sollten Sie beherrschen. Hier nochmal zu diesen Filtern ein paar Kurzbeschreibungen. Grundsätzlich ist das Studium der jeweiligen Handbuchseiten hier sehr empfehlenswert...

sed

sed ist ein Stream-Editor, ein Editor, der Datenströme oder Dateien nach bestimmten Regeln bearbeitet. sed ist ein sehr komplexes Programm, von dem in der LPI-101-Prüfung nur sehr geringe Kenntnisse verlangt werden. Wichtig ist zu wissen, daß dieses Werkzeug in der Lage ist, immer wiederkehrende Aufgaben mit Hilfe einer Scriptdatei zu lösen.

Ein Beispiel mag das verdeutlichen:

Sie haben 500 Textdateien, in denen immer wieder der Begriff "Synopsis" vorkommt. Sie wollen diesen Begriff nach "Syntax" ändern. Außerdem soll jeweils die 3. Zeile jeder dieser Dateien gelöscht werden. Das wäre unglaublich viel Arbeit, hätten wir nicht das Programm sed. Der Streameditor erlaubt uns verschiedene Aktionen auf einen Datenstrom auszuführen. Dazu erstellen wir eine Datei, die die notwendigen Befehle enthält. Nennen wir sie "befehle":

```
1,$s/Synopsis/Syntax/g
3d
```

Jede Zeile enthält einen Befehl. Jeder Befehl besteht aus einer Adress-Angabe und dem Befehl selbst. Die erste Zeile hat die Adressangabe `1,$`, was soviel heißt wie *von der ersten Zeile bis zum Dateiende*. Der Befehl ist `s` - das steht für *substitute* - also ersetzen. Der Befehl s hat die Form:

`s``/`*Suchbegriff*`/`*Ersetzungsbegriff*`/`*Optionen*

Als Option haben wir `g` benutzt, was heißt, daß eine Zeile global durchsucht werden soll, nicht nur das erste Auftreten, sondern jedes soll ersetzt werden.

Die zweite Zeile unserer Datei ist noch einfacher, ihr Adress-Teil ist einfach die 3 und meint also Zeile 3. Der Befehl ist `d` und steht für *delete* also löschen. Es ist also die Anweisung, die Zeile 3 zu löschen.

Um diese Befehlsdatei jetzt anzuwenden, schreiben wir nur noch:

```
sed -f befehle Datei > Datei_neu
```

und schon wurde die Datei `Datei` bearbeitet. Das Ergebnis steht jetzt in `Datei_neu`. Eine kleine Schleife konstruiert, und schon könnten wir alle 500 Dateien mit einem Aufwasch bearbeiten...

Wie gesagt, die LPI 101-Prüfung besteht nicht darauf, daß Sie den sed vollständig beherrschen, aber ein paar Übungen dazu sind nicht verkehrt. Genauere Beschreibungen, was dieses Programm für Befehle versteht finden Sie auf der sed-Handbuchseite.

sort

Dieses kleine Programm sortiert einen Eingabedatenstrom oder eine Datei zeilenweise und gibt das Ergebnis wieder auf die Standard-Ausgabe aus.

Normalerweise werden die ganzen Zeilen verglichen, um sie zu sortieren. Es ist aber auch möglich, Zeilen nach einem bestimmten Positionsfeld zu sortieren. Ein Beispiel:

Der Befehl `ls -l` gibt uns eine Liste aller Dateien im aktuellen Verzeichnis aus. Ab Spalte 31 steht die Dateigröße. Würden wir den Befehl

```
ls -l | sort
```

eingeben, dann würden die gesamten Zeilen sortiert. Das macht wenig Sinn, denn die Zeilen beginnen ja immer mit der Angabe der Zugriffsrechte und die zu sortieren ist sicher nicht das, was wir wollen. Um die Ausgabe nach Dateigröße zu sortieren können wir eingeben:

```
ls -l | sort +0.32
```

Dabei bedeutet `0.32` das erste Feld (0) und davon alles, ab dem 31. Buchstaben. (Die Angabe von Feldern bezieht sich auf Dateien, die tatsächlich feldorientiert aufgebaut sind, wie etwa `/etc/passwd`)

Alle Optionen und Möglichkeiten entnehmen Sie wieder der sort-Handbuchseite.

uniq

uniq ist ein Programm, das aus einer Datei (oder einem Eingabedatenstrom) hintereinanderliegende gleiche Zeilen löscht, so daß nur noch eine Zeile übrig bleibt. Die Betonung liegt auf hintereinanderliegend. Am häufigsten wird uniq in Kombination mit **sort** eingesetzt, weil gerade beim sortieren häufig vorhandene Zeilen hintereinander geschrieben werden. Hat eine Datei (oder ein Datenstrom) z.B. viele Leerzeilen, so werden all diese Leerzeilen hintereinander in die sortierte Ausgabe geschrieben. Mit **uniq** können diese Leerzeilen auf eine Zeile reduziert werden.

Über Kommandozeilenparameter kann zudem eingestellt werden, daß nur bestimmte Teile einer Zeile verglichen werden, um die Gleichheit festzustellen. Die genauen Parameter sind der uniq-Handbuchseite zu entnehmen.

cut

Mit cut werden bestimmte Spalten einer Datei ausgeschnitten und auf die Standard-Ausgabe geschrieben. Dabei können Spalten entweder durch absolute Positionen angegeben werden, oder als Felder, die durch bestimmte Trennzeichen abgegrenzt sind.

Um z.B. alle Usernamen aus der Datei /etc/passwd zu schneiden benutzen wir die Feldbegrenzer `:` und wollen nur das erste Feld sehen:

```
cut -d: -f1 /etc/passwd
```

Wenn wir aus der Liste aller laufender Prozesse die PIDs ausschneiden wollen, so müssen wir das mit absoluten Positionsangaben machen. Die Ausgabe von ps huax sieht ja etwa so aus:

```
at         172  0.0  0.4   892   448  ?  S    10:09   0:00 /usr/sbin/atd
bin        128  0.0  0.3   816   360  ?  S    10:09   0:00 /sbin/portmap
...
```

Die ProzeßIDs stehen also von Zeichen 9 bis 14. Um genau diese Spalte auszuschneiden schreiben wir also:

```
ps huax | cut -c9-14
```

So bietet uns cut also die Möglichkeit, jeden beliebigen Teil einer Zeile auszuschneiden und auf die Standard-Ausgabe zu schreiben. Das ist natürlich hervorragend geeignet, um in Pipes weiterverarbeitet zu werden. Die genaue Beschreibung entnehmen Sie wieder der cut-Handbuchseite

expand

Das Programm expand wird benutzt, um Tabulatorzeichen aus einer Textdatei oder einem Eingabedatenstrom durch eine beliebige Menge Leerzeichen im Ausgabedatenstrom zu ersetzen. Das klingt trivial, ist aber sehr häufig gebraucht, um Listen, die durch Tabs getrennt sind in ein druckbares Format zu verwandeln.

Die genaue Beschreibung ist der expand-Handbuchseite zu entnehmen.

unexpand

unexpand arbeitet genau gegenläufig zu expand. Es verwandelt eine bestimmte Menge Leerzeichen aus einer Datei (oder einem Eingabedatenstrom) in Tabulatorschritte. Dabei kann mit Parametern angegeben werden, wieviele Leerzeichen jeweils in einen Tab verwandelt werden sollen.

Die genaue Beschreibung ist der unexpand-Handbuchseite zu entnehmen.

fmt

fmt (steht für format) ist ein Programm, das Textdateien oder Eingabedatenströme absatzweise formatiert. Dabei sind Einstellungen möglich, wie etwa die Breite jedes Absatzes, die Anfangseinrückung oder das Zusammenziehen von mehreren Leerzeichen. So können etwa Textdateien mit Zeilenlängen über 80 für eine Druckausgabe auf eine Zeilenbreite von 75 umgewandelt werden.

Die genaue Beschreibung ist der fmt-Handbuchseite zu entnehmen.

head

Head schreibt die ersten Zeilen einer Datei oder eines Eingabedatenstroms auf die Standard-Ausgabe. Ohne Parameter werden die ersten 10 Zeilen ausgegeben.

Durch Kommandozeilenparameter kann bestimmt werden, wieviele Zeilen oder Bytes vom Anfang der Datei ausgegeben werden sollen. Dazu wird die gewünschte Anzahl Zeilen einfach mit einem Bindestrich versehen an den Befehl angehängt.

Die genaue Beschreibung ist der head-Handbuchseite zu entnehmen.

tail

Tail arbeitet wie head, nur daß nicht die ersten, sondern die letzten Zeilen einer Datei oder eines Eingabedatenstroms ausgegeben werden. Auch hier kann die gewünschte Anzahl der zu zeigenden Zeilen angegeben werden.

In Kombination mit head können damit beliebige Teile einer Datei ausgeschnitten werden. Wollen wir z.B. die Zeilen 7-12 der Datei `versuch.txt`, so können wir zuerst mit head die ersten 12 Zeilen der Datei auschneiden und uns dann daraus die letzten 6 Zeilen mit tail entnehmen. Elegant wird das in einer Pipe:

```
cat versuch.txt | head -12l | tail -6l
```

tail hat noch den speziellen Parameter -f, der es anweist, nicht nach der Ausgabe der letzten 10 Zeilen abzubrechen, sondern zu warten, ob die Datei noch weiter wächst und die dann angehängten Zeilen ebenfalls auszugeben. Wollen wir so etwa die Meldungen des Syslog-Daemon ständig überwachen, so schreiben wir:

```
tail -f /var/log/messages
```

Solange wir tail nicht mit Strg-C abbrechen, wird es nun die Datei /var/log/messages dauernd überwachen und jede neue Zeile auf dem Bildschirm anzeigen. Wir können also der Datei "beim Wachsen zusehen".

Die genaue Beschreibung ist der tail-Handbuchseite zu entnehmen.

join

join verknüpft zwei (alphabetisch) sortierte Dateien, indem je zwei Zeilen mit identischen Schlüsselfeldern zu einer Ausgabezeile verbunden werden.

Die Schlüsselfelder sind - wenn nicht anders angegeben - durch Leerzeichen voneinander getrennt. Führende Leerzeichen werden ignoriert. Wenn nicht anders angegeben, ist das erste Feld einer jeden Zeile Schlüsselfeld. Die Ausgabefelder sind ebenfalls durch Leerzeichen voneinander getrennt. Die Ausgabe besteht aus dem Schlüsselfeld, gefolgt von den übrigen Feldern der Datei1 und schließlich aller Felder der passenden Zeilen von Datei2 ohne das Schlüsselfeld.

Damit können verschiedene Dateien nach inhaltlichen Kriterien zusammengefügt werden. Nehmen wir ein Beispiel:

Die Datei `Adressen.txt` beinhaltet Zeilen wie die folgenden:

```
...
Huber Peter Schillerstr. 23 54321 Musterhausen
```

```
Maier Hans Goethestr. 3 12345 Musterstadt
Schmidt Stefan Kleistweg 34 55555 Musterhofen
...
```

Die Datei `Jobs.txt` enthält jetzt etwa

```
Huber Geschäftsführer
Kohler Fensterputzer
Schmidt Aufzugführer
```

Wenn wir jetzt beide Dateien mit dem Befehl

```
join Adressen.txt Jobs.txt
```

miteinander verknüpfen, so werden nur die Zeilen miteinander verbunden, die gemeinsame Schlüsselfelder haben. Voreingestellt ist das erste Feld, also der Name. Das Ergebnis wäre:

```
Huber Peter Schillerstr. 23 54321 Musterhausen Geschäftsführer
Schmidt Stefan Kleistweg 34 55555 Musterhofen Aufzugführer
```

Die genaue Beschreibung ist der join-Handbuchseite zu entnehmen.

nl

nl gibt die Zeilen einer oder mehrerer Dateien (oder der Standardeingabe) mit Zeilennummern auf die Standardausgabe. Es können dabei die Zeilen einer (logischen) Seite in einen Kopf, einen Körper und einen Fuß unterteilt werden, die jeweils einzeln und in unterschiedlichen Stilen nummeriert werden. Jeder Teil kann auch leer sein. Wenn vor dem ersten Kopfteil bereits Zeilen vorhanden sind, werden diese Zeilen wie ein Seitenkörper nummeriert.

Die Nummerierung beginnt auf jeder Seite neu. Mehrere Dateien werden als ein einziges Dokument betrachtet, und die Zeilennummer nicht zurückgesetzt.

Der Kopfteil wird durch eine Zeile eingeleitet, die nur die Zeichenkette \:\:\: enthält. Der Körper wird entsprechend durch \:\: und der Fuß durch \: eingeleitet. In der Ausgabe werden diese Zeilen als Leerzeilen ausgegeben.

Die genaue Beschreibung ist der nl-Handbuchseite zu entnehmen.

od

od (Octal Dump) gibt Dateien oder einen Eingabedatenstrom als dezimalen, oktalen oder hexadezimalen Dump aus. So können Binärdateien in eine lesbare Form gebracht werden. Jede Ausgabezeile enthält neben der eigentlichen Bytedarstellung am Zeilenanfang noch ein siebenstelliges Adressfeld.

Die genaue Beschreibung ist der od-Handbuchseite zu entnehmen.

paste

paste verknüpft zwei oder mehrere Dateien zeilenweise, so daß die ersten Zeilen der Dateien zur ersten Zeile der Ausgabedatei werden.

Damit können also einzelne Dateien, die Spalten enthalten, die etwa von cut erzeugt wurden, wieder zu einer Datei zusammengesetzt werden. Nehmen wir ein simples Beispiel:

Wir wollen eine Datei erstellen, in der in der ersten Spalte die UserID, in der zweiten der Username steht. Die Datei soll alle User des Systems enthalten. Wir können das nicht allein mit cut erledigen, weil ja in /etc/passwd zuerst der Username und erst im 3. Feld die UserID steht. Zunächst erzeugen wir also zwei Dateien, die jeweils nur eine Spalte der /etc/passwd beinhalten, dann fügen wir diese beiden Dateien umgekehrt wieder zusammen:

```
cut -d: -f1 /etc/passwd > Datei1
cut -d: -f3 /etc/passwd > Datei2
paste -d: Datei2 Datei1 > Datei3
```

Im Gegensatz zu join (siehe oben) werden die Zeilen der angegebenen Dateien ohne inhaltliches Kriterium miteinander verknüpft. Die genaue Beschreibung ist der paste-Handbuchseite zu entnehmen.

pr

pr dient zur Formatierung von Textdateien für die Druckerausgabe. pr wird heute nur noch selten verwendet, es gibt einfach andere, bessere Druckerfilter. Trotzdem kann es manchmal noch gute Dienste leisten.

Immerhin bietet pr auch Features wie Spaltensatz, Seitennummerierung und Kopfzeilen, genaueres siehe bei der pr-Handbuchseite

split

Die Anwendung von split ist einfach und sie wird häufig gebraucht, um z.B. große Dateien in viele kleinere aufzuteilen, damit sie mit Disketten transportiert werden können. Die Größe der anzulegenden Zieldateien kann sowohl in Zeilen, als auch in Bytes angegeben werden.

Die kleineren Zieldateien heißen standardmäßig xaa, xab, xac, ..., wobei - falls die Buchstaben nicht ausreichen - auch mehrere Buchstaben verwendet werden (xaaaa, xaaab,...). Weil das ein wenig unhandlich ist, kann man dem split-Programm ein Präfix mitgeben, das dann das x ersetzt.

Um also z.B. eine Datei netscape.tgz mit ca 13 MByte in Dateien der Größe 1 MByte aufzuteilen, um sie auf Disketten verteilen zu können, schreiben wir:

```
split -b 1m netscape.tgz
```

Als Ergebnis bekommen wir jetzt die Dateien xaa, xab, xac, ... Das ist verwirrend - daher benutzen wir eben das Präfix:

```
split -b 1m netscape.tgz netscape_
```

Jetzt bekommen wir die Dateien netscape_aa, netscape_ab, ...

Um diese Dateien wieder zusammenzubauen benutzen wir einfach das cat-Programm. wir könnten so also schreiben:

```
cat netscape_aa netscape_ab netscape_ac netscape_ad netscape_ae > netscape.tgz
```

Das ist zugegebenerweise etwas unhandlich, aber jetzt kommt der Vorteil ins Spiel, daß die Shell alphabetisch ihre Ausgaben sortiert. Weil die aa, ab, ac ... Aufteilung streng alphabetisch ist, können wir auch viel einfacher schreiben:

```
cat netscape_* > netscape.tgz
```

Die genaue Beschreibung ist der split-Handbuchseite zu entnehmen.

cat

Das Programm cat arbeitet ähnlich wie das DOS-Programm TYPE. Es gibt die Dateien, die als Parameter angegeben wurden auf der Standard-Ausgabe aus. Falls keine Parameter angegeben wurden, so nimmt cat die Standard-Eingabe als Datenstrom an, der auf der Standar-Ausgabe ausgegeben werden soll.

Das klingt auf den ersten Blick trivial, wird aber sehr häufig gebraucht, um Dateien in eine Pipe oder auf ein Gerät zu schicken:

```
cat datei.txt > /dev/tty10
```
Schickt die Datei datei.txt statt auf die Standard-Ausgabe in die Gerätedatei /dev/tty10 und damit auf das Terminal 10.

Außerdem können mit cat mehrere Dateien zu einer zusammengefügt werden, indem die Ausgabe wieder in eine Datei umgeleitet wird:

```
cat teil1.txt teil2.txt teil3.txt > gesamt.txt
```
Mit Optionsschaltern kann cat auch dazu gebracht werden, mehrere aufeinanderfolgende Leerzeilen zu einer Leerzeile zu reduzieren oder Zeilen in der Ausgabe zu nummerieren.

Die genaue Beschreibung ist der cat-Handbuchseite zu entnehmen.

tac

Das Programm tac (cat umgekehrt geschrieben) gibt Dateien in umgekehrter Reihenfolge aus, also die letzte Zeile zuerst, dann die vorletzte usw. Dabei müssen es nicht zwangsläufig Zeilen sein, tac kann auch andere Zeichen als den Zeilentrenner als Trennzeichen benutzen.

Die genaue Beschreibung ist der tac-Handbuchseite zu entnehmen.

tr

tr (translate) löscht oder ändert einzelne Zeichen eines Eingabedatenstroms und schreibt das Ergebnis wieder auf die Standard-Ausgabe.

Die Angabe der zu übersetzenden Zeichen erfolgt dabei in sogenannten Mengen. Eine Menge ist einfach eine Zeichenkette aus Buchstaben. Um z.B. in einer Datei alle vorkommenden a,b und x in A,B und X zu verwandeln, benutzen wir die Mengen "abx" und "ABX".

tr ist ein reiner Filter, d.h., das Programm kann ausschließlich in Pipes verwendet werden. Wenn wir also die Datei `Test.txt` übersetzen wollen, so müssen wir schreiben:

```
cat Test.txt | tr abx ABX > Ergebnis
```
Zugegebenermaßen ein unsinniges Beispiel. tr erlaubt es aber auch, Steuerzeichen und andere undruckbare Zeichen in den Mengen zu verwenden, indem ihr Oktalwert angegeben wird. Dazu muß dem Wert ein Backslash vorangestellt werden. Hier sind dann aber auch Anführungszeichen nötig, sonst würde die Shell uns den Backslash weginterpretieren.

Außerdem bietet tr mehrere bereits vordefinierte Mengen, wie etwa alle Großbuchstaben und alle Kleinbuchstaben. So können wir eine Datei z.B. in Großbuchstaben verwandeln indem wir schreiben:

```
cat Test.txt | tr [:lower:] [:upper:] > Ergebnis
```

Die genaue Beschreibung ist der tr-Handbuchseite zu entnehmen.

wc

Dieses Programm (word count) zählt Zeichen, Wörter und Zeilen einer Datei oder eines Eingabedatenstroms. Es ist in der LPI 101 Prüfung relativ häufig gefragt, aber auch wirklich sehr einfach anzuwenden.

Ohne Parameter aufgerufen gibt wc alle drei Angaben aus, in der Reihenfolge Zeilen, Wörter, Zeichen. wc kennt aber auch Parameter, die es veranlassen nur die Zeilen, Wörter oder Zeichen zu zählen. Es gibt diese Parameter jeweils in einer langen und kurzen Form, aber auch jeweils leicht zu merken und zu verstehen:

- `-l` oder `--lines` zählt nur die Zeilen
- `-w` oder `--words` zählt nur die Wörter
- `-c` oder `--chars` zählt nur die Zeichen

Die genaue Beschreibung ist der wc-Handbuchseite zu entnehmen.

1.103.3

Durchführung eines allgemeinen Datei-Managements

Beschreibung: Prüfungskandidaten sollten in der Lage sein, die grundlegenden Unix Kommandos zum Kopieren und Verschieben von Dateien und Verzeichnissen zu benutzen. Dieses Lernziel beinhaltet das Durchführen fortgeschrittener Dateiverwaltungs-Operationen wie dem rekursiven Kopieren mehrerer Dateien, das rekursive Löschen von Verzeichnissen und das Verschieben von Dateien, deren Namen einem bestimmten Muster entsprechen. Dies beinhaltet das Benutzen einfacher und komplexerer Wildcard-Muster für das Bestimmen von Dateien sowie die Verwendung von **find**, um Dateien aufgrund von Typ, Größe oder Zeitstempel ausfindig zu machen und Operationen an ihnen durchzuführen.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **cp**
- **find**
- **mkdir**
- **mv**
- **ls**
- **rm**
- **rmdir**
- **touch**
- *File-Globbing* (Suchmuster mit Wildcards)

Kommandos zum Umgang mit Dateien

Grundsätzlich stehen uns unter Linux die gleichen Programme zur Verfügung, wie unter jedem anderen textorientierten System. Wir können Dateien kopieren, verschieben (umbenennen) und löschen. Die jeweils notwendigen Kommandos sind cp (copy), mv (move) und

rm (remove).

Im Gegensatz zu MSDOS Befehlen haben diese Linux/Unix-Befehle aber eine Eigenschaft, die nennenswert ist; sie können beliebig viele Dateien bearbeiten. Das heißt, es ist möglich mehrere Dateinamen anzugeben, die kopiert, verschoben oder gelöscht werden sollen.

Das bringt zumindestens beim kopieren und verschieben ein Problem mit sich, das so unter DOS nicht entsteht. DOS-Befehle zum kopieren und verschieben erlauben es, das Ziel der Operation wegzulassen. Fehlt das Ziel, so wird das aktuelle Verzeichnis als Ziel angenommen. Das ist bei Unix Befehlen **nicht** möglich, und zwar eben genau weil diese Befehle mehrere Dateien bearbeiten können. Dürften wir das Ziel weglassen, so wäre der Befehl, sofern ihm mehrere Dateien mitgegeben wurden, nicht in der Lage zu ermitteln, ob der letzte Parameter eine zu kopierende Datei sein soll oder das Zielverzeichnis... Trotzdem ist das leicht zu verschmerzen, heißt das aktuelle Verzeichnis doch einfach `.` Es reicht also, als letzten Parameter einen Punkt anzugeben, das ist nicht viel Mehraufwand.

Alle drei Befehle sind in der Lage, auch Verzeichnisse zu bearbeiten, cp und rm jedoch nur, wenn ihnen das mittels Kommandozeilenparameter mitgegeben wurde. Näheres dazu siehe unten unter dem Punkt "Rekursive Bearbeitung".

Manchmal ist es einfach nur nötig, eine leere Datei neu anzulegen. Diese Aufgabe kann auf unterschiedliche Weise gelöst werden, hier drei Beispiele:

- **Kopieren der Gerätedatei /dev/null in eine Datei**
  Die Gerätedatei /dev/null ist eigentlich dazu gedacht, unerwünschte Aufgaben ins Nirwana zu schicken, indem sie dorthin umgeleitet werden. Man kann diese Datei aber dazu nutzen, eine neue, leere Datei anzulegen, inem man schreibt:

  ```
  cp /dev/null Neudatei
  ```

  Das Ergebnis ist eine leere Datei mit Namen Neudatei.
- **Einfache Umleitung**
  Mit einem Umleitungssymbol (>) kann auch eine leere Datei angelegt werden. Das ist der einzige Fall, in dem in einer Unix-Kommandozeile kein Befehl am Anfang steht. Das Kommando

  ```
  >Neudatei
  ```

  legt eine leere neue Datei mit Namen Neudatei an.
- **Das Programm touch**
  Das Programm touch ist eigentlich dazu gedacht, die Zeitmarkierung einer Datei auf das aktuelle Datum und die aktuelle Uhrzeit zu ändern. Die Datei wird sozusagen "berührt" und damit werden die Zeitmarken verändert. Wenn die angegebene Datei jedoch noch nicht existiert, so wird sie angelegt. Der Befehl

  ```
  touch Neudatei
  ```

legt also auch eine leere neue Datei mit Namen Neudatei an.

Alle drei Methoden sind im Ergebnis gleich. In jedem Fall haben wir eine leere, neue Datei der Größe 0 Byte.

Kommandos zum Umgang mit Verzeichnissen

Neue Verzeichnisse werden unter Linux mit dem Befehl mkdir (make directory) erstellt. Mit dem Parameter `-p` oder `--path` kann dieser Befehl sogar einen ganzen Verzeichnisbaum mit beliebig vielen Unterverzeichnissen anlegen.

Leere Verzeichnisse können mit dem Befehl rmdir (remove directory) gelöscht werden. Verzeichnisse können so aber nicht gelöscht werden, wenn sie noch Dateien enthalten. Eine leere Verzeichnishierarchie kann wieder mit dem Parameter `-p` oder `--path` komplett gelöscht werden.

Das aktuelle Arbeitsverzeichnis kann mit dem Befehl cd gewechselt werden. Dieser Befehl funktioniert wie unter DOS auch, allerdings beharrt er auch beim Wechsel in das Mutterverzeichnis (`..`) auf ein Leerzeichen zwischen `cd` und `..`

Wird der Befehl cd ohne Parameter eingegeben, so wechselt man in sein Homeverzeichnis.

Wird der Befehl in der Form

```
cd ~username
```

verwendet, so wechselt man ins Homeverzeichnis des angegebenen Users.

Um sich den Inhalt eines Verzeichnisses anzusehen, gibt es den Befehl ls (list). Er kann auf vielfältige Weise angewandt werden, wichtig sind die folgenden Parameter:

Befehl	Bedeutung
`ls`	Nur die Namen der Dateien und Verzeichnisse werden angegeben
`ls -l`	Langes Listing. Zugriffsrechte, Anzahl der Hardlinks, Eigentümer, Gruppe, Datum, Größe und Namen werden angegeben.
`ls -i`	Die Inode Nummern der Dateien werden mit angegeben.
`ls -a`	Auch die "versteckten" Dateien (deren Namen mit einem Punkt beginnt) werden angezeigt.

Werden ls keine Dateinamen mitgegeben, so zeigt der Befehl den Inhalt des aktuellen Verzeichnisses. Ansonsten wird ls die Dateien eines

angegebenen Verzeichnisses oder die Informationen über die angegebenen Dateien zeigen.

Rekursive Bearbeitung

Die genannten Befehle zum Kopieren, Verschieben und Löschen von Dateien sind alle auch in der Lage, ganze Unterverzeichnisäste zu bearbeiten. Dazu muß folgendes erwähnt sein:

- Der Befehl cp (copy) kann Unterverzeichnisäste nur dann kopieren, wenn ihm als Parameter ein `-r, -R` oder `--recursive` mitgegeben wurde.
- Der Befehl rm (remove) kann Unterverzeichnisse nur dann löschen, wenn ihm als Parameter ein `-r, -R` oder `--recursive` mitgegeben wurde.
- Der Befehl mv (move) verschiebt grundsätzlich auch Unterverzeichnisse, allerdings nur innerhalb der Grenzen einer Partition (eines Dateisystems). Normale Dateien können auch von Partition zu Partition verschoben werden, Verzeichnisse nicht. mv kennt keinen Parameter `-r`
- Der Befehl ls (list) zeigt alle Dateien auch die der Unterverzeichnisse, wenn ihm der Parameter `-R` oder `--recursive` mizgegeben wurde.

Wildcards

Wildcards (oder auch Jokerzeichen) werden unter Linux nicht von den einzelnen Programmen, sondern von der Shell interpretiert. Alle der genannten Programme können ja - wie oben schon erwähnt - mehrere Dateien als Parameter mitbekommen. Die Shell ersetzt die Wildcard Konstruktion durch eine Liste von passenden Dateinamen - der Befehl selbst bekommt also das Wildcard Konstrukt selbst nie zu sehen.

An Wildcards gibt es unter Linux folgende Möglichkeiten:

Wildcard	Bedeutung
*	Das Sternchen steht für eine beliebige Folge beliebiger Zeichen, auch die leere Folge. Ein einzelnes Sternchen meint also alle Dateien mit Ausnahme der Dateinamen, die mit einem Punkt beginnen. Das MSDOS-Konstrukt `*.*` würde nur alle Dateien ansprechen, die irgendwo (auch am Anfang oder Ende) einen Punkt im Namen vorweisen.
?	Das Fragezeichen steht für genau ein beliebiges Zeichen.
`[...]`	Eine Menge von Zeichen in eckigen Klammern steht für genau ein Zeichen aus dieser Menge.

`[...-...]`	Weiterentwicklung des Mengenprinzips. So ist es auch möglich, Bereiche anzugeben, ohne alle Zeichen anzugeben. Beispiel `[0-9]` oder `[a-zA-Z]`. Wieder ist genau ein Zeichen aus der Menge gemeint.
`[!...]`	Steht am Anfang der Menge ein Ausrufungszeichen, so wird der Wahrheitswert der Menge umgedreht (NOT) - das Konstrukt steht also für genau ein Zeichen, aber dieses Zeichen darf **nicht** in der angegebenen Menge stehen.

All diese Konstrukte dürfen beliebig miteinander kombiniert werden, jede einzelne Wildcard darf auch mehrfach vorkommen. So würde etwa die Konstruktion

```
[A-Z]*.[0-9][0-9][0-9]_[!A]
```

jede Datei ansprechen, deren Name mit einem Großbuchstaben (`[A-Z]`) beginnt. Danach darf irgendwas (`*`) folgen, beliebig lang. Dann muß ein Punkt kommen, gefolgt von drei Ziffern (`[0-9]`). Dem folgt ein Unterstrich und dann ein beliebiges Zeichen - nur eben kein A.

Dateien finden mit find

Ein universelles Werkzeug für den Umgang mit Dateien ist das Programm find. Es ermöglicht das Suchen von Dateien anhand aller denkbaren Kriterien wie

- Namensmuster
- Zugriffs-, Modifikations- und Statusveränderungsdatum
- Eigentümer und Gruppenzugehörigkeit
- Dateiart und Zugriffsrecht
- Größe
- ...

find ermöglicht es darüber hinaus, beliebige Befehle auf die gefundenen Dateien anzuwenden, sie also zu löschen, kopieren, verschieben oder was auch immer. Im Abschnitt 1.104.8 - Auffinden von Systemdateien und Platzieren von Dateien an den korrekten Ort wird der Befehl genauer besprochen.

Zurück

1.103.4

Benutzen von Unix Streams, Pipes und Umleitungen

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Ein- und Ausgabeströme umzuleiten und sie zu verknüpfen, um Textdaten effizient zu verarbeiten. Dieses Lernziel beinhaltet das Umleiten von Standardeingabe, Standardausgabe und Standardfehlerausgabe, das Umleiten der Ausgabe eines Kommandos in die Eingabe eines anderen Kommandos, das Verwenden der Ausgabe eines Kommandos als Argument für ein anderes Kommando und das gleichzeitige Senden einer Ausgabe sowohl an die Standardausgabe als auch in eine Datei.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **tee**
- **xargs**
- <
- <<
- >
- >>
- |
- ``

Grundlegendes Prinzip

Jedes Programm unter Linux besitzt drei Standardkanäle für die Datenein/ausgabe. Diese drei Kanäle sind

Standard-Eingabe (STDIN)

Der Kanal, von dem das Programm seine Eingabe erwartet. Dieser Kanal besitzt die Kenn-Nummer 0 und ist normalerweise mit der Tastatur verbunden.

Standard-Ausgabe (STDOUT)

Der Kanal, auf den das Programm seine Ausgaben schreibt. Dieser Kanal besitzt die Kenn-Nummer 1 und ist normalerweise mit dem Monitor verbunden.

Standard-Fehlerausgabe (STDERR)

Der Kanal, auf den das Programm seine Fehlerausgaben schreibt. Dieser Kanal besitzt die Kenn-Nummer 2 und ist wie STDOUT normalerweise mit dem Monitor verbunden.

Schematisch könnten wir das etwa folgendermaßen darstellen:

Der Grund für die Auftrennung von Standard Ausgabe und Standard Fehlerausgabe ist einfach der, daß wir diese Kanäle oftmals umleiten. Und eine Umleitung sorgt natürlich auch dafür, daß die Ausgabe jetzt nicht mehr auf dem Schirm erscheint. Käme es jetzt zu einem Fehler, so würden wir es nicht mitbekommen, weil die Ausgabe nicht sichtbar ist. Aus diesem Grund gibt es die separate Fehlerausgabe, die dafür sorgt, daß Fehlerausgaben immer noch auf dem Bildschirm zu lesen sind, auch wenn die Ausgabe umgeleitet wurde.

Umleitungen

Die Shell verwaltet die Ein- und Ausgabekanäle. Sie ist es auch, die diese Kanäle umleiten kann. Das Programm selbst merkt von dieser Umleitung nichts, es schreibt weiterhin auf die Standard Ausgabe und liesst von der Standard Eingabe. Die Shell bietet uns vier Möglichkeiten der Umleitungen. Zwei für die Eingabe und zwei für die Ausgabe:

Umleitung	Bedeutung
Programm > Datei	Das Programm leitet seine Standard Ausgabe in die Datei um, statt sie auf den Bildschirm zu schreiben. Existiert die Datei schon, so wird sie überschrieben, existiert sie noch nicht, so wird sie neu angelegt.
Programm >> Datei	Das Programm leitet seine Standard Ausgabe in die Datei um, statt sie auf den Bildschirm zu schreiben. Existiert die Datei schon, so wird die Ausgabe hinten an die bestehende Datei angehängt, existiert sie noch nicht, so wird sie neu angelegt.
Programm < Datei	Das Programm ließt seine Eingabe aus der Datei statt von der Tastatur.

Programm << `EOM`
`...`
`EOM`
	Das Programm ließt seine Eingaben statt von der Tastatur aus dem Block zwischen den beiden `EOM` Marken. Dabei dürfen diese Marken beliebige Worte sein, es gilt immer vom ersten bis zum zweiten Auftreten der Marke. Diese Konstruktion wird "here-document" genannt und findet Anwendung fast ausschließlich in der Scriptprogrammierung.

Manchmal ist es auch sinnvoll oder gewünscht, den Fehlerausgabekanal umzuleiten. Das ist über die Nennung seiner Kenn-Nummer vor dem Umleitungssymbol möglich. So würde die Anweisung

```
Programm 2> Fehlerprotokoll
```

die Fehlerausgabe des Programms in die Datei Fehlerprotokoll umleiten. Selbstverständlich ist das auch mit 2 Umleitungssymbolen (anhängend) möglich.

Und in einigen Fällen sollen beide Ausgabekanäle (STDOUT und STDERR) doch in eine Datei umgeleitet werden. Dazu kann man beide Kanäle zu einem zusammenfassen, der dann umgeleitet wird. Das geschieht mit der kryptischen Anweisung `2>&1`. Aber vorsicht, um wirklich beide Kanäle in eine Datei umzuleiten muß die Bündelung nach der eigentlichen Umleitung vorgenommen werden:

```
Programm > Datei 2>&1
```

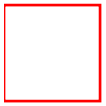
Diese Anweisung bündelt die Kanäle 2 und 1 (STDERR und STDOUT) und leitet beide in die Datei um.

Verkettung (piping)

Wenn wir jetzt zwei Programme haben, von denen das zweite die Ausgabe des ersten weiterverarbeiten sollte, so könnten wir jetzt die Ausgabe des ersten Programms in eine Datei umleiten und dann das zweite Programm aufrufen und seine Eingabe aus der Datei lesen lassen:

```
Programm1 > Datei
Programm2 < Datei
```

Das können wir auch ohne den Umweg über die Datei haben, indem wir zwei Programme direkt miteinander verbinden. Genauer gesagt verbinden wir die Standard-Ausgabe des ersten Programms mit der Standard-Eingabe des zweiten.



Dazu benutzen wir das Symbol `|` so daß wir einfach nur schreiben müssen:

```
Programm1 | Programm2
```

Das läßt sich beliebig fortsetzen, es ist also möglich, auch mehrere Programme hintereinanderzuhängen.

xargs

Das Programm xargs bietet eine Möglichkeit, die stark an die Kommandosubstitution erinnert. Hauptsächlich wird dieses Programm in Shells benötigt, die keine Substitution bieten, aber es gibt auch Fälle, in denen mit xargs elegantere Lösungen möglich sind, als mit der

Kommandosubstitution.

xargs führt ein beliebiges Kommando aus, und gibt diesem Kommando als Parameter das mit, was xargs von der Standard-Eingabe gelesen hat. In einer Pipe ist es also möglich, daß ein Programm seine Ausgaben an xargs weiterleitet und xargs führt dann ein anderes Programm aus, das mit eben der Ausgabe des ersten Programms als Parameter bestückt wird. Ein simples Beispiel:

Wir haben eine Datei mit Namen `Liste`, die folgenden Inhalt hat:

```
datei1
datei2
datei3 datei4
datei5 /home/hans
```

Jetzt rufen wir folgenden Befehl auf:

```
cat Liste | xargs cp
```

Der Befehl `cat Liste` gibt den Inhalt der Datei `Liste` auf die Standard-Ausgabe aus. Diese ist aber mit der Standard-Eingabe von `xargs cp` verbunden. xargs ruft also den Befehl `cp` auf und gibt ihm das mit, was es selbst von der Standard-Eingabe gelesen hat - also in unserem Fall den Inhalt der Liste. Zeilentrenner werden dabei wie Leerzeichen interpretiert. xargs ruft also folgenden Befehl auf:

```
cp datei1 datei2 datei3 datei4 datei5 /home/hans
```

Den selben Effekt hätten wir mit Kommandosubstitution mit folgender Zeile erreichen können:

```
cp `cat Liste`
```

Der Vorteil von xargs liegt darin, daß beliebig lange Ergebnisse mitgegeben werden können, weil xargs die Parameterkette aufteilt und das Programm entsprechend oft hintereinander aufruft. Die Kommandosubstitution kann das nicht und stößt irgendwann an die Grenzen der erlaubten Kommandolänge.

xargs kennt noch viele verschiedenen Kommandozeilenparameter, die aber für die LPI-101 Prüfung eher irrelevant sind.

Zwischensicherung

Die Ausnutzung komplexer Pipekonstruktionen erfordert es oft, daß einzelne Zwischenschritte während des Ablaufs in eine Datei gespeichert werden. Dazu gibt es das Programm tee, das einfach seine Standard-Eingabe auf seine Standard-Ausgabe weitergibt und daneben aber auch den Datenstrom in eine Datei zwischenspeichert. Es handelt sich also um eine Art T-Stück in einer Pipe (Rohrleitung) - daher stammt auch der Name.

Das folgende Konstrukt gibt ein Beispiel für die Anwendung:

```
Programm1 | tee Datei1 | Programm2 | tee Datei2 | Programm3 > Datei3
```

Das Programm1 schickt seine Ausgabe an das Programm tee, das sie dann in die Datei1 schreibt, aber gleichzeitig auch wieder an das Programm2 weitergibt. Dieses Programm gibt wiederum seine Ausgabe auf eine weitere Instanz von tee weiter, welches sie wieder in die Datei2 schreibt und an das Programm3 weiterleitet. Das dritte Programm schließlich speichert sein Ergebnis in der Datei3. Wir haben also alle Zwischenergebnisse einzeln gespeichert, also das gleiche Ergebnis, als hätten wir geschrieben:

```
Programm1 > Datei1
Programm2 < Datei1 > Datei2
Programm3 < Datei2 > Datei3
```

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

1.103.5

Erzeugung, Überwachung und Terminierung von Prozessen

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Prozesse zu verwalten. Dieses Lernziel beinhaltet das Ausführen von Prozessen in Vorder- und Hintergrund, das Bringen eines Jobs vom Hintergrund in den Vordergrund und umgekehrt, das Starten eines Prozesses, der ohne Verbindung zu einem Terminal laufen soll, und die Mitteilung an ein Programm, daß es nach Abmelden weiterlaufen soll. Ebenfalls enthalten ist die Überwachung aktiver Prozesse, die Auswahl und das Sortieren von Prozessen für die Ausgabe, das Senden von Signalen an Prozesse, das Terminieren von Prozessen sowie das Erkennen und Terminieren von X-Anwendungen, die nach dem Schließen der X-Sitzung nicht ordnungsgemäß beendet wurden.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- &
- **bg**
- **fg**
- **jobs**
- **kill**
- **nohup**
- **ps**
- **top**

Ausführen von Prozessen im Vorder- und Hintergrund

Prozesse werden von der Shell grundsätzlich im Vordergrund gestartet, indem einfach eine Kommandozeile eingegeben wird. Die Shell wartet dann bis der Prozess beendet ist und kehrt erst danach zur Eingabeaufforderung zurück.

Damit die Eingabeaufforderung sofort wieder zur Verfügung steht, kann ein Prozess im Hintergrund gestartet werden, indem der Kommandozeile einfach ein Ampersand (&) angehängt wird.

Dadurch kehrt die Shell sofort wieder zur Eingabeaufforderung zurück, der gestartete Prozess läuft jetzt im Hintergrund.

Diese Ausführungsart eignet sich jedoch nur für Prozesse, die nichts auf die Standard-Ausgabe schreiben (oder deren Ausgaben umgeleitet werden), weil auch die Hintergrundausführung die Ausgaben auf der Standard-Ausgabe belässt. Ein Hintergrundprozeß, der permanent auf den Bildschirm Meldungen ausgibt, ist wohl eher störend.

Häufig wird diese Fähigkeit benutzt, um in der graphischen Oberfläche aus einem XTerm heraus graphische Programme aufzurufen. Durch die Hintergrundausführung kehrt die Shell sofort zur Eingabeaufforderung zurück, das XTerm ist also nicht solange gesperrt, solange das graphische Programm läuft.

Wechsel zwischen Vorder- und Hintergrundausführung

Die Shell bietet auch Möglichkeiten, Prozesse aus dem Vordergrund in den Hintergrund zu schieben und umgekehrt. Dazu sind fünf Funktionen erforderlich:

Strg-Z

Hält einen Vordergrundprozess an. Dadurch wird dem Job jegliche Rechenzeit entzogen und die Shell kehrt zur Eingabeaufforderung zurück.

Strg-C

Killt einen Vordergrundprozess. Dadurch wird der Job beendet und die Shell kehrt zur Eingabeaufforderung zurück.

`bg` [*Jobnummer*]

Startet einen angehaltenen Job im Hintergrund. Der Job bekommt wieder Rechenzeit, die Shell kehrt aber zur Eingabeaufforderung zurück.

`fg` [*Jobnummer*]

Startet einen angehaltenen Job oder einen im Hintergrund laufenden Job im Vordergrund. Die Shell kehrt nicht mehr zur Eingabeaufforderung zurück.

`jobs`

Zeigt eine Liste aller Jobs, Vordergrund- Hintergrund und gestoppte Jobs samt ihren JobIDs.

Zusammengefasst kann gesagt werden, daß jede Shell beliebig viele Hintergrundjobs und maximal einen Vordergrundprozess besitzen kann. Mit den oben genannten Funktionen ist es möglich jede beliebige Kombination herzustellen. Die Befehle `fg, bg` und `jobs` sind eingebaute Shellfunktionen. Wichtig ist die Unterscheidung zwischen JobID und ProzessID. Die JobID bezieht sich auf die jeweilige Shell, die ProzessID ist von überall her gültig und im ganzen System eindeutig.

Prozesse unabhängig von der Terminalsitzung starten

Normalerweise wird ein Programm von einer Shell aufgerufen. Die Shell startet dann einen weiteren Prozeß, dessen Elternprozeß sie selbst darstellt. Wenn die Shell jetzt beendet wird, etwa durch ein **exit**, durch ein **Strg-d** oder den Befehl **logout**, wird der neu gestartete Befehl dadurch automatisch auch beendet.

In manchen Fällen kann es jedoch auch gewünscht sein, daß ein Prozeß auch nach dem Abmelden weiterläuft. Diese Möglichkeit bietet uns das Programm **nohup**. Es startet einen Befehl, und ignoriert dann das Hangup-Signal (SIGHUP) für diesen Prozeß. Dieses Signal wird gewöhnlich beim Abmelden an alle Prozesse gesendet, um ihnen mitzuteilen, daß die Sitzung jetzt geschlossen wird. (Hangup steht für Auflegen, also das Beenden einer Modemverbindung)

Die Aufrufform von **nohup** ist einfach, außer den Kommandozeilenparametern `--help` und `--version` existieren keine weiteren Schalter. Dem Programmnamen wird einfach der Name des zu startenden Befehls mitgegeben:

```
nohup Befehl
```

Dadurch wird der angegebene Befehl ausgeführt, und bleibt auch nach dem Abmelden weiter aktiv im Speicher. Die Ausgaben des gestarteten Prozesses werden nicht mehr auf die Standard-Ausgabe geschrieben (weil die ja nach dem Abmelden gar nicht mehr zur Verfügung steht), sondern an die Datei `nohup.out` angehängt. Der Prozeß hat also seine Verbindung zum Terminal verloren, er kann nur noch über seine ProzeßID angesprochen werden.

Überwachung von Prozessen

Laufende Prozesse können überwacht werden mit den Programmen ps und top.

Das Programm ps bietet einen Schnappschuß der gerade laufenden Prozesse. Es zeigt verschiedene Daten dieser Prozesse an, das wichtigste ist dabei die ProzessID (PID) des jeweiligen Prozesses. Ohne Parameter aufgerufen zeigt ps nur die eigenen Prozesse. Häufige Kombinationen von Parametern sind

`ps uax`

Zeigt eine Liste aller (a) Prozesse des Systems inclusive der Angabe der User (u) denen die Prozesse gehören, es werden auch

daemon-Prozesse ohne eigenes Terminal (x) angezeigt.

`ps fax`

Zeigt eine Liste aller (a) Prozesse des Systems wieder mit den daemon-Prozessen (x) an. Die Prozesse werden als Baumstruktur (f) angezeigt, die klar macht, welche Prozesse von welchen Eltern-Prozessen abstammen.

Im Gegensatz zu ps zeigt top eine ständig aktualisierte Liste der Prozesse mit der höchsten CPU-Auslastung. Top ist ein interaktives Programm, das auch Modifikationen der Prozesse zulässt. top läuft solange, bis es mit q beendet wird.

Ansprechen und Beenden von Prozessen

Unter Linux können laufende Prozesse angesprochen werden, genauer gesagt können ihnen Signale geschickt werden. Wie die jeweiligen Prozesse auf Signale reagieren hängt davon ab, was der Programmierer des jeweiligen Programms vorgesehen hat.

Die wichtigste Aufgabe von Signalen ist es, Prozesse zu beenden. Das Programm zum Versenden von Signalen heißt daher folgerichtig kill. Es gibt aber auch viele andere Aufgaben für Signale, so kann etwa ein Prozess durch ein Signal dazu aufgefordert werden, siene Konfigurationsdatei neu einzulesen. Damit kann ein laufender Prozess verändert werden, ohne ihn neu starten zu müssen.

Das Programm kill erwartet eine oder mehrere ProzessIDs als Parameter, die die Prozesse beschreiben, denen das Signal geschickt werden soll. Wenn kein spezielles Signal angegeben wird, so schickt kill den Prozessen das Signal Nr. 15 (SIGTERM), das einen Prozess auffordert, sich selbst zu beenden. Es gibt ein Signal, das immer "tödlich" ist, es hat die Nummer 9 (SIGKILL).

Die Syntax von kill ist einfach:

```
kill [-Signal] PID [PID ...]
```

Ein Parameter, mit führendem Bindestrich wird als Signal interpretiert, alle anderen als PIDs. Signale können entweder in ihrer numerischen Form oder über ihern Namen angegeben werden - also z.B. entweder `-9` oder `-KILL`.

Eine Liste aller unterstützten Signale bekommen Sie mit `kill -l`

1.103.6

Modifizeren von Prozeßprioritäten

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Prozeßprioritäten zu verwalten. Dieses Lernziel beinhaltet das Ausführen von Prozessen mit höherer oder niedrigerer Priorität, das Bestimmen der Priorität eines Prozesses und das Ändern der Priorität eines laufenden Prozesses.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **nice**
- **ps**
- **renice**
- **top**

Ausführen von Programmen mit verschiedenen Prioritäten

Prozesse, die ohne spezielle Handhabe gestartet werden, haben eine "normale" Priorität, das heißt, sie sind mit einem sogenannten Nice-Wert von 0 ausgestattet. Ein Prozess ist dann "nice" (engl. für nett), wenn er anderen Prozessen mehr Rechenzeit übriglässt, also selber weniger verbraucht.

Sowohl das Programm ps, als auch top sind in der Lage, die NICE-Werte eines Prozesses darzustellen.

Ein normaler User kann Programme mit einem höheren Nice-Wert starten, indem er das Programm nice benutzt. Dieses Programm startet Prozesse mit einem veränderten Nice-Wert, also mit mehr oder weniger Priorität.

Hoher Nice-Wert heißt geringere Priorität

Der Superuser (root) kann Prozesse auch mit negativen Nice-Werten starten, also ihre Priorität erhöhen.

Der maximale Nice-Wert ist 19, der minimale -20.

Das Programm `nice` selbst ist einfach anzuwenden, es wird aufgerufen mit dem gewünschten Nice-Wert und der Angabe, welches Programm gestartet werden soll. Wollen wir also etwa als Normaluser das Programm `foo` mit einem Nicewert von 15 starten, dann schreiben wir:

```
nice -n15 foo
```

oder einfach

```
nice -15 foo
```

Vorsicht: im letzten Beispiel geht es nicht um -15 sondern um 15. Der Bindestrich hat nur die Aufgabe, dem nice-Programm klar zu machen, daß es sich nicht um das Programm 15 handelt! Damit der Superuser auch negative Werte angeben kann, muß er den Parameter -n benutzen. Damit könnte er also schreiben:

```
nice -n-15 foo
```

und der Nice-Wert wäre auf -15 gesetzt.

Das Programm nice ist nur gemacht, um Programme mit unterschiedlichen Prioritäten zu starten. Ein nachträgliches Ändern der Priorität ist damit nicht möglich.

Festlegen und Verändern der Prioritäten laufender Prozesse

Um die Priorität bzw. den Nice-Wert bereits laufender Prozesse zu verändern, gibt es zwei Möglichkeiten. Zum einen das speziell zu diesem Zweck entwickelte Programm renice und zum anderen das Programm top, das wir schon bei der Prozessbeobachtung kennengelernt haben.

Beide Programme erlauben es, daß ein laufender Prozess seinen Nice-Wert verändert bekommt. Normaluser dürfen den Nice-Wert bei beiden Programmen nur vergrößern. Das heißt, das auch ein Normaluser, der selbst seinem eigenen Prozess einen Nicewert von beispielsweise 10 gegeben hat, diesen Wert nicht mehr auf 5 runtersetzen kann.

Das Programm renice ist im Gegensatz zu top auch in der Lage, alle Prozesse eines Users oder einer Gruppe gleichzeitig zu verändern. top kann dagegen nur einzelnen PIDs einen neuen Nice-Wert zuweisen.

1.103.7

Durchsuchen von Textdateien mittels regulärer Ausdrücke

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Dateien und Textdaten mittels regulärer Ausdrücke zu bearbeiten. Dieses Lernziel beinhaltet das Erzeugen einfacherer regulärer Ausdrücke mit verschiedenen Elementen. Ebenfalls enthalten ist die Verwendung von Regex-Werkzeugen zum Durchführen von Suchen über ein Dateisystem oder Dateiinhalte.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **grep**
- **regexp**
- **sed**

Das Grundprinzip regulärer Ausdrücke

Reguläre Ausdrücke (regular expressions - regexp) sind nah verwandt mit den Wildcards, aber nicht das Selbe. Es handelt sich um Suchmuster, die auch eine Art Jokerzeichen beinhalten, jedoch wesentlich mächtiger sind, als die Wildcards der Shell.

Die unterschiedlichsten Programme unter Linux können mit regulären Ausdrücken umgehen, allerdings unterscheiden sie sich manchmal im Detail, in der Interpretation. Es gibt aber eine gemeinsame Grundmenge von Ausdrücken, die von allen Programmen verstanden werden.

Die LPI-101-Prüfung verlangt nur das Wissen um einfache reguläre Ausdrücke, die die folgenden Elemente enthalten dürfen:

Ausdruck	Bedeutung
c	Ein einzelner Buchstabe passt auf sich selbst.
.	Ein Punkt steht für genau einen beliebigen Buchstaben, außer das Zeilenende.

\?	Das dem Operator \? voranstehende Zeichen null oder einmal.
*****	>Das dem Operator * voranstehende Zeichen null mal oder öfter.
\+	Das dem Operator \+ voranstehende Zeichen ein mal oder öfter.
^	Das Caret (^) bedeutet Zeilenanfang.
$	Das Dollarzeichen ($) bedeutet Zeilenende.
\<	Bedeutet Wortanfang.
\>	Bedeutet Wortende.
[*Buchstaben*]	Ein Buchstabe aus der Menge.
[^*Buchstaben*]	Ein Buchstabe, der NICHT in der Menge steht.
\(...\)	Zusammenfassung eines Ausdrucks. Wichtig zum Ersetzen.
|	Ein logisches ODER mit dem verschiedene Ausdrücke verknüpft werden können.
****	Jedes Zeichen nach dem Backslash verliert seine Sonderbedeutung.

Mit Hilfe dieser Ausdrücke sind sehr komplexe Suchmuster möglich, nehmen wir ein Beispiel:

Wir haben eine Datei mit Namen und wir wollen alle Formen von Maier (Maier, Mayer, Meier, Meyer, Mayr) ansprechen. Der reguläre Ausdruck würde nun lauten:

```
M[ae][iy]e\?r
```

Wir suchen also ein Wort, das mit einem `M` beginnt, danach entweder ein `a` oder ein `e` vorweist. Dem folgt entweder ein `i` oder ein `y`. Dann kommt ein oder kein `e`, gefolgt vom Buchstaben `r`.

Wichtig ist zu wissen, daß die einzelnen Ausdrücke auch kombinierbar sind. So bedeutet etwa `.*` *eine Folge von beliebigen Zeichen, auch die leere Folge.* Warum? Ganz einfach. Der Punkt meint *ein beliebiges Zeichen*, das Sternchen meint *das dem Sternchen vorangehende Zeichen belibig oft, auch Null mal*. In Kombination wird daraus eben die beliebige Folge beliebiger Zeichen.

Zu beachten ist, daß diese Suchmuster viele Zeichen enthalten, die von der Shell interpretiert würden. Wenn wir also ein solches Muster auf der Kommandozeile eingeben, dann müssen wir es in Anführungszeichen setzen.

Aber genug der Theorie, jetzt folgen die zwei Anwendungen, mit denen wir diese Ausdrücke nutzen können:

Dateien durchsuchen mit grep

Das Programm grep durchsucht entweder Dateien oder einen Datenstrom (seine Standard-Eingabe) nach einem regulären Ausdruck. Dabei wird standardmäßig jede Zeile, die den Ausdruck enthält auf die Standard-Ausgabe geschrieben.

Damit ist es sowohl möglich, bestimmte Dateien zu durchsuchen, als auch die Ausgabe anderer Programme. Im einfachsten Fall durchsucht grep die Dateien nur nach einem festen Suchbegriff. Wenn Sie z.B. alle Useraccounts sehen wollen, die als Startshell die bash haben, könnten Sie das mit grep ganz leicht erledigen:

```
grep bash /etc/passwd
```

Das heißt also, die Datei /etc/passwd wird nach dem Auftreten des Begriffs "bash" durchsucht und alle Zeilen, die den Begriff enthalten werden ausgegeben. Wenn mehrere Dateien durchsucht werden sollen, dann gibt grep in der Ausgabe auch noch den Dateinamen der Datei aus, in der der Suchbegriff gefunden wurde.

Wenn grep einen Datenstrom durchsuchen soll, so wird ihm einfach kein Dateiname mitgegeben. Ein Beispiel: Wir wollen alle Prozesse des Users "hans" aufgelistet bekommen. Der Befehl "ps uax" listet uns alle laufenden Prozesse samt ihrer Eigentümer auf. Der Befehl

```
ps uax | grep hans
```

filtert die Ausgabe des ps uax Befehls durch grep und gibt nur die Zeilen aus, die den Begriff "hans" enthalten. Allerdings werden wir hier auch noch die Zeile angezeigt bekommen, die den Prozess "grep hans" beschreibt. Um das zu vermeiden, bedienen wir uns der regulären Ausdrücke:

```
ps uax | grep "^hans"
```

Das "^" Zeichen steht in einem regulären Ausdruck für den Zeilenanfang. Also suchen wir jetzt nach dem Auftauchen des Wortes "hans" am Zeilenanfang. Jetzt haben wir tatsächlich nur die Prozesse des Users hans.

Ein wichtiger Parameter von grep ist das `-v`. Mit dieser Kommandozeilenoption gibt grep nur die Zeilen aus, die **nicht** den Suchbegriff enthalten.

Anstatt eines einfachen Suchbegriffs können wir natürlich auch mit komplexen regulären Ausdrücken suchen. Um etwa aus einer Datei mit Adressangaben alle Adressen aller Formen des Namens Maier zu suchen, könnten wir schreiben:

```
grep "[Mm][ae][iy]e\?r" Adressen.txt
```

Bitte beachten Sie die Anführungszeichen. Sie verhindern, daß die Shell selbst versucht, die eckigen Klammerm und das Fragezeichen als Wildcard zu interpretieren. Grundsätzlich ist es von Vorteil, wenn Sie den Suchbegriff in solche Anführungszeichen setzen, sobald er mehr

als nur Buchstaben und Zahlen beinhaltet.

Suchen und Ersetzen mit sed

Mit grep haben wir Dateien nach Suchbegriffen durchsucht, wir konnten diese Begriffe aber nicht ändern. Dazu sind Editoren notwendig. Alle Unix-Editoren wie etwa vi, ex, ed oder emacs können reguläre Ausdrücke verwenden, um ein "Suchen und Ersetzen" durchzuführen. Aber eine kommandozeilenorientierte Funktion, die dieses Ersetzen bietet, wird mit einem ganz speziellen Editor durchgeführt, dem Stream-Editor oder kurz sed.

Dieser Editor ist dazu entworfen, einen Datenstrom oder eine Datei automatisch zu bearbeiten. Dazu werden bestimmte Befehle, die auf diese Datei oder diesen Datenstrom anzuwenden sind, entweder in einer Datei formuliert oder gleich auf der Kommandozeile miteingetragen. Der Stream-Editor bearbeitet dann die Datei oder den Datenstrom zeilenweise und führt die jeweiligen Befehle darauf aus. Das Ergebnis wird dann wiederum auf die Standard-Ausgabe geschrieben. Wir haben das schon auf der Seite über die Textfilter besprochen.

Die mit Abstand häufigste Anwendung dieses Stream-Editors ist das Suchen und Ersetzen mit regulären Ausdrücken. Diese Vorgehensweise soll hier eingehend beschrieben werden.

Ein sed-Kommando ist immer in einen Adressbereich und einen Befehl aufgeteilt. Das Adressfeld kann dabei folgende Werte aufnehmen:

Eine Zahl *n*

Eine einfache Zahl *n* beschreibt die Zeile *n*. Der folgende Befehl wird nur auf diese Zeile angewandt.

Ein Zahlenbereich *n,m*

Ein Bereich meint die Zeilen *n* bis *m*. Start- und Endzeile werden durch Komma getrennt. Ein Dollarzeichen ($) meint die letzte Zeile. So bedeutet also `1,$` alle Zeilen der Datei (von der Zeile 1 bis zur letzten Zeile). Der folgende Befehl wird auf jede Zeile angewandt, die innerhalb des genannten Bereiches liegt.

Ein regulärer Ausdruck */Ausdruck/*

Wenn als Adresse ein regulärer Ausdruck steht, dann ist jede Zeile gemeint, die ein Element enthält, das auf den Ausdruck passt. Damit sed das Konstrukt als Ausdruck erkennt, muß der Ausdruck in Slash-Zeichen (/) geklammert sein.
Wenn ein regulärer Ausdruck innerhalb einer Adressbereichsangabe steht, dann ist immer die erste Zeile gemeint, auf die der Ausdruck passt.
Damit ein Ausdruck auch nach dem Auftreten von Slashes suchen kann, erlaubt sed auch die Verwendung einer anderen Klammerung, in der Form `\#`, wobei für # jedes beliebige Zeichen stehen kann. Also ist auch die Konstruktion `\+Ausdruck\+` eine legale Angabe einer Adresse.

Wird bei sed der Adressteil weggelassen, so wird jede Zeile des Datenstroms bearbeitet, es wird also die Adresse `1,$` angenommen.

Der Befehl zum Suchen und Ersetzen für sed ist das `s`, das wie folgt angewandt werden muß:

```
s/Suchbegriff/Ersetzung/[Optionen]
```

Die Angabe der Optionen ist optional (daher die eckigen Klammern), der abschließende Slash nach dem Ersetzungsteil ist aber zwingend, auch wenn keine Optionen angegeben werden sollen.

Als Optionen stehen zur Verfügung:

Eine Zahl *n*

Eine Zahl von 1 bis 512 ersetzt nur das *n*-te Auftreten des Musters.

g

Global - jedes Auftreten des Begriffes innerhalb einer Zeile wird bearbeitet. Ohne diese Option wird immer nur das erste Auftreten des Begriffs bearbeitet.

p

Print -wenn eine Ersetzung stattgefunden hat, wird der Inhalt des Arbeitsspeichers von sed in die Standardausgabe geschrieben

Im einfachsten Fall können wir also z.B. das Auftreten des Wortes "Huber" in "Herr Huber" mit dem folgenden Befehl ersetzen:

```
s/Huber/Herr Huber/g
```

Wie wird das nun angewandt? Entweder, wir geben diesen Befehl gleich auf der Kommandozeile ein (mit der Option `-e`, dann würde das etwa folgendermaßen aussehen:

```
cat Datei.txt | sed -e "s/Huber/Herr Huber/g" > Datei2.txt
```

oder einfacher

```
sed -e "s/Huber/Herr Huber/g" Datei.txt > Datei2.txt
```

Der Editorbefehl wurde hier in Anführungszeichen gesetzt, weil er ein Leerzeichen enthält. Wie schon bei grep ist es grundsätzlich zu empfehlen, diese Konstruktion in Anführungszeichen zu setzen um die Shell davon abzuhalten, Sonderzeichen zu interpretieren.

Was hat unser Befehl jetzt bewirkt? Jedes Vorkommen des Wortes "Huber" in der Datei `Datei.txt` wurde in "Herr Huber" verwandelt. Die gesamte Datei mit den Änderungen wurde in die Datei `Datei2.txt` geschrieben.

Wir hätten den Befehl aber auch in eine separate Datei (nennen wir sie mal `Befehle`) schreiben können. Das macht insbesondere dann Sinn, wenn es sich um mehrere Befehle handelt, oder wir die Ersetzung immer wieder brauchen werden. Dann hätte der sed-Aufruf einfach statt der Option `-e` ein `-f` enthalten, gefolgt vom Namen der Befehlsdatei:

```
sed -f Befehle Datei.txt > Datei2.txt
```

Die eigentliche Stärke dieses Suchen und Ersetzen liegt aber natürlich wieder in der Angabe von regulären Ausdrücken. Es können hier beliebige Ausdrücke verwendet werden, wie sie oben beschrieben sind. Als besondere Möglichkeit sei hier noch auf die Klammerung mit der Konstruktion `\(...\)` verwiesen. Auf jedes Element, das im Suchen-Teil derart geklammert ist, kann im Ersetzen-Teil wieder zugegriffen werden. Dazu muß nur eine Nummer *n* mit vorgestelltem Backslash (\) angegeben werden. Das meint die *n*-te Klammer. Ein Beispiel:

Wir haben eine Datei `Namen.txt` mit folgendem Inhalt:

```
Hans Müller
Peter Schmidt
Michael Huber
Gabi Maier
Thomas Gruber
```

Wir wollen jetzt die Namen in der Form Nachname, Vorname haben. Kein Problem mit sed. Wir schreiben

```
sed -e "s/\(.*\) \(.*\)/\2, \1/" Namen.txt > Namen2.txt
```

Schon steht in der Datei `Namen2.txt`

```
Müller, Hans
Schmidt, Peter
Huber, Michael
Maier, Gabi
Gruber, Thomas
```

Was ist passiert? Schauen wir uns den sed-Befehl einmal genauer an. Der reguläre Ausdruck `.*` steht für eine beliebige Zeichenfolge beliebiger Länge. Also können wir die zwei Namen (Vor- und Nachname) auch als `.* .*` angeben. Also zwei beliebige Zeichenfolgen, getrennt durch ein Leerzeichen. Jetzt klammern wir die beiden Konstrukte jeweils mit `\(...\)` ein. So entsteht zweimal der Ausdruck `\(.*\)`. Auf jeden dieser geklammerten Ausdrücke können wir jetzt im Ersetzen-Teil des sed-Befehls wieder zugreifen, mit `\1` für den ersten gefundenen Ausdruck und `\2` für den zweiten.

Dadurch entsteht folgende Zuordnung, am Beispiel der ersten Zeile:

`\(.*\)`	`\(.*\)`
Hans	Müller

```
\1      \2
```

Im Ersetzen-Teil schreiben wir einfach `\2, \1`, das wird dann eben ersetzt durch den Inhalt der zweiten Klammer, gefolgt von einem Komma, einem Leerzeichen und dem Inhalt der ersten Klammer. Also in unserem Beispiel durch `Müller, Hans`.

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

1.103.8

Allgemeine Dateibearbeitung mit vi

Beschreibung: Prüfungskandidaten müssen in der Lage sein, Textdateien mit **vi** zu bearbeiten. Dieses Lernziel beinhaltet Navigation im vi, die grundlegenden vi Modi, Einfügen, Bearbeiten, Löschen, Kopieren und Auffinden von Text.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **vi**
- `/, ?`
- `h,j,k,l`
- `G, H, L`
- `i, c, d, dd, p, o, a`
- `ZZ, :w!, :q!, :e!`
- `:!`

Der Unix-Standard-Editor ist - unabhängig vom verwendeten System - immer das Programm **vi**. Dieser Editor ist auf allen Terminals lauffähig, das bedeutet aber auch, daß er sich nicht auf die Existenz bestimmter Funktionstasten verlassen kann, um bestimmte Befehle auszuführen. Einfache Terminals haben nicht einmal Pfeiltasten, der Editor **vi** muß also auch die Möglichkeit haben, mit anderen Befehlen Aktionen wie die Bewegungen des Cursors vorzunehmen.

Aus diesem Grund kennt der Editor **vi** verschiedene Modi. Die beiden wichtigsten sind der Kommandomodus und der Eingabemodus. Im Kommandomodus werden alle eingegebenen Zeichen als Befehle interpretiert, statt sie als in den Text einzugebende Zeichen zu interpretieren. Im Eingabemodus werden alle Zeichen an der Cursorposition in den Text eingefügt. Der Wechsel vom Eingabemodus in den Kommandomodus erfolgt immer durch die **Esc**-Taste. Beim Aufruf von **vi** befinden wir uns grundsätzlich im Kommandomodus.

Der Editor **vi** wird in der Regel zusammen mit einem Dateinamen aufgerufen.

```
vi Datei
```

Dadurch wird **vi** angewiesen, die angegebene Datei zu editieren. Existiert die Datei noch nicht, so startet **vi** mit einem leeren Puffer, beim Speichern wird dann aber der angegebene Dateiname verwendet.

Befehle zum Bewegen des Cursors im Kommandomodus

Nachdem **vi** nicht davon ausgehen kann, daß dem Terminal Pfeiltasten zur Verfügung stehen, muß er Befehle kennen, die die Aufgabe dieser Pfeiltasten übernehmen. Dazu dienen die Tasten `h j k l`, die im Kommandomodus folgende Bedeutung haben:

- **h** Bewegt den Cursor nach links
- **j** Bewegt den Cursor nach unten
- **k** Bewegt den Cursor nach oben
- **l** Bewegt den Cursor nach rechts

Jedem dieser Befehlszeichen kann eine Zahl vorangestellt werden, die angibt, wie oft der Befehl ausgeführt werden soll. Der Befehl `23j` bewegt den Cursor also 23 Zeilen nach unten.

Neben diesen grundlegenden Befehlen gibt es noch weitere Möglichkeiten den Cursor zu bewegen:

- **w** Bewegt den Cursor auf den Beginn des nächsten Wortes
- **b** Bewegt den Cursor auf den Beginn des aktuellen Wortes
- **0** Bewegt den Cursor auf den Beginn der aktuellen Zeile
- **$** Bewegt den Cursor auf das Ende der aktuellen Zeile
- **Strg-f** (Forward) Eine Bildschirmseite vorwärts (PgDn)
- **Strg-b** (Back) Eine Bildschirmseite zurück (PgUp)
- **G** Bewegt den Cursor zum Dateiende (letzte Zeile)
- ***n*G** Bewegt den Cursor in die *n*te Zeile der Datei
- **H** Bewegt den Cursor in die erste Zeile des aktuellen Bildschirms
- ***n*H** Bewegt den Cursor in die *n*te Zeile des aktuellen Bildschirms
- **L** Bewegt den Cursor in die letzte Zeile des aktuellen Bildschirms
- ***n*L** Bewegt den Cursor in die *n*te Zeile des aktuellen Bildschirms von unten

Auch diese Befehle (bis auf 0) können alle nach einer Zahl auftreten, die angibt, wie oft der Befehl ausgeführt werden soll. Mit `3w` bewegt sich der Cursor also auf den Beginn des dritten Wortes nach dem aktuellen Wort. Bei den Befehlen `G, H` und `L` ist angegeben, was die Zahl jeweils bedeutet.

Die modernen Versionen von **vi**, die unter Linux zum Einsatz kommen, kennen natürlich auch dei entsprechenden Funktionstasten der PC-Tastatur. Es ist also sehr wohl möglich, die Pfeiltasten, Home, End, PgUp und PgDn zu verwenden. Trotzdem sollten die hier besprochenen Befehle bekannt sein, zumindestens die vier grundlegenden Cursorbewegungen `h j k l`.

Befehle zum Wechsel in den Eingabemodus

Ein Editor ist dazu gedacht, Text in eine Datei zu schreiben, die wichtigsten Befehle sind also die, wie vom Kommandomodus in den Eingabemodus gewechselt werden kann. **vi** bietet hierfür viele verschiedenen Möglichkeiten an, die sich auch wieder aus der alten terminalbasierten Arbeitsweise erklären. Heute - mit Verwendung der Pfeiltasten - können wir bedenkenlos im Eingabemodus den Cursor bewegen. Früher - ohne Pfeiltasten - musste für jede Bewegung des Cursors in den Kommandomodus gewechselt werden. Aus diesem Grund gibt es so viele verschiedene Möglichkeiten, in den Eingabemodus zu wechseln:

- **i** Einfügen vor der aktuellen Cursorposition
- **I** Einfügen am Anfang der aktuellen Zeile
- **a** Einfügen nach der aktuellen Cursorposition
- **A** Einfügen am Ende der aktuellen Zeile
- **o** Einfügen nach der aktuellen Zeile
- **O** Einfügen vor der aktuellen Zeile
- **R** Überschreiben des Textes (Replace)
- **C** Ersetzen der aktuellen Zeile ab Cursorposition

Achtung: Auch diesen Befehlen kann eine Zahl vorangestellt werden. Der Befehl `23i` führt zwar auch nur einmal in den Eingabemodus, die geschriebenen Zeilen werden aber 23 mal eingefügt.

Suchen im Text

Um ein bestimmtes Wort - bzw. einen regulären Ausdruck - zu suchen, bietet **vi** zwei Methoden:

- **/*Suchmuster*** Sucht nach dem *Suchmuster* von der Cursorposition in Richtung Dateiende und bewegt den Cursor auf das gefundene Ergebnis.
- **?*Suchmuster*** Sucht nach dem *Suchmuster* von der Cursorposition in Richtung Dateianfang und bewegt den Cursor auf das gefundene Ergebnis.

Beiden Methoden kann wiederum eine Zahl vorausgehen, die dann den Cursor nicht zum ersten, sondern zum *n*ten Auftreten des Suchmusters bewegt.

Löschen von Text

Um aus dem Kommandomodus heraus Text aus der Datei zu löschen, existieren viele verschiedenen Möglichkeiten. die wichtigsten sind

x	Löscht das Zeichen, auf dem der Cursor gerade steht.
dl	Löscht das Zeichen, auf dem der Cursor gerade steht (wie x).
dd	löscht die aktuelle Zeile
dw	löscht von der Cursorposition bis zum Anfang des folgenden Wortes
db	löscht von der Cursorposition nach links bis zum Anfang des aktuellen Wortes
d$	löscht von der Cursorposition bis zum Zeilenende
d0	löscht von der Cursorposition nach links bis zum Anfang der aktuellen Zeile

Die Löschbefehle speichern den gelöschten Text in einem Zwischenpuffer. Es ist möglich, diesen Zwischenpuffer wieder an einer beliebigen Stelle des Textes auszugeben. Somit ist auch eine Art Kopieren oder Verschieben von Textblöcken möglich. Die dazu nötigen Befehle sind:

p	Fügt den Inhalt des Zwischenpuffers nach der aktuellen Cursorposition ein.
P	Fügt den Inhalt des Zwischenpuffers vor der aktuellen Cursorposition ein.
yy	Liesst eine Zeile in den Zwischenpuffer, ohne sie zu löschen.

Es gibt auch noch eine raffinierte Methode, Text zu löschen oder zu ersetzen. Dazu wird ein Befehl zusammen mit einem der Befehle zur Cursorbewegung benutzt. Das Löschen bzw. Ersetzen bezieht sich dann auf den Bereich zwischen der aktuellen Cursorposition und dem Punkt, an den der Befehl zur Cursorbewegung den Cursor bewegt hätte:

d*Cursor-Bewegung*	Löscht den Bereich zwischen aktueller Cursorposition und dem Punkt, an den die *Cursor-Bewegung* den Cursor geführt hätte.
c*Cursor-Bewegung*	Ersetzt den Bereich zwischen aktueller Cursorposition und dem Punkt, an den die *Cursor-Bewegung* den Cursor geführt hätte durch den Eingabetext.

Der Befehl `d3k` würde also den Text zwischen der aktuellen Cursorposition bis 3 Zeilen weiter nach oben (3k bewegt den Cursor 3 Zeilen nach oben) löschen. Als *Cursor-Bewegung* kann jeder beliebige Befehl angegeben werden, der den Cursor bewegt, also auch ein Suchbefehl.

Beenden und Sichern

Die meisten Befehle des **vi** zum Beenden und Sichern sind sogenannte Ex-Befehle, die durch einen führenden Doppelpunkt eingeleitet werden. Diese Befehle verweigern unter bestimmten Umständen ihren Dienst, wenn ihnen kein Ausrufungszeichen angefügt wird. So kann der **vi** nicht mit `:q` beendet werden, wenn der Text, den der Editor bearbeitet hat, verändert wurde, aber noch nicht gesichert ist. Durch das Anhängen eines Ausrufungszeichen an den Befehl (`:q!`) wird diese Einschränkung aufgehoben. Folgende Befehle sind in diesem Zusammenhang interessant:

ZZ	Sichert die Datei und beendet den Editor
:w *Dateiname*	Sichert die Datei. Wurde beim Aufruf von **vi** schon ein Dateiname angegeben, kann er hier weggelassen werden.
:w! *Dateiname*	Wie `:w`. Speichert aber auch, wenn es Gründe gibt, die das Speichern normalerweise verunmöglichen (Datei existiert schon)
:wq *Dateiname*	Speichern der Datei und Beenden des Editors (auch hier kann ein ! angegeben werden, wenn ansonsten das Speichern unmöglich wäre). Der Dateiname kann weggelassen werden, wenn er schon bekannt ist.
:x *Dateiname*	Wie `:wq`. Datei wird aber nur gespeichert, wenn es Veränderungen gab.
:q	Beendet den Editor
:q!	Beendet den Editor auch wenn vorher nicht gesichert wurde.

Sonstige wichtige Kommandos

Neben den bereits vorgestellten Befehlen gibt es noch hunterte anderer. Hier seien nur noch ein paar genannt, die wirklich wichtig sind:

:e *Dateiname*	Läd die angegebene Datei in den Editor. Nur ausführbar, wenn die aktuelle Datei schon gesichert wurde.
:e! *Dateiname*	Läd die angegebene Datei in den Editor. Auch ausführbar, wenn die aktuelle Datei noch nicht gesichert wurde.
:r *Dateiname*	Läd die angegebene Datei an der aktuellen Cursorposition in den Text.
:! *Shellbefehl*	Führt den angegebenen Shellbefehl aus und kehrt dann zum Editor zurück.
:*Bereich*! *Shellbefehl*	Filtert den angegebenen Bereich durch das angegebene Shellkommando und ersetzt ihn durch die Ausgabe des Shellkommandos. Bereiche werden durch ihre Zeilennummern angegeben in der Form *erste_Zeile,letzte_Zeile*. Ein Dollarzeichen bezeichnet die letzte Zeile der Datei. Soll die ganze Datei gefiltert werden, kann also die Angabe `1,$` benutzt werden. Die aktuelle Zeile wird durch einen Punkt (.) repräsentiert.
u	Undo der letzten Aktion.

Das I-Node System

Unix-Dateisysteme sind nach einem anderen Prinzip aufgebaut, als etwa DOS-FAT-Systeme. Es gibt zwar viele verschiedene Dateisysteme unter Unix, gemeinsam haben sie jedoch eben das Prinzip der Funktionsweise. Und dieses Prinzip beruht auf den sogenannten I-Nodes.

Jede Partition enthält ein Dateisystem, dieses Dateisystem wiederum enthält eine Art Inhaltsverzeichnis, die *I-Node-Liste*. Die einzelnen Elemente der *I-Node-Liste* sind die Dateiköpfe, also die Orte wo Dateiattribute, Größe usw. gespeichert sind. Diese Dateiköpfe werden *I-Nodes* genannt.

Wie unter DOS auch, so verwalten Unix-Dateisysteme nicht zwangsläufig die Sektoren einer Festplattenpartition sondern Zuordnungseinheiten, die hier aber nicht Cluster sondern *Block* heißen. Beim Anlegen eines Dateisystems kann die Blockgröße angegeben werden, die auf dieser Partition verwendet werden soll. Typische Blockgrößen sind 512, 1024 oder 2048 Byte. Voreingestellt sind meist 1024 Byte pro Block.

Anders als unter DOS werden diese Blöcke aber nicht in einer Tabelle (FAT) zusammengefasst, sondern die I-Nodes enthalten selbst die Verweise auf diese Blöcke. Das Format eines typischen I-Nodes sieht etwa so aus:

Typ und Zugriffsrechte
Anzahl der Hardlinks
Benutzernummer (UID)
Gruppennummer (GID)
Größe der Datei in Bytes
Datum der letzten Veränderung (mtime)
Datum der letzten Statusänderung (ctime)
Datum des letzten Zugriffs (atime)
Adresse von Datenblock 0
...
Adresse von Datenblock 9
Adresse des ersten Indirektionsblocks

Adresse des Zweifach-Indirektionsblocks
Adresse des Dreifach Indirektionsblocks

Nach den Datumsfeldern stehen zehn Felder, die direkt die Adressen der Datenblöcke beinhalten können. Benutzt die Datei weniger Platz sind die Felder einfach leer.

Ist die Datei größer als 10 Blöcke (also größer als 10*1024 oder 10*2048 oder 10*512), so enthält das nächste Feld der I-Node eine Adresse eines Blockes, der wiederum bis zu 128 Adressen anderer Blöcke enthalten kann. Sollte das auch noch nicht ausreichen, so enthält der nächste I-Node-Eintrag eine Adresse eines Zweifach-Indirektionsblocks, eines Blocks, der bis zu 128 Adressen auf Blöcke mit wiederum 128 Adressen enthält. Und falls auch das noch zu wenig sein sollte, so enthält der nächste Eintrag die Adresse eines Blockes, der wiederum 128 Adressen von Zweifach-Indirektionsblöcken enthalten kann. Damit sind dann Dateigrößen von 1, 2 oder 4 Gigabyte (je nach Blockgröße von 512, 1024 oder 2048 Byte) möglich.

Zu beachten ist, daß der Dateiname nicht in der I-Node auftaucht. Die I-Node ist sozusagen nur die Referenz auf die Datenblöcke, die eine Datei benutzt und der Ort, an dem die Attribute wie Eigentümer, Gruppe, Größe und Zugriffsdaten gespeichert sind.

Je nach verwendetem Dateisystem liegt das Wurzelverzeichnis auf einer festgelegten I-Node, meist 1 oder 2. Grundsätzlich ist es aber dem Dateisystem bekannt, welche I-Node das Wurzelverzeichnis enthält.

Jedes Verzeichnis (Ordner, Directory) - auch das Wurzelverzeichnis ist unter Unix nichts anderes als eine Datei, deren Inhalt einfach die Dateinamen der enthaltenen Dateien samt ihren I-Node-Nummern enthält. Damit wird auch klar, warum Unix kein Problem mit Hard-Links hat, also mit Dateien mit mehreren Namen. Es handelt sich ja nur um verschiedene Namenseinträge, die eben die selbe I-Node-Nummer besitzen.

Das Standard Linux-Dateisystem ext2 hat zusätzlich zu den gezeigten I-Node Einträgen noch verschiedene andere, die das System in mancherlei Hinsicht noch leistungsfähiger macht. Für den Anwender ergenen sich dadrch keinerlei Veränderungen, in der Praxis sind aber einige positive Effekte feststellbar.

So benutzt das ext2 System beispielsweise bis zu 12 direkte Datenblockadressen, es hat noch ein zusätzliches Datumsfeld für das Datum des Löschens der Datei (für später zu entwickelnde Undelete-Funktion) und es bietet weitere Attribute, die hier nicht genauer dargestellt werden sollen weil das den allgemeinen Unix-Rahmen dieses Kurses sprengen würde.

Der Superblock

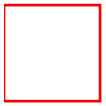
Ein Unix-Dateisystem besitzt einen sogenannten Superblock, einen Block, der die grundlegenden Informationen zum Dateisystem selbst enthält. Einige wichtige Daten des Superblocks sind:

- Die Größe des Dateisystems in Blöcken

- Die Größe der Blöcke in Bytes
- Zeiger auf den ersten freien Datenblock
- Zeiger auf erste freie I-Node
- Verschiedene Statusbits (Flags)

Auch hier unterscheiden sich die verschiedenen Unix-Dateisysteme voneinander, was an zusätzlichen Informationen im Superblock gespeichert ist. Das wesentliche an dieser Struktur ist, daß der Superblock beim Mounten eines Dateisystems in den Speicher gelesen wird und alle Veränderungen dort vorgenommen werden. Erst beim Wiederabhängen des Dateisystems werden diese Veränderungen physikalisch auf der Platte gespeichert. Das erklärt auch, daß es nach einem Systemabsturz zu Inkonsistenzen in einem Dateisystem kommen kann.

Noch vor dem Superblock steht der sogenannte Bootblock auf der Platte, der in etwa dem Bootsektor der DOS-Partitionen entspricht. Zusammengefasst kann man also ein Unix-Dateisystem etwa wie folgt darstellen:



[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Zurück

Das EXT2 Dateisystem

Im Wesentlichen entspricht das Second Extended Filesystem (EXT2) von Linux dem Unix-Inode-System, das an anderer Stelle schon beschrieben wurde. Es gibt aber ein paar wesentliche Unterschiede, die hier kurz erläutert werden sollen.

Der Aufbau einer EXT2 Inode

Die wesentlichen Unterschiede des EXT2 Systems zum "normalen" I-Node-System liegen in der Struktur der I-Nodes selbst. Hier sind einige Felder hinzugekommen, die für bestimmte Erweiterungen vorgesehen wurden, die im folgenden genauer erklärt werden. Doch zunächst einmal die Struktur der Inode selbst:

permission	**links**	**owner**	**group**
size		**creation time**	
modification time		**access time**	
deletion time		**blockcount**	
flags		**file version (NFS)**	
file ACL		**dir ACL**	
fragment addr.	**fr. size**	**frag. nr**	**reserved**
1. block data		**2. block data**	
3. block data		**4. block data**	
5. block data		**6. block data**	
7. block data		**8. block data**	
9. block data		**10. block data**	
11. block data		**12. block data**	
simple indirect		**double indirect**	
triple indirect		**reserved**	
reserved		**reserved**	

Die wesentlichen Unterschiede zum bisher beschriebenen I-Node-System sind:

- Neben den drei Standard-POSIX-Zeitmarken (ctime, atime, mtime) gibt es hier eine vierte Zeitmarke, die die Löschzeit (*deletion time*) speichert. Das ist gedacht für Spezialprogramme zum Wiederherstellen versehentlich gelöschter Dateien.
- Ein Feld für *flags* ist hinzugekommen. Diese Flags ermöglichen es, daß Dateien bestimmte Eigenschaften bekommen, die weiter unten genauer beschrieben werden.
- Der Eintrag *file version* kann vom NFS-Server zur Unterscheidung verschiedener Versionen einer Datei verwendet werden.
- Zwei Einträge für eine erweiterte Zugriffskontrolle für Dateien (*file ACL*) und Verzeichnisse (*dir ACL*) sind hinzugekommen. ACL steht für Access Control List.
- Die Unterstützung fragmentierter Datenzonen ist vorgesehen.
- Es stehen 12 direkte Adressfelder für Datenblocks zur Verfügung.

Die Flags des EXT2 Filesystems

Jede Datei in einem EXT2 Dateisystem kann neben den normalen Unix-Zugriffsrechten noch einen Satz zusätzlicher Flags bekommen, die hier noch erläutert werden sollen. Um sich die Flags (oder auch Attribute) anzusehen, wird das Programm **lsattr** benutzt, zum Ändern der Attribute wird **chattr** benutzt. Es gibt die Attribute A,a,c,d,i,S,s und u.

- Wenn eine Datei mit gesetztem A-Attribut verändert wird, so wird das *atime* Feld nicht verändert.
- Eine Datei, die das a-Attribut gesetzt hat, kann nur im Append-Mode geöffnet werden. Das heißt, Daten können an diese Datei angehängt werden, bestehende Daten jedoch nicht gelöscht.
- Eine Datei mit dem c-Attribut wird automatisch vom Kernel komprimiert, bevor sie auf die Platte geschrieben wird. Wenn sie gelesen wird, dann entkomprimiert der Kernel die Datei wieder, ein Anwender merkt also nichts von der Kompression.
- Eine Datei mit gesetztem d-Attribut ist **kein** Kandidat für ein Backup mit dem *dump* Befehl.
- Eine Datei mit gesetztem i-Attribut kann nicht verändert werden, sie kann nicht gelöscht oder umbenannt werden, kein Link kann auf sie erstellt werden und keine Daten können in ihr verändert werden. Dieses Attribut kann nur vom Superuser gesetzt oder entfernt werden.
- Wenn eine Datei mit gesetztem s-Attribut gelöscht wird, dann werden die Bytes der Datei mit Nullen überschrieben. Ein sicheres Löschen, das keine Wiederherstellung erlaubt.
- Wenn eine Datei, deren S-Attribut gesetzt ist, verändert wird, dann wird diese Veränderung synchron auf die Platte geschrieben, ohne lange im Cache zu liegen.
- Wenn eine Datei mit gesetztem u-Attribut gelöscht wird, dann wird ihr Inhalt gesichert, so daß sie wiederhergestellt werden kann.

Erweiterungen des Superblocks

Der Superblock eines EXT2-Dateisystems enthält ein sogenanntes *Valid-Flag*. Sobald dieses Dateisystem gemountet wird, wird das Valid-Flag gelöscht. Erst beim Umount wird es wieder gesetzt. Sollte das System abstürzen, so bleibt das Valid-Flag ungesetzt - beim nächsten Start kann also erkannt werden, daß dieses Dateisystem nicht ordnungsgemäß abgehängt worden ist.

Das Programm zur Überprüfung eines EXT2 Dateisystems (e2fsck) überprüft dieses Valid-Flag und prüft (falls es nicht durch die -f Option gezwungen wurde) nur dann das Dateisystem, wenn das Valid-Flag nicht gesetzt war. So kann beim Systemstart einfach für jede Partition das e2fsck-Programm aufgerufen werden. Es wird nur dann aktiv, wenn es nötig ist.

Der Superblock enthält weiterhin eine Angabe über die maximale Anzahl von Mount-Vorgängen, die zwischen zwei System-Checks ablaufen dürfen. Jedesmal, wenn ein Dateisystem gemountet wird, wird ein Zähler (*mount-count*) um eins erhöht. Wird bei einem Systemstart festgestellt, daß der Wert des Zählers den Wert der maximalen Mountvorgänge überschreitet, so wird ein Systemcheck erzwungen.

Im Superblock steht noch eine Angabe, die festlegt, wieviel Prozent eines Dateisystems für den Superuser root reserviert wird. Standardmäßig sind es 5 Prozent.

Um diese Werte zu manipulieren steht das Programm **tune2fs** zur Verfügung. Dieses Programm sollte aber nur für Dateisysteme verwendet werden, die gerade **nicht** read-write gemountet sind!

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

Journaling Dateisysteme

Ein großer Nachteil der traditionellen Unix-Dateisysteme ist die interne Verwaltung von Dateisysteminformationen. Der Superblock eines Dateisystems wird beim Mounten des Dateisystems in den Arbeitsspeicher geladen und alle Veränderungen des Superblocks werden dort vorgenommen. Die eigentlichen Veränderungen am Dateisystem (Anlegen, Verändern oder Löschen von Dateien) werden aber tatsächlich minütlich auf der Platte vorgenommen. Erst beim Abhängen des Dateisystems (in der Regel beim Herunterfahren) wird der im Speicher liegende Superblock physikalisch auf die Platte geschrieben.

Dieser Mechanismus spart zwar Zeit, weil Zugriffe auf den Arbeitsspeicher wesentlich schneller erledigt werden können, als Zugriffe auf die Platte, aber er birgt große Risiken. Wenn ein solches Dateisystem nicht korrekt abgehängt wird, etwa bei einem Systemabsturz oder Stromausfall, dann kann der aktuelle Superblock nicht mehr auf die Platte geschrieben werden. Der Zustand des Dateisystems ist dann inkonsistent, das bedeutet, daß der Superblock nicht mehr die tatsächliche aktuelle Situation des Dateisystems beschreibt, sondern die, die das Dateisystem beim vorletzten Mounten aufwies.

Linux reagiert auf diese Inkonsistenz mit einem erzwungenen Dateisystemcheck beim Booten, wenn das System merkt, daß eine Platte nicht korrekt abgehängt wurde. Dieser Systemchek kann - je nach Partitionsgröße - sehr lange dauern und im schlimmsten Fall einen manuellen Eingriff erfordert. Das bedeutet, daß ein abgestürzter Server lange nicht mehr zur Verfügung steht und eventuell sogar nicht mehr automatisch wieder hochfährt.

Um dieses Problem zu lösen, wurden moderne Dateisysteme mit einem neuen Mechanismus ausgestattet, dem Journaling. Dabei wird jede Transaktion zwischen dem System und der Platte in einem Journal mitprotokolliert, so daß nach einem Absturz in sehr kurzer Zeit wieder ein konsistenter Zustand hergestellt werden kann. Das bedeutet nicht, daß alle Daten wiederhergestellt werden können, die durch den Absturz womöglich verlorengingen, sondern nur, daß es zu keiner Diskrepanz zwischen Superblock und Dateisystem kommen kann.

Dazu wird beim Systemstart - falls es zu einem Absturz gekommen war - das Journal gelesen und die darin enthaltenen Transaktionen werden quasi Stück für Stück rückgängig gemacht, bis es wieder zu einem konsistenten Zustand kommt. Dieser Vorgang kann wesentlich schneller ausgeführt werden, als ein Dateisystemcheck eines herkömmlichen Dateisystems, weil nicht die ganze Platte überprüft werden muß. Die Dauer ist auch nicht abhängig von der Größe der Partition.

Linux bietet heute mehrere Dateisystemtypen an, die das Journaling beherrschen. Bemerkenswert sind dabei die folgenden:

Ext3fs

Ext3fs ist nicht wie der Name vermuten ließe ein echter Nachfolger von Ext2fs, dem Standard-Dateisystem unter Linux. Ext3fs ist vielmehr eine Erweiterung von Ext2fs, da sich am Dateisystem so gut wie nichts entscheidendes zum "Vorgänger" verändert hat. Die Erweiterung beschränkt sich - etwas übertrieben gesagt - im wesentlichen auf das Hinzufügen einer Datei - dem Logfile. Das Ziel bei der Entwicklung von Ext3fs war es mit minimalen Änderungen eine komplette Lösung zu finden, die das Journaling unterstützt. Hierfür wurde dann eine Kopie des Ext2fs-Quellcodes gemacht und eine textuelle Ersetzung durchgeführt mit der "ext2" durch "ext3" ersetzt wurde. Hinzugefügt wurde dann eine vom Dateisystem selbst völlig unabhängige Schicht die das Journaling übernimmt. Das Dateisystem selber hat nichts mit dem Journaling zu tun. Die einzige Änderung die das System hinter sich hat ist, dass es jede Aktion die Schreibend auf die Festplatte zugreift in eine Transaktion packt. D.h. es wird der Journaling-Schicht mitgeteilt, welche Blöcke innerhalb einer Transaktion geändert werden sollen. Den Rest übernimmt dann die neue Journaling-Schicht.

ReiserFS

Beim Dateisystem des Hans Reiser werden im Journal alle Metadatenänderungen protokolliert, die mehr als einen Block betreffen. Bei einem Systemabsturz ist also lediglich sichergestellt, dass das System sehr schnell wieder in einen konsistenten zustand gesetzt wird. Datenverluste kann es weiterhin geben, da ja "nur" die Metadaten im Journal gesichert werden.

Die Datenstruktur wird hier in einem B*-Baum gespeichert. In den Blättern werden vier verschiedene Arten von Datensätzen gespeichert:

- direkte Datensätze: Alle kleinen Dateien (< 1 Block) werden direkt im Baum gespeichert. (Eine 10 Zeichen große Datei landet bei Ext2fs in einem Block von z.B. 1024 Bytes) Hierdurch wird Plattenplatz und ein zusätzlicher Plattenzugriff gespart
- indirekte Datensätze: Hier werden Zeiger auf größere Dateien, die dann außerhalb des Baumes liegen, gespeichert
- Verzeichnisse: Enthält die einzelnen Einträge eines Verzeichnisses
- stat data: Was Ext2fs in Inodes speichert wird hier direkt im Baum gespeichert.

Jeder Datensatz besitzt einen eindeutigen Schlüssel (Hashfunktion), der zum Sortieren und Wiederfinden benutzt wird. Durch die Verwaltung des Baumes (ausbalancieren beim Hinzufügen bzw. Löschen von Einträgen) nimmt das Speichern etwas mehr Rechenzeit in Anspruch als der einfache Mechanismus beim ext2.

XFS

XFS ist eine Portierung des Dateisystems von IRIX, das bereits seit langem auf den bekannten Grafik-Workstations und Servern von Silicon-Graphics läuft. Die Version 1.0 erschien am 1.Mai 2001. Da es in diesem Bereich besonders auf Sicherheit und Geschwindigkeit ankommt ist XFS besonders für große Dateien ausgelegt. Wenn man bedenkt, dass Dateien bis zu 9.000 PByte groß sein können (zur Information die Reihenfolge: Kilo-, Mega-, Giga-, Terra-, Peta-, Exa-Byte).

Die technischen Details füllen ganze Seiten. Zu erwähnen wäre z.B. dass mit dem plattformübergreifenden xfsdump/xfsrestore sogar Dumps von IRIX nach Linux transferiert werden können.

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.104.1

Erzeugen von Partitionen und Dateisystemen

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Plattenpartitionen zu konfigurieren und Dateisysteme auf Medien wie Festplatten zu erzeugen. Dieses Lernziel beinhaltet verschiedene **mkfs** Kommandos zur Erzeugung von verschiedenen Dateisystemen auf Partitionen, einschließlich *ext2*, *ext3*, *reiserfs*, *vfat* und *xfs*.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **fdisk**
- **mkfs**

Wie in jedem Betriebssystem, so müssen Festplatten auch unter Linux vorbereitet werden, um vom System benutzt werden zu können. Die beiden Schritte, die nötig sind, sind

- Partitionieren einer Festplatte
- Dateisystem auf den Partitionen erstellen (unter DOS wird das "formatieren" genannt)

Für jede dieser beiden Aufgaben gibt es ein (bzw. mehrere) Programm(e) um sie zu bewältigen.

Partitionieren einer Festplatte mit fdisk

Eine neue Festplatte ist komplett leer. Damit ein beliebiges Betriebssystem auf die Platte zugreifen kann, müssen zunächst einmal bestimmte Bereiche auf der Platte angelegt werden, die das System nutzen kann - die sogenannten Partitionen. Selbst wenn eine Platte nicht unterteilt werden soll, also als ein Bereich genutzt werden soll, muß eine Partition erstellt werden.

Festplatten bestehen aus einer bis mehreren runden übereinanderliegenden Magnetscheiben, die in konzentrische Kreise (Spuren) eingeteilt werden. Alle übereinanderliegenden Spuren werden Zylinder genannt und gemeinsam angesprochen. Das liegt an der Tatsache, daß für jede Scheibenoberfläche ein Schreib/Lesekopf existiert. Alle diese Schreib/Leseköpfe sind starr miteinander verbunden. Wenn einer davon auf der Spur 5 seiner Plattenoberfläche ist, so sind alle anderen auch auf der Spur 5 jeweils ihrer Oberfläche. Daher werden Schreib- und

Lesevorgänge immer auf allen untereinanderliegenden Spuren gleichzeitig durchgeführt. So entsteht der Begriff Zylinder, eben alle übereinanderliegenden Spuren.

Partitionen sind jetzt ringförmige Bereiche auf einer Platte. Eine Partition wird durch einen Start- und einen Endzylinder begrenzt. Der erste Zylinder (Zylinder 0) ist nie Teil einer Partition, sondern enthält Verwaltungsinformationen, wie den Master-Boot-Sektor und die Tabelle mit den Partitionsinformationen.

Wenn eine Platte z.B. 1024 Zylinder besitzt und in zwei Partitionen aufgeteilt werden soll, so entstünde etwa eine Aufteilung wie die folgende:

Partition	Startzylinder	Endzylinder
1	1	512
2	513	1023

Jede physikalische Festplatte kann bis zu vier sogenannte primäre Partitionen (primary partitions) aufnehmen. Das liegt daran, daß auf dem Zylinder 0 nur Platz gelassen wurde für vier Einträge in der Partitionstabelle (partition-table).

Eine dieser vier primären Partitionen darf eine sogenannte erweiterte Partition (extended partition) sein, die ihrerseits wieder behandelt wird, wie eine physikalische Festplatte und selbst wieder Unterpartitionen (logical partitions) aufnehmen kann. Linux kann bis zu 11 solcher logischer Partitionen pro erweiterter Partition verwalten.

Linux hat eine eindeutige Namenskonvention für Festplatten und Partitionen. Dabei wird unterschieden zwischen IDE- und SCSI-Festplatten. Die Namen für die IDE-Festplatten sind:

`/dev/hda`
: Der Master des ersten IDE-Kanals

`/dev/hdb`
: Der Slave des ersten IDE-Kanals

`/dev/hdc`
: Der Master des zweiten IDE-Kanals

`/dev/hdd`
: Der Slave des zweiten IDE-Kanals

`/dev/hde`
: Der Master des dritten IDE-Kanals

`/dev/hdf`

Der Slave des dritten IDE-Kanals

...

SCSI-Festplatten werden der Reihe nach mit Buchstaben von a bis z durchnummeriert, zuerst alle Platten des ersten SCSI-Hostadapters, dann die des zweiten usw. Für jeden Hostadapter gilt, daß die Platten der Reihe nach durchnummeriert werden, die Platte mit der niedrigsten SCSI-ID zuerst, bis zur Platte mit der höchsten SCSI-ID. Der Name der SCSI-Platten beginnt immer mit **/dev/sd**. Also wäre die erste SCSI-Platte **/dev/sda**, die nächsten **/dev/sdb**, **/dev/sdc** usw.

Die Partitionen jeder Platte sind jetzt einfach als Nummern an diese Namen angehängt. Die primären Partitionen tragen die Nummern 1 bis 4, wobei eine dieser primären Partitionen - unabhängig von ihrer Nummer - eine erweiterte Partition sein kann. Die logischen Partitionen (Unterpartitionen) innerhalb der erweiterten Partition tragen die Nummern 5 bis 15. Daraus lassen sich also aus den Namen der Partitionen eindeutige Informationen gewinnen. Ein paar Beispiele:

/dev/hda1

Die erste primäre Partition des Masters des ersten IDE-Kanals

/dev/hdb5

Die erste logische Partition des Slaves des ersten IDE-Kanals

Wenn eine solche Bezeichnung also eine Nummer hat, so ist die jeweilige Partition angesprochen, wenn sie keine Nummer hat, so ist die ganze physikalische Platte gemeint.

Um jetzt eine bestimmte Platte zu partitionieren, müssen wir also zunächst einmal wissen, wie diese Platte heißt. Das Programm zum Partitionieren von Festplatten heißt fdisk und wird zusammen mit dem Namen der zu partitionierenden Platte aufgerufen. Wollen wir also auf der ersten SCSI-Festplatte im System (`/dev/sda`) Partitionen einrichten, so schreiben wir:

```
fdisk /dev/sda
```

Es erscheint zunächst mal die wenig aussagekräftige Zeile

```
Command (m for help):
```

Also drücken wir doch mal das m:

```
Command action
   a   toggle a bootable flag
```

```
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   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)

Command (m for help):
```

Für uns sind nur ein paar wenige dieser Befehle notwendig, insbesondere die folgenden:

a

Bootflag ein- und ausschalten. Manche Systeme, wie etwa DOS/Windows können nur von Partitionen booten, die den sogenannten Bootflag gesetzt haben oder - im DOS-Jargon - aktiviert sind. Linux benötigt diesen Flag überhaupt nicht.

d

Delete - Löschen einer Partition. Hier muß die jeweilige Partitionsnummer angegeben werden. Mit dem `p`-Befehl kann die Partitionstabelle angezeigt werden, die uns die gewünschte Nummer zeigt. Erweiterte Partitionen, die noch logische Partitionen enthalten können nicht gelöscht werden.

`l`

Liste aller Partitionstypen. Jede Partition hat eine hexadezimale Kenn-Nummer, die vom jeweiligen System gesetzt wird und dieser Partition einen bestimmten Typ zuweist. Linux benutzt dabei drei Kenn-Nummern, 82 für eine Swap-Partition (eine Erweiterung des Arbeitsspeichers), 83 für eine normale Linux-Plattenpartition und 8e für eine Partition des Logical Volume Managers.

n

Neue Partition anlegen. Wenn die Platte bisher keine erweiterte Partition besitzt, dann wird zunächst gefragt, ob eine primäre oder eine erweiterte Partition erstellt werden soll. Gibt es schon eine erweiterte Partition, so wird statt dessen nach primärer und logischer

Partition gefragt. Existieren schon vier primäre Partitionen und eine davon ist eine erweiterte, dann fällt diese Frage weg, denn es können dann ja nur logische Partitionen angelegt werden.

Die nächste Frage ist die nach der Nummer der anzulegenden Partition und dann müssen Start- und Endzylinder angegeben werden. Statt einem End-Zylinder kann auch eine Größenangabe gemacht werden, die mit einem Pluszeichen (+) eingeleitet werden muß und einen Größenpostfix M oder K für Mega- oder Kilobyte besitzen darf. So bedeutet `+256M` eben eine Partitionsgröße von 256 Megabyte.

`p`

Print Partition-Table - Gibt die Partitionstabelle der Platte aus.

`t`

Wechselt den Typ der Partition. Hier kann dann einer der Typen angegeben werden, die mit dem l-Befehl angezeigt wurden.

`q`

Quit - Beendet das Programm ohne die vorgenommenen Änderungen tatsächlich vorzunehmen. Es bleibt also alles beim Alten.

`w`

Write Partition-Table - Schreibt die vorgenommenen Änderungen in die Partitionstabelle und beendet das Programm.

Mit diesen Befehlen sind alle denkbaren Aktionen durchzuführen, die wir für den Umgang mit Linux normalerweise brauchen. Wenn Sie allerdings nur die Partitionstabelle einer physikalischen Festplatte anzeigen wollen, so können Sie fdisk auch einfach mit dem Parameter `-l` aufrufen, dann zeigt es nur die Partitionstabelle des anfgegebenen Laufwerks (oder aller Laufwerke, wenn keines angegeben wurde) und beendet sich dann sofort wieder.

Erstellen eines Dateisystems mit mkfs und seinen Verwandten

Bevor eine Partition (oder eine Diskette) von Linux benutzt werden kann, muß zunächst ein Dateisystem erstellt werden. Das ist die gleiche Prozedur, die unter DOS/Windows "Formatieren" genannt wird. Das Prinzip eines Unix/Linux Dateisystems unterscheidet sich aber grundlegend von dem des DOS/Windows Dateisystems. So finden Sie hier zunächst einmal ein kurzen Überblick über den Aufbau von Unix-Inode-Systemen, dessen Verständnis im Folgenden vorausgesetzt wird, wenn wir im Einzelnen das Erstellen des Dateisystems besprechen. Dieses Verständnis wird insbesondere auch im nächsten Kapitel wichtig sein.

Der häufigste Dateisystemtyp unter Linux ist das *Second Extended Filesystem* oder kurz EXT2. Es unterscheidet sich in einzelnen Details vom Standard-Unix-Dateisystem, benutzt aber die selben Techniken, was I-Nodes angeht. Für das Verständnis der LPIC-Level1 Prüfung reicht das Wissen um diesen Standard. Für die speziell Interessierten findet sich hier noch Informationen über das EXT2 Filesystem.

Das Programm, mit dem wir Dateisysteme erstellen heißt `mkfs` (Make FileSystem) und ist eigentlich nur ein Frontend für weitere Programme, die dann für jedes Linux-Dateisystem extra zur Verfügung stehen, wie etwa `mkfs.ext2, mkfs.minix, mkfs.msdos`

oder `mkfs.xiafs`.

Der grundsätzliche Aufruf von mkfs ist

```
mkfs  [ -t Dateisystemtyp ] [ Optionen ] Gerätedatei [ Blocks ]
```

Dabei sind die Optionen abhängig vom verwendeten Dateisystemtyp. Wird der Dateisystemtyp weggelassen wird heutzutage EXT2 angenommen, aber darauf gibt es keine Garantie. Werden die Blocks weggelassen, so werden alle zur Verfügung stehenden Blocks verwendet.

Das Programm zur Erstellung von EXT2-Dateisystemen kennt die folgenden wichtigen Parameter:

-b *Blockgröße*

Spezifiziert die Größe der Blocks in Byte. Gültige Größen sind 1024, 2048 und 4096.

-c

Checkt das Gerät nach schlechten Blocks ab, bevor das Dateisystem erstellt wird.

-i *Bytes-Per_Inode*

Stellt das Verhältnis von Plattenplatz zur Menge der Inodes ein. Je größer der angegebene Wert ist, um so weniger Inodes werden erstellt. Dieser Wert sollte grundsätzlich kleiner als die Blockgröße des Dateisystems sein.

-N *Anzahl der Inodes*

Gibt die absolute Anzahl der zu erstellenden Inodes an. Wird anstatt des -i Parameters verwendet bzw. überschreibt dessen Einstellungen.

In der Regel ist es ausreichend, ein Dateisystem mit den Standard-Parametern anzulegen, also einfach gar keine Parameter anzugeben. ein einfaches

```
mkfs /dev/hdb5
```

legt ein solches Dateisystem auf der Partition 5 der zweiten IDE-Platte an. In Ausnahmefällen ist es aber notwendig, bestimmte Parameter zu verändern. So werden beispielsweise beim Anlegen eines EXT2-Dateisystems auf einer Diskette nur 184 Inodes angelegt. Normalerweise würde das sicher auch reichen, um die 1,44 MByte Diskettenplatz aufzuteilen. Wenn wir aber etwa eine Bootdiskette erstellen wollen, die schon alleine hunderte von Gerätedateien im `/dev`-Verzeichnis braucht, ist diese Einstellung definitiv nicht brauchbar. Hier müssten wir also mit dem -N Parameter manuell die Anzahl der zu erstellenden Inodes hochsetzen.

Explizites Anlegen eines EXT2- oder EXT3-Dateisystems

Um sicherzugehen, daß tatsächlich ein Dateisystem des Typs EXT2 (second extended) oder seine Erweiterung zum Journaling Dateisystem (ext3) erzeugt wird, muß entweder der Dateisystemtyp mit der Option `-t ext2` angegeben werden, oder es wird gleich das entsprechende Programm mke2fs aufgerufen.

Mit diesem Programm können beide Dateisystemtypen erzeugt werden. Um ein Journaling-Dateisystem vom Typ ext3 zu erzeugen muß der Parameter `-j` (journaling) mit angegeben werden.

Anstatt **mke2fs** kann auch **mkfs.ext2** oder **mkfs.ext3** angegeben werden.

Explizites Anlegen eines Reiser-Dateisystems

Wenn ein Dateisystem des Typs REISERFS angelegt werden soll, so muß dem **mkfs**-Programm der Parameter `-t reiserfs` angegeben werden, oder es wird das Programm mkreiserfs direkt aufgerufen.

Anstatt **mkreiserfs** kann auch **mkfs.reiserfs** angegeben werden.

Eine explizite Angabe der anzulegenden Inodes entfällt bei dieser Dateisystemart, weil das Reiserfs die Inodes dynamisch anlegt, je nach Bedarf.

Explizites Anlegen eines XFS-Dateisystems

Wenn ein Dateisystem des Typs XFS angelegt werden soll, so muß dem **mkfs**-Programm der Parameter `-t xfs` angegeben werden, oder es wird das Programm **mkfs.xfs** direkt aufgerufen.

Explizites Anlegen eines VFAT-Dateisystems

Wenn ein Dateisystem des Typs VFAT (Windows Dateisystem) angelegt werden soll, so muß dem **mkfs**-Programm der Parameter `-t vfat` angegeben werden, oder es wird das Programm **mkdosfs** direkt aufgerufen.

Anstatt **mkdosfs** kann auch **mkfs.vfat** angegeben werden.

1.104.2

Erhaltung der Dateisystemintegrität

Beschreibung: Prüfungskandidaten sollten in der Lage sein, die Integrität von Dateisystemen zu prüfen, freien Speicherplatz und Inodes zu überwachen und einfache Dateisystemprobleme zu beheben. Dieses Lernziel beinhaltet die Kommandos, die für die Verwaltung eines Standard-Dateisystems notwendig sind sowie die zusätzlichen Notwendigkeiten eines *Journaling* Dateisystems.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **du**
- **df**
- **fsck**
- **e2fsck**
- **mke2fs**
- **debugfs**
- **dumpe2fs**
- **tune2fs**

Dieser Bereich hat eine verhältnismäßig hohe Bewertung und es kommen in der LPI 101 Prüfung wirklich einige Fragen zu diesem Thema an die Reihe. Zunächst einmal ist es notwendig, die Architektur der Unix-Dateisysteme zu verstehen, das wurde bereits auf den letzten Seiten erklärt. Ein großer Schwerpunkt liegt im Verständnis des I-Node-Systems. So ist z.B. eine häufige Fehlerursache das Fehlen von freien I-Nodes. Obwohl auf einer Platte noch haufenweise Megabytes frei wären kann keine Datei mehr angelegt werden. Daher sind Techniken notwendig, die das Erkennen solcher Probleme ermöglichen.

Das Programm fsck und seine Verwandten

Wie schon beim Anlegen von Dateisystemen, so ist auch beim Reparieren bzw. Prüfen der Systeme für jedes Dateisystem ein spezielles Programm vorhanden, das genau das jeweilige System kennt. Wie beim Anlegen gibt es aber eben auch wieder ein sogenanntes Frontend, das dann die jeweiligen Programme aufruft. Dieses Frontend heißt **fsck** (FileSystemChecK). Dieses Frontend ruft dann die einzelnen Filesystem-Checker für die jeweiligen Dateisysteme auf, als da wären:

- e2fsck bzw. fsck.ext2 für das Second Extended Filesystem
- reiserfsck für das ReiserFS
- fsck.minix für Minix Dateisysteme
- fsck.msdos für DOS-FAT-Systeme
- fsck.vfat für Windows-VFAT-Systeme
- fsck.xfs für XFS-Dateisysteme

Grundsätzlich ist jedes dieser Programme dafür gedacht, die Konsistenz eines Dateisystems zu überprüfen und gegebenenfalls zu reparieren.

Die Anwendung des fsck-Programms sollte immer nur auf nicht gemounteten Dateisystemen stattfinden, da sonst die Gefahr droht, daß ein Schreibzugriff eines anderen Prozesses (wie etwa die ständige Synchronisation) Veränderungen vornimmt, die den Check bzw. die Reparatur durcheinanderbringen und so den Schaden nur vergrößern. Das ist allerdings ein Problem beim Überprüfen des Wurzel-Dateisystems, da es nicht so einfach möglich ist, es zu checken, ohne es zu mounten. In diesem Fall sollte grundsätzlich in den Single-User-Mode gewechselt werden und das Wurzeldateisystem sollte Read-Only gemountet sein!

Beim Systemstart wird das Programm fsck mit der Option -A aufgerufen, was das Programm veranlasst, alle Dateisysteme zu überprüfen, die in /etc/fstab aufgelistet sind. Die Reihenfolge ist dabei grundsätzlich durch die Angabe des sechsten Feldes innerhalb der /etc/fstab Datei geklärt. Das Wurzeldateisystem wird zuerst überprüft und dann werden entsprechend den Nummern in diesem sechsten Feld (fs_passno) die anderen Dateisysteme der Reihe nach abgearbeitet. Wenn mehrere solcher Systeme die gleiche Nummer haben, dann wird versucht, sie gleichzeitig zu bearbeiten.

Soll ein Dateisystem manuell (nicht beim Systemstart) überprüft werden, so gibt es ein paar Dinge zu bedenken. Neben der Tatsache, die oben schon erwähnt wurde, daß das System nicht gemountet sein sollte, gibt es ein paar zu bemerkende Optionsschalter, die bekannt sein müssen.

Die grundsätzliche Aufrufform ist

```
fsck Optionen  Dateisystem
```

Um das Dateisystem jetzt manuell zu überprüfen, sollten zumindestens die wichtigsten Optionen bekannt sein, sonst kann es schlimmstenfalls dazu kommen, daß es gar nicht überprüft wird. Die folgenden Optionen beziehen sich hauptsächlich auf das EXT2 Dateisystem, das im Augenblick sicherlich der Standard unter Linux ist.

`-f`

force - die Überprüfung wird erzwungen, auch wenn das Dateisystem ein gesetztes Valid-Flag hat. Das ist im Handbetrieb fast immer der Fall, daher ist das -f ein sehr wichtiger Parameter.

`-p`

preen - Automatische Reparatur ohne jede Nachfrage.

`-n`

no - Das Dateisystem wird ReadOnly geöffnet und alle Fragen, ob eine bestimmte Aktion durchgeführt werden soll, werden automatisch mit n(ein) beantwortet. Es werden also keine Veränderungen durchgeführt, aber man kann sehen, was passieren würde...

`-y`

yes - Das genaue Gegenteil von -n. Alle gestellten Fragen werden mit y(es) beantwortet.

Die Angabe des Dateisystems erfolgt in Form der entsprechenden Gerätedatei. Ein manueller Aufruf könnte also z.B. so aussehen:

```
e2fsck -f /dev/hda7
```

Damit würde das Dateisystem auf der siebten Partition (der dritten logischen Partition innerhalb der erweiterten Partition) des Masters des ersten IDE-Kanals zwingend (-f) überprüft.

Die Überprüfung und Reparatur von Journaling-Dateisystemen läuft grundsätzlich anders ab, als die von herkömmlichen Dateisystemen. Hier wird - im Falle einer Inkonsistenz - der Transaction-Log - eben das Journal - zurückverfolgt und alle Transaktionen rückgängig gemacht, bis das System wieder konsistent ist. Dieser Vorgang ist wesentlich schneller, als der bei einem traditionellen Dateisystem, da nicht die ganze Platte überprüft werden muß.

Das Programm df

Das Programm df (Disk Free) dient dazu, die Belegung einzelner Dateisysteme (oder aller gemounteten Dateisysteme) zu ermitteln. Die Anwendung ist sehr einfach, wird df ohne Parameter angewandt, so zeigt es alle gemounteten Dateisysteme etwa in der folgenden Form:

```
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda2              2071328   1051656    914448  53% /
/dev/hda5              3099108   1737096   1204580  59% /usr
```

```
/dev/hda6              2071296    767708   1198364  39% /opt
/dev/hda7              2071296    215212   1750860  11% /home
```

Aus dieser Ausgabe ist also zu entnehmen, welche Dateisysteme gerade gemountet sind, wieviel 1 Kilobyte-Blocks insgesammt zur Verfügung stehen (1k-blocks), wieviel davon belegt sind (Used), wieviel also noch frei sind (Available), die prozentuale Auslastung - also wieviel Prozent sind belegt (Use%) und schließlich der Mountpoint, an dem das Dateisystem eingehängt ist.

Wird stattdessen der Befehl df mit einem bestimmten Dateisystem aufgerufen, entweder durch die Nennung des Mountpoints oder durch die Angabe der entsprechenden Gerätedatei, so werden nur die Angaben über dieses Dateisystem ausgegeben. Hätten wir also entweder

```
df /usr
```

oder

```
df /dev/hda5
```

eingegeben, so wäre es zur folgenden Ausgabe gekommen:

```
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda5              3099108   1737096   1204580  59% /usr
```

Ein wichtiger Parameter für df ist noch die Angabe -i oder --inodes. Wird **df** mit dieser Option aufgerufen, so werden statt den Angaben über die Kilobyte-Blöcke jetzt Angaben über die I-Nodes gemacht. die Ausgabe sähe jetzt also folgendermaßen aus:

```
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hda2             263296   16769  246527    6% /
/dev/hda5             393600  100095  293505   25% /usr
/dev/hda6             263296   32595  230701   12% /opt
/dev/hda7             263296   11088  252208    4% /home
```

Jetzt sehen wir also die Anzahl aller Inodes, die Anzahl der benutzten Inodes (IUsed), die Anzahl der freien Inodes (IFree) und wieder die prozentuale Auslastung (IUse%).

Diese Angaben sind auserordentlich wichtig, weil - wie oben schon erwähnt - es dazu kommen kann, daß zwar noch reichlich Platz in Kilobyte auf einer Partition sein kann, jedoch keine Inodes mehr frei sind, weil sehr viele, sehr kleine Dateien darauf gespeichert sind.

Auch hier gibt es einen wichtigen Hinweis für moderne Journaling-Dateisysteme. Diese Systeme speichern ihre Daten in einer völlig anderen Struktur ab, in sogenannten B-Bäumen. Das hat zur Folge, daß solche Systeme keinen reservierten Platz für die Inodes aufweisen, also Inodes dynamisch anlegen können, wenn sie benötigt werden. Aus diesem Grund zeigt **df -i** bei solchen Systemen keine realen Werte

für die Inodes an.

Das Programm du

Das Programm du (Disk Usage) zeigt den Platzbedarf einzelner Dateien bzw. Verzeichnisse an. Das ist z.B. wichtig, wenn es darum geht, herauszufinden, welches Verzeichnis besonders viel Platz benötigt auf einer Partition, von der der df-Befehl gezeigt hatte, daß der Platz langsam knapp wird.

Das Programm ist in der Regel nur sinnvoll mit Kommandozeilenparametern anwendbar, weil es sonst alle Dateien und Verzeichnisse zeigt und die Ausgabe so etwas unübersichtlich wird. Der wichtigste Optionsschalter ist **`-s`**, der dafür sorgt, daß nur die Summe der verwendeten Bytes aller übergebenen Verzeichnisse ausgibt. So ist schnell feststellbar, wieviel Platz ein Verzeichnis mit allen darin enthaltenen Dateien und Unterverzeichnissen benötigt. Der Aufruf

```
du -s /opt
```

führt dann nur noch zu einer Ausgabe, die etwa folgendermaßen aussehen könnte:

```
767708 /opt
```

Die Angaben erfolgen normalerweise in Kilobyte, die obige Ausgabe besagt also, daß 760 Megabyte im /opt-Verzeichnis belegt sind. Wenn wir jetzt wissen wollen, wie sich diese Summe zusammensetzt, dann können wir einfach alle Verzeichnisse mitangeben, indem wir schreiben:

```
du -s /opt/*
```

und bekommen jetzt eine Auflistung wie folgt:

```
20      /opt/Office51
20      /opt/fsuite
88556   /opt/gnome
157568  /opt/kde
184220  /opt/kde2
16      /opt/lost+found
31372   /opt/netscape
20924   /opt/netscape6
4       /opt/nps
258728  /opt/office52
```

```
4       /opt/oracle
8       /opt/skyrix
9476    /opt/slab
11764   /opt/tfd
8       /opt/tngfw
4       /opt/virtuoso-lite
5012    /opt/www
```

Jetzt lässt sich also schon ziemlich genau sagen, wer hier den vielen Platz braucht...

Werkzeuge für den Umgang mit Ext2-Dateisystemen

In dieser Stufe der Linux-Zertifizierung werden für die hier genannten Werkzeuge noch keine umfassenden Kenntnisse verlangt. Wichtig ist, zu wissen, daß sie existieren und welche Aktionen damit vorgenommen werden können. An dieser Stelle also nur eine kurze Beschreibung der entsprechenden Tools mit jeweils einem Hinweis auf die Handbuchseiten. Ich habe die entsprechenden Handbuchseiten übersetzt, ein Studium dieser Information kann sicherlich nicht schaden.

- mke2fs
  Das Werkzeug zum Anlegen des Dateisystems wurde in Abschnitt 1.104.1 - Erzeugen von Partitionen und Dateisystemen bereits ausführlich besprochen.
- debugfs
  Ein mächtiges Werkzeug, um ein EXT2-Dateisystem zu bearbeiten. Mit diesem Programm ist es möglich, manuell alle möglichen Einstellungen des Dateisystems zu verändern, aber auch es auf einen Schlag unbrauchbar zu machen.
- dumpe2fs
  Gibt den Superblock und Block-Gruppeninformationen eines EXT2-Dateisystems auf die Standard-Ausgabe aus. So kann diese wichtige Information zwischengespeichert werden um damit ein Dateisystem manuell wiederherzustellen.
- tune2fs
  Ermöglicht wichtige Einstellungen von Dateisystem-Parametern des EXT2-Dateisystems.

1.104.3

Kontrolle des Ein- und Aushängen von Dateisystemen

Beschreibung: Prüfungskandidaten sollten in der Lage sein, das Einhängen (Mounten) eines Dateisystems zu konfigurieren. Dieses Lernziel beinhaltet die Fähigkeit, Dateisysteme manuell ein- und auszuhängen, die Konfiguration des Mountens von Dateisystemen bei Systemstart und das Konfigurieren von wechselbaren Datenträgern, die von Benutzern gemountet werden können, wie z.B. Bänder, Disketten und CDs.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/fstab`
- **mount**
- **umount**

Unter Linux/Unix werden die verschiedenen Laufwerke nicht wie etwa unter Windows als einzelne Dateisysteme mit eigenem Laufwerksbuchstaben verwaltet, sondern sie werden zu einem einzigen Baum zusammengebaut. Der englische Begriff *to mount* (anbringen, montieren) steht für das Einhängen der verschiedenen Dateisysteme in den Dateibaum. Dieses Einhängen geschieht entweder automatisch beim Systemstart, oder einzelne Dateisysteme werden manuell während des Betriebs *gemountet*. Dieses manuelle Mounten bezieht sich in der Regel auf Wechsemedien, wie etwa Disketten, CD-ROMs oder auch Netzverbindungen (im Windows-Jargon: *Netzlaufwerke*)

In diesem Zusammenhang sind für uns zwei Programme und zwei Dateien wichtig. Die Programme sind diejenigen, die wir zum Einhängen (*mount*) und Abhängen (*umount*) benötigen, die Dateien zeigen, was wohin eingehängt werden soll (/etc/fstab) und was gerade eingehängt ist (/etc/mtab). Diese vier Elemente sollen hier dargestellt werden.

Das Programm mount

Mit dem Programm mount werden Dateisysteme an bestimmte Plätze, also in bestimmte Verzeichnisse eingehängt, neudeutsch gemountet. Die grundsätzliche Form des Programms ist simpel:

```
mount [-t Dateisystemtyp] [ -o Optionen] [Gerätedatei] [Mountpoint]
```

Die meisten Dateisystemtypen werden heute von mount selbstständig erkannt, die Angabe des Dateisystemtyps kann daher meist weggelassen werden. Die Optionen sind abhängig vom verwendeten Dateisystemtyp, sie werden später noch etwas genauer dargestellt. Wichtig ist also zunächst einmal die Angabe, welches Gerät (Gerätedatei in /dev) soll wohin eingehängt werden.

Um z.B. eine Diskette (/dev/fd0) ins Verzeichnis /floppy einzuhängen würde der Befehl

```
mount /dev/fd0 /floppy
```

genügen. Nachdem dieser Befehl ausgeführt wurde ist der Inhalt der Diskette im Verzeichnis /floppy zu finden. Eventuelle Inhalte, die vorher in diesem Verzeichnis waren sind jetzt unsichtbar, sobald das Diskettenlaufwerk aber wieder abgehängt ist, sind sie wieder vorhanden.

Sobald ein Dateisystem mit dem mount-Befehl eingehängt wurde (und dabei nicht die Option -n gesetzt wurde) wird ein Eintrag in die Datei /etc/mtab geschrieben, der das Dateisystem und den Mountpoint beschreibt.

Existiert bereits ein Eintrag in /etc/fstab, der das zu mountende Dateisystem beschreibt, so genügt dem mount-Befehl die Angabe entweder der Gerätedatei oder des Mountpoints, um das Dateisystem einzuhängen.

Soll statt eines lokalen Dateisystems ein NFS-Verzeichnis gemountet werden, so muß statt der Gerätedatei der Rechnername des NFS-Servers, gefolgt von einem Doppelpunkt (:) und dem freigegebenen Verzeichnispfad angegeben werden. So würde die Zeile

```
mount einstein.my.domain:/usr/public /mnt
```

das Verzeichnis `/usr/public` des Rechners `einstein.my.domain` in das lokale Verzeichnis `/mnt` einhängen.

Der mount-Befehl kann auch einfach mit dem Parameter -a aufgerufen werden, ohne Nennung eines Dateisystems. Dann werden alle Dateisysteme eingehängt, die in der Datei /etc/fstab stehen und dort nicht die Option noauto gesetzt haben. Das passiert gewöhnlich beim Hochfahren des Systems.

Die Optionen zum Einhängen der Dateisysteme

Das Programm mount kann mit dem -o Kommandozeilenparameter verschiedene Optionen setzen, wie ein Dateisystem gemountet werden soll. Die selben Optionen können auch in der Datei /etc/fstab angegeben werden (siehe unten).

Folgende Optionen werden von allen Dateisystemen verstanden:

`async`
: Alle Ein-/Ausgabe Operationen des Dateisystems werden asynchron vorgenommen.

`atime`
: Die Zugriffszeit der Inodes (atime) werden bei jedem Zugriff gesetzt. (Voreinstellung)

`auto`
: Wenn diese Option in /etc/fstab steht, wird das Dateisystem gemountet, wenn mount mit der Option -a aufgerufen wird.

`defaults`
: Entspricht den voreingestellten Optionen: rw, suid, dev, exec, auto, nouser und async

`dev`
: Gerätedateien auf dem Dateisystem sind gültig und werden interpretiert.

`exec`
: Erlaubt die Ausführung von Binärdateien auf dem Dateisystem.

`noatime`
: Die Inode-Zugriffszeiten (atime) werden nicht bei jedem Zugriff gesetzt.

`noauto`
: Wenn ein Eintrag in /etc/fstab diese Option gesetzt hat, so wird dieses Dateisystem nicht durch den Befehl mount -a eingehängt. Also wird es auch nicht beim Systemstart automatisch gemountet.

`nodev`
: Gerätedateien auf diesem Dateisystem werden nicht interpretiert.

`noexec`
: Verbietet die Ausführung von Binärdateien auf diesem Dateisystem.

`nosuid`
: Die SUID und SGID Bits von Programmen auf diesem Dateisystem werden ignoriert.

`nouser`

Ein Normaluser (nicht root) darf dieses Dateisystem nicht mounten.

`remount`

Ein bereits gemountetes Dateisystem soll neu gemountet werden. Das wird meist benutzt, um die Optionen eines bereits gemounteten Dateisystems neu zu setzen, insbesondere um ein ReadOnly Dateisystem wieder beschreibbar zu mounten.

`ro`

Read Only - Das Dateisystem wird ReadOnly (nur lesbar) gemountet.

`rw`

ReadWrite - Das Dateisystem wird ReadWrite (les- und schreibbar) gemountet.

`suid`

SUID und SGID Bits werden interpretiert.

`sync`

Alle Ein- und Ausgabeoperationen werden synchron durchgeführt.

`user`

Erlaubt einem Normaluser, das Dateisystem zu mounten.

Neben diesen Optionen, die für alle verwendeten Dateisystemtypen gelten, existieren noch viele verschiedene Optionen für jeden einzelnen Dateisystemtyp. Diese Optionen hier darzustellen würde einerseits den Rahmen dieser Darstellung sprengen und andererseits sind sie auch nicht notwendiger Bestandteil der LPIC 101 Prüfung. Wer sie nachlesen will, kann das auf der Handbuchseite des mount-Befehls tun.

Das Programm umount

Das Programm umount (**nicht** unmount !!) hängt ein oder mehrere eingehängte Dateisysteme wieder ab. Das abzuhängende Dateisystem kann entweder durch die Nennung der entsprechenden Gerätedatei oder durch die Angabe des Mountpoints spezifiziert werden.

Ein Dateisystem kann nicht abgehängt werden, wenn es in Benutzung (busy) ist. Das ist in der Regel dann der Fall, wenn ein Prozess ein Verzeichnis dieses Dateisystems als aktuelles Arbeitsverzeichnis hat.

Wird umount mit dem Parameter -a aufgerufen, so werden alle Dateisysteme abgehängt, die in der Datei /etc/mtab aufgelistet sind. Das passiert gewöhnlich während des Shutdowns.

Die Datei /etc/mtab

Die Datei /etc/mtab enthält immer eine Liste aller gerade gemounteten Dateisysteme. Diese Datei wird niemals von Hand editiert, sie ist ausschließlich von den Programmen mount und umount zu beschreiben. Es existieren zwar für beide Befehle jeweils die Option -n, die verhindert, daß dieser Eintrag gemacht wird, das dient aber nur dazu, daß auch auf Systemen, deren /etc-Verzeichnis ReadOnly gemountet ist, beide Befehle funktionieren.

Die Datei /etc/fstab

Die Datei /etc/fstab ist sozusagen die Bauanleitung des Systems, in der genau steht, welches Dateisystem wohin gemountet werden soll. Es obliegt dem Systemadministrator, diese Datei zu erstellen und zu pflegen. Jedes Dateisystem wird durch eine separate Zeile in der fstab repräsentiert; innerhalb einer Zeile werden die Felder durch Tabs oder Leerzeichen getrennt. Die Reihenfolge der Zeilen in der fstab ist wichtig, da fsck(8), mount(8), und umount(8) diese Datei sequentiell abarbeiten.

Das erste Feld, (fs_spec), beschreibt das zu mountende blockorientierte Device oder remote filesystem.

Das zweite Feld, (fs_file), gibt den Mountpunkt für das Dateisystem an. Bei Swap-Partitionen sollte hier `none" stehen.

Das dritte Feld, (fs_vfstype), beschreibt den Typ des Dateisystems. Hier steht entweder (bei festen Partitionen) das jeweilige Kürzel für das Dateisystem, das auf der Partition angelegt ist (ext2, minix, vfat, swap, ...) oder (bei Wechselplattenlaufwerken wie Zip- oder Diskettenlaufwerken) auto, damit der mountbefehl den jeweiligen Typ selbst erkennt. Bei NFS-Verbindungen steht hier entsprechend der Begriff nfs.

Das vierte Feld, (fs_mntops), beschreibt die zum Dateisystem gehörenden Mountoptionen. Hier werden die Optionen, die oben näher beschrieben wurden, als eine durch Kommas getrennte Liste angegeben. Falls keine speziellen Optionen gewünscht sind, wird hier der Begriff defaults eingegeben.

Das fünfte Feld, (fs_freq), wird von dump(8) benutzt um zu entscheiden welche Dateisysteme gedumpt werden müssen. Eine 1 bedeutet, daß das Dateisystem mit dump bearbeitet werden soll, eine 0 bedeutet, daß es nicht gedumpt werden muß. Ist das fünfte Feld nicht vorhanden, wird für diesen Wert Null angenommen und dump geht davon aus, daß das Dateisystem nicht gedumpt werden muß.

Das sechste Feld, (fs_passno), wird von fsck(8) benutzt um die Reihenfolge, in der die Dateisysteme während des Reboots geprüft werden, festzulegen. Das root Dateisystem sollte mit einer fs_passno von 1 versehen sein, andere Dateisysteme mit einer fs_passno von 2. Dateisysteme innerhalb eines Laufwerks werden sequentiell geprüft, Dateisysteme auf verschiedenen Laufwerken jedoch gleichzeitig, um parallel arbeitende Hardware zu unterstützen. Ist das sechste Feld nicht vorhanden oder Null, wird sinnigerweise eine Null zurückgegeben und fsck geht davon aus, daß das Dateisystem keiner Prüfung bedarf.

Normalerweise darf nur root Dateisysteme an- und abhängen. Das macht bei Festplattenpartitionen durchaus Sinn, ist aber für

PC-Hardware mit Wechselmedien wie Disketten, CD-Laufwerke oder ZIP-Disks sehr unpraktisch. Wenn also gewünscht wird, daß ein Normaluser bestimmte Laufwerke auch mounten darf, so sollte bei den Optionen nach dem "defaults" noch ein "user" stehen. Damit dieses Laufwerk dann aber beim Start nicht automatisch gemountet wird (und so eine Fehlermeldung provoziert, wenn z.B. keine Diskette eingelegt ist) sollte auch noch die Option "noauto" benutzt werden.

Ein typischer Eintrag in einem sehr einfachen Linux-System könnte also folgendermaßen aussehen:

```
/dev/hda2       /       ext2    defaults 1 1
/dev/hda3       swap    swap    defaults 0 2
/dev/hda5       /usr    ext2    defaults 1 2
/dev/hdb        /cdrom  auto    ro,noauto,user 0 0
/dev/fd0        /floppy auto    defaults,noauto,user 0 0
```

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.104.4

Verwalten von Diskquotas

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Diskquotas für Benutzer zu verwalten. Dieses Lernziel beinhaltet das Einrichten von Diskquotas für ein Dateisystem, das Bearbeiten, Prüfen und Erstellen von Berichten über Userquotas.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **quota**
- **edquota**
- **repquota**
- **quotaon**

Das Erstellen von Diskquotas hatte in der ersten Version der LPI101 Prüfung nur eine Bewertung von 1, es ist also fragwürdig, ob überhaupt nur eine Frage zu diesem Thema in der Prüfung vorkommt. Trotzdem ist es ein offizielles Thema und wird daher hier behandelt. Abgesehen von der niedrigen Bewertung ist es aber doch so, daß es ein sehr praktischer Mechanismus ist,m den man in der Systemverwaltung häufig brauchen kann.

Worum geht es? Quotas sind Mechanismen, die es erlauben, bestimmten Usern oder Gruppen einen eingeschränkten Platz auf einem bestimmten Dateisystem zu gewähren. Das heißt, es ist z.B. möglich, einem User fest vorzugeben, wieviel Platz er in seinem Homeverzeichnis nutzen darf. Damit kann verhindert werden, daß ein User übermäßig viel Platz in Anspruch nimmt und so den anderen Usern Platz wegnimmt. In der Praxis findet man diesen Mechanismus häufig bei Webservern, die bestimmten Usern eine eingeschränkte Menge Platz anbieten, um Webseiten darauf abzulegen.

Das Prinzip der Quotas läuft darauf hinaus, daß der Kernel bei jedem schreibenden Zugriff eines Users auf ein Dateisystem überprüft, ob der User noch Platz hat, oder ob er seine Quote schon erreicht hat. Um das zu gewährleisten, muß schon beim Mounten des Dateisystems festgelegt werden, daß dieses Dateisystem eine Quotierung des Platzes haben soll. Das wird in der Regel in der Datei /etc/fstab festgelegt, wo bei den Optionen eines Dateisystems die Begriffe **`usrquota`** bzw. **`grpquota`** angefügt werden. Userquotas sind Einschränkungen für einzelne User, Gruppenquotas entsprechend Einschränkungen für bestimmte Usergruppen.

Voraussetzungen

Die erste Voraussetzung zur Verwendung von Disk-Quotas ist die Verwendung eines Kernels, der quotas unterstützt. Das sollte heute standardmäßig jeder Kernel anbieten.

Als nächstes muß in /etc/fstab angegeben werden, welche Dateisysteme Disk-Quota benutzen sollen. Jedes Dateisystem, das dieses Feature anbieten soll muß hier bei den Mount-Optionen den Begriff `usrquota` für Userquotas und/oder `grpquota` für Gruppenquotas enthalten. Eine /etc/fstab-Datei könnte also dann folgendermaßen aussehen:

```
/dev/hda2       /       ext2    defaults,usrquota 1 1
/dev/hda3       swap    swap    defaults 0 2
/dev/hda5       /usr    ext2    defaults 1 2
/dev/hda6       /opt    ext2    defaults 1 2
/dev/hda7       /home   ext2    defaults,usrquota 1 2
```

Wir haben also sowohl für das Wurzeldateisystem (hier /dev/hda2), als auch für das Dateisystem, auf dem die Homeverzeichnisse liegen (/home auf /dev/hda7) Userquotas angegeben.

Anlegen der Quotafiles mit quotacheck

Die Angabe, welche User auf welchem Dateisystem wieviel Platz bzw. wieviel Inodes benutzen dürfen, steht auf der Wurzel des jeweiligen Dateisystems in den Dateien `quota.user` bzw. `quota.group`. Diese Dateien sind Binärdateien, die zunächst einmal angelegt sein müssen. Dazu dient das Programm **quotacheck**. Um dieses Programm zu benutzen sind root-Privilegien nötig.

Das Programm quotacheck kann entweder für jedes Dateisystem einzeln aufgerufen werden, indem ihm die entsprechende Gerätedatei als Parameter mit angegeben wird, oder es wird mit dem Parameter -a aufgerufen und arbeitet so alle Dateisysteme ab, die in der Datei /etc/fstab eine Quotaangabe gesetzt haben.

Im einfachsten Fall schreiben wir also (als root) die Zeile

```
quotacheck -avug
```

was bedeutet, daß alle Dateisysteme bearbeitet werden, die Quotas unterstützen (-a), daß dort sowohl Userquotas (-u), als auch Gruppenquotas (-g) berücksichtigt werden und daß das Programm uns auch mitteilt, was es gerade tut (-v).

Dieser Befehl sollte immer dann angewandt werden, wenn ein neues Dateisystem mit quotas erstellt wurde oder wenn Dateisysteme nicht sauber heruntergefahren wurden, also typischerweise nach einem Systemabsturz, wenn auch fsck ausgeführt wird. Die meisten

Distributionen bieten bereits fertige Startdateien an, die diese Aufgabe übernehmen.

Nach der Abarbeitung dieses Befehls existieren auf allen Dateisystemen, die die Mountoption `usrquota` gesetzt hatten die Datei `quota.user` und auf allen Dateisystemen, die die `grpquota`-Option aktiviert hatten die Datei `quota.group`. Diese Dateien enthalten binär codiert alle wichtigen Angaben über die festgelegten User- bzw. Gruppenquotas, insbesondere auch die Angaben, welche User wieviel Platz bzw. wieviele Dateien auf diesem Dateisystem im Augenblick in Anspruch nehmen.

Quotas definieren mit edquota

Nachdem die Quotadateien jetzt angelegt sind, müssen wir als nächstes die Beschränkungen definieren, die für den jeweiligen User bzw. die Gruppe gewünscht sind. Auch das darf natürlich nur root vornehmen. Für diese Aufgabe existiert das Programm **edquota**.

Der Aufruf von edquota erfolgt entweder zusammen mit einem Usernamen, oder mit der Option -g und einem Gruppennamen. Dadurch wird ein Editor aufgerufen (standardmäßig der vi) und alle Angaben über die Quotas in lesbarer Form in eine Temporärdatei geschrieben. Diese Angaben können jetzt editiert werden. Wird der Editor jetzt mit Sichern der Datei beendet, so schreibt das Programm edquota die entsprechenden Einstellungen wieder binär codiert in die jeweiligen Quotadateien der jeweiligen Dateisysteme.

Typischerweise werden Userquotas für jeden User einzeln definiert. Ein Aufruf von

```
  edquota hans
```

würde jetzt etwa folgende Ausgabe im Editor nach sich ziehen:

```
Quotas for user hans:
/dev/hda2: blocks in use: 121, limits (soft = 0, hard = 0)
        inodes in use: 12, limits (soft = 0, hard = 0)
/dev/hda7: blocks in use: 1220, limits (soft = 0, hard = 0)
        inodes in use: 204, limits (soft = 0, hard = 0)
```

Um jetzt Einschränkungen vorzunehmen, können die jeweiligen Limits verändert werden. Dazu stehen uns jeweils ein Soft-Limit und ein Hard-Limit sowohl für die Kilobyte-Blocks (Speicherplatz), als auch für die Inodes (Anzahl der Dateien) zur Verfügung. Der Hard-Limit kann nicht überschritten werden, der Soft-Limit kann überschritten werden, es erfolgt aber eine Warnung. Ein User kann einen Soft-Limit nur für eine bestimmte Zeit überschreiten (standardmäßig eine Woche), dann wird der überschrittene Soft-Limit zum Hard-Limit.

Aus der Beispielausgabe oben können wir entnehmen, daß der User Hans auf der Wurzel (/dev/hda2) 121 Kilobyte in 12 Dateien und im Homeverzeichnis (/dev/hda7) 1,2 Megabyte in 204 Dateien belegt.

Nehmen wir an, wir wollten eine Einschränkung vornehmen, die besagt, daß Hans in seinem Homeverzeichnis 200 Megabyte nutzen darf,

auf dem Wurzelverzeichnis aber nur 10 Megabyte. Dann verändern wir die Angaben folgendermaßen:

```
Quotas for user hans:
/dev/hda2: blocks in use: 121, limits (soft = 10000, hard = 11000)
        inodes in use: 12, limits (soft = 0, hard = 0)
/dev/hda7: blocks in use: 1220, limits (soft = 200000, hard = 210000)
        inodes in use: 204, limits (soft = 0, hard = 0)
```

Wenn wir auch noch die Anzahl der Dateien einschränken wollten, so können wir das mit Hilfe der Inode-Limits machen. Um z.B. festzulegen, daß Hans nicht mehr als 100 Dateien auf dem Wurzelsystem anlegen darf schreiben wir:

```
Quotas for user hans:
/dev/hda2: blocks in use: 121, limits (soft = 10000, hard = 11000)
        inodes in use: 12, limits (soft = 100, hard = 100)
/dev/hda7: blocks in use: 1220, limits (soft = 200000, hard = 210000)
        inodes in use: 204, limits (soft = 0, hard = 0)
```

Jetzt beenden wir den Editor mit sichern und alle nötigen Veränderungen sind vorgenommen.

Damit wir in einem System mit vielen hundert Usern diesen Vorgang nicht hunderte Male wiederholen müssen, gibt es die Möglichkeit, mit edquota die Einstellungen eines Users auf einen anderen User zu kopieren. Nehmen wir an, wir wollen die Einstellungen, die wir für Hans getroffen haben, jetzt auch für Peter, Michael und Gabi treffen, dann schreiben wir einfach:

```
  edquota -p hans -u peter michael gabi
```

Das -u könnten wir sogar noch weglassen. Jetzt haben alle drei User (peter, michael und gabi) die selben Einschränkungen wie hans.

Wird edquota mit der Option -g *Gruppenname* aufgerufen und es existieren auf dem System Gruppenquotas, so sind entsprechend die Gruppenquotas zu editieren.

Wird edquota mit der Option -t aufgerufen, so erscheint eine Angabe, die die Gnadenfrist (grace period) einstellbar macht, die zwischen Soft- und Hard-Limit liegt. Gültige Werte für die Zeit sind hier sec, min, hour(s), day(s), week(s) oder month(s).

Berichte erstellen mit repquota

Damit der Systemverwalter (root) einen Überblick behalten kann, welche Quotas für welche Dateisysteme vergeben wurden bzw. wieviel davon tatsächlich benutzt wird, gibt es das Programm **repquota**. Dieses Programm zeigt für ein angegebenes Dateisystem (oder - falls die -a Option gesetzt wurde - für alle Dateisysteme mit Quotas) eine Zusammenfassung der Quotas und der aktuellen Belegung.

Wird die Option -g benutzt, so werden die Gruppenquotas angezeigt.

Eigene Quotas ansehen mit quota

Alle bisherigen Befehle sind dem Systemverwalter vorbehalten. Damit ein Normaluser überprüfen kann, welche Einschränkungen er hat bzw. wieweit er seinen Platz schon nutzt, gibt es das Programm **quota**.

Eine typische Ausgabe dieses Programms würde folgendermaßen aussehen, wenn in unserm Beispiel hans den Befehl quota ausführen würde:

```
Disk quotas for user hans (uid 501):
   Filesystem  blocks   quota   limit   grace   files   quota   limit  grace
    /dev/hda2     121   10000   11000              12     100     100
    /dev/hda7    1200  200000  210000             204       0       0
```

Das Feld `grace` (engl. Gnade) zeigt, wieviel Zeitraum zwischen Softlimit und Hardlimit gelassen wird. Steht hier keine Angabe, so ist die Standardeinstellung (1 Woche) gültig.

Wird quota mit der Option -g aufgerufen, so wird statt der Userquota die Gruppenquota des aufrufenden Users dargestellt.

Wenn ein oder mehrere Quotas überschritten wurden, dann gibt quota einen Rückgabewert ungleich Null zurück.

Quotas aktivieren bzw. deaktivieren mit quotaon und quotaoff

Damit all das oben angeführte überhaupt funktioniert, muß die Unterstützung für quotas beim Systemstart angeschaltet werden. Dazu steht das Programm **quotaon** zur Verfügung. Wie schon beim quotacheck kann entweder ein spezielles Dateisystem angegeben werden, oder mit der Option -a veranlasst werden, daß auf allen Dateisysteme, die in der Datei /etc/fstab die Mountoption `usrquota` oder `grpquota` gesetzt haben, die Quotas aktiviert werden. Üblicherweise wird in einer Startdatei beim Systemstart die Zeile

```
quotaon -avug
```

stehen, die die Quotas auf allen (-a) Dateisystemen aktiviert, zeigt, was passiert (-v) und sowohl Userquotas (-u), als auch Gruppenquotas (-g) berücksichtigt.

Um von Hand die Quotas auf einzelnen Dateisystemen zu aktivieren, kann der Befehl quotaon aber auch entsprechend mit dem Dateisystem angegeben werden:

```
quotaon -u /dev/hda7
```

würde also die Userquotas auf dem Dateisystem der Partition /dev/hda7 aktivieren.

Entsprechend funktioniert der Befehl **quotaoff** um die Quotas abzuschalten.

1.104.5

Zugriffskontrolle auf Dateien mittels Zugriffsrechten

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Dateizugriff mittels Zugriffsberechtigungen zu steuern. Dieses Lernziel beinhaltet Zugriffsrechte auf reguläre und spezielle Dateien sowie Verzeichnisse. Ebenfalls enthalten sind Zugriffsmodi wie suid, sgid und *sticky bit*, die Verwendung des Gruppenfeldes für die Vergabe von Zugriffsrechten an Arbeitsgruppen, das *immutable flag* und der voreingestellte Dateierstellungsmodus.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **chmod**
- **umask**
- **chattr**

Die Zugriffsrechte unter Linux/Unix sind im Prinzip sehr einfach aufgebaut, jedoch aber auch sehr wirkungsvoll, wenn man weiß, wie sie im Einzelnen angewandt werden. Das Wissen um diese Rechte ist absolutes Grundwissen für alle, die mit Systemverwaltung unter Linux beschäftigt sind und sollte daher im Schlaf beherrscht werden.

Grundsätzlicher Aufbau von Zugriffsrechten

Jede Datei in einem Linux-System - also auch Gerätedateien, Sockets, Pipes, Verzeichnisse, usw. - besitzt in ihrer Inode eine Angabe über die Zugriffsrechte. Außerdem hat jede Datei genau einen Eigentümer und genau eine Gruppe, der sie zugehört. Die Zugriffsrechte beziehen sich immer auf eben diesen Eigentümer der Datei, auf Gruppenmitglieder der Gruppe, der die Datei angehört und auf den Rest der Welt.

Für diese drei Kategorien (Eigentümer, Gruppe, Rest) existiert jeweils eine Angabe, die beschreibt, ob die Datei für die jeweilige Kategorie lesbar (r), beschreibbar (w) und ausführbar (x) ist.

Der Befehl `ls -l` *Dateiname* zeigt uns für jede Datei eben diese Angaben. So bedeutet die Ausgabe:

```
-rw-r-----   1 hans     autoren         1519 Jul 29  2000 Testdatei
```

Die Datei **Testdatei** ist eine reguläre Datei (-). Sie gehört dem User **hans**, der sie lesen und verändern darf (**rw-**). Die Datei gehört zur Gruppe **autoren**. Mitglieder dieser Gruppe dürfen die Datei lesen (**r--**). Der Rest der Welt hat keinerlei Rechte auf diese Datei (**---**).

Das erste Zeichen der Ausgabe zeigt uns also, um was für eine Art Datei es sich handelt. Folgende Dateiarten sind unter Linux definiert:

Zeichen	Dateiart
-	Reguläre (normale) Datei
d	Verzeichnis (directory)
l	Symbolischer Link (symlink)
b	Blockorientierte Gerätedatei (block device)
c	Zeichenorientierte Gerätedatei (character device)
p	Feste Programmverbindung (named pipe)
s	Netzwerk Kommunikationsendpunkt (socket)

Das dem ersten Zeichen folgende Konstrukt ist die Beschreibung des Zugriffsmodus. Die ersten drei Zeichen beschreiben die Rechte des Eigentümers der Datei, die nächsten drei die Rechte eines Gruppenmitglieds der Gruppe, der auch die Datei zugehört und die letzten drei die Rechte aller anderen User. Ein `r` bedeutet Leserecht (read), ein `w` Schreibrecht (write) und ein `x` Ausführungsrecht (execute). Diese Rechte können numerisch dargestellt werden. Dazu werden die Rechte wie folgt bezeichnet:

Eigentümer			**Gruppenmitglied**			**Rest der Welt**		
r	**w**	**x**	**r**	**w**	**x**	**r**	**w**	**x**
4	**2**	**1**	**4**	**2**	**1**	**4**	**2**	**1**

Die Nummern werden für jede der drei Kategorien einzeln addiert. Ein Zugriffsrecht von `rw-r-----` wäre so also numerisch darstellbar als 640. Die 6 errechnet sich aus dem r (4) plus w (2) des Eigentümerrechtes, die 4 ist einfach das Leserecht (r) des Gruppenmitglieds und die 0 entspricht keinem gesetzten Recht.

Für reguläre Dateien sind diese Rechte einfach zu durchschauen. Das Leserecht bedeutet, daß der Inhalt der Datei gelesen werden darf. Das Schreibrecht bedeutet, daß der Inhalt der Datei verändert werden darf (und somit die Datei auch gelöscht werden darf) und das Ausführungsrecht bedeutet, daß die Datei ein Programm ist, das ausgeführt werden darf.

Etwas anders sieht es mit Verzeichnissen aus. Hier bedeutet das Leserecht, daß der Inhalt eines Verzeichnisses aufgelistet werden darf, das Schreibrecht, daß Dateien im Verzeichnis angelegt und gelöscht werden dürfen und das Ausführungsrecht, daß in das Verzeichnis gewechselt werden darf. Das hat einen Haken, zu dem wir später noch kommen werden. Hat nämlich ein User Schreibrecht auf ein Verzeichnis, so darf er darin auch Dateien löschen, auf die er selbst keinerlei Rechte besitzt. Das liegt an der Tatsache, daß ein Verzeichnis

genau genommen nur eine Datei ist, die Dateinamen und die dazu passenden Inode-Nummern gespeichert hat. Eine Datei in einem Verzeichnis ist also letztlich nichts anderes als eine Zeile Text in einer Textdatei. Und wer Schreibrechte auf eine Textdatei hat, kann Zeilen daraus löschen!

Spezielle Rechte

Neben diesen sichtbaren Rechten existieren noch drei weitere, die man sich als weitere (führende) Nummer vorstellen kann. Dabei handelt es sich um das *Substitute UserID Bit* (4), das *Substitute GroupID Bit* (2) und das *Sticky Bit* (1).

Substitute UserID Bit (SUID)

Dieses Recht gilt ausschließlich für ausführbare Dateien. Hat eine ausführbare Datei dieses Recht gesetzt, so erscheint in der Darstellung durch das ls -l Kommando statt dem x beim Eigentümerrecht ein s. Jeder User, der dieses Programm ausführt, tut dies unter der effektiven UserID des Users, dem die Datei gehört.

So hat z.B. das Programm /usr/bin/passwd die Aufgabe, daß auch normale User damit ihr eigenes Passwort ändern können. Dieses Passwort wird aber gespeichert in einer Datei, die nur von root beschrieben werden darf. Das Programm /usr/bin/passwd hat als Eigentümer root und hat das Substitute UserID Bit gesetzt. Jeder User, der dieses Programm ausführt tut dies also unter der effektiven UserID von root und hat daher die Rechte von root während der Ausführung. Das ermöglicht es dem Programm, das veränderte Passwort zu speichern.

Das ist beim Passwort-Programm nicht weiter problematisch, eine richtige Sicherheitslücke entsteht, wenn eine Shell mit diesem Bit ausgestattet ist und als Eigentümer root hat. Dann könnte jeder User, der diese Shell ausführt, root-Rechte benutzen!

Substitute GroupID Bit (SGID)

Dieses Recht gilt einerseits für ausführbare Dateien und andererseits für Verzeichnisse. Hat ein ausführbares Programm dieses Recht gesetzt, so gilt der gleiche Mechanismus, wie beim Substitute UserID Bit, nur diesmal eben die Gruppenmitgliedschaft betreffend. Ein User, der dieses Programm ausführt, tut dies als Gruppenmitglied der Gruppe, der das Programm gehört, statt seiner eigentlichen Gruppenkennung. Er hat also die Rechte eines Gruppenmitglieds dieser Gruppe, auch wenn er selbst nicht Mitglied dieser Gruppe ist. Statt dem x beim Gruppenrecht stellt das ls -l Kommando hier ein s dar.

Hat ein Verzeichnis dieses Recht gesetzt, dann liegt der Fall etwas anders. Legt ein User, der Schreibrecht auf ein Verzeichnis hat, in diesem Verzeichnis eine Datei an, so erhält diese Datei normalerweise die Gruppenmitgliedschaft der primären Gruppe des Users, der sie eben angelegt hat. Das führt schnell zum Chaos, wenn z.B. mehrere User zusammen an einem Projekt arbeiten. Alle diese User sind Gruppenmitglieder einer bestimmten Gruppe, aber sie haben diese Gruppe nicht als primäre Gruppe. Das heißt, es kann sein, daß die einzelnen User die Dateien der jeweils anderen Projektmitarbeiter nicht lesen können. Wenn dieses Verzeichnis aber der gemeinsamen Gruppe gehört und eben das Substitute GroupID Bit darauf gesetzt ist, dann werden Dateien in diesem Verzeichnis grundsätzlich unter der GruppenID abgespeichert, der das Verzeichnis gehört. Mit diesem Mechanismus sind Arbeitsverzeichnisse für bestimmte Projektgruppen realisierbar.

Sticky Bit

Das Sticky Bit hat nur eine Bedeutung für Verzeichnisse. Oben wurde schon das Problem erwähnt, daß ein User, der Schreibrecht auf ein Verzeichnis hat, innerhalb dieses Verzeichnisses alle Dateien löschen kann, auch wenn sie ihm nicht gehören und er keinerlei Rechte auf diese Dateien besitzt. Das wird durch das Sticky Bit verhindert.

Ein Verzeichnis in dem jeder User Schreibrecht haben muß, wie etwa das /tmp-Verzeichnis, sollte unbedingt dieses Bit gesetzt haben. Ansonsten kann ein böswilliger User dort die Dateien anderer User löschen.

Das gesetzte Sticky-Bit wird vom ls -l Kommando durch ein t statt des x in der Kategorie "Rechte des Restes der Welt" dargestellt.

Um diese speziellen Rechte auch numerisch darzustellen, wird vor den oben genannten "normalen" Rechten noch eine Oktalziffer angefügt, so daß sich das folgende Bild ergibt:

Sonderrechte			Eigentümer			Gruppenmitglied			Rest der Welt		
SUID	SGID	Sticky	r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1	4	2	1

Ein Recht von 4755 bedeutet also, daß das Substitute UserID Bit gesetzt ist (4), der Eigentümer Lese-, Schreib- und Ausführungsrechte (1+2+4=7) hat während sowohl Gruppenmitglieder, als auch der Rest der Welt nur Lese- und Ausführungsrecht (1+4=5) besitzen.

Ändern von Zugriffsrechten mit chmod

Um die oben beschriebenen Rechte vergeben bzw. ändern zu können existiert das Programm chmod (change mode). Es erlaubt, die Rechte von Dateien und Verzeichnisse zu verändern. Die prinzipielle Anwendung ist einfach:

`chmod [`*`Optionen`*`] `*`Modus Datei(en)`*

Die wichtigste Option ist dabei `-R` oder `--recursive`, mit dem ganze Unterverzeichnisse mit allen Dateien darin auf einmal bearbeitet werden können.

Als Modus kann entweder ein numerischer, oder ein symbolischer Modus angegeben werden. Der numerische Modus entspricht genau dem, was im letzten Abschnitt dieser Seite beschrieben wurde, er wird hier nicht nochmal erklärt. Nur ein kurzes Beispiel noch, der Aufruf

```
chmod 644 foo.txt
```

würde der Datei `foo.txt` ein Zugriffsrecht von `rw-r--r--` geben. Mit numerischem Modus ist grundsätzlich immer ein absoluter Wert gesetzt, egal, welche Rechte vorher gesetzt waren.

Der symbolische Modus besteht aus einer Angabe der Rechte durch Buchstaben. Die Grundsätzliche Form ist:

```
[ugoa] +|-|= [rwxstugo], ...
```

Das führende Zeichen steht für die zu verändernde Kategorie, u steht für User (Eigentümer), g für group (Gruppenmitglied), o für other (Andere) und das a steht für all (Alle). Wird dieses führende Zeichen weggelassen, dann wird standardmäßig a (Alle) angenommen.

Vorsicht, oft wird diese Angabe verwechselt und das o mit dem Begriff Owner (Eigentümer) verwechselt. Das kann fatale Folgen haben, weil man dann die Rechte, die man eigentlich dem Eigentümer geben will, dem Rest der Welt verleiht!

Das folgende Rechenzeichen +, - oder = beschreibt, ob ein Recht den bestehenden Rechten hinzugefügt (+) werden soll, davon abgezogen (-) werden soll oder absolut (=) gesetzt werden soll.

Dem Rechenzeichen folgt die Angabe der zu setzenden (oder addierenden oder subtrahierenden) Rechte in der Form einer Zeichenkette, die aus den Buchstaben r,w,x,s,t,u,g,o besteht. Dabei meinen r, w und x wie üblich Read, Write und Execute. Wird dem User das s-Recht gesetzt, so meint das das Substitute UserID Bit (beispielsweise durch `u+s`), bekommt hingegen die Gruppe dieses Recht, so ist das Subtitute GroupID Bit gemeint (`g+s`). Das t steht für das Sticky-Bit und kann nur dem Rest der Welt vergeben werden (`o+t`).

Folgen dieser Angabe noch ein u, g oder o, so ist dies ein Ausschlußkriterium in Verbindung mit dem a am Anfang. Das nachgestellte u, g oder a schützt also die Rechte der User, Gruppenmitglieder bzw. Anderen vor Veränderung.

Das ganze kann jetzt mehrmals hintereinander angewandt werden, indem mehrere solcher Modi durch Kommas getrennt angegeben werden. So ist es etwa möglich zu schreiben:

```
chmod u=rwx,g=rx,o-rwx foo
```

um dem Programm foo das Recht `rwxr-x---` zu setzen. Gewöhnlich tut man sich aber in diesem Fall leichter mit einer numerischen Angabe (hier 750).

Das praktische an der symbolischen Angabe von Modi ist die Fähigkeit, bestimmte Rechte auf die bestehenden Rechte aufzuaddieren bzw. sie von den bestehenden Rechten abzuziehen. Ein einfaches `+x` bedeutet z.B., daß allen drei Kategorien User, Group und Other ein Ausführungsrecht zu den schon vorhandenen Rechten gegeben wird.

Wird als Option ein -R oder ein --recursive angegeben, so verändert chmod die Rechte eines ganzen Verzeichnisbaums, inclusive aller enthaltenen Dateien und Unterverzeichnisse.

Voreingestellten Zugriffsmodus mit umask bestimmen

Es bleibt jetzt natürlich die Frage, welche Zugriffsberechtigungen beim Anlegen einer Datei verwendet werden. Um das festzulegen, kennt Unix das Kommando umask, dessen Anwendung aber etwas merkwürdig ist.

Der Befehl umask erwartet eine Maske als Parameter, die sich auf den Zugriffsmodus bezieht, den neu zu erstellende Dateien bekommen sollen. Das Wort Maske bedeutet, daß nicht die Werte eingegeben werden, die gesetzt werden sollen, sondern umgekehrt, die Werte, die nicht gesetzt (maskiert) werden sollen. Die einfachste Möglichkeit besteht darin, die gewünschten oktalen Werte jeweils von 7 abzuziehen:

Wollen wir z.B. dafür sorgen, daß alle unsere Dateien die Zugriffsberechtigung `rw-r-----` bekommen, also Lese-und Schreibrecht für den Eigentümer, Leserecht für Gruppenmitglieder und keine Rechte für den Rest der Welt, dann entspräche das der oktalen Darstellung 640.

Die dafür notwendige umask wäre dann:

```
7-6=1
7-4=3
7-0=7
```

Mit dem Befehl

```
umask 137
```

würden also alle Dateien, die wir anlegen die gewünschte Zugriffsberechtigung 640 bekommen. Das hat noch einen kleinen Haken, weil die Verzeichnisse, die wir erstellen würden auch diesen Modus bekämen. Damit wäre für uns selbst das Durchsuchungsrecht (x) nicht gesetzt. Linux hat aus diesem Grund dafür den Mechanismus entwickelt, daß selbst wenn im umask-Kommando das x-Recht gesetzt ist, beim Erzeugen von normalen Dateien dieses Recht nicht gesetzt wird, beim Erzeugen von Verzeichnissen hingegen schon. Ein vernünftiges umask-Kommando setzt also zumindestens für den Eigentümer auch das x-Recht. Damit wäre ein typischer Wert für umask z.B. 022 (`rwxr-xr-x`) oder 027 (`rwxr-x---`).

Neben dieser umständlichen Methode gibt es aber auch die symbolische Form, die die Rechte direkt bezeichnet. Sie wird in der Form `u=...,g=...,o=...` eingegeben, wobei für ... immer die entsprechenden Rechte eingesetzt werden. Also würde der Befehl

```
umask u=rwx,g=rx,o=
```

die gleiche Wirkung haben wie

```
umask 027
```

Typischerweise steht eine umask-Anweisung in einer der Shell-Startdateien wie z.B. `/etc/profiles` oder `~/.profile`. Jeder User kann seine Voreinstellung also selbst einstellen, eine der wenigen Möglichkeiten, mit denen er dem Systemverwalter ins Handwerk pfuschen kann, wenn der versucht, sein System sicher zu machen. Allerdings bezieht sich diese Einstellung ja nur auf die neu anzulegenden Dateien des jeweiligen Users...

Erweiterte Dateiattribute im EXT2-Dateisystem

Dateien auf EXT2-Dateisystemen besitzen neben den oben genannten Attributen noch weitere, die einen zusätzlichen Schutz bieten können. Diese Attribute können mit dem Befehl lsattr angezeigt und mit chattr verändert (gesetzt) werden.

Die wichtigsten dieser Attribute sind:

a (append)

Eine Datei mit a-Attribut kann schreibend nur im Anhängen-Modus geöffnet werden. Das bedeutet, daß ein User, der Schreibrecht auf diese Datei hat, zwar Daten an das Ende der Datei anhängen kann (etwa durch die Verwendung der >>-Umleitung), jedoch keine Veränderungen am bestehenden Inhalt der Datei vornehmen darf. Nur der Superuser kann dieses Attribut setzen oder entfernen.

i (immutable)

Eine Datei mit gesetztem i-Attribut kann nicht modifiziert werden. Sie kann weder gelöscht, noch umbenannt werden, es kann kein Hardlink auf sie angelegt werden und keine Daten können angehängt werden. Nur der Superuser kann dieses Attribut setzen oder entfernen.

s (save-delete)

Wenn eine Datei das s-Attribut gesetzt hat, werden alle Blöcke dieser Datei beim Löschen der Datei mit Nullzeichen überschrieben. Dadurch ist die Datei auch durch Methoden wie dem Dateisystemdebuger nicht wieder herstellbar oder ihr Inhalt ist nicht mehr einsehbar, wenn sie einmal gelöscht wurde.

Eine Liste aller möglichen Attribute entnehmen Sie der Handbuchseite von chattr.

1.104.6

Verwaltung von Dateieigentum

Beschreibung: Prüfungskandidaten sollten in der Lage sein, den Benutzer- und Gruppenbesitz von Dateien zu steuern. Dieses Lernziel beinhaltet das Ändern des Besitzers und der Gruppenzugehörigkeit einer Datei sowie des Standardeigentümers von neuen Dateien.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **chmod**
- **chown**
- **chgrp**

Im letzten Abschnitt haben wir gesehen, wie Zugriffsrechte gesetzt bzw. verändert werden. All diese Zugriffsrechte beziehen sich ja auf Eigentümer bzw. Gruppen von Usern. Von daher benötigen wir noch die Technik, Eigentümer und Gruppenzugehörigkeit einer Datei zu verändern. Dazu gibt es die Programme chown und chgrp. Diese Programme sollen hier vorgestellt werden.

Ändern des Dateieigentümers mit chown

Um den Eigentümer einer Datei zu wechseln bzw. besser ausgedrückt, eine Datei einem anderen User zu übereignen, existiert das Programm chown. Dieses Programm darf nur vom Systemverwalter ausgeführt werden, ein normaler User kann also nicht seine Dateien anderen Usern übereignen.

Das Programm chown hat eine einfache Aufrufform:

```
chown [Optionen] Username Datei(en)
chown [Optionen] [Username][.][Gruppenname] Datei(en)
chown [Optionen] --reference=Referenzdatei Datei(en)
```

Eine oder mehrere Dateien bekommen so als Eigentümer den genannten Usernamen zugewiesen. Der Username kann hier entweder als

Name oder als Nummer (UserID - UID) angegeben werden.

Wenn nur ein Username oder eine UserID angegeben wurde, so wird dieser User zum Eigentümer jeder angegebenen Datei und die Gruppenmitgliedschaft der Dateien wird nicht verändert. Wenn dem Username ein Doppelpunkt oder Punkt und ein Gruppenname (oder eine GruppenID) ohne Leerzeichen folgt, dann wird auch die Gruppenzugehörigkeit der Dateien geändert. Wenn dem Usernamen ein Punkt oder Doppelpunkt folgt, aber kein Gruppenname angegeben wird, so wird der angegebene User zum Eigentümer der angegebenen Dateien und die Dateien bekommen die Gruppenzugehörigkeit zu der Gruppe, die die Login-Gruppe dieses Users ist (die Gruppe, die im Gruppenfeld in /etc/passwd für diesen User angegeben ist). Wenn aber der Username weggelassen wurd, aber ein Punkt oder Doppelpunkt, gefolgt von einem Gruppennamen angegeben wurde, so wird nur die Gruppenzugehörigkeit der Dateien verändert, der Eigentümer bleibt unverändert. In diesem Fall arbeitet chown genau wie chgrp.

Wenn statt einem Usernamen und/oder Gruppennamen die Option `--reference=`*`Referenzdatei`* angegeben wurde, so werden Eigentümer und Gruppenzugehörigkeit der angegebenen Dateien so gesetzt, wie die der Referenzdatei.

Wird als Option ein `-R` oder ein `--recursive` angegeben, so verändert chown die Eigentümer eines ganzen Verzeichnisbaums, inclusive aller enthaltenen Dateien und Unterverzeichnisse.

Ändern der Gruppenzugehörigkeit einer Datei mit chgrp

Die Gruppenzugehörigkeit einer Datei kann auch mit dem Befehl **chgrp** gewechselt werden. Die Aufrufform ist ähnlich der von chown:

```
chgrp [Optionen] Gruppe Datei(en)
```

Auch hier kann die Gruppe wieder entweder als Gruppenname oder als GruppenID angegeben werden. Die Benutzung von chgrp ist nur dem Eigentümer und dem Superuser (root) erlaubt. Der Eigentümer kann eine Datei nur den Gruppen zuordnen, denen er selbst auch angehört.

Wie schon bei chown und chmod kann auch chgrp mit der Option `-R` oder `--recursive` ein ganzer Verzeichnsast rekursiv bearbeitet werden, d.h., daß alle Verzeichnisse und enthaltene Dateien bearbeitet werden.

Sicherstellen der Gruppenzugehörigkeit von Dateien in Verzeichnissen

Wenn ein Verzeichnis für eine bestimmte Gruppe von Usern beschreibbar ist und diese User darin eine gemeinsame Arbeit leisten sollen, so wäre es ja praktisch, wenn alle User innerhalb dieses Verzeichnisses alle Dateien, die sie anlegen, eben der Gruppe zuweisen, der das Verzeichnis angehört. Das ist aber nicht automatisch der Fall.

Jeder User kann beliebig vielen Gruppen angehören. er ist aber immer Mitglied einer sogenannten Login-Gruppe (manchmal auch Initialgruppe genannt). Wenn ein User eine Datei anlegt, so wird diese Datei normalerweise die Gruppenzugehörigkeit der Login-Gruppe

dieses Users zugewiesen bekommen.

Um sicherzustellen, daß alle Dateien in einem bestimmten Verzeichnis beim Anlegen immer die Gruppenzugehörigkeit zu der Gruppe bekommen, der das Verzeichnis gehört, muß das Verzeichnis das SGID-Bit gesetzt bekommen. Das wird mit dem Befehl chmod eingestellt, wie auf der letzten Seite beschrieben.

Wenn ein User allerdings nur vorübergehend die Standard-Gruppe wechseln will, so steht im dazu das Kommando newgrp zur Verfügung. Das würde allerdings das Mitdenken aller User voraussetzen, was immer eine Schwachstelle bedeutet...

Wenn also der Systemverwalter ein Verzeichnis für eine bestimmte Projektarbeit anlegt, tut er gut daran, diesem Verzeichnis das SGID-Bit zu setzen, um die Gruppenmitgliedschaft der Dateien in diesem Verzeichnis von vorneherein festzulegen.

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.104.7

Erzeugen und Ändern von harten und symbolischen Links

Beschreibung: Prüfungskandidaten sollten in der Lage sein, harte und symbolische *Links* auf eine Datei zu verwalten. Dieses Lernziel beinhaltet das Erzeugen und Bestimmen von Links, das Kopieren von Dateien über Links und das Verwenden von verknüpften Dateien zur Unterstützung von Tätigkeiten der Systemadministration.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **ln**

Links sind eine Spezialität von Unix-Dateisystemen, die gerne und häufig verwendet werden. Der Umgang mit ihnen ist eine wichtige und manchmal etwas verkopfte Angelegenheit, die ein Systemverwalter sicher beherrschen muß.

Unix und Linux unterscheiden zwischen Hardlinks (direkte Links) und Symlinks (symbolische Links). Die Eigenschaften dieser beiden Typen von Links sowie die Techniken, um sie anzulegen bzw. mit ihnen umzugehen sind Inhalt dieses Kapitels.

Hardlinks

In einem Unix-Dateisystem ist ein Dateiname nur ein Verzeichniseintrag, der in einem Verzeichnis gespeichert ist. Neben diesem Namen ist immer auch ein Verweis auf die Inode gespeichert, die dann die eigentlichen Eigenschaften (Eigentümer, Gruppenmitgliedschaft, Zugriffsrechte, usw.) der Datei und ihre physikalischen Speicheradressen auf der Platte enthält. Dieses Prinzip wurde auf den Seiten über Inodes und über das EXT2 Dateisystem bereits umfassend dargestellt.

Ein Hardlink ist nichts anderes, als ein neuer Verzeichniseintrag auf eine schon bestehende Inode. Also genau genommen ein zweiter Dateiname für eine Datei. Weil die Zugriffsrechte, Eigentümer usw. ja in der Inode stehen, haben alle Hardlinks einer Datei die selben solchen Attribute.

Wenn ein oder mehrere Hardlinks auf eine Datei zeigen, so ist nicht mehr zu unterscheiden, welcher davon das Original und welche die Links sind. Es handelt sich ja einfach nur um Namenseinträge, die auf die selbe Inode zeigen. Das heißt auch, daß Hardlinks immer noch gültig sind, wenn die Datei, auf die sie zeigen gelöscht wurde. Es wurde ja eben nicht die Datei gelöscht, sondern nur einer ihrer Namen. Solange noch weitere Namen existieren, wird die Datei nicht physikalisch gelöscht.

Die Ausgabe des `ls -l` Kommandos zeigt für jede Datei gleich nach dem Zugriffsmodus die Anzahl der Hardlinks (also der Namenseinträge), die diese Datei besitzt.

```
-rw-r--r--   1 root     root         4326 Apr  8 15:02 datei1.txt
-rw-r--r--   5 root     root         1578 Apr  8 15:02 datei2.txt
```

Auch hier wird nicht zwischen Original und Link unterschieden (geht ja eben auch gar nicht) so daß eine Datei mit nur einem Namen hier eine 1 anzeigt. Die zweite Datei des obigen Beispiels hat also 5 Namen, das könnte bedeuten, daß die Datei erstellt wurde und anschließend vier Hardlinks auf sie erstellt wurden. Insgesamt existieren also 5 Namenseinträge für diese Datei.

Da die Hardlinks sozusagen innerhalb der Mechanismen eines Dateisystems arbeiten indem sie eigentlich nur Verweise auf schon bestehende Inodes sind, arbeiten diese Links nur innerhalb der Grenzen eines Dateisystems also einer Partition. Es ist also nicht möglich, Hardlinks auf einer zweiten Partition zu erstellen, die auf eine Datei zeigen, die auf der ersten Partition liegt!

Eine weitere Einschränkung von Hardlinks ist, daß es nicht möglich ist, Hardlinks auf Verzeichnisse zu legen. Es existiert zwar die Option -d (oder -F oder --directory) des Befehls ln, die die Fähigkeit erlauben soll, auf allen Linux-Dateisystemen ist dieses Feature aber verboten.

Symbolische Links

Die genannten Einschränkungen der Hardlinks (keine Links auf Verzeichnisse/ keine Links über die Dateisystemgrenze hinaus) können mit sogenannten *symbolischen Links* umgangen werden. Symbolische Links arbeiten nicht auf der Dateisystemebene, sondern sind einfach Dateien, die nichts anderes enthalten, als den Pfad zu der Datei (oder dem Verzeichnis), auf die sie zeigen. Damit sie als Links zu erkennen sind, haben sie einen eigenen Dateityp (`l`), der es dem Betriebssystem klar macht, daß es sich hier um einen Link und nicht um eine reguläre Datei handelt.

Das `ls -l` Kommando zeigt einen symbolischen Link also als solchen an:

```
-rw-r--r--   1 root     root       276295 Apr 21 19:46 Datei1
lrwxrwxrwx   1 root     root            6 Apr 21 19:46 Datei2 -> Datei1
```

Sowohl an der Angabe des Dateityps (`l`, als auch am Dateinamen, dem ein Pfeilsymbol und das Ziel des Links folgt, ist ersichtlich, daß es sich hier um einen symbolischen Link handelt. Genauso ist ersichtlich, worauf der Link zeigt. Beim Hardlink konnten wir Original und Link nicht unterscheiden, beim symbolischen Link sind sie eindeutig unterscheidbar.

Weil symbolische Links nicht auf der Ebene der Dateisysteme selbst arbeiten reagieren sie aber auch in anderen Beziehungen völlig anders, als die Hardlinks:

- Wenn die Datei (oder das Verzeichnis), auf die ein symbolischer Link zeigt nicht mehr existiert (z.B. gelöscht wurde), dann zeigt der symbolische Link "ins Leere", der Link existiert zwar weiter, er funktioniert aber nicht mehr.
- Wird ein symbolischer Link mit einer relativen Pfadangabe erstellt und anschließend in ein anderes Verzeichnis kopiert, dann wird er womöglich nicht mehr weiter funktionieren, weil vom neuen Verzeichnis aus dieser Pfad nicht existiert.

Andererseits sind die symbolischen Links nicht eingeschränkt, was die Grenzen eines Dateisystems angeht oder die Verwendung für Verzeichnisse.

Das Programm ln

Um Links anzulegen, existiert das Programm ln. Die Aufrufform ist einfach,

```
ln [-s] Datei [Link]
```

Wird das Programm `ln` ohne den Parameter `-s` oder `--symbolic` aufgerufen, so wird ein Hardlink erstellt, mit einem dieser beiden Optionsschaltern wird ein symbolischer Link erstellt.

Wenn ein Hardlink erstellt wird, so muß die Datei, auf die der Link verweist existieren, wenn ein symbolischer Link erstellt wird, so gilt das nicht.

Wird beim Aufruf von ln der Linkname weggelassen, so wird ein Link mit gleichem Namen wie die Datei im aktuellen Verzeichnis erzeugt. Das setzt aber natürlich voraus, daß die Datei nicht im aktuellen Verzeichnis liegt.

Werden mehr als zwei Dateinamen angegeben (Datei und Link), so muß der letzte Parameter ein Verzeichnisname sein. In diesem Verzeichnis werden dann Links auf all die Dateien angelegt, die vor diesem letzten Parameter angegeben wurden.

Normalerweise überschreibt das Programm ln keine Dateien, wenn ein Link angelegt werden soll, dessen Namen schon existiert. ln bietet aber eine Fülle von Optionen, die diese Eigenschaft ändert, inclusive der Frage der automatischen Umbenennung von überschriebenen Dateien.

Identifikation von Hardlinks

Wie oben schon erwähnt, gibt es keine Möglichkeit, zwischen Hardlink und Original zu unterscheiden, weil es ja kein Original gibt, sondern nur mehrere Namen, die alle auf die gleiche Inode verweisen. Wenn wir nun herausfinden wollen, welche Dateinamen alle die selbe Inode verwenden, gibt es da durchaus eine Möglichkeit:

Zunächst einmal müssen wir herausbekommen, welche Inode von unserer Datei überhaupt belegt wird. Das `ls`-Kommando bietet uns mit der Option `-i` die Möglichkeit, das zu erfahren. Ein `ls -i` gibt uns zu den Dateinamen die verwendeten Inode-Nummern mit aus.

Dann können wir mit dem Programm `find` nach allen Dateien suchen, die diese Inode-Nummer benutzen. Allerdings sollten wir das ausschließlich innerhalb der Partition tun, auf der die Datei liegt, die wir untersuchen. Denn zufälligerweise kann ja eine ganz andere Datei auf einem ganz anderen Dateisystem (also auf einer anderen Partition) die selbe Inode-Nummer benutzen.

Spielen wir es einmal durch: Wir befinden uns im Verzeichnis `/usr/local/data` und das `ls -l` Kommando zeigt uns eine Datei mit Namen `BEISPIEL.DAT` folgendermaßen an:

```
-rw-r--r--   5 root     root         1895 Apr  8 15:02 BEISPIEL.DAT
```

Aus der Angabe direkt nach dem Zugriffsmodus entnehmen wir, daß es insgesamt fünf Dateinamen gibt, die auf ein und dieselbe Inode verweisen. Also neben dieser Datei noch vier weitere. Zunächst müssen wir wissen, welche Inode diese Datei überhaupt benutzt. Dazu geben wir den Befehl

```
ls -i BEISPIEL.DAT
```

ein und bekommen die folgende Ausgabe:

```
92550 BEISPIEL.DAT
```

Die benutzte Inode ist also die Inode Nummer 92550. Jetzt müssen wir herausbekommen, auf welcher Partition wir uns eigentlich befinden und wo sie im Verzeichnisbaum eingehängt ist. Wir geben den Befehl `df` ohne weitere Parameter ein. Die Ausgabe lautet:

```
/dev/hda2              2071328   1047620    918484  53% /
/dev/hda5              3099108   1737192   1204484  59% /usr
/dev/hda6              2071296    767708   1198364  39% /opt
/dev/hda7              2071296    215212   1750860  11% /home
```

Nachdem wir uns im Verzeichnis `/usr/local/data` befinden, können wir also aus dieser Ausgabe schließen, daß wir uns auf der Partition `/dev/hda5` befinden und daß diese Partition ins Verzeichnis `/usr` gemountet ist. Jetzt rufen wir den `find`-Befehl auf und weisen ihn an, ab dem Verzeichnis `/usr` alle Dateien zu suchen, die die Inode 92550 benutzen. Zusätzlich weisen wir ihn noch darauf hin, daß er nur dieses eine Dateisystem durchsuchen soll. Dazu kennt find die Option `-xdev`. Wir geben folgenden Befehl ein:

```
find /usr -xdev -inum 92550 -print
```

Der Befehl `find` sucht ab dem Verzeichnis `/usr` aber nur innerhalb der Partition (`-xdev`) alle Dateien, die die Inode 92550 (`-inum`

92550) besitzen. Die gefundenen Dateien werden ausgegeben (-print). Die Anweisung -print hätten wir unter Linux auch weglassen dürfen, es ist hier die voreingestellte Aktion. Das Ergebnis sieht dann etwa wie folgt aus:

```
/usr/lib/Beispiel/Datei.txt
/usr/local/data/BEISPIEL.DAT
/usr/local/lib/Beispiel/Noch_eine
/usr/openwin/Hier_auch
/usr/src/abc
```

Hätten wir dem find-Befehl statt der Aktion -print die Aktion -ls mitgegeben, die alle gefundenen Dateien im selben Format wie ls -dils ausgibt, dann hätten wir die folgende Ausgabe bekommen:

```
 92550    5 -rw-r--r--   1 root     root         1895 Apr  8 15:02
/usr/lib/Beispiel/Datei.txt
 92550    5 -rw-r--r--   1 root     root         1895 Apr  8 15:02
/usr/local/data/BEISPIEL.DAT
 92550    5 -rw-r--r--   1 root     root         1895 Apr  8 15:02
/usr/local/lib/Beispiel/Noch_eine
 92550    5 -rw-r--r--   1 root     root         1895 Apr  8 15:02
/usr/openwin/Hier_auch
 92550    5 -rw-r--r--   1 root     root         1895 Apr  8 15:02 /usr/src/abc
```

Eine weitere Möglichkeit der Identifikation von Hardlinks wäre es, ein ls -ilR Kommando (langes Listing mit Inode-Nummern, rekursiv) auszuführen und dessen Ausgabe an grep weiterzuleiten. Grep würde dann nach der entsprechenden Inode-Nummer am Zeilenanfang suchen. Also für unser opiges Beispiel etwas in der Art:

```
ls -ilR /usr | grep "^ *92550"
```

Das hat allerdings den Nachteil, daß wir nicht festlegen können, daß nur die Dateien innerhalb einer Partition gesucht werden. Zum anderen ist diese Art der Suche wesentlich langsamer als die mit dem find-Befehl.

Kopieren symbolischer Links

Wenn ein symbolischer Link mit dem Befehl cp kopiert wird, ohne daß dazu bestimmte Optionen gegeben werden, so wird nicht etwa der Link kopiert, sondern die Datei, auf die der Link zeigt. Das Ergebnis (die Zieldatei) ist jetzt also kein Link, sondern eine reguläre Datei. Man spricht in diesem Zusammenhang von der Dereferenzierung eines Links.

Dereferenzierung bedeutet, daß der Link zurückverfolgt wird, und durch das ausgetaucht wird, auf das er zeigt.

Um das zu vermeiden, muß dem `cp`-Befehl eine spezielle Option mitgegeben werden, die diese Dereferenzierung unterbindet. Wenn wir also einen symbolischen Link als solchen kopieren wollen, das heißt wenn das Ziel der Kopieraktion wiederum ein Link sein soll, so müssen wir dem Kopierbefehl die Option `-d` oder `--no-dereference` mitgeben.

Aber bitte Vorsicht walten lassen! Wenn ein symbolischer Link als Link kopiert wird, dann wird der Link genauso kopiert, wie er war. Falls er keine absolute, sondern eine relative Pfadangabe zu dem Objekt beinhaltet, auf das er verweist, so wird die Kopie genau die Selbe Pfadangabe haben. Es ist nicht gewährleistet, daß diese Angabe von der neuen Lokalität aus immer noch stimmt.

Am Rande bemerkt: Der `cp`-Befehl ist selbst auch in der Lage, Links zu erzeugen statt Dateien zu kopieren. Mit der Option `-l` erzeugt er Hardlinks statt Kopien und mit `-s` symbolische Links.

Verwenden von Links zur Systemadministration

In der Systemverwaltung werden Links auf vielfältige Weise eingesetzt. Dabei kommen sowohl Hard- als auch symbolische Links zur Anwendung. Ein paar Beispiele:

Vermeidung versehentlichen Löschens mit Hardlinks

Wichtige Systemdateien können an andere Orte gelinkt werden, wo sie sozusagen eine Versicherung gegen versehentliches Löschen bieten. Nehmen wir an, Sie erstellen ein Verzeichnis `/etc2` und verlinken alle wichtigen Dateien in `/etc` mit Hardlinks in dieses Verzeichnis. Sollte jetzt eine Datei in `/etc` gelöscht werden, so steht sie uns immer noch in der aktuellsten Version in `/etc2` zur Verfügung.

Zentrale Verwaltung wichtiger Startdateien in den Userverzeichnissen

Jeder User hat in seinem Home-Verzeichnis verschiedene Startdateien oder andere Konfigurationsdateien. Soll sichergestellt werden, daß alle User immer die gleichen Einstellungen besitzen, so könnten all diese Dateien Hardlinks auf einen Prototyp sein. Wenn der Systemverwalter jetzt für alle User eine Veränderung vornehmen will, so muß er nur eine dieser Dateien (etwa den Prototyp) verändern und alle Dateien der User sind mitverändert.

Symbolische Links auf Verzeichnisse

Wenn bestimmte Teile des Verzeichnisbaums ReadOnly gemountet sein sollen, aber andererseits Unterverzeichnisse dieser Teile beschreibbar sein müssen, so können diese Unterverzeichnisse symbolische Links auf andere Unterverzeichnisse sein, die nicht auf ReadOnly-gemounteten Partitionen liegen. Das `/usr` Verzeichnis wird z.B. gerne ReadOnly gemountet, enthält aber Verzeichnisse wie `tmp` oder `spool`, die beschreibbar sein müssen. Diese Verzeichnisse sind oft symbolische Links auf entsprechende Verzeichnisse unter `/var`. Dort sind diese Verzeichnisse tatsächlich beschreibbar.

1.104.8

Auffinden von Systemdateien und Platzieren von Dateien an den korrekten Ort

---

Beschreibung: Prüfungskandidaten sollten ausreichend vertraut mit dem *Filesystem Hierarchy Standard*, einschließlich typischer Speicherort von Dateien und der Einteilung der Verzeichnisse, sein. Dieses Lernziel beinhaltet das Auffinden von Dateien und Kommandos auf einem Linux-System.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **find**
- **locate**
- **slocate**
- **updatedb**
- **whereis**
- **which**
- `/etc/updatedb.conf`

Die Standard-Hierarchie der Unix-Dateisysteme war und ist schon immer einer der großen Vorteile gegenüber anderen Systemen gewesen. Hat man einmal ein Unix-System egal welcher Art kennengelernt, so findet man sich nahezu auf jedem anderen Unix-System mühelos zurecht. Linux bietet einen solchen Standard genauso an und es gibt weltweite Bestrebungen, die Positionen der Dateien und Verzeichnisse innerhalb des Dateisystems, sowie deren Namen festzulegen. Trotzdem finden sich immer Unterschiede zwischen verschiedenen Unixen und selbst zwischen verschiedenen Linux-Distributionen. Um mit diesen Unterschieden umgehen zu können finden sich verschiedene Werkzeuge, die hier näher besprochen werden sollen.

Die Standard-Hierarchie des Dateisystems

Um zu gewährleisten, daß die verschiedenen Linux-Distributionen eine einheitliche Dateisystem-Hierarchie benutzen, gibt es den Versuch, einen Standard für diese Hierarchie zu erarbeiten, den *Filesystem Hierarchy Standard*, der unter www.pathname.com/fhs/ genau nachzulesen ist.

Die wesentlichen Aufgaben dieses Standards sind die Festlegungen, welche Dateien in welche Verzeichnisse gehören und zwar dergestalt, daß es möglich ist, bestimmte Verzeichnisse mit statischen Daten als ReadOnly-Dateisystem einzuhängen, um eine größtmögliche Stabilität des Systems zu gewährleisten.

Im folgenden sollen die wesentlichen Verzeichnisse kurz beschrieben werden, um ihre Aufgaben bzw. die Philosophie, die hinter ihnen steckt zu klären:

Das Wurzeldateisystem /

Dieses Dateisystem enthält die Wurzel des gesammten Systems. In einem professionellen System wird versucht, dieses Dateisystem möglichst klein zu halten. Es soll die Verzeichnisse enthalten, die notwendig sind, um ein System auch nach einem Ausfall anderer Partitionen noch hochfahren und reparieren zu können.

Weil dieses Wurzeldateisystem selbst sehr viele systemspezifische Dateien enthält, ist es nicht geeignet, über ein Netz von mehreren Rechnern verwendet zu werden.

Viele der Unterverzeichnisse dieses Dateisystems sind nur Mountpoints, um andere Partitionen dort einzuhängen. Einige Verzeichnisse sollten aber unter keinen Umständen auf anderen Plattenpartitionen liegen, da sie sonst nicht zur Verfügung stehen, wenn während des Bootvorgangs bisher nur das Wurzelverzeichnis gemountet ist. Ein simples Beispiel ist das Programm *mount* selbst. Dieses Programm muß natürlich irgendwo auf der Partition liegen, die das Wurzelverzeichnis enthält, sonst steht es in dem Moment nicht zur Verfügung, wo andere Platten eingehängt werden sollen!

In der folgenden Darstellung wird immer auch beschrieben, ob ein Verzeichnis geeignet ist, auf eine andere Partition ausgelagert zu werden, oder ob es zwingend auf der Wurzelpartition sein muß.

Ein weiterer Gesichtspunkt ist die Frage, ob ein Verzeichnis statische oder dynamische Dateien enthält. Wenn in einem Verzeichnis ausschließlich statische Dateien liegen - also Dateien, die im laufenden Betrieb nicht verändert werden müssen - so bietet es sich an, dieses Verzeichnis ReadOnly zu mounten. Das Wurzeldateisystem enthält zwingend sowohl statische, als auch dynamische Dateien, muß also ReadWrite gemountet sein.

/bin - Grundlegende Programmdateien

Das Verzeichnis /bin enthält grundlegende Programmdateien, die während des Bootvorgangs und zur Reparatur des Systems

zwingend notwendig sind. Es muß auf der Wurzelpartition positioniert sein.

/boot - Statische Dateien für den Boot-Loader

Das Verzeichnis /boot enthält die statischen Dateien, die für den Bootloader notwendig sind, um das System überhaupt zu booten. Es ist durchaus möglich, dieses Verzeichnis auf eine eigene kleine Partition zu positionieren. Bei Platten mit mehr als 1024 Zylindern und einem älteren Bootloader ist es sogar notwendig, dieses Verzeichnis auf eine Partition unterhalb des 1024ten Zylinders zu legen. Es macht in der Regel keinen Sinn, diese Partition übers Netz zu teilen.

/dev - Gerätedateien

Dieses Verzeichnis enthält die Gerätedateien, die notwendig sind, um auf jede Art von Hardware zugreifen zu können. Es muß zwingend auf der Wurzelpartition angelegt sein, sonst sind Hardwarezugriffe während des Bootvorgangs unmöglich, was ihn sofort zum Abbruch führen würde. Die hier liegenden Dateien benötigen keinen physikalischen Speicherplatz auf der Partition, wohl aber Inodes.

/etc - Rechnerspezifische Konfigurationsdateien

Auch dieses Verzeichnis enthält Informationen, die notwendigerweise beim Booten gebraucht werden, also muß es auf der Wurzelpartition liegen. Hier finden sich alle wichtigen Konfigurationsdateien des Systems, unter anderem eben auch die Datei `fstab`, die beschreibt, welche Partitionen wohin gemountet werden müssen.

/home - Die Heimatverzeichnisse der User

Hier liegen die Heimatverzeichnisse der einzelnen User. Dieses Verzeichnis ist geradezu dafür vorgesehen, auf einer eigenen Partition zu liegen, damit die User nicht durch eine volle Platte das System zum Absturz bringen können. Alles was hier liegt ist dynamisch, hier werden die Daten der User abgelegt. Sehr sinnvoll ist es auch, dieses Verzeichnis über das Netz mit mehreren Rechnern zu teilen, so daß die verschiedenen User auf jedem Rechner ihre Daten vorfinden.

/lib - Grundlegende Libraries und Kernelmodule

Dieses Verzeichnis muß beim Systemstart zur Verfügung stehen, darf also auf keinen Fall auf einer anderen Partition als der Wurzelpartition liegen. Es enthält sowohl wichtige *shared libraries*, also Programmbibliotheken, ohne die die wichtigsten Programme des Systems nicht arbeiten können, als auch die Kernelmodule, mit anderen Worten die Gerätetreiber. Alles Informationen, die beim Booten zur Verfügung stehen müssen.

/mnt - Ein leerer Mountpoint

Das ist einfach nur ein leeres Verzeichnis, um temporär andere Dateisysteme dorthinein zu mounten. Die meisten modernen Linux-Distributionen enthalten neben diesem Verzeichnis noch mindestens `/cdrom` und `/floppy`, die auch nur leere Verzeichnisse sind, gedacht um eine CDROM bzw. Diskette dort hinein zu mounten.

/opt - Platz für große zusätzliche Programmpakete

Was früher in `/usr` abgelegt wurde, wird heute standardmäßig in dieses Verzeichnis abgelegt: große zusätzliche Programmpakete wie z.B. Netscape, StarOffice, KDE, Postgres usw. Nichts was zum Booten notwendig wäre, also ist dieses Dateisystem bestens

geeignet, um auf einer eigenen (großen) Partition Platz zu finden.

Dieses Verzeichnis enthält zumeist statische Daten, kann also im normalen Betriebsablauf ReadOnly gemountet werden. Es enthält keine rechnerspezifischen Daten, ist also auch geeignet, um gemeinsam im Netz genutzt zu werden.

proc - Prozessinformationen

In diesem Verzeichnis finden sich Dateien, die eigentlich keine sind. Es handelt sich hier um Schnittstellen zum Kernel, die es ermöglichen, bestimmte Informationen vom Kernel zu erhalten und bestimmte Einstellungen im laufenden Betrieb des Kernels zu setzen.

Außerdem besitzt hier jeder laufende Prozess ein Unterverzeichnis, das den Namen der PID trägt und prozess-spezifische Informationen bereitstellt.

/root - Heimat des Systemverwalters

Dieses Verzeichnis ist das Home-Verzeichnis des Systemverwalters. Weil der Systemverwalter im Notfall, etwa bei einer Reparatur des Systems im Single User Mode auf seine Dateien zugreifen können muß, ist dieses Verzeichnis eben nicht unter `/home` sondern auf der Wurzelpartition direkt abgelegt. Es eignet sich also nicht dazu, auf eine andere Partition ausgelagert zu werden.

Bei älteren Unix-Systemen hatte der Systemverwalter kein Home-Verzeichnis, sondern die Wurzel des Systems (root) diente ihm als solches. Das führte aber dazu, daß auf der Wurzel dann einige Dateien lagen, die den Überblick über die Konsistenz des Systems erschwerten. Aus diesem Grund wurde dieses Verzeichnis eingeführt.

/sbin - Grundlegende Systemprogramme

Hier liegen wichtige Systemprogramme, die im Gegensatz zu den Programmen in `/bin` nicht für alle User, sondern nur für den Systemverwalter nötig sind. Alles, was hier liegt ist für den Bootvorgang nötig, darf also nicht auf einer anderen Partition als der Wurzelpartition liegen.

/tmp - Temporärer Speicherplatz

Dieses Verzeichnis ist ein Platz, in dem alle User Temporärdateien anlegen können. Es sollte das Sticky-Bit gesetzt haben, damit boshafte User nicht die Dateien anderer User löschen können. Es ist kein Problem, dieses Verzeichnis auf eine andere Partition zu legen, es enthält nichts, was für den Bootvorgang nötig wäre. Das Verzeichnis ist logischerweise nur für dynamische Dateien benötigt, darf also nicht ReadOnly gemountet werden.

/usr - Die zweite Dateihierarchie

Das /usr-Verzeichnis ist die zweite Hauptsektion des Unix-Dateisystems. Es enthält übers Netz teilbare, statische Daten. Es ist übliche Praxis, dieses Verzeichnis auf eine eigene Partition zu legen, die im Netz geteilt verwendet wird und jeweils ReadOnly gemountet ist. Große Programmpakete sollten - mit Ausnahme des X11-Systems - nicht hier, sondern unter `/opt` abgelegt werden. Der genaue Inhalt dieses Dateisystems wird im nächsten Abschnitt dargestellt.

/var - variable Daten

Frühere Unixe haben bestimmte variable Daten, wie etwa ein Temporärverzeichnis, die Systemlogbücher, Spoolverzeichnisse, usw. im `/usr` Verzeichnis abgelegt. Da dieses Verzeichnis heute unbedingt ReadOnly mountbar sein muß, wurde es nötig, einen Platz für variable (dynamische) Daten zu schaffen. Diese Aufgabe übernimmt heute das Verzeichnis /var.

Damit auch ältere Programme, die noch immer ins `/usr`-Verzeichnis schreiben wollen, nicht abstürzen, existieren heute zumeist die folgenden symbolischen Links im `/usr`-Verzeichnis, die ins `/var`-Verzeichnis verweisen:

- `/usr/spool -> /var/spool`
- `/usr/tmp -> /var/tmp`

Es ist durchaus möglich, dieses Verzeichnis auf eine eigene Partition zu legen und auch der zumindest teilweisen gemeinsamen Nutzung im Netz steht nichts im Weg. Allerdings enthält das `/var`-Verzeichnis einige Rechnerspezifische Daten, die im Netz nicht umbedingt geteilt werden sollten (etwa `/var/lock` - die Lockfiles oder `/var/run` - die ProzessIDs verschiedener Programme).

Das /usr Dateisystem

Die zweite Ebene der Dateisystemhierarchie liegt im `/usr` Verzeichnis. Hier finden sich sehr ähnliche Verzeichnisse, wie auf der Wurzel des Systems. Allerdings sind alle Daten, die hier liegen statisch, müssen also im normalen Systembetrieb nicht verändert werden. Im einzelnen sind hier folgende Unterverzeichnisse wichtig:

X11R6 - Das X Window System, Version 11, Release 6

Hier liegen verschiedene Verzeichnisse, die wiederum die Binärdateien (bin), die Libraries (lib), Include-Dateien (include) und vieles mehr beinhalten, die alle zum X11-System gehören. Das ist das einzige große Programmpaket, das immer noch unter /usr liegt und nicht unter /opt.

bin - Die meisten User Kommandos

Hier liegen all die Userkommandos, die nicht unbedingt während des Systemstarts oder einer Notfallreparatur benötigt werden.

games - Spiele

Hier finden sich Spiele und andere zum Teil erzieherische Software.

include - Die Include-Dateien des C-Compilers

Hier finden sich die ganzen Include-Dateien, die sowohl für den C-, als auch für den C++ Compiler notwendig sind. Das Verzeichnis enthält auch viele Unterverzeichnisse, die wiederum Includedateien enthalten.

lib - Bibliotheken für Programme und Pakete

Dieses Verzeichnis enthält Objektdateien, Programmbibliotheken und interne Binärdateien, die nicht direkt von Usern aufgerufen werden sollen. Anwendungen können unter `/usr/lib` ein eigenes Verzeichnis haben, um ihre Daten dort abzulegen.

local - Die dritte, lokale Hierarchie

Dieses Verzeichnis sollte nach der Hauptinstallation leer sein. Hier ist Platz für eine weitere Dateisystemhierarchie, mit dem gleichen Inhalt wie `/usr` selbst. Nur sollten hier die lokalen Besonderheiten abgelegt sein, statt der systemspezifischen Standards.

sbin - Nicht lebensnotwendige Systemprogramme

Hier finden sich die Systemprogramme, die nicht für alle User, sondern nur für den Systemverwalter gedacht sind und die nicht während des Boot- oder Reparaturvorgangs benötigt werden.

share - Geteilte Daten

Hier liegen architekturunabhängige Daten, wie etwa Handbuchseiten, Dokumentationen, Wörterbücher, Sound usw.

src - Der Quellcode für Programme

Aller Quellcode für Systemprogramme, inclusive der des Systems selbst ist hier zu finden. Um hier Code zu übersetzen muß natürlich Schreibmöglichkeit existieren. Aus diesem Grund wird dieses Verzeichnis entweder als eigene Partition realisiert, die ReadWrite gemountet wird, oder der Systemverwalter remountet das /usr Verzeichnis ReadWrite, um Programme zu übersetzen.

Das /var Dateisystem

Vereinfacht gesagt gehört alles, was nicht statische Daten ist aber eigentlich unter `/usr` zu finden wäre, heute in dieses Verzeichnis. Insbesondere ist hier zu finden:

lib

Variable Statusinformationen der Programmbibliotheken

lock

Lockfiles, die anzeigen, daß ein bestimmtes Gerät gerade in Benutzung ist.

log

Die Systemlogbücher und Unterverzeichnisse für die Logdateien einzelner Anwendungen (z.B. Webserver, Mailsystem)

run

Prozessinformationen über gerade laufende Prozesse. Viele Daemon-Prozesse schreiben hier eine Datei hinein, die nur die ProzessID des laufenden Daemons enthält.

spool

Das Spoolverzeichnis für alle verschiedenen Dienste, die Warteschlangen unterstützen. Dazu zählen z.B. Druckerdienste, Faxdienste, Maildienste und Programme wie at oder cron.

tmp

Das Temporärverzeichnis, das eigentlich in `/usr` liegt, dort aber nur ein symbolischer Link auf dieses Verzeichnis ist.

Auffinden von Dateien und Programmen

Die oben beschriebene Struktur des Linux-Dateisystems macht es zwar leichter, einzelne Dateien und Programme innerhalb des Systems zu finden, aber wir brauchen natürlich noch Mechanismen und Programme, die uns das Auffinden einzelner Dateien ermöglichen.

Bei dieser Frage müssen wir zwischen Programmen und Dateien unterscheiden. Ein Programm ist eine ausführbare Datei, im folgenden wird der Begriff aber etwas genauer definiert. Linux enthält, wie oben gesehen, einige Verzeichnisse, die für die Aufnahme von Programmen (Binärdateien und Scripts) vorgesehen sind. Zumeist enden diese Verzeichnisse mit den Buchstaben `bin`. Damit einzelne Programme von überall her aufrufbar sind, ohne jeweils ihren ganzen Pfad eingeben zu müssen, existiert ein sogenannter Programmsuchpfad. Das ist eine Liste all der Verzeichnisse, die von der Shell durchsucht werden, wenn ein Programmaufruf eingegeben wurde. Diese Liste ist in einer Umgebungsvariable namens `PATH` gespeichert. Ein Normaluser hat hier meist andere Einstellungen als der Systemverwalter, der in seinem Pfad noch die verschiedenen `sbin`-Einträge hat.

Um herauszufinden, welche Verzeichnisse im Suchpfad stehen reicht der Aufruf von

```
echo $PATH
```

Eine typische Ausgabe für den Systemverwalter wäre z.B

```
/sbin:/usr/sbin:/usr/local/sbin:/root/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:
/bin:/usr/games/bin:/usr/games:/opt/gnome/bin:/opt/kde/bin:.
```

Ein Normaluser würde aber eher etwas in der Art vorfinden:

```
/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/usr/games/bin:/usr/games:
/opt/gnome/bin:/opt/kde/bin:.
```

Wenn wir jetzt Programme suchen, so existieren Tools, die nur diesen Suchpfad durchforsten und uns sagen, welche Programme von wo aus aufgerufen werden. Suchen wir hingegen Dateien, so sind wir gezwungen, den gesammten Bestand an Verzeichnissen zu durchsuchen, was natürlich erheblich mehr Arbeit ist und länger dauert.

Für die Programme ist noch anzumerken, daß die Shell, wenn sie einen Programmaufruf ausführt, den Suchpfad von vorne nach hinten (von links nach rechts) durchsucht und das erste gefundene Programm mit dem passenden Namen ausführt. Das kann zu Konflikten führen, wenn wir z.B. zwei Programme gleichen Namens auf dem System haben, das eine in /bin und das andere in /usr/bin. Nachdem /usr/bin im Suchpfad vor /bin steht wird immer das Programm in /usr/bin aufgerufen.

Programme finden mit which, whereis und type

Wenn wir Programme suchen, so stehen uns dafür drei Tools zur Verfügung: which, whereis und type. Alle durchsuchen jeweils nur den Suchpfad, finden also nur die Programme, die die Shell auch finden würde.

Das Programm **which** wird zusammen mit einem oder mehreren Programmnamen aufgerufen und gibt uns dann den vollen Pfad des gefundenen Programms zurück. which arbeitet exakt wie die Shell, durchsucht eben den Suchpfad (`PATH`) und gibt das erste Kommando mit passendem Namen samt Pfad zurück. Der Aufruf von `which mount` hätte also die folgende Ausgabe gebracht:

```
/bin/mount
```

Das Programm **type** ist eigentlich gar kein Programm sondern eine interne Funktion der Shell (bash). Es zeigt etwas mehr Information als which, weil auch die hash-Mechanismen der Shell berücksichtigt werden und angezeigt wird, ob das gewünschte Kommando eine Shellfunktion, ein Alias, ein interner Shell-Befehl oder ein Programm ist.

Auf vielen Systemen ist tatsächlich etwa das Programm which nur ein Alias auf `type -p`.

Die Ausgabe von `type mount` würde folgendermaßen aussehen:

```
mount is /bin/mount
```

Das Programm whereis ist schließlich neben dem Auffinden von Programmdateien auch in der Lage, die zugehörigen Handbuchseiten und den Quellcode (falls installiert) von Programmen zu lokalisieren.

Der find-Befehl

Um jetzt nicht nur nach Programmen im Suchpfad zu suchen, sondern nach allen möglichen Dateien im System und um auch alle denkbaren Suchkriterien formulieren zu können, gibt es das Programm find.

find bietet aber neben der Fähigkeit Dateien zu suchen auch noch an, beliebige Aktionen mit den gefundenen Dateien auszuführen. Aus diesem Grund ist dieses Programm ein unentbehrliches Hilfsmittel für den Systemvewalter.

Die genauen Parameter, Tests und Aktionen von **find** sind auf der Handbuchseite aufgelistet, hier soll noch einmal in Beispielen erklärt werden, wie find aufgerufen wird und wie man damit vernünftig arbeiten kann.

Die einfache Aufrufform von **find** ist

find ***Startverzeichnis Test Aktion***

Dabei sucht **find** alle Verzeichnisse ab *Startverzeichnis* nach den Dateien ab, für die die Kriterien *Test* zutreffen. Für jede gefundene Datei wird dann die *Aktion* durchgeführt. Wenn die Aktion weggelassen wird, so wird standardmäßig die Aktion **-print** ausgeführt, d.h. der Dateiname der gefundenen Datei wird zusammen mit seinem Suchpfad (relativ zum angegebenenen Startverzeichnis) ausgegeben.

Um Dateien anhand ihres Namens zu finden, wird der Test **-name *Muster*** benutzt. Wenn der Name nicht vollständig bekannt ist, oder nach verschiedenen Dateien mit bestimmten Namen gesucht werden soll, dann kann das Suchmuster auch die Jokerzeichen verwenden, die auch die Shell benutzt (*,?,[...]). Aber Vorsicht: Das Namensmuster muß dann in Anführungszeichen gesetzt werden, weil sonst die Shell die Jokerzeichen interpretieren und durch evt. gefundene Dateien im aktuellen Verzeichnis ersetzen würde.

Um also z.B. alle Dateien des gesamten Systems zu finden, deren Namen mit einem großen oder kleinen M beginnt schreiben wir:

```
find / -name "[mM]*"
```

Der Slash steht für das Startverzeichnis, hier also das Wurzelverzeichnis. **-name** leitet den Namenstest ein und **"[mM]*"** ist das Suchmuster. Die Aktion haben wir hier einfach weggelassen, es wird also die **-print** Aktion durchgeführt. Das Ergebnis sieht dann etwa so aus:

```
/usr/X11R6/bin/macptopbm
/usr/X11R6/bin/mgrtopbm
/usr/X11R6/bin/mtvtoppm
/usr/X11R6/bin/makedepend
/usr/X11R6/bin/makeg
/usr/X11R6/bin/mergelib
/usr/X11R6/bin/mkdirhier
/usr/X11R6/bin/mkfontdir
/usr/X11R6/bin/mksusewmrc
/usr/X11R6/bin/manix
/usr/X11R6/bin/mirrormagic
/usr/X11R6/bin/mogrify
/usr/X11R6/bin/montage
/usr/X11R6/bin/mtv
/usr/X11R6/bin/mtvp
/usr/X11R6/bin/mpeg_play
/usr/X11R6/include/GL/MesaDrawingArea.h
/usr/X11R6/include/GL/MesaDrawingAreaP.h
/usr/X11R6/include/GL/MesaMDrawingArea.h
/usr/X11R6/include/GL/MesaMDrawingAreaP.h
```

```
/usr/X11R6/include/GL/MesaWorkstation.h
/usr/X11R6/include/GL/MesaWorkstationP.h
/usr/X11R6/include/X11/Xaw3d/MenuButtoP.h
...
```

Wenn mit Tests gearbeitet wird, die nummerische Parameter haben, dann gibt es eine Besonderheit. Wird die Zahl einfach, ohne Vorzeichen angegeben, dann heißt das, daß genau diese Zahl gemeint ist. Hat die Zahl als Vorzeichen ein Pluszeichen (+), dann heißt das, daß die Zahl oder eine größere Zahl gemeint ist, ein Minuszeichen bedeutet entsprechend die Zahl oder eine kleinere.

Der Test **-size** sucht Dateien nach ihrer Größe. Schreiben wir also

```
find . -size 12k
```

so sucht **find** nach allen Dateien im aktuellen Verzeichnis (.), die genau 12 Kilobyte groß sind. Wenn wir alle Dateien suchen, die höchstens 12 Kilobyte groß sind, so schreiben wir

```
find . -size -12k
```

und Dateien, die mindestens 12 Kilobyte groß sind finden wir mit

```
find . -size +12k
```

Wenn wir mehrere Tests angeben, so werden die Dateien gesucht, auf die alle Tests zutreffen, nicht die, die einen von beiden Tests bestehen. Der Befehl

```
find . -size +12k -name "M*"
```

sucht also nach allen Dateien im aktuellen Verzeichnis, die mindestens 12 Kilobyte groß sind **UND** deren Namen mit einem M beginnt.

Richtig interessant wird **find** erst mit der Anwendung von Aktionen. Die wichtigste und sicherlich meistgebrauchte Aktion ist **-exec**. Mit dieser Aktion lassen sich beliebige Unix-Kommandos ausführen, die mit den gefundenen Dateien angewendet werden. Dabei gibt es zwei wichtige Punkte zu beachten.

1. Die Befehlszeile der **-exec** Aktion muß mit einem **\;** abgeschlossen werden, damit eindeutig klar wird, was ist Befehlszeile und was weitere Aktionen. Der Strichpunkt muß mit einem Backslash versehen sein, damit ihn die Shell nicht interpretiert. Außerdem muß er durch ein Leerzeichen von der Befehlszeile getrennt sein.
2. Der Namen der gefundenen Datei wird mit einem **{}** dargestellt. Trifft **find** auf eben diese zwei geschweiften Klammern, so ersetzt es diese durch den gefundenen Dateinamen.

Um also etwa alle Dateien des ganzen /usr Verzeichnisses, die größer als 12 Kilobyte sind und deren Namen mit M beginnt in das

Verzeichnis /tmp zu kopieren schreiben wir:

```
find /usr -size +12k -name "M*" -exec cp {} /tmp \;
```

Oder, um ein etwas praktischeres Beispiel zu nennen, wenn der Systemverwalter alle Dateien löschen will, die Usern gehören, die es nicht mehr gibt, dann kann er schreiben:

```
find / -nouser -exec rm {} \;
```

Das ist zweifellos ein gewisses Risiko, es könnten ja auch wichtige Dateien dabei sein, die durch einen Zufall einer UserID zugewiesen wurden, die nicht bekannt ist. Statt **-exec** können wir in so einem Fall auch **-ok** benutzen. Es macht exakt genau das Gleiche wie **-exec**, fragt aber bei jeder Ausführung vorher nach, ob es die Befehlszeile wirklich ausführen soll.

Die zentrale Dateidatenbank

Linux (und andere Unixe) verwalten eine zentrale Datenbank, die alle Dateien, die auf dem System gespeichert sind als Namenseintrag mit komplettem Pfad speichern und die so auch das Suchen nach Dateien ermöglicht. Damit diese Datenbank auf einem möglichst aktuellen Stand gehalten wird, muß z.B. täglich ein Programm laufen, das die gesammte Festplatte durchscannt und die gefundenen Dateinamen in dieser Datenbank abspeichert.

Dabei kommen unterschiedliche Techiken zum Zuge, es kann sein, daß die Datenbank nur in einer einzigen Datei liegt (z.B. `/var/lib/locatedb`) oder daß sie verteilt auf verschiedenen Dateisystemen vorliegt. Unter Linux ist bei den meisten Distributionen der Standort `/var/lib/locatedb` voreingestellt.

Das Programm, das diese Datenbank anlegt und täglich auffrischen sollte heißt **updatedb** und wird gewöhnlich via cron einmal täglich aufgerufen. Es erstellt eine Datenbank indem es alle angeschlossenen Laufwerke durchscannt und jede einzelne Datei samt Pfad speichert. Da das eine sehr aufwendige Aktion ist, wird dieser Programmaufruf in der Regel mitten in der Nacht ausgeführt und es existieren Möglichkeiten, bestimmte Verzeichnisse oder auch bestimmte Dateisysteme (wie etwa Netzlaufwerke) von der Suche auszunehmen.

In älteren Versionen von **updatedb** wurde diese Ausnahmeregelung in der Datei `/etc/updatedb.conf` vorgenommen. Hier wurden einfach Umgebungsvariablen definiert, die verschiedene Pfade ausschlossen, eine typische solche Datei könnte wie folgt aussehen:

```
# File: /etc/updatedb.conf
# This file sets environment variables which are used by updatedb

# filesystems which are pruned from updatedb database
```

```
PRUNEFS="NFS nfs afs proc smbfs autofs auto iso9660 ncpfs coda"
export PRUNEFS

# paths which are pruned from updatedb database
PRUNEPATHS="/tmp /usr/tmp /var/tmp /afs /amd /alex /var/spool"
export PRUNEPATHS

# netpaths which are added
NETPATHS=""
export NETPATHS
```

Die erste Anweisung (`PRUNEFS="NFS nfs afs proc smbfs ..."`) definiert Dateisystemtypen, die nicht durchsucht werden sollen. Die zweite (`PRUNEPATHS="/tmp /usr/tmp /var/tmp ..."`)) definiert Pfade, die nicht durchsucht werden sollen.

Neuere Versionen von **updatedb** arbeiten mit Kommandozeilenoptionen, die den selben Effekt bieten.

Wozu diese Datenbank? Ein schnelles Auffinden von Dateien anhand von Namensmustern ist mit `find` nicht so einfach möglich. Denn `find` durchsucht ja tatsächlich die Festplatten nach den Dateien und das dauert. Da aber das Suchen nach Namensmustern sicherlich die häufigste Form ist, existiert das Programm **locate**. Dieses Programm durchsucht nicht die Festplatten, sondern eben die Datenbank, die durch **updatedb** erstellt wurde. Das geht enorm schnell und effektiv, hat jedoch den Nachteil, daß Dateien, die seit dem letzten Aufruf von **updatedb** erstellt wurden natürlich nicht gefunden werden können.

locate erlaubt die Verwendung von Suchmustern wie die Shell sie anbietet, also *, ?, [...]. Allerdings wird tatsächlich der ganze Dateinamenstring sammt Pfad durchsucht, ein Suchmuster von `foo*bar` würde also auch eine Datei `/usr/foo/bla/bar` finden.

Neben **locate** existiert alternativ auch das Programm **slocate**, das einen sichereren Zugriff gewährleistet. **slocate** erstellt die zentrale Datenbank und speichert aber zusätzlich auch die Eigentümerschaft und Zugriffsrechte der Dateien ab. So kann ein Normaluser nur die Dateien mit **slocate** finden, auf die er auch Zugriff hätte.

Insgesammt kann man zusammenfassen, daß locate und slocate eine schnelle und für normale Systemdateien durchaus zufriedenstellende Möglichkeit darstellt, Dateien anhand ihres Namens oder eines Suchmusters im Dateisystem zu finden. Wenn es aber darum geht, Dateien zu finden, die womöglich nach dem letzten Aufruf von updatedb erstellt wurden oder die Suchkriterien nicht den Namen der Datei benutzen, dann muß find benutzt werden.

1.110.1

Installation und Konfiguration von XFree86

Beschreibung: Prüfungskandidaten sollten in der Lage sein, X und einen X-Fontserver zu installieren und zu konfigurieren. Dieses Lernziel beinhaltet die Überprüfung, ob Grafikkarte und Monitor von einem X-Server unterstützt werden, und das Anpassen und Trimmen von X für Grafikkarte und Monitor. Ebenfalls enthalten ist die Installation eines X-Fontservers, die Installation von Schriftarten und das Konfigurieren von X zur Benutzung des Fontservers (möglicherweise durch manuelles Bearbeiten des Abschnitts "Files" in `/etc/X11/XF86Config`).

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **XF86Setup**
- **xf86config**
- **xvidtune**
- `/etc/X11/XF86Config`
- `.Xresources`

Grundlagen

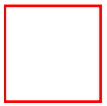
Das X-Window-System oder kurz X11 ist ein netzbasiertes graphisches Fenstersystem, das auf einer *Client/Server* Architektur beruht. Programme, die ihre Ausgaben in Fenstern machen wollen sind Clients, die den Service in Anspruch nehmen, den der *Display-Server* anbietet, nämlich Ausgabe auf dem Schirm zu bringen und Eingaben von der Tastatur und Mouse entgegenzunehmen.

Dabei ist wichtig, daß es möglich ist, daß Client und Server über ein Netzwerk miteinander kommunizieren können. Es ist also unerheblich, ob die Programme, die auf einem Bildschirm dargestellt werden auf dem lokalen Rechner laufen oder auf einem anderen Rechner im Netz.

Client und Server verständigen sich mittels einem Protokoll, dem sogenannten *X-Protokoll*. Dieses Protokoll nutzt als Transportbasis meist TCP/IP, aber auch DECNet oder der Betrieb auf reinen Unix-Sockets ist möglich. Technisch gesehen ist das X-Protokoll die eigentliche Definition des X-Window-System.

Durch die Definition eines eindeutigen Protokolls, das alle Fähigkeiten zur Übermittlung von graphischen Ausgaben beinhaltet wird X11 zur hardwareunabhängigen Graphikplattform. Das X11-System ist aufgegliedert in einen Displayserver und in Programme, die den Display-Service in Anspruch nehmen, also Clients. Ein Server kann für jede beliebige Art von Graphiksystemen geschrieben werden, über das vordefinierte X-Protokoll können beliebige Clients dann ihre Ausgaben auf dem Server machen.

Natürlich können Clients (also X11-Programme) auch lokal auf dem selben Rechner laufen, auf dem auch der Display-Server läuft, das ist sogar die Regel. Aber es können eben auch beliebige andere Rechner Programme laufen haben, deren Ausgaben dann auf unserem Server zu sehen sind.



Zusammenfassend können wir also sagen, das X11-System beruht auf drei Einzelelementen:

1. Der **Display-Server**, der alle Ausgaben auf den Bildschirm steuert und die Eingaben von der Tastatur und der Maus entgegennimmt. Er ist ein Programm, das lokal auf dem Rechner läuft und Kontrolle über die Grafik-Hardware hat.
2. Ein oder mehrere **Clients**, also Programme, die ihre Ein/Ausgabe nicht selbst erledigen sondern diese Aufgabe vom Display-Server erledigen lassen. Kurz gesagt sind alle X-Anwendungen (xterm, xsnow, xteddy, ...) Clients. Es ist unerheblich, auf welchem Rechner im Netz diese Programme laufen, solange sie mit dem Display-Server verbunden sind geht ihre Ausgabe auf den Bildschirm dieses Servers.
3. Das **X-Protokoll**, das die Kommunikation zwischen den Clients und dem Display-Server regelt. Dieses Protokoll ist das einzige, das X11 eigentlich ausmacht. Prinzipiell kann für jedes graphikfähige Gerät ein X-Server geschrieben werden, wenn er das X-Protokoll versteht, kann er X-Clients erlauben, ihre Ausgaben auf dem Gerät auszugeben.

Bei der hier gestellten Aufgabe geht es also um die Installation eines X-Servers, also des Programms, das anderen Programmen (den X-Clients) den Dienst anbietet, sich auf dem Bildschirm darstellen zu können und die Eingabegeräte Tastatur und Mouse zu nutzen. Beim Umgang mit X11 ist es grundsätzlich wichtig, sich dieser Client-Server-Architektur immer bewußt zu sein, um tatsächlich zu verstehen, was eigentlich vorgeht.

Es gibt X-Server von verschiedensten Herstellern für verschiedenste Hard- und Softwaresysteme. Der hier verwendete X-Server ist der X-Free-86 Server. Dieser Server ist freie Software und für die Verwendung auf 80x86 Systemen (und kompatiblen Prozessoren) gedacht. Aus diesen Eigenschaften bezog er seinen Namen. XF86 ist der unter Linux verbreitetste Server, andere angebotene Server sind meist kommerzielle Produkte und spielen in der Praxis kaum eine Rolle.

Die Konfiguration des XFree86-Servers

Der Display-Server verwaltet die Hardware, deren Verwendung er anderen Clients ermöglichen soll. Er muß daher sehr genau über die installierte Graphik- und Eingabehardware Bescheid wissen. Insbesondere die folgenden Hardware-Elemente muß er verwalten:

- Graphikkarte
- Monitor
- Tastatur
- Mouse
- Andere Eingabegeräte (Joystic, Grafiktablett,...)

Die gesamten Angaben über diese Geräte und auch alle anderen Konfigurationseinstellungen befinden sich in einer einzigen Datei. Diese Datei kann manuell bearbeitet werden, es ist eine einfache Textdatei. Sie heißt `XF86Config` und befindet sich in der Regel im Verzeichnis `/etc/X11`. Die Orginalversion von XF86 hatte eigentlich andere Orte für die Konfigurationsdatei vorgesehen, aber im neuen Filesystem Hierarchy Standard ist `/etc/X11` festgelegt worden. Die meisten Distributionen halten sich heute an diesen Standard. Der Vollständigkeit halber seien hier nochmal die ursprünglichen Konfigurationsdateien genannt:

- **~/XF86Config**
  Wenn der User, der den X-Server manuell startet der Systemverwalter ist, so wird zuerst in seinem Homeverzeichnis nach der Konfigurationsdatei gesucht.
- **/etc/XF86Config**
  Das war früher der Standard für die systemweite Konfigurationsdatei des X-Servers.
- ***X-Wurzel*/lib/X11/XF86Config.*hostname***
  Falls in /etc keine Konfigurationsdatei gefunden wurde, so wurde in diesem Verzeichnis danach gesucht. *X-Wurzel* bezeichnet die Wurzel des X-Systems, meist `/usr/X11R6`. Damit geteilte `/usr`-Verzeichnisse benutzt werden konten, konnte für verschiedene Rechner je eine Konfigurationsdatei angelegt werden, die dann den Hostnamen des Rechners als Namenserweiterung trug, für den sie gedacht war.
- ***X-Wurzel*/lib/X11/XF86Config**
  Falls an dieser Stelle keine Datei mit dem passenden Hostnamen gefunden wurde, wurde nach einer Konfigurationsdatei ohne Hostnamenendung gesucht.

Noch heute funktionieren diese Dateien, werden aber kaum noch benutzt.

Weil heute zwei XFree86 Versionen im Umlauf sind (Version 3.x und Version 4.x) und die beiden unterschiedliche Formate ihrer Konfigurationsdatei erwarten, haben verschiedene Distributionen unterschiedliche Mechanismen entwickelt, um einen gleichzeitigen Betrieb beider Versionen zu ermöglichen. Entweder liegen beide Konfigurationsdateien in `/etc/X11`, dann trägt die Datei für die Version

4.x den Namen `XF86Config-4`, oder es wurde die Konfigurationsdatei für Version 3.x in `/etc` belassen und die von Version 4.x liegt in `/etc/X11`. Im weiteren Verlauf werden wir hier keine Unterschiede machen, sondern die Datei einfach immer `/etc/X11/XF86Config` nennen.

Früher musste die Konfigurationsdatei von Hand erstellt werden und es bestand die Gefahr, daß - bei falschen Monitor-Einstellungen - die Hardware tatsächlich beschädigt werden konnte. Die Errechnung der Bildschirmparameter war eine umständliche und schwierige Arbeit, die nicht immer das erreicht hatte, was gewünscht war. Durch die Architektur moderner Graphikkarten und Monitore ist diese Konfigurationsarbeit heute erheblich erleichtert worden. Zudem gibt es eine ganze Reihe von Hilfsmitteln, um die Konfigurationsdatei erstellen zu lassen.

xf86config

Ein kleines textorientiertes Programm, das die ganze Abfrage aller notwendigen Hardwareinformation vornimmt, und anschließend eine XF86Config Datei daraus erstellt. Dieses Programm war die erste automatisierte Methode von XFree86 und gehört zur Standard-Ausstattung.

XF86Setup

Auch dieses Programm wird von XFree86 selbst betreut und ausgeliefert. Im Gegensatz zu **xf86config** arbeitet es aber bereits in einem graphischen Modus. Das Programm startet in einem VGA16 Graphikmodus, der von praktisch jeder Graphikkarte unterstützt wird, und arbeitet mit einer Bildschirmauflösung von 640x480 Pixeln, die jeder Bildschirm darstellen können sollte. Auch dieses Programm erstellt eine XF86Config Datei aus den gemachten Angaben, ist aber dann in der Lage, die Konfiguration gleich auszuprobieren.

SAX und SAX2

Sind Programme der SuSE-Distribution (SAX heißt SuSE Advanced X-Configuration) und stehen nur dort zur Verfügung. SAX erstellt Konfigurationsdateien für Version 3.x, Sax2 für Version 4.x. Beide Programme arbeiten wie **XF86Setup** bereits im graphischen Modus, versuchen jedoch bereits vor dem Start des Graphikmodus durch einen Scan des PCI-Busses herauszufinden, welche Einstellungen notwendig sind.

Xconfigurator

ist das X11-Konfigurationsprogramm von RedHat. Es handelt sich um eine textorientierte verbesserte Version von **xf86config**, die auch vorher den PCI-Bus scant, um die angeschlossenen Geräte zu erkennen. Über die SLANG-Library wurde die Benutzerführung verbessert. Das Programm kann im interaktiven Modus laufen oder automatisch aus den gefundenen Informationen eine Konfigurationsdatei erstellen (kickstart).

dexconf

ist das Programm der Debian Distribution, das aus den gemachten Angaben der Debian-Konfiguration eine entsprechende XF86Config Datei erstellt. Das Programm wird automatisch während der Installation aufgerufen, ist also kein wirkliches usergeführtes Konfigurationsprogramm für XFree86. Sollte unter Debian eine manuelle Konfiguration erwünscht sein, so wird der Befehl

```
dpkg-reconfigure xserver-xfree86
```

verwendet. **dexconf** wird dann abschließend automatisch aufgerufen.

Als distributionsunabhängige Zertifizierung genügt für LPI das Wissen um die ersten beiden Programme, alle anderen sind spezielle Lösungen der Distributionen. All diese Programme erledigen die selben Aufgaben (auf unterschiedliche Weise), die hier nochmal genauer dargestellt werden sollen. Im Wesentlichen handelt es sich um zwei Aufgaben:

1. Abfrage der Hardware und erstellen einer passenden XF86Config Datei.
2. Erstellen eines Links auf den zu verwendenden X-Server.

Beide Aufgaben werden jetzt nochmal im Einzelnen besprochen:

Abfrage der Hardware und erstellen einer passenden XF86Config Datei

Der X-Server steuert die Graphik- und Eingabehardware um sie dann den Clients zur Verfügung zu stellen. Er muß also sehr genau über die verwendeten Geräte Bescheid wissen und benötigt zum Teil sehr technische Informationen. Folgende Informationen werden abgefragt, müssen also bekannt sein:

Tastatur

Diese Einstellung ist sicher die einfachste. Tastaturen sind stark standardisiert und dem Betriebssystem auch im Textmodus bekannt. Wichtige Einstellungen sind hier

- Anzahl der Tasten (pc101, pc102, pc104)
- Tastaturlayout (Sprache - de, en, ...)
- Zusätzliche Optionen, wie mit Akzent-Tasten umgegangen werden soll (nodeadkeys)

Normalerweise sollten diese Einstellungen kein Problem darstellen.

Maus

Bei der Mouse haben wir schon wesentlich mehr Fragen zu beantworten. Dazu zählen

- An welcher Schnittstelle ist die Mouse angeschlossen (seriell, PS/2, USB)?
- Welches Mouse-Protokoll verwendet die Mouse (imps/2, microsoft, logitech, mmseries, ...)?
- Wieviele Tasten hat die Mouse?
- Soll (bei 2-Tasten Mäusen) die Emulation der mittleren Taste durch gleichzeitiges Drücken der beiden Tasten aktiviert werden (Emulate3Buttons)?
- Soll ein eventuell vorhandenes Scroll-Rädchen aktiviert werden (ZAxisMapping)?

Die Einstellungen können - zumindest bei den graphischen Setup-Programmen - gleich ausprobiert werden. Wichtig ist, daß zumindestens die Anschluß- und Mouseart stimmen, weil XFree86 ohne Mouse gar nicht startet.

Graphikkarte

Hier wird in der Regel bei modernen Rechnern alles automatisch erkannt. Durch die Verwendung von PCI/AGP Karten ist eine automatische Konfiguration möglich. Ansonsten müssen hier - bei alten Rechnern - folgende Angaben gemacht werden:

- Chipsatz der Graphikkarte (oder Auswahl aus einer Liste bekannter Karten).
- Größe des Graphikspeichers.
- Zu verwendender X-Server (Nur Version 3.x)
- Clocks (nur bei alten Karten)

Die Angabe der Clocks ist bei modernen Karten nicht mehr notwendig, weil sie diese Angaben auf Anfrage selbst machen. Bei sehr alten Karten, die nicht in der Kartendatenbank vorhanden waren, war diese Angabe sowohl lebenswichtig, als auch nahezu unlösbar...

Bildschirm

Um einen Bildschirm richtig ansteuern zu können, benötigt XFree86 die folgenden Angaben, die im Handbuch des Monitors zu finden sein sollten:

- Horizontale Synchronisation in kHz (entweder eine Liste von festen Frequenzen durch Kommas voneinander getrennt, oder - bei fast allen modernen Monitoren - ein Bereich, der durch Bindestrich verbunden ist, etwa 30-96).
- Vertikale Synchronisation (Refreshrate) in Hz. (Wie bei der horizontalen Synchronisation entweder eine Liste oder ein Bereich).

Diese Angaben müssen stimmen, sonst wird der Monitor falsch angesteuert. Bei modernen Monitoren kann zwar dabei nichts mehr kaputtgehen, aber es wird nichts oder die Fehlermeldung *out of range* ausgegeben.

Neben der Angabe der Monitordaten werden von den Konfigurationsdateien auch noch die gewünschten Bildschirm-Auflösungen und Farbtiefen erfragt. Es können mehrere angegeben werden, die dann später im laufenden Betrieb umgeschaltet werden können.

Neben diesen Hardware-Angaben werden eventuell noch verschiedene Pfade abgefragt, die zu weiteren Informationen führen. Diese Pfade werden später noch einer genaueren Betrachtung unterworfen. Für die Konfiguration reicht hier die Übernahme der Vorgaben.

Erstellen eines Links auf den zu verwendenden X-Server

Der eigentliche X-Server wird später auf verschiedene Art und Weise gestartet. Gemeinsam ist allen Methoden aber, daß sie das Programm `/usr/X11R6/bin/X` aufrufen. Es wird also immer davon ausgegangen, daß der X-Server einfach nur X heißt.

In den XFree86 Versionen vor 4.0 gab es nicht nur einen, sondern sehr viele unterschiedliche X-Server. Jeder Server war für eine bestimmte Graphikkarte (oder eine Klasse von Karten) zuständig. Typische Beispiele dieser Server waren **XF86_Mach64, XF86_VGA16, XF86_SVGA** oder **XF86_S3**. Damit das jeweils verwendete Startprogramm auch den richtigen Server startet, ist das Programm `/usr/X11R6/bin/X` nur ein symbolischer Link auf den entsprechenden Server. Genauer gesagt ist X ein Link auf `/var/X11R6/bin/X` und das ist wiederum ein Link auf den entsprechenden X-Server. Der Grund für diese doppelte Verlinkung ist, daß ansonsten bei einem ReadOnly gemounteten `/usr`-Verzeichnis keine Änderungen möglich wären. Die Verlinkung bei XFree86 Version 3.x sieht also folgendermaßen aus:

```
/usr/X11R6/bin/X => /var/X11R6/bin/X => /usr/X11R6/bin/XF86_Servertyp
```

Diese Links werden auch vom jeweils verwendeten Konfigurationsprogramm angelegt.

Ab der Version 4.0 von XFree86 hat sich diese Architektur geändert. Es war einfach zu viel Aufwand, für jede Graphikkarte einen eigenen X-Server zu schreiben. Aus diesem Grund wurde ein modularisierter X-Server erstellt, der die kartenabhängigen Informationen als Modul nachläd. Der eigentliche X-Server heißt jetzt `/usr/X11R6/bin/XFree86`. Trotzdem existiert meist immer noch der Link von `/usr/X11R6/bin/X` auf `/usr/X11R6/bin/XFree86`. Nur bei Debian ist `/usr/X11R6/bin/X` ein kleines Ladeprogramm, das dann aber auch `/usr/X11R6/bin/XFree86` läd.

Tunen der Konfiguration

Nach abgeschlossener Konfiguration kann der X-Server gestartet werden. Im einfachsten Fall wird das durch die Eingabe des Befehls **startx** erfolgen. (Andere Startmethoden werden im Abschnitt 1.110.2 - Einrichten eines Display Managers beschrieben.)

Falls die Bildschirmeinstellungen nicht zufriedenstellend sind, gibt es zwei Möglichkeiten, daran noch etwas zu verändern. Entweder der Monitor wird mittels seiner Hardwarekonfiguration (Monitormenü oder Drehknöpfe) an die Konfiguration angepasst, oder die Konfiguration wird im Nachhinein noch verändert. Dazu steht uns das Programm **xvidtune** zur Verfügung.

Mit Hilfe dieses Programms können die Bildschirmeinstellungen ähnlich verändert werden, wie mit dem Menü des Monitors selbst. Im Unterschied dazu wird **xvidtune** aber die Einstellungen in der XF86Config Datei verändern, statt sie hardwareseitig einzustellen.

Mögliche Einstellungen sind:

- **Left, Right, Up, Down**
  Verändert den Video-Modus dahingehend, daß sich das sichtbare Bild in die angegebene Richtung verschiebt.
- **Wider, Narrower, Shorter, Taller**
  Verändert den Video-Modus dahingehend, daß das Bild breiter, schmäler, flacher oder höher wird.

Die beste Methode ist es, die Bildschirmeinstellungen mit **xvidtune** soweit wie möglich einzustellen, und dann anschließend das Feintuning

mit der Monitoreinstellung vorzunehmen.

Struktur der Datei XF86Config

Alle Konfigurationseinstellungen des X-Servers werden in der Datei `/etc/X11/XF86Config` abgelegt. Diese Datei kann manuell mit einem Texteditor verändert werden. Ihr Format unterscheidet sich zwischen den Versionen 3.x und 4.x erheblich.

Wichtige Einstellungen für die LPI101 Prüfung beziehen sich nur auf einen Abschnitt, den Abschnitt Files. Dieser Abschnitt ist in beiden Versionen vorhanden.

Die weiteren Abschnitte bezeichnen die jeweiligen Konfigurationen der verschiedenen Hardware und Bildschirmauflösungen. Der Vollständigkeit halber seien sie hier kurz aufgezählt:

- **Section "Module"**
  Angabe über ladbare Module.
- **Section "Files"**
  Pfadangaben zu Schriften, Farbinformationen und Modulen
- **Section "ServerFlags"**
  Angaben zu bestimmten Grundeinstellungen des X-Servers
- **Section "InputDevice"**
  Angaben zu allen denkbaren Eingabegeräten (Tastatur, Mouse, ...) Erst ab Version 4.0
- **Section "Keyboard"**
  Angaben zur Tastatur. Nur bei Versionen bis 3.x
- **Section "Pointer"**
  Angaben zur Mouse. Nur bei Versionen bis 3.x
- **Section "Monitor"**
  Angaben zum Monitor.
- **Section "Device"**
  Angaben zur Graphikkarte
- **Section "Screen"**
  Angaben zur Bildschirmauflösung und Farbtiefe

Optional können noch weitere Abschnitte enthalten sein, die sich auf Extras wie 3D-Beschleuniger (DRI) oder ähnliches beziehen.

Die Sektion "Files"

In diesem Abschnitt werden dem X-Server die Pfade zu drei verschiedenen Informationen übermittelt. Die Pfade zu den Verzeichnissen, in denen die Schriften liegen (FontPath), der Pfad zu der Datei, die die Farbbeschreibungen enthält (RgbPath) und optional der Pfad zu dem Verzeichniss, das die Module für den Server enthält (ModulePath). Diese Angabe muß nur gemacht werden, wenn der Pfad vom Standard abweicht.

Nachdem sowohl der ModulPath, als auch der RgpPath sehr statisch sind, werden wir an diesen beiden Einstellungen eigentlich niemals Veränderungen vornehmen müssen. Die einzigen Einstellungen, die wir tatsächlich verwalten und verändern müssen sind die Pfade zu den Schriftverzeichnissen.

Auf einem X-Server sind normalerweise mehrere Verzeichnisse mit Schriften installiert. Wir können diese Verzeichnisse entweder in einem Eintrag durch Kommas getrennt hintereinanderhängen, oder - was sicherlich zu einer besseren lesbarkeit führt - für jedes Verzeichnis einen eigenen Eintrag vornehmen. Also entweder

```
FontPath="/usr/X11R6/lib/X11/fonts/local/","/usr/X11R6/lib/X11/fonts/misc/"
```

oder

```
FontPath   "/usr/X11R6/lib/X11/fonts/local/"
FontPath   "/usr/X11R6/lib/X11/fonts/misc/"
```

In modernen Versionen wird nur noch die zweite Variante benutzt. Jede der Pfadangaben bezieht sich auf ein Verzeichnis, das Schriftdateien enthält. Die Reihenfolge dieser Angaben ist nicht unwichtig, weil der Server auf der Suche nach einer Schrift immer die Pfade in der angegebenen Reihenfolge durchsucht. Die erste gefundene Schrift, die den Anforderungen der Suche entspricht, wird verwendet. Wenn beispielsweise sowohl TrueType, als auch klassische X11-Schriften installiert sind, und es dabei zu einer Namensüberschneidung kommt (beispielsweise Times und Helvetica), dann kommt es darauf an, ob das Verzeichnis mit den TrueType Schriften vor oder nach dem mit den klassischen Schriften hier angegeben wurde.

Eine Veränderung der Datei XF86Config wird erst wirksam, wenn der X-Server neu gestartet wird. Es gibt aber zumindestens für FontPath-Änderungen auch die Möglichkeit, die neuen Einträge manuell zu aktivieren oder zu deaktivieren. Dazu wird das Programm **xset** benutzt:

```
xset +fp neuer_Pfad
xset fp+neuer_Pfad
```

fügen einen angegebenen neuen Pfad zum FontPath hinzu,

```
xset -fp Pfad
xset fp-Pfad
```

entfernen den angegebenen Pfad aus dem FontPath.

Das Prinzip der Schriftnamen von X11

Schriften können bei X11, wie bei fast jeder anderen graphischen Oberfläche auch, beliebig hinzugefügt oder entfernt werden. Dabei existieren mehrere mögliche Formate, so daß es also möglich ist, daß ein und dieselbe Schrift in einer Installation mehrfach vorhanden ist.

Schriftnamen haben eine etwas gewöhnungsbedürftige Form, hier einmal ein Beispiel, an dem die einzelnen Elemente des Namens erklärt sind.

```
 Hersteller   Gewicht Breite     Pixels  XAufl Sp     Zeichensatz
    |            |        |         |      |     |         |
  -b&h-lucida-medium-r-normal-sans-18-180-75-75-p-106-iso8859-1
        |           |          |      |     |     |          |
     Schriftart    Neigung    Stil   Punkte YAufl Durchschn. Optionen des
                                                 Zeichen-   Zeichensatzes
                                                 breite
```

Dabei bedeuten im Einzelnen:

Hersteller

Meist die Herstellerfirma der Schriftart, z.B. adobe, dec, sony, manchmal aber auch nur eine weitere Schublade wie misc oder bitstream.

Schriftart

Die "Schriftfamilie" also etwa Times, Symbol, Utopia,...

Gewicht

Das "Gewicht" der Schwärze wie black, bold, demibold oder medium.

Neigung

Die Schräge der Schrift. i steht für italic (kursiv), r für roman (aufrecht).

Breite

Die Breite einer Schrift wie normal, narrow oder semicondensed.

Stil

Zusätzliche Stilangaben

Pixels

Das Höhenmaß der Schrift in Pixel (in einer bestimmten Punktgröße und Auflösung)

Punkte

Die Größe der Schrift in typographischen Punkten (1/72 Zoll) angegeben in Zehntel-Punkten

X-Auflösung

Horizontale Auflösung in Dots per Inch (DPI) in der die Schrift ursprünglich erstellt wurde.

Y-Auflösung

Vertikale Auflösung in Dots per Inch (DPI) in der die Schrift ursprünglich erstellt wurde.

Spacing

Die Beschreibung der Raumaufteilung einer Schriftart. Hier geht es darum, ob eine Schriftart proportional aufgebaut ist, also für ein i einen kleineren Platz als z.B. für ein o benötigt, oder nicht. Mögliche Werte sind hier p für proportional, m für monospaced also nicht proportional und c für charactercell, einer Technik, in der allen Zeichen ein gleichgroßer Rahmen gegeben wird, was sie auch wieder zur nicht-proportionalen Schrift macht.

Durchschnittsbreite

Die durchschnittliche Breite eines Zeichens dieser Schrift.

Zeichensatz

Die ISO-Standard, die diese Schrift enthält.

Die Nummer der ausgewählten Zeichensatzseite des Standards.

Oft werden statt einzelner Angaben auch ein Sternchen angegeben, dann wird die erste passende Einstellung genommen. So wird ein * bei der Angabe der Neigung automatisch zu einem i.

Verwaltung von Schriften unter X11

Wenn neue Schriften installiert werden sollen, dann werden ein paar Einstellungen nötig, damit sie benutzbar werden.

Verzeichnisse, die Schriften für X11 enthalten, enthalten neben den eigentlichen Schriftdateien (Endungen .bdf .snf .pcf .spd .fb .ps .ttf) noch die folgenden Dateien:

fonts.dir

Diese Datei enthält die Angaben, welche Schrift in welcher Datei zu finden ist. Am Anfang der Datei steht eine Nummer, die Anzahl der im Verzeichnis enthaltenen Schriften. Die weiteren Zeilen der Datei enthalten paarweise die Dateinamen mit zugehörigem Schriftnamen, also etwa

```
courBO10.snf  -adobe-courier-bold-o-normal--10-100-75-75-m-60-iso8859-1
```

Damit diese Datei immer auf dem neuesten Stand ist, muß nach jeder Neuinstallation von Schriftdateien in einem Verzeichnis in diesem Verzeichnis das Programm **mkfontdir** aufgerufen werden. Alternativ kann auch der Name des Verzeichnisses dessen fonts.dir Datei neu erstellt werden soll, als Parameter von **mkfontdir** angegeben werden.

fonts.scale

Neben den Bitmap Fonts existieren auch noch skalierbare Schriften. In den Verzeichnissen mit solchen skalierbaren Schriften (z.B. speedo) existiert neben der fonts.dir auch die Datei fonts.scale. Sie enthält je einen Eintrag pro skalierbarer Schrift, mit den Größenangaben jeweils auf 0 gesetzt.

fonts.alias

Diese Datei enthält beliebige Alias-Namen, mit den dazugehörigen Schriftnamen. So kann etwa mit der Zeile

```
school15 -adobe-new century schoolbook-bold-r-normal-*-15-150-75-*-*-105-iso8859-1
```

die Schrift -adobe-new ... schlicht mit school15 angesprochen werden. Dieser Mechanismus hat aber Vor- und Nachteile. Es ist zwar so für einen User einfacher, eine Schrift auszuwählen, es kann aber bei einer Anwendung, die auf verschiedenen Rechnern läuft fatal sein, wenn es die Schrift bzw. eben das Alibi auf einem anderen Rechner nicht gibt.

Wenn ein neues Verzeichnis mit Schriften also hinzugefügt wird, indem es beispielsweise neu erstellt und mit Schriftdateien aus dem Internet gefüllt wird, so sind folgende Schritte nötig, damit die Schriften benutzbar werden.

- Ausführen des Programmes **mkfontdir** *Verzeichnis*
- Verzeichnis in den FontPath aufnehmen (indem ein Eintrag in `/etc/X11/XF86Config` vorgenommen wird - siehe oben)
- Neustart des X-Servers oder manuelle Aufnahme des Verzeichnisses mit **xset** +fp *Verzeichnis*.

Ab diesem Zeitpunkt sind die Schriften auf dem X-Server verfügbar.

Installation eines FontServers

Das Schriftenmodell unter X11 erfordert relativ viel Speicherplatz, weil bei den meisten installierten Schriftarten jede Schrift eine eigene Schriftdatei benutzt. Aus diesem Grund gibt es die Möglichkeit, einen Fontserver zu installieren, der beliebig vielen X-Servern im Netz Schriften zur Verfügung stellt. So müssen die Schriftdateien nur auf einem Rechner vorhanden sein und stehen trotzdem allen X-Servern im Netz zur Verfügung.

Ein weiterer Vorteil dieser Technik ist es, daß ein Unternehmen bestimmte Schriften für seine Präsentationen benutzen kann und zentral festlegen kann, daß alle X-Server die selben Schriften benutzen.

Für die Ansprüche der LPI101 Prüfung muß sowohl ein Fontserver eingerichtet werden können, als auch ein X-Server an einen Fontserver angebunden werden, so daß er die vom Server angebotenen Schriften nutzen kann.

Konfiguration eines Fontservers

Der Fontserver für X11 heißt **xfs** und ist ein eigenständiger Daemon. Er wird in der Regel über ein init-Script gestartet und bietet dann seine Dienste an.

Wie so oft bei Linux/Unix-Servern besitzt **xfs** eine einzige Konfigurationsdatei, in der die gesamte Konfiguration des Servers erledigt wird. Diese Konfigurationsdatei liegt standardmäßig unter `/usr/X11R6/lib/X11/fs/config`. Da diese Plazierung dem neuen Dateisystem-Standard widerspricht, kann alternativ dazu dem **xfs** per Kommandozeilenoption eine andere Datei mitgeteilt werden, die er als Konfigurationsdatei nutzt. In modernen Versionen von Linux wird meist `/etc/X11/xfs.conf` oder `/etc/X11/fs/config` benutzt.

Der Fontserver muß die Schriftdateien in Verzeichnissen ablegen, die exakt den oben genannten Ansprüchen der Schriftverzeichnisse eines normalen X-Servers entsprechen. Er kann jetzt diese Verzeichnisse an andere X-Server weitergeben, ohne daß dabei zusätzlicher Verwaltungsaufwand entsteht.

Eine einfache Konfigurationsdatei eines Fontservers könnte folgendermaßen aussehen:

```
# Erlaubt maximal 10 Clients, die sich mit diesem Fontserver verbinden
client-limit = 10

#
# Wenn ein Fontserver die maximale Anzahl Clients erreicht hat,
# startet er einen neuen um weitere Clients bedienen zu können
clone-self = on

#
# Angabe der Schriftverzeichnisse
#
catalogue = /usr/local/fonts/tr_misc/,
            /usr/local/fonts/tr_75dpi/,
            /usr/local/fonts/freefont/,
            /usr/local/fonts/sharefont/,
            /usr/local/fonts/Adobe/,
            /usr/X11R6/lib/X11/fonts/misc:unscaled,
            /usr/X11R6/lib/X11/fonts/75dpi:unscaled,
```

```
                /usr/X11R6/lib/X11/fonts/100dpi:unscaled,
                /usr/X11R6/lib/X11/fonts/Type1,
                /usr/X11R6/lib/X11/fonts/Speedo,
                /usr/X11R6/lib/X11/fonts/misc,
                /usr/X11R6/lib/X11/fonts/75dpi


  # Voreingestellte Schriftgröße ist 12 Punkte (120 Dezipunkte)
  default-point-size = 120

  # 100 x 100 und 75 x 75 Auflösung
  default-resolutions = 100,100,75,75

  # Keine Meldungen an den Syslogdaemon
  use-syslog = off
```

Die Kommentare in der Datei erklären die jeweiligen Zeilen ausreichend. Wenn diese Datei jetzt als `/etc/X11/fs/config` abgespeichert wird, dann lautet der Aufruf des XFontServers:

```
  xfs -config /etc/X11/fs/config -daemon
```

Diese Zeile sollte jetzt aus einem init-Script heraus aufgerufen werden. Nachdem sie ausgeführt wurde, ist der Fontserver aktiv und hört auf den TCP-Port 7100.

Soll einem laufenden Fontserver ein neues Schriftverzeichnis hinzugefügt werden, so muß es in die Konfigurationsdatei eingefügt werden und danach dem **xfs**-Prozeß ein HUP-Signal geschickt werden. Das zwingt ihn, seine Konfigurationsdatei neu einzulesen und sich entsprechend zu verhalten.

Anbindung eines X-Servers an einen Fontserver

Damit ein beliebiger X-Server jetzt den Fontserver benutzen kann, reicht es, in der Datei XF86Config in der Section "Files" einen FontPath Eintrag in der Art

Transportprotokoll/*Fontservername*:*Portnummer*

zu machen. Wenn unser Font-Server auf dem Rechner **marvin** auf dem Standard-TCP-Port 7100 läuft, so kann jeder X-Server im Netz den Eintrag

```
  FontPath   "TCP/marvin:7100"
```

in die Section "Files" aufnehmen und nach einem Neustart des X-Servers (nicht des Rechners) stehen die Schriften des Font-Servers zur Verfügung. Auch hier spielt die Reihenfolge der Angaben eine Rolle. Der FontPath wird von vorne nach hinten (oben nach unten) durchgearbeitet und die erste passende Schrift wird gewählt. Steht also der Font-Server Eintrag am Anfang des FontPath, so werden wahrscheinlich niemals lokale Schriften benutzt (es sei denn es gibt lokale Schriften, die der FontServer nicht anbietet), steht er am Ende, werden immer lokale Schriften verwendet, außer es gibt eine bestimmte Schrift nicht lokal, dann wird der Font-Server herangezogen.

Verwendung von Schriften

Die in den Prüfungszielen angesprochene Datei `.Xresources` kann benutzt werden, um bestimmte Schriften für bestimmte Anwendungen fest einzustellen. Diese Technik wird im Abschnitt 1.110.4 - Installation und Anpassung einer Window Manager Umgebung genau besprochen.

1.110.2

Einrichten eines Display Managers

Beschreibung: Prüfungskandidaten sollten in der Lage sein, einen Display Manager einzurichten und anzupassen. Dieses Lernziel beinhaltet das Aktivieren und Deaktivieren des Display Managers und das Ändern der Willkommensmeldung. Ebenfalls enthalten ist das Ändern der voreingestellten Bitplanes des Display Managers. Weiters enthalten ist die Konfiguration des Display Managers für die Verwendung auf X-Stationen. Das Lernziel deckt die Display Manager XDM (X Display Manager), GDM (Gnomde Display Manager) und KDM (KDE Display Manager) ab.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/inittab`
- `/etc/X11/xdm/*`
- `/etc/X11/kdm/*`
- `/etc/X11/gdm/*`

Startmöglichkeiten des X-Servers

Es existieren grundsätzlich zwei Möglichkeiten, einen X-Server zu starten. Er kann entweder als Anwenderprogramm gestartet werden, nachdem sich der aufrufende User normal eingeloggt hat, oder er stellt bereits den Login graphisch zur Verfügung.

Die erste Methode wird gewöhnlich dadurch erreicht, daß der Befehl **startx** aufgerufen wird. **startx** ist ein Shellscript, das den Befehl **xinit** aufruft. Dieses Programm startet dann sowohl den X-Server, als auch zusätzliche Programme, wie den Window-Manager oder die Desktop Umgebung. Die Konfiguration von **xinit** wird über die Datei `$HOME/.xinitrc` oder falls diese Datei nicht existiert über `/var/X11R6/lib/xinit/xinitrc` konfiguriert. Der X-Server wird also als normale Anwendung gestartet. Dieses System hat erhebliche Sicherheitslücken, weil "hinter" dem X-Server eine offene Konsole liegt, von der aus der Befehl **startx** eingegeben wurde.

Um diese Sicherheitslücke zu schließen, kann die gesamte Kontrolle des Einloggens auch an X11 weitergegeben werden. In diesem Fall wird der X-Server nicht von einem Anwender per Befehl gestartet, sondern von einem Daemon-Prozess, dem **X-Display-Manager**. Dieser

Daemon läuft ständig im Hintergrund und stellt für jeden angeschlossenen X-Server (in der Regel nur einer) ein graphisches Login-Fenster zur Verfügung. Wenn sich ein User über dieses Fenster anmeldet, dann startet der X-Display-Manager gleich den X-Server, der User bekommt also sofort eine graphische Oberfläche, ohne überhaupt auf einem Textterminal eingeloggt zu sein.

Der Display-Manager ist ein Daemon-Prozeß, er sollte also über ein init-Script gestartet werden. Ist der Display-Manager aktiv, so ist es - zumindestens ohne Tricks - nicht mehr möglich, über ein Textterminal mit **startx** den X-Server als normale Anwendung zu starten. Aus diesem Grund wird der Display-Manager in der Regel nur in einem ganz bestimmten Runlevel gestartet.

Diese Einschränkung erlaubt es, den Display-Manager beliebig an- und auszuschalten, indem einfach der Runlevel entsprechend verändert wird. Die meisten Distributionen haben eine entsprechende Voreinstellung, also einen Runlevel, der alle Netzwerkdienste startet und zusätzlich den Display-Manager aktiviert und einen anderen, der alle Netzwerkdienste startet aber keinen Display-Manager.

Welche Runlevel dazu verwendet werden ist nicht standardisiert. Hier ein paar Beispiele für wichtige Distributionen:

Distribution	**Runlevel ohne DM**	**Runlevel mit DM**
SuSE bis 7.2	3	5
SuSE ab 7.3	2	3
RedHat	3	5
Caldera	3	5
Slackware	3	4

Nur bei Debian gibt es keinen solchen Runlevel voreingestellt, da die Debian Distribution davon ausgeht, daß sobald der Display-Manager installiert ist, auch sein Einsatz gewünscht wird.

In der Regel sollte in der Datei `/etc/inittab` eine Liste der zur Verfügung stehenden Runlevel als Kommentare stehen, und danach die Einstellung, welcher Runlevel beim Systemstart verwendet werden soll (*initdefault*). Beispielsweise könnte hier stehen:

```
...
#
# runlevel 0  is  System halt   (Do never use this for initdefault)
# runlevel 1  is  Single user mode
# runlevel 2  is  Local multiuser without remote network (e.g. NFS)
# runlevel 3  is  Full multiuser with network
# runlevel 4  is  Not used
# runlevel 5  is  Full multiuser with network and xdm
# runlevel 6  is  System reboot (Do never use this for initdefault)
```

```
#

id:5:initdefault:
...
```

Dem wäre also zu entnehmen, daß der Runlevel 5 der Runlevel ist, in dem der Display-Manager (xdm) gestartet werden soll und daß dieser Runlevel auch der voreingestellte Runlevel ist.

Um einen Linux-Rechner also zu zwingen, den Display-Manager zu benutzen, muß der Eintrag in `/etc/inittab` entsprechend auf den Runlevel geändert werden, der für den Display-Manager Runlevel steht. Um Linux zu zwingen, den Display-Manager **nicht** zu benutzen, so muß entsprechend der Eintrag geändert werden, so daß in unserem obigen Beispiel die Zeile jetzt lauten müßte:

```
...
id:3:initdefault:
...
```

Der Display-Manager-Runlevel sollte erst aktiviert werden, wenn sichergestellt ist, daß die Konfiguration des X-Servers tatsächlich funktioniert. Ansonsten geht man Gefahr, daß sich das System nicht mehr vernünftig bedienen läßt, weil beim Systemstart gleich die graphische Oberfläche geladen wird und die womöglich nur einen flackernden Bildschirm produziert.

Verschiedene Display-Manager und ihre Konfiguration

Der klassische Display-Manager des XFree86-Projekts war und ist der **xdm**. Dieser Display-Manager stellt ein relativ anspruchsloses Login-Fenster zur Verfügung, das nur je eine Eingabezeile für Username und Passwort anbietet.

Sowohl das KDE-, als auch das Gnome-Projekt haben jeweils einen eigenen Display-Manager im Angebot. Der Displaymanager von KDE heißt **kdm**, der von Gnome ist **gdm**. Der primäre Unterschied zum **xdm** liegt im entsprechenden Look&Feel. Allerdings bieten **gdm** und **kdm** noch einige interessante Zusatzfeatures:

- User werden als Bilder (evt. Passfotos) dargestellt und können angeklickt werden, statt den Usernamen direkt einzugeben.
- Der gewünschte Window-Manager der zu startenden X-Session kann per Menü ausgewählt werden.
- Das System kann heruntergefahren oder neu gestartet werden, ohne sich vorher einloggen zu müssen.

Der **kdm** basiert sehr stark auf dem **xdm**, das heißt, er nutzt sowohl XDM-Konfigurationsdateien, als auch Teile des Quellcodes von **xdm** Im Gegensatz dazu ist der **gdm** komplett neu geschrieben und daher etwas unabhängiger von **xdm**.

Die Konfiguration der jeweiligen Displaymanager findet nach dem neuen Dateisystemstandard immer in den folgenden Verzeichnissen

statt:

Displaymanager	Konfigurationsverzeichnis
xdm	`/etc/X11/xdm`
gdm	`/etc/X11/gdm`
kdm	`/etc/X11/kdm`

In diesen Verzeichnissen liegen unterschiedliche Konfigurationsdateien. Die Verzeichnisse von **kdm** und **xdm** ähneln sich sehr stark, was auf ihre Verwandschaft zurückzuführen ist, das Verzeichnis von **gdm** ist vollkommen anders aufgebaut.

Konfiguration von xdm

Der optische Aufbau von **xdm** ist sehr simpel gehalten. Ein Login-Fenster enthält eventuell ein Logo und eine einfache Willkommensmeldung. Die einzige Einstellungsmöglichkeit außer der des Logos und der Willkommensmeldung bezieht sich auf den Hintergrund des Bildschirms, auf dem das Login-Fenster erscheint.

xdm ist ein typisches X11-Programm, wird also durch Resources konfiguriert. Die Einstellung der von **xdm** benötigten Resources liegt in der Datei `/etc/X11/xdm/Xresources`.

In dieser Datei können die folgenden Einstellungen getroffen werden:

- Die Willkommensmeldung kann mittels der Ressource `xlogin*greeting` verändert werden. Ein eventuelles Vorkommen des Wortes `CLIENTHOST` wird in der Ausgabe durch den Hostnamen des Rechners ersetzt.
- Die Schriftart der Willkommensmeldung wird über die Ressource `xlogin*greetFont` eingestellt.
- Das darzustellende Logo wird über die Ressource `xlogin*logoFileName` eingestellt. Das Dateiformat des Logos sollte XPM sein.
- Die Farben des Login-Fensters und seine Rahmenbreite können über Ressourcen wie `xlogin*borderWidth, xlogin*frameWidth, xlogin*background, xlogin*foreground` und `xlogin*greetColor` angepasst werden.

Um schließlich den Bildschirmhintergrund zu verändern, kann eine weitere Datei verwendet werden. Im Verzeichnis `/etc/X11/xdm` liegt ein Shellscript `Xsetup`. Dieses Script wird jedesmal abgearbeitet, wenn xdm eine Loginmeldung auf einen Bildschirm schreibt. In dieses Script können jetzt schon Befehle eingetragen werden, die den Bildschirmhintergrund verändern, wie etwa **xpmroot**, **xsetbg** oder jedes beliebige andere Programm, das den Bildschirmhintergrund modifiziert.

Nehmen wir an, im Verzeichnis `/etc/X11/xdm` befindet sich eine Datei `background.xpm`. Diese Datei enthält das gewünschte Hintergrundbild. Dann tragen wir in die Datei `/etc/X11/xdm/Xsetup` die folgende Zeile ein:

```
/usr/bin/xpmroot /etc/X11/xdm/background.xpm
```

Damit wird das gewünschte Bild jedesmal auf den Hintergrund dargestellt, wenn **xdm** ein Loginfenster darstellt.

Konfiguration von kdm

Im Prinzip funktionieren grundsätzlich alle Einstellungen, die für **xdm** gezeigt wurden auch für **kdm**. Der einzige Unterschied ist der, daß die Einstellungen jetzt in `/etc/X11/kdm` vorgenommen werden müssen.

Allerdings gibt es eine zweite Konfigurationsmöglichkeit für **kdm**, die wesentlich mehr Einstellungen erlaubt, als die Möglichkeiten, die wir bereits besprochen haben. Normalerweise werden diese zusätzlichen Einstellungen über ein graphisches Programm erledigt, das Teil des K-Control-Centers ist. Unter dem Menüpunkt *System/Anmeldungsmanager* können die verschiedensten Einstellungen vorgenommen werden. Alle Einstellungen, die über diese Methode vorgenommen wurden, werden in der Datei `/etc/X11/kdm/kdmrc` gespeichert und können dort auch manuell verändert werden.

Einer der wesentlichen Unterschiede zwischen **kdm** und **xdm** ist die Darstellung der User in Bildform zum Auswählen. Auch die Frage, welche Bilder verwendet werden sollen, kann mit dem k-Control-Center eingestellt werden. Allerdings kann auch einfach je ein Bild (im Format XPM) für jeden User in das Verzeichnis *kde-Wurzel*`/apps/kdm/pics/users` gelegt werden, dessen Namen dem des Users entspricht. Das Bild `hans.xpm` würde also für den User **hans** benützt.

Konfiguration von gdm

Der Display-Manager des Gnome-Projektes arbeitet nicht mit dem Quellcode von **xdm**, daher ist seine Konfiguration auch entsprechend anders, als die der beiden anderen Displaymanager. Die Konfiguration wird grundsätzlich mit dem Programm **gdmconfig** vorgenommen. Es handelt sich auch um ein graphisches Programm mit entsprechenden Einstellungsmöglichkeiten.

Die Einstellungen dieses Programmes werden in der Datei `/etc/X11/gdm/gdm.conf` abgespeichert und können hier auch manuell verändert werden. In der Regel wird das aber nie der Fall sein, da die graphische Konfiguration wesentlich konfortabler ist.

Damit auch der **gdm** die Bilder der User darstellt, muß dieses Feature ersteinmal aktiviert werden (über **gdmconfig**). **gdm** bietet jetzt aber die Möglichkeit, daß jeder User sein eigenes Bild einstellt. Dazu kann er das Programm **gdmphotosetup** benutzen.

XTerminals mit dem Displaymanager bedienen

Die Möglichkeiten des graphischen Logins beschränken sich nicht alleine auf den lokalen Rechner. Es ist auch möglich, daß sich User fremder Rechner auf unserem Rechner graphisch einloggen. Dazu stellt das X-Protokoll ein spezielles Netzwerkprotokoll zur Verfügung, XDMCP (X-DisplayManager Control Protocol). Damit ist es möglich, daß ein fremder X-Server das Bild unseres Displaymanagers auf seinem Bildschirm zu sehen bekommt und sich dann graphisch auf unserem Rechner einloggt. Alle Programme - außer dem X-Server selbst - die während dieser Session auf dem fremden Rechner dargestellt werden, laufen nicht mehr auf seiner CPU sondern auf der des Servers, auf dem er sich eingeloggt hat. So kann z.B. auch noch ein alter 486er als graphischer Arbeitsplatz genutzt werden.

Damit diese Möglichkeit besteht, muß der Displaymanager zunächst einmal so konfiguriert werden, daß er Zugriffe von fremden Rechnern auch zuläßt. Das geschieht bei **xdm** und **kdm** wieder exakt gleich, nur der **gdm** hat wiederum seine eigenen Einstellmöglichkeiten.

XDMCP aktivieren

Bei **xdm** und **kdm** müssen zunächst einmal die grundsätzliche Bereitschaft des Display-Managers aktiviert werden, XDMCP-Pakete anzunehmen. Das geschieht in der Datei `xdm-config` bzw. `kdm-config` im jeweiligen Verzeichnis des Display-Managers. Dort steht - meist in der letzten Zeile - der Eintrag

```
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with kdm
DisplayManager.requestPort:     0
```

Die Zeile, die den `requestPort` auf 0 setzt muß auskommentiert werden, wenn ein Login von fremden Rechnern aus möglich sein soll. In beiden Fällen wird durch ein Ausrufungszeichen diese Zeile deaktiviert, also:

```
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with kdm
! DisplayManager.requestPort:     0
```

Damit ist die grundsätzliche Bereitschaft zur Bedienung aktiviert.

Um den selben Schritt mit **gdm** zu vollziehen, muß über das Programm **gdmconfig** im Submenü *Expert* im Einstellungstab *XDMCP* das Schaltkästchen *XDMCP aktivieren* angeschaltet werden. Dadurch wird der Eintrag `Enable=true` in der Sektion `[xdmcp]` der Datei `/etc/X11/gdm/gdm.conf` vorgenommen.

Zugriffe erlauben

Für **xdm** und **kdm** existiert jetzt noch die Notwendigkeit, einzustellen, welche Rechner das Recht haben sollen, sich graphisch

einzuloggen. Dazu existiert die Datei `Xaccess` im jeweiligen Verzeichnis des Displaymanagers. Dort kann festgelegt werden, welcher Rechner

- ein Login-Fenster bekommen soll
- ein Auswahlfenster weiterer Server, die xdm-Logins anbieten, bekommen soll

Dazu werden in dieser Datei Einträge nach dem Muster

Hostnamensmuster

für Direct und Broadcast Zugriffe gemacht. Der einfachste Eintrag wäre also eine Zeile mit nur einem Sternchen (*), der Zugriff für alle Rechner erlaubt.

Den Zugriff auf die Auswahl anderer Rechner wird mit

Hostnamensmuster CHOOSER BROADCAST

geregelt.

Eine solche Datei, die also allen Rechnern der Domaine `foo.bar` die Benutzung unseres graphischen Logins erlauben soll könnte folgendermaßen aussehen:

```
*.foo.bar
*.foo.bar       CHOOSER BROADCAST
```

Clientseitige Einstellungen

Normalerweise - wenn der X-Server entweder über xinit (startx) oder xdm gestartet wird, wird nach dem Start des X-Servers sofort noch ein Fenster-Manager (fvwm2, kwm, gwm, ...) gestartet, der dann wiederum die Oberfläche konfiguriert. Das ist anders, wenn wir uns auf einem fremden System graphisch einloggen wollen.

Zu diesem Zweck wird der X-Server tatsächlich direkt aufgerufen, also weder durch **startx**, noch durch **xdm** sondern durch den Befehl **X**. Dazu kommen drei Techniken in Betracht. DIRECT, INDIRECT und BROADCAST.

Die DIRECT-Methode spricht direkt einen Rechner im Netz an um sich bei ihm einzuloggen. Die Systax dazu ist

```
X :0.0 -query Hostname
```

Jetzt wird der Server 0 gestartet, seine gesammte Konfiguration übernimmt aber der **xdm** des Rechners *Hostname*. Der Rechner, auf dem wir diesen Befehl gestartet haben dient also nur als X-Terminal für den unter *Hostname* angegebenen Rechner.

Die zweite Methode (INDIRECT) geht einen anderen Weg. Sie frägt einen Rechner nicht nach dem Login-Fenster, sondern nach einer

Liste von bekannten Servern, die den xdm-Dienst (aso den Dienst sich graphisch einloggen zu können) anbieten. Der gefragte Rechner startet jetzt diese Auswahlliste (den sogenannten CHOOSER) und der aufrufende Rechner kann sich aus dieser Liste aussuchen, wo er sich einloggen will. Um diese Methode anzuwenden lautet der Befehl:

```
X :0.0 -indirect Hostname
```

Die dritte Methode ist schließlich ein Broadcast, der ins lokale Netz ausgesandt wird, um einen Rechner oder CHOOSER zu bekommen. In diesem Fall wird kein Rechnername angegeben, sondern einfach nur ein

```
X :0.0 -broadcast
```

Statt dem ersten zu startenden Server kann bei jedem der drei Methoden auch ein zweiter, dritter, usw gestartet werden, was dann statt der Angabe `:0.0` entsprechend die Angaben `:1.0, :2.0` usw. erfordert.

[Zurück zur LPIC 101 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.110.4

Installation und Anpassung einer Window Manager Umgebung

Beschreibung: Prüfungskandidaten sollten in der Lage sein, eine systemweite Desktopumgebung und/oder einen Window Manager einzurichten und ein Verständnis der Anpassungsprozedur für Menüs der Fenstermanager und oder der Desktop-Panele demonstrieren. Dieses Lernziel beinhaltet die Auswahl und Konfiguration des gewünschten X-Terminals (xterm, rxvt, aterm etc.), das Prüfen und Auflösen von Bibliotheksabhängigkeiten von X-Anwendungen und das Exportieren der Ausgabe von X-Anwendungen auf einen Client.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `.xinitrc`
- `.Xdefaults`
- **xhost**
- `DISPLAY` Umgebungsvariable

Window-Manager

Der X-Server selbst enthält noch keinerlei Mechanismen zum Bewegen, zur Größenveränderung oder zum "Iconifizieren" von Fenstern. Auch die Ausstattung von Fenstern mit Rahmen findet noch nicht statt, wenn nur der Server läuft. All diese Aufgaben übernehmen Fenster-Manager (Windowmanager) wie der Motif Window Manager (mwm), der Open-Look Window Manager (olwm) oder fvwm. Auch die KDE-Desktop-Umgebung bringt einen eigenen Window-Manager mit, der kwm, während die Gnome-Desktop-Umgebung mit vielen Window-Managern zusammenarbeitet und keinen eigenen besitzt.

Erst durch das Starten eines Window-Managers wird der X-Server zur arbeitsfähigen Graphikstation. Erst der Window-Manager erlaubt es dem User, Fenster zu verschieben, in der Größe zu verändern oder in ein Icon zu verwandeln. Er definiert meist auch Menüs, die durch Drücken der verschiedenen Maustasten aktiviert werden können (wenn sich der Mauszeiger nicht innerhalb eines Fensters befindet).

Die Konfiguration der einzelnen Window-Manager ist so unterschiedlich, daß sich keinerlei allgemeine Regeln dazu formulieren lassen außer der, daß fast immer eine persönliche Konfigurationsdatei im jeweiligen Home-Verzeichnis eines Users befindet, die immer mit einem Punkt beginnt und auf rc endet. dazwischen steht meist der Name des Window-Managers. Also hat Motif eine `.mwmrc`-Datei, olwm wird in `.olwmrc` konfiguriert und fvwm2 in `.fvwm2rc`.

Die beiden Desktop-Umgebungen **KDE** und **Gnome** bieten zusätzlich zu den Features der Window-Manager noch eine vereinheitlichte Darstellung der jeweils verwendeten Programme und eine Schnittstelle für Programmierer, um ihre Programme entsprechend für diese Umgebungen auszustatten. Auch Features wie Drag&Drop werden dabei unterstützt.

Diese beiden Spezialfälle, die heute wohl eher als die Regel statt die Ausnahme zu bezeichnen sind, bieten vor allem eine Möglichkeit, ihre Konfiguration auf eine Art und Weise vornehmen zu lassen, wie sie die BenutzerInnen von Windows gewohnt sind. Kontextmenüs über die rechte Maustaste und Controlcenter in denen die gesamte Konfiguration graphisch vorgenommen werden kann, gehören zur Grundausstattung.

Konfiguration von X-Clients

Weil X-Clients (also X-Anwendungsprogramme) auf allen X-Servern in einem Netz dargestellt werden können, muß jede X-Anwendung konfigurierbar sein. So portierbar das X-System auch ist, so unterschiedlich sind doch einige Voraussetzungen auf unterschiedlichen Servern. Bildschirme können unterschiedliche Auflösungen haben, Graphikkarten unterschiedliche Farbtiefen. Ein Programm, das auf einem Server gut dargestellt wird kann auf einem anderen nicht auf den Schirm passen oder farblich falsch dargestellt werden. Eine Taste, die auf einer Tastatur leicht erreichbar ist kann auf einer anderen dazu führen, daß man sich fast den Finger bricht...

Auf anderen Fenstersystemen wie MS-Windows oder dem Macintosh-Finder werden alle Anwendungen auf der Anwendungsebene konfiguriert. Das macht Sinn, weil sowohl DOS als auch das Mac-OS Single-User-Systeme sind. Alle Konfiguration kann zentral gespeichert werden.

X11-Programme müssen anderen Anforderungen standhalten. Erstens ist Unix ein Mehrbenutzersystem und es ist daher nötig, daß alle User ihre individuellen Einstellungen machen können, zweitens weiß das X11-Programm noch nicht, auf welchem Server es dargestellt werden soll, es muß also auch in dieser Hinsicht konfigurierbar sein.

Es stehen bei allen X11-Clients zwei Konfigurationsmöglichkeiten zur Verfügung, die Kommandozeilenparameter und die Resourcen.

Konfiguration von X-Clients über Kommandozeilenparameter

Anhand des Clients **xterm** kann schön dargestellt werden, wie die klassischen Kommandozeilenparameter eines X-Clients angewandt werden. All diese Parameter werden von allen X-Clients verstanden, **xterm** ist hier also nur ein Beispiel.

Schriftenauswahl

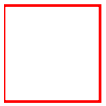
Die Auswahl von Schriften erfolgt mit dem Kommandozeilenparameter -fn. Um ein xterm zu starten, das die Schrift `-misc-fixed-bold-r-normal--13-100-100-100-c-70-iso8859-1` benutzen soll reicht die Angabe

```
xterm -fn -misc-fixed-bold-r-normal--13-100-100-100-c-70-iso8859-1 &
```

Damit ist ein xterm mit anderer Schrift gestartet. Vorsicht ist übrigens angesagt bei der Verwendung von anderen Schriftfamilien. In der Regel ist das kein Problem, aber ein xterm-Fenster simuliert eben ein Terminal. Werden hier Proportionalschriften verwendet, die für ein i weniger Platz als für ein o benutzen, so kann es zu heftigen Problemen beim Arbeiten mit bildschirmorientierten Programmen kommen. Eine Aufteilung in untereinanderliegende Spalten ist ja dann nicht mehr möglich. Xterm sollte also immer mit Schriften der Familie fixed benutzt werden.

Fenstergeometrie

X11 verwaltet den Bildschirm als ein Koordinatensystem. Die obere linke Ecke entspricht der Position 0,0. Die unteren Ecken werden durch die maximale Auflösung bestimmt, genau wie die obere linke Ecke. Bei einer Auflösung von 1024x768 sieht das beispielsweise so aus:



Ein X-Client kann mit dem Parameter `-geometry` auf zweierlei Arten konfiguriert werden. Die Größe und die Position des Fensters auf dem Schirm kann angegeben werden. Zunächst zur Größe:

Wir wollen ein xterm-Fenster, das statt der Standardgröße 80 Spalten in 24 Zeilen jetzt 90 Spalten in 30 Zeilen besitzt. Kein Problem:

```
xterm -geometry 90x30 &
```

Nach dem `-geometry` folgt also die Angabe der Spalten und Zeilen durch ein x getrennt. (Wem das geometry zu lang ist, der kann beruhigt sein - in den meisten Fällen funktioniert auch ein -g statt dessen).

Achtung: Die Angabe von Zeilen und Spalten bezog sich beim xterm auf Zeichen statt auf Pixel. Graphikorientierte Programme rechnen hier mit Pixel, zeichenorientierte Programme mit Zeichen. Um also etwa ein Programm wie xeyes (die wohlbekannten Augen) in der Größe zu verändern, muß mit Pixeln gerechnet werden.

Die andere Aufgabe, die mit `-geometry` erfüllt werden kann ist die Positionierung eines Fensters auf dem Schirm. Dazu wird der Offset zum Rand des Schirms angegeben, ein Pluszeichen meint den oberen bzw. den linken Rand, ein Minuszeichen den unteren bzw. rechten:

```
xterm -geometry +10+50 &
```

positioniert die obere linke Ecke des Fensters also 10 Pixel vom linken Rand entfernt und 50 Pixel vom oberen Rand. Bei dieser Angabe sind übrigens immer Pixel die Maßangabe, nie Zeichen. Um sich auf die untere linke Ecke zu beziehen können wir schreiben:

```
xterm -geometry -30-100 &
```

Wir erhalten ein Fenster, dessen untere rechte Ecke 30 Pixel vom rechten Rand des Bildschirms und 100 vom unteren Rand entfernt ist. Es ist sogar möglich, diese Angaben zu mischen. Die Zeile

```
xterm -geometry +10-100 &
```

meint also ein Fenster, dessen linker Rand 10 Pixel vom linken Rand des Bildschirms entfernt ist und dessen untere Begrenzung 100 Pixel Abstand zum unteren Rand aufweist.

Die Angaben von Größe und Position lassen sich auch kombinieren. Dabei wird immer die Größe zuerst angegeben, danach (ohne Leerzeichen aber mit Vorzeichen + oder -) die Positionsangaben. Um etwa ein xterm mit 90 Spalten in 30 Zeilen an der Position 10,100 zu erhalten schreiben wir:

```
xterm -geometry 90x30+10+100 &
```

Wird keine Angabe zur Position gemacht, so übernimmt der Window-Manager die Positionierung. Dabei kann meist eingestellt werden, auf welche Art und Weise diese Positionierung erfolgt (Userdefiniert mit Maus, zufällig, auf freien Bildschirmteilen, ...).

Farbauswahl

X-Clients haben fast immer die Möglichkeit, ihre Farbdarstellungen einzustellen. Dazu dienen in der Regel die Parameter `-fg` für die Vordergrundfarbe (foreground) und `-bg` für den Hintergrund (background).

Um also ein xterm mit blauem Hintergrund und gelber Schrift zu erhalten benutzen wir die folgende Zeile:

```
xterm -bg blue -fg yellow &
```

Konfiguration von X-Clients über Resources

Die Steuerung der Programmeigenschaften über Kommandozeilenparameter ist zwar eine mächtige Möglichkeit, jedoch kann es auch ganz schön lange Befehlszeilen geben, wenn wir etwa ein xterm an einer bestimmten Position in einer bestimmten Größe und Farbe mit einer anderen Schrift haben wollen. Dadurch entstünden Befehlszeilen wie etwa:

```
xterm -geometry 90x30-10-10 \
>   -fn -misc-fixed-bold-r-normal--13-100-100-100-c-70-iso8859-1\
>   -fg blue -bg bisque2 &
```

Zugegebenerweise wäre das etwas umständlich zu tippen. Um oft gebrauchte Einstellungen fest zu verändern steht uns das Resource-System zur Verfügung. Resources sind einfach nur Variablen, die ein Programm benutzt und die von außerhalb des Programms verändert werden können.

Die Grundeinstellungen dieser Resourcen stehen im Verzeichnis `/etc/X11/app-defaults`. Jeder User kann aber diese Werte verändern, so daß für ihn (und nur für ihn) Programme mit anderen Werten gestartet werden. Das geschieht in der Regel über die Datei `.Xresources` oder `.Xdefaults` im jeweiligen Home Verzeichnis des Users. Dort können Zeilen wie die folgenden eingetragen werden:

```
XTerm*font: -misc-fixed-bold-r-normal--13-100-100-100-c-70-iso8859-1
XTerm*Background: bisque2
XTerm*Foreground: blue
XTerm*geometry: 90x40
```

Diese Werte werden dann beim nächsten Start des X-Servers in den Speicher geladen und aktiviert. Jedesmal, wenn Sie jetzt xterm ohne Parameter aufrufen erscheint ein Fenster mit Hintergrundfarbe bisque2, Vordergrundfarbe blau, fetter Schrift und der Größe 90x40...

Um die Datei während des Betriebs zu laden (ohne den Server neu zu starten) kann der Befehl

```
xrdb -merge .Xresources
```

angegeben werden. Der Parameter -merge sorgt dafür, daß die bestehenden Resourcendefinitionen, die durch die Veränderungen nicht betroffen wurden erhalten bleiben.

Welche Resourcen ein Client versteht kann in der Regel in seiner Handbuchseite nachgelesen werden.

Die grundsätzliche Form der Eingabe ist immer die gleiche:

*Client (oder Clientname)*Resource: Wert*

In der Regel haben Clients sehr viele Resourcen, wesentlich mehr als mögliche Kommandozeilenparameter. So kann einem Xterm beispielsweise durch die Resource

```
XTerm*scrollbar:true
```

eine Scrollbar (Rollbalken) zugefügt werden. Damit alle diese Resourcen auch über die Kommandozeile möglich sind, gibt es den Parameter `-xrm`, dem ein Resourcenstring folgen kann. Damit ist es also auch möglich, Programmeinstellungen über die Kommandozeile

zu machen, für die es zwar Resourcen gibt aber keine Parameter.

Starteinstellungen

Wenn der X-Server über den Befehl **startx** gestartet wird, so wird beim Start automatisch die Datei `~/.xinitrc` oder - falls sie nicht vorhanden ist, die Datei `/etc/X11/xinit/xinitrc` abgearbeitet. Dabei handelt es sich um ein einfaches Shellscript, das verschiedene Aufgaben übernimmt:

- Die systemweite und die userbezogene Tastaturdefinition wird geladen (`/etc/X11/Xmodmap, ~/.Xmodmap`).
- Die systemweiten und die userbezogenen Resource-Dateien werden geladen (`~.Xdefaults, ~.Xresources, /etc/X11/Xresources`)
- Beliebige X-Clients können gestartet werden
- Der gewünschte Window-Manager wird gestartet

Einen ähnlichen Vorgang gibt es, wenn der X-Server vom Display-Manager gestartet wird. In diesem Fall wird die Datei `/etc/X11/xdm/Xsession` abgearbeitet.

Wenn im Homeverzeichnis eines Users ebenfalls eine Datei `~/.Xsession` existiert, so wird auch diese Datei abgearbeitet. So kann jeder User seine Einstellungen persönlich verändern.

In vielen Distributionen rufen sowohl xinitrc, als auch Xsession einfach nur ein anderes Script auf, um zu einer gemeinsamen Konfiguration zu gelangen.

X11 im Netz

Da das X-Protokoll ein Netzwerkprotokoll ist, kann jeder X-Client auch entsprechend so aufgerufen werden, daß er seine Ausgaben nicht auf dem lokalen Server macht, sondern auf einem anderen X-Server. Dazu gibt es verschiedene Mechanismen, die sowohl den Aufruf des Clients, als auch die Einstellungen auf dem Server betreffen.

Die Client-Seite

Jeder X-Client kann mit einem Parameter `-display` gestartet werden, mit dem angegeben werden kann, auf welchem Display der Client dargestellt werden möchte. Die Angabe des gewünschten Displays hat die Form:

Hostname:Servernummer.Displaynummer

Der Hostname kann weggelassen werden, wenn das Programm auf dem lokalen Server zugreifen soll. Servernummer und Displaynummer beziehen sich auf die einzelnen Server und Displays des jeweiligen Hosts. Das ist nur von Bedeutung, wenn auf einem Rechner mehrere

Server laufen (etwa wenn mehrere XTerminals angeschlossen sind) - im Normalfall steht hier ein `:0.0` also der Server 0 und das Display 0. In den meisten Fällen kann auch die Angabe der Displaynummer weggelassen werden (Displaynummern fallen nur an, wenn ein Server mehrere Graphikkarten anspricht) so daß einfach ein :0 als Serverbezeichnung reicht.

Damit es unter X11 möglich ist, lokale X-Clients ohne die Nennung eines Displays zu starten, wird beim Starten von X eine Umgebungsvariable `DISPLAY` definiert, die den Wert :0.0 beinhaltet. Dadurch wissen XClients auf welchen XServer sie zugreifen sollen, wenn keiner explizit angegeben wurde.

Ein Beispiel:

Wir sind im Rechner marvin eingeloggt und wollen hier das Programm netscape starten. Die Ausgaben dieses Programmes sollen aber auf dem XServer des Rechners hal ausgegeben werden. Wir schreiben also:

```
netscape -display hal:0.0
```

Natürlich kann auch hier wieder statt dem Rechnernamen die IP-Adresse stehen.

Die Server-Seite

Ein X-Server, auf dem Ausgaben von Clients anderer Hosts zugelassen werden sollen, muß natürlich zunächst diesen Hosts erlauben, auf seinen Dienst zuzugreifen. Dazu existiert eine Hostbasierte Zugriffskontrolle.

Das Programm, mit dem diese Kontrolle realisiert wird heisst **xhost** und ist sehr einfach anzuwenden:

```
xhost [+|-]
xhost Hostname
xhost -Hostname
```

Wenn **xhost** ohne Argumente aufgerufen wird, so zeigt es den Status der Zugriffskontrolle (an- oder ausgeschaltet) und die Liste der authorisierten Rechner an.

Durch den Aufruf von `xhost +` wird alle Zugriffskontrolle ausgeschaltet, jeder Rechner kann jetzt auf den Server zugreifen.

Mit `xhost -` werden die Zugriffskontrollmechanismen wieder aktiviert, nur authorisierte Rechner dürfen auf den Server zugreifen.

Durch die Nennung eines Rechnernamens nach **xhost** (also etwa `xhost hal`) wird der angegebene Rechner in die Liste der authorisierten Rechner aufgenommen. Programme auf diesem Rechner können also jetzt ihre Ausgaben auf dem Server machen.

Die Zeile `xhost -hal` würde den Rechner `hal` wieder von der Liste der authorisierten Rechner streichen.

Selbstverständlich können diese Einstellungen auch beim Start von X11 automatisch ablaufen. Es ist nur nötig, sie in die jeweils verwendete xinitrc Datei zu schreiben. Also entweder in die systemweite in `/etc/X11/xinit/xinitrc` oder in die private `$HOME/.xinitrc`

Eleganter ist allerdings die Möglichkeit, eine Datei namens `/etc/Xn.hosts` zu verwenden, die eine Liste aller Rechner enthält, die Zugriff auf den Server haben sollen. Das *n* im Dateinamen steht für die Servernummer, in unserem Fall also immer die 0. Diese Datei ist nicht standardmäßig vorhanden, sie muß bei Bedarf vom Systemverwalter angelegt werden.

Auflösung von Bibliotheksabhängigkeiten

Zur Prüfung und Auflösung von Library-Abhängigkeiten wurde im Abschnitt 1.102.4 - Verwaltung von Shared Libraries schon alles notwendige erläutert. Es gibt keinen Unterschied zwischen X11-Anwendungen und normalen Linux-Programmen hinsichtlich der Libraries.

Handbuchseiten

Hier habe ich die wichtigsten Handbuchseiten auf Deutsch abgelegt, die für die LPI 101 Prüfung notwendig sind. Zum Teil lagen sie schon auf Deutsch vor, zum anderen Teil habe ich sie übersetzt. Es gibt natürlich noch einige Handbuchseiten, die auch für die LPI-Prüfung wichtig sind, aber noch nicht übersetzt sind, ich arbeite daran... Die Seiten liegen hier in alphabetischer Reihenfolge. Es existiert aber auch eine thematische Sortierung

---

- apropos - sucht die Manualkurzbeschreibung in der Indexdatenbank
- at -Jobs zur späteren Ausführung in eine Warteschlange stellen,
- cat - gib Dateien aus
- catman - erzeugt oder aktualisiert die formatierten Manualseiten
- chfn - ändert das Kommentarfeld eines Usereintrags (Name und Informationen)
- chgrp - (change group) ändert die Gruppenzugehörigkeit einer Datei oder eines Verzeichnisses
- chmod - (change mode) ändert die Zugriffsrechte auf Dateien und Verzeichnisse
- chsh - wechselt die Loginshell
- compress - komprimiert Dateien
- cp - (copy) kopiert eine oder mehrere Dateien
- cron - Daemon zur Ausführung regelmäßig geplanter Aufgaben
- crontab - wartet crontab Dateien für einzelne User
- cut - schneidet bestimmte Teile aus den Zeilen einer Datei aus
- df - (disk free) zeigt den freien Festplattenplatz
- du - (disk usage) zeigt die Verteilung des belegten Plattenplatzes auf die Verzeichnisse
- expand - ersetzt Tabulatorzeichen durch Folgen von Leerzeichen
- find - sucht nach bestimmten Dateien
- fmt - einfacher Textformatierer
- gpasswd - Verwaltung der Datei /etc/group

- grep - durchsucht Dateien nach Ausdrücken
- groupadd - Anlegen einer neuen Gruppe
- groupdel - Löschen einer Gruppe
- groupmod - Ändern einer Gruppe
- groups - zeigt alle Gruppen, denen der Benutzer angehört
- grpck - Überprüfung der Integrität der Gruppen-Dateien
- gzip - komprimiert Dateien
- head - schreibt den Anfang einer Datei auf die Standardausgabe
- id - gibt die reale und die effektive User ID und Gruppen ID aus
- join - verknüpft zwei Dateien nach Schlüsselfeldern
- kill - beendet einen Prozess
- ln - (link) erzeugt einen Verzeichniseintrag einer existierenden Datei unter anderem Namen
- ls - (list) zeigt den Inhalt eines Verzeichnisses
- man - Programm zum Einsehen der Online-Manuale
- mandb - erzeugt oder aktualisiert die Indexdatenbank
- manpath - ermittelt den aktuellen Suchpfad für die Manualseiten
- mkdir - erzeugt ein leeres Verzeichnis
- mv - (move) verschiebt eine Datei oder benennt sie um
- newgrp - Wechselt die Logingruppe
- nice - läßt ein Programm mit veränderter Priorität laufen
- nl - nummeriert die Zeilen in einer Datei
- od - (octal dump) zeigt Dateien im oktalen, hexadezimalen oder in anderen Formaten an.
- passwd - Wechseln eines Userpassworts
- paste - fügt die Zeilen von zwei oder mehr Dateien horizontal zusammen
- pr - formatiert Textdateien zur Druckerausgabe
- ps - (process status) zeigt die Prozesse mit ihrem Status an

- pwck - Überprüfung der Integrität der Passwort-Dateien
- renice - verändert Prioritäten laufender Prozesse
- rm - löscht Dateien
- rmdir - löscht Verzeichnisse
- sed - (stream editor) ist ein Editor zur nicht-interaktiven Textbearbeitung
- sort - sortiert die Zeilen einer Textdatei
- split - spaltet eine Datei in mehrere kleinere
- tac - wie cat, nur umgekehrt
- tail - zeigt das Ende einer Datei
- tar - (tape archiver) verwaltet Dateiarchive
- tee - verzweigt die Ausgabe auf eine Datei
- top - Zeigt laufende CPU-Prozesse
- touch - ändert die Zeitmarkierung einer Datei
- tr - Ändert oder löscht einzelne Zeichen
- umask - bestimmt oder zeigt den voreingestellten Zugriffsmodus
- useradd - Legt einen neuen User an oder verändert die Grundeinstellungen zum Anlegen neuer User
- userdel - Löscht einen Useraccount und zugehörige Dateien
- usermod - Modifiziert einen Useraccount
- wc - zählt die Anzahl von Zeichen, Worten oder Zeilen
- whatis - durchsucht die Indexdatenbank nach Kurzbeschreibungen
- xargs - bildet Kommandozeilen aus der Standard-Eingabe und führt sie aus

Name

apropos - sucht die Manualkurzbeschreibung in der Indexdatenbank

Syntax

apropos [**-dhV**] [**-e**|**-r**|**-w**] [**-m** *System*[,...]] [**-M** *Pfad*] *Schlüsselwort* ...

Beschreibung

Innerhalb jeder Manualseite ist eine Kurzbeschreibung vorhanden. **apropos** sucht das Schlüsselwort in den Kurzbeschreibungen der *Indexdatenbank.* Falls es eine solche nicht im Manualpfad gibt, durchsucht es die *whatis*- Datenbank nach *Schlüsselwort.* Das Schlüsselwort kann Wildcards oder reguläre Ausdrücke enthalten. Bei der Standardsuche (nicht bei **-r** oder **-w**) wird nach Treffern am Anfang von Worten in den Beschreibungen und in Befehlsnamen gesucht. Die Nicht-Standardsuche ist weiter unten beschrieben.

Optionen

-d, --debug

Ausgeben von Fehlerinformationen.

-e, --exact

Sucht nach einem genau passenden Wort, im Gegensatz zu der sonst üblichen fuzzy-Übereinstimmung. Beispielsweise wird in diesem Fall das Schlüsselwort `string' nicht in dem Text `strings' gefunden. Ein Treffer muß also sowohl am Anfang als auch am Ende des Wortes übereinstimmen.

-r, --regex

Interpretiert jedes Schlüsselwort als regulären Ausdruck. Jedes Schlüsselwort wird im Befehlsnamen und in

der Beschreibung gesucht. Die Übereinstimmung ist nicht durch Wortgrenzen beschränkt.

-w, --wildcard

Das Schlüsselwort kann Wildcards wie in der Shell (z. B. *, ?, ...) beinhalten. Das Schlüsselwort wird in den Befehlsnamen und in den Kurzbeschreibungen gesucht. Nur wenn ein expandiertes Schlüsselwort auf eine komplette Beschreibung oder einen Befehlsnamen paßt, wird eine Übereinstimmung gefunden.

-m *System* [,...]**, --systems=***System*[,...]

Wenn auch Manualseiten von einem anderen Betriebssystem installiert sind, so kann auf sie mit dieser Option zugegriffen werden. Um beispielsweise die Manualseiten von NewOS zu durchsuchen, muß **-m NewOS** angegeben werden.

Das angegebene *System* kann eine durch Kommata abgetrennte Aufzählung von Betriebssystemnamen sein. Die normalen Seiten werden durch den Betriebssystemnamen **man** angesprochen. Diese Option überschreibt die Umgebungsvariable $**SYSTEM.**

-M *Pfad***, --manpath=***Pfad*

Gibt die Manualpfade an. Wenn mehrere Pfade angegeben werden, müssen diese durch Doppelpunkte getrennt sein. Als Default benutzt **apropos manpath,** um den Suchpfad zu bestimmen. Diese Option überschreibt die $**MANPATH** Umgebungsvariable.

-h, --help

Zeigt einen Hilfstext an.

-V, --version

Zeigt Programmversion und Autor an.

Dateien

/var/catman/.../index.(bt|db|dir|pag)

Die **FSSTND** complianten globalen *Indexdatenbanken.*

/usr/man/.../whatis

Traditionelle *whatis* Datenbank.

Siehe Auch

whatis(1) , **man**(1) , **mandb**(8) .

Fehler

Wenn die **-r** oder **-w** Optionen benutzt werden und in einem *Schlüsselwort* Shell-Metazeichen vorkommen, kann es notwendig sein, *Schlüsselwort* zu quoten, damit die Shell nicht versucht, diese Zeichen zu interpretieren. Die **-r** und **-w** Optionen können verwirrende Ausgaben erzeugen.

Autor

Wilf. (G.Wilford@ee.surrey.ac.uk), die deutsche Übersetzung ist von Anke Steuernagel (a_steuer@informatik.uni-kl.de) und Nils Magnus (magnus@informatik.uni-kl.de).

Name

at, atq, atrm - Jobs zur späteren Ausführung in eine Warteschlange stellen, den Inhalt der Warteschlange lesen oder Jobs aus der Warteschlange löschen.

Syntax

at [**-V**] [**-q** *Warteschlange*] [**-f** *Datei*] [**-mldbv**] **Zeit**
at -c *Job* [*Job...*]
atq [**-V**] [**-q** *Warteschlange*]
atrm [**-V**] *Job* [*Job...*]

Beschreibung

at ließt Kommandos von der Standard-Eingabe oder aus einer angegebenen Datei, die zu einem späteren Zeitpunkt durch **/bin/sh** ausgeführt werden sollen.

at

Führt Kommandos zu einer angegebenen Zeit aus

atq

Zeigt die Jobs des Users, die er bereits mit at in die Warteschlange aufgenommen hat. Ist der User der Superuser, so zeigt **atq** alle Jobs aller User. Das Ausgabeformat jedes Jobs ist: Jobnummer, Datum, Stunde, Jobklasse.

atrm

Löscht Jobs aus der Warteschlange. Die zu löschenden Jobs werden über ihre Jobnummer angegeben.

At erlaubt ziemlich komplexe Zeitangaben, entsprechend dem POSIX.2 Standard. Es akzeptiert Zeitangaben der Form **HH:MM** um Jobs zu einer bestimmten Uhrzeit des Tages auszuführen. (Wenn dieser Zeitpunkt schon

vorbei ist, wird der Folgetag angenommen.) Stattdessen können auch die Worte **midnight** (Mitternacht), **noon** (Mittag) oder **teatime** (16 Uhr) angegeben werden und der Zeitangabe darf ein **am** oder **pm** folgen, um einer 12-stündige Zeitangabe Vor- oder Nachmittag hinzuzufügen. Auch der Tag, an dem der Job ausgeführt werden soll, kann bestimmt werden, indem ein Datum im Format **MMDDYY** oder **MM/DD/YY** oder **DD.MM.YY.** angegeben wird. Die Angabe des Datums muß nach der Angabe der Uhrzeit erfolgen und wird durch ein Leerzeichen von ihr getrennt. Auch Angaben wie **now + *Zahl Zeiteinheiten*** sind erlaubt, wobei für die Zeiteinheiten die Werte **minutes, hours, days** oder **weeks** erlaubt sind. Außerdem können Zeitangaben der Begriff **tomorrow** folgen, um den Job am folgenden Tag auszuführen.

Um zum Beispiel einen Job um 4 Uhr Nachmittag in drei Tagen auszuführen würde der Befehl lauten: **at 4pm + 3 days**, um einen Job morgen um 13 Uhr auszuführen genügt **at 13:00 tomorrow**

Die Kommandos selbst, die at ausführen soll werden entweder von der Standard-Eingabe gelesen, oder aus der Datei, die mit der **-f** Option angegeben wurde. Wenn die Kommandos also von Hand über die Standard-Eingabe eingegeben werden, so wird diese Eingabe durch ein Dateiendezeichen (Strg-D) in einer eigenen Zeile abgeschlossen.

Das Arbeitsverzeichnis, der Umgebungsspeicher (mit Ausnahme der Variablen `TERM, DISPLAY` und _) sowie die umask-Einstellungen werden so gesetzt, wie sie beim Aufruf des at-Kommandos gesetzt waren. Ein At-Kommando, das aus einer su-Shell aufgerufen wurde behält die aktuelle UserID bei. Dem User der das at-Kommando abgeschickt hatte, werden alle Ausgaben (STDOUT und STDERR) per Mail zugeschickt. Wenn at über eine su-Shell aufgerufen wurde, so wird die Mail an den Eigentümer der Login-Shell und nicht den der su-Shell geschickt.

Der Systemverwalter kann die at-Kommandos auf jeden Fall benutzen. Für alle anderen User gilt eine Zugriffskontrolle, die über die Dateien **/etc/at.allow** und **/etc/at.deny** geregelt wird.

Wenn die Datei */etc/at.allow* existiert, dürfen nur die darin aufgelisteten User das at-Kommando benutzen.

Wenn */etc/at.allow* nicht existiert, aber die Datei */etc/at.deny* existiert, dann dürfen alle User, außer den in dieser Datei genannten das at-Kommando benutzen.

Existieren keine der genannten Dateien, dann darf nur der Superuser das at-Kommando ausführen.

Eine leere, aber existierende Datei */etc/at.deny* meint, daß alle User at-Kommandos benutzen dürfen. Das ist die voreingestellte Konfiguration.

Optionen

-V

gibt die Versionsnummer auf der Standard-Fehlerausgabe aus.

-q *Warteschlange*

benutzt die angegebene Warteschlange. Eine Warteschlange wird durch einen einzelnen Buchstaben von "a" bis "z" definiert. Die voreingestellte Warteschlange für at ist "a". Warteschlangen mit späteren Buchstaben laufen mit einem höheren Nice-Wert (also niedriger Priorität).

Wenn **atq** eie Warteschlange mitgegeben wurde, zeigt er nur die Jobs dieser Warteschlange.

-m

Schickt dem User, der den Job aufgetragen hat auch dann eine Mail, wenn der Job selbst keine Ausgaben produziert hat.

-f *Datei*

Ließt den Job aus der *Datei* statt von der Standard-Eingabe.

-l

Ist ein Alias für **atq.**

-d

Ist ein Alias für **atrm.**

-v

Zeigt die Zeit, zu der der Job ausgeführt werden soll.

Das Format der Zeitangabe ist wie bei "1997-02-20 14:50" wenn nicht die Umgebungsvariable **POSIXLY_CORRECT** gesetzt ist. In diesem Fall wäre die Angabe im Format "Thu Feb 20 14:50:00 1996".

-c

gibt die auf der Kommandozeile mitgegebenen Jobs auf der Standard-Ausgabe aus.

Dateien

/var/spool/atjobs
/var/spool/atspool
/var/run/utmp
/etc/at.allow
/etc/at.deny

Siehe auch

cron(1) , **nice**(1) , **sh**(1) , **umask**(2) , **atd**(8) .

Autor

At wurde hauptsächlich von Thomas Koenig, ig25@rz.uni-karlsruhe.de geschrieben.

Bezeichnung

cat - gib Dateien aus

Syntax

cat [**-benstuvAET**] [**--number**] [**--number-nonblank**] [**--squeeze-blank**] [**--show-nonprinting**] [**--show-ends**] [**--show-tabs**] [**--show-all**] [*Datei*...]

Beschreibung

cat liest beliebige Dateien und schreibt sie ohne Veränderung in die Standardausgabe. Durch Umlenkung der Ausgabe auf eine Datei können so Dateien verkettet werden. Außerdem wird **cat** häufig benutzt, um Dateien an Programme zu übergeben, die nur von der Standardeingabe lesen. (Solche Programme werden im allgemeinen als Filter bezeichnet.) Für die Übergabe steht entweder die Ausgabeumlenkung der Shell mit > und >> zur Verfügung, oder die Ausgabe wird durch eine Pipeline | an den Filter weitergeleitet.

Optionen

-b

alle nicht leeren Zeilen erhalten eine Zeilennummer

-e

das gleiche wie **-vE**

-n

sämtliche Zeilen werden nummeriert

-s

mehrere leere Zeilen in Folge werden zu einer einzigen leeren Zeile zusammengefaßt

-t

das gleiche wie **-vT**

-u

ohne Funktion

-v

alle Kontrollzeichen außer **tab** und **newline** werden angezeigt

-A

das gleiche wie **-vET**

-E

gibt ein **$** Zeichen am Ende jeder Zeile aus

-T

die **tab** werden als **^I** angezeigt

Siehe Auch

tac(1)

Autor

Torbjorn Granlund und Richard Stallman

Name

catman - erzeugt oder aktualisiert die formatierten Manualseiten

Syntax

catman [**-dhV**] [**-M** *Pfad*] [*Abschnitt*] ...

Beschreibung

catman wird benutzt, um einen aktuellen Satz von vorformatierten Manualseiten zu erzeugen, die als cat-Seiten bezeichnet werden. Die cat-Seiten werden wesentlich schneller angezeigt als die unformatierten Manualseiten, benötigen aber zusätzlichen Speicherplatz auf der Festplatte. Die Entscheidung zur Unterstützung von cat-Seiten liegt beim lokalen Administrator, der für diese Seiten passende Verzeichnisse anlegen muß.

Die Optionen für **catman** geben die Manualseiten und die Abschnitte an, die vorformatiert werden sollen. Im Normalfall werden die Manualseiten verwendet, die in der Manualkonfigurationsdatei beschrieben sind. Normalerweise werden die Abschnitte formatiert, die entweder in der Umgebungsvariable $**MANSEC** stehen oder fest in **man**(1) einkompiliert sind, wenn $**MANSEC** undefiniert ist. Wird **catman** eine durch Whitespace abgetrennte Liste von Abschnitten übergeben, so werden nur die Manualseiten in den betreffenden Abschnitten bearbeitet.

Um festzustellen, welche Dateien formatiert werden müssen, macht **catman** Gebrauch von den *Indexdatenbanken* der jeweiligen Manualseiten.

Optionen

-d, --debug

Ausgeben von Fehlerinformationen.

-M *Pfad***, --manpath=***Pfad*

Gibt einen alternativen Manualpfad an. Als Default sind dies alle Pfade, die in der Manualkonfigurationsdatei als global deklariert sind. Die Umgebungsvariable $**MANPATH** wird von **catman** nicht benutzt.

-h, --help

Zeigt einen Hilfstext an.

-V, --version

Zeigt Programmversion und Autor an.

Dateien

/var/catman/.../index.(bt|db|dir|pag)

Die **FSSTND** complianten globalen *Indexdatenbanken.*

/usr/man/.../whatis

Traditionelle *whatis* Datenbank.

Siehe Auch

man(1) , **manpath**(5) , **mandb**(8) .

Fehler

Auf Systemen mit kleinem (~4 kByte) Zwischenspeicher für Argumente von **execve**(2) kann es nötig sein, daß **catman** mehrere kleine im Gegensatz zu wenigen großen Batchjobs ausführen muß. In diesem Fall ist es für **man** möglich, diesen Eintrag (korrekt) aus einer Datenbank zu löschen (vielleicht existiert er in Wirklichkeit gar nicht mehr), der vom aktuellen Aufruf von **catman** benötigt wird, um die nächste Seite zu bestimmen. Wenn die *Indexdatenbank* einen Hashtable-Lookup (ndbm, gdbm) benutzt, führt dies zu einem `sich am eigenen Schopf aus

dem Sumpf ziehen'. Obgleich dies selten auftritt, führt es dazu, daß **catman** nicht mehr die weiteren Seiten des Abschnittes bearbeitet.

Hinweis: **Linux** hat einen Zwischenspeicher von 128 kByte für **execve (2)** Argumente.

Autor

Wilf. (G.Wilford@ee.surrey.ac.uk), die deutsche Übersetzung ist von Anke Steuernagel (a_steuer@informatik.uni-kl.de) und Nils Magnus (magnus@informatik.uni-kl.de).

Name

chfn - Ändert das Kommentarfeld eines Usereintrags (Name und Informationen)

Syntax

chfn

[**-f** ***Klarname***] [**-r** ***Büroraum Nr.***]
[**-w** ***Telephon Arbeit***] [**-h** ***Telephon Privat***] [**-o** ***Anderes***] [*Username*]

Beschreibung

chfn ändert den Klarnamen, Büroraumnummer, Bürotelephonnummer und Privattelephonnummer eines Useraccounts. Diese Informationen werden typischerweise von **finger**(1) und ähnlichen Programmen dargestellt. Ein Normaluser darf nur die Einträge seines Accounts ändern, und er darf grundsätzlich nicht das Feld für **Anderes** mit der Option **-o** ändern. Der Superuser kann die Informationen für alle User verändern, inklusive der **Anderes** Angabe.

Die einzige Einschränkung für diese Felder ist, daß keine Steuerzeichen und weder Komma, Doppelpunkt noch Gleichheitszeichen verwendet werden dürfen. Das Feld **Anderes** kennt diese Einschränkungen nicht.

Wenn keine Option angegeben wurde, arbeitet **chfn** in einem interaktiven Modus, in dem der User nach allen Angaben gefragt wird.

Dateien

/etc/passwd - Useraccount Informationen
/etc/shadow - Sichere Useraccount Informationen

Siehe auch

passwd(5)

Autor

Julianne Frances Haugh (jfh@bga.com)

Handbuchseiten

Hier habe ich die wichtigsten Handbuchseiten auf Deutsch abgelegt, die für die LPI 101 Prüfung notwendig sind. Zum Teil lagen sie schon übersetzt vor, zum anderen Teil habe ich sie übersetzt. Es gibt natürlich noch einige Handbuchseiten, die auch für die LPI-Prüfung wichtig sind, aber noch nicht übersetzt sind, ich arbeite daran... Sie liegen hier in thematischer Sortierung. Es gibt alle Handbuchseiten aber auch in alphabetischer Reihenfolge.

Textfilter

- cat - gib Dateien aus
- cut - schneidet bestimmte Teile aus den Zeilen einer Datei aus
- expand - ersetzt Tabulatorzeichen durch Folgen von Leerzeichen
- fmt - einfacher Textformatierer
- grep - durchsucht Dateien nach Ausdrücken
- head - schreibt den Anfang einer Datei auf die Standardausgabe
- join - verknüpft zwei Dateien nach Schlüsselfeldern
- nl - nummeriert die Zeilen in einer Datei
- od - (octal dump) zeigt Dateien im oktalen, hexadezimalen oder in anderen Formaten an.
- paste - fügt die Zeilen von zwei oder mehr Dateien horizontal zusammen
- pr - formatiert Textdateien zur Druckerausgabe
- sed - (stream editor) ist ein Editor zur nicht-interaktiven Textbearbeitung
- sort - sortiert die Zeilen einer Textdatei
- split - spaltet eine Datei in mehrere kleinere
- tac - wie cat, nur umgekehrt
- tail - zeigt das Ende einer Datei
- tr - Ändert oder löscht einzelne Zeichen

- wc - zählt die Anzahl von Zeichen, Worten oder Zeilen

Befehle zum Umgang mit Dateien, Verzeichnissen und Links

- chgrp - (change group) ändert die Gruppenzugehörigkeit einer Datei oder eines Verzeichnisses
- chmod - (change mode) ändert die Zugriffsrechte auf Dateien und Verzeichnisse
- cp - (copy) kopiert eine oder mehrere Dateien
- find - sucht nach bestimmten Dateien
- ln - (link) erzeugt einen Verzeichniseintrag einer existierenden Datei unter anderem Namen
- ls - (list) zeigt den Inhalt eines Verzeichnisses
- mkdir - erzeugt ein leeres Verzeichnis
- mv - (move) verschiebt eine Datei oder benennt sie um
- rm - löscht Dateien
- rmdir - löscht Verzeichnisse
- touch - ändert die Zeitmarkierung einer Datei
- umask - bestimmt oder zeigt den voreingestellten Zugriffsmodus

Befehle für Pipes

- tee - verzweigt die Ausgabe auf eine Datei
- xargs - bildet Kommandozeilen aus der Standard-Eingabe und führt sie aus

Befehle für Prozesse

- kill - beendet einen Prozess
- nice - läßt ein Programm mit veränderter Priorität laufen
- ps - (process status) zeigt die Prozesse mit ihrem Status an
- renice - verändert Prioritäten laufender Prozesse
- top - Zeigt laufende CPU-Prozesse

Befehle zum Verwalten von Usern und Gruppen

- chfn - ändert das Kommentarfeld eines Usereintrags (Name und Informationen)
- chsh - wechselt die Loginshell
- gpasswd - Verwaltung der Datei /etc/group
- groupadd - Anlegen einer neuen Gruppe
- groupdel - Löschen einer Gruppe
- groupmod - Ändern einer Gruppe
- groups - zeigt alle Gruppen, denen der Benutzer angehört
- grpck - Überprüfung der Integrität der Gruppen-Dateien
- id - gibt die reale und die effektive User ID und Gruppen ID aus
- newgrp - Wechselt die Logingruppe
- passwd - Wechseln eines Userpassworts
- pwck - Überprüfung der Integrität der Passwort-Dateien
- useradd - Legt einen neuen User an oder verändert die Grundeinstellungen zum Anlegen neuer User
- userdel - Löscht einen Useraccount und zugehörige Dateien
- usermod - Modifiziert einen Useraccount

Befehle fürs Dateisystem

- df - (disk free) zeigt den freien Festplattenplatz
- du - (disk usage) zeigt die Verteilung des belegten Plattenplatzes auf die Verzeichnisse

Befehle für das Handbuchsystem

- apropos - sucht die Manualkurzbeschreibung in der Indexdatenbank
- catman - erzeugt oder aktualisiert die formatierten Manualseiten
- man - Programm zum Einsehen der Online-Manuale

- mandb - erzeugt oder aktualisiert die Indexdatenbank
- manpath - ermittelt den aktuellen Suchpfad für die Manualseiten
- whatis - durchsucht die Indexdatenbank nach Kurzbeschreibungen

Befehle zur zeitgesteuerten Ausführung von Programmen

- at -Jobs zur späteren Ausführung in eine Warteschlange stellen,
- cron - Daemon zur Ausführung regelmäßig geplanter Aufgaben
- crontab - wartet crontab Dateien für einzelne User

Befehle zum archivieren

- compress - komprimiert Dateien
- gzip - komprimiert Dateien
- tar - (tape archiver) verwaltet Dateiarchive

Bezeichnung

chgrp - (change group) ändert die Gruppenzugehörigkeit einer Datei oder eines Verzeichnisses

Syntax

chgrp [**-Rcfv**] [**--recursive**] [**--show-changes**] [**--silent**] [**--quiet**] [**--verbose**] *Gruppe* Datei ...

Beschreibung

Der Befehl **chgrp** ändert die Gruppenzugehörigkeit einer Datei oder eines Verzeichnisses. Die Benutzung von **chgrp** ist nur dem Eigentümer und dem Superuser (root) erlaubt. Der Eigentümer kann eine Datei nur den Gruppen zuordnen, denen er selbst auch angehört.

Die *Gruppe* kann als ganze Zahl oder als Name angegeben werden. Die möglichen Gruppen und ihre Kennzahlen sind in der Datei /etc/group festgelegt und können mit **id** angezeigt werden.

Optionen

-c

(changes) diese Option zeigt die Dateien an, deren Gruppe geändert wird

-f

(force) es werden keine Fehlermeldungen ausgegeben

-v

(verbose) alle Aktionen werden angezeigt

-R

(recursive) die Gruppenzugehörigkeit der Dateien in den Unterverzeichnissen wird ebenfalls geändert

Siehe Auch

chmod(1) , **chown**(1) .

Autor

David MacKenzie

Bezeichnung

chmod - (change mode) ändert die Zugriffsrechte auf Dateien und Verzeichnisse

Syntax

chmod [**-Rcfv**] *Modus* Datei ...

Beschreibung

chmod setzt oder ändert die Zugriffsrechte auf Dateien oder Verzeichnisse. Die Benutzung von **chmod** ist nur dem Eigentümer oder dem Superuser (root) erlaubt.

Die Zugriffsrechte werden als Modus bezeichnet. Der *Modus* kann entweder als (drei- oder vierstellige) Oktalzahl oder durch Buchstabenkennungen angegeben werden. Bei Angabe als Oktalzahl legen die letzten drei Ziffern jeweils die Rechte für den Besitzer, die Gruppe und die Anderen fest. Die einzelnen Bits der Oktalziffer stehen dabei für Lesen (4) , Schreiben (2) und Ausführen.

Wenn vier Ziffern angegeben werden, so setzt die erste Ziffer spezielle Ausführungsmodi:

Wenn das erste Bit (4) dieser Zahl gesetzt ist, wird ein Programm mit der effektiven Benutzerkennung (**EUID** für Effective User-ID) des Besitzers dieser Datei ausgeführt.

Wenn das zweite Bit (2) dieser Zahl gesetzt ist, wird ein Programm mit der Gruppenkennung dieser Datei anstelle der realen Gruppenkennung des aufrufenden Benutzers ausgeführt.

Das dritte Bit (1) schließlich hat unter Linux nur bei Verzeichnissen eine Bedeutung.

Die Buchstabenkennung setzt sich aus den folgenden Teilen zusammen:

[**ugoa**...] [[**+-=**] [**rwxstugo**...]...] [**,**...]

Dabei steht **u** (user) für Besitzer, **g** (group) für Gruppe, **o** (other) für Andere und **a** (all) für Alle. Die arithmetischen Symbole **+-=** geben an, ob eine Berechtigung hinzugefügt (+), gelöscht (-) oder gesetzt (=) werden soll. Die Berechtigungen sind **r** (read) für Lesen, **w** (write) für Schreiben, **x** (execute) für Ausführen. Die Option **s** (set user/group ID on execution) setzt die Identitätskennung bei der Programmausführung. Die Option **t** (text) schützt die Dateien eines beschreibbaren Verzeichnisses vor Löschung durch fremde Systembenutzer. Die nachgestellten **u** , **g** und **o** schützen die entsprechenden Rechte für Besitzer, Gruppe und Andere vor Veränderung (zur Benutzung im Zusammenhang mit **-a**).

Die Rechte von symbolischen Links werden von chmod nicht geändert. Es gelten hier immer die Rechte der Datei, auf die der Link zeigt.

Optionen

-c

(changes) es werden nur die Dateien angezeigt, deren Zugriffsrechte tatsächlich verändert werden

-f

(force) Fehlermeldungen wegen fehlgeschlagener Änderungsversuche werden unterdrückt

-v

(verbose) alle Aktionen werden angezeigt

-R

(recursive) die Zugriffsrechte aller Dateien in den Unterverzeichnissen werden ebenfalls geändert

Siehe Auch

chgrp(1) , **chown**(1) .

Autor

David MacKenzie

Name

chsh - Wechselt die Loginshell

Syntax

chsh

[**-s *Loginshell***] [*Username*]

Beschreibung

chsh wechselt die Loginshell des Users. Ein Normaluser darf nur seine Loginshell wechseln, der Superuser kann die Loginshell jedes Accounts wechseln.

Die einzige Einschränkung für diese Angabe ist, daß das Programm, das als neue Loginshell verwendet werden soll in */etc/shells* aufgeführt sein muß, wenn es nicht der Superuser ist, der **chsh** aufruft. Der kann jedes beliebige Programm als Loginshell für jeden beliebigen User angeben.

Wenn die **-s** Option nicht angegeben wurde, arbeitet **chsh** in einem interaktiven Modus, in dem der User die gewünschte Shell eingeben kann.

Dateien

/etc/passwd - Useraccount Informationen
/etc/shadow - Sichere Useraccount Informationen

Siehe auch

chfn(1) , **passwd**(5)

Autor

Julianne Frances Haugh (jfh@bga.com)

Bezeichnung

compress - komprimiert Dateien

Syntax

compress [**-cdfrvV**] [**-b** *Maxbits*] [*Datei*...]

Beschreibung

compress komprimiert Dateien mit LZW (einem veränderten Lempel-Ziv) Algorithmus. **compress** überprüft erst, ob die *Datei* nach der Kompression wirklich kleiner ist als vorher. Nur in diesem Fall wird die *Datei* durch die komprimierte *Datei* **.Z** ersetzt. Wenn keine Datei angegeben wird, liest **compress** von der Standardeingabe und schreibt in die Standardausgabe. Wenn die Ausgabedatei *Datei* **.Z** schon existiert, wird sie nicht überschrieben. **compress** erhält den Zeitstempel der Datei.

Die Programme **uncompress**und **zcat** sind Links auf **compress**, bei denen bestimmte Optionen vorbelegt sind.

uncompress arbeitet wie **compress -d**, das heißt, es entkomprimiert die mit **compress** gepackten Dateien.

zcat arbeitet wie **compress -dc**, das heißt, es schreibt die entkomprimierten Dateien auf die Standardausgabe. Diese Funktion wird in den Shellscripts {bsf zdiff}, **zmore** oder **zless** benutzt, um den Inhalt komprimierter Dateien direkt an bestimmte Filter weiterzuleiten.

Das **compress** Programm kann als Standardpacker für Unix bezeichnet werden. Es wird im Zusammenhang mit **tar** auch für gepackte Dateiarchive verwendet. Das GNU **tar** bietet eine direkte Unterstützung von **compress**.

Allerdings wird in letzter Zeit dazu übergegangen, das neue **gzip** Programm zum Packen einzelner Dateien zu benutzen, weil es deutlich höhere Kompressionsraten aufweisen kann. Es ist damit zu rechnen, daß **compress** durch **gzip** verdrängt wird.

Optionen

-c

die (de-)komprimierte Datei wird in die Standardausgabe geschrieben

-d

dekomprimiert die Datei; die komprimierte Datei wird ersetzt

-f

überschreibt existierende Ausgabedateien und ersetzt *Datei*, selbst wenn sie durch die Kompression nicht kleiner wird

-v

gibt den Dateinamen und das Größenverhältnis aus

-V

gibt die Versionsnummer aus

-r

komprimiert rekursiv alle Dateien in den Unterverzeichnissen

-b *Maxbits*

setzt die Kompressionstiefe (Voreinstellung ist 16 Bit)

Siehe Auch

das LunetIX Linuxhandbuch

Autoren

Spencer W. Thomas, Jim McKie, Steve Davies, Ken Turkowski, James A. Woods, Joe Orost, Dave

Mack und Peter Jannesen

Bezeichnung

cp - (copy) kopiert eine oder mehrere Dateien

Syntax

cp [*Optionen*] *Quelle* Ziel

cp [*Optionen*] *Quelle* ... Verzeichnis

Optionen

-a

(archiv) das gleiche wie -dpR

-b

(backup) sichert Dateien im Zielverzeichnis vor dem Überschreiben

-d

(no-dereference) kopiert die Links und nicht die Dateien, auf die der Link zeigt

-f

(force) Dateien im Zielverzeichnis werden überschrieben

-i

(interactive) erwartet Bestätigung vor dem Überschreiben bereits existierender Dateien

-l

(link) macht Links anstelle von Kopien (nur bei normalen Dateien)

-P

(path) die Quelldateien werden mit Pfad relativ zum Zielverzeichnis kopiert

-p

(preserve) erhält die Zugriffsrechte und Eigentümer des Originals (nicht die SUID und SGID Bits)

-r

kopiert die Dateien der Unterverzeichnisse mit

-s

(symbolic link) macht symbolische Links anstelle von Kopien (absolute Pfadnamen)

-u

(update) überschreibt Ziel- nur durch neuere Quelldateien

-v

(verbose)

-x

(one file-system) ignoriert Unterverzeichnisse, die in anderen Dateisystemen angesiedelt sind

-R

(recursive)

-S *Endung*

(Endung) sichert die Dateien vor dem Überschreiben durch Umbenennung mit der *Endung*; Voreinstellung ist **~**

-V {**numbered**, **existing**, **simple**}

(version-control) erhält auch frühere Versionen einer Datei, indem jeweils neue Backups erzeugt werden. Die Art der Backups kann auch durch die Umgebungsvariable **VERSION_CONTROL** bestimmt werden. Die Option **-V** überschattet **VERSION_CONTROL**. Wenn weder die Option noch die Environmentvariable gesetzt sind, wird *existing* benutzt. Gültige Werte für **VERSION_CONTROL** sind:

numbered

Backups werden nummeriert

existing

Backups werden nur für Dateien nummeriert die bereits nummerierte Backups haben.

simple

es werden immer einfache Backups gemacht

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Torbjorn Granlund, David MacKenzie und Jim Meyering

Name

cron - Daemon zur Ausführung regelmäßig geplanter Aufgaben

Syntax

cron

Beschreibung

Cron sollte aus /sbin/init.d/cron, /etc/rc oder /etc/rc.local heraus gestartet werden. Er startet automatisch als Daemon im Hintergrund, so daß es nicht notwendig ist, ihn mit einem '&' zu starten.

Cron durchsucht /var/spool/cron/tabs nach crontab Dateien die nach den Namenseinträgen in /etc/passwd benannt sind; die gefundenen crontabs werden in den Arbeitsspeicher geladen. *Cron* sucht auch nach /etc/crontab das in einem unterschiedlichen Format vorliegt (siehe *crontab(5)).* Zusätzlich ließt cron die Dateien in /etc/cron.d; die Dateien in /etc/cron.d werden als Erweiterungen zur /etc/crontab-Datei verstanden, das heißt, sie müssen der Syntax dieser Datei folgen und daher das *User* Feld besitzen. Die Zielsetzung dieses Features ist es, es Paketen zu ermöglichen, die eine feinere Kontrolle ihrer geplanten Ausführung bedürfen, als es die /etc/cron.{daily,weekly,monthly} Verzeichnisse erlauben, eine crontab-Datei ins Verzeichnis /etc/cron.d zu plazieren.

Wie /etc/crontab, so werden auch die Dateien in /etc/cron.d ständig überwacht, ob sich Änderungen ergeben haben. *Cron* erwacht dann jede Minute und überprüft alle gespeicherten Crontabs, ob sie Jobs enthalten, die in der aktuellen Minute ausgeführt werden sollten. Wenn solche Jobs gefunden werden, dann führt cron sie aus. Wenn Jobs ausgeführt werden, wird jede Ausgabe der Jobs an den User gemailt, dessen Crontab den Jobauftrag enthielt bzw. an den User, der in der MAILTO-Umgebungsvariable genannt ist, falls diese Variable existiert.

Zusätzlich überprüft *cron* jede Minute, ob sich die Zeitmarke der letzten Veränderung (mtime) seines

Spoolverzeichnisses oder die der Datei /etc/crontab verändert hat. Falls sie sich verändert hat, überprüft cron die Zeitmarken aller Dateien in diesen Verzeichnissen und läd die neu, deren Zeitmarken sich verändert haben. Das **crontab**(1) Kommando verändert die Zeitmarke des Spoolverzeichnisses jedesmal, wenn ein crontab verändert wurde.

Siehe auch

crontab(1) , crontab(5)

Autor

Paul Vixie <paul@vix.com>

Name

crontab - wartet crontab Dateien für einzelne User

Syntax

crontab [-u user] Datei
crontab [-u user] { -l | -r | -e }

Beschreibung

crontab ist das Programm, mit dem die Tabellen, die den Cron-Daemon steuern installiert, deinstalliert oder aufgelistet werden können. Jeder User kann seine eigene crontab-Datei besitzen, die - obwohl sie im var-Verzeichnis liegen - nicht dazu gedacht sind, direkt editiert zu werden.

Wenn die Datei /etc/cron.allow existiert, muß ein User, der dieses Kommando ausführen will, darin aufgeführt sein. Wenn die Datei /etc/cron.allow nicht existiert, aber dafür die Datei /etc/cron.deny existiert, so darf ein User dort nicht aufgeführt sein, um dieses Kommando auszuführen. Wenn weder die eine, noch die andere Datei existiert, so kann - abhängig von der rechnerspezifischen Konfiguration - entweder nur der Superuser (root) oder alle User das Kommando benutzen.

Wenn die *-u* Option angegeben wurde, dann spezifiziert sie den Namen des Users, dessen Crontab bearbeitet werden soll. Wenn diese OPtion nicht gesetzt ist, bearbeitet *crontab* die Datei des Users, der das *crontab* Kommando ausführt. Wenn dieser User seine Identität mit *su*(8) gewechselt hatte, so kann das das crontab-Kommando verwirren. In diesem Fall ist es stets vorzuziehen, den gewünschten Usernamen mit der *-u* Option mit anzugeben.

Die erste Aufrufform dieses Kommandos dient dazu, eine neue crontab-Datei aus der angegebenen Datei (oder der Standard-Eingabe wenn statt der Datei ein - angegeben wurde) anzulegen.

Die *-l* Option zeigt den Inhalt des aktuellen Crontabs des aufrufenden (oder angegebenen) Users.

Die *-r* löscht den aktuellen Crontab des aufrufenden (oder angegebenen) Users.

Die *-e* Option dient dazu, den aktuellen Crontab des aufrufenden (oder angegebenen) Users zu editieren. Der dazu aufgerufene Editor ist der in der Variable VISUAL oder EDITOR definierte (vi). Nachdem der Editor mit Sichern verlassen wurde wird der modifizierte Crontab automatisch installiert.

Siehe auch

crontab(5) , cron(8)

Dateien

```
/etc/cron.allow
/etc/cron.deny
```

Standards

Das *crontab* Kommando ist konform zum IEEE Standard 1003.2-1992 (POSIX).

Autor

`Paul Vixie <paul@vix.com>`

Bezeichnung

cut - schneidet bestimmte Teile aus den Zeilen einer Datei aus

Syntax

cut -b *Bereich* [**-n**] [*Datei*...]

cut -c *Bereich* [*Datei*...]

cut -f *Bereich* [**-d** *Trenner*] [**-s**] [*Datei*...]

Beschreibung

cut liest aus den angegebenen Dateien oder von der Standdardeingabe und gibt bestimmte Teile jeder Eingabezeile auf die Standardausgabe. Welcher Teil der Eingabezeile ausgegeben wird, hängt von der gewählten Option und der Wahl eines Bereiches ab. Ein *Bereich* ist eine Liste von einzelnen Zahlen oder Zahlenbereichen. Ein Zahlenbereich ist ein Ausdruck der Form *m* - *n*. Wird eine der Zahlen *m* oder *n* weggelassen, so wird der Zeilenanfang bzw. das Zeilenende angenommen.

Optionen

-b *Bereich*

gibt nur die Bytes (Zeichen) im *Bereich* aus. **tab** und **backspace** werden als ein Zeichen behandelt

-c *Bereich*

gibt nur die Zeichen im *Bereich* aus. Diese Option ist identisch mit der Option **-b**. **tabs** und **backspace** werden als ein Zeichen behandelt

-f *Bereich*

gibt die Felder im *Bereich* aus; die einzelnen Felder sind durch **tab** getrennt

-d *Trenner*

benutzt den *Trenner* anstelle eines **tab** bei der Option **-f**

-n

ohne Funktion; vorgesehen für spätere Unterstützung internationaler Zeichensätze mit mehreren Bytes pro Zeichen

-s

unterdrückt die Ausgabe von Zeilen, die den *Trenner* nicht enthalten

Beispiel

Mit dem Kommando

cut -d: -f1,5 /etc/passwd

werden die Benutzernamen (das 1. Feld) und die Realnamen (das 5. Feld) aller in /etc/passwd eingeragenen Systembenutzer ausgegeben.

Siehe Auch

das LunetIX Linuxhandbuch

Autor

David M. Ihnat und David

MacKenzie

Bezeichnung

df - (disk free) zeigt den freien Festplattenplatz

Syntax

df [**-aikPv**] [**-t** *fstype*] [**--all**] [**--inodes**] [**--type** *fstype*] [**--kilobytes**] [**--portability**] [*Pfad*...]

Beschreibung

df zeigt den freien Festplattenplatz für das Dateisystem in dem das Verzeichnis *Pfad* angesiedelt ist. Wenn kein Verzeichnis angegeben ist, wird der freie Platz für alle aufgesetzten Dateisysteme angezeigt. Die Angabe erfolgt in Kilobyte, wenn nicht die Umgebungsvariable **POSIXLY_CORRECT** gesetzt ist. In diesem Fall wird der freie Platz in 512--Byte Sektoren angezeigt.

Wird als *Pfad* der absolute Name der Gerätedatei eines aufgesetzten Dateisystems angegeben, so wird der freie Platz auf diesem Gerät (Partition einer Festplatte) angegeben und nicht der des Dateisystem, in dem sich die Gerätedatei befindet. Der freie Platz auf einem abgesetzten Dateisystem ist auf diese Weise nicht zu ermitteln.

-a

(all) zeigt alle Dateisysteme an, einschließlich der mit 0 Bytes Kapazität und der vom Typ **ignore** oder **auto**

-i

(inodes) zeigt die Auslastung der Inodes anstelle der Blockauslastung

-k

(kilobytes) zeigt den freien Platz in Kilobyteblöcken, auch wenn die Umgebungsvariable **POSIXLY_CORRECT** gesetzt ist

-P

(portability) erzwingt die Ausgabe in einer Zeile pro Dateisystem

-t *Fstyp*

(type) schränkt die Ausgabe auf die Dateisysteme vom Typ *Fstyp* ein

Siehe Auch

das LunetIX Linuxhandbuch

Autor

David MacKenzie

Bezeichnung

du - (disk usage) zeigt die Verteilung des belegten Plattenplatzes auf die Verzeichnisse

Syntax

du [**-abcklsxDLS**] [**--all]** [**--total**] [**--count-links**] [**--summarize**] [**--bytes**] [**--kilobytes**] [**--one-file-system**] [**--separate-dirs**] [**--dereference**] [**--dereference-args**] [*Verzeichnis*...]

Beschreibung

du zeigt den belegten Plattenplatz für das *Verzeichnis* und für alle Unterverzeichnisse (in Kilobyte). Wenn die Umgebungsvariable **POSIXLY_CORRECT** gesetzt ist, wird die Menge in 512 Byte Blöcken angegeben.

Optionen

-a

(all) zeigt auch den Platzbedarf aller Dateien

-b

(bytes) zeigt den Platzbedarf in Bytes

-c

zeigt den (summierten) Platzbedarf der in der Kommandozeile übergebenen Dateien

-k

(kilobytes) gibt den Platzbedarf in Kilobytes, auch wenn die Umgebungsvariable **POSIXLY_CORRECT** gesetzt ist

-l

zählt die Größe der (harten) Links mit, auch wenn sie dadurch doppelt vorkommen

-s

gibt nur die Summe für jedes Verzeichnis in der Kommandozeile

-x

ignoriert Verzeichnisse, die in anderen Dateisystemen liegen

-D

folgt dem Verweis auf ein anderes Verzeichnis bei einem symbolischen Link, wenn dieser als Kommandozeilenargument übergeben wird. Andere symbolische Links werden nicht dereferenziert.

-L

alle symbolischen Links werden dereferenziert, das heißt es wird der Platzbedarf des referenzierten Verzeichnisses anstelle des Linkfiles gezeigt

-S

zeigt den Platzbedarf jedes Verzeichnisses einzeln, ohne die Unterverzeichnisse

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Torbjorn Granlund und David MacKenzie

Bezeichnung

expand - ersetzt Tabulatorzeichen durch Folgen von Leerzeichen

Syntax

expand [-*Tab1* [**,** *Tab2*] [**,** ...]]] [**-t** *Tab1* [**,** *Tab2* [**,** ...]]] [**-i**] [**--tabs=** *Tab1* [**,** *Tab2* [**,** ...]]] [**--initial**] [*Datei*...]

Beschreibung

expand liest eine Textdatei und ersetzt alle Tabulatoren durch entsprechende Folgen von Leerzeichen. In der Voreinstellung werden alle Tabulatoren ersetzt und eine Tabellenbreite von 8 Zeichen angenommen. Backspace Zeichen werden unverändert ausgegeben.

Wenn anstelle eines Dateinamens ein **-**, oder gar kein Dateiname angegeben ist, wird von der Standardeingabe gelesen.

Optionen

-*Tab1*

setzt die Tabellenbreite für alle Spalten auf *Tab1* anstelle von 8

-*Tab1*, *Tab2*, ...

setzt die erste Tabellenspalte auf *Tab1*, die zweite auf *Tab2* und so weiter; wenn im Text mehr Tabulatoren auftauchen als bei der Option angegeben, werden alle folgenden Tabulatoren durch einzelne Leerzeichen ersetzt

-t *Tab1* ...

macht nichts anderes als die oben beschriebenen Optionen

-i

konvertiert nur die führenden Tabulatoren jeder Zeile in Leerzeichen

Siehe Auch

das LunetIX Linuxhandbuch

Autor

David

MacKenzie

Bezeichnung

find - sucht nach bestimmten Dateien

Syntax

find [*Verzeichnis*] [-*Option*...] [-*Test*...] [-*Aktion*...]

Beschreibung

find durchsucht eine oder mehrere Verzeichnishierarchien nach Dateien mit bestimmten Eigenschaften, und führt damit bestimmte Aktionen aus. Die Eigenschaften können durch Tests bestimmt werden.

Optionen, Tests und Aktionen können mit Operatoren zusammengefaßt werden. **find** bewertet für jede Datei in den Verzeichnishierarchien die Optionen, Tests und Aktionen von links nach rechts, bis ein falscher Wahrheitswert auftaucht oder die Kommandozeilenargumente zu Ende sind.

Das erste Argument, das mit einem **-**, einer Klammer **(**, **)**, einem Komma **,** oder einem Ausrufezeichen **!** beginnt, wird als Anfang einer Option oder Test interpretiert. Alle Argumente davor werden als Verzeichnisnamen interpretiert.

Wenn kein Verzeichnis angegeben ist, wird das aktuelle Verzeichnis genommen. Wenn keine Aktion angegeben ist, wird die Aktion **-print** ausgeführt.

Der Status von **find** ist Null, wenn alle Aktionen erfolgreich waren, im Fehlerfall ist der Status größer als Null.

Optionen

Die Optionen bestimmen das allgemeine Verhalten des Kommandos, und beziehen sich nicht auf spezielle Dateien. Die Optionen sind immer wahr.

-daystart

mißt die Zeiten für die -amin, -atime, -cmin, -ctime, -mmin und -mtime Eigenschaften vom Beginn des aktuellen Tages anstelle der letzten 24 Stunden

-depth

bearbeitet den Inhalt jedes Verzeichnisses vor dem Verzeichnis selbst

-follow

folgt den symbolischen Links; diese Option schließt **-noleaf** mit ein

-maxdepth *Ebenen*

steigt bis zu der gegebenen Zahl von Ebenen im Verzeichnisbaum auf (in der Hierarchie ab); bei 0 Ebenen werden die Tests nur auf die in der Kommandozeile übergebenen Dateien und Verzeichnisnamen angewendet

-mindepth *Ebenen*

steigt mindestens die gegebene Zahl von Ebenen im Verzeichnisbaum auf (in der Hierarchie ab); bei einer Ebene werden die in der Kommandozeile genannten Dateien und Verzeichnisnamen nicht bearbeitet

-noleaf

erzwingt die Bearbeitung aller Verzeichniseinträge; normalerweise kann davon ausgegangen werden, daß jedes Linuxverzeichnis wenigstens zwei (harte) Links enthält: das Verzeichnis **.** ist ein Link auf das Verzeichnis selbst, und jedes Unterverzeichnis enthält den Eintrag **..** als Link auf das Oberverzeichnis; wenn **find** bei der Untersuchung eines Verzeichnisses zwei Unterverzeichnisse weniger untersucht hat, als das Verzeichnis Links zählt, kann deshalb normalerweise die weitere Suche beendet werden

-version

gibt die Versionsnummer auf die Standardfehlerausgabe

-xdev

durchsucht keine Verzeichnisse in anderen Dateisystemem (auf anderen Partitionen)

Tests:

Alle numerischen Argumente können auf drei Arten angegeben werden:

+ *N*

steht für alle Zahlen größer als *N*

- *N*

steht für alle Zahlen kleiner als *N*

N

steht für genau *N*

Alle Tests werden auf die Dateien in den angegebenen Verzeichnissen einzeln angewendet. Die Tests liefern einen Wahrheitswert von 0 (Wahr), wenn der Test erfolgreich war.

Die Tests auf die erweiterten Zeitmarken (Zugriff und Erstellung) werden nur in solchen Verzeichnissen korrekt behandelt, die auf einem der neuen Linuxdateisysteme angesiedelt sind (e2fs, xiafs, new minix). Auf den anderen Dateisystemen wird nur das Datum der letzten Änderung zuverlässig getestet. Das Ergebnis der anderen Tests hängt davon ab, ob der letzte Zugriff bzw. die letzte Änderung so kurz zurückliegen, daß die veränderte I-Node noch im Arbeitsspeicher (Cache) ist. Dann können auch für die Dateien der alten Dateissysteme alle drei Zeitmarken unterschieden werden.

-amin *N*

auf die Datei ist vor *N* Minuten zugegriffen worden

-anewer *Referenzdatei*

auf die Datei ist vor weniger Zeit zugegriffen worden, als seit der letzten Veränderung der *Referenzdatei* vergangen ist; im Zusammenhang mit **-follow** tritt **-anewer** nur in Effekt, wenn **-follow** vor **-anewer** in der Kommandozeile steht

-atime *N*

auf die Datei ist vor *N**24 Stunden zugegriffen worden

-cmin *N*

der Status der Datei wurde vor *N* Minuten geändert

-cnewer *Referenzdatei*

der Status der Datei wurde vor weniger Zeit verändert, als seit der letzten Veränderung der *Referenzdatei* vergangen ist; zusammen mit **-follow** tritt **-cnewer** nur in Effekt, wenn **-follow** vor **-cnewer** in der

Kommandozeile steht

-ctime *N*

der Dateistatus wurde vor *N**24 Stunden geändert

-empty

die reguläre Datei oder das Verzeichnis ist leer

-false

ist immer falsch

-fstype *Typ*

die Datei ist in einem Dateisystem vom angegebenen *Typ*; unter anderem werden **minix**, **msdos**, **ext** und **proc** erkannt

-gid *N*

die Datei gehört der Gruppe mit der Kennzahl *N*

-group *Name*

die Datei gehört der Gruppe *Name*

-inum *IN*

die Datei belegt die Inode mit der Nummer *N*

-links *N*

die Datei hat *N* (harte) Links

-lname *Muster*

die Datei ist ein symbolischer Link auf eine Datei oder ein Verzeichnis mit einem zum *Muster* passenden Namen

-mmin *N*

der Inhalt der Datei wurde vor *N* Minuten verändert

-mtime *N*

der Inhalt der Datei wurde vor *IN**24 Stunden verändert

-name *Muster*

der Name der Datei paßt zu dem *Muster*

-newer *Referenzdatei*

die Datei ist später verändert worden als die *Referenzdatei*; zusammen mit *-follow* tritt *-newer* nur in Effekt,

wenn **-follow** vor **-newer** in der Kommandozeile steht

-nouser

die Datei gehört keinem im System eingetragenen Benutzer

-nogroup

die Datei gehört keiner im System angemeldeten Gruppe

-path *Muster*

der Pfadname der Datei paßt zum *Muster"*

-perm *Modus*

die Zugriffsrechte auf die Datei entprechen exakt dem *Modus*; der *Modus* kann als Oktalzahl oder mit den bei **chmod**(1) beschriebenen Kennungen beschrieben werden, die Kennungen werden auf Modus **000**bezogen

-perm -*Modus*

(mindestens) die Zugriffsrechte für den *Modus* sind gesetzt

-perm +*Modus*

die Zugriffsrechte entsprechen höchstens dem *Modus* (oder sind weiter eingeschränkt)

-regex *Muster*

der Pfadname paßt zu dem regulären Ausdruck *Muster*

"**size** ***N***[{**c**,**k**}]"

die Datei belegt *N* Datenblöcke zu 512 Bytes, bzw. N Bytes und N Kilobytes mit nachgestelltem **c** oder **k**

-true

ist immer wahr

-type *C*

die Datei ist vom Typ *C*; folgende Typen werden unterschieden:

b

gepufferte Gerätedatei für ein blockorientiertes Gerät

c

ungepufferte Gerätedatei für ein zeichenorientiertes Gerät

d

Verzeichnis

p

benannte Pipeline (FiFo)

f

normale Datei

l

symbolischer Link

s

Socket

-uid *N*

die Kennziffer des Eigentümers ist *N*

-used *N*

auf die Datei ist *N* Tage nach der letzten Änderung zugegriffen worden

-user *Username*

die Datei gehört dem Anwender *Username*

-xtype *C*

das gleiche wie **-type** für alle Dateien, die keine symbolischen Links sind; wenn die Datei ein symbolischer Link ist und die Option **-follow** nicht gesetzt ist, wird die Datei, auf die der Link zeigt, auf den Typ C geprüft; wenn die Option **-follow** gesetzt ist, ist der Test wahr, wenn C = **l** ist

Aktionen:

-exec *Kommando;*

führt das *Kommando* aus; die Aktion ist wahr, wenn das Kommando einen Status von Null liefert; alle auf den Kommandonamen folgenden Argumente bis zu einem Semikolon **;** werden als Kommandozeilenargumente für das Kommando interpretiert; das Semikolon kann nicht weggelassen werden, und es muß durch mindestens ein Whitespace von der letzen Option getrennt werden; die Konstruktion **{}** wird durch den Pfadnamen der Datei ersetzt; die Klammern und das Semikolon müssen in der Kommandozeile für **find** quotiert werden, damit sie nicht von der Shell bearbeitet werden

-fprint *Ausgabedatei*

schreibt den Pfadnamen der getesteten Datei in die *Ausgabedatei*; wenn die Ausgabedatei nicht existiert,

wird sie erzeugt, sonst wird sie erweitert; die Standardausgabe und die Standardfehlerausgabe werden als **/dev/stdout** und **/dev/stderr** angesprochen

-fprint0 *Ausgabedatei*

schreibt den Namen der getesteten Datei in die *Ausgabedatei* und schließt die Ausgabe mit einem Nullbyte ab, wie **-print0**

-fprintf *Ausgabedatei Format*

schreibt den Namen der getesteten Datei in die *Ausgabedatei* und benutzt dabei das *Format* mit Sonderzeichen wie bei **printf**

-ok *Kommando;*

wie **-exec**, vor der Ausführung des Kommandos wird aber noch eine Bestätigung erwartet; nur eine Eingabe, die mit einem **B** oder einem **y** beginnt, führt zur Ausführung des Kommandos

-print

gibt den vollständigen Pfadnamen der getesteten Datei auf die Standardausgabe

-print0

gibt den Pfadnamen der getesteten Datei, von einem Nullbyte abgeschlossen, auf die Standardausgabe; auf diese Weise können auch Pfadnamen korrekt weiterverarbeitet werden, die ein Zeilenende enthalten

-printf *Format*

gibt für die getestete Datei die Zeichenkette *Format* auf der Standardausgabe aus; Format kann verschiedene Sonderzeichen und Platzhalter enthalten, die von **find** bearbeitet werden:

\a

Alarmton

\b

Rückschritt

\c

Abbruch der Ausgabe

\f

Seitenvorschub

\n

Zeilenvorschub

\r

Wagenrücklauf

\t

horizontaler Tabulator

\v

vertikaler Tabulator

\\

der Backslash selbst

ein Backslash gefolgt von irgendeinem anderen Zeichen wird als normales Zeichen interpretiert und einfach ausgegeben

%%

das Prozentzeichen selbst

%a

die Zeit des letzten Zugriffs auf die Datei, in dem Format der **ctime** Funktion

%A *k*

die Zeit des letzten Zugriffs auf die Datei, in dem von *k* bestimmte Format; k hat dabei das gleiche Format wie der entprechende Parameter der **strftime** Funktion in C:

@

Sekunden seit dem 1.1.1970 0 Uhr GMT

H

Stunde (00 bis 23)

I

Stunde (01 bis 12)

k

Stunde (0 bis 23)

l

Stunde (1 bis 12)

M

Minute (00 bis 59)

p

PM oder AM

r

Zeit, 12 Stunden (hh:mm:ss: AM/PM)

S

Sekunden (00 bis 61)

T

Zeit, 24 Stunden (hh:mm:ss)

X

Zeit (H:M:S)

Z

Zeitzone, oder nichts

a

abgekürzter Wochentag

A

ausgeschriebener Wochentag

b

abgekürzter Monatsname

B

ausgeschriebener Monatsname

c

Datum und Zeit

d

Tag im Monat

D

Datum (mm/dd/yy)

h

das gleiche wie **b**

j

der Tag im Jahr

m
die Zahl des Monats
U
die Nummer der Woche, Sonntag als erster Wochentag
w
die Zahl des Wochentags
W
die Nummer der Woche, Montag als erster Wochentag
x
Datum (mm/dd/yy)
y
die letzten beiden Stellen der Jahreszahl
Y
die Jahreszahl

%b
die Dateigröße in 512 Byte Blöcken (aufgerundet)

%c
das Datum der letzten Statusänderung im Format der C **ctime** Funktion

%C*k*
das Datum der letzten Statusänderung im Format der BR strftime " Funktion; Parameter wie oben"

%d
die Höhe der Datei im Verzeichnisbaum; Null bedeutet, daß die Datei Kommandozeilenargument ist

%f
der Name der getesteten Datei, ohne Verzeichnisse

%g
der Gruppenname der getesteten Datei oder die Kennzahl, wenn die Gruppe nicht eingetragen ist

%G
die Gruppenkennzahl

%h

die Verzeichnisnamen des Pfadnamen der getesteten Datei

%H

das Kommandozeilenargument (Test), mit dem die Datei gefunden wurde

%i

die Nummer der Inode der getesteten Datei

%k

die aufgerundete Größe der getesteten Datei in Kilobytes

%l

das Objekt, auf die ein symbolischer Link zeigt; leer, wenn die getestete Datei kein symbolischer Link ist

%m

die Zugriffsrechte als Oktalzahl

%n

die Anzahl der harten Links auf die getestete Datei

%p

der Pfadname der Datei

%P

der Pfadname und das Kommandozeilenargument (Test), mit dem die Datei gefunden wurde

%s

die Größe der getesteten Datei in Bytes

%t

die Zeit der letzten Änderung, im **ctime** Format

%T*k*

die Zeit der letzten Änderung, im **strftime** Format (siehe oben)

%u

der Name des Eigentümers der getesteten Datei oder die Kennzahl, wenn der Benutzer nicht eingetragen ist

%U

die Benutzerkennzahl des Eigentümers der getesteten Datei

-prune

wahr, wenn die Option **-depth** gesetzt ist; sonst falsch

-ls

zeigt das Verzeichnis in dem die getestete Datei gefunden wurde mit **ls -dils** an

Operatoren:

Die Optionen, Tests und Aktionen können mit Operatoren verknüpft werden. Die Bearbeitung erfolgt prinzipiell von links nach rechts.

(*Ausdruck* **)**

die Klammern fassen den *Ausdruck* zu einer Operation zusammen

! *Ausdruck*

:RI "ist wahr, wenn der " Ausdruck " falsch ist"

-not *Ausdruck*

ist ebenfalls wahr, wenn der *Ausdruck* falsch ist

Ausdruck1 Ausdruck2

UND Verknüpfung; wenn *Ausdruck1* wahr ist, wird *Ausdruck2* bewertet (ausgeführt)

Ausdruck1 **-a** *Ausdruck2*

auch eine UND Verknüpfung

Ausdruck1 **-and** *Ausdruck2*

auch eine UND Verknüpfung

Ausdruck1 **-o** *Ausdruck2*

ODER Verknüpfung; *Ausdruck2* wird bewertet (ausgeführt), wenn *Ausdruck1* falsch ist

Ausdruck1 **-or** *Ausdruck2*

auch eine ODER Verknüpfung

Ausdruck1 **,** *Ausdruck2*

Liste; beide Ausdrücke werden immer bewertet (ausgeführt); der Wahrheitswert des gesamten Ausdrucks entspricht dem von Ausdruck2

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Eric Decker, David MacKenzie, Jay Plett

und Tim Wood

Bezeichnung

fmt - einfacher Textformatierer

Syntax

fmt [-*DIGITS*] [*OPTION*]... [*DATEI*]...

Beschreibung

Formatiert jeden Absatz der Datei(en) um und schreibt sie auf die Standard-Ausgabe. Wenn keine Datei angegeben wurde oder DATEI ein "-" ist, wird stattdessen die Standard-Eingabe gelesen.

Zwingende Argumente für lange Optionen sind auch für kurze Optionen zwingend

-c, **--crown-margin**

Einrückung der ersten zwei Zeilen beibehalten

-p, **--prefix**=*STRING*

Nur die Zeilen, die STRING als Präfix haben, werden zusammengezogen

-s, **--split-only**

Lange Zeilen aufspalten aber nicht auffüllen

-t, **--tagged-paragraph**

Einrückung der ersten Zeile unterschiedlich zu der der zweiten

-u, **--uniform-spacing**

Ein Leerzeichen zwischen Worten, zwei zwischen Sätzen

-w, **--width**=*NUMBER*

Maximale Zeilenbreite (Voreingestellt sind 75 Zeichen)

--help

Zeigt kurzen Hilfetext und beendet dann das Programm

--version

Zeigt Versionsinformation und beendet dann das Programm

Bei der Angabe **-wNUMMER**, darf der Buchstabe w weggelassen werden.

Copyright

Name

gpasswd - Verwaltung der Datei /etc/group

Syntax

gpasswd ***Gruppe***
gpasswd -a *User Gruppe*
gpasswd -d *User Gruppe*
gpasswd -R *Gruppe*
gpasswd -r *Gruppe*
gpasswd [**-A** *User*] [**-M** *User,...*] *Gruppe*

Beschreibung

gpasswd wird benutzt, um die Dateien /etc/group und /etc/gshadow zu verwalten und zu bearbeiten. Jede Gruppe des Systems kann einen Verwalter und ein Passwort haben. Der Systemverwalter kann die **-A** Option benutzen, um Gruppenverwalter zu definieren und die **-M** Option um Gruppenmitglieder aufzunehmen. Er hat alle Rechte auf alle Gruppen.

Der Gruppenverwalter einer Gruppe kann mit der Option **-a** (add) Gruppenmitglieder aufnehmen, mit **-d** (delete) Gruppenmitglieder aus der Gruppe ausschließen. Mit **-r** kann er das Gruppenpasswort löschen. Wenn kein Gruppenpasswort gesetzt ist, können nur eingetragene Gruppenmitglieder mit **newgrp** diese Gruppe zu ihrer Primärgruppe machen. Die Option **-R** macht einen Zugriff auf die Gruppe durch **newgrp** ganz unmöglich.

Wird **gpasswd** vom Gruppenadministrator nur mit einem Gruppennamen aufgerufen, so kann er damit das Gruppenpasswort ändern. Wenn ein Passwort gesetzt ist, können eingetragene Mitglieder immer noch mit **newgrp** arbeiten, ohne ein Passwort eingeben zu müssen, Nichtmitglieder brauchen das Passwort.

Dateien

/etc/group - Gruppeninformationen
/etc/gshadow - Sichere Gruppeninformationen

Siehe auch

newgrp(1) , **groupadd**(8) , **groupdel**(8) , **groupmod**(8) , **grpck**(8)

Bezeichnung

grep - durchsucht Dateien nach Ausdrücken

Syntax

grep [**-CVbchilnsvwx**] [*-Anzahl*] [**-AB** *Anzahl*] [[**-e**] *Ausdruck* |**-f** *Datei*] [*Datei*...]

Beschreibung

grep durchsucht die angegebenen *Dateien* (oder die Standardeingabe) nach einem *Ausdruck* und gibt die entsprechenden Zeilen aus. Der Status von grep ist 0, wenn der *Ausdruck* gefunden wurde und sonst 1.

Als *Ausdruck* akzeptiert grep reguläre Ausdrücke mit den folgenden Steuerzeichen:

c
: ein einzelner Buchstabe paßt auf sich selbst

.
: ein Punkt paßt auf jeden Buchstaben außer auf das Zeilenende

\\?
: das dem Operator **\\?** vorangehende Muster kann null oder einmal vorkommen

***
: das dem Operator ***** vorangehende Muster kann 0 mal oder öfter vorkommen

\\+
: das dem Operator **\\+** vorangehende Muster kann einmal oder öfter vorkommen

\\|
: die durch den Operator **\\|** verbundenden Argumente werden **oder** verknüpft

^

(Caret) paßt auf den Zeilenanfang

$

paßt auf das Zeilenende

\\<

paßt auf den Wortanfang

\\>

paßt auf das Wortende

[*Buchstaben***]**

paßt auf alle *Buchstaben*; dabei können einzelne Buchstaben, aber auch Bereiche in der Form *von-bis* angegeben werden; wenn der erste Buchstabe nach **[ein** ^ ist, paßt der Ausdruck auf alle Buchstaben, außer den Aufgeführten

\\(\\)

die Klammern fassen Ausdrücke zusammen; außerdem wird der auf den in Klammern eingeschlossene Teil des Musters passende Text markiert und mit einem folgenden ***N*** Ausdruck referenziert (Tag)

N

referenziert die auf das in der *N*-ten runden Klammern eingeschlossene Muster passende Zeichenkette.

\\

jedes der Sonderzeichen kann, durch einen \\ (Backslash) eingeleitet, sich selbst suchen

\\b

paßt auf kein Zeichen, sondern auf den Anfang oder das Ende eines Wortes

\\B

symbolisiert den Raum innerhalb eines Wortes

\\w

paßt auf alle alphanumerischen Zeichen [A-Za-z0-9]

\\W

paßt auf alle nichtalphanumerischen Zeichen [^A-Za-z0-9]

Die Rangfolge der Operatoren ist (von der höchsten zur niedrigsten):**(**, **)**, **?**, *****, **+** und |. Die anderen Operatoren sind mit den anderen Buchstaben gleichrangig.

Optionen

-A *Anzahl*

gibt *Anzahl* Zeilen Kontext **nach** jeder gefundenden Zeile aus

-B *Anzahl*

gibt *Anzahl* Zeilen Kontext **vor** jeder gefundenden Zeile aus

-C

gibt *2* Zeilen Kontext **vor und nach** jeder gefundenden Zeile aus

-*Anzahl*

gibt *Anzahl* Zeilen Kontext **vor und nach** jeder gefundenden Zeile aus

-V

gibt die Versionsnummer auf die Standardfehlerausgabe

-b

gibt die Position jeder gefundenen Stelle mit aus

-c

gibt nur die Gesamtzahl der gefundenen Stellen aus

-e *Ausdruck*

sucht nach *Ausdruck*

-f *Datei*

Datei enthält die Ausdrücke, nach denen gesucht werden soll.

-h

unterdrückt die Dateinamen vor jeder Fundstelle

-i

ignoriert Groß und Kleinschreibung

-l

gibt nur die Dateinamen mit Fundstellen aus

-n

gibt die Zeilennummer zu jeder Fundstelle aus

-r oder **--recursive**

durchsucht rekursiv ganze Verzeichnisbäume

-s

(silent) keine Ausgabe außer Fehlermeldungen

-v

gibt nur Zeilen aus, die den *Ausdruck* nicht enthalten

-w

gibt nur Zeilen aus, in denen der *Ausdruck* als komplettes Wort vorkommt

-x

gibt nur Zeilen aus, die den *Ausdruck* als ganze Zeile enthalten

Beispiel

Mit dem Kommando

grep '\(.... \).+\1' Handbuch.tex

wird in der Datei Handbuch.tex nach Zeilen gesucht, in denen ein Wort mit vier Buchstaben doppelt, aber nicht unmittelbar nach dem ersten vorkommt.

Siehe Auch

egrep(1) und das LunetIX Linuxhandbuch

Autor

Mike Haertel, James A. Woods und David Olson

Name

groupadd - Anlegen einer neuen Gruppe

Syntax

groupadd [**-g** ***GID*** [**-o**]] *Gruppenname*

Beschreibung

Das **groupadd** Kommando legt anhand den übergebenen Kommandozeilen-Optionen und der voreingestellten Werte des Systems eine neue Gruppe an. Die neue Gruppe wird in die nötigen Systemdateien eingetragen. Gültige Optionen sind:

-g ***GID***

> Die numerische GruppenID der neuen Gruppe. Dieser Wert muß eindeutig sein, außer die Option **-o** ist zusätzlich angegeben. Der Wert darf nicht negativ sein. Voreingestellt ist die kleinste ID, deren Wert größer als 99 und größer als alle bestehenden GruppenIDs ist. Werte zwischen 0 und 99 sind typischerweise für System Gruppen reserviert.

Dateien

/etc/group - Gruppen Account Informationen
/etc/gshadow - Sichere Gruppen Account Informationen

Siehe auch

chfn(1) , **chsh**(1) , **useradd**(8) , **userdel**(8) , **usermod**(8) , **passwd**(1) , **groupdel**(8) , **groupmod**(8)

Autor

Julianne Frances Haugh (jfh@bga.com)

Name

groupdel - Löschen einer Gruppe

Syntax

groupdel *Gruppenname*

Beschreibung

Das **groupdel** Kommando verändert die notwendigen Systemdateien dahingehend, daß alle Einträge, die sich auf die genannte Gruppe beziehen gelöscht werden. Die genannte Gruppe muß existieren.

Um zu vermeiden, daß im System noch Dateien verbleiben, die dieser Gruppe angehören, muß von Hand danach gesucht werden.

Einschränkungen

Gruppen, die die Logingruppe eines oder mehrerer User sind können mit **groupdel** nicht gelöscht werden.

Dateien

/etc/group - Gruppen Account Informationen
/etc/gshadow - Sichere Gruppen Account Informationen

Siehe auch

chfn(1) , **chsh**(1) , **useradd**(8) , **userdel**(8) , **usermod**(8) , **passwd**(1) , **groupadd**(8) , **groupmod**(8)

Autor

Julianne Frances Haugh (jfh@bga.com)

Name

groupmod - Ändern einer Gruppe

Syntax

groupmod [**-g** ***GID*** [**-o**]] [**-n** ***Gruppenname***] *Gruppe*

Beschreibung

Das **groupmod** Kommando modifiziert die nötigen Systemdateien dahingehend, daß die auf der Kommandozeile angegebenen Veränderungen für eine Gruppe eingetragen werden. Gültige Optionen sind

-g ***GID***

Die numerische GruppenID der neuen Gruppe. Dieser Wert muß eindeutig sein, außer die Option **-o** ist zusätzlich angegeben. Der Wert darf nicht negativ sein. Voreingestellt ist die kleinste ID, deren Wert größer als 99 und größer als alle bestehenden GruppenIDs ist. Werte zwischen 0 und 99 sind typischerweise für System Gruppen reserviert. Es werden nicht automatisch die Gruppenzugehörigkeiten von Dateien geändert, die dieser Gruppe zugeteilt waren. Das muß von Hand erledigt werden.

-n ***Gruppenname***

Der Name der angegebenen Gruppe wird von *Gruppe* zu *Gruppenname* geändert.

Dateien

/etc/group - Gruppen Account Informationen
/etc/gshadow - Sichere Gruppen Account Informationen

Siehe auch

chfn(1) , **chsh**(1) , **useradd**(8) , **userdel**(8) , **usermod**(8) , **passwd**(1) , **groupadd**(8) , **groupdel**(8)

Autor

Julianne Frances Haugh (jfh@bga.com)

Bezeichnung

groups - zeigt alle Gruppen, denen der Benutzer angehört

Syntax

groups [*Benutzer*...]

Beschreibung

groups gibt die Namen aller Gruppen vom *Benutzer* aus. Wird kein Name angegeben, werden die Gruppen des aktuellen Prozesses gezeigt.

Siehe Auch

groups ist ein Shellscript und benutzt **id**

Name

grpck - Überprüfung der Integrität der Gruppen-Dateien

Syntax

grpck [**-r**] [*group shadow*]

Beschreibung

grpck überprüft die Integrität der System-Authentifizierungsinformation. Alle Einträge in */etc/group* und */etc/gshadow* werden auf ihr korrektes Format hin überprüft und es wird sichergestellt, daß jedes Feld gültige Werte besitzt. Der Anwender (der Systemverwalter) wird dazu aufgerufen, Einträge zu löschen, die ungültige Formate vorweisen oder andere - nicht automatisch reparierbare - Fehler vorweisen.

Die Überprüfung stellt sicher, daß jeder Eintrag folgende Eigenschaften besitzt:

- Die richtige Anzahl von Feldern
- Einen eindeutigen Gruppennamen
- Eine gültige Liste der Mitglieder und Administratoren

Die Überprüfung der richtigen Anzahl von Feldern und einheitlichen Usernamen sind schwerwiegend. Wenn ein Eintrag eine falsche Anzahl Felder vorweist, wird der Anwender aufgefordert, den Eintrag zu löschen. Antwortet der Anwender nicht mit ja, so wird die weitere Überprüfung abgebrochen. Ein Eintrag mit doppeltem Gruppennamen führt auch zu so einer Aufforderung zum Löschen. Wird diese mit nein beantwortet werden aber trotzdem alle weiteren Überprüfungen vorgenommen. Alle anderen Fehler werden durch Warnungen angezeigt und der Systemverwalter wird aufgefordert, sie mit Hilfe des Programms **groupmod** zu ändern.

Optionen

Voreingestelltermaßen arbeitet **grpck** mit den Dateien */etc/group* und */etc/gshadow*. Der Anwender kann mit den *group*- und *shadow*-Parametern alternative Dateien angeben. Zusätzlich kann mit der Option **-r** ein ReadOnly Modus erzwungen werden, in dem **grpck** keine Veränderungen an den Dateien vornehmen kann. Das entspricht der grundsätzlichen Antwort "no" auf alle Fragen.

Dateien

/etc/group - Gruppeninformationen
/etc/gshadow - Verschlüsselte Password- und Gruppenadministratorinformationen
/etc/passwd - Useraccount Informationen

Siehe auch

groupmod(8) , **group**(5) , **passwd**(5) , **shadow**(5)

Diagnose

Das **grpck** Kommando gibt folgende Rückgabewerte zurück:

0
: Erfolgreiche Ausführung

1
: Syntax Error

2
: Ein oder mehrere schlechte Gruppen-Einträge

3
: Kann Gruppen-Dateien nicht öffnen

4
: Kann Gruppen-Dateien nicht sperren

Kann Gruppen-Dateien nicht sichern

Autor

Julianne Frances Haugh (jfh@bga.com)

Bezeichnung

gzip - komprimiert Dateien

Syntax

gzip [**-cdfhLrtvV19**] [*Datei*...]

Beschreibung

gzip komprimiert Dateien mit dem LZ77 Lempel-Ziv Algorithmus. **gzip** erzielt erheblich bessere Kompressionsraten als das mit dem LZW Algorithmus arbeitende **compress** Programm. Weil es sich ansonsten sehr ähnlich verhält, ist abzusehen daß es **compress** als Standardpacker im Bereich der freien Software verdrängen wird. Mit **gzip** können auch Dateien ausgepackt werden, die mit **compress** gepackt wurden.

gzip ist kein Archivpacker wie **lharc**, **arj** oder **pkzip**.

gzip komprimiert einzelne Dateien, unabhängig davon ob die resultierende Datei tatsächlich kleiner ist, und ersetzt die Urdatei durch die komprimierte, indem es an den Dateinamen die Endung **.gz** anhängt. Dabei bleiben die Zugriffsrechte und das Erstellungsdatum der Urdatei erhalten.

Wenn der Dateiname durch Anhängen der Endung unzulässig lang würde, schneidet **gzip** automatisch die erforderliche Anzahl Zeichen vom Dateinamen ab, speichert aber den vollständigen Namen zur Restaurierung beim Auspacken in der Datei ab.

Wenn **gzip** ohne Dateinamen aufgerufen wird, liest es von der Standardeingabe und schreibt auf die Standardausgabe.

Wie bei **compress** kann auch **gzip** auf andere Namen gelinkt werden um bestimmte Aufgaben zu erfüllen.

Unter dem namen **gunzip** arbeitet es wie **gzip**-d**,** packt also komprimierte Dateien aus. Dabei kann es auch die von **compress** (ab Version 3.0) gepackten Dateien bearbeiten. **gunzip** erwartet die Endung **.z** oder **.Z** an dem Dateinamen. Nach dem Auspacken bleiben die Zugriffsrechte und das Erstellungsdatum der Datei erhalten.

zcat schreibt die entkomprimierte Datei auf die Standardausgabe und läßt die komprimierte Datei unberührt.

Optionen

-c

schreibt die (ent)komprimierte *Datei* auf die Standardausgabe, anstatt die Datei zu ersetzen

-d

(decompress) dekomprimiert die *Datei*

-f

(force) ersetzt bestehende Dateien mit Endung **.z**; normalerweise fragt **gzip** vor dem Überschreiben solcher Dateien nach

-h

(help) gibt eine Kurzhilfe zum Programm aus

-L

(license) gibt eine Kurzfassung des Lizenztextes aus

-r

(recursive) packt alle Dateien in den angegebenen Unterverzeichnissen

-t

(test) prüft die Integrität der angegebenen *Datei*

-v

(verbose) gibt den Kompressionsfaktor für jede *Datei* aus

-V

(Version) gibt die Versionsnummer des Programms aus

-Ziffer

bestimmt mit einer Ziffer von 1 bis 9 die Kompressionstiefe; 1 bedeutet schnell und schlecht komprimiert, 9 bedeutet langsam und optimal komprimiert

Siehe Auch

compress(1) , **tar**(1) und das LunetIX Linuxhandbuch

Autor

Jean-Loup Gailly

Bezeichnung

head schreibt den Anfang einer Datei auf die Standardausgabe

Syntax

head [**-c** *Anzahl* [**bkm**]] [**-n** *Anzahl*] [**-qv**] [**--bytes=** *Anzahl* [**bkm**]] [**--lines=** *Anzahl*] [**--quiet**] [**--silent**] [**--verbose**] [*Datei*...]

head [*-nr*] [**-bcklmqv**] [*Datei*...]

Beschreibung

head schreibt die ersten (10) Zeilen von der *Datei* auf den Bildschirm. Wenn keine *Datei* oder - angegeben wird, liest **head** von der Standardeingabe. Wird mehr als eine *Datei* angegeben, so wird der Dateiname, in ==> und <== eingeschlossen, der Ausgabe vorangestellt.

Optionen

-c *Anzahl*

gibt die angegebene *Anzahl* Bytes aus. Optional kann die Blockgröße durch einen der Zahl folgenden Buchstaben verändert werden:

b

Blocks zu 512 Byte

k

Blocks zu 1 Kilobyte

m

Blocks zu 1 Megabyte

-l *Anzahl*

gibt die ersten *Anzahl* Zeilen aus

-q

(quiet) unterdrückt die Ausgabe des Dateinamen

-v

(verbose) gibt die Dateinamen immer aus

Siehe Auch

das LunetIX Linuxhandbuch

Autor

David MacKenzie

Bezeichnung

id - gibt die reale und die effektive User ID und Gruppen ID aus

Syntax

id [**-gnruG**] [**--group**] [**--name**] [**--real**] [**--user**] [**--groups**] [*Username*]

Beschreibung

Das **id** Kommando zeigt die reale Benutzerkennzahl (**BUID**) mit dem Benutzernamen und alle Gruppen, in denen der Anwender eingetragen ist, mit ihren Kennzahlen (**GID**)s und Namen an. Wenn die effektive Benutzerkennung nicht der realen entspricht, wird die effektive Benutzerkennung ebenfalls angezeigt.

Optionen

-g

gibt nur die Gruppen ID aus

-n

gibt den User- bzw. den Gruppennamen anstelle der ID (nur in Verbindung mit **-u**, **-g** oder **-G**)

-r

gibt die reale anstelle der effektiven User- bzw. Gruppen ID aus (nur in Verbindung mit **-u**, **-g** oder **-G**)

-u

gibt nur die User ID aus

-G

gibt die ID's aller Gruppen von User aus

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Arnold Robbins und

David MacKenzie

Bezeichnung

join - verknüpft zwei Dateien nach Schlüsselfeldern

Syntax

join [**-a 1**|**2**] [**-v 1**|**2**] [**-e** *Zeichenkette*] [**-o** *Feldliste*...] [**-t** *Buchstabe*] [**-j** [**1**|**2**] *Feldnr*] [**-1** *Feldnr*] [**-2** *Feldnr*] *Datei1* Datei2

Beschreibung

join verknüpft zwei (alphabetisch) sortierte Dateien, indem je zwei Zeilen mit identischen Schlüsselfeldern zu einer Ausgabezeile verbunden werden.

Die Schlüsselfelder sind durch Leerzeichen voneinander getrennt. Führende Leerzeichen werden ignoriert. Wenn nicht anders angegeben, ist das erste Feld einer jeden Zeile Schlüsselfeld. Die Ausgabefelder sind ebenfalls durch Leerzeichen voneinander getrennt. Die Ausgabe besteht aus dem Schlüsselfeld, gefolgt von den übrigen Feldern der Datei1 und schließlich aller Felder der passenden Zeilen von Datei2 ohne das Schlüsselfeld.

Optionen

-a *Dateinummer*

fügt in die Ausgabe eine Leerzeile ein, wenn eine Zeile aus *Dateinummer* (1 oder 2) kein Gegenstück hat.

-e *Zeichenkette*

ersetzt fehlende Eingabefelder in der Ausgabe durch die *Zeichenkette*

-1 *Feldnr*

benutzt in Datei1 *Feldnr* als Schlüsselfeld

-2 *Feldnr*

benutzt in Datei2 *Feldnr* als Schlüsselfeld

-j *Feldnr*

benutzt *Feldnr* als Schlüsselfeld

-o *Feldliste*

stellt die Ausgabezeilen anhand der *Feldliste* zusammen. Ein Eintrag in der *Feldliste* besteht aus einer *Dateinummer*, einem Punkt und einer *Feldnr*. Beliebig viele solcher Paare *Dateinummer***.***Feldnr* können, durch Komma oder Leerzeichen getrennt, in der *Feldliste* stehen.

-t *Buchstabe*

verwendet *Buchstabe* als Feldtrenner

-v *Dateinummer*

gibt nur die Zeilen aus *Dateinummer* aus, die kein Gegenstück haben.

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Mike Haertel

Bezeichnung

kill - beendet einen Prozess

Syntax

kill [-*Signr*] *Prozeßnr*

Beschreibung

kill wird benutzt, um außer Kontrolle geratene (,,aufgehängte") Prozesse, die sich nicht mehr auf normale Art beenden lassen, zu terminieren (beenden). **kill** sendet dazu das Signal *Signr* an den Prozeß *Prozeßnr*. Standardwert ist **SIGTERM** (15) zum terminieren des Prozesses. Es können aber auch beliebige andere Signale gesendet werden. Weil das Signal **SIGTERM** nicht von allen Programmen bearbeitet wird, wird ein Prozeß manchmal erst mit dem Signal **SIGKILL**(9) vom Kernel beendet. Der ,,normalen" Terminierung mit **SIGTERM** ist aber der Vorzug zu geben, weil dadurch dem Prozeß noch die Möglichkeit gegeben wird, die Bühne geordnet zu Verlassen. Es können nur die eigenen Prozesse beendet werden.

In der **bash** ist ein **kill** Kommando eingebaut, daß dieses externe Programm verdeckt, wenn nicht ausdrücklich mit dem **command** Shellkommando das externe Programm aufgerufen wird.

Optionen

-Signr

sendet *Signr* anstelle von **SIGTERM** (15)

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Peter

MacDonald

Bezeichnung

ln - (link) erzeugt einen Verzeichniseintrag einer existierenden Datei unter anderem Namen

Syntax

ln [*Optionen*] *Quelle* [*Ziel*]

ln [*Optionen*] *Quelle* ... Zielverzeichnis

Beschreibung

Jede Datei wird bei ihrer Erzeugung mit ihrem Namen in ein Verzeichnis eingetragen. Dieser Eintrag enthält außerdem einen Verweis auf eine Inode, in der die Zugriffsrechte auf die Datei, der Dateityp und gegebenenfalls die Nummern der belegten Datenblöcke eingetragen sind.

Mit dem **ln** Kommando wird ein neuer Eintrag in einem Verzeichnis angelegt, der auf die Inode einer existierenden Datei zeigt. Diese Art Link wird als **Hardlink** bezeichnet. Weil die Zugriffsrechte auf die Datei in der Inode bestimmt werden, sind die Zugriffsrechte auf alle Links einer Datei gleich.

Hardlinks können nur auf dem Datenträger angelegt werden, auf dem sich die Datei (und damit die Inode) selbst befindet. Um Links über die Dateisystemgrenzen hinweg anlegen zu können, bietet Linux die Möglichkeit **symbolischer Links**. In diesen Links ist der absolute Pfad gespeichert, auf dem die gelinkte Datei gefunden werden kann. Ein Zugriff auf diese Datei wird dann vom Betriebssystem automatisch auf die gelinkte Datei umgelenkt.

Ist das letzte Argument des Aufrufs ein existierendes Verzeichnis, so werden alle als *Quelle* aufgelisteten Dateien mit entsprechenden Namen im *Zielverzeichnis* verbunden. Wird nur eine einzige *Quelle* benannt, so wird ein Link unter diesem Namen im aktuellen Verzeichnis angelegt. Normalerweise löscht **ln** keine existierenden Dateien. Es werden standardmäßig ,,hardlinks" angelegt. Links auf Verzeichnisse oder auf Dateien in anderen Dateisystemen

können nur mit symbolischen Links realisiert werden.

Gelegentlich verändert sich das Verhalten eines Programms, wenn es durch einen Link unter einem anderen Namen aufgerufen wird. (Das funktioniert natürlich nur, wenn diese Änderung im Programm vorgesehen ist.)

Optionen

-b

sichert Dateien, anstatt sie zu löschen (mit Option -f)

-d

ermöglicht Hardlinks auf Verzeichnisse

-f

löscht bestehende Dateien

-i

fragt vor dem Löschen nach Bestätigung

-s

macht symbolische Links anstelle von harten

-v

gibt die Dateinamen auf den Bildschirm

-S *Endung*

setzt die Endung für die Sicherung von Dateien auf *Endung*. Standardwert ist **~**. Die Endung kann auch mit der Umgebungsvariablen **SIMPLE_BACKUP_SUFFIX** bestimmt werden. Die Option -S hat Vorrang vor der Umgebungsvariablen.

-V {**numbered**, **existing**, **simple**}

bestimmt die Art der Sicherungskopien. Die Art der Sicherung kann auch mit der Umgebungsvariablen **VERSION_CONTROL** bestimmt werden. Die Option **-V** hat auch hier die höhere Priorität. Die Optionen bedeuten hierbei:

numbered

macht immer nummerierte Backups

existing

macht nummerierte Backups nur für bereits nummerierte Dateien

simple

macht immer nur einfache Backups

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Mike Parker und David MacKenzie

Bezeichnung

ls - (list) zeigt den Inhalt eines Verzeichnisses

Syntax

ls [**-abcdgiklmnpqrstuxABCFLNQRSUX1**] [**-w** *Spalten*] [**-T** *Spalten*] [**-I** *Muster*] [**--all**] [**--escape**] [**--directory**] [**--inode**] [**--kilobytes**] [**--numeric-uid-gid**] [**--hide-control-chars**] [**--reverse**] [**--size**] [**--width=** *Spalten*] [**--tabsize=** *Spalten*] [**--almost-all**] [**--ignore-backups**] [**--classify**] [**--file-type**] [**--ignore=** *Muster*] [**--dereference**] [**--literal**] [**--quote-name**] [**--recursive**] [**--sort**={**none**,**time**,**size**,**extension**}] [**--format**={**long**,**verbose**,**commas**,**across**, **vertical**,**single-column**}] [**--time**={**atime**,**access**,**use**,**ctime**,**status**}] [*Pfad*...]

Beschreibung

ls gibt den Inhalt der Verzeichnisse des Dateisystems an.

Das Standardausgabeformat von **ls** hängt vom Typ des Ausgabegerätes ab. Auf einem Terminal ist die mehrspaltige Ausgabe das Standardformat. In allen anderen Fällen wird die Ausgabe einspaltig ausgeführt.

Das Verhalten des **ls** Kommandos läßt sich nicht mehr durch Umbenennen in **ll dir vdir** etc. verändern. Stattdessen sind die Kommandos **dir** und **vdir** als separate Binärdateien mit entsprechenden Standardformaten verfügbar.

Optionen

-a

zeigt alle Dateien im Verzeichnis, auch die deren Name mit **.** beginnt

-b

zeigt nichtdruckbare Zeichen in Dateinamen als ,,Backslash Sequenz" mit alphabetischen oder oktalen Werten wie sie in C üblich sind

-c

sortiert die Dateien nach der Zeit der letzten Statusveränderung

-d

zeigt Unterverzeichnisse wie normale Dateien anstelle ihres Inhaltes

-i

zeigt die Nummer der Inode zu jeder Datei

-k

die Dateigröße wird in Kilobytes angegeben, auch wenn **POSIXLY_CORRECT** gesetzt ist

-l

außer dem Namen werden der Typ, die Rechte, die Anzahl der Hardlinks, der Besitzer, die Gruppe, die Größe und die Zeitmarke angezeigt

-m

gibt die Dateinamen in einer Reihe, getrennt durch Kommas, aus

-n

gibt die Benutzer und Gruppen mit ihren IDs anstelle der Namen aus

-q

gibt Fragezeichen anstelle von nicht druckbaren Zeichen in Dateinamen

-r

zeigt das Verzeichnis in umgekehrter Reihenfolge

-s

zeigt die Größe der Dateien in Kilobytes. Wenn **POSIXLY_CORRECT** gesetzt ist, wird die Größe in Blöcken zu 512 Bytes angezeigt

-t

sortiert nach Zeit anstelle des Namens

-u

sortiert nach letzter Zugriffszeit anstelle der Änderungszeitw (zusammen mit Option -t)

-x
sortiert in horizontaler Richtung
-A
zeigt alle Dateien außer **.** und **..**
-B
ignoriert Backups (mit Endung ~)
-C
listet in vertikal sortierten Spalten
-F
hängt verschiedene Symbole an die Dateinamen:
*****
steht hinter ausführbaren Dateien
/
steht hinter Verzeichnissen
@
markiert symbolische Links
|
markiert FiFo's
=
markiert sockets
Alles andere sind reguläre Dateien
-L
zeigt den Inhalt der symbolisch gelinkten Verzeichnisse anstelle des Linkfiles
-N
gibt Dateinamen ohne Quotes aus
-Q
gibt Dateinamen in Quotes aus
-R
zeigt rekursiv den Inhalt aller Unterverzeichnisse

-S
: sortiert nach Größe

-U
: unsortiert

-X
: sortiert nach Endung

-1
: einspaltig

-w *Spalten*
: Bildschirmbreite in *Spalten*

-T *Spalten*
: Tabulatorbreite in *Spalten*

-I *Muster*
: ignoriert Dateien mit *Muster* im Namen

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Richard Stallman

und David MacKenzie

Name

man - Programm zum Einsehen der Online-Manuale

Syntax

man [**-acdhwutZV**] [**-m** *System*[,...]] [**-L** *locale*] [**-p** *Zeichenkette*] [**-M** *Pfad*] [**-P** *Pager*] [**-r** *Prompt*] [**-T** *Format*] [**-S** *Liste*] [**-e** *Erweiterung*] [[*Abschnitt*] *Seite* ...] ...
man -l [**-tZ**] [**-p** *Zeichenkette*] [**-P** *Pager*] [**-r** *Prompt*] [**-T** *Format*] *Datei ...*
man -k *Schlüsselwort ...*
man -f *Seite ...*

Beschreibung

man ist der Manualbrowser des Systems. Jedes Argument *Seite* ist normalerweise der Name eines Programmes oder einer Funktion. Gefunden und angezeigt wird die *Manualseite,* die auf alle Argumente paßt. Wenn ein *Abschnitt* angegeben wird, sucht **man** nur in diesem *Abschnitt* der Manualseiten. Ohne Angabe eine explizite Angabe werden alle verfügbaren *Abschnitte* in einer vorher definierten Reihenfolge durchsucht. Wenn die *Seite* in mehreren *Abschnitten* vorkommt, wird nur die jeweils erste *Seite* angezeigt, die gefunden wird.

Die folgende Tabelle zeigt die Nummern der *Abschnitte* der Manualseiten gefolgt vom Typ der dort zu findenden Seiten.

1 Ausführbare Programme oder Shellbefehle
2 Systemaufrufe (Kernelfunktionen)
3 Bibliotheksaufrufe (Funktionen in System-Bibliotheken)
4 Spezielle Dateien (gewöhnlich in */dev*)
5 Dateiformate und Konventionen, z. B. */etc/passwd*

6 Spiele

7 Makropakete und Konventionen, z. B. **man**(7) , **groff**(7)

8 Systemadministrationsbefehle (in der Regel nur für root)

9 Kernelroutinen [Nicht Standard]

n neu [veraltet]

l lokal [veraltet]

p öffentlich [veraltet]

o alt [veraltet]

Eine Manualseite besteht aus mehreren Teilen. Die üblichen Bezeichnungen sind u. a. **NAME**, **SYNTAX**, **BESCHREIBUNG**, **OPTIONEN**, **DATEIEN**, **SIEHE AUCH**, **FEHLER**, und **AUTOR**.

Die folgenden Konventionen gelten für den Abschnitt **SYNTAX** und können für andere Abschnitte als Anleitung benutzt werden.

bold text	Literale Angaben wie in der Anzeige.
italic text	Ersetzen durch passendes Argument.
[**-abc**]	Ein oder mehrere Argumente innerhalb der [ ] sind optional.
-a|**-b**	Optionen, die durch | abgegrenzt sind, können nicht zusammen benutzt werden.
Argument **...**	*Argument* kann wiederholt werden.
[*Ausdruck*] **...**	gesamter *Ausdruck* innerhalb [ ] kann wiederholt werden.

Die Befehls- oder Funktionsbeispiele sind Muster, die auf alle möglichen Aufrufe passen sollten. In manchen Fällen ist es ratsam, die verschiedenen sich ausschließenden Aufrufe zu illustrieren, wie es im **SYNTAX** Abschnitt dieser Manualseite gezeigt ist.

Beispiele

man *ls*

zeigt die Manualseite für das Programm *ls* an.

man -a *intro*

zeigt alle vorhandenen *intro* Manualseiten an. Mit (q)uit ist es möglich, das aufeinanderfolgendende Anzeigen der Seiten abzubrechen oder mit (s)kip eine Seite zu überspringen.

man -t *alias* | *lpr -Pps*

Formatiert die Manualseite, die sich auf alias bezieht, in das Default **troff** oder **groff** Format und schickt es an den Drucker *ps*. Die Defaultausgabe für **groff** ist Postscript. Das Default für **troff** - ditroff, benötigt weitere Bearbeitung durch grops, bevor es von einem Postscriptdrucker gedruckt werden kann. **man --help** sollte zu bestimmen helfen, welcher Prozessor an die **-t** Option gebunden ist.

man -l -T*dvi ./foo.1x.gz > ./foo.1x.dvi*

Dieser Befehl formatiert die komprimierte nroff-Manualseite *./foo.1x.gz* in eine **device independent (dvi)** Datei. Diese Datei wird zuerst dekomprimiert, bevor sie durch passende Filter und Programme bearbeitet wird. Die Umlenkung ist notwendig, da die **-T** Option die Ausgabe unformatiert an die **Standardausgabe** schickt. Die Ausgabe kann mit einem Programm wie **xdvi** betrachtet oder mit einem Programm wie **dvips** in Postscript weiterverarbeitet werden.

man -k *printf*

Sucht die Kurzbeschreibungen und die Namen der Manualseiten zum *Schlüsselwort printf* und gibt alle Treffer aus.

man -f *smail*

Sucht die Manualseiten, die sich auf *smail* beziehen und gibt alle gefundenen Kurzbeschreibungen aus.

Uebersicht

Um dem Benutzer eine größtmögliche Flexibilität zu bieten, sind in **man** viele Optionen verfügbar. Veränderungen können am Suchpfad, in der Reihenfolge der Abschnitte oder am Ausgabeprozessor vorgenommen werden. Andere Verhaltensweisen und Operationen sind weiter unten beschrieben.

Um die Arbeitsweise von **man** zu bestimmen, werden verschiedene Umgebungsvariablen benutzt. Mit dieser Version ist es möglich, die Variable $**MANOPT** auf einen Ausdruck im Kommandozeilenformat zu setzen. Es gibt folgende Ausnahme: Da jede Option in $**MANOPT** von Leerzeichen eingeschlossen ist, müssen Leerzeichen, die Teil eines Argumentes sind, gequotet werden. **man** bearbeitet diese Variable vor der eigenen Kommandozeile. Die Optionen, die ein Argument benötigen, können durch die gleichen Optionen in der Kommandozeile überschrieben werden. Um alle Optionen zurückzusetzen, die in $**MANOPT** gesetzt werden, kann **-D** als initiale Kommandozeilen-Option angegeben werden. Dies erlaubt **man** alle Optionen zu `überschreiben', die in $**MANOPT** gesetzt werden, obwohl diese weiterhin gelten. Damit wird die größtmögliche Flexibilität für einen Benutzer erzeugt, der **man** auf seine eigenen Bedürfnisse anpassen will, ohne zahllose Optionen bei der Suche einer Manualseite einzugeben.

Die Hilfsprogramme, die im **man_db-** Paket zusammengefaßt sind, machen umfassenden Gebrauch von *Indexdatenbanken.* Diese Zwischenspeicher enthalten Informationen über den Ort und die zugehörige *whatis* Information (einzeilige Kurzbeschreibung der Manualseite), sowie darüber, wo sich eine Manualseite im Dateisystem befindet. Eine Aufgabe von **man** ist die Konsistenzsicherung der Datenbank. Die Datenbanken verhindern die Notwendigkeit einer manuellen Bearbeitung zur Aktualisierung der *whatis*- Textdatenbank und erlauben **man** schneller zu arbeiten, als bei der Suche nach der passenden Manualseite im gesamten Dateisystem.

Wenn **man** keine von **mandb** erstellte *Indexdatenbank* zu einer speziellen Manualhierarchie finden kann, wird dennoch nach der gewünschten Manualseite gesucht. In diesem Fall ist es jedoch wieder nötig, nach der alten Methode alle in Frage kommenden Verzeichnisse zu durchsuchen (sog. globbing). Wenn **whatis** oder **apropos** keine *Indexdatenbank* finden können, versucht es die Information aus der *whatis*- Datenbank zu beziehen. Auch anwenderspezifische Manualhierarchien werden während der Benutzung in *Indexdatenbanken* zusammengefaßt.

Die Hilfsprogramme unterstützen komprimierte nroff-Quelldateien, die normalerweise die Erweiterung **.Z**, **.z** oder **.gz** besitzen. Jede andere Erweiterung kann unterstützt werden, wenn sie zur Übersetzungszeit bekannt ist. Als Default werden alle cat-Seiten mit **gzip** komprimiert. Jede globale Manualhierarchie wie */usr/man* oder */usr/X11R6/man* kann jedes Verzeichnis als cat-Seiten-Hierarchie besitzen. Üblicherweise werden cat-Seiten unter der gleichen Hierarchie wie die Manualseiten gespeichert. Allerdings kann es aus Gründen, die im **Linux File System Standard (FSSTND)** erläutert sind, besser sein, sie an anderer Stelle zu speichern. Details, wann dies der Fall ist, beschreibt **manpath**(5) . Für Details, warum dies empfohlen wird, siehe den **Linux File System Standard (FSSTND)**.

Dieses Paket unterstützt internationale Anpassungen (sog. NLS-Support, Native Language Support). Durch den Gebrauch von *locale* Funktionen ist es möglich, Manualseiten der Landessprache zu verwalten, wenn sie auf dem System vorhanden sind. Um diese Unterstützung zu aktivieren, muß man entweder in $**LC_MESSAGES**, $**LANG** oder anderen systemabhängigen Umgebungsvariablen die gewünschte Sprache einstellen. Die Sprache wird normalerweise in dem durch **POSIX 1003.1** definierten Format angegeben: <*Sprache*>[**_** <*Region*>[**.**<*Zeichensatz*>[**,**<*Version*>]]]

Wenn die angeforderte Seite in der *locale* vorhanden ist, wird sie anstelle der Standardseite (normalerweise in amerikanischem Englisch) angezeigt. Darüber hinaus werden auch Sammlungen von landessprachlichen Systemmeldungen unterstützt und auf dieselbe Weise aktiviert - ebenfalls unter der Vorraussetzung, daß die übersetzten Meldungen vorliegen. Wer diese Manualseiten und die Ausgaben der Hilfsprogramme gerne in seiner Landessprache hätte, aber diese nicht vorfindet, ist aufgefordert, eine Übersetzung anzufertigen und sie dem Autor zuzusenden, damit spätere Versionen davon profitieren können.

Die anderen Eigenschaften und Erweiterungen von **man** sind in den beiliegenden Dokumenten beschrieben.

Einen umfassenden Einblick in die **mandb** zugrundeliegenden Konzepte sind in der Dokumentation *man_db-2.3* - the database cached manual pager suite beschrieben. Die Dokumentation sollte auf denselben Server wie das **mandb-** Paket selbst zu finden sein.

Normaleinstellungen

man sucht nach der gewüschten Manualseite in der *Indexdatenbank.* Wenn die Suche fehlschlägt, wird ein Konsistenztest durchgeführt, um die korrekte Wiedergabe des Dateisystems zu sichern. Nachdem die Datenbanken erzeugt wurden, ist es i. A. nicht notwendig, **mandb** zu starten, es sei denn, die Datenbank wurde verfälscht.

Wenn eine Manualseite gefunden wurde, wird getestet, ob dazu bereits eine vorformatierte cat-Seite existiert und diese neuer als die nroff-Datei ist. In diesem Fall wird die vorformatierte Datei dekomprimiert und mit einem Browser angezeigt. Die Auswahl des Browsers kann auf unterschiedliche Weise erfolgen (für Details siehe **-P** Option). Wenn keine cat-Seite gefunden wird oder wenn sie älter als die nroff-Datei ist, wird die nroff-Datei durch diverse Programme gefiltert und entweder sofort angezeigt oder zuerst als komprimierte cat-Datei gespeichert und dann angezeigt.

Eine cat-Datei wird erzeugt, wenn ein relatives cat-Verzeichnis existiert und **man** dort das Schreibrecht hat.

Die Filter werden in mehreren Schritten zusammengestellt: Zuerst wird die Kommandozeilen-Option **-p** oder die Umgebungsvariable $**MANROFFSEQ** untersucht. Wenn **-p** nicht benutzt wird und die Umgebungsvariable nicht gesetzt ist, wird die Anfangszeile der nroff-Datei nach einer Zeichenkette für den Präprozessor untersucht. Eine solche Präprozessor-Zeichenkette muß folgendes Aussehen haben:

'\" <**Zeichenkette**>

wobei **Zeichenkette** jede Kombination von Buchstaben sein kann, die unter Option **-p** weiter unten beschrieben sind.

Wenn keine der obigen Methoden eine Filter-Information enthält, wird **tbl** als Default verwendet.

Als primärer Formatierer wird entweder **nroff**, **troff** oder **groff** gestartet.

Optionen

Eine Argumentoption, die entweder in der Kommandozeile, in $**MANOPT** oder in beiden doppelt vorkommt , ist nicht schädlich. Für Optionen, die ein Argument benötigen, überschreibt jedes Duplikat den vorhergehenden Wert.

-l, --local-file

Aktiviert den lokalen Modus. Formatiert und zeigt lokale Manualdateien an, anstatt die System-Manualsammlung zu durchsuchen. Jedes Manualseiten-Argument wird als nroff-Quelle im richtigen Format interpretiert. Komprimierte nroff-Quelldateien mit einer unterstützten Kompressions-Erweiterung werden von **man** dekomprimiert, bevor sie über den üblichen Filter angezeigt werden. Es wird keine cat-Datei erzeugt. Wenn eines der Argumente `-' ist, wird die Eingabe von der Standardeingabe übernommen.

-L *locale***, --locale=***locale*

Normalerweise bestimmt **man** die aktuelle **locale** durch einen Aufruf der C Funktion **setlocale(3) ,** die diverse Umgebungsvariablen, darunter u. U. $**LC_MESSAGES** und $**LANG** untersucht. Diese Funktion kann dazu verwendet werden, kurzzeitig den so gefundenen Wert zu überschreiben. Dazu kann diese Option mit einer Zeichenkette, die die temporäre **locale** enthält, angegeben werden. Man beachte, daß dieser Effekt erst beim konkreten Suchen der Seite in Erscheinung tritt. Daher werden Ausgaben wie die Hilfeseite immer in der ursprünglichen Sprache ausgegeben.

-D, --default

Diese Option wird normalerweise nur als allererste angegeben und setzt das Verhalten von **man** in allen Belangen wieder zum Normalverhalten zurück. Der Zweck dieser Option ist es, Optionen wieder rückgängig zu machen, die bereits in der Umgebungsvariable $**MANOPT** gesetzt sind. Alle Optionen, die **-D** folgen, haben wieder ihren gewohnten Effekt.

-M *Pfad***, --manpath=***Pfad*

Ermöglicht es, einen alternativen Manualpfad anzugeben. Normalerweise verwendet **man** dieselben Methoden wie in **manpath,** um den Suchpfad zu ermitteln. Diese Option überschreibt die Umgebungsvariable $**MANPATH.**

-P *Pager***, --pager=***Pager*

Gibt an, welcher Pager verwendet wird. Die Normaleinstellung ist **exec /usr/bin/pager -s**. Diese Option überschreibt die Umgebungsvariable $**PAGER** und wird nicht in Zusammenhang mit **-f** oder **-k** verwendet.

-r *Prompt***, --prompt=***Prompt*

Wenn eine hinreichend neue Version von **less** als Pager verwendet wird, versucht **man** dort einige sinnvolle Optionen zu setzen. Die Eingabeaufforderung in der letzten Zeile sieht in dem Fall so aus:
Manual page *name***(***sec***) line** *x*,

wobei *name* die Manualseite bezeichnet, die gerade angezeigt wird, *sec* der Abschnitt ist, in dem sie gefunden wurde, und *x* die aktuelle Zeilennummer ist. Diese Anzeige wird durch Verwendung der Umgebungsvariable $**LESS** erreicht. Man beachte, daß einzelne Bezeichnungen sich bei der Verwendung von landessprachlichen Meldungen ändern können. Die Option **-r** ermöglicht es, durch Angabe einer Formatierungszeichenkette, das Ausgabeformat selbst zu bestimmen. Wenn diese Zeichenkette **$MAN_PN** enthält, wird dieser Text durch den Namen der Manualseite gefolgt von der Abschnittsnummer in runden Klammern ersetzt. Die Zeichenkette, die im Normalfall verwendet wird ist:

\ Manual\ page\ \$MAN_PN\ ?ltline\ %lt?L/%L.:
byte\ %bB?s/%s..?\ (END):?pB %pB\\%..

Die zweizeilige Darstellung wurde nur der besseren Lesbarkeit wegen gewählt. Nähere Informationen liefert **less**(1) . Da die Zeichenkette zuerst vom Kommandointerpreter ausgewertet wird, müssen entsprechende Zeichen durch einen Backslash geschützt werden. Weitere Optionen für **less** können nach einem geschützten $ am Ende der Zeichenkette hinzugefügt werden. Der Default ist hier **-ix8.**

-S *Liste***, --sections=***Liste*

Eine durch Doppelpunkte getrennte Liste von Abschnitten definiert bei Benutzung dieser Option die Reihenfolge, in der die Abschnitte durchsucht werden. Diese Option überschreibt die Umgebungsvariable $**MANSECT.**

-a, --all

Wird eine Manualseite in einem Abschnitt gefunden, so terminiert **man** nach Anzeige dieser Seite. Wird jedoch diese Option angegeben, so werden alle passenden Manualseiten nacheinander angezeigt.

-c, --catman

Diese Option überprüft nur, ob die zur angegebenen Manualseite passende cat-Seite aktuell ist und erzeugt ggf. eine neue. Es wird dabei nichts angezeigt.

-d, --debug

Bei dieser Option werden keine Manualseiten angezeigt, sondern nur eine Menge von Diagnoseinformation.

-e *Erweiterung***, --extension=***Erweiterung*

Einige Systeme enthalten große Pakete an Manualseiten, wie z. B. in dem **Tcl** Paket, die in die normalen

Manualabschnitte integriert werden. Um beispielsweise das Problem zweier unterschiedlicher Manualseiten mit gleichem Namen, wie **exit**(3) , zu lösen, wurden früher alle **Tcl** Seiten dem Abschnitt **l** zugeordnet. Dieses erwies sich als keine gute Lösung. Bei dieser Version von **man** ist es möglich, die Seiten in die richtigen Abschnitte einzuordnen und ihrem Seitennamen eine spezifische Erweiterung, hier z. B. **exit**(3tcl) anzuhängen. Unter normalen Umständen zeigt **man** bevorzugt **exit**(3) gegenüber **exit**(3tcl) an. Um dieses Verhalten umzukehren, ist **man** die Zeichenkette *Erweiterung* zu übergeben, die angibt, in welchem Paket die Manualseite zu finden ist. Im obigen Fall beschränkt **man** seine Suche auf Seiten mit der Erweiterung ***tcl,** wenn es mit **-e tcl** aufgerufen wurde. Die Suche wird dabei in allen Abschnitten durchgeführt.

-f, --whatis

Diese Option ist das Äquivalent zu **whatis**. Es wird eine Kurzbeschreibung der gewünschten Manualseite angezeigt, wenn sie gefunden wurde. Zu Details siehe **whatis**(1) . Mit dieser Option ist nur eine Standardsuche möglich. Verbesserte Suchmöglichkeiten bieten die Optionen von **whatis.**

-k, --apropos

Diese Option ist das Äquivalent zu **apropos**. Es werden die Kurzbeschreibungen zu allen Manualseiten nach dem angegebenen Stichwort durchsucht. Zu Details siehe **apropos**(1) . Mit dieser Option ist nur eine Standardsuche möglich. Verbesserte Suchmöglichkeiten bieten die Optionen von **apropos.**

-m *System* [,...]**, --systems=***System*[,...]

Wenn auch Manualseiten von einem anderen Betriebssystem installiert sind, so kann auf sie mit dieser Option zugegriffen werden. Um beispielsweise die Manualseiten von NewOS zu durchsuchen, muß **-m NewOS** angegeben werden.

Das angegebene *System* kann eine durch Kommata abgetrennte Aufzählung von Betriebssystemnamen sein. Die normalen Seiten werden durch den Betriebssystemnamen **man** angesprochen. Diese Option überschreibt die Umgebungsvariable $**SYSTEM.**

-p *Zeichenkette***, --preprocessor=***Zeichenkette*

Diese Option gibt die Reihenfolge an, in der die Präprozessoren vor **nroff** oder **troff**/**groff** abgearbeitet werden. Nicht alle Installationen haben notwendigerweise alle Präprozessoren. Einige der Präprozessoren und die Zeichen, um sie zu repräsentieren, sind: **eqn** (**e**), **grap** (**g**), **pic** (**p**) , **tbl** (**t**), **vgrind** (**v**), **refer** (**r**). Diese Option überschreibt die Umgebungsvariable $**MANROFFSEQ.** Der Präprozessor **zsoelim** wird immer als erster gestartet.

-u, --update

Die *Indexdatenbanken* werden immer während des laufenden Betriebes auf neuestem Stand gehalten, was

insbesondere bedeutet, daß **mandb** nicht benötigt wird, um sie konsistent zu halten. Wenn die ausgewählte Manualseite nicht im Index gefunden wurde oder die **-a** Option verwendet wurde, macht **man** einen Konsistenztest auf Verzeichnisebene, um sicherzustellen, daß der Index immer noch eine gültige Repräsentation der Manualseiten im Dateisystem darstellt. Wenn dieser Test auf Inode-Ebene durchgeführt werden soll, muß man die Option **-u** benutzen.

-t, --troff

Mit dieser Option wird */usr/bin/groff -mandoc* verwendet, um die Manualseite zu formatieren und an die Standardausgabe zu liefern. Im Zusammenhang mit **-T** oder **-Z** ist diese Option nicht nötig.

-T *Format*, **--troff-device** [*=Format*]

Diese Option wird dazu verwendet, um das Ausgabeformat von **groff** (oder möglicherweise **troff**) zu ändern. Diese Option impliziert die Angabe von **-t**. Verfügbare Ausgabeformate (von Groff-1.09) beinhalten **dvi**, **latin1**, **X75** und **X100**.

-Z, --ditroff

Das traditionelle **troff** erzeugt ditroff. **groff** startet zunächst **troff** und leitet danach dessen Ausgabe an einen für das gewählte Ausgabeformat geeigneten Postprozessor weiter. Wenn */usr/bin/groff -mandoc* **groff** ist, zwingt diese Option **groff** dazu, traditionelles **troff** zu emulieren und impliziert **-t**, andernfalls wird es ignoriert.

-w, --where, --location

Zeigt nicht die Manualseiten, sondern die Position der Dateien im Dateisystem an, die formatiert oder angezeigt würden. Wenn die Datei eine cat-Seite ist, wird auch der Ort ihrer nroff-Quelldatei angezeigt.

-h, --help

Zeigt einen Hilfstext an.

-V, --version

Zeigt Programmversion und Autor an.

Umgebung

MANPATH

Wenn $**MANPATH** gesetzt ist, wird dessen Wert als Suchpfad für die Manualseiten benutzt.

MANROFFSEQ

Wenn $**MANROFFSEQ** gesetzt ist, wird dessen Wert benutzt, um die Abfolge der Präprozessoren zu

bestimmen, die jede Manualseite vor **nroff** oder **troff** durchläuft. Als Default durchlaufen die Seiten den **tbl** (**t**) Präprozessor.

MANSEC

Wenn der Wert von $**MANSEC** eine durch Doppelpunkte getrennte Liste von Abschnitten ist, wird dieser Wert dazu benutzt, um die zu durchsuchenden Abschnitte und deren Reihenfolge zu bestimmen.

PAGER

Wenn $**PAGER** gesetzt ist, wird dieses Programm zur Anzeige benutzt. Default ist **exec /usr/bin/pager -s.**

SYSTEM

Wenn $**SYSTEM** gesetzt ist, hat dies den gleichen Effekt wie die Option **-m** *System* wobei *System* als Inhalt der Umgebungsvariable $**SYSTEM** benutzt wird.

MANOPT

Wenn $**MANOPT** gesetzt ist, wird der Wert dieser Variablen vor der **man** Kommandozeile durchsucht und abgearbeitet. Wie auch alle anderen Umgebungsvariablen, die als Kommandozeilen-Optionen ausgedrückt werden können, ist es möglich, den Inhalt von $**MANOPT** durch die Kommandozeile zu überschreiben. Alle Leerzeichen, die Teil eines Argumentes sind, müssen gequotet werden.

LANG, **LC_MESSAGES**

Abhängig von System und Implementation werden entweder $**LANG** oder $**LC_MESSAGES** oder beide nach der gegenwärtigen *locale* Information durchsucht. **man** wird (wenn möglich) seine Nachrichten in dieser *locale* anzeigen. Für Details siehe **setlocale**(3) .

Dateien

/etc/manpath.config

Die Manualkonfigurationsdatei.

/usr/.../man

Globale Manualhierarchien.

/var/catman/.../index.(bt|db|dir|pag)

Die **FSSTND** complianten globalen *Indexdatenbanken.*

Siehe Auch

mandb(8) , **manpath**(1) , **manpath**(5) , **apropos**(1) , **whatis**(1) , **catman**(8) , **less**(1) , **nroff**(1) , **troff**(1) , **groff**(1) , **zsoelim**(1) , **setlocale**(3) .

Fehler

Die Option **-t** funktioniert nur, wenn ein troff-ähnliches Programm installiert ist.

Die Option **-e** funktioniert momentan nur in Manualhierarchien, für die mit **mandb** eine *Indexdatenbank* erzeugt wurde.

Manualseiten, die die erweiterte Darstellung im Zusammenhang mit der Option **-e** unterstützen, dürfen keinen Punkt beinhalten, denn sonst werden diese Seiten als fehlerhaft zurückgewiesen.

Geschichte

1990, 1991 - Originale geschrieben von John W. Eaton (jwe@che.utexas.edu).

23. Dez. 1992: Fehlerbereinigung durch Rik Faith (faith@cs.unc.edu) unterstützt durch Willem Kasdorp (wkasdo@nikhefk.nikef.nl).

Zwischen dem 30. April 1994 und dem 12 Juli 1995 hat Wilf (G.Wilford@ee.surrey.ac.uk) unter Mithilfe von einigen wenigen engagierten Menschen dieses Paket entwickelt und weitergeführt.

Die deutsche Übersetzung wurde von Anke Steuernagel (a_steuer@informatik.uni-kl.de) und Nils Magnus (magnus@informatik.uni-kl.de) angefertigt.

Für persönliche Würdigungen und Zusätze siehe Quelltexte.

Name

mandb - erzeugt oder aktualisiert die Indexdatenbank

Syntax

mandb [**-dqsuc**|**-h**|**-V**] [*Pfad*]

Beschreibung

Um die interne *Indexdatenbank,* die von **man** verwendet wird, zu initialisieren oder auf den neusten Stand zu bringen, wird **mandb** verwendet. Die *Indexdatenbank* enthält Informationen über den aktuellen Zustand der Manualseiten.

Wenn eine *Indexdatenbank* erzeugt oder erneuert werden soll, warnt **mandb** vor möglichen Inkonsistenzen, Seiten ohne nroff-Quelle oder Seiten ohne **whatis-** Beschreibungen.

Wenn **mandb** beim Aufruf ein optionaler, durch Doppelpunkte abgetrennter Pfad übergeben wird, so wird der interne durch die Konfigurationsdatei bestimmte Manualpfad verdeckt.

Indexdatenbanken

Von **mandb** wird einer der folgenden Datenbanktypen verwendet (Auswahl während der Übersetzungszeit):

Name	Typ	asyncron	Dateiname
Berkeley db	Binärbaum	Ja	*index.bt*
GNU gdbm v >= 1.6	Hashtabelle	Ja	*index.db*
GNU gdbm v < 1.6	Hashtabelle	Nein	*index.db*

UNIX ndbm	Hashtabelle	Nein	*index.(dir|pag)*

Die Datenbanktypen, die asyncrone Veränderungen erlauben, erlauben eine höhere Geschwindigkeit zulasten der Möglichkeit, daß bei einer nichtgewöhlichen Beendung des Programms die Datenbank fehlerhaft sein kann.

In diesem seltenen Fall kann es nötig sein, **mandb** mit der **-c** Option zu starten, um die Datenbanken von Grund auf neu zu erstellen.

Optionen

-d, --debug

Erzeugt Debug-Informationen.

-q, --quiet

Unterdrückt die Warnungen.

-s, --no-straycats

Beachtet beim Aufbau keine cat-Seiten ohne zugehörige nroff-Quellen (sog. *stray-cats).*

-c, --create

Im Normalfall versucht **mandb** zunächst eine bestehende Datenbank auf den aktuellen Stand zu bringen und erzeugt eine neue nur dann, wenn noch keine solche existiert. Durch Angabe dieser Option wird **mandb** dazu gezwungen, die Datenbank von Grund auf neu zu erstellen.

-u, --user-db

Erzeugt nur die benutzerspezifischen Datenbanken, selbst wenn Schreibrechte auf systemweite Manualhierarchien bestehen.

-h, --help

Zeigt einen Hilfstext an.

-V, --version

Zeigt Programmversion und Autor an.

Dateien

/var/catman/.../index.(bt|db|dir|pag)

Die **FSSTND** complianten globalen *Indexdatenbanken.*

/usr/man/.../whatis

Traditionelle *whatis* Datenbank.

Siehe Auch

man(1) , **manpath**(5) , **catman**(8) .

Autor

Wilf. (G.Wilford@ee.surrey.ac.uk), die deutsche Übersetzung ist von Anke Steuernagel (a_steuer@informatik.uni-kl.de) und Nils Magnus (magnus@informatik.uni-kl.de).

Name

manpath - ermittelt den aktuellen Suchpfad für die Manualseiten

Syntax

manpath [**-qgdc**] [**-m** *System*[,...]]

Beschreibung

manpath versucht den aktuellen Suchpfad der Manualseiten aus den Standardeinstellungen und der Umgebungsvariable $**PATH** zu ermitteln. Das Ergebnis wird auf der Standardausgabe ausgegeben, Warnungen und Fehler auf der Standardfehlerausgabe. Wenn ein Verzeichnis nicht in der Konfigurationsdatei aufgelistet ist, sucht **manpath** nach den Unterverzeichnissen `man' und `MAN'. Wenn diese existieren, werden sie dem Suchpfad hinzugefügt.

Da **manpath** von **man** benutzt wird, um den Suchpfad zu bestimmen, ist es für einen Benutzer im Normalfall nicht notwendig, die Umgebungsvariable $**MANPATH** direkt zu setzen.

Optionen

-q, --quiet

Gibt nur den endgültigen Manualpfad aus.

-d, --debug

Ausgeben von Fehlerinformationen.

-c, --catpath

Erzeugt einen cat-Suchpfad. Vorhandene Manualpfade werden in relative cat-Suchpfade umgewandelt.

Diese Option kann mit **-g** kombiniert werden.

-g, --global

Erzeugt einen Manualpfad, der aus allen Pfaden besteht, die in der Konfigurationsdatei als global bezeichnet sind. Diese Option kann mit **-c** kombiniert werden.

-m *System* [,...]**, --systems=***System*[,...]

Wenn auch Manualseiten von einem anderen Betriebssystem installiert sind, so kann auf sie mit dieser Option zugegriffen werden. Um beispielsweise die Manualseiten von NewOS zu durchsuchen, muß **-m NewOS** angegeben werden.

Das angegebene *System* kann eine durch Kommata abgetrennte Aufzählung von Betriebssystemnamen sein. Die normalen Seiten werden durch den Betriebssystemnamen **man** angesprochen. Diese Option überschreibt die Umgebungsvariable $**SYSTEM.**

-h, --help

Zeigt einen Hilfstext an.

-V, --version

Zeigt Programmversion und Autor an.

Umgebung

MANPATH

Wenn $**MANPATH** gesetzt ist, gibt **manpath** diesen Wert auf die Standardausgabe aus. Warnungen werden auf die Standardfehlerausgabe ausgegeben.

SYSTEM

Wenn $**SYSTEM** gesetzt ist, hat dies die gleiche Wirkung wie die Option **-m** *System* wobei *System* als Inhalt von $**SYSTEM** angenommen wird.

Dateien

/etc/manpath.config

Die Manualkonfigurationsdatei.

Siehe Auch

apropos(1) , **whatis**(1) , **man**(1) .

Autor

Wilf. (G.Wilford@ee.surrey.ac.uk), die deutsche Übersetzung ist von Anke Steuernagel (a_steuer@informatik.uni-kl.de) und Nils Magnus (magnus@informatik.uni-kl.de).

Bezeichnung

mkdir - erzeugt ein leeres Verzeichnis

Syntax

mkdir [**-p**] [**-m** *Modus*] [**--path**] [**--mode=** *Modus*] *Verzeichnis* ...

Optionen

-m *Modus*

setzt die Rechte des Verzeichnisses auf *Modus*; der *Modus* wird wie beim Kommando *chmod* angegeben; der Standard, und damit der Ausgangswert für relative Modes, ist **0777** minus der Bits von **umask**

-p

wenn ein Unterverzeichnis in einem nicht existierenden Verzeichnis angelegt werden soll, werden alle fehlenden Verzeichnisse angelegt

Siehe Auch

ln(1) , **rmdir**(1) und das LunetIX Linuxhandbuch

Autor

David MacKenzie

Bezeichnung

mv - (move) verschiebt eine Datei oder benennt sie um

Syntax

mv [*Optionen*] *Quelle* Ziel

mv [*Optionen*] *Quelle* ... Verzeichnis

Beschreibung

mv verschiebt eine oder mehrere Datei(en) bzw. Verzeichnis(se) oder benennt sie um. Ein Verzeichnis kann nicht über die Grenzen eines Dateisystems hinweg verschoben werden.

Optionen

-b

sichert Dateien vor dem Überschreiben

-f

überschreibt existierenden Zieldateien rücksichtslos

-i

erwartet interaktiv eine Bestätigung vor dem Überschreiben existierender Zieldateien

-u

verschiebt Dateien nur, wenn sie neuer sind als die gleichnamigen Zieldateien

-v

meldet jede Aktion

-S *Endung*

bestimmt die *Endung* für einfaches Backup. Voreinstellung ist **~**

-V {**numbered**, **existing**, **simple**}

kontrolliert die Art des Backups. Die Parameter sind beim Befehl **cp**(1) erklärt.

Siehe Auch

ln(1) , **cp**(1) und das LunetIX Linuxhandbuch

Autor

Mike Parker und David MacKenzie

Name

newgrp - Wechselt die Logingruppe
sg - Führt ein Kommando unter einer anderen Login GruppenID aus

Syntax

newgrp [-] [*Gruppe*]
sg [-] [*Gruppe* [[**-c**] *Kommando*]]

Beschreibung

newgrp wird benutzt, um die aktuelle GruppenID (standardmäßig die GID der Login-Gruppe) während einer Sitzung zu wechseln. Wenn der optionale **-** Parameter gesetzt ist, wird der Umgebungsspeicher des Users reinitialisiert, als hätte sich der User neu eingeloggt, wenn nicht, bleibt der Umgebungsspeicher unverändert.

newgrp wechselt die aktuelle reale GruppenID des Users in die der genannten Gruppe oder in die der Login-Gruppe des Users, wenn kein Gruppenname angegeben wurde. Der User wird nach einem Passwort gefragt, wenn sie selbst kein passwort haben und die Gruppe ein Passwort erfordert, oder wenn der User nicht Gruppenmitglied der Gruppe ist und die Gruppe ein Passwort besitzt. Wenn die Gruppe kein Passwort besitzt und der User nicht Mitglied der Gruppe ist, wird ihm der Zugriff verweigert.

Das **sg** Kommando arbeitet ähnich wie **newgrp**, ersetzt aber nicht die Usershell des Benutzers, wenn also das **sg** Kommando beendet wurde, wird der User wieder seine vorherige Gruppen ID haben. Wenn **sg** ein Kommando mitgegeben wurde, wird es unter der genannten GID durch die Bourne Shell ausgeführt. Es muß in Anführungszeichen gesetzt werden.

Dateien

/etc/passwd - Useraccount-Informationen
/etc/group - Gruppeninformationen

Siehe auch

login(1) , **id**(1) , **su**(1)

Autor

Julianne Frances Haugh (jfh@bga.com)

Bezeichnung

nice - läßt ein Programm mit veränderter Priorität laufen

Syntax

nice [**-n** *Nettigkeit*] [**-** *Nettigkeit*] [**--adjustment=** *Nettigkeit*] [*Kommando* [*Argument*...]]

Beschreibung

In einem Multitasking Betriebssystem wie Linux muß die Prozessorzeit auf verschiedene Prozesse verteilt werden. Dazu gibt es einen **Scheduler** der dafür sorgt, daß die Prozessorzeit möglichst optimal zugeteilt wird. Prozesse, die auf das Ergebnis eines anderen Prozesses, ein Ereignis oder ein Signal warten, können beispielsweise solange `schlafen', bis das erwartete Ereignis eingetreten ist. Der Scheduler weckt die schlafenden Prozesse dann auf und teilt ihnen wieder Prozessorzeit zu. Trotzdem gibt es natürlich in der Regel mehr als einen lauffähigen Prozess. Und der Scheduler muß nach bestimmten Regeln den lauffähigen Prozessen Rechenzeit zuteilen. Dabei benutzt er unter anderem den **nice** Wert. Will ein Anwender nice, das heißt nett, zu den anderen Benutzern des Systems sein, startet er die Prozesse, die ruhig etwas länger dauern dürfen mit, dem **nice** Kommando. Ein negatives nice ist nur dem Superuser **root** erlaubt.

Wenn kein Wert angegeben ist, wird die Nettigkeit um 10 Punkte erhöht. Die maximale Nettigkeit ist 19 Punkte. Nach unten kann der Superuser seine Nettigkeit bis -20 abkühlen.

Optionen

-n *Wert*

-*Wert*

setzt **nice** auf *Wert*

Siehe Auch

nohup(1) und das LunetIX Linuxhandbuch

Autor

David MacKenzie

Funktion

nl - nummeriert die Zeilen in einer Datei

Syntax

nl [**-h** *Stil*] [**-b** *Stil*] [**-f** *Stil*] [**-p**] [**-d** *zwei Zeichen*] [**-v** *Nummer*] [**-i** *Nummer*] [**-l** *Nummer*] [**-s** *Zeichenkette*] [**-w** *Nummer*] [**-n**{**ln**,**rn**,**rz**}] [**--header-numbering=** *Stil*] [**--body-numbering=** *Stil*] [**--footer-numbering=** *Stil*] [**--first-page=** *Nummer*] [**--page-increment=** *Nummer*] [**--no-renumber**] [**--join-blank-lines=** *Nummer*] [**--number-separator=** *Zeichenkette*] [**--number-width=** *Nummer*] [**--number-format=**{**ln**,**rn**,**rz**}] [**--section-delimiter=** *zwei Zeichen*] [*Datei*...]

Beschreibung

nl gibt die Zeilen einer oder mehrerer Dateien (oder der Standardeingabe) mit Zeilennummern auf die Standardausgabe. Es können dabei die Zeilen einer (logischen) Seite in einen Kopf, einen Körper und einen Fuß unterteilt werden, die jeweils einzeln und in unterschiedlichen Stilen nummeriert werden. Jeder Teil kann auch leer sein. Wenn vor dem ersten Kopfteil bereits Zeilen vorhanden sind, werden diese Zeilen wie ein Seitenkörper nummeriert.

Die Nummerierung beginnt auf jeder Seite neu. Mehrere Dateien werden als ein einziges Dokument betrachtet, und die Zeilennummer nicht zurückgesetzt.

Der Kopfteil wird durch eine Zeile eingeleitet, die nur die Zeichenkette **\:\:\:** enthält. Der Körper wird entsprechend durch **\:\:** und der Fuß durch **\:** eingeleitet. In der Ausgabe werden diese Zeilen als Leerzeilen ausgegeben.

Optionen

-h *Stil*

bestimmt die Art der Zeilennummerierung für die Kopfzeile; das Nummerntrennzeichen wird auch den nicht nummerierten Zeilen vorangestellt; als *Stil* werden folgende Zeichen erkannt

a

alle Zeilen werden nummeriert

t

die leeren Zeilen werden nicht nummeriert (Voreinstellung für den Körper)

n

die Zeilen werden nicht nummeriert (Voreinstellung für Kopf und Fuß)

p *Ausdruck*

nur die Zeilen, in denen der reguläre *Ausdruck* vorkommt, werden nummeriert

-b *Stil*

bestimmt die Art der Zeilennummerierung für den Körper

-f *Stil*

bestimmt die Art der Zeilennummerierung für den Fuß

-p

die Zeilen aller Seiten werden fortlaufend nummeriert

-v *Nummer*

die erste Zeile jeder logischen Seite bekommt die angegebene Nummer

-i *Nummer*

die Schrittweite für die Nummerierung

-l *Nummer*

die angegebene Anzahl aufeinanderfolgender Leerzeilen werden als eine Zeile angesehen, und die letzte Zeile nummeriert; wenn weniger Leerzeilen in Folge auftreten, werden sie nicht nummeriert; Leerzeilen dürfen auch keine Leerzeichen oder Tabulatoren enthalten

-s *Zeichenkette*

setzt die *Zeichenkette* als Nummerntrennzeichen zwischen Zeilennummer und Text; Voreinstellung ist **tab**

-w *Nummer*

die Zeilennummern erhalten die angegebene Anzahl Stellen; Voreinstellung ist 6

-n {**ln**, **rn**, **rz**}

die Zeilennummern werden in dem angegebenen Stil ausgegeben; dabei bedeutet

ln

linksbündig, ohne führende Nullen

rn

rechtsbündig, ohne führende Nullen

rz

rechtsbündig, mit Nullen auf die volle Stellenzahl aufgefüllt

-d *zwei Zeichen*

die zwei Zeichen werden zur Trennung von Kopf, Körper und Fußteil benutzt, Voreinstellung ist **\:**

Siehe Auch

pr(1) . und das LunetIX Linuxhandbuch

Autor

Scott Bartram, David MacKenzie

Bezeichnung

od - (octal dump) zeigt Dateien im oktalen, hexadezimalen oder in anderen Formaten an.

Syntax

od [**-abcdfhiloxv**] [**-s** [*Länge*]] [**-w** [*Anzahl*]] [**-A** *Positionsformat*] [**-j** *Anzahl*] [**-N** *Anzahl*] [**-t** *Format*] [**--skip-bytes=** *Anzahl*] [**--address-radix=** *Positionsformat*] [**--read-bytes=** *Anzahl*] [**--format=** *Format*] [**--output-duplicates**] [**--strings**[= *Anzahl*]] [**--width**[= *Anzahl*]] [*Datei*...]

Beschreibung

od liest die angegebenen Dateien oder die Standardeingabe (wenn keine Datei oder anstelle einer Datei **-** angegeben ist), und gibt die Bytes formatiert und kodiert auf die Standardausgabe.

Jede Ausgabezeile enthält in der ersten Spalte die Positionsnummer des ersten in der Zeile dargestellten Bytes (vom Dateianfang gezählt). In den darauffolgenden Spalten werden die Daten aus der Datei in einem durch die Optionen kontrollierten Format angezeigt.

In der Standardeinstellung ohne Optionen gibt **od** die Position als 7-stellige Oktalzahl und die Daten in 8 Spalten zu je zwei Bytes in oktaler Darstellung aus.

Optionen

-A *Positionsformat*

zeigt die Position des ersten in einer Zeile dargestellten Bytes im Positionsformat; die folgenden Formate stehen zur Auswahl:

-d

siebenstellige Dezimalzahl

-o

siebenstellige Oktalzahl (Voreinstellung)

-x

sechsstellige Hexadezimalzahl

-n

keine Positionsangabe

-j *Anzahl*

überspringt die ersten Anzahl Bytes der Datei und beginnt erst danach mit der Ausgabe; wenn die Zahl mit **0x** oder **B0X** beginnt, wird sie als Hexadezimalzahl interpretiert, beginnt sie mit einer Null wird sie als Oktalzahl behandelt, sonst als Dezimalzahl; der Zahl kann einer der Buchstaben **b** (blocks=512), **k** (kilo=1024) oder **m** (mega=1048576) folgen, die die Anzahl mit den entsprechenden Einheiten multiplizieren

-N *Anzahl*

gibt nur die angegebene Anzahl Bytes von der Datei aus; die Anzahl kann wie bei der **-j** Option von einer Einheit gefolgt werden

-t *Format*

wählt die Codierung für die Datenausgabe; wenn die **-t** Option mehrfach benutzt wird, oder mehrere Formate gleichzeitig angegeben werden, gibt od für jedes Format jeweils eine entsprechende Zeile aus; folgende Formate werden unterstützt

a

(ascii) setzt das achte Datenbit aller Zeichen auf null, die druckbaren ASCII Zeichen werden als solche ausgegeben, und die nichtdruckbaren Steuerzeichen werden mit ihren in der ASCII Tabelle verwendeten ``Namen'' bezeichnet; so wird das Zeilenende als **cr** bezeichnet, ein horizontaler Tabulator mit **tab** und so weiter

c

(character) gibt die druckbaren ASCII Zeichen als solche aus, die nichtdruckbaren Zeichen werden, sofern möglich, als Backslashsequenz ausgegeben; \f ist hier z.B. ein Zeilenende, \t ein Tabulator und so weiter; Bytes die nicht druckbar sind und auch nicht als Backslashsequenz ausgegeben werden können, werden als Oktalzahl dargestellt

d

(decimal) gibt die Daten als vorzeichenbehaftete Dezimalzahlen aus; voreingestellt sind vier Bytes je Dezimalzahl

f

(float) gibt die Daten als Fließkommazahlen aus; voreingestellt sind acht Bytes je Fließkommazahl

o

(octal) gibt die Daten als Oktalzahlen aus; voreingestellt sind vier Bytes je Oktalzahl

u

(unsigned) gibt die Daten als vorzeichenlose Dezimalzahl aus; voreingestellt sind vier Bytes je Dezimalzahl

x

(heX) gibt die Daten als Hexadezimalzahlen aus; voreingestellt sind vier Bytes je Hexadezimalzahl

Außer die Typen **a** und **c**, die immer einzelne Bytes anzeigen, kann die Anzahl der Bytes, die in jeweils eine Zahl des bestimmten Typs umgewandelt werden sollen, durch eine dem Typkennzeichner unmittelbar folgende Zahl bestimmt werden. Die Anzahl der Bytes je Zahl kann auch durch Buchstabenkennungen angegeben werden, die den C-Datentypen entsprechen. Für die Ganzzahltypen (**d**, **u**, **x**, **o**) gibt es die folgenden Möglichkeiten:

C

(Char) ist ein Byte lang

S

(Short) ist zwei Bytes lang

I

(Integer) ist vier Bytes lang

L

(Long) ist auch vier Bytes lang

für Fließkommazahlen können die folgenden Optionen verwendet werden

F

(Float) ist vier Bytes lang

D

(Double) ist acht Bytes lang

L

(Long Double) ist auch acht Bytes lang

-v

gibt auch die doppelten Zeilen aus; normalerweise werden ganze Zeilen, die der zuletzt angezeigten Zeile vollständig entsprechen, durch ein Asterisk ***** dargestellt

-s [*Länge***]**

gibt nur die gültigen C-Zeichenketten (Folgen von druckbaren ASCII Zeichen, durch ein Nullbyte beendet) mit mindestens der angegebenen *Länge* aus; Voreinstellung für Länge ist drei Zeichen

-w [*Anzahl***]**

setzt die Anzahl der umgewandelten Bytes, die in einer Zeile ausgegeben werden; die Anzahl muß ein vielfaches der Länge jedes Ausgabetyps sein; Voreinstellung ist 16

Die folgenden Optionen übersetzen die nicht dem POSIX Format entsprechenden alten Optionen in die neuen dem POSIX Standard entsprechenden Optionen, wie sie oben aufgeführt sind.

-a

gibt benannte ASCII Zeichen aus, wie **-t a**

-b

gibt die Daten Byteweise als Oktalzahlen aus, wie **-t oC**

-c

gibt druckbare Zeichen oder Backslashsequenzen aus, wie **-t c**

-d

gibt die Daten als vorzeichenlose kurze Dezimalzahlen aus, wie **-t u2**

-f

gibt die Daten als Fließkommazahlen mit vier Bytes je Zahl aus, wie **-t fF**

gibt die Daten als vierstellige Hexadezimalzahl aus, wie

-t xL

-i

gibt die Daten als vorzeichenbehaftete Dezimalzahl mit zwei Bytes je Zahl aus, wie **-t d2**

-l

gibt die Daten als vorzeichenbehaftete Dezimalzahl mit vier Bytes je Zahl aus, wie **-t dL**

-o

gibt die Daten als Oktalzahlen mit zwei Bytes je Zahl aus, wie **-t oS**

-x

gibt die Daten als Hexadezimalzahlen mit zwei Bytes je Zahl aus, wie **-t x2**

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Jim Meyering

Name

passwd - Wechseln eines Userpassworts

Syntax

passwd [**-f**|**-s**] [*Name*]
passwd [**-g**] [**-r**|**R**] *Gruppe*
passwd [**-x** *max*] [**-n** *min*] [**-w** *warn*] [**-i** *inact*] *Name*
passwd {**-l**|**-u**|**-d**|**-S**} *Name*

Beschreibung

passwd wechselt Passwörter für User- und Gruppenaccounts. Ein Normaluser darf nur sein Passwort ändern, der Superuser darf die Passwörter aller Accounts ändern. Der Administrator einer Gruppe darf das Passwort dieser Gruppe ändern. **passwd** kann auch die Account-Informationen wie Kommentar, Loginshell oder Passwort-Auslaufzeiten und -intervalle ändern.

Passwortwechsel

Ein Normaluser wird zunächst nach seinem alten Passwort gefragt, wenn er eins hatte. Er hat nur einen Versuch, das alte Passwort richtig einzugeben. Der Superuser wird nicht nach dem alten Passwort gefragt, so daß er vergessene Passwörter ändern kann.

Nachdem das alte Passwort überprüft und für richtig befunden wurde, wird die Passwort-Alter-Information aus /etc/shadow überprüft, um festzustellen, ob ein User zur gegebenen Zeit überhaupt sein Passwort ändern darf. Wenn nicht, beendet sich **passwd**.

Der User wird dann aufgefordert, ein neues Passwort einzugeben. Dieses Passwort wird dann auf Schwächen hin

überprüft. Als eine generelle Anforderung gilt hierbei, daß Passwörter 6 bis 8 Zeichen lang sein sollten und eine oder mehrere Elemente aus den folgenden Mengen enthalten sollte:

- Kleinbuchstaben
- Großbuchstaben
- Ziffern
- Punktierungszeichen

passwd wird einem Normaluser alle Passwörter ablehnen, die nicht mindestens zwei der genannten Kriterien erfüllen.

Wenn das Passwort akzeptiert wurde, muß es ein zweites Mal eingegeben werden, um zu verhindern, daß bei der Eingabe ungewollte Tippfehler übersehen wurden. Nur wenn beide Eingaben identisch waren, wird das Passwort tatsächlich gewechselt.

Gruppenpasswörter

Wenn die **-g** Option angegeben wurde, wird das Passwort der angegebenen Gruppe gewechselt. Der ausführende User muß entweder der Superuser oder der Verwalter der Gruppe sein, deren Passwort geändert werden soll. Das alte Gruppenpasswort wird nicht abgefragt. Die **-r** Option in Zusammenhang mit der **-g** Option löscht das Passwort der Gruppe. Die **-R** Option dagegen erstellt einen ungültigen Passworteintrag und sperrt so die Möglichkeit für Nichtmitglieder der Gruppe, sich mit newgrp in ein Mitglied zu verwandeln.

Passwort Auslauf-Informationen

Der Superuser darf mit den Optionen **-x**, **-n**, **-w**, und **-i** die Passwort Auslaufzeiten und -intervalle verändern. Die **-x** Option setzt die Maximale Anzahl von Tagen, die ein Passwort gültig bleibt. Nach *max* Tagen wird verlangt, das Passwort zu wechseln. Die **-n** Option gibt die Anzahl der Tage an, vor deren Ablauf ein Passwort nicht geändert werden darf. Mit der **-w** Option werden die Anzahl der Tage angegeben, die vor dem Ablauf der Gültigkeit eines Passworts als Warnzeit für den User verwendet werden. Mit der **-i** Option wird die Pufferzeit in Tagen eingestellt, die das Passwort nach dem Ablaufdatum noch gültig bleibt.

Accountwartung

Useraccounts können mit **-l** (lock) und **-u** (unlock) gesperrt und wieder geöffnet werden. Die **-l** Option sperrt den Account, indem das Passwort dergestalt verändert wird, daß es keinen möglichen verschlüsselten Wert mehr darstellt. Mit **-u** wird ein gesperrter Account wieder freigegeben, indem das ursprüngliche Passwort wieder hergestellt wird.

Der Status des Accounts kann mit der **-S** Option abgefragt werden. Die Statusinformation besteht aus sechs Teilen. Der erste Teil zeigt, ob der Useraccount gesperrt (L) ist, ein gültiges Passwort hat (P) oder kein Passwort hat (NP). Der zweite Teil zeigt das Datum des letzten Passwortwechsels. Die nächsten vier Teile zeigen Minimales Alter, Maximales Alter, Warnzeit und Pufferzeit für das Passwort.

Einschränkungen

Eventuell werden nicht alle Optionen auf allen Systemen unterstützt. Die Sicherheitsüberprüfung der Passwörter kann von System zu System verschieden sein.

Dateien

/etc/passwd - Useraccount Informationen
/etc/shadow - Sichere Useraccount Informationen

Siehe auch

passwd(3) , **shadow**(3) , **group**(5) , **passwd**(5)

Autor

Julianne Frances Haugh (jfh@bga.com)

Bezeichnung

paste - fügt die Zeilen von zwei oder mehr Dateien horizontal zusammen

Syntax

paste [**-s**] [**-d** *Liste*] [**--serial**] [**--delimiters=** *Liste*] [*Datei*...]

Beschreibung

paste fügt die Zeilen mehrerer Dateien zusammen. Die Zeilen werden standardmäßig durch **tab** getrennt und die Ausgabe einer kompletten Zeile (das heißt die Ausgabe der entsprechenden Zeilen aller Dateien) wird mit einem Zeilenende abgeschlossen.

Optionen

-d *Liste*

benutzt die Zeichen aus *Liste* zur Trennung der Zeilen aus den einzelnen Dateien beim zusammenfügen; *Liste* ist dabei ein Wort oder eine Zeichenkette aus beliebigen druckbaren Zeichen oder den Sonderzeichen \n \t \\ und \0 für Zeilenende, Tabulator, Backslash oder Leerstring; wenn die *Liste* abgearbeitet ist, wird sie von vorne angefangen

-s

fügt alle Zeilen einer ganzen Datei zu einer Zeile zusammen; werden mehrere Dateien angegeben, so werden die Zeilen der nächsten Dateien als jeweils eine neue Zeile angefügt.

Siehe Auch

das LunetIX Linuxhandbuch

Autor

David M. Ihnat

Bezeichnung

pr - formatiert Textdateien zur Druckerausgabe

Syntax

pr [+ *SEITE*] [- *SPALTEN*] [**-abcdfFmrtv**] [**-e** [*Eintab* [*Schritt*]]] [**/$-$h** *Kopf*] [**-i** [*Austab* [*Schritt*]]] [**-l** *Seitenlänge*] [**-n** [*Separator* [*Stellen*]]] [**-o** *Lrand*] [**-s** [*Separator*]] [**-w** *Breite*] [*Datei*...]

Beschreibung

pr produziert seitenweise nummerierte und einfach formatierte Ausgabe von Textdateien.

Optionen

+Seite

beginnt die Ausgabe mit der angegebenen Seite

-Spalten

setzt den Text in die angegebene Anzahl Spalten. Ein Zeilenumbruch findet nicht statt; die Spaltenbreite wird an die Seitenbreite angepaßt

-a

teilt den Text zeilenweise in die Spalten anstatt seitenweise

-b

schließt die letze Seite mit balancierten Spalten ab, das heißt alle Spalten sind gleichlang

-c

*zeigt Controlzeichen als Caretsequenz (^**G** für Control-G)*

-d

gibt den Text mit doppeltem Zeilenabstand aus

***-e** Eintab Schritt*

*übersetzt das Zeichen Eintab in Schritt Leerzeichen; Voreinstellungen sind **tab** für **Eintab** und 8 für **Schritt***

-f

gibt einen Seitenvorschub anstelle von Zeilenvorschüben am Seitenende

***-h** Kopf*

schreibt die Zeichenkette Kopf anstelle des Dateinamen als Seitenkopf

***-i** Austab Schritt*

*ersetzt Folgen von Schritt Leerzeichen durch ein Austab Zeichen; voreingestellt sind **tab** für **Austab** und 8 für **Schritt***

***-l** Seitenlänge*

setzt die Druckseitenlänge; Voreinstellung ist 66; bei einer Seitenlänge unter 10 Zeilen werden die Kopf- und Fußzeilen weggelassen

-m

gibt alle Dateien parallel in Spalten aus

***-n** Separator Stellen*

*setzt Zeilennummern vor jede Spalte; werden Dateien parallel angezeigt, bekommt jede Zeile nur eine Nummer; der Separator steht zwischen der Nummer und der Zeile, Voreinstellung ist **tab**; die Zeilennummer hat die angegebene Anzahl Stellen, Voreinstellung ist 5*

***-o** Lrand*

*setzt den linken Rand auf Lrand; die Seitenbreite ist die Summe von Lrand und Breite von Option **-w***

-r

gibt keine Fehlermeldung für Dateilesefehler aus

***-s** Separator*

*trennt die Spalten durch den Buchstaben Separator;Voreinstellungist **tab***

-t

Der 5-zeilige Kopf und der 5-zeilige Fußbereich werden nicht ausgegeben, und die Seiten werden nicht durch Leerzeilen oder mit einem Seitenvorschub aufgefüllt

-v

gibt nichtdruckbare Sonderzeichen als Oktalzahl aus

-w *Breite*

setzt die druckbare Breite, Voreinstellung ist 72 Zeichen

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Pete TerMaat

Bezeichnung

ps - (process status) zeigt die Prozesse mit ihrem Status an

Syntax

ps [-acehjlmnrsuvwxS] [-t *xx*] [**-U** [*Systempfad* [*Swappfad*]]

Beschreibung

Mit **ps** lassen sich Daten über die Prozesse in der Prozeßtabelle anzeigen. Die Prozeßtabelle wird mit einer Titelzeile ausgegeben. Die Spalten haben folgende Bedeutung:

PRI

ist die Priorität eines Prozesses; je niedriger der Wert ist, desto mehr Rechenzeit bekommt der Prozeß

NI

ist der Nicewert des Prozesses; Nice erhöht die Priorität des Prozesses und gibt damit Prozessorzeit für andere Prozesse frei

SIZE

ist die Größe von Text, Daten und Stack

WCHAN

ist der Name der Kernelfunktion, in der der Prozeß schläft

STAT

ist der Status des Prozesses

R

lauffähig

S

schlafend

D

nicht störbarer Schlaf

T

angehalten

Z

Zombie

W

der Prozeß belegt keine Seiten im Arbeitsspeicher

%CPU

Anteil an der Prozessorzeit

TT

die Nummer des kontrollierenden Teminal

TPGID

Gruppen ID des kontrollierenden Terminal

PAGEIN

Anzahl der Seitenfehler (das ist der Versuch auf eine ausgelagerte Seite zuzugreifen)

TRS

Größe des Textsegments (enhält keine shared Librarys)

DRS

Größe des Datensegments (enthält benutzte Libraryseiten)

SWAP

ausgelagerte Speicherseiten in Kilobyte (oder Seiten mit **-p**)

SHRD

shared Memory

DT

benutzte Libraryseiten in Kilobyte (oder Seiten mit **-p**)

F

Flags

- Prozeß hat die Mathe Emulation benutzt
- Der Prozeß wurde verfolgt (traced)

Es gibt zwei unterschiedliche Versionen von **ps**. Die eine greift direkt auf den Kernelspeicher zu, aus dem sie die Prozeßtabelle ausliest. Dazu braucht **ps** die Datei */etc/psdatabase*, in der die Speicheradressen für die entsprechenden Kernelvariablen abgelegt sind. Diese Datei muß für jeden Kernel mit dem Kommando **ps -U** neu erzeugt werden. Bei größeren Veränderungen am Kernel (in der Regel bei neuen Kernelversionen) wird auch ein Neuübersetzen von **ps** notwendig.

Das andere **ps** hat die gleiche Funktionalität und mit Ausnahme der **-U** Option auch die gleichen Optionen, es arbeitet aber mit dem Prozeßdateisystem. Dieses Dateisystem enthält Verzeichnisse für alle Prozesse des Systems, in deren Unterverzeichnissen und Dateien alle für **ps** interessanten Daten zu finden sind. Das **ps** Kommando bereitet diese Daten auf und zeigt sie dem Anwender in genau der gleichen Weise an wie die andere Version. Der Vorteil der Methode mit dem Prozeßdateisystem besteht in der Unabhängigkeit von der Kernelversion.

Das Verzeichnis, auf dem das Prozeßdateisystem aufgesetzt ist, kann in der aktuellen Version des **procps** nicht angegeben werden. Es erwartet das Prozeßdateisystem unter dem Verzeichnis */proc*.

Optionen

-a

zeigt die Prozesse aller User

-c

zeigt den Namen des Kommandos

-e

zeigt die Prozeßumgebung

-h

unterdrückt die Kopfzeile

-j

jobs Format: **PGID** und **SID**

-l

langes Format: **FLAGS WCHAN NICE PRIO**

-m

zeigt Speichernutzung

-X

zeigt **EIP ESP TIMEOUT** und **ALARM**

-n

gibt numerische Werte für **USER** und **WCHAN**

-r

zeigt nur die laufenden Prozesse

-s

zeigt die Signale

-u

zeigt die Besitzer der Prozesse

-v

vm Format

-w

ausführliche Ausgabe, kann mehrmals angegeben werden

-x

zeigt Prozesse, die von keinem Terminal kontrolliert werden

-S

addiert die Prozessorzeit der Kindprozesse zu den Eltern

-U

aktualisiert die Datei */etc/psdatebase*, die den Zugang zu den Kerneldaten vermittelt; diese Aktualisierung muß immer durchgeführt werden, nachdem der Kernel neu übersetzt wurde; diese Option fällt bei dem **ps** Programm, das mit dem Procdateisystem arbeitet, weg

t *xx*

zeigt nur die Prozesse die von Terminal *xx* kontrolliert werden

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Branko Lankester

Name

pwck - Überprüfung der Integrität der Passwort-Dateien

Syntax

pwck [**-r**] [*passwd shadow*]

Beschreibung

pwck überprüft die Integrität der System-Authentifizierungsinformation. Alle Einträge in */etc/passwd* und */etc/shadow* werden auf ihr korrektes Format hin überprüft und es wird sichergestellt, daß jedes Feld gültige Werte besitzt. Der Anwender (der Systemverwalter) wird dazu aufgerufen, Einträge zu löschen, die ungültige Formate vorweisen oder andere - nicht automatisch reparierbare - Fehler vorweisen.

Die Überprüfung stellt sicher, daß jeder Eintrag folgende Eigenschaften besitzt:

- Die richtige Anzahl von Feldern
- Einen eindeutigen Usernamen
- Eine gültige User- und GruppenID
- Eine gültige Logingruppe
- Ein gültiges Homeverzeichnis
- Eine gültige Startshell

Die Überprüfung der richtigen Anzahl von Feldern und einheitlichen Usernamen sind schwerwiegend. Wenn ein Eintrag eine falsche Anzahl Felder vorweist, wird der Anwender aufgefordert, den Eintrag zu löschen. Antwortet der Anwender nicht mit ja, so wird die weitere Überprüfung abgebrochen. Ein Eintrag mit doppeltem Usernamen führt auch zu so einer Aufforderung zum Löschen. Wird diese mit nein beantwortet werden aber trotzdem alle

weiteren Überprüfungen vorgenommen. Alle anderen Fehler werden durch Warnungen angezeigt und der Systemverwalter wird aufgefordert, sie mit Hilfe des Programms **usermod** zu ändern.

Optionen

Voreingestelltermaßen arbeitet **pwck** mit den Dateien */etc/passwd* und */etc/shadow*. Der Anwender kann mit den *passwd*- und *shadow*-Parametern alternative Dateien angeben. Zusätzlich kann mit der Option **-r** ein ReadOnly Modus erzwungen werden, in dem **pwck** keine Veränderungen an den Dateien vornehmen kann. Das entspricht der grundsätzlichen Antwort "no" auf alle Fragen.

Dateien

/etc/passwd - Useraccount Informationen
/etc/shadow - Verschlüsselte Passwordinformationen
/etc/group - Gruppeninformationen

Siehe auch

usermod(8) , **group**(5) , **passwd**(5) , **shadow**(5)

Diagnose

Das **pwck** Kommando gibt folgende Rückgabewerte zurück:

0
: Erfolgreiche Ausführung

1
: Syntax Error

2
: Ein oder mehrere schlechte Passwort-Einträge

3
: Kann Passwort-Dateien nicht öffnen

4
: Kann Passwort-Dateien nicht sperren

5
: Kann Passwort-Dateien nicht sichern

Autor

Julianne Frances Haugh (jfh@bga.com)

Bezeichnung

renice - verändert Prioritäten laufender Prozesse

Syntax

renice priority [[-p] pid ...] [[-g] pgrp ...] [[-u] user ...]

Beschreibung

Renice ändert die Zeitscheiben-Priorität eines oder mehrerer laufender Prozesse. Die angegebenen Parameter werden als ProzessIDs, Prozess GruppenIDs oder Usernamen interpretiert. Das Ändern von Prozess-Gruppen bezieht sich auf alle Prozesse der angegebenen Gruppe, das Ändern von Usernamen auf alle Prozesse des angegebenen Users. Voreingestellt bezieht sich renice auf die ProzessID

Optionen

-g

Zwingt renice, den folgenden Parameter als GruppenID zu interpretieren

-u

Zwingt renice, den folgenden Parameter als Usernamen zu interpretieren

-p

Zwingt renice, den folgenden Parameter als ProzessID zu interpretieren

Beispiel

```
renice +1 987 -u daemon root -p 32
```

würde die Priorität der Prozesse mit den IDs 987 und 32 sowie aller Prozesse der User `daemon` und `root` verändern.

User, die nicht der Superuser sind dürfen nur ihre eigenen Prozesse verändern und dürfen den Nice-Wert nur vergrößern (von 0 bis PRIO_MAX (20)). Der Superuser darf die Prioritäten aller Prozesse ändern und ihnen jeden beliebigen Nice-Wert zwischen PRIO_MIN (-20) und PRIO_MAX (20) zuweisen. Nützliche Nice-Werte sind: 20 (der Prozess bekommt nur dann Rechenzeit, wenn kein anderer Prozess Rechenzeit anfordert), 0 (Die Grundeinstellung), alle negativen Werte bis -20 (um Dinge zu beschleunigen)

Dateien

/etc/passwd um Usernamen in UserIDs umzuwandeln

Siehe auch

getpriority(2), setpriority(2)

Bugs

Nicht-Super-User können ihre Prozess-Prioritäten nicht erhöhen (den Nice-Wert niedriger machen), selbst wenn sie es waren, die den Nice-Wert zuerst erhöht haben.

Geschichte

Das renice-Kommando tauchte zuerst unter 4.0BSD auf.

Bezeichnung

rm - löscht Dateien

Syntax

rm [**-dfirvR**] [**--directory**] [**--force**] [**--interactive**] [**--recursive**] [**--verbose**] *Pfad* ...

Beschreibung

rm löscht Dateien. Normalerweise werden die Verzeichnisse nicht mitgelöscht. Wenn eine Datei gelöscht werden soll, für die keine Schreibberechtigung besteht, muß der Befehl für diese Datei extra bestätigt werden. In Verzeichnissen, bei denen das Stickybit gesetzt ist, kann eine Datei nur von ihrem Eigentümer gelöscht werden.

Die Option **--** zeigt an, daß die folgenden Argumente keine Optionen mehr sind. Dadurch ist es möglich auch Dateinamen, die mit einem **-** anfangen, zu löschen.

Optionen

-d

löscht Verzeichnisse mit dem **unlink** Systemaufruf, anstelle von **rmdir** (nur für den Superuser **root**); weil die in einem so gelöschten Verzeichnis enthaltenen Dateien nicht mitgelöscht werden, ist eine anschließende Reparatur des Dateisystems angesagt.

-f

keine Nachfragen, keine Fehlermeldungen

-i

vor dem Löschen jeder Datei wird nochmal nachgefragt

-r

der Inhalt aller Unterverzeichnisse und die Verzeichnisse werden mitgelöscht

-v

zeigt die Namen aller Dateien noch ein letztes mal an, bevor sie gelöscht werden

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Paul Rubin, David MacKenzie und

Richard Stallman

Bezeichnung

rmdir - löscht Verzeichnisse

Syntax

rmdir [**-p**] [**--path**] *Verzeichnis* ...

Beschreibung

rmdir löscht leere Verzeichnisse. Es gibt keine Möglichkeit, Verzeichnisse zu löschen, die noch normale Dateien enthalten.

Optionen

-p

löscht mehrere Verzeichnisse rekursiv, wenn alle Verzeichnisse leer sind (nachdem das Verzeichnis im Verzeichnis gelöscht ist)

Siehe Auch

das LunetIX Linuxhandbuch

Autor

David MacKenzie

Bezeichnung

sed - (stream editor) ist ein Editor zur nicht-interaktiven Textbearbeitung

Syntax

sed [**-nV**] [**--quiet**] [**--silent**] [**--version**] [**-e** *Editorkommando*] [**-f** *Scriptdatei*] [**--expression=** *Editorkommando*] [**--file=** *Scriptdatei*] [*Datei*...]

Beschreibung

sed ist ein Editor zur automatischen Textbearbeitung.

Die Bearbeitung erfolgt mit Editorkommandos, die dem **sed** in einer separaten *Scriptdatei* oder direkt in der Kommandozeile übergeben werden. Um in der Kommandozeile mehrere *Editorkommandos* zu übergeben, kann die **-e** Option mehrfach verwendet werden. Die Editorkommandos können auch durch ein Semikolon getrennt werden. Wird nur ein einziges Editorkommando in der Kommandozeile übergeben, kann die Kennzeichnung mit der **-e** Option auch weggelassen werden. Damit die Shell keine Veränderungen an der Zeichenkette mit dem Editorkommando vornimmt, muß sie in Hochkommata eingeschlossen werden.

Eine Scriptdatei kann beliebig viele Editorkommandos enthalten, die durch Zeilenende oder Semikolon von einander getrennt werden müssen.

Jedes Kommando besteht aus einem Adreßteil und einem Funktionsteil. Der Adreßteil gibt an, welche Zeilen einer Textdatei mit diesem Kommando bearbeitet werden sollen, und der Funktionsteil beschreibt die Veränderung, die an den im Adreßteil bestimmten Zeilen vorgenommen werden soll. Wenn kein Adreßteil angegeben ist, wird die Funktion mit jeder Zeile ausgeführt.

Die Bearbeitung eines Textes erfolgt, indem die Eingabe zeilenweise in einen Arbeitsspeicher gelesen wird, und dann die Adreßteile aller Editorkommandos der Reihe nach mit dem Text im Arbeitsspeicher verglichen werden.

Die Funktionen der passenden Kommandos werden in der Reihenfolge ihres Auftretens ausgeführt. Normalerweise wird nach der Bearbeitung aller Kommandos der Inhalt des Arbeitsspeichers auf die Standardausgabe ausgegeben und danach durch die nächste Eingabezeile ersetzt. Die automatische Ausgabe des Arbeitsspeichers nach jeder Zeile kann mit der Option **-n** unterdrückt werden.

Zusätzlich zu dem Arbeitsspeicher gibt es noch einen Zwischenspeicher (Puffer), der von verschiedenen Funktionen benutzt werden kann.

Der Arbeitsspeicher kann auch mehrere Zeilen auf einmal enthalten.

Im Adreßteil können die Zeilen entweder durch ihre Zeilennummern, oder durch reguläre Ausdrücke ausgewählt werden. Alle Funktionen außer den **a**, **i**, **q** und **=** akzeptieren einen Adressbereich, bei dem eine Start- und eine Endadresse durch ein Komma getrennt angegeben werden. Ein Dollarzeichen steht für die letzte Zeile. Wenn eine Endadresse mit einem regulären Ausdruck bezeichnet ist, wird die erste passende Zeile als Bereichsende eingesetzt.

Reguläre Ausdrücke müssen in einfachen Schrägstrichen (Slashes /) eingeschlossen werden. **sed** benutzt die gleichen Routinen zur Auswertung regulärer Ausdrücke wie **emacs** oder **grep**. Darüberhinaus kann auch die an die **sed** Syntax angelehnte Konstruktion **\#*Muster*\#** (mit jedem beliebigen Zeichen für \#) benutzt werden, die wie /*Muster*/ interpretiert wird.

Im *Muster* kann auch ein **\n** vorkommen, das auf das Zeilenende paßt.

Der sed kann folgende Funktionen ausführen:

a*Text*

schreibt den *Text* in die Standardausgabe, bevor die nächste Eingabezeile gelesen wird

b *Marke*

springt zur der mit der : *Marke* markierten Zeile im Script (nicht im Text) und fährt dort mit dem Programm fort

c*Text*

die im Arbeitsspeicher von **sed** befindliche Zeilen werden gelöscht, und der Text in die Standardausgabe geschrieben; wenn ein Adressbereich angegeben ist, wird der Text erst am Bereichsende einmal ausgegeben

d

alle aktuell im Arbeitsspeicher von **sed** befindlichen Zeichen werden gelöscht und die nächste Eingabezeile gelesen; die auf diesen Befehl folgenden Befehle werden nicht mehr bearbeitet, auch wenn die Zeilen im

Arbeitsspeicher im passenden Bereich liegen

D

die erste Zeile im Arbeitsspeicher von **sed** wird gelöscht und die nächste Zeile wird gelesen; die auf diesen Befehl folgenden Befehle werden nicht mehr bearbeitet, auch wenn die Zeilen im Arbeitsspeicher im passenden Bereich liegen

g

der Arbeitsspeicher von **sed** wird durch den Inhalt des Puffers ersetzt; der Inhalt des Arbeitsspeichers geht dabei verloren

G

der Pufferinhalt wird an den Inhalt des Arbeitsspeichers angehängt

h

der Inhalt des Arbeitsspeichers wird in den Puffer geschrieben; der Inhalt des Puffers geht dabei verloren

H

der Inhalt des Arbeitsspeichers von **sed** wird an den Puffer angehängt

i*Text***

(insert) der Text wird sofort in die Standardausgabe geschrieben

l

der Inhalt des Arbeitsspeichers von **sed** wird ausgegeben; nichtdruckbare Zeichen werden als Oktalzahl dargestellt

n

der Inhalt des Arbeitsspeichers wird unverändert in die Ausgabe geschrieben und der Arbeitsspeicher durch die nächste Eingabezeile ersetzt

N

die nächste Eingabezeile wird an den Arbeitsspeicher angehängt; das Zeilenende wird mit in den Arbeitsspeicher geschrieben; die Zeilennummer des aktuellen Bereiches erhöht sich um eins

p

der Inhalt des Arbeitsspeichers wird in die Standardausgabe geschrieben

P

die erste Zeile im Arbeitsspeicher wird in die Standardausgabe geschrieben

q

beendet **sed**; es werden keine weiteren Befehle ausgeführt und keine Zeilen mehr gelesen

r *Datei*

der Inhalt der Datei wird ausgegeben, bevor die nächste Zeile gelesen wird

s/*Ausdruck/Ersetzung/*[*Modus*]

(substitute) ersetzt den (ersten) auf den regulären Ausdruck passenden Text durch den Ersetzungstext; es kann auch ein beliebiges anderes Zeichen anstelle von **/** benutzt werden; als Modus können ein oder mehrere der folgenden angegeben werden

n

eine Zahl von 1 bis 512 ersetzt nur das n-te Auftreten des Musters

g

(global) alle auf den Ausdruck passenden Textteile werden ersetzt

p

wenn eine Ersetzung stattgefunden hat,

wird der Inhalt des Arbeitsspeichers von **sed** in die Standardausgabe geschrieben

w *Datei*

wenn eine Ersetzung stattgefunden hat, wird der Inhalt des Arbeitsspeichers in die *Datei* geschrieben

t *Marke*

verzweigt zur mit der :*Marke* versehenen Zeile in der Programmdatei, wenn eine Ersetzung am Inhalt des Arbeitsspeichers vorgenommen wurde, seit die letzte Eingabezeile gelesen wurde, oder seit der letzte **t** Befehl bearbeitet wurde; wenn keine *Marke* angegeben ist, wird an das Ende der Programmdatei verzweigt

w *Datei*

schreibt den Inhalt des Arbeisspeichers in die benannte *Datei*

x

vertauscht den Inhalt des Puffers mit dem Arbeitsspeicher

y/*Zeichenkette1*/*Zeichenkette2*/

vertauscht jedes auftretende Zeichen aus der Zeichenkette1 mit dem entsprechenden Zeichen der Zeichenkette2; die beiden Zeichenketten müssen gleich lang sein

! *Funktion*

führt die Funktion für alle Zeilen aus, die NICHT in den Bereich passen

:*Marke*

setzt eine *Marke* für den b und den t Befehl

{...}

die von den Klammern eingeschlossenen und durch Zeilenende oder Semikolon getrennten Funktionen werden als Einheit behandelt

=

gibt die aktuelle Eingabezeilennummer aus

\\#

leitet einen Kommentar ein; alle folgenden Zeichen bis zum Zeilenende werden ignoriert

Optionen

-n

gibt nur die Zeilen aus, die explizit (durch die Anweisung p) ausgedruckt werden sollen

-V

gibt die Versionsnummer und eine Kurzhilfe aus

-e*Zeichenkette*

wendet die Editorbefehle aus *Zeichenkette* auf den Text an

-f *Datei*

liest die Editorbefehle aus der *Datei*

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Unbekannt

Bezeichnung

sort - sortiert die Zeilen einer Textdatei

Syntax

sort [**-cmus**] [**-t** *Separator*] [**-o** *Ausgabedatei*] [**-bdfiMnr**] [**+** *POS1* [**-** *POS2*]] [**-k** *POS1* [**,** *POS2*]] [*Datei*...]

Beschreibung

sort wird normalerweise zum Sortieren von Dateien verwendet. Es kann aber auch Dateien daraufhin überprüfen, ob sie sortiert sind; oder mehrere sortierte (oder unsortierte) Dateien zu einer sortierten zusammenfügen.

Dazu existieren drei Modi:

- der

  check Modus, der prüft, ob eine Datei bereits sortiert ist. Dieser Modus wird durch die Option **-c** eingeleitet

- der

  merge Modus, der mehrere vorsortierte Dateien zusammenfügt. Die Dateien so zusammenzufügen ist schneller, als sie komplett sortierten zu lassen. Dieser Modus wird durch die Option **-m** eingeleitet

- der Standardmodus ist

  sort

Wenn Schlüsselfelder bezeichnet sind, vergleicht **sort** die Schlüsselfelder in der Reihenfolge ihrer Bezeichnung, bis ein Unterschied gefunden wurde oder keine weiteren Felder vorhanden sind.

Wenn eine der globalen Optionen **Mbdfinr** benutzt wird, und kein Schlüsselfeld angegeben ist, vergleicht **sort**

die ganzen Zeilen.

Optionen

-b

ignoriert führende Leerzeichen

-c

stellt fest, ob die Dateie(en) bereits sortiert ist/sind; wenn eine Datei nicht sortiert ist, wird eine Fehlermeldung ausgegeben und mit dem Status 1 abgebrochen

-d

sortiert in alphabetischer Reihenfolge

-f

unterscheidet nicht zwischen Groß- und Kleinschreibung

-i

ignoriert alle nicht druckbaren Zeichen (außerhalb 040-126 ASCII)

-M

sortiert die (amerikanischen) Monate jan feb mar ... dec in der korrekten Reihenfolge; führende Leerzeichen werden wie bei **-b** ignoriert

-m

fügt bereits sortierte Dateien Zeilenweise zusammen

-n

sortiert Zeilen mit Zahlen; ignoriert führende Leerzeichen und behandelt - als Vorzeichen

-r

sortiert in umgekehrter Reihenfolge

-o *Datei*

schreibt in die *Datei* anstelle der Standardausgabe; wenn eine der Eingabedateien als Ausgabedatei bestimmt wird, legt **sort** erst eine Kopie der Eingabedatei an und sortiert dann in die Ausgabedatei

-t *Separator*

benutzt *Separator* als Feldtrenner für die Suchschlüssel; Standard ist der Leerstring zwischen einem Nichtblank und einem Blank; Der Trenner ist nicht Teil eines der getrennten Felder

-u

im merge Modus wird nur die erste von einer Reihe gleichwertiger Zeilen ausgegeben; im check Modus wird geprüft, ob nicht zwei Zeilen gleichwertig sind

+*POS1* **[-***POS2***]**

bestimmt die Zeichen zwischen *POS1* und *POS2* zum Sortierschlüssel; wenn *POS2* fehlt, werden alle Zeichen bis zum Zeilenende zum Schlüssel; Positionen der Felder und Buchstaben zählen von 0

-k *POS1***[,***POS2***]**

bestimmt die Zeichen zwischen *POS1* und *POS2* zum Sortierschlüssel; wird das Schlüsselfeld so spezifiziert, zählen die Felder und Buchstaben von 1

Eine Position hat die Form *feld.buchstabe*

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Mike Haertel

Bezeichnung

split - spaltet eine Datei in mehrere kleinere

Syntax

split [**-** *Zeilen*] [**-l** *Zeilen*] [**-b** *Bytes* [**bkm**] [**-C** *Bytes* [**bkm**]] [**--lines=** *Zeilen*] [**--bytes=** *Bytes* [**bkm**]] [**--line-bytes=** *Bytes* [**bkm**]] [*Datei* [*Prefix*]]

Beschreibung

split teilt eine Datei in mehrere Teile. Wenn keine weiteren Optionen gegeben sind, wird die *Datei* in Teile zu je 1000 Zeilen aufgeteilt. Die Ausgabe erfolgt in Dateien mit der Endung *Prefix* oder **x** wenn kein Prefix angegeben wird.

Optionen

-*Zeilen*

die Ausgabedateien sind *Zeilen* lang

-l *Zeilen*

die Ausgabedateien sind *Zeilen* lang

-b ***Bytes***[**bkm**]

die Ausgabedateien sind *Bytes* lang; die optionale Endung setzt die Einheit auf

b

512 Byte Blöcke

k

1 Kilobyte (1024) Blöcke

m

1 Megabyte Blöcke

-C ***Bytes***[**bkm**]

schreibt so viele Zeilen wie möglich in die Ausgabedatei, ohne *Bytes* zu überschreiten; ist eine Zeile länger als *Bytes*, wird die Zeile auf mehrere Dateien aufgeteilt, bis der Rest weniger als *Bytes* lang ist; die Optionen **bkm** werden benutzt wie bei **-b**

Siehe Auch

csplit(1) und das LunetIX Linuxhandbuch

Autor

tege@sics.se

Bezeichnung

tac - wie cat, nur umgekehrt

Syntax

tac [**-br**] [**-s** *Trenner*] [**--before**] [**--regex**] [**--separator=** *Trenner*] [*Datei*...]

Beschreibung

tac arbeitet in vieler Hinsicht wie **cat**, es werden die einzelnen Felder der *Datei* zeilenweise ausgegeben, allerdings das letzte zuerst. Als Feldtrenner dient das Zeilenende, wenn kein anderer angegeben ist. Der Feldtrenner wird zu dem Feld gezählt, das er abschließt, also an dessen Ende ausgegeben.

Optionen

-b

(before) der Feldtrenner wird zum darauffolgenden Feld gezählt, also an dessen Anfang ausgegeben.

-r

der Feldtrenner ist ein regulärer Ausdruck

-s *Trenner*

benutzt *Trenner* als Feldtrenner

Siehe Auch

cat (1) und das LunetIX Linuxhandbuch

Autor

Jay Lepreau, David MacKenzie

Bezeichnung

tail - zeigt das Ende einer Datei

Syntax

tail [**-c** [+] *N* [**bkm**]] [**-n** [+] *N*] [**-fqv**] [**--bytes=** [+] *N* [**bkm**]] [**--lines=** [+] *N*] [**--follow**] [**--quiet**] [**--silent**] [**--verbose**] [*Datei*...]

tail [{-,+}**Nbcfklmqv**] [*Datei*...]

Beschreibung

tail druckt die letzten (10) Zeilen einer *Datei* oder von der Standardeingabe, wenn keine Datei angegeben wird. Ein einzelnes **-** anstelle eines Dateinamens meint ebenfalls die Standardeingabe. Werden mehrere Dateien angegeben, so wird das Ende jeder Datei mit dem Dateinamen eingeschlossen in ==> und <== eingeleitet.

Optionen

-c *N*

zeigt *N* Bytes vom Ende der *Datei*. Der Anzahl kann eine Einheit folgen. Möglich sind:

b

Blöcke mit 512 Bytes

k

Blöcke mit Kilobytes

m

Blöcke mit Megabytes

-f

(follow) gibt immer wieder das Dateiende aus, dadurch kann die Entwicklung einer wachsenden Datei beobachtet werden; diese Option funktioniert nur, wenn nur eine einzige Datei angegeben ist

-l *N*

gibt *N* Zeilen aus

-q

unterdrückt die Dateinamen zu Beginn der Ausgabe

-v

druckt immer die Dateinamen zu Beginn der Ausgabe

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Paul Rubin, David MacKenzie

Bezeichnung

tee - verzweigt die Ausgabe auf eine Datei

Syntax

tee [**-ai**] [**--append**] [**--ignore-interrupts**] [*Datei*...]

Beschreibung

tee liest von der Standardeingabe und verzweigt die Ausgabe auf die Standardausgabe und *Datei*. Wird auf eine existierende Datei verzweigt, so wird sie überschrieben, anderenfalls wird sie angelegt.

Die Ausgabe von make bzw gcc beim compilieren eines Programms kann beispielsweise mit

make -k 2>&1 | tee make.out (**bash**-Syntax)

in der Datei make.out gesichert werden.

Optionen

-a

die *Datei* wird nicht überschrieben, sondern die Ausgabe daran angehängt

-i

ignoriert Interrupt Signale

Siehe Auch

das LunetIX Linuxhandbuch

Autor

Mike Parker, Richard M. Stallman und

David MacKenzie

Bezeichnung

top - Zeigt laufende CPU-Prozesse

Syntax

top [-] [**d** *delay*] [**p** *pid*] [**q**] [**c**] [**S**] [**s**] [**i**] [**n** *iter*] [**b**]

Beschreibung

top bietet einen fortlaufenden Blick auf die Prozessoraktivitäten in Echtzeit. Es zeigt eine Liste der Prozesse, die die meiste CPU-Zeit nutzen und kann eine Schnittstelle zur Manipulation der Prozesse anbieten. Es kann die Tasks nach CPU-Nutzung, Speicher-Nutzung und Laufzeit sortieren, kann besser konfiguriert werden, als das Standard-top aus dem procps-Paket. Die meisten Features können entweder durch interaktive Kommandos oder durch Einträge im persönlichen oder systemweiten Konfigurationsdateien ausgewählt werden. Siehe weiter unten für mehr Informationen.

Kommandozeilen-Optionen

d

Spezifiziert die Wartezeit zwischen den Updates des Bildschirms. Dieser Wert kann mit dem interaktiven Kommando **s** während der Laufzeit geändert werden.

p

Zeigt nur die Prozesse mit den angegebenen PID. Diese Option kann bis zu 20 mal angegeben werden. Diese Option ist weder interaktiv nutzbar, noch kann sie in einer Konfigurationsdatei eingetragen werden.

q

Veranlasst **top** ohne jede Wartezeit zwischen den Bildschirm-Updates zu arbeiten. Wenn der Aufrufer Superuser Privilegien hat arbeitet top mit der höchstmöglichen Priorität.

S

Spezifiziert kumulativen (anwachsenden) Modus, in dem jeder Prozess mit der CPU-Zeit aufgeführt wird, die er selbst und all seine abgeschlossenen Kind-Prozesse verbraucht haben. Entspricht dem **-S** Flag von **ps**(1). Siehe weiter unten bei der Besprechung des interaktiven Kommandos **S**.

s

Weist **top** an, im sicheren Modus zu laufen. Das schaltet alle potentiell gefährlichen interaktiven Kommandos inaktiv (siehe unten). Ein sicheres **top** ist eine elegante Lösung für ein Überwachungsterminal.

i

Startet **top** indem alle untätigen und Zombie Prozesse ignoriert werden. Siehe das interaktive Kommando **i** weiter unten.

c

Zeigt die ganze Kommandozeile statt nur den Kommandonamen. Die voreingestellte Eigenschaft, weil sich das als brauchbarer herausgestellt hat.

n

Anzahl der Wiederholungen. **top** arbeitet die angegebene Anzahl von Bildschirm-Updates ab und beendet sich dann.

b

Batch Modus. Nützlich, um die Ausgabe von **top** an andere Programme oder Dateien weiterzuleiten. In diesem Modus akzeptiert **top** keine interaktiven Kommandos. Es läuft solange, bis die mit der **n** Option angegebene Anzahl Wiederholungen abgelaufen ist oder es gekillt wurde. Die Ausgabe ist reiner Text, brauchbar auch für "dumme" Terminals.

Feld Beschreibungen

top zeigt eine Vielzahl von Informationen über den Prozessor-Status. Voreingestellt ist ein Refresh des Displays alle 5 Sekunden, aber das kann mit der **d** Kommandozeilenoption oder mit dem interaktiven Kommando **s** geändert werden.

uptime

Diese Zeile zeigt die Zeit, die das System bereits läuft und drei Durchschnittswerte für die Systemauslastung. Diese Durchschnittswerte zeigen die durchschnittliche Anzahl der Prozesse, die während der letzten 1, 5 und 15 Minuten "ready to run" waren. Diese Zeile entspricht dem Ausgabeformat des **uptime**(1) Befehls. Diese Zeile kann durch das interaktive Kommando `l` an- und ausgeschaltet werden.

processes

Die Gesamtanzahl der Prozesse, die zur Zeit des letzten Updates liefen. Diese Angabe ist noch weiter aufgespaltet in die Angabe der Prozesse, die laufen, schlafen, gestoppt sind oder Zombie-Prozesse sind. Diese Zeile kann mit dem interaktiven Kommando `t` an- und ausgeschaltet werden.

CPU states

Zeigt den Prozentsatz der CPU-Zeit im User-Modus, System-Modus, "niced"-Prozesse und untätig an. ("niced"-Prozesse sind nur diejenigen Prozesse, die einen negativen Nice-Wert besitzen.) Die Prozessorzeit der niced-Prozesse wird auch bei den User- und System-Prozessen angezeigt, so daß die Summe aller Prozesszeit mehr als 100% ist. Auch diese Zeile wird mit dem `t` Kommando aus- und angeschaltet.

Mem

Statistik der Speichernutzung, beinhaltet die Gesamtgröße des Speichers, die Größe des freien Speichers, gebrauchten Speichers, geteilten Speichers und Speichers, der für IO-Buffer reserviert ist. Diese Zeile kann mit dem interaktiven Kommando `m` ein- und ausgeschaltet werden.

Swap

Statistik des Swap-Speichers, beinhaltet die Gesamtgröße des Swap-Speichers, freier und benutzter Swap-Speicher.

PID

Die Prozess ID jedes Tasks.

PPID

The Eltern ProzessID (parent process ID) jedes Tasks.

UID

Die User ID des Eigentümers des Tasks.

USER

Der Username des Eigentümers des Tasks.

PRI

Die Priorität des Tasks.

NI

Der Nice-Wert des Tasks. Negative Nice-Werte bedeuten höhere Priorität.

SIZE

Die Größe des Programmspeichers plus Datenspeicher plus Stack des Tasks in Kilobyte.

TSIZE

Die Größe des Programmspeichers des Tasks. Diese Ausgabe gibt falsche Angaben für Kernel-Prozesse und ist aufgebrochen für ELF-Prozesse.

DSIZE

Data + Stack Größe. Aufgebrochen für ELF-Prozesse.

TRS

Residente Text Speichergröße.

SWAP

Größe des ausgelagerten Teils dieses Tasks.

D

Größe der Speicherseiten, die als "dirty" markiert sind.

LIB

Größe der Library Pages. Funktioniert nicht für ELF-Prozesse.

RSS

Die totale Größe des physikalischen Speichers, die von dem task benutzt wird in Kilobyte. Bei ELF-Prozessen zählen die Library-Pages hier mit, bei a.out-Prozessen nicht.

SHARE

Die Größe des geteilten Speichers des Tasks.

STAT

Der Status des Tasks. Status ist einer von **S** für schlafend, **D** für nicht unterbrechbar schlafend (dead), **R** für laufend (running), **Z** für Zombies, **T** für gestoppt. Dieser Wert wird noch ergänzt durch ein angehängtes < für Prozesse mit negativen Nice-Werten, **N** für Prozesse mit positiven Nice-Werten und **W** für ausgelagerte Prozesse (das funktioniert nicht korrekt für Kernel-Prozesse).

WCHAN

Abhängig von der Verfügbarkeit von entweder */boot/psdatabase* oder der Kernel-Link-Map */boot/System.map* die Adresse oder der Name der Kernel-Funktion, die der Task augenblicklich benutzt.

TIME

Gesamte CPU-Zeit, die der Task seit seinem Start benutzt hat. Wenn der kumulative Modus aktiviert ist, auch die Zeit der abgeschlossenen Kindprozesse des Tasks. Kumulativer Modus kann entweder durch die Kommandozeilenoption **S** oder durch den interaktiven Befehl **S** aktiviert werden. In diesem Modus heißt dieses Feld dann **CTIME** statt TIME.

%CPU

Der Anteil der CPU-Zeit des Tasks seit dem letzten Bildschirm-Update, ausgedrückt als Prozentsatz der totalen CPU-Zeit pro Prozessor.

%MEM

Der Anteil an physikalischem Speicher des Tasks.

COMMAND

Der Kommandoname des Tasks, der abgeschnitten wird, wenn er zu lang ist, um noch dargestellt zu werden. Tasks im Arbeitsspeicher werden mit ganzer Kommandozeile dargestellt, Tasks im Swap nur mit dem Kommandonamen in Klammern.

A , WP

Diese Felder des kmem-top werden nicht unterstützt.

Interaktive Kommandos

Verschiedenste Ein-Buchstaben Kommandos werden erkannt, während **top** läuft. Manche von ihnen sind deaktiviert, wenn die **s** Option auf der Kommandozeile angegeben war.

Leertaste

Das Display wird sofort aufgefrischt.

Strg-L

Löscht den Bildschirm und stellt ihn neu dar.

h oder ?

Zeigt einen Hilfebildschirm, der eine kurze Beschreibung der Kommandos und den Status von Sicherem- und Kumulativem Modus enthält.

k

Sendet ein Signal an einen Prozess (kill). Sie werden nach der PID des Prozesses und der Signalnummer gefragt, die dem Prozess geschickt werden soll. Dieses Kommando ist im Sicheren Modus nicht verfügbar.

i

Ignorieren von untätigen und Zombie-Prozessen. Das ist ein An-/Ausschalter.

n oder #

Ändert die Anzahl der angezeigten Prozesse. Sie werden nach der neuen Anzahl gefragt. Diese Einstellung überschreibt die automatische Erkennung der Zeilenanzahl des Fensters, in dem **top** läuft. Wird 0 eingegeben, so zeigt **top** so viele Prozesse, wie auf den Bildschirm passen. Das ist die Grundeinstellung.

q

Quit - Beendet das Programm.

r

Re-nice. Der Nice-Wert eines Prozesses wird geändert. Sie werden nach der PID des Prozesses und dem neuen Nice-Wert gefragt. Ein positiver Wert verringert die Priorität, ein negativer erhöht die Priorität. Nur root darf einen negativen Wert eingeben. Dieses Kommando steht im sicheren Modus nicht zur Verfügung.

S

Schaltet den kumulativen Modus ein und aus. Das heißt, daß die CPU-Zeit nicht nur für einen Prozess gerechnet wird, sondern auch die Zeit der abgeschlossenen Kindprozesse dieses Prozesses mitgerechnet wird.

s

Wechselt die Wartezeit zwischen den Bildschirm-Updates. Sie werden nach der Wartezeit (in Sekunden) gefragt. Gebrochene Werte werden bis zu Microsekunden erkannt. Eine 0 meint kontinuierliches Update ohne Wartezeit. Dieses Kommando steht im sicheren Modus nicht zur Verfügung.

f oder F

Neue Felder dem Display hinzufügen oder bestehende entfernen. Siehe weiter unten.

o oder O

Wechsel der Sortierreihenfolge der dargestellten Felder. Siehe unten.

l

Schaltet die Anzeige der Durchschnittsauslastung und Uptime an oder aus.

m

Schaltet die Anzeige der Speicherauslastung ein oder aus.

t

Schaltet die Anzeige von Prozess- und CPU Status ein oder aus.

c

Wechselt zwiscen der Darstellung von Kommandonamen und vollen Kommandozeilen.

N

Sortiert die Tasks nach PID.

A

Sortiert die Tasks nach Alter (neuester zuerst).

P

Sortiert die Tasks nach CPU-Benutzung (Voreingestellt).

M

Sortiert die Tasks nach Speichernutzung.

T

Sortiert die Tasks nach Zeit/kumulativer Zeit.

W

Schreibt die aktuellen Einstellungen nach *~/.toprc*. Das ist der beste Weg, eine Konfigurationsdatei für **top**

Der Feld und Reihenfolge-Bildschirm

Nachdem Sie **f**, **F**, **o** oder **O** eingegeben haben, wird Ihnen ein Bildschirm angezeigt, der die Reihenfolge der Feldanzeige in der obersten Zeile beschreibt und Kurzbeschreibungen der Feldinhalte zeigt. Die Zeichenkette für die Feldreihenfolge benutzt die folgende Syntax: Wenn ein Buchstabe in der Zeichenkette ein Großbuchstabe ist, wird das entsprechende Feld dargestellt. Das wird auch durch ein Sternchen vor der jeweiligen Feldbeschreibung angezeigt. Die Reihenfolge der Felder entspricht der Position der Zeichen in der Zeichenkette. Haben Sie den Bildschirm mit dem f oder F Befehl aufgerufen, so können Sie durch Drücken des entsprechenden Zeichens ein Feld hinzufügen oder deaktivieren. Haben Sie den Schirm mit dem o oder O Befehl aktiviert, so können Sie die Reihenfolge der Darstellung ändern. Sie bewegen das Feld um eine Position nach links, indem Sie den entsprechenden Großbuchstaben drücken, nach rechts, indem Sie den entsprechenden Kleinbuchstaben drücken.

Konfigurationsdateien

Top ließt seine voreingestellte Konfiguration aus zwei Konfigurationsdateien, */etc/toprc* und *~/.toprc*. Die globale Konfigurationsdatei kann benutzt werden, um den Gebrauch von top für normale User auf den sicheren Modus zu beschränken. Wenn das gewünscht ist, sollte die Datei ein "s" für den sicheren Modus und eine Zahl d (2<=d<=9)

für die voreingestellte Wartezeit zwischen den Bildschirmupdates in einer Zeile enthalten.

Die persönliche Konfigurationsdatei enthält zwei Zeilen. Die erste Zeile enthält Klein- und Großbuchstaben, die die Darstellung der Felder und ihre Reihenfolge beschreibt (siehe oben). Weil das nicht sehr einfach zu merken ist, wird vorgeschlagen, mit dem Feld- und Reihenfolgebildschirm eine passende Einstellung zu erstellen und diese mit dem W-Befehl zu sichern. Die zweite Zeile ist interessanter und wichtiger. Sie enthält die Information über die anderen Optionen. Sehr wichtig, wenn Sie eine Konfiguration im sicheren Modus gespeichert haben. Sie werden nie wieder ein unsicheres top bekommen, wenn Sie nicht den Buchstaben "s" aus der zweiten Zeile Ihrer *~/.toprc* entfernen.

Siehe auch

ps(1) , **free**(1) , **uptime**(1) , **kill**(1) , **renice**(1)

Bugs

Wenn das Fenster kleiner als 70x7 ist, kann top die Informationen nicht korrekt formatieren.

Viele Felder haben noch Probleme mit ELF-Binaries.

Der Hilfe-Bildschirm ist noch nicht für Fenster optimiert, die weniger als 25 Zeilen aufweisen.

Autoren

top was originally written by Roger Binns, based on Branko Lankester's <lankeste@fwi.uva.nl> ps program. Robert Nation <nation@rocket.sanders.lockheed.com> re-wrote it significantly to use the proc filesystem, based on Michael K. Johnson's <johnsonm@redhat.com> proc-based ps program. Michael Shields <mjshield@nyx.cs.du.edu> made many changes, including secure and cumulative modes and a general cleanup. Tim Janik <timj@gtk.org> added age sorting and the ability to monitor specific processes through their ids.

Helmut Geyer <Helmut.Geyer@iwr.uni-heidelberg.de> Heavily changed it to include support for configurable fields and other new options, and did further cleanup and use of the new readproc interface.

The "b" and "n" options contributed by George Bonser <george@captech.com> for CapTech IT Services.

Michael K. Johnson <johnsonm@redhat.com> is now the maintainer.

Please send bug reports to <procps-bugs@redhat.com>

Bezeichnung

touch - ändert die Zeitmarkierung einer Datei

Syntax

touch [**-acm**] [**-r** *Referenzdatei*] [**-t** *MMDDhhmm* [[*CC*]*YY*] [**.** *ss*]] [**-d** *Zeit*] [**--time=** {**time**,**access**,**use**,**mtime**,**modify**} [**--date=** *Zeit*] [**--file=** *Referenzdatei*] [**--no-create**] *Datei* ...

Beschreibung

touch setzt die Zugriffs- und die Änderungszeit der *Datei* auf die aktuelle Zeit. Wenn die *Datei* nicht existiert, wird sie erzeugt.

Wenn als erster Dateiname ein gültiges Argument für die {it -t/} Option übergeben wird, und keine der Optionen {it -d, -r/} oder {it -t} ausdrücklich angegeben ist, wird dieses Argument als Zeitmarke für die folgenden Dateien verwendet.

Optionen

-a

ändert nur die Zugriffszeit

-c

unterdrückt die Erzeugung nicht existierender Dateien

-d *Zeit*

setzt *Zeit* anstelle der aktuellen Uhrzeit. Für *Zeit* können verschiedene gebräuchliche Formate verwendet werden.

-m

ändert nur die Änderungszeit

-r *Referenzdatei*

setzt die Zeit von *Referenzdatei* anstelle der aktuellen Zeit

-t ***MMDDhhmm***[[*CC YY*][***.ss***]]

benutzt das Argument als Zeitangabe

Siehe Auch

das LunetIX Linuxhandbuch

Autoren

Paul Rubin, Arnold Robbins, Jim Kingdon, David

MacKenzie und Randy Smith

Bezeichnung

tr - Ändert oder löscht einzelne Zeichen

Syntax

tr [*OPTION*]... *Menge1* [*Menge2*]

Beschreibung

Übersetzt und/oder löscht Zeichen aus der Standard-Eingabe und schreibt auf die Standard-Ausgabe.

-c, **--complement**

Dreht den Wahrheitswert für Menge1 um (Nicht ...)

-d, **--delete**

Löscht Zeichen aus Menge1, keine Übersetzung

-s, **--squeeze-repeats**

Ersetzt Sequenzen gleicher Zeichen durch jeweils eines.

-t, **--truncate-set1**

Schneidet die Länge von Menge1 so ab, daß sie der Länge von Menge2 entspricht.

--help

Zeigt kurzen Hilfetext und beendet dann das Programm

--version

Zeigt Versionsinformationen und beendet dann das Programm

Mengen sind als Zeichenketten, bestehend aus einzelnen Zeichen spezifiziert. Die meisten Zeichen representieren sich selbst. Folgende Konstrukte werden speziell interpretiert:

\NNN
: Zeichen mit dem oktalen Wert NNN (1 bis 3 oktale Stellen)

\\
: Der Backslash

\a
: Klingelzeichen (BEL)

\b
: Backspace

\f
: Form Feed (Seitenvorschub)

\n
: New Line (Zeilentrenner)

\r
: Return

\t
: Horizontaler Tabulator

\v
: Vertikaler Tabulator

CHAR1-CHAR2
: Alle Zeichen von CHAR1 bis CHAR2 in aufsteigender Folge

[CHAR1-CHAR2]
: Das Gleiche wie CHAR1-CHAR2, wenn beide Mengen es gebrauchen

[CHAR*]
: Gültig für Menge2, Kopien der Zeichen, bis die Länge der Menge1 erreicht ist

[CHAR*REPEAT]
: REPEAT Wiederholungen von CHAR, REPEAT ist eine Zahl, die oktal interpretiert wird, wenn sie mit einer 0 beginnt, ansonsten dezimal

[:alnum:]
: Alle Buchstaben und Ziffern

[:alpha:]
: Alle Buchstaben

[:blank:]
: Alle horizontalen Whitespaces (Leerzeichen, Tab)

[:cntrl:]
: Alle Control-Sequenzen

[:digit:]
: Alle Ziffern

[:graph:]
: Alle druckbaren Zeichen - Leerzeichen nicht eingeschlossen

[:lower:]
: Alle Kleinbuchstaben

[:print:]
: Alle druckbaren Zeichen, Leerzeichen eingeschlossen

[:punct:]
: Alle Punktierungszeichen

[:space:]
: Alle horizontalen oder vertikalen Whitespaces

[:upper:]
: Alle Großbuchstaben

[:xdigit:]
: Alle hexadezimalen Ziffern (0-9, a-f, A-F)

[=CHAR=]
: Alle Zeichen, die äquivalent zu CHAR sind

Ersetzung der Zeichen wird vorgenommen, wenn **-d** nicht angegeben wurde und beide Menge1 und Menge2 erscheinen. **-t** darf nur für Ersetzung benutzt werden. Die Menge2 wird - falls sie kürzer als Menge1 ist - solange mit dem letzten Zeichen von Menge1 aufgefüllt, bis sich die Längen entsprechen. Überschüssige Zeichen der Menge2 werden ignoriert. Nur [:lower:] und [:upper:] garantieren, daß sie in aufsteigender Reihenfolge angewandt werden, wenn sie in Menge2 benutzt werden. Sie dürfen nur paarweise benutzt werden, um Groß- und

Kleinschreibung zu ändern. **-s** benutzt Menge1 wenn weder Ersetzung, noch Löschen vorliegt. Ansonsten benützt es Menge2 zum Zusammenziehen, nachdem die Ersetzung bzw. Löschung vorgenommen wurde.

Copyright

Bezeichnung

umask - bestimmt oder zeigt den voreingestellten Zugriffsmodus

Syntax

umask [-S]

umask ***Maske***

Beschreibung

Das **umask**-Kommando zeigt, bzw. verändert die Dateierzeugungsmaske, nach der berechnet wird, welche Zugriffsrechte eine neu erstellte Datei bekommt.

Wird umask ohne Parameter aufgerufen, so wird die aktuelle Maske ausgegeben.

Wird umask mit dem Parameter -S aufgerufen, so wird die eingestellte Zugriffsberechtigung symbolisch dargestellt etwa in der Form `u=rwx,g=rx,o=rx`.

Wird umask eine Maske als Parameter angegeben, so wird diese Maske in Zukunft verwendet, um die Zugriffsberechtigung neu angelegter Dateien festzulegen. Als Maske kommen zwei Formen in Betracht:

1. **Die oktale Form**
   In dieser Form errechnet sich die Maske durch die Formel `7-`*`gewünschtes_Recht`*`=`*`Maske`*. Das `rwx` Recht (7) wäre also 0, das `r-x` Recht (5) demnach 2 usw.
2. **Die symbolische Form**
   In dieser Form werden die gewünschten Zugriffsrechte symbolisch eingegeben, in genau der Form, die durch `umask -S` ausgegeben werden würde. Also nicht als Maske, sondern direkt als Zugriffsmodus. Die Form ist immer `u=...,g=...,o=...` wobei für `...` die entsprechenden Werte rwx, rx, oder ähnliches

eingegeben werden müssen. Soll kein Recht gesetzt werden, folgt dem Gleichheitszeichen nichts.
(`u=rwx,g=,o=`)

Das Ausführungsrecht (x) sollte in jedem Fall immer mindestens für den Eigentümer (u) gesetzt sein, da sonst die Verzeichnisse, die er erstellt, von ihm selbst nicht durchsucht werden können. Normale Dateien bekommen, auch wenn das x-Recht gesetzt ist dieses Recht nicht, wenn sie neu angelegt werden.

Name

useradd - Legt einen neuen User an oder verändert die Grundeinstellungen zum Anlegen neuer User

Syntax

useradd

[**-c** *Kommentar*] [**-d** *Homeverzeichnis*]
[**-e** *Auslaufdatum*] [**-f** *Pufferzeit*]
[**-g** *Login Gruppe*] [**-G** *Gruppe*[,...]]
[**-m** [**-k** *Template-Verzeichnis*]] [**-p** *Passwort*]
[**-s** *Startshell*] [**-u** *UserID* [**-o**]] *Username*

useradd

-D [**-g** *Voreingestellte Gruppe*] [**-b** *Voreingest. Homeverz.*]
[**-f** *Voreing. Pufferzeit*] [**-e** *Voreing. Auslaufdatum*]
[**-s** *Voreingest. Startshell*]

Beschreibung

Neue User anlegen

Wenn **useradd** ohne die Option **-D** aufgerufen wird, erstellt es einen neuen Useraccount, der die auf der Kommandozeile angegebenen Parameter und die vom System voreingestellten Werte benützt. Der neue Useraccount wird in die Systemdateien eingetragen, in die Einträge nötig sind. Abhängig von den angegebenen Kommandozeilenoptionen wird das Home-Verzeichnis erstellt und die initialen Dateien werden hineinkopiert. Die Optionen zum Anlegen neuer User sind:

-c Kommentar

Das Kommentarfeld des Eintrags des neuen Users in /etc/passwd

-d *Homeverzeichnis*

Der Eintrag für das Homeverzeichnis des neu anzulegenden Users in /etc/passwd. Der voreingestellte Wert ist der Username, der dem Wert des *voreigestellten Homeverzeichnisses* (siehe unten) nachgestellt wird.

-e *Auslaufdatum*

Das Datum, an dem der Useraccount ungültig werden soll. Dieses Datum wird im Format *JJJJ-MM-TT* angegeben.

-f *Pufferzeit*

Die Anzahl von Tagen, nach dem Auslaufen des Accounts, bis der Account permanent gesperrt wird. Ein Wert von 0 bedeutet keine Pufferzeit, ein Wert von -1 schaltet dieses Feature ab. Der voreingestellte Wert ist -1

-g *Login Gruppe*

Der Gruppenname oder die GruppenID der Login-Gruppe des neuen Users. Diese Gruppe muß existieren.

-G *group,[...]*

Eine Liste weiterer Gruppen, denen der User auch angehören soll. Jede angegebene Gruppe wird von der nächsten durch ein Komma getrennt. Auch diese Gruppen müssen bereits existieren. Voreingestellt ist, daß der neue User nur Mitglied seiner Login Gruppe ist.

-m

Wenn das Home-Verzeichnis des Users noch nicht existiert, so wird es angelegt. Die Dateien, die im *Template-Verzeichnis* existieren, werden in dieses Verzeichnis kopiert, wenn die **-k** Option gesetzt wurde, ansonsten werden die Dateien aus */etc/skel* kopiert. Auch entsprechende Verzeichnisse dort werden kopiert. Ohne die Angabe von **-m** wird weder das Homeverzeichnis erstellt, noch irgendwelche Dateien kopiert.

-p *Passwort*

Das verschlüsselte Passwort, wie es von crypt erstellt wird. Voreingestellt ist, den Account zu deaktivieren, indem ein * oder ! im Passwortfeld des /etc/shadow-Eintrags steht.

-s *Startshell*

Der Name der Loginshell des Users.

-u *UserID*

Die numerische UserID des neuen Users. Dieser Wert muß eindeutig sein, wenn nicht die Option **-o** angegeben wurde. Der Wert darf nicht negativ sein. Voreingestellt ist der Wert, der größer als 99 und größer als jeder andere bisher bestehende UID-Wert ist. Werte von 0 bis 99 sind typischerweise für

Systemaccounts reserviert.

Grundeinstellungen verändern

Wird **useradd** mit der Option **-D** aufgerufen, so zeigt es entweder die Voreinstellungen oder ändert sie, wie in der Kommandozeile angegeben. Die gültigen Optionen sind:

-b *Voreingest. Homeverzeichnis*

Das initiale Pfadpräfix für das Homeverzeichnis eines neuen Users. Wenn ein neuer User angelegt wird, ohne die **-d** Option, so wird der Name des Users an den hier angegebenen Wert angehängt.

-e *Voreingest. Auslaufdatum*

Das Datum, an dem der Useraccount ungültig wird.

-f *Voreingest. Pufferzeit*

Die Anzahl der Tage nach dem Auslaufen eines Passworts, bis der Account permanent gesperrt bleibt.

-g *Voreingestellte Gruppe*

Der Gruppenname oder die GruppenID der Logingruppe eines neuen Users.

-s *Voreingestellte Startshell*

Der Name der Shell eines neu anzulegenden Users.

Wenn keine Optionen außer **-D** angegeben werden, dann zeigt **useradd** die aktuellen Voreinstellungen. If no options are specified, **useradd** displays the current default

Dateien

/etc/passwd - Useraccount Informationen
/etc/shadow - Sichere Useraccount Informationen
/etc/group - Gruppen Informationen
/etc/default/useradd - Voreingestellte Werte
/etc/skel - Template-Verzeichnis für neue User

Siehe auch

chfn(1) , **chsh**(1) , **crypt**(3) , **groupadd**(8) , **groupdel**(8) , **groupmod**(8) , **passwd**(1) , **userdel**(8) , **usermod**(8)

Autor

Julianne Frances Haugh (jfh@bga.com)

Name

userdel - Löscht einen Useraccount und zugehörige Dateien

Syntax

userdel [**-r**] *Username*

Beschreibung

Das **userdel** Kommando modifiziert die notwendigen Systemdateien und löscht dort alle Einträge, die sich auf den angegebenen Usernamen beziehen. Der angegebene User muß existieren.

-r

Das Homeverzeichnis des Users und alle darinliegenden Dateien werden gelöscht. Andere Dateien des Users, die in anderen Verzeichnissen des Systems liegen, müssen von Hand gesucht und gelöscht werden.

Dateien

/etc/passwd - User Account Informationen
/etc/shadow - Sichere User Account Informationen
/etc/group - Gruppen Informationen

Einschränkungen

userdel kann keine Usereinträge von Usern löschen, die gerade eingeloggt sind.

Siehe auch

chfn(1) , **chsh**(1) , **groupadd**(8) , **groupdel**(8) , **groupmod**(8) , **passwd**(1) , **useradd**(8) , **usermod**(8)

Autor

Julianne Frances Haugh (jfh@bga.com)

Name

usermod - Modifiziert einen Useraccount

Syntax

usermod

[**-c** *Kommentar*] [**-d** *Homeverzeichnis* [**-m**]]
[**-e** *Auslaufdatum*] [**-f** *Pufferzeit*]
[**-g** *Logingruppe*] [**-G** *Gruppe*[,...]]
[**-l** *Login Name*] [**-p** *Passwort*]
[**-s** *Startshell*] [**-u** *UID* [**-o**]] *Username*

Beschreibung

Das **usermod** Kommando verändert die notwendigen Systemdateien dahingehend, daß die auf der Kommandozeile übergebenen Werte eingetragen werden. Gültige Optionen sind:

-c *Kommentar*

Der neue Wert des Kommentarfelds in /etc/passwd für den angegebenen User. Dieses Feld wird normalerweise mit dem Befehl **chfn**(1) modifiziert.

-d *Homeverzeichnis*

Das neue Homeverzeichnis eines Users. Wird zusätzlich die Option **-m** angegeben, so wird der Inhalt des alten Homeverzeichnisses ins neue Verzeichnis verschoben, das angelegt wird, wenn es noch nicht existiert.

-e *Auslaufdatum*

Das Datum, an dem der Useraccount ungültig werden soll. Dieses Datum wird im Format *JJJJ-MM-TT*

angegeben.

-f *Pufferzeit*

Die Anzahl von Tagen, nach dem Auslaufen des Accounts, bis der Account permanent gesperrt wird. Ein Wert von 0 bedeutet keine Pufferzeit, ein Wert von -1 schaltet dieses Feature ab. Der voreingestellte Wert ist -1

-g *Login Gruppe*

Der Gruppenname oder die GruppenID der Login-Gruppe des neuen Users. Diese Gruppe muß existieren.

-G *group,[...]*

Eine Liste weiterer Gruppen, denen der User auch angehören soll. Jede angegebene Gruppe wird von der nächsten durch ein Komma getrennt. Auch diese Gruppen müssen bereits existieren. Wenn der User im Augenblick Mitglied einer Gruppe ist, die hier nicht genannt ist, dann wird ihm diese Gruppenmitgliedschaft gestrichen.

-l *Login Name*

Der neue Name des Users. Durch diese Angabe wird nur der Name gewechselt, keine weiteren Änderungen (wie etwa der Name des Homeverzeichnisses) werden dadurch automatisch vorgenommen.

-p *Passwort* Das verschlüsselte Passwort, wie es von crypt erstellt wird.

-s *Startshell*

Der Name der Loginshell des Users.

-u *UserID*

Die numerische UserID des neuen Users. Dieser Wert muß eindeutig sein, wenn nicht die Option **-o** angegeben wurde. Der Wert darf nicht negativ sein. Voreingestellt ist der Wert, der größer als 99 und größer als jeder andere bisher bestehende UID-Wert ist. Werte von 0 bis 99 sind typischerweise für Systemaccounts reserviert. Alle Dateien und Verzeichnisse innerhalb des Homeverzeichnisses des Users bekommen automatisch auch diese UID als Eigentümer gewechselt. Dateien des Users, die außerhalb seines Homeverzeichnisses liegen müssen von Hand geändert werden.

Einschränkungen

usermod kann nicht die Eigenschaften eines eingeloggten Users verändern. Auch die Eigentümerschaft der Crontabs müssen von Hand gewechselt werden, wenn die UserID verändert wurde.

Dateien

/etc/passwd - Useraccount Informationen
/etc/shadow - Sichere Useraccount Informationen
/etc/group - Gruppen Informationen

Siehe auch

chfn(1) , **chsh**(1) , **crypt**(3) , **groupadd**(8) , **groupdel**(8) , **groupmod**(8) , **passwd**(1) , **useradd**(8) , **userdel**(8)

Autor

Julianne Frances Haugh (jfh@bga.com)

Bezeichnung

wc - zählt die Anzahl von Zeichen, Worten oder Zeilen

Syntax

wc [**-clw**] [**--bytes**] [**--chars**] [**--lines**] [**--words**] [*Datei*...]

Beschreibung

wc zählt in jeder *Datei* oder in der Standardeingabe die Zeilen, die Worte und die Zeichen. Für jede Datei wird eine Zeile mit den Zahlen gefolgt vom Dateinamen ausgegeben. Wird mehr als eine Datei angegeben, wird zusätzlich die Summe der einzelnen Werte in der letzten Zeile ausgegeben.

Ohne weitere Optionen gibt wc alle drei Werte aus

Optionen

-l

gibt nur die Anzahl der Zeilen aus

-w

gibt nur die Anzahl der Worte aus

-c

gibt nur die Anzahl der Zeichen aus

Name

whatis - durchsucht die Indexdatenbank nach Kurzbeschreibungen

Syntax

whatis [**-dhV**] [**-r**|**-w**] [**-m** *System*[,...]] [**-M** *Pfad*] *Schlüsselwort* ...

Beschreibung

Innerhalb jeder Manualseite ist eine Kurzbeschreibung vorhanden. **whatis** sucht *Schlüsselwort* in den Kurzbeschreibungen der *Indexdatenbank.* Falls es eine solche nicht im Manualpfad gibt, durchsucht es die *whatis*- Datenbank nach *Schlüsselwort.* Diese Version von **whatis** kann Schlüsselwörter suchen, die Wildcards oder reguläre Ausdrücke enthalten.

mandb erzeugt die *Indexdatenbank* für jeden Manualpfad. Um eine Datei im alten Stil als *whatis*- Datenbank aus der *Indexdatenbank* zu erzeugen, ist folgender Befehl einzugeben

whatis -M *Pfad* **-w '*' | sort >** *manpath/whatis*

wobei *Pfad* eine Manualhierarchie wie */usr/man* ist.

Optionen

-d, --debug

Ausgeben von Fehlerinformationen.

-r, --regex

Interpretiert jedes Schlüsselwort als regulären Ausdruck. Wenn ein Schlüsselwort irgendeinem Teil eines

Seitennamens entspricht, wird eine Übereinstimmung gefunden.

-w, --wildcard

Das Schlüsselwort kann Wildcards wie in der Shell beinhalten. Um eine Übereinstimmung zu finden, muß ein expandiertes Schlüsselwort mit einem kompletten Seitennamen übereinstimmen.

-m *System* [,...]**, --systems=***System*[,...]

Wenn auch Manualseiten von einem anderen Betriebssystem installiert sind, so kann auf sie mit dieser Option zugegriffen werden. Um beispielsweise die Manualseiten von NewOS zu durchsuchen, muß **-m NewOS** angegeben werden.

Das angegebene *System* kann eine durch Kommata abgetrennte Aufzählung von Betriebssystemnamen sein. Die normalen Seiten werden durch den Betriebssystemnamen **man** angesprochen. Diese Option überschreibt die Umgebungsvariable $**SYSTEM.**

-M *Pfad***, --manpath=***Pfad*

Gibt die Manualpfade an. Wenn mehrere Pfade angegeben werden, müssen diese durch Doppelpunkte getrennt sein. Als Default benutzt **whatis manpath,** um den Suchpfad zu bestimmen. Diese Option überschreibt die Umgebungsvariable $**MANPATH.**

-h, --help

Zeigt einen Hilfstext an.

-V, --version

Zeigt Programmversion und Autor an.

Dateien

/var/catman/.../index.(bt|db|dir|pag)

Die **FSSTND** complianten globalen *Indexdatenbanken.*

/usr/man/.../whatis

Traditionelle *whatis* Datenbank.

Siehe Auch

apropos(1) , **man**(1) , **mandb**(8) .

Fehler

Wenn die **-r** oder **-w** Optionen benutzt werden und Shell-Metazeichen in einem *Schlüsselwort* vorkommen, kann es notwendig sein, *Schlüsselwort* zu quoten, damit die Shell nicht versucht, diese Zeichen zu interpretieren.

Die **-r** und **-w** Optionen können verwirrende Ausgaben erzeugen und verlangsamen die Suche.

Autor

Wilf. (G.Wilford@ee.surrey.ac.uk), die deutsche Übersetzung ist von Anke Steuernagel (a_steuer@informatik.uni-kl.de) und Nils Magnus (magnus@informatik.uni-kl.de).

Bezeichnung

xargs - bildet Kommandozeilen aus der Standard-Eingabe und führt sie aus

Syntax

xargs [-0prtx] [-e[eof-str]] [-i[replace-str]] [-l[max-lines]] [-n max-args] [-s max-chars] [-P max-procs] [--null] [--eof[=eof-str]] [--replace[=replace-str]] [--max-lines[=max-lines]] [--interactive] [--max-chars=max-chars] [--verbose] [--exit] [--max-procs=max-procs] [--max-args=max-args] [--no-run-if-empty] [--version] [--help] [command [initial-arguments]]

Beschreibung

Diese Handbuchseite dokumentiert die GNU Version von **xargs**. **xargs** ließt Argumente aus der Standard-Eingabe, die durch Leerzeichen oder Zeilentrenner voneinander getrennt sind (das kann durch Ausklammerung durch einfache oder doppelte Anführungszeichen oder einen führenden Backslash verhindert werden) und führt dann das angegebene *command* (voreingestellt ist /bin/echo) ein oder mehrmals aus. Jedesmal mit jedem *initial-arguments*, gefolgt von den Argumenten aus der Standard-Eingabe. Leerzeilen in der Standard-Eingabe werden ignoriert.

xargs liefert die folgenden Rückgabewerte:
0 wenn erfolgreich
123 wenn ein Aufruf des Kommandos mit einem Rückgabewert 1-125 endet
124 wenn das Kommando mit dem Rückgabewert 255 endet
125 wenn das Kommando durch ein Signal unterbrochen wurde
126 wenn das Kommando nicht ausführbar ist
127 wenn das Kommando nicht gefunden wurde
1 wenn ein anderer Fehler auftrat

Optionen

--null, -0

Eingabedateinamen werden durch ein Nullzeichen statt durch ein Leerzeichen beendet und Anführungszeichen und Backslash haben keine Sonderbedeutung (jedes Zeichen wird gewertet wie es ist). Auch das Dateiendezeichen wird als normales Zeichen gewertet und nicht als solches interpretiert. Nützlich, wenn Argumente Leerzeichen, Anführungszeichen oder Backslashs enthalten können. Das GNU-find Programm produziert mit der Option -print0 Ausgaben, die für diesen Modus brauchbar sind.

--eof[=eof-str], -e[eof-str]

Setzt die Dateiendemarke auf *eof-str*. Wenn diese Marke in einer eigenen Zeile des Eingabedatenstroms existiert, wird der Rest der Eingabe ignoriert. Fehlt die Angabe des *eof-str*, so gibt es kein Dateiende. Wenn diese Option nicht verwendet wird wird die Dateiendemarke auf "_" gesetzt.

--help

Zeigt eine Zusammenfassung der Optionen von **xargs** und beendet dann das Programm.

--replace[=replace-str], -i[replace-str]

Ersetzt Vorkommen von *replace-str* in den initialen Argumenten durch Namen, die aus der Standard Eingabe gelesen wurden. Nicht ausgeklammerte Leerzeichen beenden Argumente nicht. Wenn *replace-str* weggelassen wird, wird "{}" angenommen (wie bei "find -exec"). Schließt *-x* und *-l 1* ein.

--max-lines[=max-lines], -l[max-lines]

Benutze maximal *max-lines* nichtleere Eingabezeilen pro Kommandozeile. *max-lines* wird voreingestellt auf 1 wenn es weggelassen wird. Leerzeichen am Ende einer Eingabezeile führen dazu, daß diese Zeile logisch auf der nächsten Zeile fortgesetzt wird. Schließt *-x* ein.

--max-args=max-args, -n max-args

Benutze maximal *max-args* Argumente pro Kommandozeile. Weniger als *max-args* Argumente werden nur benutzt, wenn die Größe (siehe die -s Option) überschritten wurde, wenn nicht die -x Option gesetzt wurde in der xargs abbrechen würde.

--interactive, -p

Frage den User ob eine Kommandozeile ausgeführt werden soll, und lese eine Eingabezeile vom Terminal. Das Kommando wird nur ausgeführt, wenn diese Eingabe mit 'y' oder 'Y' beginnt. Schließt *-t* ein.

--no-run-if-empty, -r

Wenn die Standard-Eingabe keine Nicht-Leerzeichen enthält, führe das Kommando nicht aus. Normalerweise würde das Kommando mindestens einmal ausgeführt.

--max-chars=max-chars, -s max-chars

Benutze maximal *max-chars* Zeichen pro Kommandozeile, einschließlich Kommando und Initialargumenten. Voreingestellt ist so groß wie möglich, bis zu 80000 Zeichen.

--verbose, -t

Gibt das Kommando auf die Standard-Fehlerausgabe aus, bevor es ausgeführt wird.

--version

Gibt die Versionsnummer von **xargs** aus und beendet dann das Programm.

--exit, -x

Beendet das Programm, wenn die Größe (siehe die -s Option) überschritten wurde.

--max-procs=max-procs, -P max-procs

Führe bis zu *max-procs* Prozesse gleichzeitig aus; voreingestellt ist 1. Wenn *max-procs* 0 ist, so führt **xargs** soviele Prozesse wie möglich gleichzeitig aus. Sollte zusammen mit der *-n* Option benutzt werden, weil sonst evt. nur ein Prozess gestartet wird.

Siehe Auch

find(1L) , **locate**(1L) , **locatedb**(5L) , **updatedb**(1)

1.105.1

Verwalten/Abfragen von Kernel und Kernelmodulen zur Laufzeit

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Kernel und ladbare Kernelmodule zu verwalten und/oder abzufragen. Dieses Lernziel beinhaltet die Verwendung von Kommandozeilen-Utilites, um Informationen über den gerade laufenden Kernel und die Kernelmodule zu erhalten. Ebenfalls enthalten ist das manuelle Laden und Entladen von Modulen nach Bedarf. Dies beinhaltet auch die Bestimmung, wann Module entladen werden können und welche Parameter ein Modul akzeptiert. Prüfungskandidaten sollten in der Lage sein, das System so zu konfigurieren, daß Module auch durch andere als ihre Dateinamen geladen werden können.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/lib/modules/`*`kernel-version`*`/modules.dep`
- `/etc/modules.conf & /etc/conf.modules`
- **depmod**
- **insmod**
- **lsmod**
- **rmmod**
- **modinfo**
- **modprobe**
- **uname**

Kernelmodule

Der Linux-Kernel war anfangs (in der Version 1.x.x) ein rein monolithisches Gebilde. Das hatte zur Folge, daß jede Hardware-Unterstützung fest in den Kernel eingebunden werden musste. So entstanden entweder sogenannte Allround-Kernel, die riesengroß waren und Unterstützung für die abenteuerlichste Hardware zur Verügung stellten, oder es musste für jede neue Hardware, die in einen Rechner eingebaut wurde, ein neuer Kernel erstellt werden. Beides war keine praxistaugliche Lösung.

Aus diesem Grund wurde der Kernel modularisiert. Das heißt, daß bestimmte Teile des Kernels nicht mehr beim Compilieren fest eingebunden werden müssen, sondern später, während der Laufzeit, ge- und auf Wunsch auch wieder entladen werden können. Diese Fähigkeit ermöglicht es, daß Gerätetreiber eben nicht immer fest eingebunden werden müssen, sondern je nach Bedarf nachgeladen werden können. Distributoren können so schlanke Kernel ausliefern, die trotzdem alle Fähigkeiten von Linux unterstützen, weil die jeweiligen Fähigkeiten bei Bedarf geladen werden können. Die Teile des compilierten Kernels, die nicht fest eingebunden sind, sondern ge- und entladen werden können, werden *Kernelmodule* genannt.

Die Kernelmodule des Linux-Kernels werden im Dateisystem unter `/lib/modules/`*`Kernelversion`* abgespeichert, wobei *Kernelversion* für die Versionsnummer des Kernels steht, für den die Module passen. Die Module bestehen aus Objektdateien (Endung `.o`), die in diesem Verzeichnis in diverse Unterverzeichnisse sortiert sind.

Kernelmodule sind Teile des Kernels, zwar ausgelagert, aber doch Teile des Ganzen. Wenn solche Module ein- und ausgehängt werden, so spielen bestimmte Abhängigkeiten (dependencies) eine Rolle. Beispielsweise kann der Kernel nichts mit einem Modul anfangen, das ihm die Fähigkeit verleihen soll, mit SCSI-CDROMs umzugehen, wenn er gar nicht weiß, was SCSI ist. Das Modul für die SCSI-CDROMs ist also abhängig vom generellen Modul für SCSI. Diese Abhängigkeiten müssen den Programmen bekannt sein, die die Module laden und entladen. Aus diesem Grund existiert das Programm **depmod**, das die Abhängigkeiten aller Module untersucht und die Ergebnisse in die Datei `/lib/modules/`*`Kernelversion`*`/modules.dep` schreibt. In dieser Datei stehen dann - in klar lesbarer Form - die entsprechenden Abhängigkeiten jedes einzelnen Moduls. Jedesmal, wenn neue Module ins Modulverzeichnis kopiert werden, muß anschließend dieses Programm aufgerufen werden.

Programme zum Umgang mit den Modulen

Es gibt jetzt eine Reihe von Befehlen, die es ermöglichen mit Kernelmodulen umzugehen. Diese Befehle werden jetzt der Reihe nach kurz beschrieben.

Auflistung der geladenen Module

Um herauszufinden, welche Module bereits geladen sind, existiert der Befehl **lsmod**. Er zeigt alle geladenen Module im Format Name, Größe, Wie-oft-gebraucht und Liste der Module, die das Modul benötigen. Dieser Befehl dient also hauptsächlich dazu, zu überprüfen, ob ein bestimmtes Modul geladen ist oder nicht. **lsmod** wird ohne Parameter aufgerufen.

Laden von einzelnen Modulen

Um ein einzelnes Modul zu laden, kann der Befehl **insmod** verwendet werden. Als Parameter erwartet **insmod** den Namen des zu ladenden Moduls. Der Name wird **ohne** die Endung `.o` angegeben. Auch Pfade werden keine benötigt. Das Programm weiß ja selbst, in welchem Verzeichnis die ladbaren Module des aktuellen Kernels zu finden sind. Um also beispielsweise das Modul `/lib/modules/`*`Kernelversion`*`/kernel/drivers/usb/printer.o` zu laden genügt die Angabe

```
insmod printer
```

Anhand der aktuellen Kernelversionsnummer ermittelt **insmod** das notwendige Basisverzeichnis für die Module und kann dort über die Datei `modules.dep` den genauen Pfad zum gewünschten Modul ermitteln.

Wenn ein Modul selbst noch Parameter erwartet, z.B. die I/O-Adresse einer ISA-Netzwerkkarte, so werden sie nach dem Modulnamen angegeben. Diese Parameter haben immer die Form

Symbolname=Wert

Für die ISA-Netzwerkkarte könnte das z.B. der Befehl

```
insmod ne io=0x300 irq=5
```

sein. Damit wird das Modul `ne` geladen (ein Modul für eine NE2000 Netzwerkkarte) und bekommt als Parameter die Werte für IO-Adresse und IRQ übergeben.

Das Programm **insmod** läd nur das angegebene Modul. Wenn dieses Modul andere Module benötigt, so werden sie **nicht** mitgeladen.

Laden eines Moduls mit allen nötigen Zusatzmodulen

Damit man auch Module laden kann, und automatisch alle notwendigen Zusatzmodule mitgeladen werden, existiert der Befehl **modprobe**. Er hat heute den Befehl **insmod** völlig verdrängt, denn er ist wesentlich intelligenter. Die Anwendung entspricht der von **insmod**. Als Parameter wird das zu ladenden Modul, wiederum ohne Pfad und ohne der Endung `.o` angegeben. Auch eventuell notwendige Parameter werden wie bei **insmod** angegeben.

Im Gegensatz zu **insmod** überprüft **modprobe** die Abhängigkeiten der Module untereinander (über die Informationen aus `modules.dep` und läd alle Module, die das angegebene Modul benötigt, um richtig zu funktionieren. Ein Beispiel:

Wenn wir das Modul für ein am Parallelport angeschlossenes Zip-Laufwerk (ppa) laden wollen, so benötigt dieses Modul, das generellen Zugriff auf Parallelports ermöglicht (parport). es reicht zu schreiben

```
modprobe ppa
```

und **modprobe** erkennt, daß eine nicht erfüllte Abhängigkeit vorliegt, läd zuerst das Modul `parport` und erst danach das Modul `ppa`.

Anstatt Parameter für bestimmte Module direkt anzugeben, unterstützt **modprobe** (nicht **insmod**) eine Datei, die die notwendigen Modulparameter enthält. Diese Datei heißt entweder `/etc/conf.modules` (alt) oder `modules.conf` (aktuell) und wird weiter unten noch genauer beschrieben.

Um genau zu sein, benutzt **modprobe** das Programm **insmod** um die notwendigen Module zu laden. **modprobe** ist also eine Art High-Level Programm zum Laden der Module oder ein Frontend für **insmod**.

Entfernen von geladenen Modulen

Wenn ein Modul nicht mehr benötigt wird, so kann es mit dem Befehl **rmmod** wieder aus dem Kernel entfernt werden. **rmmod** kann keine Module entfernen, die gerade in Gebrauch sind oder die von anderen geladenen Modulen benötigt werden. Allerdings ist es möglich, mit dem Kommandozeilenparameter `-r` dafür zu sorgen, daß **rmmod** so funktioniert, wie ein umgekehrtes **modprobe**, also nicht nur das angegebene Modul entfernt, sondern alle, die nur deshalb geladen sind, damit das zu entfernende Modul funktioniert.

Auch dieses Programm erwartet den Namen des zu entfernenden Moduls ohne Endung und Pfad.

Auch das Programm **modprobe** kann verwendet werden, um Module wieder zu entfernen. Mit dem Parameter `-r` werden Module entfernt, die auf der Kommandozeile angegeben wurden.

Modulparameter erkennen und einstellen

Module sind häufig Gerätetreiber für bestimmte Hardware. In vielen Fällen ist es notwendig, für die Ansteuerung der Hardware bestimmte Parameter wie IO-Port, IRQ usw. anzugeben, wenn ein Modul geladen werden soll. Wie oben gesehen, werden Parameter für Module immer in der Form

Symbolname=Wert

angegeben. Dazu muß aber klar sein, welche Symbolnamen das jeweilige Modul kennt, also welche Parameter es versteht. Um das herauszufinden gibt es das Programm **modinfo**. Dieses Programm gibt Informationen über ein Modul aus, unter anderem welche Parameter angegeben werden können. Auch **modinfo** arbeitet mit den Modulnamen ohne Endung und Pfad. Über Kommandozeilenparameter kann angegeben werden, welche Informationen ausgegeben werden sollen, ohne solche Schalter werden die Informationen über

- Dateiname und Pfad der Moduldatei (-n)
- Beschreibung (-d)
- Author (-a)
- Lizenz (-l)

- Parameter (-p)

ausgegeben. Um z.B herauszufinden, welche Parameter das Modul 3c509 (ein Gerätetreiber für eine 3Com Netzwerkkarte) benötigt, können wir schreiben

```
modinfo -p 3c509
```

Die entsprechende Ausgabe des Programms wäre:

```
debug int, description "EtherLink III debug level (0-6)"
irq int array (min = 1, max = 8), description "EtherLink III
   IRQ number(s) (assigned)"
xcvr int array (min = 1, max = 8), description "EtherLink III
   tranceiver(s) (0=internal, 1=external)"
max_interrupt_work int, description "EtherLink III maximum events
   handled per interrupt"
nopnp int, description "EtherLink III disable ISA PnP support (0-1)"
```

Daraus können wir entnehmen, daß dieses Modul die Parameter `debug, irq, xcvr, max_interrupt_work` und `nopnp` versteht. Alle diese Parameter erwarten ganzzahlige (int) Werte. Wenn hier die Angabe `array` steht, so ist gemeint, daß wenn mehrere gleichartige Karten existieren, die alle dieses Modul benötigen, dann an Stelle der normalen Angabe einer Zahl, eine durch Komma getrennte Liste von Zahlen verwendet wird. Bei drei solcher Netzwerkkarten hieße der Parameter, der den IRQ einstellt also `irq=3,5,7`. Die erste Karte benutzt IRQ3, die zweite 5 und die dritte 7.

Die Datei /etc/modules.conf

Um den Kernelmodulen ihre Parameter an einer festen Stelle angeben zu können, wurde **modprobe** so programmiert, daß es die Datei `/etc/modules.conf` (oder `/etc/conf.modules`) abarbeitet und dort entsprechende Parameter für die Module entnimmt.

Die Datei kann auch noch mehr, als nur Parameter für Module zur Verfügung stellen. Insbesondere können damit

- Parameter für Module festgelegt werden,
- Aliasnamen für Module vergeben werden,
- Module angegeben werden, die vor oder nach dem eigentlichen Modul geladen werden sollen,
- Module, die vor oder nach dem Laden des eigentlichen Moduls entfernt werden sollen

Die Datei hat ein einfaches Format. Jede Zeile, die nicht mit einem Kommentarzeichen (#) beginnt oder leer ist, wird als Anweisung interpretiert. Um einem Modul bestimmte Parameter zu setzen, wird der Befehl **options** benutzt. Er hat die folgende Syntax:

options *Modulname Optionen*

Ein Eintrag für unsere NE2000 Netzwerkkarte, könnte also so aussehen:

```
options ne io=0x320 irq=7
```

Jedesmal, wenn jetzt mit **modprobe** das Modul `ne` geladen wird, werden automatisch diese Optionen mitgeladen.

Um ein Modul auch mit anderem Namen anzusprechen, können wir Aliasnamen vergeben. Das bedeutet, daß mit **modprobe** auch Module geladen werden, die es gar nicht gibt, sondern die nur Aliasnamen sind. Die Syntax ist

alias *Aliasname Modulname*

Diese Methode wird gerne benutzt um bestimmte Hardware an einen Begriff zu binden, der mehr aussagt und allgemeiner handhabbar ist als der Modulname. So kann z.B. der Aliasname `eth0` an die tatsächliche Netzwerkkarte gebunden werden. Der Eintrag

```
alias eth0 ne
alias eth1 off

options ne io=0x320 irq=7
```

würde es also ermöglichen, daß wir (oder ein init-script) den Befehl

```
modprobe eth0
```

eingeben und so automatisch das Modul `ne` mit den angegebenen Parametern geladen wird. Versucht aber jemand mit **modprobe** das Modul `eth1` zu laden, dann weigert sich **modprobe** irgendein Modul zu laden, da der Wert des Aliases **off** ist.

Mit pre-install können Befehle angegeben werden, die vor dem Laden des Moduls ausgeführt werden sollen. Diese Befehle können (müssen aber nicht) andere Module laden. Die Syntax lautet:

pre-install *Modul Kommando*

Bevor das Modul geladen wird, wird erst das Kommando ausgeführt. Analog dazu gibt es die Befehle

post-install *Modul Kommando*
pre-remove *Modul Kommando*
post-remove *Modul Kommando*

die entsprechend nach dem Laden, vor dem Entfernen und nach dem Entfernen bestimmter Module bestimmte Kommandos ausführen. Alle vier Kommandos können auch auf Aliasnamen angewandt werden.

Der Befehl uname

Der Befehl uname gibt Informationen über den Kernel und das System aus. Um z.B. die aktuelle Kernelversion zu erfragen, genügt ein Aufruf von

```
uname -r
```

Das ist z.B. die Methode, mit der erfragt werden kann, in welchem Verzeichnis die Module des Kernels liegen. Es ist tatsächlich möglich zu schreiben

```
cd /lib/modules/`uname -r`/kernel
```

Durch die Kommandosubstitution wird das Konstrukt `` `uname -r` `` durch die Ausgabe des Befehls ersetzt und die entspricht genau dem, was als Verzeichnisnamen vorliegt.

Alle Informationen können mit `uname -a` ausgegeben werden, die restlichen Parameter entnehmen Sie der Handbuchseite.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.105.2

Konfiguration, Erstellung und Installation eines angepaßten Kernels und seiner Module

---

Beschreibung: Prüfungskandidaten sollten in der Lage sein, einen Kernel und ladbare Kernelmodule aus Quellcode anzupassen, zu erzeugen und zu installieren. Dieses Lernziel beinhaltet das Anpassen der aktuellen Kernelkonfiguration, das Erzeugen eines neuen Kernels und das Erzeugen von Kernelmodulen nach Bedarf. Ebenfalls enthalten ist die Installation des neuen Kernels sowie jedweder Module und das Sicherstellen, daß der Bootmanager den neuen Kernel und die dazugehörigen Dateien findet (generell in `/boot`, siehe Lernziel 1.102.2 für mehr Details über Bootmanager und deren Konfiguration).

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/usr/src/linux/*`
- `/usr/src/linux/.config`
- `/lib/modules/`*`kernel-version`*`/*`
- `/boot/*`
- **make**
- make targets:**config, menuconfig, xconfig, oldconfig, modules, install, modules_install, depmod**

Eine der wesentlichen Unterschiede von Linux gegenüber praktisch allen anderen Betriebssystemen ist, daß der Quellcode verfügbar ist und es so möglich ist, seinen eigenen Kernel zu compilieren. Einen Kernel, der also genau an die Bedürfnisse des eigenen Rechners angepasst ist und keinen unnötigen Ballast enthält. Seit der Einführung des modularen Kernels gibt es zwar keine wirkliche Notwendigkeit mehr, den Kernel selbst zu erzeugen, es ist jedoch immer noch in manchen Fällen besser, einen eigenen Kernel zu benutzen, anstelle eines Standard-Kernels. Ob es sich nun um besonders sicherheitsrelevante Systeme handelt, oder bestimmte Fähigkeiten eines Standard-Kernels nicht erwünscht sind, jedesmal ist es die Lösung, einen entsprechenden eigenen Kernel zu erzeugen.

Einen eigenen Kernel compilieren

Der Vorgang der Compilierung eines Kernels ist kein Kunststück, es werden keine C-Kenntnisse benötigt. Ähnlich wie beim Übersetzen von Paketen, die im Quellcode vorliegen (siehe Abschnitt 1.102.3 der LPI101 Vorbereitung) wird wieder das Programm **make** benötigt, das anhand eines Makefiles den Kernel erstellt. Im Gegensatz zu einem "normalen" Programm ist aber etwas mehr Konfigurationsarbeit nötig, bevor der Übersetzungsvorgang gestartet werden kann.

Ein Kernel besteht aus vielen hundert Quellcode-Dateien (`*.c`), die alle für bestimmte Fähigkeiten des Kernels stehen. Die wesentliche Arbeit bei der Neuerstellung eines Kernels ist es, festzulegen, welche dieser Fähigkeiten erwünscht sind, welche als Modul erstellt werden sollen und welche weggelassen werden sollen. Diese Aufgabe alleine wäre nahezu unlösbar, wenn es dafür keine speziellen Hilfsprogramme gäbe. Aber diese Hilfsprogramme existieren. Sie sind Teil des Kernel-Quellcodes selbst und werden auch über das Makefile (also mit dem Programm **make**) aufgerufen.

Konfiguration des Kernels

Die Quellen des Kernels werden in ein Verzeichnis unter `/usr/src` entpackt. Die Kernelversion, die compiliert werden soll, muß unter `/usr/src/linux` zu finden sein. Am einfachsten wird das über einen symbolischen Link erreicht. `/usr/src/linux` ist also ein Symlink auf das Verzeichnis, in dem wirklich die Kernelquellen liegen. Innerhalb dieses Verzeichnisses wird alle Konfigurations- und Übersetzungsarbeit geleistet.

Um jetzt den Konfigurationsvorgang zu starten, gibt es drei unterschiedliche Möglichkeiten:

- **make config**
  Startet ein sehr einfaches textorientiertes Programm, das der Reihe nach alle Einstellungen des Kernels abarbeitet und den Anwender auffordert zu entscheiden, ob das jeweilige Feature integriert, als Modul erzeugt oder weggelassen werden soll. Diese Methode ist sehr umständlich, weil immer alle Fragen von vorne bis hinten bearbeitet werden müssen. (Screenshot)
- **make menuconfig**
  Startet ein dialog-basiertes textorientiertes Programm, das auch die Beantwortung aller Fragen ermöglicht. Im Gegensatz zu **make config** erlaubt es aber das Navigieren durch alle Fragen, Wiederholung von bereits beantworteten Fragen oder nur die Veränderung einzelner Punkte. (Screenshot)
- **make xconfig**
  Ein Tcl/tk basiertes graphisches Programm, das es wie **make menuconfig** erlaubt, durch die verschiedenen Fragen zu navigieren, nur eben als graphische Anwendung unter X11. (Screenshot)

Alle drei Programme tun im Prinzip das selbe. Sie ermöglichen es dem Anwender, bestimmte Fragen über die Kernelkonfiguration zu beantworten und speichern das Ergebnis in die Datei `/usr/src/linux/.config`

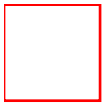
Diese Datei enthält die Ergebnisse der Konfiguration, also die Antworten auf alle Fragen, die während der Konfiguration beantwortet wurden. Die Form entspricht einer Shellvariablendefinition, also beispielsweise

```
#
# Automatically generated make config: don't edit
#
CONFIG_X86=y
CONFIG_ISA=y
# CONFIG_SBUS is not set
CONFIG_UID16=y

#
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y

#
# Loadable module support
#
CONFIG_MODULES=y
# CONFIG_MODVERSIONS is not set
CONFIG_KMOD=y

...
```

Die Datei `/usr/src/linux/.config` ist also die Datei, die die gesamte Kernelkonfiguration enthält. Es ist also möglich, verschiedene Konfigurationen herzustellen, indem einfach diese Datei umbenannt wird, um so für verschiedene Rechner jeweils einen Kernel anzupassen.

Die Beantwortung der Fragen während der Kernelkonfiguration ist ein langwieriger Prozess, bei der Version 2.4.18 sind es beispielsweise 656 Stück. Diese Fragen können hier nicht einzeln durchgesprochen werden. Viele Distributionen legen aber bereits eine `.config` Datei ihres speziellen Kernels bei, die dann benutzt werden kann, um einzelne Veränderungen zu machen.

Die Datei `.config` ist also auch die Datei, aus der die Voreinstellungen der Fragen entnommen werden. Schematisch können wir das also folgendermaßen darstellen:



Die drei verschiedenen Konfigurationsprogramme bekommen ihre Voreinstellung aus `.config` und schreiben die veränderten Ergebnisse wieder nach `.config`.

Normalerweise bringt der Wechsel von einer Kernelversion auf eine andere natürlich auch eine neue `.config` Datei mit. In der Regel gibt es ja in neueren Versionen auch neuere Einstellmöglichkeiten und damit neue Fragen und Antworten. Um eine bestehende `.config` Datei auch in einem neuen Kernel zu benutzen, kann der Befehl

- **make oldconfig**
  Benutzt eine alte Konfigurationsdatei und stellt nur die Fragen, die dort noch nicht beantwortet wurden weil es sie noch nicht gab.

benutzt werden. Diese Form bedingt natürlich, daß zunächst die alte `.config` Datei in das neue Kernelverzeichnis kopiert wurde.

Übersetzung des Kernels

Nachdem jetzt eingestellt ist, welche Teile des Kernels fest eingebaut werden sollen, welche als Modul erstellt werden sollen und welche einfach weggelassen werden sollen, muß der Kernel anhand dieser festgestellten Regeln compiliert werden. Dieser Vorgang läuft in drei Schritten ab, die alle wieder über das Programm **make** abgewickelt werden.

Der erste Schritt ist nur bei der ersten Kernelcompilation notwendig, bei späteren Übersetzungen kann er weggelassen werden. Es ist allerdings kein Schaden, wenn er auch dann ausgeführt wird. Es geht um die Erstellung der Abhängigkeitsinformationen (dependancies). Der Befehl lautet

- **make dep**
  Erstellt die Abhängigkeitsinformationen, welche `.c` Datei welche andere benötigt.

Der zweite Schritt ist die Übersetzung des Kernels selbst. Je nach gewünschtem Kernel werden hier verschiedene Befehle benutzt.

- **make zImage**
  Die alte Anweisung, die den Kernel komprimiert in einer Datei ablegt. Bei Intel-basierten Systemen liegt der fertige Kernel dann unter `/usr/src/linux/arch/i386/boot/zImage`.
- **make zdisk**
  Compiliert den Kernel und kopiert ihn auf eine Diskette, die so zur Bootdisk wird. Auch das die alte Methode.
- **make bzImage**
  Die moderne Methode, einen großen (b wie big) Kernel herzustellen und komprimiert abzulegen. Bei Intel-basierten Systemen liegt der fertige Kernel dann unter `/usr/src/linux/arch/i386/boot/bzImage`
- **make bzdisk**
  Die neue Methode, eine Bootdisk mit großem (big) Kernel zu erstellen.

Der Vorgang des Compilierens dauert eine Weile, abhängig vom verwendeten Rechner und dessen Ressourcen. Es werden jetzt alle .c Dateien, die durch die Konfigurationsinformation angewählt wurden, in Objektdateien (.o) compiliert die dann später zum Kernel zusammengefasst (linked) werden.

Als dritter Schritt müssen jetzt noch die Module übersetzt werden, also die Teile des Kernels, die nicht fest eingebunden werden sollen, sondern während der Laufzeit ein- und aushängbar sind. Dazu dient der Befehl

- **make modules**
  Übersetzt die ladbaren Module. Die übersetzten Module bleiben in den Verzeichnissen, in denen ihr Quellcode liegt.

Damit ist der eigentliche Compiliervorgang abgeschlossen. Der Kernel und die Module sind jetzt übersetzt, stehen dem System aber noch nicht zur Verfügung, da sie ja noch unterhalb des Verzeichnisses `/usr/src/linux` liegen.

Installieren des neuen Kernels und der Module

Damit die übersetzten Module auch dahin kommen, wo sie später erwartet werden (in das Verzeichnis `/lib/modules/`*`Kernelversion`*), wird der Befehl

- **make modules_install**
  Kopiert die Moduldateien in eine Verzeichnisstruktur unterhalb von `/lib/modules/`*`Kernelversion`*.

ausgeführt. Dieser Befehl startet dann auch gleich automatisch das Programm **depmod** um die notwendigen Modulabhängigkeiten zu analysieren und in der Datei `/lib/modules/`*`Kernelversion`*`/modules.dep` abzulegen.

Um schließlich auch den Kernel selbst zu benutzen, muß die Datei `/usr/src/linux/arch/i386/boot/bzImage`, die ja den neuen Kernel enthält, unter beliebigem Namen (etwa bzImage.neu) in das Verzeichnis `/boot` kopiert werden. Auch die Datei `/usr/src/linux/System.map` sollte dorthin kopiert werden und zwar als `/boot/System.map.`*`Kernelversion`*. Jetzt kann entsprechend der verwendete Bootmanager konfiguriert werden, damit er den neuen Kernel als Bootoption anbietet.

Bei der Installation eines neuen Kernels ist es immer ratsam, zunächst einmal den alten (funktionierenden) Kernel nicht zu überschreiben. So bleibt die Möglichkeit bestehen, den alten Kernel zu laden, falls der neue nicht erwartungsgemäß funktioniert.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.106.1

Booten des Systems

Beschreibung: Prüfungskandidaten sollten in der Lage sein, das System durch den Bootprozeß zu führen. Dieses Lernziel beinhaltet das Übergeben von Kommandos an den Bootlader und die Übergabe von Optionen an den Kernel zum Bootzeitpunkt sowie das Überprüfen von Ereignissen in den Logdateien.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **dmesg**
- `/var/log/messages`
- `/etc/conf.modules` oder `/etc/modules.conf`
- LILO
- GRUB

Der Bootvorgang von Linux kann auf verschiedene Art und Weise gestartet, beeinflusst und nachvollzogen werden. Für die Anforderungen des LPIC 101 Examens genügt das Wissen um die Techniken, die mit dem Linux Loader (lilo) oder dem alternativen Bootmanager **grub** zu tun haben. Das soll auf dieser Seite näher erläutert werden.

Die Funktion von Bootmanagern wurde bereits in der Vorbereitung auf die erste LPI-Prüfung in Abschnitt 1.102.2 ausführlich dargestellt und wird hier als bekannt vorausgesetzt.

Kernelparameter angeben

Jeder Linux-Kernel kann beim Start, also bevor er geladen wird, noch Parameter übergeben bekommen, die sich auf die Art und Weise auswirken, wie er startet bzw. wie er sich zur Laufzeit verhalten soll. Es kann sich dabei um Parameter für bestimmte Geräte handeln (etwa SCSI-Adressen des Bootlaufwerks) oder um Eigenschaften des Kernels selbst wie z.B. die oben als Beispiel angegebene Anweisung bei einem Reboot einen Warmstart statt einem Kaltstart zu machen.

Kernelparameter für Geräte werden nur notwendig, wenn mit Hardware gearbeitet wird, die der Kernel nicht selbstständig erkennt. Ein Standard-PC mit Standard-Komponenten benötigt keine zusätzlichen Kernelparameter.

Es würde den Rahmen dieser Darstellung bei weitem sprengen, alle denkbaren Kernel-Parameter hier zu beschreiben, das ist für die LPI101 Prüfung auch gar nicht notwendig. Wenn die Kernel-Quellen installiert sind, findet sich eine Beschreibung aller unterstützer Parameter in der Datei `/usr/src/linux/Documentation/kernel-parameters.txt`. Wichtig ist die Tatsache, daß es möglich ist, bestimmte Parameter dem Kernel zu übergeben.

Wo werden diese Parameter übergeben, das ist jetzt die Frage. Es gibt mehrere mögliche Antworten, die hier kurz angefügt werden sollen:

Auf dem Bootprompt von LILO oder GRUB

Wenn der Bootprompt beim Start des Systems erscheint, dann erwartet der Bootmanager die Nennung des Systemnamens, das gestartet werden soll (also des in /etc/lilo.conf angegebenen Labels). Diesem Namen können Kernelparameter mitgegeben werden. Das könnte z.B. folgendermaßen aussehen:

```
LILO boot: linux aha152x=0x300,10,7
```

Dieser Kernel-Parameter würde dem Kernel also mitteilen, daß ein Adaptec AHA152X SCSI Hostadapter an der Adresse 0x300 sitzt, den IRQ 10 benutzt und die SCSI-ID 7 belegt.

Analog dazu kann ein solcher Parameter auch auf der Boot-Zeile von **grub** eingegeben werden, statt LILO steht hier dann entsprechend GRUB.

Diese Form der Parameterübergabe ist natürlich lästig, weil man jedesmal beim Booten die erforderlichen Parameter angeben muß. Sie eignet sich daher nur für experimentelles booten, also das Ausprobieren einzelner Parameter, um sicherzustellen, daß sie auch richtig funktionieren.

Andere Bootprompts

Auch die Bootprompts von SYSLINUX (Startdisketten vieler Distributionen) oder als Parameter von LOADLIN (Linux von DOS/Windows aus starten) können Kernelparameter mitgegeben werden. Auch hier gilt das oben gesagte, es handelt sich um die Möglichkeit zu experimentieren und sollte keine Lösung für den Alltag sein.

In der Datei /etc/lilo.conf

Sollen Parameter dauerhaft eingerichtet werden, so ist die Datei /etc/lilo.conf der richtige Ort. Wie oben schon gezeigt, gibt es in dieser Datei die Möglichkeit, sowohl in der globalen ersten Sektion, als auch für jeden Kerneleintrag einzeln mit dem Schlüsselwort `append=` Kernelparameter anzugeben.

Wird diese Angabe wie im obigen Beispiel in der ersten, globalen Sektion der lilo.conf vorgenommen, so gilt sie für alle zu bootenden Linux-Kernel, steht sie stattdessen in der zweiten Sektion, so gilt sie nur für den jeweiligen Kernel, in dessen Kontext sie

angegeben wurde.

In der Datei /boot/grub/grub.conf

Wird statt **lilo** das Programm **grub** als Bootmanager verwendet, so muß die Angabe der Parameter entsprechend in seiner Konfigurationsdatei vorgenommen werden. In diesem Fall werden die Parameter direkt an die Kernelangabe gehängt, also etwa in der Art

```
kernel /bzImage-2.2.14 ro root=/dev/hda3 aha152x=0x300,10,7
```

In der Datei /etc/conf.modules bzw. modules.conf

Nachträglich ladbare Module können (und sollen) in der Datei /etc/conf.modules konfiguriert werden. Streng genommen handelt es sich hier auch um Kernel-Parameter, aber eben um die für ladbare Module. Eine genaue Beschreibung dieser Datei folgt im nächsten Abschnitt.

Mit Ausnahme der Einstellungen in der Datei /etc/conf.modules (oder modules.conf) gilt für Kernelparameter immer die folgende Regel:

Alle für einen Treiber relevanten Parameter müssen unmittelbar hintereinander, durch Kommas getrennt, angegeben werden. Es darf kein Leerzeichen zwischen den einzelnen Parametern eines Treibers sein. Sollen mehrere Kernelparameter angegeben werden, so werden diese mit Leerzeichen voneinander getrennt. Also z.B.

```
aha152x=0x300,10,7 reboot=warm
```

Die Angaben für die Module und ihre Form wird im nächsten Abschnitt dargestellt.

Kernelmodule konfigurieren

Ein Linux-Kernel kann prinzipiell auf zwei verschiedene Arten aufgebaut werden. Entweder alle notwendigen Gerätetreiber sind fest in den Kernel hineinkompiliert, oder einzelne Gerätetreiber sind als ladbare Module kompiliert und werden zur Laufzeit ge- und entladen. Im ersten Fall spricht man von einem **monolithischen Kernel**, im zweiten von einem **modularisierten Kernel**.

Die Vorteile des zweiten Modells liegen auf der Hand. Müssten alle Gerätetreiber immer fest im Kernel eingebaut sein, dann hätten wir entweder einen riesigen (und demzufolge langsamen) Kernel oder wir müssten wegen jeder Kleinigkeit (etwa einer neuen Netzwerkkarte) den Kernel neu kompilieren.

Durch die Modularisierung haben wir die Möglichkeit, die Treiber unabhängig vom Kernel zu kompilieren und dann zu laden, wenn wir sie brauchen. Es ist sogar möglich, die geladenen Treiber während der Laufzeit wieder abzuhängen, so daß unnötiger Speicherbedarf vermieden werden kann. So kann z.B. ein ZIP-Laufwerk an der parallelen Schnittstelle betrieben werden. Immer dann, wenn wir es einstecken laden wir den dafür notwendigen Treiber, sobald wir es abhängen entfernen wir auch den Treiber wieder...

Die Kernel-Module benötigen - genauso, als wären sie fest im Kernel eingebaut - zum Teil Angaben über ihre verwendeten Adressen, IRQs, DMA-Kanäle und ähnliches. Damit diese Angaben nicht jedesmal beim Laden der Module von Hand eingegeben werden müssen (und damit so auch ein automatisches Laden der Module ermöglicht werden kann), müssen wir diese Parameter fest in einer Datei eintragen. Diese Datei heißt /etc/conf.modules oder /etc/modules.conf. Meist ist modules.conf ein Link auf conf.modules oder umgekehrt.

Diese Datei beinhaltet die notwendigen Parameter für die verwendeten Module, allerdings werden diese Parameter hier auf eine andere Art - mit einer anderen Syntax - als bei der direkten Übergabe an den Kernel angegeben.

Die vollständige Dokumentation über diese Datei und deren Format ist in der Handbuchseite modules.conf(5) nachzulesen. Hier werden nur die wichtigsten Elemente beschrieben.

Ein einfaches Beispiel für eine solche Datei (reduziert auf die Beschreibung der Netzwerkkarte) könnte wie folgt aussehen:

```
alias eth0 ne
alias eth1 off

options ne io=0x320 irq=7
```

Wir definieren mit der ersten Zeile einen Alias eth0 (erste Ethernetkarte) und weisen ihm den Wert ne (der Name des passenden Moduls) zu. Die zweite Zeile definiert den Alias eth1 mit dem Wert off. Wird dieser Wert gegeben, so wird dieses Modul nie geladen, sogar dann nicht, wenn es explizit von Hand aufgerufen werden würde.

Die eigentliche Definition der Parameter für das Modul ne wird mit der dritten Zeile vorgenommen. Das Schlüsselwort options legt fest, daß das Modul ne die Parameter io=0x320 und irq=7 bekommt.

Im Gegensatz zur Übergabe der Parameter an den Kernel werden hier die Parameter genau bezeichnet. Das führt häufig zu Verwechslungen, weil ein und das selbe Modul eine unterschiedliche Sytax für die Übergabe der Parameter benutzt, es ist aber nicht zu ändern.

Ein weiterer interessanter Aspekt der Datei /etc/conf.modules ist die Möglichkeit, mit den Schlüsselwörtern **pre-install** und **post-install** festzulegen, daß bestimmte Kommandos vor (pre-install) bzw. nach (post-install) dem Einhängen eines Moduls ausgeführt werden. Bei den Kommandos handelt es sich in den häufigsten Fällen wiederum um Kommandos zum Laden (oder Entladen) von Modulen.

Die Syntax dieser beiden Befehle ist dabei sehr einfach:

```
pre-install Modul Kommando
```

```
post-install Modul Kommando
```

Den gleichen Mechanismus gibt es genauso wiederum zum Entladen von Modulen, hier heißen dann die Schlüsselwörter:

```
pre-remove Modul Kommando
post-remove Modul Kommando
```

Wenn also ein Stück neue Hardware installiert werden soll und diese neue Hardware über ein Modul angesprochen wird, so ist die Datei /etc/conf.modules der Ort, an dem die entsprechenden Parameter für die Module eingetragen werden. Die meisten Distributionen liefern bereits eine solche Datei mit, die schon verschiedenste Einstellungen beispielhaft gesetzt hat. Diese Einstellungen sind mit dem #-Zeichen auskommentiert und können durch das Entfernen dieses Zeichens aktiviert werden.

Die Meldungen des Bootvorgangs nachlesen

Jedes Linux-System protokolliert alle Vorkommnisse in einem Logbuch mit. Der Daemon, der dieses Logbuch schreibt ist der **syslogd** über den im Abschnitt 1.111.3 noch mehr zu lernen sein wird. Für die Meldungen des Kernels selbst steht noch ein weiterer Daemon zur Verfügung, der **klogd**, also der Kernel Log Daemon. Auch er schreibt Logbuchmeldungen, allerdings die des Kernels selbst. Beide diese Logbuchdaemonen schreiben ihre Meldungen in der Regel in die Datei **/var/log/messages**.

Diese Datei ist also das Systemlogbuch, ein Fundus von Informationen, über alle möglichen Vorkommnisse, die das System betreffen. Auch die Startmeldungen des Kernels sind hier größtenteils zu finden. Allerdings eben nur größtenteils, denn um die entsprechenden Daemonen zu starten, muß der Kernel schon geladen, und seine Dateisysteme müssen eingehängt sein. Die ersten - und für die Fehlersuche meist interessantesten - Meldungen des Kernels sind in dieser Datei also nicht zu finden.

Der Kernel hält aber einen internen Ringbuffer aufrecht, in den er alle Meldungen hineinschreibt, die etwa beim Start ausgegeben werden. Um diesen Ringbuffer zu lesen, gibt es das Programm **dmesg**. Es ließt den Buffer und gibt den Inhalt auf der Standard-Ausgabe aus. Hier finden sich dann alle Meldungen des Kernels, die er seit dem Booten erstellt hat, bzw. wenn der Rinbuffer voll ist, die letzten Meldungen, die in den Buffer passen. Eine typische Form der Ausgabe des dmesg-Programms ist hier einsehbar

Mit diesem Programm stehen uns also auch die Kernel-Meldungen zur Verfügung, die vom Syslog-Daemon bzw. Klog-Daemon noch nicht protokolliert werden konnten, weil diese Daemonen noch gar nicht geladen waren als die Meldungen anfielen.

Das ist eine gerne benutzte Möglichkeit, wenn z.B. beim Start etwas nicht funktioniert und man Rat sucht. Mit dem Befehl

```
dmesg > Bootmeldungen
```

können die Meldungen in eine Datei geschrieben werden, die dann ausgedruckt oder per Mail an einen Supporter geschickt werden kann.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.106.2

Ändern des Runlevels und Niederfahren oder Neustart des Systems

---

Beschreibung: Prüfungskandidaten sollten in der Lage sein, den Runlevel des Systems zu verwalten. Dieses Lernziel beinhaltet den Wechsel in den Single-User Modus und das Niederfahren oder Rebooten des Systems. Kandidaten sollten auch in der Lage sein, Benutzer vor dem Wechsel des Runlevels zu benachrichtigen und Prozesse sauber zu beenden. Dieses Lernziel beinhaltet auch das Setzen des Standard-Runlevels.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **shutdown**
- **init**
- `/etc/inittab`

Runlevel und ihr Wechsel

Linux bietet verschiedene Betriebszustände an, sogenannte Runlevel (manchmal auch init-level genannt). Im Wesentlichen geht es dabei um die Frage, welche Dienste zur Verfügung gestellt werden und welche nicht.

Der erste Prozess eines Linux-Systems (der Prozess mit der ProzessID 1) ist der init-Prozess. Dieser Prozess startet (initialisiert) sowohl die Benutzerprozesse (bzw. die getty-Prozesse, bei denen sich die Benutzer dann später anmelden können), als auch die Daemon-Prozesse, also die, die kein eigenes Termonal benutzen sondern im Hintergrund als Dienst arbeiten.

Um nicht nur einen Betriebszustand zu ermöglichen, der immer die selben Dienste anbietet oder nicht, gibt es eben die sogenannten Runlevel. Welche Dienste in welchem Runlevel gestartet werden sollen, lässt sich in der Datei `/etc/inittab` einstellen. Das Format und die verschiedenen Möglichkeiten dieser Datei können auf der Handbuchseite von inittab(5) nachgelesen werden.

Grundsätzlich stehen die Runlevel 0 bis 6 und S (oder s) zur Verfügung. Dabei haben die Runlevel 0, 6 und S eine festgelegte Bedeutung, alle anderen stehen frei zur Verfügung. Verschiedene Linux-Distributionen belegen diese übriggebliebenen Runlevel mit verschiedenen Bedeutungen, es existiert kein Standard. Meist werden mindestens drei Runlevel definiert, einer für Multiuser ohne Netz, einer für Multiuser mit Netz und einer für Multiuser mit Netz und graphischem Login (xdm). Multiuser heißt in diesem Fall, daß das System in der Lage ist, mehrere User gleichzeitig zu bedienen.

Die drei festgelegten Runlevel sind:

Runlevel	Bedeutung
0	System herunterfahren ohne anschließenden Neustart. (halt)
6	System herunterfahren und neu starten. (reboot)
S	System im Single User Mode betreiben

Der Single User Modus ist ein Modus, der in der Systemverwaltung immer dann benötigt wird, wenn Arbeiten durchgeführt werden sollen, bei denen Aktivitäten anderer Benutzer stören könnten. Zum Beispiel die Reparatur eines Dateisystems oder die Spiegelung einer Platte.

Die Datei /etc/inittab enthält einen Eintrag, der dem init-Prozess mitteilt, welcher Runlevel der voreingestellte ist, in dem das System hochfahren soll (initdefault). Das wird in der Regel ein benutzerdefinierter Runlevel (1-5) sein.

Um einen Runlevel manuell zu wechseln, kann der Systemverwalter (und nur er!) das Programm init mit dem gewünschten Runlevel aufrufen. Der Befehl

```
init S
```

würde also das System in den Single User Modus bringen, der Befehl

```
init 3
```

würde es in den Runlevel 3 bringen. Statt init kann auch das Programm **telinit** benutzt werden, das exakt genauso funktioniert. In der Regel ist telinit nur ein symbolischer Link auf init, der aus Kompatibilitätsgründen existiert.

Sauberes Herunterfahren des Systems

Um ein Linux-System jetzt herunterzufahren oder neu zu starten könnten wir jetzt einfach schreiben `init 0` bzw. `init 6`. Diese Lösung funktioniert zwar, ist aber aus verschiedenen Gründen nicht die beste Art.

Es sind verschiedene Aufgaben zu erledigen, um ein System sauber herunterzufahren. Diese Aufgaben beinhalten die saubere Beendigung laufender Dienste und Benutzerprogramme, die Synchronisation der Plattenlaufwerke (das physikalische Schreiben des Cache-Inhalts auf

die Platte) und nicht zuletzt die Benachrichtigung der noch eingeloggten User, daß das System jetzt heruntergefahren wird.

Es existieren heute verschiedene Programme, die diese Aufgaben zum Teil erledigen, die aber den für die Systemverwaltung wichtigsten Teil weglassen - die Benachrichtigung der User. Die Programme **reboot**, **poweroff**, **suspend** und **halt** dienen alle dazu, das System auf verschiedene Arten herunterzufahren (in Wahrheit sind reboot, poweroff und suspend nur Links auf halt). Diese Programme teilen dem Kernel mit, daß er eben das System runterfahren, neu starten usw. soll. Moderne Versionen dieser Programme ermitteln, ob sie im Runlevel 0 oder 6 ausgeführt wurden und wenn nicht, rufen sie das Programm **shutdown** auf.

Das Programm **shutdown** ist der beste Weg, ein System herunterzufahren oder neu zu starten. Shutdown tut dies auf einem sicheren Weg. Alle eingeloggten User bekommen eine Nachricht, die sie auf das Herunterfahren hinweist und ein einloggen wird blockiert. Es ist möglich, das System sofort herunterzufahren, oder zu einer anzugebenden Uhrzeit oder Zeitdifferenz.

Shutdown schickt allen Prozessen vor dem Herunterfahren zunächst ein TERM-Signal. Das gibt Programmen wie z.B. vi oder joe die Möglichkeit, die ungesicherten Dokumente noch zu speichern, bevor die Programme beendet werden.

Shutdown wird folgendermaßen aufgerufen:

```
shutdown [Optionen] Zeit
```

Die wichtigsten Optionen sind

`-r`

Reboot. Shutdown fährt das System herunter und startet es dann erneut. Genau genommen wechselt shutdown in den Runlevel 6 nachdem es alle anderen Arbeiten erledigt hat.

`-h`

Halt. Shutdown fährt das System nur herunter ohne einen Neustart danach. Also ein Wechsel in den Runlevel 0.

`-c`

Cancel. Ein schon laufender Shutdown wird unterbrochen und somit das System nicht heruntergefahren.

Das Argument *Zeit* kann in mehreren Formen angegeben werden. Entweder es enthält eine Uhrzeit in der Form `hh:mm` also Stunde und Minute. Dann wird das System zur angegebenen Zeit heruntergefahren. Die Warnmeldung erscheint aber trotzdem schon gleich nach dem Programmaufruf und vom Zeitpunkt des Programmaufrufs bis zum eigentlichen Shutdown kann sich niemand mehr einloggen. Die zweite Form der Zeitangabe ist die Form `+m` was einfach die Anzahl der Minuten ist, bis das System heruntergefahren wird. Steht das Wort `now` statt einer Zeitangabe, so wird es als `+0m` interpretiert.

Die häufigste Anwendung von Shutdown ist das sofortige Herunterfahren mit

```
shutdown -h now
```

oder das gleiche mit reboot statt halt:

```
shutdown -r now
```

Linux kennt die Möglichkeit, der Tastenkombination Strg-Alt-Entf einen Befehl zuzuordnen (einstellbar in /etc/inittab). In den meisten Fällen ist dieser Befehl ein `shutdown -r now`. Das bedeutet aber, daß jeder User, auch ein Normaluser, die Möglichkeit hat, das System herunterzufahren, wenn er Zugriff auf die Tastatur der Systemkonsole hat. Um das einzuschränken kann shutdown in /etc/inittab mit der Option `-a` aufgerufen werden. In diesem Fall überprüft shutdown, ob ein User, der in der Datei `/etc/shutdown.allow` aufgeführt ist, an einer der Consolen arbeitet. Nur dann wird shutdown ausgeführt. Das Format der Datei /etc/shutdown.allow ist einfach ein Username pro Zeile.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.107.2

Verwaltung von Druckern und Druckerwarteschlangen

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Druckerwarteschlangen und Druckjobs von Benutzern zu verwalten. Dieses Lernziel beinhaltet die Kontrolle von Druckservern und Druckerwarteschlangen und das Lösen allgemeiner Druckprobleme.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **lpc**
- **lpq**
- **lprm**
- **lpr**
- `/etc/printcap`

Drucken unter Linux läuft immer nach einem ganz bestimmten Schema ab. Zunächst einmal geht es in diesem Abschnitt um die Verwaltung von Druckerwarteschlangen. Die Einrichtung verschiedener Drucker wird in den folgenden Kapiteln erläutert.

Druckerwarteschlangen - Das Prinzip

Druckerwarteschlangen arbeiten unter Unix nach einem feststehenden Prinzip, nach dem auch alle anderen Systeme mit Warteschlangen arbeiten. Es gibt viele völlig unterschiedliche Subsysteme, die dieses Prinzip benutzen, so etwa das **at**-System oder das Fax-System. Das zugrundeliegende Prinzip ist einfach zu durchschauen, aber sehr funktional.

Eine Warteschlange (engl. spool) ist immer ein Verzeichnis irgendwo unter `/var/spool`. In dieses Verzeichnis werden die Druckaufträge in zwei verschiedenen Dateien abgelegt. Die erste Datei trägt einen Namen, der mit `cf` beginnt, diese Datei enthält die Kontrollinformationen des Jobs. (`cf` - control file). Kontrollinformationen sind z.B. der Name der zu druckenden Datei, der Name des Users, der den Druckauftrag aufgegeben hat und der Hostname des Rechners, von dem aus der Auftrag abgegeben wurde. Die zweite Datei trägt einen Namen, der mit den Buchstaben `df` beginnt und enthält die zu druckende Datei selbst (`df` - data file).

Wird also ein Druckauftrag abgegeben, so werden eigentlich nur diese zwei Dateien erstellt. Das eigentliche Drucken wird dann von einem Daemon-Prozeß erledigt, der diese Warteschlange abarbeitet. So können auch Aufträge abgeschickt werden, wenn der Drucker gerade nicht zur Verfügung steht. Denn sie werden ja nur in das Spool-Verzeichnis geschrieben. Selbst wenn der Rechner danach ausgeschaltet wird, sind die Aufträge nicht verschwunden. Sobald der Druckerdaemon das nächste Mal läuft und der Drucker zur Verfügung steht, wird die zu druckende Datei dann ausgedruckt.

Definition von Druckerwarteschlangen in /etc/printcap

Jeder Drucker, der dem System bekannt ist, also auch Netzwerkdrucker auf anderen Rechnern, erhält einen Eintrag in der Datei `/etc/printcap`. Dieser Eintrag wird im Abschnitt 1.107.4 genauer erklärt. Hier interessieren uns jetzt nur zwei Kleinigkeiten dieses Eintrags.

Ein Eintrag in `/etc/printcap` enthält - neben anderen Informationen - den Namen des zu verwendenden Druckers und die Angabe, welches Verzeichnis das zu verwendende Spoolverzeichnis ist. Zum Beispiel zeigt der folgender Eintrag einen Drucker:

```
lp|hplaser|PS;r=600x600;q=medium;c=gray;p=a4;m=auto:\
  :sd=/var/spool/lpd/lp:\
  :lf=/var/spool/lpd/lp/log:\
  :af=/var/spool/lpd/lp/acct:\
  :if=/etc/apsfilter/basedir/bin/apsfilter:\
  :lp=/dev/lp0:\
  :sh:\
  :mx#0:
```

Die für uns interessanten Einträge sind hier:

lp|hplaser

Diese beiden Werte sind die Namen des Druckers, unter denen er angesprochen werden kann. Der Name `lp` hat eine Sonderbedeutung. Der Drucker, der diesen Namen trägt ist der Standard-Drucker, an den alle Aufträge geschickt werden, die keine expliziten Angaben über einen zu verwendeten Drucker machen.

sd=/var/spool/lpd/lp

Der `sd`-Eintrag bezeichnet das Spoolverzeichnis (`sd` - spool directory) in das die Aufträge abgelegt werden sollen.

Mit Hilfe dieser Information weiß das Drucksystem von Linux also, daß der Drucker `lp` oder `hplaser` das Spoolverzeichnis `/var/spool/lpd/lp` benutzt. Wenn also ein Druckauftrag an diesen Drucker abgeschickt wird, dann werden sowohl die Kontrolldatei, als auch die zu druckende Datei in diesem Verzeichnis abgelegt.

Befehle für den Umgang mit Druckerwarteschlangen

Zum Umgang mit den Druckerwarteschlangen stehen uns verschiedene Befehle zur Verfügung, die in dieser Form auch wieder sehr typisch für alle Unix/Linux Warteschlangensysteme sind. Diese Befehle werden im Folgenden kurz beschrieben:

Druckaufträge losschicken

Um einen Druckauftrag loszuschicken, wird der Befehl **lpr** benutzt. **lpr** wird entweder ein Dateinamen mitgegeben, um eine bestimmte Datei zu drucken, oder - falls kein Dateinamen angegeben wurde - druckt **lpr** seine Standard-Eingabe.

Der Name des zu verwendenden Druckers kann mit der Option `-P` angegeben werden. Wird kein Name angegeben, so wird standardmäßig der Drucker mit dem Namen `lp` benutzt.

Die Anzahl der zu druckenden Kopien kann mit der Option `-#` angegeben werden. Vorsicht bei der Anwendung dieses Parameters auf der Shell. Die Shell mißinterpretiert das #-Zeichen als Kommentarzeichen, es muß also vor Interpretation geschützt werden (`-\#`).

Um also die Datei `/etc/passwd` auf dem Drucker `lp2` vier mal auszudrucken, wird der Befehl

```
lpr -Plp2 -\#4 /etc/passwd
```

angegeben. Soll die Datei nur einmal auf dem Standarddrucker ausgegeben werden, so genügt ein

```
lpr /etc/passwd
```

Der Befehl **lpr** druckt nicht wirklich, sondern legt die Aufträge nur in die entsprechenden Spool-Verzeichnisse ab. Erst der Druckerdaemon **lpd** erledigt das Drucken selbst.

Inhalt der Warteschlange anzeigen

Um den Inhalt einer Druckerwarteschlange anzuzeigen, existiert der Befehl **lpq**. Er zeigt die aufgegebenen Druckaufträge für einen mit `-P` angegebenen Drucker oder - falls der Druckername weggelassen wurde - für den Standarddrucker.

Die Druckaufträge werden zusammen mit einer Jobnummer ausgegeben, die wir später brauchen werden, um einzelne Aufträge wieder zu löschen. Die Ausgabe des Befehl könnte etwa so aussehen:

```
lp is ready and printing
Rank   Owner      Job  Files                                 Total Size
active root       447  /etc/passwd                           1204 bytes
```

Wir entnehmen also daraus, daß nur ein Druckauftrag vorliegt, der dem User `root` gehört, also von ihm aufgegeben wurde. Der Auftrag hat die JobID 447 und druckt die Datei `/etc/passwd`, die insgesamt 1204 Byte groß ist.

Die erste Zeile ist für die Diagnose von Fehlern sehr wichtig. Sie zeigt den Status des Druckerdaemons an. Es könnten hier auch Fehlermeldungen wie

```
Warning: lp is down:
Warning: lp queue is turned off
```

ausgegeben werden, wenn der Drucker deaktiviert wurde oder einfach eine Leerzeile, wenn kein Druckerdaemon vorhanden ist.

Druckaufträge aus der Warteschlange löschen

Um einzelne Druckaufträge wieder aus einer Warteschlange zu löschen, gibt es den Befehl **lprm**. Auch er kennt den Parameter `-P`, mit dem der gewünschte Druckername angegeben werden kann und auch er bezieht sich auf den Standarddrucker, wenn dieser Parameter weggelassen wurde.

lprm erwartet die Angabe einer oder mehrerer JobIDs als Parameter, die die zu löschenden Jobs beschreiben. Diese JobIDs können vor dem Aufruf von **lprm** durch **lpq** festgestellt werden.

Drucker kontrollieren

Um verschiedene Drucker zu kontrollieren, existiert das Printer-Control-Program **lpc**. Dieses Programm kann verschiedene Befehle verstehen, die entweder auf der Kommandozeile eingegeben werden, oder an einen Interpreter weitergegeben werden, wenn **lpc** ohne Parameter aufgerufen wurde.

Für jeden Drucker, der in `/etc/printcap` angegeben wurde, kann **lpc** benutzt werden um

- den Drucker zu aktivieren oder zu deaktivieren,
- die Warteschlange des Druckers zu aktivieren oder zu deaktivieren,
- die Reihenfolge der Druckjobs in der Warteschlange zu ändern,
- den Status der Drucker, ihrer Warteschlangen und ihrer Daemonen zu erfragen.

Wird **lpc** ohne Parameter aufgerufen, so wartet er auf Kommandos von der Standard-Eingabe. Sind Parameter angegeben, so interpretiert **lpc** den ersten Parameter als Kommando und die folgenden Parameter als Argumente dieses Kommandos. **lpc** kennt folgende Kommandos:

? [*Kommando*]

help [*Kommando*]

Gibt eine kurze Hilfsmeldung über das *Kommando* aus. Wurde kein Kommando angegeben, so gibt der Befehl eine Liste aller verstandenen Kommandos aus.

abort all | *Drucker*

Beendet einen aktiven Druckerdaemon und verunmöglicht die Angabe weiterer Druckaufträge. Entweder gilt dieser Befehl für alle (all) oder für den angegebenen Drucker.

clean all | *Drucker*

Alle Temporärdateien, Daten- oder Control-Dateien von Aufträgen, die nicht ausgeführt werden konnten, werden gelöscht. Entweder gilt dieser Befehl für alle (all) oder für den angegebenen Drucker.

disable all | *Drucker*

Schaltet die Warteschlange ab. Das verunmöglicht die Angabe weiterer Druckjobs durch **lpr**. Entweder gilt dieser Befehl für alle (all) oder für den angegebenen Drucker.

down all | *Drucker* Nachricht

Schaltet die Warteschlange ab und schreibt die angegebene Nachricht in die Status-Datei des Druckers (oder aller Drucker). Die Statusmeldung wird dann den Usern angezeigt, die einen Druckauftrag abgeben wollen.

enable all | *Drucker*

Schaltet eine Warteschlange wieder an. Damit ist es wieder möglich, Druckaufträge mit **lpr** an den entsprechenden Drucker zu schicken.

restart all | *Drucker*

Startet für den angegebenen Drucker (oder alle Drucker) einen neuen Druckerdaemon. Ein eventuell vorher vorhandener Daemon wird vor dem Neustart abgebrochen (wenn der root-User den Befehl aufgibt).

start all | *Drucker*

Ermöglicht das Drucken und startet den Druckerdaemon für den angegebenen Drucker (oder alle).

status all | *Drucker*

Zeigt den Status des angegebenen Druckers (oder aller Drucker).

stop all | *Drucker*

Hält den Druckerdaemon an, nachdem der aktuelle Auftrag abgearbeitet wurde und verunmöglicht die Aufgabe weiterer Aufträge.

topq *Drucker Jobids*

Verschiebt die JobIDs in der angegebenen Reihenfolge an den Anfang der Warteschlange.

up all | *Drucker*

Ermöglicht alles und startet einen neuen Druckerdaemon. Das Gegenteil von **down**.

All diese Befehle können entweder als Kommandozeilenparameter eingegeben werden, wie etwa

```
lpq up all
```

oder - wenn lpc ohne Parameter aufgerufen wurde - als Befehl von der Standard-Eingabe:

```
lpc
lpc> up lp
lp:
      printing enabled
      daemon started
lpc> exit
```

Wurde **lpq** ohne Parameter aufgerufen, so kann der Interpreter über die Befehle `exit` oder `quit` wieder verlassen werden.

1.107.3

Druck von Dateien

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Druckerwarteschlangen zu verwalten und Druckjobs zu bearbeiten. Dieses Lernziel beinhaltet das Hinzufügen und Entfernen von Jobs an konfigurieren Druckerwarteschlangen und das Konvertieren von Textdateien in PostScript für den Ausdruck.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **lpr**
- **lpq**
- **mpage**

Die Verwaltung von Druckerwarteschlangen und das Hinzufügen und Entfernen von Druckjobs wurde bereits im letzten Abschnitt besprochen. Interessant ist also das Konvertieren von Textdateien nach Postscript.

Das Prinzip von Drucken unter Linux ist immer mit Postscript verbunden. Da es keine wirklich eindeutige Definition von Druckertreiber unter Linux gibt, wird immer erstmal davon ausgegangen, daß ein Drucker Postscriptfähig ist. Die Ausgabe einer Postscript-Datei an einen Drucker kann dann über einen Filter laufen, der aus dem Postscript die für den Drucker verständliche Sprache generiert. Nahezu jede Anwendung unter Linux kann ihre Ausgaben in Postscript vornehmen. Dadurch ist also eine Art einheitlicher Druckerschnittstelle geschaffen. Die genaue Implementierung solcher Filter wird Thema des nächsten Abschnitts sein.

Für uns ist hier jetzt interessant, wie Textdateien in Postscript verwandelt werden können. Textdateien sind in diesem Zusammenhang beileibe nicht nur echte Dateien, sondern immer auch Datenströme, die beispielsweise von Linux-Texttools erstellt wurden. Das Programm **lpr** mit dem Druckaufträge gegeben werden, kann ja auch seine Daten aus der Standardeingabe lesen, es ist also durchaus möglich, das Ergebnis eines Befehls an dieses Programm zu pipen.

Für die Konvertierung von Textdateien nach Postscript existieren verschiedene Programme, die im Prinzip alle in der Lage sind, aus einer Textdatei oder einem Datenstrom Postscript zu erzeugen. Die jeweiligen Fähigkeiten sind in Einzelheiten unterschiedlich, so können

manche Programme die Ausgabe in Spalten setzen oder mehrere Seiten auf eine Druckseite zusammenfassen. Andere Möglichkeiten sind die Formatierung von Text nach bestimmten Regeln, das Syntax-Highlighting für bestimmte Programmiersprachen oder die automatische Einrückung zur leichteren Lesbarkeit. Die bekanntesten Programme für diese Aufgabe sind

- **a2ps**
  Verwandelt Textdateien in Postscript. Direkte Weitergabe an den voreingestellten Drucker (oder an jeden beliebigen Drucker mit der Option `-P`*Druckername*) ist genauso möglich, wie Ausgabe in eine Postscriptdatei (`-o` Dateiname) oder Ausgabe auf die Standard-Ausgabe (`-o -`). Standardmäßig werden zwei Seiten auf ein Blatt gedruckt. Seiten werden gerahmt und mit Dateinamen, Usernamen, Druckdatum versehen. (Beides kann über Optionen auch abgeschaltet werden) Die Option `-E` ermöglicht Pretty-Printing für verschiedenste Formate wie C, Bash, ...

  Sehr viele Parameter erlauben die verschiedensten Einstellungen, wie etwa Anzahl der Zeilen pro Blatt, Anzahl der Zeichen pro Zeile, Titeleinstellungen, Zeilennummerierung, ...
- **enscript**
  Dieses Programm hat praktisch alle Fähigkeiten von **a2ps** und noch einiges mehr. Es ermöglicht beispielsweise die Definition eigener Pretty-Printing Stile und den Druck von Textdateien in beliebig vielen Spalten. Außerdem kann als Ausgabeformat nicht nur Postscript, sondern auch HTML, ANSI (Terminalausgabe) und RTF erzeugen. Es sind verschiedene Ausgabeformate möglich, so daß entweder 1,2,4 oder 8 Seiten auf ein Blatt gedruckt werden können.
- **mpage**
  Dieses Programm ließt normale Textdateien oder Postscript-Dateien und druckt sie auf einem Postscript-Drucker mit einer verkleinerten Ausgabe, so daß mehrere Seiten auf ein Blatt Papier passen. Diese Fähigkeit haben zwar auch die letzten beiden genannten Programme **a2ps** und **enscript**, **mpage** ist aber im Gegensatz zu diesen Programmen auch in der Lage, schon existierende Postscript-Dateien inklusive aller Bilder, Graphiken und sonstiger Features zu verkleinern. Das kann sehr praktisch sein, um einen großen Ausdruck auf wenige Blätter auszudrucken um das Erscheinungsbild zu überprüfen.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

1.107.4

Installation und Konfiguration von lokalen und Netzwerkdruckern

---

Beschreibung: Prüfungskandidaten sollten in der Lage sein, einen Printerdämon zu installieren und einen Druckerfilter (z.B. apsfilter oder magicfilter) zu installieren und zu konfigurieren. Dieses Lernziel beinhaltet das Konfigurieren von lokalen und Netzwerkdruckern für ein Linux-System, inklusive PostScript-, Non-PostScript- und Samba-Druckern.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **lpd**
- `/etc/printcap`
- `/etc/apsfilter/*`
- `/var/lib/apsfilter/*/`
- `/etc/magicfilter/*/`
- `/var/spool/lpd/*/`

Die Datei /etc/printcap

Der Standard-Weg zur Installation eines Druckers unter Linux läuft über die Datei `/etc/printcap`. In dieser Datei hat jeder Drucker, der an das System angeschlossen ist, aber auch alle Drucker, die über das Netz erreichbar sein sollen, einen eigenen Eintrag. Jeder dieser Einträge ist eigentlich nur eine Zeile lang, wird aber (durch Verwendung eines Backslash vor dem Zeilentrenner) zur leichteren Lesbarkeit auf mehrere Zeilen verteilt.

Ein Eintrag für einen Drucker beginnt immer mit dem oder den Namen des Druckers. Danach folgen verschiedene Angaben, die jeweils durch Doppelpunkte voneinander getrennt sind und meist die Form

Einstellung=Wert

besitzen. Typische Beispiele für die möglichen Einstellungen sind:

af=*Dateiname*

Name der Account-Datei, für diesen Drucker

if=*Dateiname*

Name des Input-Filters, den dieser Drucker benutzen soll. Input-Filter sind Programme, die die zu druckende Datei in ein dem Drucker verständliches Format bringen, siehe weiter unten.

lf=*Dateiname*

Name der Logdatei, in die Fehlermeldungen aufgenommen werden sollen.

lp=*Dateiname*

Name der Gerätedatei des Gerätes, an dem der Drucker angeschlossen ist.

mx#*Zahl*

Maximale Dateigröße in Blöcken. Die Angabe 0 bedeutet, daß es keine maximale Dateigröße gibt.

rm=*Hostname*

Name der entfernten Maschine (remote machine), die einen Netzwerkdrucker zur Verfügung stellt.

rp=*Druckername*

Name des Druckers auf der entfernten Maschine.

sd=*Verzeichnis*

Das Verzeichnis, in dem die Aufträge abgespeichert werden. (Spool Directory)

sh

Es wird keine Titelseite gedruckt. (suppress header)

Ein sehr einfacher Eintrag für einen Drucker könnte also folgendermaßen aussehen:

```
lp|hplaser:\
      :lp=/dev/lp0:\
      :sd=/var/spool/lp:\
      :mx#0:\
      :lf=/var/spool/lp/hp-log:
```

Der Drucker heißt entweder `lp` oder `hplaser` (der zweite Name dient als Beschreibung). Er ist an die Schnittstelle `/dev/lp0`

angeschlossen und sein Spoolverzeichnis ist `/var/spool/lp`. Es existiert keine Größenbeschränkung (`mx#0`) und Fehlermeldungen werden in die Datei `/var/spool/lp/hp-log` geschrieben.

Die Verwendung von Input-Filtern wird weiter unten beschrieben.

Um auch Netzwerkdrucker an anderen Maschinen ansprechen zu können, benötigen wir entsprechend Einträge wie der folgende:

```
lp1|remote printer on marvin:\
      :sd=/var/spool/lp1:\
      :rm=marvin.mydomain.net:\
      :rp=lp:\
      :sh:mx#0:
```

Dieser Eintrag beschreibt einen Drucker, der auf unserem System mit dem Namen `lp1` angesprochen wird. Er benutzt das lokale Spoolverzeichnis `/var/spool/lp1`. Anstatt einem Eintrag `lp=` haben wir jetzt den Eintrag `rm=`, der besagt, daß der Drucker auf einer *remote machine* also einem entfernten Rechner angeschlossen ist. Dieser Rechner ist in unserem Beispiel der Rechner `marvin.mydomain.net`. Der Druckername auf dem entfernten Rechner (`rp` - remote printer) ist `lp`. Es wird keine Trennseite ausgegeben und es existiert keine Größenbeschränkung.

Es ist problemlos möglich, daß ein physikalischer Drucker mehrere Printcap-Einträge besitzt. So können z.B. verschiedene Inputfilter für den gleichen Drucker verwendet werden, oder ein Drucker kann entweder farbig oder in Graustufen angesprochen werden, je nachdem, unter welchem Namen er angesprochen wurde. Der Name `lp` ist dabei der voreingestellte Drucker.

Apsfilter

apsfilter erlaubt es, verschiedene Dateitypen direkt zu drucken, ohne daß der User die Datei von Hand in eine dem Drucker verständliche Sprache konvertieren muß.

apsfilter benutzt verschiedene Programme anderer Hersteller um die verschiedenen Dateiformate zunächst in Postscript zu verwandeln. Wenn ein echter Postscript-Drucker vorliegt, so werden diese konvertierten Daten dann einfach an den Drucker weitergereicht, ansonsten werden sie an Ghostscript weitergeleitet, der sie dann in eine für den jeweiligen Drucker verständliche Sprache konvertiert.

Die Frage, ob ein Drucker von **apsfilter** unterstützt wird, ist also primär die Frage, ob er entweder ein Postscript-Drucker ist, oder von Ghostscript unterstützt wird.

Welche Filter (siehe letzten Abschnitt) **apsfilter** benutzt, kann über seine Konfigurationsdatei `/etc/apsfilter/apsfilterrc` eingestellt werden. Hier können auch alle möglichen anderen Einstellungen vorgenommen werden.

apsfilter bietet ein kleines Programm, mit dem die komplette Einrichtung eines Druckers, inklusive der `printcap`-Einträge, vorgenommen wird. Dieses Programm heißt **apsfilterconfig**.

In `/var/lib/apsfilter` werden alle Aktionen von **apsfilterconfig** zwischengespeichert, so daß sie auch bei erneuter Installation wieder zur Verfügung stehen.

Ein typischer Eintrag in `/etc/printcap` für einen Drucker, der mit **apsfilter** angesteuert wird wäre

```
lp|PS;r=600x600;q=high;c=gray;p=a4;m=auto:\
    :lp=/dev/lp0:\
    :if=/etc/apsfilter/basedir/bin/apsfilter:\
    :sd=/var/spool/lpd/lp:\
    :lf=/var/spool/lpd/lp/log:\
    :af=/var/spool/lpd/lp/acct:\
    :mx#0:\
    :sh:
```

Wichtig ist hier die Angabe des Input-Filters (`if=/etc/apsfilter/basedir/bin/apsfilter`). Jede Datei, die an diesen Drucker geschickt wird, wird zuerst durch das angegebene Programm (hier `/etc/apsfilter/basedir/bin/apsfilter`) geschickt. Erst dieses Programm schickt den Druckauftrag an den eigentlichen Drucker.

Magicfilter

magicfilter ist ein erweiterbarer und einstellbarer automatischer Druckerfilter. Der Typ einer zu druckenden Datei wird anhand ihrer Magic-Nummern ermittelt und **magicfilter** wendet dann einen entsprechenden Filter an, der diesen Dateityp in die Ausgabesprache des jeweiligen Druckers konvertiert.

magicfilter ist primär dazu gedacht, als Eingabefilter (`if=`) für den LPD-Druckspooler zu arbeiten. Die Optionen, die Magicfilter versteht, sind genau die, die **lpd** an den Inputfilter weiterreicht.

Um **magicfilter** zusammen mit dem LPD-Spooler zu nutzen, muß im entsprechenden printcap-Eintrag der input-filter (`if=`) Eintrag auf das Programm magicfilter gesetzt sein. Da aber ein printcap-Eintrag keine Optionen erlaubt, wird hier einfach ein Script als Eingabefilter gewählt, das ausführbar ist und als erste Zeile den Eintrag

```
#!/usr/sbin/magicfilter
```

aufweist.

Dieses Script liegt standardmäßig im Verzeichnis `/etc/magicfilter` und trägt den Namen des zu verwendenden Druckers. Im Lieferumfang von **magicfilter** sind bereits etwa 80 verschiedene solcher Scripte für die gängigsten Drucker enthalten. Der Inhalt dieser Scripte ist - neben der bereits erwähnten ersten Zeile - immer nach folgendem Schema aufgebaut:

Offset Zeichenkette Art

Wobei *Offset* die Stelle innerhalb der zu druckenden Datei bezeichnet, an der eine bestimmte Zeichenkette zu finden ist, *Zeichenkette* ist die Zeichenkette, die an dieser Stelle erwartet wird und *Art* beschreibt die Aktion, die dann mit der Datei ausgeführt wird, um sie zu drucken.

Ein typischer Eintrag dieser Datei wäre z.B.

```
# GIF files
0       GIF87a          pipe    /usr/bin/giftopnm  2>/dev/null
0       GIF89a          pipe    /usr/bin/giftopnm  2>/dev/null
```

Wenn also an der Position 0 (Anfang der Datei) entweder die Zeichenkette `GIF87a` oder `GIF89a` zu finden ist, dann handelt es sich bei der zu druckenden Datei um ein GIF-Bild. Die Aktion, die durchgeführt werden soll ist `pipe /usr/bin/giftopnm 2>/dev/null`, der Inhalt der GIF-Datei wird also an das Programm `/usr/bin/giftopnm` weitergeleitet, das die GIF-Datei in ein PNM-Bild verwandelt. Das kann dann entsprechend von einem Postscript-Drucker gedruckt werden.

Damit auch nicht-Postscript Drucker mit **magicfilter** benutzt werden können, gibt das Programm die Ausgaben anschließend an Ghostscript weiter, das dann ein für den jeweiligen Drucker verständliches Format erzeugt.

Damit diese Arbeit nicht manuell ausgeführt werden muß, wird **magicfilter** zusammen mit einem Installationsscript ausgeliefert, das den Namen **magicfilterconfig** trägt und die notwendigen Einstellungen vornimmt, nachdem der User ein paar Fragen zum angeschlossenen Drucker beantwortet hat.

Ein typischer `printcap` Eintrag für **magicfilter** lautet beispielsweise:

```
lp|hplj4l|HP Laserjet 4L:\
      :lp=/dev/lp1:sd=/var/spool/lpd/hplj4l:\
      :sh:pw#80:pl#72:px#1440:mx#0:\
      :if=/etc/magicfilter/ljet4l-filter:\
      :af=/var/log/lp-acct:lf=/var/log/lp-errs:
```

Die Angaben `pw, pl` und `px` beziehen sich auf die Seitenbreite in Zeichen, die Seitenlänge in Zeilen und die Seitenbreite in Pixeln. Ansonsten enthält der Eintrag nichts neues, der Inputfilter ist jetzt `/etc/magicfilter/ljet4l-filter`.

Einbinden eines Windows-Drucker

Über die Technik der Input-Filter ist es auch möglich, einen Drucker anzusteuern, der von einem Windows-Rechner freigegeben wurde (SMB-Drucker). Dazu wird **samba** benötigt, bzw. die Programme **smbclient** (Ein Programm zur Nutzung von SMB-Freigaben auch Druckern) und **smbprint** (ein Shellscript, das **smbclient** nutzt und als Input-Filter nutzbar ist. Ein Eintrag in `/etc/printcap` für einen SMB-Drucker wäre beispielsweise

```
lp2|remote-smbprinter:\
  :lp=/dev/null:sh:\
  :sd=/var/spool/lp2:\
  :if=/usr/local/sbin/smbprint:
```

Im Script `/usr/local/sbin/smbprint` müssen jetzt noch entsprechend die Einträge für

- Hostname des Windows-Rechners
- Freigabename des Druckers auf dem Windows-Rechner
- Username auf dem Windows-Rechner
- Passwort für den Windows-Rechner

angegeben werden. Wenn mehrere Windows-Drucker angesprochen werden sollen, so muß für jeden dieser Drucker ein eigenes `smbprint` Script existieren. Die Scripts sind einfache Shellscripts und können entsprechend kopiert und verändert werden.

Spoolerdaemon starten

Normalerweise wird für jedes System ein Druckerdaemon gestartet. Dieser Daemon trägt den Namen **lpd** und wird automatisch beim Systemstart durch ein Init-Script aufgerufen. Er ließt die Einträge in `/etc/printcap` und kennt so die vorhandenen Drucker. Sobald der Spoolerdaemon geladen ist, arbeitet er die Warteschlange ab und kann außerdem Anfragen über das Netz beantworten, und die Druckaufträge von dort entgegennehmen.

Zugriffskontrolle, wer den Drucker verwenden darf, kann über die Datei `/etc/hosts.lpd` erreicht werden.

1.108.1

Benutzung und Verwaltung lokaler Systemdokumentation

Beschreibung: Prüfungskandidaten sollten in der Lage sein, **man** und die Materialien in `/usr/share/doc/` zu benutzen und zu verwalten. Dieses Lernziel beinhaltet das Auffinden relevanter man Pages, das Durchsuchen von man Page-Abschnitten, das Auffinden von Kommandos und dazugehöriger man Pages und die Konfiguration des Zugangs zu man Sourcen und dem man System. Ebenfalls enthalten ist die Verwendung der Systemdokumentation in `/usr/share/doc/` und das Bestimmen, welche Dokumentation in `/usr/share/doc/` zu behalten ist.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **man**
- **apropos**
- **whatis**
- `MANPATH`

Alleine das Lernen für die LPI101 Prüfung hat sicherlich jedem schon gezeigt, daß eine Online-Hilfe, die zudem für jedes einzelne Programm alle denkbaren Parameter aufführt und erklärt, von größter Wichtigkeit ist. Zu diesem Zweck stellt Linux (wie jedes andere Unix auch) ein Handbuchsystem zur Verfügung, das im Folgenden dargestellt werden soll.

Zu den reinen Handbuchseiten kommen noch weitere Dokumentationen hinzu, die entweder von den Autoren der jeweiligen Programme geschrieben wurden, oder von Gruppen, die sich zum Ziel gesetzt haben, Linux für viele Anwender durchschaubarer zu machen. Eine der wichtisten solchen Gruppen ist das Linux Documentation Project (LDP), das verschiedene Dokumentationen erstellt hat, die viele Fragen beantworten. Zu nennen sind hier im Wesentlichen die sogenannten HOWTOs (wie geht was) und die FAQs (frequently asked questions - häufig gestellte Fragen).

Diese Dokumentationen wurden bisher meist unterhalb des Verzeichnisses `/usr/doc` gespeichert, seit der neuen Version 2.0 des

Filesystem Hierachy Standard wurde daraus `/usr/share/doc`.

Das Handbuchsystem von Linux

Linux bietet ein Handbuchsystem an, das offen für zusätzliche Seiten ist. Im Prinzip sollte jedes zu installierende Programm seine Handbuchseite gleich mitbringen. Der Begriff Handbuchseite mag etwas verwirren, eine Handbuchseite besteht gewöhnlich aus vielen einzelnen Seiten, die des Befehls **mount(8)** hat auf meinem System z.B. 18 physikalische Seiten.

Die Handbuchseiten sind in logische Sektionen aufgeteilt, deren Bezeichnung bei der Referenz eines Befehl dem Befehl in runden Klammern nachgestellt wird. So bedeutet das obige Beispiel **mount(8)** eben, daß die Handbuchseite für den Befehl mount aus Sektion 8 gemeint ist.

Folgende Handbuchsektionen werden traditionell unterstützt:

1 - Kommandos (User Commands)
: Die Kommandos, die vom Benutzer aus einer Shell heraus ausgeführt werden können.

2 - Systemaufrufe (System Calls)
: Funktionen, die vom Kernel selbst für den Aufrufenden ausgeführt werden.

3 - C-Bibliotheksfunktionen (Subroutines)
: Der Großteil der Funktionen der Library libc, so wie sort(3))

4 - Gerätedateien (Devices)
: Dateien, die in /dev zu finden sind.

5 - Dateiformate und Konventionen (File Formats)
: Formate von menschenlesbaren Dateien wie /etc/passwd, /etc/inittab, usw.

6 - Spiele (Games)
: Spiele und Unterhaltung

7 - Makropakete und Konventionen
: Eine Beschreibung des standardmäßigen Layouts des Dateisystems, Formatbeschreibungen für GROFF Makropakete und ähnliche Dinge.

8 - Kommandos für die Systemverwaltung (Sys. Administration)
: Kommandos wie mount(8), die nur durch den Superuser ausführbar sind.

9 - Kernelroutinen (Kernel)
: Dies ist kein standardisiertes Kapitel und wird geführt, da der Quellcode des Linux Kernels frei verfügbar ist und viele Leute an

Änderungen des Kernels parallel arbeiten.

Jede einzelne Handbuchseite ist in einer Datei abgelegt, die aber nicht etwa die formatierte ASCII Ausgabe der Handbuchseite enthält, sondern einen Quelltext für das groff-Textformatiersystem. Damit ist gewährleistet, daß die verschiedenen Handbuchprogramme, die diese Seiten darstellen wollen, das jeweils führ ihre Darstellungsform optimale Ausgabeformat erhalten. So können die Seiten auf graphischen Systemen korrekt dargestellt werden, mit verschiedenen Schriften oder Darstellungsattributen, während sie auf textbasierten Systemen einfacher, aber eben auch korrekt formatiert dargestellt werden.

Die Dateien, die diese Handbuchseiten im Quelltext beinhalten liegen sektionsweise in Verzeichnissen, die jeweils den Namen der Sektion tragen (wie etwa man1, man2, man3, ...). In der Regel werden die Dateien in diesen Verzeichnissen komprimiert abgelegt um Festplattenplatz zu sparen. Daher tragen sie normalerweise die Endung .gz

Um Handbuchseiten schneller darstellen zu können, besteht die Möglichkeit, die Formatierung von GROFF nach ASCII schon vorher vorzunehmen. Dann werden die Handbuchseiten im sogenannten CAT-Format (weil sie mit cat anzeigbar sind) gespeichert. Der Nachteil dieser Technik ist der erhöhte Bedarf an Speicherplatz auf den Platten, der Vorteil die schnellere Darstellung der Seiten mit dem man-Kommando. In diesem Fall werden die vorformatierten Seiten in Unterverzeichnissen gespeichert, die statt man jetzt cat heißen, also cat1, cat2, cat3, ...

Damit das Handbuchsystem nicht immer alle denkbaren Verzeichnisse mit Handbuchseiten durchsuchen muß, bevor eine Seite angezeigt werden kann, existiert eine Indexdatenbank, die die Namen der Handbuchseiten, ihre Kurzbeschreibung und den Platz, wo die eigentliche Seite zu finden ist. Das Programm mandb zur Verwaltung dieser Datenbank wird weiter unten noch genauer beschrieben.

Programme zum Umgang mit den Handbuchseiten

Es existieren verschiedene Programme, um sich Handbuchseiten anzeigen zu lassen. Die Palette dieser Programme reicht vom Standard-Handbuchprogramm man über Filterprogramme wie **rman** bis zu graphischen Programmen wie **xman** oder **tkman**.

Das wichtigste dieser Programme ist das Programm man. Es ist ein sehr komplexes Dienstprogramm, das hier nur kurz dargestellt werden soll, Details sind der Handbuchseite zu entnehmen...

man zeigt Handbuchseiten in einem Textformat an, das auf jeder Konsole darstellbar ist. Es hat sehr viele Parameter, von denen hier nur die wichtigsten erklärt werden, die für die LPI-101 Prüfung notwendig sind.

Die Aufrufform von man ist:

```
man [Optionen] [Sektion] Name
```

Im einfachsten Fall wird also nur der Name der gesuchten Seite angegeben, genauer gesagt der Name des Programms, dessen

Handbuchseite Sie lesen wollen. Um also Informationen über den Befehl `fdisk` zu bekommen, wird einfach der Befehl

```
man fdisk
```

eingegeben. Dieser Befehl sucht jetzt nach dem ersten Vorkommen einer Handbuchseite mit Namen `fdisk` und stellt sie mittels less dar. Dabei ist es unerheblich, in welcher Sektion des Handbuchsystems diese Seite liegt. Da aber die Sektionen der Reihe nach durchsucht werden, wird immer die Handbuchseite der 1. Sektion dargestellt, wenn sie existiert.

Das kann zu Problemen führen, wenn es mehrere Handbuchseiten gleichen Namens gibt. In diesem Fall kann der Parameter `-a` angegeben werden. Jetzt zeigt man alle gefundenen Handbuchseiten dieses Namens hintereinander an.

Um ganz gezielt nach einer Handbuchseite in einer bestimmten Sektion zu suchen, kann aber auch die gewünschte Sektion mit angegeben werden. Ein Beispiel:

Linux kennt sowohl ein Programm mit Namen `passwd` (zum Ändern von Passwörtern), als auch eine Datei `/etc/passwd` (in der die Userinformationen gespeichert sind). Die Handbuchseite für das Programm `passwd` liegt in der Sektion 1 (User Kommandos), die Beschreibung der Datei `passwd` finden wir in der Sektion 5 (Dateiformate).

Wenn wir jetzt die Handbuchseite für `passwd` aufrufen, indem wir schreiben

```
man passwd
```

so bekommen wir zwangsläufig die Seite über das Programm und nicht über die Datei. Wie oben schon erwähnt, werden die Sektionen ja der Reihe nach (1-9) durchsucht und die erste gefundene Seite wird angezeigt. Um also direkt die Handbuchseite der Datei `passwd` anzuzeigen müssen wir die gewünschte Sektion (hier 5) mit angeben. Wir schreiben also

```
man 5 passwd
```

und bekommen so die Handbuchseite, die uns das Format der Datei `/etc/passwd` anzeigt.

Es existieren noch zwei weitere kleine Dienstprogramme, die mit den Handbuchseiten arbeiten, whatis und apropos. Beide bedienen sich ursprünglich einer kleinen Textdatenbank, die jeweils nur den Namen einer Handbuchseite und die Kurzbeschreibung des Befehls enthält. Die sogenannte `whatis-Datenbank`. Nachdem heute das Linux-Handbuchseitensystem sowieso eine Indexdatenbank aufrechterhält, die auch die Namen und Kurzbeschreibungen speichert, arbeiten whatis und apropos heute mit dieser Indexdatenbank. Nur wenn diese Indexdatenbank nicht gefunden wird, wird die alte whatis-Datenbank durchsucht.

Die Kurzbeschreibungen haben immer das Format:

```
Name (Handbuchsektion) - Beschreibung
```

Das Programm whatis wird zusammen mit einem Suchbegriff eingegeben. Es durchsucht jetzt die Kurzbeschreibungen, allerdings nur die linke Spalte dieser Beschreibungen, also die Spalte mit dem Namen. Als Ausgabe werden alle Zeilen dargestellt, die den Suchbegriff in dieser linken Spalte der Kurzbeschreibung haben. Der Programmaufruf

```
whatis passwd
```

könnte also (wenn auch die deutschen Handbuchseiten installiert sind) die folgende Ausgabe haben:

```
passwd (1)           - ändert das Paßwort zum System
passwd (5)           - Paßwort-Datei
passwd (5)           - The password file
passwd (1)           - change user password
```

Wie der Name schon andeutet, zeigt whatis also einfach, was ein Programm oder eine Datei ist. Das hilft uns natürlich nur weiter, wenn wir den Namen des Programms schon kennen. Wenn wir aber nicht mehr wissen, mit welchem Programm ein Passwort geändert wird, dann kann uns das Programm apropos weiterhelfen. Wie bei whatis, so geben wir auch dem Programm apropos einen Suchbegriff mit. Wie whatis durchsucht apropos auch wieder die Indexdatenbank bzw. die whatis-Datenbank. Nur durchsucht apropos nicht nur die linke spalte, sondern die ganze Zeile dieser Kurzbeschreibung. Wir suchen also nach dem Begriff Paßwort und schreiben

```
apropos paßwort
```

und bekommen als Ausgabe alle Zeilen der Indexdatenbank, die diesen Suchbegiff enthalten:

```
getpwuid (3)         - arbeite mit Paßworteintrag
passwd (1)           - ändert das Paßwort zum System
fgetpwent (3)        - arbeite mit Paßworteintrag
endpwent (3)         - arbeite mit Paßworteintrag
chsh (1)             - ändert den Loginshell Eintrag in der Paßwortdatei
getpwnam (3)         - arbeite mit Paßworteintrag
setpwent (3)         - arbeite mit Paßworteintrag
passwd (5)           - Paßwort-Datei
getpwent (3)         - arbeite mit Paßworteintrag
getpw (3)            - Re-konstruiere eine Paßwortdateieintrag
```

Jetzt können wir uns aus dieser Liste das Programm aussuchen, das unseren Ansprüchen gerecht wird...

Die Verwaltungsarbeiten für die Handbuchseiten

Physikalisch liegen die Handbuchseiten in verschiedenen Teilen der Dateisystem-Hierarchie. So kann jede wichtige Unterhierarchie des Dateisystems ein eigenes Handbuchverzeichnis besitzen, das in der Regel dann den Namen **man** trägt. Typische Orte für Handbuchseiten waren früher also:

```
/usr/man
/usr/local/man
/usr/X11R6/man
/usr/openwin/man
/usr/share/man
```

Nach der Version 2 des Linux Dateisystemstandards sollten Handbuchseiten heute grundsätzlich unter `/usr/share/man` liegen. Auch dort können aber wiederum mehrere Verzeichnisse mit Handbuchseiten abgelegt sein, beispielsweise für unterschiedliche Sprachen. Also etwa

```
/usr/share/man
/usr/share/man/de
...
```

Jedes dieser Verzeichnisse enthält wiederum Unterverzeichnisse für jede Sektion des Handbuchs. Enthalten diese Verzeichnisse GROFF-Quelltext (unformatierte Handbuchseiten), so tragen sie die Namen `man1, man2, man3, ...`, enthalten sie aber bereits vorformatierten ASCII-Text, so werden sie `cat1, cat2, cat3, ...` genannt.

Weil die Handbuchseiten praktisch alle im /usr/share-Dateisystem oder darunter zu finden sind, und dieses Dateisystem ja oftmals ReadOnly gemountet ist, wird meist im /var-Verzeichnis noch eine Handbuchhierarchie gehalten, die für die Speicherung der Index-Datenbank, der whatis-Datenbank und der cat-manpages dient. Diese Hierarchie liegt oft unter /var/cache/man oder /var/catman

Damit die Indexdatenbank aktualisiert werden kann (wenn z.B. neue Programmpakete installiert wurden), existiert das Programm mandb. Dieses Programm aktualisiert oder erstellt die Indexdatenbank für alle Handbuchseiten.

Damit dieses Programm tatsächlich alle Verzeichnisse findet, die Handbuchseiten enthalten, bzw. damit auch das man-Kommando die Reihenfolge der Durchsuchung weiß, werden alle Verzeichnisse, die Handbuchseiten enthalten, in einer Variable mit Namen **`MANPATH`** gespeichert. Sie ist aufgabaut wie der Suchpfad, also werden die Namen der Verzeichnisse mit Doppelpunkt voneinander getrennt.

Ein Aufruf von

```
echo $MANPATH
```

zeigt also den Suchpfad für die Handbuchseiten und hauptsächlich dadurch auch die Reihenfolge, in der diese Verzeichnisse durchsucht werden. Das ist wichtig, da ja das man-Kommando beispielsweise nur die erste gefundene Handbuchseite zeigt, wenn nicht die Option `-a` mitgegeben wurde.

Diese Handbuchpfadvariable wird entweder in einer Startdatei wie `/etc/profile` angelegt, oder - seltener - durch ein Programm namens manpath gesetzt. Dieses Programm wird auch dann automatisch von **man** aufgerufen, wenn die Variable **MANPATH** nicht existiert. Das kann in einem System ohne mehrsprachige Handbuchseiten ganz praktisch sein, sobald aber mehrere Sprachen benutzt werden zeigt es Schwächen.

Da das Programm manpath sich den Handbuchpfad einfach dadurch erstellt, daß es alle Verzeichnisse durchscannt und Unterverzeichnisse namens MAN oder man sucht, kann die Reihenfolge der gefundenen Verzeichnisse nicht manipuliert werden. Sind aber z.B. deutsche und englische Handbuchseiten installiert, so wird man es wahrscheinlich vorziehen, dem Pfad der deutschen Seiten vor dem der englischen anzugeben. Das führt dann dazu, daß - wenn eine deutsche Handbuchseite zu einem Programm existiert - die deutsche Seite aufgerufen wird, ansonsten die englische. In diesem Fall bleibt es notwendig, die MANPATH-Variable zu setzen, um diese Reihenfolge festzulegen.

Wenn es gewünscht wird, daß alle (oder bestimmte) Handbuchseiten als ASCII Dateien vorformatiert werden sollen, dann kann diese Aufgabe durch das Programm catman erledigt werden. Dieses Programm legt - meist im /var-Verzeichnis - die entsprechenden cat-Verzeichnisse an und speichert die Handbuchseiten in reinem Textformat lesbar dort hinein. Da heute Festplattenplatz kein großes Kriterium ist, kann man davon durchaus Gebrauch machen, es ersetzt aber nicht die GROFF-Dateien. Denn für alle anderen Handbuchprogramme wie xman, tkman oder rman benötigen wir diese GROFF-Quellen.

Info und seine Seiten

Die Free Software Foundation hat ein weiteres Prinzip ins Leben gerufen, um Informationen über Programmpakete dem User bereitzustellen, das info-System. Dieses System ist gedacht, um größere Informationsmengen als die Handbuchseiten zu transportieren und funktioniert ein bischen ähnlich wie HTML. Es gibt Hypertext-Verweise auf andere Seiten und ein Thema hat oftmals mehrere Dateien, die Informationen darüber bereitstellen. Das Programm, mit dem diese Info-Dokumentationen dargestellt werden können heißt **info**

Dieses Programm ist etwas spartanisch in der Benutzung, trotzdem kann es in vielen Fällen Informationen vermitteln, die wesentlich weitergehen, als die der Handbuchseiten. Nebenbei kann info heute auch Handbuchseiten darstellen, falls also im Verzeichnis der Info-Dateien (meist /usr/info, /usr/share/info) keine Informationen zu finden sind, durchsucht info auch noch den MANPATH und zeigt Handbuchseiten an.

Wird info ohne Parameter aufgerufen, so startet es in einem globalen Menü, sozusagen dem Hauptmenü, in dem dann alle vorhandenen Nodes angezeigt sind. Eine solche Node wird angewählt, indem der Cursor auf den entsprechenden Text plaziert wird und dann die Enter-Taste gedrückt wird.

Wird info mit einem Parameter aufgerufen, so zeigt es gleich die Informationen über diesen Parameter, sozusagen die Hauptseite zum angegebenen Thema. Ein Aufruf von

```
info emacs
```

bringt uns also auf die Hauptseite der Info-Dateien über Emacs. Dort finden wir wiederum Nodes, auf die wir weiterspringen können. Diese Unternodes sind aber auch direkt anwählbar z.B. mit

```
info emacs buffers
```

kommen wir sofort auf die Node Buffers der Datei Emacs des Info-Systems.

Um genauere Informationen über info zu erhalten bietet das info-System eine Art Selbsthilfe, die durch den Aufruf

```
info info
```

gestartet wird. Dort werden dann Schritt für Schritt der Umgang mit info erklärt.

HOWTOs und FAQs

Neben den bisher geschilderten Informationsquellen gibt es natürlich noch weitere Formen, deren Existenz bekannt sein muß. Im Wesentlichen handelt es sich dabei um die sogenannten HOWTOs und FAQs. Im Gegensatz zu den bisher besprochenen Dokumentationen enthalten diese Quellen nicht Online-Hilfen, die als schnelle Nachschlagmöglichkeiten gedacht sind, sondern die HOWTOs bieten fundierte Anleitungen, wie bestimmte Probleme zu lösen sind, wie bestimmte Dienste zu konfigurieren sind oder ähnliches. Die FAQs (Frequently Asked Questions - Häufig gestellte Fragen) dagegen zeigen häufig gestellte Fragen zu bestimmten Themen und natürlich deren Antworten.

Typischerweise finden sich diese Dokumente im Verzeichnis `/usr/doc` oder `/usr/share/doc`. Sie sind aber nicht grundsätzlich installiert, die neuesten Versionen finden sich im Internet. Um zu vermeiden, daß es zu einem unübersichtlichen Chaos bei dieser Form der Dokumentation kommt, werden die HOWTOs und FAQs zentral vom Linux Documentation Project (LDP) verwaltet. Das LDP stellt all seine Dokumentationen unter der Adresse http://www.linuxdoc.org zur Verfügung.

Welche dieser Dokumentationen auf dem System installiert sein sollen, hängt von mehreren Faktoren ab. Neben der Frage des Plattenplatzes stellt sich die der Brauchbarkeit. Wenn ein System keinerlei ISDN-Anbindung hat muß wohl kaum ein ISDN-HOWTO installiert werden. Andererseits schaden diese Dokumente nicht, wenn genug Platz zur Verfügung steht. Die meisten dieser Dokumente stehen inzwischen in einem HTML-Format zur Verfügung, so daß man auch einen zentralen Dokumentationsserver im lokalen Netz aufbauen kann, von dem aus alle Dokumente gelesen werden können.

Andere Dokumentationen

Neben den zentral verwalteten Dokumentationen der HOWTOs und FAQs bringt jedes größere (manchmal auch kleinere) Paket noch Dokumentationen mit. Es hat sich inzwischen durchgesetzt, daß solche Dokumentationen unter `/usr/share/doc/packages/`*`Paketname`* oder `/usr/share/doc/`*`Paketname`* abgelegt werden. Hier sollten sich zumindest eine Datei mit Namen README befinden, die eine erste Beschreibung des Paketes beinhaltet. Oft sind in diesen Verzeichnissen aber ganze Dokumentationen zu finden, die jeden Schritt, von der Installation über die Benutzung bis zur Administration beschreiben.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.108.2

Finden von Linux-Dokumentation im Internet

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Linux-Dokumentation zu finden und zu verwenden. Dieses Lernziel beinhaltet die Verwendung von Linux-Dokumentation bei Quellen wie dem *Linux Documentation Project* (LDP), auf den Webseiten von Distributoren und Drittanbietern, Newsgruppen, Newsgruppen-Archiven und Mailing-Listen.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

Nicht verfügbar

Hier geht es im Wesentlichen darum, die wichtigsten Dokumentationsquellen im Internet zu finden und benutzen zu können. Die Bewertung dieses Abschnitts ist nicht sehr hoch, es gibt jedoch tatsächlich Fragen, die einzelne Newsgroups oder Webseiten zur Auswahl stellen und fragen, in welchen Informationen zu einem bestimmten Thema zu finden sind. Hier finden Sie eine Auflistung wichtiger Informationsquellen in englischer Sprache. Schauen Sie sich um, lesen Sie etwas nach, kurz gewöhnen Sie sich an den Umgang mit solchen Dokumentationen. Es gibt keine wichtigeren Hilfsmittel unter Linux...

Es gibt natürlich auch diverse Newsgroups und Mailinglisten auf deutsch, aber Sie tun gut daran, sich auch bei den englischsprachigen etwas auszukennen, in der LPI101-Prüfung werden sicher keine deutschen nachgefragt...

Webseiten

- www.linuxdoc.org
  Die Seiten des Linux Documentation Projects. Handbuchseiten, Bücher, HOWTOs, FAQs und vieles mehr.
- rute.sourceforge.net
  Ein kompletter Kurs und Nachschlagwerk für Linux.
- www.linux.org/docs/
  Eine weitere Sammlung von Dokumentationen auf der offiziellen Linux Webseite

Newsgruppen

- comp.os.linux.advocacy
  Generelle Diskussion über die Vorteile von Linux gegenüber anderen Betriebssystemen.
- comp.os.linux.announce
  Moderierte Nachrichten über Ankündigungen bezüglich Linux
- comp.os.linux.answers
  Moderierte Versendung von Linux FAQ's. HOWTO's, und README's.
- comp.os.linux.apps
  Generelle Diskussion über Linux Anwendungssoftware.
- comp.os.linux.development.apps
  Diskussion über Anwendungsprogrammierung und -portierung für Linux.
- comp.os.linux.development.system
  Diskussion über den Linux-Kernel, Gerätetreiber und ladbare Module.
- comp.os.linux.hardware
  Generelle Diskussion über Linux Hardware-Kompatibilitätsfragen.
- comp.os.linux.misc
  Vermischte Themen, die in keiner anderen Newsgruppe Platz fanden.
- comp.os.linux.networking
  Generelle Diskussion über Netzwerk- und Kommunikationsfragen.
- comp.os.linux.setup
  Generelle Diskussion über Linux-Installation und Systemverwaltung.
- comp.os.linux.x
  Diskussion zum Thema X Window System unter Linux.
- alt.os.linux
  Generelle Diskussion zum Thema Linux.

Newsgroup-Archive

- www.dejanews.com
  Archiv aller Newsgroups. Ist inzwischen von Google übernommen worden.

Mailinglists

Die folgenden Mailinglisten laufen alle auf einem zentralen Majordimo-Server. Um eine Liste zu abonieren, muß eine Mail an majordomo@vger.kernel.org geschickt werden, die im Text der Mail die Zeile

```
subscribe Listenname
```

stehen hat. Als Listenname muß einer der unten aufgeführten Namen angegeben werden. Kommandos in der Überschrift (im Subjekt) werden nicht bearbeitet.

- linux-8086 - für 8086 Systeme
- linux-admin - Linux Administration
- linux-alpha - Linux auf dem DEC Alpha
- linux-apps - Software Anwendungen
- linux-arm
- linux-bbs - Für Mailbox-Systeme (Bulletin Board Systems)
- linux-c-programming - Progammieren und Entwickeln mit C
- linux-config - System Konfiguration
- linux-console
- linux-diald
- linux-doc - Dokumentationsprojekte
- linux-fido
- linux-fsf
- linux-ftp
- linux-gcc - Wichtige Punkte für die, die unter Linux entwickeln
- linux-gcc-digest - Digest der linux-gcc Liste
- linux-hams - Amateurfunk und Linux

- linux-hppa
- linux-ibcs2
- linux-ipx
- linux-isdn
- linux-japanese - Japanische User
- linux-kernel - Generelle Kerneldiskussion
- linux-kernel-announce - Kernel Ankündigungen
- linux-kernel-digest - Digest der linux-kernel Liste
- linux-kernel-patch
- linux-laptop - Linux auf dem Laptop Diskussion
- linux-linuxss - Linux Mach Single Sever Entwicklung
- linux-lugnuts
- linux-mca
- linux-mips
- linux-msdos
- linux-msdos-digest
- linux-msdow-devel
- linux-net
- linux-new-lists
- linux-newbie - Grundlegende Einführung in Linux
- linux-newbiew
- linux-nys
- linux-oasg
- linux-oi - Gebrauch des Object Interface Toolkit
- linux-opengl
- linux-pkg - Diskussion über Paketmanager
- linux-ppp - PPP Netzwerke unter Linux
- linux-pro - DisKussion über die Linux PRO Distribution
- linux-qag - Linux Quality Assurance Group

- linux-raid
- linux-scsi - Scsi Laufwerke Entwicklung und Gebrauch
- linux-serial - Serielle Geräte unter Linux nutzen
- linux-seyon
- linux-smp - Linux auf Symetrischen Multiprozessor Systemen
- linux-sound - Soundcards und Utilities unter Linux
- linux-standards - Standardisierung verschiedener Aspekte von Linux
- linux-svgalib - SVGA Library Diskussion
- linux-tape - Bandlaufwerke unter Linux nutzen
- linux-term
- linux-training@lists.iphil.net - Linux Training
- linux-userfs
- linux-word
- linux-x11 - Gebrauch des X-Window System unter Linux
- linux-x25
- sparclinux - Linux auf der Sparc
- ultralinux

Auf der Linux Mailing Lists-Seite kann man sich auf all den oben genannten und weiteren Mailinglisten direkt eintragen lassen.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.108.5

Benachrichtigen von Benutzern über systemrelevante Ereignisse

---

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Benutzer über aktuelle Themen bezüglich des Systems zu benachrichtigen. Dieses Lernziel beinhaltet die Automatisierung des Kommunkationsprozesses, z.B. durch Login-Meldungen.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/issue`
- `/etc/issue.net`
- `/etc/motd`

Linux hat von Unix ein paar standardisierte Wege übernommen, um Usern aktuelle Meldungen über das System zukommen zu lassen. Es handelt sich dabei um Textdateien, die zu bestimmten Zeiten auf dem Bildschirm dargestellt werden. In diese Dateien kann der Systemverwalter bestimmte Informationen schreiben, die ein User dann zu den entsprechenden Gelegenheiten zu sehen bekommt.

Die Datei /etc/issue

Der Inhalt dieser Datei wird vor dem Login auf den Bildschirm geschrieben. Konkret ausgedrückt schreibt der Getty-Prozeß, der auf die Eingabe eines Usernamens wartet, den Inhalt dieser Datei auf die Terminalleitung und fordert erst dann zur Eingabe des Usernamens auf.

Normalerweise wird diese Datei dazu benutzt, dem User, der sich einloggen will, Informationen über das System zukommen zu lassen. Typische Einträge in dieser Datei sind einzeilige Meldungen über die verwendete Systemsoftware, etwa

```
  Debian GNU/Linux testing/unstable marvin
```

oder

```
Welcome to SuSE Linux 7.3 (i386) - Kernel 2.4.18
```

Die meisten Gettys sind zusätzlich in der Lage, bestimmte Substitutionen in diese Meldung aufzunehmen, die dann durch reale Werte ersetzt werden. Normalerweise sind das

- **\b** Baudrate der Terminalleitung
- **\d** Aktuelles Datum
- **\s** Systemname (Name des Betriebssystems)
- **\l** Name der TTY-Leitung
- **\m** Architektur des Systems (z.B. i486)
- **\n** Hostname des Rechners
- **\o** Domainname des Rechners
- **\r** Release-Nummer des Betriebssystems
- **\t** Aktuelle Uhrzeit
- **\u** Anzahl der eingeloggten User
- **\U** Anzahl der eingeloggten User und das Wort User(s)
- **\v** Version des Betriebssystems (Build-Date)

Der Inhalt einer `/etc/issue` Datei könnte also auch folgendermaßen aussehen:

```
Willkommen bei \n.\o (\s \m \r)
```

Das würde dann in der Ausgabe etwa folgendermaßen dargestellt werden:

```
Willkommen bei marvin.mydomain.org (Linux i686 2.4.18)
```

Die Angabe von Datum, Uhrzeit und Anzahl der eingeloggten User ist weniger sinnvoll als auf den ersten Blick zu vermuten wäre. Die Substitution der Werte findet statt, während getty die Begrüßungsmeldung auf den Schirm schreibt. Danach findet keine Aktualisierung dieser Werte statt. Bei einem Server, der Tag und Nacht läuft und auf dessen Terminal sich nur alle paar Wochen jemand einloggt, können diese Werte mehrere Tage alt und damit wenig aussagekräftig sein.

Die Datei /etc/issue.net

Speziell für die Verwendung von telnet und rlogin existiert noch die Datei `/etc/issue.net`, die exakt die selbe Funktion wie `/etc/issue` hat, jedoch nur für Logins über eine Netzwerkverbindung gedacht sind. Entweder enthält diese Datei die selbe Information wie `/etc/issue` oder weniger Information, um zu verhindern, daß Angreifer von außen womöglich z.B. aus der Versionsnummer des Kernels heraus Schwachstellen entnehmen könnten, die sie für einen Angriff ausnützen können.

Die Datei /etc/motd

Die Datei `/etc/motd` wird einem User nach dem erfolgreichen Login dargestellt. motd steht für "Message Of The Day" also "Meldung des Tages". In dieser Datei kann der Systemverwalter wichtige Informationen für alle User veröffentlichen und jeder User bekommt diese Datei beim Login angezeigt. Typische solcher Informationen sind Termine, wann ein System zu Wartungszwecken heruntergefahren werden muß oder welche neuen Programme installiert wurden und jetzt zur Verfügung stehen.

Da diese Datei erst nach erfolgreichem Login angezeigt wird, kann sie wesentlich internere Informationen enthalten, als `/etc/issue`. Die User, die diese Datei zu sehen bekommen haben sich ja schon als gültige User mit Namen und Passwort ausgewiesen, sind also weniger als potentielle Angreifer zu betrachten.

1.109.1

Anpassung und Verwendung der Shell-Umgebung

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Shell-Umgebungen auf die Bedürfnisse der Benutzer hin anzupassen. Dieses Lernziel beinhaltet das Setzen von Umgebungsvariablen (z.B. PATH) beim Login oder beim Aufruf einer neuen Shell. Ebenfalls enthalten ist das Schreiben von Bash-Funktionen für oft benutzte Kommandoabfolgen.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `~/.bash_profile`
- `~/.bash_login`
- `~/.profile`
- `~/.bashrc`
- `~/.bash_logout`
- `~/.inputrc`
- **function** (eingebautes Bash-Kommando)
- **export**
- **env**
- **set** (eingebautes Bash-Kommando)
- **unset** (eingebautes Bash-Kommando)

Login-Shell versus NoLogin-Shell

Unix unterscheidet beim Aufruf einer Shell, ob es sich um eine Login-Shell handelt, oder um eine No-Login-Shell. Der Unterschied ist einfach: Die Login-Shell ist die Shell, die beim Login vom login-Programm gestartet wurde. Jede weitere Shell, die von der Login-Shell aufgerufen wird ist eine No-Login-Shell.

Der Grund für diese Unterscheidung liegt in der Frage der Konfigurationsdateien, die später noch genauer dargestellt werden. Zunächst nur so viel: Die Login-Shell muß konfiguriert werden, sie erhält beim Start viele Variablen, die sie an spätere Shells weitervererbt.

Eine Shell, die wiederum von der Loginshell aufgerufen wird hat diese Konfiguration nicht nötig. Durch die Tatsache, daß Variablen an aufgerufene Shells weitergegeben werden können, erledigt sich das von selbst. Außerdem wäre es eine sehr zeitaufwendige Methode, wenn jede Subshell jedesmal alle Konfigurationsdateien abarbeiten müsste.

Die Frage, die sich grundsätzlich stellt ist ja auch die, wozu eine Shell denn überhaupt eine Subshell aufruft. Das kommt sehr viel häufiger vor, als auf den ersten Blick ersichtlich. Jedesmal, wenn die Shell ein Script abarbeitet, wird dazu von ihr eine Subshell aufgerufen. Wenn bei jedem Scriptaufruf dann alle Konfigurationsdateien abgearbeitet werden würden, wäre das eine echte Zeitfrage.

Auch in der graphischen Benutzeroberflache, die ja oft aus einer Shell heraus gestartet wird (startx) erscheinen wieder mehrere xterm-Fenster, in denen jeweils eine Shell läuft. All diese Shells sind keine Login-Shells.

Eine weitere Unterteilung der No-Login-Shells ist die Frage, ob es sich om eine interaktive Shell handelt, oder nicht. Diese Frage ist aber unerheblich, wenn es um die Konfiguration geht.

Die Geltungsbereiche von Shell-Variablen

Die Shellvariablen und Befehle wie **set, unset** und env wurden schon im Abschnitt 1.103.1 der Vorbereitung auf die LPI101 Prüfung besprochen. Hier geht es jetzt um die Frage des Geltungsbereichs von Variablen.

Wenn innerhalb einer Shell eine Variable definiert wird, und danach eine zweite Shell aufgerufen wird, so ist die Variable innerhalb der zweiten Shell nicht automatisch gültig. Das läßt sich einfach ausprobieren, indem die folgenden Befehle eingegeben werden:

```
$ Testvariable=Hallo
$ echo $Testvariable
Hallo
$ bash
$ echo $Testvariable

$ exit
$ echo $Testvariable
Hallo
```

Wir haben also zunächst eine Variable `Testvariable` mit dem Wert `Hallo` angelegt. Im zweiten Schritt geben wir den Inhalt der Variable aus, das funktioniert erwartungsgemäß. Jetzt starten wir eine zweite Shell mit dem Aufruf von **bash**. Der erneute Versuch, sich den Inhalt der Variablen anzusehen schlägt fehl. Erst wenn wir die zweite Shell mit `exit` wieder beenden, funktioniert die

Variablenausgabe wieder.

Damit eine Shell eine Variable an ihre Subshells weitergibt, muß die Variable exportiert werden. Das geschieht mit dem Shellbefehl **export**. Es gibt zwei Formen der Anwendungen, entweder wird eine Variable zuerst definiert und dann exportiert, oder die Variablendefinition geschieht gleich zusammen mit `export`:

```
Variable1=Versuch
export Variable1
export Variable2="Noch ein Versuch"
```

Eine exportierte Variable wird grundsätzlich an alle folgenden Subshells weitergegeben und ist dort verfügbar. Selbst wenn die Subshell wiederum eine Shell aufruft, ist die Variable auch innerhalb dieser dritten Shell gültig.

Wird allerdings die Variable dann in einer Subshell verändert, so bezieht sich diese Veränderung nur auf die Subshell. Innerhalb der aufrufenden Shell behält die Variable ihren alten Wert. Das beweist, daß es sich eben nicht um die selbe Variable handelt, sondern daß eine exportierte Variable in einer Subshell nochmal extra angelegt wird.

```
$ export name=hans
$ echo $name
hans
$ bash
$ echo $name
hans
$ name=otto
$ echo $name
otto
$ exit
$ echo $name
hans
```

Wie schon bei der Darstellung des Unterschiedes zwischen LoginShell und NoLoginShell gezeigt, wird jedesmal eine Subshell aufgerufen, wenn ein Shellscript abgearbeitet werden soll. Wenn in diesem Script Variablen definiert werden, dann gelten diese Variablen natürlich nur innerhalb der Subshell. Die aufrufende Shell bekommt von diesen Variablen überhaupt nichts mit.

Das ist aber manchmal sehr unpraktisch. Wenn wir etwa ein Script schreiben, um verschiedene benötigte Shellvariablen anzulegen, dann hilft uns das herzlich wenig, denn jedesmal wenn wir das Script aufrufen werden diese Variablen zwar in der Subshell, die das Script ausführt erzeugt - aber jedesmal, wenn das Script beendet wurde (und damit auch die Subshell) sind die Variablen nicht mehr gültig.

Damit es trotzdem möglich ist, ein Script zu verwenden um Variablen zu definieren, gibt es die Möglichkeit die Shell dazu zu zwingen, keine Subshell zur Abarbeitung des Scripts aufzurufen. Das geschieht einfach durch das Voranstellen eines Punktes vor dem Scriptaufruf. Selbstverständlich muß zwischen dem Punkt und dem Scriptnamen dabei ein Leerzeichen stehen.

Dadurch wird die Shell gezwungen, das Script selbst abzuarbeiten. Die Variablen, die innerhalb dieses Scripts definiert wurden, sind jetzt auch nach der Abarbeitung des Scriptes noch gültig.

Das klingt schön, hat aber einen bedeutenden Haken. Wenn innerhalb des Scrips ein exit vorkommt, normalerweise dazu benutzt, um das Script zu beenden, wird ja die Shell, die das Script ausführt, beendet. Wurde das Script jetzt mit vorgestelltem Punkt aufgerufen, also von unserer Loginshell selbst, dann wird das exit die LoginShell beenden! Wir müßten uns also erneut einloggen.

Scripts, die mit führendem Punkt aufgerufen werden sollen, sollten daher auf keinen Fall - auch nicht zur Fehlerbehandlung - ein exit benutzen.

Alias und Funktion in der interaktiven Shell

Neben den Variablen gibt es noch zwei weitere Formen, die innerhalb einer Shell definiert werden können. Das Alias und die Funktion.

Alias

Ein Alias ist sozusagen ein Mechanismus, der der Shell klar macht, daß ein bestimmter Name eine bestimmte Bedeutung hat. Jedesmal, wenn die Shell auf einen Befehl trifft, überprüft sie zuerst, ob es sich dabei um einen Alias handelt, erst wenn es klar ist, daß es kein Alias ist, wird der Unix-Befehl gesucht. Das heißt, daß damit die Bedeutung von Unix-Befehlen überlagert werden kann. Ein Alias wird nur ein einziges Mal aufgelöst, so daß es möglich ist, bestehende Unix-Befehle umzudefinieren.

Beispiele:

```
alias cp=cp -i
```

Damit wird der Befehl cp grundsätzlich als cp -i ausgeführt, das heißt, er frägt vor dem Überschreiben einer Zieldatei nach, ob das in Ordnung ist.

```
alias ...=cd ..;cd..
```

Der Befehl `...` ist ein Alias auf den Befehl `cd ..;cd ..`, also zweimal ins nächsthöhere Verzeichnis zu wechseln.

```
alias werbinich="echo $LOGNAME \($UID\)"
```

Der Befehl `werbinich` gibt in einer Zeile den Usernamen und die UserID des Users aus, der ihn ausführt.

Um Aliase wieder loszuwerden gibt es den Befehl **unalias** mit dem ein Alias wieder gelöscht werden kann.

Zur Gültigkeit von Aliasen gilt genau das selbe, wie zur Gültigkeit von Shellvariablen. Sie haben nur Gültigkeit in der Shell, in der sie definiert wurden. Im Gegensatz zu Variablen lassen sich Aliase aber nicht exportieren - zumindestens nicht bei modernen Shells. Wenn ein Alias auch innerhalb einer Subshell gelten soll, so muß er in einer Konfigurationsdatei definiert werden, die auch von einer No-Login-Shell abgearbeitet wird, also etwa `~/.bashrc`.

Funktionen

Der Alias-Mechanismus erlaubt eine vielseitige Anwendung auch kombinierter Befehle, die unter einem neuen Namen zusammengefasst werden. Er hat aber zwei wesentliche Einschränkungen. Innerhalb des Alias ist es nicht möglich, auf einzelne Parameter des Aliases zuzugreifen, die ihm mitgegeben wurden. Es ist ja tatsächlich ein Textersatz, der hier vorgenommen wird.

Der zweite Nachteil des Aliases ist, daß seine Interpretation nur einmal stattfindet, dadurch kann er nicht rekursiv aufgerufen werden.

Beide Nachteile (die in Wahrheit keine Nachteile sind, sondern nur Unterschiede zur Funktion) werden von den Shellfunktionen abgeschafft. Es handelt sich hier um einen echten Funktionsaufruf, der die mitgegebenen Parameter bearbeiten kann und der rekursiv laufen kann, das heißt, die Funktion kann sich selber aufrufen.

Shellfunktionen sind kleine Unterprogramme, die einen eigenen Namen haben. Sie können sowohl im Script, als auch in der interaktiven Shell verwendet werden.

Die prinzipielle Aufgabe von Funktionen in Programmiersprachen ist, immer wiederkehrende Aufgaben oder logisch zusammenhängende Teile eines Programms zu separieren und als extra Funktion zu lösen. Dabei können einer Funktion, genauso wie dem Script selbst, Parameter übergeben werden; dazu kann eine Funktion einen eigenen (numerischen) Rückgabewert liefern.

Der prinzipielle Aufbau einer Shellfunktion ist:

```
function Funktionsname()
{
  Kommando1
  ...
}
```

Das Schlüsselwort `function` kann auch weggelassen werden, weil die Shell eine Funktion an den Klammern () am Ende des Funktionsnamens erkennt.

Parameter, die einer Funktion mitgegeben werden, werden nicht in den Klammern im Funktionskopf vordefiniert, wie in anderen

Programmiersprachen. Innerhalb der Funktion gelten für die übergebenen Parameter die gleichen Regeln, wie beim Script selbst. `$1` bezeichnet den ersten übergebenen Parameter, `$2` den zweiten usw. Auch `$*, $@` und `$#` beziehen sich auf die Parameter der Funktion und nicht mehr auf die des Scripts. Allein der Parameter `$0` behält den Namen des Scripts und nicht den der Funktion.

Genauer gesagt gelten all die genannten Variablen (`$1 - $9, $#, $*, $@`) innerhalb einer Funktion als lokale Parameter.

Funktionen können direkt eingegeben werden, das ist aber eher selten. Meist werden sie in den Konfigurationsdateien erstellt und exportiert. Sie müssen wie Variablen und Aliase exportiert werden, damit sie auch in späteren Shells gültig sind.

Wollen wir z.B. eine Funktion schreiben, die beim Verzeichniswechsel gleich den Inhalt des Verzeichnisses anzeigt, in das gewechselt wurde, so können wir das folgendermaßen erledigen:

```
  function lscd()
  {
    cd $*
    ls
  }
```

In einem Alias wäre das nicht möglich gewesen, weil wir ja den Aufrufparameter ($*) nicht gehabt hätten. Was aber, wenn wir diese Funktion nicht `lscd` sondern einfach nur `cd` genannt hätten?

Das hätte ziemlich schnell dazu geführt, daß der Computer keinen Arbeitsspeicher mehr übrig gehabt hätte. Die Funktion hätte sich nämlich permanent selbst aufgerufen. Ohne eine vernünftige Abbruchbedingung hätte das dazu geführt, daß es zu einer Endlosschleife kommt, die in jedem Durchlauf Speicher anfordert. Es wäre also ein sogenannter rekursiver Aufruf gewesen.

Damit zumindestens interne Befehle der Shell wie `cd` überlagert werden können, bietet die Shell das reservierte Wort `builtin` an, das anzeigt, daß in jedem Fall der interne Befehl und nicht die Funktion aufgerufen werden soll. Um unser Beispiel also arbeitsfähig zu machen müßten wir schreiben:

```
function cd()
{
  builtin cd $*
  ls
}
```

Die Konfigurationsdateien der bash

Bei den Konfigurationsdateien für die Shell handelt es sich um einfache Shellscripts, deren Ausführungsrecht nicht gesetzt sein muß. Die Shell führt diese Scripts selbst aus, ohne eine Subshell aufzurufen. Die Variablen, Aliase und Funktionen, die in diesen Scripts definiert wurden, gelten also innerhalb der aufrufenden Shell. In der Regel werden diese Variablen, Aliase und Funktionen exportiert, damit sie auch in kommenden Subshells gültig sind.

Natürlich können auch andere Programme aus den Konfigurationsscripts heraus gestartet werden, typisch ist z.B. der Aufruf von umask, um festzulegen, mit welchen Zugriffsrechten Dateien versehen werden sollen, die neu angelegt werden. Auch Tastaturdefinitionen o.ä. können hier eingestellt werden.

Grundsätzlich unterscheidet die Shell bei Konfigurationsdateien, zwischen Login-Shells und Nicht-Login-Shells. Nur bei Login-Shells werden alle Konfigurationsdateien abgearbeitet. Bei allen anderen Shells wird nur optional eine Datei abgearbeitet, die nicht von der Login-Shell benutzt wird. Konkret werden die folgenden Dateien in der angegebenen Reihenfolge und Abhängigkeit von der Login-Shell abgearbeitet:

Wenn die Datei `/etc/profile` existiert, wird sie abgearbeitet

Wenn im Heimatverzeichnis des Users die Datei `.bash_profile` existiert wird sie abgearbeitet.

Wenn diese Datei nicht existiert wird die Datei `.bash_login` im Heimatverzeichnis des Users gesucht und falls gefunden, abgearbeitet.

Wenn auch diese Datei nicht existiert, so wird im Heimatverzeichnis des Users die Datei `.profile` gesucht und falls gefunden abgearbeitet.

Diese Konstruktion ermöglicht es, sowohl die systemweiten Einstellungen vorzunehmen, als auch jedem User die Freiheit zu lassen, selbst Einstellungen vorzunehmen. Die `/etc/profile`-Datei kann nur vom Systemverwalter verändert werden, alle anderen Dateien befinden sich ja in den Heimatverzeichnissen der einzelnen User. Sie sind meist frei editierbar.

Eine einzige Datei wird von der Nicht-Login-Shell abgearbeitet, falls sie existiert - `.bashrc` im Heimatverzeichnis jedes Users.

Es gibt auch noch eine Datei, die beim Logout - also dem Beenden der LoginShell - abgearbeitet wird, sofern sie existiert. Sie heißt `.bash_logout` und befindet sich auch im Heimatverzeichnis der einzelnen User. Damit ist es z.B. möglich, daß temporäre Dateien gelöscht werden oder ähnliche Aufräumarbeiten erledigt werden.

Die Bourne Again Shell benutzt eine Library, die für den Umgang mit Eingabezeilen gemacht ist. Diese Library heiß Readline und ist ihrerseits konfigurierbar. Ihre Einstellungen nimmt sie in der Datei `.inputrc` im Homeverzeichnis jedes Users vor. In dieser Datei können Tastenbindungen vorgenommen werden, um die Standard-Tastenbelegung für die Eingabezeile zu verändern.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.109.2

Anpassen und Schreiben einfacher Scripts

Beschreibung: Prüfungskandidaten sollten in der Lage sein, existierende Scripts anzupassen und einfache neue (ba)sh Scripts zu schreiben. Dieses Lernziel beinhaltet die Verwendung der Standard sh Syntax (Schleifen, Tests), das Verwenden von Kommandosubstitution, das Prüfen von Kommando-Rückgabewerten, Testen des Status einer Datei und Schicken von Mails an den Superuser unter bestimmten Bedingungen. Ebenfalls enthalten ist die Vorsorge, daß der korrekte Interpreter auf der ersten Zeile (#!) von Scripts aufgerufen wird. Weiters enthalten ist die Verwaltung von Speicherort, Eigentum und Ausführungs- und suid-Rechten von Scripts.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **while**
- **for**
- **test**
- **chmod**

Das Programmieren der Shell

Der Shell ist es prinzipiell egal, ob sie ihre Anweisungen interaktiv über die Tastatur bekommt, oder ob sie diese Angaben aus einer Datei herauslesen soll. Es ist also möglich, mehrere Befehlszeilen in eine Datei zu schreiben, die dann von der Shell ausgeführt werden, als ob sie nacheinander eingegeben wurden.

Dabei gehen die Möglichkeiten der Shell aber weit über die Batch-Programmierung von DOS hinaus. Sie bietet echte Schleifen, vernünftige Ein/Ausgabe, flexible Variablen, Bedingungsüberprüfung und vieles mehr. Es ist sogar möglich, Funktionen zu nutzen und so genauso strukturiert zu arbeiten, wie es in modernen Hochsprachen üblich ist.

Shellscripts werden in der Systemverwaltung sehr gerne und häufig eingesetzt, weil sie sehr schnell und effizient Lösungen für häufig vorkommende Probleme ermöglichen. Das Programmieren der Shell ist aber ebn eine richtige kleine Programmiersprache, es würde den Rahmen des Kurses sicher sprengen, wenn das hier ein kompletter Programmierkurs wäre. Im Folgenden werden alle wichtigen Elemente

der Shellscripts beschrieben, die sichere Anwendung kommt aber sicher nur mit viel Praxis...

Das grundlegende Prinzip der Shellscripts

Prinzipiell ist ein Shellscript nichts anderes als eine Textdatei, die einzelne Befehlszeilen enthält, die von der Shell nacheinander abgearbeitet werden. Jede Zeile, die nicht mit einem Doppelkreuz (#) beginnt, wird von der Shell als Befehlszeile interpretiert und ausgeführt. Zeilen, die mit dem Doppelkreuz beginnen werden als Kommentarzeilen bewertet und nicht ausgeführt. Die einfachste Form wäre also, eine Textdatei zu erstellen, die ein paar Unix-Befehle enthält, diese Datei unter einem bestimmten Namen zu speichern z.B. script1 und dann eine Shell aufzurufen und ihr diesen Namen als Parameter mitzugeben also etwa

```
bash script1
```

Das ist aber natürlich nicht die eleganteste Lösung, schöner wäre es ja, das Script direkt wie ein Programm aufrufen zu können. Dazu müssen wir dem Script einfach nur das Ausführungsrecht geben.

```
chmod +x script1
```

Wenn wir uns jetzt das Script mit **ls -l** ansehen, so hat es jetzt neben den bisherigen Zugriffsrechten auch das x-Recht. Es ist jetzt durch die Nennung seines Namens aufrufbar. (Näheres zu **chmod** auf der entsprechenden Handbuchseite und im Abschnitt 1.104.5 der Vorbereitung auf die LPI101 Prüfung.)

Das geht solange gut, solange wir immer mit der selben Shell arbeiten oder das Script keine Elemente enthält, die eine andere Shell nicht verstehen würde. Aber sobald wir in unserem Script ein paar bash-Spezialitäten einbauen würden und dann plötzlich mit der C-Shell arbeiten würden fielen wir auf die Nase.

Das Problem ist also, daß unser Script selbst noch keine Angaben enthält, von welcher Shell es ausgeführt werden soll. Aber keine Angst, auch hierfür gibt es eine einfache Lösung. Jede Shell unter jedem Unix versteht eine ganz spezielle Kommentarzeile.

Wenn ein Script in der ersten Zeile die Anweisung

```
#!Programm
```

aufweist, so startet jede Shell das *Progamm* und gibt ihm als Parameter den Namen des Scripts mit. Wenn wir also in Zukunft grundsätzlich als erste Zeile unserer Scripts schreiben:

```
#!/bin/bash
```

sind wir auf der sicheren Seite. Selbst wenn wir mit der C-Shell oder einem ganz anderen Kommandointerpreter arbeiten wird für die

Abarbeitung unseres Scripts jetzt grundsätzlich die bash benutzt.

Ein häufiger Fehler ist es, daß ein Schreibfehler in diese erste Zeile gemacht wird. Wenn dann das Script ausgeführt werden soll, dann findet die Shell den Interpreter nicht, da der ja falsch geschrieben ist. Also bringt sie eine etwas mißverständliche Fehlermeldung:

```
bash: Scriptname: No such file or directory
```

Das könnte man jetzt mißverstehen und denken, die Shell hätte das Script nicht gefunden. In Wahrheit hat sie den Interpreter nicht gefunden...

Kommandozeilenparameter

Was wäre ein Shellscript, wenn wir ihm keine Parameter übergeben könnten? Natürlich bietet die Shell diese Möglichkeit, noch dazu auf eine sehr einfache Art und Weise.

Zunächst noch einmal die Erinnerung, was sind Kommandozeilenparameter? Wenn wir ein Script schreiben, das z.B. `addiere` heißt und das zwei Zahlen addieren soll, dann werden wir vom Anwender des Scripts erwarten, daß er die beiden zu addierenden Zahlen als Parameter übergibt, daß er also z.B. schreibt:

```
addiere 10 20
```

In diesem Fall wäre der Kommandozeilenparameter Nummer 1 also die 10, der Parameter2 die 20. Innerhalb der Shell können diese Parameter angesprochen werden mit `$1` und `$2`. Unser Script könnte also folgendermaßen aussehen:

```
#!/bin/bash
let ergebnis=$1+$2
echo $1 plus $2 ergibt $ergebnis
```

Spezielle Variablen für die Kommandozeilenparameter

Grundsätzlich unterstützt die Bourne-Shell bis zu neun Parameter (`$1 - $9`), die direkt angesprochen werden können. Die Bourne-Again-Shell (bash) kann bis zu 99 Parameter direkt ansprechen, indem die Nummern der Parametr oberhalb des neunten in geschweifte Klammern gesetzt werden (`${10} - ${99}`). Der Parameter `$0` enthält, wie in allen anderen Hochsprachen auch, den Namen des Scripts, wie es aufgerufen wurde:

Script	Parameter1	Parameter2	Parameter3	Parameter4	...
$0	$1	$2	$3	$4	...

Oft ist es gar nicht notwendig, die Parameter einzeln anzusprechen. Wenn wir z.B. einen Unix-Befehl umbenennen wollten, z.B. cp in kopiere, dann wäre es ja lästig, wenn wir innerhalb des Scripts die ganzen denkbaren Parameter einzeln angeben müssten. Dazu gibt es die Sonderform $@ und $*, die beide alle angegebenen Parameter bezeichnen. In unserem Script müßten also nur die Zeilen stehen:

```
#!/bin/bash
cp $*
```

Egal, wieviele Parameter jetzt angegeben wurden, alle werden einfach durch das $* übermittelt.

Sehr häufig ist es notwendig zu erfahren, wieviele Parameter überhaupt angegeben wurden. Dazu dient die spezielle Variable $#.

Zusammenfassend existieren also folgende spezielle Variablen für die Kommandozeilenparameter:

${n}	Der nte Parameter bei mehr als 9 Parametern (nur bash)
$@	Steht für alle angegebenen Parameter
$*	Steht für alle angegebenen Parameter
$#	Steht für die Anzahl aller angegebenen Parameter

Der Unterschied zwischen $* und $@ ist marginal und wird sehr selten gebraucht. Er bezieht sich nur auf die Frage, wie die Shell reagiert, wenn $* oder $@ in doppelten Anführungszeichen stehen.

Das	wird ersetzt durch
"$*"	"$1 $2 $3 ..."
"$@"	"$1" "$2" "$3" "..."

Der Befehl shift

Der Befehl shift verschiebt die ganze Kette der Kommandozeilenparameter um eines nach links. Das bedeutet, der Parameter 2 wird zum Parameter1, der Parameter3 zum Parameter2 usw. Der erste Parameter fällt weg. Damit kann eine unbestimmte Anzahl von Parametern bearbeitet werden, indem in einer Schleife immer nur der erste Parameter verarbeitet wird und anschließend der Befehl shift aufgerufen wird. Die Schleife wird solange wiederholt, bis keine Parameter mehr übrig sind.

Auch wenn die Schleifenkonstruktion noch unbekannt ist folgt hier ein Beispiel. Das folgende Script gibt alle eingegebenen Parameter untereinander aus:

```
#!/bin/bash
while [ $# -gt 0 ]   #Solange die Anzahl der Parameter ($#) größer 0
```

```
do
  echo $1            #Ausgabe des ersten Parameters
  shift              #Parameter verschieben $2->$1, $3->$2, $4->$3,...
done
```

Der Befehl set

Normale Variablen bekommen ihre Werte durch das Gleichheitszeichen. Die Konstruktion

Variablenname=Wert

ist aber für Kommandozeilenparameter nicht möglich. Es ist also verboten zu schreiben

```
1=Hans
```

um damit `$1` den Wert Hans zu geben. Falls in seltenen Fällen es doch notwendig sein sollte, die Kommandozeilenparameter zu ändern, so kann das mit dem Befehl **set** erledigt werden. Allerdings können nur alle Parameter gleichzeitig verändert werden. **set** schreibt einfach die gesamte Parameterkette neu.

Das heißt, alle Parameter, die dem Befehl **set** übergeben werden, werden so behandelt, als wären sie dem Script übergeben werden. Die echten Parameter, die dem Script wirklich übergeben wurden fallen dadurch alle weg, auch wenn **set** weniger Parameter erhält, als dem Script mitgegeben wurden.

Diese Anwendung von **set** macht zum Beispiel Sinn, wenn wir ein Script schreiben wollen, das zwingend zwei Kommandoparameter braucht. Wenn wir am Anfang die Anzahl der Parameter überprüfen und merken, daß das Script keine Parameter bekommen hat, so können wir mit **set** voreingestellte Parameter definieren.

Bedingungsüberprüfungen

Eine der grundlegendsten Dinge beim Programmieren ist die Überprüfung von verschiedenen Bedingungen, um damit auf bestimmte Gegebenheiten zu reagieren. Die Überprüfung von Bedingungen hat in der Regel bei Programmiersprachen zwei verschiedene Formen, die if-Anweisung und eine Mehrfachauswahl. Beide werden von der Shell unterstützt.

Die if-Anweisung

Die einfache if-Anweisung

Grundsätzlich hat die if-Anweisung der Bourne-Shell eine sehr einfache Form. Nach dem if steht ein Befehl, der ausgeführt wird. Gibt dieses Kommando eine 0 als Rückgabewert zurück, so gilt die Bedingung als erfüllt und die Aktionen, die zwischen dem folgenden then

und fi stehen werden ausgeführt.

```
if Kommando
then
  Aktion
  Aktion
  ...
fi
```

Natürlich sind die Aktionen auch wieder normale Unix-Befehle. Das "fi", das den Block beendet, der durch "if ... then" begonnen wurde, ist einfach nur das "if" rückwärts geschrieben.

Das Programm test

Damit es jetzt sinnvolle Möglichkeiten gibt, Bedingungen zu überprüfen brauchen wir ein Programm, das verschiedene Tests durchführt und jeweils bei gelungenem Test eine 0 als Rückgabewert zurückgibt, bei mislingenem Test eine 1. Dieses Programm heißt **test** und ermöglicht alle wesentlichen Bedingungsüberprüfungen, die für das Shell-Programmieren notwendig sind.

Damit wir nicht jedesmal schreiben müssen

```
if test ...
```

gibt es einen symbolischen Link auf das Programm **test**, der einfach **[** heißt. Allerdings verlangt das Programm **test**, wenn es merkt, daß es als **[** aufgerufen wurde, auch als letzten Parameter eine eckige Klammer zu. Damit ist es also möglich zu schreiben:

```
if [ ... ]
```

Wichtig ist hierbei, daß unbedingt ein Leerzeichen zwischen if und der Klammer und zwischen der Klammer und den eigentlichen Tests stehen muß. Es handelt sich bei der Klammer ja tatsächlich um einen Programmaufruf!

Die verschiedenen Bedingungsüberprüfungen mit test bzw. [

-r *Dateiname*

: Die Datei *Dateiname* existiert und ist lesbar

-w *Dateiname*

: Die Datei *Dateiname* existiert und ist beschreibbar

-x *Dateiname*

Die Datei *Dateiname* existiert und ist ausführbar

-d *Dateiname*

Die Datei *Dateiname* existiert und ist ein Verzeichnis

-s *Dateiname*

Die Datei *Dateiname* existiert und ist nicht leer

-b *Dateiname*

Die Datei *Dateiname* existiert und ist ein blockorientiertes Gerät

-c *Dateiname*

Die Datei *Dateiname* existiert und ist ein zeichenorientiertes Gerät

-g *Dateiname*

Die Datei *Dateiname* existiert und das SGID-Bit ist gesetzt

-k *Dateiname*

Die Datei *Dateiname* existiert und das Sticky-Bit ist gesetzt

-u *Dateiname*

Die Datei *Dateiname* existiert und das SUID-Bit ist gesetzt

-p *Dateiname*

Die Datei *Dateiname* existiert und ist ein Named Pipe

-e *Dateiname*

Die Datei *Dateiname* existiert

-f *Dateiname*

Die Datei *Dateiname* existiert und ist eine reguläre Datei

-L *Dateiname*

Die Datei *Dateiname* existiert und ist ein symbolischer Link

-S *Dateiname*

Die Datei *Dateiname* existiert und ist ein Socket

-O *Dateiname*

Die Datei *Dateiname* existiert und ist Eigentum des Anwenders, unter dessen UID das test-Programm gerade läuft

-G *Dateiname*

Die Datei *Dateiname* existiert und gehört zu der Gruppe, zu der der User gehört, unter dessen UID das test-Programm gerade läuft

Datei1* -nt *Datei2

Datei1 ist neuer als *Datei2* (newer than)

Datei1* -ot *Datei2

Datei1 ist älter als *Datei2* (older than)

Datei1* -ef *Datei2

Datei1 und *Datei2* benutzen die gleiche I-Node (equal file)

-z *Zeichenkette*

Wahr wenn *Zeichenkette* eine Länge von Null hat.

-n *Zeichenkette*

Wahr wenn *Zeichenkette* eine Länge von größer als Null hat.

Zeichenkette1* = *Zeichenkette2

Wahr wenn *Zeichenkette1* gleich *Zeichenkette2*

Zeichenkette1* != *Zeichenkette2

Wahr wenn *Zeichenkette1* ungleich *Zeichenkette2*

Wert1* -eq *Wert2

Wahr, wenn *Wert1* gleich *Wert2* (equal)

Wert1* -ne *Wert2

Wahr, wenn *Wert1* ungleich *Wert2* (not equal)

Wert1* -gt *Wert2

Wahr, wenn *Wert1* größer *Wert2* (greater than)

Wert1* -ge *Wert2

Wahr, wenn *Wert1* größer oder gleich *Wert2* (greater or equal)

Wert1* -lt *Wert2

Wahr, wenn *Wert1* kleiner *Wert2* (less than)

Wert1* -le *Wert2

Wahr, wenn *Wert1* kleiner oder gleich *Wert2* (less or equal)

!*Ausdruck*

Logische Verneinung von *Ausdruck*

Ausdruck* -a *Ausdruck

Logisches UND. Wahr, wenn beide Ausdrücke wahr sind

Ausdruck* -o *Ausdruck

Logisches ODER. Wahr wenn mindestens einer der beiden Ausdrücke wahr ist

Mit diesen Tests sind so ziemlich alle denkbaren Bedingungsüberprüfungen möglich, die in einem Shellscript notwendig sind.

Die erweiterte if-else Anweisung

Natürlich bietet die if-Anweisung auch eine Erweiterung zur normalen Form, die sogenannte if-else Anweisung. Es ist also möglich zu schreiben:

```
if [ Ausdruck ]
then
  Kommandos
else
  Kommandos
fi
```

Die if-elif-else Anweisung

Um noch einen Schritt weiterzugehen bietet die if-Anweisung sogar ein weiteres if im else, das sogenannte elif, das wieder eine Bedingung überprüft:

```
if [ Ausdruck ]
then
  Kommandos
elif [ Ausdruck ]
then
  Kommandos
else
  Kommandos
fi
```

Mehrfachauswahl mit case

Oft kommt es vor, daß eine Variable ausgewertet werden muß und es dabei viele verschiedenen Möglichkeiten gibt, welche Werte diese Variable annehmen kann. Natürlich wäre es mit einer langen if-elif-elif-elif... Anweisung möglich, so etwas zu realisieren, das wäre aber sowohl umständlich, als auch schwer zu lesen. Damit solche Fälle einfacher realisiert werden können, gibt es die Mehrfachauswahl mit case. Der prinzipielle Aufbau einer case-Entscheidung sieht folgendermaßen aus:

```
case Variable in
  Muster1) Kommando1 ;;
  Muster2) Kommando2 ;;
  Muster3) Kommando3 ;;
  ...
esac
```

Zu beachten sind zunächst die Klammern, die das Muster abschließen. Das Kommando, das zum jeweiligen Muster passt muß mit zwei Strichpunkten abgeschlossen werden. Statt einem Kommando können nämlich auch mehrere Kommandos, durch Strichpunkt getrennt stehen. Nur die doppelten Strichpunkte machen der Shell klar, daß das Kommando für den bestimmten Fall jetzt fertig ist.

Der Abschluß mit esac ist wieder einfach das Wort case rückwärts geschrieben.

Programmschleifen im Script

Programmierung, insbesondere die heute übliche Form der strukturierten Programmierung ist ohne Schleifen nicht möglich. Unter Programmschleifen versteht man eine Wiederholung eines Teils des Programms, bis eine bestimmte Bedingung erfüllt ist.

Die Shell bietet zwei Formen der Schleifen. die Kopfgesteuerte Schleife und die for-Schleife, die eine Liste abarbeitet.

Die kopfgesteuerte Schleife mit while

Die kopfgesteuerte Schleife überprüft vor jedem Schleifendurchgang die Bedingung, die festlegt, ob die Schleife tatsächlich nochmal durchlaufen werden soll. Im Extremfall wird diese Schleife also kein einziges Mal durchlaufen, wenn nämlich die Bedingung schon von vorneherein nicht wahr ist.

Als Bedingung wird wieder jedes Programm akzeptiert, das einen Rückgabewert liefert. Ein Rückgabewert von 0 bedeutet, die Bedingung ist wahr, jeder andere bedeutet unwahr. Wie bei der if-Anweisung wird hier meistens wieder das Programm **test** oder eben dessen abgewandelte Form **[** benutzt. Die verschiedenen Überprüfungen wurden bei der if-Anweisung detailiert dargestellt.

Die prinzipielle Form der while-Schleife mit dem [-Programm als Bedingungsüberprüfung sieht dann so aus:

```
while [ Ausdruck ]
```

```
do
  Kommandos...
done
```

Die Listenschleife mit for

Eine Listenschleife durchläuft die Schleife so oft, wie die Liste Elemente hat. Bei jedem Schleifendurchlauf bekommt die Variable den Wert des jeweiligen Listenelements.

```
for Variable in Liste
do
  Kommandos...
done
```

Als Liste gilt dabei jede mit Leerzeichen, Tabs oder Zeilentrennern getrennte Liste von Worten. Damit diese Funktionsweise etwas klarer wird ein einfaches Beispiel:

```
#!/bin/bash
for i in Hans Peter Otto
do
  echo $i
done
```

Diese Schleife würde also dreimal durchlaufen. Im ersten Durchgang bekommt die Variable i den Wert "Hans", im zweiten "Peter" und im dritten "Otto". Die Ausgabe der Schleife wäre also einfach

```
Hans
Peter
Otto
```

Richtig interessant wird diese Schleife jedoch, wenn als Liste ein Jokerzeichenkonstrukt steht wie etwa `*.txt` - Die Shell löst dieses Konstrukt ja in eine Liste aller Dateien im aktuellen Verzeichnis auf, die auf das Muster passen. Die Schleife wird also sooft durchlaufen, wie es Dateien gibt, die die Endung `.txt` vorweisen...

Eine andere häufige Form der Anwendung ist die Abarbeitung aller Kommandozeilenparameter eines Shellscripts. Die Variable $@ bietet ja diese Parameter alle zusammen. Diese Schleife wird so oft benutzt, daß es dafür eine Sonderform gibt, statt zu schreiben:

```
for name in $@
do
  ...
done
```

reicht es zu schreiben

```
for name
do
  ...
done
```

Eine große Stärke der for-Schleife ist es, als Liste die Ergebnisse eines Unix-Befehls einzugeben. Mit Hilfe der Kommandosubstitution ist es ohne weiteres möglich, die komplexesten Unix-Befehle einzugeben, die als Ergebnis eine Liste ausgeben und diese Liste dann in der Schleife zu verarbeiten.

So kann man beispielsweise eine Liste aller User, die dem System bekannt sind dadurch erhalten, daß man mit dem Befehl **cut** die erste Spalte der Datei `/etc/passwd` ausschneidet. Der Befehl würde folgendermaßen aussehen:

```
cut -d: -f1 /etc/passwd
```

Um die Liste, die dieser Befehl ausgibt, als Liste für die for-Schleife zu nutzen muß der Befehl ja nur in Grave-Zeichen gesetzt werden, also

```
#!/bin/bash
for i in `cut -d: -f1 /etc/passwd`
do
  echo $i
done
```

Shellfunktionen

Shellfunktionen sind kleine Unterprogramme, die einen eigenen Namen haben. Sie können sowohl im Script, als auch in der interaktiven Shell verwendet werden. Hier ist natürlich die Funktionsweise innerhalb eines Scripts Objekt der Darstellung.

Die Syntax von Funktionen wurden bereits im letzten Kapitel beschrieben, hir nur noch ein paar Anwendungsbeispiele im Script.

Als (zweifellos sinnloses) Beispiel folgt hier ein kleines Script, das eine Funktion enthält, die von einem bestimmten Startwert zu einem

bestimmten Endwert zählt. Start- und Endwert werden im Hauptprogramm erfragt, die eigentliche Aufgabe des Zählens erledigt die Funktion:

```
#!/bin/bash

function zaehle_von_bis()
{
  i=$1                          # i bekommt den Wert des ersten Parameters
  while [ $i -le $2 ]           # Solange i kleiner/gleich Parameter2
  do
    echo $i                     # Ausgabe der Zahl i
    let i=$i+1                  # i wird um 1 erhöht
  done
}

# Das Hauptprogramm

read -p "Startwert: " zahl1   # Startwert einlesen
read -p "Endwert: " zahl2     # Endwert einlesen
zaehle_von_bis $zahl1 $zahl2  # Aufruf der Funktion mit den gelesenen Werten
```

Rückgabewerte überprüfen

Wenn innerhalb eines Scripts ein Befehl aufgerufen wurde, so kann es für den weiteren Ablauf sehr wichtig sein, ob der Befehl erfolgreich war oder nicht. Diese Frage beantworten uns die Rückgabewerte (exit-codes). Jedes Programm gibt dem Betriebssystem einen ganzzahligen Rückgabewert zurück, wenn es abgeschlossen ist. Ist dieser Wert 0, dann bedeutet es eine fehlerfreie Beendigung des Programms, in allen anderen Fällen bedeutet es einen Fehler.

Das Programm **test** beispielsweise nutzt diese Fähigkeit, um bestimmte Tests durchzuführen und die Ergebnisse als Rückgabewert (0 bedeutet Test war erfolgreich also wahr) zurückzuliefern.

Die Shell kennt drei Methoden, diesen Rückgabewert zu analysieren:

Die Variable $?

Die Variable `$?` enthält immer den Rückgabewert des letzten ausgeführten Programms. Sie kann nach dem Aufruf eines Programms z.B. mit der if-Bedingungsüberprüfung abgefragt werden:

```
...
Programm
if [ $? -eq 0 ]
then
  ...       # Programm war erfolgreich
fi
```

Sobald ein anderes Programm abgelaufen ist, hat die Variable `$?` aber bereits einen anderen Wert, den des neuen Programms eben.

Verknüpfung zweier Programme mit &&

Wenn zwei Programme mit zwei direkt aufeinanderfolgenden Ampersands (&)verknüpft werden, so wird das zweite Programm nur ausgeführt, wenn das erste mit Rückgabewert 0 abgeschlossen wurde.

Verknüpfung zweier Programme mit ||

Wenn zwei Programme mit zwei direkt aufeinanderfolgenden Pipe-Symbolen (|)verknüpft werden, so wird das zweite Programm nur ausgeführt, wenn das erste mit Rückgabewert ungleich 0 abgeschlossen wurde.

Eine häufig benutzte Zeile, die die dritte Möglichkeit ausnutzt ist

```
test -x /usr/bin/foo || exit
```

Wenn das Programm `/usr/bin/foo` nicht existiert und ausführbar ist, wird das Script mit `exit` beendet.

Mail an root unter bestimmten Bedingungen

Ein Script wird häufig automatisch abgearbeitet auch wenn kein User am Terminal anwesend ist. Wenn es jetzt eine Meldung an den Benutzer oder den Systemverwalter ausgeben will, bleibt nur das Mail-System. Oder es soll eine Nachricht an den Systemverwalter abschicken, wenn bestimmte Bedingungen erfüllt sind, auch hier ist die E-Mail die beste Methode. Dazu existiert der Befehl **mail**.

Dieser Befehl schickt eine Mail mit bestimmter Titelzeile an eine bestimmte Adresse. Die Mail selbst wird entweder aus einer Datei oder aus einem Here-Dokument von der Standard-Eingabe gelesen. Im Script ist die Form des Here-Documents am verbreitetsten:

```
mail -s Titel Adresse <<EOM
Beliebige Textzeilen
EOM
```

Alles, was zwischen den beiden EOMs steht, wird an die angegebene Adresse per Mail verschickt. Beinhaltet der Titel Leerzeichen, so

muß er in Anführungszeichen gesetzt werden. Als Adresse für den Systemverwalter kann einfach `root` eingegeben werden:

```
#!/bin/bash
Auslastung=`df /dev/hda1| grep ^/ |cut -c52-54`
if [ $Auslastung -gt 90 ]
then
  mail -s "Alarm: Platte bald voll" root <<EOM
    Hallo Systemverwalter. Die Platte /dev/hda1 ist bald voll.
    Sie ist zu $Auslastung belegt.
    Mach was!!!
  EOM
fi
```

Es können aber auch Programme direkt ihre Ausgaben an **mail** weiterpipen, also etwa

```
df | mail -s "Ausgabe von df" root
```

In beiden Fällen wird eine Mail an root verschickt, im ersten Beispiel, wird eine Warnung an root weitergegeben, wenn die Platte `/dev/hda1` voller als 90% ist, im zweiten wird einfach die Ausgabe von **df** gemailt.

Speicherort von Scripts

Scripte sind nur dann ausführbar, wenn sie innerhalb des Suchpfades liegen. Bei der Wahl einer geeigneten Stelle für eigene Scripts sollte unterschieden werden, wofür sie gedacht sind.

Soll nur der Systemverwalter diese Scripts ausführen dürfen, dann empfiehlt es sich,

1. die Scripts entweder in `/usr/local/sbin` oder `/root/bin` abzuspeichern. Beide Verzeichnisse liegen nur im Suchpfad des Systemverwalters.
2. die Zugriffsmodi auf 700 zu setzen (`rwx------`), und dafür zu sorgen, daß sie root gehören. Somit kann sie nur root ausführen.

Sollen sie aber von allen Usern ausführbar sein, so sollten sie nach `/usr/local/bin` gelegt werden und den Zugriffsmodus 755 (`rwxr-xr-x`) besitzen. `/usr/local/bin` ist im Suchpfad aller User.

Grundsätzliche Vorsicht ist mit der Verwendung des SUID-Bits angeraten. Die Shell reagiert aber seit einigen Jahren sehr vorsichtig darauf und bezieht sich ihre Informationen über die Identität nicht aus der Effektiven UID sondern aus der echten UID. Ältere Versionen können hier aber erhebliche Schwierigkeiten machen...

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.111.1

Verwalten von Benutzer- und Gruppenkonten und verwandten Systemdateien

---

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Benutzerkonten hinzuzufügen, zu löschen, zu deaktivieren und zu ändern. Enthaltene Tätigkeiten beinhalten das Hinzufügen und Löschen von Gruppen und das Ändern der Benutzer- und Gruppeninformation in den passwd/group Datenbanken. Ebenfalls enthalten ist das Erstellen von speziellen und eingeschränkten Konten.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **chage**
- **gpasswd**
- **groupadd**
- **groupdel**
- **groupmod**
- **grpconv**
- **grpunconv**
- **passwd**
- **pwconv**
- **pwunconv**
- **useradd**
- **userdel**
- **usermod**
- `/etc/passwd`
- `/etc/shadow`

- /etc/group
- /etc/gshadow

Um mit einem Linux-Rechner arbeiten zu können, muß man eine gültige Userkennung auf diesem Rechner besitzen. Ein wichtiger Teil der administrativen Aufgaben eines Systemverwalters bezieht sich auf die Verwaltung dieser User-Accounts. Dieser Abschnitt behandelt die dazu notwendigen Fähigkeiten.

Ein User auf einem Linux-Rechner wird intern über seine UserID (UID), eine Nummer von 0 bis 65535, verwaltet. Der User mit der Nummer 0 ist der Systemverwalter. Nicht der Name root sondern die UserID 0 machen ihn zum Superuser. Alle anderen User sind sogenannte Normaluser, deren Rechte zwar stark unterschiedlich sein können, wobei der Unterschied aber nicht über die UserID definiert ist.

Jeder User gehört mindestens einer Gruppe zu, kann aber Mitglied beliebig vieler Gruppen sein. Eine dieser Gruppen ist die Login-Gruppe (manchmal auch primäre Gruppe genannt) des Users, die Gruppe, der später die Dateien gehören werden, die der User anlegt.

Die Dateien, in denen die Informationen über User und Gruppen abgelegt werden und die Befehle, die wir zum Anlegen und Bearbeiten dieser Userinformationen brauchen werden im Folgenden dargestellt.

Die Datei /etc/passwd

Die Datei `/etc/passwd` enthält alle Usereinstellungen, bis auf das verschlüsselte Passwort. In älteren Versionen ist auch das Passwort hier abgelegt, aus Sicherheitsgründen wurde es aber in den neueren Versionen in die Datei `/etc/shadow` ausgelagert.

Die Datei `/etc/passwd` muss für alle User lesbar sein, denn sie ist der einzige Ort, an dem die Verbindung zwischen UserID und Usernamen hergestellt werden kann.

```
-rw-r--r--   ...    /etc/passwd
```

Das wird praktisch bei jedem Aufruf von Programmen benötigt, die auch nur irgendetwas mit Usernamen zu tun haben, sogar ein `ls -l` braucht also diesen Zugriff. Wenn das verschlüsselte Passwort hier abgelegt wäre, so könnte ein Angreifer die Datei lesen, das Passwort nehmen und über eine *brute force attack* (Ausprobieren aller denkbaren Möglichkeiten) das Passwort entschlüsseln. Das war der Grund für die Auslagerung.

Die Datei `/etc/passwd` ist eine Textdatei, die für jeden User eine Zeile angelegt hat. Diese Zeile enthält verschiedene Felder, die durch einen Doppelpunkt voneinander getrennt sind. Das Format ist:

```
Username:x:UserID:GruppenID:Beschreibung:Homeverzeichnis:Startshell
```

Dabei bedeuten die einzelnen Felder:

Username

Der Name des Users. Alphanumerische Zeichen.

x

Hier stand früher das verschlüsselte Passwort. Das x bedeutet, daß die Passwort-Information jetzt in /etc/shadow ausgelagert wurde.

UserID

Die numerische UserID (UID). Eine einheitliche Nummer von 0 bis 65534. Wenn mehrere User die selbe ID benutzen, dann sind sie aus der Sich des Systems die selben User. Die UID 0 ist für den Systemverwalter reserviert.

GruppenID

Die numerische GruppenID (GID) der Login-Gruppe des Users. Der User kann noch Mitglied anderer Gruppen sein, diese Gruppe ist aber seine initiale Gruppe, d.h. alle Dateien, die der User anlegt werden normalerweise die Gruppenzugehörigkeit zu dieser Gruppe haben. (Ausnahme: Dateien, die in Verzeichnissen angelegt werden, die eine andere Gruppenzugehörigkeit haben und das SGID-Bit gesetzt haben)

Beschreibung

Hier kann eine kurze Beschreibung des Users stehen. Das Format ist technisch gesehen frei wählbar. Inzwischen hat sich ein Standard durchgesetzt, der die folgenden Informationen hier ablegt:

- Ausgeschriebener Name des Users (Vor- und Zuname)
- Büro-Raumnummer
- Telephonnummer Büro
- Telephonnummer Privat
- Sonstiges

Diese Informationen werden mit Kommata voneinander getrennt. Programme wie `finger` greifen auf diese Information zu. Dieses Feld kann aber auch leergelassen werden oder einfach nur eine Kurzbeschreibung beinhalten.

Homeverzeichnis

Das Verzeichnis des Users. Beim Login wird automatisch in dieses Verzeichnis gewechselt.

Startshell

Die Shell, die beim Login gestartet werden soll. Hier wird bei normalen Usern meist die /bin/bash oder /usr/bin/csh stehen. Ein Sonderfall sind Useraccounts, die nötig sind um bestimmte Dienste anzubieten (z.B. Samba, pop3, ...). Solche User sollen evt. keine Möglichkeit haben, sich regulär einzuloggen. In einem solchen Fall wird hier zweckmäßigerweise /bin/false eingetragen. Das ist ein Programm, das nichts tut, als einen Rückgabewert von 1 zu produzieren. Wenn ein User diesen Eintrag hat, kann er sich nicht

einloggen, bzw. er kann schon, fliegt aber sofort wieder raus, weil seine "Shell" sich sofort wieder verabschiedet.

Eine typische Zeile aus /etc/passwd könnte also folgendermaßen aussehen:

```
root:x:0:0:Hans Maier,1,089/12345,089/54321,Systemverwalter:/root:/bin/bash
```

Oft wird das Beschreibungsfeld gar nicht ausgefüllt oder nur mit einem kurzen Namen versehen, dann wird es noch einfacher:

```
root:x:0:0::/root:/bin/bash
```

Wenn ein Feld also leergelassen wird, folgen die Doppelpunkte direkt aufeinander.

Die Datei /etc/shadow

Nachdem das verschlüsselte Passwort jetzt nicht mehr in der Datei /etc/passwd abgelegt wird, ist dazu eine andere Datei notwendig. Diese Datei heißt /etc/shadow und birgt neben dem Passwort noch einige Einstellmöglichkeiten, die sich alle um das Passwort drehen. Wie schon in /etc/passwd handelt es sich um eine reine Textdatei, die für jeden User eine Zeile veranschlagt. Wieder ist jede Zeile in Felder aufgeteilt, die durch Doppelpunkte getrennt werden.

Die Datei /etc/passwd ist **nicht lesbar für alle** sondern meist nur durch den Systemverwalter. Manche Distributionen legen eine Gruppe `shadow` an, deren Mitglieder noch ein Leserecht auf diese Datei haben, das ist aber nicht zwingend erforderlich. Ein Zugriffsrecht von

```
-rw-------    ...   /etc/shadow
```

ist völlig ausreichend.

Die interne Struktur jeder Zeile ist folgende:

```
Username:Passwort:Alter:min_Alter:max_Alter:Warnzeit:Pufferzeit:Ungültigkeit:Reserviert
```

Die Bedeutungen der einzelnen Felder sind:

Username

Der Username, genau so wie in /etc/passwd. Das ist sozusagen das Schlüsselfeld des Datensatzes, mit dem die Verbindung zu den Zeilen aus /etc/passwd hergestellt wird.

Passwort

Hier steht das verschlüsselte Passwort. Linux verschlüsselt seine Passwörter mit einem nicht-reversiblen Verschlüsselungsmechanismus durch den Systemaufruf `crypt`. Das Ergebnis ist immer eine 13-24-stellige Zeichenkette, egal wie lange das eingegebene Passwort war. Wenn hier also eine Zeichenkette steht, die nicht 13-24-stellig ist, so existiert kein Passwort, das durch Aufruf von crypt in diese Zeichenkette verschlüsselt werden würde. Damit ist der Eingang faktisch gesperrt. Häufig steht hier z.B. ein einzelnes Sternchen (*) oder ein Ausrufungszeichen (!). Das heißt, daß dieser Eingang verschlossen ist, es existiert kein gültiges Passwort. Der Systemverwalter kann sich aber mit dem Befehl **su** in diesen Benutzer verwandeln und unter seiner Effektiven UID arbeiten. Steht in diesem Feld nichts, folgt also ein Doppelpunkt dem nächsten, so gibt es kein Passwort, der Eingang ist frei, der User kann sich ohne Nennung eines Passworts einloggen.

Alter

Hier stehen die Anzahl der Tage seit dem 1. Januar 1970 bis zu dem Tag, an dem das Passwort das letzte Mal verändert wurde. Keine Angst, das müssen Sie nicht selbst ausrechnen, das Programm passwd tut das für Sie und trägt diese Zahl hier ein.

min_Alter

Das Mindestalter eines Passworts, bevor es geändert werden darf. Also genau genommen, die Anzahl der Tage, die seit dem letzten Passwort-Wechsel vergehen müssen, damit es erneut verändert werden darf. Das hat in der Praxis eigentlich nur dann eine Bedeutung, wenn einem User verboten werden soll, das Passwort zu ändern (z.B. wenn mehrere User sich einen Account teilen). Wenn dieses Feld eine größere Zahl aufweist, als das nächste, dann darf der User sein Passwort nie ändern.

max_Alter

Die maximale Anzahl von Tagen, bevor ein Passwort geändert werden muß. Viele Hochsicherheitssysteme verlangen, daß die Passwörter regelmäßig geändert werden müssen. Hier wird die Anzahl der Tage eingetragen, die maximal zwischen zwei Wechseln des Passworts vergehen dürfen.

Warnzeit

Damit ein User merkt, das das Passwort bald geändert werden muß, kann hier die Anzahl der Tage vor dem Ablauf der Gültigkeit eines Passworts angegeben werden, ab der der User beim Login auf den baldigen Ablauf hingewiesen wird.

Pufferzeit

Weil es sein kann, daß ein User ein paar Tage nicht eingeloggt war, kann er womöglich die Warnmeldung nie gesehen haben, die ihn auf das baldige Ablaufen des Passworts hingewiesen hätte. Hier kann man die Pufferzeit in Tagen angeben, die nach dem Ablaufen des Passworts verstreichen darf, bevor der Account tatsächlich ungültig wird.

Ungültigkeit

Dieses Feld wird nur für Accounts benötigt, die zeitlich befristet sind. Hier steht die Anzahl von Tagen seit dem 1. Januar 1970, bis zu dem Tag, an dem der Account ungültig wird. Auch dazu gibt es selbstverständlich Programme, die diesen Wert in ein Datum umrechnen.

Reserviert

Ein reserviertes Feld für zukünftige Erweiterungen.

Ein minimaler Eintrag besteht nur aus dem Usernamen, gefolgt von 8 Doppelpunkten. Das wäre dann ein User, der sich ohne Passwort einloggen darf. Die anderen Felder sind optional und können - je nach Sicherheitsbedarf des Systems - gesetzt werden, wie gewünscht.

Die Datei /etc/group

Diese Datei enthält die Gruppeninformationen für die Usergruppen des Systems. Sie ist - wie die anderen Dateien der Userverwaltung - wieder eine Textdatenbank mit Feldern, die durch Doppelpunkte voneinander getrennt sind. Jede Zeile beschreibt eine Gruppe.

Die Zeilen definieren die Mitglieder der Gruppen, die **nicht** die Login Gruppen der User sind. Die Login-Gruppen-Mitgliedschaft ist in dieser Datei also **nicht** verzeichnet.

Wie schon bei der Datei /etc/passwd, so gilt auch bei der Datei /etc/group, daß sie für alle Welt lesbar sein muß. Daher sind wieder die Passwörter der Gruppen in die Datei /etc/gshadow ausgelagert worden, deren Format gleich im Anschluß besprochen wird. Wie in /etc/passwd, so steht im Passwort-Feld in /etc/group heute einfach ein x.

Das Format der Zeilen in /etc/group ist:

```
Gruppenname:x:GruppenID:Liste der Mitglieder
```

Die Bedeutung der einzelnen Zeilen ist schnell erklärt:

Gruppenname
: Der Name der Gruppe, wie er dann z.B. von ls -l ausgegeben wird.

x
: Das Feld, in dem früher das Passwort stand.

GruppenID
: Die numerische GruppenID (GID), die die Gruppe kennzeichnet. Die Datei /etc/passwd referenziert die Login-Gruppen der Usereinträge z.B. mit dieser Nummer.

Liste der Mitglieder
: Eine durch Kommata getrennte Liste von Usernamen. User, die hier aufgeführt sind, sind durch diesen Eintrag Mitglieder der jeweiligen Gruppe. Die Gruppenmitgliedschaft der Login-Gruppe eines Users muß hier nicht aufgeführt sein.

Gruppenmitgliedschaften sind oft nicht ganz richtig verstanden. Aus diesem Grund folgt hier noch eine kurze Beschreibung der Mechanismen:

Jeder User kann Mitglied beliebig vieler Gruppen sein, hat aber immer genau eine Login-Gruppenmitgliedschaft. Diese Login-Gruppenmitgliedschaft wird in /etc/passwd definiert. Jede weitere Gruppenmitgliedschaft eines Users wird in der Datei /etc/group definiert.

Die pure Mitgliedschaft in einer Gruppe gibt dem User die Rechte eines Gruppenmitglieds auf Dateien, die dieser Gruppe zugehören. Es sind keine weiteren Befehle nötig, um diese Rechte wahrzunehmen. Ein Beispiel:

Wenn der User `foo` Mitglied der Gruppen `users, bar, foobar` und `uucp` ist, dann kann er alle folgenden Dateien lesen:

```
-rw-r-----   1 root     users          294 Jul 29  2000 Datei1
-rw-r-----   1 root     bar            294 Jul 29  2000 Datei2
-rw-rw----   1 root     foobar         294 Jul 29  2000 Datei3
-rw-r-----   1 root     uucp           294 Jul 29  2000 Datei4
```

Die Datei `Datei3` dürfte er sogar beschreiben.

Um diese Rechte auszuüben, muß der User `foo` keinerlei zusätzliche Befehle wie newgrp oder ähnliches benutzen, er hat die Rechte einfach durch die bloße Gruppenmitgliedschaft.

Wenn ein User eine Datei anlegt, dann bekommt diese Datei automatisch die Gruppenzugehörigkeit zur Login-Gruppe des Users. Das kann durch Befehle wie newgrp oder sg verändert werden.

Die Datei /etc/gshadow

Gruppenpasswörter werden nicht dazu benötigt, eine Gruppenmitgliedschaft zu beweisen. Sie dienen nur dazu, einem User, der **nicht** Mitglied einer bestimmten Gruppe ist, zeitweilig die Rechte eines Mitglieds zu geben. Dazu muß er das Gruppenpasswort bekommen. Das wird nur sehr selten benötigt, denn es ist ja nicht viel Aufwand, einen User zu einem Gruppenmitglied zu machen.

Auch wenn ein User selbst kein Passwort hat, aber die Gruppe, zu der er mit newgrp wechseln will eines besitzt, muß es angegeben werden.

Damit die Passwörter für Gruppen nicht einlesbar sind, werden sie wieder in einer Datei abgelegt, die nicht für alle Welt lesbar ist. Diese Datei heißt `/etc/gshadow`. Das Format dieser Datei ist wieder ähnlich wie bei den anderen besprochenen Dateien. Zeilen, die in Felder aufgeteilt sind, durch Doppelpunkte voneinander getrennt. Das genaue Format ist:

```
Gruppenname:Passwort:Gruppenverwalter:Liste der Mitglieder
```

Die einzelnen Felder haben folgende Bedeutung:

Gruppenname

Der ausgeschriebene Name der Gruppe, wie er in /etc/group steht

Passwort

Das verschlüsselte Passwort der Gruppe. Steht hier eine Zeichenkette, die kein gültiges verschlüsseltes Passwort darstellt, also eine nicht 13-24 stellige Zeichenkette, dann hat die Gruppe kein gültiges Passwort. Das heißt, User, die nicht Mitglied der Gruppe sind, sich nicht mit newgrp zum Gruppenmitglied erklären können.

Gruppenverwalter

Jede Gruppe kann einen User haben, der als Gruppenverwalter fungiert. Ein Gruppenverwalter darf

- Andere User in die Gruppe aufnehmen oder aus der Gruppe ausschließen.
- Das Gruppenpasswort ändern.
- Das Gruppenpasswort löschen.

In diesem Feld steht der Name des Gruppenverwalters der Gruppe, so wie er auch in /etc/passwd aufgeführt ist.

Liste der Mitglieder

Eine Liste der Mitglieder der Gruppe, die sich mit newgrp zum Mitglied der Gruppe machen dürfen. Diese Liste ist nicht zwansläufig deckungsgleich mit der der gleichen Gruppe in /etc/group. Ein User, der Mitglied hier in der Liste ist, aber nicht als Mitglied der Gruppe in /etc/shadow aufgeführt ist, ist nicht automatisch Mitglied der Gruppe, kann sich aber ohne Nennung eines Passworts mit newgrp zum Mitglied der Gruppe ernennen.

Der wichtigste Grund für die Verwendung der /etc/gshadow-Datei ist wohl der, daß jede Gruppe einen eigenen Gruppenverwalter haben kann. Diese Tatsache erleichtert das Leben des Systemverwalters, weil er sich nicht um alles selbst kümmern muß.

Programme zum Umgang mit den User-/Gruppendatenbanken

Die Arbeiten an all diesen vier Dateien (passwd, shadow, group und gshadow) können prinzipiell immer mit einem Texteditor wie dem vi vorgenommen werden, es gibt aber auch viele wichtige Programme, die einem Systemverwalter die zeitaufwendige Handarbeit ersparen und auch in Schleifen bestens dazu geeignet sind, Veränderungen für viele User (oder Gruppen) automatisch vorzunehmen. Dazu kommt, daß es Programme geben muß, mit deren Hilfe auch Normaluser ihre Einstellungen verändern können, etwa ihr Passwort wechseln. Diese Programme sollen hier noch kurz dargestellt werden. Ich habe - wo immer es noch keine deutsche Handbuchseite gab - die jeweiligen Handbuchseiten übersetzt und hier mit aufgenommen.

Die beste Art, diese Programme kennenzulernen, ist es, möglichst viel mit ihnen herumzuspielen, neue User anzulegen, zu verändern,

wieder zu löschen usw. Nur die Praxis mit Programmen führt im Endeffekt zur Vertrautheit mit dem Umgang...

Das Anlegen neuer User mit useradd

Das Programm useradd dient dazu, einen neuen User anzulegen, ohne die Handarbeit mit Editoren auf den Dateien passwd, shadow, group und gshadow. Wir haben oben schon gesehen, daß das Format z.B. der Datumsangaben (Tage seit dem 1. Januar 1970) nicht unbedingt geeignet ist, von Hand gesetzt zu werden. Das Programm **useradd** nimmt uns diese Arbeit ab.

Die genaue Anwendung dieses Programms entnehmen Sie der Handbuchseite, hier nur noch ein paar Tipps aus der Praxis:

Wenn mit useradd ein neuer User angelegt werden soll, dann reicht oft nur die Angabe des Namens, weil die Voreinstellungen in der Regel sehr vernünftige Werte beinhalten. Oft wird jedoch vergessen, daß das Programm ohne die Option **-m** das Homeverzeichnis des Users nicht anlegt. Daran sollten Sie denken.

Die Angabe eines Passwortes ist selten von Nöten. Wird useradd ohne Passwort angegeben, so sperrt es den Account zunächst einmal durch ein ! im Passwortfeld der Datei /etc/shadow. Später können Sie mit dem Programm passwd ein Passwort vergeben.

Löschen von Usereinträgen mit userdel

Das Löschen von Useraccounts übernimmt das Programm userdel. Wird zusätzlich noch die Option **-r** angegeben, so wird gleich das Homeverzeichnis des Users mitgelöscht. Andere Dateien im System, die diesem User gehören werden nicht gesucht. Hier muß "von Hand" gesucht werden, das nimmt uns **userdel** nicht ab. Aber wir kennen ja jetzt das Programm find, mit dem wir solche Suchen in einem Rutsch erledigen können...

Ändern von Usereinstellungen mit usermod

Um die ganzen Eigenschaften eines bereits bestehenden Users zu verändern gibt es das Programm usermod. Hier können die ganzen Einstellungen verändert werden, die wir schon beim Anlegen eines Users angeben konnten. Zusätzlich kann auch der Name des Users noch verändert werden.

Bei der Änderung verschiedener Eigenschaften kommt es aber immer wieder zu Komplikationen, ein paar davon seien hier genannt:

- **Ändern der UserID**
  Wird die UserID eines Users verändert, dann stimmen die Rechte des Users nicht mehr. Denn die Rechte des Users auf Dateien werden ja über den Eintrag der UID in der Inode bestimmt. Aus diesem Grund verändert **usermod** zwar die UserID aller Dateien und Verzeichnisse innerhalb des Homeverzeichnisses, jedoch nicht die der restlichen Dateien im System. Das muß wiederum mit find erledigt werden.

- **Ändern des Usernamens**
  Wird der Username geändert, so verändert sich damit nicht automatisch der Name des Homeverzeichnisses. Das kann zu Inkonsistenzen führen, wenn es die Regel ist, daß alle User ihre Homeverzeichnisse in `/home/`*Username* vorfinden, nur der User Otto hat seins in /home/maier... Es ist kein technisches Problem, aber das Leben wird einfacher, wenn man sich an Prinzipien hält.
- **Ändern des Homeverzeichnisses**
  Wenn also auch das Homeverzeichnis geändert werden soll, so darf nicht vergessen werden, die Option **-m** mit anzugeben, sonst werden die Dateien des alten Homeverzeichnisses nicht ins neue verschoben bzw. das neue wird gar nicht erst angelegt.

Ändern der Passwort-Gültigkeit mit chage

Speziell für die Einträge in `/etc/shadow` existiert noch das Programm chage, das entweder über Kommandozeilenparameter oder - falls keine Parameter angegeben wurden - interaktiv die Einstellungen dieser Werte erlaubt.

Ändern von Usereigenschaften mit passwd, newgrp, chsh und chfn

Alle drei Programme **useradd, userdel** und **usermod** sind Werkzeuge des Systemverwalters und werden nur von ihm aufgerufen. Damit auch Normaluser ein paar Dinge - zumindestens für ihren eigenen Account - ändern dürfen gibt es noch ein paar Programme, deren Aufgaben sich zum Teil mit denen der drei letztgenannten überschneiden - passwd, newgrp, chsh und chfn.

Diese Programme ändern alle (mit Ausnahme von newgrp) verschiedene Einstellungen in einer oder mehreren der vier oben beschriebenen Dateien passwd, shadow, group und gshadow. Die genaue Anwendung entnehmen Sie wiederum den Handbuchseiten. Hier eine kurze Beschreibung, was die einzelnen Programme anbieten und was Normaluser damit anstellen können:

- **passwd**
  Mit diesem Programm kann ein Normaluser sein Passwort ändern. Ein Verwalter einer Gruppe kann damit zusätzlich auch das Gruppenpasswort dieser Gruppe ändern, der Systemverwalter kann alle Passwörter aller User und Gruppen ändern.
- **newgrp**
  Mit diesem Kommando kann ein User kurzfristig seine Logingruppe wechseln. Das heißt, daß Dateien, die er anlegt jetzt zu dieser neuen Gruppe gehören, statt zur eigentlichen Logingruppe des Users. Wenn der User selbst nicht Mitglied der gewünschten Gruppe ist, jedoch das Gruppenpasswort kennt, so kann er nach der Eingabe des Gruppenpassworts trotzdem kurzfristig Mitglied dieser Gruppe werden. Dieses Programm ändert nichts an den Systemdateien, beim nächsten Einloggen ist alles so wie vorher.
- **chsh**
  Jeder User kann mit diesem Programm den Eintrag der Startshell in seinem Useraccount in /etc/passwd ändern. Dieses Programm ändert nur diesen Eintrag, erst durch ein Logout mit anschließendem erneuten Login wird die Änderung spürbar. Damit ein User nicht jedes beliebige Programm hier eintragen kann, muß die gewünschte Shell in `/etc/shells` aufgelistet sein. Der

Systemverwalter kann mit diesem Programm jedem User eine andere Startshell zuweisen.

- **chfn**
  Jede Zeile der /etc/passwd enthält ein Kommentarfeld, in dem verschiedene Informationen über einen User eingegeben werden können. Jeder User kann diese Informationen selbst ändern oder löschen (mit Ausnahme eines Feldes). Dazu benutzt er das Programm **chfn** (CHange Full Name). Selbstverständlich kann ein Normaluser damit nur das Feld seiner eigenen Zeile ändern, der Systemverwalter kann aber damit die Einträge aller User editieren.

Die beiden letzten Programme arbeiten auch interaktiv. Wenn keinerlei Parameter angegeben werden, so fragen diese Programme die Angaben ab.

Verwaltung der Gruppen

Die Aufgaben zum Verwalten von Gruppen entsprechen im Wesentlichen denen zur Verwaltung der User. Und so existieren für den Systemverwalter (und nur für ihn) hier genau die drei Befehle, die wir für die Userverwaltung schon hatten:

- **groupadd**
  Zum Anlegen von Gruppen
- **groupmod**
  Zum Ändern der bestehenden Eigenschaften der Gruppen
- **groupdel**
  Zum Löschen von Gruppen.

Mit der Einführung der Shadow-Gruppen, also der verschlüsselten Speicherung der Gruppenpasswörter in der Datei `/etc/gshadow` hielt eine andere Neuerung ihren Einzug, die Möglichkeit, daß bestimmte User zu Gruppenadministratoren werden. Ein Gruppenadministrator hat bestimmte Rechte für die Gruppe, deren Verwalter er ist. Er kann

- User in eine Gruppe als Mitglied aufnehmen
- Mitglieder einer Gruppe von der Gruppenmitgliedschaft ausschließen
- Gruppenpasswörter setzen, verändern und löschen.

Für diese Aufgabe steht ihm das Programm gpasswd zur Verfügung. Dieses Programm ist prinzipiell für zwei Aufgabenbereiche geeignet:

- Der Systemverwalter kann damit den Gruppenverwalter bestimmen.
- Der Gruppenverwalter kann damit die oben genannten Aufgaben erledigen.

Und schließlich gibt es noch das Programm groups, das jeder User dazu nutzen kann, herauszufinden in welchen Gruppen er eigentlich Mitglied ist. Dieses Programm ist eigentlich nur ein Shellscript, das wiederum den Befehl id nutzt. Id ist ein Programm, mit dem

herausgefunden werden kann, welche UserID, GruppenIDs usw ein User besitzt.

Überprüfung der Konsistenz mit pwck und grpck

Da all diese Verwaltungsaufgaben für die oben beschriebenen User- und Gruppendateien viel Aufwand ist, gibt es noch die Programme pwck (Password-Check) und grpck (Group-Check), die die interne Konsistenz dieser Dateien überprüfen. Diese Programme sind naturgemäß Werkzeuge des Systemverwalters, da außer ihm niemand die shadow-Dateien lesen darf.

Konvertierung der User- und Gruppendatenbanken

In seltenen Fällen kann es notwendig sein, die Passwort- und Gruppendatenbanken entweder von der alten (shadow-losen) Version in die moderne (shadow-basierte) Version zu konvertieren oder umgekehrt. Zu diesem Zweck stehen uns die folgenden Programme zur Verfügung:

pwconv	Konvertiert die alte /etc/passwd Datei in eine passwd/shadow Kombination.
pwunconv	Konvertiert eine moderne passwd/shadow Kombination in eine alte passwd Datei.
grpconv	Konvertiert die alte /etc/group Datei in eine group/gshadow Kombination.
grpunconv	Konvertiert eine moderne group/gshadow Kombination in eine alte group Datei.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.111.2

Einstellen von Benutzer- und Systemumgebungsvariablen

Beschreibung: Prüfungskandidaten sollten in der Lage sein, globale und Benutzerprofile zu modifizieren. Dieses Lernziel beinhaltet das Setzen von Umgebungsvariablen, das Verwalten von skel Verzeichnissen für neue Benutzerkonten und das Setzen des Suchpfades auf die richtigen Verzeichnisse.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **env**
- **export**
- **set**
- **unset**
- `/etc/profile`
- `/etc/skel`

Profildateien

Wenn sich ein User an einem Linux-System anmeldet, sich also dort mit Usernamen und Passwort einloggt, so wird als erste Aktion, nach dem Login eine Shell gestartet. Welche Shell gestartet werden soll steht in dem jeweiligen Eintrag des Users in `/etc/passwd`. Jede Shell kann beim Start konfiguriert werden, das heißt, sie durchläuft mehrere Konfigurationsdateien, in denen bestimmte Umgebungsvariablen gesetzt werden können und evt. andere Programme aufgerufen werden, die im Hintergrund Arbeiten verrichten, die bei jedem Login erneut ausgeführt werden müssen.

Bei den Startdateien gibt es grundsätzlich zwei verschiedene Kategorien. Zum Einen, die Dateien, die bei jedem Login jedes Users ausgeführt werden und zum Anderen, die userbezogenen Dateien, in denen jeder User seine Einstellungen selbst treffen kann. Die

systemweiten Startdateien liegen grundsätzlich in /etc, die userbezogenen grundsätzlich im Home-Verzeichnis des jeweiligen Users.

Linux/Unix kennt verschiedene Standard-Shells, die zum Teil unterschiedliche solche Startdateien benutzen. Im weiteren Verlauf dieser Darstellung werde ich mich auf die der Linux-Standardshell BASH reduzieren. Hier nur einmal kurz eine Auflistung verschiedener Startdateien:

<table>
<tr><th>Art</th><th colspan="2">Bourne Again Shell</th><th>Korn Shell</th><th>TC-Shell</th></tr>
<tr><td>Systemweit</td><td colspan="2"><code>/etc/profile</code></td><td><code>/etc/csh.cshrc</code>
<code>/etc/csh.login</code></td></tr>
<tr><td rowspan="2">Userbezogen</td><td><code>~/.bash_profile</code>
<code>~/.bash_login</code></td><td></td><td rowspan="2"><code>~/.tcshrc</code> oder <code>~/.cshrc</code>

<code>~/.history</code>
<code>~/.login</code>
<code>~/.cshdirs</code></td></tr>
<tr><td colspan="2"><code>~/.profile</code></td></tr>
</table>

Aus der Tabelle ist zu ersehen, daß die **bash** und die **ksh** zumindestens zwei Dateien gemeinsam benutzen, während die **csh** nur eigene Startdateien benutzt. Das liegt an der Tatsache, daß **bash** und **ksh** scriptkompatibel sind (also die selbe Syntax bei Shellscripts haben) während die **csh** eine völlig andere Syntax für ihre Scripts benutzt.

Startdateien der Bourne Again Shell (bash)

Die Bourne Again Shell benutzt verschiedene Startdateien, die in unterschiedlichen Fällen abgearbeitet werden und in denen verschiedene Aufgaben erledigt werden können. Zunächst muß aber bei der Shell zwischen verschiedenen Kategorien unterschieden werden. Die erste grundsätzliche Unterscheidung liegt zwischen *interaktiver Shell* und *nicht interaktiver Shell*. Eine Shell ist immer dann eine interaktive Shell, wenn der User darin Kommandos eingeben kann. Das ist jedesmal der Fall, wenn

- ein User sich neu am System anmeldet,
- ein X-Term geöffnet wird,
- eine Shell eine andere Shell durch Programme wie su aufruft,
- eine Shell durch den User direkt aufgerufen wird.

Eine Shell ist eine nicht interaktive Shell immer dann, wenn z.B. eine Subshell aufgerufen wird, um ein Shellscript abzuarbeiten. Eine nicht-interaktive Shell arbeitet keine Startdateien ab.

Bei den interaktiven Shells muß zusätzlich noch zwischen einer *Login-Shell* und einer *Nicht Login-Shell* unterschieden werden. Die Shell ist dann eine Login-Shell, wenn sie direkt vom Login-Programm beim Anmelden eines Users gestartet wurde, oder wenn sie explizit mit dem Parameter `-l` aufgerufen wurde. Jede andere Shell, also wenn z.B. aus einem XTerm ein anderes Xterm aufgerufen wird, ist eine *nicht*

Login-Shell.

Der wesentliche Unterschied zwischen Login-Shell und Nicht-Login-Shell besteht darin, daß bei einer Login-Shell davon ausgegangen werden muß, daß noch keinerlei Umgebungsvariablen gesetzt sind. Hier ist es also nötig, daß der gesammte Konfigurationsvorgang durchgearbeitet werden muß, um sicherzustellen, daß die Umgebung den Anforderungen entspricht. Eine Nicht-Login-Shell hingegen wurde ja durch eine Login-Shell aufgerufen, die ihrerseits wieder den Konfigurationsvorgang durchwandert hat. Ausgehend von der Forderung, daß alle wichtigen Umgebungsvariablen durch die **export**-Anweisung weitervererbt werden, muß diese Shell also nicht nochmal konfiguriert werden. Der Aufruf wird so wesentlich schneller.

Konfigurationsdateien der Loginshell

Eine Login-Shell arbeitet zuallererst immer die Datei **/etc/profile** ab. Diese Datei enthält alle wichtigen Konfigurationen, die für alle User des Systems gültig sein sollen. Hier werden insbesondere die folgenden Variablen gesetzt:

- **PATH**
  Der Suchpfad für ausführbare Dateien. Die Reihenfolge der genannten Verzeichnisse ist wichtig. Das erste gefundene Programm wird ausgeführt.
- **MANPATH**
  Der Pfad, in dem die verschiedenen Handbuch-Hierarchien zu finden sind. Auch hier ist die Reihenfolge wichtig. Liegen etwa die deutschen Handbuchverzeichnisse hier nach den englischen, dann werden sie nur angezeigt, wenn mit `man -a` alle Seiten angefordert werden würden.
- **INFODIR** bzw. **INFOPATH**
  Der Pfad zu den Info-Seiten.
- **PAGER** und **EDITOR**
  Die Standardprogramme zum Ansehen und Editieren von Textdateien (z.B. less und vi).
- **PS1** und **PS2**
  Die Prompt-Einstellungen für den normalen und den Folgeprompt der Shell.

Daneben können hier weitere Variablen wie **LS_OPTIONS** oder **LESSCHARSET** gesetzt werden.

Auch Aliase und Shellfunktionen, die für alle User gelten sollen, werden in dieser Datei definiert. Eines ist aber für alle hier gesetzten Variablen, Funktionen und Aliase wichtig: **Alle hier gesetzten Variablen, Aliase und Funktionen, die auch in später aufgerufenen Shells Geltung haben sollen, müssen mit der Anweisung `export` exportiert werden!**

Absolut wichtig ist in **/etc/profile** die Angabe des initialen **umask**-Kommandos, das eine vernünftige Grundeinstellung für die Zugriffsmodi neu zu erstellender Dateien und Verzeichnisse setzt.

Des weiteren können hier noch Grenzwerte für die User gesetzt werden, was die Verwendung von Rechenzeit und Speicher angeht. Das

wird durch Aufrufe des Programms **ulimit** erreicht.

Wenn die Datei /etc/profile abgearbeitet ist, werden noch die userspezifischen Konfigurationsdateien abgearbeitet. Hier stehen drei verschiedene Dateien zur Verfügung, von denen nur eine einzige wirklich abgearbeitet wird. Die drei Dateien liegen alle im Heimatverzeichnis des jeweiligen Users und heißen **`.bash_profile .bash_login`** und **`.profile`**. Die Shell entscheidet nach der folgenden Regel, welche dieser drei Dateien abgearbeitet wird:

```
if [ -e ~/.bash_profile ]
then
  . ~/.bash_profile
elif [ -e ~/.bash_login ]
then
  . ~/.bash_login
elif [ -e ~/.profile ]
then
  . ~/.profile
fi
```

Oder - für Nicht-Bash-Programmierer:

Wenn im Heimatverzeichnis des Users die Datei .bash_profile existiert wird sie abgearbeitet.

Wenn diese Datei nicht existiert wird die Datei .bash_login im Heimatverzeichnis des Users gesucht und falls gefunden, abgearbeitet.

Wenn auch diese Datei nicht existiert, so wird im Heimatverzeichnis des Users die Datei .profile gesucht und falls gefunden abgearbeitet.

Diese drei Konfigurationsdateien bieten eigentlich jeweils die selben Möglichkeiten, wie die Einstellungen in /etc/profile, mit dem wesentlichen Unterschied, daß diese Einstellungen nur für den jeweiligen User gelten, in dessen Verzeichnis sich die Dateien befinden. Da immer nur eine dieser Dateien abgearbeitet wird, kann hier elegant jongliert werden. Ein paar Beispiele:

Will ein User experimentieren, so kann er alle Experimente in einer der beiden ersten Dateien machen, während er eine gut funktionierende dritte Datei besitzt. Falls etwas schiefgeht, muß er nur die ersten Dateien löschen und es wird wieder die dritte abgearbeitet.

Will ein Systemverwalter seinen Usern verbieten, Experimente oder eigene Einstellungen zu machen, muß er nur die Einstellungen in die erste abzuarbeitende Datei machen und dem User kein Schreibrecht darauf geben.

Neben diesen Startdateien kann auch noch eine Datei mit Namen **.bash_logout** existieren, die dann beim Logout abgearbeitet wird. Hier ist Platz für eventuell anstehende Aufräumarbeiten oder ähnliches.

Die Konfigurationsdatei der Nicht-Loginshell

Auch eine Nicht-Loginshell muß eventuell noch etwas konfiguriert werden. Das ist zwar nicht zwingend der Fall, insbesondere dann nicht, wenn die obigen Konfigurationsdateien der Loginshell alle wichtigen Einstellungen exportieren, aber die Shell kennt die Möglichkeit dazu. Allerdings ist diese Möglichkeit immer nur userbezogen.

Wenn im Verzeichnis eines Users die Datei **.bashrc** existiert, so wird sie von einer Nicht-Loginshell abgearbeitet, sobald diese Shell aufgerufen wird.

Grundsätzliches zu Umgebungsvariablen

Jede Shell und natürlich auch unsere Standard-Shell **bash** kann Variablen definieren. Die Summe aller definierten Variablen nennen wir die Umgebung der Shell (engl. environment).

Shellvariablen werden einfach durch die Angabe

Variablenname=Wert

definiert. Durch diese Definition steht uns die Variable mit dem gewählten Namen in **dieser und nur dieser** Shell zur Verfügung. Damit eine definierte Variable auch in Subshells zur Verfügung steht, die von unserer Shell geöffnet werden, muß die Variable **exportiert** werden. Das kann entweder im Nachhinein durch die Anweisung

export *Variablenname*

oder schon während der Definition durch

export *Variablenname=Wert*

geschehen. Erst jetzt werden solche Variablen auch in späteren Subshells zur Verfügung stehen. Allerdings stehen sie dort nur als Kopien der Variable der aufrufenden Shell zur Verfügung. Das hat zur Folge, daß die Variablen, die in der Subshell verändert werden, in der aufrufenden Shell unverändert bleiben.

Folgendes Beispiel mag das verdeutlichen:

```
NAME=hans       Eine Variable wird erzeugt
export NAME     Sie wird exportiert
bash            Eine Subshell wird geöffnet
echo $NAME      Wir überprüfen den Export
  hans          Hat funktioniert
NAME=otto       Ein neuer Wert für die Variable
```

`echo $NAME`	Wir überprüfen den neuen Wert
`otto`	Hat funktioniert
`exit`	Subshell wird geschlossen
`echo $NAME`	Inhalt der Variable in der ersten Shell
`hans`	Ist immer noch der alte Wert.

Jede Shell hat ihren eigenen Umgebungsspeicher, in den zwar jeweils Kopien der exportierten Variablen abgelegt werden, der aber beim Beenden der Shell wieder gelöscht wird. So kann in keinem Fall eine geänderte Variable einer Subshell wieder der ursprünglichen Shell "reimportiert" werden.

Jedes Shellscript, das aufgerufen wird, wird in einer Subshell abgearbeitet. Variablen, die in diesem Script erzeugt oder verändert werden, sind nach Beendigung des Scripts (und damit der ausführenden Subshell) nicht mehr aktuell.

Damit es aber trotzdem möglich ist, Shellscripts zur Definition von Variablen heranzuziehen, gibt es die Möglichkeit ein Script in der ursprünglichen Shell auszuführen, statt eine Subshell zu öffnen. Zu diesem Zweck wird ein einfacher Punkt (.) mit anschließendem Leerzeichen vor der Nennung des Scriptnamens angefügt.

`meinscript`	meinscript wird von einer Subshell ausgeführt
`. meinscript`	meinscript wird von der ursprünglichen Shell ausgeführt.

Durch den Punkt wird unsere Shell also gezwungen, das Script selbst abzuarbeiten. Die Variablen, die innerhalb dieses Scripts definiert wurden, sind jetzt auch nach der Abarbeitung des Scriptes noch gültig.

Das klingt schön, hat aber einen bedeutenden Haken. Wenn innerhalb des Scrips ein `exit` vorkommt, normalerweise dazu benutzt, um das Script zu beenden, wird ja die Shell, die das Script ausführt, beendet. Wurde das Script jetzt mit vorgestelltem Punkt aufgerufen, also von unserer Loginshell selbst, dann wird das exit die LoginShell beenden! Wir müßten uns also erneut einloggen.

Scripts, die mit führendem Punkt aufgerufen werden sollen, sollten daher auf keinen Fall - auch nicht zur Fehlerbehandlung - ein `exit` benutzen.

Eine Liste aller Umgebungsvariablen, die in der aktuellen Shell bekannt sind (zusammen mit ihren Inhalten) wird durch das Shellkommando **set** ausgegeben.

Eine Variable kann durch den Befehl

`unset` *`Variablenname`*

wieder aus dem Umgebungsspeicher entfernt werden.

Schließlich ist es noch möglich, einen Befehl in einer speziellen Umgebung auszuführen, die wir mit dem Befehl env erzwingen können.

Das Verzeichnis /etc/skel

Wenn ein neuer User angelegt werden soll, dann hätte der Systemverwalter viel Arbeit damit, jedem neu anzulegenden User all diese persönlichen Konfigurationsdateien eigens zu erstellen und einzurichten. Damit diese Arbeit gar nicht erst anfällt, existiert das Verzeichnis **/etc/skel**. Es enthält alle wichtigen Konfigurationsdateien, die in einem Userverzeichnis existieren sollten, damit er vernünftig arbeiten kann.

Beim Anlegen eines neuen Users werden diese Konfigurationsdateien dann in das neu angelegte Userverzeichnis kopiert und dem neuen User übereignet. Das wird automatisch dann ausgeführt, wenn das **useradd**-Kommando mit der Option `-m` aufgerufen wurde.

Ein Systemverwalter kann also sozusagen einen Musteruser anlegen und alle nötigen Konfigurationsdateien dieses Musterusers in dieses Verzeichnis speichern. Jeder neu angelegte User wird dann exakt die Konfiguration des Musterusers bekommen.

Neben den oben besprochenen Konfigurationsdateien der Shells finden sich hier auch noch viele weiteren Dateien, deren Namen praktisch immer mit einem Punkt beginnt und somit von einem normalen `ls` Kommando nicht angezeigt wird. Diese Dateien sind Konfigurationsdateien für verschiedene Programme, die ein Normaluser ausführen können soll. Welche Dateien im Einzelnen hier zu finden sind, muß der Systemverwalter entscheiden. Ein paar Beispiele:

`.Xdefaults`
: Persönliche Resourcen für X-Clients. Hier kann das Verhalten und Aussehen verschiedener X-Clients eingestellt werden.

`.Xresources`
: Persönliche Resourcen für X-Clients. Hier kann das Verhalten und Aussehen verschiedener X-Clients eingestellt werden. Exakt die selbe Funktion wie .Xdefaults.

`.Xmodmap`
: Tastaturbelegung für X11. Spezielle Einstellungen, wie die Tasten belegt werden sollen.

`.bash_history`
: Eine leere Datei zur Aufnahme der bereits eingegebenen Befehle der Shell (Befehlswiederholung).

`.dayplan`
: Termine des Users für das Programm plan

`.dayplan.priv`
: Private Termine des Users, auch für plan

`.emacs`

Persönliche Konfiguration für den Editor emacs

`.exrc`

Persönliche Konfiguration für den Editor ex

`.vimrc`

Persönliche Konfiguration für den Editor vim

`.gimprc`

Persönliche Konfiguration für das Bildbearbeitungsprogramm gimp

`.xinitrc`

Persönliche Konfiguration des Starts von X11

`.xfm`

Verzeichnis mit Konfigurationsdateien des X-Filemanagers

Und viele anderen mehr...

Wenn es gewünscht ist, mehrere verschiedene Musteruser anzulegen, dann können neben dem skel-Verzeichnis noch weitere solcher Verzeichnisse angelegt werden (z.B. skel_verwaltung, skel_produktion) und beim Anlegen der jeweiligen User muß dann useradd mit dem Parameter **-m -k *Verzeichnis*** aufgerufen werden, wobei *Verzeichnis* das gewünschte Musterverzeichnis ist.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.111.3

Konfigurieren und Nutzen der Systemlogs im Rahmen der administrativen und Sicherheitsanforderungen

---

Beschreibung: Prüfungskandidaten sollten in der Lage sein, die Systemaufzeichnungen zu konfigurieren. Dieses Lernziel beinhaltet das Einstellen der Art und Menge der aufgezeichneten Informationen, das manuelle Prüfen von Logdateien auf wichtige Aktivitäten, das Überwachen der Logdateien, das Einrichten automatischer Rotation und Archivierung der Logs und das Verfolgen von Problemen, die in den Logdateien aufgezeichnet

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **logrotate**
- **tail -f**
- `/etc/syslog.conf`
- `/var/log/*`

Unter Linux verwaltet ein spezieller Daemon-Prozess ein Systemlogbuch, das von allen Programmen und insbesondere von allen anderen Daemon-Prozessen benutzt werden kann, um Meldungen abzugeben. Dabei kann es sich um Meldungen des normalen Systemablaufs handeln oder um Alarmmeldungen, die sofortiges Eingreifen notwendig machen. Man spricht hier von unterschiedlichen **Prioritäten**.

Jedes Programm kann also über einen Systemaufruf solche Meldungen abgeben. Der Daemon, der diese Meldungen entgegennimmt und entscheidet, was mit ihnen zu geschehen hat heißt **syslogd**. Damit der Systemverwalter entscheiden kann, was mit welchen Meldungen geschehen soll kann dieser Daemon über die Datei **/etc/syslog.conf** konfiguriert werden.

In der Regel werden die verschiedenen Meldungen In Dateien geschrieben, die im Verzeichnis `/var/log` oder einem darinliegenden Unterverzeichnis abgelegt werden. Aber genau diese Frage, wohin wird welche Meldung geschrieben, wird in der Konfigurationsdatei ja eingestellt.

Die Datei /etc/syslog.conf

Jede Zeile dieser Datei, die nicht leer ist oder mit einem # beginnt, beschreibt eine Regel, was mit einer bestimmten Meldung passieren soll. die grundlegende Form jeder Zeile ist immer gleich und lautet:

```
Herkunft.Priorität      Aktion
```

Sollen in einer Regel mehrere Herkunftskategorien angegeben werden, so können sie, durch Kommas voneinander getrennt, nacheinander aufgeführt werden. Sollen mehrere Herkunfts/Prioritäts-Paare gleichzeitig angegeben werden, so müssen diese durch Strichpunkt getrennt angegeben werden. Sowohl Herkunft, als auch Priorität können durch ein Sternchen (*) ersetzt werden, das als Wildcard dient. In diesem Fall sind alle Herkunfts- bzw. Prioritätskategorien angesprochen.

Für Herkunft, Priorität und Aktion sind die folgenden Werte gültig:

Herkunftskategorien

kern	Systemmeldungen direkt vom Kernel
auth	Meldungen vom Sicherheitsdienst des Systems (login, ...)
authpriv	Vertrauliche Meldungen der internen Sicherheitsdienste
mail	Meldungen des Mail-Systems
news	Meldungen des News-Systems
uucp	Meldungen des UUCP-Systems
lpr	Meldungen des Druckerdaemons
cron	Meldungen des Cron-Daemons
syslog	Meldungen des syslog-Daemons selbst
daemon	Meldungen aller anderer Daemon-Prozesse
user	Meldungen aus normalen Anwenderprogrammen
local0-local7	frei verwendbar

Prioritäten in absteigender Reihenfolge

emerg	Der letzte Spruch vor dem Absturz
alert	Alarmierende Nachricht, die sofortiges Eingreifen erforderlich macht
crit	Meldung über eine kritische Situation, die gerade nochmal gut gegangen ist

err	Fehlermeldungen aller Art aus dem laufenden Betrieb
warn	Warnungen aller Art aus dem laufenden Betrieb
notice	Dokumentation besonders bemerkenswerter Situationen im Rahmen des normalen Betriebs
info	Protokollierung des normalen Betriebsablaufes
debug	Mitteilungen interner Programmzustände bei der Fehlersuche
none	Ist keine Priorität im eigentlichen Sinn, sondern dient zum Ausschluß einzelner Herkünfte

Eine angegebene Priorität meint immer die genannte oder eine höhere. Wenn jedoch vor der Priorität ein Gleichheitszeichen (=) steht, so ist nur die genannte Priorität gemeint.

Aktionen

Eine Aktion ist immer eine Weiterleitung einer Nachricht. Es gibt vier verschiedene Arten, wie solche Nachrichten weitergeleitet werden können:

1. **Ausgabe der Nachricht in eine Datei.**
   Dazu muß als Aktion der Dateiname mit absolutem Pfad (mit führendem Slash) angegeben werden. Normalerweise wird nach jedem Schreibvorgang des Syslog-Daemons eine Synchronisation des Dateisystems durchgeführt, weil sonst evt. Nachrichten bei einem Absturz nicht mehr physikalisch in die Datei geschrieben werden. Das ist allerdings eine sehr zeitaufwendige Aktion, daher gibt es die Möglichkeit, diese Synchronisation zu übergehen. Dazu wird dem absoluten Pfadnamen ein Bindestrich (-) vorangestellt.
2. **Weiterleitung der Nachricht an einen Syslog-Daemon eines anderen Rechners im Netz.**
   Dazu muß als Aktion der Rechnername des Rechners angegeben werden, an dessen Syslog-Daemon die Messages geschickt werden sollen. Dem Rechnernamen muß ein Klammeraffe (@) vorangestellt werden. Damit der angesprochene Syslog-Daemon auf dem anderen Rechner auch die Meldungen annimmt, muß er mit der Kommandozeilenoption **-r** (remote) gestartet worden sein.
3. **Ausgabe der Nachricht auf den Bildschirm von bestimmten Usern**
   Durch die Nennung des Usernamens (oder einer durch Kommas getrennten Liste von Usernamen) wird die Nachricht auf dem Bildschirm dieser User angezeigt, sofern sie eingeloggt sind.
4. **Ausgabe der Nachricht auf den Bildschirm aller eingeloggten User**
   In diesem Fall steht einfach ein Sternchen (*) im Aktionsfeld.

Ein paar Beispiele

Wie können nun solche Regelzeilen aussehen? Nehmen wir an, wir wollen dafür sorgen, daß alle Meldungen des Systems in die Datei /var/log/messages geschrieben werden. Das erledigt die Zeile

```
*.*          /var/log/messages
```

Wollen wir aber die Meldungen des Login-Systems, die eventuell Passwörter in Klarschrift enthalten ausschließen, so können wir das mit der Priorität none erledigen:

```
*.*;authpriv.none          /var/log/messages
```

Wir können dafür sorgen, daß Meldungen des Kernels, ab der Priorität warn immer auf den Bildschirm von root geschickt werden, sofern er eingeloggt ist. Falls der User foo eingeloggt ist, soll er die Meldungen auch bekommen:

```
kern.warn      root,foo
```

Auch Gerätedateien sind Dateien. Schreiben wir alle Meldungen des Kernels die mindestens die Priorität warn haben und alle Meldungen ab error aller anderen Kategorien auf die Konsole 10 (/dev/tty10). Die Meldungen von authpriv lassen wir aber wieder weg. Auf Konsole 10 können sie dann auf dem Bildschirm mitgelesen werden:

```
kern.warn;*.err;authpriv.none   /dev/tty10
```

Die nächsten drei Zeilen schreiben

- Nur die kritischen Meldungen des News-Systems in die Datei /var/log/news/news.crit
- Nur die Fehlermeldungen des Newssystems in die Datei /var/log/news/news.err
- die bemerkenswerten Meldungen des News Systems in die Datei /var/log/news/news.notice

Alle Dateien werden nicht sofort synchronisiert (-)

```
news.=crit                      -/var/log/news/news.crit
news.=err                       -/var/log/news/news.err
news.=notice                    -/var/log/news/news.notice
```

Wenn wir einen zentralen Rechner haben, nennen wir ihn admhost, der das Logbuch für alle Rechner im Netz mitschreiben soll, so müssten alle Rechner im Netz die Zeile

```
*.*;authpriv.none          @admhost
```

in der syslog.conf eingetragen haben. Der syslogd auf admhost müsste dazu mit der Option -r gestartet sein.

Durch eine durchdachte Einteilung kann sich ein Systemverwalter viel Arbeit ersparen. Es ist ja möglich, daß alle denkbaren Ereignisse in verschiedene Dateien geschrieben werden, die dann schnell und effizient durchsucht werden können. Werden z.B. alle Meldungen ab der Priorität error (oder für vorsichtige Systemverwalter besser warn) in eine spezielle Datei geschrieben, so kann der Systemverwalter diese

Datei regelmäßig überprüfen und muß sich nicht lange durch einen Berg von alltäglichen Meldungen kämpfen, um kritische Meldungen zu entdecken.

Verfolgen von Logdateien in Echtzeit

Manchmal ist es notwendig, Logdateien während einer bestimmten Aktion zu beobachten, um bestimmte Meldungen zu verfolgen. Um eine solche Beobachtung in "Echtzeit" zu ermöglichen gibt es zwei verschiedene Möglichkeiten. Die erste und am häufigsten angewandte Methode wird mit dem Befehl tail realisiert. tail kennt den Parameter -f, der eben diese Fähigkeit zur Verfügung stellt. Der Befehl

```
tail -f /var/log/messages
```

zeigt die letzten 10 Zeilen der Datei `/var/log/messages` am Bildschirm an. Statt aber dann zur Eingabeaufforderung zurückzukehren verharrt tail (durch den Parameter -f - forever) und wartet, ob die Datei "wächst", also neue Zeilen angefügt bekommt. Sobald neue Zeilen in die Datei geschrieben werden, werden sie auch auf dem Bildschirm ausgegeben.

Somit kann die Datei also "beim Wachsen beobachtet werden", erst ein Strg-C beendet diese fortlaufende Beobachtung. Diese Fähigkeit ist natürlich ideal geeignet, Log-Dateien zu beobachten, während ein Fehler gesucht wird.

Die zweite angesprochene Möglichkeit bietet das Programm less, der uns wohlbekannte Pager. less kennt auch die Möglichkeit einer Datei beim Wachsen zuzusehen, dazu muß der Befehl **F** eingegeben werden. Auch dieser Modus muß mit einem Strg-C abgebrochen werden.

Automatische Rotation der Logbuchdateien

Logbuchdateien werden zwangsläufig ziemlich schnell wachsen. Im normalen Betriebsablauf eines Linux-Rechners, der nicht nur untätig in der Ecke steht können pro Tag problemlos 50 bis 100 Kilobyte Meldungen entstehen. Da bietet es sich doch an, diese Logbücher hin und wieder zu löschen oder zu archivieren.

Im professionellen Umfeld sollten Logdateien niemals gelöscht werden. Sie sollten besser archiviert und abgespeichert werden, am besten zusammen mit dem Datum der Speicherung oder noch besser mit dem Start- und Enddatum. Einmal archiviert kann eine solche Datei dann komprimiert werden, Programme wie gzip sind bestens geeignet, Textdateien wie Logbücher mit immer wiederkehrenden Textpassagen auf eine sehr geringe Größe zu bringen. Eine Logdatei kann so tatsächlich weniger als ein Zehntel ihrer ursprünglichen Größe bekommen.

Verschiedene Linux-Distributionen gehen hier verschiedene Wege. Aber es zeichnet sich ein Standard ab, der dazu geeignet ist, zu große Logbuchdateien regelmäßig zu komprimieren und optional auch regelmäßig per Mail zu verschicken. Das Programm, das diesen Job übernimmt heißt logrotate und wird regelmäßig von cron (siehe Kapitel 1.111.4) aufgerufen.

logrotate bezieht seine Informationen aus einer (oder mehrerer) Konfigurationsdateien, die ihm als Kommandozeilenparameter mitgegeben wurden. Das genaue Format dieser Dateien ist auf der Handbuchseite erklärt.

Vielleicht sollte der Begriff Rotation noch einmal genauer erklärt werden. Er stammt aus der Welt des Backup, in der meist mehrere Bänder für ein Backup verwendet werden. Wenn wir beispielsweise jeden Freitag ein Backup machen, dann benutzen wir nicht jedesmal das selbe Band, sondern wir haben vielleicht 5 Bänder. Nun "rotieren" wir diese Bänder, das heißt, das wir in der ersten Woche das erste Band benutzen, in der zweiten das zweite usw. In der sechsten Woche benutzen wir dann also wieder das erste Band. So stehen uns immer die letzten 5 Wochen zur Verfügung.

Genauso läuft es mit den Logdateien. Das Programm **logrotate** rotiert die Logdateien, das heißt die gegenwärtige Logdatei wird - wenn ein bestimmtes Kriterium (Größe, Tag, Woche, Monat) erfüllt ist - in eine andere Datei kopiert (und optional komprimiert). Die Orginal-Logdatei wird dann gelöscht und neu angelegt. Die Anzahl der Rotationen (analog zu der Anzahl der verwendeten Bänder aus dem Backup-Beispiel) ist einstellbar. Das würde bedeuten, daß nach beispielsweise fünf Rotationen die älteste Datei gelöscht wird und statt dessen die neue Rotationsdatei angelegt wird. So haben wir die Sicherheit, daß immer nur eine begrenzte Anzahl von Dateien (und eine begrenzte Menge Speicherplatz) verwendet wird.

Optional kann man es aber auch so einrichten, daß die Logdatei, bevor sie dann endgültig gelöscht wird, an eine anzugebende E-Mail-Adresse gemailt wird.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

1.111.4

Automatisieren von Systemadministrationsaufgaben durch Planen von zukünftig laufenden Jobs

---

Beschreibung: Prüfungskandidaten sollten in der Lage sein, **cron** oder **anacron** zu verwenden, um Prozesse in regelmäßigen Intervallen ausführen zu lassen, und **at** zu benutzen, um Jobs zu einer bestimmten Zeit auszuführen. Dies beinhaltet das Verwalten von **cron** und **at** Jobs und die Konfiguration von Benutzerzugang zu **cron** und **at** Diensten.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **at**
- **atq**
- **atrm**
- **crontab**
- `/etc/anacrontab`
- `/etc/at.deny`
- `/etc/at.allow`
- `/etc/crontab`
- `/etc/cron.allow`
- `/etc/cron.deny`
- `/var/spool/cron/*`

Systemverwaltungsaufgaben bestehen zu einem großen Teil aus immer wiederkehrenden Routinejobs. Diese Routine kann durch Kommandos vereinfacht werden, die automatisch immer wieder in bestimmten Intervallen oder zu bestimmten Zeiten ausgeführt werden. Für diese Aufgabe existiert ein eigener Daemon, der **cron**-Daemon. Er kann auf verschiedene Weisen konfiguriert werden, die hier genauer

beschrieben werden sollen.

Manche Aufgaben sind aber nicht regelmäßig auszuführen, sondern nur genau einmal, aber zu einem ganz bestimmten Zeitpunkt. Diese Aufgabe wird vom **at**-Daemon gelöst, der genau dazu gemacht ist.

Der cron-Daemon und seine Konfiguration

Der Cron-Daemon überwacht verschiedene Dateien und Verzeichnisse, in denen Anweisungen liegen, die in regelmäßigen Abständen ausgeführt werden sollen. Diese Anweisungen werden Cron-Tabellen oder eben **crontabs** genannt.

Cron läd beim Start all die Dateien und Verzeichnisse, die er überwacht in den Arbeitsspeicher und überprüft jede Minute einmal, ob darin Jobs enthalten sind, die in der aktuellen Minute ausgeführt werden sollen. Wenn ja, so führt er sie aus. Außerdem überprüft cron jede Minute, ob sich an den Dateien oder Verzeichnissen etwas geändert hat und - falls ja - übernimmt er diese Änderungen im Speicher.

Es gibt also mehrere Möglichkeiten, dem Cron-Daemon einen Job zur Ausführung zu übergeben. Die einzelnen Möglichkeiten sind:

User-Crontabs

Jeder User des Systems kann eine eigene Crontab (also eine Datei mit Anweisungen für Cron) erstellen und bearbeiten. Die Jobs, die darin aufgeführt sind werden von Cron unter der Userkennung des Users ausgeführt, um dessen Crontab-Datei es sich handelt. Die User-Crontabs werden nicht direkt mit einem Editor verändert, sondern mit dem Befehl crontab(1).

Systemweite Crontab-Datei

Im Verzeichnis /etc existiert eine Datei `crontab`, die ebenfalls Cronjobs definiert. Das Format dieser Datei unterscheidet sich etwas von dem der Usercrontabs - siehe weiter unten...

cron.d Verzeichnis

Im Verzeichnis /etc kann ein Verzeichnis cron.d existieren, das Dateien im selben Format wie /etc/crontab enthalten darf. Auch diese Dateien werden von cron überwacht und darin enthaltene Befehle werden ausgeführt.

cron.{hourly,daily,weekly,monthly} Verzeichnisse

Ebenfalls im /etc/verzeichnis können Verzeichnisse der Namen cron.hourly, cron.daily, cron.weekly und cron.monthly existieren. Diese Verzeichnisse enthalten **nicht** crontabs, sondern Shellscripts, die entsprechend stündlich, täglich, wöchentlich oder monatlich ausgeführt werden.

Das Format der User-Crontabs

Das Format der Crontabs der User ist etwas speziell, hier das Grundprinzip: Crontab-Einträge sind entweder Definitionen von Shellvariablen oder zeitgesteuerte Cron-Befehle. Leere Zeilen oder Zeilen, die mit einem # beginnen werden ignoriert. Das Kommentarzeichen (#) darf aber nur am Anfang der Zeile stehen mitten in einer Zeile wird es als Teil des Kommandos interpretiert. Die

Definition einer Shellvariable hat immer die Form:

```
Name=Wert
```

Einige Variablen werden von Cron selbst gesetzt. `SHELL` wird auf `/bin/sh` gesetzt, `HOME` und `LOGNAME` werden den Informationen über den User aus der /etc/passwd entnommen. Die Variablen `SHELL` und `HOME` dürfen verändert werden, `LOGNAME` nicht.

Normalerweise schickt cron alle Ausgaben von ausgeführten Kommandos per Mail an den User, der das Cron-Kommando ausführen liess. Die Variable `MAILTO` kann diese Eigenschaft verändern. Wenn sie den Namen eines Users enthält, so werden ihm und nicht dem ursprünglichen Cron-User die Mail geschickt. Ist die Variable `MAILTO` definiert, aber leer, so wird gar keine Mail verschickt.

Alle anderen Zeilen der Crontabs, die also nicht Variablen definieren, beschreibt Kommandos, die zu bestimmten Zeiten ausgeführt werden sollen. Das grundlegende Format dieser Zeilen ist:

```
Minute  Stunde  Tag  Monat  Wochentag          Kommando
```

Die einzelnen Felder definieren also die Zeit, zu der das angegebene Kommando ausgeführt werden soll. Jedes der fünf Zeitfelder darf statt einem Wert auch ein Sternchen (`*`) enthalten, das als Jokerzeichen gilt und für jeden beliebigen Wert steht.

Die jeweiligen Zeitfelder dürfen folgende Werte aufnehmen:

Minute	0-59
Stunde	0-23
Tag	1-31
Monat	1-12
Wochentag	0-7 (0 und 7 bedeutet Sonntag)

Auch Bereichsangabe sind erlaubt. Bereiche werden mit Bindestrich definiert. So würde die Angabe `8-11` im Stundenfeld die Stunden 8, 9, 10 und 11 meinen.

Auch Listen sind erlaubt. Listen werden durch Kommata aufgebaut. So würde die Angabe `1,2,5` im Wochentagfeld die Tage Montag, Dienstag und Freitag meinen.

Auch Kombinationen von Listen und Bereichen sind möglich. Die Angabe `1,3,15-19,30` spricht also die Zahlen 1, 3, 15, 16, 17, 18, 19 und 30 an.

Im Zusammenhang mit Bereichen können sogar Schrittweiten angegeben werden. Die Angabe `0-23/2` im Stundenfeld meint also die Stunden 0-23 aber in der Schrittweite 2 Stunden. Also 0, 2, 4, 6, 8, 10, ... Um diesen Schritt zu vervollständigen kann z.B. die Angabe *alle zwei Stunden"* einfach mit `*/2` ausgedrückt werden. Gleiches gilt für alle anderen Zeitfelder.

Das Wochentagfeld und das Tag-Feld sind - wenn beide angegeben sind - aditiv gemeint. Die Angabe

```
   * * 13 * 5       ...
```

meint also nicht alle Freitag der 13., sondern alle Freitage **und** alle 13. des Monats.

Die letzte Angabe in einer Crontab-Zeile ist der Befehl, der zur gegebenen Zeit ausgeführt werden soll. Er wird von /bin/sh oder der in der Variable `SHELL` angegebenen Shell ausgeführt.

Eine Beispiel-Crontab könnte also folgendermaßen aussehen - mit Kommentaren zum besseren Verständnis:

```
# Variablendefinition
SHELL=/bin/bash
PATH=/usr/bin:/bin

# Cronbefehle
# Das Programm foo wird täglich um 23 Uhr 56 ausgeführt
56 23 * * *         foo

# Das Programm backup wird jeden Freitag um 17 Uhr 30 ausgeführt
30 17 * * 5         backup

# Alle 2 Stunden zwischen 6 und 23 Uhr wird das Programm fetchmail
# ausgeführt
0 6-23/2 * * *      fetchmail

# Am ersten jedes Monats wird um 8 Uhr morgens das Programm bar ausgeführt
0 8 1 * *           bar
```

Wenn der User, dem der crontab gehört die UserID 0 hat, also der Systemverwalter ist, dann kann eine Crontab-Zeile mit einem Bindestrich beginnen. In diesem Fall wird cron die ausgeführte Aktion nicht an den Syslog-Daemon weiterleiten. In jedem anderen Fall wird jede Cron-Aktion ins Logbuch übernommen und kann dort im Nachhinein überprüft werden.

User Crontabs verwalten mit crontab(1)

Die Crontabs der User sind zwar im Dateisystem als einzelne Dateien vorhanden (/var/spool/cron/tabs/*Username*), können aber dort nicht

direkt editiert werden, weil sie nicht dem User gehören. Damit ein User trotzdem eigene Jobs definieren kann, steht ihm das Programm crontab zur Verfügung.

Dieses Programm erlaubt einem User, die folgenden Aktionen:

Anlegen einer neuen Crontab-Datei

Wenn ein User mit seinem Lieblings-Editor eine Datei erstellt hat, die seine Crontab-Befehle im oben beschriebenen Format enthält, und der User bisher keine Crontab besaß oder die bisherige Crontab durch diese neue Datei ersetzen will, dann kann er diese Datei mit dem Befehl

```
crontab Dateiname
```

zu seinem Crontab machen.

Ansehen der aktuellen Crontab

Mit dem Befehl

```
crontab -l
```

kann sich ein User seine Crontab am Bildschirm anzeigen lassen.

Löschen der Crontab-Datei

Wenn ein User seine Crontab-Datei löschen will, kann er dazu den Befehl

```
crontab -r
```

benutzen. Die ganze Crontab-Datei des Users wird dadurch gelöscht - alle bisherigen Jobs sind damit automatisch auch gelöscht. Die Veränderung wird sofort aktiv.

Einträge in der Crontab editieren

Mit dem Befehl

```
crontab -e
```

kann ein User seine Crontab-Datei editieren. Dazu wird der Editor aufgerufen, der in der Umgebungsvariable `VISUAL` bzw. `EDITOR` definiert ist - meist also der vi. Alle Veränderungen, die vom User dort vorgenommen werden, werden sofort nach dem Sichern und Verlassen des Editors aktiv.

Der Systemverwalter kann dem Programm crontab mit der Option `-u username` auch noch den User mitgeben, dessen Eintrag bearbeitet werden soll. Er kann also die Einträge jedes Users erstellen, löschen, ansehen und editieren. Ein Normaluser darf

selbstverständlich nur seine eigene Datei bearbeiten.

Das Format der Crontabs in /etc

Die Datei `/etc/crontab` und alle Dateien (beliebigen Namens) im Verzeichnis `/etc/cron.d` werden auch als Crontabs gewertet und vom Cron-Daemon überwacht und ausgeführt. Im Wesentlichen entspricht das Format genau dem der User-Crontabs, mit einer Ausnahme. Zwischen dem fünften Zeitfeld (Wochentag) und dem Befehl wird hier noch ein Username angegeben, unter dessen ID der Befehl ausgeführt werden soll. Das Format ist also:

```
Minute Stunde Tag Monat Wochentag Username Kommando
```

Alle anderen oben genannten Eigenschaften bleiben unverändert, also auch die Bereichs- und Listenangabe der Zeitfelder.

Diese Dateien dürfen nur vom Systemverwalter angelegt und editiert werden, er kann aber durch das zusätzliche Userfeld bestimmen, unter wessen UserID der Befehl ausgeführt werden soll.

Die Verzeichnisse in /etc

Neben der Datei /etc/crontab und dem Verzeichnis /etc/cron.d existieren in vielen Linux-Distributionen (RedHat, Debian, SuSE,...) auch noch die Verzeichnisse

```
/etc/cron.hourly
/etc/cron.daily
/etc/cron.weekly
/etc/cron.monthly
```

Diese Verzeichnisse enthalten **keine** Crontabs, sondern Programme, die entsprechend stündlich, täglich, wöchentlich oder monatlich einmal ausgeführt werden sollen. In der Regel handelt es sich bei diesen Programmen um Shellscripts, die dann wiederum bestimmte Programme aufrufen. Standardeinstellungen, wann diese Programme ausgeführt werden sind:

- **cron.hourly** - Stündlich jeweils um *:30 Uhr
- **cron.daily** - Täglich jeweils um 0:00 Uhr
- **cron.weekly** - Wöchentlich jeweils Samstags um 0:00 Uhr
- **cron.monthly** - Monatlich jeweils am ersten eines Monats um 0:00 Uhr

In der Regel sind diese Verzeichnisse für regelmäßige Kommandos gemacht, die schon beim Installieren bestimmter Pakete festgelegt sind. Zumeist tragen die Scripts innerhalb dieser Verzeichnisse die Namen der Pakete, die sie installiert haben.

Cron-Dienste für User erlauben oder verbieten

Normalerweise kann jeder User eines Linux-Systems seine eigene Crontab besitzen und beliebige Aufgaben zu beliebigen Zeiten damit erledigen lassen. Das kann er aber nur - wie oben gesehen - durch die Verwendung des Programms crontab. Dieses Programm bietet aber auch eine Art Zugriffsschutz, mit dem der Systemverwalter einzelnen Usern die Verwendung dieses Programms erlauben oder verbieten kann.

Wie so oft, gibt es hier zwei verschiedene Lösungsansätze. Entweder dürfen nur die User Crontab benutzen, die die ausdrückliche Erlaubnis besitzen, oder alle User dürfen crontab benutzen, denen es nicht explizit verboten wurde.

Dazu kann der Systemverwalter im Verzeichnis `/etc` mit zwei Dateien arbeiten, die er dort anlegen muß. Die folgenden Möglichkeiten existieren:

Existiert die Datei `/etc/cron.allow`, dann dürfen nur die User mit crontab arbeiten, die in dieser Datei aufgelistet sind.

Wenn die Datei `/etc/cron.allow` **nicht** existiert, stattdessen aber die Datei `/etc/cron.deny`, dann dürfen alle User mit crontab arbeiten, außer denen, die in der Datei `cron.deny` aufgelistet sind.

Beide Dateien sind reine Textdateien, die pro Zeile einen Usernamen enthalten dürfen.

In älteren Versionen von Cron lagen diese beiden Dateien nicht in /etc sondern hießen `/var/spool/cron/allow` und `/var/spool/cron/deny`. Die Funktionsweise war aber die selbe. Auf manchen älteren Linux-Versionen finden sich noch diese älteren Einstellungen. Nach der Revision hat LPI aber auf die neuen Dateien umgestellt.

anacron

Anacron ist wie cron ein periodischer Kommandoscheduler. Er führt bestimmte Befehle in Intervallen aus, die aber - im Gegensatzt zu cron - nur in Tagen angegeben werden. Der wesentliche Unterschied ist aber der, daß anacron nicht annimmt, daß der Rechner Tag und Nacht läuft.

Wenn cron einen bestimmten Job zu einer Zeit ausführen soll, zu der der Rechner nicht angeschalten ist, dann verfällt dieser Job. Oft ist es zum Beispiel so, daß die täglichen cron-Jobs um 0:00 Uhr ausgeführt werden sollen oder die wöchentlichen am Sonntag (Tag 0). Wenn ein Rechner aber weder Sonntags, noch Nachts läuft, werden diese Jobs niemals ausgeführt. Meist handelt es sich bei diesen Jobs um Verwaltungsaufgaben, wie der Auffrischung bestimmter Systemdatenbanken (locate und man) oder der Rotation der Logdateien.

Anacron kann dieses Problem lösen. Man kann einfach diese Jobs in Intervallen von einem, sieben oder 30 Tagen starten um tägliche, wöchentliche oder monatliche Ausführung zu erzwingen. Anacron führt seine Jobs aus, wenn der letzte Ablauf eines Jobs länger als die genannte Intervallzeit in Tagen her ist. Somit wird ein Job auch dann ausgeführt, wenn der Rechner das nächste Mal angeschalten wird und nicht - wie bei cron - wenn das nächste Wochen- oder Monatsende erreicht ist.

Jedesmal, wenn anacron aufgerufen wird, ließt es seine Konfigurationsdatei (/etc/anacrontab) in der seine Jobs mit den Perioden eingestellt werden. Wenn ein Job die letzten *n* Tage nicht ausgeführt wurde, wobei *n* die angegebene Periode dieses Jobs ist, führt anacron ihn aus. Anacron erzeugt dann einen Eintrag in einer speziellen Zeitmarkendatei, die es für jeden Job erstellt, so daß es weiß, wann der Job zuletzt ausgeführt wurde. Wurden alle Kommandos ausgeführt, wird anacron beendet.

Anacron ist also kein Daemon, der die ganze Zeit läuft, sondern muß entweder über init-Scripts oder cron regelmäßig gestartet werden.

Die Konfigurationsdatei von anacron ist sehr einfach aufgebaut. Sie enthält entweder Variablenzuweisungen, die der Umgebung zugewiesen werden, in der der entsprechende Befehl ausgeführt werden soll, oder Befehlszeilen der Form

```
Periode   Verzögerung   Job-Identifikation   Kommando
```

Die Periode ist eine Zahl, die die Anzahl von Tagen angibt, die zwischen der letzten Ausführung des Jobs und der nächsten Ausführung mindestens vergangen sein müssen. Die Verzögerung ist ein Wert in Minuten, der angegeben wird, damit nicht alle anacron-Jobs gleichzeitig gestartet werden und so den Rechner unnötig strapazieren. Somit können die verschiedenen Jobs leicht zeitversetzt gestartet werden. Die Job-Identifikation ist ein beliebiges Wort, daß alle Zeichen außer Leerzeichen und Slashs enthalten darf. Mit Hilfe dieses Wortes wird der Dateiname der Zeitmarkendatei erstellt (das ist der Grund für die verbotenen Slashs). Am Ende der Zeile steht dann der auszuführende Befehl. Eine simple `/etc/anacrontab` Datei könnte also folgendermaßen aussehen:

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

 1    1    updatedb    updatedb
 7   10    logrotate   logrotate /etc/logrotate.conf
31   15    tmpclean    rm /tmp/*
```

Jeden Tag, eine Minute nachdem anacron gestartet wurde, wird der Befehl **updatedb** ausgeführt. Die Zeitmarke wird in eine Datei heißt ebenfalls updatedb.

Alle 7 Tage jeweils 10 Minuten nach dem Start von anacron wird der Befehl **logrotate /etc/logrotate.conf** ausgeführt. Die Zeitmarkendatei heißt nur logrotate.

Alle 31 Tage werden alle Dateien im /tmp Verzeichnis gelöscht, die Zeitmarkendatei heißt tmpclean. Der Befehl wird 15 Minuten nach Aufruf von anacron gestartet.

Hat anacron all diese drei Befehle abgearbeitet, dann beendet sich das Programm wieder.

Das at-Spoolsystem

Wenn ein Kommando nicht regelmäßig ausgeführt werden soll, sondern nur zu einem ganz bestimmten Zeitpunkt, dann können wir das mit dem at-Kommando erledigen. At ist ein typisches Unix-Spoolsystem, das Aufträge entgegennimmt, in eine Warteschlange stellt und zur gegebenen Zeit dann ausführt. Die Architektur ist wie bei allen anderen Unix-Spoolsystemen ausgeführt:

- Der Befehl **at** gibt Aufträge in die Warteschlange
- Der Befehl **atq** zeigt alle Aufträge, die in der Warteschlange stehen.
- Der Befehl **atrm** löscht einzelne Aufträge aus der Warteschlange.
- Der Daemon **atd** arbeitet die Warteschlange ab. In älteren Versionen von Linux findet sich kein eigener Daemon, in diesem Fall wurde von cron jede Minute einmal das Programm **atrun** aufgerufen, das die Warteschlange nach Aufträgen durchsuchte, die im Augenblick ausgeführt werden sollten und - sofern gefunden - diese Jobs ausführte.

Um mit **at** einen Job in Auftrag zu geben, muß einfach nur der Befehl

```
at Zeit
```

aufgerufen werden. Danach kann über die Standard-Eingabe die gewümschte Befehlszeile eingegeben werden, die zu der Angegebenen *Zeit* ausgeführt werden soll. Um die Eingabe abzuschließen, wird in einer eigenen Zeile das Dateiendezeichen Strg-D eingegeben. Alternativ kann eine Datei angelegt werden, die die Befehle enthält, die zu der bestimmten Zeit ausgeführt werden sollen. Dann kann **at** entweder mit

```
at Zeit < Datei
```

oder mit

```
at -f Datei Zeit
```

aufgerufen werden.

Die verschiedenen Formen der Zeitangaben enthehmen Sie bitte der Handbuchseite.

Die Ausgaben der Kommandos, die von **at** zu bestimmten Zeiten ausgeführt wurden, werden dem User als Mail zugeschickt, der den Auftrag losgeschickt hatte.

Wie schon bei cron, so können wir auch bei at festlegen, wer mit diesem Programm arbeiten darf bzw. wer nicht. Dazu dienen die Dateien `/etc/at.allow` und `/etc/at.deny`. Existiert die Datei /etc/at.allow, so dürfen nur die User mit **at** arbeiten, die dort aufgeführt sind (ein User pro Zeile). Existiert diese Datei nicht, aber die Datei /etc/at.deny, so dürfen alle User **at** benutzen, außer den Usern, die in at.deny

aufgelistet sind. Existieren beide Dateien nicht, so darf nur der Systemverwalter mit **at** arbeiten.

1.111.5

Aufrechterhaltung einer effektiven Datensicherungsstrategie

Beschreibung: Prüfungskandidaten sollten in der Lage sein, eine Sicherungsstrategie zu planen und Dateisysteme automatisch auf verschiedene Medien zu sichern. Die Tätigkeiten beinhalten das Sichern einer rohen Partition in eine Datei oder umgekehrt, das Durchführen teilweiser und manueller Backups, das Überprüfuen der Integrität von Backupdateien und teilweises oder vollständiges Wiederherstellen von Backups.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **cpio**
- **dd**
- **dump**
- **restore**
- **tar**

Dieser Abschnitt der LPI102 Prüfung erfordert nicht nur das Wissen um die konkrete Syntax einzelner Befehle, sondern auch die grundsätzliche Architektur von Backup-Strategien. Backups werden heute mit den unterschiedlichsten Programmen erstellt und auf unterschiedlichsten Medien abgelegt. Die genaue Syntax aller hier denkbarer Programme ist sicherlich etwas, was den Rahmen sowohl der LPI-Prüfung, als auch dieser Darstellung sprengen würde. Die wichtigsten Bordmittel werden aber hier genau besprochen.

Vorüberlegungen

Grunsätzlich ist zu sagen, daß ein Backup auf jedem Rechner notwendig ist, dessen Dateien nicht rein statisch sind. Ein Rechner, der etwa nur als Router ins Internet Dienst tut, muß sicherlich kein regelmäßiges Backup ausführen, hier würde die einmalige Sicherung nach der abschließenden Konfiguration ausreichen. Jeder Rechner, auf dem gearbeitet wird, dessen Dateien also nicht statisch sind, muß aber

regelmäßige Backups erledigen, um die getane Arbeit zu sichern. Wie im einzelnen die Backups aussehen, was gesichert wird, das muß gründlich geplant werden.

Ein wesentliches Kriterium für ein Backup ist die Frage, auf welches Medium die gesicherten Daten gespeichert werden sollen. Es bietet sich grundsätzlich an, Wechselmedien zu benutzen, die an einem anderen Ort gelagert werden können. Wird ein Backup nur auf eine zweite Platte im Rechner gemacht, so haben wir vielleicht eine Absicherung gegen eine physikalische Zerstörung der Platte (Headcrash), Feuerschäden, Wasserschäden oder Diebstahl des Rechners würde diese Form des Backups nicht abdecken, die Daten wären verloren.

Wechselmedien stehen in verschiedenen Formen zur Verfügung. Neben Wechselplatten, die auf der Festplattentechnik beruhen, sind es meist Bandlaufwerke, die für Backups verwendet werden. Hier spielen drei wesentliche Familien eine Rolle, Viertelzoll-Cartridges (QIC), DAT-Cassetten und Videobänder.

Die Speicherkapazität eines Backup-Mediums ist von großer Bedeutung. Muß das Medium während des Backups gewechselt werden, weil es nicht genügend Speicherplatz aufweist, so kann ein Backup nicht vollständig automatisiert werden. Der Aufwand (und damit auch die Hemmschwelle) ist wesentlich größer, als bei einem Medium, das das ganze Backup in einem Rutsch aufnehmen kann.

Die Frage, was überhaupt gesichert werden muß, ist die nächste Überlegung. Wir unterscheiden zwischen verschiedenen Backup-Strategien. Insbesondere zwei Begriffe werden hier immer wieder fallen: *volles Backup* und *inkrementelles Backup*.

Ein Vollbackup ist ein Backup, das alle Daten, die gesichert werden müssen, in einem Arbeitsgang auf ein Medium speichert. Ein inkrementelles Backup speichert im Gegensatz dazu nur die Daten, die sich seit dem letzten Backup verändert haben. **Ein inkrementelles Backup ist nur in Zussammenarbeit mit einem Vollbackup möglich!**. Abhängig von der Sicherheitsanforderung kann z.B. überlegt werden, jeden Tag der Woche ein inkrementelles Backup vorzunehmen, und Freitags ein Vollbackup. Das bedingt aber, daß alle Medien, die diese Backups aufgenommen haben, zu einer Wiederherstellung (restore) der Daten benötigt werden.

Nehmen wir ein Beispiel: Wir haben folgenden Backup-Plan:

- Montag - inkrementelles Backup - Band 1
- Dienstag - inkrementelles Backup - Band 2
- Mittwoch - inkrementelles Backup - Band 3
- Donnerstag - inkrementelles Backup - Band 4
- Freitag - volles Backup - Band 5

Es ist jetzt Mittwoch, wir haben einen Headcrash, der uns zwingt, alle Daten auf eine neue Platte aufzuspielen. Wie müssen wir vorgehen?

Die Reihenfolge des Wiederaufspielens ist hier wichtig. Wir brauchen:

- Das Band mit dem letzten Vollbackup (Band 5)
- Die Bänder der letzten inkrementellen Backups vom letzten Vollbackup bis zum Eintreten der Katastrophe. Also in unserem Beispiel

die Bänder von Montag und Dienstag (Band 1 und 2)

Das Wiederaufspielen muß jetzt in genau der Reihenfolge passieren, wie das Backup selbst. Zunächst spielen wir die Daten des Vollbackups wieder auf die Platte, dann die des Montag-Bandes und erst dann die des Dienstag-Bandes. Nur so stellen wir sicher, daß wir nicht alte Versionen von Dateien aufspielen, die die neueren wieder überschreiben...

Das ist ein aufwendiges Verfahren, das in der Praxis selten vorkommt. Um variabel mit Kombinationen von vollen und inkrementellen Backups arbeiten zu können, werden sogenannte Backup-Level definiert. Dabei wird ein Vollbackup immer Level 0 genannt, ein inkrementelles Backup hat dann Levelnummern größer als 0. Jedes inkrementelle Backup, egal welche Levelnummer es trägt, sichert immer alle Daten, die sich seit dem letzten Backup mit einer Nummer niedriger verändert haben. Ein Beispiel:

- **Erster Montag des Monats**
  Level 0 (Vollbackup)
- **Jeder andere Montag**
  Level 1 (wöchentliches inkrementelles Backup relativ zu Level 0)
- **Dienstag**
  Level 2 (tägliches inkrementelles Backup relativ zu Level 1)
- **Mittwoch**
  Level 2 (tägliches inkrementelles Backup relativ zu Level 1)
- **Donnerstag**
  Level 2 (tägliches inkrementelles Backup relativ zu Level 1)
- **Freitag**
  Level 2 (tägliches inkrementelles Backup relativ zu Level 1)

In diesem Fall brauchen wir für ein Wiederherstellen (restore) der Daten immer nur das aktuellste Band von jedem Level. Da jeder Level über 0 immer alle Daten sichert, die seit dem letzten Backup des vorgehenden Levels verändert wurden hat z.B. das Donnerstagsband immer auch die Daten gespeichert, die am Mittwoch verändert wurden. Es speichert ja alle Daten, seit dem letzten Backup mit niedrigerer Nummer, also seit Montag.

Zum Wiederherstellen der Daten nach einer Katastrophe am Freitag brauchen wir also

- Das letzte Vollbackup (Level 0)
- Das letzte inkrementelle Wochenbackup (Level 1)
- Das letzte inkrementelle Tagesbackup vom Donnerstag (Level 2)

Das Wiederaufspielen der Daten erfolgt jetzt in genau dieser Reihenfolge; zuerst das Vollbackup, dann Level 1 und erst dann Level 2. Jetzt ist garantiert, daß immer die aktuellsten Versionen der Dateien wiederaufgespielt sind.

Planung von Backups

Bei der Planung einer Backup-Strategie müssen ein paar Fragen durchüberlegt werden, deren Beantwortung das Backup leichter planbar machen. Wenn wir schon bei der Planung der Architektur des ganzen Systems auf solche Kriterien achten, können wir uns das Leben wesentlich erleichtern.

Die wichtigste aller Fragen ist die, welche Dateien überhaupt gesichert werden müssen. Bei einer vernünftigen Planung eines Systems können einige Teilbereiche des Systems völlig statisch angelegt werden. Das kann seinen Ausdruck dann darin finden, daß ganze Partitionen Read-Only gemountet werden, so daß selbst versehentliche Änderungen nicht möglich sind. Ein Beispiel hierfür ist die Tatsache, daß das `/usr`-Verzeichnis geradezu dafür gemacht ist, nur lesend eingehängt zu werden. Solche Bereiche können wir einmal - direkt nach der Installation - sichern und müssen sie dann nie wieder in unsere Backup-Überlegungen aufnehmen...

Eine weitere Überlegung beim Aufbau des Systems ist die, wo die User eines Systems Daten ablegen dürfen und wo nicht. Gibt es hier strenge Richtlinien, die etwa festlegen, daß User ihre Daten grundsätzlich nur in ihren Home-Verzeichnissen ablegen dürfen und diese Homeverzeichnisse auf einer separaten Partition liegen, dann ist durch das Backup dieser einen Partition schon der wesentliche Datenbesatz gesichert. Die Daten der User sind auf alle Fälle nicht verloren.

Ein weiterer wichtiger Bereich, der immer Teil des Backups sein sollte, ist der Bereich der Systemkonfigurationsdateien in /etc. Diese Konfiguration ist vielleicht nicht so hochdynamisch wie der Userdatenbereich, aber er enthält doch auch einige Daten, die sich oft verändern können. Neue oder veränderte Passwörter, die gesammte Arbeit der Systemkonfiguration usw. Um zu vermeiden, daß ein zu großer Datenbestand gesichert werden muß, ist es auch möglich, ein Verzeichnis auf einem Dateisystem anzulegen, das regelmäßig im Backup aufgenommen wird (z.B. /home). Dann wird ein Shellscript erstellt, das alle wichtigen Systemkonfigurationen aus /etc dorthin kopiert. Etwas wie

```
#!/bin/bash

cp /etc/passwd /home/backup
cp /etc/shadow /home/backup
cp /etc/group /home/backup
cp /etc/gpasswd /home/backup
cp /etc/fstab /home/backup
...
```

Dieses Script sollte jetzt täglich von cron aufgerufen werden. Damit ist sichergestellt, daß die wichtigsten Konfigurationsdateien ins Backup aufgenommen sind.

Backup mit dump

Das Programm **dump** bietet alle Funktionalität, um entweder ganze Dateisysteme (Partitionen) oder einzelne Verzeichnisse in ein Backup aufzunehmen. Die ursprüngliche Version von dump kommt aus der BSD Unix Welt, das Linux-Dump Programm ist aber eine eigene Version, die speziell für das EXT2 Dateisystem geschrieben wurde. Es gibt heute aber auch schon Versionen für weitere Dateisystemtypen, insbesondere für das ReiserFS.

Dump untersucht Dateien eines Dateisystems und entscheidet, welche Dateien in ein Backup aufgenommen werden müssen. Diese Dateien werden dann auf das angegebene Backup-Medium geschrieben, das entweder eine Platte, ein Band oder eine Datei sein kann. (Einstellbar mit der **-f** Option)

Passt das Backup nicht auf das angegebene Medium, so fordert dump zu einem Medienwechsel auf, und verteilt das Backup auf mehrere solcher Medien.

Dump unterstützt 10 Backup-Level (0-9), wobei wie oben beschrieben das Level 0 ein Vollbackup meint und jedes weitere Level ein inkrementelles Backup relativ zum letzten Backup des nächst-niedrigeren Levels bedeutet. Die grundsätzliche Aufrufform von dump ist

```
dump [-Level] [-u] [-f Backupdatei] Dateisystem
```

oder

```
dump [-Level] [-u] [-f Backupdatei] Verzeichnis
```

Wobei meist also zuerst das Backup-Level angegeben wird (0-9), gefolgt von einem u (update /etc/dumpdates). Mit der Anweisung -f *Backupdatei* wird das Medium gewählt. Hier steht also entweder eine Gerätedatei des verwendeten Bandlaufwerks oder ein Dateiname, der Datei, die das Backup aufnehmen soll. Die abschließende Angabe des Dateisystems oder Verzeichnisses gibt an, was gesichert werden soll.

Passt das Backup nicht auf das angegebene Medium, so fordert dump zu einem Medienwechsel auf, und verteilt das Backup auf mehrere solcher Medien.

Dump unterstützt 10 Backup-Level (0-9), wobei wie oben beschrieben das Level 0 ein Vollbackup meint und jedes weitere Level ein inkrementelles Backup relativ zum letzten Backup des nächst-niedrigeren Levels bedeutet. Die grundsätzliche Aufrufform von dump ist

```
dump [-Level] [-ua] [-f Backupdatei] Dateisystem
```

oder

```
dump [-Level] [-ua] [-f Backupdatei] Verzeichnis
```

Wobei meist also zuerst das Backup-Level angegeben wird (0-9), gefolgt von einem u (update /etc/dumpdates) und einem a, das die automatische Bandendeerkennung aktiviert. Dieses -a Argument ist bei modernen Bandlaufwerken besser als die Kalkulation, wieviel Daten auf das Band passen und wann es gewechselt werden muß. Bei der Verwendung von Dateien als Medium ist es unabdingbar diese Option zu setzen! Mit der Anweisung -f *Backupdatei* wird das Medium gewählt. Hier steht also entweder eine Gerätedatei des verwendeten Bandlaufwerks oder ein Dateiname, der Datei, die das Backup aufnehmen soll. Die abschließende Angabe des Dateisystems oder Verzeichnisses gibt an, was gesichert werden soll.

Die Datei /etc/dumpdates speichert die Daten alle bisherigen Backups in der Form:

```
/dev/hda8 0 Sat May  5 21:56:49 2001
/dev/hda8 1 Mon May  7 21:55:58 2001
...
```

Hier zeigt sich also, daß die Partition /dev/hda8 am Samstag, den 5. Mai 2001 um 21:48 Uhr ein Vollbackup (Level 0) erhalten hat, während am darauffolgenden Montag ein inkrementelles Backup (Level 1) ausgeführt wurde.

Der Aufruf

```
dump -0ua -f /dev/tape /dev/hda8
```

hatte das erste dieser beiden genannten Backups erstellt, das zweite wurde mit

```
dump -1ua -f /dev/tape /dev/hda8
```

erstellt.

Es sei hier an dieser Stelle noch einmal ausdrücklich an die Datei /etc/fstab erinnert, deren fünftes Feld die Frage klärt, ob die jeweilige Partition von dump bearbeitet werden soll, oder nicht.

Daten wiederherstellen mit restore

Das Gegenstück zum Programm dump ist **restore**, mit dem sich Daten von einem Backup-Medium wiederherstellen lassen. Dieses Programm hat sehr viele verschiedene Anwendungen, die alle mit Kommandozeilenparametern einstellbar sind. Die wichtigsten Aktionen für die wir dieses Programm brauchen sind nicht nur das Wiederherstellen von Daten, sondern auch der Vergleich eines Backups mit dem Orginaldatenbestand. Damit können wir bei wichtigen Backups die Konsistenz des Backups überprüfen.

Wichtige Aufrufformen von restore sind:

`restore -C -f *Datei*`

Das Backup, das sich in der *Datei* (Gerätedatei eines Tapes oder Datei) befindet wird mit dem Orginaldatenbestand verglichen. Eventuell auftretende Unstimmigkeiten werden angezeigt.

`restore -i -f *Datei*`

Das Backup, das sich in der *Datei* (Gerätedatei eines Tapes oder Datei) befindet wird mit einem interaktiven - shellähnlichen - Mechanismus durchwandert. Einzelne Dateien und Verzeichnisse können markiert werden und dann können alle markierten Dateien wiederhergestellt werden. Die wichtigsten Befehle dieser shellähnlichen Oberfläche sind:

cd *Verzeichnis*

Wechselt in das angegebene Verzeichnis.

ls [*Verzeichnis*|*Datei*]

Zeigt den Inhalt eines Verzeichnisses oder den Namen einer Datei. Ist eine Datei zur Wiederherstellung markiert, wird das durch ein * angezeigt.

add *Verzeichnis*|*Datei*

Das angegebene Verzeichnis oder die angegebene Datei wird zur Wiederherstellung markiert. Ist es ein Verzeichnis so werden alle darin enthaltenen Dateien auch in die Liste der zu wiederherstellenden Dateien aufgenommen (wenn nicht die -h Option auf der Kommandozeile gesetzt war).

delete *Verzeichnis*|*Datei*

Das angegebene Verzeichnis oder die angegebene Datei werden von der Liste der wiederherzustellenden Dateien gestrichen.

extract

Die Dateien und Verzeichnisse, die in der Liste der wiederherzustellenden Dateien aufgelistet sind werden wiederhergestellt.

quit

Das Programm wird beendet.

Zu beachten ist, daß restore beim Widerherstellen die Dateien im aktuellen Verzeichnis auspackt. Sollen also Dateien eines Dateisystems /dev/XXX wiederhergestellt werden, so bietet es sich an, auf die Wurzel dieser Partition zu wechseln.

`restore -r -f *Datei*`

Ein komplettes Dateisystem wird wiederhergestellt. Idealerweise ist das Ziel eine leere, frische Partition, in die dann gewechselt wird um den restore-Befehl auszuführen. Also z.B.:

```
mke2fs /dev/sda1
mount /dev/sda1 /mnt
cd /mnt

restore -r -f /dev/tape
```

Das ist die beste Lösung nach einem Plattencrash, wenn die orginale Platte zerstört ist und eine neue Platte komplett restauriert werden muß.

`restore -x -f` *`BackupDatei Datei1 Datei2 ...`*

Die angegebenen Dateien (*Datei1 Datei2* ...) werden von dem Backup der Backupdatei (Gerätedatei oder Datei) gelesen und extrahiert. Auch hier wird wieder ins aktuelle Verzeichnis extrahiert. Sind einzelne der angegebenen zu extrahierenden Dateien Verzeichnisse, dann werden diese Verzeichnisse samt aller Dateien darin wiederhergestellt.

Restore ist also das Programm, mit dem alle Bearbeitung der bereits gemachten Backups vorgenommen werden, insbesondere die Wiederherstellung der gesicherten Dateien.

Ist ein Backup auf mehrere Medien verteilt, so fordert restore zum gegebenen Zeitpunkt zum Medienwechsel auf..

Partielle und manuelle Backups mit tar

Um schnell ein Verzeichnis oder ein paar Dateien zu sichern ist die Anwendung von dump und restore sicherlich übertrieben. Hier findet das Programm tar seine Anwendung. tar steht für Tape Archiver, ist also grundsätzlich auch gedacht gewesen, Backups auf Bandlaufwerke zu schreiben. Heute wird tar aber sehr häufig eingesetzt, um Verzeichnisse oder Dateien in eine Archivdatei zu packen, die dann auf anderen Medien abgespeichert werden kann oder auch an andere User weitergegeben werden kann. Vor der Einführung der modernen Paket-Manager (RPM, DPKG) war das tar-Format das bestimmende Format für Linux-Distributionen.

Tar wird sowohl zum Erstellen, als auch zum Entpacken von Archiven benutzt. Es ist also kein zusätzlicher Entpacker wie etwa restore nötig.

Wie schon dump und restore kennt auch tar die Option **-f *Dateiname*** mit der angegeben wird, welche Datei als Medium zum Speichern (oder entpacken) des Archivs verwendet werden soll. Auch hier kann entweder eine Gerätedatei eines Bandlaufwerks, eine reguläre Datei oder ein Bindestrich (für StdIn bzw StdOut) angegeben werden.

Tar selbst nimmt keinerlei Komprimierung vor, arbeitet jedoch gerne mit dem gzip-Programm zusammen. Musste man früher die Archivdatei noch manuell komprimieren, so kennt tar heute die Optionen -z (komprimieren oder dekomprimieren mit gzip) und -Z (komprimieren oder dekomprimieren mit compress - veraltet) um selbstständig gleich gepackte Archive zu erstellen oder zu entpacken. Die

jeweiligen Programme gzip und compress müssen dazu aber vorhanden sein, tar ruft sie nur auf, es kann selbst keine Kompression vornehmen.

Die normale Endung einer Archivdatei, die mit tar hergestellt wurde heisst in der Regel `.tar` - wurde das Archiv mit gzip komprimiert trägt es standardmäßig die Endung `.tar.gz` oder (um auch DOS-konform zu sein) `.tgz`. Wurde die Archivdatei mit compress komprimiert, so endet ihr Name meist auf `.tar.Z`

Die drei wichtigsten Funktionen von tar sind

- Erstellen (create) von Archiven (-c)
- Inhaltsverzeichnis (table of content) von Archiven anzeigen (-t)
- Archive entpacken (extract) (-x)

Um ein Archiv mit dem Namen `Test.tgz` anzulegen, das mit gzip komprimiert ist und alle Dateien des Verzeichnisses `Testdir` enthält, rufen wir tar folgendermaßen auf:

```
tar -czf Test.tgz Testdir
```

Um den Inhalt dieses frisch geschaffenen Archivs anzusehen schreiben wir:

```
tar -tzf Test.tgz
```

Und um das Archiv im aktuellen Verzeichnis wieder zu entpacken ist der Befehl

```
tar -xzf Test.tgz
```

nötig. Wird zusätzlich noch die Option -v angegeben, so arbeitet tar "geschwätzig" (verbose), gibt uns also mehr Informationen mit. Insbesondere beim Ansehen des Inhaltsverzeichnis ist das eine gute Idee. Aber vorsicht: Der erste Parameter von tar legt die Aktion fest (c oder t oder x), die Parameter v und z folgen beliebig. Der f-Parameter kommt am Schluß, denn ihm folgt der Dateiname.

Archive mit cpio bearbeiten

Anstelle von tar kann auch das Programm cpio verwendet werden, um Archive zu erstellen oder zu bearbeiten. cpio kann dabei verschiedene Formate - unter anderem auch das tar-Format - bearbeiten. Ein großer Vorteil von cpio ist der, daß die Liste der zu archivierenden Dateien nicht auf der Kommandozeile (wie bei tar) erwartet wird, sondern auf der Standard-Eingabe. Dadurch ist es möglich, die Ergebnisse etwa des find-Befehls direkt an cpio weiterzupipen und so ein Archiv mit den Suchergebnissen aufzubauen.

cpio arbeitet in drei verschiedenen Modi:

- **copy-out**
  Dateien werden in ein Archiv geschrieben (out meint die Ausgabe in ein Archiv)
- **copy-in**
  Dateien werden aus einem Archiv extrahiert (in meint die Eingabe ins Dateisystem)
- **copy-pass**
  Dateien werden von einem Ort zu einem anderen kopiert, quasi eine Kombination aus copy-out und copy-in, ohne jedoch tatsächlich ein Archiv zu erstellen.

Durch die Verwendung unterschiedlicher Formate und die Fähigkeit die zu archivierenden Dateien sowohl aus einer Dateiliste, als auch von der Standard-Eingabe zu lesen, ist cpio wesentlich flexibler als tar. Allerdings ist es auch um einiges komplizierter anzuwenden.

Ein- und Ausgabe mit dd

Um Dateien beliebiger Art (in unserem Fall speziell Archivdateien) von und auf Gerätedateien zu kopieren, existiert das sehr nützliche (und sehr häufig verwendete) Programm dd. dd steht für DiskDump und ermöglicht es, Daten von Gerätedateien zu regulären Dateien zu kopieren oder umgekehrt. Es ist natürlich auch möglich, von Gerätedatei zu Gerätedatei zu kopieren. Die grundsätzliche Form ist dabei

```
dd if=Eingabedatei of=Ausgabedatei bs=Blockgröße count=Anzahl
```

Dieser Befehl kopiert *Anzahl* Blöcke der Größe *Blockgröße* aus der *Eingabedatei* in die *Ausgabedatei*. Dabei können sowoh Ein-, als auch Ausgabedatei entweder Gerätedateien oder reguläre Dateien sein. Wird die Angabe der Blockgröße weggelassen, so werden die "natürlichen" Blockgrößen der jeweiligen Geräte verwendet. Wird die Anzahl der zu lesenden Blöcke weggelassen, so werden alle Blöcke der Datei (oder des Gerätes) gelesen. So kann z.B ein komplettes Image einer Festplatte oder Partition (oder Diskette oder CDROM) in eine Datei durch den Befehl

```
dd if=Gerätedatei_des_Laufwerks of=Imagedateiname
```

erstellt werden. Mit dem umgekehrten Befehl

```
dd of=Gerätedatei_des_Laufwerks if=Imagedateiname
```

wird das entsprechende Image wieder auf das Gerät geschrieben.

Durch die Angabe von ibs= (Input-Blocksize) und obs= (Output-Blocksize) kann sogar noch zwischen verschiedenen Blockgrößen bei Ein- und Ausgabe unterschieden werden.

Zusätzlich kann dd auch noch Konvertierungen der zu kopierenden Blöcke vornehmen, wie etwa die Reihenfolge der Bytes in einem Datenwort oder Klein-Großschreibung.

Für die Backup-Technik ist dd besonders interessant, weil damit Archivdateien direkt auf Band geschrieben werden können oder von Bändern gelesen werden können.

1.111.6

Verwalten der Systemzeit

Beschreibung: Prüfungskandidaten sollten in der Lage sein, die Systemzeit richtig zu verwalten und die Uhr über NTP zu synchronisieren. Dieses Lernziel beinhaltet das Setzen von Systemdatum und -zeit, das Setzen der BIOS-Uhr auf die korrekte Zeit in UTC, die Konfiguration der korrekten Zeitzone des Systems und die Konfiguration der automatischen Korrektur der Zeit auf die NTP-Uhr.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **date**
- **hwclock**
- **ntpd**
- **ntpdate**
- `/usr/share/zoneinfo`
- `/etc/timezone`
- `/etc/localtime`
- `/etc/ntp.conf`
- `/etc/ntp.drift`

Linux arbeitet mit zwei verschiedenen Uhren, die beide eine große Rolle spielen, die Hardware-Uhr (oder auch RTC, RealTimeClock, CMOS-Uhr, BIOS Uhr) und die Systemzeit. Um mit Linux und der Uhrzeit korrekt umzugehen müssen diese beiden Systeme genau unterschieden werden.

Die Hardware-Uhr ist ein Uhrenbaustein, der völlig unabhängig vom verwendeten Betriebssystem arbeitet. Es handelt sich um eine batteriegepufferte Uhr, die also selbst dann arbeitet, wenn der Computer ausgeschaltet (und von der Stromversorgung getrennt) ist.

Die Systemuhr von Linux ist eine Software-Uhr, die ein Teil des Linux-Kernel ist und durch einen regelmäßigen Timer-Interrupt gesteuert wird. Diese Software-Uhr zählt die Sekunden, die seit dem 1. Januar 1970 um 0:00 Uhr vergangen sind.

Unter Linux zählt nur die Systemzeit, also die kernelgesteuerte Software-Uhr. Der einzige Punkt, wo die Hardware-Uhr ins Spiel kommt ist der Moment des Systemstarts, wo die Systemzeit gestellt werden muß. Hier wird die Hardware-Uhr gelesen und die Systemzeit entsprechend eingestellt. Danach wird die Hardware-Uhr nicht weiter benutzt bis zum nächsten Systemstart.

Es gibt im Wesentlichen zwei Programme, die benutzt werden um auf die beiden Uhren zuzugreifen:

- **date**
  Anzeigen und Stellen der Systemzeit
- **hwclock**
  Anzeigen und Stellen der Hardware-Uhr

Bei Unix/Linux-Systemen bestehen jetzt zwei grundsätzliche Möglichkeiten, wie die Uhr gestellt werden sollte. Zum einen können wir die Uhr einfach nach der lokalen Zeit stellen, zum anderen haben wir die Möglichkeit, die Hardware-Uhr (und damit auch die Systemzeit) nach der UTC (Greenwich Mean Time) zu stellen und uns anhand einer entsprechenden Zeitzoneninformation die lokale Zeit daraus ausrechnen zu lassen. Diese zweite Möglichkeit ist heute die, der im Allgemeinen der Vorzug gegeben wird. Auch die Lernziele der LPI102 Prüfung zielen auf diese zweite Möglichkeit ab, also werden wir uns entsprechend daran halten.

Der Vorgang ist eigentlich ganz einfach. Wir stellen die Hardware-Uhr manuell (über das BIOS-Setup) auf Greenwich-Zeit und weisen der Umgebungsvariable `TZ` (TimeZone) den Wert zu, der unsere lokalen Zeitzone entspricht. Das verbreitete Shellscript **tzselect** bietet eine Auswahlmöglichkeit für die zur Verfügung stehenden Zeitzonen. In einigermaßen modernen Linuxen besteht zudem die Möglichkeit, in der Datei `/etc/timezone` die aktuelle Zeitzone einzutragen.

Die moderne Standard-Library von Linux kennt noch eine genauere Methode. Im Verzeichnis `/usr/share/zoneinfo` befinden sich binäre Informationsdateien für jede Zeitzone der Welt (inkl. Antarktis). Im Verzeichnis `/etc` kann jetzt ein symbolischer Link mit Namen `localtime` angelegt werden, der auf die entsprechende Zonendatei zeigt, also z.B. für Deutschland

```
ln -s /usr/share/zoneinfo/Europe/Berlin /etc/localtime
```

Um das zu vereinfachen existiert auf vielen Linuxen ein Script mit Namen **tzconfig**, das genau die Zeitzone abfrägt und anschließend den Link erstellt.

Ist die Zeitzone richtig eingestellt, so verhalten sich alle Linux-Programme, die mit Uhrzeiten arbeiten, automatisch korrekt. Jetzt können wir z.B. auch unsere Hardware-Uhr oder die Systemzeit stellen und die Programme hwclock und date rechnen aus der angegebenen Lokalzeit und der Information der Zeitzone die Greenwich-Zeit aus und stellen die Uhren entsprechend.

Damit Linux jetzt weiß, daß unsere Hardware-Uhr (und die Systemuhr) auf UTC (Greenwich) eingestellt ist, benutzen wir wieder das Programm hwclock. In einem Init-Script sollte die Anweisung

```
hwclock --utc --hctosys
```

stehen. (Falls die Uhr aber auf lokaler Zeit steht sollte statt --utc eben --localtime eingetragen sein)

date

Der Befehl **date** ist der normale Weg, wie auf die Systemzeit von Unix zugegriffen werden kann. Er bezieht sich aber immer auf die Systemzeit und nicht auf die CMos-Uhr (RTC Real-Time-Clock).

Als einfachste Form der Anwendung kann date einfach ohne Parameter aufgerufen werden. Dann gibt er die Systemzeit in einem String der Form

```
Sat Aug 21 14:17:40 MEST 1999
```

auf die Standard-Ausgabe aus. Das ist aber, gerade für Nicht-Amerikaner, nicht immer die gewünschte Form. Aus diesem Grund bietet **date** auch die Möglichkeit, eine beliebige Ausgabeform für ein Datum zu erstellen.

Wenn **date** einen Parameter erhält, der mit einem Pluszeichen (+) beginnt, so interpretiert es die folgende Zeichenkette und ersetzt bestimmte Elemente darin mit den entsprechenden Werten der Systemzeit.

Diese interpretierten Elemente beginnen immer mit einem Prozentzeichen (%) dem ein Buchstabe folgt. So bedeutet etwa das %H die Stunden im 24 Stunden-Modus, %M die Minuten. Wenn wir also nur wissen wollen, wie spät es gerade ist, könnten wir schreiben:

```
date "+Es ist jetzt %H Uhr und %M Minuten"
```

Die Anführungszeichen sind nötig, weil in unserer Zeichenkette Leerzeichen enthalten sind. Die Ausgabe des Befehls wäre jetzt etwas in der Art:

```
Es ist jetzt 14 Uhr und 25 Minuten
```

Eine genaue Darstellung aller unterstützter Parameter ist in der Handbuchseite aufgeführt.

Der Systemverwalter kann mit dem Befehl **date** auch die Systemuhr stellen, allerdings wird nur die Systemzeit, nicht die CMos-Zeit verändert. Dazu gibt es zwei Möglichkeiten:

1. Dem Kommando **date** folgt ein Parameter in der Form MMDDhhmm, also eine Zeichenkette, deren erste zwei Zeichen die Monatszahl (01-12) enthält, die nächsten zwei den Tag des Monats (01-31), die nächsten die Stunden (00-23) und die letzten die Minuten (00-59). Optional kann die Zeichenkette auch noch eine zwei oder vierstellige Jahreszahl und danach, durch einen Punkt getrennt die Angabe der Sekunden (zweistellig) enthalten. (MMDDhhmmYYYY.ss)

2. Das Kommando **date** wird mit dem Schalter **-s** und einem darauffolgendem Datum der Art "Sat Aug 21 14:17:40 MEST 1999" aufgerufen. Das macht in der Regel nur dann einen Sinn, wenn vorher das date-Programm benutzt wurde um diese Zeit abzuspeichern, oder wenn ein date eines Rechners benutzt wird, die Uhren anderer Rechner zu stellen.

Verlässlichere Uhren

Sowohl die Hardware-Uhr, als auch die kernelbasierte Systemuhr sind nicht besonders verlässlich. Zwar bietet die Systemuhr die Möglichkeit über ausgeklügelte Mechanismen wie die Datei `/etc/adjtime` die Genauigkeit der Zeit zu erhöhen, aber eine wirklich hundertprozentige Zeit ist das auch nicht. Gerade aber bei Servern, die monatelang laufen, summiert sich auch eine kleine Ungenauigkeit auf eine störende Größenordnung. Aus diesem Grund gibt es ein spezielles Protokoll, das eine Synchronisation der Zeit über das Netz ermöglicht: das Network Time Protocol (NTP).

Damit NTP aber vernünftig arbeiten kann, benötigt es eine verläßliche Quelle für die Uhrzeit. Ein NTP-Server wird in der Regel eine Funkuhr besitzen, die eine absolut genaue Uhrzeit für den Rechner zur Verfügung stellt. Diese genaue Zeit kann er jetzt an NTP-Clients weitergeben.

Der Haken an dieser Sache ist, daß Pakete, die über das Netz transportiert werden, auch eine bestimmte Zeit benötigen, von einem Rechner zum anderen zu kommen. In einem lokalen Netz ist dieser Wert vernachlässigbar, im Internet können schon einige Sekunden zusammenkommen, je nach aktueller Verbindungsqualität und der Anzahl der Router, die zwischen den beiden Kommunikationspartnern liegen.

NTP hat auch für diese Frage eine Lösung parat. In der Regel werden nicht nur ein, sondern mehrere Server im Internet als Zeitquelle benutzt. Zu jedem Server wird eine Verbindung aufgebaut, und der Client errechnet sich aktuell, wieviel Zeit eine solche Verbindung dauert. Dazu sendet er eine Anfrage und wartet auf die Antwort. Er misst die entsprechend vergangene Zeit und errechnet sich so ein Mittel der Dauer, die ein Paket vom NTP-Server zu ihm benötigt. So kann er - mit mehreren Servern - auf eine Genauigkeit im Millisekundenbereich kommen, was völlig ausreicht, da unsere Uhr als kleinste Einheit sowieso mit Sekunden arbeitet.

Grundsätzlich geht es in diesem Zusammenhang nicht um den Aufbau eines NTP-Servers, der mit einer Funkuhr ausgestattet ist, sondern um die Anbindung eines Clients an einen existierenden Server. Diese Aufgabe ist sehr einfach zu realisieren, es gibt aber zwei verschiedene Möglichkeiten, einer solchen Anbindung:

- Anbindung über einen ständig laufenden Daemon-Prozess mit **ntpd**
- Anbindung über einen cron-gesteuerten Aufruf von **ntpdate**

Beide Möglichkeiten arbeiten mit dem NTP-Protokoll, aber im ersten Fall arbeitet unser Rechner auch als Server, während er im zweiten Fall nur Client ist.

Zeitsynchronisation mit ntpdate

Wenn es nur gewünscht ist, eine regelmäßige Anpassung der Uhrzeit über einen bestehenden Zeitserver vorzunehmen, so genügt es, mit dem Programm **ntpdate** die Uhrzeit eines öffentlichen Zeitservers einzuholen. In diesem Fall kann unsere Systemzeit vollkommen verstellt sein, die Uhrzeit (und das Datum) werden einfach vom genannten Server übernommen. Der Aufruf lautet:

```
ntpdate Zeitserver
```

wobei der Zeitserver entweder über seinen Namen oder seine IP-Adresse angegeben werden kann. Eine Liste öffentlicher Zeitserver im Internet ist unter http://www.eecis.udel.edu/~mills/ntp/clock1.htm erhältlich.

Natürlich bietet es sich an, diesen Aufruf regelmäßig über cron vorzunehmen. Ein entsprechender Eintrag in /etc/crontab könnte also lauten:

```
12 * * * *   root  /usr/sbin/ntpdate ntp3.fau.de
```

Damit würde einmal stündlich (jeweils um 12 Minuten nach der vollen Stunde) die aktuelle Uhrzeit vom Server ntp3.fau.de (einem öffentlichen Zeitserver der Universität Erlangen) geholt und die Systemzeit danach gestellt.

Zeitsynchronisation mit ntpd

Komplexer ist die Aufgabe, die Zeit über einen Daemon zu stellen. Der **ntpd** oder auch **xntpd** ermöglicht diese Form der Synchronisation. Der Daemon synchronisiert sich alle 5 Minuten mit einem oder mehreren Zeitservern oder einer lokalen Funkuhr. Seine Konfiguration erhält er über die Datei `/etc/ntp.conf`. In dieser Datei stehen im einfachsten Fall nur zwei Einträge:

```
server ntp3.fau.de
driftfile /etc/ntp.drift
```

Die erste Zeile bestimmt den verwendeten Zeitserver, von dem regelmäßig (alle 5 Minuten) die Zeit geholt werden soll. Die zweite Zeile bestimmt eine Datei, die **ntpd** selbst anlegt und in der die Abweichung der Systemzeit zur realen Zeit gespeichert wird. **ntpd** ermittelt über Stunden hinweg die Ungenauigkeit der lokalen Zeit und ermittelt einen Wert der Ungenauigkeit in PPM (Parts per Million), der als einfache Fließkommazahl in der Datei `/etc/ntp.drift` abgelegt wird. Bei jedem Neustart von ntpd kann dann mit diesem Wert gerechnet werden.

Bei der Verwendung von **ntpd** als Daemon haben wir jetzt neben einer genauen Uhrzeit die Möglichkeit, selbst als Zeitserver für lokale Clients aufzutreten.

Im Gegensatz zur Verwendung von **ntpdate** darf die Systemzeit unseres Rechners aber nie wirklich größere Abweichungen enthalten. Sobald unsere lokale Zeit mehr als 1000 Sekunden Verschiebung zur Zeit des Servers enthält, weigert sich **ntpd** die Zeit tatsächlich zu stellen, sondern fordert (in einer Syslog-Meldung) dazu auf, die Zeit manuell zu stellen.

Es ist nicht möglich, beide Techniken auf einmal zu betreiben. Ein Versuch, **ntpdate** aufzurufen, obwohl ein lokaler **ntpd**-Prozess läuft, endet mit der Fehlermeldung:

```
13 Aug 14:37:05 ntpdate[3209]: the NTP socket is in use, exiting
```

Beide Programme benutzen den selben Socket, können also nicht nebeneinander existieren. Es ist aber sehr wohl möglich, im entsprechenden init-Script, das **ntpd** starten soll, einen Aufruf von **ntpdate** vor dem Aufruf des Daemons zu veranlassen, um sicherzustellen, daß die Systemzeit keine zu große Abweichung aufweist.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

Name

pwconv, pwunconv, grpconv, grpunconv - Konvertiert in und aus Shadow-Passwort- und Gruppendateien.

Syntax

pwconv
pwunconv
grpconv
grpunconv

Beschreibung

All diese vier Programme arbeiten mit den normalen und shadow- Dateien für Passwörter und Gruppen: /etc/passwd, /etc/group, /etc/shadow, und /etc/gshadow.

pwconv erzeugt shadow aus passwd und einer optionalen shadow-Datei.
pwunconv erzeugt passwd aus der passwd und shadow-Datei und löscht anschließend die shadow-Datei.
grpconv erzeugt gshadow aus group und einer optionalen gshadow-Datei.
grpunconv erzeugt group aus group und gshadow und löscht anschließend die gshadow-Datei.

Jedes dieser Programme erzeugt die notwendigen Lock-Dateien, vor der Konvertierung.

pwconv und **grpconv** sind ähnlich. Zunächst werden Einträge aus der jeweiligen shadow-Datei entfernt, die in der Haupt-Datei nicht existieren. Danach werden Shadow-Einträge, deren Passwortfeld in der Haupt-Datei kein x enthalten aktualisiert. Alle fehlenden Shadow Einträge werden angefügt. Abschließend werden die Passwort-Einträge aus der Haupt-Datei durch ein x ersetzt. Diese beiden Programme können sowohl für die initiale Konvertierung, als auch für ein nachträgliches Update benutzt werden, wenn die Haupt-Dateien von Hand verändert wurden.

pwconv benutzt die Werte **PASS_MIN_DAYS**, **PASS_MAX_DAYS** und **PASS_WARN_AGE** aus der Datei `/etc/login.defs`, wenn neue Einträge in die Shadow-Datei gemacht werden.

Entsprechend sind auch die Programme **pwunconv** und **grpunconv** ähnlich. Passwörter in der Haupt-Datei werden aus den Shadow-Dateien übernommen. Einträge, die in der Haupt-Datei existieren, nicht jedoch in der entsprechenden Shadow-Datei bleiben unangetastet. Abschließend wird die Shadow-Datei gelöscht.

Einige Informationen über die Gültigkeit von Passwörtern geht durch die Anwendung von **pwunconv** verloren.

Bugs

Fehler in den Passwort- oder Gruppendateien (z.B. unvollständige oder doppelte Einträge) können zu Endlosschleifen oder ähnlich unerfreulichen Fehlern führen. Um das zu vermeiden, empfielt es sich, vor der Konvertierung die Programme **pwck** und **grpck** laufen zu lassen.

1.112.1

Grundlagen von TCP/IP

Beschreibung: Prüfungskandidaten sollten ein richtiges Verständnis der Netzwerk-Grundlagen demonstrieren können. Dieses Lernziel beinhaltet das Verständnis von IP-Adressen, Netzwerkmasken und ihrer Bedeutung (d.h. Bestimmung einer Netzwerk- und Broadcast-Adresse für einen Host auf Grundlage seiner Subnetzmaske in "dotted quad" oder abgekürzter Notation oder die Bestimmung der Netzwerk-Adresse, Broadcast-Adresse und Netzwerkmaske bei gegebener IP-Adresse und Anzahl von Bits). Dies deckt auch das Verstehen der Netzwerkklassen und klassenloser Subnetze (CIDR) und der für private Netzwerke reservierten Adressen ab. Ebenfalls enthalten ist das Verständnis der Funktion und Anwendung der Default Route. Weiters enthalten ist das Verstehen der grundlegenden Internet-Protokolle (IP, ICMP, TCP, UDP) und der gebräuchlicheren TCP- und UDP-Ports (20, 21, 23, 25, 53, 80, 110, 119, 139, 143, 161).

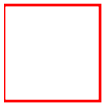
Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/services`
- **ftp**
- **telnet**
- **host**
- **ping**
- **dig**
- **traceroute**
- **whois**

Dieses Kapitel ist wenig Linux-spezifisch, sondern bezieht sich auf das Verständnis von TCP/IP im Allgemeinen. Natürlich werden die jeweiligen Linux-Techniken auch mit erwähnt, wo sie anfallen, wichtig ist jedoch das fundierte Verständnis der Technik von IP-Netzwerken, von Netzadressen usw. Ich beginne hier also mit ein paar ausführlichen Abschnitten über diese Themen. Abschließend werden noch die hier aufgeführten Programme besprochen.

Die TCP/IP Protokollfamilie

TCP/IP hält sich nicht an das OSI-Schichtenmodell, es hat ein eigenes Modell, das sich nicht eins zu eins übertragen lässt.

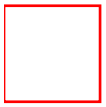


Jede der vier Schichten des TCP/IP Schichtenmodells verfügt über ein bzw. mehrere Protokolle, die den Umgang mit den jeweiligen Daten regeln. Unter Protokollen werden hier Regelsammlungen bezeichnet, die in sogenannten RFCs (Request for comment) veröffentlicht werden.

Das Entscheidende an der Konstruktion von TCP/IP sind die beiden mittleren Schichten, die Transport- und die Vermittlungsschicht. Hier arbeiten immer die gleichen Protokolle. In der Transportschicht stehen zwei zur Verfügung, eins für verbindungsorientierten Datenverkehr (TCP) und eins für verbindungslosen Datentransfer (UDP). In der Vermittlungsschicht arbeitet nur ein Protokoll in der Datenübertragung (IP), so steht also eine einheitliche und verbindliche Übertragungsform zur Verfügung.

In der obersten und untersten Schicht können so die verschiedensten Protokolle arbeiten, von denen die beiden mittleren Schichten nichts wissen müssen. In der Anwendungsschicht können so Standardprotokolle wie etwa FTP (Dateiübertragung), TELNET (Terminal EmuLation über NET), HTTP (WWW-Seiten), SMTP (E-Mail) oder NNTP (News) arbeiten, aber auch eigens entwickelte Protokolle etwa für ein Schachspiel übers Netz. Jedes dieser Anwendungsprotokolle hat eine Identitätsnummer zugewiesen bekommen, die sogenannte Portnummer. Mit Hilfe dieser Portnummer kann das Transportprotokoll wissen, an welche Anwendung in der Anwendungsschicht es Daten weitergeben soll. Unter Linux ist die Liste aller bekannten Portnummern in der Datei `/etc/services` zu finden.

Die unterste Schicht übernimmt die Steuerung der Netzhardware, also die der Netzwerkkarten o.ä. Diese Schicht kann mit beinahe jedem beliebigen Protokoll arbeiten, wie etwa die IEEE 802.3 802.4 802.5 oder FDDI, ATP aber auch mit PPP oder SLIP auf seriellen oder mit PLIP auf paralelle Schnittstellen.



Um die Funktion des Netzsystems genau zu verstehen ist ein Wissen über die einzelnen Schichten und Protokolle unabdingbar. Anhand einiger Beispiele sollen hier die einzelnen Aufgaben der Schichten erklärt werden.

Die Netzzugriffsschicht mit ihren Protokollen

In dieser Schicht arbeiten, wie oben schon erwähnt, beinahe alle Protokolle von Netzhardware, von einfachen seriellen Verbindungen bis zu Hochleistungssysteme mit Glasfaserkabeln. Ein typisches Beispiel soll hier näher betrachtet werden, das Protokoll IEEE-802.3, das Protokoll des Ethernet. Das hier dargestellte Protokoll ist das der 10Mbit Version von Ethernet, die moderneren Versionen unterscheiden sich in Kleinigkeiten, die aber nicht technisch relevant sind.

Die Aufgabe des IEEE-802.3 Protokolls ist es, Daten, die von der Vermittlungsschicht kommen, in Pakete zu packen, und diese Pakete dann über die Netzhardware zu verschicken. Umgekehrt nehmen sie Pakete vom Netz entgegen und geben sie gegebenenfalls an die Vermittlungsschicht weiter. Der Inhalt der Daten spielt dabei keinerlei Rolle.

Physikalisch gesehen, ist ein Ethernet eine parallele elektrische Verbindung aller Rechner, die an einem Netzsegment hängen. Das heißt, wenn ein Rechner einem anderen Rechner ein Paket schickt, so bekommen alle angeschlossenen Rechner dieses Paket. Anhand einer (weltweit eindeutigen) Adresse, die jede Netzwerkkarte besitzt, kann dann aber entschieden werden, ob das Paket tatsächlich für diesen Rechner bestimmt ist. Jedes Paket, das über ein IEEE-802.3 Protokoll verschickt wird enthält sowohl diese Adresse des Absenders, als auch die des Empfängers. Alle angeschlossenen Rechner im Netz (genauer gesagt, alle angeschlossenen Netzwerkkarten im Netz) empfangen das gesendete Paket, aber nur die Netzwerkkarte, die tatsächlich die Adresse aufweist, die im Paket als Empfängeradresse angegeben ist, gibt das Paket entpackt an die Vermittlungsschicht weiter. Diese Adressen sind die sogenannten MAC-Adressen.

Die detaillierte Darstellung des IEEE 802.3 Frameformat ist hier einsehbar.

Natürlich gibt es noch viele weitere Protokolle auf der Netztzugriffsschicht, etwa **ppp** und **slip** für die serielle Anbindung an ein IP-Netz, oder andere Hardware-Netzfamilien wie Token Ring. Diese Darstellung ist also nur ein Beispiel von vielen.

Die Vermittlungsschicht und das InternetProtokoll (IP)

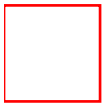
Die Vermittlungs- oder auch Internetschicht hat ein allumfassendes Protokoll, das zum Transport der Daten zuständig ist, das Internet Protokoll (IP). Daneben arbeitet noch ein weiteres wichtiges Protokoll auf dieser Schicht, das für technische Informationen statt für Daten zuständig ist, das Internet Control Message Protokoll (ICMP).

Das Internet Protokoll (IP)

Das Internet-Protokoll ist für die Vermittlung zwischen zwei Rechnern zuständig. Hier wird mit den IP-Adressen gearbeitet, statt mit den physikalischen Adressen im Netzwerk. Die zentrale Aufgabe des IP ist das Versenden und Empfangen von Daten, die für diesen Zweck wieder mal in Pakete aufgeteilt werden müssen. Die Pakete des IP werden Datagramme genannt und beinhalten neben den eigentlichen Daten wieder verschiedenste Steuerinformationen.

- Der Aufbau von IP-Datagrammen

Wenn Pakete (Datagramme) im Internet verschickt werden oder zumindest eine etwas komplexere Topologie in einem lokalen Netz existiert, so müssen Datagramme oft über Gateways geroutet werden. Das heißt, sie müssen durch verschiedene Rechner hindurch zum Empfänger geleitet werden. Auch diese Aufgabe erfüllt das IP-Protokoll, so daß solche Pakete im Gateway nie über die IP-Schicht hinweg kommen:



Zusammenfassend kann man sagen, das IP hat folgende Aufgaben:

- Es definiert die Datagramm-Struktur, die die Grundlage für allen Datenverkehr im TCP/IP-Netz darstellt.
- Es definiert die IP-Adress-Schematik, von der später noch die Rede sein wird.
- Es bewegt Daten zwischen der Transportschicht und der Netzzugriffsschicht.
- Es routet Datagramme zu externen Rechnern.
- Es teilt Datenströme in Datagramme um sie zu versenden und baut aus empfangenen Datagrammen wieder Datenströme zusammen.

Das Internet Controll Message Protocol

Das *Internet Control Message Protocol* (ICMP) ist ein integraler Bestandteil von IP und dient zur Übermittlung technischer Meldungen, die übers Netz gehen, aber nicht über die IP-Schicht hinauskommen müssen. D.h., daß dieses Protokoll in der Lage ist, Pakete zu schicken, die nicht an ein Transportprotokoll der Transportschicht gebunden sind, sondern eben nur von Vermittlungsschicht zu Vermittlungsschicht gehen.

Die Aufgaben, die dieses Protokoll zu erfüllen hat sind hier kurz aufgelistet:

Flußkontrolle

Wenn ein Rechner Datagramme so schnell schickt, daß der Empfänger sie nicht rechtzeitig verarbeiten kann, so schickt der Empfänger über ICMP eine Meldung an den Sender, daß die Sendungen vorübergehend stoppen sollen. Nachdem der Empfänger alle anstehenden Datagramme verarbeitet hat, schickt er wieder eine Meldung, daß er wieder empfangsbereit ist.

Erkennen von unerreichbaren Zielrechnern

Wenn ein Gateway erkennt, daß ein bestimmter Rechner nicht erreichbar ist, so schickt er an den Absender des Paketes eine *Destination unreachable* Meldung über ICMP

Routenoptimierung

Wenn ein Gateway erkennt, daß er einen Umweg darstellt, so schickt er an den Absender eine Meldung, in der die schnellere Route steht. Der Absender (bzw. seine IP-Schicht) kann dann das nächste Paket schon über den besseren Weg übermitteln.

Überprüfen von erreichbaren Hosts

Mit Hilfe des *ICMP Echo Message* kann ein Rechner überprüfen ob ein Empfänger ansprechbar ist. Das ping-Kommando nutzt diese Echo-Meldung.

Der Aufbau von IP-Adressen

Das IP-Adressen Format

Jeder Rechner im Netz hat eine (Host) oder mehrere (Gateway) eindeutige IP-Adressen. Streng genommen hat eigentlich jede Netzwerkschnittstelle eine solche Adresse. Dabei handelt es sich um die Adressen, die in den IP-Datagrammen angegeben werden. Aus der Aufteilung des IP-Datagramms geht schon hervor, daß diese Adressen 32 Bit breite Zahlen sind. In der Regel werden diese Zahlen in vier Oktetts dargestellt.

Die hexadezimale IP-Adresse `954C0C04` wird also in seine vier Oktetts (Bytes) aufgeteilt, die durch Punkte getrennt werden. Dadurch entsteht hexadezimal `95.4C.0C.04` oder die bekanntere, dezimale Form: `149.76.12.4`.

IP-Adressen werden zentral vergeben, das heißt, es sollten nicht einfach beliebige Kombinationen verwendet werden. Im Prinzip ist es in reinen lokalen Netzen möglich, beliebige Adressen zu benutzen, sobald ein Anschluß ans Internet besteht, müssen aber zentral vergebene Adressen benutzt werden. Jedes Land der Erde unterhält dazu ein NIC (Network Information Center), das für die Adressvergabe zuständig ist.

Netz- und Hostadressen

Die Aufteilung der IP-Adressen in vier Oktette hat noch einen anderen wesentlichen Grund. Das Internet besteht zum großen Teil nicht aus vernetzten Einzelrechnern, sondern aus Computernetzen, die miteinander über das Internet kommunizieren können. Die IP-Adresse ist immer aufgeteilt in eine Netz- und eine Hostadresse. Diese Eigenschaft spielt für das Routing eine wesentliche Rolle, weil immer davon ausgegangen werden kann, daß IP-Adressen mit der gleichen Netzadresse im lokalen Netz sind.

Je nachdem, welcher Teil der Gesamtadresse als Netz- und welcher als Hostadresse gilt, entstehen unterschiedliche Obergrenzen für die mögliche Anzahl von Netzen bzw. Hosts pro Netz. Weil ja alle Adressen im Internet einmalig sein müssen, kommt es heute bereits zu Engpässen bei der Adressvergabe. Ein neuer Standard (IP6) ist schon in Arbeit, der dann 128 Bit breite Adressen verwenden soll.

Die Frage, welcher Teil der Adresse Netzadresse ist wird durch Adressklassen gelöst. Es existieren fünf Klassen, von denen aber nur drei relevant sind, die anderen beiden dienen der Forschung und Experimentierarbeit. Die Klassen sind:

Netz klasse	Netzadressen breite	Hostadressen breite	Max Anzahl Netze	Max Anzahl Hosts/Netz	Startadresse	Endadresse
A	1	3	126	16777214	1.0.0.0	126.0.0.0
B	2	2	16382	65534	128.0.0.0	191.255.0.0
C	3	1	2097150	254	192.0.0.0	223.255.255.0

Die Adressen von 224.0.0.0 bis 255.255.255.255 sind für zukünftige Verwendungen reserviert, werden aber wohl nie zur Geltung kommen, weil demnächst das IPV6 kommen wird das diese Erweiterungen schon beinhaltet.

Um Konflikte mit lokalen TCP/IP Netzen und dem Internet zu vermeiden, gibt es für jede Klasse einige Adressen, die nicht im Internet geroutet werden. Sie sind ausschließlich für lokale Netze gedacht und sollten auch immer Verwendung finden, wenn es um die Frage der IP-Adressen geht und kein echter Internetanschluß vorliegt. Diese Adressen sind:

Klasse A

10.0.0.0

Klasse B

172.16.0.0 bis 172.31.0.0

Klasse C

192.168.0.0 bis 192.168.255.0

Wo immer lokale Netze ohne direkte Anbindung des ganzen Netzes ans Internet vorliegen, werden Adressen aus diesem Pool verwendet.

Spezielle Adressen

Die Adressen, deren Hostadressenbits alle auf 0 oder 1 gesetzt sind, haben eine Sonderbedeutung. Alle auf Null gesetzt meint das jeweilige Netz selbst, alle auf 1 gesetzt, meint alle angeschlossenen Hosts. Das ist die sogenannte Broadcast-Adresse. So bedeutet die Adresse 149.76.255.255 nicht eine bestimmte Adresse im Netz 149.76.0.0 sondern alle Rechner, die an diesem Netz hängen.

Es gibt auch zwei reservierte Netzadressen mit Sonderbedeutung. Das sind die Adressen 0.0.0.0 und 127.0.0.0 Die erste ist die sogenannte *default route* (voreingestellte Route) und die andere das sogenannte *Loopback*.

Das Netz 127.0.0.0 bedeutet intern im Netz immer das lokale Netz, egal welche reale Adresse es hat. Gewöhnlich steht dann auch noch die Adresse 127.0.0.1 als *loopback interface* des jeweiligen Hosts zur Verfügung. Jedes Paket, das darauf ausgegeben wird kommt sofort wieder an, als käme es aus einem Netz. Damit kann z.B. Software entwickelt werden, für Netze, auf einem Rechner, der nicht an eins angeschlossen ist.

Die Adresse 0.0.0.0 hat eine Bedeutung für das Routing. Jedes Datagramm, das an eine Adresse geschickt wird, zu der keine Route bekannt

ist, wird an die *default route* geschickt. Diese voreingestellte Route weist dann zum nächsten Gateway, der schon wissen wird, wohin er das Paket senden soll. Falls er keine Route kennt, die zum entsprechenden Ziel führt, so hat auch er eine default route, an die er das Paket dann schickt...

Subnetze und die Netmask

Ein Teil der Host Adresse kann auch intern als Subnetzadresse gewertet werden um große Netze nochmal zu trennen. So kann also der Host mit der Nummer 12.4 im B--Klasse Netz 149.76 auch uminterpretiert werden, in den Host Nummer 4 im Subnetz 12 des Netzwerks 149.76. Welche Bits der 32 Bit Adresse als Netz- und welche als Hostadresse gewertet werden wird in der sogenannten Netzmaske festgelegt.

Die Netzmaske (Netmask) ist wie die Adresse eine 32 Bit Zahl, die in vier Oktetts aufgeteilt ist. Jedes Bit dieser Maske, das gesetzt ist (1), bestimmt daß das entsprechende Bit in der Adresse ein Teil der Netzadresse ist. Zum Verständnis am Besten mal ein Beispiel:

Die B-Klasse Adresse 149.76.0.0 wird normalerweise gewertet als Adresse, deren erste 2 Bytes als Netzadresse gelten, und deren letzte 2 Bytes als Hostadresse. Die Netmask würde also einfach berechnet:

```
                  Netzadresse       Hostadresse
149.76.0.0 --> 10010101.01001100.00000000.00000000
Netmask        11111111.11111111.00000000.00000000
           --> 255.255.0.0
```

Wollen wir jetzt dieses Netz in mehrere Unternetze aufteilen, so können wir z.B. das dritte Byte als Unternetzadresse und das letzte Byte als Hostadresse werten. Für die Netmask käme folgendes Bild heraus:

```
                  Netzadresse        Hostadresse
149.76.0.0 --> 10010101.01001100.00000000.00000000
Netmask        11111111.11111111.11111111.00000000
           --> 255.255.255.0
```

Die Netzmaske kann entweder in dieser Form angegeben werden (dotted quad) oder als verinfachte Angabe, indem einer IP-Adresse einfach die Anzahl der Bits angegeben wird, die Netzadressenbits sind. Die folgenden Adressenangaben sind also äquivalent:

Dotted Quad	**Vereinfacht**
192.168.100.1 Netmask 255.255.255.0	192.168.100.1/24
175.12.34.56 Netmask 255.255.0.0	175.12.34.56/16

Schwieriger wird es, wenn wir etwa ein C-Klasse Netz in weitere Unternetze aufteilen wollen. Hier muß dann jede Adresse errechnet

werden, je nach der Aufteilung. Die Netmask wäre dann entsprechend auch nicht mehr nur 255 sondern würde sich aus anderen Zahlen zusammensetzen. Nochmal ein Beispiel:

Das C-Klasse Netzwerk 192.168.200.0 soll in vier Unternetze aufgeteilt werden. Um vier Netze anzusprechen benötigen wir zwei Bits der Hostadresse für die Subnetzangabe:

```
                                          Subnetzbits
                                               ||
                           Netzadresse         VV
192.168.200.0 --> 11000000.10101000.11001000.00000000
Netmask           11111111.11111111.11111111.11000000
              --> 255.255.255.192
```

Statt der üblichen 255.255.255.0 haben wir jetzt also 255.255.255.192 als Netzmaskierung (oder - vereinfacht /26 weil die ersten 26 Bits der Adresse jetzt die Netzadresse sind). Damit weiß IP, daß diese Adresse kein normales C-Klasse Netz ist. Allerdings müssen wir jetzt auch bei der Vergabe der einzelnen Rechneradressen aufpassen, daß die verwendeten Zahlen im jeweiligen Netz liegen:

```
Subnetz 0
   Netzadresse   0 -> 00000000
   Hostadressen  1 -> 00000001
                 2 -> 00000010
                 3 -> 00000011
                        ...
   Broadcast    63 -> 00111111

Subnetz 1
   Netzadresse  64 -> 01000000
   Hostadressen 65 -> 01000001
                66 -> 01000010
                67 -> 01000011
                        ...
   Broadcast   127 -> 01111111

Subnetz 2
   Netzadresse 128 -> 10000000
```

```
    Hostadresse 129 -> 10000001
                130 -> 10000010
                131 -> 10000011
                         ...
    Broadcast   191 -> 10111111

  Subnetz 3
    Netzadresse 192 -> 11000000
    Hostadresse 193 -> 11000001
                194 -> 11000010
                195 -> 11000011
                         ...
    Broadcast   255 -> 11111111
```

Netzadressen und Broadcast errechnen sich nach der Regel

- Alle Bits der Hostadresse auf 0 ist die Netzadresse.
- Alle Bits der Hostadresse auf 1 ist die Broadcastadresse.

Nur daß eben die Hostadresse jetzt nur noch aus den letzten 6 Bits besteht.

Die Transportschicht (TCP und UDP)

In der Transportschicht arbeiten zwei unterschiedliche Protokolle, TCP und UDP. Der wesentliche Unterschied zwischen beiden Protokollen ist, daß TCP *verbindungsorientiert* arbeitet und UDP *verbindungslos*.

Beiden Protokollen gemeinsam ist die Verwendung sogenannter Portnummern. Dieses System beruht auf der Tatsache, daß es mehrere Verbindungen über ein Transportprotokoll gleichzeitig geben kann. So kann ein Rechner beispielsweise gleichzeitig über FTP Daten von einem anderen Rechner kopieren und via TELNET auf dem gleichen Rechner eingeloggt sein. Das erfordert es, daß das Transportprotokoll in der Lage ist, zu unterscheiden, welche Pakete an FTP und welche an TELNET adressiert sind. Diese Adressierung übernehmen die Portnummern. Die Standard-Internet-Protokolle haben festgelegte Portnummern (definiert in `/etc/services`).

User Datagram Protocol (UDP)

UDP ist ein sehr einfaches Protokoll, das Anwendungen die Möglichkeit bietet, direkten Zugriff auf die Datagramm-Services von IP zu nutzen. Das ermöglicht die Übermittlung von Daten mit einem Minimum an Protokollinformationen.

- Das UDP-Message Format

UDP wird überall dort verwendet, wo entweder die Datenmenge so klein ist, daß es sich nicht lohnen würde, einen großen Header zu benutzen, weil der größer als die eigentlichen Daten wäre oder wo die Anwendungen selbst noch Überprüfungen des Paketinhaltes vornehmen.

Lohnenswert ist der Einsatz auch dort, wo reine Frage-Antwort Mechanismen auftreten. Dort ist kein verbindungsorientiertes Protokoll nötig, weil ja nach dem Senden einer Frage klar ist, wenn nach einer bestimmten Zeit keine Antwort eingeht, so wird das Paket verlorengegangen sein und die Frage muß nochmal gestellt werden.

Wichtig ist die Feststellung, daß das UDP-Protokoll keinen Datenstrom verarbeitet, sondern Datagramme direkt. Es stellt also keinerlei Mechanismen zur Verfügung um einen Datenstrom in Pakete aufzuteilen und wieder zusammenzubauen.

Transmission Control Protocol (TCP)

TCP ist im Gegensatz zu UDP ein verbindungsorientiertes Protokoll, das Datenströme verarbeitet, die von der Anwendungsschicht kommen. TCP garantiert die Versendung von Daten durch eingebaute Handshake-Mechanismen.

TCP bietet einen verlässlichen Datentransfer durch die Verwendung eines Mechanismus, der Datenpakete (sogenannte Segmente) solange an einen Empfänger schickt, bis der eine Bestätigung des Empfangs rückmeldet. Dabei überprüft der Empfänger auch wieder eine Prüfsumme, erst wenn die richtig ist, schickt er das Signal des Empfangs.

- Das TCP-Segment Format

TCP ist verbindungsorientiert, d.h., daß dieses Protokoll nicht einfach Daten losschickt wie UDP sondern zuerst einen Handshake durchführt, um sich mit dem Empfänger über dessen Bereitschaft zu synchronisieren. Das Prinzip ist einfach, eine TCP Übertragung beginnt immer erst mit der Nachfrage ob der Empfänger bereit ist (SYN). Der Empfänger sendet ein entsprechendes Signal (SYN.ACK) und erst nachdem dieses Signal erhalten wurde startet TCP die Übertragung der Segmente.

TCP ist streamorientiert, d.h., daß es seine Daten als kontinuierlichen Datenstrom ansieht. Durch die Verwendung von Sequenznummern kann das empfangende TCP die Segmente wieder in richtiger Reihenfolge zusammenbauen und sie wieder zu einem Datenstrom formieren.

Die Anwendungsschicht

Auf dieser Schicht laufen die Anwendungen, die übers Netz miteinander kommunizieren. In der Regel handelt es sich hier um Client/Server Paare, also um zwei verschiedene Programme, eins, das einen Dienst anbietet (server), ein anderes, das diesen Dienst in Anspruch nimmt (client).

Damit diese Anwendungen sich nicht gegenseitig stören, hat jede Anwendung eine eigene Portnummer. Sie dient als Adresse, an die die Transportschicht einen Datenstrom oder eine UDP-Meldung weiterleitet.

Die wichtigsten Portnummern sollte man parat haben, um die LPI102-Prüfung abzulegen. Es sind

Port	Protokoll	Transportprotokoll	Beschreibung
20	FTP-Data	TCP	Datenkanal einer FTP-Verbindung.
21	FTP	TCP	Kontrollkanal einer FTP-Verbindung.
22	SSH	TCP oder UDP	SecureShell (Verschlüsselter Login)
23	TELNET	TCP	Terminal Emulation over Network
25	SMTP	TCP	Simple Mail Transfer Protocol
53	DNS	TCP oder meist UDP	Nameserver
80	WWW/HTTP	meist TCP oder UDP	Hypertext Transfer Protokoll
110	POP3	TCP oder UDP	Post Office Protocol zum Holen von Mails
119	NNTP	TCP	Net News Transfer Protocol
139	NetBIOS-SSN	TCP oder meist UDP	Windows Netzwerk Sitzungsdienste
143	IMAP2	TCP oder UDP	Interim Mail Access Protocol (verschlüsselt)
161	SNMP	UDP	Simple Network Management Protocol

Programme in diesem Zusammenhang

Die geforderten Kenntnisse der ganz oben genannten Programme dient zunächst nur dafür, zu wissen, wozu welches Programm benutzt werden kann. Hier jeweils eine kurze Beschreibung über die Programme und ihre Anwendung:

ping

Das Programm **ping** benutzt das ICMP-Protokoll um herauszufinden, ob ein Rechner im Netz verfügbar ist. Dazu sendet es eine ICMP-Meldung (Typ 8) an den Zielrechner. Der Zielrechner gibt dann eine Kopie des Paketes mit dem Typ 0 (*pong*) zurück. Das wieder ankommende Paket wird auf die Standard-Ausgabe geschrieben. **ping** sendet ohne Unterlass weiter, bis es mit Strg-C abgebrochen wird.

Um **ping** von vorneherein auf eine bestimmte Anzahl von Paketen zu beschränken, gibt es den Parameter `-c`, mit dem die Anzahl der zu sendenden Pakete angegeben werden kann:

```
ping -c4 foo.bar.com
```

sendet 4 Pakete an den Rechner `foo.bar.com`. Wenn **ping** mindestens eine Antwort erhält, so gibt es in diesem Fall eine 0 zurück,

ansonsten einen Wert ungleich 0. Somit kann **ping** auch in Scripts benutzt werden, um festzustellen, ob ein bestimmter Rechner verfügbar ist.

traceroute

Das Programm **traceroute** ermöglicht es, genau zu bestimmen, über welche Gateways ein Paket zu einem Zielrechner geroutet wird. Es schickt ein Paket an den nächsten Gateway (unsere default route), das das Lebensdauerfeld auf 0 gesetzt hat. Der Gateway schickt daraufhin eine ICMP-Fehlermeldung Typ 11 (`time-exceeded`) an uns zurück. Anschließend wird das selbe Paket mit einer Lebensdauer von 1 geschickt. Jetzt bleibt es beim nächsten Gateway hängen, auch der schickt eine ICMP-Typ 11 Fehlermeldung zurück. So wird die Lebensdauer jedesmal um eins erhöht und das Paket kommt so immer einen Gateway weiter. Aus den Fehlermeldungen der Gateways werden ihre IP-Adressen extrahiert und auf die Standard-Ausgabe geschrieben. Das Ganze wird solange wiederholt, bis der Zielrechner erreicht ist.

Der einzig nötige Parameter für **traceroute** ist der Name oder die IP-Adresse des Zielrechners.

host

Das Programm **host** wird benutzt, um Nameserverabfragen direkt von der Kommandozeile aus vorzunehmen. Es kann über sehr viele Kommandozeilenparameter genau eingestellt werden, wie und was gesucht werden soll. Der einzig notwendige Parameter ist der gesuchte Domainnamen. Die Antwort besteht in diesem Fall aus seiner IP-Adresse.

Wird statt einem Domainnamen eine IP-Adresse eingegeben, so wird umgekehrt der passende Domainnamen ausgegeben, sofern er existiert und ermittelbar ist.

Wenn nicht nur die Information über die IP-Adresse eines Domainnamens gesucht wird, kann der Parameter `-a` angegeben werden, der alle verfügbaren Informationen über einen Rechner anfordert und darstellt.

dig, nslookup, nsquery

Auch diese Programme machen Nameserverabfragen. Sie unterscheiden sich in ihren Fähigkeiten und ihrer Anwendung. So bietet beispielsweise **nslookup** eine interaktive Abfrageshell. **dig** ist in der Lage auch komplexe Probleme zu lösen, so kann zum Beispiel die aktuelle Liste aller Root-Nameserver mit dem Befehl

```
dig . ns
```

erfragt werden oder **dig** kann eine Liste aller Nameserver für eine bestimmte Zone erfragen. An sich werden alle vier Programme (**host, dig, nslookup** und **nsquery**) im Alltag dazu benutzt, explizit einen Nameserver abzufragen.

whois

whois sendet eine Nachfrage an eine RFC-812 Datenbank. In einer solchen Datenbank werden die Daten von Domaininhabern abgelegt. So kann mit **whois** herausgefunden werden, wer eine bestimmte Domain registriert hat, wer der Administrator davon ist usw. Die Menge der Ausgaben wird entweder durch die entsprechende Datenbank vorgegeben oder es wird über Kommandozeilenparameter genau angegeben, welche Information gesucht wird.

Im einfachsten Fall wird **whois** nur mit dem Parameter der Domain aufgerufen, über die Informationen erfragt werden sollen:

```
whois lpi.org
```

gibt als Ausgabe

```
Registrant:
Linux Professional Institute Inc.
78 Leander St.
Brampton, ON L6S 3M7
CA

Domain Name: LPI.ORG

Administrative Contact:
   Silberman, Wilma  nic@lpi.org
   78 Leander St.
   Brampton, ON L6S 3M7
   CA
   (905) 874-4822

Technical Contact:
   starnix, DNS  dns@starnix.com
   175 Commerce Valley Dr. W.
   Thornhill, ON L3T 7P6
   CA
   (416) 410-9342

 ...
```

ftp

Das File Transfer Protokoll dient zur Übertragung von Dateien über das Netzwerk. Es nutzt die TCP-Ports 21 (Kommandokanal) und 20 (Datenkanal) um über TCP/IP Dateien von einem Rechner zum anderen zu kopieren. Auch hier gibt es unter Unix Befehle (**rcp**), die besser auf das System abgestimmt sind, aber auch hier gilt, in heterogenen Netzen hat FTP den Vorteil.

Wie bei Telnet, so gibt es auch hier einen Server (ftpd) und einen Client (ftp, xftp, Netscape), die miteinander kommunizieren. Früher war es üblich, mit der FTP-eigenen Kommandosprache zu arbeiten, heute gibt es graphische Frontends, die FTP wesentlich konfortabler machen.

FTP besitzt zwei Übertragungsmodi, Text und Binärformat. Heute ist es üblich, alle Übertragungen im Binärformat vorzunehmen, es entstehen so weniger Probleme beim Umgang mit Textdateien, die 8 Bit Zeichensätze verwenden (Umlaute...).

Die klassische Art, FTP zu nutzen war interaktiv, mit der FTP-eigenen Kommandosprache. Dabei war fast der Eindruck einer kleinen Terminalsitzung zu erahnen, um sich im Dateibaum zu bewegen und dort Dateien zu kopieren. Die wichtigsten Befehle von FTP sind:

open *Hostname*

: Stellt eine Verbindung zum gewünschten Host her. Dabei wird ein Passwort abgefragt.

close

: Schließt eine bestehende Verbindung.

dir

: Zeigt das aktuelle Inhaltsverzeichnis.

cd *Verzeichnis*

: Wechselt in das angegebene Verzeichnis.

lcd *Verzeichnis*

: Wechselt das Verzeichnis auf der Client Seite (local change dir).

pwd

: Print Working Dir - gibt das aktuelle Verzeichnis an.

get *Datei*

: Kopiert eine Datei vom Server zum Client.

put *Datei*

: Kopiert eine Datei vom Client zum Server.

del *Datei*

Löscht eine Datei auf dem Server.

binary

Wechselt in den Binärmodus.

ascii

Wechselt in den Textmodus.

quit

Beendet FTP.

Moderne FTP-Clients wie Netscape oder xftp steuern diesen Vorgang über eine graphische Benutzeroberfläche, die den Umgang mit ftp stark vereinfacht.

Eine große Bedeutung im Internet hat anonymes FTP (ohne Username und Passwort), das als Möglichkeit verwendet wird, um Public-Domain-Software downzuloaden.

telnet

Telnet (Terminal EmuLation over NETwork) dient dazu, Zugriff auf einen, am Netz angeschlossenen Rechner in Form einer Terminalsitzung zu liefern. Auf Unix-Systemen läuft auf der Clientseite das Programm telnet, auf der Serverseite ein Daemon namens telnetd. Der Telnet-Service liegt auf dem TCP-Port 23.

Auf Unix-Systemen wird heute das Kommando **rlogin** verwendet, das in etwa die gleiche Funktionalität besitzt, wie Telnet, dabei aber die Unix Eigenheiten besser unterstützt. In heterogenen Umgebungen, die nicht nur aus Unix-Systemen bestehen ist Telnet aber flexibler. Sicherer ist allerdings die Anwendung von **ssh** statt Telnet, da es die gesamte Kommunikation (inklusive der Login-Prozedur und Passwortübergabe) verschlüsselt abwickelt. Telnet überträgt alles, auch das Passwort unverschlüsselt.

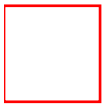
Eine Telnetsitzung wird gewöhnlich durch den Client mit dem Befehl

```
telnet Hostname
```

gestartet. Das Programm stellt dann eine TCP-Verbindung mit dem gewünschten Host her und kommuniziert dort mit dem Telnet-Daemon telnetd. Dieser Daemon stellt jetzt ein sogenanntes Pseudo-Terminal (oder auch Network Virtual Terminal - NVT) zur Verfügung. Dieses Terminal lässt sich genauso steuern, wie eine Unix-Konsole, so daß auch bildschirmorientierte Programme via Telnet benutzt werden können (z.B. vi, mc, ...)

Die interne Struktur des TCP-Segments

Im Gegensatz zu UDP braucht TCP als streamorientiertes Protokoll immer Informationen, an welcher Stelle des Datenstroms das Segment eingefügt werden soll. Außerdem ist Information über die Größe des Empfängerbuffers nötig, um zu verhindern, daß es zu Überläufen kommt. Es ist also ein entsprechend komplexer Header nötig:



Absender Portnummer

bezeichnet die Portnummer (/etc/services) des Anwenderschichtprotokolls, von dem der Datenstrom abgeschickt wurde.

Empfänger Portnummer

bezeichnet die Portnummer des Empfängerprotokolls auf der Anwendungsschicht.

Sequenznummer

Gibt die Position im Datenstrom an, an der dieses Segment eingefügt werden soll.

Bestätigungsnummer

Dient zur Bestätigung des Empfangs eines Segments. Dieses Feld enthält immer die Nummer, die im nächsten Segment als Sequenznummer stehen soll.

Daten-Offset

Gibt die Größe des Headers in 32 Bit Worten an. Ohne Optionen sind es 5. Damit kann genau bestimmt weden, wo der Datenbereich des Segments beginnt.

Flags

Verschiedene Flags zur Kommunikationssteuerung

Fenster

Mit diesem Wert wird die Größe des Buffers angezeigt, der von einem Endknoten für diese Verbindung reserviert wurde. Der sendende Knoten darf keinesfalls mehr Daten als die angegebene Puffergröße senden, ohne auf den Eingang einer Bestätigung zu

warten.

Prüfsumme

Eine Prüfsumme über das gesamte Segment (Daten und Header)

Dringlichkeitszeiger

Wird bei besonders dringend zu verarbeitenden Paketen gesetzt. Wenn gesetzt enthält dieses Feld als Wert die Endadresse des Datenfeldes, das als dringlich gilt.

Optionen

Verschiedene Optionen zur Kommunikation wie etwa die maximale Segmentgröße

Füllzeichen

Zum Auffüllen der Optionszeile auf 32 Bit

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.112.3

TCP/IP Konfiguration und Problemlösung

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Konfigurationseinstellungen und den Status verschiedener Netzwerkschnittstellen anzusehen, zu ändern und zu überprüfen. Dieses Lernziel beinhaltet die manuelle und automatische Konfiguration von Schnittstellen und Routing-Tabellen. Dies bedeutet im speziellen das Hinzufügen, Starten, Stoppen, Neustarten, Löschen und Rekonfigurieren von Netzwerkschnittstellen. Dies beinhaltet auch das Ändern, Listen und Konfigurieren der Routing-Tabelle und die manuelle Korrektur einer falsch gesetzten Default-Route. Kandidaten sollten auch in der Lage sein, Linux als DHCP-Client und TCP/IP-Host zu konfigurieren und Probleme im Zusammenhang mit der Netzwerkkonfiguration zu lösen.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/HOSTNAME` oder `/etc/hostname`
- `/etc/hosts`
- `/etc/networks`
- `/etc/host.conf`
- `/etc/resolv.conf`
- `/etc/nsswitch.conf`
- **ifconfig**
- **route**
- **dhcpcd, dhcpclient, pump**
- **host**
- **hostname (domainname, dnsdomainname)**
- **netstat**
- **ping**
- **traceroute**

- **tcpdump**
- die Netzwerk-Scripts, die während der Systeminitialisierung ausgeführt werden.

Netzwerkkonfiguration ist etwas, was im Normalbetrieb immer automatisch beim Systemstart abläuft. Init-Scripts übernehmen die Konfiguration der Schnittstellen, das Anlegen der Routen und vieles mehr. Trotzdem ist das Wissen um die manuelle Konfiguration wichtig, erstens für Problemlösungen und zweitens, weil damit auf die Schnelle auch eine Umkonfiguration von Netzwerkkarten oder ein experimenteller Aufbau möglich ist.

Notwendig ist also beides, das Wissen um die manuelle und automatische Konfiguration. Im Prinzip ist die zweite Frage kein großer Wurf mehr, wenn die erste beantwortet ist. Da die automatische Konfiguration die selben Befehle und Mechanismen einsetzt, wie die manuelle - nur eben in Scripts - sollte das Verständnis der automatischen Konfiguration kein Problem darstellen, wenn die Befehle aus der manuellen bekannt sind.

Manuelle Konfiguration von Netzwerkkarten

Die erste Grundvoraussetzung für die Konfiguration von Netzwerkschnittstellen ist, daß TCP/IP im Kernel aktiviert ist. Das ist eigentlich immer der Fall, selbst bei Rechnern, die nicht für Netze geplant waren, weil unter Linux viele lokale Dienste ohne diese Fähigkeit nicht funktionieren würden.

Die Darstellung der notwendigen Konfigurationsarbeiten wird sich jetzt auf "normale" Netzwerkkarten (Ethernet) beziehen, die Einstellungen für andere Netzwerkschnittstellen, seien es andere Kartentypen (Token Ring, FDDI, ...) oder andere Schnittstellen wie ppp oder slip, werden mit den selben Befehlen vorgenommen.

Damit die Konfiguration einer Netzwerkkarte überhaupt funktionieren kann, muß der Kernel die Karte zuallererst mal erkannt haben. Das heißt, daß der Kernel die Unterstützung für eine bestimte Netzwerkkarte bereits fest eingebaut haben muß oder daß das entsprechende Modul geladen sein muß. Diese Technik wurde bereits im Abschnitt 1.105.1 besprochen. Wir gehen hier also davon aus, daß das Modul für die zu konfigurierende Netzwerkkarte bereits geladen ist.

Das IP-Protokoll definiert eine abstrakte Schnittstelle *interface* um auf verschiedene Hardware zugreifen zu können. Diese Schnittstellen bieten eine Reihe von standardisierten Operationen an, die alle mit Versand und Empfang von Daten zu tun haben. Durch das Laden von Gerätetreibern (Modulen) werden diese Schnittstellen mit physikalischen Geräten (Netzwerkkarten o.ä.) verbunden. Dadurch entstehen symbolische Schnittstellennamen wie *eth0, eth1* für Ethernetkarten, *tr0, tr1* für Token Ring Karten oder *arc0, arc1* für Arcnet-Karten. Der lokale Loopback heißt *lo*. Selbst serielle Verbindungen ins Internet mit ppp oder slip haben dann Interfacenamen wie *sl0, sl1* oder *ppp0, ppp1*.

Diese Zuweisung geschieht in der Regel schon beim Booten, es können bei einem modularen Kernelaufbau aber auch während der Laufzeit

des Kernels noch Gerätetreiber geladen werden. Dann erfolgt natürlich auch die Zuweisung von symbolischen Namen erst während der Laufzeit.

Konfigurieren des Interfaces

Nachdem wir ein funktionsfähiges Interface haben, z.b. *eth0* muß jetzt dafür gesorgt werden, daß dieser Schnittstelle eine IP-Adresse zugewiesen wird. In der Regel geschieht das natürlich automatisch, aber um zu sehen was in den Init-Scripts passiert, ist es wichtig, diese Schritte auch mal von Hand durchgeführt zu haben.

Zum Konfigurieren eines Interfaces gibt es das Programm **ifconfig**, das einem beliebigen Interface IP-Adresse, Netzmaske und Broadcast-Adresse zuweist. Die Anwendung ist erstmal simpel:

```
ifconfig lo 127.0.0.1
ifconfig eth0 192.168.200.55
```

Damit haben wir dem Interface *lo*, also dem lokalen Loopback die ihm zustehende Adresse `127.0.0.1` zugewiesen, die Ethernetkarte *eth0* bekommt die Adresse `192.168.200.55`. Besser wäre noch gewesen, gleich die Netmask und Broadcast-Adresse mit anzugeben, mit den Schlüsselwörtern *netmask* und *broadcast* ist das möglich:

```
ifconfig eth0 192.168.200.55 broadcast 192.168.200.255 netmask 255.255.255.0
```

Somit wäre unser Interface jetzt komplett konfiguriert. Das Programm **ifconfig** kann aber noch mehr. Wird es ohne Optionen aufgerufen, so zeigt es Informationen zu allen bestehenden aktiven Netzschnittstellen an. (mit der Option -a werden auch die Schnittstellen dargestellt, die *down* geschalten sind) Mit

`ifconfig` *Interface*

werden Informationen zum angegebenen *Interface* angezeigt. Neben der Informationsausgabe kann das Programm **ifconfig** noch folgende Optionen verarbeiten:

down

Schaltet ein Interface schlagartig aus. Es ist danach für den Kernel nicht mehr erreichbar. Vorsicht, bei alten Versionen von **ifconfig** werden Routing Einträge nicht mitgelöscht, es können also noch Routen existieren, denen kein Interface zugeordnet ist.

up

Schaltet ein abgeschaltetes Interface wieder ein.

pointopoint *ADRESSE*

für SLIP und PPP, beides Protokolle für serielle Verbindungen. Hier ist das Interface etwas anders zu steuern, weil ja sozusagen nur ein Netz aus zwei Rechnern (Point to Point) besteht.

metric ***NUMMER***

Nur für RIP. RIP summiert alle anfallenden *metric* Werte einer Verbindung um daraus die Kosten zu errechnen (heute selten benutzt) und damit Routenwahl auch hinsichtlich der Kosten zu optimieren.

mtu ***BYTES***

Maximum Transmission Unit - Größte Übertragungseinheit des verwendeten Netzprotokolls. Bei Ethernet (802.3) 1500, bei SLIP 296.

promisc

Schaltet ein Interface in den *promiscuos mode* in dem alle Pakete des Netzwerks, egal ob an diesen oder einen anderen Rechner geschickt, empfangen werden. Brauchbar zur Analyse des Netzverkehrs, der Fehlersuche, aber auch zum Abhören von Netzleitungen.

Alle diese Optionen werden nach der Nennung des Interfacenamens angehängt, z.B.

```
ifconfig eth0 metric 2 mtu 1024
```

Der Befehl **ifconfig** ist also sowohl zu Konfiguration eines Interfaces, als auch zur Diagnose ein wichtiges Hilfsmittel.

Erstellen der Routen

Nachdem die Interfaces konfiguriert worden sind, müssen noch die Routing-Tables (Routen-Tabellen) erstellt werden. Das geschieht mit dem Programm **route**. Die Tabelle existiert nicht als Datei, sondern wird im Kernelspeicher verwaltet. Das heißt, der Aufruf von **route** muß also bei jedem Start erfolgen.

Die heutigen modernen Versionen von **ifconfig** erledigen bereits das Anlegen der Route ins eigene Netz sobald eine Schnittstelle mit einer IP-Adresse versehen wird. Diese Aufgabe musste bei älteren Systemen auch mit dem Befehl **route** erledigt werden.

Wird **route** ohne Parameter (oder nur mit `-n`) aufgerufen, so zeigt es den aktuellen Routing-Table an. (`-n` zeigt Adressen immer nummerisch, auch wenn Hostnamen bekannt sind)

Ansonsten muß **route** immer entweder mit dem Schlüsselwort `add` oder `del` versehen werden. `add` steht für das Hinzufügen einer Route, `del` für das Löschen einer Route.

```
route add [-net | -host] XXXX [gw GGGG]
          [metric MMMM] [netmask NNNN] [dev DDDD]
```

oder

```
route del XXXX
```

Dabei stehen `XXXX` für die Route (Netz- oder Hostadresse), `GGGG` für die Adresse eines Gateways, `MMMM` für einen numerischen Wert, der als metric-Wert benutzt werden soll, `NNNN` für eine Netzmaskierung und `DDDD` für ein TCP/IP Interface wie etwa *eth0*.

Für einen einfachen Rechner in einem lokalen Netz würde also der folgende Eintrag genügen:

```
route add -net 192.168.200.0
```

Damit wäre eine Route definiert, die alle Pakete, die ans Netz 192.169.200.0 gerichtet sind an die Ethernetkarte schickt. Das weiß das System, weil die Ethernetkarte ja die Adresse 192.168.200.55 hat, also die gleiche Netzadresse. Dieser Befehl ist bei modernen Versionen von **ifconfig** unnötig geworden, weil diese Route eben automatisch bei der Konfiguration angelegt wird.

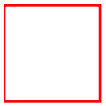
Nehmen wir an, in unserem Netz ist ein Gateway (z.B. `192.168.200.77`) ans Internet angeschlossen. Wir brauchen jetzt also zwei Routen, eine ins lokale Netz und einen an den Gateway. An diesen schicken wir alle Pakete, die nicht fürs lokale Netz gedacht sind.

```
route add -net 192.168.200.0
route add default gw 192.168.200.77
```

Alle Pakete, die wir losschicken und die nicht für das lokale Netz bestimmt sind, werden jetzt direkt an den Gateway `192.168.200.77` geschickt. Das Schlüsselwort `default` kann auch mit `-net 0.0.0.0` ausgetauscht werden, der schon bekannten Adresse der *default route*.

Auch in diesem Beispiel ist die erste Zeile bei modernen **ifconfig** Versionen unnötig.

Jetzt bleibt nur noch die Frage, wie die Routing-Tables eines Gateways konfiguriert werden müssen. Nehmen wir an, der Gateway `192.168.200.77` hat eine FDDI-Karte und hängt direkt am Rechner des Providers. Der Provider hat eine eigene Netzadresse für sein FDDI-Netz. Nehmen wir an, die Provider-Netzadresse ist `192.100.200.0` Der Rechner des Providers (`192.100.200.1`) leitet alle Pakete ans eigentliche Internet weiter. Das heißt, er dient als nächster Gateway ins Internet. Unser Gateway hat eine FDDI-Karte (Interface *fddi0*) mit der Adresse `192.100.200.7`.



Also müssen wir in unserem Gateway drei Routen konfigurieren. Eine ins lokale Netz, eine ins FDDI-Netz (beide werden wieder automatisch durch **ifconfig** angelegt und sind hier nur für ein besseres Verständnis angegeben) und eine Default-Route zum Providergateway. Selbstverständlich müssen hier auch beide Interfaces richtig konfiguriert sein:

```
ifconfig eth0 192.168.200.77 broadcast 192.168.200.255 netmask 255.255.255.0
```

```
ifconfig fddi0 192.100.200.7 broadcast 192.100.200.255 netmask 255.255.255.0
route add -net 192.168.200.0
route add -net 192.100.200.0
route add default 192.100.200.1
```

Der Gateway hat jetzt also zwei statische Routen in die jeweiligen Netze, die er verbindet (das geht selbstverständlich auch mit zwei Ethernetkarten) und eine Default Route, die wiederum auf den Gateway des Providers verweist. Wenn jetzt ein beliebiger Rechner in unserem lokalen Netz ein Paket an eine gänzlich unbekannte Adresse schickt (etwa `123.45.67.89`) dann wird sie (weil es keine feste Route dorthin gibt) über die Default-Route geschickt. Diese ist mit unserem Gateway verbunden. Der hat auch keine feste Route zu `123.45.67.89`, schickt das Paket wiederum auf seine Default Route, die mit dem Gateway des Providers verbunden ist. Der gibt das Paket seinerseits ans Internet weiter.

Das Programm **route** ist - wie schon **ifconfig** in der Praxis ein wichtiges Hilfsmittel zur Diagnose. Nachdem die Routen in der Regel beim Starten des Rechners automatisch gesetzt werden, ist die Anwendung zu Diagnosezwecken sehr viel häufiger, als die manuelle Erstellung von Routen.

Das Programm netstat

Dieses Programm gibt Informationen über den Status bestimmter Netzverbindungen zurück. Interessant sind folgende Möglichkeiten:

netstat -r oder -rn Wie route oder route -n Die angezeigten Flags haben folgende Bedeutung:

- **G** Route ist Gateway
- **U** Interface ist UP
- **H** Nur ein Host kann über die Route erreicht werden (PPP/SLIP/loopback)
- **D** Route wurde von ICMP eingetragen
- **M** Route wurde von ICMP verbessert (modified)

netstat -i Gibt die Statistik der Interfaces an, Flags haben folgende Bedeutung:

- **B** Broadcast ist gesetzt
- **L** ist Loopback
- **M** promiscous mode - Alle Pakete werden empfangen
- **O** ARP ist ausgeschaltet
- **P** Point to Point Verbindung
- **R** Running
- **U** Up

netstat -t, -u, -w, -x, -a zeigt die aktiven Sockets für TCP, UDP, RawIP, Unix oder Alle.

Automatische Konfiguration über Init-Scripts

All die oben erwähnten Befehle müssen natürlich nicht jedesmal von Hand eingegeben werden, wenn ein Linuxrechner startet. Sie werden von entsprechenden Init-Scripts gestartet, sobald in einen Runlevel gewechselt wird, der das Netzwerk aktivieren soll.

Die unterschiedlichen Distributionen gehen hier denkbar unterschiedliche Wege. Gemeinsam ist ihnen, daß die Einstellungen für die Netzwerkkarten (Adressen und Routen) in bestimmten Dateien (unter `/etc`) eingetragen werden, die dann von einem entsprechenden Init-Script (z.B. `/etc/init.d/networking`) ausgelesen werden. Das Init-Script startet dann die Befehle **ifconfig** und **route** mit den gefundenen Werten aus den Dateien.

Eine weitere Vereinfachung ist mit den Programmen **ifup** und **ifdown** möglich. Beide Programme sind Frontends für **ifconfig** und **route**. Im Verzeichnis `/etc/network` erwarten diese Programme eine Datei mit Namen `interfaces`, die Informationen über die benutzten Netzwerkschnittstellen beinhalten. In dieser Datei steht beispielsweise:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
      address 192.168.100.123
      netmask 255.255.255.0
      network 192.168.100.0
      broadcast 192.168.100.255
      gateway 192.168.100.20
```

Mit dieser Information kann **ifup** dann entsprechend die Netzwerkschnittstellen mit Adressen versehen und die notwendige Route zum Gateway (default route) setzen.

Einsatz eines DHCP-Servers

Wenn im Netz ein DHCP-Server läuft, dann ist die gesamte Konfiguration der Netzwerkkarten nicht mehr nötig. Ein solcher Server hält alle notwendigen Informationen bereit, die ein Rechner benötigt, um seine Netzwerkkarte zu konfigurieren. Beim Hochfahren des Systems wird ein Broadcast ins Netz gesendet, um einen DHCP-Server zu finden. Der Server antwortet daraufhin mit einem Paket, das alle notwendigen Angaben zur Konfiguration beinhaltet.

Die Prüfungsziele für LPI102 beinhalten nicht die Installation eines DHCP-Servers, sondern nur die Anbindung eines Rechners an einen bestehenden Server.

Es gibt unterschiedliche Programme, die einen Linux-Rechner zum DHCP-Client machen. Die bekanntesten sind

- **dhcpcd**
  Der DHCP-Client-Daemon. Er arbeitet vollkommen selbstständig und durchsucht das Netz nach einem DHCP-Server. Anschließend weisst er jeder gefundenen Netzwerkschnittstelle eine entsprechende IP-Adresse zu.
- **dhclient**
  Auch dieses Programm arbeitet als Daemon, es bezieht seine Informationen, welche Interfaces über DHCP konfiguriert werden sollen aus der Konfigurationsdatei `/etc/dhclient.conf`.
- **pump**
  Auch **pump** ist ein Daemon, der Netzwerkinterfaces über DHCP konfiguriert. Er kann aber auch als Alternative mit dem *bootp* Protokoll arbeiten, das eine ähliche Aufgabenstellung wie DHCP besitzt, allerdings auch das Booten plattenloser Workstations unterstützt. **pump** bezieht seine Konfigurationsinformationen aus der Datei `/etc/pump.conf`.

Jeder dieser Daemonen wird über ein Init-Script beim Systemstart gebootet und bleibt die ganze Zeit im Speicher. Das ist notwendig, weil DHCP-Server in regelmäßigen Abständen überprüfen, ob ein Rechner noch aktiv ist, um gegebenenfalls eine wieder frei gewordene Adresse neu vergeben zu können.

Konfigurationsdateien für Netzwerke

Neben den Init-Scripts und anderer Netzwerk-relevanten Einstellungen, die von Distribution zu Distribution unterschiedlich sind, gibt es eine Reihe Konfigurationsdateien, die allen Linux-Versionen (und auch allen anderen Unixen) gemeinsam sind. Diese Dateien werden hier jetzt der Reihe nach besprochen.

/etc/HOSTNAME oder /etc/hostname

Diese Datei enthält den Hostnamen des Rechners. Beim Systemstart wird ein Init-Script ausgeführt, das diese Datei ließt und den entsprechenden Namen darin als Hostnamen setzt. Das dazu verwendete Programm heißt ebenfalls **hostname** und ermöglicht es auch, im laufenden Betrieb den Hostnamen zu erfragen (ohne weitere Parameter) oder ihn zu verändern.

Das Programm **hostname** gibt als Ausgabe nur den Hostnamen, ohne Domainnamen an. Soll der volle Name (full qualified domain name) angegeben werden, so muß **hostname** mit der Option `--fqdn` aufgerufen werden. Der Domainname alleine kann mit dem Befehl **dnsdomainname** erfragt werden. Dieser Befehl ist aber nicht in der Lage den Domainnamen zu verändern, da es stark von der Organisation des Netzes abhängt, wo dieser Domainname eingestellt ist. Der Befehl **domainname** gibt statt der echten DNS-Domain den Namen der NIS-Domain an.

/etc/hosts

Die Datei `/etc/hosts` enthält IP-Adressen und die dazu passenden Hostnamen. Überall, wo eigentlich IP-Adressen erwartet werden, können stattdessen auch Hostnamen angegeben werden, wenn diese in der Datei `/etc/hosts` angegeben sind. Die Datei ist also - übertragen gemeint - eine Art Mini-Nameserver für das lokale Netz. Der Aufbau ist einfach, jede Zeile enthält zuerst eine IP-Adresse, dann den dazu passenden Namen und eventuelle Aliase auf diesen Namen.

```
192.168.100.1    marvin.mydomain.com marvin
```

bedeutet also, daß immer, wenn der Name `marvin.mydomain.com` oder der Name `marvin` verwendet wird, dieser Name durch die Adresse `192.168.100.1` ersetzt wird.

/etc/networks

Die Datei `/etc/networks` entspricht der Datei `/etc/hosts`, mit dem Unterschied, daß hier nicht Hostnamen, sondern Netzwerknamen verwendet werden. Diese Netzwerknamen werden mit Netzwerkadressen verbunden.

Im Unterschied zu `/etc/hosts` werden in dieser Datei zuerst die Namen und dann die Adressen angegeben. Eventuelle Aliase werden nach der Adresse angegeben:

```
buero1 192.168.1.0
buero2 192.168.2.0 meinnetz
backbone 192.168.100.0
```

Die angegebenen Adressen müssen Netzwerkadressen sein, also alle Bits des Hostadressenteils auf 0 gesetzt haben.

/etc/host.conf

Die Auflösung von Hostnamen zu IP-Adressen wird von einer speziellen Library erledigt (resolv+). Diese Library ist es auch, die entsprechend Nameserver und die Datei `/etc/hosts` abfrägt, wenn statt einer IP-Adresse ein symbolischer Name gefunden wurde.

Die Datei `/etc/host.conf` ist die Konfigurationsdatei für diese Bibliothek. Die wichtigste Aufgabe dieser Datei ist es, die Suchreihenfolge festzulegen, in der Namen in IP-Adressen aufgelöst werden. Entweder wird zuerst der Nameserver gefragt, und dann die Datei `/etc/hots`, oder umgekehrt. Diese Einstellung wird über die Zeile

```
order hosts,bind
```

erledigt. Das Beispiel gibt an, daß zuerst die Datei `/etc/hosts` und erst dann der Nameserver (bind) gefragt werden soll. Ist umgekehrt

gewünscht, daß zuerst der Nameserver gefragt wird, so hieße die Zeile

```
order bind,hosts
```

Weitere Parameter sind:

multi

Gültige Werte sind `on` und `off`. Wenn `on` gesetzt ist, liefert die Resolverbibliothek alle für den gesuchten Host gültigen Adressen aus `/etc/hosts` zurück, anstatt nur den ersten gefundenen Eintrag. Der Standardwert für diesen Parameter ist `off`, da seine Verwendung auf Installationen mit sehr großer /etc/hosts zu Performanceproblemen führen kann.

nospoof

Gültige Werte sind `on` und `off`. Wenn dieser Parameter auf `on` gesetzt ist, versucht die Resolverbibliothek, das Spoofing von Hostnamen zu unterbinden, um die Sicherheit von rlogin und rsh zu verbessern. Nachdem für einen Hostnamen die Adresse gefunden wurde, wird geprüft, ob für diese Adresse auch genau dieser Hostname gefunden werden kann. Liegt keine eindeutige Zuordnung vor, schlägt die Namensauflösung fehl.

spoofalert

Wenn dieser Parameter auf on gesetzt und gleichzeitig **nospoof** aktiv ist, protokolliert die Resolverbibliothek Fehler im Zusammenhang mit dem Spoofing-Schutz an Syslog. Der Standardwert hierfür ist off.

Zumeist enthält diese Datei nur zwei Zeilen, die die Reihenfolge definieren und das **multi** auf `on` setzen.

/etc/resolv.conf

Diese Datei wird auch von der Resolver-Library ausgelesen. Die wichtigste Aufgabe ist die Angabe der verwendeten Nameserver. Es können hier bis zu drei verschiedene Nameserver angegeben werden, die dann in der Reihenfolge der Angaben befragt werden, wenn ein DNS-Lookup stattfindet.

Der hierfür verwendete Befehl lautet

nameserver *IP-Adresse*

Dieser Befehl kann bis zu dreimal in der Datei auftauchen.

Ein weiterer Befehl für die vernünftige Arbeit mit dem Resolver ist

search *Domain-Name*

Hier wird die lokale Domain (ohne Hostnamen) angegeben, um Namen, die ohne Domainnamen angegeben wurden, als lokale Namen festzulegen. Genauer gesagt, werden alle Rechnernamen (mit oder ohne Domainnamen) zunächst einmal mit dieser Endung versehen, die dort angegeben wurde. Erst wenn ein Rechner nicht mit der Endung gefunden wurde, wird nach einem Rechner ohne diese Endung

gesucht.

Normalerweise enthält die Datei `/etc/resolv.conf` nur diese beiden Einträge.

/etc/nsswitch.conf

Verschiedene Funktionen der C-Standard-Library müssen anhand verschiedener Konfigurationsdateien konfiguriert werden. Traditionell wird das z.B. mit Dateien wie `/etc/passwd` erledigt. Später wurden dann zusätzliche Dienste eingeführt, die auch bestimmte Informationen anbieten, wie etwa NIS/YP, das auch Informationen über Usernamen und UIDs bereitstellen kann, nur eben netzweit und nicht nur lokal. Um festzulegen, welche dieser Dienste in welcher Reihenfolge verwendet werden sollen, existiert die Datei `/etc/nsswitch.conf`. Ihr Mechanismus ist in etwa zu vergleichen mit der Angabe der Reihenfolge in der Datei `/etc/host.conf`, nur daß sie sich nicht nur auf Namensauflösung, sondern auf alle verwendeten Konfigurationsmöglichkeiten bezieht.

Konkret werden die folgenden Datenbanken von `/etc/nsswitch.conf` verwaltet (in Klammern immer die normalerweise verwendeten Konfigurationsdateien):

aliases
: Mail-Aliases von **sendmail** (`/etc/aliases`).

ethers
: Ethernet-Adressen

group
: User-Gruppen (`/etc/group`).

hosts
: Hostnamen und Adressen (`/etc/hosts`).

netgroup
: Netzweite Listen von Usern und Hosts, die für Bestimmungen von Zugriffsrechten wichtig sind.

network
: Netzwerknamen und -adressen (`/etc/networks`).

passwd
: User-Passwörter und andere Informationen (`/etc/passwd`).

protocols
: Netzwerkprotokolle (`/etc/protocols`).

publickey

Public- und Secret-Keys für Secure_RPC

rpc

Remote Procedure Call Namen und Portnummern (`/etc/rpc`).

services

Portnummern von Netzwerkdiensten (`/etc/services`).

shadow

Shadow-Passwörter (`/etc/shadow`).

In der Datei kann jetzt für jede dieser Datenbanken die Suchreihenfolge angegeben werden, ob zuerst über NIS, dann über die Dateien (files) oder umgekehrt gesucht werden soll.

Troubleshooting mit tcpdump

Wenn Probleme in einem Netz auftauchen, so kann mit Hilfe des Programms **tcpdump** jedes einzelne Paket des Netzes protokolliert und analysiert werden. Dazu wird die Netzwerkkarte in den *promisquous-mode* geschaltet, das heißt, sie gibt jedes Paket, auch wenn es nicht die eigene Mac-Adresse hat, an die Vermittlungsschicht weiter.

tcpdump zeigt jetzt die empfangenen Paket-Header an und ermöglicht so eine Diagnose der aufgetretenen Fehler. Die Anwendung ist ziemlich kompliziert und würde den Rahmen dieser Darstellung sprengen. Es genügt, zu wissen, daß es dieses Programm gibt und daß es folgendermaßen aufgerufen wird.

tcpdump `-i` *Interface*

Damit werden alle Pakete des genannten Interfaces abgehört. Das Interface wird mit seinem symbolischen Namen angegeben, also beispielsweise *eth0*.

tcpdump hat eine eigene Art Abfragesprache, die Befehle ermöglicht wie

```
tcpdump host foo
```

Zeigt nur Pakete an den oder vom Rechner foo.

```
tcpdump host foo and bar
```

Zeigt nur Pakete, die zwischen den Rechnern foo und bar ausgetauscht werden.

Andere Programme

Die in der Liste für dieses Prüfungsziel erwähnten Programme **host, ping** und **traceroute** wurden bereits im Abschnitt 1.112.1 besprochen.

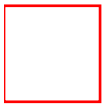
Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

Zurück

Die interne Struktur des IEEE 802.3 Frames

Der IEEE802.3 Standard (Ethernet) bedient sich auch der Aufteilung des zu transportierenden Datenstromes in Pakete. Diese Pakete werden auch Rahmen (Frames) genannt, weil sie vor und nach den eigentlichen Daten noch Steuerinformationen ablegen. Die folgende Abbildung zeigt das verwendete Rahmenformat:



Jeder Rahmen beginnt mit einer **Präambel** von 7 Bytes, die jeweils die Bitfolge `10101010` enthalten. Durch die Kodierung dieser Bitfolge (Manchester) entsteht eine 10 MHz Schwingung, die 5,6 µs dauert. Dadurch kann sich der Taktgeber des Senders mit dem des Empfängers synchronisieren.

Als nächstes folgt ein Rahmenstartbyte (**Rahmenbegrenzer**) also die Markierung, daß hier der Frame beginnt. Dieses Byte enthält immer die Bitfolge `10101011`

Die beiden folgenden Blöcke, **Zieladresse** und **Quelladresse** enthalten die physikalischen Netzwerkadressen (Ethernetadressen, MAC-Adressen) des Empfängers und Senders. In etwa also das, was ein Briefumschlag enthält, Absender und Empfängerangaben...

Dabei wird intern bei der Zieladresse noch zwischen Einzeladressen, Gruppenadressen und Broadcastadressen unterschieden, also Pakete, die entweder an einen bestimten Rechner gehen, oder an eine bestimmte Gruppe von Rechnern oder an alle Rechner im Netz.

Im nächsten Feld (**Länge**) steht die Angabe, wie lang das folgende Datenpaket ist, gültige Werte sind hier 0 bis 1500. Theoretisch sind also Datenpakete mit der Länge 0 möglich, aus physikalischen Gründen ist aber eine minimale Länge von 46 Bytes erforderlich. Dafür sorgt dann das **Pad** Feld, das Datenfelder mit einer Länge unter 46 Bytes bis auf desen Minimalwert auffüllt.

Im **Datenfeld** zwischen Länge und Pad stehen die eigentlich zu übertragenen Daten. Durch die Angabe der tatsächlichen Länge dieses Feldes im Feld Länge kann der Empfänger des Paketes mit unterschiedlich großen Datenfeldern immer richtig umgehen.

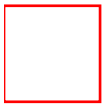
Das letzte Feld des Rahmens ist die **Prüfsumme**, die vom Sender errechnet wurde und sich natürlich auf das Datenfeld bezieht. Der Empfänger kann nun das Datenfeld ebenso überprüfen und, falls er zu einem anderen Ergebnis kommt, eine erneute Zusendung des Pakets

beantragen. Dieses Verfahren wird übrigens in der IEEE Norm 802.2 beschrieben, die für alle drei folgenden Normen (802.3, 802.4, 802.5) angewendet wird.

Zurück

Der Aufbau von IP-Datagrammen

Das Internet Protokoll (IP) packt den Datenstrom, den es aus der Transportschicht bekommt wiederum in Pakete, die aber diesmal mit den logischen IP-Adressen versehen sind statt mit den physikalischen Netzadressen. Die interne Struktur eines solchen Datagramms (Paketes) ist in der folgenden Graphik dargestellt.



Der Aufbau eines IP-Datagramms entspricht also nicht dem Frame System, sondern stellt nur einen Header (Kopfteil) zur Verfügung. Nach dem Header folgen nur noch Daten, keine Frame-Ende-Zeichen o.ä.

Die einzelnen Felder sollen kurz noch dargestellt werden:

Version - 4 Bit

Dieses Feld enthält die Versionsnummer des IP. Aktuelle Version ist 4, bald kommt 6...

Internet Header Length (IHL) - 4 Bit

Gibt die Länge des Headers in 32 Bit Worten an. Damit kann ermittelt werden, ob Optionen gesetzt sind. Normalerweise (ohne Optionen) hat dieses Feld den Wert 5.

TOS (Type of Service - Diensttyp) - 8 Bit

Enthält Flags, die zur Steuerung der Zuverlässigkeit des Netzes dienen. Sie werden automatisch erstellt.

Gesamtlänge - 16 Bit

Enthält die Gesamtlänge des Datagramms (incl. Kopf) in Byte. Aus der Breite dieses Feldes (16 Bit) ergibt sich so eine maximale Größe von Datagrammen von 65535 Byte. Dieser Wert liegt weit über dem maximalen Wert der einzelnen Netzsysteme, daher reichen 16 Bit aus.

Kennung - 16 Bit

Jedes IP-Datagramm muß eine eigene Kennung haben, um beim Wiederzusammenbau richtig eingeordnet zu werden.

Flags - 3 Bit

Flags zur Steuerung der Fragmentierung.

Fragment-Offset - 13 Bit

Unter Fragmentierung von Datagrammen ist zu verstehen, daß einzelne Datagramme, die durch Gateways in unterschiedliche Netze geroutet werden, dort oft unterschiedlichen Größenbestimmungen unterliegen. IP fragmentiert solche Datagramme und baut sie wieder zusammen. Dieses Feld enthält die Information, an welcher Stelle das Fragment ursprünglich war.

Lebensdauer - 8 Bit

In Netzen wie dem Internet, in denen viele Datagramme auf unterschiedlchsten Wegen versuchen zum Ziel zu kommen, muß eine maximale Lebensdauer haben, sonst werden sie unendlich oft geroutet. Dieses Feld enthält eine Zahl, die vom Sender eingetragen wurde. Jeder Gateway muß diese Zahl um eins herunterzählen, wenn er ein Datagramm routet. Ist der Wert Null, so darf kein Router das Datagramm mehr weiterleiten. Übliche Startwerte sind 32 oder 4 (nur lokale Netze setzen so niedrige Lebensdauer an)

Protokoll - 8 Bit

Dieses Feld gibt an, von welchem Protokoll der Transportschicht dieses Datagramm kommt. Das ist wichtig für den Empfänger, weil ja mehrere Netzanwendungen gleichzeitig laufen können. Nur durch diese Information kann das Empfänger IP das Datagramm an das richtige Transportschichtprotokoll weiterleiten. Übliche Werte sind:

- 17 - UDP
- 6 - TCP
- 1 - ICMP

Kopf-Prüfsumme - 16 Bit

Prüfsumme über den Inhalt des Headers, nicht der Daten. Sicherung der Daten wird von der Transportschicht überprüft.

Sender IP-Adresse - 32 Bit

32 Bit IP-Adresse des Senders

Empfänger IP-Adresse - 32 Bit

32 Bit IP-Adresse des Empfängers

Optionen - <= 32 Bit

Hier können verschiedene Optionen gesetzt werden, die mit Debugging (Fehlersuche) oder Routenprüfung zu tun haben.

Füllzeichen

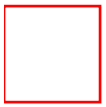
Füllt den Bereich zwischen den jeweils gesetzten Optionen und dem Ende des 32 Bit Wortes auf.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

Die interne Struktur der UDP-Message

UDP als datagramorientiertes Protokoll benötigt keinerlei Informationen über Sequenznummern oder ähnliches, eine sehr einfache Headerstruktur genügt hier:



Quellportnummer
: bezeichnet die Portnummer (/etc/services) des Anwenderschichtprotokolls, von dem die UDP-Message abgeschickt wurde.

Zielportnummer
: bezeichnet die Portnummer des Empfängerprotokolls auf der Anwendungsschicht.

Länge
: Hier steht die Länge der gesamten UDP-Message

Prüfsumme
: Eine Prüfsumme über das Datenfeld

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.112.4

Konfiguration von Linux als PPP-Client

Beschreibung: Prüfungskandidaten sollten die Grundlagen des PPP-Protokols verstehen und in der Lage sein, PPP für ausgehende Verbindungen zu konfigurieren und zu verwenden. Dieses Lernziel beinhaltet die Beschreibung der Sequenz des Verbindungsaufbaus (bei gegebenem Login-Beispiel) und das Einrichten von automatisch bei Verbindungsaufbau auszuführenden Kommandos. Ebenfalls enthalten ist die Initialisierung und die Beendung einer PPP-Verbindung mittels Modem, ISDN oder ADSL und die Einstellung der automatischen Neuverbindung bei Verbindungsabbruch.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

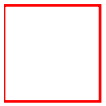
- `/etc/ppp/options.*`
- `/etc/ppp/peers/*`
- `/etc/wvdial.conf`
- **/etc/ppp/ip-up**
- **/etc/ppp/ip-down**
- **wvdial**
- **pppd**

Normalerweise werden IP-Verbindungen über Netzwerkkarten hergestellt. Die Protokolle von TCP/IP sind für den Umgang mit Netzwerken geschrieben, sie arbeiten z.B. mit den MAC-Adressen (ARP) und mit der Aufteilung eines Datenstroms in Pakete, wie es bei Netzen üblich ist.

Um zwei Computer statt über Netzwerkkarten über serielle Schnittstellen zu vernetzen, existieren mehrere Protokolle wie **SLIP** (Serial Line IP) und **PPP** (Point to Point Protocol). **SLIP** ist veraltet und wird nur noch sehr selten eingesetzt. Heute wird meist **PPP** benutzt.

Die wirkliche Aufgabenstellung von **PPP** ist es nicht, zwei alleinstehene Computer über eine serielle Leitung zu vernetzen. Heute sind Netzwerkkabel und -karten so billig, daß es sich nicht lohnen würde eine solche Art der Vernetzung zu realisieren. Zumal damit nur zwei

Rechner vernetzt werden können, das System also nicht ohne weiteres erweiterbar ist. Die wirkliche Aufgabe von **PPP** ist es, Rechner über serielle Leitungen mit anderen Rechnern zu verbinden, die an ein tatsächliches Netz angeschlossen sind, so daß auch der einzelne Rechner Zugriff auf das echte Netz hat. In der Regel ist das echte Netz das Internet und die serielle Leitung ist eine Modem/ISDN/ADSL Verbindung zum Provider:



Das **PPP** Protokoll ist ein spezielles Protokoll zur Realisierung einer IP-Verbindung über ein Medium, das nur zwei Kommunikationspartner verbindet. Also etwa serielle oder parallele Verbindungen. Spezielle Versionen von **PPP** sind für ISDN (syncPPP) oder für DSL (PPPoE) verfügbar, das klassische **PPP** ist typischerweise für Modemverbindungen gedacht.

Linux kann natürlich beide Rollen übernehmen, die der Workstation, die über **PPP** an einen Gateway angeschlossen wird (PPP-Client) oder die des Gateways, der eine **PPP**-Verbindung an ein echtes Netz knüpft (PPP-Server). Für die LPI102 Prüfung ist nur die Konfiguration eines PPP-Clients gefragt, diese Konfiguration wird im weiteren Verlauf dieses Abschnitts beschrieben.

Physikalischer Vorgang

Grundsätzlich ist der Vorgang des Aufbaus einer PPP-Verbindung in drei Teile aufzuteilen:

- Aufbau der Modemverbindung
- Loginvorgang
- Aufbau der PPP-Verbindung

Das heißt, zunächst einmal muß das Modem soweit gebracht werden, daß es eine Verbindung zu der bestimmten Nummer aufbaut. Das heißt das Modem muß initialisiert werden (`ATZ`) und die Initialisierung mit einem `OK` beantworten, anschließend muß die Nummer gewählt werden (`ATD` *`Nummer`*) und auf die Antwort `CONNECT` gewartet werden. Wenn diese Antwort empfangen wurde ist der erste Teil der Aufgabe erledigt. Damit haben wir eine bestehende serielle Verbindung zu dem anderen Rechner (Gateway). Auf dem Gateway hat das Modem abgenommen und ein normaler Login-Vorgang über getty wird jetzt gestartet. Das heißt, der Gateway schickt uns jetzt eine Meldung, die wahrscheinlich den Rechnernamen und das Wort `login:` beinhaltet. Diese Meldung wird von uns mit unserem Usernamen beantwortet. Daraufhin schickt der Gateway die Frage nach dem Passwort, die wir mit unserem Passwort beantworten. Wenn alles geklappt hat, ist der Login-Vorgang damit abgeschlossen.

Jetzt muß der eigentliche PPP-Daemon gestartet werden. Das muß auf beiden Seiten der Verbindung geschehen. Der Gateway hat den PPP-Daemon (**pppd**) als Startshell für den Usernamen eingetragen, er startet den Daemon also automatisch. Wir müssen den Daemon jetzt

von Hand starten mit

```
pppd Schnittstelle Geschwindigkeit defaultroute
```

Die Schnittstelle ist dabei unsere serielle Schnittstelle, auf der das Modem hängt, die Geschwindigkeit ist meist 38400 (für moderne schnelle Modems) und die Option `defaultroute` sorgt dafür, daß die Verbindung zum Gateway als Default Route eingetragen wird. Der PPP-Daemon initialisiert jetzt die neu entstandene symbolische Netzschnittstelle *ppp0* mit dem Programm **ifconfig** (mit einer dynamischen IP-Adresse, die er von der Gegenstelle bekommt) und legt die default route auf dieses Interface.

Jetzt besteht eine IP-Verbindung zu dem Garteway und unser Rechner hat Zugriff auf das hinter dem Gateway liegende Netz. PPP ist also jetzt aktiv.

Dieser Vorgang kann natürlich automatisiert werden und muß niemals von Hand ausgeführt werden. Diese Darstellung dient nur dazu, den Vorgang zu verstehen, der im nächsten Abschnitt beschrieben wird.

Automatischer Aufbau einer PPP-Verbindung mit chat

Damit der oben beschriebene Vorgang automatisiert werden kann, gibt es das Programm **chat**. Es dient dazu, eine Kommunikation mit dem Modem (oder jeden anderen seriellen Verbindung) zu automatisieren. **chat** erwartet seine Befehle aus einer Scriptdatei, die eine sehr einfache Struktur hat. Jede Zeile eines chat-scripts besteht aus einem Paar:

"Warte auf" "Sende"

Das heißt, **chat** wartet immer auf ein angegebenes Wort, das vom Modem kommt, und sendet nach dem Empfang dann das entsprechende Wort *Sende*. Nehmen wir ein Beispiel. Wir wollen eine Verbindung zu einem Gateway aufbauen, der die Telefonnummer 12345 hat. Unser Username auf diesem Gateway ist `hans`, unser Passwort ist `12sd45gf`. Das entsprechende **chat**-Script würde also folgendermaßen aussehen:

```
''   ATZ
OK   ATD12345
CONNECT  ''
ogin: hans
ssword: 12sd45gf
```

Das bedeutet, wir warten auf nichts und senden ein `ATZ`. Dadurch wird das Modem initialisiert. Das Modem antwortet mit einem `OK`. Sobald wir das `OK` empfangen haben, senden wir den Modembefehl `ATD12345`, der das Modem veranlasst, die Nummer 12345 anzuwählen. Sobald das gegenüberliegende Modem des Gateways abgenommen hat und die beiden Modems ihren Handshake beendet haben, antwortet unser Modem mit dem Wort `CONNECT`. Wenn wir das empfangen haben, schicken wir ein Return (`''`).

Jetzt wird der Gateway eine Loginaufforderung schicken. Wir wissen nicht, was genau er schickt, aber wir wissen, daß die Aufforderung mit den Buchstaben `ogin:` aufhört. Der Gateway könnte beispielsweise folgendes schicken:

```
Welcome to XYZ-Gateway
Login:
```

Es ist egal, was der Gateway sendet, sobald wir die Buchstaben `ogin:` empfangen, schicken wir wiederum den Usernamen `hans`. Der Gateway antwortet mit einer Aufforderung, ein Passwort einzugeben. Wir wissen nicht, ob er das groß oder klein schreibt, aber die letzten Buchstaben, die er schickt werden sicherlich `ssword:` sein. Sobald wir die empfangen haben, schicken wir das Passwort.

Mit diesem Befehl ist der Anmeldedialog abgeschlossen und die PPP-Verbindung wird aufgebaut.

Um den PPP-Daemon mit diesem Script zu starten, wird der Befehl

```
pppd "chat -f Scriptdatei" /dev/tty2 38400
```

eingegeben. *Scriptdatei* bezeichnet die Datei, die das oben besprochene Chat-Script enthält. Dadurch wird eine PPP-Verbindung automatisch aufgebaut. Der PPP-Daemon **pppd** wird aufgerufen und er erhält als ersten Parameter den Aufruf von Chat, mit der Scriptdatei. Die Angabe `/dev/tty2 38400` bezieht sich auf die serielle Schnittstelle, an der das Modem hängt und die dort verwendete Geschwindigkeit.

Die zu verwendenden Optionen werden wir - statt sie auf der Kommandozeile einzugeben - in einer anderen Datei (`/etc/ppp/options`) angeben, dazu genaueres später.

Wenn der PPP-Daemon erfolgreich einen Verbindungsaufbau erledigt hat, so ruft er ein Shellscript mit Namen `/etc/ppp/ip-up` auf. Diesem Shellscript gibt er die folgenden Parameter mit:

```
ip-up Interfacename Gerätedate Geschwindigkeit Locale_IP Remote_IP
```

Dieses Script kann jetzt dazu verwendet werden, weitere Befehle auszuführen, die nach dem Verbindungsaufbau gewünscht werden, etwa Firewallregeln oder andere Kommandos. Beim Abbau einer Verbindung wird entsprechend das Script `/etc/ppp/ip-down` abgearbeitet, das wieder entsprechende Befehle beinhalten kann.

Automatischer Aufbau einer PPP-Verbindung mit wvdial

Die modernere Methode, mit der heute PPP-Verbindungen aufgebaut werden läuft über das Programm **wvdial**. Dieses Programm erledigt alle Aufgaben, von der Anwahl über das Modem, bis hin zum Start des PPP-Daemons. Es ersetzt das Chat-Programm und ermöglicht einen Login, ohne ein Script zu schreiben.

Wen **wvdial** startet, läd es zunächst seine Konfiguration aus der Datei `/etc/wvdial.conf`. Diese Datei enthält sowohl die Grundeinstellungen für die Schnittstelle, das Modem und die Geschwindigkeit, als auch Informationen über die anzuwählenden Provider (Telefonnummer, Username, Passwort).

Nachdem diese Datei gelesen wurde, initialisiert **wvdial** das Modem, wählt die entsprechende Nummer, authentifiziert den Login und startet den PPP-Daemon.

Die Konfigurationsdatei kann für verschiedene Provider Einträge besitzen. Sie ist aufgebaut wie eine Windows-INI Datei, also mit Sektionen, die durch Überschriften in eckigen Klammern begrenzt werden. Neben der Sektion `[Dialer Defaults]`, die die Grundeinstellungen und einen voreingestellten Providereintrag enthält, kann es beliebig viele andere Einträge in der Form `[Dialer` *`Providername`*`]`.

Wird **wvdial** ohne Parameter aufgerufen, so wählt es die Verbindung aus der Sektion `[Dialer Defaults]`, wird es aber mit einem Parameter aufgerufen, der einen Provider beschreibt, so wird der entsprechende Einträge aus der Datei `/etc/wvdial.conf` benutzt. Der Aufruf

```
wvdial foo
```

benutzt also den Eintrag `[Dialer foo]` aus der Konfigurationsdatei.

Ein Beispiel für eine solche Datei könnte folgendermaßen aussehen:

```
[Dialer Defaults]
Modem = /dev/modem
Baud = 57600
Init1 = ATZ
Dial Command = ATDT
Idle Seconds = 180
Phone = 012345678
Username = foo
Password = bar

[Dialer provider1]
Phone = 0987654321
Username = hans
Password = asdf1234
```

```
[Dialer provider2]
Phone = 0918273645
Login Prompt = Weristda:
Username = hmueller
Password = 1082.oir
```

Im Abschnitt `[Dialer Defaults]` werden die entsprechenden Grundeinstellungen gemacht, die für alle anderen Abschnitte auch benutzt werden, wenn dort nichts anderes vereinbart ist. Jeder andere Abschnitt enthält dann noch die notwendigen Informationen über den anzurufenden Provider.

Wie beim Aufbau der Verbindung mit Chat, so wird auch hier beim Aufbau der PPP-Verbindung das Script `/etc/ppp/ip-up` abgearbeitet und beim Verbindungsabbau entsprechend `/etc/ppp/ip-down`.

Im Verzeichnis `/etc/ppp/peers` muß - bei Verwendung modernerer PPP-Daemonen - die Datei `wvdial` liegen, die ein paar Optionen für den PPP-Daemon enthält, die er benötigt, wenn er über **wvdial** gestartet wurde. In der Regel enthält diese Datei nur die Zeilen

```
noauth
name wvdial
replacedefaultroute
```

Falls andere Startprogramme verwendet werden, können sie auch in diesem Verzeichnis jeweils eine Datei besitzen, die die nötigen Optionen für sie setzen.

Die Datei /etc/ppp/options

Damit der PPP-Daemon nicht jedesmal mit viele Kommandozeilenoptionen aufgerufen werden muß, können die gewünschten Optionen in die Datei `/etc/ppp/options` eingetragen werden. Es ist sogar möglich, für jede Schnittstelle (`/dev/ttyS1, /dev/modem,...`) eine eigene solche Datei anzulegen, die dann entsprechend `/etc/ppp/options.ttyS1` usw. heißen muß.

Diese Datei kennt viele verschiedenen Einträge, die wichtigsten sind hier kurz erklärt:

noipdefault

Der ppp-Client übernimmt nicht seine IP-Adresse aus `/etc/hosts` sondern bezieht seine Adresse vom Server. Diese Option sollte für normale Provider immer angegeben sein.

noauth

Keine automatische Authentifizierung. Das bedeutet, daß die normale Authentifizierung über das Chat-Script benutzt wird.

crtscts

Hardware-Flußkontrolle des Modems wird aktiviert.

lock

Lockdateien werden angelegt, um zu verhindern, daß ein anderer Prozeß das Modemgerät benützt, während wir online sind.

modem

Die Verbindung läuft über ein Modem. Das Gegenteil wäre **local** und wird nur angewandt, wenn zwei Rechner über ein Nullmodemkabel miteinander verbunden sind. Für eine Modemverbindung sollte immer **modem** gesetzt sein.

defaultroute

Die aufgebaute PPP-Verbindung wird als Default Route gesetzt.

persist

Nach einem Verbindungsabbruch wird nicht aufgelegt, sondern versucht, die Verbindung erneut aufzubauen.

maxfail *N*

Die Verbindung wird abgebrochen, wenn *N* mal die Verbindung durch einen Fehler abgebrochen wurde. Ein Wert von 0 schaltet dieses Feature ab.

idle *N*

Die Verbindung wird automatisch beendet, wenn mehr als *N* Sekunden keine Daten über sie gelaufen sind.

Zwingender Abbau einer PPP-Verbindung

Um eine PPP-Verbindung abzubrechen, muß einfach der PPP-Daemon per Signal INT gekillt werden. Da der Daemon seine ProzeßID immer in einer bestimmten Datei ablegt, kann diese Aufgabe automatisch durch ein Script erledigt werden. Die Datei, in die der Daemon seine PID ablegt ist normalerweise `/var/run/`*`interface`*`.pid`, wobei *interface* für die symbolische Schnittstelle (*ppp0, ppp1,* ...) der Verbindung steht. Um also die Verbindung von *ppp0* abzubrechen, kann der Befehl

```
kill -INT `cat /var/run/ppp0.pid`
```

ausgeführt werden. Natürlich kann dieser Befehl auch in einem Script verwendet werden.

1.113.1

Konfiguration und Verwaltung von inetd, xinetd und verwandten Diensten

---

Beschreibung: Prüfungskandidaten sollten in der Lage sein, die über inetd verfügbaren Dienste zu konfigurieren, tcpwrappers für das Erlauben oder Verweigern von Diensten auf Host-Ebene zu verwenden, Internetdienste manuell zu starten, stoppen und neu zu starten und grundlegende Netzwerkdienste einschließlich **Telnet** und **FTP** zu konfigurieren. Ebenfalls enthalten ist das Ausführen von Diensten unter einer anderen als der voreingestellten Benutzerkennung in `inetd.conf`.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/inetd.conf`
- `/etc/hosts.allow`
- `/etc/hosts.deny`
- `/etc/services`
- `/etc/xinetd.conf`
- `/etc/xinetd.log`

Der inetd-Superserver

Bei der großen Anzahl von möglichen Servern, die auf einem Rechner laufen müssten, damit alle Internetdienste immer bereit sind, wäre es schnell soweit, daß der Speicher mit untätigen Servern überfüllt wäre. Damit das vermieden wird, gibt es einen "Superserver", der alle möglichen Portnummern abhört und bei Anfrage einen entsprechenden Server startet. Damit kann es ermöglicht werden, daß nur die Server laufen, die gerade arbeiten, obwohl alle Dienste verfügbar sind.

Der zuständige Daemon heißt **inetd** und wird in der Datei `/etc/inetd.conf` konfiguriert. Der Aufbau dieser Datei ist verhältnismäßig einfach, meist sind alle notwendigen Einträge schon vorhanden. Sie müssen nur jeweils auskommentiert werden, wenn sie nicht erwünscht

sind. Das Format der Datei ist folgendermaßen organisiert:

```
Dienstname Socket-Typ Transportprotokoll Flags User Programm Argumente
```

Die Felder haben folgende Bedeutung:

Dienstname

Hier steht der Name eines Dienstes, so, wie er in der Datei `/etc/services` eingetragen ist. Die ganze folgende Zeile bezieht sich darauf, daß ein Client versucht, auf dem Port zuzugreifen, der in `/etc/services` unter diesem Namen genannt ist.

Socket-Typ

Gültige Werte für den Socket-Typ sind **stream, dgram, raw, rdm** und **seqpacket**, wobei in der Praxis meist nur die ersten beiden eine Rolle spielen. Über den Daumen gepeilt kann behauptet werden, daß jedes Programm, daß als Transportprotokoll TCP benutzt, hier ein **stream** eingetragen hat, jedes, das mit UDP arbeitet hat hier ein **dgram** benutzt.

Transportprotokoll

Das hier eingetragene Transportprotokoll muß eins der in `/etc/protocols` angegebenen Protokolle sein. In der Regel spielen hier nur **tcp** und **udp** eine Rolle.

Flags

An dieser Stelle gibt es nur zwei gültige Einträge, **wait** und **nowait**. Die Unterscheidung spielt nur für UDP-basierte Anwendungen eine Rolle. Alle anderen sollten hier ein **nowait** eingetragen haben. Wenn ein Datagram-Server (ein Server, der mit UDP arbeitet) nach der Versendung eines Datagrams den benutzten Socket löscht, so daß **inetd** weitere Anfragen auf diesem Socket empfangen kann, dann spricht man von einem *multi-threaded* Server und sollte hier **nowait** benutzen. Datagram-Server, die alle eingehenden Datagramme über einen Socket empfangen und letztendlich mit einem TimeOut reagieren, werden *single-threaded* genannt und benötigen hier den Wert **wait**. Typische Vertreter dieses Typs sind die **biff** und **talkd** Daemonen.

Optional kann diesem Eintrag noch ein Maximalwert mitgegeben werden, der angibt, wieviele dieser Server maximal innerhalb von 60 Sekunden gestartet werden dürfen. Dieser Wert wird durch einen Punkt getrennt an **wait** oder **nowait** angehängt. Seine Voreinstellung ist 40.

User

Der User-Eintrag legt fest, unter welcher UserID das Serverprogramm laufen soll. Nachdem **inetd** selbst unter der UserID 0 läuft, kann hier verhindert werden, daß bestimmte Dienste auch unter dieser gefährlichen UID laufen. Optional kann diesem Eintrag ein durch Punkt vom Usernamen getrennter Gruppennamen folgen, der entsprechend die GID setzt.

Programm

Hier wird das Programm angegeben, das gestartet werden soll, wenn eine auf diesen Eintrag passende Anfrage eingeht. Es wird der komplette Pfad und Programmname angegeben. Wenn **inetd** diesen Dienst selbstständig anbietet, so steht hier `internal`.

Argumente

Die Argumente (Kommandozeilenoptionen) für das zu startende Programm. Normalerweise wird der Programmname selbst (Argument 0) mit angegeben. Auch hier wird das Wort `internal` angegeben oder der Eintrag wird leer gelassen, wenn **inetd** den Dienst intern zur Verfügung stellt.

So bedeutet die folgende Zeile also:

```
ftp     stream  tcp     nowait  root    /usr/sbin/in.ftpd       in.ftpd
```

Wenn eine Nachfrage nach dem Service *ftp* ankommt, (über den Sockettyp stream, mit dem Protokoll TCP) wird unter der UserID root das Programm `/usr/sbin/in.ftpd` ohne weitere Parameter gestartet und mit der Anfrage verbunden.

Die nötigen Portnummern bezieht *inetd* aus der Datei `/etc/services` .

Der **inetd** ließt diese Konfigurationsdatei nur beim Start. Wenn also Veränderungen daran vorgenommen wurden, so muß entweder der **inetd** neu gestartet werden, oder ihm muß ein HUP-Signal geschickt werden. Dieses Signal zwingt ihn, seine Konfigurationsdatei neu einzulesen.

Der TCP-Daemon (tcpwrapper)

Neben dem oben gezeigten Beispiel gibt es noch eine spezielle Form des Aufrufs über den inetd. In der Datei inetd.conf finden sich oft auch Zeilen wie die folgenden:

```
klogin    stream  tcp   nowait  root    /usr/sbin/tcpd  rlogind -k
eklogin   stream  tcp   nowait  root    /usr/sbin/tcpd  rlogind -k -x
kshell    stream  tcp   nowait  root    /usr/sbin/tcpd  rshd -k
```

Hier scheint auf den ersten Blick immer das gleiche Programm aufgerufen zu werden, nämlich `/usr/sbin/tcpd`. Erst als dessen Parameter kommen die eigentlichen Server an die Reihe. In der Tat ist es so, daß hier der TCP-Daemon (**tcpd**) aufgerufen wird, der dann erst die Server aufruft. Das klingt auf den ersten Blick unsinnig, aber es ermöglicht eine Einstellung, wer diesen Service benutzen darf.

Der **tcpd** überprüft anhand der Dateien `/etc/hosts.allow` und `/etc/hosts.deny` ob der jeweilige Host überhaupt berechtigt ist, diesen Dienst in Anspruch zu nehmen. Eine genaue Darstellung dieser Technik habe ich auf eine eigene Seite ausgelagert, weil das Thema auch noch in anderen Kapiteln eine Rolle spielt.

Der xinetd-Server

Der Server **xinetd** ist ein sicherer Ersatz für **inetd**. Er bietet eine eingebaute Zugriffskontrolle und eine direkte Unterstützung der Dateien `/etc/hosts.allow` und `/etc/hosts.deny` ohne den Umweg über **tcpd**. Außerdem werden noch viele weitere Verbesserungen angeboten wie etwa

- Beschränkung der Verbindungen entweder generell, oder pro Dienst oder pro Client.
- Beschränkung von Verbindungen anhand der Tageszeit.
- Dienste können an bestimmte IP-Adressen gebunden werden, so daß z.B. interne Clients andere Server-Programme nutzen als externe Clients, obwohl sie den selben Dienst aufgerufen haben.
- Schutz vor *denial of service* Angriffen.
- Ausgiebige Log-Möglichkeiten (auch unabhängig von **syslogd**).
- Umleitung von TCP-Verbindungen zu einem anderen Rechner, der selbst nicht von außen erreichbar sein muß. Damit können Server hinter einem Masquerading-Server auch von außen benutzt werden.
- Volle Unterstützung von IPv6.
- Interaktion mit dem User des Clients (Meldungen bei erfolgtem oder gescheitertem Zugriff).

Die Konfiguration des **xinetd** erfolgt analog zu seinem Vorgänger in der Datei `/etc/xinetd.conf`. Allerdings hat diese Datei einen vollkommen anderen Aufbau als `/etc/inetd.conf`. Für UmsteigerInnen gibt es ein spezielles Shellscript, das eine bestehende `inetd.conf` Datei in das neue Format konvertiert. Dieses Programm ist ein Perl-Script (`xconv.pl`) und wird mit **xinetd** mitgeliefert.

Die Konfigurationsdatei von **xinetd** beginnt mit einem Abschnitt für die Voreinstellungen, der *default section*. Die hier angegebenen Attribute werden von jedem Dienst genutzt, den **xinetd** verwaltet. Nach diesem Abschnitt folgen für jeden einzelnen zu startenden Dienst ein eigener Abschnitt. Wenn in einem solchen dienstspezifischen Abschnitt Angaben gemacht werden, die denen der *default section* widersprechen, so gelten die des dienstspezifischen Abschnitts.

Die typische Form einer *default section* ist folgendermaßen aufgebaut:

```
defaults
{
    Attribut Operator Wert(e)
    ...
}
```

Alle weiteren Abschnitte beginnen mit dem Schlüsselwort `service`, dem der Name des Dienstes (wie er in `/etc/services` eingetragen ist) folgt.

```
service Dienstname
{
    Attribut Operator Wert(e)
    ...
}
```

Es gibt drei unterschiedliche Operatoren, mit denen Attribute mit Werten versehen werden:

`=`
: Setzt einen Wert für ein Attribut.

`+=`
: Fügt einem Attribut einen Wert zu den bereits bestehenden Werten hinzu.

`-=`
: Entfernt einem Attribut den angegebenen Wert.

Es existieren sehr viele möglichen Attribute, deren Beschreibung den Rahmen hier sprengen würde. Ein Beispiel für eine solche Datei sagt hier mehr als eine endlose Beschreibung:

```
defaults
{
  instances       = 15
  log_type        = FILE /var/log/xinetd.log
  log_on_success  = HOST PID USERID DURATION EXIT
  log_on_failure  = HOST USERID RECORD
  only_from       = 192.0.0.0/8

  disabled = shell login exec comsat
  disabled += telnet ftp
  disabled += name uucp tftp
  disabled += finger systat netstat
}

service ftp
{
  socket_type  = stream
```

```
    wait         = no
    user         = root
    server       = /usr/sbin/in.ftpd
    server_args  = -l
    instances    = 4
    access_times = 7:00-12:30 13:30-21:00
    nice         = 10
    only_from    = 192.168.1.0/24
  }
```

In der `defaults`-Section definieren wir hier das Attribut `instances` mit dem Wert 15. Das bedeutet, daß grundsätzlich nicht mehr als 15 Dienste gleichzeitig gestartet werden können. Später in der Definition des FTP-Dienstes setzen wir den Wert 4 für dieses Attribut. In diesem Fall bezieht es sich auf die Menge der gleichzeitigen FTP-Verbindungen.

Die Angabe `log_type = FILE /var/log/xinetd.log` bestimmen wir, daß nicht der **syslogd** benutzt werden soll, sondern **xinetd** direkt in die angegebene Datei schreibt. Anstatt diesem Eintrag wäre auch folgender möglich gewesen:

```
  log_type = SYSLOG daemon info
```

Dann wären alle Meldungen über den Syslog-Daemon geschrieben worden und zwar mit der Herkunft `daemon` und der Priorität `info`.

Die beiden Angaben `log_on_success` und `log_on_failure` legen fest, was alles in das Logbuch aufgenommen werden soll, wenn ein Zugriff erfolgreich war oder abgelehnt wurde.

Die Angabe `only_from = 192.0.0.0/8` bestimmt, daß grundsätzlich (wenn beim entsprechenden Dienst nichts anderes angegeben wurde) nur die Rechner Zugriff haben, die auf die angegebene Adresse passen. Das `/8` bestimmt, daß nur die ersten 8 Bit der Adresse gewertet werden. In unserem Beispiel werden also alle Rechner Zugriff bekommen, deren erstes Adressenbyte 192 ist.

Die folgenden Zeilen der *default section* schalten die angegebenen Dienste mit dem Attribut **disabled** komplett ab.

Im Abschnitt FTP werden jetzt die einzelnen Attribute für den Dienst gesetzt. Die ersten vier Einträge entsprechen dem, was auch in der alten **inetd** Konfiguration eingetragen war. Interessant sind hier noch die Angaben über die erlaubten Tageszeiten, in denen der Dienst zur Verfügung steht und der gesetzte NICE-Wert, unter dem der Daemon laufen soll. Zugreifen dürfen nur Rechner, deren Adresse mit 192.168.1 beginnt.

Eine Liste aller Attribute und ihrer Bedeutungen sind auf der Handbuchseite von xinetd.conf(5) zu finden. Eine genaue Beschreibung der Formate der Logdateien sind in der Handbuchseite xinetd.log(5) nachlesbar.

1.113.2

Verwendung und allgemeine Konfiguration von Sendmail

Beschreibung: Prüfungskandidaten sollten in der Lage sein, einfache Einstellungen in den Sendmail Konfigurationsdateien vorzunehmen (einschließlich des "Smart Host" Parameters, wenn notwendig), Mail-Aliases anzulegen, die Mail-Queue zu verwalten, Sendmail zu starten und zu stoppen, Mail-Weiterleitung zu konfigurieren und allgemeine Sendmail-Probleme zu lösen.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/sendmail.cf`
- `/etc/aliases` oder `/etc/mail/aliases`
- `/etc/mail/*`
- `~/.forward`

Die Einrichtung eines Mailservers mit sendmail gilt als eines der kompliziertesten Kapitel der Unix-Serverinstallation. Es existiert ein tausendseitiges Buch alleine über die Sendmail-Konfiguration, es gibt auch die Weisheit:

> *Ein richtiger Unix-Systemadministrator ist man erst, wenn man einmal eine Sendmail Konfigurationsdatei geschrieben hat.*
>
> *Ein richtiger Unix-Systemadministrator wird sich aber hüten, es ein zweites Mal zu tun.*

Aber so schlimm ist es heute nicht mehr, keine Angst. Die modernen Sendmail Versionen bringen heute ausgesprochen leicht zu handhabende Konfigurationsdateien mit, die die Arbeit mit Sendmail erleichtern. Wenn also kein zu komplizierter Mailserver aufgesetzt werden soll, dann können wir uns das Leben sehr einfach machen.

Die LPI102 Prüfung verlangt kein tiefergehendes Wissen über die Geheimnisse der **sendmail** Konfiguration, es genügen die notwendigen Informationen, um einen einfachen Mailserver einzurichten.

Die Architektur von E-Mail

Das SMTP Prinzip verteilt seine Aufgaben auf verschiedene Programme. Die zwei grundlegenden Programmtypen sind dabei

Der Mail User Agent (MUA)

Das Programm, mit dem der User seine Mails liesst und erstellt. Es existieren hunderte verschiedener MUAs, für jedes Betriebssystem. Beispiele hierfür sind Eudora, OutlookExpress, Outlook auf der Windows-Seite und elm, pine, xfmail, kmail, mutt auf der Linux-Seite.

Der Mail Transfer Agent (MTA)

Das Programm, das die abgehende Mail von einem beliebigen MUA erhält und sich darum kümmert, daß diese Mail weitergeleitet wird. Andererseits das Programm, das - bei permanenter Internet-Anbindung - die ankommenden Mails von anderen MTAs entgegennimmt und lokal abspeichert, damit die MUAs darauf zugreifen können. Der verbreitetste und mächtigste MTA ist das Programm sendmail.

Hier geht es prinzipiell um die Konfiguration von MTAs - konkret um die Konfiguration von sendmail. Die genannte Aufteilung der Aufgaben in Userprogramm und Versendeprogramm machen eine Unterscheidung hinsichtlich der Netzanbindung nötig. Es ist ein erheblicher Unterschied, ob unser Mailserver eine permanente Netzanbindung hat, oder nicht.

In einem lokalen Netz kann auf jeder Unix-Workstation ein sendmail-Daemon laufen, jedoch hat nur einer der Rechner die tatsächliche Anbindung ans Internet. Das bewirkt, daß die anderen Rechner ihre Mails an den einen mit Netzanbindung schicken, erst der versendet sie dann weiter ins Internet. Diese Architektur hat in vielen Fällen Vorteile, zum Beispiel

- Bei temporärer Netzanbindung bauen nicht alle User dauernd Verbindungen ins Internet auf, sondern schicken ihre Mails an einen Rechner, der die Mails regelmäßig weiterleitet.
- Bei einer Firewalleinrichtung kann eingestellt werden, daß nur SMTP-Pakete von und zu diesem einen Rechner durchgelassen werden.

Im Regelfall läuft sendmail als stand-alone Dienst. Das heißt, auch bei temporärer Anbindung ist der Daemon ständig im Speicher. Das ist notwendig, damit ein Rechner Mails von anderen MTAs empfangen kann. Ob sendmail die ausgehenden Mails gleich losschickt, oder wohin es sie weiterleitet, das muss konfiguriert werden.

Die zentrale sendmail-Konfigurationsdatei ist /etc/sendmail.cf. Diese Datei selbst zu bearbeiten oder gar zu erstellen ist ausgesprochen kompliziert - das Anfangs angeführte Zitat bezieht sich eben darauf. Wir werden hier nicht die komplette Konfiguration von **sendmail** behandeln, sondern ausschließlich kleine Veränderungen an dieser Datei vornehmen.

Grundlagen der Mailverzeichnisse im Linux-System

Jedes Unix/Linux-System hat zwei verschiedene Mail-Verzeichnisse. Zum einen existiert eine Warteschlange für ausgehende Mails (meist `/var/spool/mqueue` oder `/var/mqueue`) und zum anderen ein Verzeichnis `/var/spool/mail`, in dem eine Datei für jeden User existiert, die seine angekommene Mail bereithält. Die MUAs und MTAs arbeiten beide mit diesen Verzeichnissen.

Ein MUA, mit dessen Hilfe eine Mail versendet wird, legt diese abgehende Mail mitsamt aller dazu nötigen Headerinformation (von wem kommt die Mail, an wen geht die Mail usw.) in das Spoolverzeichnis für die ausgehende Mail. Der MTA arbeitet diese Warteschlange in regelmäßigen Abständen ab und verschickt die Mails ins Internet - entweder direkt an die empangenden Rechner oder an einen Mailserver des Providers, der dann die Weiterversendung übernimmt.

Umgekehrt bekommt der MTA eingehende Mails aus dem Netz und sortiert sie in die Eingangspostfächer (z.B. `/var/spool/mail/hans`). Beim nächsten Aufruf eines MUA findet dieser im jeweiligen Verzeichnis dann die neue Mail vor und kann sie darstellen.

Die Programme, die unter Unix ständig dafür sorgen, daß einem User mitgeteilt wird, daß er neue Mail hat (biff/xbiff), nützen ebenso diese Verzeichnisse. Der Vorteil an diesem Prinzip liegt in der Tatsache, daß das Mailsystem so integraler Bestandteil des Systems selbst ist, und nicht ein aufgepfropftes Anwenderprogramm. Nur deshalb können z.B. Programme wie **cron** oder **at** ihre Ausgaben als Mail an den User schicken, der den Auftrag gegeben hatte.

Um den Inhalt der Warteschlange anzusehen, existiert der Befehl **mailq**, der in Wahrheit nur ein symbolischer Link auf **sendmail** ist.

Starten von sendmail

Die zentrale Aufgabe von **sendmail** ist es, neben dem Empfang von Mails über SMTP, die Warteschlange für die ausgehende Mail (mqueue) zu bearbeiten. Dazu existieren zwei völlig unterschiedliche Architekturen, entweder ist der Rechner, auf dem **sendmail** läuft tatsächlich mit einer festen IP-Adresse ans Internet angebunden und von überall her unter einem bestimmten Namen zu erreichen, oder er ist nur temporär ans Internet angebunden und somit nur für das Versenden von Mails gedacht.

Starten des Mailservers mit permanenter Netzanbindung

Auf einem Rechner mit permanenter Internet-Anbindung hat **sendmail** in der Regel zwei Aufgaben:

- Empfang von Mails aus dem Internet über SMTP
- Versand von ausgehenden Mails über SMTP

Zumindestens die erste Aufgabe bedingt es, daß **sendmail** permanent als Daemon im Speicher liegt und arbeitet. Nur so ist es gewährleistet, daß eingehende Mails tatsächlich jederzeit angenommen werden können. Der Start von **sendmail** benötigt hier zwei

wesentliche Parameter, der erste gibt an, daß das Programm als Daemon gestartet werden soll und der zweite stellt die Frequenz ein, in der **sendmail** die Warteschlange bearbeiten soll (in diesem Beispiel 30 Minuten):

```
sendmail -bd -q30m
```

Auf diese Weise wird **sendmail** alle 30 Minuten die Mailqueue lesen und anstehende Aufträge (ausgehende Mails) versenden. Die eingehenden Mails werden jederzeit empfangen.

Starten des Mailservers mit temporärer Netzanbindung

In diesem Fall wird **sendmail** nur für die Versendung von Mail herangezogen. Das Empfangen wird wahrscheinlich über POP3 oder IMAP geregelt. In diesem Fall muß **sendmail** nur regelmäßig aufgerufen werden, um die ausgehenden Mails zu versenden. Dazu wird der Parameter `-bd` weggelassen, um zu verhindern, daß das Programm als Daemon gestartet wird. Der Parameter `-q` wird ohne Angabe eines Zeitintervalls angegeben. Das führt dazu, daß **sendmail** die Mailqueue abarbeitet und anschließend beendet wird.

```
sendmail -q
```

Dieser Befehl kann jetzt z.B. in regelmäßigen Abständen von cron aufgerufen werden.

Um die interne Weitergabe von Mails zu realisieren, wird meist jedoch auch auf Rechnern ohne permanente Anbindung ein Sendmail-Daemon gestartet, der allerdings nicht die Warteschlange abarbeitet. Die Versendung geschieht mit dem oben genannten Befehl über cron gesteuert, der Aufruf von **sendmail** heißt dann

```
sendmail -bd
```

So ist die interne Funktionalität weiter gewährleistet, ohne daß in regelmäßigen Abständen die Queue abgearbeitet wird.

Mailaliase

sendmail hat die Möglichkeit, Alias-Adressen zu bilden. Das sind E-Mail Adressen, die in Wahrheit keinem normalen User zugeordnet sind, sondern auf andere User verweisen. Häufig benutzte solche Adressen sind z.B.

- webmaster@...
- abuse@...
- ftpadmin@...

In den seltensten Fällen sind das wirkliche User des Systems. Normalerweise werden einfach Aliase angelegt, die diese Adressen dem entsprechenden User zuweisen, der den Job übernommen hat. Dazu kennt **sendmail** die Datei `/etc/aliases` oder manchmal auch

`/etc/mail/aliases`.

Das Format dieser Datei ist sehr einfach, pro Zeile wird ein Alias definiert:

Aliasname: Username

Um die drei obigen Beispiele bestimmten Usern zuzuordnen könnten in einer solchen Datei also folgende Einträge stehen:

```
webmaster: root
abuse:     root
ftpadmin:  hans
```

Alle Mails an `webmaster@`*`mydomain.de`* und `abuse@`*`mydomain.de`* würden also an den User `root@`*`mydomain.de`* weitergeleitet, während Mails an `ftpadmin@`*`mydomain.de`* an `hans@`*`mydomain.de`* weitergegeben würden. *mydomain.de* würde natürlich durch die entsprechende lokale Domain ersetzt.

Es ist auch möglich, bestimmte Adressen nicht an andere Adressen zu versenden, sondern an bestimmte Programme weiterzuleiten. In diesem Fall werden Mails, die an diese Adresse gingen an ein angegebenes Programm weitergepiped. So können z.B. Mailinglists organisiert werden. Statt einem Usernamen wird dann ein Programmname angegeben, dem ein Pipe-Symbol vorangestellt wurde. Programmname und Pipesymbol müssen in doppelte Anführungszeichen gesetzt werden:

```
Aliasname: "|Pfad/zu/Programm Optionen"
```

Nach jeder Veränderung der Alias-Datei muß das Programm **newaliases** aufgerufen werden, um diese Aliase für **sendmail** bekannt zu machen. Wie **mailq** ist auch **newaliases** nur ein symbolischer Link auf **sendmail**.

Mails an andere Adressen weiterleiten

Einen ähnlichen Mechanismus wie das Setzen von Aliasen bietet die Datei `~/.forward` im Homeverzeichnis eines Users. Diese Datei enthält entweder E-Mail-Adressen, Dateinamen oder Programmnamen, an die eingehende Mails des Users, in dessen Homeverzeichnis die Datei liegt, weitergeleitet werden.

Die Syntax der drei möglichen Einträge ist einfach:

E-Mail-Adressen

Enthält eine `.forward` Datei eine Zeile mit einer einfachen E-Mail-Adresse, so wird jede eingehende Mail des Users an diese Mailadresse weitergeleitet. Die E-Mail-Adresse wird ohne Anführungszeichen angegeben. Enthält sie keine Domain-Angabe, so wird eine lokale Adresse angenommen. Wenn z.B. auf einem Rechner der Useraccount `hans` der Normallogin des Systemverwalters ist, so kann der Systemverwalter in sein Homeverzeichnis eine `.forward` Datei anlegen, mit dem Eintrag

```
hans
```

Jede Mail an `root` wird jetzt an Hans weitergeleitet. So empfängt der Systemverwalter seine Mails auch, wenn er sich als `hans` anmeldet.

Dateinamen

Wenn eingehende Mails in eine bestimmte Datei kopiert werden sollen, so wird der Name der Datei in Anführungszeichen in `.forward` angegeben. Die angegebene Datei entspricht vom Aufbau her einer normalen Mailbox-Datei, wie sie normalerweise in `/var/spool/mail` liegt. Alle Mails werden an diese Datei angehängt.

```
"/var/spool/mail2/hans"
"./Mail/meinemail"
```

Die Angaben einer Datei werden immer entweder als kompletter Pfad zu der Datei oder als Pfad mit führendem Punkt angegeben. Im zweiten Fall bezieht sich der Pfad auf das Homeverzeichnis des jeweiligen Users.

Bei der Angabe von normalen Pfaden muß darauf geachtet werden, daß der User Schreibrecht auf die entsprechende Datei besitzt.

Programmnamen

Soll eine Mail an ein bestimmtes Programm weitergegeben werden, so wird der Programmname in die Datei `.forward` in Anführungszeichen mit führendem Pipe-Symbol angegeben.

```
"| /Pfad/zu/Programm"
```

Die Mail wird dann an das angegebene Programm gepiped, das heißt, das Programm muß Eingaben von der Standard-Eingabe verarbeiten können. Die Angabe eines Programms erfolgt mit vollem Pfad.

Mit Hilfe dieses Mechanismuses ist es möglich, daß auch Normaluser ihre Mails weiterleiten oder Mailinglisten aufbauen, auch wenn sie keinen Zugriff auf `/etc/aliases` besitzen.

Ausgehende Mailserver

Wenn Mails von **sendmail** versendet werden sollen, dann gibt es zwei Möglichkeiten, wie das organisiert werden kann. Entweder verschickt **sendmail** die Mails direkt an die angegebenen Adressen, oder alle Mail wird an einen sogenannten *Smarthost* weitergeleitet, der sich dann um das weitere Versenden kümmert. Die meisten Internet-Provider stellen einen solchen SMTP-Server für ihre Kunden zur Verfügung.

Die Versendung ohne Smarthost bedingt eine funktionierende DNS-Installation, da **sendmail** sowohl die Adressen der ausgehenden Mails

auflösen muß, als auch von den Empfänger-Rechnern wieder angesprochen werden muß.

Wird stattdessen aber ein Smarthost angegeben, so sendet **sendmail** alle ausgehende Mail an diesen Rechner, der dann natürlich die selben Ansprüche erfüllen muß, wie ein lokaler Rechner, der keinen Smarthost verwendet, er muß eine funktionierende DNS-Anbindung haben.

Der Eintrag für den Smarthost findet in der zentralen Konfigurationsdatei von **sendmail** statt, die Datei `/etc/sendmail.cf`. Hier muß die Einstellung

```
DSRechnername
```

eingetragen sein. Soll stattdessen kein solcher Rechner verwendet werden, so wird der Parameter *Rechnername* einfach weggelassen.

```
DS
```

Die Dateien in /etc/mail

Das Verzeichnis `/etc/mail` enthält verschiedene Dateien, die Einstellungen für **sendmail** ermöglichen. Die meisten davon existieren in zwei Formaten. Zum einen existiert eine einfache Textdatei, in der wir die Einstellungen vornehmen können, zum anderen eine Datei, die den selben Namen, aber die Endung `.db` trägt. Sie enthält die Informationen, die **sendmail** selbst verarbeitet. Nach jeder Änderung einer der Textdateien ist es notwendig, diese Datei in das Datenbankformat (`.db`) zu konvertieren. Dazu wird folgender Befehl eingegeben:

```
makemap hash -f /etc/mail/datei.db < /etc/mail/datei
```

Statt *datei* wird natürlich der Name der zu konvertierenden Datei angegeben.

/etc/mail/access

Diese Datei regelt den Zugriff auf den Mailserver. Also die Frage, welcher Rechner darf ausgehende Mail an diesen Server schicken, damit er sie weiterleitet. Ein paar Beispiele:

```
127                     RELAY
192.168.100.123         OK
mydomain.de             OK
```

Alle Rechner, deren Adresse mit 127 anfangen (das sind nur wir selbst) senden ihre Mail grundsätzlich (RELAY) über diesen Rechner. Der Rechner 192.168.100.123 darf jederzeit Aufträge an diesen Rechner weitergeben und alle Mitglieder der Domain mydomain.de ebenfalls. Sonst werden alle Aufträge zur Weiterleitung abgelehnt.

Wie oben erwähnt muß diese Datei nach jeder Veränderung in das Datenbankformat konvertiert werden.

/etc/mail/genericstable

Diese Datei dient der Übersetzung ausgehender E-Mail Adressen in weltweit gültige. Bin ich z.B. auf dem Server der Systemverwalter, so lautet meine E-Mail Adresse intern z.B. `root@server.local`. Würde meine ausgehende Mail nun diese Adresse als Absender tragen, bekäme ich wohl nie eine Antwort. Denn mit `root@server.local` kann im Internet niemand etwas anfangen. Dort gilt meine Adresse `echter.name@echter.provider.org`. In der Datei genericstable stehen jetzt Umrechnungstabellen für solche Fälle:

```
root                    echter.name@echter.provider.org
root@server.local       echter.name@echter.provider.org
hans                    hMueller@gmx.de
hans@server.local       hMueller@gmx.de
```

Auch diese Datei muß nach einer Veränderung in das Datenbankformat gebracht werden.

/etc/mail/mailertable

Hier werden bekannte Rechner genannt, samt den Methoden, mit denen Mail an sie verschickt wird. Dabei können auch Tricks angewandt werden, wie etwa der Domainname, der an einen bestimmten Rechner weitergeleitet wird:

```
marvin.local.de       smtp:marvin.local.de
hal.local.de          smtp:hal.local.de
golem.local.de        smtp:golem.local.de
local.de              smtp:marvin.local.de
```

Alle Aufträge an bestimmte Rechner (marvin, hal und golem) gehen an die Rechner selbst via smtp. Alle Adressen, die keinen Rechner sondern nur eine Domain nennen (z.B. `hans@local.de`) werden an marvin weitergeschickt.

Auch diese Datei muß nach einer Veränderung in das Datenbankformat gebracht werden.

/etc/mail/virtusertable

Hier werden - ähnlich wie bei aliases - zwei verschiedene Adressen miteinander verknüpft. Dabei darf allerdings über Domaingrenzen gegangen werden, so daß - sollte ein Rechner mehrere Domains verwalten - hier Umleitungen zur jeweils passenden Adresse vorgenommen werden können.

```
efka@local.de root@golem.local.de
```

Die Adresse `efka@local.de` wird in Wahrheit an `root@golem.local.de` weitergegeben.

Auch diese Datei muß nach einer Veränderung in das Datenbankformat gebracht werden.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.113.3

Verwendung und allgemeine Konfiguration von Apache

Beschreibung: Prüfungskandidaten sollten in der Lage sein, einfache Einstellungen in den Apache Konfigurationsdateien vorzunehmen, **httpd** zu starten, anzuhalten und neu zu starten und das automatische starten von **httpd** bei Systemstart einzurichten. Dies beinhaltet nicht fortgeschrittene Konfiguration von Apache.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **apachectl**
- **httpd**
- `httpd.conf`

Die Konfiguration des Webservers **apache** ist ein weites Feld, das problemlos ganze Bücher füllen kann. An dieser Stelle wird aber nur ein Grundwissen über den Webserver und seine Konfiguration gefordert, das relativ schnell erlernbar ist.

Konfigurationsdateien und Verzeichnisse des Webservers

Der Webserver **apache** benutzte ursprünglich drei Konfigurationsdateien, die in der folgenden Reihenfolge abgearbeitet wurden:

httpd.conf

: Die zentrale Konfigurationsdatei des Webservers. Hier werden alle Einstellungen vorgenommen, die den Server selbst, den Hauptserver und eventuell existierende virtuelle Server angehen.

srm.conf

: In dieser Datei wurden die `ResourceConfig` Anweisungen des Webservers plaziert. Seit einiger Zeit wird diese Datei leer ausgeliefert und alle Einträge werden in `httpd.conf` vorgenommen.

access.conf

In dieser Datei wurden die Anweisungen des Webservers plaziert, die mit Zugriffsrechten zu tun hatten. Seit einiger Zeit wird auch diese Datei leer ausgeliefert und alle Einträge werden in `httpd.conf` vorgenommen.

Die beiden Dateien `srm.conf` und `access.conf` sind zwar noch vorhanden, werden jedoch nicht mehr benötigt, da alle Einstellungen heute in der Datei `httpd.conf` vorgenommen werden.

Standardmäßig erwartete der Webserver seine Konfigurationsdateien in `/usr/local/etc/httpd/conf`. Dieses Verzeichnis entspricht aber nicht dem Dateisystem-Standard von Linux. Aus diesem Grund wird heute der Webserver in der Regel über eine Kommandozeilenoption `-f` *`Dateiname`* gestartet, die ihm ein anderes Verzeichnis für die Konfigurationsdatei angibt. In der Regel ist das Verzeichnis entweder `/etc/httpd` oder `/etc/apache`. Dort liegen dann alle Konfigurationsdateien des Webservers.

Neben diesem Verzeichnis gibt es noch zwei weitere, die für den Webserver eine große Rolle spielen.

ServerRoot

Das Verzeichnis, in dem die Dateien liegen, die für den Gebrauch des Webservers wichtig sind. Das sind z.B. die Icons, die er für Dateien und Verzeichnisse benutzt oder die CGI-Programme.

DocumentRoot

Das Verzeichnis, das die HTML-Dateien enthält, die der Webserver der Allgemeinheit anbieten soll.

Die Festlegung, welche Verzeichnisse für diese beiden Aufgaben benutzt werden sollen, wird in der Konfigurationsdatei `httpd.conf` gemacht.

Die zentrale Konfigurationsdatei des Webservers

Auch heute noch versucht der Server alle drei oben genannten Dateien abzuarbeiten, aber die beiden Dateien `access.conf` und `srm.conf` sind in der Regel einfach leer. Alles, was früher dort untergebracht wurde, finden wir heute in der zentralen Konfigurationsdatei `httpd.conf`.

Diese Datei ermöglicht es, den Webserver komplett zu konfigurieren, von der Einstellung, auf welchen Port er hören soll, bis hin zur Angabe welcher User welche Verzeichnisse sehen darf. Die standardmäßige Konfigurationsdatei des Webservers enthält bereits eine wohldurchdachte und sehr gut kommentierte Konfigurationsdatei, die entsprechend angepasst werden kann.

Grunsätzlich ließt der Webserver diese Datei beim Start. Das heißt, wenn in dieser Datei Veränderungen vorgenommen werden sollen, dann muß der Server danach neu gestartet werden. **Bitte nochmal zur Definition: Server meint hier das Programm und nicht den Rechner!**

Die Konfigurationsdatei des Webservers kennt hunderte verschiedener Anweisung, die hier nicht alle dargestellt werden können. Wir

werden uns hier also auf die wichtigsten Anweisungen beschränken. Die Konfigurationsdatei ist in drei Bereiche aufgeteilt: Die globalen Einstellungen, die Einstellungen des Hauptservers und die Einstellungen für die virtuellen Server.

Globale Einstellungen

Diese Einstellungen beziehen sich auf die grundsätzliche Funktionalität des Servers selbst. Folgende Einstellungen sind wichtig:

`ServerType standalone|inetd`

Diese Anweisung legt fest, ob der Webserver standalone oder durch inetd gestartet werden soll. Wie schon gesagt ist es nicht empfehlenswert, den Apache durch inetd zu starten, also solte hier der Begriff `standalone` stehen.

`ServerRoot "/usr/local/httpd"`

Die Angabe des Verzeichnisses, in dem der Apache seine Dateien vorfindet. Hier liegen in der Regel alle wichtigen Dateien und Verzeichnisse, die der Server zum Betrieb benötigt. Nicht verwechseln mit DocumentRoot.

`LockFile /var/lock/subsys/httpd/httpd.accept.lock`

Das Lock-File des Servers. Diese Datei wird vom Server beim Start angelegt und beim Herunterfahren gelöscht. Damit kann der Server selbst feststellen, ob schon ein Apache-Server im Speicher gestartet ist.

`PidFile /var/run/httpd.pid`

Auch diese Datei wird vom Server beim Start angelegt. Er schreibt seine ProzessID hier hinein, so daß ein späterer Zugriff - etwa zum Versenden eines Kill-Signals - auf den Server möglich ist, ohne erst lang mit dem ps-Kommando die PID herausfinden zu müssen.

`Timeout 300`

Die Anzahl von Sekunden, die der Server beim Senden und Empfangen von Daten wartet, bis er aufgibt und einen Timeout-Fehler ausgibt.

`KeepAlive On`

KeepAlive ermöglicht es, mehrere Transfers pro Verbindung zuzulassen (On) oder zu verbieten (Off). Damit wird verhindert, daß wegen jeder einzelnen Nachfrage erneut der komplette TCP-Handshake erfolgen muß.

`MaxKeepAliveRequests 100`

Dieser Wert bestimmt die maximale Menge der zulässigen aufeinanderfolgenden Verbindungen, ohne erneuten TCP-Handshake. Steht der Wert auf 0, so bedeutet das, daß es keine Einschränkungen gibt (unendlich).

`KeepAliveTimeout 15`

Die Anzahl von Sekunden nach der Übertragung einer Datei bis zum Beginn der nächsten Nachfrage, die ohne zusätzlichen Handshake stattfinden darf.

`StartServers 1`

Die Anzahl der Serverprozesse, die beim Start geladen werden sollen.

`MinSpareServers 1`

Die minimale Anzahl der unbenutzten Serverprozesse (Childs). Wird dieser Wert unterschritten, so werden neue Server gestartet.

`MaxSpareServers 1`

Die maximale Anzahl unbenutzter Serverprozesse (Childs). Wird dieser Wert überschritten, so werden unbenutzte Serverprozesse heruntergefahren.

`MaxClients 150`

Die maximale Anzahl gleichzeitig bedienbarer TCP-Verbindungen. Kommen gleichzeitig mehr als angegeben herein, so bekommen die übrigen eine Fehlermeldung und werden nicht mehr bedient.

`MaxRequestsPerChild 0`

Nach der angegebenen Zahl von abgearbeiteten Aufträgen wird ein Child-Prozess heruntergefahren und durch einen neuen ersetzt. Steht der Wert auf 0, so findet keine Ersetzung statt (0 = unendlich). Damit kann z.B. im Dauerbetrieb festgelegt werden, daß kein Server-Child mehr als 30 Anfragen beantwortet. So wird verhindert, daß ein eventuell hängender Prozess über Tage hinweg den Server stört.

Nach diesen Grundeinstellungen werden die Module geladen. An diesen Einstellungen ist eigentlich keinerlei Veränderung nötig, es sei denn, Sie haben eigene Module entwickelt und wollen sie einbinden.

Konfiguration des Hauptservers

Die zweite Sektion der Konfigurationsdatei bezieht sich auf die Angaben zum "Hauptserver". Das ist der eigentliche HTTP-Server, der ohne weitere Gimmicks läuft. Im Regelfall ist das der einzige Server, nur in Spezialfällen werden sogenannte virtuelle Server dazugenommen.

`Port 80`

Die Portnummer, die verwendet werden soll, wenn der Server als standalone-Server läuft. Der Standard-Port ist 80. Ports unter 1023 müssen mit root-Rechten ausgestattet sein, es handelt sich ja um die sogenannten *privilegierten Ports*. Um Sicherheitslücken zu vermeiden, werden weitere Server unter anderen UserIDs gestartet.

`User wwwrun`

Die UserID, unter der die Child-Prozesse des Servers laufen. Alle Zugriffe des Servers aufs Dateisystem werden unter dieser ID ausgeführt.

`Group nogroup`

Die Gruppenmitgliedschaft der Child-Prozesse.

`Listen 80`

Diese Anweisung entspricht der Port-Anweisung, nur sind damit auch die Angabe mehrerer Ports möglich. Eine typische Form, um z.B auch das HTTPS-Protokoll (Secure Socket Layer) zu ermöglichen, wäre:

```
<IfDefine SSL>
  Listen 80
  Listen 443
</IfDefine>
```

`ServerAdmin root@localhost`

Die E-Mail Adresse des Administrators dieses Webservers

`DocumentRoot "/usr/local/httpd/htdocs"`

Die Wurzel des Dokumentenbaums. Alle Pfade in URLs (außer CGI) beziehen sich auf die diesen Pfad.

`DirectoryIndex index.html`

Der Name (oder die Namen) der Datei, die in einem Verzeichnis aufgerufen werden soll, wenn nur der Verzeichnisname angegeben wurde. Wenn mehrere Namen angegeben werden, müssen sie durch Leerzeichen voneinander getrennt werden.

`ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi-bin/"`

Mit dieser Anweisung wird festgelegt, welches Verzeichnis CGI-Scripts enthalten darf, die vom Server ausgeführt werden sollen. Der Alias `/cgi-bin` wird also auf das Verzeichnis `/usr/local/httpd/cgi-bin/` gelegt. Damit werden URLs wie

```
http://www.myhost.mydomain.de/cgi-bin/irgendwas.pl
```

auf das genannte Verzeichnis umgeleitet. Es ist zwar zu empfehlen, daß das Script-Verzeichnis außerhalb des Dokumentenbaums liegt, jedoch nicht notwendig. Falls mehrere solcher Aliase angelegt werden sollen, müssen sie jeweils andere Aliasnamen tragen:

```
ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi-bin/"
ScriptAlias /test/cgi-bin/ "/usr/cgi-bin/"
ScriptAlias /users/hans/cgi-bin/ "/home/hans/cgi-bin/"
```

Die entsprechenden URLs für die drei verschiedenen Verzeichnisse wären jetzt:

```
http://www.myhost.mydomain.de/cgi-bin/irgendwas.pl
http://www.myhost.mydomain.de/test/cgi-bin/irgendwas.pl
http://www.myhost.mydomain.de/users/hans/cgi-bin/irgendwas.pl
```

Die meisten weiteren Einstellungen benützen jetzt ein HTML-ähnliches Format, um die Gültgkeit auf bestimmte Bereiche zu reduzieren. Dazu werden die gewünschten Bereiche (Verzeichnisse, Dateien und URLs) in spitze Klammern geschrieben und mit einem entsprechenden Abschlußtag beendet. Ein paar Beispiele:

```
<Directory "/usr/local/httpd/htdocs">
  Options FollowSymLinks
  DirectoryIndex default.htm
</Directory>
```

Diese Angabe legt fest, daß die beiden Befehle in den Klammern nur für das Verzeichnis "/usr/local/httpd/htdocs" und alle darin enthaltenen Unterverzeichnisse gelten.

Statt <Directory> kann auch <Location> stehen. Dann wird statt einer absoluten Pfadangabe eine URL angegeben. Das erspart einem Systemverwalter das Wissen über die tatsächliche Positionierung des DocumentRoot. Das muß dann selbstverständlich mit </Location> abgeschlossen werden.

Um einzelne Dateien direkt anzusprechen kann auch die Klammerung mit <Files> erfolgen. In der Klammer stehen dann Dateinamen, auf die sich die Angaben beziehen.

Konfiguration von virtuellen Servern

Die dritte Sektion der zentralen Konfigurationsdatei von Apache dient der Definition von virtuellen Servern. Virtuelle Server sind sozusagen Nebenserver, die vom selben Webserver verwaltet werden, aber eine andere Dokumentenwurzel benutzen. Dazu kommt, daß diese Server auch noch entweder über andere IP-Adressen oder andere DNS-Namen ansprechbar sind.

Die Definition eines virtuellen Servers ermöglicht es also dem Webserver zu entscheiden, an welchen DNS-Namen bzw. welche IP-Adresse die Anfrage ging. Diese Entscheidung führt dann zu unterschiedlichen DocumentRoots also werden unterschiedliche Seiten angezeigt.

Linux bietet zu diesem Zweck die Möglichkeit, daß eine einzige Ethernetkarte mehrere IP-Adressen bekommen kann. Dazu wird der Bezeichnung der Ethernetkarte (z.B. `eth0`) einfach ein Doppelpunkt und eine Nummer zugefügt (`eth0:1, eth0:2, ...`). Jeder dieser "virtuellen Netzwerkkarten" kann jetzt eine eigene IP-Adresse vergeben werden. Entweder mit dem entsprechenden Konfigurationsprogramm (yast, linuxconf,...) oder mit dem Befehl:

```
ifconfig eth0:1 Adresse netmask Maske
```

Wenn diese zweite (dritte, vierte, ...) Adresse jetzt im Nameserver einen eigenen Eintrag erhält, so kann tatsächlich der Webserver darauf reagieren. Dazu müssen in der Konfigurationsdatei ein paar zusätzliche Einträge vorhanden sein:

```
NameVirtualHost  IP-Adresse
```

Dieser Eintrag ist nur nötig, um mehrere namensbasierte Virtuelle Server aufzubauen. Die IP-Adresse ist die der "virtuellen Netzwerkkarte" wie oben beschrieben.

Jetzt können wir die einzelnen virtuellen Server definieren. Dazu werden wiederum Angaben in spitzen Klammern gemacht, entweder mit IP-Adressen (von virtuellen Karten) oder mit Hostnamen (Alias-Einträge im Nameserver, die auf die virtuelle Karte verweisen).

Jeder dieser Einträge bekommt jetzt einen eigenen DocumentRoot und kann alle bisher besprochenen Direktiven des normalen Servers enthalten. Ein Beispiel:

```
NameVirtualHost 10.230.1.101

<VirtualHost 10.230.1.101>
  ServerAdmin root@marvin.mydomain.de
  DocumentRoot /www2
  ServerName virtual1.mydomain.de
</VirtualHost>

<VirtualHost 10.230.1.101>
  ServerAdmin hans@marvin.mydomain.de
  DocumentRoot /www3
  ServerName virtual2.mydomain.de
  <Directory /www3/specialdir>
     AllowOverride All
  </Directory>
</VirtualHost>
```

Hier haben wir also zwei virtuelle Hosts definiert, beide wurden über ein und dieselbe IP-Adresse (10.230.1.101) definiert, aber beide unterscheiden sich anhand ihres Namens. Natürlich muß der zweite Name entsprechend im Nameserver vorhanden sein und als Alias auf den ersten Rechner gesetzt sein.

Die zweite Möglichkeit virtueller Hosts ist die Adressenbasierte. Hier ist die Angabe des NameVirtualHost-Befehls nicht nötig. Dafür muß jeder virtuelle Server eine eigene IP-Adresse besitzen. Mit der oben gezeigten Methode ist das bis zu einem bestimmten Punkt möglich. Ab einer gewissen Anzahl (etwa ab 4 virtuellen Hosts) bietet es sich aber an, namensbasierte Hosts zu generieren, statt mit x virtuellen Netzwerkkarten zu arbeiten...

Innerhalb der <VirtualHost> Klammerung können alle Direktiven stehen, die auch schon für die Konfiguration des Hauptservers zur Anwendung kamen. Es sind also vollständig autarke Server, denen sogar eigene CGI-Verzeichnisse gegeben werden können. Das folgende Beispiel zeigt zwei virtuelle Hosts, die Adressenbasiert aufgebaut sind und je ein eigenes cgi-bin Verzeichnis besitzen:

```
<VirtualHost 10.230.1.105>
  ServerAdmin root@marvin.mydomain.de
  DocumentRoot /www1/htdocs
  ScriptAlias /cgi-bin/ "www1/cgi-bin/"
  ServerName virtual1.mydomain.de
</VirtualHost>

<VirtualHost 10.230.1.106>
  ServerAdmin hans@marvin.mydomain.de
  DocumentRoot /www2/htdocs
  ScriptAlias /cgi-bin/ "www2/cgi-bin/"
  ServerName virtual2.mydomain.de
</VirtualHost>
```

Starten und Anhalten des Webservers

Normalerweise wird der Webserver beim Start des Systems durch ein Init-Script gestartet. Dieses Script wird beim Wechsel in den Standard-Runlevel ausgeführt und heißt meistens `/etc/init.d/apache` oder `/etc/init.d/httpd`.

In diesem Script wird - wie oben schon erwähnt - als Parameter für den Webserver die Konfigurationsdatei mit angegeben. Typische Einträge zum Starten wären also z.B.

```
/usr/bin/httpd -f /etc/httpd/httpd.conf
```

Auch das Herunterfahren des Webservers kann in der Regel mit diesem Script ausgeführt werden, indem ihm der Parameter `stop` mitgegeben wird.

Nach jeder Änderung an der Konfigurationsdatei muß der Server neu gestartet werden. Damit das nicht nur über das Init-Script erfolgen kann, gehört zum Lieferumfang von **apache** ein Serverkontrollprogramm, mit dem diese Aufgaben erledigt werden können. Dieses Programm ist ein Shellscript, das eventuell an abweichende Pfade angepasst werden muß.

Die Aufrufform des Programms ist

`apachectl` *Kommando*

Es stehen verschiedene Kommandos zur Verfügung. Die wichtigsten sind:

start

Startet den Apache-Daemon (Webserver). Gibt eine Fehlermeldung aus, wenn er bereits läuft.

stop

Hält den Daemon an.

restart

Stopt den laufenden Daemon und startet ihn neu. Zu diesem Zweck wird dem Daemon ein HUP-Signal geschickt. Wenn der Daemon noch nicht gestartet war, wird er neu gestartet.

graceful

Dieses Kommando startet den Server mit dem Signal USR1 neu. Der Unterschied zu **restart** ist, daß bestehende Verbindungen nicht unterbrochen werden.

configtest

Die Konfigurationsdatei wird einem Syntax-Test unterworfen. Rückgabe ist entweder **Syntax Ok** oder eine detailierte Information über die gefundenen Fehler in der Konfigurationsdatei.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

1.113.4

Richtiges Verwalten von NFS, smb und nmb Dämonen

Beschreibung: Prüfungskandidaten sollten wissen, wie man im Netzwerk freigegebene Dateisysteme mittels NFS einbindet, wie man NFS für den Export lokaler Dateisysteme konfiguriert und wie man den NFS-Server startet, anhält und neustartet. Ebenfalls enthalten ist die Installation und Konfiguration von Samba unter Verwendung des mitgelieferten GUI-Tools oder durch direkte Bearbeitung von `/etc/smb.conf` (Achtung: Bewußt ausgeschlossen sind fortgeschrittene Themen der NT-Domänen; einfaches Freigeben von Verzeichnissen und Druckern sowie das korrekte Einrichten von nmbd als WINS-Client sind jedoch enthalten).

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/exports`
- `/etc/fstab`
- `/etc/smb.conf`
- **mount**
- **umount**

Einhängen freigegebener Verzeichnisse

Unter Linux werden freigegebene Verzeichnisse ähnlich wie bei Windows als Netzlaufwerke betrachtet. Allerdings werden sie nicht mit einem Laufwerksbuchstaben versehen, sondern - wie alle anderen Laufwerke auch - in den Dateibaum gemountet.

Es gibt prinzipiell zwei grundlegende Techniken der Freigabe. Entweder ist ein Verzeichnis über das *Network File System* (NFS) freigegeben, oder über das *Server Message Block* Protokoll (SMB). NFS ist in der Regel von Unix/Linux Maschinen angeboten, SMB ist die Technik der Windows-Freigaben.

Einhängen von NFS-Freigaben

Eine NFS-Freigabe wird durch einen einfachen **mount** Befehl eingebunden. Dazu muß aber statt einer Gerätedatei ein NFS-Pfad eingegeben werden. Dieser Pfad hat die Form

Hostname:/absoluter/Pfad

Um beispielsweise das freigegebene Verzeichnis `/usr/public` des Rechners `hal` in unser lokales Verzeichnis `/mnt` einzuhängen, wird der Befehl

```
mount -t nfs hal:/usr/public /mnt
```

eingegeben. Normalerweise kann die Angabe des Dateisystemtyps (`-t nfs`) auch weggelassen werden, da der **mount**-Befehl aus der Struktur der Pfadangabe erkennt, daß es sich um NFS handeln muß.

Dieser Prozess kann natürlich auch automatisiert werden, damit das Verzeichnis bei jedem Start des Rechners wieder gemountet wird. Dazu muß wie bei lokalen Partitionen ein Eintrag in die Datei `/etc/fstab` gemacht werden. Unsere Beispielfreigabe würde folgenden Eintrag benötigen:

```
hal:/usr/public /mnt    nfs   defaults  0  0
```

Dabei ist allerdings darauf zu achten, daß der Fileserver (in diesem Fall `hal`) schon läuft, bevor der Computer gestartet wird, dessen `/etc/fstab` diese Zeile enthält. Sonst versucht der Rechner das Verzeichnis von `hal` zu mounten und es kann eine ganze Weile dauern, bis der Timeout dafür sorgt, daß der Bootvorgang weitergeht.

Einhängen von SMB-Freigaben

Freigaben von Windows-Rechnern (oder von Samba-Servern) werden über das SMB-Protokoll gesteuert. Diese Freigaben müssen im Gegensatz zu NFS-Freigaben nicht mit ihrem vollen Pfad angegeben werden, sondern besitzen einen sogenannten *Freigabenamen*. Dieser Name wird normalerweise unter Windows in der folgenden Form zusammen mit dem Rechnernamen zu einem Netzwerkpfad:

*Hostname**Freigabename*

Da der Backslash aber eine Sonderbedeutung für die Shell hat, gibt es unter Linux verschiedene Möglichkeiten, wie diese Angabe gemacht werden darf:

*Hostname**Freigabename*
//Hostname/Freigabename
"*Hostname**Freigabename*"

Entweder wird also für jeden Backslash ein doppelter Backslash angegeben, oder statt Backslashs werden normale Slashs benutzt. Die dritte Alternative ist die Ausblendung der Sonderbedeutung des Backslashes durch Anführungszeichen.

Um eine SMB-Freigabe einzuhängen, wird entweder der spezielle Befehl **smbmount** benutzt, oder der normale **mount**-Befehl, zusammen

mit dem Dateisystemtyp `smbfs`. Um also die Windows-Freigabe mit dem Freigabenamen `public` des Rechners `winserver1` ins lokale Verzeichnis `/mnt` zu mounten, kann einer der beiden folgenden Befehle verwendet werden:

```
mount -t smbfs //winserver1/public /mnt
smbmount //winserver1/public /mnt
```

Allerdings wird auf diese Weise immer nach einem Passwort gefragt, selbst wenn das Verzeichnis auf dem Windows-Rechner ohne Passwort freigegeben wurde. Mit den folgenden Optionen kann mit diesen Passwörtern umgegangen werden:

username=*Name*

Der Usernamen des Windows-Users wird angegeben

password=*Passwort*

Das Passwort für die Freigabe wird angegeben

guest

Es existiert kein Passwort, es muß also auch nicht nach einem gefragt werden.

Diese Optionen werden dem **mount**-Befehl zusammen mit dem Schalter **-o** angegeben, also etwa

```
mount -t smbfs -o guest //winserver1/public /mnt
```

Der **smbmount** Befehl hängt diese Optionen - auch durch **-o** eingeleitet an den Befehl hinten an:

```
smbmount //winserver1/public /mnt -o guest
```

Um den Prozeß zu automatisieren, wird wiederum ein Eintrag in die Datei `/etc/fstab` gemacht, der unbedingt als Option entweder **guest** oder Username und Passwort benötigt.

```
//winserver1/public  /mnt   smbfs  defaults,guest 0  0
```

Wenn die Windows-Freigabe die Angabe eines Usernamens und Passwortes erwartet, so könnten hier natürlich auch die Optionen `username=...` und `password=...` angegeben werden. Das ist aber keine gute Idee, denn die Datei `/etc/fstab` ist von allen Usern lesbar. Aus diesem Grund gibt es die Möglichkeit, hier einen Dateinamen anzugeben, mit der Option **credentials**=*Dateiname*. Die angegebene Datei kann dann Einträge der Form

```
username = Wert
password = Wert
```

enthalten. Diese Datei wird nicht von aller Welt lesbar sein, also sind die Passwörter sicher.

Verzeichnisse mit NFS freigeben

Um selbst Verzeichnisse mit NFS freizugeben, müssen ein paar Bedingungen erfüllt sein, die hier näher erläutert werden sollen.

Technisch gesehen baut das Network-File-System auf einer Entwicklung auf, die *Remote Procedure Call* (RPC) genannt wird. Dieser "entfernte Prozeduraufruf" nutzt die Portnummern oberhalb 10000 um es zu ermöglichen, Client-Server Software zu steuern. Der Server stellt bestimmte Prozeduren zur Verfügung, die jeweils einer bestimmten Portnummer zugeordnet sind. Ein Client im Netz kann nun diese Prozeduren aufrufen, es entsteht eine Art gemischtes Programm, ein Teil läuft auf dem Client ab, ein anderer Teil auf dem Server.

Die Zuweisung der Ports für das RPC-System übernimmt der **portmapper**, ein Programm, das immer laufen muß, wenn ein Rechner RPC-Service anbieten will.

Die oben genannte Aufteilung macht sich NFS zu Nutzen, indem dieses System Prozeduren zur Verfügung stellt, die sich auf den Zugriff auf Dateisysteme beziehen. Diese Prozeduren werden jetzt vom NFS-Client aufgerufen, um Zugriff auf ein Dateisystem zu bekommen.

Die folgenden Voraussetzungen müssen erfüllt sein, um Verzeichnisse mit NFS freizugeben:

- Der **portmapper** muß laufen.
- Die Daemonen **rpc.nfsd** und **rpc.mountd** müssen laufen.
- Beide erhalten ihre Konfigurationsinformationen über die Datei `/etc/exports` - Diese Datei enthält die Informationen, wer was mounten darf.

Probleme bei den Zugriffsrechten

Linux erweckt zwar den Anschein, als ob es die Zugriffsrechte nach Usernamen verwalten würde, hinter den Kulissen arbeitet es aber ausschließlich mit den numerischen UserIDs. Das führt beim Mounten von Dateisystemen übers Netz zu Problemen mit der Sicherheit, was Dateizugriffe angeht.

Ein Beispiel:

Auf dem Rechner hal existieren die User:

```
Username     UID
root          0
peter         501
hans          502
```

Der Rechner hal exportiert sein `/usr` Verzeichnis jetzt an alle anderen Rechner im Netz. Dabei bleiben natürlich die Zugriffsrechte

erhalten. Doch die exportierten Dateien

```
-rw-r-----  root   root  geheim.txt
-rw-------  peter  user  noch_geheimer.txt
```

werden eben nicht nach Usernamen (root, peter) verwaltet sondern nach ihren User IDs also 0 und 501.

Der Rechner marvin will das freigegebene Verzeichnis von hal jetzt mounten. marvin hat jedoch andere User:

```
Username     UID
root          0
otto          501
hugo          502
```

Die oben genannten UserIDs sind also auch auf marvin vergeben, was dazu führt, daß marvin jetzt mit den neuen Namen arbeitet, nicht nur mit den Namen sondern eben auch mit den Rechten dieser User. Die gemounteten Dateien sähen jetzt also so aus:

```
-rw-r-----  root  root  geheim.txt
-rw-------  otto  user  noch_geheimer.txt
```

Die ganze Sache hätte zur Folge, daß otto plötzlich Dateien lesen und beschreiben kann, die eigentlich peter gehören und für alle anderen unlesbar sein sollten.

In den Fällen, in denen diese Erscheinung unvermeidbar ist (weil es z.B. nicht möglich ist, alle User überall gleich zu nummerieren) kann ein Mechanismus angewandt werden, der etwas genauere Betrachtung verdient, das Squashing.

Diese Technik wird vorwiegend bei root-Zugängen angewandt; root hat immer die UserID 0 egal ob es sich um die selbe Person handelt oder nicht. Also kann dafür gesorgt werden, daß sich der root-Zugriff auf ein gemountetes Dateisystem in eine andere UserID verwandelt. NFS versucht es zunächst mit dem Usernamen nobody, wenn es den nicht gibt, so weist es dem zugreifenden root die UID und GID -2 zu. Man spricht hier von der anonymen UserID.

Manchmal ist es auch erwünscht, anderen Usern ebenfalls die anonyme UserID zuzuweisen, NFS bietet die Möglichkeit diese Zuweisung zu übernehmen.

Der Aufbau der Datei /etc/exports

Die Datei `/etc/exports` steuert die Zugriffsrechte anderer Rechner auf die Verzeichnisse des Servers. Hier werden Verzeichnisse freigegeben und die entsprechenden Optionen (etwa ReadOnly) gesetzt.

Die Datei hat eine einfache Form, zeilenweise werden die einzelnen Verzeichnisse angegeben und mit Zugriffsbestimmungen versehen. Dabei sind Wildcards möglich um möglichst flexibel in der Unterscheidung zu sein, wer was mounten darf.

In Klammern können noch Optionen gesetzt werden, die die Zugriffsrechte betreffen (Read only ...) oder die das Squashing (siehe oben) steuern. Am Besten sehen wir uns das einmal an einem Beispiel (der Handbuchseite entnomme) an:

```
# Beispielhafte /etc/exports Datei
/               master(rw) trusty(rw,no_root_squash)
/projects       proj*.mydomain.de(rw)
/usr            *.mydomain.de(ro)
/home/joey      pc007(rw,all_squash,anonuid=501,anongid=100)
/pub            (ro,all_squash)
```

Der erste Eintrag bestimmt zwei Rechner (`master` und `trusty`) die das ganze Wurzeldateisystem mounten dürfen. Das `rw` in Klammern besagt, daß sie auch schreibenden Zugriff haben. Standardmäßig ist vorgegeben, daß der root-account über Squashing auf die Anonyme UID gelegt wird. Mit der Option `no_root_squash` wird dieses Squashing ausgeschaltet. Der Systemverwalter von `trusty` hat also tatsächlich auf dem gemounteten Dateisystem Rootrechte.

Der nächste Eintrag beschreibt die Möglichkeit aller Rechner, deren Namen mit `proj` beginnen und auf `.mydomain.de` enden, das Verzeichnis `/projects` zu mounten. Schreibender Zugriff ist möglich.

Das Verzeichnis `/usr` wird in der dritten Zeile für alle Rechner der Domain `mydomain.de` freigegeben. Es kann aber nur Read-Only gemountet werden, Veränderungen sind also nicht möglich.

Die nächste Zeile beschreibt den Rechner `pc007`, der das Verzeichis `/home/joey` mounten darf. Es handelt sich um einen typischen Eintrag für PCs, alle User werden gesquasht, als Anonyme UID wird 501 angenommen, als GID 100.

Das letzte Beispiel zeigt einen Eintrag, der von der ganzen Welt (sofern sie Zugriff auf unser Netz hat) benutzt werden kann und das Verzeichnis `/pub` als Read-Only freigibt. Alle UserIDs werden auf die Anonyme UID (nobody) gesetzt.

Eine genaue Möglichkeit zu bestimmen, welche UserIDs auf die anonyme abgebildet werden sollen, gibt es auch noch. Die Optionen `squash_uids` und `squash_gids` ermöglichen eine genaue Angabe, welche UIDs und GIDs verwandelt werden sollen. Eine gültige Angabe kann so aussehen:

```
...   (squash_uids=0-15,20,25-50)
```

Jedesmal, wenn Einträge in der Datei `/etc/exports` verändert werden, muß den beiden Daemonen **rpc.mountd** und **rpc.nfsd** ein HUP-Signal geschickt werden, um sie zu zwingen, diese Datei neu einzulesen.

Statt einem HUP-Signal kann auch der Befehl **exportfs -a** eingegeben werden, der die Daemonen zwingt, die Datei neu einzulesen.

Einfache Freigaben mit Samba

In den meisten Fällen arbeiten in Computernetzen heute die Arbeitsstationen mit Windows-Betriebssystemen, entweder Windows 95/98/ME oder WindowsNT/2000/XP. Es wäre natürlich genauso möglich, alle Arbeitsstationen mit Linux auszurüsten, das würde aber eine sehr große Umstellung bedeuten, sowohl für die einzelnen Anwender, als auch für die Verwaltung. Oft kommen auch Programme zur Anwendung, die für Linux nicht zur Verfügung stehen und die Akzeptanz von Linux ist sicher auch nicht besonders hoch, bei Menschen, die eben gerade mal gelernt haben, mit Windows umzugehen...

Damit Linux als Server für Windows-Systeme eingesetzt werden kann, ist es nötig, daß ein Serverprogramm installiert wird, das die Netzwerktechnik von Windows zur Verfügung stellt. Die Protokollfamilie, mit der Windows-Netze ihre Datei- und Druckerdienste verwalten, heißt SMB (Server Message Block) und wird von allen Windows-Systemen (von Win 3.11 bis Win 2000/XP) verwendet. Die Implementierung dieser Protokollfamilie unter Linux (und anderen Unixen) heist Samba.

Samba bietet von der einfachen Fähigkeit, Datei- und Druckerdienste in Arbeitsgruppen zur Verfügung zu stellen, über die Integration in bestehende NT-Domains bis hin zur kompletten Emulation eines NT-PDC alle Funktionen, die gebraucht werden, um Linux-Rechner in ein bestehendes oder neu zu erstellendes Windows-Netz zu integrieren. Die Windows-Arbeitsstationen merken gar nicht, daß es sich bei dem Server um einen Linux-Rechner handelt, aus ihrer Sicht arbeitet hier ein Windows Server.

Das grundlegende Prinzip ist zunächst einmal sehr einfach. Um Samba auf einem Linux-Rechner zu installieren müssen zwei Daemonen gestartet werden:

- **smbd**
  Der SMB-Daemon, der die Datei- und Druckerdienste zur Verfügung stellt.
- **nmbd**
  Der NetBIOS Name Server, der das Prinzip der Rechnernamen in einem Windows-Netz unter Linux zur Verfügung stellt.

Beide dieser Daemonen erhalten ihre Konfigurationsinformationen aus einer einzigen Datei, `/etc/smb.conf`. Diese Datei enthält alle Einstellungen, die nötig sind, um SMB-Dienste zu aktivieren.

Samba ist üblicherweise ein Stand-alone Dienst, kann aber auch via **inetd** gestartet werden. Nachdem aber in den meisten Fällen ein Rechner mit Samba-Dienst hauptsächlich als solcher arbeitet, wird der stand-alone Variante meist der Vorzug gegeben. In diesem Fall erledigt das entsprechende Init-Script (z.B.: `/etc/init.d/smb/`) die Aufgabe, sowohl den SMB-Server **smbd**, als auch den NetBIOS-Nameserver **nmbd** jeweils mit dem Parameter `-D` (für Daemon) zu starten (oder zu stoppen).

Eine erste Samba Konfiguration

Samba wird grundsätzlich über eine einzige Konfigurationsdatei verwaltet, `/etc/smb.conf`. Diese Datei muß existieren, bevor der Server selbst gestartet wird, damit er überhaupt weiß, was er tun und lassen soll.

Wir werden jetzt eine sehr einfache `smb.conf` Datei durchsprechen, die die grundlegenden Prinzipien dieser Konfigurationsdatei zeigt. Die einzelnen Bereiche werden jeweils kommentiert, erklärt und denkbare Alternativen werden aufgezeigt.

Der grundsätzliche Aufbau dieser Konfigurationsdatei entspricht dem von Windows-INI-Dateien. Das heißt, die Datei besteht aus einzelnen Abschnitten, die immer durch Überschriften in eckigen Klammern voneinander getrennt sind. Kommentare innerhalb der Konfigurationsdatei werden durch Strichpunkte eingeleitet, nicht durch Doppelkreuze (Doppelkreuze funktionieren jedoch auch).

Alle Freigaben, also sowohl Drucker, als auch Dateifreigaben werden als sogenannte *shares* bezeichnet. Jedes Verzeichnis und jeder Drucker, der freigegeben wird ist also ein *share* und hat einen eigenen Abschnitt innerhalb der Konfigurationsdatei. Ein spezieller solcher Abschnitt ist `[global]`, der globale Einstellungen ermöglicht. Aber genug der Theorie, fangen wir an. Das folgende steht in unserer `/etc/smb.conf`:

```
[global]
  workgroup = ARBEITSGRUPPE
  netbios name = MARVIN
  security = SHARE
  guest account = nobody

[public]
  comment = Oeffentliches Verzeichnis
  path = /usr/public
  guest ok = Yes
  read only = Yes
```

Wir haben nur zwei einfache Abschnitte in dieser Datei, `[global]` und `[public]`. Der erste Abschnitt beschreibt die globalen Einstellungen und muß diesen Namen tragen. Der zweite beschreibt eine Freigabe, deren Namen willkürlich auf "public" gesetzt wurde. Der Name könnte auch anders heißen, es ist einfach nur ein Name.

Die Sektion [global]

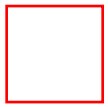
Sehen wir uns die `[global]`-Sektion einmal genauer an: Die erste Anweisung ist klar, hier wird die Windows-Workgroup definiert. Das Prinzip der Workgroups wurde von Windows 3.11 (Windows for Workgroups) eingeführt und definiert eine Arbeitsgruppe, um ein Netz überschaubarer zu halten. Windows NT (und Win 2000) arbeiten stattdessen mit dem Domain-Prinzip, aber Samba benutzt den Workgroup Eintrag später auch, um die Domain-Einstellungen vorzunehmen. Wir definieren hier also die Zugehörigkeit des Samba Servers zur Workgroup "Arbeitsgruppe".

Der zweite Eintrag (`netbios name = marvin`) definiert den Namen, der aus der Sicht von Windows Rechnern für den Samba Server gültig ist. Wir dieser Eintrag weggelassen, dann wird der Standard-Unixname (DNS Name) des Servers gewählt.

Der Eintrag `security = share` legt fest, daß die Zugangssteuerung auf Freigabeebene erfolgt, das ist die simpelste Möglichkeit, die einzige, um keine Passwörter zu brauchen. Wir werden uns später mit dieser Einstellung noch intensiv auseinandersetzen.

Der nächste (und letzte) Eintrag der Sektion `[global]` beschreibt, unter welcher UserID der Dateizugriff stattfinden soll, wenn ein nicht authorisierter User (also ein Gast ohne Passwort) auf einen Dienst zugreift. Die Tatsache, daß Windows selbst keine vergleichbaren Usereinstellungen kennt, zwingt uns hier, einen User anzugeben, damit Linux weiß, welche Rechte hier zur Verfügung stehen. In diesem Fall ist es also der User nobody, ein User ohne besondere Rechte.

Wenn wir jetzt also von einem Windows-Rechner aus die Netzwerkumgebung ansehen und die Workgroup Arbeitsgruppe aufrufen, werden wir folgendes Bild zu sehen bekommen:



Der Kommentar Samba 2.0.6 ist der voreingestellte Kommentar, den wir in der Sektion `[global]` auch ändern könnten, indem wir die Anweisung

```
server string = ...
```

eingefügen. Diese Einstellung bietet noch die Möglichkeit, den Rechnernamen mit einem %h darzustellen, die Version von Samba bekommen wir mit %v. Hätten wir also in der Sektion [global] geschrieben:

```
server string = Samba Versuchsserver auf %h (Samba %v)
```

dann stünde im Kommentarfeld der Netzwerkumgebung unter Windows

Samba Versuchsserver auf marvin (Samba 2.0.6)

Das %h und %v sind also sogenannte Zeichenkettensubstitutionen, die vom Server durch eine bestimmte Zeichenkette ausgetauscht (substituiert) werden. Alle denkbaren Zeichenkettensubstitutionen, die Samba unterstützt, finden Sie in der Zusammenfassung Zeichenkettensubstitutionen.

Die Sektion [public]

Unserer zweite Sektion in der Datei `/etc/smb.conf` heißt `[public]`. Das ist - wie oben schon erwähnt - nur ein beliebiger Name und

hat keinerlei syntaktische Bedeutung. Der Name ist allerdings der Freigabename, unter dem das freigegebene share (Verzeichnis) unter Windows dann zu sehen sein wird. Die Zeilen in diesem Abschnitt haben folgende Bedeutung:

Die erste Anweisung (`comment = Oeffentliches Verzeichnis`) definieren einen Kommentar, den wir dann in der Netzwerkumgebung unter Windows wiederfinden werden. Beachten Sie, daß Unix und Windows völlig anders mit Umlauten umgehen. Hätten wir Öffentlich statt Oeffentlich geschrieben, so hätte der Windows-User statt dem Ö ein | gesehen, was sicherlich weniger informativ gewesen wäre...

Die zweite Anweisung definiert den absoluten Pfad unter Linux, auf den sich diese Freigabe bezieht. Die Angabe `path = /usr/public` bedeutet also, daß unter dem Freigabenamen public das Verzeichnis /usr/public zu erreichen ist.

Die Angabe `guest ok = yes` bedeutet, daß dieses share ohne Passwort erreichbar ist. Stattdessen hätten wir auch schreiben können:

```
public = Yes
```

Es ist also ein öffentlich zugängliches Verzeichnis für alle Windows-User. Kein Passwort wird verlangt. Allerdings haben alle User in diesem Verzeichnis dann auch nur das Recht des Gast-Users.

Die letzte Zeile erklärt sich von selbst, `read only = Yes` meint natürlich, daß die Freigabe nur ReadOnly erfolgt, daß die Windows-User also dort nur lesen dürfen. Aus der Sicht von Windows bekommen wir also jetzt folgendes Bild, wenn wir oben den Rechner marvin angeklickt hätten:

Beachten Sie, daß die Dateien, die auf dem Rechner marvin im Verzeichnis `/usr/public` liegen nur dann lesbar sind, wenn sie - aus der Sicht von Linux - vom User nobody lesbar sind. Wir haben ja oben angegeben, daß ein Gastzugriff unter dieser UserID verwaltet wird. Jeder passwortlose Zugriff auf dieses Verzeichnis wird jetzt unter der UserID von nobody vorgenommen.

Es ist auch möglich, für jedes angegebene share einen eigenen Gast-User festzulegen. Wenn die Anweisung `guest account = ...` in einem Abschnitt vorkommt, der nicht die Sektion `[global]` ist, so gilt diese spezielle Abmachung nur für diesen share.

Zwei weitere interessante Anweisungen für jeden share sind

- **`browseable = Yes|No`**
  Wenn die Option browseable = No gesetzt wurde, dann erscheint der share nicht in der Liste der freigegebenen Verzeichnisse in der Windows Netzwerkumgebung, steht aber trotzdem zur Verfügung, wenn man den entsprechenden Pfad (\\Rechnername\Freigabename) angibt.
  Standardmäßig steht browseable auf Yes.

- **`available = Yes|No`**
  Mit dieser Option kann - wenn sie auf No gesetzt ist, ein share für den Augenblick abgeschaltet werden, ohne ihn aus der Datei /etc/smb.conf zu streichen oder langwierig auszukommentieren. Standardmäßig ist available auf Yes gestellt.

Selbstverständlich können beliebig viele shares angelegt werden, um verschiedene Verzeichnisse auf unterschiedliche Arten freizugeben. Jedes share bekommt einen eigenen Abschnitt in der `smb.conf`, beginnend mit dem Freigabenamen in eckigen Klammern, gefolgt von den entsprechenden Angaben...

Userverzeichnisse einbeziehen

Die primäre Aufgabe eines Fileservers ist es nicht nur öffentliche Dateisysteme anzubieten, sondern auch userbezogene Verzeichnisse anzubieten. Linux hat ja für jeden User der dem System bekannt ist, ein Verzeichnis, in dem dieser User alle Rechte bekommt. Er kann dort Dateien und Verzeichnisse anlegen und ist alleiniger Eigentümer dieser Daten.

Samba bietet einen sehr einfachen Mechanismus, der einem Windows User erlaubt, sein Home-Verzeichnis auf dem Linux-Rechner zu nutzen, wenn sein Username unter Linux der selbe Name wie der unter Windows ist. Erweitern wir unsere /etc/smb.conf Datei doch einmal um diesen Mechanismus zum laufen zu bringen:

```
[global]
  workgroup = ARBEITSGRUPPE
  netbios name = MARVIN
  security = SHARE
  guest account = nobody

[homes]
  comment = Heimatverzeichnis
  read only = No
  create mask = 0750
  browseable = No

[public]
   comment = Oeffentliches Verzeichnis
   path = /usr/public
   guest ok = Yes
   read only = Yes
```

Um diese Erweiterung auch aktiv zu machen müssen wir den Samba Server neu starten, am einfachsten mit

```
/etc/init.d/smb restart
```

Wenn wir uns jetzt auf unserem Windows-Rechner als - sagen wir mal - User hans anmelden - und ein User hans auf dem Linux-Rechner existiert, dann sieht die Liste der freigegebenen Shares in der Netzwerkumgebung jetzt folgendermaßen aus:

Hätten wir uns aber als otto eingeloggt (und gäbe es einen User otto unter Linux) so hätten wir statt der Angabe der Freigabe von hans jetzt eben die von otto gesehen.

Die konsequente nächste Fragestellung ist jetzt natürlich die des Passworts, das wir für diesen Zugriff benötigen. Grundsätzlich gilt:

- Wenn das Passwort unter Windows das selbe Passwort wie unter Linux ist, so gewährt Samba Zugriff.
- Wenn das Windows-Passwort nicht das selbe wie das unter Linux ist, so muß ein Passwort angegeben werden.

Windows frägt im zweiten Fall dann einfach nach dem Passwort nach und gibt auch die Möglichkeit frei, das angegebene Passwort in einer Passwortliste zu speichern. Wird das Passwort gespeichert, so muß es beim nächsten Mal nicht erneut angegeben werden.

Probleme ab Windows 98

Diese Passwort-Weitergabe funktioniert so bis Windows 95 ohne Probleme, ab Windows 98 kommt es aber nur zu der befremdlichen Fehlermeldung, das eingegebene Passwort sei falsch. Das liegt nicht daran, daß Sie es falsch eingegeben haben, sondern an einer Eigenschaft von Windows 98/NT/2000, die festlegt, daß Passwörter nicht unverschlüsselt über das Netz gegeben werden. Das ist ja eine eigentlich erfreuliche Eigenschaft, weil es die Sicherheit im Netz vergrößert, aber andererseits funktioniert jetzt der normale Mechanismus der Passwortüberprüfung des Unix-Passworts natürlich nicht mehr.

Die Lösung besteht darin, eine eigene Passwortverwaltung für den Sambadienst zu installieren. Das geschieht durch zwei einfache Schritte:

1. Die Zeile

   ```
   encrypt passwords = Yes
   ```

   wird in die `[global]` Sektion der `/etc/smb.conf` aufgenommen
2. Eine spezielle Datei `/etc/smbpasswd` enthält die Userinformationen über die Samba-User samt ihrer verschlüsselten Passwörter.

Um jetzt also diese Passwort-Datei anzulegen und verschlüsselte Passwörter zu generieren benötigen wir wiederum ein kleines Programm mit Namen **smbpasswd**. Dieses Programm erlaubt es, sofern es der Superuser root anwendet, neue Samba-User anzulegen und ihnen Passwörter zu vergeben. Ein Normaluser kann nur sein Passwort damit ändern. Um einen neuen User anzulegen schreibt root jetzt

```
smbpasswd -a Username
```

Er wird dann gleich nach dem Passwort des neuen Users gefragt, das Passwort wird verschlüsselt und zusammen mit der UserID des Users in der Datei `/etc/smbpasswd` abgelegt.

Damit also alles jetzt wirklich funktioniert muß die `[global]`-Sektion unserer smb.conf Datei jetzt folgendermaßen aussehen:

```
[global]
  workgroup = ARBEITSGRUPPE
  netbios name = MARVIN
  security = SHARE
  guest account = nobody
  encrypt passwords = Yes
```

Und schon funktioniert die Freigabe der Userverzeichnisse.

Die Sektion `[homes]` ist also - wie die Sektion `[global]` eine spezielle Möglichkeit. Sie muß grundsätzlich diesen Namen tragen denn Samba übernimmt ja sehr spezielle Ersetzungsmechanismen, wenn dieser share angesprochen wird.

Drucker freigeben

Um Drucker für Windows-Rechner freizugeben, die an Linux Rechnern hängen, sind ein paar kleine Erweiterungen an unserer smb.conf Datei nötig. Zunächst einmal muß geklärt werden, wie Linux druckt. In der Regel wird Linux das BSD-Drucksystem benutzen, manchmal - seltener - aber auch das PLP System. Samba muß in der `[global]`-Sektion einen entsprechenden Eintrag besitzen:

```
[global]
  workgroup = ARBEITSGRUPPE
  netbios name = MARVIN
  security = SHARE
  guest account = nobody
  encrypt passwords = Yes
  printing = BSD
```

Um jetzt Drucker selbst freizugeben haben wir zwei Möglichkeiten:

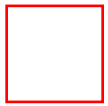
- Alle Drucker werden freigegeben
- Bestimmte Drucker werden freigegeben

Wenn alle Drucker freigegeben werden sollen wird ein Mechanismus benutzt, der dem der Userverzeichnisse ähnelt. Wir legen ein spezielles share an, das zwingend [printers] heissen muß. Dieses share enthält folgende Anweisungen:

```
[printers]
  comment = All Printers
  path = /tmp
  create mask = 0700
  printable = Yes
  browseable = No
  guest ok = Yes
```

Der angegebene Pfad bezieht sich auf ein Verzeichnis, in dem alle User Schreibrecht haben (und dem also das Sticky-Bit gesetzt sein sollte). Die Create Mask legt fest, welche Permissions Druckaufträge haben sollen, die in diesem Verzeichnis liegen. Die Angabe printable = Yes bedeutet, daß - selbst wenn das Verzeichnis selbst Read only = Yes als Parameter hat, Druckaufträge dort abgelegt werden dürfen. Die Angabe browseable = No bedeutet, daß der share `[printers]` selbst nicht erscheint.

Auf diese Weise werden alle Drucker, die unter Linux in der `/etc/printcap` Datei auftauchen, dem Windows-Rechner angezeigt und freigegeben. Das heißt, daß auch wenn - wie bei vielen Distributionen üblich - mehrere logische Drucker angelegt werden, all diese logischen Drucker auch angezeigt werden:

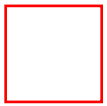


Das kann einen unbedarften Windows-User natürlich verwirren. Um aber nur einen einzigen Drucker anzuzeigen, wird statt des `[printers]` Abschnitts ein Abschnitt für den freizugebenden Drucker angelegt, dessen Name jetzt wieder frei wählbar ist:

```
[PSLaser]
  comment = PS_600dpi-a4-auto-mono-600
  path = /tmp
  create mask = 0700
  guest ok = Yes
  printable = Yes
  printer name = lp2
```

Der entscheidende Unterschied zu einem share für Verzeichnisse sind eigentlich nur die Anweisungen `printable = yes` und `printer name = lp2`

Daran merkt Samba, daß es sich hierbei um einen Drucker-Dienst und nicht um einen Dateidienst handelt. Der Windows-User bekommt jetzt folgendes Bild zu sehen:



Die Angabe printer `name = lp2` bezieht sich auf den Namen des Druckers, der unter Linux gültig ist. In diesem Fall darf natürlich auch nicht `browseable = No` eingetragen werden, sonst sieht der Windows-User den Drucker nicht in seiner Liste.

Die Angabe `guest ok = yes` ermöglicht das Drucken ohne Passworteingabe.

Sicherheitsebenen

Samba kennt verschiedene Sicherheitsebenen, die alle durch die Angabe

```
security = ...
```

in der `[global]`-Sektion eingestellt werden. Das heißt, sie gelten immer für den gesammten Server und sind nicht für jedes einzelne share einstellbar. Denkbare Werte für security sind `share, user, server` und `domain`. Diese vier Sicherheitsebenen werden hier einzeln erläutert.

security = share

Bisher haben wir in unsere Samba Konfiguration immer die Zeile

```
security = share
```

in die `[global]`-Sektion eingetragen. Dieser Eintrag entspricht der "Zugriffssteuerung auf Freigabeebene" unter Windows. Es ist die geringste Sicherheitsstufe, die Zugriff auf bestimmte shares nur über die Frage klärt, ob ein share öffentlich ist oder nicht. Allerdings kann auch unter dieser Einstellung mit Samba eine userbezogene Freigabe realisiert werden, indem dem share die Anweisung

```
guest ok = No
```

gesetzt wird. In diesem Fall wird Samba ein Passwort anfordern, das zu dem Usernamen passen muß, der unter Windows angegeben wurde. Auch hier gilt, daß ab Win98 die Passwörter verschlüsselt übermittelt werden, also die Zeile

```
encrypt passwords = Yes
```

in der `[global]`-Sektion stehen muß. Zusätzlich muß der entsprechende User sowohl Linux bekannt sein, als auch ein verschlüsseltes Passwort in der Datei `/etc/smbpasswd` besitzen. (Siehe Userverzeichnisse)

Die Frage, wer dann nun was in dem Verzeichnis anstellen darf wird wiederum von Linux selbst beantwortet. Es gelten die normalen Unix-Dateizugriffsrechte des jeweiligen Users. Das ist auch der Grund, warum jeder Samba-User auch eine gültige Unix UserID besitzen muß.

Zusammenfassend ist also zu sagen, daß Samba mit dieser Methode noch wesentlich sicherere Zugriffe zulässt, als es Windows95/98 zulassen würde. Andererseits ist diese Methode erheblich eingeschränkt, was die Möglichkeiten der userbasierten Zugriffskontrolle angeht.

`security = share` wird hauptsächlich in Systemen eingesetzt, die alle Dienste frei - ohne Passwörter - anbieten wollen, oder in Systemen, deren User nicht identisch mit den Windows-Usern sind.

security = user

Eine andere Möglichkeit wird durch die Einstellung

```
security = user
```

in der `[global]`-Sektion ermöglicht. Das ist - seit Samba 2.0 - die voreingestellte Methode. In dieser Einstellung muß sich ein Windows-User zuerst "einloggen", bevor er irgendwelche Dienste des Samba-Servers in Anspruch nehmen darf. Das heißt, er muß einen Usernamen und ein Passwort an den Samba-Server schicken, der wiederum überprüft, ob die beiden Angaben stimmen und erst nach erfolgreicher Prüfung Zugriff gewährt. Der Zugriff ist dann wiederum abhängig von den Linux-Rechten, die dieser User auf dem Samba-Server hat. Das heißt, daß diese Methode nur dann funktioniert, wenn auf dem Windows-Rechner die selben Usernamen existieren, wie auf dem Samba-Server.

Wie schon bei früheren Passwortübermittlungen wird auch hier wieder zwischen unverschlüsselter Passwortübermittlung (bis einschließlich Win95) und verschlüsselter Passwortübergabe (ab Win98) unterschieden. Das heißt, wenn die Windows-Rechner mindestens Win98 fahren, muß die Zeile

```
encrypt passwords = Yes
```

in der `[global]`-Sektion stehen und die User müssen mittels **smbpasswd** angelegt werden.

Beachten Sie, daß der Name der angeforderten Resource nicht an den Server weitergeschickt wird, bevor sich der Windows-User nicht korrekt angemeldet hat. Das ist der Grund, warum Gast-shares (guest ok) in diesem Modus nicht funktionieren, ohne daß User automatisch in einen Gastaccount umgewandelt werden. Aber auch dazu müssen sie sich zuerst korrekt anmelden, also einen Useraccount besitzen.

security = server

In diesem Modus versucht Samba die Passwortüberprüfung nicht selbst vorzunehmen, sondern die angegebenen Username/Passwort Kombinationen an einen anderen SMB-Server zur Überprüfung weiterzuleiten (z.B. einen NT-Rechner). Wenn das fehlschlägt, wird automatisch auf die Einstellung `security = user` zurückgeschaltet.

Damit Samba weiss, an welchen Server es die Passwortüberprüfung weiterleiten soll, muß der entsprechende Server mit der Angabe `password server = ...` in der `[global]`-Sektion angegeben werden. Der angegebene Name muß ein NetBIOS Name sein, also der Name, unter dem der Rechner auch unter Windows ansprechbar ist. Dieser angegebene Server muß selbst im `security = user` Modus laufen, um die entsprechende Überprüfung vorzunehmen.

Achtung: Versuchen Sie niemals hier den Samba Server selbst einzutragen! Es würde zu einer Endlosschleife kommen, die den gesammten Samba-Server blockieren würde.

Aus der Sicht des Windows-Clients ist der Server-Modus absolut identisch mit dem User-Modus. Der Unterschied bezieht sich nur auf die Frage, wie der Samba-Server die Überprüfung der Zugriffsrechte vornimmt. Im User-Modus macht er es selbst, im Server-Modus leitet er es an einen anderen Server weiter. Natürlich kann dieser andere Server auch wieder ein Samba Rechner sein, der dann aber eben im User-Modus laufen muß...

Beachten Sie, daß der Name der angeforderten Resource nicht an den Server weitergeschickt wird, bevor sich der Windows-User nicht korrekt angemeldet hat. Das ist der Grund, warum Gast-shares (guest ok) in diesem Modus nicht funktionieren, ohne daß User automatisch in einen Gastaccount umgewandelt werden. Aber auch dazu müssen sie sich zuerst korrekt anmelden, also einen Useraccount besitzen.

security = domain

Dieser Modus funktioniert nur dann, wenn der Samba-Server mittels des Programms **smbpasswd** an eine bestehende Windows NT Domain angeschlossen wurde. Der Parameter `encrypted passwords` muß aktiviert sein. In diesem Modus versucht der Samba-Server alle Authentifizierungen an einen Primären- oder Backup Domain Controller einer Windows-NT Domain weiterzuleiten, genau so, wie es eine NT-Maschine selbst tun würde.

Trotzdem müssen User dem Linux-System bekannt sein, damit eine Bestimmung ihrer Zugriffsrechte auf einzelne Verzeichnisse möglich ist. Jeder User braucht also einerseits einen Account auf dem PDC der Windows-Domain und andererseits einen gültigen Acccount auf dem Linux-Server.

Wie schon bei dem Server-Modus, so ist dieser Modus aus der Sicht des Windows-Clients einfach der User-Modus. Der Unterschied liegt nur darin, auf welche Weise der Samba-Server die Passwörter authentifiziert.

Wie schon im Server-Modus, so muß auch hier der Rechner angegeben werden, der die Passwortüberprüfung vornimmt. Wieder benutzen wir die Angabe

```
password server = ...
```

in der `[global]`-Sektion. Nur jetzt geben wir nicht einen Server an, sondern eine Liste, durch Kommata getrennt. Hier können wir den PDC und alle existierenden BDCs angeben, also etwa

```
password server = NT-PDC, NT-BDC1, NT-BDC2
```

Wird statt einer Liste einfach nur ein Pluszeichen (+) angegeben, so versucht Samba den PDC und die BDCs selbst herauszufinden.

Beachten Sie auch, daß der Name der Domain, der wir uns anschließen, wiederum mit dem Parameter `workgroup = ...` angegeben wird.

Damit unser Samba-Server aber überhaupt Mitglied einer Domain werden kann, müssen wir erst dafür sorgen, daß er davon auch weiss. Dazu kommt wiederum das Programm smbpasswd zur Anwendung, das wir ja schon zum Anlegen der User und Wechseln der Passwörter benutzt hatten.

Mit dem Aufruf von

```
smbpasswd -j Domain
```

wird der Samba-Server in die genannte Domain aufgenommen. Damit das funktioniert, muß der Administrator der NT-Domain mit dem Programm *Server Manager for Domains* den primären NetBIOS Namen des Samba-Servers in die Liste der Domain-Mitglieder aufgenommen haben.

Erst nachdem dieser Befehl ausgeführt wurde, sollte die smb.conf auf `security = domain` gesetzt werden. Von nun an sendet der Samba-Server alle Anfragen an den PDC zur Authentifizierung weiter.

NMBD Einstellungen

Alle bisherigen Einstellungen, die wir in den vorangehenden Beispielkonfigurationen getroffen hatten, bezogen sich auf den SMB-Daemon smbd. Es ging um die Fragen der Freigaben und Sicherheitskonzepte. Damit ein Windows-Netz aber richtig funktioniert, muß zusätzlich noch ein Dienst laufen, der die NetBIOS-Namen in IP-Adressen umsetzt und die Liste der Windows-Netzwerkumgebung verwaltet, der NMBD. Dieser Daemon wird grundsätzlich zusammen mit dem SMBD gestartet, entweder über inetd oder als Stand-Alone-Server.

Die primäre Aufgabe dieses Daemons ist es, die Namensauflösung im SMB-Netz zu ermöglichen. Dabei ist es unerheblich, ob die entsprechenden NetBIOS-Namen den DNS-Namen entsprechen, es gibt aber viele Fälle, wo sich Probleme vermeiden lassen, wenn zumindestens der Samba-Server hier immer den selben Namen trägt.

Die Einstellungen des NMBD werden auch in der Datei /etc/smb.conf vorgenommen, dort aber nur in der `[global]`-Sektion. Dieses

Kapitel beschreibt die notwendigen Einstellungen für die Namensauflösung, im nächsten Kapitel betrachten wir die Verwaltung der Browsing-List (Windows-Netzwerkumgebung), die auch vom NMBD übernommen wird.

Etwas genaueres zu NetBIOS Namen

NetBIOS ist die grundlegende Protokollfamilie für Windows-Netze. Es arbeitet mit den folgenden drei Transport/Vermittlungsschicht Protokollen:

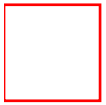
- TCP/IP
- NetBEUI
- IPX/SPX

Samba benutzt grundsätzlich nur NetBIOS über TCP/IP. NetBIOS Anwendungen (wie Samba) bieten ihre Dienste im Netz unter einem NetBIOS Namen an. Dieser Name muß vorher im Netz beansprucht worden sein. Nachdem aber diese Namen nicht zwangsläufig mit den DNS-Namen im TCP/IP Netz übereinstimmen müssen, gibt es zwei grundsätzliche Herangehensweisen, um im Netz Kommunikation zu ermöglichen. Entweder werden Broadcast-Aufrufe an alle angeschlossenen Rechner verschickt, oder es wird ein spezieller Windows Internet Name Service (WINS) angeboten, der die Auflösung von Namen in IP-Adressen vornimmt.

Größere Netze sollten grundsätzlich die zweite Möglichkeit (WINS) in Anspruch nehmen, denn sonst wird das Netz mit unnötig vielen Paketen belastet.

Um mit WINS arbeiten zu können muß ein Rechner im Netz diesen Dienst auch anbieten. Das kann entweder ein WindowsNT/2000 Rechner sein, oder eben wieder ein Samba Server. In der Regel wird der NT Rechner benutzt, wenn Samba in einer gemischten NT/Samba Umgebung arbeitet. Wenn nur Samba-Server zur Verfügung stehen, wird einer dieser Server den Dienst übernehmen.

Windows-Rechner müssen diesen Server dann entsprechend unter den Eigenschaften von TCP/IP eintragen:



Zu beachten ist, daß - um die Netzwerkumgebung richtig darstellen zu können - immer nur ein WINS-Server pro Netzsegment (bzw. Workgroup) aktiv sein sollte. Ansonsten sehen die verschiedenen Windows-Clients jeweils nur die Rechner, die den entsprechend gleichen WINS-Server benutzen.

Samba als WINS-Client

Um einen Samba Server an einen bestehenden WINS-Server anzuhängen, also ihn zu zwingen, seine Dienste und Namensauflösungen über diesen Server vorzunehmen, muß nur die folgende Zeile in der `[global]`-Sektion eingetragen werden:

```
[global]
  ...
  wins server = 123.45.67.89
  ...
```

Der Befehl `wins server =` erfordert entweder die IP-Adresse oder den DNS-Domainnamen des entsprechenden Servers.

Viele Dienste von Samba, die die Browsing-Liste betreffen funktionieren nur, wenn ein WINS-Server im Netz aktiv ist. Ob dieser Server jetzt ein NT-Rechner oder wiederum ein Samba-Server ist, spielt keine Rolle.

Samba als WINS-Server

Wenn in einem Netz keine NT-Maschinen laufen, so kann auch Samba selbst als WINS-Server arbeiten. Allerdings darf in diesem Netz dann absolut nur ein Samba-Server als WINS-Server agieren.

Die notwendige Einstellung in der smb.conf lautet:

```
[global]
  ...
  wins support = yes
  ...
```

Beachten Sie, daß NIEMALS beide Einstellungen (Server und Client) gleichzeitig benutzt werden dürfen. Wenn Samba als Server arbeitet ist die Zeile `wins server = ...` nicht nur unnötig, sondern schlicht verboten.

1.113.5

Einrichtung und Konfiguration grundlegender DNS-Dienste

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Namensauflösung zu konfigurieren und Probleme mit lokalen caching-only Nameservern zu lösen. Dies benötigt ein Verständnis der Prozesse der Domainregistrierung und der DNS-Auflösung. Ebenfalls erforderlich ist ein Verständnis der wichtigsten Unterschiede der Konfigurationsdateien von BIND und BIND 8.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/hosts`
- `/etc/resolv.conf`
- `/etc/nsswitch.conf`
- `/etc/named.boot` (v.4) oder `/etc/named.conf` (v.8)

Adressen in einem IP-Netzwerk sind immer nummerische Adressen. Diese Adressen sind zwar logisch aufgebaut, jedoch nicht unbedingt dafür geeignet, daß man sich davon 500 verschiedene merkt. Menschen erinnern sich einfach lieber an Namen als an Nummern.

Um dieser Tatsache Rechnung zu tragen bietet Linux ein System an, wie IP-Adressen mit Namen verknüpft werden können.

Die Bibliotheksfunktionen **gethostbyname** und **gethostbyaddr** sind für die Verwaltung der Hostnamen und IP-Adressen zuständig. Jedes Programm, das mit IP-Adressen oder Hostnamen umgeht, benützt diese Funktionen.

Es existieren mehrere Informationsquellen, die von diesen beiden Funktionen benutzt werden, um einen Namen mit einer Adresse zu verknüpfen.

- **Die Datei /etc/hosts**
  Diese Datei enthält IP-Adressen und dazu passende Namen (Siehe auch Abschnitt 1.112.3). Sie dient dazu, die Namen der Rechner im lokalen Netz zu verwalten und eventuell noch häufig benutzte Namen fremder Rechner mit Adressen zu versehen. Diese Technik

ist aber sehr statisch. In einem lokalen Netz können wir uns darauf verlassen, daß sich Adressen nicht dauernd ändern, aber in einem fremden Netz gibt es dafür keine Garantie.

- **Das Domain Name System (DNS)**
  Dieses System teilt das gesamte Internet in Verwaltungseinheiten (domains) auf, die autonom administriert werden. Jede Domain betreibt sogenannte Nameserver, die die Adressen und Hostnamen ihres Verwaltungsgebietes kennen. Die Nameserver sind über eine Art Baumstruktur miteinander verbunden, so daß man die Summe aller Nameserver als globale und verteilte Datenbank betrachten kann. Jeder Nameserver kann über die Informationen über die über ihm liegenden Nameserver jede beliebige Adresse im Internet herausfinden. Welche Nameserver benutzt werden sollen, wird in der Datei `/etc/resolv.conf` eingestellt.

In welcher Reihenfolge diese Informationsquellen herangezogen werden, kann in den Dateien `/etc/host.conf` und `/etc/nsswitch.conf` festgelegt werden. All diese Zusammenhänge sind bereits im Abschnitt 1.112.3 besprochen worden.

Die Technik der Nameserver bedingt natürlich eine zentrale Verwaltung. Diese Verwaltung wird von einer Organisation mit Namen IANA (Internet Assigned Numbers Authority) abgewickelt. Die Struktur des gesamten DNS-Systems wird über eine überschaubare Menge von Toplevel-Domains gesteuert.

Es gibt zwei unterschiedliche Arten von Toplevel-Domains. Die sogenannten *generischen* und die *länderspezifischen* Toplevel-Domains. Die generischen Domains sind 14 festgelegte Domains, die alle eine bestimmte Bedeutung haben:

- **.aero**
  neu, reserviert für Mitglieder der Luftfahrtindustrie.
- **.biz**
  eine neu angelegte Toplevel-Domain nur für Unternehmen (businesses).
- **.com**
  die alte Toplevel-Domain für kommerzielle Unternehmen.
- **.coop**
  eine neue Domain für Kooperativen.
- **.edu**
  reserviert für Bildungseinrichtungen, die bei einer der sechs US-amerikanischen Agenturen akkreditiert sind.
- **.gov**
  reserviert für Einrichtungen der US-amerikanischen Regierung.
- **.info**
  eine neue Domain für informative Einrichtungen.
- **.int**
  eine neue Domain, reserviert für Organisationen, die aufgrund internationaler Verträge zwischen Regierungen zustande kamen.

- **.mil**
  reserviert für Einrichtungen des US-amerikanischen Militärs.
- **.museum**
  eine neue Domain nur für Museen.
- **.name**
  eine neue Domain, für die Verwendung im Privatbereich.
- **.net**
  ursprünglich für die Netzverwaltung gedachte Domain, die heute frei verfügbar ist.
- **.org**
  eine Domain für nicht kommerzielle Organisationen.
- **.pro**
  Eine im Aufbau befindliche Domain, reserviert für Berufsverbände.

Neben diesen generischen Toplevel-Domains existieren für jedes Land der Erde eine länderspezifische Domain. Diese Domain trägt als Namen die zwei Buchstaben ISO-Länderkennung. Eine vollständige Liste der existierenden Länderdomains erhalten Sie bei der IANA selbst unter www.iana.org/cctld/cctld-whois.htm.

Jede Toplevel-Domain wird von einer bestimmten Organisation verwaltet, die wiederum Domains unterhalb ihres Verwaltungsgebietes vergeben kann. So kann etwa die Verwaltung der deutschen Topleveldomain `.de` der Firma `foo` die Domain `foo.de` überlassen. Wichtig ist dabei, daß jede echte Domain - egal ob Toplevel oder nicht - einen oder mehrere Nameserver betreiben muß, die ihren jeweiligen Zuständigkeitsbereich abdeckt.

Auf der Wurzel des Baums des DNS arbeiten noch die sogenannten *root-server*, Nameserver, die die Namen aller Toplevel-Domains kennen und die Information über die von ihnen verwendeten Nameserver bereithalten.

Jeder Nameserver, egal ob der einer Toplevel-Domain oder einer gewöhnlichen, kennt die Liste aller *root-server*. Wenn jetzt ein Nameserver nach einer bestimmten Adresse sucht, so fragt er zunächst einmal einen *root-server* nach dem Nameserver der Toplevel-Domain dieser Adresse. Der wiederum kennt den Nameserver der Subdomain und der kennt - hoffentlich - die Adresse, die gesucht wurde.

Das gesamte Domain-System des Internets kann als eine große, weltweit verteilte Datenbank verstanden werden. Die einzelnen Knoten der Datenbank sind die Nameserver der verschiedenen Domains, die über die root-Server miteinander verbunden sind. Wie jede andere Datenbank auch, so hat auch die DNS-Datenbank eine vorgegebene Struktur und vorgegebene Feldeinträge.

Jeder Datensatz einer DNS-Datenbank besteht aus folgenden Elementen:

Domain-name

Der Name der Domain, auf den sich der Datensatz bezieht.

Time-to-live

Dieser Wert ist optional und bezeichnet die Stabilität eines Eintrags, also die Frage, wie lange er gültig sein soll. Wird häufig weggelassen.

Type

Gibt an, um was für einen Typ Datensatz es sich handelt. Eine genaue Beschreibung der möglichen Typen finden Sie weiter unten.

Class

Die Informationsklasse. Für Internet Informationen steht hier immer der Begriff `IN`.

Value

Der eigentliche Wert des Datensatzes. Abhängig vom genannten Typ.

Die Datenbank besteht also aus Einträgen, die sich alle an dieses Format halten. Für den Typ der Information (Type) stehen folgende Werte zur Verfügung:

Typ	**Bedeutung**	**Eintrag**
`SOA`	Start Of Authority	Verschiedene Parameter für die Zone, die der Nameserver verwalten soll.
`A`	Address	Die Adresse eines Internet Hosts.
`MX`	Mail Exchange	Die Priorität und Name des Mailservers der Domain.
`NS`	Name Server	Name eines Nameservers der Domain.
`CNAME`	Canonical Name	Domainname eines Rechners (Aliasfunktion)
`PTR`	Pointer	Alias für eine numerische IP-Adresse
`HINFO`	Host Information	ASCII Beschreibung des Hosts (CPU, OS, ...)
`TXT`	Text	Nicht verwertbarer Text - Kommentar

Die einzelnen Einträge werden in Konfigurationsdateien geschrieben, die jeweils für eine sogenannte Zone gelten. Eine Zone entspricht entweder einem physikalischem Netz oder einer Domain selbst.

Für jede Zone existieren zwei solcher Konfigurationsdateien, eine für die Auflösung von Namen in Adressen und eine für die Auflösung von Adressen in Namen (reverse lookup).

Eine zentrale Konfigurationsdatei (natürlich im Verzeichnis `/etc`) sorgt für die Information, für welche Zonen der Nameserver verantwortlich ist und gibt die Positionen der Zoneninformationsdateien im Dateisystem an. Meist liegen diese Dateien unter `/var/named`.

Zuletzt existiert noch eine Datei `root.hint`, die nicht selbst editiert wird und die die Informationen über die root-Server im Internet enthält. Diese Information ist nötig, damit unser Nameserver in den weltweiten Verbund des DNS eingebunden ist.

Ein lokales Netz, das nicht permanent ans Internet angeschlossen ist und das vor allem nicht mit echten IP-Adressen arbeitet, muß natürlich keinen kompletten Nameserver unterhalten. In den meisten Fällen genügt es, einen sogenannten *caching only nameserver* bereitzustellen, der die Abfragen an andere Nameserver weiterleitet, aber selbst keine Informationen zur Verfügung stellt. Der Nameserver kann die Informationen über die bereits gefundenen Adressen zwischenspeichern (caching) und muß so nicht für jede Adresse erneut eine Anfrage starten.

Ein solcher Nameserver muß also keine Informationen über das lokale Netz beinhalten, sondern nur die Liste der *root-server* kennen. Diese Liste ist im Internet frei erhältlich und liegt bei der Installation eines Nameservers gewöhnlich bei. Meist heißt diese Datei `root.hint` (ab Version 8.0) oder `named.ca, named.root` (Version 4.x) und wird ins Verzeichnis `/var/named` installiert.

Es existieren heute zwei Generationen von Nameservern. Die erste benutzt den Berkley Internet Name Daemon **bind** Version 4, die zweite benutzt **bind** Version 8.0 (oder später). Dieser Generationsunterschied spielt eine große Rolle hinsichtlich der Syntax der Konfigurationsdateien. Moderne Installationen sollten heute alle mit Versionen ab 8.0 arbeiten. Die alten Versionen sind sehr unsicher!

Für den Prüfungsinhalt der LPI102 Prüfung ist noch das Wissen über beide dieser Versionen und vor allem über den Unterschied in den Konfigurationsdateien gefragt. Also werden wir beide Installationen durchspielen. Nochmal die Aufforderung: Es sollten nur noch Versionen ab 8.0 verwendet werden!

Konfiguration eines caching-only Nameservers unter bind Version 4.0

Die zentrale Konfigurationsdatei der älteren bind-Version liegt im Verzeichnis `/etc` und heißt `named.boot`. Diese Datei enthält die grundlegenden Informationen über folgende Einstellungen:

- In welchem Verzeichnis liegen die Informationsdateien des Nameservers.
- Welche Zonen werden von diesem Server verwaltet und in welchen Dateien liegen die entsprechenden Informationen.
- Ist dieser Nameserver der primäre Server einer Domain oder ein sekundärer.
- Welche Nameserver sollen gefragt werden, wenn dieser Nameserver nicht in der Lage ist, eine Adresse aufzulösen (forwarders).
- Soll der Nameserver selbst Anfragen starten, oder nur einen anderen Nameserver fragen.

Nachdem unser Nameserver nur als caching only Server betrieben werden soll, sind nur die wenigsten dieser Einstellungen für uns relevant. Eine einfache `/etc/named.boot` Datei könnte folgendermaßen aussehen:

```
;
```

```
; /etc/named.boot Datei fuer einen caching only Server
;
directory     /var/named
;
;             domain                    file
;-----------------------------------------------
cache         .                         named.root
primary       0.0.127.in-addr.arpa      named.local
```

Kommentare in dieser Datei werden durch einen Strichpunkt (;) eingeleitet. Die erste Anweisung, die kein Kommentar ist, gibt an, in welchem Verzeichnis alle Dateien gefunden werden, auf die wir uns im weiteren Verlauf der Datei beziehen. In unserem Fall ist das das Verzeichnis `/var/named`.

Die nächste (nicht-Kommentar) Zeile gibt die Datei an, die für den caching-only Server die wichtigste ist. Die Datei `named.root` wird - dank der letzten Anweisung - im Verzeichnis `/var/named` gesucht. Sie enthält die Informationen über die root-server im Internet und sollte nicht selbst verändert werden. Die Datei kann - falls sie nicht bei der Installation bereits vorhanden war - im Internet über anonymes FTP bezogen werden: Dateiname `/domain/named.root` auf dem Rechner `FTP.RS.INTERNIC.NET`.

Über diese Datei hat unser Nameserver jetzt Kontakt zum weltweiten DNS. Er kennt jetzt alle root-Server und kann somit jede Adresse im Internet auflösen. Er wird sich auch gerade aufgelöste Adressen merken, so daß nicht für jede Adresse erneut ein Lookup stattfinden muß.

Die letzte Zeile unserer Konfigurationsdatei verweist auf die Datei `/var/named/named.local`, die ausschließlich die Information zum *localhost* (127.0.0.1) also zu unserem eigenen Rechner enthält. Diese Datei entspricht vom Aufbau her der Form, die auch "echte" Informationsdateien des Nameservers hätten, wenn sie verantwortlich für eine Domain wären:

```
;
; /var/named/named.local       Reverse mapping of 127.0.0
;
@                   IN  SOA   ns.mydomain.de. (
                              root.mydomain.de.
                              1          ; serial
                              360000     ; refresh: 100 Std
                              3600       ; retry:   1 Std
                              3600000    ; expire:  42 Tage
                              360000     ; minimum: 100 Std
                              )
                    IN  NS    ns.mydomain.de.
```

```
1                   IN  PTR   localhost.
```

Diese Datei enthält 3 Datensätze. Der erste ist der sogenannte *Start of Authority* (SOA) Eintrag. Er definiert den Namen unseres Nameservers (`ns.mydomain.de`), die E-Mail Adresse des Verwalters, wobei statt dem @ Zeichen ein normaler Punkt verwendet wird (`root.mydomain.de`) und eine zusammengesetzte Seriennummer, die festlegt, wie lange Einträge des Nameservers gültig sein sollen.

Der zweite Eintrag definiert den Nameserver (NS) der verwendet werden soll - in unserem Fall ist das der Rechner `ns.mydomain.de` selbst.

Der dritte und letzte Eintrag ist die Angabe, daß die Adresse, die auf 1 endet (bei 127.0.0.1 immer der Localhost) eben der *localhost* ist.

Wenn diese Einträge jetzt existieren, kann der Nameserver gestartet werden. Normalerweise wird das über ein Init-Script (z.B. `/etc/init.d/named start`) erledigt. Bei den **bind4** Nameservern existierten meist auch noch zwei Hilfsprogramme **named.restart** und **named.reload**, mit denen ein Nameserver neu gestartet werden kann, oder gezwungen werden kann, die Konfigurationsdateien neu einzulesen.

Konfiguration eines caching-only Nameservers unter bind Version 8.0

Der wesentliche Unterschied bei den Konfigurationsdateien der beiden Nameserver-Generationen liegt in der zentralen Konfigurationsdatei. Diese Datei enthält auch bei **bind8** die selbe Information, wie bei **bind4**, jedoch heißt die Datei anders und hat einen völlig anderen Aufbau.

Die zentrale Konfigurationsdatei der modernen Nameserver heißt `/etc/named.conf` und ist ähnlich aufgebaut, wie ein C-Programm. Logische Blöcke werden in geschweifte Klammern zusammengefasst. Nach einer geschlossenen Klammer folgt - wie auch nach jeder Anweisung - ein Strichpunkt. Die Konfigurationsdate eines caching only Servers könnte folgendermaßen aussehen:

```
// Konfigurationsdatei für einen caching-only Nameserver
// /etc/named.conf

options {
        directory "/var/named";
};

zone "." {
        type hint;
        file "root.hints";
};
```

```
zone "0.0.127.in-addr.arpa" {
        type master;
        file "local.zone";
};
```

Der erste Block (`options`) definiert globale Optionen. In unserem einfachen Beispiel findet sich hier nur die Information, in welchem Verzeichnis die Nameserver-Dateien zu finden sind, auf die sich der Rest der Einträge hier bezieht. Der Eintrag entspricht also genau der entsprechenden Angabe der alten Konfigurationsdatei.

Die einzelnen Zonen werden jetzt auch als Blöcke verwaltet, nicht mehr in einer Zeile, wie oben. Jede Zone hat mindestens einen Typ und eine Angabe einer Datei, die die Informationen über diese Zone beinhaltet. Wie oben haben wir wieder zwei Einträge, einmal für die Zone "." (die Wurzel des DNS-Baums) und einmal für unsere lokale Zone, das lokale Netz. Wie oben schon wird diese Angabe umgekehrt geschrieben, also statt `127.0.0` steht hier `0.0.127.in-addr.arpa`.

Die einzelnen Dateien in `/var/named` unterscheiden sich nicht, außer in ihrem Namen. Und der wäre prinzipiell frei wählbar. Immerhin können wir zumindest bei der Zonendatei `local.zone` den SOA-Eintrag etwas verständlicher gestalten:

```
;       Zonendatei für den localhost
@               IN      SOA     ns.mydomain.de. root.mydomain.de. (
                                1       ; Serial
                                8H      ; Refresh
                                2H      ; Retry
                                1W      ; Expire
                                1D)     ; Minimum TTL
                        NS      ns.mydomain.de.
1                       PTR     localhost.
```

Die Angaben über die Zeiten können hier mit Prefixen versehen werden, die die Zeiteinheiten bezeichnen, statt sie immer in Sekunden angeben zu müssen. Die Prefixe `H, D, W` stehen für Stunde (hour), Tag (day) und Woche.

Auch hier wird der Nameserver über ein Init-Script gestartet, allerdings existieren die beiden Programme **named.restart** und **named.reload** nicht mehr. Stattdessen kann dem Nameserverprozeß ein HUP-Signal geschickt werden, das ihn zwingt, seine Konfigurationsdatei neu einzulesen.

Beide Nameserver-Versionen geben Diagnosemeldungen an den Syslog Daemon weiter. Ein Studium der Datei `/var/log/messages`

sollte also im Fehlerfall immer der erste Schritt zur Problemlösung sein.

1.113.7

Einrichten von Secure Shell (OpenSSH)

Beschreibung: Prüfungskandidaten sollten in der Lage sein, OpenSSH zu installieren und zu konfigurieren. Dieses Lernziel beinhaltet grundlegende Installation und Problemlösung von OpenSSH sowie die Konfiguration von **sshd** für den automatischen Start beim Booten.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/hosts.allow`
- `/etc/hosts.deny`
- `/etc/nologin`
- `/etc/ssh/sshd_config`
- `/etc/ssh_known_hosts`
- `/etc/sshrc`
- **sshd**
- **ssh-keygen**

Die Techniken zum Einloggen in einen fremden Rechner mittels **telnet** und **rlogin** haben einen großen Nachteil, der in einem lokalen (vertrauenswürdigen) Netz keine große Rolle spielt, im Internet aber fatal sein kann. Bei beiden Protokollen werden alle Übertragungen unverschlüsselt realisiert. Das heißt, daß sowohl die Übertragung von Username und Passwort, als auch die der eigentlichen Sitzung von Lauschern mitgelesen werden kann.

Aus diesem Grund existiert eine sichere Lösung, die sogenannte *Secure Shell* oder kurz **ssh**. Dieses Protokoll besteht, genauso wie die beiden anderen oben genannten, aus einem Client-Server Paar. Auf dem Rechner, auf dem man sich sicher einloggen soll, läuft der Server (**sshd**), der User, der sich dort einloggen will, startet den Client (**ssh**).

Der Server **sshd** erlaubt nicht nur das sichere Einloggen, sondern ist auch noch ein Ersatz für **rsh**, das heißt, er kann auch einfach Befehle ausführen, und er ist auch in der Lage, Dateien sicher über das Netz zu transportieren, wenn der User statt **ssh** das Programm zum Kopieren

(**scp**) aufruft.

Der **sshd** Daemon wird in der Regel als Stand-Alone Dienst gestartet, also über ein Init-Script und nicht - wie seine Vorgänger - über inetd. Für jede eingehende Verbindung wird dann ein neuer Prozeß gestartet. Dieser neue Prozeß kümmert sich dann um die Schlüsselübergabe, die Verschlüsselung, Authentifizierung, Kommandoausführung und den Datentransfer.

Jeder Rechner hat einen rechnerspezifischen RSA-Schlüssel (normalerweise 1024 Bit breit) der ihn eindeutig identifiziert. Der Server erstellt zusätzlich in regelmäßigen Abständen einen speziellen Serverschlüssel (meist 768 Bit), der niemals abgespeichert, sondern immer nur im Speicher gehalten wird.

Sobald ein Client eine Anfrage startet, schickt der Server ihm seine beiden Public-Keys (Server und Host Schlüssel). Der Client überprüft dann, ob der Hostschlüssel des Servers mit seinem gespeicherten übereinstimmt und hat so die Gewißheit, daß es sich tatsächlich um den gewünschten Server handelt und nicht um eine Fälschung. Anschließend erzeugt der Client eine Zufallszahl und verschlüsselt sie mit den beiden Schlüsseln des Servers. Diese verschlüsselte Zahl wird an den Server geschickt. Nur dieser Server, der den passenden Secret-Key hat, kann die Zahl wieder entschlüsseln.

Die Zahl wird im weiteren Verlauf benützt, um die gesamte Kommunikation zwischen Client und Server zu verschlüsseln. Da diese Zahl schon verschlüsselt übertragen wurde, ist es auch einem Lauscher nicht möglich, sie zu verwenden um die Kommunikation abzuhören.

Optional kann **sshd** auch die Mechanismen der **r-Befehle** benutzen, die auf die Datei `~/.rhosts` basieren. Standardmäßig ist dieses Feature aber abgeschaltet, da es die Sicherheit kompomittieren kann.

Konfiguration des Daemons

Der **sshd** wird über die Datei `/etc/ssh/sshd_config` konfiguriert. Diese Datei erlaubt alle notwendigen Einstellungen für den Server. Eine genaue Beschreibung aller Direktiven dieser Datei ist in der Handbuchseite **sshd_config**(5) nachzulesen. Wichtige Einstellungen sind:

Port ***N***

Die Angabe auf welchem Port **sshd** laufen soll. Normalerweise ist es Port 22.

HostKey ***Dateiname***

Spezifiziert die Datei, in der der Host-Schlüssel des Servers abgelegt ist. Normalerweise ist das die Datei `/etc/ssh/ssh_host_key`. Diese Datei darf nicht für alle Welt lesbar sein, ansonsten verweigert sich **sshd** die Datei zu verwenden.

ServerKeyBits ***Zahl***

Hier wird die Breite des Server-Schlüssels angegeben. Normalerweise 768.

KeyRegenerationInterval ***Zeit***

Die Anzahl der Sekunden, nach denen der Server-Schlüssel neu erstellt werden soll. Standardmäßig sollte hier 3600 (eine Stunde) stehen. Steht hier eine 0, so wird der Schlüssel niemals neu erstellt.

PermitRootLogin ***yes*****|*****no***

Ist es erlaubt, daß sich der Systemverwalter über **ssh** einloggen kann?

IgnoreRhosts ***yes*****|*****no***

Sollen die Dateien `~/.rhosts` und `~/.shosts` ignoriert werden?

Sind diese Einstellungen erledigt, so kann der Daemon gestartet werden.

Weitere Sicherheitseinstellungen

Obwohl **sshd** nicht über inetd gestartet wird, arbeitet er trotzdem mit den TCP-Wrappern. Diese Technik wurde bereits an anderer Stelle detailiert dargestellt. Der Servername für die Steuerung von **sshd** in den Dateien `/etc/hosts.allow` und `/etc/hosts.deny` ist einfach **sshd**.

Existiert die Datei `/etc/nologin`, so verweigert **sshd** jeden Einlogvorgang außer dem von root (sofern root-Eingänge in der Konfigurationsdatei erlaubt waren). Der Inhalt der Datei wird dem Client übermittelt. Ein Systemverwalter kann also diese Datei anlegen und eine kurze Erklärung hineinschreiben, warum der Zugang gerade verboten ist.

Die Dateien `/etc/ssh/ssh_known_hosts` und `$HOME/.ssh/known_hosts` enthalten die Public-Keys aller bekannten Rechner. Die globale Datei wird vom Systemverwalter administriert, die userbezogene wird automatisch angelegt und erweitert, sobald der User sich von einem bisher unbekannten Rechner einloggt.

Jede Zeile dieser Dateien enthält die folgenden durch Leerzeichen getrennten Felder:

```
Hostnamen Bits Exponent Modulus Kommentar
```

Hostnamen ist eine durch Kommas getrennte Liste von Namen oder Mustern (* und ? dürfen benutzt werden). Jedes Muster wird mit dem vollständigen Hostnamen des Rechners verglichen, der sich anmelden will.

Bits, Exponent und Modulus werden direkt den Host-Schlüsseln entnommen, die in den Dateien `/etc/ssh/ssh_host_*_key.pub` gespeichert sind. Das optionale Kommentarfeld wird nicht ausgewertet.

Über diesen Mechanismus können Hosts als bekannt (known) definiert werden. Jeder neue Host, wird - sobald er sich eingeloggt hat, als bekannter Host eingetragen. So können Versuche enttarnt werden, wenn ein fremder Rechner vorgibt, ein anderer - bekannter - zu sein.

Der Login-Vorgang

Wenn ein User sich über **ssh** erfolgreich angemeldet hat, dann erledigt **sshd** folgende Schritte:

- Wenn es sich um ein terminalbasiertes Login (nicht um eine Kommandoausführung) handelt, dann gibt **sshd** aus, wann der letzte Login war und zeigt den Inhalt der Datei `/etc/motd` an. Anschließend wird die Zeit des aktuellen Logins abgespeichert (um beim nächsten wieder sagen zu können, wann der letzte war).
- Wenn die Datei `/etc/nologin` existiert, wird ihr Inhalt ausgegeben und die Verbindung getrennt (außer bei einem root-Login).
- **sshd** wechselt zur Benutzerkennung des Users, der sich eingeloggt hat.
- **sshd** erzeugt eine grundlegende Umgebung (Shellvariablen wie PATH)
- **sshd** ließt die Datei `$HOME/.ssh/environment` wenn sie existiert und nimmt die dort gemachten Einstellungen in die Umgebung auf.
- **sshd** wechselt in das Homeverzeichnis des Users.
- Wenn `$HOME/.ssh/rc` existiert, wird es abgearbeitet. Ansonsten wird die Datei `/etc/ssh/sshrc` gesucht und falls gefunden ausgeführt.
- Startet die Shell (oder das angegebene Kommando)

Schlüsselerzeugung mit ssh-keygen

Der Befehl **ssh-keygen** erzeugt und verwaltet die RSA und DSA Schlüssel für **ssh** Verbindungen. Normaluser können sich damit einen Schlüssel erzeugen, der Systemverwalter kann auch den Host-Schlüssel damit anlegen.

Diese Schlüssel sind nicht zwingend für den Gebrauch von **ssh** erforderlich, bestimmte Server verlangen ihn aber.

Es stehen zwei verschiedene Schlüsseltypen zur Verfügung, **rsa** und **dsa**. Mit dem Befehl

```
ssh-keygen -t Typ
```

wird ein neues Schlüsselpaar angelegt. Speicherort und Name werden beim Anlegen erfragt. Als *Typ* stehen **rsa** und **dsa** zur Auswahl.

Normalerweise wird dieses Programm hauptsächlich vom Init-Script aufgerufen, wenn das erste Mal der SSH-Server oder -Client betrieben wird, um einen Host-Schlüssel zu erzeugen.

Es ist aber auch möglich, über diese Schlüsselpaare eine Authentifizierung zu steuern, so daß ein User beim Einloggen via ssh kein Passwort mehr angeben muß.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.114.1

Ausführung von sicherheitsadministrativen Tätigkeiten

Beschreibung: Prüfungskandidaten sollten in der Lage sein, die Systemkonfiguration zu überprüfen, um die Sicherheit des Hosts in Übereinstimmung mit lokalen Sicherheitsrichtlinien sicherzustellen. Dieses Lernziel beinhaltet die Konfiguration von tcpwrappers, das Finden von Dateien mit gesetztem SUID/SGID-Bit, das Überprüfen von Softwarepaketen, das Setzen oder Ändern von Benutzerkennwörtern und des Ablaufs von Kennwörtern, das Aktualisieren von Programmdateien nach Empfehlung von CERT, Bugtraq und/oder Sicherheitswarnungen des Distributors. Ebenfalls enthalten ist ein grundsätzliches Wissen über **ipchains** und **iptables.**

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/proc/net/ip_fwchains`
- `/proc/net/ip_fwnames`
- `/proc/net/ip_masquerade`
- **find**
- **ipchains**
- **passwd**
- **socket**
- **iptables**

Bereits besprochene Techniken

Viele der hier verlangten Techniken sind an anderer Stelle dieses Study-Guides schon besprochen worden. Dazu zählen insbesondere

- Verwaltung der tcpwraper.

- Setzen und Ändern von Passwörtern und Auslauffristen von Passwörtern (Abschnitt 1.111.1)
- Überprüfung von Softwarepaketen (Abschnitt 1.102.6 der Vorbereitung auf die LPI101 Prüfung)

Diese Techniken werden hier nicht näher beschrieben, sondern als Wissen vorausgesetzt.

Auffinden von Programmen mit gesetztem SUID/SGID Bit

Programme, deren SUID-Bit oder SGID-Bit gesetzt sind, und die gleichzeitig als Eigentümer `root` haben, sind in dem Moment eine Sicherheitslücke, in dem es sich um Programme handelt, die einen Zugriff auf eine Shell ermöglichen. Viele Programme benützen dieses Feature um ganz gewöhnliche Aktionen vorzunehmen. Das klassische Beispiel ist das Programm **passwd**. Jeder normale User kann mit diesem Programm sein Passwort ändern. Die Datei, in der das verschlüsselte Passwort gespeichert wird (`/etc/shadow`) ist aber nicht für jeden User beschreibbar. So muß das Programm **passwd** das SUID-Bit gesetzt haben. Nur so kann es - unabhängig davon, wer es aufgerufen hat - die Datei verändern.

Um das ganze System nach Dateien zu durchsuchen, die dieses Bit (oder eines dieser Bits) gesetzt haben, benutzen wir den Befehl find. Mit der Suchoption `-perm` ist es möglich, Dateien zu suchen, die bestimmte Rechte gesetzt haben. Ein Zugriffsrecht von 4XXX bedeutet, daß das SUID-Bit gesetzt ist, ein 2XXX gibt an, daß das SGID-Bit gesetzt ist. Die Summe davon (6XXX) wären beide Bits gesetzt. Wenn dem angegebenen Zugriffsrecht ein Minuszeichen vorangeht, so ist gemeint, daß mindestens diese Rechte gesetzt sein müssen, es reicht also, nach 4111 oder gar 4000 zu suchen, wenn wir alle Programme mit SUID-Bit suchen.

```
find / -perm -4000 -user root
```

sucht alle Dateien des gesamten Systems, die das SUID-Bit gesetzt haben und dem User root gehören. Das sind zunächst einmal überraschend viele Programme, es wäre also ratsam, die Ausgabe dieses Befehls direkt nach der Installation in eine Datei umzuleiten und regelmäßig den Befehl erneut aufzurufen und mit dieser Datei zu vergleichen.

Das entsprechende zum SGID-Bit wäre der Befehl

```
find / -perm -2000 -group root
```

Die Angabe des Users wurde hier weggelassen, stattdessen suchen wir nach Dateien, die der Gruppe root zugehören. Ohne diese Angabe, würden alle Dateien mit gesetztem SGID-Bit angezeigt.

Grundlegendes zu Firewaltechniken

Der Begriff Firewall überspannt ein weites Feld an Missinterpretationen und falschen Vorstellungen. Es existieren verschiedenste Ansätze, die Sicherheit eines Netzes durch verschiedene Techniken zu verbessern, die alle diesen Namen für sich reklamieren. Die gesamte Thematik umfassend darzustellen würde den Rahmen dieses Kurses bei weitem sprengen, hier geht es nur um das grundlegende Verständnis von Paketfilterfirewalls unter Linux.

Die Firewalltechnik unter Linux basiert auf sogenannten Paketfilterregeln. Ein- und ausgehende Pakete können nach folgenden Kriterien durchgelassen oder abgewiesen werden:

- Sender IP-Adresse
- Empfänger IP-Adresse
- Sender-Port
- Empfänger-Port
- SYN und ACK-Flag
- Verwendete Netzwerkschnittstelle

Die Regeln werden mit einem kommandozeilenorientierten Programm formuliert und stehen dann im Kernelspeicher. Die eigentliche Firewall, also das Programm, das Pakete durchlässt oder abweist, ist kein separates Programm sondern der Kernel selbst.

Dieses Programm, zur Formulierung der Regeln hat zwischen der Kernel-Version 2.2 und 2.4 gewechselt. Das Standard-Programm bei Kerneln der 2.2 Reihe war **ipchains**. Es funktioniert unter gewissen Umständen auch noch bei neueren Kerneln. Seit der Version 2.4 existiert stattdessen das Programm **iptables**, das die selbe Aufgabe hat, sie aber etwas anders löst. Eine Grundkenntnis beider Programme wird für die LPI102 Prüfung vorausgesetzt.

Bevor jedoch die einzelnen Programme besprochen werden, noch etwas Theorie, die für das Verständnis beider Programme unabdingbar ist.

Prinzipielles zur TCP/IP Datenübertragung

Um eine Paketfilter-Firewall selbst aufzubauen sind fundierte Kenntnisse der TCP/IP Datenübertragung notwendig. Denn wir müssen ja verschiedenste Regeln formulieren, die sich auf die Datenpakete und deren Protokollheader beziehen. So soll hier nochmal kurz das wichtigste zum Thema Datenübertragung im IP-Netz dargestellt werden.

Der prinzipielle Ablauf einer Datenübertragung

Grundsätzlich müssen wir, angesichts des TCP/IP Schichtenmodells, unterscheiden, was eigentlich wie übertragen werden soll. Uns stehen

im Prinzip drei Protokolle zur Verfügung, die vorkommen können:

ICMP

Das Internet Control Message Protocol arbeitet noch auf der Vermittlungsschicht und dient der Übertragung von Kontrollnachrichten von Rechner zu Rechner. Es benützt selbst keine Ports, aber es werden verschiedene ICMP Nachrichtentypen unterschieden, die in Firewallregeln wie Ports behandelt werden. Ein Beispiel ist Ping, dessen ausgehendes Format vom Typ 8 ist, während das Antwortpaket (pong) den Typ 0 hat.

TCP

Das Transmission Control Protocol ist das verbindungsorientierte Protokoll auf der Transportschicht. Es benutzt Portnummern, um die jeweiligen Anwendungen auf der obersten Schicht zu adressieren. TCP benutzt einen dreiteiligen Handshake, dessen Ablauf für die Firewallregeln noch Bedeutung haben wird.

UDP

Das User Datagram Protocol ist das verbindungslose Protokoll auf der Transportschicht. Auch dieses Protokoll benutzt Portnummern, es hat jedoch keinerlei Handshake, also auch keine Flags, die wir bei der Formulierung der Regeln nutzen könnten.

Eine Datenübertragung besitzt - aus der Sicht der Firewall - zwei grundsätzlich unterschiedliche Abläufe. Entweder unser Rechner schickt eine Anfrage an einen Server eines anderen Rechners und bekommt Antwort von ihm, oder ein fremder Client schickt eine Anfrage an einen unserer Server und bekommt Antwort von uns.

Im Prinzip laufen alle Übertragungen nach dem gleichen Muster ab. Ein Client fängt mit einer Anfrage an einen Server an und der antwortet ihm. Dabei spielen die Portnummern eine große Rolle. Um einen Dienst auf einem anderen Rechner anzusprechen, muß der Client also die IP-Adresse des Servers und die Portnummer des gewünschten Dienstes kennen. Damit der Server ihm auch antworten kann, muß der Client auch seine eigene IP-Adresse angeben und eine Portnummer, auf der er selbst lauscht. Da der Client keine fest zugewiesene Portnummer besitzt, wählt er sich einfach eine beliebige Nummer aus dem Pool der unprivilegierten Portnummern (1024 bis 65535) aus.

Wir können also davon ausgehen, daß ein Paket mit einer Anfrage des Clients an den Server folgendes an Header-Information beinhaltet:

- Die IP-Adresse des Servers
- Die bekannte Portnummer des gewünschten Dienstes auf dem Server
- Die IP-Adresse des Clients
- Eine beliebige Portnummer zwischen 1023 und 65535, die sich der Client selbst gewählt hat.

Eine Ausnahme dieses Prinzips haben wir bei ICMP, dort hat der Sender selbst keinen Port bzw. keinen Pakettyp.

Handshake bei TCP

Bei der Verwendung von TCP als Protokoll gibt es eine weitere Besonderheit. Hier werden noch sogenannte Flags gesetzt, also Optionen,

die besondere Steuerfunktionen haben. Eine Verbindung von Client zu Server beginnt bei TCP immer folgendermaßen:

Der Client schickt ein Paket mit folgender Header-Information an den Server:

Empfänger IP:	**Empfänger Port:**	**Sender IP:**	**Sender Port:**	**Flags:**
IP-Adresse des Servers z.B. 123.45.67.89	*Portnummer des gewünschten Dienstes z.B. 80*	*Eigene IP-Adresse des Clients z.B. 111.22.33.44*	*Zufällige Portnummer z.B. 12345*	SYN

Der Client setzt also das SYN-Flag, gleichbedeutend mit der Nachfrage nach einem Verbindungsaufbau (Synchronisation). Der Server schickt jetzt ein Paket zurück, das folgende Information beinhaltet:

Empfänger IP:	**Empfänger Port:**	**Sender IP:**	**Sender Port:**	**Flags:**
IP-Adresse des Clients-111.22.33.44	*Portnummer des Clients-12345*	*IP-Adresse des Servers-123.45.67.89*	*Portnummer des Server-Dienstes-80*	SYN und ACK

Der Server setzt also einerseits das ACK (Acknowledge) Flag um zu zeigen, daß er den Verbindungsaufbau gewährt und setzt zusätzlich nochmal das SYN-Flag um seinerseits die Verbindung anzufordern. Das nächste Paket des Clients enthält jetzt nur noch das ACK-Flag:

Empfänger IP:	**Empfänger Port:**	**Sender IP:**	**Sender Port:**	**Flags:**
IP-Adresse des Servers-123.45.67.89	*Portnummer des Server-Dienstes-80*	*IP-Adresse des Clients-111.22.33.44*	*Portnummer des Clients-12345*	ACK

Von diesem Moment an gilt die Verbindung als etabliert und alle weiteren Pakete haben jetzt grundsätzlich das ACK-Flag und niemals mehr das SYN-Flag gesetzt. Aus dieser Handshake-Methode heraus ist anhand der Flags grundsätzlich festzustellen, ob ein Paket eine Nachfrage eines Clients an einen Server darstellt oder nicht. Zusammengefasst kann man sagen, daß ein Paket immer dann eine Client-Nachfrage beinhaltet, wenn das SYN-Flag gesetzt und das ACK-Flag nicht gesetzt ist.

Diese fünf (TCP) bzw. vier (UDP) Kriterien stehen uns also für die Firewall zur Verfügung. Anhand dieser Eigenschaften müssen wir entscheiden, ob wir ein Paket durchlassen oder nicht.

Architektur der Firewall-Regelketten

Eine Linux-Paketfilter Firewall besteht aus sogenannten Regelketten (chains). Standardmäßig existieren drei solcher Ketten:

- Die input-chain, die alle Pakete betrifft, die den Rechner verlassen
- Die output-chain, die alle Pakete betrifft, die der Rechner empfängt
- Die forward-chain, die alle Pakete betrifft, die der Rechner routen soll.

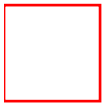
Alle drei Regelketten können unterschiedliche Grundeinstellungen haben, gemeinsam ist ihnen, daß sie eine Sammlung (Verkettung) von Regeln besitzen, die nacheinander abgearbeitet werden, solange bis eine Regel zutrifft oder das Ende der Kette erreicht wurde.

Es stehen drei Grundeinstellungen zur Verfügung:

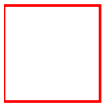
- ACCEPT - Ein Paket wird akzeptiert
- DENY - Ein Paket wird wortlos abgewiesen
- REJECT - Ein Paket wird mit Fehlermeldung abgewiesen.

Hat eine Kette die Grundeinstellung (Policy) ACCEPT, so bedeutet das, das grundsätzlich alles erlaubt ist, was nicht explizit verboten ist. Hat sie aber DENY oder REJECT als Grundeinstellung, so ist alles verboten, was nicht explizit erlaubt wurde.

Der zweite Fall ist mit Sicherheit die bessere Wahl, wenn es darum geht, ein System richtig abzusichern, auch wenn es zunächstmal mehr Arbeit bedeutet. Schematisch dargestellt könnte man die Grundeinstellung ACCEPT folgendermaßen darstellen:



Die DENY-Grundeinstellung hingegen sähe so aus:



Grundlagen von ipchains

Das Programm, mit dem die Linux-Paketfilterfirewall seine Regeln definiert, heisst bis zur Kernelversion 2.2 **ipchains**. Dieses Programm hat verschiedene Aufrufformen, die alle zu besprechen den Rahmen dieser Darstellung sprengen würde. Ich werde mich hier auf die notwendigen Formen beschränken, die den Umgang mit dem Programm klären.

ipchains formuliert Regeln, die dann direkt in den Speicher geschrieben werden. Das hat zur Folge, daß diese Regeln nach einem Neustart allesamt wieder verschwunden sind. Es empfielt sich also, die Regeln in Form eines Shellscripts zu formulieren.

Grundlegende Parameter von ipchains

Einer der folgenden Parameter muß immer der erste von ipchains sein. Diese Parameter schließen sich aus, es darf in einem Aufruf also immer nur einer der folgenden Parameter auftauchen:

-A *Regelkette ...*

Eine Regel wird ans Ende der angegebenen Regelkette (input, output oder forward) angehängt. Wird keine Regelkette angegeben, so

gilt die Regel für alle Regelketten.

-I *Regelkette ...*

Eine Regel wird an den Anfang der angegebenen Regelkette (input, output oder forward) eingefügt. Wird keine Regelkette angegeben, so gilt die Regel für alle Regelketten.

-F *Regelkette*

Alle bestehenden Regeln der angegebenen Kette werden gelöscht. Die Grundeinstellung (Policy) bleibt jedoch erhalten. Wird keine Regelkette angegeben, so werden alle Regeln aller Regelketten gelöscht.

-P *Regelkette Policy*

Die angegebene Policy (DENY, REJECT oder ACCEPT) wird zur Grundeinstellung der genannten Regelkette.

Um Regeln zu definieren werden wir also mit ipchains -A arbeiten. Der Angabe der Kette folgen dann die entsprechenden Regelformulierungen.

Darstellungsart von Ports und Adressen

Wir werden bei jeder zu formulierenden Regel sowohl mit Portnummern, als auch mit IP-Adressen arbeiten müssen. Dabei stehen uns folgende Darstellungsmöglichkeiten zur Verfügung:

IP-Adressen können mit einer Maske versehen werden, die die signifikannten Bits der Adresse angibt. Diese Maske wird als Bitmaske realisiert und einfach mit einem Slash (/) hinten an die Adresse angehängt. Die Bedeutung ist einfach, die Adressangabe

```
192.168.100.123/24
```

bedeutet, daß die ersten 24 Bit der Adressangabe mit der gefundenen Adresse übereinstimmen müssen, damit die Regel greift. In diesem Beispiel sind das also alle Adressen, die vorne 192.168.100 stehen haben. Eine Maske /32 bedeutet also, daß die Adresse exakt übereinstimmen muß, eine Maske /0 bedeutet, daß kein Bit übereinstimmen muß, also alle Adressen gemeint sind. Dafür ist auch die Abkürzung any/0 zulässig.

Ports werden als Nummern angegeben. Soll ein ganzer Bereich von gültigen Portnummern angegeben werden, so wird das in der Form

Startport:Endport

dargestellt.

Parameter zur Darstellung der Regeln

Nach der Angabe des Parameters -A (oder -I) und der gewünschten Kette folgen verschiedene Angaben, die alle erfüllt sein müssen, wenn die Regel zutreffen soll. Die wichtigsten Parameter sind:

`-i` *Interface*

Die Netzwerkschnittstelle für die die Regel gilt. Hier werden die symbolischen Schnittstellennamen wie `eth0, eth1, ppp0, ippp0,..` eingesetzt.

`-p` *Protokoll*

Hier wird das vom Paket verwendete Protokoll angegeben, also `icmp, tcp` oder `udp`.

`-y`

Das SYN-Flag einer TCP-Nachricht muß gesetzt, das ACK-Flag darf nicht gesetzt sein. Das Paket ist also das erste eines Verbindungsaufbaues und kommt vom Client.

`! -y`

Das ACK-Flag einer TCP-Nachricht muß gesetzt sein. Das heißt, das Paket ist entweder der zweite Teil des Verbindungsaufbaues oder, es ist ein Teil einer bestehenden Verbindung. Ist weder -y noch ! -y gesetzt, werden die TCP-Flags nicht überprüft.

`-s` *IP-Adresse* [*Portnummer*]

Absenderadresse (Source) des Paketes und optional die Absenderprtnummer.

`-d` *IP-Adresse* [*Portnummer*]

Empfängeradresse (Destination) des Paketes und optional der Empfängerport.

`-j` *Policy*

Policy dieser Regel. Gültige Policies sind ACCEPT, REJECT und DENY. In der forward-Chain ist auch die Policy MASQ für Masquerading zulässig.

Formulierung von Regeln

Für jede Art von Datenverbindung müssen wir mindestens zwei Regeln formulieren, eine für die input-chain und eine für die output-chain. In der Regel werden diese Formulierungen sich immer an den folgenden Aufbau halten:

```
ipchains -A Regelkette -i Interface -p Protokoll \
         -s Absenderadresse Absenderport \
         -d Empfängeradresse Empfängerport -j Policy
```

In manchen Fällen wird dem Protokoll noch die Angabe `! -y` folgen oder einzelne Ports werden weggelassen. Aber der grundsätzliche Aufbau hält sich immer an diese Struktur.

Aufbau eines eigenen Firewall-Scripts

Um ein komplettes Firewallscript zu schreiben fehlt uns hier der benötigte Rahmen, wir können aber die Regeln für die wichtigsten Dienste hier formulieren und alles weitere kann dann - sofern erwünscht - hinzugefügt werden.

Wir erstellen die Regeln am Besten in einem Shellscript, das wir dann jedesmal einfach aufrufen können, dann stellt sich gar nicht erst die Frage, wie einzelne Regeln gespeichert werden.

Das Script beginnt üblicherweise mit einem ganzen Satz von Variablendefinitionen, damit wir uns später leichter mit der Formulierung der Regeln tun:

```
#!/bin/bash

EXTERN_INTERFACE=eth0    # Unsere Netzschnittstelle
LOOP_INTERFACE=lo        # Das Loopback Interface
IPADDR=10.230.1.100      # Unsere IP-Adresse
ANYWHERE=any/0           # Jede Adresse im Netz
MYNET=10.230.1.0/24      # Unsere Netzadresse
UNPRIVPORTS=1024:65535   # Die unprivilegierten Ports
```

Als nächstes löschen wir alle bestehenden Regeln. Falls wir das Script einmal mehrfach hintereinander starten kommt es so nicht zu Doppelregeln. Dadurch, daß wir keine Regelkette angeben, werden alle bestehenden Regelketten gelöscht.

```
ipchains -F
```

Alle Ketten sind jetzt leer. Allerdings haben wir noch nicht die Grund-Policies gelöscht bzw. verändert. Sie bleiben auch nach dem Löschen vorhanden. Also stellen wir jetzt diese Policies ein, für jede Kette extra. Dazu stellt **ipchains** den Parameter -P (nicht verwechseln mit -p) zur Verfügung. Wir setzen alle drei Regelketten auf die Grundeinstellung DENY, also ist alles verboten, was nicht explizit erlaubt ist.

```
ipchains -P input    DENY
ipchains -P output   DENY
ipchains -P forward  DENY
```

Ab hier ist jetzt also alles verboten. Es existieren keinerlei Regeln mehr, die Grundeinstellung ist DENY. Wir können uns ab diesem Moment nicht einmal mehr selbst anpingen.

Damit das Loopback-Interface wieder funktioniert erlauben wir in den nächsten beiden Regeln grundsätzlich alles auf diesem Interface.

```
ipchains -A input  -i $LOOP_INTERFACE -j ACCEPT
ipchains -A output -i $LOOP_INTERFACE -j ACCEPT
```

Jetzt erlauben wir alle ICMP-Pakete auf der Netzschnittstelle. Wir könnten zwar hier verschiedene Einschränkungen machen, aber das ist langwierig und kompliziert. Viele Programme benötigen ICMP-Nachrichten um z.B. eine Nichtverfügbarkeit oder ähnliche Fehlermeldungen zu übertragen.

```
ipchains -A input -i $EXTERN_INTERFACE -p icmp -j ACCEPT
ipchains -A output -i $EXTERN_INTERFACE -p icmp -j ACCEPT
```

Jetzt funktioniert wenigstens ein Ping schonmal wieder. Allerdings nur mit numerischen IP-Adressen. Denn wir haben noch keinerlei Zugriff auf Nameserver. DNS funktioniert in den meisten Fällen über UDP auf Port 53. Wir können jetzt die erste vollständige Regel anwenden, indem wir den Zugriff auf Nameserver freischalten. Da uns über Port 53 wenig Gefahr droht geben wir ihn für alle frei, nicht nur für einen bestimmten Nameserver. Falls das nicht gewünscht wöre, müsste der Eintrag $ANYWHERE durch die IP-Adresse des Nameservers ersetzt werden.

```
ipchains -A output -i $EXTERN_INTERFACE -p udp\
         -s $IPADDR $UNPRIVPORTS \
         -d $ANYWHERE 53 -j ACCEPT

ipchains -A input -i $EXTERN_INTERFACE -p udp\
         -s $ANYWHERE 53 \
         -d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Jetzt können wir anfangen, die wichtigen Dienste einzeln zu erlauben. Zunächst einmal HTTP (TCP Port 80) und HTTP mit SSL (TCP Port 443) Wir erlauben nur Antworten eines Servers an uns.

```
ipchains -A output -i $EXTERN_INTERFACE -p tcp\
         -s $IPADDR $UNPRIVPORTS \
         -d $ANYWHERE 80 -j ACCEPT

ipchains -A input -i $EXTERN_INTERFACE -p tcp ! -y\
         -s $ANYWHERE 80 \
         -d $IPADDR $UNPRIVPORTS -j ACCEPT

ipchains -A output -i $EXTERN_INTERFACE -p tcp\
         -s $IPADDR $UNPRIVPORTS \
```

```
          -d $ANYWHERE 443 -j ACCEPT

ipchains -A input -i $EXTERN_INTERFACE -p tcp ! -y\
          -s $ANYWHERE 443 \
          -d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Sind wir selbst auch Webserver, dann müssen wir auch Zugriffe fremder Clients auf unseren Server zulassen und unsere Antwortpakete an diese Clients. Das könnte mit der Einschränkung passieren, daß wir nur Clients aus dem lokalen Netz ($MYNET) akzeptieren.

```
ipchains -A input -i $EXTERN_INTERFACE -p tcp\
          -s $MYNET $UNPRIVPORTS \
          -d $IPADDR 80 -j ACCEPT

ipchains -A output -i $EXTERN_INTERFACE -p tcp ! -y\
          -s $IPADDR 80 \
          -d $MYNET $UNPRIVPORTS -j ACCEPT
```

Schwieriger wird es mit FTP. Hier existieren zwei Portnummern (20 und 21), eine für den FTP-Befehlskanal und eine für den Datenkanal. Zudem existieren zwei Modi, der sogenannte aktive Modus und der passive Modus, der von den meisten Browsern verwendet wird. Der Unterschied liegt darin, daß der passive Modus beim Datenkanalaufbau auf beiden Ports (Sender und Empfänger) unprivilegierte Portnummern verwendet. Wir brauchen also 6 Regeln um als Client an einen FTP Server zu kommen:

- Anfrage des lokalen Clients beim fremden Server
- Antwort des fremden Servers
- Datenkanalaufbau durch den fremden Server (aktiv)
- Antwort auf Datenkanalaufbau durch den lokalen Client (aktiv)
- Datenkanalaufbau des Clients zum fremden Server (passiv)
- Antwort des fremden Servers auf Datenkanalaufbau (passiv)

In der oben genannten Reihenfolge sieht das also folgendermaßen aus:

```
ipchains -A output -i $EXTERN_INTERFACE -p tcp \
          -s $IPADDR $UNPRIVPORTS\
          -d $ANYWHERE 21 -j ACCEPT

ipchains -A input -i $EXTERN_INTERFACE -p tcp ! -y \
          -s $ANYWHERE 21\
```

```
          -d $IPADDR $UNPRIVPORTS -j ACCEPT

  ipchains -A input -i $EXTERN_INTERFACE -p tcp\
          -s $ANYWHERE 20\
          -d $IPADDR $UNPRIVPORTS -j ACCEPT

  ipchains -A output -i $EXTERN_INTERFACE -p tcp ! -y\
          -s $IPADDR $UNPRIVPORTS\
          -d $ANYWHERE 20 -j ACCEPT

  ipchains -A output -i $EXTERN_INTERFACE -p tcp\
          -s $IPADDR $UNPRIVPORTS\
          -d $ANYWHERE $UNPRIVPORTS -j ACCEPT

  ipchains -A input -i $EXTERN_INTERFACE -p tcp ! -y\
          -s $ANYWHERE  $UNPRIVPORTS\
          -d $IPADDR $UNPRIVPORTS -j ACCEPT
```

Das Gleiche in umgekehrter Reihenfolge wäre dann für einen eigenen FTP-Server nötig.

Dabei belassen wir es einmal, die Grundzüge des Aufbaus der Firewallscripts sind damit eigentlich geklärt. Jedes verwendete Protokoll benötigt also einen Regelsatz, der sowohl die eingehenden, als auch die ausgehenden Pakete berücksichtigt.

Unsere Firewall schützt bisher natürlich nur uns selbst, Aber die Mechanismen sind natürlich auch bei einer Firewall mit zwei Netzwerkkarten im Grunde die selben.

Grundlagen von iptables

Der grundsätzliche Aufbau von **iptables** ist sehr ähnlich dem von **ipchains**. Nur werden die Regelketten jetzt in Tabellen verwaltet, die jeweils einzelne oder mehrere Regelketten enthalten können. Das sollte zu einer größeren Übersichtlichkeit führen.

Drei Standard-Tabellen stehen zur Verfügung:

nat

Diese Tabelle ist zuständig für das IP-Masquerading oder Network Address Translation (NAT). Datenpakete, die durch einen Rechner geroutet werden, passieren diese Tabelle nur einmal. Die Regeln dieser Tabelle werden nur auf das erste Paket eines Datenstroms angewandt. Wenn dieses Paket passieren darf, so werden alle weiteren Pakete des gleichen Datenstroms automatisch

maskeriert, ohne daß die Regeln jedesmal überprüft werden. Die Tabelle enthält drei Regelketten (chains), `PREROUTING`, `OUTPUT` und `POSTROUTING`.

mangle

Diese Tabelle ist dazu gedacht, einzelne Elemente des Headers zu verändern. Beispiele wären die Veränderung der Felder `TTL`, `TOS` oder `MARK`. Sie hat zwei Regelketten, `PREROUTING` und `OUTPUT`. Die erstere wird verwendet, um Pakete zu verändern, in dem Moment, in dem sie die Firewall erreichen, die zweite verändert Pakete, die innerhalb der Firewall erzeugt werden, bevor die Routing-Entscheidungen anstehen.

filter

Diese Tabelle entspricht im Wesentlichen der Aufgabenstellung des alten **ipchains**. Hier werden die Firewall-Regeln erstellt. Die drei bekannten Regelketten `INPUT, FORWARD` und `OUTPUT` sind hier eingebaut.

Der Befehl **iptables** erhält die Information, welche Tabelle er bearbeiten soll durch den Parameter `-t` *`Tabelle`*. Wird diese Angabe weggelassen, so wird standardmäßig die Tabelle **filter** angenommen.

Die eigentliche Formulierung der Regeln selbst entspricht ziemlich genau der von **ipchains**.

Sicherheitswarnungen

In der Welt der Informatik gibt es zwei wesentliche Philosophien, wie mit Sicherheitslücken in Programmen umgegangen werden soll. Die eine geht davon aus, daß Sicherheitslücken am Besten geheim bleiben sollen und stillschweigend - etwa mit Hilfe eines Service-Packs - beseitigt werden sollten. Die typische Vertreterin dieser These ist die Firma Microsoft. Die zweite Philosophie geht genau andersherum vor. Ihre These ist, daß Sicherheitslücken sofort öffentlich gemacht werden müssen, um es den entsprechenden Systemverwaltern sofort zu ermöglichen, Maßnahmen zu ergreifen, die die Löcher schließen können. Linux-Software wird in der Regel mit dieser zweiten Philosophie arbeiten.

Es gibt zwei große Einrichtungen, die zentral solche Sicherheitslücken aufdecken und veröffentlichen: BUGTRAQ und CERT.

BUGTRAQ ist eine moderierte Mailingliste, die für die detailierte Diskussion und Veröffentlichung von Sicherheitsproblemen zuständig ist. Hier wird beschrieben, was Sicherheitslücken genau sind, wie herausgefunden wird, ob sie ein bestimmtes System betreffen und wie sie beseitigt werden können.

Um sich an dieser Mailingliste zu beteiligen - lesend oder schreibend - muß eine Mail an `LISTSERV@SECURITYFOCUS.COM` geschickt werden, die im Mailkörper - nicht in der Überschrift - die folgende Anweisung besitzt:

```
SUBSCRIBE BUGTRAQ Nachname, Vorname
```

Man erhält anschließend eine Bestätigungsmail, die man beantworten muß. Sobald diese Antwort auf die Bestätigungsmail eingegangen ist,

ist man Mitglied der Liste und erhält regelmäßig die Nachrichten über neu entdeckte Sicherheitslücken.

Auf der Webseite von www.securityfocus.com können die Archive der Mailingliste eingesehen werden.

Die zweite große Institution für die Veröffentlichung von Sicherheitslücken ist CERT. Auf der Webseite von www.cert.org werden regelmäßig neue Sicherheitslücken und Lösungen dazu veröffentlicht.

Weitere wichtige Adressen im Internet sind

- www.ciag.org
- www.sans.org

Zusätzlich veröffentlichen die Linux-Distributoren selbst Nachrichten über Sicherheitslücken und liefern etsprechende Bug-Fixes. Ein paar Beispiele wären:

- SuSE-Linux Updates und Bugfixes
- RedHat Security Alerts and Bugfixes
- Debian Sicherheits-Informationen

Diese Informationsquellen sollten regelmäßig konsultiert werden, um über neue Sicherheitsprobleme informiert zu sein und auftretende Schlupflöcher zu schließen.

Das Programm socket

Das Programm socket(1) erlaubt es, auf beliebigen Ports entweder Client- oder Serversockets aufzubauen, deren Datenströme mit der Standardein- und -ausgabe verbunden werden. Dieses Programm kann dazu benutzt werden, bestimmte Ports des eigenen Rechners zu überprüfen, ob von außen dort noch Zugriffe möglich sind.

Selbst wenn alle möglichen Dienste, die von außen zugreifbar sind, geschlossen wurden, so existiert mit **socket** eine Möglichkeit, Angriffe von innen heraus zuzulassen. Das Programm sollte also mit Vorsicht benutzt werden und sicherheitshalber nicht für alle User zugänglich sein.

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.114.2

Einrichten von Host-Security

Beschreibung: Prüfungskandidaten sollten über das Einrichten einer grundlegenden Host-Security Bescheid wissen. Diese Tätigkeiten enthalten die Konfiguration von syslog, Shadow-Paßwörter, das Einrichten eines Mail-Alias für die Mails von root und das Abdrehen der nicht verwendeten Netzwerkdienste.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- `/etc/inetd.conf` oder `/etc/init.d/*`
- `/etc/nologin`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/syslog.conf`

Auch für dieses Kapitel gilt, daß die meisten der hier geforderten Techniken bereits an anderer Stelle beschrieben wurden. Aus diesem Grund werden neben den Verweisen auf die bereits besprochenen Techniken nur noch einige Details geklärt.

Syslog Konfigurieren

Die Konfiguration des Syslog-Daemons wurde im Abschnitt 1.111.3 bereits ausführlich beschrieben. Sicherheitsrelevante Systemmeldungen können mit Hilfe dieser Technik in eine spezielle Datei geschrieben werden, die dann entsprechend beobachtet werden muß.

Typische Herkünfte von sicherheitsrelevanten Meldungen sind:

kern

Meldungen der Firewalls

auth und **authpriv**

Meldungen und vertrauliche Meldungen des Sicherheitsdienstes. Hier sind die logins verzeichnet.

Neben der Ausgabe in Dateien sei hier auch nochmal an die Möglichkeit erinnert, die Meldungen auch auf das Terminal bestimmter User (vorzugsweise des **root**-Users) zu schreiben.

Shadow-Passwörter

Alle modernen Linux-Distributionen arbeiten heute standardmäßig mit den Shadow-Passwörtern. Die genauen Beschreibungen der Formate der Shadow-Dateien sind in Abschnitt 1.111.1 nachlesbar.

Um ein System mit dem alten Passwortsystem in ein Shadow-Passwort-System zu konvertieren, existieren die Programme pwconv und grpconv. Umgekehrt können moderne Shadow-Systeme mit pwunconv und grpunconv wieder zurück in das alte Format verwandelt werden.

Logins zeitweise verbieten

Wenn der Systemverwalter für eine bestimmte Zeit verbieten will, daß sich User auf dem Rechner einloggen, so kann er einfach eine Datei `/etc/nologin` anlegen. Diese Datei kann einen kurzen Text enthalten, der erklärt, warum gerade kein Login möglich ist.

Sowohl das Programm **login**, als auch Netzwerkdienste wie **ssh, telnet, rlogin** oder **rsh** verweigern den Login, wenn diese Datei existiert. Statt dessen zeigen sie dem User den Inhalt der Datei `/etc/nologin` an.

Diese Beschränkung gilt nicht für den **root**-User.

Mail Alias von root

Normalerweise sollte der Systemverwalter nicht immer unter seinem **root**-login arbeiten, außer er arbeitet tatsächlich an der Verwaltung. Wenn er jedoch normale Arbeiten am Computer erledigt, sollte er mit einem Normallogin arbeiten, wie alle anderen auch. Das hat mehrere Gründe. Zum einen können fatale Fehler im Normalbetrieb nicht stattfinden, weil ein Normaluser diese Fehler nicht machen kann. Will ein User etwa ein Verzeichnis löschen und vertippt sich, so daß ein Leerzeichen in die Befehlszeile rutscht:

```
rm -r / foo
       ^
      /|\
       |
```

```
unbeabsichtigtes Leerzeichen
```

so wäre das der Befehl, das Wurzelverzeichnis und alle darunterliegenden Verzeichnisse zu löschen. Das kann aber nur der Systemverwalter (root). Arbeitet **root** ständig mit seinem Systemverwalter-Account, so gibt es keine Instanz, die ihn hindern würde, solche Fehler zu machen. Arbeitet er als Normaluser, so werden solche Fehler mangels Berechtigung nicht ausgeführt.

Damit aber **root** trotzdem seine Mails bekommt, die an **root@localhost** gesendet wurden, kann er einen Mailalias einrichten. Diese Technik wurde bereits im Abschnitt 1.113.2 besprochen. Der Eintrag in `/etc/aliases` müsste z.B. lauten:

```
root:  hans
```

Nach dieser Veränderung müßte der Befehl

```
newaliases
```

ausgeführt werden, damit **sendmail** den Alias benutzen kann.

Eine andere Möglichkeit wäre die Weiterleitung der Mails von root an einen anderen User über die Benutzung der Datei `~/.forward` im Heimatverzeichnis von **root**

Nicht benötigte Netzwerkdienste abschalten

Netzwerkdienste können unter Linux grundsätzlich auf zwei verschiedene Arten gestartet werden. Entweder sie sind grundsätzlich schon im Arbeitsspeicher geladen und warten auf Aufträge, die sie abarbeiten sollen, oder sie werden - jeweils wenn sie gebraucht werden - vom **inetd** aufgerufen. Beide Techniken haben ihre Vor- und Nachteile die Fall für Fall abgewogen werden sollten, um zu entscheiden, auf welche Weise ein bestimmter Dienst gestartet werden soll. Meist ist die Voreinstellung der Distributionen ein guter Anhaltspunkt, um zu entscheiden, ob ein Dienst als Stand-Alone oder inetdbasierter Dienst laufen soll.

Die Voreinstellungen der verschiedenen Distributionen ermöglichen oft Zugriffe, die in der Praxis nicht gewünscht sind. Um solche Dienste abzuschalten, muß entsprechend erstmal klar sein, wie er gestartet wird. Die beiden folgenden Abschnitte gehen die beiden möglichen Methoden durch und zeigen Möglichkeiten der Abschaltung.

Inetd-basierte Dienste abschalten

Wenn ein Dienst über den **inetd** (siehe auch Abschnitt 1.113.1) gestartet wird, so ist er in der Datei `/etc/inetd.conf` eingetragen. Wurde stattdessen der **xinetd** benutzt, liegen die Einstellungen in `/etc/xinetd.conf`. Einträge in diesen Dateien können einfach auskommentiert werden, indem ihnen ein Kommentarzeichen am Zeilenanfang vorangestellt wird.

Das alleinige Auskommentieren dieser Zeile reicht aber nicht aus, denn der **inetd** ließt diese Datei nur beim Start. Um einem laufenden

inetd diese Veränderungen bekannt zu geben, wird ihm ein **HUP**-Signal geschickt.

In vielen Fällen ist es überhaupt nicht erwünscht, daß der **inetd** läuft. Sollte das der Fall sein, so kann dafür gesorgt werden, daß dieser Dienst selbst gar nicht gestartet wird. Der **inetd** selbst ist ein Stand-Alone-Dienst, dessen Abschaltung im nächsten Abschnitt beschrieben wird.

Stand-Alone Dienste abschalten

Stand-Alone Dienste werden in den modernen Linux-Distributionen alle auf eine gemeinsame Art und Weise gestartet. Im Verzeichnis `/etc/init.d` liegen Shellscripts, die die einzelnen Dienste starten und stoppen. Diese sogenannten Init-Scripts verstehen jeweils mindestens die Parameter **start** und **stop**. Häufig gibt es noch weitere Parameter wie **status** (gibt eine Ausgabe aus, ob der entsprechende Dienst läuft oder nicht), **reload** (zwingt den Dienst seine Konfigurationsdatei neu einzulesen) oder **restart** (Kombination aus **stop** und anschließendem **start**).

Jedesmal, wenn ein neuer Dienst installiert wird, installiert er ein entsprechendes Init-Script in das Verzeichnis `/etc/init.d`. Jeder Dienst kann jetzt mit Hilfe dieser Scripts gestartet werden. Installieren wir den Dienst **foo**, dann existiert also in diesem Verzeichnis ein Script, wahrscheinlich ebenfalls mit dem Namen `/etc/init.d/foo`. Um den Dienst von Hand zu starten, wird dieses Script mit dem Parameter **start** aufgerufen:

```
/etc/init.d/foo start
```

entsprechend kann ein laufender Dienst mit dem Script wieder heruntergefahren werden:

```
/etc/init.d/foo stop
```

Die alleinige Existenz dieser Scripte legt aber noch nicht fest, daß diese Dienste auch automatisch gestartet werden. Für diese Einstellungen sind die sogenannten Runlevel-Verzeichnisse gedacht.

Im Verzeichnis `/etc` (manchmal auch in `/etc/init.d` liegen für jeden Runlevel ein weiteres Verzeichnis. Diese Runlevelverzeichnisse haben immer die Namen

`rc`*Runlevelnummer*`.d`

also etwa `rc0.d, rc1.d, rc2.d` oder `rcS.d` (Single User Mode). Innerhalb dieser Verzeichnisse finden sich symbolische Links auf die Init-Scripts, die in dem jeweiligen Runlevel ausgeführt werden sollen. Der Namen dieser Links beginnt entweder mit einem `S` (Start) oder mit einem `K` (Kill). Dem folgt eine zweistellige Nummer, die angibt, in welcher Reihenfolge die Scripts ausgeführt werden sollen. Anschließend folgt der Name des Scripts. Die Nummer ist kein absoluter Wert, sondern nur eine Hilfe für die alphabetische Sortierung der Scriptnamen für die Ausführung.

Um unser **foo**-Script in `/etc/init.d` automatisch im Runlevel 5 zu starten, genügt also die Erstellung eines symbolischen Links im

Verzeichnis `/etc/rc5.d` in der Form:

```
lrwxrwxrwx 1 root root   13  5. Sep 17:22 S20foo -> ../init.d/foo
```

Die 20 bedeutet nicht, daß dieser Dienst an 20. Stelle gestartet wird. Alle Links in diesem Verzeichnis, die mit einem `S` beginnen, werden beim Eintritt in diesen Runlevel alphabetisch sortiert ausgeführt. Die Nummer dient also dazu, die Reihenfolge anzugeben, nicht die absolute Position. Es ist ohne Problem möglich, daß mehrere Links mit der selben Nummer arbeiten, sie werden dann hintereinander (`S20bar` vor `S20foo`) aufgerufen. Wird ein Script durch einen Link aufgerufen, dessen Namen mit einem `S` beginnt, so wird das entsprechende Script mit dem Parameter **start** aufgerufen

Analog dazu werden die Scripts, deren Namen mit `K` beginnen, beim Verlassen des Runlevels ausgeführt, wieder in alphabetischer Reihenfolge. Nur diesmal wird das Script, auf das sie verweisen mit dem Parameter **stop** aufgerufen.

Um also Dienste grundsätzlich abzuschalten und zu verhindern, daß diese Dienste bei einem Neustart wieder automatisch geladen werden, muß im entsprechenden Runlevelverzeichnis der entsprechende Link einfach gelöscht werden.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

1.114.3

Einrichten von Sicherheit auf Benutzerebene

Beschreibung: Prüfungskandidaten sollten in der Lage sein, Sicherheit auf Benutzerebene zu implementieren. Die Tätigkeiten beinhalten das Verwalten von Beschränkungen auf Benutzerlogins, Prozesse und Speichernutzung.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **quota**
- **usermod**

Die Verwaltung von Einschränkungen einzelner Useraccounts ist ein vielfältiger Bereich, der in einzelnen Fällen schon in anderen Abschnitten beschrieben wurde. Die Haupt-Möglichkeiten der Einschränkungen liegen in einer durchdachten Organisation der Dateien und Programme hinsichtlich ihrer Gruppenzugehörigkeit. Sind bestimmte Programme nur von Mitgliedern einer bestimmten Gruppe ausführbar, so kann z.B. die Einstellung, welcher User ein solches Programm ausführen darf einfach über die Gruppenmitgliedschaft des Users verwaltet werden. Diese Techniken sind im Abschnit 1.111.1 bereits beschrieben worden.

Auch die Möglichkeit, die Gültigkeit eines Accounts und andere einen User betreffende Einstellungen mit usermod zu verändern wurde dort bereits ausführlich besprochen.

Die Einschränkungen des Festplattenplatzes, den einzelne User zur Verfügung haben mittels diskquotas wurde bereits bei der Vorbereitung auf die Prüfung LPI101 im Abschnitt 1.104.4 ausführlich besprochen und wird daher hier nicht weiter behandelt.

Einschränkungen mit ulimit

Eine Sache, die noch keine Beschreibung erfahren hat, für userbezogene Einstellungen aber sehr wichtig ist, ist der Umgang mit dem Shellbefehl **ulimit**. Es handelt sich hier um ein eingebautes (internes) Shellkommando, das der Shell selbst Einschränkungen hinsichtlich der von ihr benötigten Ressourcen macht. In der Regel werden diese Einstellungen in systemweiten Startdateien der Shell (wie etwa `/etc/profile`) gemacht. Einstellungen in usereigenen Startdateien sind in der Regel sinnlos, weil jeder User diese Einschränkungen

selbst wieder aufheben könnte.

ulimit kennt verschiedene Möglichkeiten für Beschränkungen von Ressourcen, die für die Shell und alle Prozesse, die von ihr gestartet werden gelten. Die Syntax ist:

ulimit [-SHacdflmnpstuv [*Limit*]]

Die Optionen `-H` und `-S` spezifizieren jeweils den Hard- bzw. Softlimit einer Ressource. Ein Hardlimit kann nicht mehr erhöht werden, sobald er einmal gesetzt ist. Ein Softlimit kann bis zum Wert des Hardlimits erhöht werden. Werden beide Angaben weggelassen, so werden sowohl Hard-, als auch Softlimit gesetzt.

Die Angabe des *Limit* kann entweder eine Zahl sein, in der für die angegebene Ressource gültigen Einheit, oder einer der Spezialwerte **hard, soft** oder **unlimited**. Diese Spezialwerte bezeichnen die für diese Ressource eingestellten Hard- und Softlimits bzw. keinen Limit.

Ist kein *Limit* angegeben, so wird der gegenwärtige Wert des Softlimits der entsprechenden Ressource ausgegeben. Ist zusätzlich ein `-H` angegeben, so wird stattdessen der Hardlimit angezeigt.

Die einzelnen Ressourcen werden über Parameter eingestellt und haben die folgenden Bedeutungen:

-a

Alle aktuellen Limits werden dargestellt

-c

Stellt die maximale Größe von Core-Dateien ein. Das sind Dateien, die erstellt werden, wenn ein Programm abstürzt. Diese Dateien enthalten dann den Daten- und Programmbereich des abgestürzten Programms für die Fehlersuche mit einem Debugger. Ein Wert von 0 schaltet Core-Files grundsätzlich ab.

-d

Die maximale Größe des Datensegments eines Prozesses.

-f

Die maximale Größe von Dateien, die die Shell anlegt.

-l

Die maximale Speichergröße, die gesperrt werden darf

-m

Die maximale Größe von residentem Speicher

-n

Die maximale Anzahl gleichzeitig geöffneter Dateien

-p

Die Größe von Pipes in 512 Byte Blöcken

-s

Die maximale Stack-Größe

-t

Der maximale Wert von CPU-Zeit in Sekunden

-u

Die maximale Anzahl von Prozessen für einen einzelnen User

-v

Die maximale Größe von virtuellem Speicher, der für die Shell verfügbar ist.

Ist *Limit* angegeben, so bezeichnet er den neuen Wert für die angegebene Ressource (außer bei `-a`, das nur Ausgaben macht). Wird keine Ressource angegeben, so wird `-f` angenommen. Die Werte beziehen sich in der Regel auf 1024 Byte Blöcke, außer bei der Angabe von `-t` (Sekunden), `-p` (512-Byte Blöcke) und `-n` und `-u`, die keinen Multiplikator haben sondern genau so gewertet werden, wie angegeben.

Ein typischer Eintrag in `/etc/profile` wäre z.B.

```
ulimit -Sc 0        # Der Soft-Limit für Core-Dateien wird
                    # auf 0 gesetzt
                    # Es werden also keine Corefiles angelegt.
ulimit -d 15000     # Hard- und Softlimit für die Datengröße
                    # eines Programmes werden auf 15 MByte
                    # gesetzt (15000*1024)
ulimit -s 15000     # Hard- und Softlimit für die Stack-Größe
                    # eines Programmes werden auf 15 MByte
                    # gesetzt.

# Die folgende Konstruktion gilt nur für den User hans.
# Er darf nicht mehr als 10 Prozesse gleichzeitig laufen haben.
if [ $LOGNAME = "hans" ]
then
  ulimit -u 10
fi
```

Mit diesen Einstellungen ist es möglich, die einzelnen User in ihrem Ressourcenverbrauch einzuschränken. Diese Ressourcen beziehen sich

aber nur auf eingeloggte User, nicht auf Netzwerkdienste. Genau besehen sind es ja Einstellungen der Shell. Also greifen diese Einstellungen nur da, wo die Shell aktiv ist.

[Zurück zur LPIC 102 Seite] [Zurück zur Startseite] [Zurück zur LPI Seite] [Guestbook] [Kontakt] [Impressum]

Diese und die darunterliegenden Seiten wurden erstellt von F. Kalhammer

---

Name

chage - Ändert die Informationen über die Gültigkeit eines Userpassworts

Syntax

chage

[**-m** *mindays*] [**-M** *maxdays*]
[**-d** *lastday*] [**-I** *inactive*]
[**-E** *Ablaufdatum*] [**-W** *warndays*] *Username*

chage

-l *Username*

Beschreibung

chage ändert die Anzahl der Tage zwischen Passwort-Wechseln und dem Datum der letzten Passwort-Änderung. Diese Information wird vom System benutzt, um zu entscheiden, wann ein User sein Passwort ändern muß. Das **chage** Kommando ist auf den Systemverwalter beschränkt, mit Ausnahme der -l Option, die von einem Normaluser benutzt werden kann, um herauszufinden, wann sein Passwort oder Account ausläuft.

Mit der **-m** Option kann die minimale Anzahl von Tagen eingestellt werden, die zwischen zwei Passwortwechseln vergehen müssen. Ein Wert von 0 stellt ein, dass der User sein Passwort immer ändern kann.

Mit der **-M** Option wird die maximale Anzahl von Tagen eingestellt, die ein Passwort gültig ist. Wenn *maxdays* plus *lastday*, weniger als der aktuelle Tag sind, wird der User aufgefordert, sein Passwort zu ändern, bevor er seinen Account nutzen kann. Mit der -W Option kann die entsprechende Warnzeit eingestellt werden.

Mit der **-d** Option wird die Anzahl der Tage eingestellt, die seit dem 1. Januar 1970 vergangen sind, bis zu dem Tag, an dem das Passwort das letzte Mal verändert wurde. Dieses Datum kann auch im Format JJJJ-MM-TT

eingegeben werden.

Die **-E** Option wird benutzt, um ein Datum anzugeben, an dem der Account eines Users nicht mehr gültig sein wird. Die Angabe *expiredate* enthält entweder die Anzahl der Tage seit dem 1. Januar 1970 bis zu dem gewünschten Datum, oder wiederum ein Datum im Format JJJ-MM-TT. Ein User, dessen Account auf diese Weise abgelaufen ist, muß den Systemverwalter kontaktieren, bevor er wieder Zugriff auf den Rechner bekommt.

Die **-I** Option wird benutzt, um eine Pufferzeit anzugeben, die gültig ist, nachdem ein Passwort ausgelaufen ist, bis zu dem Tag, an dem der Account tatsächlich gesperrt wird. Die Angabe erfolgt in Tagen, ein Wert von 0 schaltet dieses Feature ab.

Die **-W** Option stellt die Anzahl der Tage ein, an denen eine Warnung an den User ausgesprochen wird, bevor das Passwort ausläuft. Die Angabe von *warndays* ist die Anzahl von Tagen vor dem Auslaufen des Passwortes, ab denen ein User gewarnt wird, daß sein Passwort ausläuft.

Wird keine der genannten Optionen benutzt, so fällt **chage** in einen interaktiven Modus, der alle genannten Informationen abfragt. Ein neuer Wert kann eingegeben werden, um ein Feld zu ändern, ein Return behält den jeweils gültigen Wert. Der aktuelle Wert wird in eckigen Klammern dabei angezeigt.

Beachten Sie, daß dieses Feature nicht zur Verfügung steht, wenn die Shadow-Passwörter abgeschaltet sind.

Dateien

/etc/passwd - Useraccount Informationen
/etc/shadow - Sichere Useraccount Informationen

Autor

Julianne Frances Haugh (jfh@austin.ibm.com)

TCP-Wrapper

Ursprünglich alleine für die Verwendung mit **inetd** gedacht, hat sich das Prinzip des TCP-Wrappers inzwischen auch für Stand-Alone-Dienste wie **ssh** durchgesetzt. Worum geht es?

Wenn ein bestimmter Dienst aufgerufen wird, so kann er - anstatt direkt aufgerufen zu werden - durch das Programm **tcpd** aufgerufen werden. Dazu wird einfach statt dem zu startenden Dienst der **tcpd** aufgerufen und ihm wird der Name des zu startenden Dienstes als Parameter mitgegeben. Das Programm **tcpd** überprüft jetzt anhand von Einträgen in den Dateien

- `/etc/hosts.allow`
- `/etc/hosts.deny`

ob der Dienst von dem entsprechenden Host in Anspruch genommen werden darf. Analog überprüfen auch Stand-Alone-Dienste diese beiden Dateien.

Die Überprüfung erfolgt auf eine etwas eigenwillige Weise:

- Existiert ein passender Eintrag in der Datei /etc/hosts.allow, so wird Zugriff gegeben. Wenn nicht, dann
- Existiert ein passender Eintrag in der Datei /etc/hosts.deny, so wird kein Zugriff gegeben. Wenn nicht, dann
- wird Zugriff gegeben.

Die klassische Form der TCP-Wrapper

Das Format beider Dateien ist in der Handbuchseite hosts_access(5) genauestens dargestellt, es handelt sich um Zeilen des Formats:

```
Serverliste : Clientliste [: Shellkommando]
```

- Serverliste ist eine Liste von Servern (Programmnamen), oder Wildcards. Server werden mit ihrem Programmnamen - nicht über ihr Protokoll - angegeben, also z.B.: **in.telnetd**
- Clientliste ist eine Liste von einem oder mehreren Hostnamen, IP-Adressen, Suchmustern oder Wildcards,
- Shellkommando ist ein Kommando, das die lokale Shell ausführt, wenn der Dienst angefordert wurde und der Eintrag ausschlaggebend für seine Ausführung oder Nichtausführung war. Damit kann etwa eine Warnmeldung an root gegeben werden, wenn jemand versucht auf einen verbotenen Service zuzugreifen.

Als Suchmuster kommen zwei einfache Verfahren in Frage:

1. Beginnt ein Suchmuster mit einem Punkt (z.B. .foo.bar), so gelten alle Hostnamen als Treffer, deren Ende mit dem Muster übereinstimmt also etwa hal.foo.bar
2. Endet ein Suchmuster mit einem Punkt (z.B. 192.168.200.), so gelten alle Namen und Adressen als Treffer, deren erster Teil mit dem Muster übereinstimmt.

Als Wildcards können unter anderem folgende Werte verwendet werden:

ALL

Die universelle Wildcard, alles gilt...

LOCAL

Alle Hostnamen ohne Punkt (also lokale Namen) gelten.

UNKNOWN

Passt auf alle Usernamen, die unbekannt sind und alle Hosts, deren Namen oder Adressen nicht bekannt sind. Wird gerne in /etc/hosts.deny verwendet.

KNOWN

Passt auf alle Hosts und User, die bekannt sind

EXEPT

Ist ein Operator, um zwei Listen auszuschließen (Liste1 EXEPT Liste2) also etwa ALL EXEPT UNKNOWN

Das Shellkommando sollte grundsätzlich mit einem & beendet werden, weil sonst auf seine vollständige Abarbeitung gewartet wird, bevor ein Service evt. gestartet wird. Je nach Kommando kann das natürlich dauern...

Als zusätzliche Platzhalter in Shellkommandos können folgende Konstrukte verwendet werden:

%a

Die IP-Adresse des anfordernden Hosts

%A

Die IP-Adresse des aufgerufenen Servers

%c

Clientinformationen - User@Host oder User@IP-Adresse oder nur IP-Adresse des Anrufers, je nach dem, wieviel Informationen zur Verfügung stehen.

%d

Der Name des Daemon-Prozesses, der angefordert wurde.

%h

Name (oder falls nicht vorhanden IP-Adresse) des Clients

%H

Name (oder falls nicht vorhanden IP-Adresse) des Servers

%p

Die ProzessID des Daemon-Prozesses

%s

Serverinformationen - Daemon@Hostname oder Daemon@Adresse oder nur Daemon, je nach dem, wieviel Informationen zur Verfügung stehen.

%u

Der Username des Anrufers oder "unknown"

%%

Das %-Zeichen

Die modernere Form der Wrapper

Moderne Systeme arbeiten heute zwar immer noch mit dem dargestellten Prinzip, bieten aber auch die Möglichkeit, alle Einstellungen in nur einer der beiden Dateien vorzunehmen. Statt der Handbuchseite `hosts_access(5)` wird diese Methode in `hosts_options(5)` beschrieben. Der Aufbau der beiden Dateien sieht jetzt folgendermaßen aus:

```
  Serverliste : Clientliste [: Option ] [: Option ...]
```

Statt einem Shellkommando können also Optionen angegeben werden. Die wichtigsten Optionen sind:

ALLOW

Erlaubt den angegebenen Dienst für die angegebenen Clients.

DENY

Verbietet den angegebenen Dienst für die angegebenen Clients.

spawn *Shellkommando*

Führt das angegebene Shellkommando aus. Wie in der klassischen Form werden die oben beschriebenen Ersetzungen vorgenommen.

twist *Shellkommando*

Führt das angegebene Shellkommando aus und schickt seine Ausgaben an den Client, anstatt den gewünschten Dienst zu starten. Wie in der klassischen Form werden die oben beschriebenen Ersetzungen vorgenommen.

user *Username*[.*Gruppe*]

Startet den angegebenen Dienst unter der angegebenen User (und optional Gruppen) ID.

Der Vorteil dieser Methode liegt darin, daß alle Einstellungen in einer Datei vorgenommen werden können. Da die Optionen ALLOW und DENY zur Verfügung stehen, können in `/etc/hosts.allow` auch Verbote ausgesprochen werden und umgekehrt.

Aber Achtung: Es gilt immer noch die oben angegebene Reihenfolge. Es nützt also nichts, einem Host etwas zu erlauben, ohne allen anderen es zu verbieten. Die Einträge werden der Reihe nach gelesen und der erste passende wird benutzt. Um also nur dem Rechner `marvin.foo.bar` zu erlauben, den FTP-Daemon zu benutzen, müssten wir schreiben:

```
in.ftpd : marvin.foo.bar : ALLOW
in.ftpd : ALL : DENY
```

Wenn der Rechner `marvin.foo.bar` jetzt den Dienst anfordert, greift die erste Zeile und der Zugriff wird gewährt. Versucht aber ein anderer Rechner den Zugriff, so greift die erste Zeile nicht, dafür aber die zweite - der Zugriff wird verwehrt.

Name

socket - erzeugt TCP-Sockets und verbindet sie mit STDIN/OUT

Syntax

socket [**-cfqrvw**] [**-p** *Kommando*] *Host Port*
socket [**-cfqrvw**] [**-p** *Kommando*] **-s** [**-l**] *Port*

Beschreibung

socket erzeugt einen Internet Domain TCP-Socket und verbindet ihn mit der Standard Ein- und Ausgabe (stdin/stdout). Der angegebene *Host* kann entweder über eine IP-Adresse oder einen Domain-Namen angegeben werden. Der letztere Fall setzt ein funktionierendes DNS-System voraus, so daß der angegebene Name über einen Aufruf von **gethostbyname**(3) aufgelöst werden kann. Die Angabe des *Port* kann entweder über eine Portnummer oder einen Portname erfolgen. Der angegebene Name muß mit dem Systemaufruf **getservbyname**(3) auflösbar sein (also etwa in `/etc/services` aufgeführt sein.

Optionen

-b (background)

Das Programm wird in den Hintergrung gebracht, löst sich vom kontrollierenden Terminal, schließt die Dateideskriptoren, die mit dem Terminal verbunden waren und wechselt das aktuelle Arbeitsverzeichnis zum Wurzelverzeichnis.

-c (crlf)

Zeilenvorschubszeichen (LineFeed - LF) werden in eine Wagenrücklauf (Carriage Return - CR)/Zeilenvorschub Kombination (CRLF) konvertiert. CRLF Kombinationen, die vom Socket gelesen

werden, werden in ein einfaches Linefeed konvertiert.

-f (fork)

Wenn eine Serververbindung aufgebaut wurde, so wird ein separater Prozeß erzeugt (forked) um die Verbindung im Hintergrund auszuführen.

-l (loop)

(nur in Zusammenhang mit **-s** gültig). Nachdem eine Verbindung geschlossen wurde, werden weitere Verbindungen akzeptiert.

-p (program)

Das spezifizierte Kommando wird für jede Verbindung ausgeführt. Seine Standard-Kanäle sind mit dem Socket verbunden. Das angegebene Kommando kann jedes beliebige Shellkommando sein.

-q (quit)

Die Verbindung wird abgebrochen, wenn ein Dateiendezeichen empfangen wurde.

-r (read only)

Keine Daten werden von der Eingabe gelesen und auf den Socket geschrieben. Nur Daten, die vom Socket kommen, werden gelesen.

-s (server)

Ein Server-Socket wird aufgebaut. In diesem Fall muß kein Hostname angegeben werden.

-v (verbose)

Nachrichten über Verbindungen werden auf die Standard-Fehlerausgabe (stderr) geschrieben.

-w (write only)

Keine Daten werden vom Socket gelesen und auf die Standard-Ausgabe geschrieben.

-version

Die Programmversion von **socket** wird ausgegeben und das Programm danach beendet. Das muß das erste Argument sein.

Beispiele

Das Kommando

```
socket -v coma.cs.tu-berlin.de nntp
```

verbindet die Standardein- und -ausgabe mit dem NNTP-Port (119) des Rechners coma.cs.tu-berlin.de.

Das Kommando

```
socket -sl 3425
```

erzeugt einen Server-Socket auf Port 3425 auf dem lokalen Rechner und wartet auf eingehende Verbindungen. Nachdem eine Verbindung abgebrochen wurde, werden weitere Verbindungen akzeptiert.

Autor

Juergen Nickelsen <nickel@cs.tu-berlin.de>