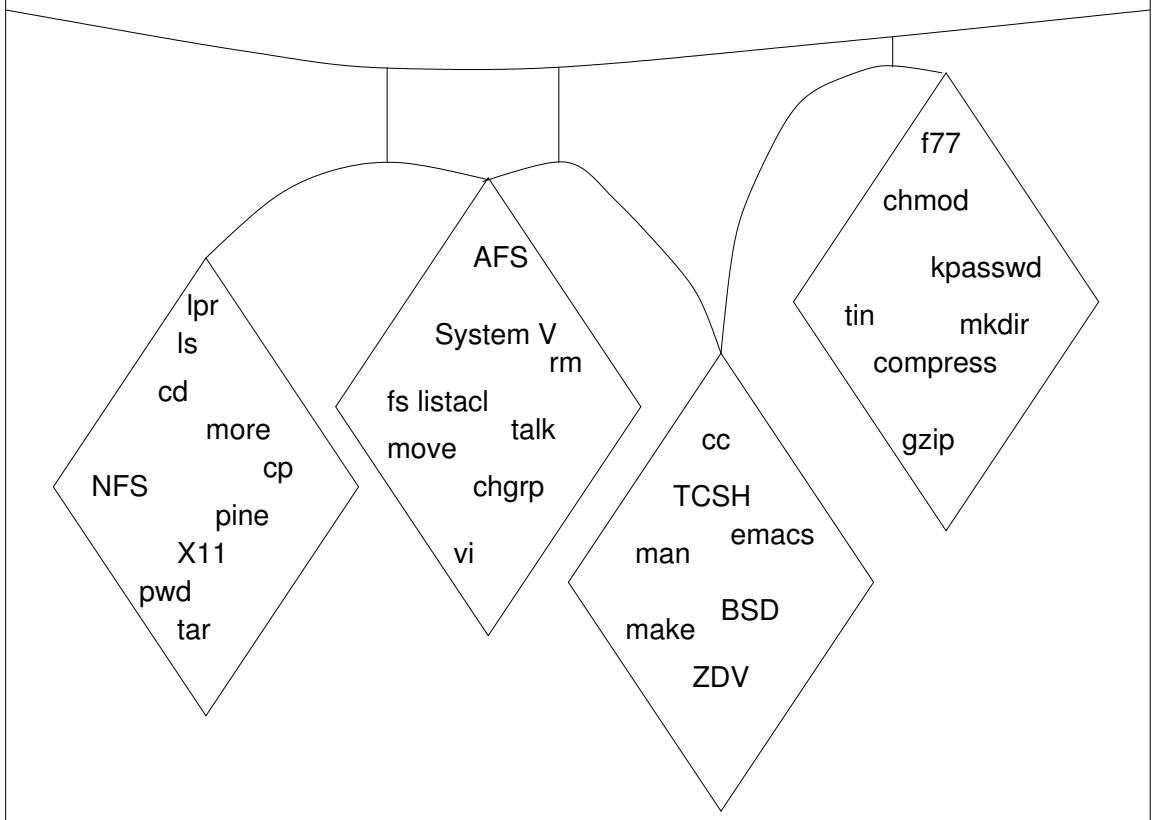
Einführung in

UNIX und Linux



Ein-/Zweitägiger Kurs am Zentrum für Datenverarbeitung (ZDV), Tübingen

Inhaltsverzeichnis

1 Vorbemerkungen

1.1

Das ZDV

Das Zentrum für Datenverarbeitung (ZDV) ist das zentrale Rechenzentrum der Universität Tübingen.

Seine Aufgabe als Dienstleistungseinrichtung besteht darin, das Kommunikationsnetzwerk der Universität und zentrale Datenverarbeitungssysteme, das sind beispielsweise Hochleistungsrechner und Peripheriegeräte wie Laserdrucker, Scanner, etc., für die Universität zu betreiben.

Unter anderem bietet das ZDV für die Studenten zwei Workstationpools an, die ganz oder teilweise unter **UNIX** bzw. **Linux** laufen:

1. Einen Computer-Pool in der Wilhelmstraße 106. Dieser besteht aus Windows NT- und Linux-Workstations (Rechnernamen: linux25 - linux45).
2. Einen Linux-Pool auf der Morgenstelle (Gebäude C, Ebene 2, Raum F07, Rechnernamen: linux01 - linux21).

Neben diesen WS-Pools stehen den Institutsangehörigen die zentralen Server zur Verfügung:

- Text-, Graphik-, Compute-, Kommunikations- und Print-Server
- Mailserver, WWW-Server, Newsserver

In dem vorliegenden UNIX-Kurs wird auf die Besonderheiten am ZDV eingegangen.

Literaturverzeichnis

Einführende Werke:

- „Standard-Betriebssystem UNIX“, *Hans-Josef Heck*, Rowohlt TB: 1990, ISBN 3-499-18167-3, 278 Seiten, **sehr günstig**.
- „Standard-Betriebssystem UNIX für Fortgeschrittene“, *Hans-Josef Heck*, Rowohlt TB: 1991, ISBN 3-499-18187-8, ca. 300 Seiten.
- „UNIX System V.4 für Einsteiger und Fortgeschrittene. Bourne-Shell, Korn-Shell, C-Shell, TCP/IP, UUCP, E-Mail & News, X Windows System“, *Stefan Stapelberg*, Bonn: Addison-Wesley 1993, 864 Seiten, ISBN 3-89319-433-9, UB edv N 3108.
- „UNIX – System V.4. Begriffe, Konzepte, Kommandos, Schnittstellen“, *J. Gulbins, K. Obermayr*, Berlin: Springer 1995, 840 Seiten, ISBN 3-540-58864-7, UB inf N 3040.
- „Unix in a Nutshell“, *Daniel Gilly*, O'Reilly & Associates 1995, ISBN 3-930673-14-2.

Weiterführende Literatur:

- „UNIX Power Tools“, *Jerry Peek, Tim O'Reilly und Mike Loukides*, O'Reilly & Associates, 1120 Seiten, ISBN 3-446-15798-0.
- „X Window System und OSF/Motif. Eine schrittweise systematische Einführung mit zahlreichen praxisnahen Beispielen“, *Uwe Wittig*, IWT/VVA 1992, ISBN 3-88322-412-X.

Sonstiges:

- Verschiedene UNIX-Kurse, WWW-pages der Fachschaft Informatik, unter *http://www-ti.informatik.uni-tuebingen.de/~fsi/graf.html*.

Die Autoren

Die Autoren des vorliegenden UNIX-/Linux-Kurses waren und sind bemüht, die Unterlagen auf dem neuesten Stand zu halten. Dennoch können immer wieder Druck- und Schreibfehler übersehen werden. Aus diesem Grund ergeht die Bitte und der Wunsch an die Leserinnen und Leser, Bemerkungen, Verbesserungsvorschläge und Kritiken an die Autoren zu richten.
Die Autoren (in alphabetischer Reihenfolge):

W. Dilling, M. Goller, D. Hänle, U. Hahn, A. Keck und L. Servissoglou

Wir sind per E-Mail erreichbar: *markus.goller@zdv.uni-tuebingen.de*

Beratung am ZDV

Da ein Kurs niemals alle Fragen „präventiv“ beantworten kann, die beim Arbeiten unter UNIX auftreten, bietet das ZDV mehrere Anlaufstellen bei Problemen:

- Die **Allgemeine Beratung**, Besuchszeit Montag bis Freitag täglich von 10.30-12.00 Uhr, im Raum 104 in der Wilhelmstraße 106, E-Mail: *beratung@zdv.uni-tuebingen.de*, Tel. 29-70250 (bevorzugt Montag bis Freitag von 9-10.30 Uhr).
- Die **Pool-Beratung Wilhelmstraße**: Dienstag bis Freitag 12-17 Uhr, im Durchgang zum Poolraum (R. 010), Tel. 29-70230.
- Das **Informationssystem des ZDV im WWW** unter der Adresse *http://www.uni-tuebingen.de/zdv/*, mit Informationen über die Beratung am ZDV, Kursankündigungen, Antworten zu häufig gestellten Fragen (FAQ) und vielem mehr.

Eine aktuelle Version dieses Skriptes findet sich im Informationssystem des ZDV unter der Adresse
http://www.uni-tuebingen.de/zdv/schriften/

2 Einleitung

2.1

Allgemeines zu Betriebssystemen

In der EDV werden generell **Hardware** und **Software** unterschieden.

- Zur Hardware gehört das „Innenleben" des Computers wie Prozessor, Arbeitsspeicher, Festplatte, Graphikkarte usw. ebenso wie die externen Komponenten, etwa Bildschirm, Tastatur und Diskettenlaufwerk(e).
- Software ist die Bezeichnung für Programme, die auf Computern laufen.

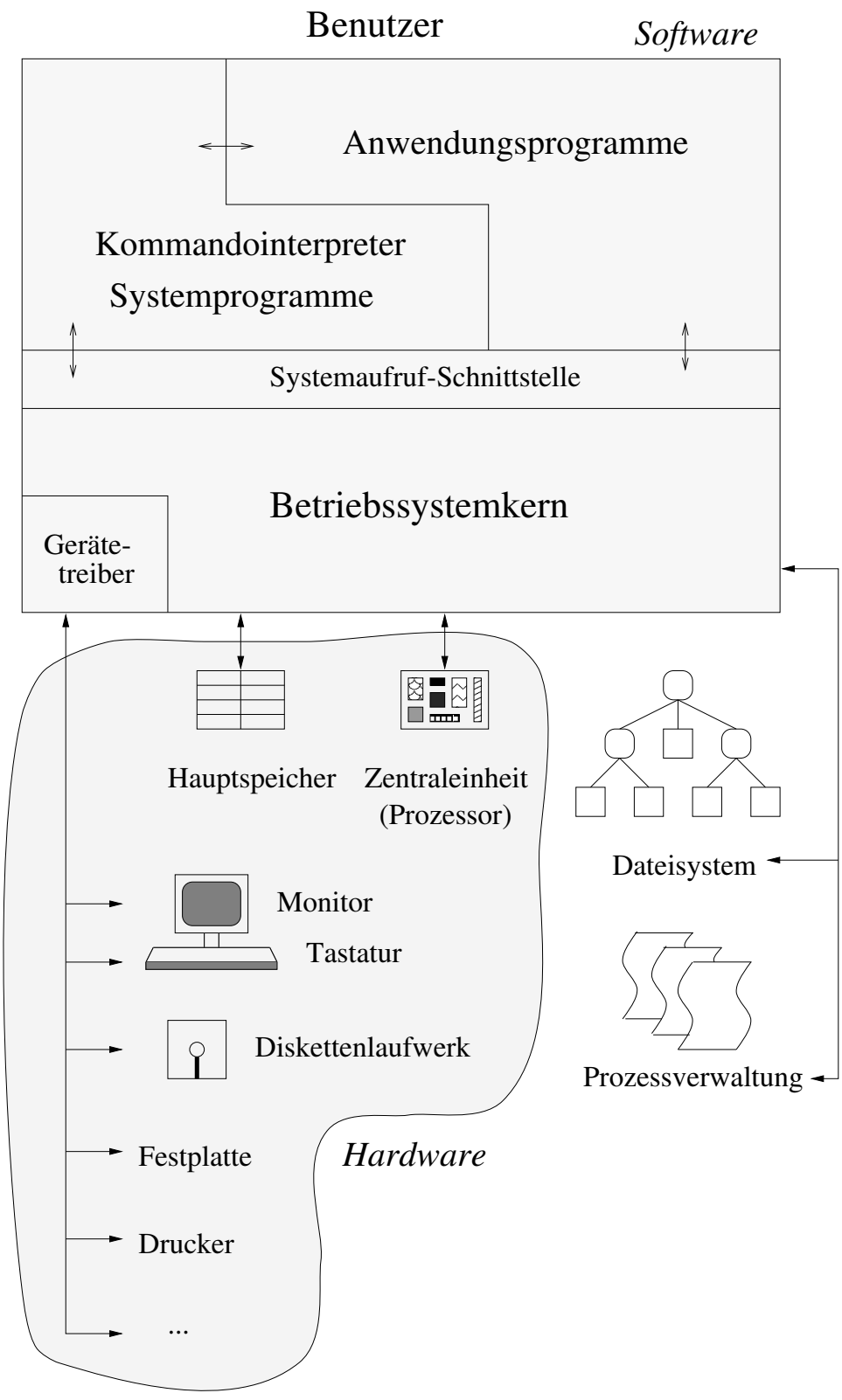
Dre wichtigste Teil der Software ist das **Betriebssystem**, das Kontrollprogramm des Computers.

Das Betriebsystem

- wird beim Einschalten des Rechners automatisch geladen und gestartet,
- verwaltet Speicher, Festplatte, Laufwerke, Monitor, Drucker etc.,
- und stellt **Befehle** zur Verfügung, die es dem Benutzer überhaupt erst ermöglichen, mit dem Rechner zu arbeiten.

Bekannte Betriebssysteme sind MS-DOS, Windows 95/98 und Windows NT von Microsoft, OS/2 von IBM und eben – **UNIX** und **Linux**.

Die Schnittstellen des UNIX-Systems



Allgemeines zu UNIX

- UNIX ist ein **Multitasking-Betriebssystem**, d.h. es können auf einem Rechner mehrere Programme, auch verschiedener Benutzer, zur gleichen Zeit laufen.
- Man kann unter UNIX entweder am Rechner selbst arbeiten (**Konsole**) oder an einem Terminal.
- UNIX ist ein **Multiuser-Betriebssystem**, d.h. UNIX unterscheidet verschiedene Benutzer. Jeder Benutzer hat dabei seine eigene Arbeitsumgebung und eigene Daten.
- Zur Identifikation erhält jeder Benutzer
  - einen **Login-Namen** (auch **User-ID** genannt), z.B. *zrago01*
  - und ein Paßwort, das man selbst ändern kann (und soll - dazu mehr in 3.2).

  Diese Zugangsberechtigung nennt man auch **Account**.
- Der Systemverwalter (login-Name **root**) weist jeden Benutzer einer Gruppe zu, z.B. zur Gruppe **unixkurs** oder zur Gruppe der Studenten **zx**.
- Ein UNIX- oder Linux-Rechner (Workstation) sollte (im Unterschied zu einem „stand-alone"-PC) vom Benutzer nicht ausgeschaltet werden !
- Ein modernes Betriebssystem enthält gewöhnlich eine **graphische Benutzeroberfläche**, z.B. WINDOWS unter MS-DOS oder FINDER auf dem Macintosh. Unter UNIX ist dies **X11**, unter Linux die freie X11-Portierung **XFree86**.

UNIX und Linux I

Zur Geschichte:

- Die Entwicklung von **UNIX** begann Ende der 60er-Jahre in den Bell Laboratories von AT&T aus dem Wunsch heraus, ein hardwarenahes, leistungsfähiges und netzwerkfähiges Betriebssystem für Größrechner und Workstations zu haben. Große Teile sind in der eigens dazu entwickelten Programmiersprache *C* geschrieben.
- Da sich UNIX als effizient und stabil erwiesen hat, setzte es sich insbesondere im Großrechnerbereich und vernetzten Systemen durch. Daher entwickelten zahlreiche Firmen kommerzielle Varianten von UNIX.
  Mittlerweile gibt es eine Reihe von Standards (POSIX, ANSI), die den Wechsel zwischen verschiedenen UNIX-Varianten erleichtern.
- **Linux** wurde Anfang der 90er-Jahre von einem finnischen Studenten, Linus Torvalds, entwickelt, der UNIX auch privat auf dem PC nutzen wollte. Linux ist ebenfalls in C geschrieben; die Quelltexte waren jedoch von Beginn öffentlich zugänglich. Daher kann via Internet eine weltweite Entwicklergemeinde ständig zur Aktualisierung, Erweiterung und Verbesserung von Linux beitragen. Der größte Vorteil davon ist aber, das Linux grundsätzlich nicht kommerziell, also frei erhältlich ist.
- Bei Linux wurde von Anfang an darauf geachtet, daß es die Vorteile von UNIX (Multitasking- und Multiuser-Funktionalität, Netzwerkfähigkeit) beinhaltet und weitgehend zum POSIX-Standard konform ist. Das erleichtert die Portierung von Software zwischen UNIX und Linux ungemein.

UNIX und Linux II

Unix für Workstations, Linux für PC's ?

- Mit **Workstation** wird i.d.R. ein leistungsfähiger, in ein Netzwerk eingebundener Rechner im Multiuser-Betrieb bezeichnet.
- Im Gegensatz dazu war ein **PC** (= Personal Computer) ein vergleichsweise schwacher Rechner, der nur einem Benutzer zur Verfügung stand und nicht vernetzt war (also ein sog. *Stand-Alone-Rechner*).
- Durch Fortschritte in der Speicher- und Chiptechnologie kam es zu einer Leistungsexplosion bei PC's; zudem haben PC's vermehrt auch Anschluß an Netzwerke (insbesondere ans Internet).
- Die begriffliche Trennung zwischen PC und Workstation hat also zunehmend weniger Aussagekraft, denn auch ein PC (z.B. unter Linux) ist inzwischen als Workstation einsetzbar.
- Das ZDV hat daher vor kurzem im Rahmen des Umzugs die beiden UNIX-Workstation-Pools durch Linux-Pools ersetzt.
- Selbst Großrechner werden mittlerweile von Linux-PC's verdrängt. So ist beispielsweise der neue Textserver des ZDV (*textserv1*) ein PC unter Linux.
- Mit der steigenden Beliebtheit von Linux nimmt auch die Verfügbarkeit von (ebenfalls gratis erhältlicher) Software zu.
  Selbst kommerzielle Produkte werden inzwischen für Linux entwickelt; beispielsweise stehen mit *StarOffice 4.0* und *ApplixWare 4.4.1* zwei leistungsfähige Office-Pakete (mit Textverarbeitung, Tabellenkalkulation, Datenbanken etc.) für Linux zur Verfügung.

Informationen zur Installation und Konfiguration von Linux erhalten Sie in den entsprehenden Kursen des ZDV .

3 Einloggen und Ausloggen

3.1

Einloggen

Gewöhnlich findet man am Rechner oder Terminal eine Aufforderung zum Einloggen vor, die ungefähr so aussieht:

```
Login:
Password:
```

- Geben Sie den Login-Namen ein (<ENTER>)
- Geben Sie das Paßwort ein (<ENTER>)
- Achtung: UNIX unterscheidet Groß- und Kleinschreibung !

Verlassen des Systems (**Ausloggen**)

Dies ist abhängig von der Konfiguration des Rechners, geschieht aber gewöhnlich durch eine der drei Methoden:

- Eingabe von `exit`
- Eingabe von `logout`
- Anklicken eines `logout`-Menüpunkts mit der Maus
- Geschieht das Login in einer graphischen Umgebung, schließt `exit` u.U. nur das gerade aktive Fenster. In diesem Fall sollte man mit der Tastenkombination STRG+ALT+⇐ (Backspace) die graphische Benutzeroberfläche beenden.

Paßwörter

Das „initiale“ Paßwort, das bei der Einrichtung eines Accounts von der Benutzerverwaltung oder einem Systemadministrator zugeteilt wird, sollte aus Sicherheitsgründen schnellstmöglichst geändert werden. Aber auch später ist regelmäßiges Ändern des Paßworts ratsam, um einen Mißbrauch des eigenen Accounts zu erschweren. Man sollte bei der Auswahl eines Paßworts folgende Empfehlungen beachten:

- Ein Paßwort sollte aus mindestens acht Zeichen bestehen.
- Es sollte ein Paßwort gewählt werden, das in keinem Wörterbuch steht. Es sollte daher mindestens ein nicht-alphanumerisches Zeichen (d.h. ein Sonderzeichen, nicht einen Buchstaben oder eine Zahl) enthalten und sowohl aus Groß- als auch Kleinbuchstaben bestehen.
- Namen, Geburtstage, Firmennamen, Telefonnummern etc. sind leicht herauszufinden und daher ungeeignet, ebenso wie Abwandlungen der User-ID.
- Die einzelnen Zeichen des Paßworts sollten auf der Tastatur nicht direkt nebeneinanderliegen, damit das Paßwort beim Eingeben nicht einfach „abgeguckt“ werden kann.
- Beliebte Methoden sind das Ersetzen von Zeichen (etwa 's' durch '$', 'i' durch '!' etc.) oder Paßwörter aus Anfangsbuchstaben von Sätzen (z.B. *IoR&RbIIi!* - It's only Rock & Roll but I like it!).
- Man sollte sein Paßwort niemals aufschreiben oder weitersagen und vor dem Verlassen des Computers immer darauf achten, daß man ausgeloggt ist.
- Das Paßwort sollte man von Zeit zu Zeit ändern (Faustregel: einmal pro Semester).

4 Dateien, Directories und Pfade

4.1

Dateien

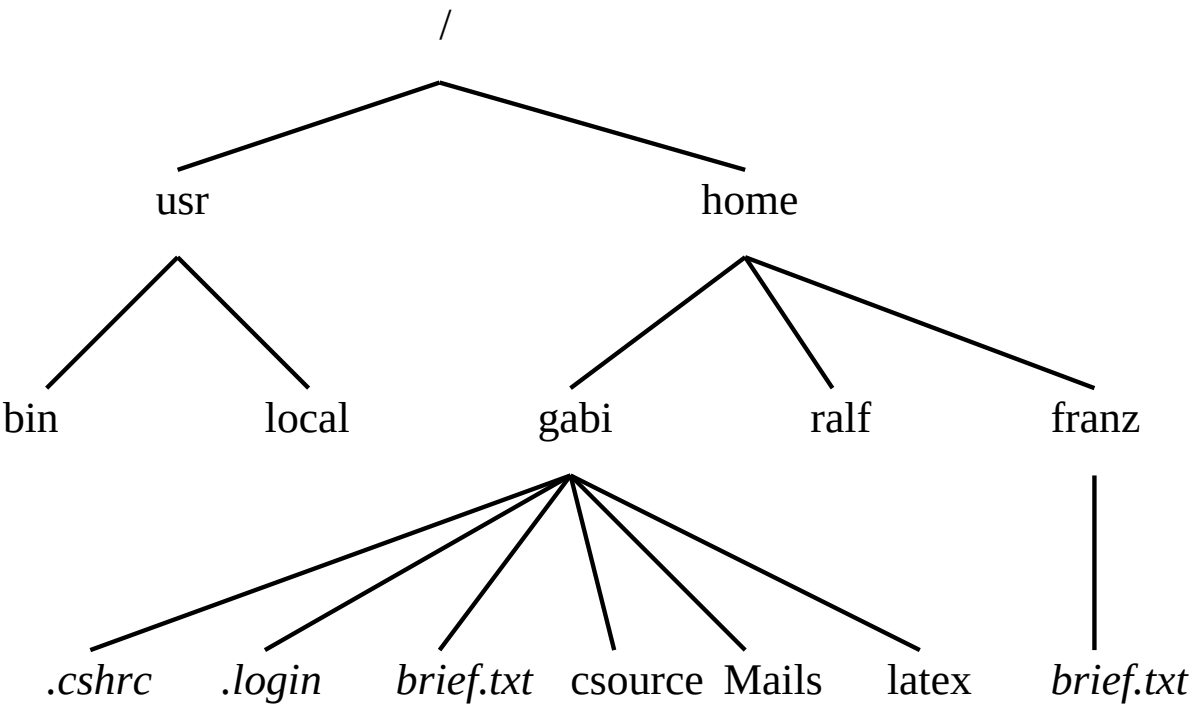
- enthalten beliebige Daten (z.B. Texte, ausführbare Programme, Liste von Adressen, usw). Auch **Files** genannt.
- werden von bestimmten Programmen (z.B. einem Editor) erzeugt.
- Mit der Datei werden auch Informationen über den **Eigentümer** der Datei abgespeichert.
- Der Eigentümer kann mithilfe von **Zugriffsrechten** bestimmen, wer auf seine Dateien zugreifen darf.
- haben einen Namen. Dabei wird zwischen Groß- und Kleinschreibung unterschieden !
- können mit sehr langen Namen versehen werden.
- sollten eine **Endung** haben, die etwas über den Inhalt der Datei aussagt (z.B. *Brief.txt* anstelle *Brief*).
- Verschiedene Programme sind auf bestimmte Endungen angewiesen.
- übliche Endungen sind

.c	Quelltexte von C-Programmen
.f	Quelltexte von Fortran-Programmen
.p	Quelltexte von Pascal-Programmen
.doc	Dokumentationsdateien
.tex	TEX- oder LATEX-Texte
.ps	Bilddateien im PostScript-Format

Directories (Verzeichnisse)

- Mehrere Dateien werden in einem **Directory** (auch **Verzeichnis** genannt) zusammengefaßt.
- Ein Directory kann weitere **Subdirectories** (bzw. **Unterverzeichnisse**) enthalten und ist selbst in einem **Parent Directory** enthalten.
- Directories „halten Ordnung“ und kennzeichnen logische Zusammenhänge (z.B. alle Briefe im Verzeichnis *Briefe*).
- Alle Directories und Dateien bilden (grafisch veranschaulicht) einen Baum, der **Directory-Baum** genannt wird.
- Das unterste Verzeichnis wird **Root Directory** genannt und mit „/“ bezeichnet.
- Jedem Benutzer wird ein **Homedirectory** (oder **Arbeitsverzeichnis**) zugewiesen, in dem er Dateien und Directories erzeugen kann.
- Im Homedirectory dürfen Benutzer meist ein oberes Speicherplatz-Limit, **Quota** genannt, nicht überschreiten (in den Linux-Pools derzeit 20 MB).

Beispiel eines Directory-Baumes



Pfade

- **Pfade** oder **Pfadnamen** geben an, wo eine Datei oder ein Verzeichnis zu finden ist.
- Man folgt den „Ästen“ des Directorybaumes und trennt Dateien und Directories durch das Zeichen „**/**“ (**slash**).

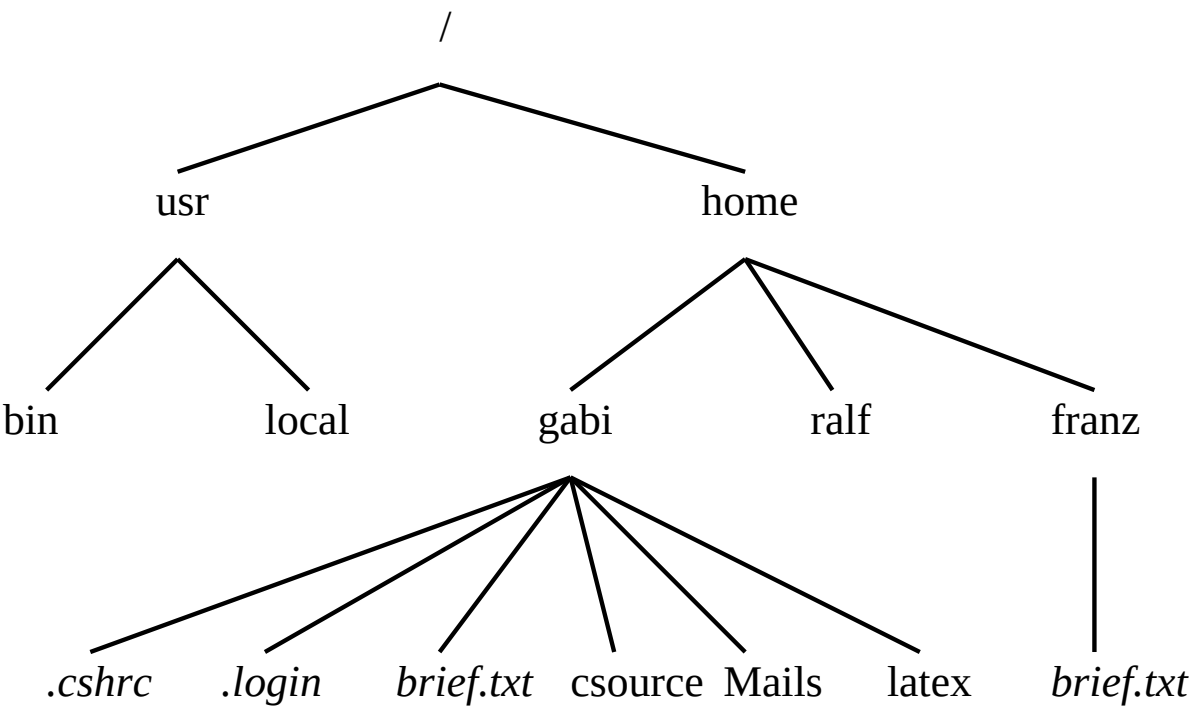
Beispiele:

- Im obigen Baum wird Gabis Homedirectory mit */home/gabi* bezeichnet.
- In Gabis Homedirectory wird die Datei *brief.txt* mit */home/gabi/brief.txt* bezeichnet. Der Brief von Franz ist */home/franz/brief.txt*.

Klettern im Directory-Baum

- Man „befindet“ sich stets an einer Stelle (in einem Verzeichnis) des Baumes, dem **aktuellen Verzeichnis** bzw. **current working directory (cwd)**.
- Mit dem Befehl `ls -al` läßt sich herausfinden, welche Dateien und Directories es im cwd gibt. Der Benutzer sieht also immer nur einen kleinen Ausschnitt des Directorybaumes.
- Nach dem Einloggen befindet man sich in seinem Homedirectory.
- „`.`“ ist ein Verweis ins cwd
  „`..`“ ist ein Verweis ins parent-Verzeichnis von cwd
  „`~`“ ist eine Abkürzung der Shell für das Homedirectory
  „`~`benutzer“ ist die Abkürzung für das Homedirectory von benutzer
- Man kann das aktuelle Verzeichnis mit dem Befehl `cd` *Pfad* (change directory) wechseln („vor Ort arbeiten“ erspart Tipparbeit für lange Pfadnamen !)
- Der Befehl `pwd` (print working directory) zeigt das Directory an, in dem man sich gerade befindet.

Beispiel



- Franz loggt sich ein und befindet sich in */home/franz*.
- Nach Eingabe des Befehls `cd ..` befindet er sich in */home*.
- Nach `cd` *gabi/latex* ist das aktuelle Verzeichnis */home/gabi/latex*.
- `cd` *../gabi/latex* oder `cd ~`*gabi/latex* wäre kürzer gewesen.
- Nach Eingabe von `cd` *../../../usr/local* ist das cwd das Verzeichnis */usr/local*.
  Äquivalent dazu ist `cd` */usr/local*.
- `cd /` wechselt ins Root Directory.
- Hat man sich „verirrt“, so führt die Eingabe von `cd` (ohne Argument) zurück ins Homedirectory.

Das verteilte Dateisystem AFS

- **Verteilt** bedeutet, daß ein(e) BenutzerIn von einem Rechner aus auf den Dateibaum (oder einen Teil davon) eines anderen Rechners zugreifen kann.
- Ein verteiltes Filesystem hat den Vorteil, daß ein Benutzer immer dasselbe Homedirectory vorfindet, unabhängig davon, an welcher Pool-Workstation er sich einloggt.
- Ein solches verteiltes Filesystem ist das **AFS** (Andrew File System).
- **AFS** ist ein „weltweiter" Dateibaum, der aus Teilbäumen einzelner Universitäten, Firmen und anderer Institutionen aufgebaut ist.
- Das Rootdirectory dieses Baumes ist */afs*.
- Universitäten und Firmen hängen ihren eigenen Teilbaum unterhalb von */afs* ein. So entstehen Verzeichnisse wie z.B. */afs/zdv-pool.uni-tuebingen.de*.
- Ist ein Rechner **im AFS**, kann er auf das Verzeichnis */afs* und somit auch auf dessen Unterverzeichnisse zugreifen. Der Systemadministrator legt fest, welche Unterverzeichnisse von */afs* „sichtbar" sind.
- Auch die Homedirectories der BenutzerInnen am ZDV liegen im **AFS**, z.B. */afs /zdvpool.uni-tuebingen.de /home/zrakr01*.
- Für das Ändern des Paßworts oder das Setzen von Zugriffsrechten gelten **NICHT** die Standard UNIX-Befehle (`passwd`, `chmod`), sondern die AFS-spezifischen Befehle (`kpasswd`, `fs setacl`).

Mehr Information zum AFS und den zugehörigen Befehlen finden Sie im WWW-Informationssystem des ZDV unter
http://www.uni-tuebingen.de/zdv/zrsinfo/pools-n/afs-pool-zdv.html.

5 Befehle, Optionen und Argumente

5.1

Eingabe von Befehlen

- Befehle bestehen aus **Kommandonamen**, **Optionen** und **Argumenten**, die, jeweils durch Leerzeichen getrennt, an den Befehl angehängt werden. Beispiel:

  `ls -alg` *`/home/franz`*

- **Optionen** sind Anweisungen an den Befehl, dessen Arbeitsweise gegenüber der Voreinstellung zu ändern. Optionen werden meist mit „–“ eingeleitet.
- Eine Option besteht gewöhnlich aus einem Buchstaben. Mehrere Optionen können aneinandergereiht werden, wie in obigem Beispiel die drei Optionen **a**, **l** und **g**.
- Jeder Befehl hat seine eigenen Optionen, die man „nachschlagen“ kann (siehe Manual Pages).
- **Argumente** sind Zeichenketten, die vom Befehl interpretiert werden. Meist geben sie Dateien an, mit denen etwas gemacht werden soll.
- Befehle werden eingegeben, wenn die Aufforderung zur Eingabe (**Prompt**) erscheint. Auf den Rechnern des ZDV zeigt der Prompt den Rechnernamen und das aktuelle Verzeichnis an, z.B.

  `linux40:~/unixkurs>`

- Durch Drücken der Taste <ENTER> wird der Befehl ausgeführt.

Optionen und Argumente - Ein Beispiel

Das Kommando `ls -al` */home/gabi* erzeugt folgende Ausgabe:

```
drwxr-x---  gabi  unixkurs   512   Oct 19 16:47  .
drwxr-xr-x  root  wheel      512   Oct 19 16:37  ..
-r--r-----  gabi  unixkurs   836   Jan 23 15:37  .cshrc
-r--r-----  gabi  unixkurs   394   Jan 24 12:14  .login
-rw-r-----  gabi  unixkurs  1137   Feb 28 09:28  brief.txt
drwxr-xr-x  gabi  unixkurs   512   Apr 11 13:08  csource
drwxr-----  gabi  unixkurs   512   Apr 17 10:07  Mails
drwxr-xr-x  gabi  unixkurs  1024   Mar 12 13:44  latex
```

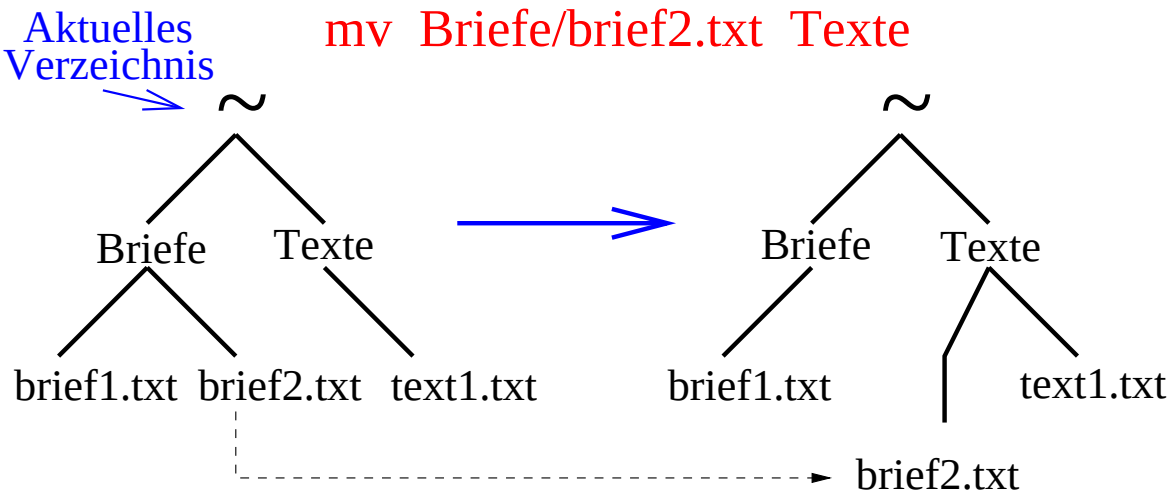
- Das erste Zeichen ist ein Typbezeichner
  - – für Dateien
  - d für Directories
- Es folgen je 3 Zeichen, die die **Zugriffsrechte** des Besitzers, seiner Gruppe und der anderen Benutzer angeben. Mehr hierzu erfahren Sie im Kapitel *Zugriffsrechte*.
- Schließlich in je einer Spalte
  - – Name des Besitzers (z.B. gabi)
  - – Gruppe des Besitzers (z.B. unixkurs)
  - – Größe der Datei
  - – Datum und Zeit der letzten Änderung
  - – Name der Datei

Die wichtigsten Befehle

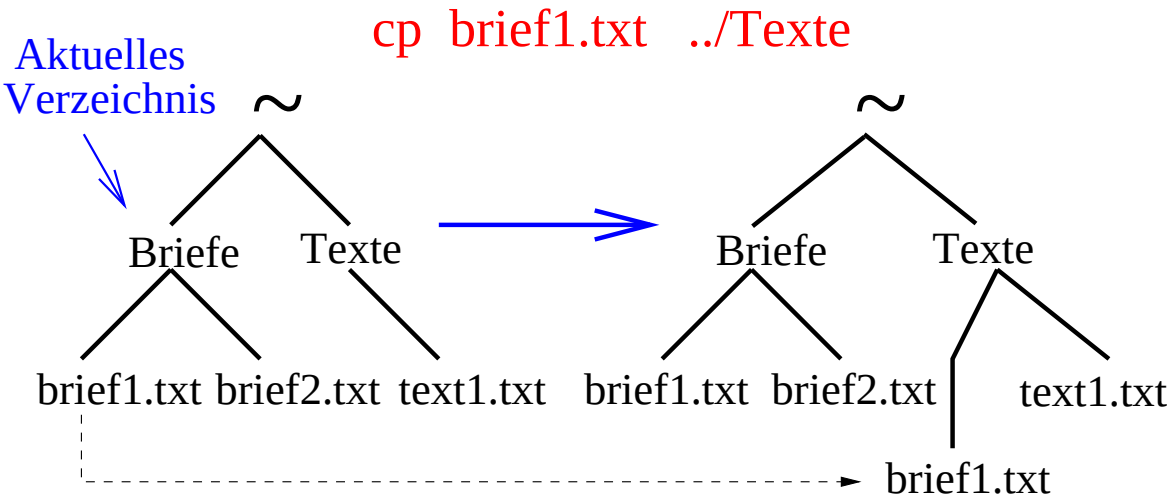
`cd` *directory*	wechseln in das Verzeichnis *directory*
`ls` *directory*	listet den Inhalt des Verzeichnisses *directory* auf
`mkdir` *directory*	erzeugt ein neues Verzeichnis mit dem Namen *directory*
`mv` *alt directory*	verschiebt die Datei *alt* in das Verzeichnis *directory*
`mv` *alt neu*	benennt die Datei *alt* in *neu* um
`cp` *alt neu*	erzeugt eine Kopie von *alt* mit dem Namen *neu*
`rm` *Datei*	löscht die Datei mit dem Namen *Datei*
`rmdir` *directory*	löscht das Verzeichnis *directory* (sofern leer)
`pwd`	gibt das aktuelle Verzeichnis (**cwd**) aus
`more` *text*	gibt eine Textdatei auf dem Bildschirm aus und hält dabei nach jeder Bildschirmseite an
`passwd`	Ändern des Paßworts
`kpasswd`	Ändern des Paßworts unter AFS
`chmod`	Ändern der Zugriffsrechte einer Datei
`chown`	Ändern des Eigentümers einer Datei
`startx`	Startet die grafische Benutzeroberfläche X11 (falls diese noch nicht läuft).

Beispiele für `mv` und `cp`

Die Datei *brief2.txt* soll ins Verzeichnis *Texte* **verlegt** werden:

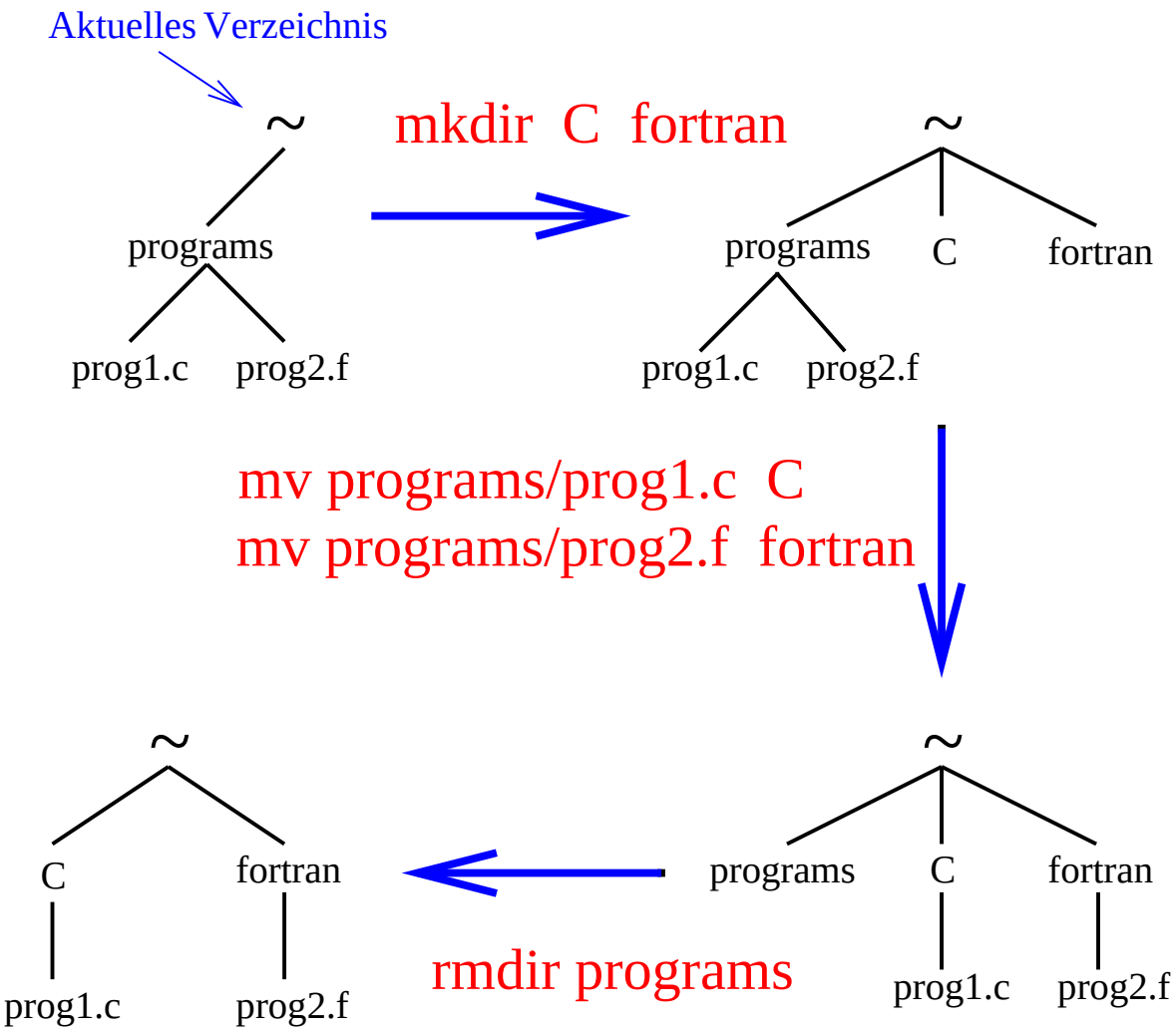


Die Datei *brief1.txt* soll ins Verzeichnis *Texte* **kopiert** werden:



Beispiel: Aufräumen im Homedirectory

Im Verzeichnis *programs* liegen sowohl C- als auch Fortran-Programme. Diese sollen in eigene Verzeichnisse kopiert werden:



Druckkommandos

Wird auf einem Drucker ausgedruckt, so wird der **Druckjob** in eine Warteschlange eingereiht.

`lpr` *text*	gibt eine Text- oder PostScript-Datei (!!!) auf einem Drucker aus
`lpq`	gibt die Warteschlange eines Druckers aus (einschließlich der JobNr)
`lprm` *JobNr*	entfernt den Ausdruck mit Nummer *JobNr* aus einer Warteschlange

- Bei jedem dieser Kommandos muß man mit der Option *–P* angeben, auf welchen Drucker sich das Kommando bezieht, z.B.

```
lpr -Pps007.d4 Brief.txt
lpq -Pps007.d4
lprm -Pps007.d4 23
```

Derzeit stehen für Studenten die folgenden Drucker zur Verfügung:

Auf der Morgenstelle (C2):

ps001 / ps001_d4	einseitig / doppelseitig	300 dpi
ps032	einseitig	600 dpi

Im ZDV, Wilhelmstraße 106 (UG), Computer-Pool:

ps007 / ps007.d4	einseitig / doppelseitig	1200 dpi
ps009	einseitig	1200 dpi
ps014	einseitig	1200 dpi
ps033 / ps033_fl	Papier- / Foliendruck	300 dpi (farbig)

- Auf manchen UNIX-Systemen (**System V**) heißen die Befehle `lp`, `lpstat` und `cancel`. Der Option *–P* entspricht dort *–d* .

Mehr Informationen zu den ZDV-Druckern finden Sie unter
http://www.uni-tuebingen.de/zdv/zrsinfo/zdv-peripherie/drucker/drucker-in-pools.html.

Das Kommando `who`

- Mit `who` läßt sich herausfinden, wer auf dem Rechner, auf dem man gerade arbeitet, eingeloggt ist.
- Mit ausgegeben werden
  - das logische Terminal (Fenster) (Spalte 2).
  - Datum und Uhrzeit des Einloggens (Spalte 3,4,5).
  - der Rechner bzw. das Terminal, von dem aus die Verbindung hergestellt wurde (Spalte 6).

```
zrssg01  pts/0  Apr 19 11:02  (hamster.zdv.uni-)
tgifi01  pts/1  May 22 09:55  (ge.gsmed.medizin)
tgihe01  pts/2  May 22 08:12  (tgi-gw.zdv.uni-t)
zrncl01  pts/3  May 10 15:28  (ambixpc2.zdv)
zrssg01  pts/4  Apr 04 08:04  (hamster)
```

Das Kommando `finger`

- Die Eingabe von `finger` liefert eine ähnliche Ausgabe wie `who`.
- Mit `finger LoginId` oder `finger Vor-/Nachname` lassen sich Informationen über andere Accountbesitzer herausfinden.
- Mit `finger @Rechnername` läßt sich auch herausfinden, wer auf einem anderen Rechner eingeloggt ist.
  Dieses ist jedoch häufig durch den Systemadministrator „abgeklemmt“.

```
textserv:~> finger @sunap3
[sunap3.zdv.uni-tuebingen.de]
Login       Name      TTY Idle   When  Where
zrakr01 Uwe Kreppel   p3  3d Wed 10:41 :0.0
zraun01 Koaunghi Un   p4  9d Fri 08:55 134.2.1.15:0.0
zrakr01 Uwe Kreppel   p5  3d Wed 12:29 sunap3.zdv.uni-t
zrakr01 Uwe Kreppel   p7   5 Mon 10:19 :0.0
zrahd01 Daniel Haenle p8     Mon 10:54 moraix.zdv.uni-t
```

6 Zugriffsrechte

6.1

Die „traditionellen“ Zugriffsrechte

Das UNIX-System unterscheidet drei Arten von Benutzern:

- den Besitzer der Datei (bezeichnet mit **user**)
- die Gruppe, in die man eingeteilt wurde (z.B. Gruppe „Unixkurs“) (**group**)
- alle anderen Benutzer mit einem Account auf dem Rechner (**others**)

Die Zugriffsrechte

- erlauben oder verbieten den Zugriff (Lesen, Schreiben und Ausführen) auf Dateien und Directories. Auch **protections** genannt.
- kann jeder Benutzer nur für seine eigenen Dateien vergeben (Ausnahme: der Systemadministrator).
- können gezielt für **user**, **group** und **others** vergeben werden.
- werden mit dem Befehl `chmod` geändert.
- werden mit dem Befehl `ls -al` angezeigt:

```
-|rw-|r--|---  gabi unixkurs 1137 Feb 28 09:28  brief.txt
d|rwx|r-x|r-x  gabi unixkurs  512 Apr 11 13:08  csource
T| U | G | O
```

Bei dieser Ausgabe kann man die Zugriffsrechte an den zehn Spalten des linken Blocks ablesen:
Die erste Spalte (T) gibt den Typ (Verzeichnis oder Datei) an. In der nächsten Dreiergruppe (U) stehen die Rechte des Eigentümers (user). Die mittlere Dreiergruppe (G) enthält die Rechte der Gruppe (hier: unixkurs). Die letzte Dreiergruppe (O) gibt die Rechte aller anderen Benutzer (others) an.

Benutzerrechte von Dateien und Directories

Mögliche Benutzerrechte bei Dateien:

- r Read: Erlaubt ist das Lesen der Datei
- w Write: Erlaubt sind Ändern oder Löschen
- x Execute: Erlaubt ist das Ausführen eines Programmes

Mögliche Benutzerrechte bei Directories:

- r Read: Die Liste der Dateien und Subdirectories darf mit `ls` gelesen werden
- w Write: Dateien in diesem Directory dürfen erzeugt und gelöscht werden
- x Search: Das Directory darf mit dem Befehl `cd` „betreten“ werden

Beispiel: Ausgabe des Kommandos `ls -al`

```
drwxr-x---  gabi  unixkurs   512   Oct 19 16:47  .
drwxr-xr-x  root  wheel      512   Oct 19 16:37  ..
-r--r-----  gabi  unixkurs   836   Jan 23 15:37  .cshrc
-r--r-----  gabi  unixkurs   394   Jan 24 12:14  .login
-rw-r-----  gabi  unixkurs  1137   Feb 28 09:28  brief.txt
drwxr-xr-x  gabi  unixkurs   512   Apr 11 13:08  csource
drwxr-----  gabi  unixkurs   512   Apr 17 10:07  Mails
drwxr-xr-x  gabi  unixkurs  1024   Mar 12 13:44  latex
```

- Vergeben Sie **nie** das Schreibrecht für Ihr Homedirectory an andere Benutzer !!!

Wie ändere ich nun Zugriffsrechte ?

`chmod <wer><darf><was>` *Datei*

(Keine Leerzeichen zwischen `<wer>`, `<darf>` und `<was>`, jeweils ein Leerzeichen nach `chmod` und vor *Datei* !)

`<wer>` steht für

- **u** (user) für sich selbst
- **g** (group) für die Gruppe
- **o** (others) für alle anderen
- **a** (all) für alle (entspricht `ugo`)

`<darf>` steht für eines der Zeichen

- \+ ein Recht wird (zusätzlich) vergeben („darf“)
- − ein Recht wird entzogen („darf nicht“)
- = genau die angegebenen Rechte werden vergeben („darf nur“)

`<was>` steht für eine Kombination von `r`, `w` und `x`.

Beispiele:

- `chmod o+rw brief.txt`
  „others dürfen *brief.txt* lesen und schreiben“
- `chmod go-rwx ~`
  „group und others dürfen in meinem Homedirectory weder lesen, schreiben noch dürfen sie es mit `cd` betreten“
- `chmod a=r Mails`
  „Alle (ugo) dürfen jetzt nur noch lesen“

Zugriffsrechte unter AFS

- Manche Verzeichnisse (z.B. die Homedirectories der ZDV - Pools) sind Teil des großen **AFS**-Dateibaums und können nur mit speziellen AFS-Befehlen geschützt werden (`chmod` funktioniert also nicht!)
- Unter AFS können nur Directories geschützt werden; die darin liegenden Dateien erhalten die gleichen Zugriffsrechte.
- Man kann auch einzelnen Benutzern Rechte geben/nehmen.
- Rechte sind entweder **normal** (man hat das Recht) oder **negativ** (man hat ein Recht nicht).
- Zugriffsrechte werden mit dem Befehl `fs setacl` gesetzt und mit `fs listacl` angezeigt.

Die wichtigsten Rechte unter AFS:

r	(read) Dateien im Directory dürfen gelesen werden
w	(write) Dateien im Directory dürfen geändert werden
l	(lookup) man kann mit `ls` den Inhalt des Directories ausgeben
i	(insert) Dateien und Subdirectories dürfen erzeugt werden
d	(delete) Dateien und Subdirectories dürfen gelöscht werden

Es gibt vorformulierte, sinnvolle Kombinationen von Rechten:

read	für **rl**
write	für **rlidw**
all	alle Rechte (Vorsicht!)
none	löscht alle Zugriffsrechte

Genauere Informationen zu den AFS-Zugriffsrechten finden Sie unter
http://www.uni-tuebingen.de/zdv/zrsinfo/pools-n/afsdoc/zugriff.html

7 Editoren

7.1

Allgemeines

Wann wird ein Texteditor benötigt? Einige Beispiele:

- Schreiben von elektronischer Post (**Emails**)
- Erstellen von Programmen
- Gestaltung der eigenen Benutzerumgebung
- Schreiben von längeren Dokumenten (Diplomarbeit)

vi:

+ Gehört zur Standardausstattung von UNIX und steht somit auf jedem UNIX-Rechner zur Verfügung
+ Keine „Fingerakrobatik“ (gleichzeitiges Drücken mehrerer Tasten ist nicht notwendig)
+ Schneller und „kompakter“ Editor
– Die Handhabung ist gewöhnungsbedürftig, da drei verschiedene Betriebsmodi existieren.
– Da keine grafische Benutzerführung vorhanden ist, ist die Bedienung nicht intuitiv.

Emacs:

+ Grafische Benutzerführung (Fenster, Menüs usw.)
+ Tutorial, Online-Dokumentation
+ Ist dank eines Lisp-Interpreters universell einsetzbar.
+ besitzt verschiedene Modi (C-Mode, TEX-Mode usw.)
+ Die Funktionalität ist erweiter- und konfigurierbar.
– Verbraucht viel Speicher.
– Die Tastenkombinationen sind teilweise schwer zu merken (und einzugeben).

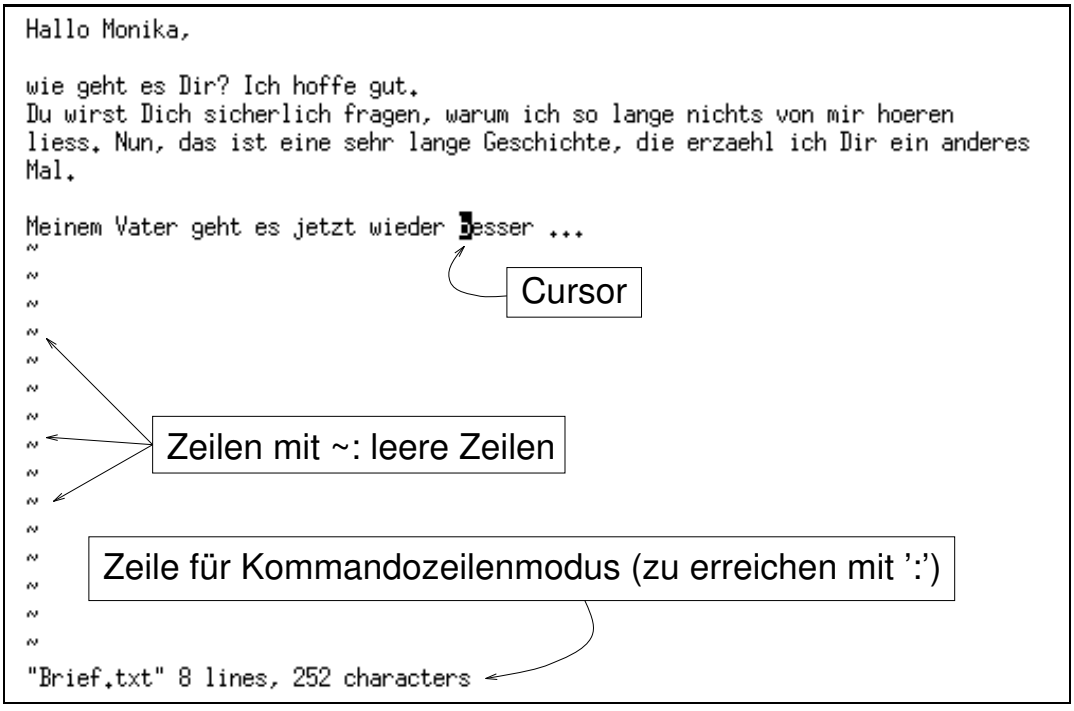
Start des vi

`vi`

Der Editor verlangt am Ende einer Sitzung entweder die Eingabe eines Filenamens, oder der erstellte Text wird verworfen.

`vi` *filename*

- Wenn ein File mit dem Namen *filename* schon existiert, wird dieses File auf dem Bildschirm angezeigt. Das File kann **editiert** werden.
- Existiert kein gleichnamiges File, wird am Ende der Sitzung der eingegebene Text in einem neuen File unter dem Namen *filename* gespeichert.



vi besitzt drei Arbeitsmodi:

- Kommando- / Einfüge- / Kommandozeilenmodus

Kommandomodus

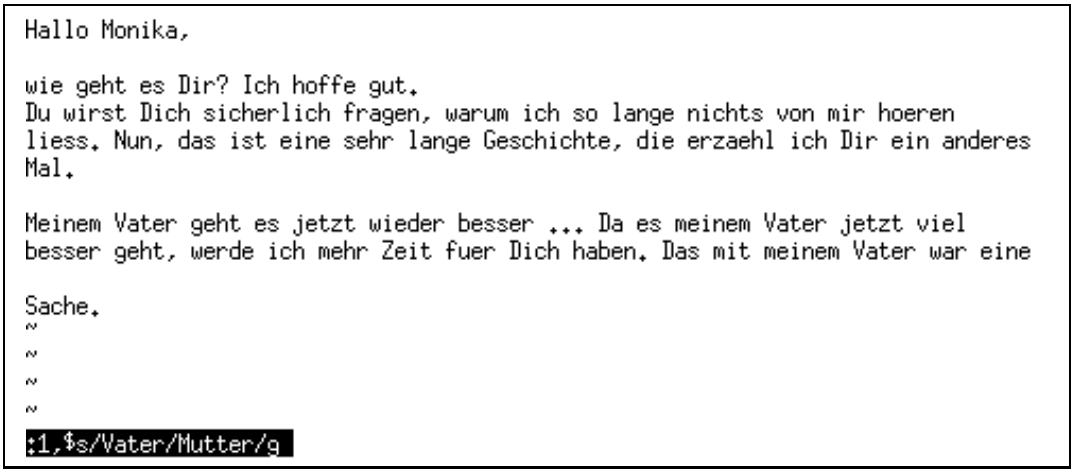
- Aktivierung des Kommandomodus durch die [ESC]-Taste
- Nur in diesem Modus bewegen die Pfeiltasten den Cursor
- Belegung der Tasten mit einem Befehl:
  z.B.: <r> = Replace (Ersetzen eines Buchstabens)
- Eine Zahl (n) vor dem Tastenkommando, wiederholt diesen Befehl n-Mal

Taste	Erkärung
⇐⇑⇒⇓	Tasten, die den Cursor bewegen
.	Wiederholung des letzten Kommandos (Tastenbefehl)
dd	Löschen der Zeile, in der der Cursors steht
dw	Löschen des Wortes, auf dem der Cursor steht
i	Ab der Stelle, wo der Cursor steht, kann Text eingegeben werden
I	Text ab Anfang der Zeile eingeben
o	Text unter der aktuellen Zeile eingeben
O	Text über der aktuellen Zeile eingeben
r	Ersetzt das Zeichen unter dem Cursor
R	Ersetzt ab dem Zeichen auf dem der Cursor steht
u	**undo**: letztes Kommando wird zurückgesetzt
w	**word**: Bewegen des Cursors um ein Wort nach rechts
x	Das Zeichen unter dem Cursor wird gelöscht
ZZ	Text sichern und **vi** verlassen

Einfügemodus

- Aktivierung durch eine der Tasten `a, A, i, I, o, O`
- Rückkehr in den Kommandomodus mit der `[ESC]`-Taste

Kommandozeilenmodus



- Aktivierung durch Eingabe von <:>
- Eingabe komplexer Befehle mit Argumenten und Zeilenangaben: z.B. das Ersetzen des Wortes „Vater“ durch „Mutter“ im ganzen Text:

  `:1,$s/Vater/Mutter/g`

- mit <:w> läßt sich der Text abspeichern
- VERLASSEN DES EDITORS MIT <:q>
  Verlassen des Editors ohne Abspeichern mit <:q!>

Start des Emacs

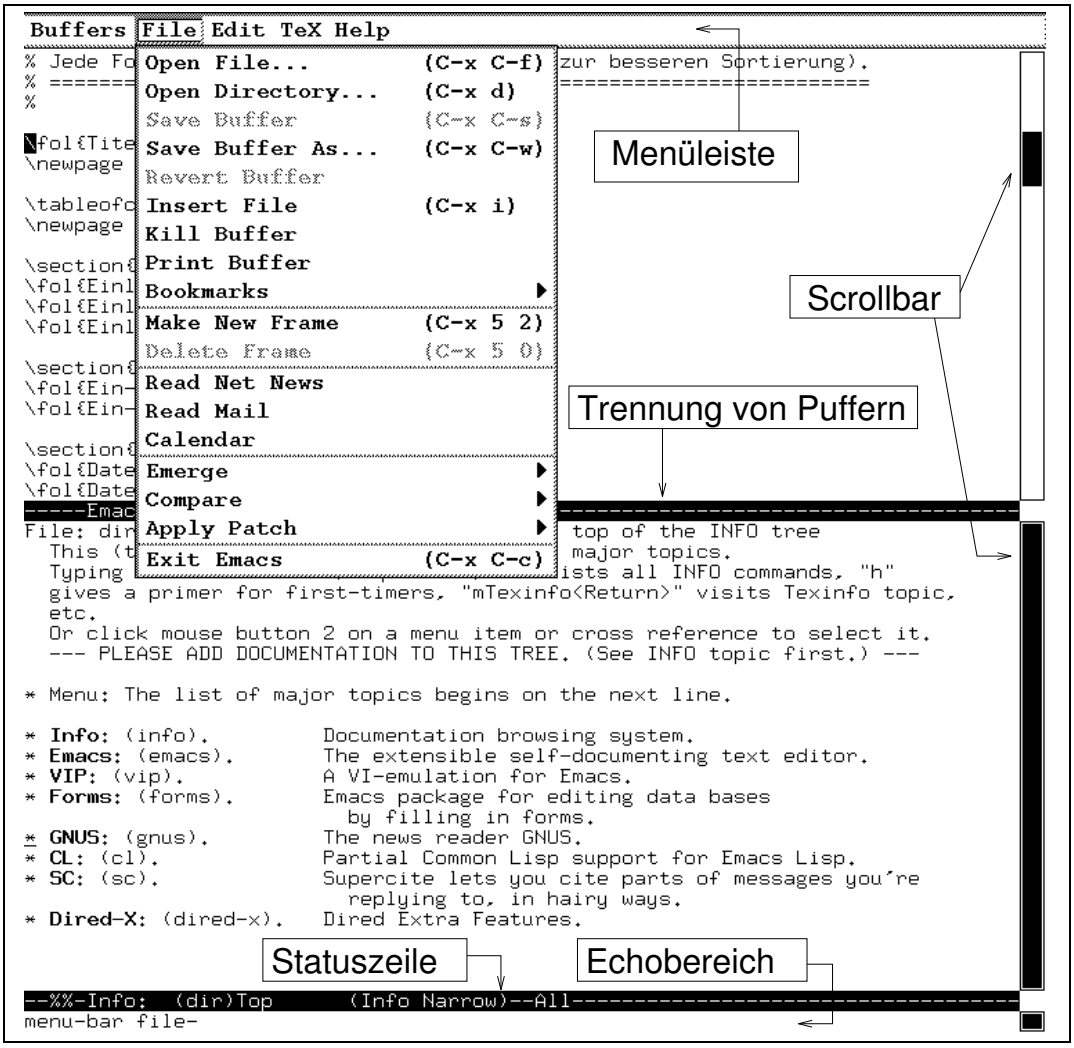
Der Aufruf des Emacs erfolgt mit

```
emacs file1 [file2 ...]  [optionen]
```

wobei `file1 [file2 ...]` die zu bearbeitenden Texte sind.

- Funktionen lassen sich entweder durch Benutzung der Maus oder durch Tastenkombinationen aktivieren.
- Beispiel: Abspeichern einer Datei entweder durch Anklicken von `Buffers/Save Buffer` oder durch Eingabe der Tastenkombination `Ctrl-x Ctrl-s`.
- Der Emacs öffnet beim Start ein eigenes Fenster.
  Daher sollte beim Aufruf ein „&“ angehängt werden, damit der Emacs als Hintergrundprozess gestartet wird und das Fenster, aus dem der Emacs aufgerufen wurde, benutzbar bleibt. Mehr zu dieser sog. Jobkontrolle erfahren Sie im Kapitel *Shell* (14.12).

Aufbau des Emacs-Fensters



- **Menüleiste** zum Aktivieren von Pulldown-Menüs.
- **Scrollbar** zum Bewegen im Text
- **Statuszeile** zum Anzeigen von Buffername, Mode, Cursorposition, etc
- **Echobereich** zum Anzeigen der gerade gewählten Funktion.

Kommandotasten

Eingabe von Kommandos durch Kombination gewöhnlicher Tasten mit Kommandotasten:

- `C-g` bedeutet gleichzeitige Betätigung der Tasten `Ctrl` und g.
- `M-v` bedeutet Taste `Escape` gefolgt von Taste „v“ (blättert hier eine Seite zurück: **page-up**).
- Komposition solcher Tastenkombinationen, z.B. `C-x C-c` (**save-buffers-kill-emacs**)
- Eingabe von Befehlen im Klartext, z.B. `M-x goto-line`.

Wichtige Kommandos:

- `C-g`: Abbruchsequenz von Emacs, beendet die aktuelle Aktion („Notbremse“). `C-g` ist immer gut, wenn man gerade nicht weiß, was das Programm will.
- `C-x C-c`: Verlassen des Editors; Emacs fragt, ob Text gespeichert werden soll oder nicht.
- `C-/` (**Undo**): Macht die letzte Aktion rückgängig, kann fast beliebig wiederholt werden.
- `C-s` (**isearch-forward**): Inkrementelle Suche nach einem Begriff.
- `C-i` (**insert-file**): Einfügen einer Datei.
- `C-x C-w` (**write-file**): Speichern des Textes in anzugebender Datei.

Hilfesystem des Emacs

Anfrage nach Hilfe mit `C-h`, gefolgt von weiterer Taste, z.B.

- `C-h ?`: Überblick über Hilfeangebote.
- `C-h t`: Tutorial, Emacs gibt eine Einführung in sich selbst: Laden und Speichern von Texten, Bewegung im Text, Suchen und Ersetzen usw.
- `C-h a`: Liste von Kommandos zu gegebenem Begriff (**command-apropos**).
- `C-h f`: Beschreibung einer Funktion (**describe-function**).
- `C-h k`: Beschreibung einer Taste (**describe-key**).
- `C-h m`: Beschreibung eines Modus, z.B. **c-mode** (**describe-mode**).
- `C-h i`: Interaktive Dokumentation (**info-mode**); der Text enthält sogenannte Nodes:

```
The character 'M-y' copies text from the kill
ring into the search string. It uses the same
text that 'C-y' as a command would yank.
*Note Yanking::.
```

  In diesem Beispiel ist *Note Yanking:: ein Node, d.h. ein Querverweis auf eine andere Textstelle, zu der man springen kann.

  Der Info-Mode stellt die komplette Emacs-Dokumentation innerhalb des Programms zur Verfügung.

8 Zu Hilfe!

8.1

Manual Pages, apropos und whatis

Auf jedem UNIX-System gibt es mehrere Möglichkeiten, Informationen über Kommandos abzurufen: **man, apropos** und **whatis**.

Manualpages

- enthalten die abstrakte Syntax eines Kommandos oder Anwendungsprogrammes.
- enthalten eine Liste aller Optionen und deren Wirkung.
- enthalten Querverweise zu anderen Befehlen/Manual Pages.
- enthalten manchmal Beispiele.
- sind nicht zum Lernen, sondern eher zum Nachschlagen geeignet.
- werden aufgerufen durch

`man` *Kommando*

Beispiel: Ausgabe von `man cd`

```
CD(1)                    USER COMMANDS                    CD(1)

NAME
     cd - change working directory

SYNOPSIS
     cd [ directory ]

DESCRIPTION
     directory becomes the new working directory.
     The  process must have execute (search)
     permission in directory. If cd is used
     without arguments, it returns you to your
     login directory. In csh(1) you may specify a
     list of directories in which directory is to
     be sought as a subdirectory if it is not a
     subdirectory of the current directory; see
     the description of the cdpath variable in
     csh(1).

SEE ALSO
     csh(1), pwd(1), sh(1)
```

Das Kommando `apropos`

- liefert die Antwort auf die Frage „Was gibt's denn alles zum Stichwort xyz ?“
- gibt eine Liste der zu diesem Stichwort verfügbaren Manual Pages aus.
- wird aufgerufen durch

apropos *Stichwort*

Beispiel: Ausgabe des Kommandos `apropos directory`

```
adv (8)         - advertise a directory for remote
                  access with RFS
cd (1)          - change working directory
chdir (2V)      - change current working directory
chroot (2)      - change root directory
...             - ...
```

Auf **System V** Rechnern (HP, IBM...) ist statt apropos der Befehl `man -k` (k steht für *keyword*) einzugeben.

Das Kommando `whatis`

Das Kommando `whatis` liefert eine einzeilige Kurzinformation, wenn der Name des Kommandos bekannt ist:

Beispiel: Ausgabe des Kommandos `whatis cd`

```
cd (1)          - change working directory
```

9 Die Graphische Benutzeroberfläche X11

9.1

Allgemeines

Die meisten Rechner unter UNIX verfügen über eine grafische Benutzeroberfläche: **X11**. **X11** bietet **Fenster** (**windows**) an, in denen einzelne Programme (Editor, Shell, WWW-Browser, etc.) arbeiten.

Eine wichtige X11-Anwendung ist der **Windowmanager**, der die Verwaltung von Fenstern übernimmt (Aussehen, Bewegen, Vergrößern von Fenstern, Reaktion auf das Drücken von Mausknöpfen, usw.). Es gibt davon etliche:

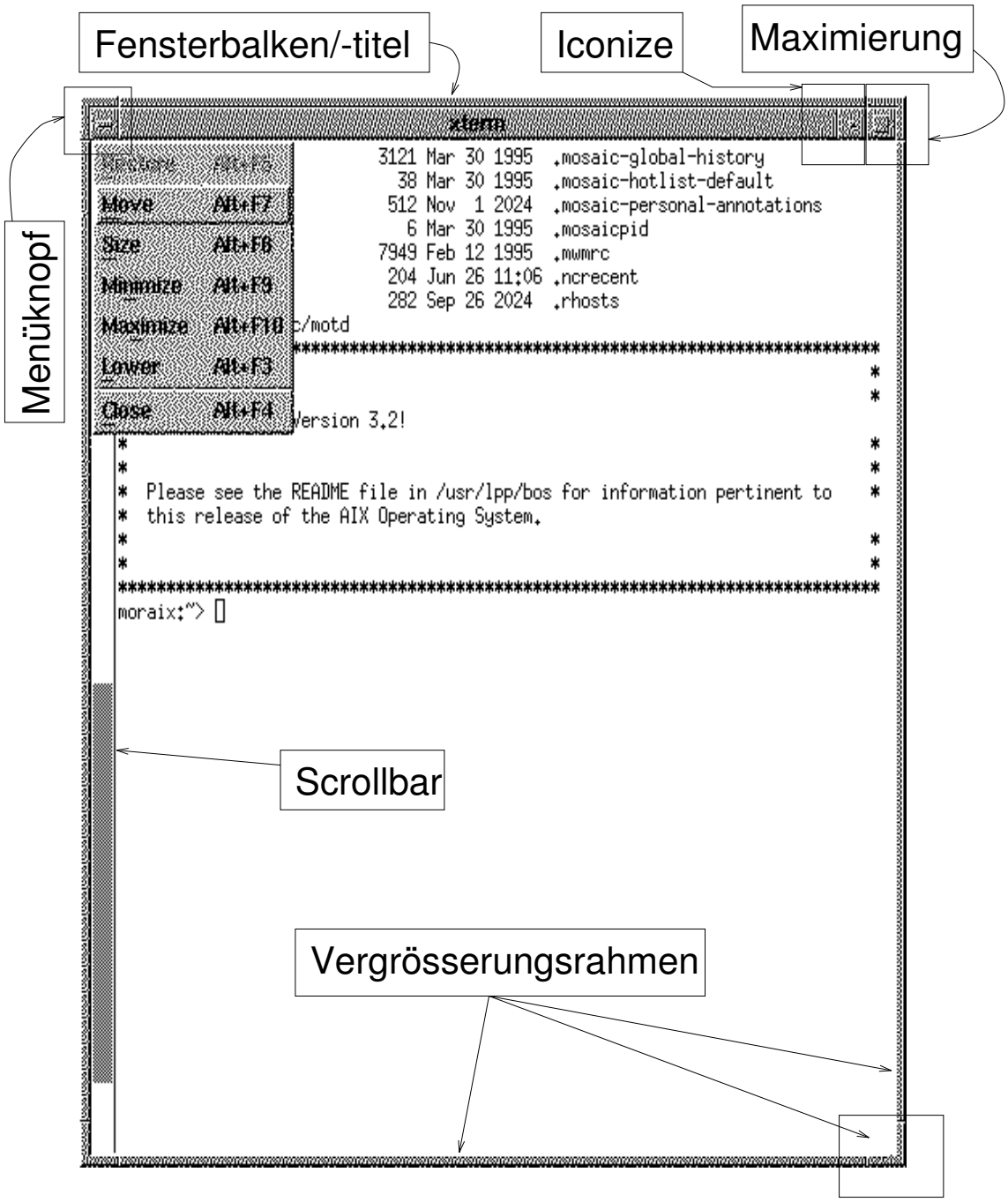
mwm	=	**Motif Window Manager**
twm	=	**Tab Window Manager**
ctwm	=	**Coloured Tab Window Manager**
fvwm	=	**F(?) Virtual Window Manager**
4Dwm	=	**IRIX Extended Motif Window Manager**

Läuft die graphische Benutzeroberfläche nicht bereits beim Einloggen, so kann man sie durch Eingabe des Befehls `startx` starten. Auf einem Linux-Rechner kann man sie entweder durch Anklicken eines Menüpunktes oder die gleichzeitige Eingabe der Tastenkombination `Ctrl`, `Alt` und `Backspace` beenden.

Achtung: Dies ist nicht gleichbedeutend mit dem Ausloggen! Man bleibt danach im sog. Textmodus eingeloggt und muß dort noch `exit` oder `logout` eingeben.

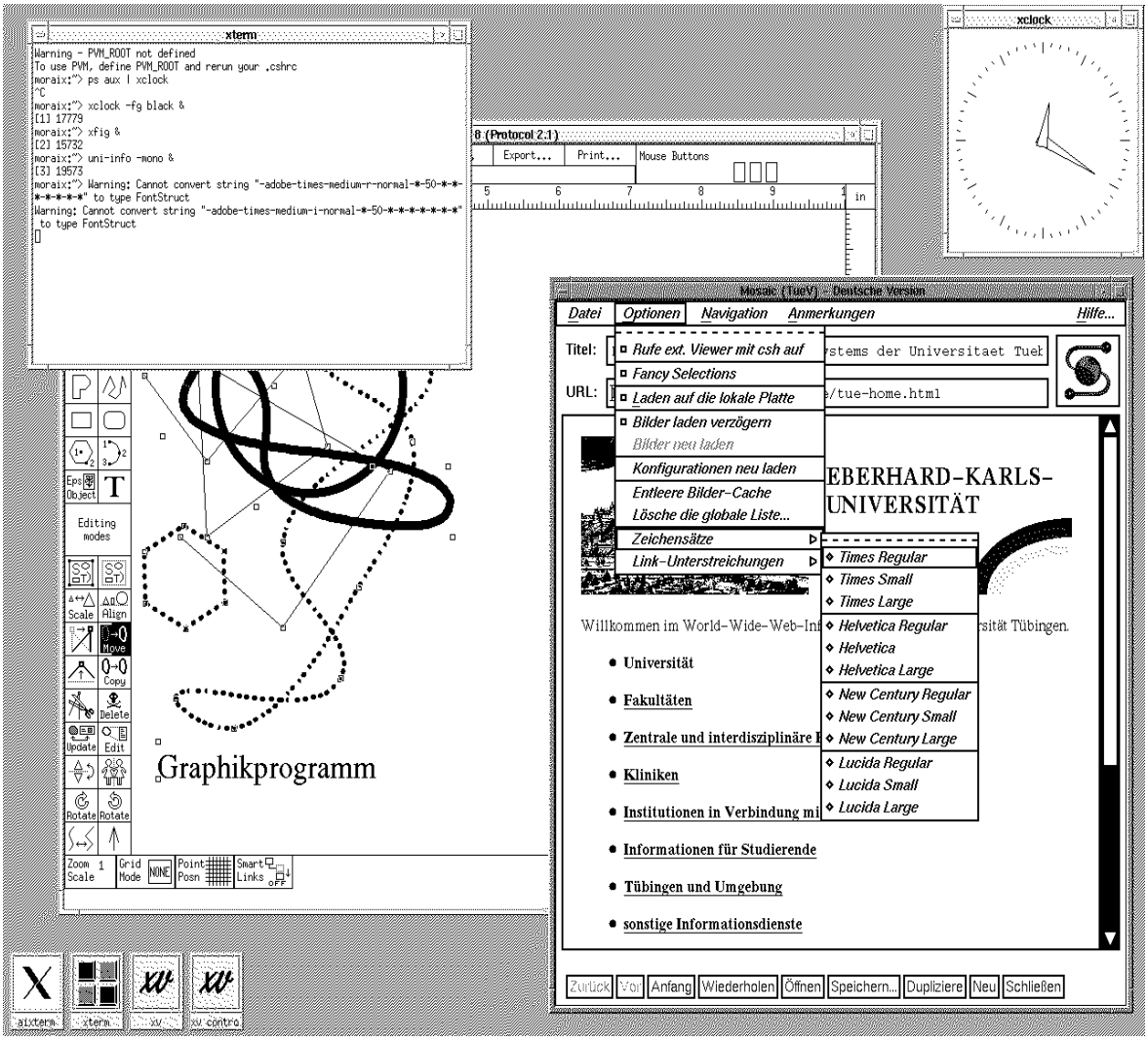
Fensterstrukturen

Ein einzelnes Fenster unter mwm kann folgende Grundstrukturen oder -bausteine aufweisen:



„Voller Bildschirm“

Um einen Gesamteindruck von einer grafischen Benutzeroberfläche zu gewinnen, ist unten ein Bildschirmausdruck gezeigt, der den mwm und mehrere X11-Anwendungen auf einem UNIX-Rechner zeigt:



Deutlich ist zu sehen, daß Fenster übereinander liegen können. Unten links sieht man „Icons“ von Fenstern, die im Augenblick nicht benötigt werden und deshalb als solche nicht sichtbar sind.

10 Kommunikation

10.1

Allgemeines

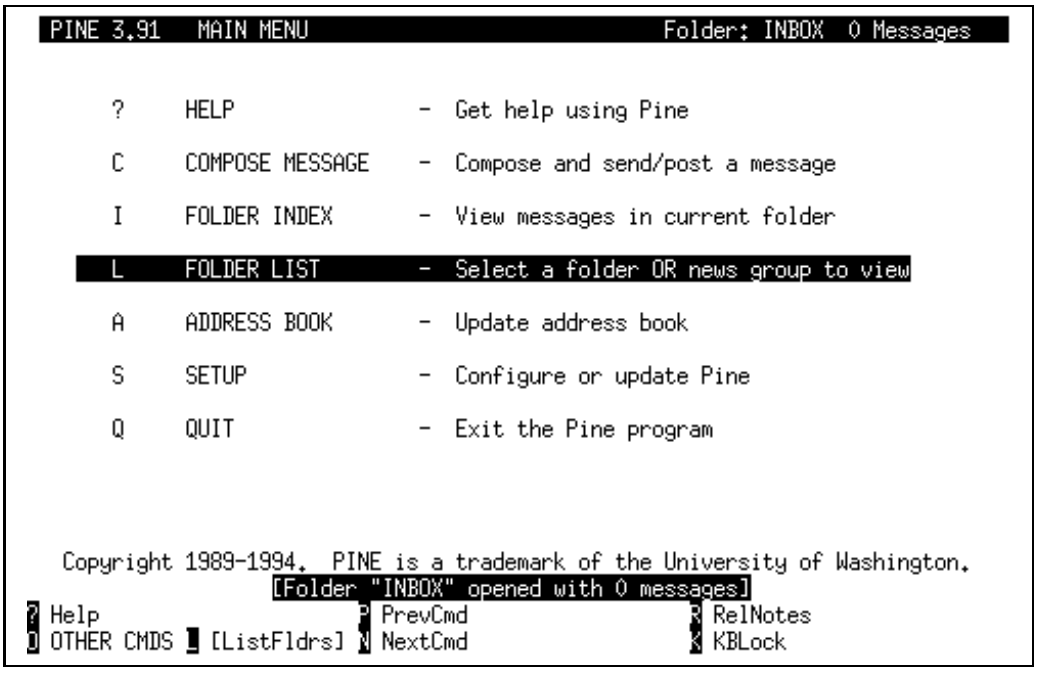
Unabhängig vom Betriebssystem wird einem Benutzer durch Anschluß an das weltweite Kommunikationsnetz die Möglichkeit gegeben, Post zwischen zwei Personen auszutauschen. Diese Form von elektronischer Post wird **E-Mail** genannt und steht im ZDV auch unter UNIX/Linux zur Verfügung.

Die E-Mail-Kommunikation wird am ZDV über einen eigenen Server abgewickelt, auf dem die Benutzer/-innen einen eigenen Account (mit einem eigenen Passwort) haben. Dieses können Sie ändern, indem Sie sich auf dem Mailserver einloggen:

`telnet` *LoginId*`.mail.zdv.uni-tuebingen.de`

E-Mail

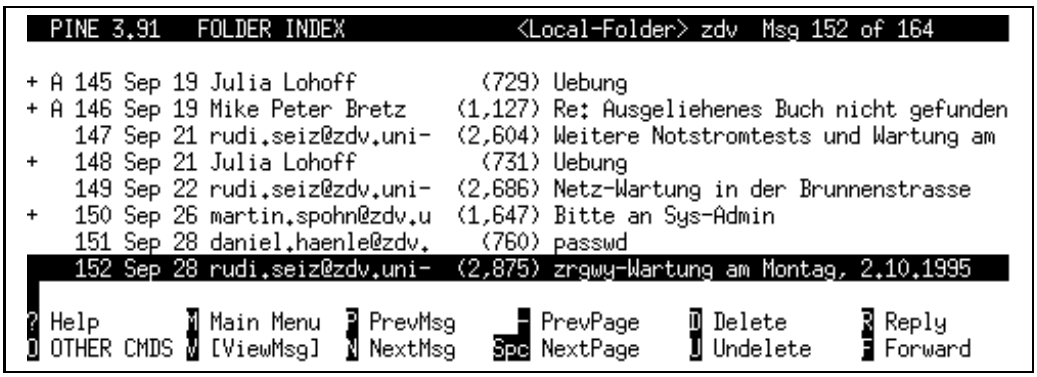
Am ZDV wurde als Standard die E-Mail-Benutzeroberfläche `pine` gewählt:



E-Mail

Das E-Mail-Programm `pine` bietet in einer textbasierten Umgebung dem Anwender die Möglichkeit, Nachrichten zu archivieren, zu versenden, zu erstellen:

und (selbstverständlich) zu empfangen:



E-Mail-Adressen

Voraussetzung für eine erfolgreiche elektronische Post ist die Kenntnis der **E-Mail-Adresse** des jeweiligen Benutzers. Für die Adressierung eines Benutzers gibt es, je nach Organisation der Rechner, folgende Möglichkeiten:

- `login-Name@domain`

  z.B. zrago01@zdv.uni-tuebingen.de
- `login-Name@Rechnername.domain`

  z.B. zrahd01@moraix.zdv.uni-tuebingen.de
  In diesem Fall geht die E-Mail an einen Benutzer auf einem bestimmten Rechner.
- `Vorname.Nachname@domain`

  z.B. markus.goller@zdv.uni-tuebingen.de
  Das **E-Mail-Alias** markus.goller wird am Zielort automatisch in den für den Rechner verständlichen Login-Namen zrago01 umgewandelt. Diese Art der Adressierung ist jedoch nicht überall realisiert, wo es E-Mail gibt.

Um die E-Mail-Adresse eines Gesprächspartners herauszufinden, benutzt man am besten den DFN-Verzeichnisdienst im **X.500**, in dem weltweit die meisten Universitäten und viele andere Einrichtungen erfaßt sind. Er ist unter der WWW-Adresse

http://ambixhp2.uni-tuebingen.de:10211/

zu finden.

Mehr darüber erfahren Sie im **E-Mail-Kurs** des ZDV
(Kursunterlagen unter
http://www.uni-tuebingen.de/zdv/mail-dienst/mailkurs/folien.html)

11 News

11.1

Das Usenet

- Das **Usenet** ist eine Sammlung von **Newsgroups** (Fachzeitschriften/Pinbrettern); derzeit etwa 6500.
- Newsgroups sind Sammlungen von **Artikeln** (Zettel am Pinbrett).
- Durch Beantworten fremder Artikel entstehen **Threads**.
- Neue Threads entstehen auch durch **Posting** eigener Artikel.
- Alle Artikel werden auf **Servern** gesammelt, z.B. *news.uni-tuebingen.de*.
- Lesen der Artikel durch **Clients**, z.B. **tin**, **nn** oder **GNUS** (Emacs).
- **Moderierte Newsgruppen** erlauben kein direktes Posten, sondern nur indirekt durch Mail an den **Moderator**.
- **FAQs** (Frequently asked questions) werden in den Newsgruppen gesammelt und archiviert (news.answers).

Newsgruppen

Hierarchien

Newsgruppen sind baumartig angeordnet, z.B.

- **cl** = Culture (Deutschsprachig)
- **cl.recht** = Unterhierarchie Recht
- **cl.recht.umwelt** = Newsgruppe Umwelt

Beispiele:

- **alt.*** - beliebige Themen, können frei eingerichtet werden, z.B. *alt.aeffle.und.pferdle* oder *alt.folklore.computers*.
- **cl.*** - deutschsprachige Kultur.
- **comp.*** - Computer, z.B. *comp.sys.mac.hardware* oder *comp.lang.c*.
- **de.*** - deutschsprachig, ähnliche Unterhierarchie, z.B. *de.alt.fan.harald.schmidt* oder *de.comp.sys.amiga.misc*.
- **news.*** - rund ums Usenet, z.B. *news.newusers* oder *news.announce*.
- **sci.*** - wissenschaftliche Themen, z.B. *sci.math.stochastics* oder *sci.biology.genetics*.

Die Newsgruppe mit den **FAQ** (Frequently Asked Questions) zu UNIX findet sich in *comp.unix.questions*.

Der News-Reader Tin

- Aufruf mit `tin -r`
- ganz oben Statuszeile (Gruppenübersicht, Artikelübersicht oder Artikelname)
- in der Mitte: Liste der abonnierten Newsgruppen oder der Artikel
- Das Gruppenübersichts-Menü des tin sieht so aus:

```
          Group Selection (newsserv.zdv.uni-tuebingen.de  6280)   h=help

     196  1160  alt.books.stephen-king
     197    57  alt.books.technical
     198   362  alt.books.tom-clancy
     199    46  alt.boomerang
     200    73  alt.brother-jed
     201 19535  alt.buddha.short.fat.guy
     202    31  alt.building.construction
     203    40  alt.building.realestate
     204  5042  alt.business
     205   162  alt.business.accountability
     206  1832  alt.business.import-export
     207   166  alt.business.insurance
     208    88  alt.business.internal-audit

  <n>=set current to n, TAB=next unread, /=search pattern, c)atchup,
g)oto, j=line down, k=line up, h)elp, m)ove, q)uit, r=toggle all/unrea,
 s)ubscribe, S)ub pattern, u)nsubscribe, U)nsub pattern, y)ank in/out
```

- Die Menüauswahl (unten) erfolgt durch Drücken einer Taste, z.B. `h` (help) für Liste aller möglichen Kommandos.
- Newsgruppen müssen abonniert (**subscribe**) oder abbestellt (**unsubscribe**) werden.
- Beim ersten Starten von tin sind alle Newsgruppen abonniert. Abhilfe: Bearbeiten der Datei *~/.newsrc* mit einem Editor. Diese Datei enthält die Namen aller Newsgruppen und wird beim ersten Start eines News-Readers erstellt.

12 Das World Wide Web

12.1

Allgemeines

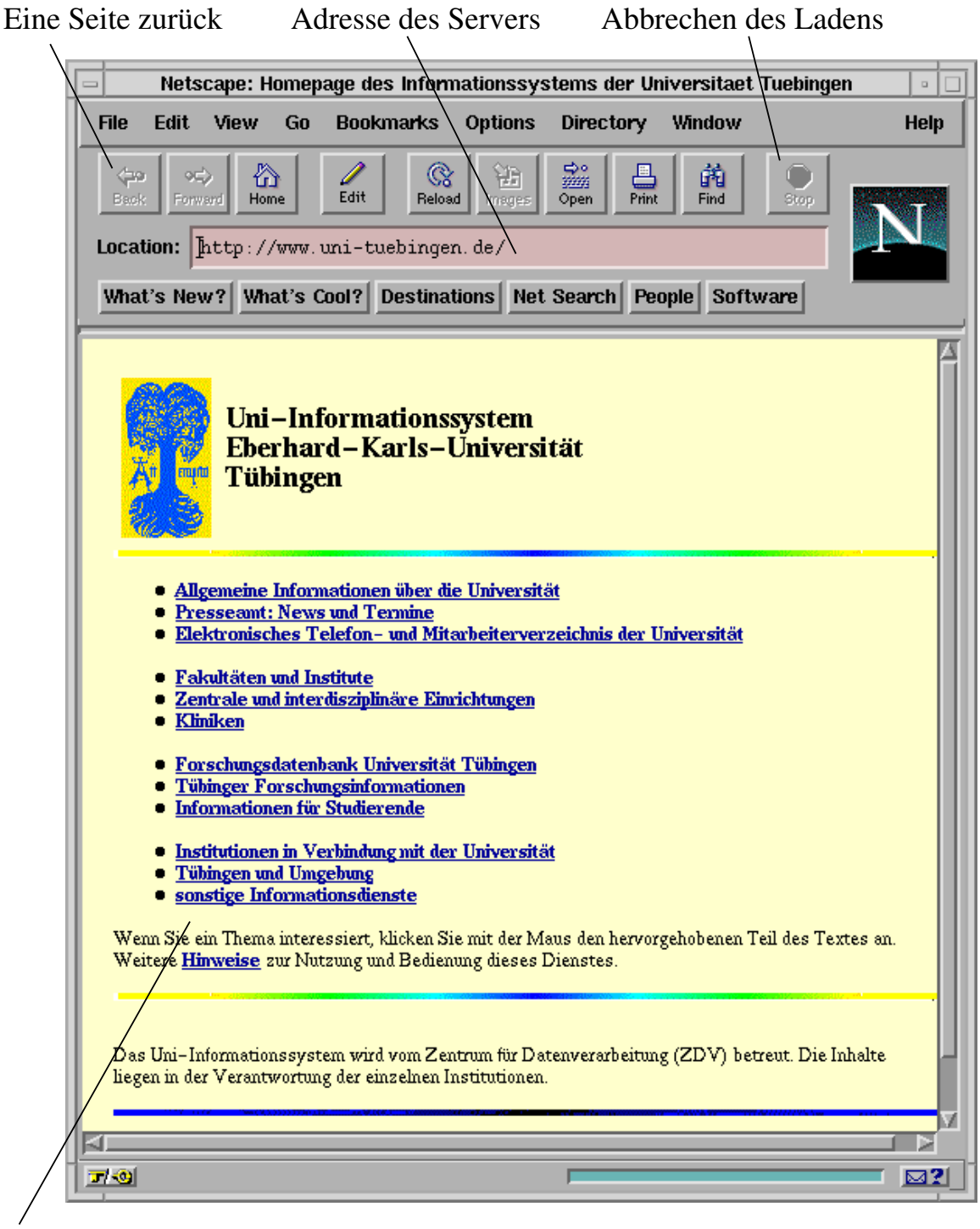
- Das **WWW** ist ein weltweites Informationsnetz.
- Benutzer im Internet stellen dabei ihre Informationen in einem bestimmten Format zur Verfügung: z.B. Text, Bilder (PostScript, gif...), Audio (meist digitalisierte Stimmen oder Musik).
- Das häufigste Format ist **HTML** (Hypertext Markup Language), welches die oben genannten Formate in einfacher Weise kombiniert.
- HTML ist eine **Hypertext**-Sprache, d.h. ein Text kann Verweise auf andere Texte (sogar Texte auf anderen Rechnern) enthalten, zu denen man dann „springen" kann.
- Mit einem sogenannten **Netbrowser**-Programm kann man all diese Informationen letztendlich sehen bzw. hören. Man „klickt" sich dabei einfach durch die verschiedenen Texte (**pages**).
- Jeder, der Informationen zur Verfügung stellt, legt eine „erste" Seite fest: die **Homepage**. Dort existieren dann meist Verweise auf andere Seiten oder Homepages.
- Möchte man eine Homepage lesen, muß der Benutzer dem Netbrowser den Namen des Rechners, auf dem die Homepage liegt, mitteilen.
  Der WWW-Server des ZDV heißt beispielsweise

  ```
  www.uni-tuebingen.de
  ```

- Das ZDV bietet unter UNIX und Linux als Standard den Netbrowser **Netscape** an. Der Aufruf von **Netscape** erfolgt im ZDV durch

  ```
  netscape &
  ```

Der Netbrowser Netscape



13 Anwendungsprogramme unter UNIX

13.1

Linux/UNIX-Software am ZDV

Aus der Vielfalt der zur Verfügung stehenden Software wird im folgenden eine kleine Auswahl kurz vorgestellt. Für weitere Informationen seien auf die Beratung und das WWW-Informationssystem verwiesen:
http://www.uni-tuebingen.de/zdv/zrsinfo/pools-n/software-linux.html

- Software, die an anderer Stelle in diesem Skript beschrieben wird:
  - Emacs (Editor, 7.5)
  - vi (Editor, 7.2)
  - Netscape Navigator (WWW-Browser, 12.1)
  - Pine (E-Mail-Reader und -Editor, 10.1)
  - Tin (News-Reader, 11.3)
  - Tar (Datenarchivierungsprogramm, 15.1)
  - Zip - Unzip (Daten(ent-)komprimierungsprogramme, 15.1)
  - telnet / rlogin / ssh (Einloggen auf anderen Rechnern, 16.1)
  - Ftp (Datentransferprogramm, 16.4)
  - mtools (Arbeiten mit DOS-Disketten, 16.7)
  - Compiler für mehrere Programmiersprachen (17.1)
- GhostView
  - Ein sog. Previewer, mit dem man sich Dateien im PostScript-Format auf dem Bildschirm anschauen kann.
  - Aufruf: `ghostview` *dateiname.ps* &
  - Anwendungsbeispiel: Man hat sich eine Datei *unix.ps* mit `ftp` oder mit `netscape` aus dem Internet geholt, und möchte sich die Datei erst anschauen, um zu entscheiden, ob sich das Ausdrucken lohnt.

Software auf den UNIX-Workstations II

- LaTeX
  - Ein Textsatz- und Layoutprogramm zum Erstellen von Texten jeder Art
  - Benutzung (Kurzüberblick): Erstellung des Textes als *dateiname.tex* mit einem Editor (vi, emacs), Erstellen einer (dvi-) Druckdatei mit `latex` *dateiname.tex*, Ausdrucken mit `dvips -P druckername` *dateiname.dvi*
  - Anwendungsbeispiel: Das Skript und die Folien zu diesem Kurs wurden komplett mit Latex erstellt.
  - Zu Latex wird zweimal im Jahr ein Kurs am ZDV angeboten.
- Xfig
  - Ein Zeichenprogramm zum Erstellen von Grafiken
  - Aufruf: `xfig` *dateiname.fig* &
  - Anwendungsbeispiele: Alle Grafiken des Unix-Skripts (z.B. die Verzeichnisbäume oder die Netscape-Seite) wurden mit Xfig erstellt.
- Xv
  - Ein Programm zur Bearbeitung und Darstellung von Bilddaten
  - Aufruf: `xv` *dateiname* &
  - Anwendungsbeispiel: Eine aus dem World Wide Web gezogene Bildatei (*.gif, *.jpeg, ...) kann vergrößert, verkleinert, beschriftet etc. werden.

Software auf den UNIX-Workstations III

- MuPAD
  - Ein freies Computeralgebrasystem, das symbolisches und numerisches Rechnen aus fast allen Teilbereichen der Mathematik beherrscht, mit weitreichenden grafischen Möglichkeiten.
  - Aufruf: `mupad`
  - Anwendungsbeispiele: Integrieren, Differenzieren, Lösen von linearen Gleichungssystemen, Zeichnen von zwei- oder dreidimensionalen Funktionen, parametrisierten Kurven etc.
- Gimp
  - Ein Programm, zur Bearbeitung bzw. Erstellung von Bildern nach Art von Photoshop oder PaintShop Pro. Der `gimp` kennt sehr viele Effekte und Bildformate.
  - Aufruf: `gimp &`
  - Anwendungsbeispiele: Icons oder Schriftzüge erstellen, Photoretuschen, Bildformatkonvertierung, ...

14 Shell

14.1

Shells

- **Shells** sind spezielle Programme, die Kommandos des Anwenders **interaktiv** empfangen und ausführen.
- Die **Login-Shell** wird beim Einloggen gestartet.
- Shells definieren eine eigene Programmiersprache mit **eingebauten** Kommandos und eigener Syntax. Außerdem erlauben sie den Aufruf **externer** Programme.

Bekannte Shells sind

sh Die Bourne-Shell war die erste Shell. Komfortablere Abkömmlinge der Bourne-Shell sind

ksh die Korn-Shell und

bash die Bourne-Again-Shell.

csh Die C-Shell unterscheidet sich stark von der Bourne-Shell. Die **tcsh** ist eine komfortable Erweiterung der C-Shell.

Bourne-Shell und C-Shell sind auf jedem Unix-Computer vorhanden. Empfohlen wird die Benutzung der `tcsh`, die auf allen ZDV –Rechnern als **Login-Shell** verwendet wird. Alle weiteren Angaben beziehen sich deshalb auf die `tcsh`.

Eingabe eines Kommandos

- Durch Ausgabe des sog. **Prompts** zeigt die Shell an, daß sie bereit ist, Kommandos entgegenzunehmen.
- Am ZDV besteht ist der Prompt so voreingestellt, daß er aus Rechnername, aktuellem Pfad und dem Zeichen „>“ besteht: `linux40:~>`
- Fehler in der Eingabezeile können mit Hilfe der Tasten `Back-space` (Löschen eines Zeichens) oder `^x` (Löschen einer Zeile) korrigiert werden. Abschluß der Eingabe mit der `Return-` bzw. `Enter-`Taste.
- tcsh, bash und ksh erlauben auch die Benutzung der **Cursor-(Pfeil-)tasten** zur Korrektur der Eingabezeile ($\leftarrow$ und $\rightarrow$) und zur Benutzung der **Command History** ($\uparrow$ and $\downarrow$). Mit $\uparrow$ erhält man den jeweils vorhergehenden Befehl - das ist sehr hilfreich, da man u.U. erheblich Tipparbeit spart!
- Kommandos und externe Programme haben folgende Syntax: `programm [optionen] <argumente> [argumente]`, z.B. `lpr -Pps001 Brief`.
- Leerzeichen trennen die einzelnen Optionen und Argumente eines Befehls. Die Eingabe von `lpr-Pps001 Brief` hätte die Ausgabe der Fehlermeldung
  `lpr-Pps001: Command not found.`
  zur Folge, da UNIX nach einem (nicht vorhandenen) Befehl aus diesen zehn Zeichen (am Stück) sucht.
- UNIX unterscheidet Groß- und Kleinschreibung! Daher würde `LPR -Pps001 Brief` diesselbe Fehlermeldung hervorrufen. Ein häufiger Fehler besteht darin, daß man versehentlich die **Caps-Lock**-Taste gedrückt hat und dann das Gefühl hat, daß der Rechner einen nicht mehr versteht (was ja auch der Fall ist).

Wildcards in Dateinamen

- Viele Kommandos (z.B. `ls` oder `cat`) haben als Argumente die Namen von Dateien oder Verzeichnissen.
- Namen, die **Wildcards** enthalten, werden zu einer Liste von Namen **expandiert**.
- Beispiele von Wildcards sind

?	beliebiges Zeichen
*	beliebige Zeichenkette, auch Länge Null
[abd]	die einzelnen Zeichen a,b oder d
[a-dg-i]	die einzelnen Zeichen a,b,c,d,g,h, oder i
[^abd]	nicht die einzelnen Zeichen a,b oder d
{dies,das}	optionale Zeichenketten

- Es werden nur Namen existierender Dateien generiert.

Beispiele:

prog*	*prog.c*, *prog.tex* oder *prog*
*.[ch]	*datei.c*, *prog.c*, *prog.h*
prog.?	*prog.c*, *prog.h*, *prog.f*
acc.{jan,feb}	*acc.jan* und *acc.feb*
/usr/bin/*/*	alle Dateien in Unterverzeichnissen von */usr/bin*, z.B. */usr/bin/X11/xterm*
~ralf/*	alle Dateien im **home directory** von ralf

- Wildcards sind nahe verwandt zu **regular expressions**.

Ein-/Ausgabeumlenkung

Eingabeumlenkung

- Erwartet ein Kommando Eingaben des Benutzers, so erfolgen diese gewöhnlich von der Tastatur.
- Durch `< infile` kann man ein Programm veranlassen, den in *infile* vorhandenen Text zu benutzen, als würde er an der Tastatur eingegeben.
- Beispiel:

  ```
  mail zrasl01@zdv.uni-tuebingen.de < brief
  ```

  verschickt die Datei *brief* als Mail an die angegebene E-Mail-Adresse.

Ausgabeumlenkung

- Die Ausgabe eines Kommandos erfolgt normalerweise auf den Bildschirm.
- Durch `> outfile` kann man die Ausgabe auf die angegebene Datei *outfile* umlenken.
- Beispiel:

  ```
  ls -al ~ > homedir
  ```

  schreibt die Liste aller Dateien im Home-Directory in die Datei *homedir*.

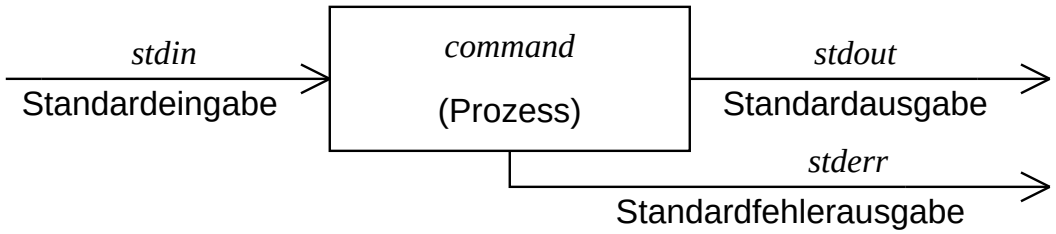
Ein-/Ausgabeumlenkung II

- Die Shell eröffnet bei der Ausführung eines Programmes Dateien:

  stdin Die **Standardeingabe**, normalerweise die Tastatur.

  stdout Die **Standardausgabe**, normalerweise der Bildschirm.

  stderr Die **Standardfehlerausgabe**, meist ebenfalls der Bildschirm.



- Die Shell kann die Vorgabedateien durch andere ersetzen:

  `<infile` ersetzt Tastatur durch die Datei *infile*. Beispiel (14.4).

  `>outfile` ersetzt die Standardausgabe durch die Datei *outfile*. Beispiel (14.4).

  `>&outfile` schreibt Ausgabe und Fehlermeldungen in *outfile*, z.B.

```
> cc prog.c >& errors
```

Ein-/Ausgabeumlenkung III

- Bei Ausgabeumlenkung wird die Ausgabedatei neu erzeugt. Verwendet man statt `>outfile` die Syntax `>>outfile`, so wird die Ausgabe an die bereits bestehende Datei *outfile* angehängt, d.h.

  ```
  ls /bin >unixcommands
  ls /usr/bin >>unixcommands
  ```

  entspricht

  ```
  ls /bin /usr/bin >unixcommands
  ```

- Analog kann man mit `>>&outfile` Standardausgabe und Standardfehlerausgabe an den Inhalt von *outfile* anhängen.
- Eine spezielle Datei ist */dev/null*. Bei Verwendung als Eingabedatei bewirkt sie ein sofortiges Eingabeende, als Ausgabedatei ignoriert sie alle eingehenden Daten („Mülleimer“).

  Beispiel:

  ```
  cat </dev/null >leeredatei
  ```

  erzeugt eine leere Datei.

  ```
  cat file >/dev/null
  ```

  gibt *file* nach /dev/null aus, es werden nur evtl. Fehlermeldungen auf der Standardausgabe angezeigt.

Pipes und Filter

- Häufig soll die Ausgabe eines Kommandos durch ein anderes weiterverarbeitet werden, z.B.

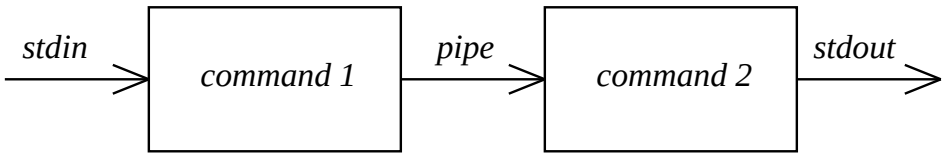
  ```
  ls /bin /usr/bin >unixcommands
  lpr -Pdrucker unixcommands
  ```

- Durch eine **Pipeline** oder **Pipe** kann man die Standardausgabe eines Kommandos zur Standardeingabe eines anderen weiterleiten:

  ```
  ls /bin /usr/bin | lpr -Pps007
  ```

  Allgemein:

  ```
  command1 | command2
  ```



- **Filter** sind Programme, die von der Standardeingabe lesen und zur Standardausgabe schreiben, insgesamt also filtern.
- Wichtige Filter sind:

  `sed` **stream editor**

  `grep` durchsucht Texte nach Mustern

  `awk` Muster-Such- und Verarbeitungssprache

  Bsp: `grep music ~/.newsrc` durchsucht die Datei *.newsrc* nach allen Newsgruppen, die etwas mit Musik zu tun haben.

- Filter kann man kombinieren, z.B. durchsucht

  ```
  cat *.txt | grep Unix | sed s/alt/neu/g
  ```

  alle mit *.txt* endenden Dateien nach dem Wort **Unix** und ersetzt das Wort **alt** durch **neu**. Die Ausgabe von sed erfolgt auf den Bildschirm.

Umgebungsvariablen

- **Umgebungsvariablen** enthalten Zeichenketten als Werte.
- Umgebungsvariablen werden vom Benutzer gesetzt. Sie werden von Programmen ausgewertet, die daraufhin ihr Verhalten danach ausrichten.
- Wichtige Umgebungsvariablen sind:

  EDITOR Enthält den Namen des bevorzugten Texteditors, Vorgabe `vi`.

  PRINTER Enthält den Namen des bevorzugten Druckers

  DISPLAY Enthält den Namen des Rechners, auf dessen Bildschirm eine Anwendung gestartet werden soll. Wird vor allem im Zusammenhang mit `telnet` und `rlogin` benutzt (siehe 16.3).

- Manche Programme definieren eigene Umgebungsvariablen, die vom Anwender gesetzt werden können, z.B. benutzt der WWW-Browser **Netscape** die Variable `HOME_DOCUMENT`, um die Adresse einer bevorzugten Homepage zu setzen.
- Auch bei der Eingabe von Kommandos können Umgebungsvariablen verwendet werden:

```
echo Mein Lieblingsdrucker ist der $PRINTER.
```

- Mit dem Kommando `setenv` werden Umgebungsvariablen **gesetzt**, z.B.

```
setenv EDITOR /soft/bin/emacs
```

- Das Kommando `setenv` ohne Argumente gibt eine Liste aller Umgebungsvariablen aus. Tip:

```
setenv | sort | more
```

- Umgebungsvariablen werden mit dem Kommando `unsetenv` wieder aufgehoben.

Die Suchpfadvariable

- Der **Suchpfad** der Shell ist eine Liste von Directories.
- Der Suchpfad wird durch die Umgebungsvariable **PATH**, eine durch Doppelpunkte getrennte Liste, definiert.

  Beispiel eines Pfades:
  /usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/afs/bin:/usr/afsws/bin:/usr/afsws/etc:/usr/bin/X11:/sbin:/usr/local/bin:/soft/bin:.
- Wird ein Programm aufgerufen, so werden alle in **PATH** definierten Verzeichnisse nach diesem Programm durchsucht.
- Der Suchpfad wird durch

  ```
  setenv PATH "$PATH:/usr/bin/X11"
  ```

  um das Verzeichnis */usr/bin/X11* erweitert.
- Die Reihenfolge der Directories im Suchpfad ist wichtig: Im ersten Directory wird zuerst gesucht.
- Den aktuellen Suchpfad kann man sich mit

  `echo $PATH` ausgeben lassen.
- Programme, die nicht in einem Verzeichnis des **Suchpfades** zu finden sind, können nur durch Angabe ihres Pfades aufgerufen werden, z. B.

  ```
  /soft/bin/gcc program.c
  ```

  statt

  ```
  gcc program.c
  ```

Quotierung und Kommandosubstitution

- Eine Reihe von Zeichen haben für die Shell besondere Bedeutung:

  <SPACE>, <TAB>, <CR>, $, *, [,], ?, {, }, ~, -, <, >,&, !, |, ;, (,), \, ', ", `.

- Um diese Zeichen benutzen zu können, ohne daß die Shell sie interpretiert, müssen diese **quotiert** werden.
- Ein einzelnes Sonderzeichen wird durch vorstellen eines Backslash (\) quotiert:

  ```
  echo Ein Hamburger kostet 2\$.
  ```

- Verwendet man doppelte Anführungsstriche, so werden von der Shell nur noch !, $, \ und ` als Sonderzeichen behandelt. Man kann also z.B. noch Umgebungsvariablen verwenden:

  ```
  echo "$EDITOR"
  ```

  gibt den Wert der Variablen EDITOR aus.

- Verwendet man einfache Anführungsstriche (', nicht `), so werden von der Shell nur noch ! und \ als Sonderzeichen behandelt.

  ```
  echo '$, >, <, \' sind Sonderzeichen.
  ```

- Mit `programm` (hier `, nicht ') wird die Ausgabe von `programm` in die Kommandozeile einbaut. Dies bezeichnet man als **Kommandosubstitution**. Beispiel:

  ```
  lpr -Pps007 `ls ~`
  ```

  gibt die Namen aller Dateien im Homedirectory auf den Drucker **ps007** aus.

Abkürzen von Befehlen durch `alias`

- Häufig benutzte Kommandos wie z.B. `ls -al` kann man mit dem `alias`-Kommando abkürzen, um Tipparbeit zu sparen.
- Syntax: `alias <Abkürzung><Befehl>`
- Beispiele für alias-Definitionen sind
```
alias ll ls -al
alias drucke lpr -Pps001_d4
alias e emacs -fn 9x15
alias dismor setenv DISPLAY moraix:0.0
```
- Die obigen (Neu-)Definitionen lassen sich dann folgendermaßen benutzen:
`ll /soft/bin` (entspricht `ls -al /soft/bin`)
`drucke Brief` (entspricht `lpr -Pps001_d4 Brief`)
`e Diplomarbeit.tex`
- Der Befehl `alias` ohne Argumente zeigt alle bereits vergebenen Abkürzungen an.
- Die alias-Definitionen muß man nicht jedesmal eingeben, sondern kann sie in der Datei *.alias* oder auch in *.cshrc* (s. 14.16) unterbringen. Sie stehen dann in jedem Fenster zur Verfügung, da diese Dateien beim Einloggen automatisch gelesen werden.

Jobkontrolle

- Ein laufendes Kommando kann mit `^Z` (Gleichzeitiges Drücken der Tasten `Ctrl` und `Z`) angehalten werden. Nach dem Anhalten meldet sich die Shell mit ihrem Prompt wieder, neue Kommandos können eingegeben werden.
- Das unterbrochene Kommando kann mit dem Befehl `fg` (**Foreground**) wieder aktiviert werden.
- Mit dem Befehl `bg` (**Background**) kann das Kommando im Hintergrund fortgesetzt werden, sofern das Kommando keine Eingabe über die Tastatur (**stdin**) erfordert.
- Man kann ein Kommando auch direkt im Hintergrund starten, indem man **&** an das Kommando anhängt. Weitere Kommandos können dann eingegeben werden, während das erste ausgeführt wird. Beispiele:
  - `gzip unix-kurs.tar &` komprimiert die Datei *unix-kurs.tar*. Es gelten die gleichen Einschränkungen wie bei bg.
  - `xterm &` öffnet ein neues, parallel zu benutzendes xterm-Fenster unter X-Windows.
  - `emacs unix-kurs.tex &`
  - `netscape &`
- Unter X-Windows bedeutet das Anhängen von &, daß das Fenster, aus dem eine Anwendung gestartet wurde, weiter benutzt werden kann. Daher sollte dies bei Anwendungen, die längere Zeit laufen sollen (z.B. netscape, ghostview, emacs, xv, xfig), immer gemacht werden.
- Ein im Hintergrund laufendes oder unterbrochenes Programm benötigt unter Umständen die Tastatur bzw. den Bildschirm zur Ein- bzw. Ausgabe. Die Shell kann deshalb nicht beendet werden, und es erscheint beim Versuch des Ausloggens die Fehlermeldung „There are suspended jobs“.

Skripte und Skriptkommandos

- Mehrere Kommandos können in einer Textdatei zusammengefaßt werden, z.B. um immer wiederkehrende Kommandofolgen nicht jedesmal explizit eintippen zu müssen. Diese Textdateien heißen **Shell-Skripte** und stellen neue Kommandos dar.
- In Shell-Skripten sind mit # beginnende Zeilen Kommentare, diese Zeilen werden nicht ausgeführt.
- Beginnt die erste Zeile mit `#!` so wird zur Ausführung des Shell-Skripts die danach angegebene Shell gestartet:

  `#!/bin/csh`
- Shell-Skripte müssen mit

  `chmod u+x skript`

  als ausführbare Kommandos gekennzeichnet werden.

Im folgenden benutzen wir die **(t)csh**.

Shell-Skripte besitzen eine erweiterte Syntax:

- Es stehen mathematische Operationen und Vergleichsoperationen wie in der Programmiersprache C zur Verfügung.
- verschiedene Kontrollstrukturen (siehe Beispiele)

Beispiele:

- Mit `if` können Bedingungen abgefragt werden:

```
#!/bin/csh
if ( $?DISPLAY ) then
  setenv TERM xterm
else
  setenv TERM vt100
endif
```

Skripte und Skriptkommandos II

- Eine erweiterte Variante von `if` ist `switch`:

```
#!/bin/csh
switch ( $HOSTNAME )
  case compserv:
    setenv EDITOR /soft/bin/gemacs
    breaksw
  case sun*:
    setenv EDITOR /usr/ucb/vi
    breaksw
  default:
    setenv EDITOR /soft/bin/emacs
    breaksw
endsw
```

- Schleifen kann man mit dem `foreach`-Kommando implementieren:

```
#!/bin/csh
# Sichere alle .f-Dateien in .f.orig
foreach file (*.f)
  cp $file $file.orig
end
```

- Mit `exit n` wird ein Skript beendet. **n** gibt dabei den Fehlerstatus an (0 = kein Fehler).

Das Shellscript numfiles (Beispiel)

```
#!/bin/csh
# Dieses script zaehlt die Anzahl von Dateien
# und Verzeichnissen im Argument1 oder im
# current working directory
# Usage: numfiles [directory]

if ($#argv == 0) then       # kein argument ?
   set dir = "."            # benutze cwd
else
   set dir = $argv[1]       # benutze argv 1
endif

if (! -d $dir) then         # kein directory ?
   echo $0\: $dir ist kein Verzeichnis.
   exit 1                   # Fehlermeldung
                            # und Fehlerstatus
endif

echo $dir\:                 # Verzeichnis anzeigen
@ fcount = 0                # Zaehler setzen
@ dcount = 0

cd $dir
foreach file (*)            # alle Eintraege
   if (-f $file) then       # Datei ?
      @ fcount++            # Dateizaehler erhoehen
   else if (-d $file) then  # Verzeichnis ?
           @ dcount++       # Verzeichniszaehler
        endif
   endif
end

# Ergebnis anzeigen
echo $fcount Dateien $dcount Verzeichnisse
```

Einrichten der individuellen Benutzerumgebung

Die Datei *.cshrc*

- befindet sich im Homedirectory.
- wird bei jedem Start einer csh oder tcsh ausgeführt.
- enthält Einstellungen wie z.B. Umgebungsvariablen (**PATH** !), `alias` Abkürzungen oder den eigenen Prompt (`set prompt="Ja, Chef ?"`).
- kann bei heterogenen Pools (Workstations unterschiedlicher Hersteller) Abfragen über den Rechnertyp enthalten.

Die Datei *.login*

- befindet sich im Homedirectory.
- wird nur einmal beim Einloggen ausgeführt.
- enthält gewöhnlich Umgebungsvariablen (z.B. PATH, den Suchpfad).

Die Datei *.xinitrc*

- enthält die Kommandos, die beim Starten der Benutzeroberfläche X11 ausgeführt werden (also ebenfalls nur einmal).
- muß mindestens den Aufruf eines **xterm** (Fenster mit Shell zum Eingeben von Befehlen) und eines Windowmanagers enthalten.
- Weitere mögliche Aufrufe von Programmen sind eine Uhr (`xclock`), ein Fenster für Systemmeldungen (`xterm -C`) oder ein Bild als Arbeitshintergrund (mit `xv` oder `xsetroot`).
- Als Vorlage für ein eigenes *.xinitrc* dient die Datei */usr/lib/X11/xinit/xinitrc*, die man unter dem Namen *.xinitrc* in sein Homedirectory kopieren und dann verändern kann.
- muß als letzten Befehl entweder den Aufruf eines Windowmanagers oder eines xterms beinhalten (und zwar ohne **&** !).

Beispiel eines *.xinitrc*

```
# Starten der verschiedenen X11-Programme

# Zuerst wird ein X-Terminal gestartet, das 65
# Spalten breit ist und 42 Zeilen hoch. Im Titel-
# fenster erscheint "SUNAPS" und der Font hat
# die Groesse 9x15.

xterm -geometry 65x42+0-5 -fn 9x15 -n "SUNAPS" &

# Jetzt starte eine Uhr mit der Breite und Hoehe
# von 150 Pixeln an den x-y-Koordinaten -1 +1

xclock -geometry 150x150-1+1 &

# Als letzten Client starte ich den Window-
# manager "mwm". Dieser Client laeuft nicht im
# Hintergrund, d.h. wenn der Windowmanager
# beendet wird, wird auch die aktuelle Sitzung
# beendet.

mwm
```

15 Packen und Komprimieren von Daten

15.1

Dateien packen mit `tar`

- Möchte man viele zusammenhängende Dateien kopieren oder weitergeben, sollte man sie als Einheit behandeln können.
- Mit dem Kommando `tar` können Dateien oder ganze Verzeichnisse zu einem großen Block zusammengefaßt werden (**Einpacken**).
- Am Zielort können die in den Block eingepackten Dateien dann wieder **ausgepackt** werden.
- Auch die Verzeichnisstruktur wird gespeichert, d.h. die Dateien liegen nach dem Auspacken in denselben Verzeichnissen.
- Dieser Block ist eine Datei (**Tarfile**) und hat gewöhlich die Endung *.tar* .
- Mit `tar` wird noch kein Platz gespart. Das tarfile läßt sich jedoch mit einem Komprimierungsprogramm „verkleinern“.

Das Komprimierungsprogramm `compress`

- ist auf jedem UNIX-System vorhanden.
- reduziert die Größe von Dateien bis auf 60% .
- ist bei eventuellem quota zu empfehlen.
- Eine komprimierte Datei erhält die Endung *.Z*.
- Ein komprimiertes Tarfile heißt also z.B. *tarfile.tar.Z*.
- Eine mit `compress` komprimierte Datei wird mit `uncompress` wieder entkomprimiert.
- ist nicht besonders leistungsfähig. Es gibt bessere Komprimierungsprogramme wie z.B. `gzip` (steht im ZDV zur Verfügung). Weitere Informationen erhält man durch Eingabe von

  `gzip --help`.

Die Anwendung von `tar`

`tar c|x|t [fv] [`*tarfile*`] `*file1 file2* `...`

c (**create**): Erstellen/Einpacken eines tarfiles.

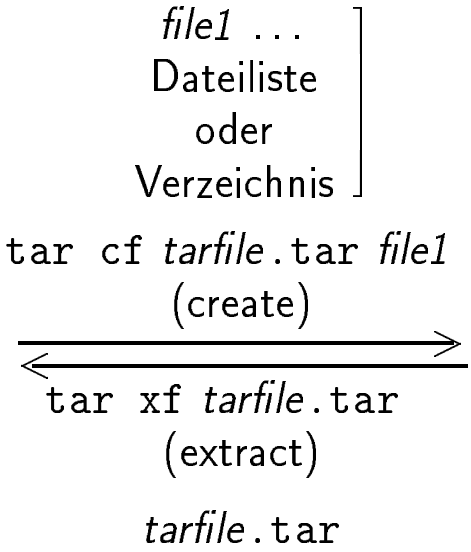
x (**extract**): Auspacken eines tarfiles. Vorsicht, vorhandene gleichnamige Dateien werden überschrieben !

t (**type**): zeigt den Inhalt eines tarfiles an, ohne dieses auszupacken.

v (**verbose**) liefert Zusatzinformationen über die Aktivitäten von `tar`.

f (**filename**) bestimmt den Namen des tarfiles.

Beispiel:



Ein Beispiel

Einpacken und komprimieren der beiden Dateien *datei1.txt* und *datei2.txt*:

- Erzeugen des tarfiles *demo.tar* **(verbose)**:

  `tar cvf demo.tar datei1.txt datei2.txt`

  ```
  a datei1.txt 7 blocks
  a datei2.txt 7 blocks
  ```

- Inhalt von *demo.tar*:

  `tar tvf demo.tar`

  ```
  rw-------425/100 3237 Apr 30 11:42 1992 datei1.txt
  rw-------425/100 3237 Apr 30 11:43 1992 datei2.txt
  ```

- Komprimieren mit `compress`:

  `compress demo.tar` (erzeugt *demo.tar.Z*)

- Ausgabe des Inhalts von demo.tar.Z, ohne es zu entpacken (Trick 17):

  `tar -ztvf demo.tar.Z`

  Die Option `z` bewirkt dabei die Dekomprimierung mit `gzip`.

- Entkomprimieren mit `uncompress`:

  `uncompress demo.tar.Z` (erzeugt *demo.tar*)

- Auspacken von *demo.tar*:

  `tar xvf demo.tar`

16 Verbindung mit anderen Rechnern

16.1

telnet - rlogin

- Mit dem Kommando `telnet` kann man sich auf einem anderen Rechner einloggen (sofern man ein account hat) und den eigenen Rechner als „Terminal“ benutzen.
- Aufruf durch

  `telnet Rechnername`

  Beispiel: `telnet linux40`
- Es werden wie beim Einloggen die User-ID und das Paßwort abgefragt.
- Abbrechen von telnet erfolgt entweder durch die Tastenkombination **CTRL-D** oder durch Warten auf Timeout (meist eine Minute).
- Ausloggen am anderen Ende wie gewohnt durch `exit` oder `logout`.

- `rlogin` funktioniert im Wesentlichen wie `telnet`, bietet aber die Möglichkeit des Einloggens ohne Paßwort, sofern eine Datei *.rhosts* existiert.
- Die *.rhosts* muß sich im homedirectory befinden und enthält Einträge in folgender Form:

  `Rechnername.domain  login-Name`

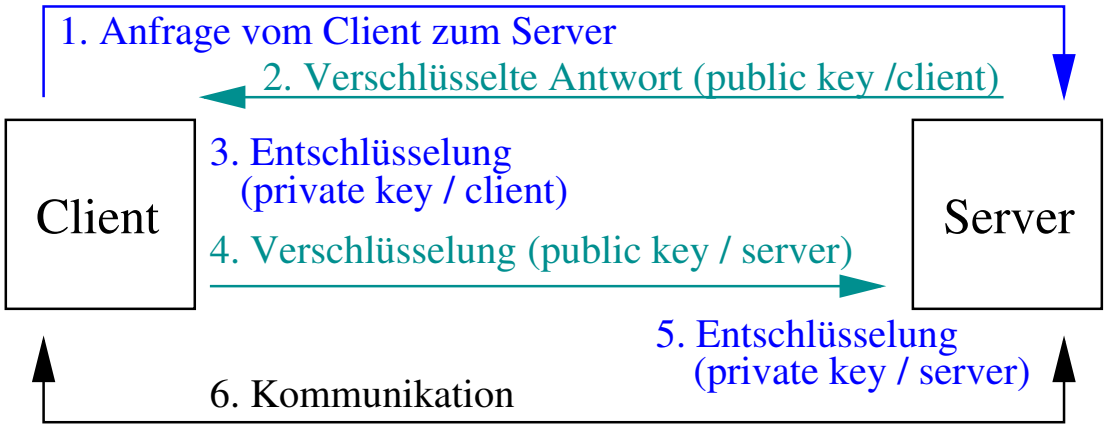
- Der Aufruf erfolgt durch

  `rlogin Rechnername`
- Da dieses Verfahren eine Sicherheitslücke darstellt, sollte in Zukunft das Secure-Shell-Paket **ssh** (s.16.2) eingesetzt werden.

Die Secure Shell ssh

- Das Einloggen mit `telnet` oder `rlogin` hat den Nachteil, daß die Kommunikation zwischen den Rechnern unverschlüsselt abläuft und daher „abgehört“ werden kann.
- Die **Secure Shell (ssh)** ermöglicht eine verschlüsselte Verbindung mittels eines sog. **Public Key-Kryptoverfahrens**.
- Dabei wird der Datentransfer mit einem sog. Public Key codiert. Der Decodierschlüssel (Private Key) ist dagegen privat.
- Um die `ssh` verwenden zu können, muß man folgendes tun:
  1. Je einen privaten und öffentlichen Schlüssel auf dem Arbeitsplatzrechner (Client, hier: linux40) erzeugen:
     `linux40:~> ssh-keygen`
     Es wird ein Unterverzeichnis *.ssh* erzeugt, in dem die Dateien *identity* (Public Key) und *identity.pub* (Private Key) gespeichert werden.
  2. Den so erzeugten öffentlichen Schlüssel *identity.pub* auf den entfernten Rechner (Server, hier: textserv1) kopieren (z.B. per `ftp`) und in die Datei *~/.ssh/authorized_keys* einfügen:
     `textserv1:~/.ssh> cat identity.pub >> authorized_keys`
- Die Verwendung der **ssh** funktioniert analog zu `telnet` oder `rlogin` durch Verwendung der Befehle `ssh` oder `slogin`:
  `linux40:~> slogin textserv1`

Graphisch dargestellt sieht der Verbindungsaufbau so aus:



X11-Anwendungen auf anderen Rechnern

Will man auf einem anderen Rechner nicht nur mit Unix-Befehlen arbeiten, sondern auch Anwendungsprogramme benutzen, die auf die grafische Benutzeroberfläche X11 zurückgreifen, sind einige zusätzliche Schritte nötig.

- Zunächst muß dem anderen Rechner der Bildschirmzugriff auf dem gegenwärtigen Arbeitsplatzrechner gestattet werden. Dies geschieht mit

  `xhost Entfernter_Rechnername`

- Dann erfolgt die Einlogprozedur mit `telnet` oder `rlogin` wie beschrieben.
- Als nächstes muß die Ausgabe („Display“) auf den lokalen Arbeitsplatzrechner umgelenkt werden. Dazu setzt man die Variable `DISPLAY` mittels

  `setenv DISPLAY Lokaler_Rechnername:0`

  wobei der Rechnername die Internetadresse des Rechners ist.
- Nun können Programme wie `Netscape`, `Emacs` oder auch ein neues `xterm`-Fenster wie gewohnt gestartet werden.

Beispiel: Einloggen vom Rechner moraix.zdv.uni-tuebingen.de auf dem Rechner indiaka.zdv.uni-tuebingen.de und Starten von `netscape`:

```
moraix:~> xhost indiaka.zdv.uni-tuebingen.de
indiaka.zdv.uni-tuebingen.de being added to access control list
moraix:~> rlogin indiaka
IRIX Release 5.3 IP22 indiaka
Copyright 1987-1994 Silicon Graphics, Inc. All Rights Reserved.
Last login: Thu Feb 20 11:41:30 MEZ 1997
by zrago01@moraix.zdv.uni-tuebingen.de
indiaka:~> setenv DISPLAY moraix.zdv.uni-tuebingen.de:0.0
indiaka:~> netscape &
[1] 2747
indiaka:~>
```

Filetransfer per FTP

Das Programm **FTP** (File Transfer Program) ist installiert auf

- jedem UNIX-Rechner des Rechenzentrums.
- den öffentlichen PC's des Rechenzentrums („Expreßstation“) auf der Morgenstelle, Raum C2P14.
- den Linux-Rechnern im Computer-Pool Wilhelmstraße 106 (UG).

Aufruf von FTP mit

`ftp` **Internetnummer**

oder

`ftp` **Internetadresse**

Beispiele:

- `ftp 134.2.3.48`
- `ftp softserv.zdv.uni-tuebingen.de`

Nun wird man wie beim „gewöhnlichen“ Einloggen nach der User-Id und dem Paßwort gefragt. Möchte man von einem **Softwareserver** Software beziehen, wird man auf diesem in der Regel kein account besitzen. Um dennoch als „Gast“ eine Zugangsberechtigung zu erhalten, akzeptieren die Softwareserver meist eine der folgenden Eingaben:

- login: ftp
  Password: ftp
- login: anonymous
  Password: anonymous

Ansonsten wird das Format der erwünschten Eingabe vom Server explizit angegeben (z.B. wird als Paßwort häufig die eigene E-Mail-Adresse verlangt).

Die wichtigsten FTP-Kommandos

Nach dem Einloggen befindet man sich auf der Kommandoebene von FTP. Hier stehen nicht der gesamte UNIX-Befehlswortschatz, sondern nur einige Befehle zur Dateiverwaltung und zum Datentransport zur Verfügung. Im folgenden sei mit **Arbeitsrechner** der Rechner gemeint, auf dem FTP gestartet wurde, und **Zielrechner** sei der Rechner, mit dem man per FTP die Verbindung hergestellt hat.

`cd, mkdir, rmdir`	Verzeichnisse wechseln, anlegen, löschen auf dem Zielrechner
`lcd` *Verzeichnis*	Verzeichnis auf Arbeitsrechner wechseln
`dir`	zeigt das directory des Zielrechners an
`ldir` (nur auf PC)	zeigt das directory des Arbeitsrechners an
`!Kommando`	aktiviert eine Shell auf dem Arbeitsrechner und führt *Kommando* aus
`put` *Dateiname*	kopiert *Dateiname* auf den Zielrechner
`mput` *Dateiname(n)*	wie `put`, jedoch sind hier die Platzhalter * und ? erlaubt
`get` *Dateiname*	kopiert *Dateiname* vom Zielrechner auf den Arbeitsrechner
`mget` *Dateiname(n)*	entspricht `get` mit Platzhaltern
`ascii`	schaltet auf den für den Transport von ASCII-Dateien notwendigen ASCII-Modus um, Voreinstellung von FTP
`bin`	schaltet auf den für den Transport von binären Daten notwendigen Binärmodus um
`?` oder `help`	zeigt eine Liste der zur Verfügung stehenden Befehle an
`quit`	beendet FTP

Eine Beispielanwendung von FTP

Transport des Quellcodes eines Programmes *perl5.001.tar.gz* im Verzeichnis */pub/gnu* vom Rechner **ftp.uni-tuebingen.de** (Zielrechner) zum lokalen Rechner (Arbeitsrechner)

Einloggen auf dem Soft-Server des ZDV:

```
ftp ftp.uni-tuebingen.de
softserv FTP server (Version wu-2.4(1) Fri Feb 24 11:44:12 MEZ 1995) ready.
Name (ftp.uni-tuebingen.de:zrasl01): ftp
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Suchen und „holen“ eines Files:

```
ftp> cd pub/
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 46
drwxr-sr-x  12 softserv zdv         1024 Aug 21 09:16 .
drwxr-xr-x   9 softserv zdv         1024 Aug 31 17:09 ..
-rw-r--r--   1 softserv zdv          574 May 26 15:33 .message
-rw-r--r--   1 softserv zdv            0 Jul  5  1994 .notar
drwxr-sr-x   5 softserv zdv         1024 Jul 10 16:10 SimTel
drwxr-xr-x   6 softserv zdv          512 Sep 25 17:37 WWW
drwxr-sr-x   6 softserv zdv         1024 Aug  9 11:00 X11
drwxr-sr-x   3 softserv zdv         1024 Feb  7  1995 ZDV
drwxr-sr-x   6 softserv zdv         8192 Oct 20 02:25 gnu
drwxr-sr-x   6 443      zdv         1024 Sep 11 15:14 i386
drwxr-sr-x   9 softserv zdv         1024 Oct  5 17:31 linux
drwxr-sr-x  27 zrasl01  zdv         1024 Sep 20 09:45 parallel
drwx------   6 softserv zdv         1024 Aug  2  1994 sources
drwxr-xr-x  13 softserv zdv         1024 Aug 22 10:45 sw
226 Transfer complete.
ftp> cd gnu
ftp> get perl5.001.tar.gz
200 PORT command successful.
150 Opening BINARY mode data connection for perl5.001.tar.gz (1130765 bytes).
226 Transfer complete.
1130765 bytes received in 7.1 seconds (1.6e+02 Kbytes/s)
ftp> bye
221 Goodbye.
```

Arbeiten mit Disketten

Will man Dateien oder Programme, die man sich per `ftp` beschafft hat, auf Disketten abspeichern und transportieren, stehen dafür zum einen die PC's in der „**Expreßstation**“ des ZDV auf der Morgenstelle zur Verfügung.
Darüberhinaus besitzen manche UNIX-Workstations sowie die Linux-PC's ein Diskettenlaufwerk. Dort steht auch ein eingeschränkter Befehlssatz zum Arbeiten mit MS-DOS-formatierten Disketten zur Verfügung, die sog. **mtools**. Diese haben ihren Namen daher, daß den entsprechenden MS-DOS-Befehlen ein „m“ vorangestellt ist.
Das Standardlaufwerk für die mtools ist das 3.5-Zoll-Diskettenlaufwerk „A:“ . Die wichtigsten Befehle sind:

`mcopy`	Kopieren von DOS nach UNIX oder umgekehrt. Beispiel: *mcopy Brief.txt A:\Brief.txt* kopiert die Datei Brief.txt auf Diskette.
`mdel`	Löschen einer DOS-Datei. Beispiel: *mdel A:\Brief.txt* löscht die Datei Brief.txt von der Diskette.
`mdir`	Anzeigen eines DOS-Verzeichnisses. Beispiel: *mdir A:* zeigt den Inhalt von A:\ an.
`mmd`	Anlegen eines DOS-Verzeichnisses auf der Diskette.
`mrd`	Löschen eines (leeren) DOS-Verzeichnisses.
`mren`	Umbenennen eines (existierenden) DOS-Verzeichnisses.
`mtype`	Zeigt den Inhalt einer DOS-(ASCII-)Datei an.

17 Programmentwicklung

17.1

Allgemeines

Für ProgrammiererInnen existieren auf **UNIX**-Rechnern eine Vielzahl vor Programmiersprachen:
FORTRAN 77, FORTRAN 90, C, C++, Pascal ...

Einzelne Schritte

- Als erstes sollte der/die ProgrammierIn den Quellcode mit einem Editor erstellen. Zu diesem Zweck bieten sich die weiter oben erwähnten Editoren (`vi` oder `emacs`) an.
- Je nach verwendeter Programmiersprache wird jetzt der dazu passende Compiler aufgerufen:

  cc : Auf den meisten **UNIX**-Rechnern wird damit der **C**-Compiler aufgerufen.

  f77 : Damit wird der **FORTRAN 77**-Übersetzer gestartet.

  c++ : Startet den **C++**-Compiler.

  pc : Ruft den **Pascal**-Compiler auf.

  Der Aufruf eines Übersetzers hat folgende Form:

  ```
  compiler [flag]... file ...
  ```

- Während der ganzen Programmentwicklung ist ein Debugger manchmal von großer Hilfe:

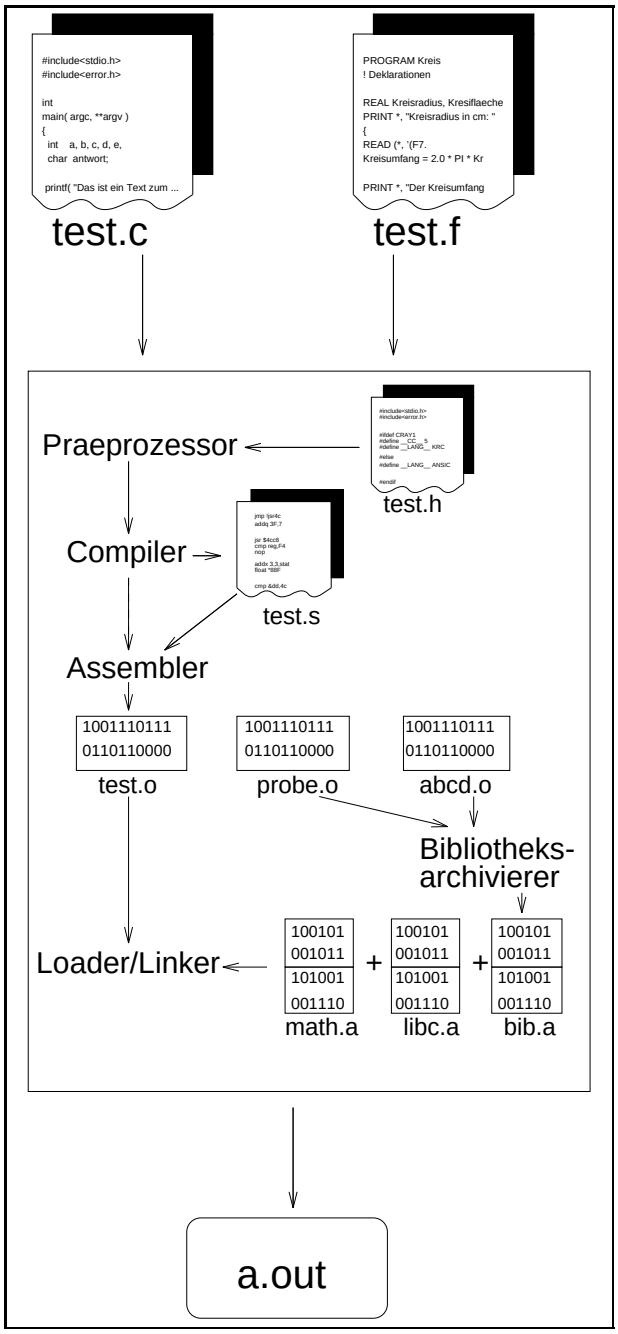
  ```
  gdb programmname
  ```

Wichtige Compilerflags/-schalter

Flags	Erklärungen
-c	Der Quellcode wird nur in **Object**-Code übersetzt und nicht in ein ausführbares Programm
-Dname=def	Definiert das Makro **name**
-E	Nur der Präprozessor wird aufgerufen; die Ausgabe geht zur Standardausgabe
-g	der Compiler erzeugt Symboltabellen; dies ist notwendig, wenn das Programm mit dem Quellcode debuggt (Fehler ausmerzen) werden soll
-Idir	Spezifiziert die Suche nach den Include-Files: Zuerst im Verzeichnis von **file**, dann im Verzeichnis **dir** und schließlich, wenn nicht bisher gefunden, in den Standardpfaden
-lx	Der Compiler verbindet (**linkt**) den in Maschinensprache übersetzten Programmcode mit der Bibliothek *lib***x**.*a*
-Ldir	Ändert die Suche nach Befehlsbibliotheken: zuerst wird im Verzeichnis *dir* gesucht und bei Mißerfolg in den Standardbibliotheken
-o output	Das ausführbare Programm heißt nicht *a.out* sondern *output*
-O[n]	Das Programm wird nach bestimmten Kriterien optimiert. **n** (0-4) gibt die Vorgehensweise beim Optimieren an.

Schaubild

Die einzelnen Schritte der Programmentwicklung, vom Quellcode zum ausführbaren Programm, sind im Schaubild schematisch dargestellt:



Zusätzlich zu den erwähnten Compilern für verschiedene Programmiersprachen existieren weitere Werkzeuge zur Programmentwicklung:

- `make`: Ein Programm, welches die Targetfiles (übersetzte Programme, Bibliotheken etc.) auf dem neuesten Stand hält. Es ist in der Lage, Shellkommandos auszuführen.
- `rcs/sccs`: Ein Werkzeug, das die verschiedenen Versionen des Quellcodes eines Programmes verwaltet (sehr hilfreich bei sehr großen Programmierprojekten).