UNIX
Eine Einführung

Zusammengestellt von Thomas Grün (gruen@cs.uni-sb.de)

Beiträge von Georg Demme, Alexander Fox, Stefan Franziskus,
Thomas Grün, Peter Klein, Martin Wanke

Universität des Saarlandes
FB 14 — Informatik

Inhaltsverzeichnis

Kapitel 1

Grundlagen

1.1 UNIX – ein Multitasking-Betriebssystem

In einem Multitasking-System können mehrere Prozesse (Task, Programm in Ausführung) zur selben Zeit laufen. Für jedes Programm sieht es so aus, als ob es alleine auf dem Rechner liefe. Tatsächlich werden die Programme hintereinander für jeweils eine kurze Zeit bearbeitet. Das Multi-Tasking-Betriebssystem kümmert sich um die Reihenfolge und um die Rechenzeitzuteilung an die einzelnen Prozesse.

Neben der *Rechenzeitvergabe* kümmert sich das Betriebssystem um die Vergabe der Ressourcen *Hauptspeicher* und *Peripheriegeräte*. Kein Benutzerprogramm darf darauf direkt zugreifen. Stattdessen beauftragt es das Betriebssystem über einen Systemaufruf (System call), um die gewünschte Aktion ausführen zu lassen. Gegebenenfalls verweigert das Betriebssystem die Ausführung und meldet einen Fehler.

In einem Multi-Tasking-Betriebssystem ist es notwendig, neue Prozesse zu erzeugen und alte Prozesse zu beenden.

Nach dem Hochfahren des Systems existiert zunächst nur ein Prozeß. Alle anderen Prozesse werden auf folgende Art erzeugt (siehe auch Abbildung 1.1).

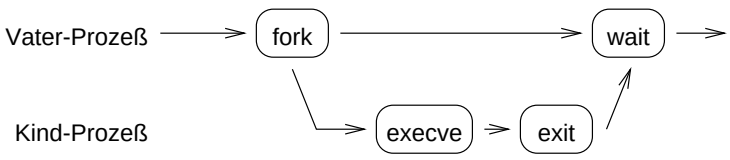


Abbildung 1.1: Prozeßerzeugung

Jeder Prozeß – im folgenden Vater-Prozeß genannt – kann einen neuen Prozeß (den sogenannten Kind-Prozeß) durch den Systemaufruf `fork()` erzeugen. Der Kindprozeß ist eine fast identische Kopie des Vaterprozesses. Die beiden Prozesse unterscheiden sich vor allem durch den Rückgabewert des System Calls: der Kind-Prozeß erhält Null zurück, während der Vater die Prozeßnummer (process identifier, PID) des neu erzeugten Prozesses erhält.

Nach der Prozeß-Erzeugung führt der neue Prozeß in den meisten Fällen einen `exec()`-Systemaufruf durch, der ein Programm von Disk lädt und ausführt. Wenn der Kind-Prozeß seine Aufgabe erfüllt hat, terminiert er mit einem `exit()`–Syscall. Dabei kann es noch einen Rückgabe-Parameter angeben. Dieser Parameter kann vom Vater-Prozeß mit dem Systemaufruf `wait()` abgefragt werden. Dann wartet er bis einer seiner Sohnprozesse terminiert und erhält neben dem Exit-Code noch die PID zurück.

Die Vater-Kind–Relation spannt den Prozeß-Baum auf. Die Wurzel ist der *Init*–Prozeß. Wenn ein Vaterprozeß früher als seine Kinder terminiert, "erbt" der Init-Prozeß diese Waisen.

1.2 Login

Jeder Benutzer in einem Unix-System hat einen *Login Namen* – normalerweise eine Abkürzung des Namens – und ein geheimes Passwort. Wenn er zu arbeiten anfangen will, muß er sich zunächst *einloggen*. Im folgenden Beispiel ist der Login-Name `hugo` und das Passwort `Zw/34!ob`. Es ist sinnvoll, neben Kleinbuchstaben auch große Buchstaben, Zahlen und Sonderzeichen zu benutzen.

Zuerst meldet sich das System mit:

```
login:
```

Der Benutzer gibt 'hugo' ein. Danach sieht der Bildschirm wie folgt aus.

```
login: hugo
```

Anschließend fragt das Betriebssystem nach dem Passwort.

```
login: hugo
password:
```

Wenn der Benutzer das Passwort "Zw/34!ob" eingibt, wird es *nicht* auf dem Bildschirm angezeigt.

Wenn ein falsches Passwort eingegeben wird, erscheint eine Meldung "login failed" und ein neues `login:` erscheint auf dem Bildschirm. Bei einem erfolgreichen Login wird ein Kommandozeileninterpreter (Shell) gestartet, der sich mit `hugo%` meldet.

Bleibt nur noch die Frage, wer den Login-Prozeß startet; es ist der Init-Prozeß, der erste und einzige Prozeß, der auf einem Unix-System vom Betriebssystem selbst erzeugt wird. Nachdem er das "Hochfahren" des Systems veranlaßt hat, hält er nach Terminals Ausschau, die von keinem Benutzerprogramm verwendet werden. Wenn er ein solches Terminal findet startet er das Programm `getty` (get teletype, get terminal) mit diesem Terminal als *Kontroll-Terminal*. Der `getty`-Prozeß wickelt die Login-Prozedur ab und

startet anschließend mit *exec()* ohne vorheriges *fork()* den Kommandozeileninterpreter. Wenn die Prozesse des Kontroll-Terminals — insbesondere also die Shell — beendet sind, beginnt das Spiel von neuem.

1.3 Die Shell — ein Kommandozeilen–Interpreter

Nach der Login-Prozedur wird eine Shell gestartet. Normalerweise ist das "csh" (sprich: ßih-Schell) oder "tcsh". Sie sind komfortabler als die ursprünglich benutzte Bourne-Shell "sh", die auf allen Unix-Systemen zur Verfügung steht. Mittlerweile gibt es viele andere Shells: ksh, zsh, bash, ...

Die Shell ist ein Kommandozeilen-Interpreter (wie `COMMAND.COM` unter DOS), d.h. sie liest eine Zeile, die der Benutzer eingibt, und versucht, sie als Kommando zu interpretieren. Eine solche Zeile besteht normalerweise aus einem Kommando, gefolgt von mehreren Parametern (Argumenten). Es gibt interne Kommandos, die direkt in der Shell eingebaut sind, und externe Kommandos, für die von der Shell eigene Prozesse gestartet werden.

1.3.1 Shell-Variablen

Da die Shell auch als Programmiersprache konzipiert ist, kennt sie Variablen, die mit dem internen Kommando `set` belegt werden können. Die Zeile

```
set path (/usr/bin /usr/etc /home/myname/bin .)
```

weist der Variablen `path` die in Klammern gefaßte Liste von Wörtern zu. Die Variable `path` hat für die Shell eine spezielle Bedeutung. Sie enthält die Pfadnamen, in denen die Shell nach externen Programmen sucht.

Variablen mit einer speziellen Bedeutung sind:

- path – Suchpfad für externe Kommandos
- user – Login Name des Benutzers
- term – Typ des Terminals, an dem der Benutzer arbeitet
- home – das Home-Directory des Benutzers

Diese Shell-Variablen werden automatisch auch als *Umgebungsvariablen* (environment variables, sozusagen öffentliche Variablen) zur Verfügung gestellt, auf die von anderen Programmen zugegriffen werden kann. Auf alle anderen Variablen (sozusagen private Variablen) kann nur die Shell zugreifen. Grundsätzlich werden Environment-Variablen mit `setenv VAR value` erzeugt.

1.3.2 Ein- und Ausgabe–Umlenkung

Jedes Unix-Programm hat drei Kommunikationskanäle, die mit *stdin* (Standardeingabe), *stdout* (Standardausgabe) und *stderr* (Standardfehlerausgabe) bezeichnet werden. Normalerweise sind alle diese Kanäle mit dem Benutzerterminal assoziiert, d.h. über *stdin* werden Eingaben von der Tastatur gelesen und Ausgaben an *stdout* oder *stderr* werden auf den Bildschirm ausgegeben.

Eingabeumlenkung bedeutet, daäs Eingaben statt von der Tastatur aus einer anzugebenden Datei gelesen werden. So bewirkt die Kommandozeile

```
wc < datei.in
```

daß das Programm `wc` (word count) die Anzahle der Zeilen, Wörter und Buchstaben der Datei `datei.in` bestimmt und nicht – wie gewöhnlich – diese Werte für den Text ausgibt, der auf der Tastatur eingegeben wird.

Die Ausgabeumlenkung unterscheidet zwischen *normalen* Ausgaben und *Fehler*ausgaben. Die Variante

```
cat bla.txt test.txt > datei.out
```

hängt die Dateien `bla.txt` und `test.txt` hintereinander (con**cat**enate) und schreibt die Ausgabe in die Datei `datei.out`. Fehlerausgaben, wie beispielsweise die Meldung "Datei bla.txt nicht vorhanden", werden weiterhin auf den Bildschirm ausgegeben, weil *stderr* nicht umgelenkt wurde.

Die Syntax für Umlenkungen, die auch die Fehlerausgabe betreffen, unterscheiden sich von Shell zu Shell. In der häufig verwendeten `csh` oder `tcsh` lautet die Umlenkung für Ausgabe und Fehlerausgabe:

```
cat bla.txt test.txt >& datei.out
```

Neben der Umlenkung in eine Datei, kommt es oft vor, daßs die Ausgabe eines Programmes gleich als Eingabe des näechsten Programmes benutzt werden soll. Dafür gibt es den sogenannten *Pipe–Mechanismus*. Die Kommandozeile

```
cat bla.txt test.txt | wc > datei.out
```

gibt aus wieviele Wörter die beiden spezifizierten Dateien zusammen haben. Bei `csh` und `tcsh` gibt es auch die Variante `|&`.

1.3.3 Prozeß– und Job–Kontrolle

Prozesse können im *Vordergrund* oder im *Hintergrund* gestartet werden. Um sie im Vordergrund zu starten, tippt man einfach ihren Namen und die Parameter. Dann wird das Programm gestartet, läuft bis zum Ende durch, und anschließend meldet sich die Shell mit einer neuen Eingabeaufforderung.

Anders als bei DOS läuft unter Unix ein Programm als Kindprozeß der Shell ab; d.h. bei Programmende wird keine neue Shell gestartet, sondern die alte Shell, die mit `wait()` auf das Ende eines Sohnprozesses gewartet hat, läuft weiter.

Das Programm wird im Hintergrund gestartet, wenn das letzte Zeichen der Kommandozeile ein Und-Zeichen (&, ampersand) ist. Dann wird von der Shell, direkt nachdem das Programm gestartet ist, ein neuer Kommandozeilenpromt geliefert. Das Programm läuft derweil "im Hintergrund" weiter. Das ist ein nützliches Feature für lange laufende Programme, wie beispielsweise Compilerläufe.

Technisch gesehen, läßt die Shell nur nach dem Starten des Kindprozesses den `wait()`-Aufruf weg. Der Benutzer kann gegebenenfalls mit dem Shell-Kommando `wait` auf die Beendigung eines Sohnprozesses warten.

Eine Kommandozeile, die von der Shell gestartet wird, heißt auch *Job*. Ein Job kann aus mehreren Prozessen bestehen, wenn z.B. der Pipe-Mechanismus verwendet wird. Man kann sich alle Jobs mit dem Kommando `jobs` anzeigen lassen; es liefert eine numerierte Liste von Jobs mit Statusinformationen (Vordergrund/Hintergrund, gestoppt/laufend).

Es ist möglich, Jobs im Vordergrund anzuhalten (zu stoppen) indem man die Tastekombination "CTRL–Z" drückt. Danach kann man sie mit `bg` (Background) im Hintergrund weiterlaufen lassen. Einen Hintergrundprozeß mit der Nummer *num* kann man mit dem Kommando `fg %`*num* in den Vordergrund bringen.

Statt Jobs anzuhalten (und später weiterlaufen zu lassen) ist es auch möglich Jobs abzubrechen, indem man die Tastenkombination "CTRL-C" drückt. Dieselbe Wirkung hat das Kommando `kill %`*num*. Wenn dieser Versuch bei hartnäckigen Programmen keine Wirkung zeigt, kann man `kill -9 %job` benutzen, wodurch das betreffende Programm mit Sicherheit abgebrochen wird.

Jobs werden von der Shell zur Vereinfachung der Prozeßhierarchie und –nummerierung eingeführt. Hinter einem Job verbergen sich eine oder mehrere Prozeßnummern (Prozeßidentifier, PID). Die Liste der eigenen Prozesse erhält man über das Kommando `ps`. Der Befehl `kill` läßt sich auch auf Prozesse anwenden; dabei wird das Prozentzeichen weggelassen.

1.3.4 Start und Beendigung der Shell

Wenn eine Shell gestartet wird, sucht sie zuerst nach einer Datei mit dem Namen `.cshrc` (C Shell Run Commands) und interpretiert alle Zeilen der Datei, als ob sie vom Benutzer eingetippt worden wären. Ist die Shell nach den Login gestartet worden, verarbeitet sie *anschließend* die Datei `.login` auf dieselbe Art und Weise, bevor sie dem Benutzer einen Promt liefert.

Beim Ausloggen wird analog eine Datei `.logout` verarbeitet. In ihr können Programme zum Aufräumen eingetragen sein; so werden hier oft automatisch angelegte Backup-Dateien gelöscht.

1.3.5 Implementierung der Shell

Die Shell ist ein gewöhnlicher Benutzerprozeß ohne besondere Rechte. Für jedes externe Programm wird ein Unterprozeß gestartet, so wie es im ersten Abschnitt gezeigt wurde. Der Unterschied zwischen Vorder- und Hintergrundprozessen ist, daß der *wait()*-Systemaufruf von der Shell nur ausgeführt wird, wenn ein Prozeß im Vordergrund läuft.

1.4 Das Dateisystem

Das Dateisystem (file system) ist baumartig strukturiert. Es gibt zwei Unterschiede zu DOS: 1) es gibt keine Laufwerksbuchstaben und 2) sind die Pfadkomponenten durch einen Slash (/) und nicht durch einen Backslash (\) getrennt.

Das Gegenstück zum DOS-Befehl `DIR` ist das Unix-Kommando `ls` (für list directory). Mit `cd dirname` kann man in ein anderes Verzeichnis wechseln. Der Befehl `pwd` (print working directory) zeigt den aktuellen Verzeichnispfad an. Im Gegensatz zu DOS wechselt das Kommando `cd` ohne weitere Parameter ins Benutzer–Home-Directory. Das ist ein dem Benutzer zugeordnetes Verzeichnis, in dem er sich nach dem Einloggen befindet. In diesem "Home-Dir" befinden sich auch die Startup-Skripts eines jeden Benutzers.

1.4.1 Zugriffsberechtigungen und Sicherheit

Unter DOS können Dateinen markiert werden als "Systemdatei", "verborgen" (hidden), etc. Unter Unix wird ein erweitertes Konzept für Datei-Attribute eingesetzt. Dateien, die mit einem Punkt (.) beginnen werden bei einem gewöhnlichen `ls` nicht angezeigt. Für ein Mehrbenutzersystem sind allerdings weitergehende Schutzmechansimen erforderlich.

Sehen wir uns den Eintrag eines Benutzers in der Passwortdatei "/etc/passwd" oder mit `ypcat passwd` an. Er enthält neben dem Login-Namen und dem verschlüsselten

Passwort noch folgende Einträge: einen eindeutigen Benutzer-Identifier (UID, user identifier), die anfängliche Gruppen-Kennung (GID, group identifier), den ausgeschriebenen Name des Benutzers, sein Home-Directory und die Shell, die beim Einloggen gestartet werden soll. Mit dem Kommando `passwd` kann sowohl das verschlüsselte Passwort, als auch (über Optionen) die sonstigen Einträge in der Passwort-Zeile geändert werden.

Die Datei "/etc/group" enthält die Zuordnung von GIDs zu Gruppennamen, und die Liste von Benutzern, die einer Gruppe angehören. Durch das Kommando `newgrp neu` kann ein solcher Benutzer seine aktuelle GID ändern. Das Kommando `groups` zeigt an, welchen Gruppen ein Benutzer angehören kann.

Jeder Datei in einem Unix-System ist *ein* Besitzer (owner) und eine Gruppe (group) zugeordnet. Beim Erzeugen einer Datei wird die UID des Benutzers und die aktuelle GID als Default-Wert benutzt. Später kann zu jedem Zeitpunkt die Besitzerschaft mit `chown newowner filename` (change owner, ändere Benutzer) und die Gruppenzugehörigkeit mit `chgrp newgroup filename` (change group) geändert werden.

Als einzeln setzbare Zugriffsberechtigungen (*access permissions* gibt es `r` für Lesen (reading), `w` für Schreiben (writing) und `x` für das Ausführen (execution) von Dateien. Sie werden unabhängig voneinander für den Besitzer (user), die Gruppe (group) und alle anderen Systembenutzer (other) gesetzt. Die Ausgabe von `ls -lg` verdeutlicht das:

```
drwxr-x---   1 hugo       compiler    1024 Jun 12 15.11 project
-rw-rw-r--   1 hugo       compiler     573 Jun 14 11.03 memo
```

Dabei zeigt der erste Buchstabe den Dateityp an, die nächsten 9 Buchstaben die Zugriffsrechte. Sie sind in drei Gruppen (user, group, others) von je 3 Zeichen (read, write, execute) aufgeteilt. Ein `x` bei einem Verzeichnis zeigt eine Sucherlaubnis (*search permission*) an, d.h. der Inhalt kann mit `ls` angezeigt werden.

Die Zugriffsrechte können mit dem `chmod` Kommando geändert werden. Die Syntax für den Befehl ist

`chmod` *class operation permission* `file` oder

`chmod` *mode* `file`

wobei `class`$\in \{u, g, o\}$ (user, group, others), `operation`$\in \{+, =, -\}$ (add, set, remove) und `permission`$\in \{r, w, x\}$ (read, write, execute). Der Parameter `mode` ist die oktale Repräsentation der neun Zugriffsrechte. Kommen wir zu einigen Beispielen:

`chmod ug=rw file`	Setze Lese– und Schreibberechtigung für Benutzer und Gruppe
`chmod ugo+x file`	Füge die Ausführungsberechtigung für alle Klassen hinzu
`chmod go-w, u=rwx file`	Lösche die Schreibberechtigung für alle Benutzer außer dem Besitzer und setze alle Rechte für den Besitzer
`chmod 644 file`	Setze die Zugriffsrechte auf "`rw-r--r--`"

Wenn Dateien neu angelegt werden, bestimmt die *umask*, eine Maske die im Oktalformat zugewiesen wird, welche Berechtigungen nicht gesetzt werden. Eine weit verbreitete Maske ist "022": sie verhindert, daß Benutzer außer dem Besitzer einer Datei Schreibrechte erhalten.

Kommen wir noch einmal auf die Ausgabe von `ls -lg` zurück. Die zweite Spalte gibt die Anzahl der Links für das File an, d.h. wie viele verschiedene Namen auf diesen Dateiinhalt verweisen; es kann mehrere "Aliase" geben. Danach werden Besitzer und Gruppe der Datei ausgegeben; anschließend die Größe, die Zeit der letzten Änderung und der Name der Datei.

Der erste Buchstabe der Zugriffsberechtigungen beschreibt den Dateityp. Die wichtigsten Typen sind:

- d Directory: Ein Verzeichnis
- – File: Eine gewöhnliche Datei (plain file) oder ein *hard link* zu einer Datei. Ein Hardlink wird mit `ln ursprung linkname` erzeugt. Danach kann auf die Datei unter zwei Namen zugegriffen werden.
- l Symbolic Link: ein symbolischer Verweis (sozusagen ein Pointer) auf einen Dateinamen. Er wird mit `ln -s ursprung linkname` erzeugt. Unter `linkname` muß nicht notwendigerweise eine Datei vorhanden sein.
- b, c Block– und Character Devices: Hinter diesen Dateien verbergen sich Gerätetreiber, die direkt auf Geräte zugreifen. Lese– und Schreibzugriffe auf diese Dateien werden in Zugriffe auf Geräte wie eine Disk-Partition oder ein Terminal abgebildet.

  Diese Dateien befinden sich normalerweise im Verzeichnis `/dev`. Das Kommando `tty` zeigt den Namen der Spezialdatei (special file) für das Terminal an.

1.4.2 Kommandos

Kommandos bestehen aus dem Kommandonamen und gegebenenfalls Parametern (arguments). Bei den Parametern werden üblicherweise zuerst Optionen und dann Textparameter (wie Filenamen) angegeben. Die Optionen verändern das Verhalten der Kommandos und beginnen normalerweise mit eine Minuszeichen (-, dash).

Die nachfolgende Tabelle zeigt einge Kommandos, die das Dateisystem betreffen. Der Übersicht halber werden nur einige, wenige Optionen angegeben.

ls	Zeigt die Filenamen des aktuellen Verzeichnisses an (DOS: dir)
ls subdir	Listet das Unterverzeichnis "subdir" auf
-l	Langformat (mit Zugriffsberechtigung, etc.)
-g	mit Gruppenname
-r	rekursiv (alle Unterverzeichnisse)
-a	alle Dateien (auch mit ˙ beginnend)
cd dir	Wechsle Verzeichnis zu "dir"(wie in DOS)
cd	Wechsle ins Home-Verzeichnis (Vorsicht!!!)
pwd	Zeige Namen des aktuellen Dir. (DOS: cd)
mkdir dir	Lege Verzeichnis "dir" an (DOS: md)
rmdir dir	Lösche Verzeichnis "dir" (DOS: rd)
rm file	Lösche eine Datei (DOS: del)
-r	rekursiv (Vorsichtig benutzen!!!)
-i	interaktiv (erst nachfragen)
-F	force (Löschen erzwingen)
cat f1 f2 ...	gibt hintereinander f1,f2, ...aus
	(DOS: type funktioniert nur mit einer Datei)
-n	setze Zeilennummer vor jede Zeile
more f1 f2 ...	wie `cat`, aber warte auf Tastendruck nach jeder ausgegebenen Seite

1.5 Manual pages

Niemand kann sich alle Kommandos oder Kommando-Optionen merken. Deswegen gibt es neben den gedruckten Referenzbüchern eine "elektronische Version", die sogenannten Manual Pages (wörtlich: Handbuch-Seiten). So zeigt das Kommando

```
man ls
```

die Beschreibung zum Programm `ls` aus dem Referenz-Handbuch auf dem Bildschirm an. Die Beschreibung gibt einen kurzen Überblick über die Funktion des Programms und über die (Unzahl von) Optionen.

Das Referenz-Handbuch hat acht Kapitel in denen viele interessante Dinge zum Unix-System beschrieben sind. Die einzelnen Kapitel umfassen:

1. **User Commands:** (Benutzerkommandos) Kommandos, die jeder Benutzer ausführen darf
2. **System Calls:** (Systemaufrufe) Beschreibung der Syscalls für Programmierer
3. **C Library Functions:** (Bibliotheksfunktionen für die Programmiersprache C)
4. **Devices and Protocols:** (Geräte und Protokolle)
5. **File Formats:** (Dateiformate)

6. **Games:** (Spiele) Mehr oder weniger interessante Sachen, wie Schach, robots, hack, ...
7. **Special:** Alles, was sonst nirgendwo paßt
8. **Maintainance Commands:** Kommandos, die normalerweise für den Systemverwalter interessant sind. Einige davon können aber auch von einem normalen Benutzer eingesetzt werden.

Um zu einem Schlüsselwort alle Einträge in den Manual Pages zu finden, benutzt man:

```
man -k keyword
```

Dadurch wird eine Liste von Kommandos ausgegeben, die das Schlüsselwort enthalten. In einer Zeile werden Kommandoname, Kapitelnummer und eine kurze Beschreibung angezeigt.

Wenn ein Eintrag in mehr als einem Kapitel auftaucht, muß man die Kapitelnummer des entsprechenden Eintrages mit angeben.

```
man chapter entry (z.B. man 2 write)
```

Es ist äußerst empfehlenswert, zu jedem Kapitel die (elektronische) Handbuch-Seite "intro" zu lesen. Sie gibt eine strukturierte Einführung und einen kuzen Überblick mit einer einzeiligen Erklärung über alle Einträge in diesem Kapitel.

1.6 CSH – Erweiterte Funktionalität

Die grundlegenden Aufgaben der CSH wurden bereits in Kapitel 1.3 behandelt. Nun zeigen wir einige Features, die das Arbeiten mit der CSH erleichtern.

Die wichtigste Neuerung der CSH im Vergleich zur SH ist die History-Funktion (history substitution), mit der alte Befehle schnell wieder ausgeführt werden können. Dazu werden die Befehle durchnummeriert. Die Shell merkt sich die letzten `$history` Befehle und schreibt beim Ausloggen die letzten `$histsave` Befehle auf Platte. Diese Befehle werden beim Einloggen neu geladen und stehen dann sofort zur Verfügung.

Wenn die Kommandozeile mit Nummer *Num* wiederholt werden soll, gibt man `!`*num* ein. Wenn man sich nicht mehr an die Nummer erinnert, kann man auch die Anfangsbuchstaben des Kommandos nach dem Ausrufezeichen angeben, das ist recht nützlich bei Edit-Compile-Test–Zyklen.

Das letzte Kommando kann durch `!!` wiederholt werden. Die Argumente des letzten Kommandos können durch `!*` aufgerufen werden, so daß man ein neues Kommando auf einfache Weise mit den Parametern des letzten Kommandos versehen kann.

Wenn Teile einer Kommandozeile falsch geschrieben sind, kann man mit der Sequenz `^alt^neu^` den falschen Teil `alt` durch `neu` ersetzen.

Die TCSH ermöglicht das Aufrufen von alten Befehlszeilen durch die Cursortasten. Auch Aenderungen innerhalb einer Zeile können auf diese Art bequem vorgenommen werden.

Ein anderes interessantes Feature sind *Aliase*. Sie werden definiert durch:

```
alias name 'definition'     (z.B. alias ll 'ls -l')
```

Danach liefert `name` dasselbe Ergebnis wie die ausgeschriebene Definition. Es ist erlaubt, innerhalb der Alias-Definition auch "history substitution" einzusetzen; allerdings muß das Ausrufezeichen durch ein vorangestelltes Backslash-Zeichen "entschärft" (escaped) werden, weil sonst die CSH schon bei der Alias-Definition die History-Substitution vornehmen würde.

Einige Variablen, wie beispielsweise `$history`, dienen nicht nur zum Speichern von Werten, sondern beeinflussen das Verhalten der CSH. So schaltet die Variable `filec` die Kommandozeilen-Ergänzung (command line completion) ein: Wenn man die ESC–Taste (escape) drückt, versucht die CSH das zuletzt eingetippte Wort (normalerweise ein Dateiname) mit der längsten eindeutigen Zeichenkette zu ergänzen. Wenn es mehrere Möglichkeiten zur Ergänzung gibt, können diese mit CTRL-D aufgelistet werden.

1.7 Mail and Messages

Unix bietet eine Möglichkeit, anderen Benutzern Nachrichten zukommen zu lassen: Email (electronic mail, elektronische Post) oder kürzer Mail. Um einem Benutzer *name* zu schreiben ruft man zuerst das Mail-Programm auf:

```
mail name
```

Danach fragt das Programm nach einer Betreffzeile (subj, subject) und einem Verteiler (cc, carbon copy, Durchschläge). Danach wird der Text der Nachricht (mail body) eingegeben. Die Nachricht wird dadurch abgeschickt, daß eine Zeile mit einem einzigen Punkt am Anfang eingegeben wird.

Wenn sich ein Benutzer einloggt wird er gegebenenfalls durch die Meldung "New mail arrived" oder "You have Mail" auf Nachrichten, die sich in seiner Mailbox gesammelt haben, aufmerksam gemacht. Um die Nachrichten zu lesen, benutzt man ebenfalls das Programm `mail`. Es meldet sich mit einer Eingabeaufforderung `&`. Dort können folgende Kommandos eingegeben werden.

- h: (Headers) Zeige den Mail-Header an. Er enthält Informationen über den Absender und den Weg, den die Mail vom Sender zum Empfänger genommen hat. Außerdem wird die Größe der Nachricht in Zeilen und Zeichen, sowie die Betreffzeile ausgegeben.
- z, z-: (Page up, down) Bei vielen Mails kann man seitenweise durch die Mailbox blättern.
- p (number) : (print) Zeigt die Mail mit der entsprechenden Nummer auf dem Bildschirm an. Ohne Nummer wird die aktuelle Mail angezeigt.
- +, -: Lese die nächste oder vorhergehende *verfügbare* Mail
- d (number): (delete) Lösche die entsprechende Mail
- s (number) name: (save) Speichere die Mail unter dem Namen *name*
- m username: (mail) Versenden einer Nachricht an einen anderen Benutzer. Das funktioniert wie beim Kommando `mail user`
- r (number): (reply) Antworten auf eine Mail. Der Empfänger wird automatisch eingetragen. Als Betreff wird "re: alter Betreff" voreingestellt. Es muß also nur noch der Mail-Text eingegeben werden.

Während des Eingebens des Mail-Textes kann man auf verschiedene Kommandos zurückgreifen. Sie beginnen alle mit einem Tilde-Zeichen (~) als erstes Zeichen einer neuen Zeile. Hier ist eine Liste der Kommandos:

- ~. Beendigung einer Nachricht wir mit einem normalen Punkt an der ersten Position einer neuen Zeile.
- ~v Startet den Editor `vi` mit dem bisher getippten Mail-Rumpf als Eingabefile.
- ~r file fügt die Datei *file* an der aktuellen Position ein.
- ~m number fügt die Mail mit der Nummer *number* an der aktuellen Position ein.
- ~? zeigt alle Tilde-Kommandos.

Zum Programm `mail` gehört das Startup-File `.mailrc`. Dort können einige Mail-Einstellungen konfiguriert werden. Nähere Informationen hierzu gibt es wie immer in den Manual-Pages.

Wenn man das Mailprogramm verläßt, werden sämtliche gelesene Nachrichten aus der System-Mailbox entfernt und in die Datei `mbox` im Homedirectory des Benutzers abgelegt. Diese Nachrichten können mit dem Mailprogramm gelesen werden, wenn es mit `mail -f mbox` aufgerufen wird.

1.8 Shell–Scripts

Shell-Skripts sind Dateien, die von einer Shell interpretiert werden. Sie enthalten neben den üblichen Befehlsfolgen auch Kommandos für eine bedingte Ausführung von Programmteilen über `if ... then` und `while`–Konstrukte. Es ist außerdem möglich Argumente an ein Shellskript zu übergeben.

Wir betrachten hier nur die grundlegensten Kommandos, um die Shell-Skripte `.login`, `.logout`, ...zu verstehen. Später werden wir eine mächtigere interpretierte Programmiersprache kennenlernen, die auf den meisten Systemen installiert ist.

Beginnen wir mit einem einfach Beispiel, das direkt in die Eingabeaufforderung der C-Shell eingetippt wird:

```
% touch out
% foreach i (*.c *.h)
? pr $i >> out
? end
```

Der Befehl `touch out` stellt sicher, daß eine Datei namens `out` existiert: wenn es sie noch nicht gab, wird sie als leere Datei angelegt, ansonsten wird das Datum einer bereits vorhandenen Datei auf den aktuellen Wert gesetzt. Das `foreach`- Kommando führt den Teil bis zum passenden `end` mehrmals aus. Dabei nimmt die Variable `$i` für jeden Schleifendurchlauf einen anderen Wert aus der Menge der Dateinamen, die mit `.c` oder `.h` enden, an. Das Kommando `pr` formatiert die in der Variablen `$i` angegebene Datei und hängt die Ausgabe an die datei `out` an.

Das obige Beispiel könnte als Shell-Skript dienen, wenn es in einer Datei enthalten wäre. Es hat sich jedoch eingebürgert für Shell-Scripte nur die Bourne-Shell (`sh`) zu benutzen, die auf allen Unix-Systemen verfügbar ist.

Ein typisches Bourne-Shell–Skript beginnt mit den beiden Zeilen:

```
#!/bin/sh
#
```

Das veranlaßt das Unix-System die Datei als Shell-Skript auszuführen, wenn sie unter ihrem Namen aufgerufen wird.

An das Skript können Kommandozeilen-Parameter übergeben werden, die in die Shell-Variablen `$1, $2`,...(positional parameters) abgebildet werden. Die Anzahl der übergebenen Parameter wird in `$#` gespeichert.

Sehen wir uns nun ein Beispiel für ein Shell-Skript mit einem Parameter an. Es ist ein kleines Telefonregister, das zu einem Namen die entsprechende Telefonnummer herausucht.

```
#!/bin/sh
#
if test $# -lt 1
then
  echo Usage: $cmd name ...
  exit 1
fi
#
grep -i $1 << END_OF_LIST
name number description
The big unknown ???? test entry
Rolf Schmitt 523643 none
Herbert Lustig 46156 PR manager
END_OF_LIST
```

Der erste Abschnitt besteht aus einer Abfrage der Anzahl der übergebenen Parameter. Wenn weniger als (-lt, less than) 1 Argument übergeben wurde, soll eine kurze Gebrauchsanweisung (usage) ausgegeben werden. Das `test` Kommando liefert Null zurück, wenn der nachfolgende Ausdruck wahr ist, ansonsten Eins. Das `if`-Kommando führt den Teil von `then` bis `fi` aus, wenn der Returnwert der Bedingung (hier `test` ...) Null ist.

Im zweiten Teil ist zunächst die Konstruktion eines *inline text* zu erwähnen. Die Zeilen nach "`grep ...`" bis vor "`END_OF_LIST`" werden durch die Anweisung "`<< END_OF_LIST`" zum `grep`-Kommando geschickt, als ob sie auf der Tastatur eingegeben worden wären. Das Programm `grep` (get regular expression) druckt alle Zeilen aus, auf die das eingegebene Suchmuster paßt.

Andere Kommandos für die bedingte Ausführung von Programmteilen sind

- `for (varname) do (command-list) done` führt (command-list) für jeden Parameter in (varname) aus.
- `for (var) in (word1) (word2) ... do (list) done` führt die Schleife für jedes Wort aus.
- `case word in [pattern [| pattern] ) list ;; ] esac`
- `if list then list fi`
- `if list then list else list fi`
- `if list then list elif list then list else list fi`
- `while list do list done`

Es können auch neue Funktionen definiert werden, und zwar mit `name() {list;}`. Parameter können beim Aufruf über eine Liste übergeben und als Positionsparameter innerhalb der Funktion angesprochen werden.

Kapitel 2

Editoren

2.1 Der Editor `vi`

2.1.1 Allgemeines zum `vi`

Der Name **vi** kommt von dem Wort "visual" (engl.: sichtbar). Bevor es den **vi** gab, waren auf Unix–Systemen sogenannte Zeileneditoren gebräuchlich, bei denen man nur innnerhalb einer Zeile Text eingeben und verändern konnte. Der **vi** ermöglichte es, sich den gesamten Text anzusehen und Veränderungen in einer beliebigen Zeile zu machen. Trotz seines Alters spielt der **vi** immer noch eine wichtige Rolle unter den Texteditoren, da er unter Unix zur Standardsoftware gehört, keine graphische Benutzeroberfläche benötigt und ohne Spezieltasten, wie z.B. die Cursortasten, zu bedienen ist. Da der **vi** ein relativ kleines Programm ist (unter 200KB) ist er schnell zu laden und wird deshalb häufig für kleinere Veränderung in Dateien benutzt, wozu man nicht den emacs (über 1MB, viele Konfigurationsdateien) laden möchte.
Der **vi** ist kein Public-Domain Programm und deshalb ist der Sourcecode des **vi** nicht öffentlich zugänglich. Es gibt allerdings **vi** Klone, wie beispielsweise **nvi** und **vi** ähnliche Programme, wie z.B. **elvis** oder der **vim**.

2.1.2 Aufruf des vi

Der vi kann mit folgender Eingabe in einer Shell aufgerufen werden:

```
vi [filename]
```

Wird der **[filename]** nicht angegeben, so öffnet der vi sein Eingabefenster und man kann einen Text eingeben, muß aber beim Verlassen des vi den Text unter einem Namen abspeichern.

Hinweis Wenn der vi in einem Fenster der X Oberfläche aufgerufen wurde, sollte man die Größe dieses Fensters nicht verändern, da der vi sich nicht der neuen Größe anpaßt.

Der vi zeigt beim ersten Öffnen in der untersten Zeile den Status des Files an:

```
".tcshrc", 24 lines, 564 chars
```

Diese Zeile verschwindet beim ersten Tastendruck wieder, kann aber durch drücken von `Ctrl-g` wiederangezeigt werden.

2.1.3 Die Modi des vi

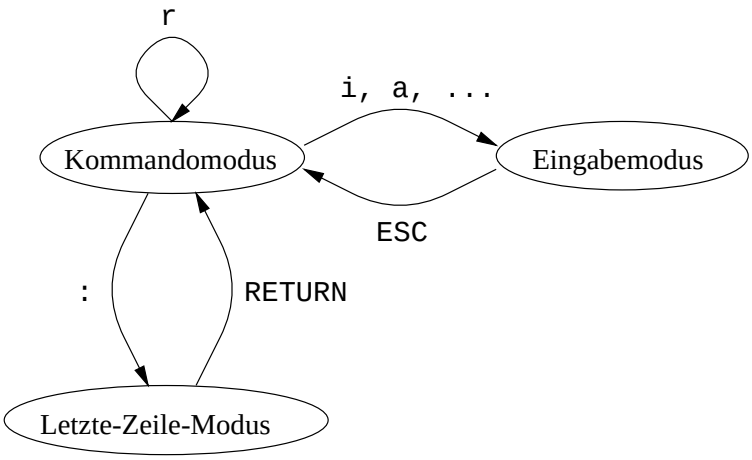
Der `vi` hat mehrere Modi, die per Default dem Benutzer nicht angezeigt werden. Das führt insbesondere bei neuen `vi`–Benutzern oft zur Verwirrung. Es gibt folgende Modi:

Der Kommandomodus *(commandmode)* dient zur Eingabe einfacher Anweisungen aus wenigen Buchstaben, die nicht auf dem Bildschirm angezeigt werden. Man kann jederzeit mit `ESC` in den Kommandomodus wechseln (Nach dem Start des vi ist man im Kommandomodus).
Beispiel:
Die Eingabe von `ESC 10dd` löscht 10 Zeilen. Dabei wird das Kommando `10dd` nicht angezeigt.

Der Letzte-Zeile-Modus *(last-line-mode)* dient zur Eingabe komplexer Kommandos, wie z.B. dem laden einer Datei, die in der letzten Zeile angzeigt werden. Zum Wechsel in den Kommandomodus gibt man `ESC :` ein und jeder Befehl wird mit `RETURN` beendet.
Beispiel:
`ESC :w ~/vi-testfile` speichert die aktuellee Datei unter dem Namen `vi-testfile` im Home–Verzeichnis ab

Der Eingabemodus *(insertmode)* dient der Eingabe oder Veränderung von Text. Es gibt mehrere Kommandos die vom Kommandomodus in diesen Modus führen. Die eingehende Beschreibung dieser Kommandos erfolgt im Abschnitt **Eingeben von Text**.

Notation Befehle, die mit einem : beginnen, sind Befehle des Letzte-Zeile-Modus. Alle anderen sind Befehle des Kommandomodus.



Die verschiedenen Modi des vi.

2.1.4 Bewegen des Cursors

... um einzelne Zeichen

`k` hoch
`h` links `l` rechts
`j` runter

... innerhalb einer Zeile
`^` oder `0` an den Anfang der Zeile
`$` an das Ende der Zeile

... um ganze Seiten
`Ctrl-f` eine ganze Seite nach unten
`Ctrl-b` eine ganze Seite nach oben

... in eine bestimmte Zeile
`<Zeilennummer> G` springt in Zeile `<Zeilennummer>`
Achtung: Groß- und Kleinschreibung bei den Kommandos beachten!

... zu einer Marke
`m<Zeichen>` Marke `<Zeichen>` setzen, wobei `<Zeichen>` ein einzelner Buchstabe ist
`'<Zeichen>` springt zur Marke `<Zeichen>`

Hinweis Hinter der letzten Zeile der Datei zeigt der vi nur noch Zeilen an, die mit einem ~ beginnen.

2.1.5 Eingeben von Text

Wenn man jetzt die richtige Textstelle gefunden hat, an der man seinen Text einfügen will, muß man vom Kommandomodus in den Eingabemodus wechseln. Die zwei häufigsten Befehle hierfür lauten:

- `i` einfügen von Text vor dem momentanen Zeichen *(insert)*
- `a` einfügen von Text nach dem momentanen Zeichen *(append)*

Weitere Kommandos:

- `A` einfügen am Ende der Zeile
- `o` einfügen einer Zeile nach der aktuellen Zeile
- `O` einfügen einer Zeile über der aktuellen Zeile

Kommandos zum Ersetzen von Text:

- `r` ersetzt das Zeichen unter dem Cursor und kehrt danach direkt wieder in den Kommandomodus zurück
- `R` ersetzt eine ganze Zeile
- `s` ersetzt das Zeichen unter dem Cursor und geht dann aber in den Eingabemodus *(substitute)*

Das change Kommando

Das *change* Kommando `c` ist eine komplexere Möglichkeit Text zu ersetzen. Mit ihm kann man ein bis mehrere Objekte, d.h. Zeichenfolgen, wie Buchstaben, Wörter, usw., ersetzen. (Weitere Ojekte sind im Abschnitt **Qualifier** zu finden.)

Syntax

`<Nummer> c object` oder `c <Nummer> object`

`<Nummer>`	die Anzahl der Objekte die berichtigt werden sollen
`c`	der Befehl
`object`	Objekt auf das die Berichtigung angewendet wird

Ein Beispiel:
Text: `The quick brown fxo jumbs over the lazy dog.`
Eingabe: `ESC 2 c w`
Text: `The quick brown fxo jumb$ over the lazy dog.`
Eingabe: `fox jumps`
Text: `The quick brown fox jumps_over the lazy dog.`
Eingabe: `quick`
Text: `The quick brown fox jumps quick_over the lazy dog.`

Was bedeutet `ESC 2 c w`?

ESC	vi wechselt in den Kommandomodus
2	die Korrektur wird auf zwei Objekte angewandt
w	die Korrektur betrifft Wörter

Weitere Objekte sind:

$	Ende der Zeile
f<Zeichen>	von der Cursorposition bis zum ersten Auftreten des Buchstabens <Zeichen>

2.1.6 Löschen von Text

Um einzelne Zeichen zu Löschen wird im Kommandomodus einfach x gedrückt und das Zeichen unter dem Cursor wird entfernt. Um mehrere Zeichen zu löschen benutzt man das d Kommando.

Das d Kommando
Das *delete* Kommando d löscht das angegebene Objekt und speichert es in einem namenlosen Puffer ab.

Syntax

<Nummer> d object oder d <Nummer> object

dw	löscht ein Wort, einschließlich der Leerstelle am Ende
d$	löscht bis zum Ende der Zeile
dd	löscht die gesamte Zeile
"<Zeichen>d object	löscht das angegebene Objekt und speichert es in dem Puffer <Zeichen> <Zeichen> ist ein Buchstabe zwischen a bis z ohne Groß- und Kleinschreibung
d'<Zeichen>	löscht von der aktuellen Cursorposition bis zur Marke <Zeichen>

2.1.7 Kopieren von Text

Der vi bietet die Möglichkeit, Text in Puffer abzulegen, ohne ihn dabei zu löschen, wie das bei dem delete Kommando der Fall ist. Dies geschieht mit dem *yank* Kommando y.

Syntax

<Nummer> y object oder y <Nummer> object

`yw`	kopiert ein Wort, einschließlich der Leerstelle am Ende
`y$`	kopiert alles bis zum Ende der Zeile
`yy`	kopiert die gesamte Zeile
`"<Zeichen>y object`	kopiert das angegebene Objekt und speichert es in dem Puffer `<Zeichen>`
`y'<Zeichen>`	kopiert von der aktuellen Cursorposition bis zur Marke `<Zeichen>`

2.1.8 Qualifier

Die allgemeine Syntax eines vi Kommandos ist:

Syntax

`<Nummer> <Befehl> object` oder `<Befehl> <Nummer> object`

Das `object` bezeichnet man auch als *qualifier*, da er die Wirkungsweise des Kommandos näher bestimmt.
Hier eine Auflistung einiger *qualifier*:

`$`	Zeilenende
`0`	Zeilenanfang
`w`	Anfang des nächsten Wortes
`e`	Ende des momentanen Wortes
`f<Zeichen>`	bis zum ersten Auftreten des `<Zeichen>` in der aktuellen Zeile rechts vom Cursor
`F<Zeichen>`	bis zum ersten Auftreten des `<Zeichen>` in der aktuellen Zeile links vom Cursor
`<Befehl>`	die nochmalige Angabe des Befehls, führt die Aktion auf der gesamten Zeile aus (z.B. `dd` löscht eine ganze Zeile)
`}`	bis Ende des Paragraphen
`)`	bis Ende des Satzes

2.1.9 Texte Einfügen

Einfügen von Dateien
Um eine ganze Datei an der Cursorposition einzufügen, kann man folgendes Kommando benutzen:

`:r <Dateiname>`	die Datei `<Dateiname>` wird an der Cursorposition eingefügt Hinweis: Die Pfadvervollständigung der tcshell funktioniert im vi nicht.

Einfügen von Pufferspeichern
Die Lösch-Operationen aus dem Abschnitt **Löschen von Text** und die Kopier-Operationen aus dem Abschnitt **Kopieren von Text**, schreiben den Text in einen Pufferspeicher, aus dem man sie mit dem *put* Kommando `p` wieder abrufen kann.

`p`	fügt den Pufferinhalt nach dem aktuellen Zeichen ein
`P`	fügt den Pufferinhalt vor das aktuellen Zeichen ein

Die gelöschten Textteile werden vom vi in 9 Puffern abgelegt, die sich wie ein Stack verhalten.

`"<Puffernummer>p`	einfügen eines spezifischen Puffers mit `<Puffernummer>` zwischen 1 bis 9
`.`	abrufen der darauffolgenden Puffer

Beispiel:
Die Befehlsfolge

```
"1p . . . .
```

fügt die Puffer 1 bis 5 wieder ein.

`"<Zeichen>p`	fügt den Puffer mit dem Namen `<Zeichen>` ein

2.1.10 Speichern und Verlassen des vi

Um die Änderungen des Textes zu speichern, wird das erweiterte Kommando `:wq` benutzt. Dieses Kommando ist eigentlich aus zwei Befehlen zusammengesetzt, die beide auch getrennt voneinander benutzt werden können: (Achtung: Das Zusammensetzen von erweiterten Kommandos funktioniert nur in diesem Fall.)

`:w`	speichert den momentanen Text ab
`:q`	beendet den vi

Weitere Kommandos:

`:w <Dateiname>`	speichert den Text unter dem Namen `<Dateiname>` ab (wenn z.B. vi ohne Angabe eines Filenames gestartet wurde)
`:z1,z2w <Dateiname>`	es wird eine Datei `<Dateiname>` angelegt, die den Text von Zeile z1 bis Zeile z2 enthält
`:q!`	beendet den vi, auch wenn der geänderte Text noch nicht abgespeichert wurde

Tip `:wq` kann auch durch `:x` abgekürzt werden.

2.1.11 Suchen von Zeichenfolgen

Um in großen Texten bestimmte Zeichenfolgen zu finden, benutzt man das *search* Kommando `/`. Bei der Suche wird die Groß- und Kleinschreibung beachtet *(case-sensitive)* und wenn das Ende der Datei erreicht ist, beginnt die Suche wieder vom Anfang der Datei *(wrapped)*.

`/<pattern>`	sucht vorwärts
`?<pattern>`	sucht rückwärts
`n`	wiederhole Suche
`N`	wiederhole Suche, aber in umgekehrter Richtung

Bei der Suche kann man auch sogenannte 'magic-keys' angeben:

- `.` beliebiger Buchstabe
  (`/h.h.h.` findet `hahaha`, `hihihi`, `hihahx` ...)
- `*` Null oder mehrere Vorkommen eines Zeichens
  (`/ah*` findet `a`, `ah`, `ahhhhhhhhh` ...)
- `[]` Menge von Buchstaben
  (`/[r-t]at` findet `rat`, `sat`, `tat`)
- `^` Anfang einer Zeile
  (`/^1.Ei` findet jede Zeile die mit `1.Ei` beginnt)
  oder
  Negation einer Menge
  (`/[^a-z ]` findet alles außer `a` bis `z` und `SPACE`)
- `$` Ende einer Zeile
  (`/Ende.$` findet jede Zeile, die mit `Ende.` endet)
- `\` Zeigt die Suche nach einem Sonderzeichen an
  (`/\^2` findet ^2; nicht die Zeile, die mit 2 beginnt)

2.1.12 Ersetzen von Zeichenfolgen

Zeichenfolgen werden mit Hilfe des *substitute* Befehls `s` ersetzt.

`:s/<alte Zeichenfolge>/<neue Zeichenfolge>/`
Ersetzt das erste Vorkommen von `<alte Zeichenfolge>` in der aktuellen Zeile durch `<neue Zeichenfolge>`

`:z1,z2s/<alte Zeichenfolge>/<neue Zeichenfolge>/`
Ersetzt das erste Vorkommen von `<alte Zeichenfolge>` zwischen den Zeilen z1 und z2 durch `<neue Zeichenfolge>`

`:%s/<alte Zeichenfolge>/<neue Zeichenfolge>/`
Ersetzt alle Vorkommen von `<alte Zeichenfolge>` in der gesamten Datei durch `<neue Zeichenfolge>`

`:s/<alte Zeichenfolge>/<neue Zeichenfolge>/g`
Ersetzt alle Vorkommen von `<alte Zeichenfolge>` in der aktuellen Zeile durch `<neue Zeichenfolge>` *(global)*

`:s/<alte Zeichenfolge>/<neue Zeichenfolge>/gc`
Ersetzt alle Vorkommen von `<alte Zeichenfolge>` in der aktuellen Zeile durch `<neue Zeichenfolge>` und fragt bei jeder Ersetzung *(confirm)*

2.1.13 Verschiedenes

Rückgängigmachen von Kommandos

`u`	macht die letzte Änderung rückgängig
`U`	stellt die gesamte Zeile, in der etwas geändert wurde, wieder her

Verbinden zweier Zeilen

`J`	verbindet zwei Zeilen, die durch einen Zeilenumbruch getrennt sind *(join)*

Zwei Dateien im vi

`:e <Dateiname>`	es wird eine zweite Datei `<Dateiname>` geöffnet
`e#`	wechselt zu der vorhergehenden zurück

Bemerkung Die namenlosen delete buffer sind lokal zu dem gerade editierten file, die restlichen global.

Klammerüberprüfung

`%`	wenn der Cursor auf einer der folgenden Klammern ()[]{} steht, springt der vi zu der entsprechenden öffnenden bzw. schließenden Klammer

Shell Kommandos

`:!<command>`	es wird das Shell Kommando `<command>` ausgeführt

Beispiel:

```
:!ls
```

zeigt den Inhalt des aktuellen Verzeichnisses an

Umgebungsvariablen

Der vi benutzt sogenannte Umgebungsvariablen, die die Arbeitsweise des vi spezifizieren.

`:set all`	zeigt alle Umgebungsvariablen an
`:set <name>`	setzt die Umgebungsvariable `<name>`

Beispiele für Umgebungsvariablen:

`showmode`	setzt die Anzeige des Modus
`ts`	tabstops: setzt die Tabulatorweite
`ai`	autoindent: setzt den Indent Modus
`wrapmargin`	setzt den Zeilenumbruch
`ic`	ignore case: macht die Suche nicht case-sensitive
`magic`	magic chars: erlaubt die Benutzung von magic chars
`nu`	numbers: blendet Zeilennummer ein

Um eine Umgebungsvariable nicht zu setzten wird einfach ein no vorangestellt.
Beispiel 1:

```
:set nonu
```

blendet die Zeilennummern aus.
Beispiel 2:

```
:set wrapmargin=10
```

setzt den rechten Rand auf 10 Zeichen (von rechts gezählt).

Kommandos und Textblöcke

`!}sort`	sortiert einen Paragraph

Blöcke: (siehe hierfür auch **Qualifier**)

}	Block bis Ende des Paragraphen
)	Satz als Block
!	aktuelle Zeile
6!	6 Zeilen

Andere Programme sind spell und fmt (Blocksatzformatierung).

Das `mapping` Kommando `map`

Man kann im vi bestimmten Tastenkombinationen Befehlsfolgen zuordnen. Dazu dient das `map` Kommando.

`ESC : map <Zeichenfolge1> <Zeichenfolge2>`

Nach diesem Kommando wird im Kommandomodus bei Eingabe der `<Zeichenfolge1>` die `<Zeichenfolge2>` ausgeführt

`ESC : map! <Zeichenfolge1> <Zeichenfolge2>`

dient dazu Kommandos zu binden, die im Eingabemodus ausführbar sind
Beispiel:

```
ESC : map! ^y dd
```

bindet das Löschen einer Zeile an `C-y`

Das `abbreviation` Kommando `abbr`
Wenn man in seinem Text häufig gleiche Zeichenfolgen benutzt, so kann man diese im vi mit `abbr` abkürzen.
`ESC : abbr <Zeichenfolge1> <Zeichenfolge2>`
wenn im Eingabemodus die `<Zeichenfolge1>` gefolgt von einem `SPACE` eingegeben wird, wird die `<Zeichenfolge1>` durch die `<Zeichenfolge2>` ersetzt

Die vi Konfigurationsdatei
Die Konfigurationsdatei des vi heißt `.exrc`. In ihr werden Umgebungsvariablen gesetzt, map Kommandos ausgeführt und Abkürzungen festgelegt werden.
Bespiel:

```
set nu
set showmode
map! ^ y dd
abbr mfg Mit freundlichen Grüßen
```

2.1.14 Weitere Informationsquellen

- Auf ftp.mines.edu/pub/tutors gibt es den neusten vi-Tutor. Es ist ein Lerntext, mit dem man die grundlegenden Eigenschaften des vi kennenlernen kann.
- SSC staff, SSC reference cards
- Linda Lamb, Learning the vi
- HewlettPackard, The ultimate guide to the vi, Addison-Wesley

2.2 Der Editor `emacs`

2.2.1 Allgemeines zum emacs

Der emacs bietet einen größeren Bedienkomfort als der vi, so verfügt der emacs über Menüsteuerung, über die Möglichkeit mehrere Files gleichzeitig zu bearbeiten und viele Erweiterungen, so daß man mit dem emacs mails verschicken, Hypertext Dokumente lesen und Programme, wie z.B. scheme, ausführen kann.

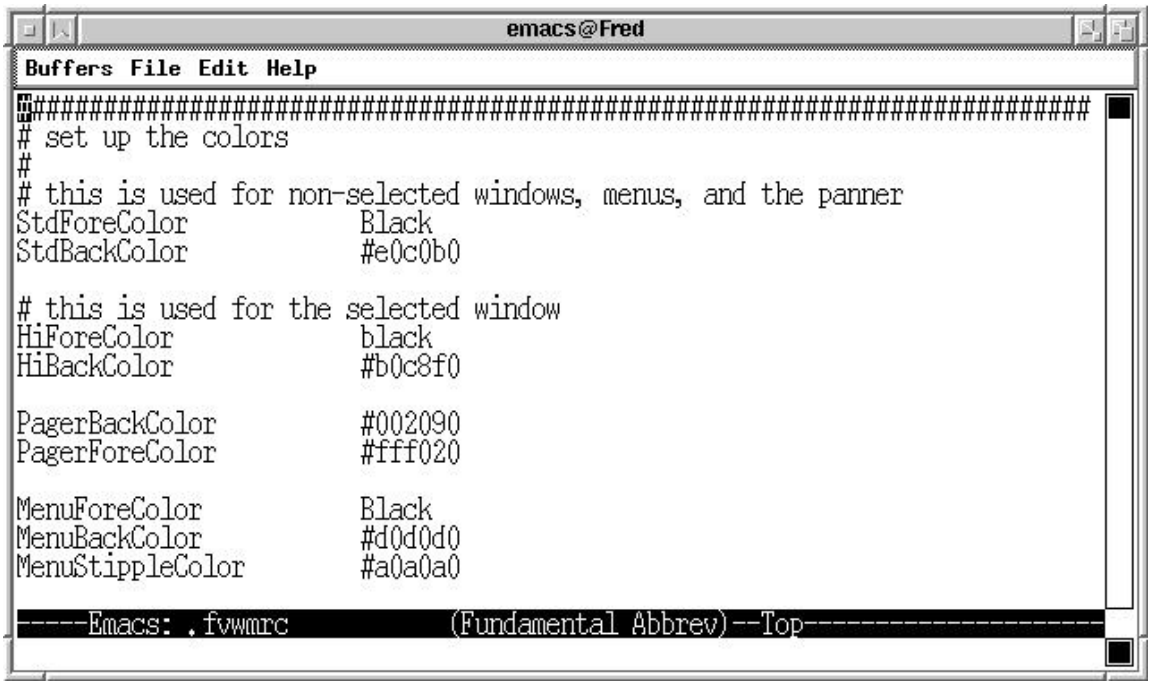
2.2.2 Aufruf des emacs

Der emacs wird aus einer Shell durch folgendes Kommando aufgerufen:

```
emacs [filename]
```

Danach öffnet der emacs sein eigenes Fenster, in dem der Text angezeigt wird.

2.2.3 Aufbau des emacs



Oberfläche des emacs.

Die Menüleiste Einige Funktionen des emacs lassen sich über Maus anwählen. Sie sind relative einfach und selbsterklärend, sodaß hier nur kurz die Tastenkombinationen der Befehle vorgestellt werden.

Das Textfenster Der emacs hat keinen Eingabemodus wie der vi. In diesem Fenster kann immer Text eingegeben werden. Auch Kommandos können jederzeit eingegeben werden.

Der Minibuffer Der Minibuffer dient dazu kurze Meldungen des emacs auszugeben und in ihm werden auch Befehle und Dateinamen, z.B. für das Laden einer Datei, eingegeben.

Die Mode Line dient zur Anzeige wichtiger Informationen:

`-----` zeigt an, ob der Text geändert wurde (`--**-` Text geändert)

`Emacs: <Dateiname>` Name der Datei im Textfenster

Lisp – mode Modus des emacs

Der emacs hat zwei verschiedene Arten von Modi:

den Hauptmodus *(major mode)*

Der Hauptmodus gibt an welche Tasten welche Befehle ausführen und

welche Menüs vorhanden sind.

den Untermodus *(minor mode)*
Die Untermodi dienen als Ergänzung des Hauptmodus und sie können unabhängig vom Hauptmodus gewählt werden. Es können einer, mehrere oder kein Untermodus aktiv sein.
Beispiel: `auto-fill-mode` schaltet den Zeilenumbruch ein.

`21%` Prozentsatz der Datei, die über dem oberen Ende des Fensters liegt.

Der Scrollbar Mit ihm und der Maus kann man sich im Text bewegen.

`LMB` *left mounse button*	bewegen nach unten
`RMB` *right mounse button*	bewegen nach oben
`MMB` *middle mounse button*	Rollen des Inhaltes

Je nachdem, wo auf dem Scrollbar der entsprechende Mausbutton gedrückt wurde, hat dieser eine andere Wirkung. Der Text wird immer um die Anzahl Zeilen bewegt, die der Mauszeiger vom oberen Rand des Scrollsbars entfernt ist.

2.2.4 Wichtige Tastenkombinationen

Bewegen des Cursors
In der Standardkonfiguration des emacs, kann man die Cursortasten nicht verwenden. Stattdessen benutzt man folgende Tastenkombinationen:

`C-p` hoch
`C-b` links `C-f` rechts
`C-n` runter

`C-a`	Anfang der Zeile
`C-e`	Ende der Zeile

Hinweis Diese Kommandos funktionieren auch in der tcshell.

Löschen und Kopieren

`C-k`	löschen bis Zeilenende *(kill)* (auch tcshell)
`C-d`	löschen des Zeichens unter dem Cursor (auch tcshell)
`C-y`	einfügen des gelöschten bzw. kopierten Textes *(yank)* (auch tcshell)
`C-SPACE`	Marke setzen
`C-w`	löscht von Cursorposition bis Marke
`M-w`	kopiert von Cursorposition bis Marke

Text laden und speichern

`C-x C-f`	Datei laden; Im Minibuffer erscheint der Pfad. Er kann mit `TAB` vervollständigt werden. Wenn sich die Datei schon unter den geladenen Buffers befindet, so wird diese angezeigt.
`C-x C-s`	Datei speichern
`C-x C-w`	Datei unter neuem Namen speichern
`C-x i`	Datei an Cursorposition einfügen *(insert)*

Suchen und Ersetzen

Die Suche des emacs ist inkrementell, d.h. während man die zu suchende Zeichenfolge tippt, sucht der emacs schon und zeigt das erste gefundene Wort in Suchrichtung. Nochmaliges drücken der entsprechenden Tastenkombination wiederholt die Suche. Um die Suche zu beenden klickt man mit der linken Maustaste in das Textfenster und der Cursor wird an die Stelle positioniert, auf die geklickt wurde, oder man drückt `C-g` und kehrt an die Stelle zurück, wo die Suche begonnen hat.

`C-s`	vorwärts suchen
`C-r`	rückwärts suchen
`C-x C-x`	zum Ausgangspunkt der Suche zurückkehren
`M-x %`	Ersetzen einer Zeichenfolge

Puffer und Fenster

Mit dem emacs kann man nicht nur mehrere Texte gleichzeitig bearbeiten. Man kann sich auch mehrere Buffers gleichzeitig anzeigen lassen.

`C-x 2`	Teilt aktuelles Fenster horizontal
`C-x o`	Wechselt in ein anderes Fenster *(other)*
`C-x b`	Wechselt im aktuellen Fentser in den zweiten Buffer der Bufferliste

Verschiedenes

`C-x C-c`	emacs beenden, bei Dateien, die noch nicht abgespeichert sind, fragt der emacs, ob er sie noch abspeichern soll
`C-g`	Abbruch von allen Befehlen (auch halb eingetippte)
`C-x u oder C-_`	undo
`C-u <Nummer><Kommando>`	wiederhole Befehl `<Kommando>` `<Nummer>` mal

emacs Info

Der emacs verfügt über ein ausführliches Hilfs- und Informationssystem, das man, während man den emacs benutzt, aufrufen kann. Das Informationssystem ist ein Hypertext–Dokument und um sich in ihm zu bewegen benutzt man folgende Tasten:

`M-x info`	Info aufrufen
`m`	makiert ein Thema
`RETURN`	Thema auswählen
`u`	einen Thema zurück
`p`	vorheriges Fenster
`q`	Info beenden

Die Maus

C-LMB im Textfenster	Bufferliste anzeigen
C-RMB im Textfenster	Font auswählen
LMB auf Mode Line	Wenn mehrere Mode Lines vorhanden sind, lassen sie sich hiermit verschieben
MMB auf Mode Line	Alle Fenster schließen
C-MMB auf Scrollbar	Fenster horizontal teilen
C-MMB auf Mode Line	Fenster vertikal teilen

2.2.5 Verschiedenes

Funktionen

Der emacs hat eine Vielzahl von Funktion, die mit `M-x <Befehl>` aufgerufen und mit `RETURN` beendet werden.

`M-x scheme-mode` Wechselt in den Hauptmodus `scheme-mode`

`M-x line-number-mode` blendet die Zeilennummer ein (Schaltet den Untermodus `line-number-mode` an)

`M-x global-set-key` weist einer Taste oder einer Tastenkombination eine Funktion oder ein Zeichen zu.
Beipiel: Taste: `C-c g`, Funktion: `goto-line`
Nun erscheint beim drücken von `C-c g` im Minibuffer `Goto line:` und man kann die Zeilennummmer angeben, in die der emacs springen soll.

2.2.6 Weitere Informationsquellen

- Der emacs bietet ein grundlegendes Tutorial an, daß man mit `C-h t` aufrufen kann. Es enthält alle wichtigen Funktion und erklärt die Arbeitsweise mit dem emacs.
- Free Software Foundation, GNU emacs reference card
- Ausführliche Dokumentationen über den emacs kann man mit `C-h i` aufrufen. Hier wird alles erklärt, von Kommandozeilenparametern bis zu Konfiguration der .emacs Datei.

Kapitel 3

Programmieren in Perl

3.1 Einleitung

Wie vorher erwähnt, stellt UNIX zur Eingabe eine Oberfläche zur Verfügung, die Shell genannt wird. Diese interprtiert die Eingabe und führt den entsprechenden Befehl aus. Mit Shell-Skripts lassen sich längere Befehlsfolgen abarbeiten. Man kann sie mit MS-DOS Batch-Dateien vergleichen. Der Befehlsfluß läßt ich mittels if- und while,-Anweisungen steuern.
Da es aber viele verschiedene Shells gibt und die Syntax zwischen diesen leicht abweicht, sind diese Shell-Skripts nicht immer kompatibel.
Es ist deshalb besser, gleich eine mächtigere Programmiersprache zu benutzen, die eine feste Syntax besitzt. Hierzu bietet sich Perl (Practical Extraction and Report Language) an, da sie die Funktionalität der Shell und einiger Standard–Tools wie `sed` und `awk` abdeckt, portabel ist und auf den meisten Systemen mittlerweile installiert ist.

3.1.1 Leistungsmerkmale

- leicht erlernbar (wie BASIC)
- gut lesbare Programme
- großer Funktionsumfang
- Portabilität (viele Plattformen)
- schnelle Entwicklungszeit
- Obermenge anderer Interpreter (awk, sed), für die Konverter verfügbar sind.

3.1.2 Verwendung

- Manipulation von Dateien und Verzeichnissen
- Datenreduktion (Mustererkennung)

3.2 Der Perl-Interpreter

3.2.1 Aufruf

- perl -e '<Anweisungen>' (Ausführung der Kommandozeilen–Befehle, execute)
- perl <datei> (Ausführung einer Datei als Perl-Skript)
- `#!/usr/bin/perl` als erste Zeile in die Datei schreiben. Das `#!` bewirkt, daß das Programm in der Datei durch den angegebenen Interpreter abgearbeitet wird. Nachdem man das Execute-Flag der Datei hat, kann das Programm mit dem Dateinamen gestartet werden.

3.2.2 Weitere Command-Line-Options

-c Syntaxprüfung des Skripts ohne Ausführung

-d Ausführung des Skripts mit dem symbolischen Debugger

-w Ausgabe der Warnungen zu möglichen Schreibfehlern und anderen 'verdächtigen' Konstruktionen im Skript

-x Nimmt das Perl-Programm aus dem Eingabestrom (STDIN)

3.3 Ein- und Ausgabe

3.3.1 Ausgabe (print)

Beispiel:

```
#!/usr/bin/perl
print "Hello World\n";
```

Das Beispielprogramm gibt `Hello World` über die Standardausgabe aus.
Die Ausgabe kann von Perl aus auch in eine Datei geleitet werden. Dazu öffnet man die Datei mit `open`, wobei eine spezielle Variable (Dateihandle genannt) mit der zu öffnenden Datei verknüpft wird. Mit Hilfe dieses Handles läßt sich danach bestimmen, wann wann diese Datei für Ein- und Ausgabe verwendet werden soll.

- beliebige Dateihandle werden geöffnet über
  open(*MYFILE, "test"*)
- Ausgabe auf diesen Handle mit
  "print *MYFILE* "Test"

- Per Default: Ausgabe über *Datei-Handle* der Standardausgabe *STDOUT*

Sonderzeichen bei der Ausgabe:

\n	Zeilenvorschub
\r	Absatz
\t	Tabulator
\f	Seitenvorschub
\b	Rückschritt

3.3.2 Eingabe

Die Eingabe wird in Perl durch eine einfache Zuweisung realisiert.

- Eingabe über beliebigen Handle

  Beispiel:
  ```
  #!/usr/bin/perl
  $eingabe = <HANDLE> # Wertübergabe
  print $eingabe;
  ```

- Eingabe über Dateinamen in der Befehlszeile
  - Argumente in der Befehlszeile werden in Vektorvariable ARGV (dazu später mehr) abgespeichert
  - Argumente werden im Beispiel direkt als Dateihandler bei der Eingabe benutzt

  Beispiel.:(cat)
  ```
  #!/usr/bin/perl
  while ($eingabe = <ARGV>){
  #   print $eingabe;
  }
  ```

  Wirkung:
  Der Inhalt aller Dateien, die als Programm-Argument übergeben werden wird auf den Bildschirm ausgegeben.

  Bemerkung:
  `<ARGV>` ist der Handler, mit dem auf jede Datei im Argument zugegriffen werden kann.

Es gibt folgende Anwendungsarten für den Befehl `open`**:**

`open(HANDLE,"> file")`	Erzeugt Datei
`open(HANDLE,">> file")`	Hängt Datei an
`open(HANDLE,"| out-pipe-command")`	Erzeugt Ausgabe-Filter
`open(HANDLE," in-pipe-command|")`	Erzeugt Eingabe-Filter

3.4 Variablen

- Filehandle: MYFILE

  Konvention: Name in Großbuchstaben

- Skalare oder einfache Variablen: $wert

  Konvention: Kleinbuchstaben, wie bei Feldern und assoziativen Arrays.

- Felder: @vector

- Assoziative Arrays: %array

- Bemerkungen

  – Pointer, kompliziertere und selbstdefinierte Datentypen gibt es in Perl nicht. Sie können allerdings mit Hilfe von assoziativen Vektoren konstruiert werden (siehe unten).

  – $v[i] : i. Element des Vektors $v

  – $v[0], $v[$#v] : erstes, bzw. letztes Element

3.5 Ablaufsteuerung

Folgende Hilfsmittel der Flußkontrolle stehen in Perl zur Verfügung. Sie sind sind größtenteils mit denen anderer Programmiersprachen identisch und werden deshalb hier nur am Beispiel erläutert.

- Anweisungen:

  if(EXPR) BLOCK [[elseif (EXPR) BLOCK...] else BLOCK]

  unless (EXPR) BLOCK [else BLOCK]
  gleichbedeutend mit if(!EXPR)

  while (EXPR) BLOCK [continue BLOCK]

  until (EXPR) BLOCK [continue BLOCK]

  for (EXPR;EXPR;EXPR) BLOCK

  foreach VAR (ARRAY) BLOCK

  BLOCK [continue BLOCK]

  goto LABEL

- BLOCK={ <Anweisung 1>; ... <Anweisung n>;}

- Sonderformen:

  do BLOCK while EXPR; do BLOCK until EXPR
  BLOCK if(EXPR)

Beispiel: Student

```
#!/usr/bin/perl
$oldtimer=time(); # sichere alten Timer
$timer=0;
while($timer<95){
 $timer=(time()-$oldtimer)/60;
 # Verbrauchte Zeit in min statt s
 if($timer>90) print {"Go Home";} # Wenn Zeit>90
 elsif($timer<85) {print "Listen";}
 # Wenn Zeit<85
 else {print "Get Ready";} # Sonst
}
continue {print "!\n"}
# Wird in jedem Schleifendurchlauf ausgeführt

Ausgabe:
 85 lange Minuten: Do Nothing!
  5 lange Minuten: Get Ready!
  5 lange Minuten: Go Home!
```

Beispiel: Zähler

```
#!/usr/bin/perl
do{
 unless($e==0){print "größer Null\n";}
 # Wenn nicht $e==0
 else{print "gleich Null\n";} # Sonst
 $e++;
} until($e>=10)

Ausgabe:
 ein mal : gleich Null
 neun mal: größer Null
```

3.6 Unterprogramme

- Unterprogramme werden mit `sub` gekennzeichnet

Beispiel: Maximumberechnung

```
sub MAX{
 local($max);
```

```
 foreach $dummy (@_){
 # Loop über alle Parameter
  $max = $dummy if $max < $dummy;
  # nachgestelltes if möglich
 }
 $max; # Letzte Auswertung ist Rückgabewert
}
```

- Aufruf mit $maximum = &MAX(...) oder mit do MAX(...)
- local erzeugt eine lokale Variable
- Prozedurparameter im speziellen Vektor @_ (siehe unten) übergeben
- (@_) ist Abkürzung für pop(@_)

z.B.: printf &MAX(1,5,3);

3.7 Operationen

Zuweisungsoperatoren	
$a=$b	Zuweisung
$a+=$b	$a=$a+$b
$a-=$b	$a=$a-$b
$a.=$b	$a = $a konkateniert mit $b

Arithmetische Operatoren	
+-*/	Grundrechenarten
%	modulo
**	Potenzieren
++$a,$a++	Autoinkrement
--$a,$a--	Autodekrement
rand($a)	Zufallszahl im Wertebereich 0..$a

Logische Operatoren	
$a && $b	und
$a || $b	oder
!$a	nicht

Zeichenkettenoperatoren	
$a . $b	Konkatenation
$a x $b	$a wird $b mal wiederholt
substr($a,$o,$l)	Teilzeichenkette von $a, ab dem Offset $o, der Länge $l

Vergleichsoperationen	
==,!=,>,>=,<,<=,<=>	Numerische Vergleiche
eq,ne,gt,ge,lt,le,cmp	Zeichenkettenvergleiche

Bemerkung:
<=>, cmp sind lexikalische Vergleiche, d.h. es wird nicht nur wahr oder falsch (0,1) zurückgeliefert, sondern es wird noch angegeben ob eine Zahl größer ist als die andere oder nicht, bzw. ob ein String lexikalisch größer ist als der andere oder nicht. Es gibt also drei Rückgabewerte (0,1,-1).

Dateioperationen	
-r $a	Wahr, wenn Datei $a lesbar
-w $a	Wahr, wenn Datei $a beschreibbar
-d $a	Wahr, wenn $a Verzeichnis ist
-T $a	Wahr, wenn $a Textdatei ist

Beispiel: Verzeichnisbaum

```
#!/usr/bin/perl
# Rekursive Suche nach allen Dateien und
# Subdirectories eines angegebenen
# Verzeichnisses

sub explore{
 local(@files); # Inhalt des Directories
 local($datei); # Eintrag in @files
 local($pd); # Present Directory
 $pd=@_[0]."/"; # eventuell Fehlendes / anhängen
 opendir(DIR,$pd); # Öffne Directory
 @files=readdir(DIR); # Lies D.-Einträge
 closedir(DIR); # Schließe Directory
 shift(@files); # Entferne 1. Eintrag (="./")
 shift(@files); # und 2. Eintrag (="../")
 foreach $datei (@files){
 # durchlaufe alle D.-Einträge
  if(-d $pd.$datei){
  # Wenn Eintrag Directory
   &explore($pd.$datei);
   # Mache rekursiv mit diesem D. weiter
  }
  else{ # sonst
   print $pd,$datei,"\n";
   # Pfad- und Dateiausgabe
  }
 }
}

&explore(@ARGV[0]);
# starte in angegebenen Directory
```

Mustervergleich	
`$a =~ /pat/`	Wahr, wenn $a Suchmuster pat enthält
`$a =~ s/p/r`	Ersetzt in $a alle vorkommenden p durch r
`$aa =~ tr/a-z/A-Z/`	Eingabe in Großbuchstaben umwandeln

3.8 Reguläre Ausdrücke

Suchmuster beim Mustervergleich sind reguläre Ausdrücke (ähnlich vi).

- Folgendes hat spezielle Bedeutung in regulären Ausdrücken:

Ausdruck	Bedeutung
( )	Gruppierung von Ausdrücken
.	Einzelbuchstabe
?	keine oder eine Wiederholung
[...]	Zeichen der Menge
[^...]	Zeichen außerhalb der Menge
^	Zeilenanfang
$	Zeilenende
a | b	a oder b
a{n,m}	a erscheint n-mal aber nicht mehr als m-mal
a{n,}	mindestens n-mal
a{n}	genau n-mal
*	dasselbe, wie {0,}
+	dasselbe, wie {1,}
?	dasselbe, wie {0,1}
\n	Zeilentrenner
\r	Wagenrücklauf
\t	Tabulator
\f	Seitenvorschub
\d	Ziffer (0-9)
\D	Nicht-Ziffer
\w	Wortzeichen
\W	kein Wortzeichen
\s	Zwischenraum
\S	kein Zwischenraum

Beispiel:
`.{80,}` : paßt auf Zeilen mit mindestens 80 Zeichen, d.h. 80 Zeichen lang kein "\n"
`Time: (..):(..):(..)` : paßt auf Zeitangaben
`(t){1,}.*perl` : paßt auf alle Worte mit t anfangen und mit perl enden

3.9 Kurzbeschreibung einiger Perl-Funktionen

- umfassende Beschreibung aller Funktionen in UNIX-Manual-Pages

3.9.1 Arithmetische Funktionen

Es stehen folgende arithmetische Funktionen zur Verfügung:

Funktion	Rückgabewert
atan2(X,Y)	Arcustangens von X/Y
cos(EXPR)	Cosinus in Radiant
exp(EXPR)	e hoch EXPR
int(EXPR)	ganzzahliger Teil
log(EXPR)	Logarithmus (Basis: e)
rand(EXPR)	Zufallszahl
sin(EXPR)	Sinus in Radiant
sqrt(EXPR)	Quadratwurzel
time	vergangene Zeit seit UNIX entwickelt wurde

3.9.2 Umwandlungsfunktionen

Funktion	Rückgabewert
gmtime(EXPR) [get meantime]	Zeit, die von time in einen Vektor ($sec,$min,$hour,...) zurückgeliefert wird (GMT)
hex(EXPR)/ oct(EXPR) sprintf("%lx",$x)	Liefert Dezimalwert des hex/oct-EXPR Liefert Hexadezimalwert von $x
localtime(EXPR)	wie gmtime, allerdings Zeit der richtigen Zeitzone
ord(EXPR)	Numerischer ASCII-Wert des ersten Zeichens von EXPR

3.9.3 Zeichenkettenfunktionen

Funktion	Rückgabewert
chop(LIST)	LIST ohne abschließendes \n
eval(EXPR)	Wert von EXPR wird wie ein kleines Perl-Prg. ausgeführt
length(EXPR)	Länge von EXPR in Zeichen
substr(EXPR, OFFSET[,LEN])	Teilstring von EXPR

3.9.4 Vektor- und Listenfunktionen

Funktion	Aktion
delete $ARRAY KEY	Löscht Wert KEY aus ARRAY und gibt KEY zurück
pop(@ARRAY)	Liefert und Löscht letzten Wert des Vektors
push(@ARRAY,LIST)	Hängt LIST an ARRAY
sort(LIST)	Sortiert LIST
splice(@ARRAY, OFFSET[,LENGTH [,LIST]])	Entfernt Elemente aus Vektor (durch OFFSET und LENGTH angegeben, Ersetzung durch Elemente von LIST)

- splice dient zur universellen Stringmanipulation
- *Beispiel:* `pop(@a)=splice(@a,-1)`

3.9.5 Dateioperationen

Funktion	Aktion
chmod(LIST) [change mode]	ändert Zugriffsrechte auf alle Dateien in LIST
chown(LIST)	wechselt den Eigentümer
mkdir(DIR,MODE)	Erstellt Verzeichnis DIR mir Zugriffsrechten MODE
rmdir(FILENAME)	Löscht Verzeichnis, falls es leer ist

3.9.6 Ein-/Ausgabefunktionen

Funktion	Aktion
binmode(HANDLE)	Zugriff im binären und nicht im ASCII-Mode
close(HANDLE)	Schließt Datei-Handle
open(HANDLE[,FILE])	Verbindet Handle mit Datei
pipe(RHDL,WRHDL)	Gibt Handle-Paar von verbundenen Pipes zurück (zum Lesen und Schreiben)
printf([HANDLE,]LIST)	entspricht print HANDLE sprintf(LIST)

3.9.7 Kommunikation mit dem System

Funktion	Aktion
chdir[(EXPR)]	Wechselt Verzeichnis
exec(LIST)	Führt Systembefehl in LIST aus; keine Rückkehr
fork	Erzeugt neuen Prozeß
system	Ausführung von Unix-Programmen

3.9.8 Funktionen zum Suchen und Ersetzen

Funktion	Aktion
/PATTERN/	Sucht in $_ nach regulärem Ausdruck PATTERN gefundenes Muster in : $& Zeichenkette vor, nach $& : $` , $´

Beispiel: Sucht aus Standardeingabe einen beliebigen regulären Ausdruck

```
#!/usr/bin/perl


while(){
  $_=<STDIN>; # Zu durchsuchender Text
  /@ARGV[0]/;
  # Suche in STDIN nach regulärem Ausdruck ARGV[0]
  print "$`:$&:$´\n";
  # Ausgabe des gefundenen Musters
}
```

Bemerkungen:

- Das Argument mit dem regulärem Ausdruck muß beim Aufruf des Programms in Anführungszeichen gesetzt werden, da es sonst von der Shell interpretiert wird
- $_ belegt mit dem Text der durchsucht werden soll
- In $` und $´: Text vor und nach dem gefundenen Text
- In $&: gefundener Text des reg. Ausdrucks

3.10 Die Perl-Bibliothek (Auszug)

- Perl-Bibliothek enthält viele grundlegenden Programme und Pakete
- zu finden im Pfad @INC (meist /usr/lib/perl)
- Funktionen der Perl-Bibliothek werden zugänglich gemacht, indem am Anfag des Programmes die Zeile: "`require ´filename´`" eingefügt wird
- Aufruf : siehe Aufruf von Unterprogrammen
- erforderliche Parameter, Rückgabewerte, sowie eine Packetbeschreibung sind in der jeweiligen Datei nachlesbar

Dateiname	Aufgabe der Unterprogramme
abbrev.pl	Bildet alle möglichen eindeutigen Abkürzungen einer Liste
bigint.pl	Arithmetik für Integers beliebiger Länge
bigfloat.pl	Arithmetik für Floats beliebiger Länge
bigrat.pl	Arithmetik für beliebig große Bruchzahlen
cacheout.pl	wie open(), aber für beliebig viele geöffnete Dateien
complete.pl	ähnlich tcsh-Readline
ctime.pl	wie ctime aus UNIX C-Bibliothek
dumpvar.pl	Gibt Inhalt der Variablen der aktuellen Packung aus
find.pl	Suche von oben nach unten im Verzeichnisbaum
finddepth.pl	Suche von unten nach oben im Verzeichnisbaum
flush.pl	Leert Ein-/Ausgabepuffer
importenv.pl	Erzeugt Variablen, die die Umgebungswerte enthalten
pwd.pl	Sucht aktuelles Arbeitsverzeichnis
timelocal.pl	Rechnet Datum zurück in Sekunden
validate.pl	Überprüft Datei (Zugriffsrechte)

3.11 Anwendung des assoziativen Vektors

Beispiel: Wort-Statistik

```
#!/usr/bin/perl
$/ = ""; # Setze auf NULL, d.h. Lese Absätze
$* = 1; # Mehrzeiligen Mustervergleich

# Lese Absätze und zerlege in Einzelworte.
# Zähle in %wordcount jedes Auftreten eines
# Wortes.

while($_=<ARGV>){ # aus angegebener Datei einlesen
 tr/A-Z/a-z/;
 # Übersetze Grossbuchstaben in
 # Kleinbuchstaben
 @words = split(/\W*\s+\W*/,$_);
 # Aufteilen des Musters ([.;,:...]<Wort> <Rest>)
 # in (<Wort>,<Rest>)
 foreach $word (@words){
  $wordcount{$word}++;
 }
}


# Ausgabe der Tabelle
foreach $word (sort keys(%wordcount)){
 printf "%20s %d\n",$word,$wordcount {$word};
}
```

Bemerkung:
Den Assoziativen Vektor kann man sich als Tabelle vorstellen. In der ersten Spalte stehen Strings, die als Zugriffsschlüssel verwendet werden und in der zweiten Spalte stehen Werte (Strings oder Zahlenwerte), auf die mit Hilfe des assoziierten Strings der ersten Spalte zugegriffen werden kann. Dieser String in der zweiten Spalte kann ein Verweis auf die erste Spalte sein (Pointer). Damit lassen sich eigene oder rekusive Datentypen konstruieren.

Beispiel: Konstruktion einer einfach verketteten Liste

```
#!/usr/bin/perl
# Aufbau einer einfachen rekursiven Liste

sub listf{
 local($dummy);
 $dummy=pop(@_);
 if($dummy>0){
  $list{$dummy.'.inh'} .= $dummy; # Wert des Listeneintrags
  $list{$dummy.'.next'} .= $dummy-1; # nächstes Element festlegen
  &listf($dummy-1);
 }
 else{
  $list{$dummy.'.next'}.=NULL; # letzter Zeiger gleich NULL
 }
}

if(@ARGV[0]>0) # Aufbau der Liste
{
 &listf(@ARGV[0]); # grösse der Liste über Kommandozeile
}

$i=@ARGV[0];
while($list{$i.'.next'}!=NULL){ # Ausgabe der Liste
 print "nr:$i inh:$list{$i.'.inh'} next:$list{$i.'.next'}\n";
 $i=$list{$i.'.next'};
}
print "nr:$i inh:$list{$i.'.inh'} next:$list{$i.'.next'}\n";
# Letztes Element ausgeben
```

3.12 Zusammenfassung der speziellen Variablen

Da in den verschiedenen Beispielen eine grosse Menge kryptischer Variablennamen verwendet wurden, die aber in Perl für verscheiedene Funktionen (z.B. Mustervergleiche) von grosser Bedeutung sind, folgt jetzt eine Aufstellung dieser Variablen:

`$_`	Variable, auf die Funktionen standardmäßig zugreifen
`@_`	Array mit Unterprogrammparameter
`$0`	Name der Datei aus der das Perl-Programm ausgeführt wird
`@ARGV`	Array mit Befehlzeilenargument
`$ARGV`	Name der aktuellen Datei, Argumente, wenn aus `<>` gelesen wird
`$&`	Variable mit letztem Muster-vergleich übereinstimmender Zeichenkett (siehe Mustervergleichfunktion)
`$´`	Variable, die einem Muster nachfolgt
`` $` ``	Variable, die einem Muster vorausgeht
`%ENV`	Vektor, der die Umgebung enthält
`STDERR`	Error-Handler
`STDIN`	Standard-Eingabe-Handler
`STDOUT`	Standard-Ausgabe-Handler

Kapitel 4

X-Windows

4.1 Allgemeines

4.1.1 Entstehungsgeschichte

Die Entwicklung von X–Windows begann 1984 am MIT Laboratory for Computer Science. Bob Scheifler und Jim Gettys benötigten, unabhängig voneinander, ein gutes Fenstersystem für UNIX–Rechner. So begann die Zusammenarbeit, von Stanford bekamen sie eine Kopie des experimentellen Fenstersystems **W**. Nach einiger Entwicklungszeit wurde das neue Kind **X** genannt.

Immer dann, wenn das System durch Änderungen inkompatibel zu früheren Versionen wurde gab es eine neue Versionsnummer. Mitte 1985 entschied man sich, **X** für jedermann zur Verfügung zu stellen.

Meilensteine:

- Ende 1985: Version 10, zu dieser Zeit begannen Organisationen außerhalb des MIT mitzuarbeiten.
- Januar 1986: DEC kündigt die VAXstation–II/GPX an (erstes kommerzielles X-Produkt).
- November 1986: Version 10 Release 4.
- Januar 1987: Erste X Technical Conference am MIT.
- September 1987: Version 11 Release 1.
- September 1987: Gründung des X–Konsortiums zur weiteren Forschung und Entwicklung von **X**.

- März 1988: Version 11 Release 2.
- Oktober 1988: Version 11 Release 3.
- Januar 1994: Version 11 Release 6 (aktuelle Release).

Dem X–Konsortium gehören mittlerweile über 30 Organisationen an, unter ihnen große Firmen wie DEC, SUN, HP und IBM.

4.1.2 Das Client–Server Modell

Beim X–Windows System unterscheiden wir zwischen dem *X–Server* und den *X–Clients*. Grob gesagt ist der Server ein Programm, das die Fenster anzeigt und die Clients sind Anwendungsprogramme, deren Ausgaben vom Server angezeigt werden. Beim Starten eines Clients wird mit dem Server eine Verbindung aufgebaut. Der Server überprüft dabei zuerst, ob der Client dazu berechtigt ist. Clients, welche nicht hierzu berechtigt sind halten mit einer Fehlermeldung an. Berechtigte Clients schicken ihre Daten an den Server, der sich dann um die Darstellung kümmert. Jede Linie, jeder Punkt und jeder Buchstabe werden nicht von den Clients selbst angezeigt, sondern kommen nur durch Requests der Clients an den Server auf den Bildschirm.

Neben der Ausgabe sammelt der Server Eingabeereignisse (z.B. Tastendruck, Mausbewegungen und -klicks) und gibt sie an die Clients weiter. Die Clients warten auf Events und verarbeiten diese.

Der Server ist der Rechner, an dem man (physisch) arbeitet. Für die Clients spielt es keine Rolle auf welchem Rechner sie gestartet werden. Man muß dem Client nur die Adresse des Servers mitteilen. Die Adresse hat das Format: "`Servername:Displaynummer[.Bildschirmnummer]`"
Beispiel: `cip64.cscip.uni-sb.de:0.0`.
Display– und Bildschirmnummer sind in der Regel 0. Abbildung 4.1 veranschaulicht das Client–Server Modell des X–Windows–Systems.

Der große Vorteil dieses Systems ist, daß man rechenintensive Anwendungen auf entfernten und teueren Groß–Rechnern laufen lassen kann, und die graphische Ausgabe auf einen relativ billigen X–Terminal anschauen kann. So können viele Benutzer gleichzeitig auf einem Groß–Rechner arbeiten ohne dabei unmittelbar physikalisch an diesem Rechner sitzen zu müssen. Umgekehrt kann ein Benutzer unterschiedliche Programme auf verschiedenen Rechnern laufen lassen und an seinem Arbeitplatz anzeigen.

Die Adresse des Servers (z.B. `machtnix:0.0`) kann man auf folgende Arten festlegen:

- Mit dem Kommandozeilenargument `-display`
  z.B. `xterm -display machtnix:0`
- Mit Hilfe der Environment–Variable `DISPLAY`. Falls diese Variable gesetzt ist, versucht automatisch jeder gestartete X-Client sich mit dem in `DISPLAY` angegebenen

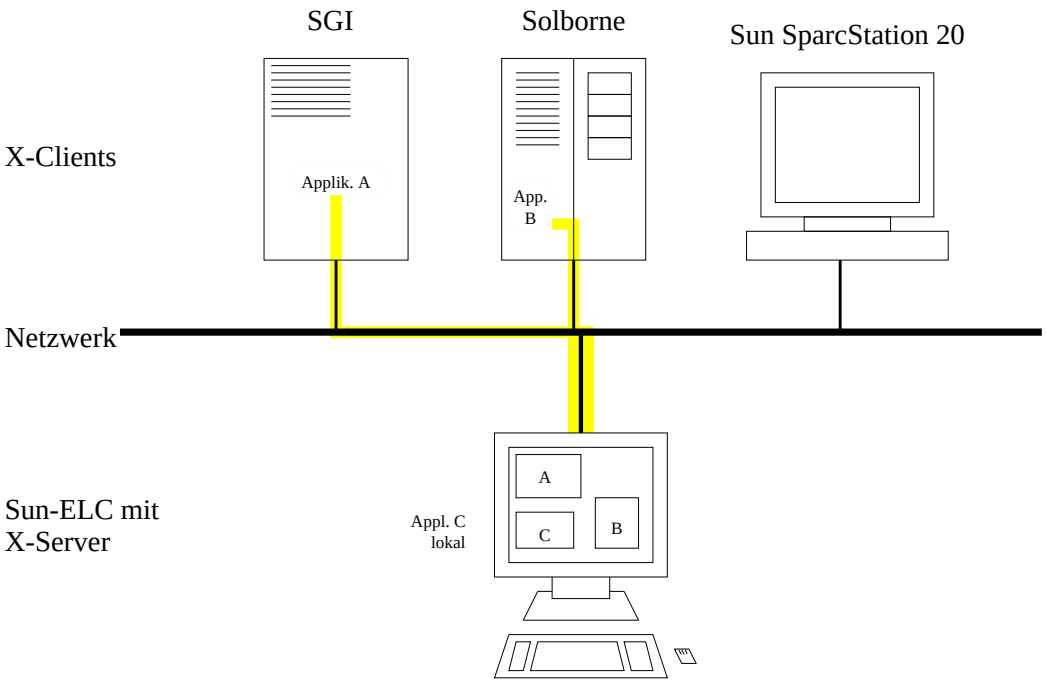


Abbildung 4.1: Das Client–Server Modell

Server zu verbinden.
(z.B. `setenv DISPLAY machtnix:0`).

Die Option `-display` hat hierbei eine höhere Priorität als die Environment–Variable `DISPLAY`. Falls das die Serveradresse nur '`:0`' ist, so werden die Clients auf dem Rechner dargestellt, auf dem sie gestartet wurden.

Bei der tcsh gibt es die Variable `REMOTEHOST`, sie wird automatisch gesetzt wenn man sich auf einem entfernten Rechner einloggt und enthält den Namen des Rechners von dem das Remote–Login ausgeht. Mit Hilfe dieser Variable kann man beim Einloggen die Environmentvariable `DISPLAY` automatisch setzen. Dazu muß man die Datei `.login` um folgende Zeilen erweitern:

```
if ( "`tty`" != "/dev/console" && $TERM == "xterm" ) then
  if ( ! $?REMOTEHOST) then
    setenv REMOTEHOST $HOST
  endif
  if ($?DISPLAY == 0) then
    if ( $REMOTEHOST == $HOST) then
      setenv DISPLAY :0
    else
      setenv DISPLAY ${REMOTEHOST}:0
    endif
  endif
endif
```

4.1.3 Sicherheitsaspekte

Natürlich möchte man nicht, daß jeder beliebige Rechner auf jedem Server Ausgaben anzeigen und Eingaben anfordern kann; dies würde eine große Sicherheitslücke bedeuten. Daher ist immer nur eine bestimmte Gruppe von Rechnern berechtigt, Clients auf einem bestimmten Server anzuzeigen. In dieser Gruppe ist mindestens der Rechner auf welchem der Server läuft und maximal alle am Internet angeschlossenen Rechner. Um Rechner in diese Gruppe aufzunehmen (oder zu entfernen) gibt es das Kommando **xhost**:

- `xhost +`*somehost.domain* nimmt einen Rechner in die Gruppe auf
- `xhost --`*somehost.domain* entfernt einen Rechner
- `xhost +` nimmt alle Rechner des Internets auf
- `xhost --` entfernt alle Rechner aus der Gruppe bis auf den X–Server

Das Kommando xhost wird auf dem zum X–Server gehörigen Rechner aufgerufen. Eine Alternative zu **xhost** ist **xauth** (siehe Man–Pages zu **xauth** und **Xsecurity**).

4.1.4 Konfigurationsdateien

Es gibt mehrere Arten von Konfigurationsdateien um die X–Oberfläche anzupassen:

1. **lokale** (private) Konfigurationsdateien legen die Vorlieben des Benutzers fest, sie beginnen mit einem Punkt und liegen im Home–Verzeichnis des Benutzers (z.b. `.xsession, .fvwmrc`).

2. **globale** Konfigurationsdateien legen Defaults des Systems für alle Benutzer fest. Diese Dateien werden in der Regel nur konsultiert, wenn die entsprechenden **lokalen** Dateien fehlen. Die globalen Konfigurationsdateien stehen normalerweise in `/usr/X11/lib/X11/` oder
`/usr/local/X11/lib/X11/` bzw. Unterverzeichnissen davon.

Die Dateien .xinitrc und .xesssion

Die Dateien `.xinitrc` bzw. `.xsession` werden beim Einloggen oder beim Starten des X–Servers gelesen.
In ihnen werden die Programme festgelegt, die immer nach dem Einloggen gestartet werden sollen. Diese Dateien arbeiten wie Shell–Skripte, jede Zeile enthält ein Programm mit Argumenten. Alle Zeilen, bis auf die letzte, müssen mit einem `&` abgeschlossen werden. Das bedeutet, jedes mit `&` abgeschlossene Programm wird im Hintergrund gestartet und danach wird zur nächsten Zeile übergegangen. Das letzte Programm wird nicht in den Hintergrund gelegt. Beendet man es, so wird auch das Skript, und damit die X–Sitzung, beendet.

Es gibt zwei Möglichkeiten X–Windows zu starten:

- Mit der Eingabe von '`startx`' von der Kommandozeile aus. In diesem Fall werden die Programme in der Datei `.xinitrc` gestartet.
- Über den `xdm`, der eine graphische Oberfläche zum Einloggen bietet. Dabei wird die Datei '`.xsession` gelesen. xdm bedeutet **X D**isplay **M**anager.

Xresource Database (.Xdefaults)

Die Xdefaults legen das Aussehen, Attribute und Verhalten von X–Clients fest. Globale Defaults sind in `/usr/X11/lib/X11/app-defaults` festgelegt, für jeden Client gibt es eine eigene Datei mit dem Namen des Client–Programms, wobei der erste Buchstabe groß geschrieben ist. Lokale Defaults stehen in der Datei `.Xdefaults`

```
clientname[.*]resourcename: value
```

In einem vollständigen Resource–Namen sind die Teile des Namens durch Punkte getrennt. Ein Stern ersetzt beliebige Textteile. Der Resource–Name enthält Angaben zur Klasse und/oder zu einer einzelnen Instanz. Die Benennungen von Klassen bginnen mit einem Großbuchstaben und charakterisieren das Aussehen von übergeordneten Klassen, z.B. von Buttons. Der Instanz–Name wird klein geschrieben und bezieht sich nur auf ein bestimmtes Element der Applikation.

Beispiele für Resource–Angaben in der Datei `.Xdefaults`:

```
XTerm*font:                 7x13
XTerm*background:           honeydew1
XTerm*foreground:           black
XTerm*cursorColor:          deeppink
XTerm*scrollbar*thickness:  10
XTerm*VT100*geometry:       80x30
```

Die Resoucen zu einem Programm sind in den Man–Pages unter RESOURCES aufgeführt. Außerdem sollte man sich einmal die globalen Defaults anzuschauen.

4.1.5 Command–Line Options

Wie in UNIX üblich, können auch Anwendungen unter X–Windows durch Kommandozeilen–Optionen konfiguriert werden. Folgende Parameter gibt es bei (fast) jedem X–Programm:

- `-display` bezeichnet den Server auf dem der Client angezeigt werden soll.
- `-geometry` legt die Größe des Fensters und evtl. die Stelle fest, wo es plaziert werden soll. Format: <*xsize*>x<*ysize*> [+<*xoffset*>+<*yoffset*>]
  z.B. legt `-geometry 320x200+50+50`
- `-fn, -font` die Schriftart für Texte in der Applikation fest.
- `-iconic` bewirkt, daß das nur Vorhandensein eines Fensters angezeigt wird und nicht das Fenster selbst. D.h. anstelle des Fensters liegt ein kleines Icon (Bild), welches das geschlossene Fenster repräsentiert, auf dem Desktop. Durch ein Doppelklick wird dieses Icon dann zum Fenster vergrößert.
- `-title` legt den Titel (Namen) des Fensters fest .

4.1.6 Sonstiges

Mit der Maus können, durch Cut and Paste, Textteile zwischen X–Applikationen ausgetauscht werden. Mit der linken und rechten Maustaste werden Anfang und Ende des Textteils markiert und mit der mittleren Maustaste wird ein vorher markierter Text an der aktuellen Cursorposition “eingeklebt“.
Beim Markieren gibt es folgende Varianten:

- linke Maustaste: Anfang markieren, rechte Maustaste: Ende markieren
- linke Maustaste gedrückt halten und bis zum Ende ziehen

- Doppelklick linke Maustaste: ganzes Wort markieren

- Dreifachklick linke Maustaste: ganze Zeile markieren

Einige Programme, z.B. der tin–Newsreader interpretieren die Maustasten für eigene Zwecke. Cut'n Paste ist dann in der Regel immer noch möglich, wenn man dabei die Shift–Taste gedrückt hält.

Bei einigen Anwendungen (z.B. openwin–Anwendungen) muß für Cut and Paste das **xclipboard** gestartet werden.

Die Manual–Page X(1) bietet eine gute Einführung in das X–Windows–System und enthält Verweise auf Manuals zu Anwendungen unter X11. Folgende Bücher enthalten sehr umfassende Einführungen in das System, sie können in der Informatik–Bibliothek eingesehen werden:

- Valerie Quercia, Tim O'Reilly
  X Window System User's Guide. 3rd Ed.
  Info–Bib Standort: NYE a 90:2

- Niall Mansfield
  Das Benutzerhandbuch zum X-Window System
  MAN n 90:1

4.2 Der Window–Manager

Der Window–Manager verwaltet die Fenster der Oberfläche und verziert diese mit einem Rahmen. Außerdem bietet der Window–Manager Popoups zum Starten von Programmen. Popups sind kleine Menus, welche bei Mausclicks auf den Desktop (Arbeitsfläche des "Schreibtisches") , oder Fensterrahmen aufspringen. Viele Window–Manager bieten sogenannte Virtual Desktops, bei denen der Bildschirm nur einen Ausschnitt des Desktops zeigt. (Größerer Schreibtisch → bessere Ordnung)

4.2.1 Der Window–Manager fvwm

Beim fvwm handelt es sich um einen recht neuen Window–Manager. Er wurde unter anderem entwickelt, um für kleine Rechner einen schnellen Window–Manager mit wenig Speicher–Verbrauch zu haben. Mittlerweile steht er anderen Managern jedoch in punkto Speicher– und Resourcen–Verbrauch um nichts mehr nach. Das globale Konfigurations–File steht in
`/usr/X11/lib/X11/fvwm/system.fvwmrc`. Die lokale Datei heißt `.fvwmrc` und liegt im Home–Verzeichnis des Benutzers. Aufgerufen wird der fvwm mit `fvwm [-f` *config-file*`]`, Üblicherweise sollte dies in der Datei `.xsession` oder `.xinitrc` erfolgen. Es ist sinnvoll den fvwm in diesen Dateien als letztes Programm aufrufen zu lassen,so daß man ihn über Kommando in einem Popup–Menu beenden kann und damit auch X–Windows beendet wird.
Die Konfiguration des fvwm läßt sich in zwei große Teile zerlegen:

1. Konfiguration des Look and Feel. Hierzu gehören Farbeinstellungen, Fensterrahmen, Pixmaps und Schriftarten (Fonts).
2. Konfiguration von Popups, Pulldown–Menues und Modulen. Module sind kleine Programme, die Aufgaben übernehmen, die aus dem Window–Manager ausgelagert wurden.

Zum fvwm gibt es, auf den CsCips, ein kommentiertes `.fvwmrc` im Verzeichnis `/home/stud/stefran/`.
Der fvwm ist (auf den cscips) in
`/usr/local/X11R6/bin` installiert, entweder man fügt `/usr/local/X11R6/bin` zu seinem Pfad hinzu oder ruft den fvwm im `.xsession` mit vollständiger Pfadangabe auf.

Konfiguration des Look & Feel

Bei der Konfiguration des Look & Feel werden nur die wichtigsten Dinge besprochen, Module werden nicht betrachtet.

Zunächst lassen sich für alle Arten von Fenstern **Farben** für Rahmen und Titel festlegen. Dazu werden die Fenster in folgende Klassen unterteilt: **Std, Hi, Sticky, Menu** und **Pager**.

- **Hi** ist das gerade aktive Fenster
- **Sticky** (klebrig) sind Fenster, die beim Wechseln des Virtual Desktop an der gleichen Stelle bleiben, sie kleben sozusagen auf dem Bildschirmglas.
- **Menu** sind Popup–Fenster und Pulldown–Menus
- **Pager** ist das Fenster, das den ganzen virtual Desktop mit allen Fenstern anzeigt. Der auf dem Bildschirm sichtbare Ausschnitt des Desktops wird hervorgehoben dargestellt.
- **Std** sind alle restlichen Fenster

Zu jeder Klasse gibt immer eine ForeColor (Schrift) und eine BackColor (Hintergrund), z.B. StdForeColor. Die Farbe ändert man mit `Farbklasse Farbe` (z.B. HiForeColor yellow). Die Namen aller Farben des Systems stehen üblicherweise in `/usr/X11/lib/X11/rgb.txt`. Auf einigen Systemen gibt es das Programm **xcolor** um die Farben mit ihrem Namen tabellenartig anzuzeigen.

Außerdem kann man die Schriftarten (`Fonts`) der Fenster und Popups konfigurieren. Auch hier gibt es wieder mehrere Klassen: **Font** für Menues, **WindowFont** für Fenstertitel, **IconFont** für Icons und **PagerFont** für den Pager. Fonts werden über `FontKlasse FontName` angegeben. Mit dem Programm **xfontsel** kann man Fonts komfortabel aussuchen.

Als letztes gibt es noch **StyleOptions** oder **Styles**, dies sind Angaben zum Aussehen und Verhalten von Fensterrahmen, die vom Window–Manager erzeugt werden sollen. Sie werden wie folgt festgelegt:

`Style` *"PrgName" StyleOptionList*

Es gibt folgende Styles:

- **NoTitle** für Fenster ohne Titelleiste
- **NoBorder** erzeugt Fenster ohne Rahmen
- **WindowListSkip** listet das Fenster nicht in der Fensterliste des Managers auf

- **StaysOnTop** besagt, daß das Fenster nicht von anderen Fenstern verdeckt werden soll
- **Sticky** besagt, daß dieses Fenster auch beim Verschieben des virtuellen Desktops an der gleichen Stelle auf dem Bildschirm bleiben soll.

Beispiele zu diesen Optionen stehen im Verzeichnis `/home/stud/stefran` in der Datei `.fvwmrc`.

Konfiguration von Popups und Menues

Wie bereits erwähnt, kann man Applikationen im fvwm durch frei konfigurierbare Popup–Menues starten, welche sich auch frei konfigurieren lassen. Außerdem kann man über diese Popups Kommandos an den Window–Manager geben (z.B. **Move** zum Verschieben von Fenstern). Diese Menues werden im fvwm folgendermaßen deklariert:

```
Popup "PopupName"
Title "This is a Popup"
Exec "Application 1" app_name1 args &
...
Exec "Application n" app_namen args &
Nop ""
Popup "Other Popup" Popup2
EndPopup
```

- **Exec** gibt ein zu startendes Programm an
- **Popup** spezifiziert ein Untermenu
- **Nop** fügt eine Leerzeile ein
- zu den eingebauten Kommandos des WM gehören z.B. **Move, Resize, Lower, Raise ...** genaueres findet man in den Manual–Pages und im .fvwmrc

Die so definierten Popups können durch verschiedene Ereignisse aktiviert werden, beispielsweise durch Mausklick auf den Desktop. Ereignisse werden nach folgendem Format deklariert:

Button Context Modifier Function

- **Button** Kann entweder eine Maustaste oder eine Tastatureingabe sein, z.B. `Mouse 1` oder `Key F7`.

- der **Context** gibt an wo sich die Maus gerade befindet, z.B. **R** für das Root–Window, **I** für ein iconifiziertes Fenster).
- **Modifier** sind Tasten, die man zusätzlich zum Button gedrückt halten muß, um die Aktion auszulösen, z.B. **C** für Control, **S** für Shift, **A** für alle Modifier und **N** (none) für keinen Modifier.
- **Funktionen** werden ähnlich wie in Popups angegeben, z.B. `Popup PopupName` Auch hier können die eingebauten Funktionen benutzt werden, z.B. Iconify, Maximize.

Beispiel:

```
#     Button    Context Modifi  Function
Mouse 1         R       A       PopUp "Aremorika"
Mouse 2         R       A       WindowList 1 3
Mouse 3         R       A       Popup "LongRootMenu"

Mouse 1         1       A       Function "Iconify-or-kill"
Mouse 2         1       A       Stick
Mouse 3         1       A       PopUp "Window Ops"
Mouse 0         2       A       Resize

Key F1          A       M       Popup "Window Ops"
Key F2          A       M       Popup "Utilities"
Key F1          A       S       Delete
Key F2          A       S       Destroy
```

4.2.2 Andere Window–Manager: twm & tvtwm

Der **twm** oder **tvtwm** ist ein älterer Window–Manager, er ist bei jeder Standard–Installation von X vorhanden und ist in der Regel als Default–WM eingestellt. So hat man auch auf den CsCips diesen WM, wenn man einen neuen Account bekommt. Das globale Konfigurations–File liegt in
`/usr/X11/lib/X11/twm/system.twmrc`. Die entsprechende private Datei des Benutzers liegt in dessen Home–Verzeichnis und heißt `.twmrc`. Wer den `twm` oder `tvtwm` nach seinem persönlichen Geschmack einrichten möchte, muß sich die globale Datei kopieren und konfigurieren. Die verfügbaren Kommandos zur Konfiguration sind in den Manual–Pages dokumentiert.

Um von `t(vt)wm` auf den `fvwm` umzusteigen, löscht man die Zeile im `.xsession`, in welcher der `t(vt)wm` aufgerufen wird und ersetzt diese durch den `fvwm` (Auf richtige Pfadangabe achten!).

4.3 Anwendungen

In den folgenden Abschnitten beschreiben wir kurz einige wichtige und häufig benötigte Anwendungen, wie beispielsweise das X–Terminalprogramm oder einige Hilfsprogramme um X–Einstellungen festzulegen. Wir betrachten dazu nur die Grundfunktionalität und die wichtigsten Optionen. Eine ausführliche Liste findet sich in der Manual–Page von X.

xterm

xterm ist ein vt102–kompatibeler Terminal Emulator für X–Windows. Damit kann man Programme, die Ausgaben auf einem Textbildschrirm vornehmen in einem X–Fenster ablaufen lassen. Die Schriftgröße und diverse andere Einstellungen des xterms kann man über Popup–Menus einstellen. Sie erreicht man über Ctrl–Maustaste.
Für xterm gibt es folgende Kommandozeilen–Parameter:

- `-e` *programm* spezifiziert das Programm, das unter xterm ausgeführt werden soll (normalerweise läuft eine Shell).
- `-sb`: xterm wird mit Scrollbar gestartet. Mit dieser Scrollbar kann man sich die Zeilen anschauen, die bereits über den oberen Fensterrand das xterm verlassen haben. `-sl` *Nummer* gibt die Anzahl der zu speichernden Zeilen an, die mit der Scrollbar noch zu erreichen sind, der Default ist 64.

xconsole

xconsole zeigt Meldungen der System–Console in einem X–Fenster an. Zu solchen Meldungen gehört z.B. eine Shutdown–Message des Operators. Falls keine **xconsole** läuft, werden die Meldungen im Text–Modus über die X–Oberfläche geschrieben.

xclock, oclock

xclock bzw. **oclock** zeigen die aktuelle Uhrzeit an. **oclock** verbraucht weniger Resourcen als **xclock**. Das Programm **xclock** kann die Uhrzeit sowohl digital (Option: -digital) als auch analog (Option: -analog) anzeigen, **oclock** hat nur eine analoge Anzeige.

xbiff, xmail

xbiff zeigt einen typischen amerikanischen Hausbriefkasten (Mailbox) an. An der Form des Briefkastens kann man erkennen, ob man Post hat oder nicht. Sobald neue Mail

ankommt meldet **xbiff** dies akustisch und optisch.

Das Programm **xmail** ist ein X-Windows basierter Mail–Reader. In iconifizierter Form zeigt **xmail** (wie xbiff) an ob Post da ist oder nicht.

xmeter

xmeter stellt die Auslastung verschiedener System–Resourcen graphisch dar. Es lassen sich viele System–Daten darstellen, wie z.B. die Last (Load) des Systems, IO– und Disk–Aktivitäten, Netz–Pakete, Interrupts u.s.w. Mit der Option **-update secs** läßt sich die Zeit zwischen zwei Aktualisierungen der Anzeige angeben. Das Programm **xmeter** erlaubt es auch fremde Rechner zu überwachen. Mehrere Zustände lassen sich gleichzeitig anzeigen und können in mehreren Spalten oder Zeilen angeordnet werden.

Bsp: `xmeter-update 2 -cols 2 -cpu ‘uname -n‘ -load ‘uname -n‘`
zeigt Last und CPU des eigenen Rechners an. Die Option **-sn** bewirkt, daß xmeter nur die kurzen Hostnamen, ohne Angabe der Domain anzeigt, also cip55, statt cip55.cscip.uni-sb.de.

xman

xman ist Manual–Browser für X–Windows. Das Hauptfenster besteht aus drei Buttons; **Help** zeigt einen Hilfetext zu xman an, **Quit** beendet xman. Durch Anwählen des Buttons **Manual Page** erhält man ein neues Fenster zur Anzeige von Man–Pages, danach wählt man im Pulldown–Menu **Sections** die gewünschte Manual–Section und erhält alle Manual–Pages zu dieser Section. Durch Anklicken eines Eintrages wird die entsprechende Man–Page angezeigt. Durch weiteres Anklicken des Buttons **Manual Page** erhält man ein weiteres Fenster. Das ist nützlich, um mehrere Manuals gleichzeitig zu lesen.

xsetroot

Mit **xsetroot** lassen sich verschiedene Parameter des Desktops einstellen. Beispielsweise kann man die Hintergrundfarbe oder den Maus–Cursor ändern. Es gibt folgende Optionen:

- `-solid` *color* bewirkt, daß der Hintergrund mit der Farbe *color* ausgefüllt wird.
- `-bitmap` *filename* spezifiziert ein Muster, mit dem der Hintergrund ausgefüllt werden soll.
- `-cursor` *cusor-file mask-file* geben das Cursor– und Masken–File für den Maus–Pointer an (bitmaps). Bevor man sich daran gewöhnt hat wie die Maske funktio-

niert, sollte man das Masken–Bitmap ganz schwarz lassen (aus der Man–Page).

bitmap

Mit dem **bitmap**–Programm lassen sich Bitmap–Dateien erstellen und editieren. **Bitmaps** sind s/w Bilder mit einem Bit pro Pixel. Im Gegensatz dazu gibt es auch **Pixmaps** (farbig, 8 Bit pro Pixel). Bitmaps und Pixmaps werden oft für Icons des Window–Managers und für Hintergrund–Muster benutzt. Mit **bitmap** lassen sich folgende Funktionen ausführen:

- Ändern der Größe den Bitmaps (Menu File:Resize)
- Setzen und löschen von Einzelpunkten, zeichnen von Kreisen, Rechtecken, Linien und freien Kurven
- Falten, drehen, spiegeln und invertieren des ganzen Bildes

xlock

xlock sperrt den Arbeitsplatz und startet einen Bildschirmschoner. Um wieder an der Console arbeiten zu können muß man sein Password eingeben. Die Option `-nice` *level* bewirkt, daß der Schoner mit einer geringeren Priorität läuft, dadurch laufen andere Prozesse schneller. Die Art des Bildschirmschoners läßt sich mit `-mode` *name* einstellen. Man sollte die Console nur für kurze Zeit sperren. Wenn man länger wegbleibt (Vorlesung), sollte man sich ausloggen

xset

mit **xset** lassen sich verschiedene Einstellungen des X–Servers manipulieren, beispielsweise Mausbeschleunigung und Bildschirmschoner.

- `xset [-]led` *num* schaltet die Led *num* auf der Tastatur an bzw. aus
- `xset m [`*accel* `[`*thr*`]]; xset m default` stellt die Mausbeschleunigung ein
- `xset s [`*cmd*`]` manipuliert Eigenschaften des Screen–Savers. Es gibt folgende Kommandos:
  - `[`*timeout* `[`*cycle*`]]` Setzt das Timeout für den Saver
  - `default` restauriert die Standardwerte
  - `on/off` schaltet die Saver–Option an/aus
  - `activate` schaltet den Saver sofort ein

Bei der Eingabe von `xset` ohne Parameter erhält man eine Übersicht mit allen Optionen

xmodmap

Oft wünscht man sich auf der guten amerikanischen Sun–Tastatur wenigstens die deutschen Umlaute. Dies kann man mit dem Programm **xmodmap** erreichen. Mit **xmodmap** lassen sich die Keycodes der Tastatur ummappen, insbesondere kann man z.B. auch Modifier wie Ctrl und CapsLock vertauschen. Dazu gibt man die nötigen Befehle entweder in einer Datei oder auf der Kommandozeile an. Die Syntax von **xmodmap** ist etwas spartanisch, daher gibt es das Programm **xkeycaps**; hiermit lassen sich die Codes der Tasten interaktiv ändern. Wenn die Einstellung stimmt, kann man sich die nötigen Befehle für **xmodmap** ausgeben lassen. **xkeycaps** liegt auf den CsCips in `/usr/local/X11R6/bin`.

Folgende Datei belegt die $< Alt > - < Vokal >$ mit Umlauten und definiert einige andere Tasten (scharfes S, Copyright–Zeichen...):

```
keycode 26 = Mode_switch
add mod5 = Mode_switch
keysym a = a A Adiaeresis
keysym e = e E Ediaeresis
keysym i = i I Idiaeresis
keysym o = o O Odiaeresis
keysym u = u U Udiaeresis
keysym s = s S ssharp
keysym 2 = 2 at twosuperior
keysym 3 = 3 numbersign threesuperior
keysym m = m M mu
keysym c = c C copyright
keysym r = r R registered
keysym t = t T trademark
```

Kapitel 5

Netzwerk

5.1 Einführung in das Internet

Das Internet wurde vor 20 Jahren als ein US Defense Department Network, ARPANet genannt, geboren. Es war ein experimentelles Netzwerk zur Unterstützung militärischer Forschung, u. a. dem Aufbau eines Netzwerkes, das selbst nach einer Katastrophe noch intakt ist. Die Kommunikation lief nur zwischen dem Quell- und dem Zielcomputer ab, die für die Übertragung einer Botschaft selbst verantwortlich waren. Beide Rechner waren somit im Netz gleichgestellt. Die Botschaft wurde für den Transfer in einen Umschlag (Internet Protocol (IP) packet) eingepackt und dann versendet. Das Netzwerk wurde als unzuverlässig angesehen, da jeder Teil zu einem beliebigen Zeitpunkt aus dem Netz treten kann. Während noch von der ISO Standards für die Kommunikation entwickelt wurden, wurde die bereits bestehende IP-Software für verschiedene Computertypen umgesetzt und verwendet. Nach und nach entstanden andere lokale Netzwerke (local area networks, LAN, oft auf Basis von Ethernet), die sich auch ans ARPANet anschließen wollten. Ein wichtigstes darunter war NSFNet von der National Science Foundation, die in den 80ern Jahren fünf Supercomputerzentren baute und Wissenschaftlern gestattete, diese Quellen zu nutzen. Um Kommunikationsprobleme zwischen den Zentren zu beseitigen, war ein Anschluß ans ARPANet geplant, dieser schlug aber aus Bürokratiegründen fehl. Aus diesem Grund baute NSF ein eigenes Netzwerk basierend auf ARPANet's IP-Technik auf, das über Telefonanschlüsse verbunden wurde. Da die direkte Verbindung der einzelnen Universitäten über Telefon zu teuer gewesen wäre, baute man regionale Netzwerke auf, die immer mit dem nächsten Nachbarn verbunden waren. Am Ende einer solchen Kette stand dann ein Supercomputer, wobei die Supercomputer untereinander direkt verbunden waren. Dieses Konzept genügte für eine gewisse Zeit. Doch durch stetiges Ansteigen des Netzwerkverkehrs reichte bald die Übertragungskapazität der Telefonleitungen und die Geschwindigkeit der Computer nicht mehr aus, und sie mußten durch schnellere ersetzt werden. Dieser Trend setzt sich bis heute fort.

5.2 Rechneradressen

Jeder Rechner im Internet hat eine sogenannte **IP-Adresse**, die 4 Byte lang ist und in der Form *Byte1.Byte2.Byte3.Byte4* notiert wird; jedes Byte wird dabei als Zahl zwischen 0 und 255 dargestellt, beispielsweise ist 134.96.4.91 die Adresse eines Rechners in Saarbrücken.

Da es schwierig ist, sich solche Adressen zu merken, haben die meisten Rechner zusätzlich einen **Hostname**, der die Funktion und den Standort des Rechners beschreibt, z. B. www.uni-sb.de. Die Adresse ist von hinten nach vorne zu lesen; den letzten Teil der Adresse bildet das Kürzel für das Land (*domain*), in dem der Rechner zu finden ist, den vorletzten die sogenannte *sub-domain* usw.

Die Domains sind unterteilt in Kürzel für die USA und den Rest der Welt:

Die folgenden Kürzel wurden ursprünglich in den USA verwendet, später dann auch für große nicht-amerikanische Einrichtungen:

.com	Kommerzielles Unternehmen *(commercial)*
.edu	Universitäten etc. *(educational)*
.gov	Regierung *(government)*
.mil	Militärische Einrichtung *(military)*
.net	Netzorganisation *(net)*
.org	Nichtkommerzielles Unternehmen *(organisation)*

Die wichtigsten Länderkennungen außerhalb der USA sind:

Kürzel:	**Land:**
.at	Österreich
.au	Australien
.ca	Kananda
.ch	Schweiz
.de	Deutschland
.fi	Finnland
.fr	Frankreich

Die Übersetzung der Hostnames in IP-Adressen wird von speziellen Rechnern, sogenannten **Nameservern** übernommen.

Üblicherweise verwendet man Hostnames, weil sich so Benutzer nur den Namen eines Rechners, auf dem bestimmte Programme laufen, merken müssen und der Betreiber dieses Rechners durch Umkonfigurieren der Nameserver einen Ersatzrechner spezifizieren kann.

5.3 Remote Login

Remote Login beschreibt den Vorgang, sich auf einem anderen, möglicherweise weit entfernten Rechner „einzuloggen“. Prinzipiell gibt es zwei Programme dafür: Telnet und Rlogin.

5.3.1 Telnet

Telnet bezeichnet einerseits das sogennante Telnet-Protokoll (=Sammlung von Absprachen bezüglich Kommunikation, usw...) zur Kommunikation zwischen heterogenen

Rechnern. Dieses Protokoll wird von vielen anderen Programmen, so auch zum Verschicken von E-mails, benutzt. Andererseits ist `telnet` ein Programm für den „Dienst“ Remote Login. Nach dem Aufruf von **telnet** befindet man sich im Kommandomode. In diesem gibt es folgende Kommandos und Parameter:

close	Beendet die gerade bestehende Verbindung
display	zeigt die Operationsparameter an
mode	ändert den Eingabemodus von Line in Char (oder umgekehrt)
open	Verbindung aufbauen
quit	verläßt Telnet
send	überträgt spezielle Zeichen
set	setzt Operationsparameter (siehe nächste Folie)
status	gibt Statusinformationen aus
toggle	Verschiedene Operationsparameter ein- bzw. ausschalten
z	Suspend Telnet

Hilfe zu den oben genannten Befehlen erhält man durch Angabe des Befehls und nachfolgendem Fragezeichen. Zum Beispiel bringt **„set ?“** folgende Ausgabe:

```
telnet> set ?
echo            character to toggle local echoing on/off
escape          character to escape back to telnet command mode
rlogin          rlogin escape character
tracefile       file to write trace information to

                The following need 'localchars' to be toggled true
flushoutput     character to cause an Abort Output
interrupt       character to cause an Interrupt Process
quit            character to cause an Abort process
eof             character to cause an EOF

                The following are for local editing in linemode
erase           character to use to erase a character
kill            character to use to erase a line
lnext           character to use for literal next
susp            character to cause a Suspend Process
reprint         character to use for line reprint
worderase       character to use to erase a word
start           character to use for XON
stop            character to use for XOFF
forw1           alternate end of line character
forw2           alternate end of line character
ayt             alternate AYT character
...usw.
```

Man kann auch gleich beim Aufruf einen remote host mit angeben, also:

`telnet` *remote-computer-name*

5.3.2 Rlogin

Dieses Programm funktioniert prinzipiell wie `telnet`, allerdings mit einem etwas anderem Kommandomodus. Man sollte versuchen, `telnet` zu verwenden, weil es das effizientere Protokoll ist.

Um sich in einen Rechner einzuloggen, benutzt man:

`rlogin` *Rechnername*

Mit der Option „**-l**“ kann ein **Username** angegeben werden, der beim login auf dem anderen Rechner verwendet werden soll; ansonsten wird die gleiche Benutzerkennung wie auf dem lokalen Rechner verwendet. Die Option „**-e**“ ändert die „*escape characters*“ für den Kommandomodus, dessen wichtigste Optionen (jeweils mit Standard-escape character) hier kurz dargestellt werden:

~.	Verbindung trennen
~CTRL-Z	Suspend mode
~CTRL-Y	Suspend input

5.3.3 Weitere R-Kommandos

Diese Kommandos sind die „Netzwerkversionen“ von anderen Standardkommandos, wie zum Beispiel `cp`. Das Kommando `rcp` ist praktisch die netzerweiterte Version von `cp`. Einige grundlegende R-Kommandos sind:

- Um Files von einem System zum anderen zu kopieren:

  rcp *hostname:/path/src hostname:/path/dest*

- Es besteht auch die Möglichkeit, eine *„Remote-Shell“* zu öffnen, um auf einem anderen Rechner Kommandos auszuführen.

  rsh *[-l username] host [command]*

5.4 File Transfer Protocol

Das *File Transfer Protocol (***FTP***)* dient zur Datenübertragung zwischen heterogenen Rechnern. Dabei unterscheidet man zwischen dem *local host* (der Rechner, von dem man die Datenübertragung gestartet hat) und dem *remote host* (der Rechner, auf den man sich per ftp eingeloggt hat).
Will man Daten mit Hilfe dieses Protokolls übertragen, benutzt man einen FTP-Client, z. B. `ftp`. Die Kontaktaufnahme mit dem remote host läuft analog zu einer Telnet-Session ab:

ftp *remote-computer-name*

Man wird auch hier mit einer Login-Aufforderung und einer Passwortabfrage konfrontiert. Bei gelungener Kontaktaufnahme mit dem FTP-Host stehen nun folgende Kommandos zur Verfügung:

put *filename*	Um ein File zum Host zu laden
get *filename*	Um ein File vom Host zu speichern
binary	Schaltet in den Modus zur Programmübertragung
ascii	Schaltet in den Modus, um (Text-)Dateien zu übertragen
hash	Schaltet einen Zustandsindikator an
quit	Beendet FTP-Session
lcd *path*	Zeigt / Setzt den Pfad am eigenen Computer

Abhängig von der Kofniguration des FTP-Servers funktionieren auch die üblichen Kommandos, um sich im Verzeichnisbaum zu bewegen (`cd`, `ls`, ...) oder auch Pipes. Beispiel:

ftp> **get indexfile | grep doom**

5.4.1 Anonymous FTP

FTP kann nicht nur verwendet werden, wenn man auf dem remote host einen Account hat; viele Server mit Shareware- oder Public-Domain-Software erlauben auch *anonymous ftp*. Dort gibt man als login **„anonymous“** oder **„ftp“** an, als Paßwort die eigene email-Adresse.
Um die Netzbelastung möglichst gering zu halten, sollte man — statt auf weit entfernte Rechner zuzugreifen — nach Möglichkeit *Mirror-Server* verwenden, die die Software von wichtigen ausländischen FTP-Servern in regelmäßigen Abständen kopieren und ihrerseits zur Verfügung stellen.
Umfangreiche Archive findet man z. B. unter *ftp.uni-stuttgart.de* und *ftp.uni-paderborn.de*.

5.4.2 Übliche Probleme

Falls man sich beim Einloggen auf ein System, auf dem man einen FTP-Account besitzt, vertippt, bekommt man eine „Login incorrect“-Nachricht und einen FTP-Prompt.
In diesem Fall kann man durch Eingabe des Kommandos `user` erneut einen Loginprompt anfordern. Es besteht auch die Möglichkeit, den Loginnamen direkt hinter das `user`-Kommando anzuhängen; man umgeht damit den Loginprompt und kommt direkt zur Paßwortabfrage.

```
atomix:[pklein] >ftp 127.0.0.1
Connected to 127.0.0.1.
220 atomix FTP server (Version wu-2.4(1) Tue Aug 8 15:50:43 CDT 1995) ready.
Name (127.0.0.1:pklein): pklein
```

```
331 Password required for pklein.
Password:
530 Login incorrect.
Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
```

```
ftp> user
(username) pklein
331 Password required for pklein.
Password:
230 User pklein logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Falls die Verbindung unterbrochen wird, besteht die Möglichkeit, mit dem `open`-Kommando wieder eine Zieladresse anzugeben; auch hier gilt wieder, daß man das Ziel direkt hinter dem Kommando angeben kann.

```
atomix:[pklein] >ftp atomix.cs.uni-sb.de
ftp: atomix.cs.uni-sb.de: Unknown server error
ftp> open
(to) 134.96.247.250
Connected to 134.96.247.250
220 atomix FTP server (Version wu-2.4(1) Tue Aug 8 15:50:43 CDT 1995) ready.
Name (127.0.0.1:pklein):
```

5.4.3 Mehrere Files

Um mehrere Files up- bzw. downloaden zu können, kann man die Befehle *mput* und *mget* benutzen. Um zum Beispiel alle Dateien, die mit „doc“ enden, downzuladen, gibt man folgendes ein:

mget **.doc*

analog beim upload:

mput **.doc*

Bei jedem File wird explizit gefragt, ob es transferiert werden soll. Falls dies als störend empfunden wird, kann man dies mit dem Kommando **prompt** abstellen.

5.4.4 .netrc

In einer Datei *$home/.netrc* lassen sich Informationen über logins, Passwörter oder aliase speichern. Dies geschieht in folgender Form:

```
machine         ftp.uni-paderborn.de
login           anonymous
password        mawan@fsinfo.cs.uni-sb.de
macdef simt
                cd /Mirrors/SimTel.Coast.NET
```

Dabei gibt *machine* an, für welchen FTP-Server die folgenden Einstellungen gelten sollen, *login* ist dann der User-Name, *password* das dazugehörige Passwort *(Hinweis: funktioniert aus Sicherheitsgründen nur, wenn die Datei keine Zugriffsrechte für* group *und* others *hat!)*. Mit *macdef* läßt sich ein Makro definieren, der dann in `ftp` durch Voranstellen von „$“ aufrufbar ist; dabei ist zu beachten, daß die eigentliche Makrodefinition erst in der nächsten Zeile steht, und zwar hinter einem führenden Tabulator-Zeichen.

5.4.5 NcFTP

NcFTP ist ein anderes User-Interface für FTP. Es bietet eine benutzerfreundlichere Oberfläche als Standard-FTP und besitzt einige nützliche Features, wie zum Beispiel:

- Automatisches Einloggen (als *anonymous*)
- Aliase für FTP-Server
- Speicherung von Informationen über bestimmte Server, z. B. auch von Passwörtern für nicht-anonymes FTP in dem Verzeichnis *$home/.ncftp — Achtung: Wenn man Passwörter mit abspeichert, so stehen diese (wie bei .netrc) im Klartext in der Datei $home/.ncftp/hosts!*
- Datei- und Verzeichnisnamen-Ergänzung

5.5 Electronic Mail

5.5.1 Grundsätzliches

Mails sind (Text-)dateien, die über das Internet verschickt werden. Dabei muß immer eine *email-Adresse* angegeben werden, die die Form

username@host

hat.
Die mail wird dabei vom einem Rechner zum nächsten weitergeleitet, bis sie schließlich den Zielrechner erreicht hat. Dort wird die mail an den *incoming folder* des entsprechenden Users angehängt. Falls es zu einem Fehler kommt, z. B. daß der Rechner oder Benutzer, an den die mail gehen sollte, nicht existiert, so erzeugt der entsprechende *mailer-daemon* eine diesbezügliche Fehlermeldung und schickt diese an den Absender zurück.
Die Email besteht sie aus einem sogenannten *header*, gefolgt von einer Leerzeile und dem eigentlichen Text der mail. Ein header kann z. B. so aussehen:

```
From kor@ep.sub.de Wed Feb 14 11:57:46 1996
Received: by fsinfo.cs.uni-sb.de with ESMTP; Wed, 14 Feb 1996 11:57:46 +0100 (MET)
Received: by cscip.uni-sb.de with ESMTP; Wed, 14 Feb 1996 11:57:38 +0100 (MET)
Received: by uni-sb.de with ESMTP; Wed, 14 Feb 1996 11:57:35 +0100 (MET)
Received: from ep.UUCP (root@localhost) by subnet.sub.net
          (8.7.3/8.7.3/1.11subnet) with UUCP id LAA17423 for
          mawan@cscip.uni-sb.de; Wed, 14 Feb 1996 11:57:33 +0100 (MET)
Received: from sol10.ep.sub.de (sol10 [194.120.231.11]) by nexus.ep.sub.de
          (8.6.12/8.6.12) with ESMTP id LAA14030 for <mawan@cscip.uni-sb.de>;
          Wed, 14 Feb 1996 11:09:26 +0100
Received: (from kor@localhost) by sol10.ep.sub.de (8.6.12/8.6.9) id LAA26004;
          Wed, 14 Feb 1996 11:10:26 +0100
```

```
0Date: Wed, 14 Feb 1996 11:10:23 +0100 (MET)
From: Robert Koenig <kor@kor.ep.sub.de>
X-Sender: kor@sol10.ep.sub.de
To: Martin Wanke <mawan@cscip.uni-sb.de>
Subject: NT
Message-ID: <Pine.SUN.3.91.960214100829.21968A-100000@sol10.ep.sub.de>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII

Hi Martin!

[hier folgt der Rest der eigentlichen mail]
```

Dem header kann man folgende Informationen entnehmen:

- wer die mail verfasst hat; hierbei sind im `From:`-Feld normalerweise sowohl die email-Adresse als auch der Name des Absender vermerkt
- Namen sämtlicher Rechner, die die mail weitergeleitet haben, jeweils einschließlich dazugehörigem Datum und verwendetem Protokoll (`Received:`)
- das Datum und die Uhrzeit, wann die mail abgeschickt wurde (`Date:`)
- der Empfänger der mail (`To:`)
- worum es in der mail geht (`Subject:`)
- die sogenannte `Message-ID`, die normalerweise uninteressant sein dürfte; allerdings kann sie einem Systemverwalter nützlich sein, wenn mal etwas nicht so läuft, wie es sollte...
- die `MIME-Version:` *(s. u.)*
- den `Content-Type:` um was für eine Art von Daten es sich bei der mail handelt
- Außerdem gibt es noch weitere Felder, die jedoch nicht immer benutzt werden, z. B. `Priority:` (wie dringend eine mail ist), `In-reply-to:` (auf welche mail sich diese mail bezieht), `Expires:` (wann eine mail nicht mehr von Bedeutung ist) sowie diverse `X-`header-Einträge, die Auskunft über den Absender (Geek-Code, WWW-Seiten, etc.) oder über die verwendete Software stehen kann

5.5.2 MIME (*M*ulti-Purpose *I*nternet *M*ail *E*xtensions)

MIME-Mailer bieten die Möglichkeit zum Senden und Empfangen von Multipart- und Multimedia-Mail. Voraussetzung zum reibungslosen Funktionieren ist geeignete Software, welche die übertragenen Dateien (Grafik, Sound, Text, Animation) korrekt darstellt.
Anzumerken wäre noch, daß die Netzbelastung dabei durch die großen Datenmengen

natürlich zunimmt.

5.5.3 ELM

Da das UNIX-Standardtool *mail* teilweise nicht besonders benutzerfreundlich ist, gibt es Mailprogramme, die interaktiv bedienbar sind und deren Bedienung daher leichter zu erlernen ist. Beispiele für diese Programme sind `pine` und `elm`, von denen wir erstgenanntes näher betrachten wollen.

Erster Aufruf

Das Programm `elm` benötigt ein Verzeichnis *$home/.elm*, in dem er seine Konfigurationsdateien ablegt. Da dieses Verzeichnis im Normalfall noch nicht existiert, fragt `elm` beim ersten Aufruf, ob es Verzeichnis anlegen darf:

```
Notice:
This version of ELM requires the use of a .elm directory in your home
directory to store your elmrc and alias files. Shall I create the
directory .elm for you and set it up (y/n/q)? y
```

Hier sollte man mit „**y**“ bestätigen.
Weiterhin wird ein Verzeichnis gebraucht für die *mail-folder*, also die Dateien, in denen die eigentlichen mails archiviert werden:

```
Notice:
ELM requires the use of a folders directory to store your mail folders in.
Shall I create the directory /home/testi/Mail for you (y/n/q)? y
```

Auch dieses Verzeichnis sollte man anlegen lassen, also nochmal „**y**“.
Sobald das geschehen ist, startet `elm`.

Das Hauptmenu

Jedesmal, wenn `elm` gestartet wird, wird erscheint ein Bildschirm, der dem folgenden sehr ähnlich ist:

```
Mailbox is 'fsinfo:/usr/spool/mail/testi' with 5 messages [ELM 2.4 PL25]
```

```
 N  1   Apr 14 Martin Wanke       (41)   html (fwd)
 O  2   Apr 3  Mario Cattaneo     (57)   Re: WWW-Referers (tja dann... ;)
 O  3   Apr 1  Bastian Kleineidam (57)   linux und anderes..
    4   Mar 28 Niklas Matthies    (40)   Re: DS9-Ignoranz
 D  5   Mar 26 SCHUSTER@bbg-mail. (42)


   You can use any of the following commands by pressing the first character;
d)elete or u)ndelete mail,  m)ail a message,  r)eply or f)orward mail, q)uit
   To read a message, press <return>.  j = move down, k = move up, ? = help

Command:
```

Natürlich kann es sein, daß beim ersten Start noch keine mails vorhanden sind; wir betrachten weiterhin obiges Beispiel, um die Bedienung von `elm` zu erläutern.

Der Bildschirm ist folgendermaßen aufgebaut:
In der oberen Zeile ist angegeben, welcher *folder* gerade gelesen wird. In diesem Fall handelt es sich um den sogenannten „*incoming folder*“, also denjenigen, in den die mails, die neu ankommen, geschrieben werden. Dieser folder liegt auch nicht in einem Unterverzeichnis von *$home*, sondern unter */usr/spool/mail/*username. In unserem Beispiel enthält er 5 mails, die darunter angezeigt werden.

Die Symbole vor der laufenden Numerierung haben folgende Bedeutung:

(kein Symbol)	mail, die bereits gelesen wurde
D	mail, die zum Löschen *(„delete“)* markiert wurde
E	mail, die nicht mehr aktuell *(„expired“)* ist
N	mail, die angekommen ist, nachdem das letzte Mal mail gelesen wurde *(„new“)*
0	mail, die schon beim letzten Maillesen vorhanden war, aber noch nicht gelesen wurde

In den nächsten Spalte stehen Monat und Tag des Absendedatums, Name oder Absenders, dann die Länge der mail in Zeilen (einschließlich header) und schließlich das Subjekt, also worum es in der mail geht.

Es folgt ein kleines Menu, in dem die wichtigsten Befehle angeboten werden:

delete	eine mail zum Löschen markieren
undelete	die Löschmarkierung einer mail wieder aufheben
mail	eine mail verschicken
reply	auf eine mail antworten
forward	eine mail an jemand anderen weiterschicken
quit	`elm` verlassen
RETURN	aktuelle mail lesen
j	Markierung abwärts (wie bei vi)
k	Markierung aufwärts (wie bei vi)
?	Hilfe

Weitere wichtige Befehle, die nicht angezeigt werden, lauten:

!	shell
$	folder neu einlesen
+, →	den nächsten Index-Bildschirm anzeigen
-, ←	den vorigen Index-Bildschirm anzeigen
/	Suche über Absender und Subject
//	Volltextsuche (auch Mail-Text)
alias	Alias-Menu öffnen
bounce	mail weiterleiten
change	folder wechseln
group reply	Antwort an Absender und sonstige Empfänger der mail
headers	anzeigen der mail einschließlich header
J	nächste mail markieren
j, ↓	wie „**J**“, läßt aber zum Löschen markierte mails aus
K	vorige mail markieren
k, ↑	wie „**K**“, läßt aber zum Löschen markierte mails aus
CTRL-L	Bildschirm neu zeichnen
options	Optionen einstellen
Quit	`elm` ohne Abfragen verlassen
save	eine mail speichern
tag	mail für weitere Operationen markieren
e**x**it	`elm` ohne Änderungen am folder verlassen
e**X**it	`elm` ohne Änderungen am folder und ohne Abfrage verlassen

Erläuterung der Befehle

- Eine mail, die man nicht länger aufheben will, kann zum Löschen markiert werden. Wirklich gelöscht wird sie, sobald man den folder verläßt, je nach Konfiguration nach einer entsprechenden Rückfrage
- Eine Löschmarkierung kann durch undelete wieder aufgehoben werden.
- Zum Mailen muß man die email-Adresse des Adressaten angeben, das kann entweder eine volle Adresse oder ein entsprechender Alias *(s. u.)* sein. Dabei wird auch abgefragt, ob man jemandem eine Kopie schicken möchte.
  Dann wird der eingestellte Editor aufgerufen, in dem die mail geschrieben und anschließend unter dem vorgegebenen Namen abgespeichert werden muß. Daraufhin meldet sich `elm` wieder mit einem kleinen Menu:

```
Please choose one of the following options by parenthesized letter: s
            e)dit message, edit h)eaders, s)end it, or f)orget it.
```

Hier kann man nun

- nochmals den Editor aufrufen, um Veränderungen vorzunehmen *(„edit“)*
- den header der mail editieren *(„edit headers“)* — vgl. oben
- die mail verschicken *(„send“)*
- die mail nicht verschicken *(„forget“)*

- Wenn man zum Antworten auf eine mail die reply- bzw. die *group-reply*-Funktion benutzt, braucht man die Empfänger-Adresse(n) nicht anzugeben, diese werden automatisch aus dem header der Originalmail übernommen. Außerdem hat man die Möglichkeit, die Originalmail mit in die Antwort zu übernehmen *(„quoting“)*. Bei einem *group-reply* erhalten automatisch alle Adressaten der ursprünglichen mail eine Kopie der Antwort.

- Eine mail kann man an jemand anderen weiterleiten, entweder mit der *forward*- oder der *bounce*-Funktion; der Unterschied zwischen beiden liegt u. a. darin, daß es bei einer gebouncten mail für den neuen Empfänger so aussieht, als sei sie vom ursprünglichen Absender gekommen; bei einer mail, bei der *forward* benutzt wurde, von demjenigen, der das forward benutzt hat.

- Beim Verlassen eines folders wird — abhängig von der Konfiguration — u. U. nachgefragt, ob Änderungen am folder tatsächlich durchgeführt werden sollen. Das kann im Einzelnen so aussehen:

```
Command: Quit          Move read messages to "received" folder?  (y/n) n
```

Hier kann man mails, die man bereits gelesen hat, in den sogenannten *„received folder“* verschieben; dies ist der Standard-folder für diesen Zweck. Man sollte zumindestens von Zeit zu Zeit mails löschen oder in entsprechenden foldern abspeichern, damit der *incoming folder* nicht zu groß und unübersichtlich wird.

```
Command: Quit                                    Delete message?  (y/n) n
```

Hier muß nun bestätigt werden, ob die mails wirklich gelöscht werden sollen.

- Um eine mail zu lesen, kann man entweder *RETURN*, *SPACE* oder auch *„h“* drücken; letzteres zeigt im Gegensatz zu den beiden anderen Möglichkeiten noch den header der mail an.

- Mit den Tasten *„j“*, *„J“*, *„k“*, *„K“*, *„↑“*, *„↓“*, *„+“* und *„-“* sowie durch Eingabe einer Nummer lassen sich mails auswählen.

- Mit *„?“* aktiviert man den Hilfemodus; durch nochmalige Eingabe von *„?“* erhält man eine kurze Hilfe, in der alle Tasten mit ihren Funktionen aufgelistet sind.

- Man kann aus `elm` heraus ein *shell* ausführen; nach deren Beendigung kann man weiterarbeiten, wo man vorher gerade war.
- „*$*“ liest den aktuellen folder neu ein; das ist z. B. interessant, wenn man gerade eine neue mail empfangen hat oder Änderungen am folder vorgenommen hat wie Löschen von mails etc.
- Mit „*/*“ bzw. „*//*“ kann man nach einem bestimmten Stichwort suchen.
- `elm` verfügt über ein recht gutes alias-system, auf das weiter unten näher eingegangen wird.
- Mit „*c*“ kann man den aktiven folder wechseln; hierbei kann man — wie bei *save* auch — folgende Abkürzungen verwenden:
  - „*!*“ für den incoming-folder (standardmäßig /var/spool/mail/*username*)
  - „*>*“ für den received-folder (standardmäßig $home/Mail/received)
  - „*<*“ für den sent-folder (standardmäßig $home/Mail/sent)
  - ein vorangestelltes „*=*“ wird durch den Pfad des Mail-Verzeichnisses ersetzt (standardmäßig $home/Mail/)
- „*CTRL-L*“ zeichnet den Bildschirm neu, was nützlich sein kann, wenn Teile der Anzeige z. B. durch einen talk-request oder die Ausgabe eines Hintergrundprozesses überschrieben wurde.
- Unter Options kann man einige Einstellungen tätigen; wesentlich mehr Möglichkeiten hat man allerdings durch das Editieren der *elmrc*-Datei *(s. u.)*.
- Beim Verlassen mit „*Q*“ werden die Sicherheitsabfragen *(s. o.)* im Gegensatz zu „*q*“ übergangen.
- „*exit*“ verläßt den folder *ohne* Veränderungen; auch hier kann die Abfrage durch den Großbuchstaben übergangen werden.
- Beim Speichern wird der header mit in die Datei übernommen; anschließend wird die mail zum Löschen markiert.
- Mit dem *tag*-Befehl lassen sich Aktionen wie z. B. „speichern“ oder „an eine bestimmte Person weiterleiten“ auf mehrere mails gleichzeitig anwenden.

Aliase

In `elm` besteht die Möglichkeit, für email-Adressen sogenannte *aliase* zu definieren, d. h. eine Abkürzung, die sich der Benutzer leicht merken kann. Darüber hinaus kann man mehrere Adressen *(„group alias“)* zusammenfassen.
Zum Anlegen und Warten der Alias-Datenbank gibt es mehrere Möglichkeiten:

- unter Zuhilfenahme des entsprechenden Menus:

  - „$“ bringt die aliase nach Veränderungen auf den aktuellen Stand
  - „/“ erlaubt eine Suche
  - *SPACE* und *RETURN* zeigen an, wofür der ausgewählte alias steht
  - mit „*a*“ kann man die Absenderadresse aus der aktuellen mail (im übergeordneten Menu) übernehmen
  - **c**hange verändert einen bereits bestehenden alias
  - **d**elete löscht einen Eintrag (**u**ndelete zum Wiederherstellen)
  - **m**ail schickt eine mail an den aktiven alias
  - **n**ew erlaubt das Anlegen eines neuen aliases
  - Ferner hat man die gewohnten Tasten zur Bewegung innerhalb des Menus zur Verfügung.

- durch Editieren des alias-files:

  - aus dem Menu heraus:
    Eingabe von **e**dit
  - von der shell aus:
    Hierbei ist zu beachten, daß nach dem Editieren der Datei `$home/.elm/-aliases.text` unbedings noch das Kommando **newalias** einzugeben ist, damit die alias-Datenbank aktualisiert wird.

Die Datei `elmrc`

In der Datei `elmrc`, die normalerweise im Verzeichnis *$home/.elm/* zu finden ist, kann man `elm` so konfigurieren, daß das Programm den persönlichen Anforderungen am ehesten entspricht. Diese Datei wird automatisch angelegt, wenn man unter *options* die Einstellungen mit „>“ speichert. Die Datei ist gut kommentiert, so daß hier nur auf die wichtigsten Optionen eingegangen wird:

- der zu benutzende Editor kann hier festgelegt werden (ansonsten wird der Editor benutzt, der in der Umgebungsvariable `EDITOR` bzw. `VISUAL` festgelegt ist:)
  Dazu gibt es die Einträge *alteditor*, *easyeditor*, *editor* und *visualeditor*.

- die Standardeinstellungen für die Abfragen beim folder-Ändern: *alwaysdelete*, *alwayskeep*, *alwaysstore*

- ob überhaupt gefragt werden soll, oder gleich die Standardwerte verwendet werden sollen

- ob man selber Kopien der ausgehenden mail haben möchte *(copy)* und wohin diese gespeichert werden soll *(sentmail)*

- der eigene Name, wie er beim Adressaten erscheinen soll *(fullname)*

- welche Datei als Signature verwendet werden soll (die Datei, die an jede mail automatisch angehängt wird; dort schreibt man normalerweise die eigene email-Adresse usw. hinein): *(loacalsignature)* für mails, die an Leute auf dem gleichen host gehen, *(remotesignature)* für den Rest
- das Verzeichnis, in dem die mail-folder liegen *(maildir)* sowie der Name des received-folder: *(receivedmail)*
- womit mails angezeigt werden *(pager)*
- welche shell verwendet werden soll *(shell)*
- Außerdem kann man u. a. einstellen, nach welchen Kriterien und in welcher Reichenfolge die mails und aliase sortiert werden sollen, ob das Anhängen an Dateien bestätigt werden muß etc.

5.6 Nachrichten (*News*) im Netzwerk

Nachrichten sind hierarchisch in sogenannte Newsgruppen unterteilt. Es existieren Obergruppen, sogenannte Resorts, an denen man den Inhalt erahnen kann. Einige Resorts sind z. B.: `sci, comp, rec, alt`. Eine Gruppe könnte nun wie folgt aussehen: **rec.music.folk**. Dies sagt uns, es handelt sich um eine Freizeitgruppe (`rec`), welche sich mit Musik (`music`), insbesondere der Stilrichtung amerikanische Volksmusik (`folk`) auseinandersetzt.
Innerhalb einer Gruppe gibt es Threads; diese bestehen aus den eigentlichen Artikeln und den (öffentlichen) Antworten („Follow-Ups“) darauf. Diese kann normalerweise jeder verfassen; allerdings gibt es auch sogenannte „moderated newsgroups“; dort wird ein Artikel bzw. Follow-Up erst an einen sogenannten Moderator geschickt, der dann nach den Kriterien der **FAQ** (=**F**requently **A**sked **Q**uestions; hier werden Fragen, die häufig zu Grundlagen des entsprechenden Themas gestellt werden, beantwortet) entscheidet, ob der Artikel in der Gruppe erscheinen soll oder nicht.

Zum Teilnehmen am Newsgeschehen braucht man zunächst einen Newsreader. Das einfachste, uns zur Verfügung stehende Programm hierfür ist `tin`.
Falls der Standard-Newsserver (bei uns `news.coli.uni-sb.de`) nicht erreichbar ist, kann man Alternativen angeben, z. B.:

setenv NNTPSERVER news.uni-stuttgart.de

Danach muß man tin mit der Option „-r“ starten: **„tin -r“**

Lokale Newsserver, die für jedermann zugänglich sind, sind z. B.:

- `news.phil.uni-sb.de`
- `coli-gate.coli.uni-sb.de`
- `nntp.rz.uni-sb.de`

Innerhalb von `tin` bewegt man sich mit den Cusortasten durch die Newsstruktur. Informationen über Nachrichten im Internet erhält man u. a. in folgenden Newsgruppen:

- `de.newusers.questions`
- `de.newusers`
- `de.newusers.questions`

sowie in den englischsprachigen Gruppen:

- `news.announce.newusers`
- `news.newusers.questions`
- `news.answers`

5.6.1 Grundbefehle von tin

Der untere Bildschirmbereich enthält folgende Informationen: *(falls das nicht der Fall ist: dieses Menu läßt sich durch Eingabe von* **„H“** *ein- und ausblenden)*

```
 <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/unread,
  s)ubscribe, S)ub pattern, u)nsubscribe, U)nsub pattern, y)ank in/out
```

Hier noch ein paar wichtige Tastaturkürzel:

Hauptmenu

SPACE	eine Seite nach unten
CTRL-L	Bildschirm neuzeichnen
RETURN	Gruppe / Artikel auswählen
TAB	nächste Gruppe mit ungelesenen Artikeln
/,?	vorwärts / rückwärts suchen
y	alle Newsgroups anzeigen
r	nur noch nicht Gelesenes anzeigen
s,u	eine Gruppe abonnieren / abbestellen
q,Q	aktuelle(n) Artikel / Gruppe bzw. tin verlassen
w	Artikel in aktuelle Gruppe posten
W	alle selbst-geposteten Artikel zeigen
H	Menu mit Kurzhilfe *(vgl. oben)* ein- / ausblenden
h	Hilfe
M	Optionsmenü
!	shell

weitere Befehle im Gruppen- bzw. Artikelmenu

a,A	Autorensuche (vorwärts / rückwärts)
l	einzelne Follow-Ups eines Threads anzeigen
c	alle Artikel als gelesen markieren
z	alle Artikel als ungelesen markieren
-	letzten Artikel nochmals anzeigen
CTRL-H	Header anzeigen
s,S	Speichern mit / ohne Abfragen
m	Artikel mailen
r	Antwort an den Autoren (per mail)
f	Follow-Up posten
D	Artikel löschen *(nur eigene!)*

Hinweis: Statt der Cursor-Tasten kann man zum Bewegen auch analog zum vi „h“, „j“, „k“ und „l“ benutzen.)

5.7 Direkte Kommunikation

Neben mail und news gibt es noch weitere Arten, mit anderen Internet-Usern in Verbindung zu treten; dazu ist es allerdings nötig, daß der entsprechende User zur gleichen

Zeit wie man selbst eingeloggt ist; dafür kann man sich dann direkt mit ihm „unterhalten“.

Hierzu gibt es mehrere Möglichkeiten:

1. Kurze Mitteilungen: **write**
2. Eine Art „Telefongespräch“: **talk**
3. Treffen mit vielen Usern: **IRC**

5.7.1 write

Wenn man jemandem, der auf dem gleichen Rechner wie man selbst eingeloggt ist, eine kurze Mitteilung schicken möchte, muß man nicht unbedingt eine mail schreiben; manchmal ist auch eine message einfacher.

Das sieht dann z. B. so aus:

```
(42) testi@fsinfo > write mawan
write: mawan logged in more than once ... writing to ttyp4
Ich gehe jetzt Essen!
```

Hinweis: Die Eingabe wird mit CTRL-D abgeschlossen.

Es ist auch möglich, ein tty mit anzugeben, z. B. so:

```
(43) testi@fsinfo > write mawan ttyp4
Kommst Du mit?
```

Auf der anderen Seite sieht das Ganze dann so aus:

```
Message from testi@fsinfo on ttyq1 at 12:42 ...
Ich gehe jetzt Essen!
EOF
```

Dies ist für kurze Mitteilungen ganz praktisch; wenn man allerdings nicht auf dem gleichen Rechner wie der entsprechende User eingeloggt ist oder sich länger „unterhalten“ möchte, sollte man dafür besser eine andere Methode wählen:

5.7.2 talk

Per talk läßt sich eine direkte Verbindung zwischen mehreren Usern herstellen; dabei erscheint das, was der eine tippt, jeweils (je nach Geschwindigkeit der Netz-Verbindung)

nahezu zeitgleich bei den anderen Beteiligten auf dem Bildschirm.
Um eine solche Verbindung herzustellen, schickt man zunächst einen *talk-request*, indem man

talk *username@hostname*

eingibt.

Daraufhin erscheint bei dem „Angetalkten“ etwa folgendes:

```
Message from Talk_Daemon@fsinfo at 15:58 ...
talk: connection requested by testi@fsinfo.cs.uni-sb.de.
talk: respond with: talk testi@fsinfo.cs.uni-sb.de
```

Hierauf muß dieser — wenn er die Verbindung herstellen will — mit dem Befehl antworten, der ihm in der unteren Zeile angegeben wird, in diesem Fall also mit:

talk testi@fsinfo.cs.uni-sb.de

Hinweis: Der Name des entsprechenden Rechners muß gegebenenfalls nicht ausgeschrieben werden; handelt es sich dabei um den Rechner, auf dem man selbst eingeloggt ist, reicht sogar der Username aus.
Tip: Statt **talk** kann man auch **ytalk** benutzen; dadurch kann man auch mit mehr als zwei Leuten gleichzeitig talken; außerdem hat man ein Menu zur Verfügung, mit dessen Hilfe sich einige interessante Optionen einstellen lassen.

5.7.3 IRC

Während die beiden eben vorgestellten Programme der Kommunikation mit bestimmten Usern dienen, ist **IRC** *(**I**nternet **R**elay **C**hat)* eher dafür gedacht, sich über ein bestimmtes Thema zu unterhalten (oder einfach nur ein bißchen Zeit zu verschwenden...). Dabei gibt es für alle möglichen Themen *(topics)* — vergleichbar mit den newsgroups im Usenet — einen Channel, den normalerweise jeder betreten und verlassen kann. Eingaben werden bei allen anderen angezeigt, die sich momentan auf dem gleichen Channel befinden, was teilweise — insbesondere auf Channels mit vielen Usern — recht unübersichtlich werden kann. Außerdem gibt es die Möglichkeit, jemandem eine persönliche Nachricht zu schicken, die nur derjenige sehen kann; man kann sich auch auf mehreren Channels gleichzeitig mit Leuten unterhalten usw.

Befehle fangen in IRC grundsätzlich mit einem „/“ an; die wichtigsten sind:

/alias *name Befehl*	einen alias definieren
/help *[Thema]*	Hilfe anzeigen
/invite *user*	jemanden in einen Channel einladen
/join *channel*	einen Channel betreten
/kick *channel user*	jemanden aus dem Channel werfen
/leave *channel*	einen Channel verlassen
/list *(Parameter)*	eine Liste von Channels anzeigen
/me *Kommentar*	einen Kommentar abgeben
/mode *(Parameter)*	Einstellungen setzen oder verändern
/msg *user Nachricht*	eine persönliche Nachricht an einen User schicken
/nick *neuer_nick*	seinen Nick-Name verändern
/quit *[Kommentar]*	IRC verlassen
/server *[neuer_server]*	den IRC-Server wechseln bzw. Infos anzeigen
/set *(Parameter)*	IRC-Variablen setzen oder verändern
/topic *[channel] [neues_thema]*	das Thema des Channels anzeigen oder verändern
/who, /whois *channel* | *user*	Informationen über User anzeigen

Hinweis: Einen Teil der Einstellungen (IRC-Server, Namen) kann man auch durch Umgebungsvariablen in der shell setzen.
Manche Befehle sind nicht immer allen Usern erlaubt.

5.8 Daten im Internet finden

Oft stellt sich das Problem, daß man ein spezielles Programm benötigt, welches irgendwo im Internet existiert. Um die Suche nach dem „Aufenthaltsort“ zu vereinfachen, wurde Archie erfunden. Archie ist ein Dienst zum Lokalisieren von Dateien. Archie durchsucht sehr viele anonymous-FTP-Server und merkt sich deren Verzeichnisstruktur inklusive aller Filenamen. Auf diesen Listen kann schnell gesucht werden.
Der nächste Archie-Server befindet sich in Darmstadt. Man erhält Zugriff z. B. via telnet:

telnet archie.th-darmstadt.de *(login: archie)*

Nun befindet man sich in einem interaktiven Abfragemodus, in welchem folgende Kommandos benutzt werden können:

Kommando	Erklärung
prog *string*	startet die Suche
whatis *string*	sucht nach Schlüsselwort in einer Index-Datenbank
set *search*	setzt den Suchtyp (z. B. `exact`)
set *match_domain*	beschränkt den Domainsuchbereich (z. B. `set match_domain de`)
set *match_path*	legt den Pfad fest, der vorkommen soll (z. B. `set match_path pub:linux`)
show *variable*	zeigt den Wert einer Variablen
maxhits *nummber*	beschränkt die Anzahl der anzuzeigenden Treffer (1–1000)
pager	Ausgabe stoppt nach jeder Bildschirmseite
help	Anzeige der Hilfeseite

Es besteht auch die Möglichkeit, Archie via Email zu benutzen. Dies bietet sich dann an, wenn gerade kein Zugang mehr zu einer interaktiven Session möglich ist (durch Überlastung o. ä.).
Die Mailadresse für den nächstgelegenen Archie-Server lautet:

archie@archie.th-darmstadt.de

Als Subject muß nichts eingetragen werden. Im eigentlichen Text kann man nun sämtliche Archie-Kommandos benutzen, welche auch in der interaktiven Session funktionieren. Die einzige Bedingung an das Format der Mail besteht darin, daß jedes Kommando in der ersten Spalte beginnt. Hat man sich vertippt oder einen anderen Syntaxfehler begangen, gibt Archie die Standard Hilfeseite aus. Antwort sollte man noch am gleichen Tag bekommen.
Vorsicht! Wenn die übliche Signature an die mail angehängt wird, sollte man die Befehle mit „quit“ abschliessen, da sonst versucht wird, diese auch noch als Befehle zu interpretieren!

5.9 Jemanden im Internet finden

5.9.1 finger

Jemanden ausfindig zu machen, von dem man weiß, auf welchem Rechner er arbeitet, ist relativ einfach. Der Befehl hierfür lautet:

finger *name@host*

Allerdings kann es durchaus passieren, daß der Hostrechner aus Sicherheitsgründen kein „finger“ zuläßt; schließlich bekommt man mit dem Loginnamen einen Teil des Zugriffcodes für dieses System.
`finger` liefert als Ausgabe — je nach finger-daemon — z. B. den Namen des Gesuchten, sowie den Inhalt seiner `.plan`- und `.project`-Files, wann er zuletzt Mail erhalten und gelesen hat sowie wann er zuletzt eingeloggt war (bzw. ob er gerade eingeloggt ist).
Bsp.:

```
atomix:[pklein] >finger pklein
Login: pklein                           Name: Peter Klein
Directory: /home/pklein                 Shell: /bin/bash
On since Sun Jan 21 19:15 (MET) on tty1,  idle 0:37
New mail received Sun Jan 21 19:58 1996 (MET)
     Unread since Fri Dec  1 12:02 1995 (MET)
```

Ist ein `.project`-file vorhanden, so wird die erste Zeile(!) dieser Datei angezeigt; weiterhin kann man ein `.plan`-file anlegen, das beliebig lang sein darf.
Im *Project* informiert man normalerweise darüber, an welchen Projekten man sich gerade beschäftigt ist, während der *Plan* für Hinweise auf Sprechstunden, Urlaub etc. genutzt wird.
Legt man sich *Plan* und/oder *Project* an, so sollte man darauf achten, daß die entsprechenden Dateien für alle lesbar sind und das eigene Home-Verzeichnis mindestens ausführbar für alle ist.

5.9.2 WHOIS

Die WHOIS-Datenbank ist analog zur Whatis-Datenbank aufgebaut. Zugriff auf eine solche Datenbank bekommt man mittels telnet (auch mittels Gopher oder WWW).

Die wichtigsten Datenbanken sind:

- `rs.internic.net`
- `ds.internic.net`

Nun kann man mit folgender Befehlszeile zum Beispiel nach einem gewissen Ed Krol suchen:

whois -h rs.internic.net krol

Allerdings findet man in diesen Datenbanken eher das „Who is Who“ des Internets als einen Erstsemesterstudenten aus Hintertupfing...

5.9.3 Die Usenet-Benutzerliste

Das MIT besitzt eine Liste aller Internetbenutzer, die jemals im Usenet (siehe *network news*) etwas gepostet haben. Um diesen Dienst zu nutzen, schickt man eine Email an:

mail-server@rtfm.mit.edu

Der Inhalt der Email sollte wie folgt aussehen:

send usenet-addresses/*name*

Also zum Beispiel:

send usenet-addresses/pklein

5.9.4 Knowbot Information Service

Ein „Frontend“ für diverse Auskunftsdienstleistungen *(finger, X.500, Whois,...)* ist der sogenannte **Knowbot Information Service**. Man loggt sich auf port 185 von `info.cnri.reston.va.us` ein:

telnet info.cnri.reston.va.us 185

Nachdem man eingeloggt ist, gibt man einfach den Namen ein, der Suchvorgang wird dann sofort gestartet.

5.9.5 Netfind

Netfind-Server verfügen über eine Datenbank von Rechnern, die man durchsuchen und auf den entsprechenden Servern fingern kann. Dazu loggt man sich per telnet oder rlogin auf einem Netfind-Server *(Username: netfind)* ein; zur Suche gibt man den Namen, den Ort und eventuell die Institution an, bei der der Gesuchte seinen Account hat.
Ein deutscher Netfind-Server ist `netfind.uni-essen.de`.

Eine Sitzung mit Netfind könnte z. B. so aussehen:

```
(33) mawan@fsinfo > telnet netfind.uni-essen.de
[...]
login: netfind
[...]
```

```
I think that your terminal can display 24 lines.  If this is wrong,
please enter the "Options" menu and set the correct number of lines.

Top level choices:
        1. Help
        2. Search
        3. Seed database lookup
        4. Options
        5. Quit (exit server)
-->  2
Enter person and keys (blank to exit) --> Wanke Uni Saarbruecken
Please select at most 3 of the following domains to search:
        0. uni-sb.de (universitaet des saarlandes, saarbruecken, germany)
        1. cad.uni-sb.de (computer aided design, universitaet des
           saarlandes, saarbruecken, germany)
        [...]
        39. stud.uni-sb.de (im stadtwald, universitaet des
            saarlandes, saarbruecken, germany)
        [...]
Enter selection (e.g., 2 0 1) --> 0 39
```

```
( 2) SMTP_Finger_Search: checking domain stud.uni-sb.de
SYSTEM: stud.uni-sb.de
        Login name: mawan                       In real life: Martin Wanke
        Martin Wanke (mawan) is not presently logged in.
        Last seen at hermes.rz.uni-sb.de on Thu Apr 11 23:32:37 1996

        No plan.

( 1) SMTP_Finger_Search: checking domain uni-sb.de
( 1) do_connect: Finger service not available on host uni-sb.de -> cannot
do user lookup


FINGER SUMMARY:
- Remote user queries (finger) were not supported on host(s) searched in
  the domain 'uni-sb.de'.
- The most promising email address for "wanke"
  based on the above finger search is
  mawan@stud.uni-sb.de.

Continue the search ([n]/y) ? --> n
```

Wie man aus diesem Beispiel erkennen kann, kennt ein Netfind-Server nicht alle möglichen Rechner, auf denen jemand einen Account haben könnte, so besteht z. B. nicht die Möglichkeit, auf diesem Wege einen `cscip-` *oder* `fsinfo`*-Account zu finden.*
Außerdem kann nicht auf allen bekannten Rechnern gefingert werden.

5.9.6 WWW-Server

Viele Institutionen haben auch einen WWW-Server, auf dem man nach Informationen finden kann, was für Rechner existieren und wer darauf Accounts hat.
Oftmals wird auch ein Skript angeboten, das die Suche nach einer Person erlaubt, z. B. *http://www.rz.uni-sb.de/dienste/WebPh.html.*

5.10 Internetsurfen mit Gopher

Gopher ist *das* Frontend für alle bisher gezeigten Dienste (solange man über kein X-Terminal verfügt...).

Man startet den Dienst mittels **gopher**.
Man sieht nun diverse Menüeinträge, die man mit Hilfe der Cusortasten aufrufen kann. Um sich besser zurechtzufinden, haben manche Menüeinträge am Ende ein Flag, welche sie als Telnetverbindung, FTP, WAIS-Datenbank oder ähnliches ausgibt:

?	indizierte Verzeichnisse
\	normale Verzeichnisse
CSO	White Page Server

Um die Gefahr des „sich Verlaufens“ etwas zu minimieren, gibt es die Möglichkeit, Lesezeichen zu setzen (mittels **„a“**. Mit **„A“** speichert Gopher außerdem noch eventuell gestellte Suchanfragen). Der Aufruf der Bookmarkdatei (die die Lesezeichen enthält) erfolgt über **„v“**.

5.10.1 Veronica

Veronica ist das Gopheräquivalent zu Archie. Um Zugriff auf Veronica zu erhalten, folgt man einem Menüeintrag, der so oder so ähnlich aussieht:

```
19. Search titles in Gopherspace using Veronica <?>
```

Danach findet man sich in einem Menü wieder, welches ein paar Suchregionen zur Auswahl stellt. Hat man sich für eine Region entschieden, wird man aufgefordert, Schlüsselwörter einzugeben, um die Suche zu starten. Die Suchergebnisse erscheinen wiederum als Menübaum.

5.11 Datenbanksuche mit WAIS

WAIS ist ein Textretrievalsystem. Es basiert auf dem Z39.50-Standard (Kurzschreibweise für eine ANSI-Norm zur Bibliographischen Recherche). Um ein Dokument einem WAIS-Server zugänglich zu machen, muß erst ein Index erstellt werden. Ist das Dokument ein Text, wird jedes Wort indiziert. Wenn man nun eine Suchanfrage stellt, gibt der Server im Gegenzug eine Liste von Dokumenten aus, in denen die Schlüsselwörter gefunden wurden. Außerdem bewertet WAIS noch die Artikel, indem es die Anzahl der Treffer pro Schlüsselwort ausgibt.
Zugriff auf WAIS erhält man mittels *Gopher* oder *WWW-Browsern*; es existiert auch ein XWAIS-Client; außerdem besteht die Möglichkeit, sich per telnet auf einem WAIS-Server einzuloggen.

Hier einige öffentlich zugängliche WAIS-Server:

Name	Login	Location
info.funet.fi	wais	Europa
swais.cwis.uci.edu	swais	USA
cnidr.org	demo	USA
sunsite.unc.edu	swais	USA
quake.think.com	wais	USA

5.12 Das World Wide Web (WWW)

Das WWW basiert auf der **Hypertextarchitektur** (*Hypertexte* sind Texte, welche mittels Hyperlinks mit anderen Daten verbunden sind.) Dabei können verschiedene Datentypen miteinander verknüpft werden, z. B. Bilder, Sounds, Animationen, aber auch telnet- und ftp-Verbindungen.

Entwickelt wurde das WWW am europäischen Elektronenbeschleuniger (CERN) in der Schweiz.

Falls man keinen graphischen Browser zur Verfügung hat, kann man auch per telnet auf WWW-Server zugreifen. Dazu gibt man folgendes ein:

telnet info.cern.ch

Nun kommt man automatisch in einen zeichenorientierten Browser.

Es existieren zwei Möglichkeiten, sich Informationen im WWW zu beschaffen:

1. Links von irgendwelchen Einstiegsseiten aus folgen
2. strukturierte Suche
   Einige gute Search Engines sind z. B.:
   - http://www.webcrawler.com/
   - http://altavista.digital.com/
   - http://www.lycos.com/

   Außerdem kann man den *„Net Search“*-Button in Netscape benutzten.

5.12.1 Navigation

Die beiden wichtigsten X-basierten Browser sind `netscape` und `Mosaic`. Außerdem existiert noch ein Textbrowser namens `lynx`.
Am weitesten verbreitet ist zur Zeit Netscape.

Einige Features von Netscape sind:

- Bookmarks (zum Speichern interessanter Seiten)
- Buttons für Search Engines, Cool Stuff, News,...

- History
- Auch als Mail- / News-Frontend verwendbar
- Zeigt die meisten Multimedia-Formate direkt (d. h. ohne Hilfe von externen Programmen) an
- unterstützt Proxy-Server, die den Zugriff auf häufig benutzte Seiten beschleunigen und Netzlast sparen
- Dokumente lassen sich in verschiedenen Formaten auf Knopfdruck downladen

Es besteht auch die Möglichkeit, das Web zu verlassen, ohne dabei den Client zu wechseln (also zum Beispiel von Netscape zu `gopher` oder `ftp`).

Wenn man in Netscape den *„Open“*-Button anklickt, erscheint ein Fenster, in welchem man aufgefordert wird, die zu öffnende URL *(„Uniform Resource Location“)* anzugeben. Nun muß man nur noch wissen, wie eine solche URL aufgebaut ist:

[Protokoll://][Login[:Passwort]@][Server][:Port] [/Verzeichnis[/Datei]]

Eine FTP-Resource hat zum Beispiel folgendes Aussehen:

ftp://internet-name/remote-path,

wobei der *internet-name* die Adresse des gewünschten Servers ist und der *remote-path* beschreibt, welches File man möchte. Wenn man den remote-path wegläßt, zeigt der Browser den Startbildschirm des FTP-Servers, genauso, also ob man sich mit `ftp` eingeloggt hätte. Wenn man allerdings einen Pfad angibt, zeigt der Browser direkt dieses Verzeichnis an. Man kann nun durch Mausklicks die gewünschten Files selektieren oder andere Verzeichnisse öffnen.
Um zum Beispiel das File *ls-lR.Z* vom Server **ftp.uu.net** zu bekommen, benutzt man folgende URL:

ftp://ftp.uu.net/ls-lR.Z

Um UUNET's Rootverzeichnis zu bekommen, gibt man einfach

ftp://ftp.uu.net

ein.

Es kann Probleme geben, wenn man versucht, auf einen Server Zugriff zu bekommen, der einen Loginnamen verlangt. Abhilfe schafft hier meist eine allgemeinere Form von URLs:

ftp://username@internet-name/remote-path

Allerdings kann es trotzdem noch Probleme beim ftp mit Netscape geben, z. B. mit IBM-Mainframes oder Rechnern, die die angegebenen logins mit Hilfe von *identd* überprüfen. Wenn alles nichts hilft, bleibt immer noch der Weg über das normale `ftp`.
Die oben gezeigte Vorgehensweise erlaubt auch, ein eventuelles Passwort mit anzugeben. Das Passwort schreibt man durch einen Doppelpunkt getrennt hinter den Loginnamen. Man kann diese Feature benutzen, um einen persönlichen FTP-Account zu erreichen. Allerdings wird das Passwort dabei in Klartext angezeigt und auch im *.netscape*-Verzeichnis gespeichert. Daher ist aus Sicherheitsgründen davon abzuraten, einen WWW-Browser für diese Art von FTP zu benutzen.

Um auf Telnet-Resourcen zugreifen zu können, gibt man folgendes ein:

telnet://internet-name[:port]

Der *internet-name* stellt wieder den zu erreichenden Rechner dar; *port* ist ein optionaler Parameter, um z. B einen nicht-Standardport angeben zu können.

Ähnlich sieht es auch mit anderen Resourcen aus, hier beispielsweise für **gopher**, **news**, **WWW** und lokale Dateien:

gopher://internet-name[:port]
news:newsgroup-name
http://internet-name[:port]/remote-path
file:/path *(für lokale Dateien)*

5.13 Mtools

Mtools ist ein Programmpaket, das den einfachen und bequemen Zugriff auf Disketten im DOS-Format erlaubt. Die `mtools`-Kommandos (etwa `mattrib, mdir`, usw.) sind durch Links auf das zentrale Programm `mtools` realisiert. `mtools` selbst kann nicht unmittelbar ausgeführt werden, sondern nur über diese Links. Prinzipiell können DOS-Disketten — wie viele andere Filesysteme — mit `mount` in das Dateisystem eingebunden werden. Diese Vorgehensweise ist allerdings umständlicher als der `mtools` Einsatz; außerdem kann der `mount`-Befehl (in den meisten Fällen) nur von root ausgeführt werden.
Die `mtools`-Kommandos haben einige gemeinsame Merkmale: Laufwerksangaben erfolgen wie unter DOS mit *A:* oder *B:*. Wenn kein Laufwerk angegeben wird, greifen die Kommandos automatisch auf A: bzw. auf das durch `mcd` eingestellte Arbeitsverzeichnis zu *(dieses wird in der Datei $home/.mcwd gespeichert)*. In Pfadangaben wird zur Trennung von Verzeichnissen „/“ benutzt; das Jokerzeichen „*“ funktioniert wie unter UNIX üblich, weswegen zur Bearbeitung von allen Dateien „*“ (und nicht „*.*“) eingegeben werden muß. Dateinamen sind auf DOS-Konventionen (acht plus drei Zeichen) limitiert; beim Übertragen auf Disketten werden die Namen von längeren Dateien entsprechend angepaßt, ferner wird der Dateiname in Großbuchstaben umgewandelt.

Die wichtigsten Befehle sind:

Befehl:	**Parameter:**	**Funktion:**
mattrib	*[+|-ahrs] Datei*	ändert die DOS-Dateiattribute
mcd	*[Verzeichnis]*	wechselt bzw. zeigt das Arbeitsverzeichnis auf der DOS-Diskette
mcopy	*[-tnvm] Quelle Ziel*	kopiert DOS-Dateien von/nach UNIX; besonders interessant ist die automatische Konvertierung von CR/LF in LF mit der Option `-t`
mdel	*[-v] Datei*	löscht eine DOS-Datei
mdir	*[-w] [Datei]*	zeigt den Inhalt eines DOS-Verzeichnisses an
mformat	*[Parameter] Laufwerk*	legt ein DOS-Dateisystem auf einer Low-Level formatierte Diskette an
mmd	*[-v] Verzeichnis*	legt ein Verzeichnis auf einer DOS-Diskette an
mrd	*Verzeichnis*	löscht ein Unterverzeichnis auf einer DOS-Diskette
mread	*[-tnm] Quelle Ziel*	liest (kopiert) eine Datei „roh“ von DOS nach UNIX
mren	*[-v] alt neu*	benennt eine existierende DOS-Datei um
mtype	*[-st] Datei*	gibt den Inhalt einer DOS-Datei aus
mwrite	*[-tnvm] Quelle Ziel*	schreibt (kopiert) eine Datei „roh“ von UNIX auf eine DOS-Diskette.
eject	*[-dfnq]*	wirft eine eingelegte Diskette aus