mikroelektronik akademie

Linux-Grundkurs

Ziele des Kurses

Der vorliegende Unix-/Linux-Kurs soll zukünftige Anwender auf die tägliche produktive Arbeit vorbereiten. Dies schließt sowohl die Arbeit in grafischen Benutzeroberflächen als auch den Umgang mit der sogenannten Shell, der Kommandozeilenumgebung, ein.

Ausgehend von den täglichen Aufgaben, die man am Rechner zu bewältigen hat, soll der Kurs folgendes ermöglichen:

- Kenntnis grundlegender Unix-Strukturen und -Features
- Fähigkeit, anfängliche Probleme mit Systemmitteln zu bewältigen
- Verzeichnisnavigation und -verwaltung
- Dateiverwaltung und -anzeige
- die Arbeit mit der Kommandozeile (shell)
- Starten und Beenden von Programmen
- effiziente Suche nach Daten
- Textbearbeitung

Was ist Linux?

Linux ist ein weitgehend UNIX-konformes, kostenloses alternatives Betriebssystem für nahezu alle Rechnerplattformen.

Unter IT-Fachleuten erfreut sich Linux einer enorm steigenden Beliebtheit, da es große Kostenersparnisse bringen, fast alle Aufgaben der Computertechnik bewältigen und sehr viel Spaß machen kann. Nicht zu unrecht wird Linux oft als das komfortabelste UNIX bezeichnet.

Linux ist sowohl als Quellcode als auch in Binärform erhältlich und besteht im Wesentlichen aus dem Betriebssystem (dem Kernel, engl. Kern), einer einheitlichen Verzeichnisstruktur und System- und Anwenderprogrammen.

Verschiedene Linuxfirmen, sog. Distributoren (S.u.S.E., Red Hat, DLD, Debian, Slackware etc.), stellen diese Grundkomponenten zu lauffähigen und benutzbaren Systemen zusammen und versehen „ihr“ Linux mit komfortablen Installations-, Wartungs- und Benutzertools. Leistungsfähige Softwarepakete, wie z.B. Webbrowser, Office-Pakete, Tools und Serverprogramme für so gut wie jeden Anwendungszweck, werden meistens vorkonfiguriert mitgeliefert.

Das Copyright auf dem Linux-Grundsystem liegt bei Linus Torvalds (dem „Ur“-Entwickler) und anderen.

Das System ist frei erhältlich und unterliegt den Bestimmungen der GNU (GNU‘s Not Unix) General Public License (GPL). Eine Kopie der GPL gehört zum Quellcode von Linux. Wichtig: „frei“ bezieht sich auf den Zugang zum Quellcode und nicht auf die Kosten.

Es ist absolut legal, Geld für Linux-Distributionen zu verlangen, so lange man auch den Sourcecode mitliefert. Falls man veränderten Code weitergeben möchte, ist man laut GPL verpflichtet, den Quellcode für diese Änderungen mitzuliefern.

Die Geschichte von Unix

UNIX ist in einer Zeit entstanden, in der Datenverarbeitungsanlagen noch ganze Turnhallen füllten und Daten mittels Lochkarten eingegeben wurden. Der enorme Zuwachs an zu bewältigenden Daten und der Wunsch nach mehr Automatisierung bei der Verarbeitung leiteten Mitte der 60er Jahre eine neue Ära der Hard- und Softwaretechnik ein.

1965 Das amerikanische Unternehmen Bell Laboratories beginnt zusammen mit dem MIT, AT&T und General Electric mit der Entwicklung eines neuen Betriebssystems, Multics, das unter anderem Multi-User- und Multi-Prozessor-Fähigkeiten sowie ein hierarchisches Dateisystem mit sich bringen soll. Ehrgeiziges Ziel des Projekts ist, ein System zu schaffen, das ganz Boston mit genug Rechen- und Verarbeitungsleistung bedienen soll.

1969 AT&T ist unzufrieden mit den Entwicklungsarbeiten und zieht sich aus dem Projekt zurück. Einige ehemalige Mitarbeiter der Bell Labs, die schon an der Entwicklung von Multics beteiligt waren, insbesondere Ken Thompson, Dennis Ritchie, Rudd Canaday und Doug McIllroy, stellen für AT&T die erste Version von UNIX inklusive einiger kleinerer Programme vor - lauffähig auf einer PDP-7.

1. Januar 1970 Die offizielle Geburtsstunde von UNIX. Dieser Zeitpunkt stellt auch eine Referenz für alle heutigen Unixsysteme dar, da die Systemzeit intern als Sekundenanzahl seit diesem Tag geführt wird.

1971 Unix läuft mittlerweile auf einer PDP-11 mit 16Kbyte Hauptspeicher, davon 8Kbyte für Userprogramme, und einem 512Kbyte großen Festspeicher.
Erstes reales Anwendungsgebiet ist die Textverarbeitung in der Patentabteilung der Bell Labs, eine Aufgabe, die das System so gut meistert, daß weitere Entwicklungsarbeiten betrieben werden. Unter Programmierern sprechen sich folgende vielversprechenden Features herum:

- enthaltene Entwicklungsumgebung
- einfaches Userinterface
- einfache Utilities, die in Kombination auch komplexe Aufgaben erledigen können
- hierarchisches Filesystem
- einfacher Zugriff auf angeschlossene Geräte, analog zu Dateien
- Multi-User und -Prozess-Fähigkeiten
- Transparenz und Plattformunabhängigkeit

1973 UNIX wird zum größten Teil auf Dennis Ritchies neue Programmiersprache C portiert. Die daraus resultierende Plattformunabhängigkeit und Portierbarkeit macht UNIX immer gefragter.

1974 AT&T, denen der kommerzielle Vertrieb von UNIX in den sechziger Jahren per Gerichtsbeschluß untersagt worden war, stellt den Sourcecode Universitäten kostengünstig zur Verfügung. Akademische Forschung treibt die Entwicklung UNIX-artiger Systeme stark voran.

1977 Weltweit existieren bereits ca. 500 UNIX-Systeme.

1980 Ein Fachbereich der Berkley University, Berkeley Software Development, gibt BSD 4.1 heraus.

1983 SunOS von Sun Microsystems, BSD 4.2 und SysV/UNIX 7 von AT&T erscheinen.

1984 Weltweit ca. 100.000 UNIX-Systeme unterschiedlicher Architekturen und Fähigkeiten im Einsatz.

1987 Professor Andrew S. Tanenbaum implementiert zu Lehrzwecken ein zu UNIX Version 7 kompatibles Minimal-UNIX namens Minix. Für geringe Lizenzgebühren stellt er auch den Quelltext zur Verfügung.

1988 AT&T und Sun Microsystems entwickeln gemeinsam System V Releas 4 (SVR4), das später als Grundlage für UnixWare und Solaris 2 dienen wird.

1991 Linux Torvalds, ein finnischer Informatikstudent, schreibt unter Zugrundelegung der Minix-Sourcen ein neues minimales Betriebssystem, um seinen 386er genauer zu verstehen. Den Sourcecode legt er unter der GPL (GNU Public License, GNU: GNU‘s Not Unix) offen.

Mitte der 90er Weltweit finden sich zigtausende interessierte Entwickler, die über das aufkommende Internet zusammen an der Weiterentwicklung des neuen Systems arbeiten. Es wird von Anfang an Wert auf kostenlose allgemeine Verfügbarkeit, frei verfügbare Sourcen und weitestgehende Kompatibilität zu bestehenden UNIX-Varianten gelegt. Auch die Integration in heterogene Netzwerke wird forciert. Die Resultate dieser innovativen, offenen Entwicklung können sich sehen lassen: Für jeden erdenklichen Zweck wird innerhalb kürzester Zeit Software entwickelt, die zumeist ebenfalls kostenlos im Internet angeboten wird.

Ende der 90er Kommerzielle Unternehmen erkennen das Potential von Linux und beginnen mit der Entwicklung bzw. Portierung eigener Software. Auch kommerzieller Support steht nun zur Verfügung. Linux-Firmen, die den Schritt an die Börse wagen, werden mit Milliarden überschüttet.

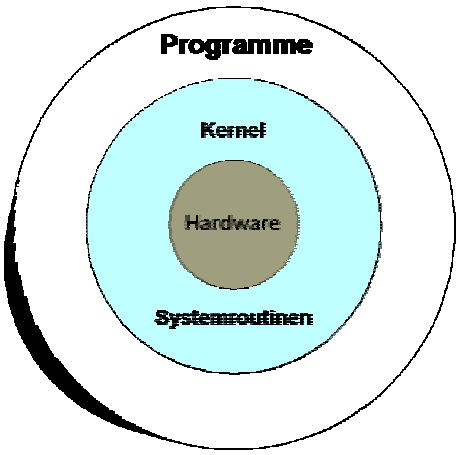
Heutzutage sind UNIX-Systeme die erste Wahl, wenn es um hochverfügbare und weitgehend automatisierbare Serversysteme geht. Zirka die Hälfte aller weltweiten Internetserver sind UNIX- oder Linux-Systeme. Aber auch auf dem Desktop- und Workstation-Markt haben sich UNIX-artige Systeme etabliert -

insbesondere Linux erfährt derzeit einen kräftigen Boom, da es kostenlos erhältlich ist und mittlerweile als extrem stabil und ausgereift gilt.

Grundlegende Systemstrukturen

Das Betriebssystem

UNIX ist ein schichtweise aufgebautes System:



Die innerste Schicht stellt die **Hardware** dar, die dem Betriebssystem Ressourcen wie Rechenzeit, Haupt- und Festspeicher oder Mensch-Maschine-Interfaces zur Verfügung stellt.

Das Betriebssystem, die nächste Schicht, in Unixkreisen als **Kernel** (Systemkern) bekannt, interagiert direkt mit der Hardware und abstrahiert den Zugriff darauf für Userprogramme.

Diese Abstraktion wird durch standardisierte Systemroutinen (**System Calls**) erreicht.

Programme unter Unix, die dritte Schicht, nutzen diese Routinen zur Anforderung bestimmter Hardware-Ressourcen, die vom Kernel bedient werden. Gängige System Calls sind z.B. Dateioperationen (öffnen, lesen, schreiben, schließen), Sicherheitseinstellungen (Setzen von Zugriffsrechten auf Dateien, Geräte) und Prozeßsteuerung (Programme starten, stoppen, Rechenzeitverteilung).

Durch diese Schichtenarchitektur wird ein hoher Grad an Plattformunabhängigkeit erzielt, da zur Portierung auf andere Rechnerarchitekturen im Idealfall nur die Kernelzugriffe auf die spezielle Hardware angepaßt werden müssen.

Das Sicherheitskonzept

Unix ist ein **Multi-User-** und **Multi-Tasking-System**.
Das bedeutet, daß auf einem Rechner gleichzeitig mehrere Benutzer voneinander unabhängig und ungestört arbeiten können, und daß beliebig viele von den Benutzern gestartete Programme scheinbar gleichzeitig abgearbeitet werden. Der Kernel kümmert sich dabei um die Vergabe der benötigten Ressourcen.

Gleichzeitig wird aber bei derartigen Systemen eine **Sicherheitskontrolle** unabdingbar, die je nach Benutzerberechtigungen Zugriff auf bestimmte Dienste, Dateien, Geräte, Programme o.ä. gewährt oder ablehnt. Nur so kann ein reibungsloser Betrieb ohne gegenseitige Störung gewährleistet werden.

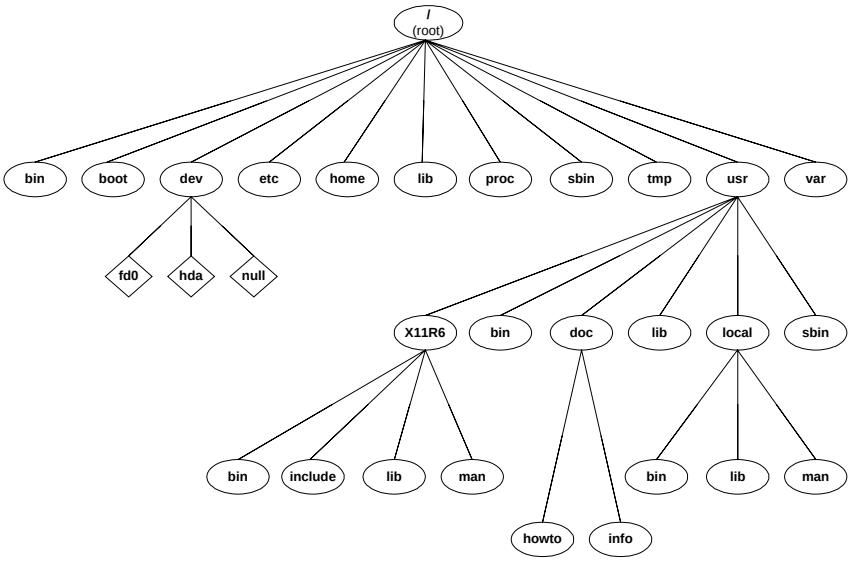
Unix kategorisiert die Benutzerberechtigungen nach **Usernamen**, dem **Eigentümer** und der **Gruppenzugehörigkeit** von Dateien und Prozessen und individuell setzbaren **Dateiberechtigungen** (lesen, schreiben, ausführen, hineinwechseln, u.a.). Durch diesen Ansatz wird bei allen laufenden Programmen, Daten und Ressourcen sichergestellt, daß es keine ungewollten Zugriffe oder Eingriffe in die Arbeit des einzelnen Users geben kann.

Eine Kleinigkeit gilt es noch zu beachten:
Auf jedem Unix-System gibt es den **Benutzer root**, der über alle Berechtigungen erhaben ist.

Der User root wird vom Administrator des Systems zu Wartungs- und Konfigurationszwecken eingesetzt, und er darf alles! Daher sollte das root-Paßwort sehr gut geschützt, niemals an andere Personen weitergegeben und unter spezieller Berücksichtigung der weiter unten genannten Paßwortkonventionen ausgewählt werden.

Das Dateisystem

Das Unix-Dateisystem sieht wie eine umgedrehte Baumstruktur aus:



In ihr können unterschiedliche Arten von Objekten enthalten sein, also sowohl Dateien, Verzeichnisse und Unterverzeichnisse, als auch Hardware-Geräte, die getreu dem Unix-Motto „*alles ist eine Datei*“ an definierten Stellen im Dateisystem abgebildet werden.

Beginnend im **Root-Verzeichnis**, dargestellt durch den Schrägstrich (slash) **/**, kann so hierarchisch jedes Objekt im Dateisystem über einen definierten **Pfad** angesteuert werden. Pfade können absolut oder relativ zum aktuellen Verzeichnis angegeben werden.

Absolute Pfade werden mit dem führenden **/** eingeleitet, folgen dem Verlauf der „Äste“ des Dateisystembaumes, jeweils getrennt durch /, bis die gewünschte Position erreicht ist, also z.B. /home/Marketing/Dokumente/Mailings/April-2000.pdf

Relative Pfade werden ohne führenden **/** angegeben. Sie bezeichnen den Weg ausgehend vom aktuellen Arbeitsverzeichnis hin zu anderen Positionen im Dateisystem.

Angenommen, wir befinden uns im Verzeichnis /home/karl und möchten den oben genannten Pfad erreichen, würde uns der relative Pfad **..**/Marketing/Dokumente/Mailings/April-2000.pdf zum Ziel führen.

An dieser Stelle müssen nun zwei spezielle Verzeichniseinträge vorstellt werden:

- **.** Platzhalter für das aktuelle Verzeichnis
- **..** Platzhalter für das übergeordnete Verzeichnis, in dem das aktuelle Verzeichnis enthalten ist

In unserem Beispiel bewegen wir uns also zunächst eine Ebene nach oben (**..**), um vom darüberliegenden Verzeichnis /home in die darunterliegenden Verzeichnisse Marketing/...usw. wechseln zu können.
Durch die Angabe relativer Pfade kann eine Menge Tipparbeit gespart werden.

Eine weitere Besonderheit im Umgang mit dem Dateiysystem stellt das Tildezeichen (~) dar. Die **Tilde ~** ist ein Platzhalter für den absoluten Pfad zum **Homeverzeichnis** des aktuellen Users, also dem Platz zur Speicherung und Verwaltung eigener Daten.

Dateinamen dürfen unter Unix bis zu 255 Zeichen lang sein, sie sollten jedoch keine Umlaute und Sonderzeichen enthalten, um Kompatibilität mit anderen Systemen zu gewährleisten. Generell gilt: GROSS- und klein-Schreibung wird unterschieden.

Versteckte Dateien sind möglich. Ihr Dateiname muß mit einem Punkt (**.**) beginnen.

Das Unix-Dateisystem erlaubt desweiteren Verknüpfungen, sog. **Links**. Sie können vielfältig eingesetzt werden, ähnlich wie unter anderen Systemen wie Windows, um schnelleren Zugriff auf Daten zu erhalten.

Ein Verzeichnis unter Unix ist nichts weiter als eine Datei, in der eine Liste aller enthaltenen Dateien und Verzeichnisse steht, und in der diese Datei- und Verzeichnisnamen mit den Nummern der dazugehörigen **Inodes** in Zusammenhang gebracht werden. Inodes sind spezielle, für den User unsichtbare Dateien, aus denen der Kernel Informationen über jede einzelne Datei im Dateisystem entnimmt, also z.B. Berechtigungen, Eigentümer, Gruppenzugehörigkeit, Erstell-, Änderungs- und letztes Zugriffsdatum und die physikalische Position der Datei auf dem Speichermedium.

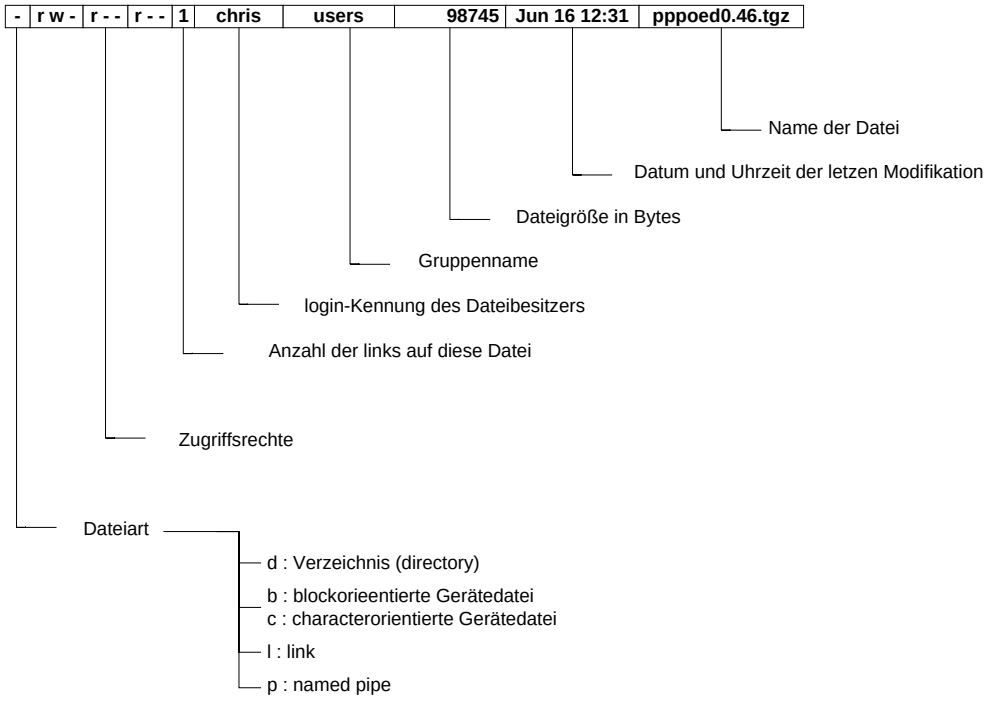
Der Verzeichnisbaum ist in seinen Grundzügen standardisiert, so daß auf unterschiedlichen Unix-Systemen bestimmte Dateien, die zur Verwaltung und Bedienung des Systems dienen, immer dieselbe Position im Dateisystem haben. Folgende **wichtige Verzeichnisse** werden einem während der Arbeit mit Unix immer wieder begegnen:

/home/...	Homeverzeichnisse der Benutzer. In seinem Homeverzeichnis darf der Benutzer alles
/bin, /usr/bin	Platz für Systemtools, die auch für normale Benutzer zugänglich sind (z.B. finger, cat, date)
/sbin, /usr/sbin	Hier sind Administrationstools untergebracht, die meistens nur von root ausführbar sind
/etc	Verzeichnis, das alle Konfigurationsdateien der installierten Software enthält
/lib, /usr/lib	Funktionsbibliotheken (Libraries), die vom System und der Software genutzt werden
/var	Variable Daten, also Druckjobs, Logdateien o.ä. werden hier zwischengespeichert
/tmp	Verzeichnis für temporäre Daten. Hier darf i.d.R. jeder User lesen und schreiben

Dateiberechtigungen

Unix-Systeme klassifizieren die Benutzer nach Username und Gruppenzugehörigkeit.
Alle Objekte im Dateisystem haben einen Eigentümer und sind einer Gruppe zugeordnet.

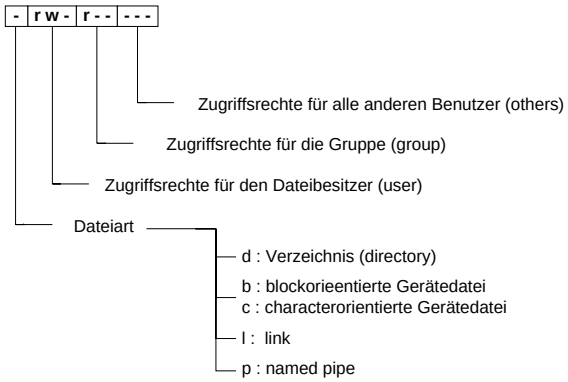
Die folgende beispielhafte Zeile eines Verzeichnislistings zeigt, wie Datei-Informationen und -Berechtigungen unter Unix dargestellt werden:



Um nun gezielt den Datenzugriff für bestimmte User oder Usergruppen zu kontrollieren, kennt Unix zu jedem Dateisystemobjekt drei Gruppen von Berechtigungsattributen:

- Rechte des Eigentümers
- Rechte der Gruppe, zu der die Datei gehört
- Rechte aller anderen User

Sie werden als zehnstellige **Rechtemaske** angegeben (s. Abb.), die aus einem Typattribut und drei Gruppen zu je drei Rechte-Attributen zusammengesetzt wird, also z.B.



Das erste Zeichen gibt den Typ der Datei an, die erste Dreiergruppe die Berechtigungen des Eigentümers, die zweite die der Gruppe und die dritte die aller anderen Benutzer.

Der Typ der Datei wird vom System automatisch korrekt gesetzt und kann vom User nicht geändert werden.

Die Rechteattribute bewirken bei Verzeichnissen und Dateien teilweise unterschiedliche Konsequenzen. Sie können mit folgenden Kürzeln jeweils für den Eigentümer, die Gruppe und die Anderen (Others) gesetzt sein:

r read
Datei ist lesbar bzw. Verzeichnisinhalt darf gelistet werden

w write
Datei darf geändert werden bzw. im Verzeichnis darf geschrieben werden

x execute
Datei ist ausführbar bzw. in das Verzeichnis darf hineingewechselt werden

\- Rechteattribut nicht gesetzt

Im obigen Beispiel darf so der Eigentümer lesen und schreiben, die Gruppe darf lesen, und alle anderen, davon nicht eingeschlossenen User dürfen gar nichts.

Die Shell

Nach erfolgreicher Anmeldung am System wird für jeden Anwender ein Kommandointerpreter, die Shell, gestartet. Jeder User hat solch eine vom Systemadministrator voreingestellte **Login-Shell**.

Während des Starts der Shell werden verschiedene user- und gruppenspezifische Konfigurationsskripte, **login scripts** genannt, abgearbeitet, die die Shellsitzung mit verschiedenen Optionen, wie z.B. dem gewünschten Terminaltyp, der lokalen Sprache, dem richtigen Zeichensatz, etc. initialisieren. Diese Konfiguration geschieht meistens durch Setzen von speicherresidenten **Environment-Variablen**, die von anderen Programmen ausgelesen werden. Die Konfigurationsskripte lassen sich vom Benutzer anpassen.

Nach dem Start meldet sich die Shell mit dem **Prompt**, der Eingabeaufforderung.

Es gibt mehrere Varianten der Shell, die im Laufe der Zeit entstanden sind, und die unterschiedliche Möglichkeiten bieten - Näheres dazu im Kapitel „Weitergehende Informationen zur Shell“.

Die Shell ist ein ganz normales Programm, das auch durch Aufruf ihres Namens gestartet werden kann.
Sie nimmt Eingaben des Anwenders mehr oder weniger komfortabel entgegen (mit oder ohne Zeileneditor, Kommando-History, etc.), führt Kommandos aus und vermittelt zwischen dem Benutzer, Programmen und dem System. Zudem bieten fast alle Shells eigene Möglichkeiten und Kommandos zur Kommandoverkettung, Skriptprogrammierung (Schleifen, Wenn/Dann-Entscheidungen, etc.) und zur Prozeßsteuerung.

Unter Linux hat sich als Standardshell die Bourne-Again-Shell oder kurz bash etabliert.
Die bash bietet viele Vorteile bei der Arbeit in der Kommandozeile, da sie alle Editorfunktionen, eine komfortable Dateinamenerweiterungsfunktion und viele andere nette Details beinhaltet.

Programme, Tools und Skripte

Unter Unix werden ausführbare Dateien nicht wie unter anderen Betriebssystemen durch bestimmte Dateiendungen wie .exe, .com, .bat o.ä. gekennzeichnet. Sie können, wie jede andere Datei auch, frei benannt werden, müssen dem System allerdings als ausführbar bekanntgemacht werden.
Dies geschieht durch das Setzen geeigneter Dateiberechtigungen, die für den Eigentümer, die Gruppe oder andere das x-Recht (x für executable, ausführbar) einschließen. Der Start dieser ausführbaren Dateien wird über den Aufruf ihres Namens in der Shell oder aus anderen Programmen heraus erreicht.

Außer kompiliertem Objektcode sind die sog. Shellskripte (Folgen von Kommandos, Batchfiles) und Skripte in interpretierenden Programmiersprachen (z.B. Perl) ausführbar, sofern sie das x-Recht gesetzt haben.

Eine große Stärke des Unix-Konzeptes ist die Fähigkeit, mehrere kleine Expertenprogramme, die eine Teilaufgabe perfekt und so flexibel wie möglich lösen können, zu umfassenden Systemen zu kombinieren.
So muß nicht immer wieder das Rad neu erfunden werden, und das spart ungemein viel Zeit.

Die grafische Benutzerumgebung X-Window

Aufbauend auf dem Textmodus jedes Linuxsystems läßt sich eine leistungsfähige grafische Benutzeroberfläche einrichten, das **X-Window**-System. X-Window wurde vom MIT (Massachusetts Institute of Technology) entwickelt und bildet die grafische Standardschnittstelle auf UNIX- bzw. Linux-Rechnern.

Es existieren verschiedene, teils kommerzielle Implementierungen des X-Standards.
Unter Linux ist **XFree86** die beste Wahl, da dieses X-Window-Paket kostenlos erhältlich ist und umfassende Hardwareunterstützung mit sich bringt.

X-Window bietet mit Windows, MacOS oder anderen grafischen Benutzeroberflächen vergleichbare Funktionen zum Starten und Stoppen von Software, zur Darstellung laufender Programme in Fenstern, zum Zugriff auf Daten und Geräte und vieles mehr.

Das X-Window-System besitzt drei besondere Eigenschaften, die es von den anderen herkömmlichen grafischen Benutzeroberflächen unterscheidet. Dazu zählen die Konzipierung und Realisierung als offenes System, die **Client/Server-Architektur** sowie als wichtigstes Merkmal die sogenannte **Netzwerktransparenz**.

Durch die Herstellerunabhängigkeit und durch die freie Verfügbarkeit der kompletten Quelltexte konnte das X-Window-System im Gegensatz zu anderen kommerziellen grafischer Oberflächen von Anfang an als offenes System konzipiert werden. Neben Standardschnittstellen, die eine komfortable Entwicklung portabler und hardwareunabhängiger Software erlauben, unterstützt das System auch Schnittstellen für herstellerspezifische Erweiterungen, um somit auch die Anbindung von spezieller Hardware zu ermöglichen.

Das X-Window-System unterscheidet aufgrund seiner spezifischen Architektur zwischen dem **X-Server** und den **X-Clients**. Der X-Server stellt ein Programm dar, das hardwareabhängig einen grafischen Bildschirm steuert. Zusätzlich stellt er das Bindeglied zwischen dem Benutzer und den verschiedenen X-Applikationen, den sogenannten X-Clients, dar, indem er Tastatureingaben bzw. Mausbewegungen des Benutzers an die entsprechenden X-Clients leitet und die von ihnen zurückgelieferten, hardwareunabhängigen Ausgabeinformationen grafisch darstellt.

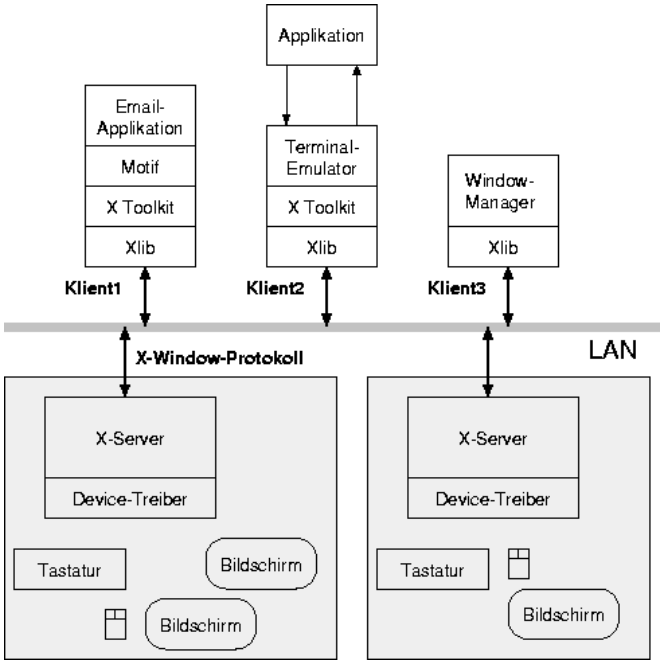
Für das optische Erscheinungsbild des Linux-Grafiksystems ist desweiteren eine noch darüber angesiedelte Softwareschicht zuständig, der **Windowmanager**. Er bringt das individuelle Aussehen der Bedienelemente (Fenster, Buttons, Farben, ...) mit sich und macht das Look-and-Feel der Oberfläche aus. Auf einem Linux-

Rechner können mehrere Windowmanager installiert sein. Sie können in der Regel vom User frei gewählt werden.

Die Kommunikation zwischen dem Server und den Clients erfolgt ausschließlich über das standardisierte X-Protokoll. Dieses Protokoll ist so flexibel ausgelegt, daß es möglich wird, den X-Server und die X-Clients nicht nur lokal auf einem Rechner zu halten, sondern sie auf beliebige Rechner in einem Netz zu verteilen. Durch diese Netzwerktransparenz besteht somit die Möglichkeit, rechenintensive Programme auf Maschinen mit den leistungsstärksten Prozessoren laufen zu lassen, Ein- und Ausgabe jedoch auf einer normalen Workstation im Netz durchzuführen. Zu bedenken ist allerdings, daß das X-Protokoll eine sehr hohe Belastung für das Netzwerk darstellt. Aus diesem Grund sollte man, um Bandbreite zu sparen, nur einzelne Anwendungen/Fenster über das Netz auf den eigenen Rechner umleiten und nicht gleich die gesamte Anmeldung an einem entfernten Rechner vornehmen.

Die Konzeption als offenes System, die Client/Server-Architektur und die Netzwerktransparenz sind natürlich auch integraler Bestandteil des XFree86-Systems, so daß dem Linux-Anwender mit XFree86 ein hervorragendes X-Window-System zur Verfügung steht, das den Vergleich mit kommerziellen Systemen nicht scheuen muß.

Schematisch läßt sich die X-Window-Architektur folgendermaßen darstellen:



Um die Grafikausgabe von X-Programmen über TCP/IP-Netzwerke auf andere Rechner umzuleiten, auf denen ebenfalls ein X-Server für die Darstellung laufen muß, meldet man sich über das Netzwerk an dem Rechner an, auf dem das gewünschte Programm ablaufen soll. Der zur Darstellung zu verwendende X-Server wird dem ausführenden Rechner über einen Identifikations-String mit folgenden Komponenten mitgeteilt:

<Rechnername/IP-Adresse>:*<Servernummer (meist 0)>*.*<Display-Nummer (meist 0)>*

Diese werden in der Environment-Variable DISPLAY eingetragen, also z.B.

setenv DISPLAY "pc12:0.0" (in der C-Shell)

oder

export DISPLAY="pc12:0.0" (in der sh bzw. bash)

Anschließend kann das gewünschte Programm auf dem entfernten PC gestartet werden, die Grafikausgabe wird auf den durch o.g. DISPLAY-Variable bezeichneten X-fähigen Rechner umgeleitet.

Dieses Konzept birgt grundsätzlich einige Sicherheitsrisiken. Ohne geeignete Sicherheitsmechanismen würde so jeder Anwender Programmausgaben gegen den Willen der Anwender auf andere PCs schicken können.

Um dies zu verhindern, überprüft der anzeigende X-Server anhand einer lokalen Liste von Rechnernamen, IP-Adressen oder Usernamen, ob vom entfernten Host X-Window-Umleitungen erwünscht sind.
Diese Access-Liste kann vom Benutzer des Systems mit dem "xhost"-Kommando beeinflußt werden.
Die Syntax lautet wie folgt:

xhost + / - <Rechnername/IP-Adresse>

xhost + gewährt X-Zugriffe für den folgenden Rechner, der Schalter - verbietet sie.

also z.B.

% xhost + pc12 um X-Umleitungen von Rechner pc12 zu akzeptieren.

Damit sind aber längst nicht alle Sicherheitslecks geschlossen. Man sollte sich immer im Klaren darüber sein, daß Remote-X-Sitzungen Unmengen von unverschlüsselten Informationen über das Netzwerk schicken. Darin können natürlich auch Klartextpaßwörter oder ähnliche sensible Informationen enthalten sein - es ist also, wie bei allen Netzwerkverbindungen, Vorsicht angesagt!

Arbeiten in der Shell

Grundlegende Funktionen

Hilfe anfordern

man pages (Online-Handbuch)

Das Unix-Online-Handbuch, die sog. **man pages** (von engl. Manual, Handbuch), ist auf allen Unix-Systemen verfügbar, um den Benutzern schnellen Zugriff auf Hilfetexte zu ermöglichen. Zu den meisten Programmen existieren man pages, teilweise auch in mehreren Sprachen.

Um Hilfe zu einem Kommando anzufordern, genügt es i.d.R., das Kommando **man** einzugeben, gefolgt vom Kommando, über das man informiert werden möchte.

Syntax

man [Kategorie] [Optionen] Kommandoname

Nützliche Optionen

-k Stichwort	alle man pages zu gewünschtem Stichwort auflisten
-a	alle Manpages zum gewünschten Kommando nacheinander anzeigen

Beispiele

man kann dazu benutzt werden, Kommandos, deren Beschreibung ein gewünschtes Stichwort enthält, herauszusuchen und anzuzeigen.
Hier alle Befehle, die etwas mit password zu tun haben:

```
% man -k password
chage (1)        - change user password expiry information
passwd (1)       - change user password
passwd (5)       - The password file
```

Die in Klammern nachgestellten Nummern oder Buchstaben stehen für die Kategorie der man page.
man pages werden unter Unix in folgende **Kategorien** eingeteilt:

1	Ausführbare Programme oder Shellbefehle
2	Systemaufrufe (Kernelfunktionen)
3	Bibliotheksaufrufe (Funktionen in System-Bibliotheken)
4	Spezielle Dateien (gewöhnlich in /dev)
5	Dateiformate und Konventionen, z. B. /etc/passwd
6	Spiele
7	Makropakete und Konventionen, z. B. man(7), groff(7)
8	Systemadministrationsbefehle (in der Regel nur für root)
9	Kernelroutinen
n	neu
l	lokal
p	öffentlich
o	alt

Für ein Stichwort können auch mehrere man pages erhältlich sein, wenn z.B. ein Kommando und eine Datei gleichen Namens, für die es ebenfalls eine man page gibt, auf dem System installiert sind.

Im obigen Beispiel gibt es zwei man pages für „passwd“: passwd (1), das Kommando, und passwd (5), die Paßwortdatei.

Ohne Angabe der Kategorie wird man stets die man page mit der geringsten Nummer anzeigen.

Um in diesem Fall z.B. die man page zur Paßwortdatei passwd (5) anzuzeigen, genügt der Befehl:

```
% man 5 passwd
PASSWD(5)          Dateiformate          PASSWD(5)
```

BEZEICHNUNG
passwd - Paßwort-Datei

BESCHREIBUNG
Passwd ist eine ASCII-Datei, die eine Liste der Benutzer des Systems und deren Paßwörter enthält. Die Paßwort-Datei sollte für alle Benutzer lesbar sein, was für die Verschlüsselung notwendig ist, aber nur vom Superuser beschreibbar. [...]

DATEIEN
/etc/passwd

SIEHE AUCH
passwd(1), login(1), group(5).

Tastaturkommandos
man wird über Tastenkommandos bedient.
Die wichtigsten sind:

q	man beenden
/	Stichwort in der man page suchen
n	nächstes Vorkommen des Stichworts finden
Return	Zeile nach unten scrollen
f bzw. Space	Seite vorwärts blättern
b bzw. Backspace	Seite zurück blättern

Hilfe zu Kommandos

Die meisten Unixbefehle bringen eine eigene Kurzanleitung mit, die die Aufrufsyntax, Optionen und Hinweise enthält. Diese Hilfetexte sind über verschiedene Standardoptionen, die fast alle Kommandos interpretieren, abrufbar, werden aber teilweise auch automatisch ausgegeben, wenn Optionen oder Argumente fehlerhaft verwendet worden sind.

Syntax
<**Kommando**> **--help**
oder
<**Kommando**> **-h**

An- und Abmeldung am System

Einloggen

Vor dem Beginn der Arbeit am Unixsystem steht immer die **Anmeldung**.
Das System muß wissen, welcher User auf welchem Wege Zugriff auf die zur Verfügung gestellten Ressourcen nehmen möchte, um anhand dessen zu entscheiden, ob der Zugriff gewährt wird oder auch nicht, und welchen Nutzern Daten zugeordnet werden sollen.

Unabhängig davon, auf welchem Wege man sich am Unixsystem anmelden möchte (über das Netzwerk, lokal, per seriellem Terminal, ...), wird Unix immer nach einem **Usernamen** und dem dazugehörigen **Paßwort** fragen.

Eine gültige User-Paßwort-Kombination nennt man **Account**. Einen Account kann der Systemverwalter einrichten.

Wie fast überall unter Unix gilt auch hier: alle **Namen und Paßwörter sind case-sensitive**!

Terminal-Typen

Der Begriff Terminal ist ein Relikt aus alten Tagen der Großrechner. Damals wurden mehrere Bildschirmarbeitsplätze mit seriellen Kabeln an leistungsfähige Zentralrechner angeschlossen, um so Geld und Administrationsaufwand einzusparen.

Obwohl die Verbindungsaufnahme zu Unixsystemen heutzutage in der Regel nicht mehr über den langsamen seriellen Port geschieht, hat sie sich doch noch ein wenig des damaligen "Terminal-Feelings" erhalten. Spricht man heute von Terminals, so sind im allgemeinen die lokalen Konsolen oder via Netzwerk angemeldete Shellsessions gemeint. Um die Flexibilität in Bezug auf heterogene Netzwerke so groß wie möglich zu halten, ist das Konzept verschiedenartiger Terminals beibehalten worden.

Um dem System, an dem man sich anmeldet z.B. die eigene Tastaturbelegung (insbesondere Sondertasten wie Backspace, Delete, Cursor, etc.), Bildschirmgröße, Farbfähigkeit usw. mitteilen zu können, meldet man sich zunächst an und gibt - je nach verwendeter Shell - eines der nachfolgenden Kommandos ein:

setenv TERM <Terminal-Typ>	für Benutzer der C-Shell csh
oder	
TERM=<Terminal-Typ>; export TERM	für Benutzer von sh bzw. bash

Auf den meisten Linuxsystemen sollten diese Einstellungen allerdings vom Systemadministrator korrekt vorkonfiguriert sein.

Passwörter

Bei der Einrichtung eines eigenen Accounts wird vom Administrator meistens ein Standardpaßwort eingetragen. Dieses, dem Administrator bekannte Paßwort sollte man natürlich schnellstmöglich ändern.
Dies geschieht unter Unix mit dem Kommando "passwd".
Benutzerpaßwörter dürfen nur der Benutzer selbst und der Benutzer root ändern.
Nach Eingabe des Befehls "passwd" in der Shell wird zunächst noch einmal das alte, dann zur Sicherheit zweimal das neue Paßwort abgefragt.

Auf manchen Systemen sind Richtlinien für die Gestaltung von Paßwörtern festgelegt, so daß z.B. nur Paßwörter einer bestimmten Mindestlänge, Kombinationen aus Klein-/Großbuchstaben und Zahlen usw. akzeptiert werden. In der Regel weist das passwd-Programm Sie aber auf derartige Zustände hin.

Generell sollte man zur Auswahl eines Paßwortes folgende Richtlinien beachten:

NIEMALS	allgemein bekannte Wörterbuchvokabeln verwenden
	Vor-, Nach-, Rechner-, sonstige Namen, Informationen aus dem pers. Umfeld wählen
	Paßwörter notieren
	Sonderzeichen wie ä,ö,ü,ß verwenden, da einige Systeme nicht darauf vorbereitet sind
	das Paßwort weitergeben, nicht einmal an die Administratoren
UNBEDINGT	Kombinationen aus Klein-/Großbuchstaben und Zahlen verwenden
	mindestens 6 Zeichen Länge vorsehen
	häufig das Paßwort wechseln
	auf Beobachtung bei der Paßworteingabe achten und ggf. Paßwort ändern

Ausloggen

Aktive Sitzungen kann man auf mehrere Arten beenden:

- aktive Shellsitzungen werden mit exit beendet
- das System verläßt man mit logout
- per Tastaturkommando STRG-D

Struktur der Unix-Kommandozeile

Ein Kommando ist ein Programm, das dem Unix-System befiehlt, eine Aufgabe zu erledigen.
Kommandos haben unter Unix eine recht einheitliche Form:

Kommando [Optionen] [Argumente] [--help bzw. -h]

Die **Argumente** geben an, worauf das Kommando angewendet werden soll, also z.B. Dateien, Verzeichnisse o.ä.
Optionen beeinflussen das Verhalten des Kommandos, sie können also z.B. verschiedene Modi des Kommandos aktivieren. Die eckigen Klammern deuten an, daß die Angabe von Optionen oder Argumenten teilweise optional ist - man muß sie also nicht immer angeben.

Kommandos, Argumente und Optionen sind case-sensitive.
Kommando ist also nicht dasselbe wie *kommando*.

Optionen werden in der Regel mit einem führenden Minuszeichen (-) eingeleitet. Mehrere Optionen können nach folgenden Schemata verkettet werden:

Kommando -[Option][Option][Option]

oder

Kommando -[Option] -[Option] -[Option]

z.B.

ls -alR

ls (das Äquivalent zum dir-Befehl unter DOS/Windows) wird in diesem Fall das Inhaltsverzeichnis des aktuellen Arbeitsverzeichnisses im langen Format (Option l), inklusive der versteckten Dateien (Option a) und rekursiv durch alle enthaltenen Unterverzeichnisse (Option R) ausgeben.

Einige Kommandos erfordern darüber hinaus die Angabe von Parametern für Optionen. Dann wird meistens jede Option einzeln mit einem Minuszeichen (-) eingeleitet:
z.B.

lpr -Plaserjet5 -#2 datei.txt

Das Kommando lpr würde demnach die Datei datei.txt zwei mal (Option -#<Parameter>) auf dem Drucker laserjet5 (Option P<Parameter>) ausdrucken.

Dies sind die Standardkonventionen für Unix-Kommandos. Alle Programmierer von Kommandozeilentools sind dazu aufgerufen, sich an diese Regeln zu halten, aber dennoch hält sich nicht jede Software daran.

Im Zweifelsfall hilft ein Blick in die Online-Hilfe des Kommandos, die sog. **man page** (von engl. Manual, Handbuch), oder der Aufruf der Kurzhilfe mit den Optionen -h oder --help.

Tastenkombinationen

Tastenkombinationen, auch oft als Control Keys oder Hot Keys bezeichnet, werden benutzt, um spezielle Aktionen in der Shell oder in Programmen auszulösen.
Meistens werden Tastenkombinationen in den Formen CTRL-<Taste>, STRG-<Taste> oder **^T**aste angegeben, also z.B. STRG-Z oder ^M.
Aufgerufen werden sie, indem man die STRG-Taste drückt und hält und dazu die angegebene Taste betätigt.
Bei den Control Keys wird nicht nach Groß-/kleinschreibung unterschieden.

Wichtige Kombinationen in gängigen Shells sind:

- STRG-C: aktuelles Kommando abbrechen
- STRG-Z: aktuelles Kommando anhalten; es kann danach mit fg bzw. bg fortgesetzt werden
- STRG-S: die aktuelle Shell für Eingaben sperren
- STRG-Q: die aktuelle gesperrte Shell wieder für Eingaben freigeben
- STRG-L: den Bildschirm löschen, entspricht cls unter DOS
- STRG-U: die aktuelle Kommandozeile in der Shell löschen
- STRG-D: aus der Shellsitzung ausloggen

Außer diesen „klassischen" Control Keys gibt es auf modernen Linuxsystemen noch einige praktische Tastenkombinationen, die das Leben einfacher machen:

- Shift-BildAuf / -BildAb:
  die vorherigen Ausgaben der Shellsitzung durchblättern; sehr nützlich, um schnell noch einmal einen Blick auf frühere Kommandoausgaben werfen zu können, z.B. Verzeichnislistings etc.

- Tabulator-Taste:
  die **wichtigste Taste** überhaupt (leider nicht in allen Shells verfügbar)!
  Durch Druck auf die Tab-Taste vervollständigen moderne Shells (insbes. bash) Kommando-, Datei- und Verzeichnisnamen und ersparen dem Benutzer so eine Menge Tipparbeit. Auch sehr von Vorteil, wenn man die genaue Schreibweise eines Befehls nicht kennt: 2x Tab-Taste zeigt alle verbleibenden Alternativen an.

- Alt-<F-Tasten>:
  Mit der Kombination aus Alt- und den F-Tasten 1-6 läßt sich zwischen bis zu sechs voneinander unabhängigen Textkonsolen hin und her schalten, in denen man sich lokal am Rechner - auch unter verschiedenen Usernamen - anmelden kann.
  Auf weiteren Konsolen (erreichbar über Alt-F8-F12) können, je nach Konfiguration des Linux-Systems, weitere Informationen abrufbar sein, z.B. Logfiles, Status des Systems oder die Kernelmeldungen.

- Alt-F7:
  Eine Sonderstellung hat bei aktiviertem Grafiksystem X-Window die Tastenkombination Alt-F7: damit schaltet man aus den Textkonsolen direkt wieder in den Grafikmodus. Zurückwechseln vom Grafikmodus in die Textkonsolen ist mit den Tastenkommandos STRG-Alt-<F-Taste> jederzeit möglich.

Verzeichnisnavigation und –Verwaltung

Um im Verzeichnisbaum zu navigieren und Verzeichnisstrukturen anzulegen und zu verwalten dienen folgende Kommandos:

Kommando	**Wofür?**
ls	Verzeichnisinhalt auflisten
cd	Verzeichnis wechseln
mkdir	Verzeichnis(-struktur) anlegen
rmdir	Verzeichnis(-struktur) löschen
pwd	aktuelles Arbeitsverzeichnis anzeigen

Wenn man mit DOS oder ähnlichen Systemen vertraut ist, hilft folgende Tabelle beim Umstieg:

Aktion	**Unix**	**DOS**
Verzeichnisinhalt auflisten	ls	dir
Verzeichnis wechseln	cd	cd
Verzeichnis(-struktur) anlegen	mkdir	md bzw. mkdir
Verzeichnis(-struktur) löschen	rmdir	rd bzw. rmdir
aktuelles Arbeitsverzeichnis anzeigen	pwd	cd

pwd - aktuelles Verzeichnis anzeigen

Oft zeigt die Shell die aktuelle Position im Verzeichnisbaum in ihrem Prompt an. Um den Namen des aktuellen Arbeitsverzeichnisses jedoch stets ausgeben zu können, dient der Befehl pwd, engl. für print working directory, z.B.

```
% pwd
/home/mark/Texte
```

cd - Verzeichnis wechseln

Mit dem Befehl cd kann man sich im Dateisystembaum „bewegen“ und das aktuelle Arbeitsverzeichnis wechseln. cd akzeptiert absolute und relative Pfadnamen.

Syntax

cd Verzeichnis

Beispiele

cd	ins Homeverzeichnis wechseln
cd /	ins root-Verzeichnis wechseln
cd ..	ins übergeordnete Verzeichnis wechseln
cd ../..	zwei Verzeichnisebenen nach oben wechseln
cd /home/mark/Texte	in den angegebenen absoluten Pfad wechseln
cd Texte/Bericht99	in den angegebenen relativen Pfad wechseln

mkdir - Verzeichnis erstellen

Um in einem Verzeichnis, in dem man Schreibberechtigung hat, eine sinnvolle Verzeichnisstruktur zur Archivierung eigener Daten zu erstellen, benutzt man das Kommando mkdir.

Syntax

mkdir [Optionen] Verzeichnis

Nützliche Optionen

-p	kompletten angegebenen Pfad anlegen, wenn Zwischenverzeichnisse fehlen
-m Modus	beim Anlegen File Permissions setzen (s. chmod)

Beispiele

```
% mkdir /home/mark/Daten
```

oder, wenn wir uns schon im Verzeichnis /home/mark befinden, mit relativem Pfad:

```
% mkdir Daten
```

rmdir - Verzeichnis löschen

Um ein Verzeichnis mit rmdir löschen zu können, muß es leer sein. Ebenso kann das aktuelle Arbeitsverzeichnis nicht gelöscht werden. Man muß also zunächst mit rm alle Dateien löschen und aus dem Verzeichnis herauswechseln.

Syntax

rmdir Verzeichnis

Nützliche Optionen

-p	explizit angegebene Parent-Verzeichnisse löschen, wenn sie leer sind

Beispiele

```
% rmdir Test
```

ls - Verzeichnisinhalt anzeigen

ls ist das Unix-Äquivalent zum dir-Befehl unter DOS/Windows - nur um einiges leistungsfähiger.
ls kann nicht nur simple Inhaltsverzeichnisse anzeigen, sondern alle Informationen, die im Dateisystem über Daten, Verzeichnisse oder Geräte bekannt sind. Unter diese Informationen fallen die Größe und Art der Datei, Berechtigungen, Anzahl der auf die Datei bezogenen Links, Erstell-, Änderungs- und Zugriffsdaten.

Syntax

ls [Optionen] [Datei-Suchmuster]

Nützliche Optionen

Wird ls ohne Optionen oder Suchmuster gestartet, wird es den Inhalt des aktuellen Verzeichnisses ausgeben. ls kennt sehr viele Optionen - ein Blick in die man page kann nicht schaden.

-a	alle Dateien, auch versteckte, anzeigen
-l	ausführliches Inhaltsverzeichnis ausgeben
-d	nur Verzeichnisnamen, nicht ihren Inhalt anzeigen
-F	Typ des Dateisystemobjekts durch folgende Zeichen kennzeichnen:
/	für Verzeichnisse
=	für Sockets
@	Links
*	ausführbare Dateien

Beispiele

```
% ls -la
insgesamt 136
drwxr-xr-x   2 chris    users        4096 Jun 16 12:31 .
drwxr-xr-x  20 chris    users        4096 Jun 16 12:29 ..
-rw-r--r--   1 chris    users        5376 Jun 16 12:30 .gimprc
-rw-r--r--   1 chris    users         341 Jun 16 12:30 .vimrc
-rw-r--r--   1 chris    users          10 Jun 16 12:30 .zsh
-rw-r--r--   1 chris    users       98745 Jun 16 12:31 pppoed0.46.tgz
drwxr-xr-x   2 chris    users        4096 Jun 16 12:31 Texte
[...]
```

Die Option -la bewirkt, daß ls den Verzeichnisinhalt inkl. versteckten Dateien (-a) und ausführlichen Informationen (-l) auflistet.

Ein einfaches ls ohne Optionen im selben Verzeichnis, diesmal absolut angegeben, ergibt folgende Ausgabe:

```
% ls /home/chris/test
pppoed0.46.tgz  temp  webcam.html
```

Dateiverwaltung

Um Dateien anzulegen, sie zu kopieren, zu verschieben, zu löschen oder Dateiberechtigungen zu setzen bietet Unix folgende Kommandos an:

Kommando	**Wofür?**
cp	Datei / Verzeichnis kopieren
mv	Datei / Verzeichnis verschieben
rm	Datei / Verzeichnis löschen
chown	Eigentümer ändern
chgrp	Gruppenzugehörigkeit ändern
chmod	Berechtigungen setzen
mc	Dateimanager für die Shell

Wenn man mit DOS oder ähnlichen Systemen vertraut ist, hilft folgende Tabelle beim Umstieg:

Aktion	**Unix**	**DOS**
Datei kopieren	cp	copy
Datei verschieben	mv	move
Datei umbenennen	mv	rename bzw. ren
Datei löschen	rm	del
Dateiattribute setzen	chmod, chown, chgrp	attrib (eingeschränkt)
Dateimanager (Textmode)	mc	nc (Norton Commander)

cp - Datei kopieren

cp kopiert den Inhalt einer Datei in eine andere.

Syntax

cp [Optionen] Quelldatei Zieldatei

Nützliche Optionen

-i	interaktives Kopieren, fragt bei jeder Datei nach
-R	Verzeichnis rekursiv inkl. aller Unterverzeichnisse kopieren
-p	Dateiberechtigungen und Eigentums-/Gruppenverhältnisse beibehalten

Beispiele

% cp -R -p /home/mark/Texte/ ./Backup/

kopiert den kompletten Inhalt des Verzeichnisses /home/mark/Texte inkl. der Unterverzeichnisse (-R) in das Verzeichnis Backup im aktuellen Arbeitsverzeichnis.
Dabei werden die Dateiattribute mitkopiert (-p).

mv - Datei verschieben oder umbenennen

Für das Verschieben oder Umbenennen von Dateien oder Verzeichnissen dient das Kommando mv.

Syntax

mv [Optionen] Quelldatei Zieldatei

Nützliche Optionen

-i	interaktives Verschieben, fragt bei jeder Datei nach
-f	force, nicht nachfragen vor dem Überschreiben von existierenden Zieldateien

Beispiele

% mv /home/karl/ /Backup/EhemaligeMitarbeiter/
verschiebt das Homeverzeichnis des Users karl in ein Backupverzeichnis

rm - Datei löschen

Mit rm werden Dateien oder Verzeichnisse unwiderruflich gelöscht.

Syntax

rm [Optionen] Dateiname

Nützliche Optionen

-i	interaktives Löschen, vor jeder Datei nachfragen
-r	rekursiv inkl. Unterverzeichnissen löschen
-f	force, Löschen nicht leerer Verzeichnisse ermöglichen (VORSICHT!)

Bei diesem Kommando ist Vorsicht angesagt, da in der Regel keine Rückfrage vor dem Löschen geschieht!
Es macht sehr viel Sinn, im Konfigurationsskript der Shell ein sogenanntes Alias für den rm-Befehl einzutragen, das den Standardmodus von rm auf „interaktiv“ setzt. So wird beim Löschen immer zunächst rückgefragt.

Tragen Sie dazu folgende Zeilen in die Konfigurationsdatei Ihrer Shell ein:

bash	`alias rm='rm -i'` in die Datei .bashrc im Homeverzeichnis
c-Shell	`alias rm "rm -i"` in die Datei .cshrc im Homeverz.

Beispiele

% rm -rf /home/mark/Texte
Vorsicht! Dieses Kommando löscht den kompletten angegebenen Ast im Verzeichnisbaum unwiderruflich.

chmod - Berechtigungen ändern

Mit chmod (engl. change mode) kann die Rechtemaske von Dateien, Verzeichnissen oder Geräten gesetzt werden, um den Zugriff auf eigene Daten zu steuern. Nur der Eigentümer oder der Benutzer root dürfen die Dateiberechtigungen ändern.

Syntax

chmod kennt zwei Modi zur Angabe der gewünschten Rechtemaske:

Symbolischer Modus:

chmod [Benutzer/-gruppe] [Aktion] [Rechte] Datei- oder Verzeichnisname
wobei

Benutzer-/Gruppe	**u**	für den Eigentümer
	g	für die Gruppe, der die Datei angehört
	o	für alle anderen Benutzer
	a	für alle Benutzer des Systems
Aktion	**+**	für „Rechte hinzufügen“
	-	für „Rechte entziehen“
	=	für „alle Rechte setzen“
Rechte	**r**	für readable, lesbar
	w	für writeable, beschreibbar
	x	für executeable, ausführbar

nach o.g. Syntax verwendet werden können.

Bei Benutzung von chmod im symbolischen Modus werden bei Benutzung der Aktionsoperatoren „+“ bzw. „-“ nur die Rechte geändert, die explizit angegeben werden. Der restliche, von der Änderung nicht betroffene Teil der Maske bleibt unverändert.

Benutzt man den Operator „=“, wird die komplette Rechtemaskte so gesetzt wie angegeben.

Numerischer Modus:

chmod [numerische Rechtemaske] Datei- oder Verzeichnisname

Mit der Angabe eines numerischen Rechtearguments wird die komplette Rechenmaske neu gesetzt. Der zu übergebende Wert läßt sich nach folgendem Schema als Summe der oktalen Werte aller zu setzenden Rechte berechnen:

	Rechte des Eigentümers			Rechte der Gruppe			Rechte aller anderen User		
Attribut	r	w	x	r	w	x	r	w	x
Dezimal	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Oktal	400	200	100	40	20	10	4	2	1

Beispiele

Im aktuellen Verzeichnis liegt die Datei datei.txt mit folgenden Berechtigungen und Eigentumsverhältnissen:

-rwxr-x--- chris users datei.txt

Sie soll auch für User, die nicht chris sind und nicht der Gruppe users angehören, lesbar und ausführbar gemacht werden.

Symbolischer Modus
% chmod o+rx datei.txt

Mit „o“ gibt man an, daß sich die Änderungen auf „andere“ beziehen soll, „+“ besagt, daß Rechte erteilt werden sollen, und „rx“ sind die beiden zu erteilenden Rechte für lesen und ausführen. Es werden nur die angegebenen Rechte verändert.

% chmod u=rwx,go=rx datei.txt

Der Aktionsoperator = bewirkt, daß die komplette Rechtemaske anhand der gegebenen Informationen neu gesetzt wird. Es werden also für den Eigentümer alle Rechte, für die Gruppe und alle anderen r und x eingetragen.

Numerischer Modus
% chmod 755 datei.txt

Anhand folgender Tabelle, in der die gewünschten Rechte mit x markiert sind, läßt sich der Wert 755 erklären:

	Rechte des Eigentümers			Rechte der Gruppe			Rechte aller anderen User		
Attribut	r	w	x	r	w	x	r	w	x
Oktal	400	200	100	40	20	10	4	2	1
gewünscht	x	x	x	x		x	x		x

Addiert man die zu den Rechten korrespondierenden Zahlenwerte, erhält man folgendes Ergebnis:

755 = 400 + 200 + 100 + 40 + 10 + 4 + 1

chown - Eigentümer ändern

Die Eigentumsverhältnisse einer Datei lassen sich mit dem Kommando chown (engl. change owner) ändern. Auf den meisten Unixsystemen darf nur der root Dateien anderen Usern zuschreiben, d.h. ein normaler User kann das Eigentum auf seine Daten nicht an andere Benutzer abgeben.

Syntax

chown [Optionen] User[:Gruppe] Datei

Nützliche Optionen

-R rekursiv, alle Unterverzeichnisse mitbehandeln
-f force, keine Fehler melden

Beispiele

% chown mark:marketing datei.txt

weist die Datei datei.txt dem User mark und der Gruppe marketing zu.

chgrp - Gruppe ändern

Mit dem Kommando chgrp dürfen auch normale User die Gruppenzugehörigkeit von Dateien, die ihnen gehören, ändern - jedoch muß der User auch in der neuen Gruppe geführt werden.

Syntax

chgrp [Optionen] Datei

Nützliche Optionen

-R	rekursiv ändern
-f	force, keine Fehlermeldungen

Beispiele

% chgrp support datei.txt

weist datei.txt der Gruppe support zu, sofern der Eigentümer der Datei auch Mitglied der Gruppe support ist.

mc - der Midnight Commander

mc ist eine Weiterentwicklung des unter DOS bekannten Norton Commanders. Er sieht genauso aus und läßt sich über die selben Tastaturkommandos wie das Original bedienen.
mc bietet Funktionen für Datei-, Verzeichnis- und Textverwaltung, einen Editor, gute Suchfunktionen, einen symbolischen Editor für Dateiberechtigungen, und er kann auf das Netzwerk zugreifen.

Wenn man schnell Dateien kopieren, Verzeichnisse umsortieren oder Textdateien editieren muß, ist mc ein sehr nützliches Tool. mc gehört nicht zu den Unix-Standardkommandos, sondern muß separat installiert werden. Bei den meisten Linuxdistributionen ist mc jedoch enthalten.

Syntax

mc [Optionen]

Nützliche Optionen

-c	color, startet mc in Farbe. Das benutzte Terminal muß diese Funktion unterstützen.

Textausgabe

Es gibt eine Reihe von Befehlen, mit denen Zeichenketten, numerische Werte oder Dateien auf den Bildschirm ausgegeben werden können:

Kommando	**Wofür?**
echo	Zeichenkette ausgeben
cat	Datei ausgeben
less	Datei seitenweise betrachten
head	die ersten n Zeilen einer Datei ausgeben
tail	die letzten n Zeilen einer Datei ausgeben

Wenn man mit DOS oder ähnlichen Systemen vertraut ist, hilft folgende Tabelle beim Umstieg:

Aktion	**Unix**	**DOS**
Zeichenketten ausgeben	echo	echo
Datei ausgeben	cat	type
Datei betrachten	less	more

echo - Textzeilen ausgeben

Das Kommando echo wird benutzt, um Text, den man als Argument übergibt, auf den Standardausgabekanal stdout auszugeben. Die Zeichenkette, die ausgegeben werden soll, wird automatisch mit einem Zeilenvorschubzeichen (\n) abgeschlossen, wenn nicht per Option geändert.

Textstrings können unter Linux mit Formatanweisungen versehen werden (s.u.).
echo interpretiert diese Formatierungssequenzen jedoch in der Standardeinstellung nicht, dieses Verhalten muß per Option aktiviert werden.

Syntax

echo [Optionen] Text

Nützliche Optionen

-n	kein abschließendes newline (\n) einfügen
-e	Interpretation der Formatierungssequenzen aktivieren

Formatierungssequenzen

\a	alert, Piepton
\b	Backspace
\c	kein abschließendes newline einfügen
\f	form feed, Seitenvorschub (insbes. für Drucker)
\n	newline einfügen
\r	carriage return, Wagenrücklauf
\t	horizontaler Tabulator
\v	vertikalen Tabulator
\\	Backslash
\NNN	ASCII-Zeichen einfügen, dessen Wert NNN (oktal) ist

Beispiele

einfacher Aufruf ohne Optionen, gibt String inkl. newline-Zeichen aus:

```
% echo Hallo, Welt!
Hallo, Welt!
%
```

ohne abschließendes newline-Zeichen (Prompt erscheint hinter der letzten Ausgabezeile):

```
% echo -n Hallo, Welt!
Hallo, Welt!%
```

mit Formatierungsanweisungen:

```
% echo -e ‘Hallo,\n\tWelt!‘
Hallo,
        Welt!
%
```

cat - Datei ausgeben

cat gibt Dateien auf den Bildschirm aus.

Syntax

cat [Optionen] Datei

Nützliche Optionen

-n	jede Zeile der Datei mit Zeilennummer versehen
-v	nicht-sichtbare Formatieranweisungen oder Zeichen anzeigen

Beispiele

```
% cat datei.txt
```

gibt die Datei datei.txt in der Shell aus.

```
% cat -n datei1.txt datei2.txt datei3.txt
```

gibt nacheinander die drei Dateien aus und setzt Zeilennummern voran

less - Dateien seitenweise anzeigen

Wenn man Dateien komfortabel betrachten, also seiten- oder zeilenweise mit den normalen Bildlauftasten (Cursortasten, Bild-auf/-ab, etc.) in der Datei herumscrollen möchte, benötigt einen sogenannten **pager**.

less ist das Unix-Standardtool für genau diese Aufgabe. Es läßt sich auch in eigenen Programmen oder Skripten zur komfortablen standardisierten Anzeige von Texten benutzen und hat daher eine große Verbreitung erreicht.

Es dient unter anderem auch als Anzeige-Subprogramm bei den manual pages.

Syntax

less [Optionen] [+/Suchmuster] Dateiname

Nützliche Optionen

-c	Anzeige vor der Ausgabe löschen
-i	ignore case, Groß-/kleinschreibung bei Suchmustern ignorieren

Tastaturkommandos

q	less beenden
/	Stichwort/Suchmuster/regulären Ausdruck im Text suchen
n	nächstes Vorkommen des Stichworts finden
Return	Zeile nach unten scrollen
f bzw. Space	Seite vorwärts blättern
b bzw. Backspace	Seite zurück blättern

Auf vielen Linuxsystemen ist ein Scrollen in der Datei auch mit den Bildlauftasten (Cursortasten, Bild-Auf/-Ab, Home, End) möglich.

Beispiele

% less datei.txt
öffnet datei.txt zur Ansicht.

Die Suche nach Stichwörtern ist mit der Taste /, gefolgt von der Eingabe des Suchstrings möglich.

head - Dateianfang anzeigen

head zeigt beliebig viele Zeilen des Dateianfangs an. Ohne Angabe von Optionen werden die ersten zehn Zeilen ausgegeben.

Syntax

head [Optionen] Datei

Nützliche Optionen

-n<Anzahl> bzw. -<Anzahl>	die ersten <Anzahl> Zeilen ausgeben

Beispiele

% head -40 datei.txt
gibt die ersten 40 Zeilen von datei.txt aus

tail - Dateiende anzeigen

tail gibt, analog zu head, beliebig viele Zeilen des Dateiendes aus.
Desweiteren kann tail offene Dateien, in die also von anderen Programmen noch hineingeschrieben wird, beobachten und sofort jede Änderung anzeigen. Diese Funktion ist sehr nützlich um z.B. wachsende Logdateien zu betrachten.

Syntax

tail [Optionen] Datei

Nützliche Optionen

-n<Anzahl> bzw. -<Anzahl>	die letzten <Anzahl> Zeilen ausgeben
-f	follow, offene Dateien beobachten

Beispiele

% tail -f /var/log/messages
Beobachtet die System-Logdatei und stellt neue Einträge direkt dar.

% tail -n25 datei.txt
gibt die letzten 25 Zeilen von datei.txt aus.

Drucken

Unter Unix existieren mehrere konkurrierende Drucksysteme. Als gängige Standards haben sich die netzwerkfähigen Druckservices lpr (BSD), lp (SysV) und neuerdings LPRNG (Next Generation, baut auf BSD auf) etabliert. Außer anderen Namen für die Drucktools und geringfügiger Unterschiede bei der Druckjobbehandlung sind sie jedoch für den User recht ähnlich.

Folgende Kommandos stehen bereit:

Kommando		**Wofür?**
BSD	SysV	
lpq	lpstat	Status des Druckers/Druckjobs anzeigen
lpr	lp	Datei drucken
lprm	cancel	Druckjob abbrechen/löschen

lp / lpr - Druckjob absenden

lp bzw. lpr (engl. line print) schicken eine zu druckende Datei an das lokale oder auf einem Druckerserver im Netzwerk installierte jeweilige Drucksystem und veranlassen deren Ausdruck. Diese Druckjobs werden vom Drucksystem mit ID-Nummern versehen, über die sie vom Benutzer jederzeit beobachtet, abgebrochen oder gelöscht werden können. Die Druckjobs landen in einer Warteschlange für den jeweiligen Drucker und werden, je nach Konfiguration des Systems, der Reihe nach, nach bestimmten Prioritäten oder sofort ausgedruckt.

Syntax

lp [Optionen] Dateiname
lpr [Optionen] Dateiname

Nützliche Optionen

lp (SysV)	**lpr (BSD)**	**Funktion**
-n <Anzahl>	-#<Anzahl>	<Anzahl> Kopien drucken
-t <Titel>	-T<Titel>	Titelzeile für den Druckjob
-d <Drucker>	-P<Drucker>	Name des zu verwendenden Druckers
-c	(Standard)	Druckjob in die Warteschlange stellen
(Standard)	-s	Druckjob nicht in die Warteschlange stellen
-o Optionen		zusätzliche Optionen angeben (s. man page)

Beispiele

% lpr -#2 -Plaserjet5 datei.txt
druckt zwei Kopien von datei.txt auf dem Drucker laserjet5 aus.

% lp -t Testdatei -d lj4 datei.txt
druckt datei.txt auf dem Drucker lj4 mit der Titelzeile „Testdatei“ aus.

lpstat / lpq - Druckstatus überprüfen

Um den Status des Druckers, der Druckerwarteschlange und der Druckjobs zu überprüfen benutzt man die Befehle lpstat (line printer status, SysV) bzw. lpq (line printer queue, BSD).

Syntax

lpstat [Optionen]
lpq [Optionen]

Nützliche Optionen

lpstat (SysV)	**lpq (BSD)**	**Funktion**
-d	(Default: lp)	Standarddrucker des Systems anzeigen
-s		Druckstatus zusammenfassen
-t		alle Statusinformationen ausgeben
-u [Userliste]		Druckjobs einzelner User anzeigen
-v		alle dem System bekannten Drucker anzeigen
-p <Drucker>	-P<Drucker>	Drucker zur Statusüberprüfung auswählen

Beispiele

Hier werden Informationen über einen Druckjob im SysV-Drucksystem angezeigt (Job-ID, Eigentümer, Größe des Druckjobs, Erstelldatum und Drucker):

```
% lpstat
lp-147    chris      288329  Jun 13 17:34 on hplj6l
```

Und so sieht ein Druckjob im BSD-Drucksystem aus:

```
% lpq
lp is ready and printing
Rank   Owner   Job  Files          Total Size
active root    32   /root/ServerLog  4217 bytes
```

cancel / lprm - Druckjob abbrechen

Benutzer können nur ihre eigenen Druckjobs abbrechen oder aus der Druckerwarteschlange löschen.
Dies geschieht mit den Kommandos cancel (SysV) bzw. lprm (BSD), eventuellen Optionen und unter Angabe der Job-ID, die man durch Aufruf der o.g. Statuskommandos erhält.

Syntax

cancel [Job-ID] [Drucker]
lprm [Optionen] [Job-ID] [Username]

Nützliche Optionen

cancel (SysV)	**lprm (BSD)**	**Funktion**
	-P<Drucker>	Drucker auswählen
	- <Username>	alle Jobs des Benutzers <Username> löschen
-u <Userliste>		Druckjobs einzelner User löschen

Beispiele

Um die o.g. Druckjobs zu löschen, gibt man folgende Kommandos ein:

cancel

```
% cancel lp-147
```

lprm

```
% lprm 32
```

a2ps - ASCII-Dateien formatiert drucken

Wenn man häufig lange unformatierte Textdateien drucken muß, ist das Tool a2ps eine große Hilfe.
Es wandelt ASCII-Daten ins Postscript-Format um und ordnet auf Wunsch mehrere Seiten auf einem Blatt an.
Desweiteren können mit a2ps der Zeichensatz, die Schriftgröße und vieles mehr eingestellt werden.
Auch ein direktes Ausdrucken der so erzeugten Postscriptdokumente ist möglich.

Syntax

a2ps [Optionen] Dateiname(n)

Nützliche Optionen

-o <Datei>	erzeugt Postscriptdatei <Datei>
-P <Drucker>	druckt Dokument auf <Drucker>
-n <Anzahl>	druckt <Anzahl> Kopien des Dokuments
-M <Medium>	paßt Druckdaten an das Papierformat <Medium>, z.B. DIN A4, an
--landscape	orientiert die Druckdaten im Querformat
--portrait	orientiert die Druckdaten im Hochformat
--columns=<Anzahl>	teilt das Blatt in <Anzahl> Spalten auf, zum Drucken mehrerer Seiten auf einem Blatt
--row=<Anzahl>	teilt das Blatt in <Anzahl> Zeilen auf, zum Drucken mehrerer Seiten auf einem Blatt
-f <Größe>	wählt die Größe des Zeichensatzes
-1, -2, ..., -9	wählt vordefinierte Seitenlayouts für eine bis neun Seiten pro Blatt

Beispiele

% a2ps --landscape --columns=2 -f 6 -P hp5 webdesign.html
druckt die Datei webdesign.html mit zwei quer nebeneinander angeordneten Seiten pro Blatt mit 6 Punkt Schriftgröße auf dem Drucker hp5 aus.

% a2ps -o /home/chris/datei.ps -M a4 -4 datei.txt
erzeugt aus der Datei datei.txt das Postscriptdokument datei.ps im Format DIN A4 mit 4 Seiten pro Blatt.

Systemkommandos

Die folgenden Kommandos dienen zur Anzeige bzw. Steuerung von Systemressourcen wie Arbeitsspeicher, Festplattenplatz, Rechenzeit, etc.

Kommando	**Wofür?**
date	Datum und Uhrzeit formatiert ausgeben
df	disk free, freien Speicherplatz verschiedener Speichermedien anzeigen
du	zeigt Größe (belegten Platz) von Dateien/Verzeichnissen an
hostname bzw. uname	Rechnernamen und Systemtyp anzeigen
ps	Prozeßstatus und -Statistik anzeigen
top	Prozeßmonitor
kill	Signale an Prozesse senden, i.d.R. zum Beenden
who	eingeloggte User anzeigen
whereis	Position installierter Softwarepakete ermitteln
which	gibt den Pfad zu Kommandos aus
script	Shellsitzungen protokollieren

df - Festplattenauslastung anzeigen

df zeigt an, wie viele Blöcke und Inodes auf den Speichergeräten oder Netzwerkshares belegt und frei sind.

Syntax

df [Optionen] [Ressource]

Nützliche Optionen

-l nur lokale Dateisysteme anzeigen
-h Ausgabe im „human readable“-Format, d.h. in KBytes, MBytes oder GBytes

Beispiele

```
% df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda1       1.9G  1.3G  516M  72% /
/dev/hda6       1.9G  199M  1.6G  11% /WG
/dev/hda7       9.8G  5.8G  3.7G  61% /DATEN
```

gibt die Plattenbelegung der Partitionen hda1, hda6 und hda7 in "menschenlesbarer" Form (-h) aus.

du - Speicherverbrauch von Dateien und Verzeichnissen

Um die exakte Größe eines Verzeichnisbaums oder Teilen davon zu ermitteln, kann man den Befehl du verwenden.

Syntax

du [Optionen] Verzeichnis

Nützliche Optionen

-a auch Speicherverbrauch einzelner Dateien anzeigen, nicht nur Verzeichnisse
-s summary, nur Zusammenfassung anzeigen

Beispiele

% du

```
23      ./.kde
7       ./.vmware
76      ./Desktop
278     .
```

zeigt die Anzahl enthaltener KBytes im aktuellen Arbeitsverzeichnis und allen Unterverzeichnissen an.

ps - Anzeige und Statistik laufender Prozesse

ps wird benutzt, um Informationen über laufende Prozesse anzuzeigen.
Die Ausgabe des Kommandos läßt sich mit einigen Optionen an die eigenen Bedürfnisse anpassen.

Syntax

ps [Optionen]

Nützliche Optionen

SysV	BSD	Funktion
-e	-a	alle Prozesse aller User anzeigen
-u <User>	-u <User>	Prozesse einzelner User anzeigen
-l	-l	long format, ausführliche Liste anzeigen
	-w, -ww, -www	bestimmt die angezeigte Länge des Kommandostrings

Anmerkung:
Da ps je nach zugrundeliegendem System in unterschiedlichen Implementationen oder Versionen vorliegen kann, sollte man einen Blick in die man page von ps werfen.

Beispiele

```
% ps
PID    TT      STAT    TIME    COMMAND
15549  p0      IW      0:00    -tcsh (tcsh)
15588  p0      IW      0:00    man nice
15594  p1      IW      0:00    sh -c less /tmp/man15588
[...]
```

ps zeigt die aktuell laufenden Prozesse des aktuellen Users an.

top - Prozeßmonitor

top beobachtet laufende Prozesse und stellt sie ständig aktualisiert und formatiert dar.

Syntax

top

Beispiele

```
% top
9:36pm  up 18 days,  7:53,  1 user,  load average: 0.05, 0.01, 0.00
64 processes: 57 sleeping, 1 running, 6 zombie, 0 stopped
CPU states:  0.7% user,  1.5% system,  0.0% nice, 97.6% idle
Mem:    46784K av,   45640K used,    1144K free,   53260K shrd,    2064K buff
Swap:  266072K av,    2772K used,  263300K free                  17976K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
15781 chris     12   0  1068 1068   880 R       0  1.5  2.2   0:01 top
15721 chris      7   0  4268 4268  3088 S       0  0.3  9.1   0:04 konsole
15613 chris      1   0  4224 4224  3172 S       0  0.1  9.0   0:01 kwm
    1 root       0   0   200  200   172 S       0  0.0  0.4   0:04 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:04 kflushd
[...]
```

kill - Prozesse beenden

kill schickt Signale an Programme. Das meistbenutzte Signal ist KILL, das zum Beenden eines Prozesses benutzt wird. Eine Liste aller ca. 30 möglichen Signale findet man in der man page zu kill.

Syntax

kill [Optionen] [-SIGNAL] Prozeß-ID

Nützliche Optionen

-l	zeigt alle möglichen Signale an

Beispiele

```
% kill -9 15599
[1] + Killed        emacs unixgrep.txt
```

beendet das Programm emacs, das dem System unter der Prozeß-ID 15599 bekannt ist.
Die PID lßt sich mit dem Kommando ps ermitteln.

who - eingeloggte User anzeigen

who berichtet über die im System eingeloggten User. Zusätzlich kann es die aktuelle Identität (den aktuell benutzten User) ausgeben.

Syntax

who [am i]

Beispiele

```
% who am i
chris
%
```

gibt den Usernamen des aktuell benutzten Users aus.

```
% who
mark    tty1    Jun 14 20:14    (192.168.102.2)
chris   tty2    Jun 13 13:45    (192.168.102.3)
[...]
%
```

gibt Informationen über eingeloggte User, deren virtuellen Zugang zum System (Terminal), die Loginzeit und den Rechner, von dem aus der User arbeitet.

whereis - wo befindet sich Programm xyz?

Mit whereis kann ermittelt werden, in welchem Verzeichnis bestimmte Kommandos installiert sind. whereis liefert den komletten Pfad zum gesuchten Kommando.

Syntax

whereis [Optionen] Kommando

Nützliche Optionen

-b	nur binäre Dateien anzeigen
-m	nur man pages anzeigen
-s	nur Sourcecode anzeigen

Beispiele

```
% whereis bash
/bin/bash
```

zeigt den absoluten Pfad zum Executable der Shell bash an

which - welches Programm wird bei Kommando xyz ausgeführt?

which ermittelt, welche ausführbare Datei bei Eingabe eines beliebigen Kommandos ausgeführt wird.

Syntax

which Kommando

Beispiele

% which mail
/usr/bin/mail
zeigt an, daß bei Aufruf des Kommandos „mail“ die ausführbare Datei /usr/bin/mail gestartet werden würde.

hostname / uname - Anzeige von Rechnername und -Typ

Um in der Kommandozeile oder in Skripten den Rechnernamen oder -typen ermitteln zu können dienen die Befehle hostname und uname.

Syntax

hostname [Optionen]
uname [Optionen]

Nützliche Optionen

hostname

-i	IP-Adresse anzeigen
-d	Unix-Domainnamen anzeigen
-f	fully qualified, vollen Unix-Hostnamen anzeigen

uname

-a	alle verfügbaren Informationen anzeigen
-m	Typ der Maschine (Architektur) anzeigen
-s	Name des Betriebssystems anzeigen

Beispiele

% hostname -i
192.168.102.5
%
zeigt die IP-Adresse des Rechners an, auf dem man eingeloggt ist.

% uname -s
Linux
%
zeigt den Namen des installierten Betriebssystems an.

script - Shell-Sessions aufzeichnen

Manchmal ist es sehr von Vorteil mitzuprotokollieren, welche Kommandos man in einer Shellsitzung eingegeben hat und welche Ausgaben die jeweiligen Kommandos geliefert haben, z.B. wenn man Vorgehensweisen dokumentieren möchte. Das Tool script schreibt, nachdem es gestartet wird, alle Ein- und Ausgaben in einer Shellsitzung in eine frei benennbare Logdatei, bis der Befehl exit gegeben wird.

Syntax

script [-a] Protokolldateiname <Sitzung> exit

Nützliche Optionen

-a	an bestehende Logdateie anfügen

Beispiele

```
% script /tmp/bashlog.txt
Script startet, file is /tmp/bashlog.txt
% hostname
pc13
% exit
exit
Script done, file is /tmp/bashlog.txt
%
```

Die aufgezeichnete Datei /tmp/bashlog.txt kann dann später weiterverwendet werden.

date - Uhrzeit und Datum anzeigen

Das Kommando date gibt das aktuelle Datum und die Uhrzeit in frei wählbaren Formaten aus.
date gibt, wenn richtig eingestellt, Daten und Uhrzeiten im Standardformat der eingestellten geographischen Region aus. Außerdem kann der root mit date die Systemzeit setzen.

Syntax

date [Optionen] +Format

Nützliche Optionen und Formatanweisungen

-u	Universal Time (bzw. Greenwich Mean Time) ausgeben
+<Format>	folgende Formate sind in beliebiger Kombination möglich:
%a	Abkürzung des Wochentages, Mon-Son
%h	Abkürzung des Monatsnamens, Jan-Dez
%j	Tag des Jahres, 1...366
%d	Tag des Monats, 1...31
%y	zweistellige Jahreszahl, 00...99
%Y	vierstellige Jahreszahl
%H	zweistelliger Stundenwert
%M	zweistelliger Minutenwert
%S	zweistelliger Sekundenwert

usw., s. man page zu date

Beispiele

```
% date
Fri Jun 16 20:14:03 MET 2000
```

gibt das Datum und die Zeit im Standardformat aus.

```
% date +%H:%M:%S
15:43:36
%
```

gibt die Uhrzeit im angegebenen Format inkl. Trennzeichen aus.

Weitergehende Informationen zur Shell

Shells bieten über die reine Kommandoeingabe und -Ausgabe hinaus noch viele weitere Möglichkeiten. Dazu zählen verschiedene eingebaute Kommandos, Verwaltung von Environment-Variablen, Jobkontrolle und Kommando-Historyfunktionen, die die zuletzt eingegebenen Befehle speichern.

Je nach verwendeter Shellvariante unterscheiden sich diese Möglichkeiten jedoch, so daß sie für die meistbenutzten Shells, die Bourne Shell und ihre freie Weiterentwicklung Bourne Again Shell und die C-Shell, separat vorgestellt werden.

Die folgenden Informationen geben eine Übersicht über mögliche Aktionen.

Shellvarianten

sh - The Bourne Shell bzw. bash - The Bourne Again Shell

Die Bourne Shell sh war eine der ersten Shells für Unixsysteme. Sie ist ein de-facto-Standard für Shells unter Unix - auf jedem Unixsystem sollte normalerweise eine sh bzw. eine zur sh kompatible Shell installiert sein. Sie bietet umfassende Möglichkeiten zur Ein-/Ausgabesteuerung, ist klein und schnell, jedoch zur interaktiven Arbeit heutzutage eher weniger geeignet, da ihr wichtige Features, wie z.B. Jobkontrolle fehlen. sh findet jedoch in Shellskripten, also kleinen ausführbaren Batchfiles, immer noch rege Anwendung.

Auf den meisten Linux-Systemen ist heutzutage jedoch eine Weiterentwicklung der Bourne Shell installiert, die Bourne Again Shell bash, die die Vorteile aller existierenden Shells (sh, csh, tcsh, ...) zu kombinieren versucht. Die bash ist weitgehend kompatibel zur sh. Die vorgestellten Methoden beziehen sich auf die modernere bash.

csh - The C Shell

Aus Wunsch nach mehr Komfort bei der Nutzung der Kommandozeile als bei der Bourne Shell wurde die C-Shell csh entwickelt. Sie existiert ebenfalls auf sehr vielen Unix-Systemen und hat sich durch eine an der Programmiersprache C angelehnte Syntax viele Freunde im Anwendungsentwicklerlager gemacht.
Sie bietet auch eine Jobkontrolle und eine Kommando-History, ist aber bei Ein-/Ausgabeaktionen nicht so flexibel wie die bash.

Eingebaute Kommandos

Die meisten Shells bieten eigene Funktionen, die von ihnen selbst ohne Zuhilfenahme externer Unix-Tools abgearbeitet werden.
Die folgenden Kommandos sind eine Auswahl der interessantesten internen Befehle der Shells:

sh bzw. bash

:	Null-Kommando
.	Datei einlesen und gefundene Befehle ausführen
case	fallbezogene Verzweigung
cd	Arbeitsverzeichnis wechseln
echo	Textstring ausgeben
eval	angegebene numerische Ausdrücke evaluieren und Ergebnis ausgeben
exec	angegebenen Befehl außerhalb der Shell ausführen
exit	die aktuelle Shell verlassen
export	Umgebungsvariable global (auch für andere Shellprozesse) setzen
for	von-bis-Schleife
if	wenn-dann-Verzweigung
pwd	aktuelles Verzeichnis anzeigen
read	Zeile von der Tastatur einlesen
set	Umgebungsvariable für die Shell setzen
test	überprüft einen logischen Ausdruck auf Wahrheit
trap	auf Signale warten und dann Kommandos ausführen
umask	Default-Rechtemaske für neue Dateien festlegen
wait	angegebene Zeit warten, dann weitermachen
while	solange-wie-Schleife
[...]	s. man page

csh

alias	einer Befehlskette einen Kurznamen zuweisen
bg	aktuellen Job in den Hintergrund schicken
cd	aktuelles Verzeichnis wechseln
echo	String ausgeben
eval	numerischen Ausdruck auswerten und Ergebnis zurückgeben
exec	externen Befehl außerhalb der Shell ausführen
exit	aktuelle Shell verlassen
fg	einen Hintergrundjob wieder in den Vordergrund holen
foreach	von-bis-Schleife
glob	Dateinamenerweiterung
history	Kommandohistory aufrufen
if	wenn-dann-Verzweigung
jobs	laufende, in der aktuellen Shell gestartete Prozesse anzeigen oder steuern
kill	Prozesse beenden
limit	Zugriff auf Systemressourcen einschränken
nice Kommando	Priorität von Prozessen setzen (Rechenzeitverteilung)
nohup Komm.	Kommando abkoppeln, nicht beenden, wenn ausführende Shell beendet wird
repeat	Kommandos mehrfach wiederholen
set	Shellvariable setzen
setenv	Environmentvariable setzen
source	Datei einlesen und enthaltene Kommandos ausführen
stop	angegebenen Hintergrundjob anhalten
switch	fallbezogene Verzweigung
umask	Default-Rechtemaske für neue Dateien festlegen
unalias	angegebenen Aliasnamen löschen
unset	Shellvariable löschen
unsetenv	Environmentvariable löschen
wait	bestimmte Zeit warten, dann fortsetzen
while	solange-wie-Schleife

[...] s. man page

Umgebungs- und Shellvariablen (environment variables)

Fast alle Shells können mit Variablen umgehen. Es gibt globale und lokale Variablentypen.
Globale Environmentvariablen werden benutzt, um bestimmte Informationen global allen Programmen zur Verfügung zu stellen, wie z.B. den Suchpfad für Programme, User- und Hostnamen, Parameter für bestimmte Software, etc.

Sollen Informationen nicht global, sondern nur innerhalb der Shell, in der man die Variable benötigt, zur Verfügung stehen, kann man nicht-globale Shellvariablen benutzen.

Die Werte aller momentan gesetzten Umgebungsvariablen lassen sich mit dem Kommando env, die Shellvariablen mit dem Kommando set ausgeben.

Einige wichtige Environmentvariablen stellen folgende Informationen global zur Verfügung:

DISPLAY	das X-Grafikdisplay, das genutzt werden soll, z.B. pc12:0.0
EDITOR	der Pfad zum Standard-Editor, z.B. /usr/bin/vi
GROUP	der Name der Gruppe des Users, z.B. marketing
HOME	der Pfad zum Homeverzeichnis des aktuellen Users, z.B. /home/mark
HOST	der Name des Rechners
LOGNAME	der Loginname des aktuellen Users
PATH	Suchpfad für ausführbare Programme, hier wird nach Software, Befehlen und Tools gesucht
SHELL	die Standard-Shell des Users
TERM	Terminal-Typ, z.B. xterm, vt100
USER	Username des aktuellen Benutzers

Einige der systemweiten Environmentvariablen werden bereits beim Login von den login scripts auf sinnvolle Werte gesetzt, die aber jederzeit verändert werden können.

In der Shell können die Werte der Variablen auch in Kommandos oder Skripten verwendet werden.

Um den Wert einer Variable anzusprechen, genügt es, sie mit einem vorangestellten Dollarzeichen ($) in Befehlen zu verwenden, z.B. gibt "echo $DISPLAY" den Wert der Variablen DISPLAY aus.

csh und sh bzw. bash verwenden unterschiedliche Mechanismen um derartige Variablen zu behandeln, daher stellen wir die Möglichkeiten gesondert vor.

sh bzw. bash

In den Bourne Shells werden Variablen nach folgenden Schemata gesetzt:

- globale Environmentvariablen mit export:
  % export DISPLAY="pc12:0.0"
  setzt den neuen Wert und "exportiert" die Variable DISPLAY (X-Window) für alle Programme.

- lokale Shellvariablen durch einfache Zuweisung:
  % TEST=abcdefgh1234567890
  weist der lokalen Variable TEST die Zeichenfolge wie angegeben zu.
  Existiert die Shellvariable noch nicht, wird sie dynamisch angelegt.

Ein Löschen der Variablen ist mit dem Kommando unset möglich.

csh

In der C-Shell wird für die gleichen Aufgaben eine geringfügig andere Notation zum Setzen von Variablen verwendet:

- globale Environmentvariablen
  % setenv DISPLAY "pc12:0"

- lokale Shellvariablen
  % set TEST "abcdefgh1234567890"

Löschen der Variablen geschieht in der C-Shell mittels unsetenv.

Login Scripts bzw. Shell-Konfiguration

Die meisten Shells lassen sich über sogenannte Login Scripts und Kofigrationsdateien beeinflussen. Sie haben je nach eingesetzter Shell verschiedene Namen, die teils nach der Form .<Shellname>rc aufgebaut sind, also z.B. .bashrc. "rc" steht dabei für Ressource Configuration.

Zu unterscheiden ist zwischen den Konfigurationsskripten, die beim Start *jeder* neuen Shell verarbeitet werden, und den Login-Skripten, die lediglich ausgeführt werden, wenn die neue Shell zum Einloggen im System genutzt wird. Diese Trennung ermöglicht, bestimmte Einstellungen, die nur bei Benutzung über ein Netzwerk Sinn machen, z.B. das Setzen verschiedener Terminaltypen etc., nur bei Bedarf auszuführen.

In diese Konfigurationsdateien können Voreinstellungen, die alle neu gestarteten Shells betreffen, eingetragen werden. So werden z.B. das X-Display, Suchpfade oder andere Variablen gesetzt, das Verhalten und die Optik der Shell eingestellt, andere Skripte (für die Vorkonfiguration weiterer externer Softwarepakete, z.B. Compiler) oder Kommandos gestartet oder Dienste, wie z.B. Virenscanner, aktiviert. Auch Kurzbefehle (Aliase), die bestimmte, immer wiederkehrende, längere Kommandos ausführen, können hier definiert werden.

In der Regel existieren auf Unix-Systemen systemweite Konfigurationsdateien, die das Verhalten der meisten Shells konsistent voreinstellen. Sie werden vor der Abarbeitung der usereigenen login scripts von allen neuen Shells ausgeführt und liegen meist im Verzeichnis /etc. Sie können jedoch nur von root verändert werden.

Eine komplette Abhandlung der Shellkonfiguration und Login-Verwaltung sprengt den Rahmen dieses Grundkurses. Einen guten Ausgangspunkt für eigene Recherchen bilden die manual pages der Shells.

sh bzw. bash

Die User-Konfigurationsdatei muß .profile (sh) oder .bashrc (bash) heißen und im Homeverzeichnis des Users liegen. Die systemweite Konfigurationsdatei für die Bourne Shells ist /etc/profile.

Ein typisches .bashrc-File kann z.B. folgende Elemente enthalten:

```
[...]
test -z "$PROFILEREAD" && . /etc/profile
# Some people like DOS like aliases
if test -f /etc/profile.dos ; then
        . /etc/profile.dos
fi
alias rm='rm -i'
export EDITOR=/usr/bin/pico
export NNTPSERVER=news.meak.de
# commands common to all logins
if ! [ $TERM ] ; then
    eval `tset -s -Q`
    case $TERM in
      con*|vt100) tset -Q -e ^?
        ;;
    esac
fi
#
# try to set DISPLAY smart (from Hans) :)
#
if test -z "$DISPLAY" -a "$TERM" = "xterm" -a -x /usr/bin/who ; then
    WHOAMI="`/usr/bin/who am i`"
    _DISPLAY="`expr "$WHOAMI" : '.*(\([^\.][^\.]*\).*)'`:0.0"
    if [ "${_DISPLAY}" != ":0:0.0" -a "${_DISPLAY}" != " :0.0" \
         -a "${_DISPLAY}" != ":0.0" ]; then
        export DISPLAY="${_DISPLAY}";
    fi
    unset WHOAMI _DISPLAY
fi
test -e ~/.alias && . ~/.alias
export DISPLAY LESS PS1 PS2
umask 022
[...]
```

Man sieht, daß auch in den Shell-Startskripten normale Unixbefehle verwendet werden. Kommentare sind mit vorangestelltem Hash # möglich.

Das Verhalten bestimmter Funktionen der Bourne Shells wird über diverse Variablen (hier z.B. LESS, PS1, PS2, ...) in den Skripten eingestellt. Welche Variable welche Aktion auslöst, verraten die man pages zu sh und bash.

csh

Ähnlich sieht es bei der C-Shell aus. Die systemweiten Voreinstellungsdateien für den interaktiven Modus und für Logins sind i.d.R. /etc/csh.cshrc und /etc/csh.login. Die Konfigurationsdatei des Users muß .cshrc heißen, wird ebenfalls im Userhome abgelegt, ist jedoch anders aufgebaut als die der Bourne Shells:

```
[...]
setenv OPENWINHOME /usr/openwin
set tmp=/usr/man
foreach man ( /usr/local/man \
              /usr/X11R6/man \
              /usr/openwin/man )
        if ( -d $man ) then
           set tmp=${tmp}:${man}
        endif
end
setenv MANPATH $tmp
unset  tmp man
setenv HOSTNAME "`hostname -f`"
setenv HOST     "`hostname -s`"
if ( -f /etc/organization ) then
   setenv ORGANIZATION "`cat /etc/organization`"
endif
[...]
```

Bei der Nutzung der C-Shell als Login-Shell wird die Datei .login im Homeverzeichnis ausgeführt, die z.B. folgende Logik zum Setzen des korrekten Terminaltypen, der Default-Rechtemaske und des Mailverzeichnisses bei Netzwerkanmeldungen enthalten kann:

```
[...]
if ( -o /dev/$tty && ${?prompt} ) then
    # Console
    if ( ! ${?TERM} )           setenv TERM linux
    if ( "$TERM" == "unknown" ) setenv TERM linux
    # No tset available on SlackWare
    if ( -x "`which stty`" ) stty sane cr0 pass8 dec
    if ( -x "`which tset`" ) tset -I -Q
    unsetenv TERMCAP
    settc km yes
endif
umask 022
setenv SHELL /bin/tcsh
if ( -f /var/spool/mail/$USER ) then
    setenv MAIL /var/spool/mail/$USER
    set mail=$MAIL
endif
[...]
```

Eine umfassende Anleitung zum Erstellen eigener Login- und Config-Skripte für die C-Shell liefert die man page zu csh.

Jobkontrolle

Moderne Shells bringen Methoden zur Verwaltung mehrerer Prozesse, die im Multitaskingbetrieb parallel ablaufen können, mit. So wird dem Benutzer die Möglichkeit gegeben, Programme oder Kommandos, die normalerweise die aktuelle Shell blockieren würden, in den Hintergrundbetrieb zu schicken, so daß sie keinerlei Ausgaben auf den Bildschirm mehr produzieren. Andererseits können Hintergrundprozesse wieder in den Vordergrund geholt werden, um sie zu beoachten, zu beenden oder mit ihnen zu interagieren.

Zu beachten ist, daß je Shell nur ein Prozeß im Vordergrund, also in Interaktion mit dem User, laufen kann, und daß nur Interaktion mit dem Prozeß im Vordergrund möglich ist (Näheres s. "Ein-/Ausgabeumleitung").

Bei den meisten Shells, die Jobkontrolle unterstützen, haben sich folgende (Tastatur-)Kommandos durchgesetzt:

jobs	zeigt die laufenden Kindprozesse der aktuellen Shell an
STRG-Z	stoppt den aktuellen Vordergrundprozess, um ihn mit fg, bg oder kill zu steuern
fg %<Job-ID>	holt Job <Job-ID> in den Vordergrund
bg %<Job-ID>	schickt Job <Job-ID> in den Hintergrund
<Kommando> &	startet <Kommando> direkt im Hintergrund
kill %<Job-ID>	beendet Prozeß <Job-ID>

Kommando-History

Die C-Shell, die Korn-Shell und neuere Bourne Shells (bash) speichern die zuletzt eingegebenen Kommandos des Benutzers, damit er schnell wieder darauf zurückgreifen kann. Dies ist besonders praktisch, wenn ein identische oder ähnliche Kommandos häufig eingegeben werden müssen.
Wie diese sogenannte Kommando-History zu bedienen ist, hängt von der eingesetzten Shell ab. Es hat sich jedoch das Kommando history etabliert, daß die Liste früher eingegebener Befehle anzeigt, z.B. hier die letzten fünf:

```
% history 5
52 cd marketing
53 ls
54 cp * /Backup
55 chmod -R 644 /Backup/
56 rm marketing
```

Auf diese numerierten (und über diese Nummer referenzierbaren) früheren Kommandos erneut aufzurufen, genügt folgende Syntax:

```
% !<Eintragsnummer des Befehls>
```

also z.B.:

```
% !53
ls
[...]
```

ruft den Befehl Nr.53 erneut auf, in diesem Fall ls.

Der letzte Befehl läßt sich mittels !! wiederholen.

Weitere Hilfen, wie die Suche in der History (STRG-R in der Shell) etc., werden ausführlich in den man pages beschrieben.

Shell wechseln

Will man andere Shells als die standardmäßig eingestellte benutzen , so hat man zunächst die Möglichkeit, ausgiebig zu testen, welche der Shells als neue Standardshell in Frage kommt. Da Shells im Prinzip normale Programme sind, können sie natürlich auch einfach aufgerufen werden.
Ein anderer Prompts deutet dann meistens an, daß die neue Shell aktiv ist.
Beenden kann man die Shell, wie gehabt, mit dem Kommando exit, und findet sich dann wieder in der alten Shell.

Um die neue Shell dauerhaft einzutragen, gibt es die Kommandos chsh (change shell) und passwd -e (bzw. -s bei BSD-artigen Systemen). Bitte überprüfen Sie vor dem Ändern der Standardshell immer die man pages der entsprechenden Kommandos, um einem Chaos beim nächsten Login vorzubeugen.

Grundsätzlich sollte immer der komplette Pfad zur gewünschten Shell eingetragen werden, um Inkonsistenzen bei unterschiedlich gesetzten Suchpfaden etc. zu vermeiden.

Das System wird desweiteren nur Logins über Shells akzeptieren, die in der Konfigurationsdatei /etc/shells gelistet werden.

Spezielle Unix-Features

Unixsysteme bieten insbesondere im Bereich der Textein-/ausgabe, der Programmdarstellung und -kombination und der effizienten Arbeit in der Kommandozeile erhebliche Vorteile gegenüber weniger textbasierten Systemen wie z.B. Dos/Windows. Durch geschickte Ein-/Ausgabeumleitungen, Kommandoverkettungen und Suchmusterbenutzung können viele kleine Tools im Zusammenspiel komplexe Aufgaben erledigen. Jedes Tool erledigt dabei seine Teilaufgabe und schickt seine Ergebnisse als Eingabewerte an das nächste Tool.

Diese Eigenschaft in Kombination mit der Tatsache, daß alle Daten unter Unix als Datei angesprochen werden, ermöglicht eine einheitliche Schnittstelle zwischen Programmen aller Art: Daten können **in** Dateien, den Bildschirm, den Drucker etc. geschrieben werden, bzw. **aus** diversen Dateien, der Tastatur, anderen Prozessen usw. ausgelesen werden, und das immer mit denselben Standardmechanismen.

Einige dieser Möglichkeiten werden wir nun vorstellen.

Ein-/Ausgabekanäle (stdin, stdout, stderr)

Jedes Unix-System bietet drei numerierte "Standardkanäle", sog. Dateideskriptoren, über die das System mit dem Benutzer und Programme untereinander kommunizieren:

- stdin (0) Standardkanal für Eingabe in Programmen
- stdout (1) Standardkanal für Ausgaben aus Programmen
- stderr (2) Standardkanal für Fehlermeldungen

Normalerweise kommt der Input für bestimmte Programme von der Tastatur oder aus Dateien, und die Ausgabewege, also stdout und stderr, liefern ihre Ausgaben in der Regel auf einem Bildschirm. Diese Standards können aber durch Umleitungen verändert werden, wenn z.B. keine Bildschirmausgabe gewünscht wird, jedoch eine Logdatei angefertigt werden soll, die alle Ausgaben des jeweiligen Programms enthält (Ausgabeuml.), oder ein Programm Daten als Parameter übernehmen soll, die von anderen Programmen zuvor ermittelt wurden (Eingabeuml.). Darüber hinaus können vom User weitere File Descriptors zu eigenen Zwecken angelegt werden. Näheres dazu verrät die man page der benutzten Shell.

Die Standard-Filedeskriptoren lassen sich unter Angabe ihrer Nummer (s.o. in Klammern) referenzieren, so daß gezielte Ein-/Ausgabeumleitungen o.ä. möglich werden.

Ein-/Ausgabeumleitung, Pipes und Kommandoverkettung

Es gibt eine ganze Menge Operatoren für die Kombination beliebiger Kommandos, daher ist bei weitergehender Anwendung der vorgestellten Techniken ein Blick in weitere Literatur zur verwendeten Shell unumgänglich.

Generell gilt:
Eingabeumleitung liest die Eingabewerte für Programme aus Dateien,
Ausgabeumleitung sorgt dafür, daß Kommandoausgaben in Dateien geschrieben werden.
Pipes sorgen dafür, daß die Ausgabe eines Programmes einem anderem Programm als Eingabe zur Verfügung gestellt wird, ohne den Umweg über Dateien zu nehmen.

Die folgende Tabelle faßt einen Teil der Möglichkeiten zusammen:

Symbol	**Umleitung**
>	Ausgabe in Datei umleiten, bestehende Dateien werden überschrieben
>>	Ausgabe in Datei umleiten, Daten werden an bestehende Dateien angehängt, wenn vorhanden
|	"Pipe", legt "Leitung" zwischen Ausgabe des ersten und Eingabe des zweiten Programms
<	Eingabe aus anderem Programm umleiten
<<Zeichenkette	Eingabe vom Standard Input stdin holen, bis Zeichenkette in eigener Zeile eingegeben wird
Kommando1; Kommando2; ...	mit ; werden Kommandos in der Zeile separiert
Kommando &	startet Kommando im Hintergrund

Komm1 && Komm2	Komm2 wird ausgeführt, wenn Komm1 erfolgreich war (UND-Verkettung)
Komm2 || Komm2	Komm2 wird ausgeführt, wenn Komm1 nicht erfolgreich war (ODER-Verkettung)
(Kommandos)	in runden Klammern genannte Befehle werden gruppiert und in einer eigenen Subshell ausgeführt
' Zeichenkette '	einfache Anführungszeichen, Sonderzeichen innerhalb der Zeichenkette nicht als Shellsysntax interpretieren, alles unverändert ausgeben
" Zeichenkette "	doppelte Anführungszeichen, erlaubt Angabe von Zeichenketten inkl. enthaltener Variablen.
\Zeichen	Zeichen "escapen", d.h. darstellen, seine eigentlichen Funktion in der Shell ignorieren und als normales Zeichen verwenden
` Kommando `	Kommandosubstitution, setzt Output des Kommandos als Wert ein
#	Kommentarzeile, bis zum nächsten newline

Beispiele

Ausgabeumleitung:
% cat datei1 datei2 > datei3
gibt datei1 und datei2 aus, alle Ausgaben werden in datei3 gespeichert. Wenn datei3 bereits existiert, wird sie überschrieben.

% cat datei2 >> datei3
datei2 wird an datei3 angehängt.

Manchmal ist es erwünscht, daß alle Ausgaben eines Programms, die normalerweise in einer Konsole landen würden, in Dateien umzuleiten und jegliche sofort sichtbare Ausgabe zu unterdrücken.
Die Shells bieten dazu mehr oder weniger komfortable Hilfsmittel. Am geeignetsten für derartige Aktionen ist wahrscheinlich die bash, die nach folgendem Schema arbeitet:

Filedeskriptornummer> Datei

z.B.

% ps 2> ps_errors.txt
leitet die Fehlermeldungen (stderr, Standard-Deskriptor Nr.2) in die angegebene Datei um.

% ps 2>&1 > ps_output.txt
leitet alle Ausgaben (stdout (1) und stderr (2)) in die Datei ps_output.txt.

% ps 2>&1 | grep netscape
piped alle Ausgaben des Kommandos ps in das Programm grep.

Eingabeumleitung:
% less < datei3
datei3 wird dem Kommando less als Eingabe eingeleitet bzw. less holt sich die Eingabe aus datei3.

Kommandoverkettung mit Pipes:
% ps | grep netscape
die Ausgabe des ersten Programms, hier ps (Prozeßstatistik), wird an das zweite Programm, grep (Suche nach dem String "netscape") weitergeleitet.

Bedingte Verkettung:
% cat /tmp/datei.txt && echo "Datei wurde angezeigt."
gibt die Statusmeldung nur aus, wenn die Datei ausgegeben werden konnte (UND-Verkettung)

% cat /tmp/datei.txt || echo "Fehler bei der Anzeige der Datei."
gibt die Fehlermeldung aus, wenn die Datei nicht ausgegeben werden konnte (ODER-Verkettung)

Wild Cards - Suchmuster

Die Shell und einige Textverarbeitungskommandos erlauben die Angabe von einfachen Suchmustern, die während der Ausführung durch die Ergebnisse der Suche (nach Dateien, Strings, etc.) ersetzt werden.

Mit diesen einfachen Suchmustern, die in der Regel nur einen zusammenhängenden String fassen können, sind folgende Operationen möglich:

Wild Card	**Funktion**
?	einzelnes beliebiges existentes Zeichen
*	beliebig lange Zeichenkette, auch Länge Null
[abc...]	beliebiges Zeichen aus Liste
[a-e]	beliebiges Zeichen aus Bereich
[!def]	beliebige Zeichen, die nicht in nachstehendem Ausdruck enthalten sind, suchen [nur sh]
{abc,bcd,def,...}	beliebige Zeichenfolgen suchen [nur csh]
~	Homeverzeichnis des aktuellen Users
~User	Homeverzeichnis beliebiger User

In welchem Umfang die Wild Cards unterstützt werden, hängt von der verwendeten Shell ab.

Beispiele

% ls [abcdABCD1234]text.txt
listet alle Dateien, deren Name mit a,b,c,d,A,B,C,D,1,2,3 oder 4 beginnt und direkt von text.txt fortgesetzt wird, also z.B. die Datei Dtext.txt

% cp [a-z].dat /Backup
kopiert alle Dateien, die kleingeschriebene, ein Zeichen lange Dateinamen haben, in das Verzeichnis /Backup.

% rm /tmp/[A-Z]*.tmp
löscht alle Dateien unterhalb von /tmp, die mit einem Großbuchstaben beginnen und beliebig lange, auf .tmp endende Dateinamen haben.

Textbearbeitung

Das vorteilhafteste Feature aller Unixsysteme ist der gelungene Ansatz, mit Texten, Textdateien usw. umzugehen. In diesem Kapitel werden daher entgegen der Erwartung der meisten Leser bei dem Titel "Textverarbeitung" nicht gängige Office-Pakete oder Editoren, sondern Unix-Mittel und -Kommandos zur direkten, mittels Skripten automatisierbaren Textverarbeitung vorgestellt.

Die im Folgenden behandelten Kommandos sind äußerst flexible Spezialistentools, die eine Unmenge an verschiedenen Optionen, Parametern, Suchmustern usw. verarbeiten können und leider teilweise auch erfordern. Daher sei wieder einmal der intensive Blick in die dementsprechenden Manual-Seiten ans Herz gelegt.

Da die von fast allen Textverarbeitungskommandos benutzte Syntax zur Mustersuche standardisiert und daher identisch ist (die sog. Regular Expressions), stellen wir zunächst diese ansatzweise vor.

Reguläre Ausdrücke - erweiterte Suchmuster

Einige mächtige Textprocessing-Kommandos sowie moderne Shells akzeptieren die Expansion bestimmter weitergehender Suchmuster. Sie werden Reguläre Ausdrücke (Regular Expresssions) genannt und sind nicht, wie die simplen Wild Cards, auf einzelne zusammenhängende Zeichenketten beschränkt.

Ein regulärer Ausdruck kann aus einer Mixtur verschiedener Bestandteile bestehen, so z.B. auch mehreren Kriterien, die erfüllt sein müssen, Kombinationen aus normalen Zeichen und sog. Meta-Zeichen (die die Suchregeln beeinflussen), usw.

Reguläre Ausdrücke kennen drei Grundformen:

- Anchors (Anker) bestimmen die Position eines Suchmusters (z.B. Zeilenanfang)
- Character Sets (Zeichenfolgen) suchen an bestimmten Positionen nach Zeichenfolgen
- Modifiers (Anweisungen) geben an, wie oft der vorhergehende reg. Ausdruck wiederholt wird

Die (unvollständige) Liste möglicher Elemente regulärer Ausdrücke ist wie folgt:

.	jedes einzelne Zeichen außer newline fassen
*	null oder mehr Vorkommen des folgenden einzelnen Zeichens erfassen
[abc]	beliebiges einzelnes der angegebenen Zeichen erfassen
[a-d]	beliebiges einzelnes Zeichen aus angegebenem Bereich erfassen
[^Ausdruck]	jedes Zeichern erfassen, das nicht vom folgenden Ausdruck erfasst würde
^abc	der reguläre Ausdruck muß am Zeilenanfang stehen (Anker)
abc$	der reguläre Ausdruck muß am Zeilenende stehen (Anker)
\	Escapezeichen; nächstes Zeichen nicht als Kommandozeichen auffassen
\{n,m\}	den vorhergehenden reg. Ausdruck minimal n, maximal m mal erfassen (n,m=0...255)
\<abc\>	erfasst den enthaltenen regulären Ausdruck nur, wenn er als separates Wort vorkommt

[...] komplette Liste: s. man page "ed"

Beispiele

Regulärer Ausdruck	**erfasst**
cat	den String "cat"
.at	beliebigen Buchstaben, gefolgt von at, z.B. fat, cat, Rat
xy*z	jedes Vorkommen von x, gefolgt von null oder mehr y's, gefolgt von einem z
^cat	"cat" am Zeilenanfang
cat$	"cat" am Zeilenende
*	jedes Vorkommen des Sterns "*"
[cC]at	"cat" bzw. "Cat"
[^a-zA-Z]	alle nicht-alphabetischen Zeichen
[0-9]$	jede Zeile, die mit einer Zahl endet
[A-Z]*	null oder mehr Großbuchstaben (also: alles!)
[A-Z][A-Z]*	einer oder mehr Großbuchstaben

Textverarbeitungskommandos

Die folgenden Programme, grep, sed und awk, mit denen sich so gut wie jede Arbeit und Suche in beliebig langen und komplexen Textdateien (ASCII) bewerkstelligen läßt, können auf Wunsch regen Gebrauch von den o.g. Regular Expressions machen:

Kommando	**Wofür?**
awk	in Dateien nach Suchmustern scannen und Resultate verarbeiten
grep	das Argument (Datei oder stdin) nach Mustern durchsuchen und Ergebnisse ausgeben
sed	"Stream Editor", zum Editieren von Dateien von der Kommandozeile oder Skripten aus

grep

grep filtert aus einem Datenstrom, der aus Dateien oder der Standardeingabe stdin kommen kann (z.B. Pipe aus anderem Programm), Zeilen heraus, die bestimmten Suchmustern entsprechen, und gibt das Ergebnis aus.

Syntax

grep [Optionen] regexp [Dateinamen]

Nützliche Optionen

-i	ignore case, Groß-/kleinschreibung ignorieren
-c	nur die Anzahl der gefundenen Zeilen anzeigen, nicht die Treffer
-v	Suche invertieren, das Gegenteil des Suchergebnisses ausgeben
-n	Zeilennummern anzeigen
-l	keine Zeilen, sondern Dateinamen, in denen Ausdruck gefunden wird, anzeigen

Beispiele

% grep -c bash /etc/passwd
gibt die Anzahl der in der Paßwortdatei /etc/passwd eingetragenen Zeilen aus, in denen das Suchwort "bash" vorkommt. So findet man heraus, wie viele Benutzer die bash als Login-Shell benutzen.

% cat /etc/passwd | grep -c bash
erledigt die gleiche Aufgabe mit einer Pipe, die die Ausgaben des Kommandos cat zum Standardeingabekanal stdin des Tools grep leitet.

% grep -l "1[234]" /etc/*
listet alle Dateien im Verzeichnis /etc, in denen die Zahlenkombinationen 12, 13 oder 14 vorkommen.

sed, awk

sed, der nicht-interaktive Stream-Editor, bearbeitet einen Eingabestrom zeilenweise, ändert gewünschte Bestandteile der Zeile (je nach sog. Edit-Kommando), und gibt die neuen Zeilen in die Standardausgabe stdout aus.

awk ist die Königslösung zur skriptgesteuerten Textverarbeitung. Es bietet eine eigene "Programmiersprache", mit der unter Zuhilfenahme regulärer Ausdrücke jede Verarbeitung gelöst werden kann.

Diese beiden Programme vorzustellen sprengt den Rahmen dieses Grundkurses. Weitere Informationen finden Sie in den man pages zu den Tools und in der unten angefügten Literaturliste.

Weitere nützliche Kommandos

Dateiverwaltung

Es gibt viele weitere Tools zur Verwaltung, zum Ver- und Abgleich und zur Analyse von Daten:

Kommando	**Wofür?**
cmp	Dateiinhalte vergleichen und Position der Unterschiede anzeigen
cut	Teile von Textzeilen auswählen
diff	Unterschiede zwischen Dateien anzeigen
paste	Dateien aneinanderhängen
touch	neue Dateien anlegen bzw. Dateidatum setzen
wc	Wort-, Zeilen- oder Zeichenanzahl zählen
ln	Dateisystem-Verknüpfungen anlegen
sort	Daten sortieren
tee	Kommandoausgabe kopieren
uniq	doppelte Zeilen löschen
file	Dateityp ermitteln
find	Dateien nach verschiedenen Kriterien finden

cmp - Dateiinhalte vergleichen

cmp vergleicht Dateien und zeigt die Position eventuell gefundener Unterschiede an.

Syntax

cmp [Optionen] Datei1 Datei2 [Offset1] [Offset2]

Nützliche Optionen

-l nicht nur den ersten, sondern alle Unterschiede anzeigen

Beispiele

Angenommen, man vergleicht die folgenden Dateien:

datei1.txt	**datei2.txt**
karl	karl
meike	m*ark*

klaus *elke*
... ...

% cmp datei1.txt datei2.txt
datei1.txt datei2.txt differ: char 7, line 2

zeigt an, daß der erste Unterschied in Zeile 2, genauer beim 7. Zeichen (nach k, a, r, l, newline, m), auftritt.

cut - Teile einer Zeile auswählen

cut ist immer dann nützlich, wenn Teile einer Textzeile separiert und weiterverarbeitet werden sollen.

Syntax

cut [Optionen] Datei

Nützliche Optionen

-c <Bereich> gibt die Zeichen in <Bereich> aus
-b <Bereich> gibt die Bytes in <Bereich aus
-d <Zeichen> benutzt <Zeichen> als Trennzeichen (Default: <TAB>)
-f Feld1,Feld2,... gibt die durch Trennzeichen (s.o.) getrennten Felder aus

Beispiele

angenommen, die Datei logins.monday enthält folgende Daten (getrennt durch <TAB>):
jdoe John Doe 195.23.99.12
lsmith Laura Smith 194.68.224.43
kmyers Kurt Myers 212.2.220.19
[...]

% cut -f 2,3 logins.monday
John Doe 195.23.99.12
Laura Smith 194.68.224.43
Kurt Myers 212.2.220.19
gibt nur die Spalten 2 und 3 der Datei aus.

touch - neue Dateien anlegen

Mit dem Kommando touch legt man, sofern noch keine Datei unter dem gewünschten Namen vorhanden ist, eine neue leere Datei an. Andernfalls werden die Daten der letzten Änderung und des letzten Zugriffs auf den aktuellen Zeitpunkt gesetzt. Datumsinformationen können mit touch flexibel gesetzt werden, um z.B. Versionskontrollen o.ä. zu implementieren.

Syntax

touch [Optionen] Dateiname

Nützliche Optionen

-a nur Datum des letzten Zugriffs setzen
-c keine Dateien anlegen
-m nur Datum der letzten Änderung setzen
-r, --reference=Datei diese Datei als Referenz für Datum benutzen

Beispiele

% touch /tmp/datei.tmp
legt die angegebene Datei an.

wc - Wortanzahl ausgeben

wc zählt die Anzahl Wörter, Zeilen, Bytes oder Zeichen in einer Textdatei.

Syntax

wc [Optionen] Datei

Nützliche Optionen

-c	Bytes zählen
-m	Zeichen zählen
-l	Zeilen zählen
-w	Wörter zählen

Beispiele

angenommen, die Datei logins.monday enthält folgende Daten:

```
jdoe    John Doe      195.23.99.12
lsmith  Laura Smith   194.68.224.43
kmyers  Kurt Myers    212.2.220.19
```

```
% wc -lw logins.monday
3       12      logins.monday
```

gibt die Anzahl der Zeilen und die der Wörter aus.

ln - Verknüpfungen anlegen

Eine wichtige Funktion im Linux-Dateisystem ist das Anlegen von Links.
Links sind sehr praktisch, um einer Datei mehrere Namen zuzuweisen oder Abkürzungen im Dateisystembaum anzulegen.

Syntax

ln [Optionen] Quelle Ziel

Quelle	Bestehende Datei/Verzeichnis
Ziel	Name des Links

Nützliche Optionen

-s	**symbolischen Link** anlegen

Ohne Option -s wird ein sog. **Hard Link** auf Dateien oder Verzeichnisse (nur für User root) erzeugt. Ein Hard Link kann nur auf bestehende Dateien oder Verzeichnisse zeigen. Er stellt einen neuen Verzeichniseintrag dar, der auf denselben **Inode** verweist.

Bei einem **Symbolic Link** wird indes nur ein "softer" Link angelegt, eine Datei des Typs "link", in der auf den Inode der Quelle verwiesen wird.

Beispiele

Wenn z.B. für eine Software eine neue Version eingespielt wurde und sich die Namen wichtiger Dateien oder Verzeichnisse geändert haben, kann man symbolische Links mit den alten Namen anlegen, die auf die neuen Daten zeigen. Dadurch wird gewährleistet, daß ein Zugriff auf die neuen Daten sowohl unter dem alten als auch unter dem neuen Namen möglich ist.

```
% ln -s emacs-20 emacs
```

legt einen symbolischen Link namens emacs (dem alten Namen) an, der auf die Datei emacs-20 zeigt.

sort - Sortieren

sort sortiert Dateien nach unzähligen einstellbaren Kriterien.
Die Standardeisntellung in Kombination mit einigen einfachen Optionen genügt allerdings meistens, um sinnvolle Textsortierung durchzuführen.

Syntax

sort [Optionen] [+Position1 [-Position2]] Datei

Nützliche Optionen

-n	numerisch sortieren
-r	reverse, Sortierung umkehren

Beispiele

Folgende Datei namens datei.txt enthält die Zeilen:

```
jdoe    John Doe        195.23.99.12
lsmith  Laura Smith     194.68.224.43
kmyers  Kurt Myers      212.2.220.19
```

```
% sort datei.txt
jdoe    John Doe        195.23.99.12
kmyers  Kurt Myers      212.2.220.19
lsmith  Laura Smith     194.68.224.43
```

sortiert die Daten zeilenweise alphabetisch nach der ersten Spalte.

uniq - doppelte Zeilen löschen

uniq löscht doppelte aufeinander folgende Zeilen. Es sucht nach doppelten Einträgen und schreibt die bereinigte Version in eine neue Datei.

Syntax

uniq [Optionen] Datei NeueDatei

Nützliche Optionen

-d nur die doppelten Zeilen speichern
+n die ersten n Zeichen ignorieren

Beispiele

Die Datei logins.monday enthält doppelte Zeilen:

```
jdoe    John Doe        195.23.99.12
kmyers  Kurt Myers      212.2.220.19
kmyers  Kurt Myers      212.2.220.19
lsmith  Laura Smith     194.68.224.43
```

Um sie herauszufiltern, nutzt man folgenden Befehl:

```
% uniq logins.monday logins.monday.neu
```

Die Unterschiede zeigen sich bei der Ausgabe der neuen Datei:

```
% cat logins.monday.neu
jdoe    John Doe        195.23.99.12
kmyers  Kurt Myers      212.2.220.19
lsmith  Laura Smith     194.68.224.43
```

file - Dateityp ermitteln

file ermittelt anhand der Informationen, die in einer Tabelle über verschiedene Dateiarten gespeichert sind (in der Datei /etc/magic), welchen Typs eine Datei ist. Dies hilft bei der Ermittlung des Zwecks von Daten.

Syntax

file [Optionen] Datei

Nützliche Optionen

-c Magic-Datei auf Fehler überprüfen (zur Wartung)
-f Datei file list, Datei enthält Liste zu überprüfender Dateien
-m Magic-Datei angegebene Magic-Datei statt /etc/magic verwenden

Beispiele

Hier einige Beispieldateien:

```
% file webcam.html
webcam.html: empty
```

```
% file public_html/
public_html/: directory
```

```
% file linuxbuch.ps
linuxbuch.ps: PostScript document text conforming at level 2.0

% file linuxbuch.pdf
linuxbuch.pdf: PDF document, version 1.2

% file pppoed0.46.tgz
pppoed0.46.tgz: gzip compressed data, deflated, last modified: Sun Feb 27 09:57:33 2000, max
compression, os: Unix
```

find - Dateien finden

find durchsucht ganze Verzeichnisbäume nach Dateien, die bestimmte angegebene Kriterien wie z.B. Datum, Größe, Name o.ä. erfüllen.

Syntax

find Verzeichnis [Suchoptionen] [Aktionen]

Nützliche Optionen

-atime n	access time, Zeit des letzten Zugriffs vor n Tagen
-ctime n	change time, Zeit der letzten Änderung des Dateistatus vor n Tagen
-mtime n	modify time, Zeit der letzten Änderung des Dateiinhalts vor n Tagen

Notation der Tage n bei o.g. Datumsoperationen:

+n	mehr als n Tage
n	genau n Tage
-n	weniger als n Tage

-newer Datei	nur neuere Dateien als <Datei>
-user User	nur Dateien von <User>
-group Gruppe	nur Dateien von <Gruppe>
-perm Mode	nur Dateien, deren Zugriffsrechte <Mode> ist
-ls	long listing der gefundenen Dateien anzeigen
-print	gefundene Dateien anzeigen (Default bei SysV, nicht bei BSD)
-exec Kommando	wendet <Kommando> auf gefundene Dateien an. **VORSICHT:** werden die falschen Dateien ergriffen, kann ein destruktives Kommando (rm o.ä.) fatale Folgen haben (kein Un-Delete)!!

[...] s. man page "find"

Beispiele

```
% find . -name \*ar\* -print
./library
./arbeitsplan.xls
./marketingkonzept.pdf
%
```

findet alle Dateien, in deren Namen der String "ar" an beliebiger Stelle vorkommt.

```
% find . -user chris -ls
288367   4 drwxr-xr-x   3 chris    users        4096 Jun 21 21:22 .
288368   1 -rw-r--r--   1 chris    users          10 Jun 16 12:30 ./.zsh
288369   6 -rw-r--r--   1 chris    users        5376 Jun 16 12:30 ./.gimprc
288370   1 -rw-r--r--   1 chris    users         341 Jun 16 12:30 ./.vimrc
288371   1 -rw-r--r--   1 chris    users          92 Jun 21 21:22 ./test.neu
288372   5 -rw-r--r--   1 chris    users        5066 Jun 16 12:31 ./webcam.html
288373   1 -rw-r--r--   1 chris    users         183 Jun 21 21:21 ./test
288374   1 -rw-r--r--   1 chris    users         183 Jun 21 21:22 ./test.sort
```

% findet im aktuellen Verzeichnis (.) alle Dateien des Users chris und zeigt sie im long format an (-ls).

Datenarchivierung und -kompression

Oft steht man vor der Aufgabe, Projektdaten oder ähnliche nur für eine bestimmte Zeit benötigte Daten, aus Platzgründen auf Backupmedien verschieben zu müssen. Wenn man Daten auf diese Art verlagert, sollte man sichergehen, daß sie im Notfall schnell wiederhergestellt werden können.

Zu diesem Zweck gibt es Unix-Programme, die komplette Unterverzeichnisse inklusive Pfadinformationen in einer einzigen Archivdatei unterbringen - ein Vorgang ähnlich zum Komprimieren mit zip, arj, rar etc. unter anderen Systemen. Diese Archivdateien lassen sich dann in einem weiteren Schritt komprimieren, auf Korrektheit testen und anschließend ruhigen Gewissens auf ein Backup-Speichermedium (CD-Rom, Tape, etc.) verschieben.

In der Regel arbeiten zur Archivierung von Daten mehrere kleinere Expertentools zusammen.

Folgende Programme sind weit verbreitet:

Kommando	**Wofür?**
tar	tape archiver, erzeugt einzelne Archivdateien, in denen Daten gesichert werden können; mit tar erzeugte Archivdateien haben die Endung .tar
gzip / gunzip	Dateien komprimieren bzw. entpacken (GNU-zip-Verfahren); komprimierte Dateien haben die Endung .gz
zcat	gibt komprimierte Dateien aus

tar - Daten archivieren

tar (tape archiver) ist ein sehr altes Unix-Programm. Es stammt aus einer Zeit, in der alle Daten sequentiell auf Magnetbändern gespeichert wurden. Um Zugriffszeiten auf die Banddaten zu optimieren, wurde tar entwickelt, um die zu speichernden Daten in einer einzigen zusammenhängenden Datei zusammenzufassen und nicht in vielen Fragmenten auf das Band schreiben zu müssen.

tar alleine komprimiert die Daten nicht! Dies geschieht erst in Zusammenhang mit einem Kompressionsprogramm, z.B. gzip, das mit der Option "z" in den Archivierungsvorgang eingebunden werden kann.

Syntax

tar [Optionen] Verzeichnisse bzw. Dateien

Nützliche Optionen

(ohne führendes -)

c	create, Archiv erzeugen
t	table of contents, Inhalt des Archivs anzeigen
x	extract, Datei(en) aus Archiv auspacken
v	verbose, mehr Informationen anzeigen
f Archivdatei	Namen des Archivs angeben
z	die Daten, die tar erhält, zunächst durch gzip / gunzip filtern, zum Entpacken von komprimierten tar-Dateien (.tar.gz oder .tgz)

Beispiele

Angenommen, im aktuellen Verzeichnis befinden sich folgende Dateien:

```
drwxr-xr-x   2 chris    users        4096 Jun 21 22:57 .
drwxr-xr-x  21 chris    users        4096 Jun 21 22:52 ..
-rwxr-----   1 chris    users      823664 Jun 21 22:55 fax-seite1.jpg
-rwxr-----   1 chris    users      832136 Jun 21 22:55 fax-seite2.jpg
-rw-r--r--   1 chris    users     1732485 Jun 21 22:55 linuxbuch.pdf
-rwxr-----   1 chris    users     1796301 Jun 21 22:55 linuxbuch.ps
-rw-r--r--   1 chris    users        4055 Jun 21 22:52 mailcap
-rw-r--r--   1 chris    users           0 Jun 21 22:55 webcam.html
```

tar wird mit den Optionen c (neues Archiv anlegen), v (anzeigen, was geschieht) und f (Name des Archivs, hier archiv.tar) gestartet:

```
% tar cvf archiv.tar *
fax-seite1.jpg
fax-seite2.jpg
linuxbuch.pdf
linuxbuch.ps
mailcap
webcam.html
%
```

Die erzeugte Datei archiv.tar ist nicht komprimiert, wie man anhand ihrer Größe erkennt - sie entspricht ungefähr der Summe der Größe der einzelnen Dateien:

```
-rw-r--r--   1 chris    users     5539840 Jun 21 22:57 archiv.tar
```

Um Daten direkt mittels gzip zu komprimieren, nutzt man die Option "z" und wählt als Namenssuffix des Archivfiles .tar.gz:
% tar czf archiv.tar.gz *

Die Datei ist nun um einiges kleiner:
-rw-r--r-- 1 chris   users   3307434 Jun 21 23:08 archiv.tar.gz

Komprimierte Dateien lassen sich mit den Optionen "x" und bei komprimierten .tar.gz-Dateien "z" wieder extrahieren:

```
% tar xvzf archiv.tar.gz          (komprimierte .tar.gz-Datei auspacken)
bzw.
% tar xvf archiv.tar              (unkomprimierte .tar-Datei auspacken)
```

gzip / gunzip - Dateien komprimieren

gzip bzw. gunzip (GNU zip / unzip) komprimieren oder dekomprimieren Dateien. Im o.g. Beispiel zu tar ist gzip bereits verwendet worden.
Der verwendete Kompressionsalgorithmus ist bei Text-, verschiedenen Grafik- und anderen Formaten sehr effizient und kann Daten um durchschnittlich 50% verkleinert archivieren.

gzip erzeugt standardmäßig komprimierte Dateien mit dem Namensformat Dateiname.gz und löscht die unkomprimierte Originaldatei.

Syntax

gzip bzw. **gunzip** [Optionen] Dateien

Nützliche Optionen

-l	Inhalt der komprimierten Datei anzeigen
-d	decompress, auspacken
-r	rekursiv, ganze Verzeichnisäste einpacken
-t	komprimierte Dateien testen
-S Suffix	<Suffix> statt .gz verwenden

Beispiele

% gzip -r ./Daten/*
komprimiert alle im Verzeichnis Daten und allen Unterverzeichnissen enthaltenen Dateien in einzelne Dateien.

% gunzip datei.gz
entpackt datei.gz und löscht die gepackte Datei.

zcat - komprimierte Dateien ausgeben

zcat ist ein kleines Tool, das komprimierte Dateien dekomprimiert nach stdout ausgibt.
Es wird nicht das Inhaltsverzeichnis des Archivfiles, sondern der wirkliche Inhalt der enthaltenen Datei angezeigt.

Syntax

zcat Datei.gz bzw. Datei.Z

Nützliche Optionen

-l	Inhalt der komprimierten Datei anzeigen
-d	decompress, auspacken
-t	komprimierte Dateien testen

Beispiele

```
% zcat datei.txt.gz
Hallo, Welt!
```

"Hallo, Welt!" ist der Inhalt der im .gz-File enthaltenen Originaldatei.

Zugriff auf das Netzwerk

Linux ist hervorragend geeignet zur Arbeit in heterogenen Netzwerken, d.h. in Netzen mit verschiedenen Rechnerarchitekturen, Betriebssystemen und Netzwerkprotokollen. Außer mit seinen Fähigkeiten als Netzwerkserver jeglicher Art unterstützt Linux den Anwender jedoch auch mit allen Client-Programmen, die man zum Zugriff auf Daten im Netzwerk benötigt.

Wir beschränken uns in diesem Kurs auf die absolut wichtigsten Netzwerkprotokolle. Sie nutzen das TCP/IP-System zur Datenübertagung und sind daher sowohl für die Arbeit im lokalen Intranet als auch im Internet geeignet.

Das sind im einzelnen:

Kommando	**Wofür?**
telnet	auf anderen Rechnern einloggen
ftp	File Transfer Protocol, Dateien übertragen
rsh	Remote Shell, Kommandos auf anderen Rechnern ausführen
lynx	Textmode-Webbrowser

telnet - auf anderem Rechner einloggen

Das Programm telnet ermöglicht, sich auf anderen Rechnern (meist Unix) einzuloggen und dort zu arbeiten, als ob man direkt an der Konsole des Rechners säße.
Es nutzt das unverschlüsselte Telnet-Protokoll und sollte daher nicht ohne vorherige Gedanken über mögliche Sicherheitsprobleme verwendet werden. So werden z.B. Paßwörter im Klartext über das TCP/IP-Protokoll gesendet, und ein fähiger Angreifer könnte diese Informationen stehlen.

Syntax

telnet Rechnername / IP-Adresse

Beispiele

Um sich auf dem entfernten Rechner lx-server anzumelden und dort in der Shell zu arbeiten:

```
% telnet lx-server
Trying 192.168.102.6...
Connected to lx-server.
Escape character is '^]'.
Welcome to SuSE Linux 6.4 (i386) - Kernel 2.2.14 (2).

lx-server login:
```

Nach Eingabe von Usernamen und Paßwort kann man mit der Arbeit beginnen.

ftp - Dateien übertragen

ftp (File Transfer Protocol) dient zur schnellen Übertragung von Daten aller Art über TCP/IP-Netzwerke.

Syntax

ftp Rechnername / IP-Adresse

Beispiele

```
% ftp lx-server
Trying 192.168.102.1...
Connected to lx-server.domain.de.
220 lx-server.domain.de FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.16) ready.
Name (lx-server:chris):
```

Nun noch Usernamen und Paßwort eingeben und man ist im ftp-Server eingeloggt.
ftp bietet eine Reihe eigener Kommandos zur Übertragung von Dateien etc.

Hier ein Auszug:

bye	ausloggen
get	Datei von entferntem Rechner auf lokalen PC übertragen
mget	mehrere Dateien "
put	Datei von lokalem PC auf entfernten Rechner übertragen
mput	mehrere Dateien "
ascii	Daten im ASCII-Modus übertragen (für alle Textdateien)
bin	Dateien im Binär-Modus übertragen (für Binärdateien)
[...]	

Weitere Informationen bietet die man page zu ftp.

lynx - Textmode-Webbrowser

Ein besonders erwähnenswertes Programm ist der Textmode-Webbrowser lynx. Er ermöglicht das Websurfen in der Textkonsole.
lynx kann zwar keine Grafiken, Java-Applets und ähnliche Elemente modernen Webdesigns anzeigen, kann aber bei Internetrecherche oder ähnlichen eher textlastigen Arbeiten eine sehr schnelle Alternative zu großen Browsern wie z.B. Netscape sein.
Darüber hinaus ist lynx ein sehr gutes Tool zum Anzeigen von Online-Dokumentation zu Programmpaketen, die oft im HTML-Format vorliegt.

Syntax

lynx [Optionen] URL / IP-Adresse / Rechnername / Datei

Tastaturkommandos

Cursortasten:	
oben / unten	in der Webseite scrollen
links / rechts	Seite zurück- bzw. vorblättern
q	lynx verlassen
h	Online-Hilfe aufrufen

Beispiele

```
% lynx www.altavista.de
```

ruft die Website www.altavista.de auf.

Editoren

Um Dateien bearbeiten zu können, braucht man einen Editor.
Es gibt eine riesige Menge an verschiedenen Text- und Binär-/Hex-Editoren, mit denen man jede Art von Dateien verändern kann. Die Vielfalt an Editoren hat sich jedoch auch in einer Unzahl unterschiedlicher Bedienkonzepte niedergeschlagen. So folgen einige Editoren den Emacs-Kommandos (der Editor emacs war einer der ersten leistungsfähigen Textverarbeitungssysteme), einige orientieren sich eher am alten DOS-Editor Word Star und viele bringen ganz einfach eigene Möglichkeiten und Befehle mit.
Welche Editoren auf Ihrem System installiert sind, kann Ihr Administrator Ihnen sagen.

vi - der Standard-Editor

Ein Standard-Editor, das Programm vi, ist jedoch auf allen UNIX-Rechnern vorzufinden.
Es ist ebenfalls ein etwas betagtes Programm, ist jedoch, wenn man die Tastaturkommandos verstanden hat, sehr leistungsfähig. Bitte beachten Sie die Befehlsreferenz, die unten beigefügt ist.

Syntax

vi Datei

Tastaturkommandos

Hier einige grundlegende Kommandos des vi:

vi <DATEI>	öffnet eine Datei zum Editieren.
i	Insert-Modus
R	Replace-Modus
A	Anfüge-Modus am Zeilenende
a	Anfüge-Modus am Punkt
ESC-Taste	Eingabemodi abbrechen
k	Zeichen vor
j	Zeichen zurück
h	Zeile aufwärts
l	Zeile abwärts
J	folgende Zeile an diese anhängen
dd	Zeile löschen
5dd	5 Zeilen löschen
D	Zeile bis zum Ende löschen
x	Zeichen löschen
Y	Zeile in Puffer kopieren
7Y	7 Zeilen in Puffer kopieren
P	Zeile(n) vor Cursor einfügen
p	Zeile(n) nach Cursor einfügen
:w	Datei speichern
:wq	Datei speichern, vi beenden
:q	vi beenden
:q!	vi beenden, ohne zu speichern (wenn mal was schiefgelaufen ist)
$	Ans Zeilenende
^	An den Zeilenanfang

[...] s. man page oder Kurzreferenz

emacs

Ein sehr verbreiteter Editor ist der emacs.
Er ist mittlerweile unter dem Namen xemacs auch in einer X-Window-Version mit grafischer Benutzeroberfläche erhältlich, bietet jedoch auch im Textmodus schon gewissen Komfort.
Er ermöglicht Dinge wie gleichzeitiges Editieren mehrerer Dateien, Nutzung als Entwicklungsumgebung für Programmiersprachen oder als Webbrowser o.ä.
In emacs ist eine eigene Programmiersprache enthalten, die Erweiterungen und Makros möglich macht.

emacs wird zu großen Teilen über Tastaturkommandos bedient, wenn auch einige neuere emacs-Versionen Mausunterstützung bieten.

Die gängigsten Kommandos sind in der Kurzanleitung weiter unten aufgeführt.

joe

Der Editor joe benutzt Tastaturkommandos, die stark an WordStar erinnern. Er ist sehr leistungsfähig und flexibel. Mit CTRL-K-H ruft man die ausführliche Hilfe auf, die alle Tastaturkommandos erklärt.

Grafische Benutzerumgebungen

Für Linux und sein Grafiksystem XFree86 gibt es eine Unmenge von verschiedenen Windowmanagern (s. Grundlegende Systemstrukturen - X-Window-System), die mal verspielter und bunt, mal professionell oder traditionell ausfallen und unterschiedliche Funktionen mitbringen.

Im Allgemeinen bieten Sie alle mehr oder weniger leistungsfähige Funktionen (teilweise auch vom X-Server bereitgestellt, z.B. 3D-Support), darunter:

- Darstellung von grafischen Programmen, mit allen Standard-Bedienelementen wie Fenstern, Buttons, Bitmaps, Menüs, Kontextmenüs
- Möglichkeiten zum Organisieren des Desktops mit Symbolen
- Prozeßverwaltung, Starten und Stoppen von Programmen
- Drag-and-Drop, also das Hin- und Herschieben von Objekten mit der Maus, teilweise auch mit damit verknüpften Aktionen (z.B. Drucken, indem man eine Datei auf das Druckersymbol zieht)
- Unterstützung für 3D-Beschleunigerkarten
- Einstellmöglichkeiten für Desktopfarben und -aussehen, Bildschirmschoner, etc.
- Multimedia-Applikationen, z.B. für Audio-CDs

Im folgenden stellen wir kurz die beiden "Standard-X-Window-Oberflächen" vor, die bei allen modernen Linuxsystemen enthalten sein sollten, wobei eher Wert auf einen schnellen Überblick über die Optik und die gebotenen Features gelegt wird als auf tiefgehende Einführung in die Bedienung.

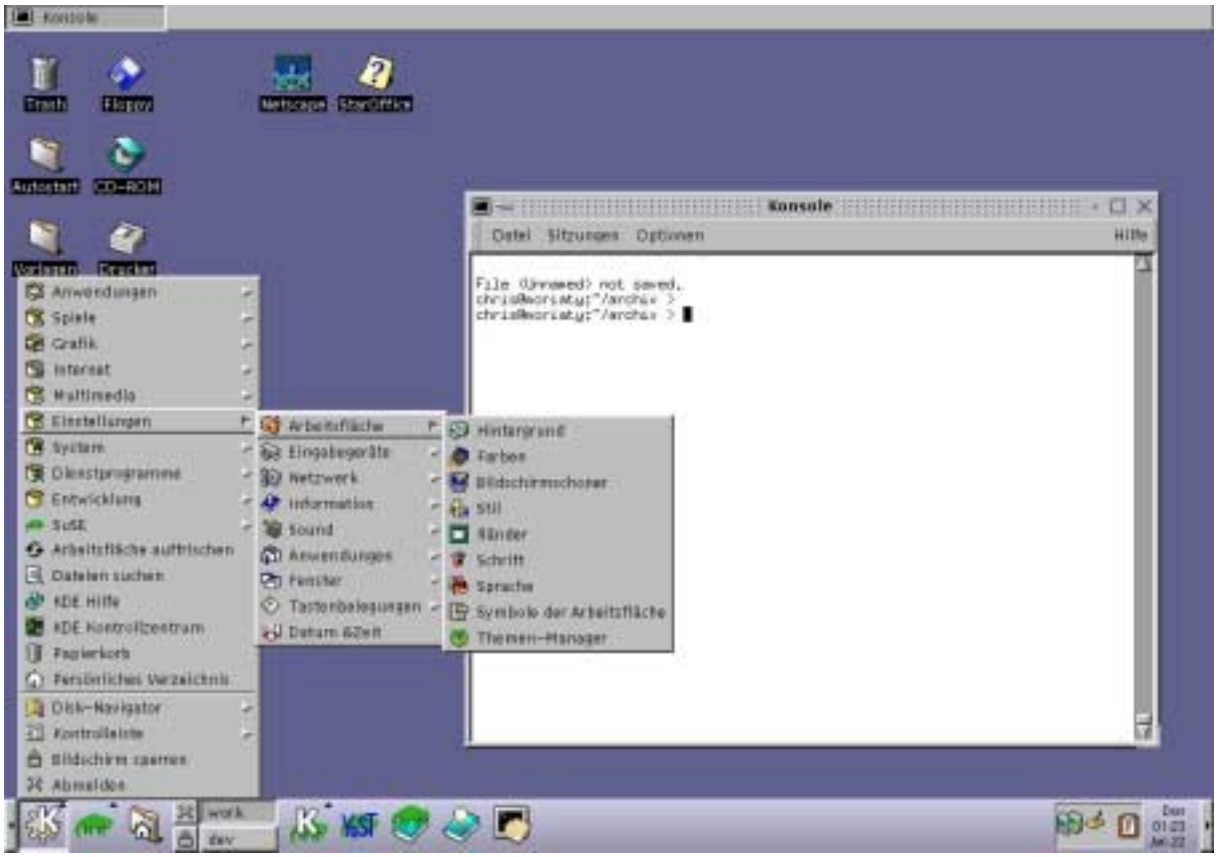
KDE

Das K Desktop Environment (KDE) ist eine Entwicklung der Netzgemeinde, die sich zum Ziel gesetzt hat, eine freie objektorientierte grafische Benutzeroberfläche inkl. Klassenbibliotheken für eigene Programme bereitzustellen. Auch die norwegische Firma TrollTech Inc. hat sich an der Entwicklung sowohl finanziell, als auch mit Entwicklerteams beteiligt.

KDE ist sehr ressourcen-hungrig (viel Speicher und schnelle Grafikkarte nötig), allerdings auch äußerst komfortabel. Fast alle Einstellungen am Linuxsystem lassen sich im KDE mit der Maus in benutzerfreundlichen Dialogboxen regeln. Darüberhinaus ist ein leistungsfähiger Dateimanager - kfm - integriert, der Dateioperationen wie Kopieren, Verschieben, Umbenennen oder Verzeichnisverwaltung in der Grafikoberfläche ermöglicht.

Bei modernen Distributionen ist die Integration der im System installierten Software sehr gut. Fast alle Programme, die unter X11 laufen, sind in die Startmenüs integriert.

Ein typischer KDE-Desktop (hier einer S.u.S.E. 6.4) sieht so aus:



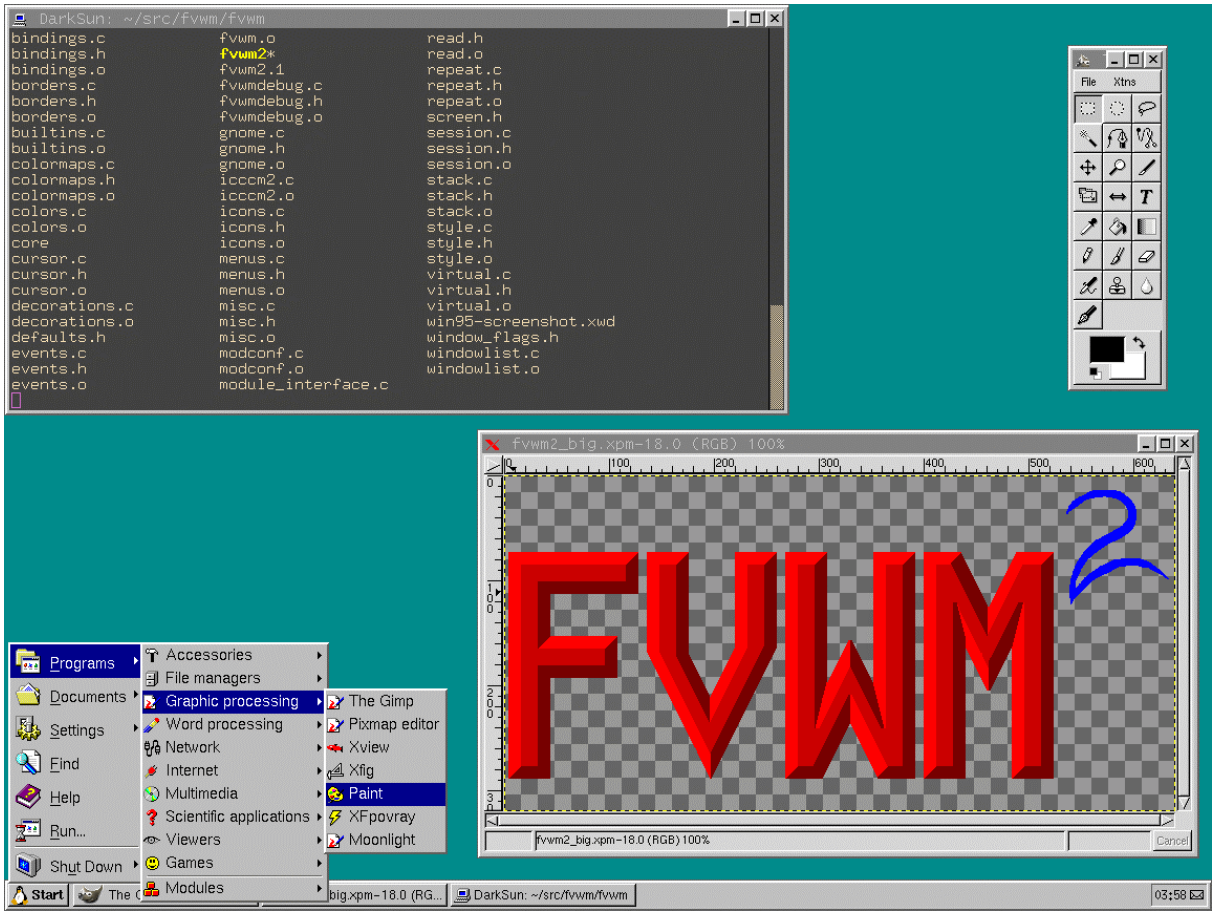
Für die KDE-Umgebung gibt es sehr viele, zum größten Teil kostenlose Softwarepakete, die man im Internet z.B. unter http://www.kde.org finden und herunterladen kann.

Viele Linux-Distributionen (RedHat, S.u.S.E., Debian) enthalten jedoch auch schon eine gute Auswahl praktischer Programme für das KDE. Konsultieren Sie für eine Übersicht über den Inhalt der Distributionen bitte die Homepages der Anbieter.

fvwm / fvwm2

Der Windowmanager fvwm, aktuell ist die Version 2, ist eine der ersten für Linux entwickelten grafischen Benutzeroberflächen. Er ist extrem stabil, begnügt sich auch mit einem mittelklassigen Rechner und läßt sich sehr weitgehend anpassen. Über vordefinierte Konfigurationen, die von den meisten Distributoren mitgeliefert werden, lassen sich Optik, Verhalten und Bedienung einstellen.

So ist z.B. auch folgende Konfiguration leicht möglich, die den Linux-Desktop ähnlich wie Windows95 erscheinen läßt:



Allerdings fehlen dem fvwm einige Fähigkeiten fortgeschrittener Windowmanager, wie z.B. Drag-and-Drop und Objekte auf dem Desktop.

Auf der FVWM-Homepage unter http://www.fvwm.org findet man Wissenswertes über die Installation, Einrichtung und Benutzung dieses Windowmanagers.

Tools und Programme im Grafikmodus

Im folgenden Abschnitt stellen wir kurz mögliche leistungsfähige Programme für bestimmte Einsatzgebiete vor. Diese Liste erhebt nicht im Geringsten den Anspruch auf Vollständigkeit. Jeden Tag werden tausende neue Versionen von Softwarepaketen für Linux im Internet veröffentlicht.

Die hier genannten Anwendungen sollten ohne größere Probleme mit allen gängigen Windowmanagern harmonieren.

Ein Blick auf die in den Literaturtips genannten Websites bringt einen guten Überblick über weitere verfügbare Software.

Dateiverwaltung

Tools zum Management von Daten und Verzeichnissen gibt es unter Linux wie Sand am Meer.
Besonders vielversprechend sind natürlich in den Windowmanager integrierte Lösungen wie **KFM**, der KDE File Manager, mit denen sich Dateien völlig transparent mit Drag-and-Drop-Mausaktionen verwalten lassen.
Aber auch Programme wie **tkDesk**, ein in der Programmiersprache tcl/tk implementierter Dateimanager, sind durchaus benutzbar.

Editoren

Ebenso vielfältig sieht es bei den Editoren unter X11 aus.
Auf den meisten Systemen dürfte der alte **xedit** installiert sein. Er ist zwar in seinem Funktionsumfang eingeschränkt, aber schnell und für eine kleine Änderung auf einem System, auf dem keine anderen Editoren verfügbar sind, eine gute Wahl.

KDE bringt die Programme **kedit** und **kwrite** mit, die im großen und ganzen den Windows-Programmen notepad und write entsprechen.

xemacs, das emacs für X11, ist sehr leistungsfähig, allerdings auch nicht sofort verständlich, da es vor Optionen, Einstellungen, Menüoptionen usw. strotzt. Für Softwareentwickler ist der xemacs die erste Wahl, da er sich weitgehend in eine Entwicklungsumgebung einfügen kann und Vorgänge automatisieren kann.

xjed ist eine gute Alternative, da xjed über Konfigurationsdateien frei angepaßt werden kann.

Grafikprogramme

Im Grafikbereich haben sich folgende Tools dauerhaft einen Namen gemacht:
The Gimp ist ein im Funktionsumfang mit Adobe's Photoshop vergleichbares Grafik- und Photobearbeitungsprogramm. Es ist unter http://www.gimp.org erhältlich, wird aber auch den meisten Distributionen beigepackt. Es ist der de-facto-Standard für Grafiksoftware im Unix-Bereich und existiert auch für andere Betriebssysteme, u.a. Windows.

xfig ist ein Illustrations-Tool, das sich gut für Skizzen, Diagramme, Zeichnungen usw. eignet. Es basiert zwar auf einer recht alten Benutzeroberfläche, ist aber dennoch sehr leistungsfähig.

Textverarbeitung / Office

Komplette Officepakete, die denen unter anderen Systemen in nichts mehr nachstehen, gibt es schon seit einigen Jahren. Insbesondere die Hamburger Firma StarDivision, die mittlerweile von Sun Microsystems gekauft wurde, hat mit ihrem Produkt **StarOffice**, aktuell ist die Version 5.1, den Officemarkt unter Linux vorangetrieben.
Es ist genauso weit entwickelt wie die Windows-Version, kann MS-Office-Dokumente lesen und Schreiben und bietet alle Funktionen einer modernen Office-Suite.

Alternativen dazu existieren mit **ApplixWare** (Applix), **WordPerfect** (Corel) und **KOffice**, einem freien Officepaket, das von der KDE-Entwicklergemeinde im Internet weitergepflegt und -entwickelt wird.

Einen anderen Ansatz, eher in Richtung professioneller Textsatzsysteme, geht man mit dem **LaTeX**-System und grafischen Editoren, wie z.B. **LyX** oder **KLyX**. TeX ist eine Dokumentenstruktur-Beschreibungssprache, die Sprachen wie HTML ähnelt. Mit ihr lasen sich, unabhängig vom Ausgabegerät (z.B. Drucker, Monitor), immer gleiche und typografisch korrekte Dokumente erstellen.

Netzwerktools

Für die Arbeit im Netzwerk findet man in den Distributionen und im Internet eine erschlagende Fülle von grafischen Applikationen für jeden nur erdenklichen Anwendungsfall.

Für die Hauptanwendungen in TCP/IP-Netzen, also WWW, FTP, Mail und News, benötigt man im Prinzip nur die Linuxversion des **Netscape Communicators**, die bei allen Distributionen beiliegt. Sie ist weitgehend identisch zur Windows-Version.

Aber auch Programme wie **Caitoo**, ein Download-Manager für KDE, der Dateien zeitgesteuert und mit Wiederaufnahme abgebrochener Downloads herunterladen kann, oder **xftp**, ein simples, aber schnelles X11-ftp-Programm sind bei der täglichen Arbeit sehr praktisch.

Linux - technische und qualitative Merkmale

Hardwareunterstützung

Linux läuft auf 386, 486, Pentium, PII und allen anderen x86-kompatiblen Rechnern mit ISA-, EISA-, PCI- oder VLB-Bus. Die Unterstützung für MCA (IBM Microchannel) ist experimentell.
Desweiteren sind Portierungen in unterschiedlichen Entwicklungsstufen für Motorola 680x0 (Amiga, Atari etc.), DEC Alpha, Sun SPARC, PowerPC, MIPS, ARM erhältlich.
Man kann davon ausgehen, daß darüber hinaus nahezu sämtliche gebräuchliche Zusatzhardware unterstützt wird. Detaillierte Listen der unterstützten Hardware und der Hardwareanforderungen erhält man in der Regel auf den Homepages der Linux-Distributoren.

Stabilität

Linux gilt längst nicht mehr als Programm in der Betaphase, da die Version 1.0 bereits am 14. März 1994 herausgebracht wurde. Es gibt immer noch Fehler im System und im Laufe der Zeit werden neue auftauchen und beseitigt werden. Da Linux dem offenen Entwicklungsmodell folgt, werden alle neuen Versionen der Öffentlichkeit zur Verfügung gestellt, ob sie nun Serienreife haben oder nicht.
Um den Anwendern trotzdem die Unterscheidung zwischen einer stabilen und einer Beta-Version zu ermöglichen, bedient man sich des folgenden Schemas:

Die Versionen 2.x.y sind bei geradem x stabile Versionen; bei Beseitigung von Fehlern wird y erhöht. Beim Wechsel von Version 2.2.2 auf 2.2.3 wurden also nur Fehler bereinigt, es gab keine neuen Eigenschaften.

Die Versionen 2.x.y mit ungeradem y, also z.B. 2.3.124, sind Vorabversionen, die hauptsächlich für Entwickler geeignet sind. Sie können instabil sein oder abstürzen, und es werden ständig neue Eigenschaften hinzugefügt.

Von Zeit zu Zeit, wenn der aktuelle Entwicklerkernel stabil ist, wird er als neuer stabiler Kernel eingefroren und die Entwicklung wird mit einer neuen Entwicklungsversion des Kernels fortgesetzt.
Den aktuellen Entwicklungsstand erfährt man aus erster Hand unter http://www.kernel.org.

Verfügbare Anwendungssoftware

Für Linux steht ein immenser Softwarepool zur Verfügung.
Standardprogramme für die tägliche Arbeit (Office, WWW, E-Mail, Grafikbearbeitung etc.) liegen ebenso vor wie Spezialsoftware (DTP, CAD, Sound- u. Videobearbeitung, Netzwerkprogramme etc.).
Der Großteil der unter Linux erhältlichen Software unterliegt ebenfalls diversen Freeware-Lizenzen und wird in unentgeltlicher Freizeitarbeit von engagierten Hobbyprogrammierern entwickelt.
Mehr und mehr Firmen erkennen mittlerweile die Marktchance Linux und entwickeln auch kommerzielle Software, die vergleichbaren Produkten für kommerzielle Betriebssysteme in nichts nachsteht.
Einen guten Überblick über den Stand der Entwicklung im Bereich der Anwendungen erhält man bei sog. Linux-Portal-Sites, wie z.B. http://www.linux.org, .de, .com usw., die aktuelle Berichte über Linux veröffentlichen.

Anwendungsgebiete

Linux-Systeme werden heutzutage in allen Bereichen der Computeranwendungen erfolgreich eingesetzt. Vom reinen Arbeitsplatz-PC über die Grafikworkstation, das Entwicklungssystem, den Internet-, Mail-, Fax-, Telefon- oder ftp-Server, das Internet-Gateway, Echzeitanwendungen zu Automatisierungszwecken bis hin zum massiven Parallelrechnen – Linux hat so gut wie jede Domäne erobert.
Aufgrund seiner ausgezeichneten Netzwerkfähigkeiten liegt der Schwerpunkt des Einsatzes von Linux natürlich bei Internet-, Intranet- und anderen Serverdiensten.

Technische Eigenschaften von Linux

- Multitasking: mehrere Programme laufen zur selben Zeit.
- Multiuser: mehrere Benutzer arbeiten gleichzeitig auf demselben.
- Multiplattform: läuft auf vielen verschiedenen CPUs, nicht nur Intel.
- Multiprozessor: SMP-Unterstützung steht auf Intel- und SPARC-Plattformen zur Verfügung (an anderen Plattformen wird gearbeitet). Linux wird für verschiedene offene Multiprocessing-Anwendungen benutzt, Beowulf-Systeme (Netzwerk-Supercomputing) und der SPARC-basierte Fujitsu AP1000+ Supercomputer eingeschlossen.

- Speicherschutz zwischen Prozessen, so daß ein Programm nicht das ganze System zum Absturz bringen kann
- Ausführbarer Code wird nach Bedarf geladen: Linux liest nur diejenigen Teile eines Programms von Platte, die tatsächlich benutzt werden.
- Virtueller Speicher benutzt Paging auf Festplatte: Es werden nicht komplette Prozesse ausgelagert, sondern nur soviele Speicherseiten, wie ein anderer Prozeß gerade anfordert. Dies geschieht auf eine separate Partition oder in eine Datei im Dateisystem oder beides, mit der Möglichkeit, während des Betriebs Auslagerungsbereiche hinzuzufügen (ja, man spricht immer noch von Auslagerungsbereichen). Insgesamt können 16 dieser 128 MB großen Bereiche auf einmal benutzt werden. Das ergibt theoretisch einen Gesamtwert von 2 GB an verwendbarem Auslagerungsbereich. Bei Bedarf kann dies leicht durch Ändern einiger Zeilen im Quellcode erweitert werden.
- Ein gemeinsamer Speicherpool für Anwender-Programme und Plattenpuffer, so daß der gesamte freie Speicher für den Cache verwendet werden kann, der Cache aber reduziert werden kann, wenn große Programme laufen.
- Dynamisch gelinkte Shared Libraries (DLL's), und natürlich auch statisch gelinkte Libraries.
- Führt „core dumps“ (Speicherabzüge) für Post-Mortem-Analysen aus. Dadurch wird die Anwendung eines Debuggers auf ein Programm nicht nur während dieses läuft, sondern auch nachdem es abgestürzt ist, möglich.
- Weitgehend kompatibel mit POSIX, System V und BSD auf der Ebene des Quellcodes.
- Der gesamte Quellcode ist verfügbar, den kompletten Kernel, alle Treiber, die Entwicklungswerkzeuge und alle Nutzerprogramme eingeschlossen. Alles ist auch frei verteilbar. Es gibt eine Vielzahl an kommerziellen Programmen für Linux, die ohne Sourcecode ausgeliefert werden, aber alles, was einmal frei war, das gesamte Betriebssystem eingeschlossen, ist immer noch frei erhältlich.
- Unterstützung für viele nationale oder speziell angepaßte Tastaturen.
- Mehrere virtuelle Konsolen: mehrere unabhängige Zugänge zugleich über die Konsole. Man kann dabei durch eine spezielle Tastenkombination hin- und herschalten (unabhängig von der Grafikhardware). Die virtuellen Konsolen werden dynamisch angelegt. Bis zu 64 Stück kann man benutzen.
- Unterstützt verschiedene gängige Dateisysteme, wie z.B. Minix, Xenix und alle gängigen System-V-Dateisysteme; hat ein ausgefeiltes eigenes Dateisystemschema, das Dateisysteme bis zu 4 TB mit Dateinamen von bis zu 255 Zeichen Länge bietet.
- Transparenter Zugriff auf MS-DOS Partitionen (oder OS/2 FAT-Partitionen) über ein spezielles Dateisystem: man braucht keine speziellen Befehle zur Benutzung der MS-DOS-Partition, sie präsentiert sich einfach wie ein normales UNIX-Dateisystem (abgesehen von Beschränkungen für Dateinamen, Berechtigungen, etc). Mit MS-DOS 6 komprimierte Partitionen funktionieren derzeit nicht ohne Patch (dmsdosfs). VFAT (Windows 95/98) wird ab Linux 2.0 unterstützt.
- Ein spezielles Dateisystem namens UMSDOS, das es ermöglicht, Linux in einem DOS-Dateisystem zu installieren.
- Lesezugriff für HPFS-2 von OS/2 2.1 wird unterstützt.
- HFS-Dateisystem-Unterstützung (Macintosh) ist separat als Modul erhältlich.
- CD-ROM-Dateisystem, das alle gängigen CD-ROM-Formate liest.
- TCP/IP Netzwerkunterstützung, einschließlich ftp, telnet, NFS etc.
- Appletalk Server
- Netware Client und Server
- Lan Manager (SMB) Client und Server
- Viele Netzwerkprotokolle: die neuen Kernel beinhalten TCP, IPv4, IPv6, AX.25, X.25, IPX, DDP (Appletalk), NetBEUI, u.a.

Literaturtips

Bücher

- **Linux / Unix Grundlagen - Kommandos und Konzepte**
  Helmut Herold
  Addison-Wesley-Verlag
  ISBN 3-8273-1435-6

- **Linux in a Nutshell (dt. Ausgabe)**
  Ellen Siever
  O'Reilly-Verlag
  ISBN 3-8972-1116-5

- **Linux, Unix- Profitools. awk, sed, lex, yacc und make**
  Helmut Herold
  Addison-Wesley-Verlag
  ISBN 3-8273-1448-8

Online-Dokumentation

- **http://freshmeat.net**
  Portalsite für Ankündigung neuer Linux-Software

- **www.linux.de, www.linux.com usw.**
  gute Startpunkte für Recherchen

- **http://www.kernel.org**
  offizielle Seiten zum Kernel und zum System

Evtl. im System installierte Dokumentation

- **im Verzeichnisast unterhalb von /usr/doc/**
  Dokumentation zu installierter Software, zum System, zur Administration, etc.

- **Hilfesysteme der jeweiligen Distribution**
  Kommando "hilfe" (S.u.S.E.), "help" o.ä.
  Bitte konsultieren Sie hierzu Ihren Administrator.

- **Hilfesysteme der Windowmanager**
  KDE-Hilfe, GNOME-Hilfe, etc.

Copyright und Disclaimer

Copyright

Kommentare, Ergänzungen, Korrekturen oder sonstiges Feedback sind immer willkommen :-)
Email: cmoeller@iam.de

Disclaimer

Die Informationen in diesem Skript wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Weder der Autor noch die Mikroelektronik-Akademie GmbH übernehmen irgendeine juristische Verantwortung oder Haftung für eventuelle fehlerhafte Angaben und deren Auswirkungen oder Folgen.
Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen. Der Autor richtet sich im wesentlichen nach der Schreibweise des Herstellers.