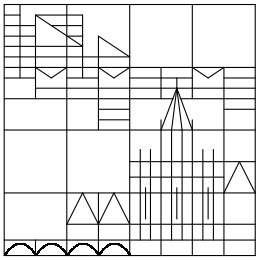

Universität Konstanz

Kurzübersicht über UNIX/X11/Internetzugang am Beispiel des Bereichs Informatik der Universität Konstanz

Karsten Weihe

Konstanzer Schriften in Mathematik und Informatik
Nr. 6, Februar 1996

ISSN 1430–3558

Kurzübersicht über UNIX[1]/X11[2]/Internetzugang am Beispiel des Bereichs Informatik der Universität Konstanz

Karsten Weihe[3]

Lehrstuhl für praktische Informatik I
(Algorithmen und Datenstrukturen)

6/1996
Universität Konstanz
Fakultät für Mathematik und Informatik

1 Allgemeines

Diese Kurzübersicht kann und soll keinen umfassenden Überblick geben, sondern nur einen (mehr oder weniger) systematischen *Ein*blick. Sie ist nicht für das selbständige Erlernen der Materie gedacht, sondern für die Nachbereitung eines entsprechenden Kurses und als Kurzreferenz. Sie ist entstanden aus einem UNIX/C–Praktikum, das der Autor im Wintersemester 1995/96 an der Universität Konstanz durchgeführt hat.

Diese Kurzübersicht bezieht sich auf Rechner der Marke *Sun* mit dem Sun–spezifischen „UNIX–Dialekt" *Solaris*. Die allermeisten Details sind allgemeingültig auch für andere Marken und Dialekte. Aus Gründen der Übersichtlichkeit gehen wir bei den einzelnen Details nicht darauf ein, ob sie allgemeingültig sind oder nicht. Nur bei Details, die wirklich für den Bereich Informatik der Universität Konstanz spezifisch sind, machen wir auf diesen Umstand jeweils durch einen kleinen Kommentar aufmerksam.

Danksagung

Ich danke Torsten Grust, Dietmar Kühl und Annegret Liebers für's Korrekturlesen und für viele Anregungen und Vorschläge.

[1] UNIX ist ein eingetragenes Warenzeichen von AT&T.

[2] X11 ist ein eingetragenes Warenzeichen des X Consortium, Inc.

[3] Universität Konstanz, Fakultät für Mathematik und Informatik, Postfach 5560/D 188, 78434 Konstanz, Germany, email: karsten.weihe@uni–konstanz.de, WWW http://www.informatik.uni–konstanz.de/~weihe

1.1 Von der Systemsoftware verwaltete Objekte

User (Benutzerin)[4]: Ist dem Computer bekannt durch eine *Benutzerinnenkennung*. Wichtige Attribute von Benutzerinnen sind Passwort und Gruppenzugehörigkeiten. Ist man unter der Benutzerinnenkennung *root* eingeloggt, dann hat man uneingeschränkte Zugriffsrechte; das *root*–Passwort ist daher nur den Administratorinnen bekannt.

Group (Gruppe): Besteht aus Benutzerinnen. Jede Benutzerin gehört zu einer Hauptgruppe und mglw. zu einer oder mehreren Nebengruppen. Die Hauptgruppe für Studentinnen im Bereich Informatik der Universität Konstanz heißt „student", die der Mitarbeiterinnen in diesem Bereich ist „inf".

File (Datei): Ein File ist aus Sicht „normaler" Benutzerinnen einfach eine lineare Sequenz von Zeichen. Alle Daten und Informationen werden in Files abgespeichert.

Directory (Verzeichnis, Ordner): Eine Sammlung von Files. Jedes File gehört zu einer Directory. Eine Directory kann neben Files wieder Directories enthalten, so daß sich eine Hierarchie von Directories ergibt.

Device (externes Gerät): Dazu gehören bspw. Bildschirm, Tastatur, Maus, Festplatte, Diskettenlaufwerk, Drucker und Modem. Zugriff auf alle diese Geräte wird durch die Systemsoftware geregelt. In einem Rechnernetz gehören dazu auch die Schnittstellen zu anderen Computern.

Program (Programm): Eine Folge von Maschinenbefehlen, die in einem File abgespeichert sind. Der Name des Files ist das Kommando, das man aufrufen

[4] Aus Gründen der sprachlichen Vereinfachung wird im folgenden immer nur die weibliche Form für natürliche Personen benutzt.

muß, damit der Computer das Programm ausführt.

Process (Prozeß): Die Abarbeitung eines Programms nennt man einen Prozeß.

1.2 Aufbau und Arbeitsteilung der Systemsoftware

UNIX Kernel: Unterste Ebene; zuständig für die Verwaltung von File- und Prozeßsystem und dafür, daß die Interaktion mit externen Geräten überhaupt läuft. Der Kernel erhält die Illusion aufrecht, daß

- Files lineare Sequenzen von Zeichen sind, die im Arbeitsspeicher jeweils als ein Stück abgelegt sind.
- Prozesse parallel ablaufen.

In Wirklichkeit werden Files auf der Festplatte gespeichert, und zwar in Blöcken gleicher Größe (*Page*), die wild verstreut auf der Festplatte liegen können und nur einzeln bei Bedarf in den Arbeitsspeicher geladen werden. Prozesse werden nicht wirklich parallel ausgeführt, sondern „reihum“ erhält jeder aktuelle Prozeß ein sehr kurzes Zeitfenster und wird danach wieder in die Warteschlange eingereiht.

UNIX System Call Library: Die Programmschnittstelle des Kernels. Mit ihrer Hilfe kann man Programme schreiben, in denen dem Kernel Anweisungen gegeben und vom Kernel Informationen abgefragt werden können. Diese Schnittstelle ist auf Programme in der Programmiersprache C zugeschnitten.

UNIX–Kommandos: Die grundlegenden Programme, die zu UNIX standardmäßig dazugehören und vernünftiges interaktives Arbeiten erst ermöglichen.

Beispiele: ls, cp, mv, rm.

Auch diese Kommandos bauen auf der System Call Library auf und sind in C implementiert.

Shell: Die interaktive Schnittstelle, die Kommandos der Benutzerin entgegennimmt und die zugehörigen Programme aufruft. Es gibt mehrere Shells, die sich in Funktionalität und Bequemlichkeit unterscheiden.

X11 Window System: Verwaltet Fenster (Windows). In jedem Window läuft ein Programm. Das Programm, das in einem Window läuft, muß nicht auf demselben Computer laufen, sondern kann im Rahmen von lokalen und weltweiten Computernetzwerken prinzipiell überall laufen.

Wie UNIX bietet auch X11 eine Schnittstelle für C–Programme, die man für graphikorientierte Programme benutzen kann.

2 Files

2.1 Filetypen

Ordinary File: ASCII–Files und Binaries; siehe folgenden Text.

ASCII–File: „Normales“ Textfile, also die Art von Files, die man sich mit einem Editor ansehen kann.

ASCII (American Standard Code for Information Interchange) ist der allgemein verwendete Standard–Zeichensatz. Er besteht aus 128 Zeichen, d.h. wird mit sieben Bits codiert. Da Zeichen im allgemeinen als Bytes (acht Bits) gespeichert werden, bleibt ein Bit ungenutzt und ist meist gleich 0.

Es gibt auch erweiterte Varianten des ASCII–Zeichensatzes, die das achte Bit benutzen, um weitere 128 Zeichen anzubieten, z.B. deutsche Umlaute. Auf IBM–Großrechnern findet sich anstelle von ASCII der IBM–eigene EBCDIC–Code. Auf einigen älteren Rechnern werden auch noch weitere Zeichensätze verwendet.

Binaries: Ein File, in dem alle acht Bits eines Bytes genutzt werden, nennt man auch *Binary*. Bspw. sind komprimierte Files Binaries. Auch andere Daten–Formate, bei denen Lesbarkeit für menschliche Augen unwichtig ist, werden aus Platzgründen oft als Binaries gespeichert.

Programm: Ein File, dessen Inhalt elementare Maschinenbefehle sind. Programme sind Binaries.

Directory: Eine spezielle Art von File, dessen Inhalt eine Liste anderer Files ist. Da Directories ebenfalls Files sind, kann diese Liste wiederum andere Directories enthalten. Alle Directories zusammen bilden daher eine hierarchische Struktur. Jede Benutzerin hat eine feste *Home Directory* (Heimatverzeichnis), in der sie volle Zugriffsrechte besitzt und beliebig über Zugriffsrechte anderer Benutzerinnen (außer root) verfügen kann. Die Home–Directory von root ist die Spitze der Hierarchie.

Device: Jedes externe Gerät wird intern durch ein File repräsentiert. Lesen von einem Gerät heißt Lesen von diesem File, und Schreiben auf ein Gerät heißt Schreiben auf dieses File. Bspw. kann man von dem File, das mit der Tastatur verbunden ist, Zeichen lesen und in das File, das mit dem Bildschirm verbunden ist, Zeichen hineinschreiben.

Es gibt *Character Devices* (character=Zeichen), bei denen Ein- bzw. Ausgabe Zeichen für Zeichen erfolgt, bspw. Tastatur und Bildschirm. Im Gegensatz dazu stehen *Block Devices* (z.B. Diskettenlaufwerk), bei denen Daten immer in größeren Blöcken transferiert werden. Typische Blockgrößen sind 1K ($2^{10} = 1024$ Bytes), 2K (2048 Bytes) oder 4K (4096 Bytes).

Symbolic Link: Ein Alias für ein anderes File, d.h. ein neuer Name für ein schon existierendes File. Gehört dieses File zu einer anderen Directory als der Symbolic Link, kann man in letzterer Directory dennoch über den Namen des Symbolic Links darauf zugreifen, als ob das File in dieser Directory stünde. Symbolic Links sind also dazu da, die strenge Hierarchie der Directories flexibler zu gestalten.

Named Pipe: Ein weiterer Filetyp, der zur Kommunikation zwischen verschiedenen Prozessen verwendet wird.

2.2 Pfadnamen

Absoluter Pfadname: Jedes File hat genau einen absoluten Pfadnamen.[5] Der absolute Pfadname der *root*–Directory ist „/". Ein File „*Filename*", das in der root–Directory enthalten ist, hat absoluten Pfadnamen „/*Filename*". Sei „*Filename*" nun in einer anderen Directory enthalten und sei „*Dirpfadname*" der absolute Pfadname dieser Directory. Dann ist

„*Dirpfadname*/*Filename*"

der absolute Pfadname von „*Filename*". Da „*Filename*" der Name einer Directory sein kann, ist diese Regel rekursiv und definiert alle absoluten Pfadnamen.

Relativer Pfadname: Man kann ein File auch relativ zu einer beliebigen Directory spezifizieren. Der relative Pfadname eines Files in bezug auf eine Directory gibt den Weg durch die Directory–Hierarchie an, über den man von der Directory zu diesem File kommt.

Der relative Pfadname der aktuellen Working Directory eines Prozesses (siehe Kapitel 3.1) ist ein einzelner Punkt: „."; der Pfadname der Directory, die die aktuelle Directory unmittelbar enthält, besteht aus zwei Punkten: „..".

Sei „*Dirname*" der Name einer Directory und sei „*Filename*" der Name eines Files. Formal erhält man den relativen Pfadnamen von „*Filename*" in bezug auf Directory „*Dirname*" wie folgt: Man nimmt die absoluten Pfadnamen von „*Filename*" und „*Dirname*" und streicht aus beiden das Anfangsstück heraus, das sie gemeinsam haben. An das Reststück des absoluten Pfadnamens von „*Filename*" hängt man so oft die Zeichenkette „../" vorne an (zwei Punkte, ein Slash, kein Blank), wie es noch Directorynamen im Reststück des absoluten Pfadnamens von „*Dirname*" gibt. Das Endergebnis ist der relative Pfadname.

[5] Der Einfachheit halber ignorieren wir in Kapitel 2.2 Pfadnamen, die Symbolic Links und Schleifen enthalten; Hard Links ignorieren wir im ganzen Text.

2.3 Attribute von Files

Jedes File hat einen Namen und eine Besitzerin und gehört zu genau einer Gruppe von Benutzerinnen. Im Normalfall ist das die Hauptgruppe der Besitzerin. Es besitzt auch eine eindeutige ID (Kennummer, Inode–Number), die eine positive ganze Zahl ist. Außerdem werden für jedes File noch Zugriffszeiten mitprotokolliert und Zugriffsrechte verwaltet.

Zugriffszeiten: Zu jedem File wird gespeichert, wann das letzte Mal

- sein Inhalt gelesen wurde (*Access Time*);
- sein Inhalt verändert wurde (*Modification Time*);
- seine Attribute geändert wurden (*Change Time*).

Zugriffsrechte: Für jedes File gibt es drei verschiedene Arten von Zugriffsrechten.

Leserecht: Inhalt darf gelesen werden. Insbesondere darf Inhalt kopiert werden.

Bei Directories heißt das: Mit Kommandos wie *ls* darf man sich die Files auflisten lassen, die in der Directory enthalten sind.

Schreibrecht: Inhalt darf verändert werden.

Bei Directories heißt das: In dieser Directory darf man Files löschen und anlegen. Man darf auch dann ein File löschen, wenn man auf dem File selbst kein Schreibrecht hat.

Ausführrecht: Bei Programmen und Shell–Skripten (siehe Kapitel 6.2): Darf aufgerufen werden.

Bei Directories: Darf mit Kommandos wie *cd* betreten oder übersprungen werden.

Jedes dieser Zugriffsrechte kann separat gesetzt oder gelöscht werden für

- die Besitzerin selbst,
- die Mitglieder der Gruppe des Files,
- alle anderen Benutzerinnen.

2.4 Wichtige Standard–Directories

/ Root–Directory; die Spitze der Directory–Hierarchie.

/bin Essentielle Programme der UNIX–Standardauslieferung.

/dev Die Files, die die Kommunikation mit externen Geräten darstellen.

/etc Files für Systemadministration.

/kernel Das File /kernel/unix ist der UNIX–Kernel.

/tmp Für temporäre Datenhaltung, üblicherweise benutzt von Programmen, um temporäre Hilfsfiles anzulegen. Daten in dieser Directory werden bei Platzbedarf oder beim Hochfahren des Systems gelöscht und

im allgemeinen auch bei Backups (Sicherheitskopien auf Magnetband) nicht gesichert.

/var Für verschiedene Files wie Incoming Mailboxes und Druckerwarteschlangen.

2.5 Extensionen für Filenamen

Viele Files sind in speziellen Formaten abgefaßt und erfüllen nur spezielle Zwecke, z.B. Quelltexte, die in einer Programmiersprache abgefaßt sind. Um Mißverständnisse zu vermeiden, gibt es die allgemein in der UNIX–Welt akzeptierte Konvention, daß Format und Zweck eines Files schon am Namen des Files zu kennzeichnen sind, nämlich indem man eine *Namensextension* am Ende des File anhängt und sie durch einen Punkt abtrennt.

Wichtige Extensionen nach allgemeiner Übereinkunft:

a: Libraries aus vorkompilierten Unterprogrammen.

c: C–Quelltext.

cc: C++–Quelltext (oft statt dessen: *C*, *c++*, *cxx*, *cpp*).

dvi: dvi–Files. Das Textlayout, das aus einem TEX- oder LATEX–File konstruiert wird, wird im sogenannten *dvi*–Format gespeichert (dvi=device independent).

f: Fortran–Quelltext.

gif: Bilder im gif–Format.

html: Hypertext–Files für den World Wide Web (oft auch: *htm*).

o: vorkompilierter Maschinencode (oft auch: *bin*).

p: Pascal–Quelltext (oft auch: *pas*).

ps: PostScript–Code. PostScript ist eine Sprache zum Layout von Graphiken und Texten (genauer: Seitenbeschreibungssprache). Viele Drucker „verstehen“ nur PostScript, weswegen Files in anderen Formaten (z.B. *ASCII*, *dvi*, *gif*) vor dem Ausdrucken zuerst in PostScript umgewandelt werden müssen (meist automatisch durch die entsprechenden Druckkommandos).

s: Assemblercode.

tar: mit Kommando *tar* zu einer Einheit zusammengefaßte Files und Directories.

tex: TEX- und LATEX–Quelltexte. TEX ist eine mathematikorientierte Textlayoutsprache, und LATEX ist eine bequeme Textformatierungssprache auf der Basis von TEX. (Diese Arbeit wurde in LATEX abgefaßt.)

Z: mit *compress* komprimierte Files (statt dessen: *z* bei *pack*, *gz* bei *gzip*).

2.6 Konfigurationsfiles

Manche Programme lesen am Beginn der Ausführung den Inhalt von *Konfigurationsfiles* und lassen ihr Verhalten davon steuern. Per Konvention beginnen die Namen von Konfigurationsfiles mit einem Punkt („*dot–Files*“) und stehen in der Home Directory. Meist enthält der Name des Konfigurationsfiles den Programmnamen, und oft endet der Filename auf *rc* (=*run commands*). In Directories, deren Namen mit einem Punkt beginnen, werden mehrere Konfigurationsfiles für dasselbe Programm gesammelt. Beim Auflisten von Programmen mit *ls* werden dot–Files nur bei Option *-a* angezeigt.

Einige wichtige Konfigurationsfiles

Die folgenden Files sind nur wirksam, wenn sie in der Home Directory stehen.

.apps Aufruf von Programmen beim Einloggen; spezifisch für den Bereich Informatik der Universität Konstanz.

.bashrc Für die Shell *bash*.

.bashrc.*kennung*: Persönliche Bash–Zusatzdefinitionen der Benutzerin mit Benutzerinnenkennung *kennung*; spezifisch für den Bereich Informatik der Universität Konstanz.

.cshrc Für die Shells *csh* und *tcsh*.

.cshrc.*kennung*: Analog zu *.bashrc.kennung*.

.elm/elmrc Für das Mailprogramm *elm*.

.elm/aliases.text Sammlung von Mail–Aliassen für den *elm*.

.emacs Für den Editor *emacs*.

.exrc Für die Editoren *ex* und *vi*.

.forward Um eingegangene Emails automatisch weiterzuschicken oder sonstwie vorab zu filtern oder zu bearbeiten.

.mailrc Für das Programm *mail* und andere Email–Programme.

.plan Persönlicher Zusatz zur Standardausgabe des Kommandos *finger*.

.profile Für die Bourne–Shell *sh* und die Korn–Shell *ksh*.

.signature Inhalt ist bei einigen Email–Programmen (z.B. *elm*) automatisches Anhängsel für alle abgeschickten Emails (bei entsprechender Konfiguration des Programms).

3 Prozesse

Unter UNIX können mehrere Prozesse gleichzeitig abgearbeitet werden. Derselbe Rechner kann auch Prozesse verschiedener User gleichzeitig ablaufen lassen.

Es kann auch gleichzeitig mehrere Prozesse geben, die dasselbe Programm ausführen. In jedem Fall laufen diese Prozesse völlig unabhängig voneinander, es sei denn sie sind absichtlich miteinander gekoppelt.

3.1 Attribute von Prozessen

- Effective User ID/Effective Group ID: Prozeß hat gleiche Zugriffsrechte wie der User und die Group, die dadurch spezifiziert sind.
- PID: Eindeutige Kennummer (ID) des Prozesses.
- PPID: Prozeß–ID des Parent Prozesses (siehe Kapitel 3.2).
- Kommandozeile: Der Aufruf des Kommandos mit allen Optionen und Argumenten (allerdings *nach* Verarbeitung durch die Shell; siehe Kapitel 6.2).
- Priorität: Eine Zahl, die angibt, durch welche Ereignisse ein Prozeß unterbrochen werden kann und wie lange er in der Warteschlange bleiben muß, bevor er wieder eine Zeitscheibe zugeteilt bekommt.
- Startzeit.
- Totale verbrauchte CPU–Zeit bisher.
- Filedeskriptoren: Prozesse können Files zum Lesen und/oder Schreiben öffnen und sind mit ihnen dann über sogenannte *Deskriptoren* verbunden. Ein Prozeß hat üblicherweise mindestens die folgenden drei Deskriptoren: *stdin* (Standardeingabe), *stdout* (Standardausgabe) und *stderr* (Standardfehlerausgabe).
- Current Working Directory (CWD): Wenn der Prozeß einen System Call mit einem relativen Pfadnamen als Argument macht, wird dieser Pfadname relativ zur CWD ausgewertet. Die CWD kann während der Ausführung des Prozesses geändert werden.
- Momentaner Zustand; die wichtigsten sind:
  - Running: Wird gerade abgearbeitet.
  - Runnable: In der Warteschlange.
  - Sleeping: Wartet auf Event (z.B. Eingabe der Benutzerin über Tastatur oder Maus).

3.2 Erzeugung von Prozessen

Ein Prozeß wird durch einen anderen Prozeß, den *Parent Process* erzeugt; nur der Prozeß Nr. 0 wird beim Hochfahren des Systems anders erzeugt. Ein Prozeß (*Child Process*) wird in zwei Schritten aus einem anderen erzeugt:

1. Der Parent Process ruft System Call „*fork*“ auf. Dadurch wird eine exakte Kopie des Parent Processes angelegt, die sich im selben Zustand der Ausführung befindet wie der Parent Process selbst. Einzige Unterschiede sind PID und PPID. Außerdem ist der Rückgabewert von *fork* für beide Prozesse verschieden, woran jeder Prozeß erkennt, ob er Parent oder Child ist.
2. Der Child Process ruft System Call „*exec*“ mit dem Namen des Programms und den Kommandozeilenargumenten auf. Dadurch wird im Child Process das eigentlich auszuführende Programm gestartet, und Startzeit und verbrauchte CPU–Zeit werden initialisiert.

Durch diese zweistufige Erzeugung „erbt“ jeder Prozeß die Attribute des aufrufenden Prozesses.

3.3 Signale

Prozesse können einander Signale schicken. Zu jedem Signal gibt es eine *Default*–Reaktion; bei den meisten Signalen ist die Default–Reaktion sofortige Beendigung (*Terminierung*). In vielen Programmen wird die Default–Reaktion auf ein Signal durch eine eigene Behandlungsroutine für dieses Signal überschrieben (z.B. um alle offenen Files vor Terminierung abzuspeichern).

CONT (Continue) Default–Reaktion: Schlafengelegter Prozeß wird wieder „aufgeweckt“.

HUP (Hangup) Wenn sich eine Benutzerin ausloggt, wird dieses Signal an alle ihre Prozesse geschickt.

Default–Reaktion: Terminierung.

KILL Default–Reaktion: Terminierung. Bei diesem Signal kann die Default–Reaktion **nicht** überschrieben werden.

SEGV (Segmentation Violation) Ein typischer Fehler bei C–Programmen (unzulässiger Pointerzugriff).

STOP Prozeß wird schlafengelegt.

TERM (Terminate) Default–Reaktion: Terminierung.

WINCH (Window Size Change) Wird an Prozeß geschickt, wenn die Größe eines Windows, das zu diesem Prozeß gehört, sich geändert hat.

Typischerweise können einige Signale auch durch Kontrollsequenzen abgeschickt werden (siehe Kommando *stty* Kapitel 8.4).

3.4 Daemons

Das sind Prozesse, die immer laufen. Bspw. gibt es einen Mail–Daemon, der ständig Ausschau nach eingehender Email hält und sie an die jeweils richtige Adressatin verteilt. Typischerweise werden Daemons beim Hochfahren des Systems automatisch mitgestartet und haben root–ID als effektive User–Id.

Ein wichtiger Daemon ist *cron*. Dieser Daemon durchsucht relativ oft in festgesetzten Zeitintervallen eine Tabelle, in der jeder Eintrag aus einem Kommando und einer Zeitangabe besteht (Tageszeiten, Wochentage, Tage im Monat). Wann immer die aktuelle Zeit dieser Zeitangabe entspricht, wird das Kommando (genannt *cron-Job*) gestartet.

4 X11 Window System

Dieser Draufsatz auf UNIX verwaltet die Windows auf dem Bildschirm und steuert die Kommunikation zwischen der Benutzerin und den Prozessen, die in den einzelnen Windows laufen.

In jedem Window läuft ein Programm, das *Client* genannt wird. X11 ordnet jeden *Event* (Eingabe der Benutzerin über Tastatur oder Maus) dem richtigen Window zu und übergibt die Eingabe an den darin laufenden Client.

4.1 Wichtige Attribute von Windows

- Client: Das Programm, das in diesem Window läuft.
- Depth: Anzahl Bits pro Bildpunkt.
- Geometry: Höhe, Breite sowie die Koordinaten des oberen linken Punktes, gemessen in Pixels (Bildschirmpunkte).
- Name:[6] Name des Windows; wird typischerweise im äußeren Rahmen eines Windows angezeigt.
- Visual Class: schwarz–weiß, Graustufen, farbig usw. Jede Visual Class erfordert eine Mindestgröße für Depth.

4.2 X–Server, Display, Screen

Ein Display besteht üblicherweise aus einem Bildschirm, einer Tastatur und einem *Pointing Device* (Zeige–Gerät, z.B. Maus). In X11 gehört zu jedem Display ein Daemon, der *X–Server*, der das Display steuert und kontrolliert. Der X–Server verwaltet einen *Pointer* auf dem Bildschirm, der durch das Pointing Device bewegt werden kann. Das Window, in dem der Pointer ist, heißt *aktiv*. Alle Events von diesem Display werden vom X–Server an den Client weitergereicht, der im aktiven Window läuft.

Ein *X–Terminal* ist eine spezielle Art von Rechner, auf dem nur der X–Server läuft (nicht zu verwechseln mit einem *Xterm*–Window). Alle Clients im Display eines X–Terminals laufen auf anderen Rechnern.

An einem Rechner können mehrere Displays hängen, etwa an einem Zentralrechner. Diese Displays sind fortlaufend numeriert, startend mit 0. Ein Display verwaltet ein oder mehrere *Screens*, die ebenfalls fortlaufend numeriert sind und mit 0 starten. Üblich ist genau eine Screen für ein Display. Ein Beispiel für zwei Screens ist ein Display, das wahlweise schwarz–weiß oder farbig laufen kann.

Wenn man die entsprechenden Rechte besitzt, kann man auch über das lokale Netz oder über das Internet eine Screen auf einem anderen Rechner ansprechen mit

<Internet–ID>:<Display–Nr>.<Screen–Nr>

Jeder X–Server verwaltet eine *Access Control List*, in der alle anderen X–Server notiert sind, die auf sein Display momentan zugreifen dürfen.

4.3 X11 Library

Außerdem bietet X11 eine Library für C–Programme, auf deren Basis alle Clients implementiert sind. Diese Library enthält hinzuladbare Unterprogramme für Manipulationen an Windows wie z.B. Öffnen, Schliessen, Bewegen, Vergrößern und Verkleinern, ebenso für das Empfangen von Events und die Darstellung von Texten und Bildern in einem Window.

4.4 Xterm

Der wichtigste Window–Typ ist Xterm. Ein *Xterm*–Window simuliert ein zeichenorientiertes Terminal, d.h. ein Terminal, bei dem sich der Bildschirm als eine Matrix von Zeichen darstellt (ohne Graphik, ohne Windows). Der Client eines Xterm–Windows ist eine Shell oder ein von einer Shell gestarteter Prozeß.

4.5 Window Manager

Ein spezieller Client zur einfachen Handhabung von Windows. Der Window Manager fügt zu jedem Window einen Rahmen hinzu, so daß man einfach durch Maus–Klick damit umgehen kann. Außerdem stammen die Hintergrund–Menüs vom Window Manager. Es stehen mehrere komfortable Window Manager zur Auswahl.

[6] Nur bei Top–Level Windows.

5 Internet

Das Internet ist ein weltweites Netz, an dem inzwischen Millionen von lokalen Rechnern und Rechnernetzen angeschlossen sind. Man kann es für viele verschiedene Dinge verwenden, z.B. elektronische Post, Filetransfer, Datenbankanfragen und Arbeiten auf weit entfernten Rechnern.

Neuerdings hinzugekommen ist das *World Wide Web (WWW oder W3)* als eine Möglichkeit, das Internet zu nutzen. Das WWW besteht aus einer unüberschaubar großen Zahl von Dokumenten (vor allem Texte, auch Bilder, Sound–Files etc.) überall auf der Welt, die von ihren Autoren auf dem Internet offen zugänglich gemacht worden sind und die durch Querverweise massiv miteinander vernetzt sind.

5.1 Domain Name System

Das Internet zerfällt in viele *Domains*, die hierarchisch gegliedert sind, d.h. eine Domain ist Teil (*Subdomain*) einer anderen. Bei der Adressierung eines Rechners gibt man zuerst den Rechnernamen und dann die Domains in aufsteigender Reihenfolge an. Zwei Domainnamen werden dabei durch einen Punkt getrennt.

Beispiel: konstanz.pool.informatik.uni–konstanz.de

Im wesentlichen gibt es genau eine Top–Level Domain für jedes Land.

Beispiele:

de Bundesrepublik Deutschland.

at Österreich (Austria).

ch Schweiz (Confoederatio Helvetica).

fr Frankreich.

uk Großbritannien (United Kingdom).

Obwohl die USA als Ganzes und jeder Bundesstaat ebenfalls jeweils ein eigenes Kürzel hat, gibt es aus historischen Gründen mehrere weitere Top–Level Domains speziell für die USA. Üblicherweise werden Adressen in den USA durch folgende Top–Level Domains spezifiziert:

com Firmen (commercial).

edu Schulen und Hochschulen (education).

gov Behörden (government).

net Zentrale Netzverwaltung.

mil Militär.

org Organisationen (z.B. wissenschaftliche).

Außerdem bietet das Internet Übergänge zu anderen Netzen über *Gateways*, z.B. zum Bitnet, zu diversen kommerziellen Netzen und zum Datex–J der deutschen Telekom. Ein Gateway ist ein spezieller Rechner, der zum Internet und zu einem anderen Netz verbunden ist und Daten von einem Netz in das andere weiterleiten kann.

5.2 Datentransfer

Eine Benutzerin, die Daten über das Internet verschicken will, gibt im Normalfall den Namen des Zielrechners oder des lokalen Zielnetzes gemäß *Domain Name System* an. Jede Domain enthält einen oder mehrere Name Server, die mindestens volle Information über die eigene Domain (nicht unbedingt über die Subdomains) haben. Ein Name Server, der volle Information über den Zielrechner hat, wird angefragt und liefert dessen *IP–Adresse* zurück. Das ist eine 32–Bit–Zahl, die die Zieldomain und — innerhalb der Domain — den Zielrechner eindeutig kodiert.

Datentransfer im Internet basiert auf dem *Internet Protocol (IP)*. Ein solches Protokoll hat eine ähnliche Funktion wie ein Briefumschlag, d.h. es hängt technische Zusatzinformationen an jedes gesendete Datenpaket an. Das IP–Protokoll enthält unter anderem die IP–Adresse des Zielrechners. Es ist sehr einfach gehalten und sieht bspw. weder eine Empfangsbestätigung noch einen Vorabtest, ob der Zielrechner momentan ansprechbar ist, vor. Für solche Zwecke werden Zusatzprotokolle benutzt. Deren spezifische Zusatzinformationen bilden einen Briefumschlag innerhalb des IP–Briefumschlags. Das am meisten verwendete Zusatzprotokoll ist das *Transmission Control Protocol (TCP)*.

Das Internet zerfällt organisatorisch in viele Subnetzwerke, die nicht unbedingt mit den Domains übereinstimmen. Datentransfer zwischen den Subnetzwerken läuft über *IP–Router*. Das sind zentrale Rechner in jedem Subnetz, die mit den Routern „benachbarter“ Subnetze verbunden sind. Jeder Router verwaltet tabellarische Informationen, die es ihm ermöglichen, für jede IP–Adresse einen Router zu bestimmen, der dieser IP–Adresse einen Schritt „näher“ ist. Das heißt, Datentransfer im Internet vollzieht sich dezentral: Jeder Router kennt nur den jeweils nächsten Router, und keine zentrale Instanz steuert den Netzverkehr.

6 Shells

Die *Shell* verarbeitet zunächst die Eingabe nach ihren eigenen Regeln und sucht das Programm (oder Shell–Skript) zum eingegebenen Kommando (siehe Kapitel 6.2). Findet sie kein solches Programm, gibt sie

eine Fehlermeldung aus. Ansonsten startet die Shell einen Child Process mit diesem Programm.

Falls der Prozeß Eingaben von *stdin* verlangt, müssen diese Eingaben in *xterm* eingegeben werden (Ausnahme: Redirection und Pipes, Kap. 6.2). Diese Eingaben werden von *xterm* direkt an das Programm weitergereicht, unter Umgehung der Shell. Die Ausgaben des Prozesses auf *stdout* und *stderr* werden wieder unter Umgehung der Shell über das Window *xterm* auf den Bildschirm geschrieben (ebenfalls mit Ausnahme Redirection und Pipes).

6.1 Verschiedene Shells

sh: Die *Bourne*–Shell, die Urmutter aller Shells.

ksh: Die *Korn*–Shell, eine Erweiterung der Bourne–Shell, speziell entwickelt zur Shell–Programmierung (siehe Kapitel 6.2).

csh: Die *C*–Shell, eine Erweiterung der Bourne–Shell, speziell entwickelt zum interaktiven Gebrauch.

tcsh: Die *TC*–Shell, eine Erweiterung der C–Shell, die bequemer interaktiv zu benutzen ist und auch viele Möglichkeiten der Korn–Shell anbietet.

bash: Bietet ähnliche Möglichkeiten wie die TC–Shell.

rsh: Unglücklicherweise bezeichnet man damit zwei verschiedene Shells: die *Remote Shell* (auch: *remsh*) und die *Restricted Shell*. Die erstere ist dazu da, kurze Aktionen auf einem anderen Rechner durchzuführen. Die zweite Shell ist dazu da, bestimmten Benutzerinnen nur eingeschränkte Zugriffsmöglichkeiten zu gewähren.

6.2 Wichtige Shell–Konzepte

- Built–in Kommandos: Eine Reihe von Kommandos, die man in einer Shell eingeben kann (z.B. *cd*), sind keine eigenständigen Programme, sondern Unterprogramme der Shell selbst.

- Alias: Ein Shell–Prozeß verwaltet eine Liste von Abkürzungen, die man definieren kann, um sich Tipparbeit zu sparen oder lange, komplizierte Kommandosequenzen durch leicht zu merkende Kürzel zu ersetzen. Üblicherweise werden einige Standard–Aliasse im Konfigurationsfile der Shell definiert.

- Shell–Skript: Shells bieten einige Built–in Kommandos, die in beschränktem Umfang typische Konstrukte höherer Programmiersprachen realisieren. Solche „*Shell–Programme*“ werden üblicherweise nicht interaktiv eingegeben, sondern in *Skripten* zusammengefaßt. Sie werden wie ausführbare Maschinenprogramme aufgerufen.

- Environment–Variablen. Auf den Wert einer Variablen wird mit $*variable* zugegriffen. Die wichtigsten sind:
  - CWD: Die Current Working Directory des Shell–Prozesses.
  - HOME: Die Home Directory der Benutzerin.
  - MANPATH: Liste von Directories. Das Programm *man* durchsucht der Reihe nach jede Unterdirectory jeder Directory in dieser Liste nach einer Man Page für sein Argument, bis eine Man Page gefunden wird (vgl. Kapitel 7). Findet *man* keine, gibt es eine Fehlermeldung.
  - PATH: Liste von Directories. Für jedes Kommando, das weder ein Built–in noch ein Alias ist, wird jede dieser Directories nach einem ausführbaren File dieses Namens durchsucht, bis eines gefunden wurde. Wird keines gefunden, gibt es eine Fehlermeldung. Wird ein Programm nicht gefunden, bedeutet das also nicht, daß dieses Programm nicht existiert, sondern nur, daß es nicht in einer dieser Directories enthalten ist.

- Alle gängigen Shells außer der Bourne–Shell interpretieren „~*benutzerinnenkennung*“ als die Home–Directory der entsprechenden Benutzerin und eine isolierte Tilde ~ als die eigene Home–Directory.

- Wird die Kommandozeile durch „&“ vor *Return* abgeschlossen, wird das Kommando „im Hintergrund“ abgearbeitet und blockiert nicht die Shell. Nur geeignet für Programme, die nicht über die Shell mit der Benutzerin kommunizieren, sondern bspw. ein eigenes Window dafür aufmachen.

- Zwei Kommandos, die durch „|“ getrennt sind, bilden wieder ein Kommando. Die Ausgabe des ersten Kommandos auf *stdout* wird *nicht* angezeigt, sondern als Eingabe auf *stdin* an das zweite Kommando weitergereicht (*Pipe*).

- Mit „< *Filename*“ werden Eingaben auf *stdin* während der Abarbeitung des Programms nicht von der Tastatur gelesen, sondern aus File *Filename* (*Input Redirection*).

- Mit „> *Filename*“ werden Ausgaben des Programms auf *stdout* nicht auf dem Bildschirm ausgegeben, sondern in File *Filename* hinausgeschrieben (*Output Redirection*).

- Ein Argument auf der Kommandozeile, das ein oder mehrere * enthält, wird ersetzt durch alle Files, deren Name sich aus dem Argument ergibt, indem jeder * durch eine beliebige Zeichenkette ersetzt wird (Zeichenkette kann auch leer sein). Ein *?* steht hingegen für genau ein beliebiges Zeichen.

- History: Bash, Csh, Ksh und Tcsh speichern immer die letzten Befehle ab (bis zu einer gewissen Zahl, die in einer weiteren Shell–Variable steht). Auf diese Befehle oder Teile davon kann man mit speziellen Kommandosequenzen zugreifen.
- Teile der Kommandozeile, die durch Double–Quotes oder Single–Quotes (d.h. "..." oder '...') zusammengefaßt sind, werden nicht von der Shell verarbeitet, sondern „wörtlich“ an das Programm weitergereicht. (Bei Double–Quotes behalten einige Sonderzeichen ihre Sonderbedeutung, z.B. $.) Ebenso wird ein einzelnes Zeichen nicht weiter verarbeitet, wenn ihm ein „\“ vorangestellt wird.
- Hash: Einige Shells speichern aus Effizienzgründen alle Namen von ausführbaren Files, die in den Directories in PATH enthalten sind, in einer internen Datenstruktur ab, einer sogenannten *Hash Table*. In diesem Fall wird nur die Hash Table für ein Kommando durchsucht. Falls sich die Menge der ausführbaren Files in diesen Directories ändert, muß die Hash Table neu aufgebaut werden. Typischerweise wird dies durch einen cron–Job erledigt (z.B. jede Nacht).

Systematische Beschreibung: in den Man Pages der einzelnen Shells (s.u.).

7 Man Pages

Zu den meisten Kommandos (aber auch zu Funktionen der C–lib u.ä.) gibt es „*Manual Pages*“ (Handbuchseiten, kurz: *Man Pages*). Die Man Page zu einem Kommando bekommt man mit:

„man *Kommandoname*“

7.1 Kapitel in einer Man Page

Die folgenden Kapitel sind die wichtigsten und in den meisten Man Pages vorhanden.

NAME: Name und Kurzbeschreibung.

SYNOPSIS: Wie das Kommando zu verwenden ist. Ein Kommandoaufruf besteht aus dem *Kommandonamen* und mglw. nachfolgenden *Optionen* und *Argumenten*. Optionen beginnen meistens mit einem Minuszeichen und können ihrerseits wieder Argumente haben.

DESCRIPTION: Ausführliche Beschreibung des Kommandos.

OPTIONS: Auflistung und genaue Beschreibung der verschiedenen Optionen und ihrer Argumente.

FILES: Files, die für das Kommando wichtig sind (z.B. Konfigurationsfiles).

DIAGNOSTICS: Auflistung der verschiedenen Fehlermeldungen, die man erhalten kann, wenn man das Kommando nicht gemäß SYNOPSIS verwendet.

BUGS: Fehler im Kommando selbst, die zwar erkannt, aber noch nicht behoben sind.

SEE ALSO: Weitere Kommandos, die in enger Beziehung zum beschriebenen Kommando stehen.

7.2 Synopsis

Die Bezeichner, die in der Synopsis verwendet werden, werden üblicherweise in den Kapiteln *DESCRIPTION* und *OPTIONS* genauer beschrieben. Die wichtigsten Regeln, eine Synopsis zu verstehen, sind folgende:

- Ausdrücke in eckigen Klammern sind optional, d.h. dürfen vorhanden sein, dürfen aber auch fehlen.
- Sind mehrere Optionen hintereinander mit einem einzigen Minuszeichen vorneweg aufgeführt, dann kann eine beliebige Kombination dieser Optionen in beliebiger Reihenfolge (ohne Blanks dazwischen) mit einem Minuszeichen angegeben werden. Die einzelnen Optionen dürfen aber auch blank–separiert mit jeweils eigenem Minuszeichen angegeben werden.
- Drei Punkte bedeuten, daß der vorhergehende Bezeichner nicht für ein einziges Objekt steht, sondern stellvertretend für eine beliebig lange Liste von Objekten gleicher Art (typischerweise Filenamen).
- Geschweifte Klammern fassen Alternativen zusammen, die durch Kommas voneinander getrennt sind. Senkrechte Striche trennen ebenfalls Alternativen.

7.3 Sektionen für Man Pages

Man Pages sind in Sektionen organisiert. Gibt es Man Pages zum selben Namen in verschiedenen Sektionen, kann man die richtige bei Aufruf von *man* durch Option *-s* auswählen. Siehe man man.

Wichtige Sektionen:

1. UNIX–Kommandos.
2. System Calls.
3. Die C–Library außer System Calls.
4. Beschreibung spezieller Files.
5. Globale Konstanten u.ä.
6. Spiele und Demonstrationen.

7.4 Ausgewählte Man Pages

In Klammern ist jeweils die Sektion notiert.

ascii (5): ASCII–Zeichensatz in Oktal- und Hexadezimalcode.

ex (1): Listet u.a. Kommandos für den Editor *vi* und andere Fileprozessoren auf.

forward (4): Beschreibt u.a. den korrekten Inhalt eines Files „ *~/.forward*“.

grep (1): Enthält neben der Beschreibung des Kommandos *grep* eine Definition von *Regular Expressions*.

intro: Gibt es üblicherweise in jeder Sektion und enthält eine Einführung in die jeweilige Sektion. Die *intro*–Page in Sektion 2 enthält außerdem Definitionen von grundlegenden UNIX–Konzepten (z.B. Files, Directories, Zugriffsrechte,...).

ls (1): Listet u.a. die Attribute von Files auf.

man (1): Beschreibung des Kommandos *man*.

man (5): Wie man Man Pages schreibt. Listet auch die Grundregeln für die Interpretation des Kapitels SYNOPSIS auf.

ps (1): Listet u.a. die Attribute von Prozessen auf.

sh (1), ksh (1), csh (1), tcsh (1), bash (1): Informationen zur jeweiligen Shell. Wenn eine Shell Weiterentwicklung einer anderen ist, werden die gemeinsamen Eigenschaften zum Teil nur in der Man Page zur ursprünglichen Shell beschrieben.

shell_builtins (1): Eine Liste aller gängigen Built–in Kommandos von Shells.

signal (5): Liste aller Signale an Prozesse.

test (1): Listet u.a. die Möglichkeiten für Boolesche Ausdrücke in eckigen Klammern auf.

X (1): Intro–Page für das X11–Windowsystem.

8 Wichtige Kommandos und Shell Built–ins

8.1 Informationen zu Kommandos

answerbook: On–line Dokumentationssystem.

apropos: Ausgabe aller NAME–Zeilen von Man Pages, die das Argument als Muster enthalten.

echo: Gibt seine Argumente wieder aus, allerdings *nach* Bearbeitung durch die Shell.

locate: Findet Programme und andere Files, die allgemein zugänglich sind auf der Basis einer internen Indexierung. Diese Indexierung wird nur durch einen cron–Job regelmäßig neu aufgebaut, so daß die Ausgabe von *locate* nicht immer absolut aktuell ist.

man: Manual Page zu einem Kommando, einem grundlegenden File oder einer C–Funktion (auch: *xman*).

whatis: Kurzbeschreibung eines Kommandos (NAME–Zeile jeder Man Page für dieses Kommando).

whereis: Mindestens Auflistung aller ausführbaren Programme mit dem Argument als Namen in den Directories in *PATH* sowie aller Man Pages in den Directories in *MANPATH*.

which: Falls das Argument ein Programm oder Shell–Skript ist, wird der absolute Pfadname des Programms ausgegeben; falls ein Alias, die Alias–Definition; falls ein Built–in Kommando der Shell, eine entsprechende Mitteilung.

8.2 Kontrolle von Prozessen

at: Prozeß wird zu angegebener Zeit gestartet.

batch: Prozeß wird erst gestartet, wenn Systembelastung niedrig ist.

bg: Prozeß in Hintergrund schieben und reaktivieren (vorher mit Signal STOP kurzfristig stoppen; STOP ist meist auf *Ctrl–z* gelegt). Dieses Kommando und die nächsten zwei sind Shell Built–ins und nicht für jede Shell verfügbar.

fg: Prozeß in Vordergrund holen.

jobs: Child–Prozesse einer Shell anzeigen.

kill: Schickt Signale an Prozesse (auch: *xkill*).

nice: Setzt für einen Prozeß eine andere Priorität fest.

nohup: Prozeß ignoriert Signal *HUP*.

time: Protokolliert Laufzeit eines Prozesses.

top: Liste der Prozesse, die momentan am meisten CPU–Zeit verbrauchen.

truss: Protokolliert alle System Calls des Prozesses und alle Signale, die während seiner Ausführung an ihn geschickt werden.

8.3 Allgemeine Informationen

cal: Kalendar für (fast) beliebige Jahre und Monate.

date: Datum und Uhrzeit.

leave: Kurzfristige, automatische Erinnerung an dringende Termine.

xclock: „Wanduhr“ (auch: *clock*, *dclock*, *oclock*, *xdaliclock*).

8.4 Informationen zum Rechner

du: Von einer Directory und allen Ihren Unterdirectories belegter Speicherplatz.

df: Momentane Auslastung der Festplatten.

pagesize: Größe einer Rechnerseite.

sysinfo: Allgemeine technische Informationen zum Rechner.

stty: Zum Setzen und Ausgeben von Terminaloptionen (z.B. Tastenbelegungen).

xhost: Gibt Access Control List aus.

xload: Momentane Auslastung der CPU.

8.5 Benutzerinnen

finger: Informationen über andere Benutzerinnen.

groups: Auflistung aller Groups, zu denen eine Benutzerin gehört.

passwd: Passwort ändern.

rusers: Alle momentan im lokalen Rechnernetz eingeloggten Benutzerinnen (auch: *rwho*).

su: Benutzerin in einem einzelnen Xterm ändern.

users: alle momentan auf eigenem Rechner eingeloggte Benutzerinnen (auch: *who*).

8.6 Informationen von X11

xdpyinfo: Informationen zum Display.

xhost: Gibt Access Control List aus.

xlsclients: Listet alle Clients auf einem Display auf.

xwininfo: Auflistung einiger Attribute eines Windows.

8.7 Rechnen mit dem Rechner

bc: Formelauswerter (auch: *dc*).

xcalc: Bildschirm–Taschenrechner (auch: *calctool*).

8.8 Behandlung von Files als Ganzem

cat: Mehrere Files zu einem zusammenfügen (*konkatenieren*).

chmod: Änderung der Zugriffsrechte.

chgrp: Gruppenzugehörigkeit eines Files ändern.

chown: Besitzerin eines Files ändern.

compress: Komprimieren von Files (auch: *gzip*, *pack*).

cp: Kopieren von Files.

file: Typ eines Files systematisch "erraten".

grep: Zeilen ausgeben, die bestimmte Zeichenketten enthalten (auch: *egrep*, *fgrep*).

filemanager: Interaktiver Umgang mit Files und Directories.

ln -s: Symbolic Link einrichten.[7]

[7] Ohne Option *-s* richtet *ln* einen Hard Link anstelle eines Symbolic Link ein. Auf Hard Links gehen wir hier aber nicht weiter ein.

ls: Auflistung von Files, mit geeigneten Optionen auch Auflistung der Attribute. Ohne Option *-a* werden dot–Files nicht aufgelistet.

mv: Änderung der Namen von Files und/oder ihrer Directory.

rm: Files löschen.

tar: Lesen von und Schreiben auf Devices (z.B. Tapes), auch Zusammenfassen von mehreren Files.

uncompress: Entkomprieren von Files (auch: *gunzip*, *unpack*).

wc: Zeichen, Wörter und Zeilen zählen.

8.9 Filterprogramme

Ein Filter ist ein Programm, das Eingaben von *stdin* liest, diese Eingabe manipuliert und das Ergebnis auf *stdout* ausgibt. Filter werden oft in Zusammenhang mit Pipes und Redirection verwendet. Die meisten bieten auch Optionen, mit denen man *stdin* und *stdout* durch ein oder mehrere Files ersetzen kann (ohne Redirection).

awk: sehr flexibles Fileprocessing (auch: *perl*).

cut: Einzelne Spalten ausfiltern.

diff: Unterschiede zwischen zwei Files herausfinden (auch: *bdiff*, *cmp*).

head: Nur Anfangszeilen der Eingabe ausgeben.

sed: Ediert jede Zeile gemäß einem vorgegebenen sed–Skript.

sort: Sortiert Zeilen nach sehr flexibel definierbarem Sortierschlüssel.

tail: Nur Endzeilen der Eingabe ausgeben.

8.10 Files edieren

less: Inhalt von Files lesen (auch: *more*, *pg*).

emacs: Inhalt von Files lesen und interaktiv ändern (auch: *textedit*, *vi*).

8.11 Ausdrucken von Files

dvips: Ausdrucken von dvi–Files und/oder Übersetzen nach PostScript.

lp: Ausdrucken von PostScript–Files.

lpstat: Zustand und Inhalt der Warteschlange für den Drucker.

lprm: Löscht Druckauftrag aus der Warteschlange für den Drucker.

prx: Ausdrucken von ASCII–Text; spezifisch für den Bereich Informatik der Universität Konstanz (auch: *a2ps*).

xpr: Inhalt eines X11–Windows drucken.

8.12 Directories

basename: Den eigentlichen Namen eines Files aus einem Pfadnamen extrahieren und ausgeben.

dirname: Komplement zu *basename*.

mkdir: Neue Directory einrichten.

rmdir: Directory löschen.

rm -r: Directoryhierarchie rekursiv löschen.

cd: Aktuelle Working Directory einer Shell ändern.

pushd, popd: Built–in von einigen Shells. Diese Shells verwalten einen Stapel von Directories, wobei oberste Directory auf diesem Stapel die aktuelle Working Directory ist. Die Kommandos *pushd* und *popd* verwalten zusammen diesen Stapel.

8.13 Finden von Files

find: Lokalisiert File(s) und tut dabei gleich etwas mit gefundenen Files.

locate: Lokalisiert File(s), auf die jedermann zugreifen darf (Kommentare siehe Kapitel 8.1).

8.14 Programmieren (außer C/C++)

ar: Erstellen von Programmbibliotheken.

emake: Eiffel–Compiler.

f77: Fortran77–Compiler.

make: Werkzeug zur Verwaltung von Kompilierarbeiten u.ä. bei größeren Projekten (auch: *makedepend*, *imake*).

p2c: Übersetzer von Pascal nach C.

pc: Pascal–Compiler.

ranlib: Vorbereiten einer Library zur Benutzung.

8.15 Programmieren mit C/C++

c++–filt: Linker–Referenzen für C++–Funktionen zurück in Signaturen übersetzen.

efence: Zum Abtesten von falschen Pointerzugriffen auf den Freispeicher.

gcc: C–Compiler (auch: *acc*, *cc*).

g++: C++–Compiler (auch: *CC*).

runtest: Umgebung zum systematischen Austesten von Programmen.

xwpe: Programmierumgebung ähnlich wie Turbo–C.

xxgdb: Debugger, unterstützt Fehlersuche in Programmen (auch: *ctrace*, *ddd*, *debugger*, *gdb*, *lint*, *xgdb*).

8.16 Textverarbeitung

bibtex: Programm zur Nutzung von Literaturdaten für LaTeX-Files.

dvips: Ausdrucken von dvi–Files und/oder Übersetzen von dvi–Files in PostScript–Format.

ghostview: Anschauen von PostScript–Files (auch: *pageview*).

imaker: Das Desktop–Publishing Programm *Framemaker*.

latex: LaTeX, ein mathematikorientiertes Textverarbeitungs- und Layoutprogramm. Übersetzt LaTeX-Quelltexte in dvi–Format.

tex: TeX, die Grundlage von LaTeX.

xdvi: Anschauen von Files in dvi–Format.

xfig: Erstellung schematischer Graphiken.

xv: Manipulation von Graphiken und Bildern.

xwd: Windowbild in einem File abspeichern.

8.17 Rechnernetze

elm: Programm zum Lesen und Verschicken von elektronischer Post (auch: *mail*, *mailtool*).

finger: Informationen über andere Benutzerinnen auf demselben oder anderen Rechnern.

ftp: Kopieren von Files weltweit von und zu anderen Rechnern (auch: *ftptool*, *ncftp*).

netscape: Zugriff auf WWW (auch: *Mosaic*).

nslookup: Informationen über Rechner und Domains auf dem Internet.

ping: Testet, ob ein Rechner auf dem Netz momentan korrekt arbeitet und bereit für Netzwerkkommunikation ist.

talk: schriftliche Rede- und Gegenrede zwischen Benutzerinnen auf verschiedenen Rechnern im lokalen Netz oder weltweit (auch: *ytalk*).

telnet: Einloggen auf anderen Rechnern (auch: *rlogin*).

xarchie: Weltweite Suche nach interessierenden, frei zugänglichen Files.

xbiff: Anmeldung von eingegangenen Emails.

xhost: Zugriff auf die Access Control List eines X–Servers.