

# Linux

# Grundlagen



# 1 Informationsquellen und Hilfen zu Linux

- **Installierte Online-Hilfen, Man-Pages und Dokumentationen**

- **Linux-Bücher von EDV-Verlagen, z.B.:**

→ Kofler, Michael: *Linux - Das umfassende Handbuch*, Rheinwerk Verlag, 50 EUR

→ Deimeke, Kania, u.a.: *Linux-Server: Das Administrationshandbuch*, Rheinwerk Verlag, 60 EUR

- **Online-Bücher**

*Linux - Das umfassende Handbuch (2012)*: [openbook.rheinwerk-verlag.de/linux](http://openbook.rheinwerk-verlag.de/linux)

*Ubuntu GNU/Linux*: [openbook.rheinwerk-verlag.de/ubuntu](http://openbook.rheinwerk-verlag.de/ubuntu)

Deutschsprachige Online-Bücher und Probekapitel: [www.oreilly.de/openbooks.php](http://www.oreilly.de/openbooks.php)

z.B.: - *Linux Wegweiser zur Installation & Konfiguration, 3 Auflage*:

[www.oreilly.de/german/freebooks/rlinux3ger/linux\\_wegIVZ.html](http://www.oreilly.de/german/freebooks/rlinux3ger/linux_wegIVZ.html)

- *Linux Wegweiser für Netzwerker, 3.Auflage*:

[www.oreilly.de/german/freebooks/linag2](http://www.oreilly.de/german/freebooks/linag2)

Diverse englische Linux Bücher: [www.oreilly.com/openbook](http://www.oreilly.com/openbook)

- **Zeitschriften**

Das Deutsche Linux-Magazin: [www.linux-magazin.de](http://www.linux-magazin.de)

Eine Linux-Zeitschrift für Anwender: [www.linux-user.de](http://www.linux-user.de)

Eine Linux-Zeitschrift für Anfänger: [www.easylinux.de](http://www.easylinux.de)

Eines der bekanntesten Linux-Magazine weltweit: [www.linuxjournal.com](http://www.linuxjournal.com)

Linux-Magazine mit kompletten Online-Archiv: [www.linux-mag.com](http://www.linux-mag.com)

- **LDP - The Linux Documentation Project** mit Anleitungen, Koch-Rezepten (HOWTOs) und Büchern (Guides)

zur Installation, Konfiguration und Administration von Linux: [www.tldp.org](http://www.tldp.org)

z.B.: - *Linux Administration Made Easy*: [www.tldp.org/LDP/lame/LAME/linux-admin-made-easy/](http://www.tldp.org/LDP/lame/LAME/linux-admin-made-easy/)

→ - *Advanced Bash-Scripting Guide*: [www.tldp.org/LDP/abs/html/abs-guide.html](http://www.tldp.org/LDP/abs/html/abs-guide.html)

- **Interessante Dokumentation, die zeigt, wie ein Linux-System von Grund auf selbst aufbaut wird**

[www.linuxfromscratch.org](http://www.linuxfromscratch.org)

- **Schulnetze mit Linux** vom Landesbildungsserver Baden-Württemberg

[www.lehrerfortbildung-bw.de/st\\_digital/netz/muster/linux/](http://www.lehrerfortbildung-bw.de/st_digital/netz/muster/linux/)

- **Linux Portale und Linkssammlungen**

Die wichtigste distributionsunabhängige Seite zu Linux: [www.linux.org](http://www.linux.org)

IBM engagiert sich stark im Linux-Sektor: [www.ibm.com/developerworks/views/linux/library.jsp](http://www.ibm.com/developerworks/views/linux/library.jsp)

- **Zertifizierungen des Linux Professional Institute (LPI):** [www.lpice.eu/](http://www.lpice.eu/)

- **Linux Distributionen** [www.debian.org](http://www.debian.org) [de.opensuse.org](http://de.opensuse.org) [www.redhat.com](http://www.redhat.com) [www.ubuntu.com](http://www.ubuntu.com)  
[www.knoppix.de](http://www.knoppix.de) Plop-Linux

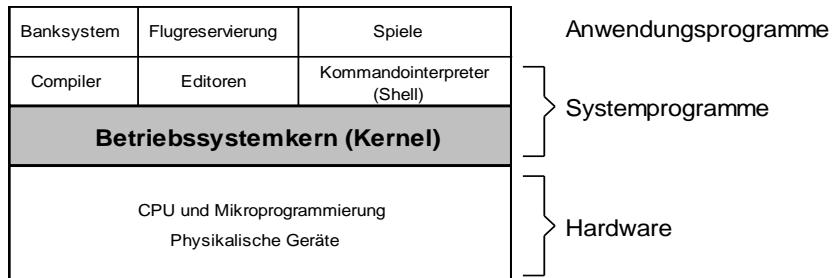
- **Linux Kernel**

Download des aktuellsten Linux-Kernels: [www.kernel.org/](http://www.kernel.org/)

## 2 Was ist ein Betriebssystem?

### 2.1 Grundlagen

Ein Rechensystem besteht aus Hardware, Systemprogrammen und Anwendungsprogrammen.



Unter einem Betriebssystem wird meist die Summe aus Betriebssystemkern (Kernel) und den wichtigsten Systemprogrammen verstanden. Ein Linux-Kernel besitzt ungefähr eine Größe von 1 - 8 MiB. Weitere Systemkomponenten benötigen zusätzlich 10 - 15000 MiB Plattenplatz.

Ein Betriebssystem versteckt die *realen* Eigenschaften der verwendeten Hardware vor dem Programmierer einer Applikation. Den Anwendungsprogrammen wird so eine verallgemeinerte Sicht auf die Hardware gewährt.

- Das Betriebssystem verallgemeinert die Benutzung der Hardware-Ressourcen des Rechners dadurch, dass es über definierte Software-Schnittstellen (APIs, *application programming interface*) den Zugriff darauf zur Verfügung stellt.

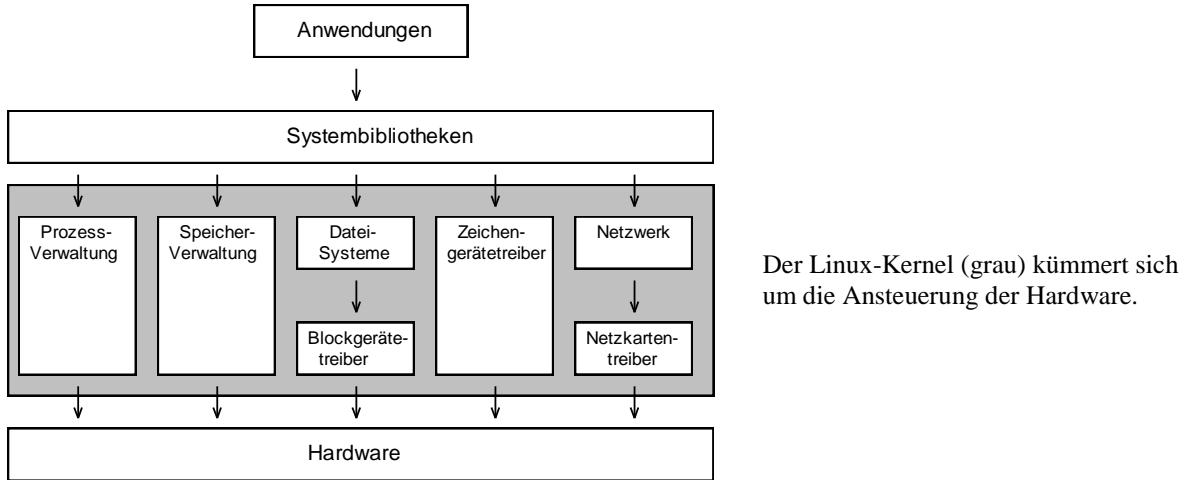
**Aber:** Was passiert, wenn mehrere Programme bzw. Benutzer gleichzeitig dieselbe Ressource nutzen wollen?

- Das Betriebssystem dient als **Ressourcenverwalter**, der Betriebsmittelanforderungen befriedigt, deren Benutzung protokolliert und abrechnet und bei Anforderungen, die von verschiedenen Programmen oder Benutzern im Konflikt stehen, vermittelt.

### 2.2 Die Aufgaben des Betriebssystemkerns (Kernel)

Siehe auch die Abbildung auf der nächsten Seite!

<b>Prozessverwaltung</b> <i>dispatching and scheduling</i>	Mehrere Benutzer können mit mehreren Aktionen gleichzeitig im System arbeiten, ohne sich gegenseitig zu beeinflussen. Damit dies möglich ist, müssen die Aktionen organisiert, koordiniert und verwaltet werden. Die zu verwaltende Einheit nennt man <b>Prozess</b> .
<b>Speicherverwaltung</b> <i>memory management</i>	Zuteilen von Speicherbereichen an laufende Prozesse, Schutz der Speicherbereiche vor unbefugten Zugriffen, bei Speicherengpässen: Auslagern ( <i>swapping</i> ) von ganzen oder von Teilen ( <i>paging</i> ) von Prozessen auf die Platte in Dateien ( <i>swap file</i> , <i>page file</i> ) oder Partitionen ( <i>swap partition</i> )
<b>Dateiverwaltung</b> <i>file handling</i>	Zuteilung von Plattenplatz für Programme, Benutzer und Systemdienste, Schreiben, Lesen und Strukturieren der Daten auf den Massenspeichern, Durch Bereitstellen einer entsprechenden Schnittstelle ( <i>VFS, virtual file system</i> ) kann der Kernel Dateisysteme unterschiedlichster Art (EXT3, FAT, NTFS, usw.) ansprechen.
<b>Geräteverwaltung</b> <i>resource management</i>	Der Kernel ermöglicht und kontrolliert den Zugriff auf die Hardware. Er steuert die Ein- und Ausgabearbeiten der Anwender. An den Kernel gebunden sind die entsprechenden Gerätetreiber ( <i>device driver</i> ), die damit eine Kommunikation zwischen Kern und Hardware ermöglichen.

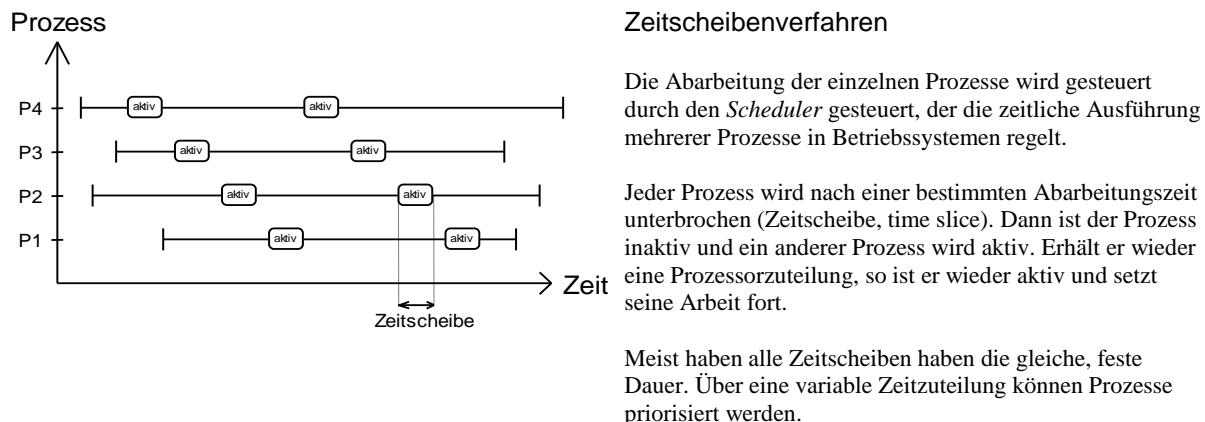


## 2.3 Eigenschaften von Betriebssystemen

### Multitasking

Multitasking bedeutet, dass ein Rechner aus Anwendersicht mehrere Aufgaben (*tasks*) gleichzeitig übernehmen kann. Das Betriebssystem unterteilt die gleichzeitig anstehenden Aufgaben in Einzelarbeitsschritte (Prozesse) und steuert durch ein Zeitscheibenverfahren (*scheduling*) das gleichmäßige Abarbeiten der Prozesse.

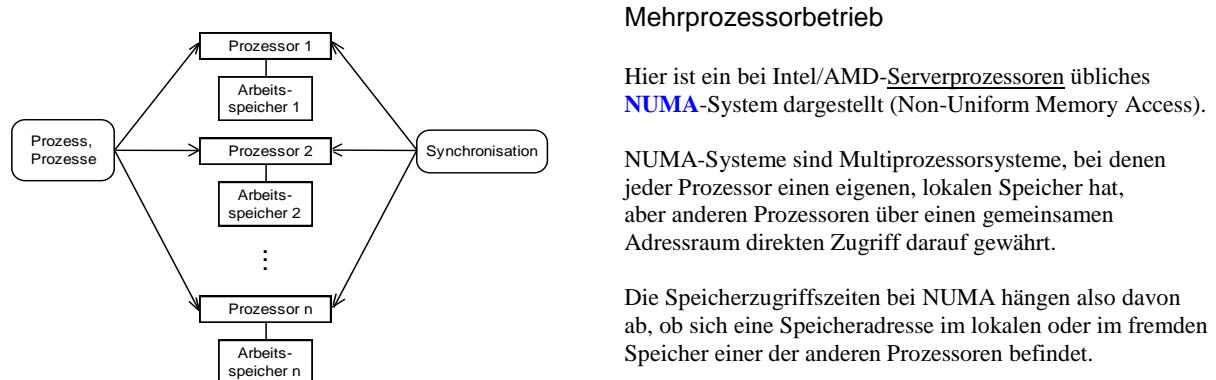
Der Betriebssystemkern steuert den Taskwechsel bzw. die Ressourcenfreigabe, indem laufend nach einer vom Betriebssystemkern festgelegten Zeit vom gerade aktiven Prozess zum nächsten weitergeschaltet wird.



### Mehrprozessorbetrieb

Praktisch alle Betriebssysteme unterstützen mehrere Prozessorkerne bzw. mehrere Prozessoren (CPU). So können mehrere Tasks auf die zur Verfügung stehenden Prozessorkerne verteilt und gleichzeitig parallel bearbeitet werden. Dies führt zu einer deutlichen Leistungssteigerung.

Problematisch dabei ist einerseits, dass viele Anwenderprogramme immer noch nicht für die Nutzung mehrerer Prozessorkerne optimiert sind. Andererseits kann es bei einer größeren Anzahl von Prozessorkernen zu einem Kommunikationsproblem zwischen den einzelnen Tasks kommen, was dann wiederum die Gesamtleistung beeinträchtigt.



## 3 Die Linux-Geschichte

### 3.1 UNIX

Das erste *Unics* wurde 1969 von Ken Thompson und Dennis Ritchie bei den Bell Laboratories (AT&T und Western Electric) geschrieben, um auf einer wenig benutzten DEC PDP-7 (*programmable data processor*) das Spiel *Space Travel* zu spielen ☺. Bis 1971 war Unix als Version 1 auf eine PDP-11 portiert; Version 4 wurde 1973 von dem ursprünglich in Assembler geschriebenen Quelltext fast vollständig in die eigens dafür entwickelte Hochsprache C umgeschrieben.

Weil AT&T durch Verträge mit der US-Bundesregierung daran gehindert war, Unix zu vermarkten, gab sie für Lehr- und Forschungszwecke die Unix-Sourcelizenzen zu sehr günstigen Konditionen an Universitäten weiter. Dieser Schritt führte zu einer sehr dynamischen Entwicklung von Unix. Es verbreitete sich schnell und aus den Universitäten flossen viele Ideen in die Entwicklung des Systems ein.

Seit Version 6 (1975) hat AT&T aber auch kommerzielle Lizenzen für Unix verkauft. Spätestens seit dem Jahr 1984, als ein weiteres Gerichtsurteil AT&T die Vermarktung von Software erlaubte, wurde Unix als System-V unter rein kommerziellen Gesichtspunkten verbreitet. Gerade wegen seiner Verbreitung an den Universitäten hat es sich im kommerziellen Sektor schnell durchgesetzt. Da Unix fast komplett in C geschrieben wurde, ist es auf praktisch alle Rechnerarchitekturen portiert ([www.bell-labs.com/history/unix/](http://www.bell-labs.com/history/unix/)).

### 3.2 Minix

Im Jahr 1987 hat Andrew Tanenbaum, Professor an der Freien Universität von Amsterdam, ein Lehrbetriebssystem für PCs veröffentlicht, das ohne jeden AT&T Code die Funktionalität von Unix Version 7 hat und inzwischen Freeware ist. Minix ist keine echte Basis für Anwenderprogramme, aber es ist ein sehr lehrreiches Spielzeug. Minix wurde auf den Atari ST, den Amiga und den Apple Macintosh portiert und es wurde an die erweiterten Möglichkeiten der Intel-386-Prozessoren angepasst ([www.minix3.org](http://www.minix3.org)).

### 3.3 Linux

Im März 1991 fing Linus Benedict Torvalds in Helsinki damit an, die Möglichkeiten des Intel-386-Prozessors in seinem neuen PC zu studieren. Er hatte das 386er Minix und das C-Entwicklungssystem der Free Software Foundation installiert. Nur ein halbes Jahr später war aus den Assemblerstudien ein kleines, lauffähiges Betriebssystem entstanden. Als Linus im September 1991 die erste Version (0.01) von Linux an interessierte Minixer verschickte, musste es noch unter Minix übersetzt und installiert werden. Diese Verbindung von Minix und Linux hat sich aber nicht im Quelltext niedergeschlagen.

Linus Torvalds hat seine eigene Entwicklung von Anfang an frei angeboten. Jeder konnte die Quelltexte bekommen und daran mitarbeiten. Die im Januar 1992 herausgegebene Version 0.12 von Linux war bereits ein stabil laufender Kernel. Es gab den GNU C-Compiler, die bash-Shell, den Emacs-Editor und viele der GNU Utilities. Dieses Betriebssystem wurde in comp.os.minix angekündigt und per anonymous FTP weltweit verteilt. Zur Kommunikation der interessierten Entwickler wurde die Rubrik alt.os.linux im USENET eingerichtet. Dieses Medium und der anonyme FTP Service im Internet ermöglichen eine Programmierung, wie sie in den Vorstandsetagen der mächtigen Softwareschmieden erträumt wird.

Innerhalb weniger Monate wurde aus dem weitgehend in Assembler geschriebenen Minikernel ein ausgewachsenes Betriebssystem mit vollständiger Unix-Funktionalität. Die weitgehende POSIX-Konformität von Linux und die umfangreichen C-Bibliotheken des GNU-C-Compilers erleichtern die Portierung von Unix- oder BSD-Software. Praktisch das gesamte Angebot an freier Software läuft auch unter Linux.

Dank der enormen Leistungen, die bereits seit vielen Jahren von der Free Software Foundation, dem X Consortium, den Universitäten und ungezählten Organisationen und Einzelpersonen in diesem Bereich erbracht wurden, haben die Linux-Distributionen heute einen Umfang erreicht, der die Angebote kommerzieller Unix leicht in den Schatten stellt.

Die Kernelentwicklung von Linux teilt sich in zwei Bäume: die Entwicklerkernel und die Produktionskernel. Entwicklerkernel sollten nur benutzt werden, wenn man experimentierfreudig ist und auch ab und an einen Systemabsturz tolerieren kann. Produktionskernel sind für den Einsatz auf Produktionssystemen getestet und laufen ausgesprochen stabil. Entwicklerkernel erkennt man an einer Versionsnummer, die eine ungerade zweite Zahl enthält (z.B. 1.3.10, 2.3.34, 2.5.3). Produktionskernel haben stets eine gerade zweite Versionszahl (z.B. 1.2.4, 2.2.10, 2.4.14, 2.6.32).

Diese Trennung ist jedoch seit Juli 2004 ausgesetzt, stattdessen werden Änderungen und Erweiterungen laufend in die 3.0er-Serie eingearbeitet.

### 3.4 Fragen zu Betriebssystem-Grundlagen

Siehe zweitletzte Seite!

## 4 20 Jahre Linux (aus c't 18/2011)

Report | Zwanzig Jahre Linux

Thorsten Leemhuis

# 20 Jahre Linux

## Vom Hobby-Projekt zur Schlüsseltechnik der modernen Computerwelt

Trotz Unkenrufen und Schwarzmalerei hat Linux viele Bereiche der Computerwelt für sich erobert – dabei waren einige der heute von Linux bewohnten Geräteklassen noch Science Fiction, als der Betriebssystem-Kern vor 20 Jahren entstand.

Sie müssen Linux nicht mögen. Und trotzdem würde Ihre Welt zusammenbrechen, wenn man Linux von heute auf morgen herausreißen würde. Das fängt beim Aktienhandel an, den vielen Börsen über Linux-Rechner abwickeln. Weiter ginge es mit Google, Amazon, Facebook oder eBay, wo Linux überall das Rück-

grat der Infrastruktur bildet. Und auch die Gerätschaften in so mancher Wohnung würden die Mitarbeit einstellen, denn viele WLAN-Router und so manches Smartphone verwendet heute den Betriebssystemkern, dessen Entwicklung ein 21-jähriger Student namens Linus Torvalds vor zwanzig Jahren begann.

Geplant war das Vordringen in diese ganz unterschiedlichen Bereiche nicht – Linux ist vielmehr in diese und andere Einsatzgebiete hineingewachsen, weil einzelne Entwickler oder Unternehmen den Betriebssystemkern für diese Bereiche fit gemacht haben. Auf diesem Weg gab es unzählige Fehlpro-

nosen und Befürchtungen, die sich in der Retrospektive als vollkommen unbegründet oder gar amüsant erweisen.

Die ersten Fehleinschätzungen lieferte Torvalds gleich selbst, als er im August 1991 seine Arbeit an Linux in der Newsgroup des Unix-artigen Lehrbetriebssystem Minix ankündigte: Linux sei ein Hobby, das nicht so groß und professionell werde wie GNU; es laufe nur auf i386-Systemen, sei nicht portierbar und würde vermutlich nie etwas anderes als AT-Festplatten unterstützen, denn etwas anderes habe er nicht zur Hand.

Der Erfolg von Linux ist mit Schuld, dass es ein „GNU“ genanntes Betriebssystem bis heute nicht gibt: Linux hat sicher viele Entwickler angelockt, die sonst am Betriebssystem-Kernel „Hurd“ mitgearbeitet hätten, an dem das GNU-Projekt schon seinerzeit arbeitete. Linux profitierte jedoch von der Vorarbeit des GNU-Projekts, denn das entwickelte schon damals Software wie Bash für die Kommandozeile oder Basis-Tools wie Cat, Cp, Sleep & Co. – und die Compilersuite GCC zum Entwickeln weiterer Programme. Ohne diese noch heute mit Linux kombinierte Software ist Linux wie ein Automotor: ohne Drumherum zu nichts nutze.

Ernsthaft arbeiten konnte man mit der ersten Linux-Version wirklich nicht: Sie konnte lediglich die Bash ausführen, Treiber fehlten und Systemabstürze waren nichts Ungewöhnliches. Enthusiasten nahmen das als Herausforderung. Einige schickten Fehlerberichte oder Verbesserungsvorschläge, denen Torvalds sich widmete. Andere Entwickler implementierten Verbesserungen, nach denen ihnen der

### 1991

Im April 1991 begann Linus Torvalds mit der Entwicklung eines Vorläufers von Linux, mit dem er die Fähigkeiten seines i386-Prozessors ausprobieren wollte. Das wuchs schnell zu einem Betriebssystem-Kern heran, dem er draufhin ein Posix-Interface verpasste, damit GNU-Programe wie Bash und GCC darauf laufen. Nach der Veröffentlichung von **Linux 0.01** im September 1991 begannen bald die ersten Entwickler, Torvalds Verbesserungen zukommen zu lassen.

### 1992

Ende des Jahres liefen viele **GNU-Tools**; im darauf folgenden Mai auch das X Window System, das eine grafische Oberfläche mit Linux möglich machte. Im Herbst folgte erste Teile eines **TCP/IP**-Stacks zur Netzwerkkommunikation. Kurz darauf erschien die erste Linux-Distribution. In einer endlosen Folge von 0.9xx-Versionen machte Linux damals viele Fortschritte; erste SCSI- und Soundtreiber zogen ein und das Dateisystem ext2 entstand. Die verbesserte Posix-Unterstützung

### 1994

machte eine Portierung des BSD-Drucksystems möglich. Schließlich erschien im Mai 1994 die **Version 1.0** – Windows 3.11 war da gerade einige Monate alt. Unter den Linux-Entwicklern fanden sich bereits noch heute aktive Größen wie **Alan Cox**. Um Anwendern eine stabile Basis zu bieten, pflegten sie diese Versionsreihe weiter, während sie die nächste größere Überarbeitung in der Unstable-Series mit Versionsnummer 1.1 vorantrieben.

### 1995

Aus der ging dann im März 1995 die Version 1.2 hervor, die auch auf Prozessor-Architekturen jenseits von Intels x86 arbeitete und erheblich ausgebauten Netzwerk-Unterstützung enthielt. Linux wurde so gut, dass allmählich ein ernsthafter Markt um Linux entstand – damals etablierten sich etwa **Red Hat** und **Suse**. Linux galt in der Zeit des Erscheinens von Windows 95 zwar noch als Hacker-Spielzeug, eroberte sich aber zusammen mit Software wie Samba,

Sinn stand, und schickten sie zur Integration an den Linux-Erfinder – darunter bald auch Treiber für andere Speichermedien. Manche Verbesserungen nahm Torvalds direkt auf, andere erst nach von ihm angeregten Änderungen; gelegentlich implementierte er eingereichte Verbesserungen auch in einer Weise neu, die er für besser hielt.

## Debatte

Anfangs unterlag der Kernel einer von Torvalds selbstgeschriebenen Lizenz, die kommerziellen Vertrieb untersagte; zur noch heute genutzten Version 2 der vom GNU-Projekt entwickelten GPL (General Public License) wechselte er erst Ende Januar 1992 bei der Version 0.12. Gleichzeitig entflammt eine mittlerweile legendäre Debatte zwischen Torvalds und Andrew S. Tanenbaum: Der Minix-Erfinder und angesehene Informatik-Professor hatte behauptet, Linux sei „obsolet“, also überholt.

Als einer der beiden Hauptgründe führte Tanenbaum an, Linux nutze wie andere unixoide Kernel einen monolithischen Ansatz – alle Funktionen und auch Treiber werden direkt in einen großen Klotz zusammengebaut. Als Technik der Zukunft galten modular aufgebaute Mikrokerneln. MacOS X nutzt heute einen solchen; Linux hingegen ist nach wie vor monolithisch, ohne dass noch viel darüber geredet wird. Seit der im Mai 1994 veröffentlichten Version 1.0 kann Linux aber Kernel-Module nachladen, was einen Nachteil des Ansatzes beseitigt.

Der zweite Kritikpunkt: Linux sei explizit auf Intels i386-Prozessoren ausgelegt und nicht auf

andere Architekturen portierbar. Das hatte Torvalds bei der ersten Ankündigung von Linux selbst erwähnt. Trotz dieser Einschränkung und der Kritik von angesehener Stelle eroberete Linux weiter Herzen und Systeme. Daraufhin passierte zum ersten Mal etwas, was sich in der Linux-Geschichte mehrfach wiederholen sollte: Einige Entwickler trieben nach und nach tief greifende Umbaumaßnahmen voran, um Linux für Aufgaben fit zu machen, für die es zuvor vollkommen ungeeignet schien.

Durch diese Bemühungen lief schließlich das im März 1995 erschienene Linux 1.2 nicht mehr nur auf x86-CPUs, sondern auch auf Workstations und Servern mit Alpha-, Mips- und Sparc-Prozessoren. Treibende Kräfte hinter den Portierungen waren Programmierer, die sich eine Alternative zu den kommerziellen und teureren Unixen wünschten; die ließen damals meist auf Servern und Workstations, die keine x86-Prozessoren enthielten. Auf den so entstandenen Grundlagen bauten andere Portierungen auf. Je nach Zählweise läuft Linux heute auf mehreren Dutzend oder an die hundert verschiedenen Architekturen – weit mehr als jeder andere Betriebssystem-Kernel.

## Geschäftsmodelle

In den Anfangstagen von Linux wusste noch niemand, ob oder wie man Geld mit Open-Source-Software verdienen kann. Vorgemacht, wie es geht, hat es Red Hat: Im aktuellen Geschäftsjahr wird die Firma aller Wahrscheinlichkeit nach das erste Mal die Marke von einer Milliarde US-Dollar Jahresumsatz durchbrechen.

## Geburtstermin

Über den Geburtstag von Linux herrscht Uneinigkeit. Das greifbarste Datum ist der 17. September 1991 – der Tag, an dem Linus Torvalds die Linux-Version 0.01 auf einem FTP-Server zum Download bereitstellte.

Dass er an einem freien Betriebssystem arbeitet, hatte er allerdings bereits am 25. August verkündet – manche legen den Geburtstag daher dorthin.

Erste Indizien auf Linux gab es bereits am 3. Juli, als Torvalds in einer Newsgruppe nach Informationen zum Posix-Standard fragte, welche die Sys-

temschnittstellen von Unix definieren.

Eine Vorstufe von Linux ist noch älter: Die hat der damals 21-jährige Torvalds im April 1991 begonnen, um die Funktionen des i386-Prozessors auszuprobieren.

Geburtstag feiert indes nur das, was manchmal auch als Linux-Kernel bezeichnet wird. Bash, GCC, der X-Server von X.org und viele der typischen Bestandteile von häufig schlicht „Linux“ genannten Linux-Distributionen wie Ubuntu haben bereits einige Jahre mehr auf dem Buckel.

Das hat das Unternehmen maßgeblich der Linux-Distribution Red Hat Enterprise Linux (RHEL) zu verdanken. Der erste Vorläufer dieser Distribution erschien im November 1994. Da feierte die allererste Linux-Distribution Yggdrasil schon fast ihr Zweijähriges und das Debian-Projekt hatte schon mehr als ein Jahr auf dem Buckel; ebenso Slackware, die älteste der heute aktiven Linux-Distributionen. Red Hat Linux wurde nicht nur schnell zu einer der beliebtesten Distributionen, sondern hatte auch bei Firmenkunden Erfolg. Im August 1999 ging das Unternehmen an die Börse. Seitdem hat die Firma den Umsatz nach und nach gesteigert und in den letzten Jahren immer Gewinn abgeworfen – sogar in gesamtwirtschaftlich eher düsteren Zeiten.

Allerdings verdient Red Hat sein Geld nicht mit Software-Li-

zenzen, sondern einem von der Firma mitgeprägten Open-Source-Geschäftsmodell: Es verkauft Support für die eigenen Produkte in einem Abonnement. Nicht-Red-Hat-Kunden, die ein RHEL-Medium in die Finger bekommen, dürfen es daher auf ihren Rechnern einsetzen. Nur für Systeme mit Service-Abo erhält man jedoch Betreuung und Updates, die Fehler und Sicherheitslücken korrigieren. Das ist für viele Unternehmen Anreiz genug, um ein Abonnement abzuschließen – dabei gibt es mit CentOS oder Scientific Linux sogar kostenlose Nachbauten der Red-Hat-Distribution, deren Updates auch frei erhältlich sind.

## Rückgrat

Viele Firmen trauten sich vor zehn oder fünfzehn Jahren nicht,

1996	1998	1999	2001
Sendmail oder Apache in so manchen Firmen einen Platz – nicht selten durch „einfache“ Administratoren ohne Wissen der IT-Entscheider.	tierten ihre Software nach Linux, auch Netscape seinen Browser und StarDivision den OpenOffice-Vorläufer StarOffice. Die KDE- und Gnome-Entwickler legten derweil die Grundsteine für ihre Desktops.	Die Kernel-Entwickler bereiteten unterdessen Linux 2.2 vor. Das erschien Anfang 1999 und brachte bessere Unterstützung für Multiprozessor-Systeme sowie Sound- und Video-Hardware. Große Hardware-Hersteller wie Compaq, Dell, Gateway und HP kündigten erste Linux-Server an. Red Hat ging an die Börse und IBM kündigte eine Linux-Initiative an; die sorgt später für Aufsehen, als die Firma Ende 2000 nachlegte und <b>1 Milliarde US-Dollar</b> in Linux investierte.	Anfang 2001 folgte <b>Linux 2.4</b> , das auf dreizehn verschiedenen Architekturen lief und auf x86-Systemen bis zu 64 GByte Arbeitsspeicher adressierte. Parallel entdeckten Embedded-Entwickler den freien Betriebssystem-Kern mehr und mehr für sich, nachdem sich Linux zur festen Größe im Server-Markt gemacht hatte. Hardware-Hersteller wie Intel und später auch AMD beteiligten sich immer intensiver an der Kernel-Entwicklung, denn ordentliche Linux-
Den nächsten großen Sprung machte Linux mit der <b>Version 2.0</b> im Juli 1996, die erstmals Firewall-Funktionen bot und auf Systemen mit mehreren Prozessoren lief – und diese auch nutzte. Zu der Zeit erhielt Linux auch sein Maskottchen, den <b>Tux</b> . Nicht nur Datenbankhersteller wie Borland und Informix por-	1998 kündigten schließlich auch Oracle und SAP Linux-Versionen ihrer Software an: Linux und Open Source waren geschäftsfähig geworden, was das amerikanische Wirtschaftsblatt Forbes unterstrich, indem es Torvalds auf das Titelblatt hob.		



Quelle: BusinessWeek

**Linus und sein Linux wurden mit der Zeit immer mehr für voll genommen: Im Sommer 1998 hievt erst Forbes den Linux-Erfinder aufs Cover, im Januar 2005 folgte die BusinessWeek.**

Linux auf unternehmenskritischen Systemen einzusetzen. Einige Start-Ups waren mutiger und nutzten Linux als einen Schlüssel zum Erfolg. Google etwa verwendete es damals wie heute auf den Rechnern der Suchmaschine, mit der das Unternehmen groß wurde. Auch die Server-Infrastruktur von Amazon, eBay, Twitter, Facebook und vielen Cloud-Anbietern fußte von Anfang an zu großen Teilen auf Linux.

Viele Webserver, das Gros der WLAN-Router für zu Hause und nahezu alle in der Top-500-Liste verzeichneten Supercomputer hat Linux erobert. Laut Jim Zemlin, dem Direktor der Linux Foundation, verwenden derzeit rund 75 Prozent der Börsen Linux auf den Servern, die den Wertpapierhandel abwickeln. Ohne Linux kann man somit heute keine Aktien kaufen –

auch nicht die von Apple und Microsoft.

Der Windows-Hersteller passte seine Einstellung zu Linux mit den Jahren an. Mit den „Halloween-Dokumenten“, schlechten Studienergebnissen und Anti-Linux-Werbung streute das Unternehmen vor zehn Jahren noch reichlich Verunsicherung.

Zum 20. Geburtstag schickte das Unternehmen nun ein Glückwunschvideo. Microsoft pflegt zu dem Kernel-Treiber für die eigenen Virtualisierungsschnittstelle Hyper-V im Rahmen des Linux-Kernels. Beim Verbessern dieses Treibers schaffte es Microsoft sogar unter die Top-Ten der Unternehmen und Gruppen, die zwischen Linux 2.6.39 und 3.0 die meisten Änderungen beigesteuert haben.

### Alphamännchen

Doch auch zwanzig Jahre haben nicht gereicht, manche Bedenken rund um Linux aus der Welt zu schaffen. Noch immer erntet man gelegentlich erstaunte Blicke von Leuten, die erfahren, dass Torvalds die Weiterentwicklung ganz alleine bestimmt.

Denn auch heute muss jede noch so kleine Änderung durch Torvalds Hände. Selbst programmiert er allerdings nur einen Bruchteil der Änderungen; mit der Zeit ist er mehr und mehr zum Lenker geworden, der Änderungen anderer einbaut, die Qualität sicherstellt und anderen Ratschläge gibt. Eine Marschrichtung zur Ausrichtung des Kernels gibt Torvalds dabei allerdings nicht vor. Genau das hat zu immer neuen und aufeinander aufbauenden Verbesserungen geführt, die Einsatzmöglichkeiten eröffnen, an die Jahre zuvor keiner gedacht hätte.



**Microsofts Verhältnis wandelte sich im Laufe der Linux-Geschichte:** Vor zehn Jahren verunsicherte das Unternehmen mit spöttischer Werbung, heute gratuliert es zum Geburtstag und trägt selbst zu Linux bei.



Ein aktuelles Beispiel dafür sind die unter anderem von Thomas Gleixner vorangetriebenen Änderungen, die Linux Echtzeitfähigkeiten verleihen. Viele davon sind in den letzten Jahren in Linux eingegangen, dabei hatte Torvalds die Idee anfangs für verrückt erklärt. Es scheint nur noch eine Frage der Zeit, bis auch die verbliebenen Änderungen in den offiziellen Linux-Kernel einziehen.

Torvalds achtet jedoch sehr darauf, dass Erweiterungen und Umbauten nicht das große Ganze ins Wanken bringen. Dabei ist er manchmal sehr direkt und ranzt ihm vertraute Kernel-Entwickler durchaus mal an, wenn er ihre Änderungen für schlecht hält. Oder er lässt Verbesserungen außen vor, selbst wenn sie viele Fürsprecher haben – darunter das Reiser4-Dateisystem oder die Desktop-Optimierungen von Con Kolivas. Dennoch genießt Torvalds ho-

hes Ansehen unter den Kernel-Entwicklern.

### Aufbegehren

Durch sein geschicktes Agieren hat Torvalds auch verhindert, dass größere Abspaltungen („Forks“) entstanden, die auf Dauer miteinander konkurrieren oder gar inkompatibel zueinander werden. Genau davor wurde und wird immer wieder gewarnt – nicht zu Unrecht, denn diese Gefahr besteht durchaus.

Ende der Neunziger sah alles nach einer Abspaltung aus: Torvalds kam mit der Integration der ihm zugesandten Änderungen nicht hinterher und lieferte häufig nicht mal eine Erklärung, warum er bestimmte Verbesserungen ignorierte. Bald war davon die Rede, Torvalds habe einen Burnout; einige unzufriedene Mitstreiter planten bereits eine Abspaltung. Die kam dann nicht zustande, weil altgediente

#### 2002

Unterstützung wurde für den Markterfolg der eigenen Produkte immer wichtiger.

Windows XP war gerade ein halbes Jahr alt, als Red Hat im Frühjahr 2002 sein erstes Enterprise Linux (**RHEL**) veröffentlichte. Ein Jahr später begann der Unix-Spezialist **SCO** damit, im Linux-Umfeld aktive Firmen auf Schadensersatz zu verklagen; Hauptziel war IBM, das im Rahmen seiner Linux-Initiative Programmcode und Programmierkonzepte von SCO gestohlen

#### 2003

und in Linux eingebaut haben soll. Das ganze sorgte jahrelang für Verunsicherung, erwies sich aber als haltlos; seit dem Sommer 2010 scheint es, als sei die Sache ausgestanden.

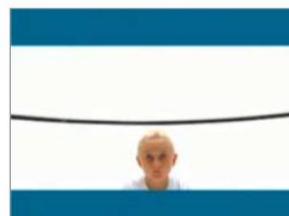
Das im Dezember 2003 freigegebene **Linux 2.6** brachte Alsaaudio-Treiber, Software-Suspend und die Sicherungserweiterung SELinux. Die ebenfalls neue Unterstützung für NUMA und die heute gängigen x86-64-Prozessoren war parallel auch in Linux 2.4 eingezogen. Red Hat hatte

#### 2004

damals gerade das Geschäft mit **Heimanwendern** weitgehend aufgegeben und das Community-Projekt Fedora gestartet, das seitdem die Basis von RHEL bildet. Das damals von Novell aufgekauft Suse setzte bald mit OpenSuse und Suse Linux Enterprise auf ein ähnliches Modell. Weiter aufgemischt wurde der Distributionsmarkt Ende 2004, als Newcomer Canonical mit dem Debian-basierten **Ubuntu** begann, die Herzen vieler Heimanwender im Sturm zu erobern.

#### 2005

Zu der Zeit schaffte Torvalds die Unstable-Serie ab und begann, Linux im Rahmen der 2.6er-Reihe weiterzuentwickeln – das a<sub>n</sub>fangs mit Skepsis aufgenommene Modell bewährte sich gut. Die **Stable-Series** führten die Kernel-Entwickler Anfang 2005 ein; sie versorgte die jeweils aktuelle Kernel-Version mit Korrekturen, während die Entwickler die nächsten vorbereiteten. Linux lernte derweil Dinge wie RAID 6, Größenanpassungen von Ext3 zur Laufzeit oder das



Entwickler die Wogen glätteten und Torvalds entlasteten, indem sie als Adjutanten einsprangen. Mittlerweile sind es „Subsystem-Maintainer“, die als Vorfilter für einzelne Bereiche zuständig sind.

Ausgestanden war das ganze aber damit noch nicht, denn zwei Jahre später entstand aus denselben Gründen abermals Unzufriedenheit; erneut machte das Wort „Burnout“ die Runde. Als ein Grund dafür galt die manuelle Verwaltung des Quellcode und der Austausch von Änderungen über Mails mit Patches, weil Torvalds damals verbreitete Quellcodeverwaltungssysteme wie CVS als ungeeignet zur Linux-Entwicklung einstufte. Er sah aber ein, dass er mit dem Wachstum von Linux zum Flaschenhals wurde; er und kurz danach auch andere Entwickler begannen daraufhin im Februar 2002 mit dem Einsatz des verteilten Quellcodeverwaltungssystems Bitkeeper, das einen einfachen Austausch von Änderungen ermöglichte.

Bitkeeper erleichterte Torvalds die Arbeit erheblich. Das beendete nicht nur die Burnout-Diskussion, sondern führte auch zu einem schnelleren Wachstum des Kernels. Drei Jahre lang erzeugte es aber eine andere Form von Aufruhr, denn das Quellcodeverwaltungssystem war proprietär; die von ihm genutzten Server stellten die Bitkeeper-Macher von BitMover Open-Source-Entwicklern unter bestimmten Bedingungen kostenlos zur Verfügung. Die sah das Unternehmen verletzt, als Samba-Urgestein Andrew „Tridge“ Tridgell eine quelloffene Client-Software für Bitkeeper entwickelte. Die Situation eskalierte, obwohl Torvalds die Gemüter zu



IBM sorgte für Aufsehen, als es 2000 begann, mehr und mehr Geld in Linux investierte; im Frühjahr 2003 war Linux sogar Gegenstand einer IBM-Werbung, die während des Super Bowl ausgestrahlt wurde.



beruhigen versuchte. Er stellte daraufhin Anfang April 2005 die Nutzung von Bitkeeper ein; BitMover kündigte parallel an, die Unterstützung für die kostenlosen Services einzustellen.

Torvalds ging aber nicht zur alten Arbeitsweise zurück, sondern nahm sich einige Tage Auszeit und startete die Entwicklung des verteilten Quellcodeverwaltungssystems Git. Das diente schon nach wenigen Tagen zur Entwicklung des damals in Arbeit befindlichen Linux 2.6.12 und wird noch heute zur Kernel-Entwicklung genutzt. In den Zeiten von Git stieg Zahl und Umfang der Änderungen abermals erheblich an – mit jeder neuen Kernel-Version werden heute rund eine bis eineinhalb Millionen Codezeilen aufgenommen, umhergeschoben oder gelöscht.

Git entpuppte sich als der zweite große Erfolg von Torvalds, denn viele Open-Source-Projekte nutzen heute das Quellcodeverwaltungssystem; darunter Android, Eclipse, Gnome, KDE, Qt, Perl und X.org. Die Wei-



terentwicklung von Git hat Torvalds schon nach vier Monaten in andere Hände gegeben, um sich wieder auf Linux konzentrieren zu können.

## Anläufe

Ende der Neunziger entdeckten Embedded-Entwickler Linux mehr und mehr für sich. Durch deren Bemühungen erlangte der Kernel Fähigkeiten, die es auch für Handys beziehungsweise Smartphones attraktiv machten. Dennoch sind mehrere Hersteller und Industrievereinigungen erst daran gescheitert, Linux in diesem Markt zu etablieren – darunter Maemo/MeeGo und LiMo. Dass sich Linux heute dennoch in den Hosentaschen vieler Menschen findet, ist Googles Android zu verdanken. Das macht vieles anders – so viel, dass es außer dem Kernel so gut wie nichts mit Linux-Distributionen für Desktops und Notebooks gemeinsam hat.

Nach dem Erfolg auf Servern hörte man bereits Ende der

Neunziger immer wieder Stimmen, die den Durchbruch von Linux auf dem Desktop prognostizierten. Bis heute hat es allerdings keine Firma geschafft, Linux-Distributionen für PCs und Notebooks von Heimanwender zum Erfolg zu führen oder signifikant Kapital aus den Heimanwendern zu schlagen. Anläufe dazu gab es indes mehrere – 1998 etwa Corel Linux, das ebenso scheiterte wie die Linux-Version von Word Perfect. Knapp zehn Jahre später sah es mit dem Aufkommen der Netbooks kurzzeitig aus, als würde Linux doch endlich einen Fuß in die Tür bekommen. Damit war dann doch schnell wieder Schluss – schuld waren daran aber nicht nur die verbilligten Windows-Lizenzen für Netbooks, sondern auch viele Hardware-Hersteller, die Kunden mit schlechten Linux-Distributionen quälten.

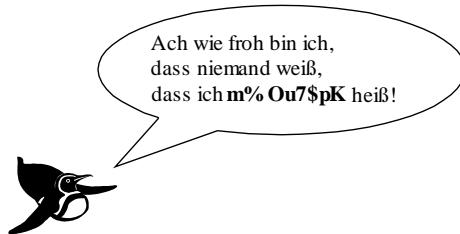
Nichts deutet darauf hin, dass Linux in nächster Zeit die Desktop-PCs erobernt. Junge oder neue Gerätelassen wie Smartphones, Tablets und Co. sind aber gerade dabei, die Computer-Welt umzukrempeln. Es würde zu Linux passen, wenn es sich dabei durch eine Hintertür irgendwann doch noch einen Platz auf Heimcomputern erobert, sodass der anfängliche Misserfolg bei Heimcomputern beim 30. Geburtstag vielleicht schon amüsant wirkt. (thl) ct

2006	2007	2008	2011
Schreiben auf das Windows-Datensystem NTFS. Kurz vor der Einführung von Windows Vista erschien Ende 2006 die Playstation 3, auf der auch Linux lief. Für Aufsehen sorgte eine Kooperationsvereinbarung zwischen Novell und Microsoft, in dessen Rahmen Microsoft den eigenen Kunden Support-Zertifikate für Suse Linux Enterprise verkaufte; außerdem stellt sie Kunden der Unternehmen von wechselseitigen Patentansprüchen frei, was in	der Open-Source-Szene für allerlei Unmut sorgte. Anfang 2007 entstand die <b>Linux Foundation</b> . Das Konsortium zur Linux-Förderung konnte seitdem immer mehr Mitglieder gewinnen und bezahlt Torvalds ein Gehalt, wie es die Vorfahrts-Organisation OSDL schon seit 2003 tat, um die Unabhängigkeit Torvalds sicherzustellen. Der Funktionsumfang von Linux stieg derweil – Version <b>2.6.20</b> erhielt etwa mit <b>KVM</b> einen eigenen Hypervisor zur Virtualisie-	zung. Mit einem neuen WLAN-Stack verbessert sich langsam die problematische Unterstützung für WLAN-Hardware. Mit dem im September 2008 vorgestellten Smartphone G1 betritt <b>Android</b> das Licht der Öffentlichkeit und bachtet Linux in die Taschen vieler Anwender, die von Open Source noch nie gehört haben. Das Dateisystem <b>Ext4</b> erreicht bei 2.6.28 die Produktionsreife; Kernel-based Mode-Setting (KMS) revolutioniert die Art, wie Linux mit Gra-	fikchips umgeht. Novell wurde Übernahmeziel und geht Ende 2010 an Attachmate, das Suse in einen in Nürnberg ansässigen Unternehmensbereich ausgliedert hat. Im Mai 2011 kündigt Torvalds überraschend an, ihm würden die Versionsnummern des Kernels zu groß: Anlässlich des Geburtstags und dem Beginn des dritten Lebensjahrezehnts gibt er dem im Juli veröffentlichten Nachfolger von 2.6.39 nicht die Bezeichnung 2.6.40, sondern <b>3.0</b> .
c't 2011, Heft 18			83

## 5 Die erste Annäherung

Nach dem Einschalten des Systems führt dieses einige Tests durch und fordert zur Anmeldung auf.

### 5.1 An- und Abmelden



#### Anmelden

Linux ist ein Multi-User-Betriebssystem, deshalb müssen sich alle Systembenutzer mit Usernamen und Passwort authentifizieren.

Während der Installation ist bereits ein besonderes Benutzerkonto angelegt worden, das des Benutzers **root**. Dies ist der Username des Administrators bzw. des Superusers der **alle** Rechte im System besitzt.

Beachten Sie bitte die Regeln für sichere Passworte (min. 8 Zeichen, kein sinnhaftes Wort, Sonderzeichen, Groß- und Kleinschreibung mischen).

Da man als **root** sehr viel zerstören kann, sollte man nur dann als **root** arbeiten, wenn dies unbedingt nötig ist!

Die Anmeldeanforderung lautet:

Login: *(Eingabe des Usernamens)* ↵  
Password: *(Eingabe des Passwortes)* ↵

Nach erfolgreicher Anmeldung erfolgt die Aufforderung zur Eingabe weiterer Kommandos durch den Prompt. Der Prompt (Eingabeaufforderung) kann je nach System und aktuellem Pfad unterschiedlich gestaltet sein.

#### Abmelden

Will ein Benutzer die Linux-Sitzung wieder beenden, muss er sich beim System abmelden.

Dazu gibt es drei Möglichkeiten: die Tastenkombination **Strg+D**, oder die Kommandos **exit** bzw. **logout**.

### 5.2 Herunterfahren des Systems

**WICHTIG!** Vor dem Abschalten des Rechners **muss** Linux ordnungsgemäß heruntergefahren werden.  
Wird einfach abgeschaltet, riskiert man im ungünstigsten Fall z.B. ein zerstörtes Dateisystem.

Mit dem Kommando **shutdown** wird das System heruntergefahren. Es kann nur von **root** ausgeführt werden.

- |                          |  |
|--------------------------|--|
| <b>shutdown -h now</b> ↵ | Das System fährt herunter und hält an.           |
| <b>shutdown -r now</b> ↵ | Das System fährt herunter und bootet neu.        |
| <b>poweroff</b> ↵        | Das System fährt herunter und schaltet sich aus. |

**Hinweis:** ↵ steht für die Eingabetaste (RETURN-Taste)!

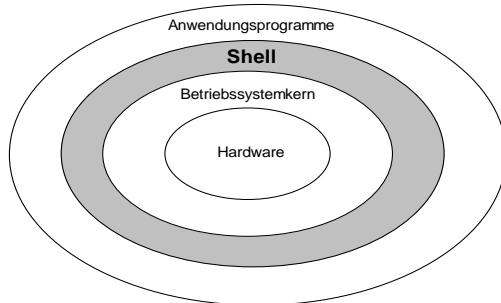
Wirkung von **shutdown** und **poweroff**

- Alle Benutzerprozesse werden beendet.
- Der Datei-Buffer-Cache und die Festplatte werden synchronisiert.
- Alle Dateisysteme werden ausgehängt.

**WICHTIG!** Auf vielen Linux-Systemen kann von allen Usern das System auch mit **Strg+Alt+Entf** neu gestartet werden. Als Administrator sollten Sie den Usern diese Möglichkeit nehmen! (es sei denn, es sprechen wichtige Gründe dagegen)  
- **SUSE ab 12.1:** Löschen Sie die Datei `/lib/systemd/system/ctrl-alt-del.target`  
- neuere Debian-/Ubuntu-Systeme: Löschen Sie die Datei `/etc/init/control-alt-delete.conf`  
- ältere Linux-Systeme: Editieren Sie die Datei `/etc/inittab`, suchen Sie die Zeile mit der Zeichenfolge `ctrlaltdel` und kommentieren Sie diese Zeile mit Hilfe des Kommentarzeichens `#` am Zeilenanfang aus. Um die geänderte Konfiguration neu einzulesen, müssen Sie `init q` eingeben.

# 6 Allgemeines zu Linux-Kommandos und zur Terminalbedienung

## 6.1 Die Shell



Die *shell* (Muschel, Schale) ist ein Programm, das als Kommandointerpreter für den Dialog und die Interaktion mit dem Benutzer zuständig ist  
(Windows: `cmd.exe`  
Linux/UNIX: `bash`, `sh`, `csh`, `zsh`, `ash` usw.).

Die Shell stellt an einem Terminal eine Kommandozeile zur Verfügung, setzt Benutzereingaben in entsprechende Betriebssystemaufrufe um, startet Systemprogramme und gibt die Ausgaben und Meldungen dieser Programme sowie die des Betriebssystems auf der Konsole aus.

Hilfe zur von uns verwendeten `bash`-Shell gibt es mit dem Befehl `help`, `help Befehl` und `man bash`.

## 6.2 Wichtige Eigenschaften der `bash`-Shell

- Eingabekomplettierung mit der Tabulator-Taste
- Spezifizieren von Dateinamen mithilfe von Mustern und Jokerzeichen (*wildcards*)
- Ein- und Ausgabeumleitung bei Programmen
- Verbinden von Ein- und Ausgabedatenströmen mehrerer Programme (*pipes*)
- Kommandozeileneditor
- Leistungsfähige Skriptsprache (*shell script*)

## 6.3 Bedienung des Terminals

- Eine Kommandozeile besteht aus einem oder mehreren Wörtern.
- Die Wörter sind durch Leerzeichen (*blank*) oder Tabulatoren (*tab*) getrennt.
- Groß- und Kleinschreibung **muss** beachtet werden.
- Alle Eingaben werden durch die RETURN-Taste ↴ abgeschlossen.

Kommando	Beschreibung
<b>Rücktaste</b> bzw. <b>Strg-h</b>	Löschen des jeweils letzten Zeichens der Kommandozeile
<b>Strg-u</b>	Löschen der gesamten Kommandozeile
<b>Strg-s</b>	Bildschirmausgabe stoppen
<b>Strg-q</b>	Bildschirmausgabe fortsetzen
<b>Shift-Bild↑</b> bzw. <b>Shift-Bild↓</b>	Text zurückholen, der schon über den Bildschirm gelaufen ist
<b>Strg-c</b>	Abbruch eines Kommandos oder Programms
↓ bzw. ↑	Kommandowiederholung

## 6.4 Virtuelle Terminals

Sie können sich an Ihrem System mehrfach unter gleichem oder unter verschiedenen Benutzernamen anmelden. Hierzu können Sie durch die Tastenkombination **Alt-Fn** (n = 1 bis 6) zwischen den virtuellen Terminals umschalten.

Aus der grafischen Oberfläche heraus muss man **Strg-Alt-Fn** für die virtuellen Terminals drücken. Zurück zur grafischen Oberfläche kommt man dann meist mit **Alt-F7**.

## 6.5 Einfache Kommandos

Syntax: **Kommandoname** dann die RETURN-Taste ↵ drücken.

Beispiele:

**who** Listet alle Benutzer auf, die im System angemeldet sind.  
Auszgabe z.B.:



**date** Datum und Systemzeit ausgeben  
Auszgabe z.B.: Thu Sep 6 10:16:52 MEST 2001

**cal** Kalender des aktuellen Monats.

## 6.6 Kommandos mit Argumenten

Syntax: **Kommandoname -Option [-weitere Optionen ... ] Argument [weitere Argumente ... ]**

Angaben, die dem Kommandonamen folgen, werden als Parameter interpretiert.  
Man unterscheidet zwischen Argumenten und Optionen:

**Argumente** Dateinamen oder sonstige Angaben, die von den Kommandos verarbeitet werden.

**Optionen** Sind eine Art Schalter, die eine Feineinstellung der Kommandos bewirken.  
Sie werden durch ein vorangestelltes Minuszeichen gekennzeichnet und  
sie werden meistens **vor** den eigentlichen Argumenten angegeben.

Beispiele:

**cal 12 1989** Kalender für Dezember 1989

**cat /etc/profile** Inhalt der Datei /etc/profile ausgeben

**cat -n /etc/services** -n ist die Option zur Zeilennummerierung

Ausgabe:  
Bei der Ausgabe der Datei /etc/services erhält  
jede Zeile eine Nummer.

## 6.7 Hilfen zu Kommandos

Fast alle Kommandos geben eine kurze Hilfestellung zu ihren Argumenten und Optionen aus, wenn man sie folgendermaßen aufruft: **Kommandoname --help** oder **Kommandoname -?**

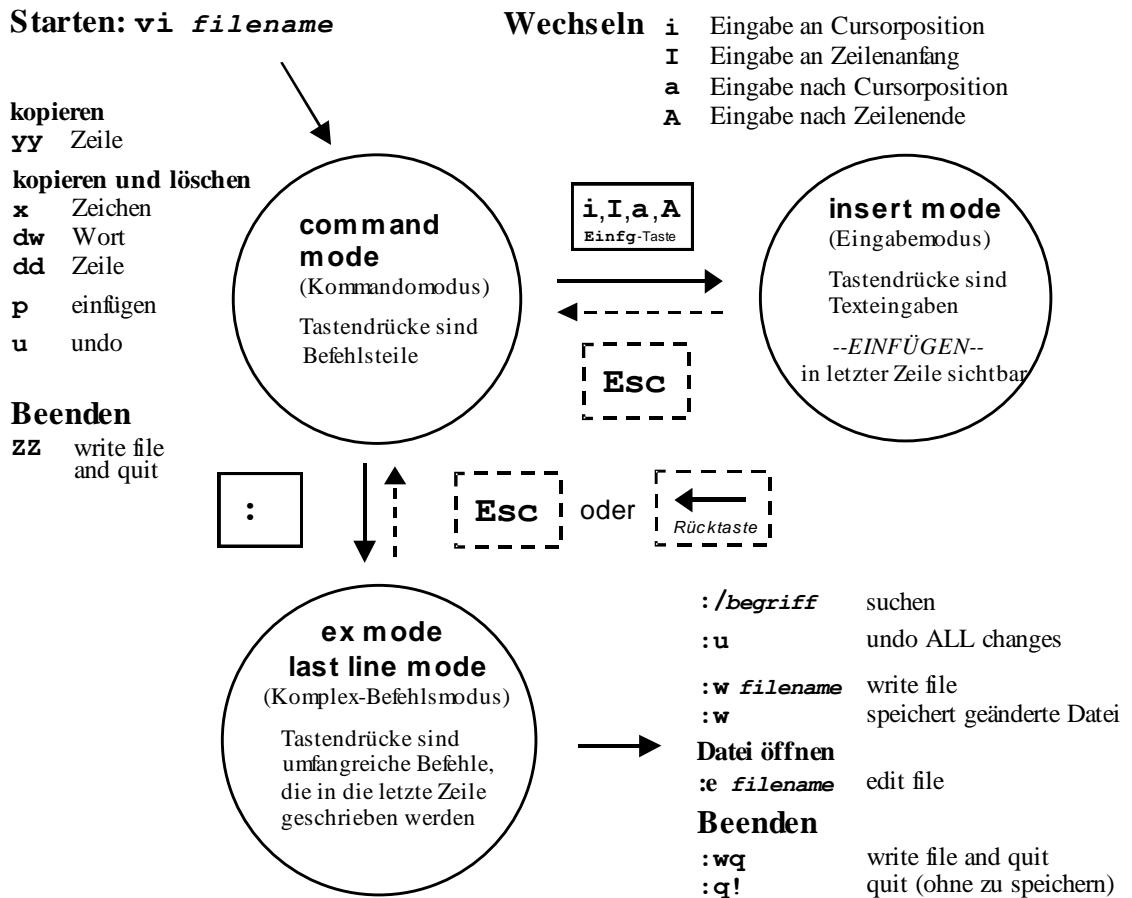
Umfangreichere Hilfestellungen geben die Man-Pages (*manual pages*). Mit **man Kommando** wird vom Kommando less die entsprechende Man-Page aufgerufen. In den Man-Pages kann mit den Pfeil- und Bildlauf-Tasten manövriert werden.

Begriffe können durch Eingabe von **/Begriff** gesucht werden. Weitere Fundstellen springt man dann mit der **n**-Taste an, vorherige mit **N**.

Mit **q** verlässt man die Man-Pages!

## 7 Der Editor vi

Das Wechseln zwischen den 3 Modi des Editors vi ist mit den angegebenen Tastenkombinationen möglich.



- Im Eingabemodus *insert mode* werden Tastendrücke als Texteingabe interpretiert. In den Einfügemodus gelangt man vom Befehlsmodus beispielsweise durch die Taste **i**.

- Im Kommandomodus *command mode* werden Tastendrücke als Befehle interpretiert.

Syntax: **[Wiederholungsfaktor n]KommandoTexteinheit**

Beispiele:

<b>4dw</b>	löscht die nächsten <b>4</b> Wörter ( <i>delete</i> )
<b>5dd</b>	löscht die nächsten <b>5</b> Zeilen
<b>J</b>	nächste Zeile hochziehen, d.h. das Zeilenende-Zeichen loeschen ( <i>join</i> )
<b>y</b>	aktuelle Zeile in die Zwischenablage kopieren ( <i>yank</i> )
<b>5y</b>	5 Zeilen in die Zwischenablage kopieren
<b>p</b>	Zeile(n) aus der Zwischenablage einfügen ( <i>paste</i> )
<b>G</b>	springt an den Beginn der letzten Zeile der Datei

- Im *last line mode* (Komplex-Befehlsmodus) können umfangreiche Befehle ausgeführt werden.

Syntax: **:[Adresse]Kommando [Optionen]**

Beispiele:

<b>:q!</b>	beendet den vi ohne zu speichern
<b>/suchbegriff</b>	sucht im Text. Wenn etwas gefunden wurde, dann drückt man <b>n</b> für das nächste, <b>N</b> für das vorherige Vorkommen des Suchmusters
<b>:set number</b>	aktiviert die Zeilenanzeige in der aktuellen vi-Session
<b>:s/Maier/Mair/g</b>	ersetzt in der aktuellen Zeile alle Vorkommen von "Maier" durch "Mair"
<b>:6,9s/\ß/ss/g</b>	ersetzt in den Zeilen 6-9 alle "ß" durch "ss"
<b>:%s/\//./g</b>	ersetzt im gesamten Dokument alle "/" durch "." (Quoting mit \ nötig!)
<b>:50,\$s/[aeiou]/e/g</b>	ersetzt ab Zeile 50 bis zum Dokumentenende in jeder Zeile alle Vokale "a", "e", "i", "o" oder "u" durch "e"

# 8 Grundlagen der Dateiverwaltung

## 8.1 Namenskonventionen im Dateisystem

- **Zwischen Groß- und Kleinschreibung wird streng unterschieden!**
- Dateinamen können bis 255 Zeichen lang sein
- Sonderzeichen in Dateinamen sind erlaubt (Vorsicht!)
- Dateinamen, die Leerzeichen enthalten, müssen in Hochkommas eingeschlossen sein
- Dateinamen, die mit einem Punkt beginnen, sind versteckte Dateien
- der *forward slash* / ist das Trennzeichen für Verzeichnisse

## 8.2 Jokerzeichen und Reguläre Ausdrücke der Shell

Der Einsatz von Jokerzeichen (*wildcards*) bei der Angabe von Dateinamen in Kommandoargumenten veranlasst die Shell, die gesuchten Dateinamen zu finden und dann erst die kompletten Namen der gefundenen Dateien als Argumente dem Kommando zu übergeben. Kombinationen von Jokerzeichen werden *Reguläre Ausdrücke* genannt.

- Achtung:**
- Die Auswertung der Regulären Ausdrücke erfolgt bereits durch die Shell, **nicht** durch das aufgerufene Kommando!
  - Reguläre Ausdrücke von grep, sed, python usw. unterscheiden sich **stark** in ihrer Syntax zu denen der Shell!

Zeichen	Bedeutung
?	Spezifikation für genau ein beliebiges Zeichen
*	Spezifikation für kein oder beliebig viele Zeichen * erfasst alle Zeichen, auch Punkte außer denjenigen am Anfang des Dateinamens
Der reguläre Ausdruck *graf*	trifft zu auf grafic.doc, junger-graf, steffigrafundherragassi, readme.graf, graf
[123]	genau eines der angegebenen Zeichen
[a-f]	genau ein Zeichen aus dem angegebenen Bereich
[!abc]	keines der angegebenen Zeichen
[^abc]	keines der angegebenen Zeichen
~	Abkürzung für das Home-Verzeichnis des gerade angemeldeten Benutzers

### Beispiele zum Kommando ls

- ```
ls [a-f]*      Dateien, die mit einem der Buchstaben von a bis f beginnen
ls *[_-]*      Dateien, die irgendwo im Namen ein _ . oder - haben
ls [!a-z]*     Dateien, die nicht mit a bis z beginnen
ls *.[!hc]      Dateien, die nicht mit .h oder .c enden
```

**Hinweis:** \* erfasst nicht nur die Dateien, sondern auch die Verzeichnisse, **ls \*** durchsucht auch die Dateien der Unterverzeichnisse. Dies kann durch die Option **-d** ausgeschaltet werden.

### Quoting

Möchte man nicht, dass Jokerzeichen bereits durch die Shell ausgewertet werden, so muss man ihnen ihre Sonderbedeutung durch *Quoting* nehmen. Es gibt mehrere Möglichkeiten:  
Voranstellen eines \ vor jedes Jokerzeichen bzw. den Regulären Ausdruck in einfache Hochkommas ('') schreiben

### Versteckte Dateien

Versteckte Dateien (z.B. .profile) beginnen mit einem Punkt und werden von der Shell nur angezeigt, wenn das Kommando **ls** mit der Option **-a** oder durch die explizite Angabe des Punktes z.B. mit **.\*** aufgerufen wird (gilt nicht für root).

Versteckte Dateien sind vor einem versehentlichen Löschen durch **rm \*** geschützt.  
Sie müssen durch **rm .dateiname** gelöscht werden (oder mit **rm .\***).

## 8.3 Einige der wichtigsten Linux/UNIX-Kommandos zur Dateiverwaltung

|              |                                                                |              |                                            |
|--------------|----------------------------------------------------------------|--------------|--------------------------------------------|
| <b>ls</b>    | Dateien und Verzeichnisse anzeigen                             | <b>grep</b>  | Suchen von Zeichenketten in Dateien        |
| <b>cd</b>    | Verzeichniswechsel                                             | <b>sort</b>  | Textdateien zeilenweise sortiert ausgeben  |
| <b>pwd</b>   | gibt das aktuell gesetzte Verzeichnis aus                      | <b>pr</b>    | Dateiinhalte formatiert ausgeben           |
| <b>cat</b>   | Inhalt von (Text-)Dateien ausgeben                             | <b>cut</b>   | Auswählen von Spalten und Feldern          |
| <b>more</b>  | seitenweises Betrachten von Texten                             | <b>chmod</b> | Zugriffsrechte ändern                      |
| <b>less</b>  | komfortablere Version von <b>more</b>                          | <b>chown</b> | Dateibesitzer ändern                       |
| <b>cp</b>    | Kopieren von Dateien                                           | <b>chgrp</b> | Gruppenzugehörigkeit einer Datei verändern |
| <b>rm</b>    | Löschen von Dateien/Unterverzeichnissen                        | <b>file</b>  | Analysieren des Inhalts von Dateien        |
| <b>mv</b>    | Umbenennen/Verschieben von Dateien                             | <b>mount</b> | Einhängen (Mounten) von Dateisystemen      |
| <b>mkdir</b> | Erstellen von Verzeichnissen                                   | <b>du</b>    | Platzverbrauch von Dateien/Verz. anzeigen  |
| <b>rmdir</b> | Löschen von leeren Verzeichnissen                              | <b>df</b>    | freien Speicherplatz ausgeben              |
| <b>find</b>  | nach bestimmten Dateien suchen                                 | <b>ps</b>    | Prozesse des Systems abfragen              |
| <b>echo</b>  | Meldung auf der Konsole ausgeben                               | <b>kill</b>  | einen Prozess/ein Programm terminieren     |
| <b>test</b>  | universelles Kommando, das z.B. prüft, ob eine Datei existiert | <b>expr</b>  | auf der Kommandozeile rechnen              |
|              |                                                                | <b>bc</b>    |                                            |

## 8.4 Dateiarten

**gewöhnliche Datei** *ordinary file* Texte, Programme, Binärcode, Systembibliotheken, ..

- Gerätedatei** *special file*
- Logische Beschreibung der Hardware (z.B. Drucker, Monitor...).
  - Über die Gerätedateien wird auf die Gerätetreiber zugegriffen.
  - Lesender und schreibender Zugriff wie auf gewöhnliche Dateien.
  - Man unterscheidet *zeichenorientierte Gerätedateien* (z.B. für Terminal) und *blockorientierte Gerätedateien* (z.B. für Festplatte)

Beispiele wichtiger Gerätedateien:

|            |                                                                                         |
|------------|-----------------------------------------------------------------------------------------|
| /dev/fd0   | erstes Floppylaufwerk, meist ein Link auf die Gerätedatei /dev/fd0H1440                 |
| /dev/sda   | erste S-ATA oder SCSI-Festplatte                                                        |
| /dev/hda   | IDE-Festplatte oder IDE-CD-ROM (Master am ersten IDE-Controller)                        |
| /dev/sda1  | erste primäre Partition der ersten S-ATA oder SCSI-Festplatte                           |
| /dev/sda5  | erstes log. Laufwerk in Erweiterter Partition auf der ersten S-ATA oder SCSI-Festplatte |
| /dev/sr0   | erstes S-ATA-CD/DVD-ROM ( <i>scsi-readable</i> )                                        |
| /dev/sde2  | zweite primäre Partition der fünften S-ATA oder SCSI-Festplatte                         |
| /dev/sdb   | zweite S-ATA oder SCSI-Festplatte, <u>evtl. auch ein USB-Stick</u>                      |
| /dev/tty1  | erstes virtuelles Terminal                                                              |
| /dev/mouse | Link auf die von der Maus verwendete Schnittstelle.                                     |

Anstelle der Größenangabe werden bei Gerätedateien mit **ls -l** zwei durch Komma getrennte Zahlen angegeben:

1. Zahl: *major device number* Identifiziert den Gerätetyp
2. Zahl: *minor device number* Wird dem Gerätetreiber übergeben, der sie nach Belieben interpretiert.

**Verzeichnis (Ordner)** *directory (folder)* Kann Dateien und Unterverzeichnisse (*subdirectories*) enthalten.

**Soft-Link** Verweis auf eine andere Datei, siehe Kapitel *Datei-Links*

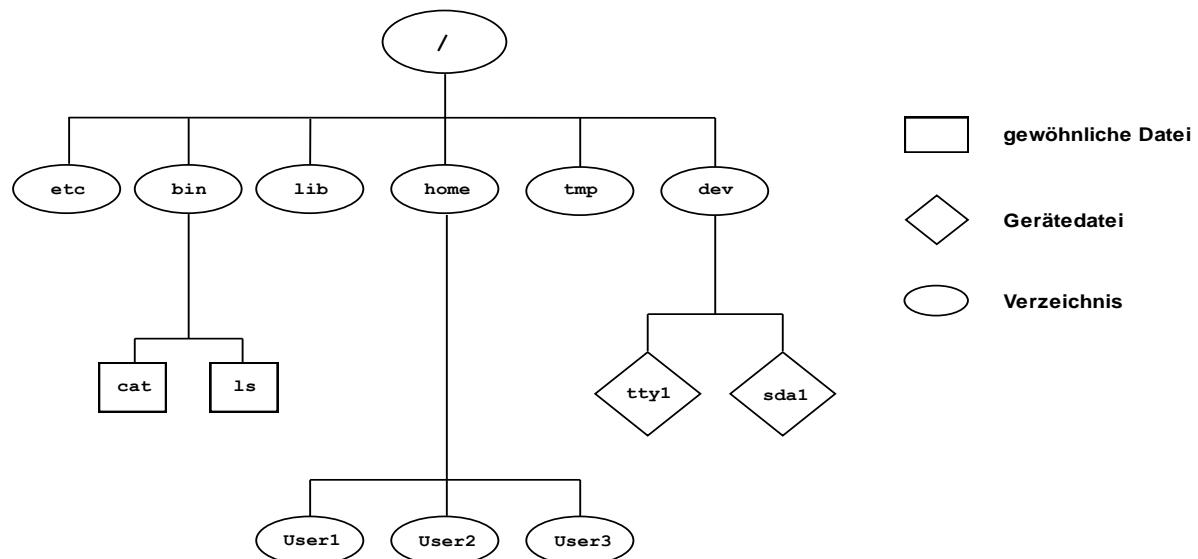
**Named Pipe** Datenstruktur zur Inter-Prozesskommunikation, siehe Kapitel *Dateiarten und Zugriffsrechte*

## 8.5 Verzeichnisstruktur von Linux-Systemen (FHS)

Linux und UNIX verfügen über eine ähnliche Verzeichnisstruktur und verfügen nicht über Laufwerksbuchstaben. Die Verzeichnisse sind so organisiert, dass sich eine hierarchische Struktur ergibt (Verzeichnisbaum). Die Wurzel dieses umgedrehten Baums wird als **root** bzw. **root directory** oder **/** bezeichnet. In jeder Ebene des Baumes können sowohl Dateien als auch Verzeichnisse liegen, letztere bilden die nächste Ebene des Baumes. Name und Position jeder Datei kann durch eine entsprechende **Pfadangabe** spezifiziert werden, die an der Wurzel beginnt und alle zu durchlaufenden Verzeichnisse auflistet.

Um bei Linux einheitliche Verzeichnisstrukturen-Strukturen zu erhalten, wurde der Filesystems Hierarchy Standard (FHS) geschaffen. Einzelne Linux-Distributionen weichen in Details voneinander ab, die Akzeptanz der FHS-Empfehlungen steigt zusehends.

**Beispiel einer Linux-Verzeichnisstruktur (unvollständig):**



Bei openSUSE befinden sich im root-Directory folgende Unterverzeichnisse:

|                |                                                                                                                                  |
|----------------|----------------------------------------------------------------------------------------------------------------------------------|
| /boot          | Kernel und Dateien, die vom Linux-Bootmanager benötigt werden                                                                    |
| /bin           | Dienstprogramme und Kommandos, die von allen Benutzern ausgeführt werden können                                                  |
| /sbin          | Dienstprogramme und Kommandos zur Systemverwaltung.<br>Sie können nur von <b>root</b> ausgeführt werden                          |
| /etc           | Konfigurationsdateien für Systemdienste und systemweite Einstellungen                                                            |
| /etc/init.d    | Skripten und Programme zur Systeminitialisierung                                                                                 |
| /dev           | Gerätedateien                                                                                                                    |
| /lib           | Systembibliotheken ( <i>shared libraries</i> ) vergleichbar mit den *.dll-Dateien unter Windows                                  |
| /proc          | Prozess-Datei-System, Schnittstelle zum Kernel z.B. zur Verwaltung von Prozessen                                                 |
| /root          | Home-Directory des Systemverwalters <b>root</b>                                                                                  |
| /home          | Home-Directories der Benutzer                                                                                                    |
| /usr           | Anwendungsprogramme, X-Window-System, Quellen zu Linux usw. (statische Daten)                                                    |
| /usr/src       | Quellen des Linux-Kernels (falls installiert)                                                                                    |
| /usr/share/doc | Hilfdateien und Dokumentationen zu Anwendungsprogrammen                                                                          |
| /var           | Vergleichbar mit <b>usr</b> , jedoch mit dynamischen Daten                                                                       |
| /var/log       | System Log-Dateien                                                                                                               |
| /opt           | Ist für eigene Anwendungssoftware vorgesehen                                                                                     |
| /lost+found    | Verloren gegangene Dateien. Diese entstehen beim nicht ordnungsgemäßen Verlassen des Systems und anschließendem Filesystem-Check |
| /tmp           | Temporäre Dateien                                                                                                                |
| /media/floppy  | Mount-Directory ( <i>mount point</i> ) für das Floppy-Laufwerk (SUSE-spezifisch)                                                 |
| /media/cdrom   | Mount-Directory für das CD-ROM-Laufwerk (SUSE-spezifisch)                                                                        |
| /mnt           | Mount-Directory für externe Dateisysteme                                                                                         |

## 8.6 Begriffe zum Dateisystem

|                                             |                                                                                                                                                                                                                                                                                                                           |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>root directory</i>                       | Die Wurzel des Dateisystems hat kein übergeordnetes Verzeichnis und wird mit / bezeichnet.                                                                                                                                                                                                                                |
| <i>working directory, current directory</i> | Momentaner Aufenthaltsort im Verzeichnisbaum.<br>Er kann mit <b>pwd</b> ( <i>print working directory</i> ) angezeigt werden.                                                                                                                                                                                              |
| <i>home directory</i>                       | Heimatverzeichnis, Platz im Dateisystem, der einem Benutzer zugeordnet ist. Der Pfadname des Home-Directories steht in der Systemvariablen <b>HOME</b> und kann mit <b>echo \$HOME</b> angezeigt werden. Das Home-Directory stellt die Privatsphäre eines Benutzers dar, es kann jederzeit mit <b>cd</b> erreicht werden. |
| <i>login directory</i>                      | Anmeldeverzeichnis, Position im Dateibaum, an der sich der Anwender nach dem Anmelden befindet, im Normalfall wird dies sein Home-Directory sein.                                                                                                                                                                         |
| <i>parent directory</i>                     | Übergeordnetes Verzeichnis<br>Das <i>parent directory</i> kann vom aktuellen Verzeichnis aus mit <b>cd ..</b> erreicht werden.                                                                                                                                                                                            |
| <i>absoluter Pfadname</i>                   | Vollständiger Pfadname, der ausgehend von / den Weg durch den Dateibaum bis zur gewünschten Datei angibt (z.B.: /home/herbert/briefe/maria.txt)                                                                                                                                                                           |
| <i>relativer Pfadname</i>                   | Die Angaben eines solchen Pfadnamens beginnen im aktuell gesetzten Verzeichnis.<br>Beispiel (für /home/herbert): briefe/maria.txt                                                                                                                                                                                         |

## 9 EXT2-Dateisystem

Laut Wikipedia ist eine Datei ein "...strukturierter Bestand inhaltlich zusammengehöriger Daten...".

Ein **Dateisystem** stellt die Ablageorganisation bzw. das Ordnungs- und Zugriffssystem für Dateien dar.

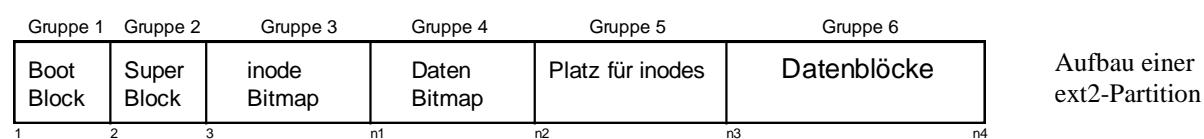
Um eine hohe Schreib-/Leseleistung zu erhalten, fassen Dateisysteme eine bestimmte Menge von Daten zu einem sog. **logischen Datenblock** zusammen und schreiben diese jeweils am Stück auf die Speichergeräte (z.B. Platte). Logische Blöcke sind die kleinste vom Dateisystem adressierbare Einheit. Auch wenn nur ein Byte eines Blocks zu ändern ist, muss der ganze Block gelesen, das entsprechende Byte geändert und dann der ganze Block geschrieben werden.

Linux unterstützt eine große Zahl von Dateisystemen. Welche Dateisysteme der aktuell installierte Kernel unterstützt, kann man in der Datei `/usr/src/linux/fs/filesystems.c` nachlesen, falls die zugehörigen Kernel-Quelltexte installiert sind.

Ein einfaches Linux-Dateisystem heißt **ext2** (*Extended Filesystem Version 2*).

Es unterstützt Dateinamen bis zu 255 Zeichen, Dateigrößen (bei 1kiB=1024Byte großen logischen Blöcken) bis zu 16GiB und kann dann Dateisystemgrößen mit bis zu 4TiB (1TiB = 1024GiB) verwalten.

Eine ext2-Partition ist eine durchnumerierte Folge von logischen Daten-Blöcken fester Größe.  
Diese Blöcke sind unterschiedlichen Gruppen zugeordnet:



| Gruppe | Block-Nr.     | Bedeutung                                                                   |
|--------|---------------|-----------------------------------------------------------------------------|
| 1      | 1             | Bootblock enthält ein kleines Lade-Programm zum Starten des Betriebssystems |
| 2      | 2             | Superblock gibt an, wie groß die folgenden vier Gruppen sind                |
| 3      | 3 bis n1      | inode-Bitmap, gibt an, welche Blöcke für inodes frei sind                   |
| 4      | n1 bis n2 - 1 | Daten-Bitmap, gibt an, welche Blöcke für Daten frei sind                    |
| 5      | n2 bis n3 - 1 | Speicherplatz für inodes                                                    |
| 6      | n3 bis n4     | Speicherplatz für Daten                                                     |

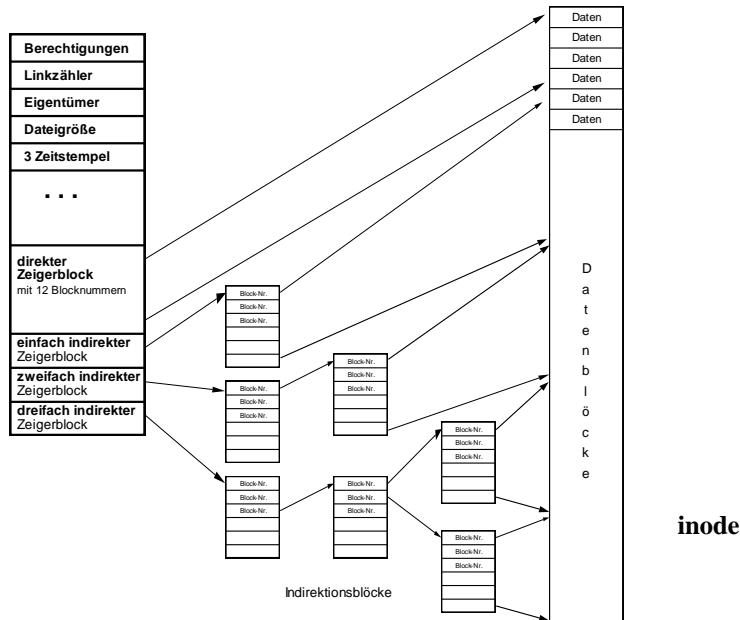
Zu den Gruppen 3 und 4: Diese Gruppen bestehen aus mehreren Blöcken und sind als *Bitmaps* organisiert. Jedes Bit innerhalb dieser Bitmaps gibt an, ob der zugehörige inode bzw. Datenblock aus der 5. bzw. 6. Gruppe frei oder belegt ist. So kann beim Anlegen einer neuen Datei schnell festgestellt werden, wo noch freie Blöcke vorhanden sind.

**Übrigens:** Windows nennt die Logischen Datenblöcke eines Dateisystems *Zuordnungseinheit* oder *Cluster*.

## inode

Inodes (Informationsknoten) liefern den Schlüssel zur Verwaltung der Daten. In ihnen werden, mit Ausnahme des Dateinamens, alle Verwaltungsinformationen einer Datei gespeichert. Bei ext2 werden für einen inode 256Byte Speicherplatz benötigt (früher: 128Byte).

Beim Formatieren eines ext2-Dateisystems durch das Kommando **mke2fs** kann die inode-Dichte angegeben werden. Normalerweise wird beim Formatieren zur Verwaltung von 4kiB Plattenplatz ein inode vorgesehen (bei einer Partition mit 100MiB also 25000 inodes, d.h. 25000 Verzeichnisse bzw. Dateien). Schöpfen die 25000 angelegten Dateien bzw. Verzeichnisse z.B. aufgrund ihrer geringen Größe die 100MiB nicht aus, bleibt Speicherraum ungenutzt. Dieses Problem kann man entschärfen, indem man eine höhere inode-Dichte wählt.



### Beschreibung des inode-Inhalts

| Feldname                                                         | Erklärungen                                                                                                                                                                                                           | Option zu <b>ls</b> |
|------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| Dateiart + Zugriffsrechte                                        | Dateiarten: gewöhnliche Datei, Verzeichnis (Siehe 8.4)<br>Zugriffsrechte: Siehe Kap. 10                                                                                                                               | <b>-1</b>           |
| Linkzähler                                                       | Anzahl der Hard-Links (s.u.)                                                                                                                                                                                          | <b>-i</b>           |
| Eigentümer- und Gruppen-Nummer                                   | siehe Kapitel zur Benutzerverwaltung                                                                                                                                                                                  | <b>-ln</b>          |
| Dateigröße                                                       | in Byte                                                                                                                                                                                                               | <b>-l</b>           |
| Datum der letzten inode-Änderung                                 | Zeitpunkt, an dem der inode selbst geändert wurde, z.B. durch eine Änderung der Besitzverhältnisse                                                                                                                    | <b>-lc</b>          |
| Datum der letzten Änderung des Dateiinhalts                      | Durch die Änderung des Dateiinhalts wird auch der Eintrag "Datum der letzten inode-Änderung" geändert.                                                                                                                | <b>-l</b>           |
| Datum des letzten Zugriffs                                       | Zeitpunkt zu dem die Datei zuletzt gelesen oder ausgeführt wurde.                                                                                                                                                     | <b>-lu</b>          |
| Verweis auf Daten-Block 0<br>.....<br>Verweis auf Daten-Block 11 | Diese 12 Verweise haben jeweils eine Größe von 32 Bit und enthalten die Blocknummern der ersten 12 Datenblöcke einer Datei. Ist die Datei kleiner als 12 Blöcke, werden die nicht benötigten Verweise auf 0 gesetzt.  |                     |
| Verweis auf den 1. Indirektionsblock                             | Ist der Inhalt einer Datei größer als 12 Blöcke, so enthält dieses Feld die Nummer eines Blocks, in welchem (bei 1kiB/Block) bis zu 256 weitere Verweise auf echte Datenblöcke zu finden sind (1. Indirektionsstufe). |                     |
| Verweis auf den Zweifach-Indirektionsblock                       | Reichen 12+256 Verweise nicht aus, enthält dieses Feld die Nummer eines Blocks, der auf bis zu 256 Blöcke mit jeweils 256 Verweisen auf echte Datenblöcke zeigt (2. Indirektionsstufe).                               |                     |
| Verweis auf den Dreifach-Indirektionsblock                       | Reichen 12+256+256*256 Verweise nicht aus, enthält dieses Feld die Nummer eines Blocks, welcher bis zu 256 Verweise auf zweifach indirekte Blöcke enthält (3. Indirektionsstufe).                                     |                     |

Bei 1kiB großen Blöcken sind so Dateigrößen von mehr als 16GiB möglich:

$$[12+256+256*256+256*256*256] * 1kiB = 17.247.252.480Byte = 16,06GiB$$

Wird die Blockgröße erhöht, so können in den Indirektionsblöcken auch mehr als 256 Blocknummern (die jeweils 32Bit groß sind) gespeichert werden: Anzahl der Blocknummern = Blockgröße/32Bit.

Beispiel für 4kiB-Blöcke:  $4kiB/32Bit = 4*1024*8Bit/32Bit = 1024$  Blocknummern

Dadurch kann die maximale Dateigröße erhöht werden. Bei 4kiB großen Blöcken wären so Dateigrößen von  $[12+1024+1024*1024+1024*1024*1024] * 4kiB = 4.402.345.721.856Byte = 4,0TiB$  möglich.

Die Inhalte des inode können teilweise mit **ls** angesehen werden. Hierzu müssen die oben in Spalte 2 angegebenen Optionen verwendet werden.

mögliche Ausgabe von **ls -l** :

| Dateiart               | Anzahl der festen Links | Zeitpunkt des letzten Schreib-Zugriffs |       |      |             |           |
|------------------------|-------------------------|----------------------------------------|-------|------|-------------|-----------|
| Zugriffsrechte         | Eigentümer              | meierst                                | users | 8391 | Aug 5 16:41 | maria.txt |
| -rwxr-xr-x             | 1                       | ↑                                      | ↑     | ↑    | ↑           | ↑         |
| Gruppe des Eigentümers |                         |                                        |       |      | Dateiname   |           |

Wenn eine neue Datei in einem Verzeichnis angelegt wird, wird zunächst ein inode für diese Datei aus der inode-Liste der Gruppe 3 reserviert und dann diese inode-Nummer zusammen mit dem Namen der neuen Datei in der dazugehörigen Verzeichnis-Datei eingetragen. Ein neuer inode wird allerdings nur dann benötigt, wenn es sich bei der neuen Datei nicht um einen Link handelt, denn im Falle eines Links ist bereits ein inode für die Originaldatei reserviert. In diesem Fall wird der Name des Links gemeinsam mit der inode-Nummer der Originaldatei in der Verzeichnis-Datei gespeichert (siehe Bild auf der nächsten Seite).

Die inode-Nummer einer Datei kann mit **ls -i Dateiname** abgefragt werden.

In frühen ext2-Versionen hatte die inode-Nummer eine Länge von 2 Byte (16Bit), dies begrenzte die Anzahl der Dateien eines Dateisystems auf 65536. Aktuell sind bei ext2 bis zu  $10^{18}$  Dateien möglich.

Beim Löschen einer Datei wird nur die inode-Nummer der Datei auf Null gesetzt, der Dateiname bleibt weiterhin in der Verzeichnis-Datei stehen. Das Löschen einer Datei verkleinert also nicht die Verzeichnis-Datei. Beim Erzeugen einer neuen Datei wird der erste Eintrag mit der inode-Nummer = 0 gesucht. Hier werden dann die neue inode-Nummer und der neue Dateinamen eingetragen. Falls keine inode-Nummer = 0 existiert, erfolgt der Eintrag am Ende der Verzeichnis-Datei.

## 9.1 Fragen zu Kommandos und Dateisystemen

- 1.) Worin unterscheiden sich Optionen und Argumente bei Kommandos?
- 2.) Geben Sie zwei Möglichkeiten an, um Hilfen zu Linux-Kommandos zu erhalten.
- 3.) Erläutern Sie die drei Modi des vi-Editors.
- 4.) Wie löschen Sie mit dem vi die nächsten 523 Zeilen?
- 5.) Was versteht man unter *Regulären Ausdrücken*. Warum wurden sie erfunden?
- 6.) Was versteht man unter *Quoting*?
- 7.) Wie gibt man alle Dateinamen eines Verzeichnisses an, die aus genau 10 Zeichen bestehen?
- 8.) Wie gibt man alle Dateinamen eines Verzeichnisses an, die nicht mit einem Großbuchstaben beginnen?
- 9.) Wie löscht man in einem Verzeichnis alle versteckten Dateien?
- 10.) Mit welchem Kommando werden nicht leere Unterverzeichnisse gelöscht?
- 11.) Was machen die folgenden Linux-Kommandos: cat, cp, echo, grep, kill, ls, mkdir, mv, ps, pwd, rm, test
- 12.) Erläutern Sie die 5 Dateiarten bei Linux.
- 13.) Welche besondere Eigenschaft hinsichtlich eines übergeordneten Verzeichnisses hat das *root directory* / ?
- 14.) Geben Sie an, welche Dateien sich bei Linux-Systemen in /etc, /bin, /home, /root befinden.
- 15.) Worin unterscheiden sich die Verzeichnisstrukturen von Linux- und Windows-Systemen?
- 16.) Was haben Verzeichnisstrukturen mit Bäumen zu tun?
- 17.) Verzeichnisstrukturen: Was ist ein Pfad?
- 18.) Wie unterscheidet sich eine relative von einer absoluten Pfadangabe?
- 19.) Was ist eine *Datei*?
- 20.) In welchem Zusammenhang stehen die Begriffe *Datei*, *Daten* und *Dateisystem*?
- 21.) Warum werden Dateien beim Löschen nicht wirklich gelöscht, sondern nur die Verweise auf die Dateiinhalte?
- 22.) Welchen Vorteil hat es, wenn der Dateiname in einer eigenen Datenstruktur (Verzeichnis) und nicht im inode selbst gespeichert wird?
- 23.) In welchen Anwendungsbereichen ist es sinnvoll, eine höhere bzw. eine geringere inode-Dichte zu wählen?
- 24.) Wann ist es sinnvoll, beim Formatieren eines Dateisystems größere log. Blockgrößen (z.B. 4kiB) zu wählen?
- 25.) Was wird bei Windows unter einer Zuordnungseinheit (*Cluster*) verstanden?
- 26.) Berechnen Sie die maximale Dateigröße bei ext2, wenn die Blöcke 2kiB groß sind.
- 27.) \*Moderne Festplatten verwenden intern eine physikalische Blockgröße von 4kiB. Die Festplattenhersteller empfehlen, bei diesen Platten Dateisysteme mit einer logischen Blockgröße von 4kiB zu formatieren. Warum?
- 28.) \*Wie würde es sich auf die Leistung des Dateisystems auswirken, wenn Sie bei modernen Festplatten, die intern eine physikalische Blockgröße von 4kiB haben, beim Formatieren eine kleinere logische Blockgröße von z.B. 1kiB verwenden würden?

## 9.2 Datei-Links

Links sind Verweise auf Dateien. Mit ihrer Hilfe kann von verschiedenen Orten in der Verzeichnisstruktur auf ein- und dieselbe Datei zugegriffen werden, ohne dass die Datei physikalisch mehrfach gespeichert werden muss. Änderungen an der physikalischen Datei sind selbstverständlich für alle darauf gelinkten Dateinamen wirksam.

Linux kennt zwei Arten von Links:

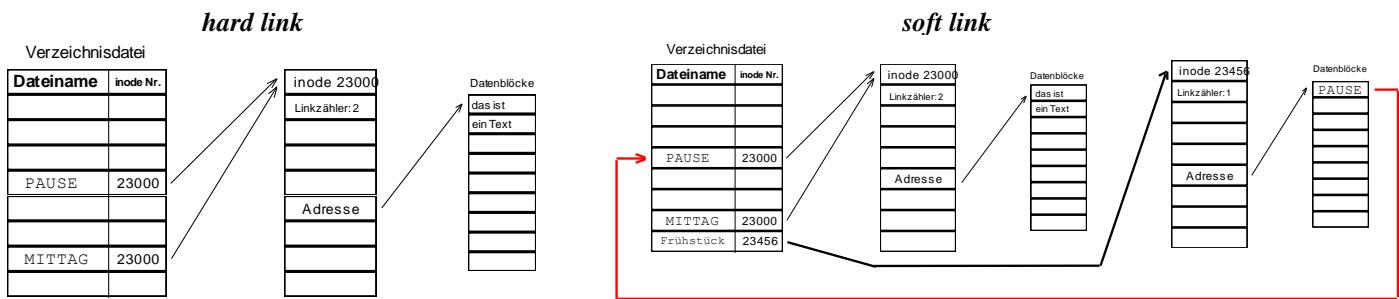
- **Hard-Link** (fester Link, *hard link*)

Beim Erzeugen eines Hard-Links erfolgt ein Directory-Eintrag mit einem neuen Dateinamen und der inode-Nummer der Originaldatei. Hard-Links können nicht auf Dateien zeigen, die sich auf anderen Festplatten befinden. Hard-Links auf Verzeichnisse sind nicht erlaubt. Für Hard-Links wird im entsprechenden inode ein Zähler verwaltet, der angibt, wie viele Hard-Links auf diese Datei bestehen.

- **Soft-Link** (symbolischer Link, *soft link*)

Soft-Links haben den Vorteil, dass sie innerhalb eines Dateisystems von einer physikalischen Platte auf eine andere verweisen können. Außerdem können sie nicht nur auf Dateien, sondern auch auf Verzeichnisse angewendet werden. **ls** zeigt bei symbolischen Links an, wo sich die Originaldatei befindet. Ein Zähler, der angibt wieviel symbolische Links auf eine Datei verweisen, wird nicht verwaltet. Außerdem wird für einen symbolischen Link ein neuer inode mit einer neuen inode-Nummer angelegt.

### Beispiel: hard link und soft link



MITTAG ist ein Link auf PAUSE. Aus diesem Grund sind die inode-Nummern und somit auch die inodes der Dateien MITTAG und PAUSE identisch. Der Dateinhalt existiert nur einmal auf der Platte und kann über den Adresseintrag im inode adressiert werden.

### Beispiel: soft link

Es existiert die Datei PAUSE, auf die ein Hard-Link MITTAG und ein Soft-Link Frühstück erzeugt wird.  
Das Kommando **ls -li** ergibt in diesem Fall folgende Ausgabe:

```
Der Linkzähler wird nur
bei Hard-Links hochgezählt
↓
23000  -rw-rw-rw-  2  user1   users    437 Aug 22 16:40 PAUSE
23000  -rw-rw-rw-  2  user1   users    437 Aug 22 16:40 MITTAG
23456  lrwxrwxrwx  1  user1   users      5 Oct 12 08:23 Frühstück → PAUSE
↑
Die inode-Nummern der Hard-Links
und der Originaldatei sind identisch.
Der Eintrag des Soft-Links
erhält eine eigene inode-Nummer.
↑
Der Verweis auf die Original-Datei
erfolgt nur bei Soft-Links.
```

### Kommando zum Erstellen von Links

#### **ln original link\_name**

Dieses Kommando erzeugt einen Hard-Link mit dem Namen `link_name` auf die Datei `original`.

#### **ln -s original symlink\_name**

Erzeugt einen Soft-Link mit dem Namen `symlink_name` auf die Datei `original`.

Die resultierenden inode-Nummern der Links können mit **ls -li** angesehen werden.

## 10 Dateiarten und Zugriffsrechte

In den Verzeichnissen werden Dateinamen und die dazugehörigen inode-Nummern gespeichert. Im inode sind die weiteren Verwaltungsinformationen einer Datei abgelegt. Viele davon werden mit **ls -l** angezeigt:

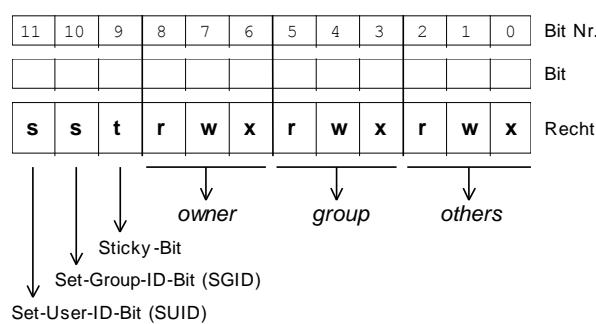
Beispiele zu Dateiarten und Zugriffsrechten (siehe auch Kap. 8.4):

```
-rw-r--r-- 1 root root 1299 Nov 23 12:45 /etc/passwd  
-rwsr-xr-x 1 root shadow 27920 Aug 15 22:33 /usr/bin/passwd  
-rw----- 1 root root 1187 Nov 22 18:33 /etc/shadow  
drwxrwxrwt 1 root root 4096 Aug 15 08:18 /tmp  
lrwxrwxrwx 1 root root 4 Oct 8 10:03 /bin/sh -> bash  
crw----- 1 root root 10, 145 Aug 15 21:52 /dev/hfmodem  
brw-rw-rw- 1 root disk 2, 0 Aug 15 23:41 /dev/fd0  
prw----- 1 root tty 0 Nov 23 02:33 /dev/xconsole
```

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Dateiart</b>            | <ul style="list-style-type: none"> <li>- gewöhnliche Datei, <i>ordinary file</i></li> <li><b>d</b> Verzeichnis, <i>directory</i></li> <li><b>l</b> <i>soft link</i></li> <li><b>c</b> zeichenorientierte Gerätedatei, <i>character device</i> z.B. Tastatur, Terminal, Drucker</li> <li><b>b</b> blockorientierte Gerätedatei, <i>block device</i> z.B. Festplatte, Diskette</li> <li><b>p</b> Named Pipe, dient der Kommunikation zwischen Prozessen</li> </ul>                               |
| <b>Zugriffsrechte</b>      | Für die Benutzerklassen Eigentümer ( <i>owner</i> ) , Gruppe ( <i>group</i> ) des Eigentümers und alle anderen ( <i>others</i> ) jeweils: <ul style="list-style-type: none"> <li><b>r</b> lesen, <i>read</i></li> <li><b>w</b> schreiben, <i>write</i></li> <li><b>x</b> ausführen, <i>execute</i>. Bei Verzeichnissen ist damit das Wechseln in das Verzeichnis erlaubt</li> <li><b>s</b> Set-User-ID-Bit (SUID-Bit) bzw. Set-Group-ID-Bit (SGID-Bit)</li> <li><b>t</b> Sticky-Bit</li> </ul> |
| <b>Referenzzähler</b>      | Gibt die Anzahl der festen Links an                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Benutzer und Gruppe</b> | Eigentümer ( <i>owner</i> ) der Datei und Gruppe des Eigentümers                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Dateigröße</b>          | Angabe in Bytes oder Anzahl der benutzten Blöcke                                                                                                                                                                                                                                                                                                                                                                                                                                               |

- ### Zugriffsrechte

  - Eigentümer/Besitzer (*owner*) ist derjenige, der die Datei erstellt hat. Er verwaltet die Zugriffsrechte.
  - Die Zugriffsrechte werden im inode in drei Gruppen zu jeweils drei Bits gespeichert.
  - Durch Setzen des Set-User-ID-Bit (SUID) bzw. Set-Group-ID-Bit (SGID) können die Zugriffsrechte des Dateibesitzers auf den aktuellen Benutzer bzw. die Gruppe des aktuellen Benutzers übertragen werden.
  - Ist das Sticky-Bit bei einem Verzeichnis gesetzt, auf das mehrere User Schreibrechte haben, so kann jeder User nur seine eigenen Dateien löschen.



## Zugriffsrechte ändern: chmod (change mode)

Syntax: chmod [*Optionen*] *Modus Dateiname*

Mit dem Befehl **chmod** können die Zugriffsrechte einer Datei geändert werden. Im oktalen Modus werden die zu setzenden Zugriffsrechte dem Befehl als 3-stellige Oktalzahl übergeben. Um die Zugriffsrechte für sich selbst, die Gruppe oder für alle anderen zu ändern, muss der Besitzer die entsprechenden Bits setzen oder zurücksetzen.

## Beispiel zu chmod

Folgende Zugriffsrechte sollen vergeben werden:

**Besitzer (owner):** lesen, schreiben, ausführen  
**Gruppe (group):** lesen, ausführen  
**alle anderen (others):** lesen

Das Bitmuster sieht dann wie folgt aus:

## Oktalzahl

Bei der Angabe als Oktalzahl beschreiben die drei Ziffern die Rechte des Besitzers, der Gruppe und der Anderen.

**Beispiel:** Oktalzahl 754<sub>8</sub> → Binärdarstellung (jede Zahl wird einzeln gewandelt): 111 101 100

- 1: Recht ist vergeben
- 0: Recht ist nicht vergeben

$$\rightarrow \quad \begin{array}{c} 7 \\ \underbrace{\phantom{0}}_{1} \end{array} \quad \begin{array}{c} 5 \\ \underbrace{\phantom{0}}_{1} \end{array} \quad \begin{array}{c} 4 \\ \underbrace{\phantom{0}}_{1} \end{array}$$


  
*owner*    *group*    *others*

|   |                            |                             |
|---|----------------------------|-----------------------------|
| → | Rechte des <i>owner</i> :  | <i>read, write, execute</i> |
|   | Rechte der <i>group</i> :  | <i>read, execute</i>        |
|   | Rechte der <i>others</i> : | <i>read</i>                 |

## Buchstabenkennung

Bei der Buchstabenkennung werden folgende Buchstaben und Symbole verwendet:

| Betroffene     | Rechte       | Symbole                                             |
|----------------|--------------|-----------------------------------------------------|
| u user (owner) | r read       | + Berechtigung hinzufügen                           |
| g group        | w write      | - Berechtigung löschen                              |
| o others       | x execute    | = nur diese Rechte werden gesetzt, alle anderen     |
| a all          | s SUID/SGID  | Rechte werden, sofern sie vergeben waren, entzogen. |
|                | t Sticky-Bit |                                                     |

### **Beispiele für Oktal- und Buchstabendarstellung:**

| <b>oktal</b> | <b>Buchstaben</b> | <b>Anzeige mit 1s -1</b> | <b>Bedeutung</b>                                                                                                                                               |
|--------------|-------------------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 640          | u=rw, g=r, o=     | rw-r-----                | Nur der Eigentümer kann die Datei lesen und verändern. Die Gruppenmitglieder können die Datei lesen, alle anderen haben keinen Zugriff.                        |
| 644          | a=r, u+w          | rw-r--r--                | Alle Systembenutzer dürfen die Datei lesen, nur der Eigentümer darf sie verändern.                                                                             |
| 750          | u=rwx, g=rx, o=   | rwxr-x---                | Der Eigentümer kann die Datei lesen, beschreiben und ausführen. Die Gruppenmitglieder dürfen die Datei lesen und ausführen, alle anderen haben keinen Zugriff. |

Die vollständigen Kommandos dafür (z.B. für Dateiname maria.txt) lauten:

```
chmod 640 maria.txt oder chmod u=rw,g=r,o= maria.txt  
chmod 644 maria.txt oder chmod a=r,u+w maria.txt  
chmod 750 maria.txt oder chmod u=rwx,g=rx,o= maria.txt
```

## ACL - Access Control List

Eine Access Control List (ACL, Zugriffssteuerungsliste), ist eine Software-Technik, die Zugriffe auf Dateien eingrenzt. Eine ACL legt fest, welcher Benutzer welche Rechte auf eine Datei hat.

Im Unterschied zu dem einfachen UNIX-Rechtesystem sind ACLs feiner einstellbar.

So können etwa für eine Datei für mehrere Benutzer und Gruppen unterschiedliche Rechte vergeben werden, während die üblichen Zugriffsrechte nur die Rechtevergabe für einen Benutzer, eine Gruppe und den „Rest der Welt“ zulassen. Unter Linux werden ACLs mit den Befehlen **getfacl** und **setfacl** verwaltet.

### Dateibesitz und Gruppenzugehörigkeit einer Datei

### Dateibesitzer ändern: chown (change owner)

Syntax: chown Benutzername Dateiname

### **Beschreibung:**

Der Besitzer einer Datei kann nachträglich mit **chown** verändert werden. Die Datei Dateiname wird an den Benutzer Benutzername verschenkt. Dieser Befehl kann nur von root ausgeführt werden.

## **Gruppenzugehörigkeit einer Datei verändern: chgrp (change group)**

Syntax:            chgrp *Gruppe* *Datei*

### **Beschreibung:**

Das Kommando **chgrp** ändert die Gruppenzugehörigkeit einer Datei. Der Befehl ist dem Eigentümer einer Datei und dem Superuser vorbehalten. Der Eigentümer kann eine Datei nur einer Gruppe übergeben, in der er selbst Mitglied ist.

**Beispiel:** `charp schüler maria.txt` Übergibt die Datei maria.txt an die Gruppe schüler

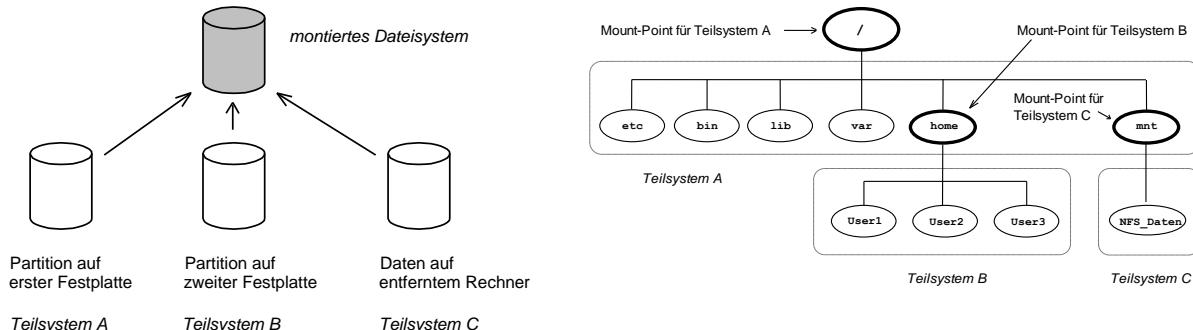
## **10.1 Fragen zu Links, Dateiarten und Zugriffsrechten**

- 1.) Worin unterscheiden sich Hard- von Soft-Links?
  - 2.) Warum sind Hard-Links nur innerhalb der Dateien einer Partition möglich?
  - 3.) Wie kann man sich die Funktion eines Soft-Links vorstellen?
  - 4.) Warum ist bei modernen Betriebssystemen im Dateisystem eine Rechteverwaltung nötig?
  - 5.) Die Shell ignoriert bei Shell-Skripten ein gesetztes SUID-Bit und führt diese unter dem Konto des gerade angemeldeten Benutzers aus. Warum?
  - 6.) Was haben Oktalzahlen mit Linux-Dateirechten zu tun?
  - 7.) Was haben die Windows-Befehle **cacls** bzw. **icacls** mit Zugriffsrechten zu tun?
  - 8.) Wozu wird das *Sticky-Bit* eingesetzt?
  - 9.) Bei Softlinks sind immer alle Rechte gesetzt, also oktal 777. Die Rechte eines Softlinks lassen sich auch mit **chmod** nicht ändern. Versuchen Sie zu erklären, warum dies so ist und welche Rechte effektiv gelten.

# 11 Montierte Dateisysteme

Linux Dateisysteme können über mehrere Partitionen, Platten oder vernetzte Rechner verteilt sein. Das Einhängen dieser Dateisysteme in den Verzeichnisbaum wird montieren (*mounten*) genannt.

**Beispiel für ein montiertes Dateisystem:**



Das Mounten ist die Fähigkeit, Dateisysteme transparent, also ohne die Verwendung von zusätzlichen Gerätbezeichnern (z.B. Laufwerksbuchstaben) in die eigene Verzeichnisstruktur einzubinden.

Mounten ist eine elegante Möglichkeit, über die eigene Verzeichnisstruktur auf fremde oder entfernte Dateisysteme zuzugreifen. Nach dem Mounten präsentiert sich der Dateibaum als homogene Einheit. Zusätzliche Gerätbezeichner, wie die Laufwerksbuchstaben unter Windows sind bei Linux nicht erforderlich. Um die einzelnen Teilsysteme (Datenträger) beim Systemstart automatisch zu mounten, müssen sie in der Datei `/etc/fstab` eingetragen sein.

Unter Windows ist das Mounten von Partitionen in leere Verzeichnisse mit `mountvol` möglich.

## 11.1 Manuelles Mounten

`mount` ohne Parameter zeigt alle gemounteten Dateisysteme an.

Ausgabebeispiel:

```
Das Dateisystem ist lesbar und beschreibbar gemountet  
↓  
/dev/sdb1 on /home type ext2 (rw)  
↑  
Das gemountete Dateisystem ist ein ext2-System  
Die erste Partition der zweiten S-ATA oder SCSI-Festplatte ist am Mount-Point /home gemountet
```

### Mounten von Dateisystemen

```
mount -t Dateisystemtyp Gerätedatei Mountpoint
```

Dieses Kommando hängt das Dateisystem, das durch eine Gerätdatei spezifiziert wird unter dem angegebenen Mount-Point in den Dateibaum ein. Der Mount-Point sollte ein möglichst leeres Verzeichnis sein. Die Option `-t` kennzeichnet die Art des zu mountenden Dateisystems (z.B.: ext2, iso9660, vfat, ntfs-3g, nfs, cifs usw.). Wenn bei lokalen Datenträgern die Option `-t` weggelassen wird, versucht `mount` den Dateisystemtyp selbst zu erkennen. Den erkannten Typ kann man ausgeben lassen, wenn `mount` ohne Argumente eingegeben wird.

Normalerweise kann nur `root` Dateisysteme mounten. Da dies für Disketten, USB-Sticks oder CD/DVDs aber unpraktisch ist, kann man einem normalen User das Mounten dieser Systeme durch einen Eintrag in die Datei `/etc/fstab` erlauben.

### Unmounten von Dateisystemen

Das zu auszuhängende Dateisystem kann entweder durch den *Mount-Point* oder die *Gerätedatei* spezifiziert werden. Nach dem Unmounten sind Zugriffe auf dieses Dateisystem nicht mehr möglich.

```
umount Mountpoint bzw. umount Gerätedatei
```

**Achtung:** Bevor ein Medium (CD/DVD, USB-Stick, Diskette usw.) entnommen wird, muss man es ummounten!

Das Unmounten kann nur der User durchführen, der das Medium auch gemountet hat. Sollen alle User das Dateisystem ummounten können, muss im Options-Feld (4. Spalte) in der Datei `/etc/fstab` `users` statt `user` beim entsprechenden Dateisystem eingetragen werden.

## Mounten einer CD bzw. DVD

```
mount -t iso9660 -o ro /dev/cdrom /media/cdrom
```

- Das Dateisystem von CDs/DVDs ist iso9660 (**-t iso9660**) bei DVDs oft auch UDF (**-t udf**)
- Die Option **-o ro** legt fest, dass das zu mountende Dateisystem nur lesbar ist (*read only*)
- Die Gerätedatei lautet **/dev/cdrom**
- Das Dateisystem wird unter dem Mount-Point **/media/cdrom** (Name willkürlich) eingehängt.

Die Datei **/dev/cdrom** ist keine echte Gerätedatei, sondern ein Link auf die Gerätedatei des verwendeten CD-ROM-Laufwerks. Dieser Link wurde bei der Installation angelegt. Zum Mounten kann man auch die laufwerksspezifische Gerätedatei angeben, z.B. **/dev/hdc** für ein IDE-CD-ROM-Laufwerk, das als Master am zweiten IDE-Controller angeschlossen ist oder **/dev/sr0** für ein S-ATA-DVD-Laufwerk.

**Unmounten der CD/DVD mit `umount /media/cdrom` oder `umount /dev/cdrom`** (Beide Kommandos bewirken, dass die CD/DVD ummountet wird!)

***Bevor man eine CD, DVD oder Diskette mountet, muss man sie ins Laufwerk legen!***

## Mounten eines USB-Sticks

Zuerst mit **fdisk -l** die Gerätedatei des USB-Sticks herausfinden, z.B. **/dev/sdb**

```
mount /dev/sdb /mnt
```

Der USB-Stick wird in das Verzeichnis **/mnt** eingehängt und ab sofort kann über dieses Verzeichnis auf die Daten des Sticks zugegriffen werden.

**Unmounten des USB-Sticks mit `umount /mnt` oder `umount /dev/sdb`**

## Mounten einer Diskette

```
mount -t ext2 /dev/fd0 /media/floppy
```

- Es wird eine Diskette mit einem ext2-Dateisystem (**-t ext2**) gemountet.
- Die Diskette befindet sich im ersten Diskettenlaufwerk (**/dev/fd0**).
- Die Diskette wird in das Verzeichnis **/media/floppy** eingehängt.

Ab sofort kann über das Verzeichnis **/media/floppy** auf die Diskette zugegriffen werden. Dass die Daten, die über dieses Verzeichnis angesprochen werden, sich auf einer Diskette in Laufwerk A befinden, ist für den User uninteressant. Das Verzeichnis **/media/floppy** als Mount-Point für die Diskette in Laufwerk A zu wählen (eine Empfehlung von SUSE) ist willkürlich. Selbstverständlich kann sich jeder ein eigenes Verzeichnis erzeugen oder ein bereits bestehendes benutzen, um eine Diskette zu mounten.

**Unmounten einer Diskette mit: `umount /media/floppy` oder `umount /dev/fd0`**

## Einrichten von Dateisystemen

Das Kommando **mount** kann nur dann ausgeführt werden, wenn auf dem entsprechenden Gerät bereits ein Dateisystem eingerichtet ist. Ist dies nicht der Fall, muss das Medium zuerst formatiert und dann das gewünschte Dateisystem eingerichtet werden. Das Kommando hierzu lautet:

```
mkfs -t Dateisystem Gerätedatei
```

Das Kommando erzeugt ein Dateisystem vom Typ *Dateisystem* auf dem durch *Gerätedatei* spezifizierten Medium. Es kann nur von *root* verwendet werden. Bevor man **mkfs** auf eine Diskette anwenden kann, muss das Medium mit **fdformat Gerätedatei** Low-Level formatiert werden.

### Beispiel 1: Erzeugung eines FAT32-Dateisystems auf einem USB-Stick

```
mkfs -t vfat /dev/sdb1
```

(Zuerst muss mit **fdisk -l** die Gerätedatei des Sticks herausgefunden werden, hier: **/dev/sdb1**)

### Beispiel 2: Erzeugen eines ext2-Dateisystems auf der Diskette in Laufwerk A

```
mkfs -t ext2 /dev/fd0
```

Der Name der Gerätedatei für Laufwerk B: lautet übrigens **/dev/fd1**

### Beispiel 3: Erzeugen eines ext3-Dateisystems auf der Partition **/dev/sda6**

```
mkfs -t ext3 /dev/sda6
```

## 11.2 Automatisches und vereinfachtes Mounten

### Automatisches Mounten über die Datei /etc/fstab

In der Datei /etc/fstab kann man eintragen, welche Dateisysteme beim Hochfahren von Linux automatisch gemountet werden. Für jedes zu mountende Dateisystem wird in die Datei /etc/fstab eine Zeile eingetragen.

Beispiel: `/dev/sda2 / ext2 defaults 1 1`

1. Spalte: Gibt die **Gerätedatei** des Datenträgers an (im Beispiel: /dev/sda2)
2. Spalte: Gibt den **Mountpoint** an, an den der entsprechende Datenträger im Dateisystem montiert werden soll (im Beispiel: / )
3. Spalte: Gibt den **Typ** des zu mountenden Dateisystems an (im Beispiel: ext2)
4. Spalte: Gibt **Optionen** für den Zugriff auf den Datenträger an. Mehrere Optionen müssen durch Kommata getrennt werden. Wichtige Optionen sind:

|                       |                                                                                                                                                                                                                                                                                                                            |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| defaults              | Dient meist Standard Mount-Option und entspricht <code>rw, suid, dev, exec, auto, nouser, async</code>                                                                                                                                                                                                                     |
| noauto                | Dieser Datenträger wird nicht automatisch montiert, er kann jedoch sehr bequem mit <code>mount name</code> ohne Angabe von Optionen montiert werden. Für <code>name</code> ist dabei entweder der Gerätename aus Spalte 1 oder der Directoryname aus Spalte 2 anzugeben. Die fehlenden Optionen werden automatisch ergänzt |
| noexec                | Vorhandene Programme auf dem Datenträger dürfen nicht ausgeführt werden.                                                                                                                                                                                                                                                   |
| ro                    | read only mounten                                                                                                                                                                                                                                                                                                          |
| user                  | Jeder User darf diesen Datenträger mounten, der gleiche User muss den Datenträger wieder ummounten (sinnvoll bei CD-ROM, USB-Sticks u. Disketten)                                                                                                                                                                          |
| users<br>(statt user) | Jeder User darf den Datenträger mounten.                                                                                                                                                                                                                                                                                   |
| umask=n               | Zugriffsbits der Dateien                                                                                                                                                                                                                                                                                                   |
| uid=n                 | Benutzerzugehörigkeit der Dateien                                                                                                                                                                                                                                                                                          |
| gid=n                 | Gruppenzugehörigkeit der Dateien                                                                                                                                                                                                                                                                                           |

Mit den Optionen `gid`, `uid` und `umask` können die Zugriffsrechte der Dateien auf dem montierten Datenträger voreingestellt werden. Das ist bei Dateisystemen sinnvoll, die über keinen Zugriffsrechte-Mechanismus verfügen (z.B. FAT).

5. Spalte: Enthält Informationen für das Programm `dump`

6. Spalte: Gibt an, ob und wie das zu mountende Dateisystem auf Konsistenz überprüft wird.

- 0: Keine Konsistenzprüfung.
- 1: Sollte beim Root-Filesystem angegeben werden (Konsistenzprüfung erfolgt zuerst).
- 2: Sollte bei veränderlichen Dateisystemen angegeben werden.

### Vereinfachtes Mounten über die Datei /etc/fstab

Für das gewünschte Dateisystem erfolgt in der Datei /etc/fstab ein Eintrag wie oben beschrieben. In der Spalte 4 des Eintrags wird die Option `noauto` angegeben. Das bezeichnete Dateisystem ist dadurch für das vereinfachte manuelle Mounten vorbereitet (s.o. Beschreibung Spalte 4).

Das Mount-Kommando für ein so vorbereitetes System lautet entweder `mount Gerätedatei` (Eintrag aus Spalte 1) oder `mount Mountpoint` (Eintrag aus Spalte 2). Weitere Optionen brauchen nicht angegeben werden.

## 11.3 Vertiefung: Windows-Partition unter Linux automatisch mounten

Auf den Übungsrechnern soll die vorhandene Windows-Partition beim Linux-Start automatisch und schreibbar in den Datebaum eingehängt werden. Mount-Point soll `/windows_c` sein.

Dazu sind folgende Arbeitsschritte nötig:

- Gerätedatei und Dateisystem der einzuhängenden Partition bestimmen
- Probeweise Dateisystem mounten
- Datei-Zugriff prüfen
- Mount-Point `/windows_c` erstellen
- In der Konfigurationsdatei `/etc/fstab` eine neue Zeile einfügen, damit das Dateisystem beim nächsten Systemstart automatisch nach `/windows_c` gemountet wird
- System neu starten
- Gemountete Dateisysteme anzeigen
- Datei-Zugriff prüfen

`fdisk -l` zeigt die verfügbaren Festplatten (auch USB) mit allen jeweils darauf befindlichen Partitionen an.

Bei unseren PCs ist die erste Partition vom Typ: HPFS/NTFS

```
mount -t ntfs-3g /dev/sda1 /mnt
```

```
ls /mnt
```

```
mkdir /windows_c
```

```
vi /etc/fstab
```

Zeile mit folgendem Inhalt einfügen:

```
/dev/sda1 /windows_c ntfs-3g defaults 0 0
```

```
shutdown -r now bzw. reboot
```

```
mount
```

```
ls -l /windows_c
```

## Home-Directories der Benutzer auf eigene Partition auslagern

Um durch vollgelaufene Dateisysteme die Systemstabilität nicht zu gefährden, ist es vorteilhaft, wenn die Inhalte des Verzeichnisses `/home` (das die Home-Directories aller Benutzer enthält) auf eine andere Partition verlagert werden. Somit beeinflusst eine volle Partition das Root-Filesystem nicht negativ (siehe Übung).

## Mounten von Windows-Freigaben

Wenn die CIFS-Protokollunterstützung installiert ist (Paket: `cifs-utils`), können Windows-Freigaben unter Linux gemountet werden.

Beispiel 1: Mounten der Freigabe `tools` des Servers SRV50 als angemeldeter Linux-Benutzer nach `/mnt`:

```
mount -t cifs //SRV50/tools /mnt
```

Beispiel 2: Sie können sich auch gegenüber dem Windows-Server als Benutzer `Administrator` ausgeben (Passwort: `geheim`) und die Freigabe `tools` über z.B. die IP-Adresse des Servers einhängen:

```
mount -t cifs -o username=Administrator,password=geheim //192.168.1.50/tools /mnt
```

## Mounten von WebDAV-Verzeichnissen

Wenn die DAVFS- und die FUSE-Protokollunterstützung installiert sind (Paket: `davfs2` und `fuse`), können unter Linux Verzeichnisse von WebDAV-Servern gemountet werden.

Beispiel: Mounten der WebDAV-Verzeichnisse der BSInfo nach `/mnt`:

```
mount -t davfs https://webdav-ad-muenchen.musin.de/intk/ /mnt
```

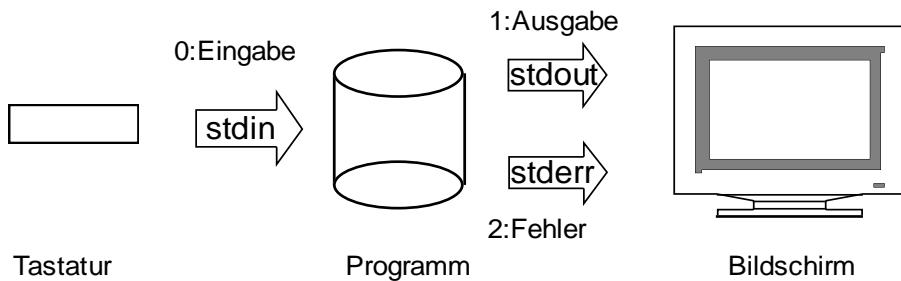
## 11.4 Fragen zum Mounten

- 1.) Was versteht man unter *mounten* (engl. für befestigt, montiert)?
- 2.) Erläutern Sie ausführlich, warum ein gemounteter USB-Stick nicht einfach abgezogen werden sollte.
- 3.) Wieviele Verzeichnisbäume sind bei Linux möglich, wieviele bei Windows?
- 4.) Wo und wie ist der "Bauplan" einer Linux-Verzeichnisstruktur abgelegt.
- 5.) Wie funktioniert das vereinfachte Mounten?
- 6.) Wie können Sie die Gerätedateien aller angeschlossenen Festplatten des Systems herausfinden?
- 7.) \*\*Die erste logische Partition einer Platte hat immer dieselbe Nummer. Welche?

## 12 Ein- und Ausgabeumleitung

Praktisch alle Kommandos/Programme verfügen über einen Eingabekanal und zwei Ausgabekanäle:

- Kanal 0: Standard-Eingabekanal **stdin**  
Der Standard-Eingabekanal ist die Tastatur.
- Kanal 1: Standard-Ausgabekanal **stdout**  
Der Standard-Ausgabekanal ist der Bildschirm.
- Kanal 2: Standard-Fehlerausgabekanal **stderr**  
Der Standard-Fehlerausgabekanal ist der Bildschirm.



### Ausgabeumleitung

Die Ausgabe kann mit den Zeichen > bzw. >> umgelenkt werden

#### Beispiele:

- ls -l > Datei1** Die Ausgabe des Kommandos **ls -l** erfolgt nicht auf dem Bildschirm, sondern in **Datei1**. Wenn **Datei1** noch nicht existiert, wird sie neu angelegt. Eine bereits existierende **Datei1** wird überschrieben.
- ls -l >> Datei1** Der alte Inhalt von **Datei1** wird nicht überschrieben. Die Ausgabe des Kommandos wird an den bestehenden Inhalt von **Datei1** angehängt.

### Eingabeumleitung

Die Eingabe für ein Kommando kann von der Tastatur auf eine Datei umgelegt werden.

#### Beispiel:

Das Kommando **wc** liest Daten vom Standard-Eingabekanal (Tastatur) und gibt die Anzahl aller Zeilen, Wörter und Zeichen auf dem Bildschirm aus. Durch die Eingabeumleitung können beispielsweise die Anzahl aller Zeilen, Wörter und Zeichen der Daten einer Datei ermittelt werden.

**wc < Datei1** gibt die Anzahl der Zeilen, Wörter und Zeichen der **Datei1** auf dem Bildschirm aus.

### Ein- und Ausgabeumleitung

**wc < Datei1 > Datei2** schreibt die Anzahl der Zeilen, Wörter und Zeichen der **Datei1** in die **Datei2**

#### Ausgabeumleitung für Fehlermeldungen:

Trotz einer „normalen“ Ausgabeumleitung in eine Datei werden Fehlermeldungen auf dem Bildschirm ausgegeben. Will man auch die Fehlermeldungen in eine Datei umlenken, muss man dies gesondert angeben.

#### Beispiel:

**wc < Datei1 > Datei2 2> Dateierr**

Der Ausdruck **2> Dateierr** lenkt Fehlermeldungen in die Datei **Dateierr** um. Die **2** von **2>** bezeichnet den Standardkanal 2 (Standard-Fehlerausgabekanal).

Mit dem Konstrukt **ls -l > Ausgabe 2>&1** werden **beide** Ausgabe-Kanäle in die Datei **Ausgabe** umgeleitet!

## Umleitung der Ausgabe eines Kommandos zur Eingabe eines Folgekommandos (Piping)

Für eine administrative Aufgabe gibt es meist genau ein passendes Kommando. Umfangreiche Problemstellungen lassen sich dann oft durch das Verketten entsprechender Einzelkommandos lösen.

Mit Hilfe einer **Pipe** (*pipeline*, Röhre) kann die Ausgabe eines Kommandos direkt als Eingabe eines weiteren Kommandos genutzt werden, ohne dass sie auf dem Bildschirm angezeigt wird. Dieses *Piping* bedeutet, dass der Standard-Ausgabekanal eines Programms mit dem Standard-Eingabekanal des nächsten Programms verbunden wird.

Das Symbol für die Pipe ist: |

### Beispiel:

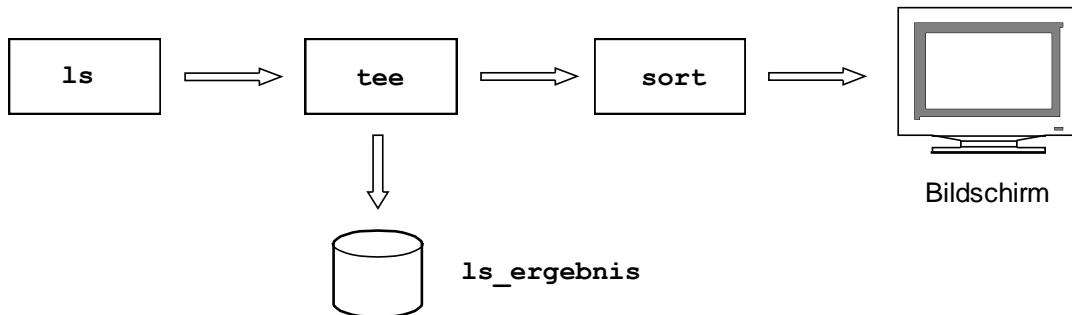
**ls -l | wc** Die Ausgabe des Kommandos **ls -l** wird als Eingabe an das Kommando **wc** übergeben.  
Die Ausgabe von **wc** erfolgt auf dem Bildschirm.

**Datenstrom duplizieren:**      **tee**

Syntax:      **tee [-i] [-a] [Datei1 ...]**

**Beschreibung:** Innerhalb einer Pipe existiert ein Datenstrom, der von Kommando zu Kommando weitergeleitet wird. **tee** ermöglicht es, Zwischenergebnisse (nach einem Kommando innerhalb der Pipeline) abzuzweigen und zu speichern. Die Abarbeitung der Pipe wird dadurch nicht beeinträchtigt.

**Beispiel:**      **ls -l | tee ls\_ergebnis | sort**



Das Ergebnis, das von **ls** erzeugt wird, wird mit Hilfe des Kommandos **tee** in der Datei **ls\_ergebnis** gespeichert. Außerdem wird es an das Kommando **sort** weitergeleitet.

**Hinweis:** Piping und Ein-/Ausgabeumlenkung funktionieren auch bei Windows-Befehlen!

### Weitere Beispiele

|                                                   |                                                                                                              |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| <b>dmesg   more</b>                               | Die Meldungen der Kernel-Initialisierung seitenweise anzeigen                                                |
| <b>dmesg   less</b>                               |                                                                                                              |
| <b>dmesg   grep "Memory"</b>                      | in den Meldungen nach <i>Memory</i> suchen                                                                   |
| <b>cat /etc/passwd   sort   less</b>              | Die Datei /etc/passwd sortieren und seitenweise ausgeben                                                     |
| <b>ps aux   grep inetd</b>                        | überprüfen, ob der Server <i>inetd</i> läuft                                                                 |
| <b>ps aux   grep smbd   grep -v grep</b>          | überprüfen, ob der Samba-Server <i>smbd</i> läuft, <i>grep</i> -Treffer aber unterdrücken                    |
| <b>ls -l   awk '{print \$5}'   sort -n   uniq</b> | ermittelt und sortiert die Dateigrößen im aktuellen Verzeichnis, gleiche Größenangaben kommen nur einmal vor |

**tar cvf - /home/user1/files | ssh -l user1@server "cd /var/files && tar xvf -"**  
den Inhalt des Verzeichnisses /home/user1/files mit tar zu einem Archiv zusammenpacken, über eine SSH-Verbindung verschlüsselt zu einem anderen Rechner senden und dort nach /var/files entpacken

## 12.1 Fragen zur Ein- und Ausgabeumleitung

- 1.) Erläutern Sie die Ein- und Ausgabeumleitung an Beispielen.
- 2.) Mit welcher Konstruktion können beide Ausgabekanäle in dieselbe Datei umgeleitet werden?
- 3.) Was ist eine *Pipe*?
- 4.) Was kann mit dem Kommando **tee** erreicht werden?

## 13 Weitere Kommandos zur Dateiverwaltung und -bearbeitung

Eine kurze Hilfe zu den meisten Kommandos erhält man mit der Option **--help** (z.B. **cp --help**).

### Inhalt von Dateien ausgeben: **cat** (concatenate)

Syntax:      **cat [Optionen] [Dateien]**

#### Beschreibung:

Gibt den Inhalt der Dateien nacheinander aus. Wird keine Datei angegeben, liest **cat** von der Standardeingabe (Tastatur). In diesem Fall wird **cat** mit **Strg+c** abgebrochen.

Wird **cat** auf eine ausführbare Datei angewendet, entsteht Chaos. Mit Hilfe des Befehls **file** kann man vorab prüfen, ob man **cat** besser nicht anwenden sollte.

### Kopieren von Dateien: **cp** (copy)

Syntax:      a)    **cp [Optionen] Quelle Ziel**  
                 b)    **cp [Optionen] Quelle1 [Quelle2, ...] Zielverzeichnis**

#### Beschreibung:

Es können absolute bzw. relative Pfade angegeben werden.

- a) *Quelle* wird unter dem Namen *Ziel* zusätzlich gespeichert.
- b) *Quelle* wird unter Beibehaltung des Namens ins *Zielverzeichnis* kopiert.

**Achtung:** Existiert die Zielfile bereits, wird deren Inhalt überschrieben!

### Löschen von Dateien: **rm** (remove)

Syntax:      **rm [Optionen] Dateien**

#### Beschreibung:

Das Kommando löscht die angegebenen Dateien. Wenn eine angegebene Datei ein Link ist, löscht rm nur den Link. Um eine Datei in einem Directory löschen zu können, benötigt man Schreibrecht in diesem Directory. Für die Datei selbst wird weder Lese- noch Schreibrecht benötigt. Fehlt das Schreibrecht für die Datei, wird vor dem Löschen noch einmal nachgefragt.

### Umbenennen oder Verlagern von Dateien: **mv** (move)

Syntax:      a)    **mv [Optionen] Datei1 Datei2**  
                 b)    **mv [Optionen] Datei1 [Datei2, ...] Verzeichnis**

#### Beschreibung:

Für *Datei1* und *Datei2* können Pfade angegeben werden.

**zu a):** Verweisen *Datei1* und *Datei2* auf das gleiche Verzeichnis (Pfad), erfolgt eine Umbenennung von *Datei1* in *Datei2*. Verweisen *Datei1* und *Datei2* auf unterschiedliche Verzeichnisse, erfolgt eine Verlagerung der *Datei1* in das Zielverzeichnis, das unter *Datei2* angegeben ist. Außerdem erfolgt eine Umbenennung.

**zu b):** Die angegebenen Dateien werden unter Beibehaltung ihrer Namen nach Verzeichnis verlagert.

Syntax:      **file Optionen Dateien**

**Beschreibung:**

**file** überprüft ob die angegebenen Dateien ausführbare Dateien sind. Bei der Ausgabe *data* oder *executable* sollte auf **cat** verzichtet werden. Lautet die Ausgabe *english text*, *ascii text* oder *commands text*, kann **cat** angewendet werden.

## 14 Benutzerverwaltung

Der Systemverwalter **root** wird bereits bei der Installation eingerichtet. Er besitzt alle Rechte im System. Mit den Administrationsrechten kann das System allerdings versehentlich zerstört werden (z.B. **rm -rf /**). Deshalb müssen Benutzer eingerichtet werden, die nur über eine Teilmenge aller Rechte verfügen. Arbeitet der Systemadministrator nicht unter der Benutzerkennung **root**, kann er mit **su -** (*switch user*) und der Eingabe des **root**-Passworts jederzeit zum Administrator werden.

### Die Dateien **/etc/passwd**, **/etc/shadow** und **/etc/group**

In diesen Dateien werden die Informationen über die Benutzer und Gruppen des Systems verwaltet. Sie werden beim Login gelesen und ausgewertet. Sie beinhalten pro Benutzer eine Zeile, die aus mehreren, durch Doppelpunkte getrennten Feldern besteht. Die Feldinhalte sind u.a. Benutzernamen, Passwort, Benutzer-ID, Gruppen-ID, Homeverzeichnis und eine Angabe, welche Shell nach dem Login gestartet werden soll.

Beim Login benötigt ein Benutzer Leserechte auf **/etc/passwd**. Dies stellt jedoch ein Sicherheitsproblem dar, das Angriffe auf das System ermöglichen könnte. Abhilfe leistet das shadow-Passwort-System, das auf den meisten UNIX/Linux-Systemen installiert ist. Die Datei **/etc/shadow** enthält jetzt die verschlüsselten Passwörter und ist nur für **root** lesbar. Beim Programm **/usr/bin/passwd** ist das SUID-Bit gesetzt. Bei seiner Ausführung läuft es unter **root**-Rechten und darf deshalb die Datei **/etc/shadow** lesen.

### Befehle zur Benutzerverwaltung

Echte Hacker editieren die **passwd** und **shadow** -Dateien direkt. Um Probleme zu vermeiden, sollten Sie jedoch Befehle wie **useradd**, **usermod**, **userdel**, **groupadd**, **groupmod**, **passwd** und **id** verwenden.

Die Programme **yast** oder der KDE User-Manager sind benutzerfreundlicher und bieten denselben Leistungsumfang.

### 14.1 Fragen zur Benutzerverwaltung

- 1.) Wie heißt der Dateiname der lokalen Benutzerdatenbank bei Linux-Systemen? Wie bei Windows?
- 2.) In welcher Datei sind die verschlüsselten Passwörter gespeichert?
- 3.) Warum muss beim Programm **/usr/bin/passwd** das SUID-Bit gesetzt sein?

# 15 Prozesse und Prozessverwaltung

## Prozesse

Alle Programme die auf einem Rechner gestartet und von der CPU bearbeitet werden nennt man Prozesse. Die dafür benötigten Verwaltungsdaten werden vom Betriebssystem in eine Prozesstabellle eingetragen.

Mit **ps** (*process status*) bekommt man einen Überblick zu allen laufenden Prozessen.

### Beispiel:

```
user1@486> ps
  PID  TTY  STAT   TIME  COMMAND
 6662  1    S      0:00  -bash
 6673  2    S      0:00  -bash
 6681  2    S      0:00  vi maria.txt
 6689  1    R      0:00  ps
user1@486>
```

Der User hat zwei virtuelle Konsolen geöffnet (1, 2). An beiden Terminals ist die bash-Shell gestartet. Am Terminal 2 wird anschließend der vi gestartet und am Terminal 1 wird das Kommando ps gestartet, dessen Ausgabe im Beispiel angezeigt ist.

|         |                                                                                                            |
|---------|------------------------------------------------------------------------------------------------------------|
| PID     | Prozessnummer ( <i>process identification</i> ), wird vom Betriebssystem beim Start des Prozesses vergeben |
| TTY     | Terminal oder Nummer des Konsolenbildschirms, an dem das Kommando gestartet wurde                          |
| STAT    | Prozesszustand ( <i>state</i> ), S: sleeping, W: waiting, R: running, Z: zombie                            |
| TIME    | Rechenzeit, die bisher vom Prozess verbraucht wurde                                                        |
| COMMAND | Kommandozeile des Programmaufrufs                                                                          |

Mit **ps aux** wird eine Liste von allen Prozessen des Betriebssystems ausgegeben. Die ausgegebene Liste ist meist umfangreich, da die internen Prozesse sehr vielfältig sind, wie beispielsweise der Initialisierungsprozess **init** des Kernels.

|      |                                                                            |
|------|----------------------------------------------------------------------------|
| PPID | Elternprozess ( <i>parent process id</i> ), der diesen Prozess erzeugt hat |
| UID  | Prozess-Eigentümer                                                         |
| NI   | nice-Wert des Prozesses                                                    |

## Prozesshierarchie

Ein Prozess entsteht nicht aus dem Nichts, sondern wird von einem anderen Prozess (Elternprozess) erzeugt. Der Vater aller Prozesse ist der **init**-Prozess mit PID=1 und PPID=0. Die im System entstandene Prozesshierarchie wird mit **pstree** visualisiert.

|                                |                                                                                                                                                                                                                                                                           |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Beispiele für Prozesse:</b> | Virtuelle Konsolen, init-Prozess, im Hintergrund laufende und ständig aktive Prozesse (sog. Dämonen, <i>daemon</i> ). Ihr Dateiname endet deshalb meist mit einem <b>d</b> ( <b>klogd</b> , <b>kswapd</b> , <b>inetd</b> , <b>httpd</b> , <b>nfsd</b> , <b>ftpd</b> ,..). |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Hintergrundprozesse

Die Shell ermöglicht es, Prozesse als Hintergrundprozesse zu starten, wenn man am Ende des Befehls ein **&** anfügt. Beispiel, um den gesamten Dateibaum nach allen Dateien abzusuchen, die den Namen xyz haben:

```
find / -name xyz &
```

Durch das **&** arbeitet der Befehl im Hintergrund und meldet nur noch kurz eine Jobnummer und seine PID.

Die Ein- und Ausgaben eines so gestarteten Prozesses sind jedoch dem Terminal zugeordnet auf dem der Prozess gestartet wurde. Möchte man an diesem Terminal ungestört weiterarbeiten, so kann dies durch eine entsprechende Ein- und Ausgabeumleitung erreicht werden

```
z.B.: find / -name xyz >ergebnis.txt 2>fehler.txt &
```

Ein Vordergrundprozess kann durch **Strg-z** schlafend in den Hintergrund geschickt werden und mit **fg Jobnummer** wieder in den Vordergrund zurückgeholt werden. Die benötigte Jobnummer kann man auch durch den Befehl **jobs** abfragen.

## Systemauslastung

Die Systemauslastung kann mit **ps u** ermittelt werden. Die Felder %CPU und %MEM geben die prozentuale Rechenzeit und Speicherbelastung einzelner Prozesse an. Abgestürzte Prozesse können so erkannt werden.

Mit **top** können ebenfalls wichtige Parameter der Systemauslastung angezeigt werden.

## Prozesskontrolle mit **nice** und **kill**

**nice [-n nice-Wert] [Befehl [Argumente ...]]**

Die Priorität mit der ein Prozess abgearbeitet wird, kann mit dem nice-Wert beeinflusst werden. Der nice-Wert kann von +19 bis -20 reichen, wobei +19 die geringste und -20 die höchste Priorität darstellt. Negative Werte sind nur für **root** erlaubt:

**nice -n 10 nano**

**kill [Signalbezeichnung | -Signalnummer] Prozessnummer**

Um einen (hängenden) Prozess zu beenden wird **kill** verwendet. Dem angegebenen Prozess wird durch eine entsprechende Signalbezeichnung bzw. Signalnummer der Abbruchwunsch (SIGTERM) oder die Aufforderung zur sofortigen Terminierung (SIGKILL bzw. -9) bekanntgegeben:

**kill -9 12345**

## 15.1 Fragen zu Prozessen und Prozessverwaltung

- 1.) Eine wesentliche Aufgabe des Betriebssystems ist die Prozessverwaltung. Was ist ein Prozess?
- 2.) Welchen praktischen Nutzen hat das Priorisieren von Prozessen?
- 3.) Mit *Affinität* ist das feste Zuordnen eines Prozesses auf einen CPU-Kern gemeint. Welche Vorteile hat das?
- 4.) Was unterscheidet ein im Hintergrund gestartetes Programm von einem im Vordergrund gestarteten?
- 5.) Worin besteht aus Sicht des betroffenen Programms der Unterschied zwischen einem Abbruchwunsch und der sofortigen Terminierung dieses Programms?

## Fragen zu Betriebssystem-Grundlagen

- 1.) Erklären Sie die Aufgabe eines Betriebssystems anschaulich mittels eines Schichtenmodells. Verwenden Sie u.a. die Begriffe *Hardware, Kernel und Anwendungsprogramme!*
- 2.) Erstellen Sie eine Grafik, welche die Komponenten eines Linux-Kernels und ihr Zusammenwirken mit den Anwendungsprogrammen und der Hardware des Rechensystems darstellt.
- 3.) \*Was ist ein API?
- 4.) In einem Dokument lesen Sie, dass ein Betriebssystem ein Ressourcenverwalter ist. Erklären Sie anhand eines Beispiels, was dies bedeutet!
- 5.) \*Was hat ein Windows *blue screen* mit der Aufgabe des Betriebssystems als Ressourcenverwalter zu tun?
- 6.) Nennen Sie vier Aufgaben eines Betriebssystemkerns und erklären Sie zwei davon anhand eines Beispiels!
- 7.) Was ist ein *Scheduler*?
- 8.) Was bedeutet *Multitasking* bei Betriebssystemen?
- 9.) Wie funktioniert ein Zeitscheibenverfahren?
- 10.) Sie haben ein Rechnersystem mit mehreren Multicore-Prozessoren. Wie muss ein Anwendungsprogramm programmiert sein, damit es die volle Prozessorleistung nutzen kann?
- 11.) \*\*Fun: Warum ist der *blue screen* bei Windows blau?

## Inhalt

|           |                                                                       |           |
|-----------|-----------------------------------------------------------------------|-----------|
| <b>1</b>  | <b>INFORMATIONSSQUELLEN UND HILFEN ZU LINUX.....</b>                  | <b>2</b>  |
| <b>2</b>  | <b>WAS IST EIN BETRIEBSSYSTEM? .....</b>                              | <b>3</b>  |
| 2.1       | GRUNDLAGEN .....                                                      | 3         |
| 2.2       | DIE AUFGABEN DES BETRIEBSSYSTEMKERNS (KERNEL) .....                   | 3         |
| 2.3       | EIGENSCHAFTEN VON BETRIEBSSYSTEMEN .....                              | 4         |
| <b>3</b>  | <b>DIE LINUX-GESCHICHTE .....</b>                                     | <b>5</b>  |
| 3.1       | UNIX .....                                                            | 5         |
| 3.2       | MINIX .....                                                           | 5         |
| 3.3       | LINUX .....                                                           | 5         |
| 3.4       | FRAGEN ZU BETRIEBSSYSTEM-GRUNDLAGEN .....                             | 5         |
| <b>4</b>  | <b>20 JAHRE LINUX.....</b>                                            | <b>6</b>  |
| <b>5</b>  | <b>DIE ERSTE ANNÄHERUNG.....</b>                                      | <b>10</b> |
| 5.1       | AN- UND ABMELDEN.....                                                 | 10        |
| 5.2       | HERUNTERFAHREN DES SYSTEMS .....                                      | 10        |
| <b>6</b>  | <b>ALLGEMEINES ZU LINUX-KOMMANDOS UND ZUR TERMINALBEDIENUNG.....</b>  | <b>11</b> |
| 6.1       | DIE SHELL .....                                                       | 11        |
| 6.2       | WICHTIGE EIGENSCHAFTEN DER BASH-SHELL .....                           | 11        |
| 6.3       | BEDIENUNG DES TERMINALS .....                                         | 11        |
| 6.4       | VIRTUELLE TERMINALS.....                                              | 11        |
| 6.5       | EINFACHE KOMMANDOS .....                                              | 12        |
| 6.6       | KOMMANDOS MIT ARGUMENTEN.....                                         | 12        |
| 6.7       | HILFEN ZU KOMMANDOS .....                                             | 12        |
| <b>7</b>  | <b>DER EDITOR VI .....</b>                                            | <b>13</b> |
| <b>8</b>  | <b>GRUNDLAGEN DER DATEIVERWALTUNG .....</b>                           | <b>14</b> |
| 8.1       | NAMENSKONVENTIONEN IM DATEISYSTEM.....                                | 14        |
| 8.2       | JOKERZEICHEN UND REGULÄRE AUSDRÜCKE DER SHELL.....                    | 14        |
| 8.3       | EINIGE DER WICHTIGSTEN LINUX/UNIX-KOMMANDOS ZUR DATEIVERWALTUNG ..... | 15        |
| 8.4       | DATEIARTEN .....                                                      | 15        |
| 8.5       | VERZEICHNISSTRUKTUR VON LINUX-SYSTEMEN (FHS) .....                    | 16        |
| 8.6       | BEGRIFFE ZUM DATEISYSTEM .....                                        | 17        |
| <b>9</b>  | <b>EXT2-DATEISYSTEM .....</b>                                         | <b>17</b> |
| 9.1       | INODE .....                                                           | 18        |
| 9.2       | FRAGEN ZU KOMMANDOS UND DATEISYSTEMEN .....                           | 19        |
| 9.3       | DATEI-LINKS .....                                                     | 20        |
| <b>10</b> | <b>DATEIARTEN UND ZUGRIFFSRECHTE.....</b>                             | <b>21</b> |
| 10.1      | FRAGEN ZU LINKS, DATEIARTEN UND ZUGRIFFSRECHTEN.....                  | 23        |
| <b>11</b> | <b>MONTIERTE DATEISYSTEME .....</b>                                   | <b>24</b> |
| 11.1      | MANUELLES MOUNTEN.....                                                | 24        |
| 11.2      | AUTOMATISCHES UND VEREINFACHTES MOUNTEN .....                         | 26        |
| 11.3      | VERTIEFUNG: WINDOWS-PARTITION UNTER LINUX AUTOMATISCH MOUNTEN .....   | 27        |
| 11.4      | FRAGEN ZUM MOUNTEN.....                                               | 27        |
| <b>12</b> | <b>EIN- UND AUSGABEUMLEITUNG.....</b>                                 | <b>28</b> |
| 12.1      | FRAGEN ZUR EIN- UND AUSGABEUMLEITUNG .....                            | 30        |
| <b>13</b> | <b>WEITERE KOMMANDOS ZUR DATEIVERWALTUNG UND -BEARBEITUNG .....</b>   | <b>30</b> |
| <b>14</b> | <b>BENUTZERVERWALTUNG .....</b>                                       | <b>31</b> |
| 14.1      | FRAGEN ZUR BENUTZERVERWALTUNG .....                                   | 31        |
| <b>15</b> | <b>PROZESSE UND PROZESSVERWALTUNG .....</b>                           | <b>32</b> |
| 15.1      | FRAGEN ZU PROZESSEN UND PROZESSVERWALTUNG .....                       | 33        |