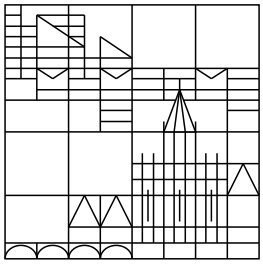

Universität Konstanz

Einführung in den Umgang mit den Sun-Workstations im Informatik-Lehre-Pool G 228 bis G 230 an der Universität Konstanz

— Revised Version —

Ulrik Brandes
Annegret Liebers

Konstanzer Schriften in Mathematik und Informatik
Nr. 4, Februar 1996, revidiert September 1998

ISSN 1430–3558

Universität Konstanz
Postfach 5560/D 188, 78434 Konstanz, Germany
Email: preprints@informatik.uni–konstanz.de
WWW: http://www.informatik.uni–konstanz.de/Schriften

Einführung in den Umgang mit den Sun-Workstations im Informatik-Lehre-Pool G 228 bis G 230 an der Universität Konstanz

Ulrik Brandes* Annegret Liebers†

Universität Konstanz
Fakultät für Mathematik und Informatik
Lehrstuhl Prof. Dr. Dorothea Wagner
4/1996
(Revidierte Fassung September 1998)

Im WS 1994/95 wurde die Vorlesung Informatik I erstmalig gehalten. Die im Rahmen der Übungen veranstaltete Rechnereinführung für die Sun-Workstations wird durch diese Ausarbeitung zusammengefaßt. Abschnitt 7 enthält eine Übersicht über alle im folgenden auftretenden Kommandos.

1 Einleitung

Diese Einführung ist notwendigerweise sehr unvollständig. Sie kann nur eine Einstiegshilfe bieten, um überhaupt erst einmal „zurechtzukommen“. Die Hörerinnen[1] werden sehr schnell Kenntnisse benötigen, die in dieser Einführung nicht vermittelt werden. Es gibt eine Vielzahl von Informationsquellen, die man dann konsultieren kann. Die Schwierigkeit besteht weniger im Mangel an Information als eher darin, in der Fülle der zur Verfügung stehenden Information die eine konkrete Fragestellung betreffende Antwort zu finden.

Aus der unüberschaubar großen Anzahl an Büchern über Rechner und Rechnernetze im allgemeinen und bestimmte Betriebssysteme oder gar die Bedienung spezieller Programme im besonderen seien die im folgenden aufgeführten empfohlen. Sie sind in der Universitätsbibliothek (zum Teil sogar in der Lehrbuchsammlung) verfügbar.

*Ulrik.Brandes@uni-konstanz.de http://www.fmi.uni-konstanz.de/~brandes/
†Annegret.Liebers@uni-konstanz.de http://www.fmi.uni-konstanz.de/~liebers/

[1] Zur besseren Lesbarkeit des Textes verwenden wir nur die weibliche (bzw. in anderen Abschnitten nur die männliche) Form. Die Personen des jeweils nicht explizit genannten Geschlechts sind dabei stets implizit mitgemeint.

[CRR96] Deb Cameron, Bill Rosenblatt, and Eric Raymond. *Learning GNU Emacs*. A Nutshell Handbook. O'Reilly & Associates, Inc., 2nd edition, 1996. ISBN 1-56592-152-6. Unibib KN: lbs 843/c15(2), kid 346:gn/c15(2).

[GH97] David F. Griffiths and Desmond J. Higham. *Learning LaTeX*. SIAM, 1997. ISBN 0-89871-383-8. Unibib KN: kid 819/g74.

[Gs] Daniel Gilly and staff of O'Reilly & Associates. *UNIX in a Nutshell: For System V & Solaris 2.0*. O'Reilly & Associates, Inc. ISBN 1-56592-001-5. Unibib KN: kid 346/g45. Ein übersichtliches und kompaktes Nachschlagewerk.

[Hah93] Harley Hahn. *A Student's Guide to Unix*. McGraw–Hill, 1993. ISBN 0-07-025511-3. Unibib KN: lbs 843/h14, kid 346:un/h14. Ein ausführliches Unix-Lehrbuch.

[Kro] Ed Krol. *The Whole Internet: User's Guide & Catalog*. A Nutshell Handbook. O'Reilly & Associates, Inc. ISBN 1-56592-025-2. Unibib KN: lbs 848/k77, kid 228/k76. Ein Buch sowohl für Einsteigerinnen als auch für Fortgeschrittene im Internet. Besonders interessant sind zu Anfang die Kapitel über EMail und über das World Wide Web (WWW). Die deutsche Übersetzung heißt *Die Welt des Internet*.

[PST] Jerry Peek, John Strang, and Grace Todino. *Learning the UNIX Operating System*. A Nutshell Handbook. O'Reilly & Associates, Inc. ISBN 1-56592-060-0. Unibib KN: lbs 843/t62, kid 346/t62b. Auf unter 100 Seiten wird alles das erklärt, was man benötigt, wenn man zum ersten Mal mit einem Unix-System und einer auf dem X-Window System beruhenden Benutzeroberfläche arbeitet.

[QO] Valerie Quercia and Tim O'Reilly. *X Window System User's Guide*, volume three of *The Definitive Guides to the X Window System*. O'Reilly & Associates, Inc. Unibib KN: kid 240.35/q92(2), kid 240.35/q92(4). Interessant für alle, die ihre Benutzeroberfläche nicht nur so benutzen wollen, wie sie ihnen anfänglich zur Verfügung gestellt wird, sondern die verstehen wollen, warum sie funktioniert, und/oder die sie ihren persönlichen Bedürfnissen anpassen wollen. Achtung: Es gibt eine Standard-Ausgabe und eine Ausgabe für Motif. Die Motif-Ausgabe entspricht am ehesten der zur Zeit auf den Sun-Workstations der Informatik verwendeten Benutzeroberfläche.

[Wei96] Karsten Weihe. Kurzübersicht über UNIX/X11/Internetzugang am Beispiel des Bereichs Informatik der Universität Konstanz. Konstanzer Schriften in Mathematik und Informatik 6, Universität Konstanz, 1996. Unibib KN: kid 2.80/k66-6, mat 2.80/k66-6. Im WWW unter `http://www.informatik.uni-konstanz.de/Preprints/` Ein zwölfseitiges Glossar.

[Wei97] Karsten Weihe. UNIX & Co. in 111 Schritten. Konstanzer Schriften in Mathematik und Informatik 31, Universität Konstanz, 1997. Unibib KN: kid 2.80/k66-31, mat 2.80/k66-31. Im WWW unter `http://www.informatik.uni-konstanz.de/Preprints/` Ausführliche Kursunterlagen zum vertiefenden Selbststudium.

1.1 Ein- und Ausloggen

In den Rechnerräumen G 228 bis G 230 des Bereichs Informatik befinden sich (u.a.) sogenannte *Arbeitsplatzrechner* (engl. Workstations) der Firma Sun Microsystems Inc. Auf diesen wird das Betriebssystem Solaris aus der Unix-Familie mit der auf dem X-Window System X11 basierenden Benutzeroberfläche OpenWindows verwendet.

Jede Hörerin hat zu Beginn der Vorlesung eine Benutzerkennung erhalten. Zum Beispiel gehört die Benutzerkennung `liebersa` zur Benutzerin Annegret Liebers. Weiterhin hat sie ein persönliches Paßwort gewählt. Beide zusammen (Benutzerkennung und Paßwort) ermöglichen den Zugang zum Arbeiten an jeder der vorhandenen Workstations. Jede Benutzerin hat dabei einen für sie reservierten sogenannten *persönlichen Arbeitsbereich* (vgl. auch Abschnitt 2.1).

Um Zugang zum persönlichen Arbeitsbereich zu erhalten, ist es nötig, zunächst die Benutzerkennung und dann das persönliche Paßwort einzugeben. Die Eingaben müssen jeweils mit der Taste `Return` abgeschlossen werden. Da das Paßwort geheim ist, wird es bei der Eingabe nicht am Bildschirm angezeigt. Damit das Paßwort geheim bleibt, darf es unter keinen Umständen weitergegeben werden und muß außerdem so gewählt werden, daß es nicht erraten werden kann[2].

Vor dem Verlassen des verwendeten Rechners muß man sich dann unbedingt wieder *ausloggen*, z.B. indem man im mit dem Wort `Console` überschriebenen Fenster das Wort `exit` tippt und dann die Taste `Return` drückt[3].

1.2 Eingabe von Kommandos

Nach erfolgreichem Einloggen erscheint die vom Window-Manager[4] verwaltete Benutzeroberfläche. Mit `xterm` überschriebene Fenster dienen der Eingabe von Kommandos.

Im mit `Console` bezeichneten Fenster zeigt das System wichtige Meldungen an, und außerdem wird es wie beschrieben zum Ausloggen verwendet.

Die Eingabe eines Kommandos beginnt hinter dem Prozentzeichen (der sogenannten *Eingabeaufforderung*, auch *Prompt* genannt; vgl. aber auch Abschnitt 5) und wird grundsätzlich durch Drücken der Taste `Return` beendet. Beispielsweise können durch Eingabe von `date` das heutige Datum und die aktuelle Uhrzeit abgefragt werden. Folgender kurzer Dialog zeigt, wie mit dem Kommando `cal` Kalender für beliebige Monate ausgegeben werden können:

[2] Vgl. die in den Rechnerräumen ausliegende Benutzungsordnung, Kapitel „Sicherheit und Datenschutz“

[3] Eine alternative Möglichkeit sich auszuloggen ist in Abschnitt 1.3 beschrieben.

[4] Der verwendete Window-Manager heißt `olwm`. Dieses Programm gehört zur Benutzeroberfläche OpenWindows. `olwm` steht für OpenLook Window-Manager.

```
% cal 1 1995
   January 1995
 S  M Tu  W Th  F  S
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

Bei der Eingabe wurden dem Kommando durch sogenannte *Parameter* nähere Angaben über den gewünschten Monat gemacht. In der Beschreibung eines Kommandos wird immer angegeben, ob Parameter erlaubt bzw. notwendig sind und welcher Art sie sein dürfen. Die Parameter zu einem Kommando bestehen aus zwei Gruppen, den *Optionen* und den *Argumenten*. Optionen beginnen i.allg. mit einem Bindestrich („`-`“) und verändern das Verhalten des Kommandos (als Beispiel siehe die Erläuterung zum Kommando `ls` in Abschnitt 2.1). Argumente bezeichnen die Objekte, mit denen das Kommando arbeiten soll (wie *monat* und *jahr* für das Kommando `cal`).

Ein besonders wichtiges Kommando ist `man`, weil man durch die Eingabe von

```
% man kommando
```

eine Beschreibung des als Parameter hinter `man` angegebenen Kommandos *kommando* erhält. `man` ruft also so etwas wie ein internes Handbuch auf und schlägt darin das gewünschte Kommando nach. Durch Tippen des Buchstabens `q` kann man das Handbuch wieder verlassen.

Weitere Fenster zur Kommando-Eingabe können jederzeit mit dem Befehl `xterm &` geöffnet werden. Das nachgestellte „`&`“ ist dabei dringend erforderlich (zur Erklärung siehe Abschnitte 4 und 5).

1.3 Umgang mit verschiedenen Fenstern

Die vom Window-Manager verwaltete Oberfläche zeigt i.allg. mehrere Fenster an, die sich auch überlappen können. Außerdem befindet sich auf dem Bildschirm ein Symbol, das sich immer dann bewegt, wenn man die Maus auf ihrem Tablett hin- und herschiebt, der sogenannte *Mauszeiger*. Der Mauszeiger kann verschiedene Formen annehmen, je nachdem, ob er sich gerade in einem Fenster befindet oder nicht, und über welchem Teil eines Fensters er sich befindet.

Mit der Maus lassen sich Fenster verschieben, in ihrer Größe verändern, teilweise verdeckte Fenster lassen sich wieder ganz sichtbar machen usw. Die drei Maustasten (linke, mittlere und rechte) haben dabei unterschiedliche Funktionen, z.B. kann man die Größe eines

Fensters verändern, indem man den Mauszeiger auf eine der vier Ecken des Fensters bewegt, die linke Maustaste niederdrückt und gedrückt hält, die Maus bewegt, und dann die linke Maustaste wieder losläßt. Den Umgang mit der Maus erlernt man am besten durch Ausprobieren („Herumspielen“).

Die vom verwendeten Window-Manager `olwm` angebotenen Fenster haben i.allg. oben links ein kleines auf der Spitze stehendes Dreieck. Fährt man den Mauszeiger auf dieses Dreieck und drückt die rechte Maustaste, so erscheint ein Menü, das im folgenden als Dreiecksmenü bezeichnet wird. Fährt man den Mauszeiger (bei weiterhin gedrückter rechter Maustaste) auf einen der Menüpunkte und läßt dann den Mauszeiger los, so wird die angewählte Aktion ausgeführt. Ein wichtiger Menüpunkt ist `Quit`, mit dem das Fenster geschlossen werden kann (d.h. das Fenser verschwindet vom Bildschirm). Aktiviert man diesen Menüpunkt für das mit `Console` überschriebene Fenster, so wird nicht nur dieses Fenster geschlossen, sondern man wird ausgeloggt, genau so, als hätte man in diesem Fenster das Kommando `exit` eingegeben. Ohne Benutzung der Maus kann der Menüpunkt `Quit` auch durch gleichzeitiges Drücken einer der beiden Karo-Tasten links und rechts neben der Leertaste und des Buchstabens `Q` aktiviert werden.

2 Das Dateisystem

Programme, Texte, Bilder und Daten aller Art werden in *Dateien* (engl. *Files*) gespeichert. Da im allgemeinen eine sehr große Zahl solcher Dateien gespeichert werden muß, werden diese in *Verzeichnissen* (engl. *Directories*) zusammengefaßt, die hierarchisch angeordnet sind, also außer Dateien auch wieder Verzeichnisse enthalten können. Daraus ergibt sich eine baumartige Verzeichnisstruktur, auf deren jeweils unterster Ebene die Dateien liegen (vgl. Abb. 1; Verzeichnisnamen sind darin im Vergleich zu Dateinamen fettgedruckt).

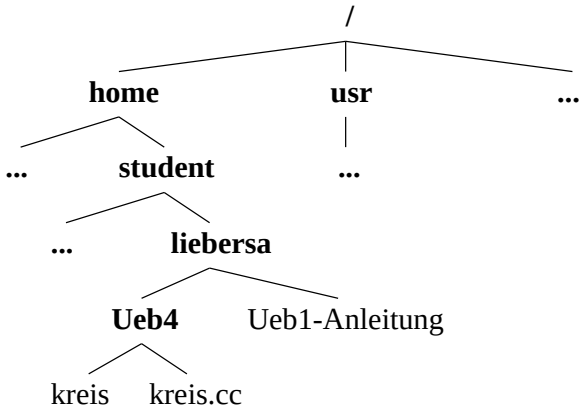

Abbildung 1: Ausschnitt aus der Verzeichnisstruktur

2.1 Verzeichnisse und Pfade

Jeder Benutzerin ist ein Heimatverzeichnis (*home directory*) zugeordnet, in dem sie i.allg. ihre Dateien ablegen wird. Dies ist ihr persönlicher Arbeitsbereich. In Abb. 1 ist ein Ausschnitt der Verzeichnisstruktur dargestellt, der das Heimatverzeichnis zur Benutzerkennung `liebersa` enthält[5].

Innerhalb eines mit `xterm` überschriebenen Fensters gibt es zu jedem Zeitpunkt ein *aktuelles* Verzeichnis (engl. *working directory*; direkt nach dem Einloggen ist das i.allg. das Heimatverzeichnis). Das aktuelle Verzeichnis kann für jedes der vorhandenen mit `xterm` überschriebenen Fenster unterschiedlich sein. Das Kommando `ls` zeigt den Inhalt des aktuellen Verzeichnisses an, unterschlägt aber Dateien, deren Name mit einem Punkt beginnt. Üblicherweise sind dies Dateien, die Daten über Voreinstellungen für bestimmte Programme enthalten und daher nur selten angezeigt werden müssen, die Ausgabe von `ls` also unnötig verlängern würden. Will man sie dennoch ausgeben lassen, so wird an `ls` die Option `-a` (für *all files*) angefügt: `ls -a`

Der Pfad zum aktuellen Verzeichnis kann durch den Befehl `pwd` ausgegeben werden[6]. Beispiel:

```
% pwd
/home/student/liebersa
```

Dies ist das Heimatverzeichnis zur Benutzerkennung `liebersa`. Dabei steht der „`/`“ ganz links für das Wurzelverzeichnis, und jeder weitere „`/`“ trennt einen Unterverzeichnisnamen vom nächsten.

Durch den Befehl `cd` *verzeichnis* kann das aktuelle Verzeichnis geändert werden. Die Angabe des Parameters *verzeichnis* kann auf zwei verschiedene Arten erfolgen, *absolut* oder *relativ*:

Absolute Pfadangabe: Die absolute Angabe eines Pfades für ein Verzeichnis beginnt grundsätzlich mit dem Wurzelverzeichnis „`/`“, es sei denn, das Zielverzeichnis ist Unterverzeichnis des Heimatverzeichnisses irgendeines Benutzers. Das eigene Heimatverzeichnis kann mit „`~`“ bezeichnet werden, das des Benutzers mit Benutzerkennung *user* durch `~`*user* (vgl. Kapitel 5). Ins eigene Heimatverzeichnis selbst gelangt man auch durch bloße Angabe von `cd`. Das Kommando `pwd` gibt den absoluten Pfad zum aktuellen Verzeichnis an.

Relative Pfadangabe: Die Pfadangabe *verzeichnis* gibt die vom aktuellen Verzeichnis aus zu durchlaufende Folge von Verzeichnissen an. Dabei muß jedes Verzeichnis im vorhergehenden enthalten sein. Ausnahme: Durch zwei Punkte gelangt man eine Ebene höher, d.h.

[5] Die Abbildung gibt die Verzeichnisstruktur wieder, wie sie sich der Benutzerin darstellt. Details der technischen Realisierung wie Softlinks oder über NFS gemountete Verzeichnisse werden hier nicht diskutiert.

[6] `pwd` steht für `print working directory`, nicht etwa für `password`.

„..“ bezeichnet immer das Verzeichnis, in dem das aktuelle enthalten ist. Ein Sonderfall ist auch ein einzelner Punkt, der das aktuelle Verzeichnis selbst bezeichnet.

Weitere Beispiele zur Verwendung von `cd` und `pwd`:

```
% cd ~liebersa/Ueb4
% pwd
/home/student/liebersa/Ueb4
% cd ../../../../usr
% pwd
/usr
% cd /home/student/liebersa
% pwd
/home/student/liebersa
```

Verzeichnisse können mit den Befehlen `mkdir` *verzeichnis* und `rmdir` *verzeichnis* angelegt und gelöscht werden. Dabei ist zu beachten, daß ein zu löschendes Verzeichnis leer sein muß, d.h. keine Dateien oder Unterverzeichnisse mehr enthalten darf. Zum nachträglichen Umbenennen einmal angelegter Verzeichnisse siehe die Erläuterung zum Kommando `mv`.

2.2 Manipulation von Dateien

Ebenso wie Verzeichnisse, können auch Dateien angelegt, gelöscht und umbenannt (verlagert) werden.

Angelegt wird eine neue, leere Datei z.B. mit dem Kommando `touch` *datei*, gelöscht wird eine beliebige Datei mit dem Kommando `rm` *datei*, Umbenennen und Verlagern einer oder mehrerer Dateien geschieht mit dem Befehl

`mv` *alter_name_und_pfad neuer_name_und_pfad*

Neue nichtleere Dateien können mit dem Befehl `cp` *quelldatei zieldatei* oder `cp` *quelldatei(en) zielverzeichnis* angelegt werden. Beispielsweise veranlaßt die Eingabe von

```
cd ~liebersa/Ueb4
cp kreis kreis.cc ~
```

daß Kopien der Dateien `Ueb4/kreis` und `Ueb4/kreis.cc` aus dem Heimatverzeichnis der Benutzerin mit Kennung `liebersa` im eigenen Heimatverzeichnis (bezeichnet mit „~“) angelegt werden. Eine andere Möglichkeit, neue Dateien anzulegen oder bestehende zu erweitern, ist z.B. die Umleitung von Ein- bzw. Ausgaben, die in Abschnitt 5 erläutert wird.

2.3 Zugriffsrechte

Die obigen Kommandos zum Erzeugen und Löschen von Dateien setzen voraus, daß die ändernde Person auch das Recht zu der Änderung besitzt. Von solchen Zugriffsrechten gibt es im wesentlichen drei verschiedene: Lese-, Schreib- und Ausführungsrechte.

Leserecht: Für Dateien bedeutet das Leserecht, daß sie einerseits mit den im nächsten Abschnitt beschriebenen Kommandos angezeigt, andererseits aber auch mit den obigen Kommandos kopiert werden dürfen. Bei Verzeichnissen bedeutet Leserecht, daß der Inhalt des Verzeichnisses z.B. mit den Kommandos `l` und `ls` aufgelistet werden kann.

Schreibrecht: Dateien und Verzeichnisse dürfen verändert werden. Das Verändern von Dateien geschieht häufig mittels sogenannter *Texteditoren* (siehe Abschnitt 3). Das Verändern eines Verzeichnisses bedeutet, in dem Verzeichnis enthaltene Dateien oder Unterverzeichnisse anzulegen, zu löschen oder umzubenennen.

Ausführungsrecht: Handelt es sich bei einer Datei um ein Programm, so ist das Ausführungsrecht nötig, um das Programm aufrufen zu dürfen. Insbesondere läßt sich durch Wegnahme dieses Rechts verhindern, daß versucht wird, nicht ausführbare Dateien (Texte, Daten) als Programm zu starten. Im Falle eines Verzeichnisses erlaubt dieses Recht das Wechseln in dieses Verzeichnis bzw. seine Benutzung auf einem Pfad in ein darin enthaltenes Unterverzeichnis (z.B. mit dem Kommando `cd`).

Die für eine Datei gesetzten Zugriffsrechte können für unterschiedliche Personengruppen verschieden sein. Dabei unterscheidet man zwischen den ganz persönlichen Zugriffsrechten der Benutzerin, den Rechten einer Benutzergruppe, der sie angehört, und den Rechten aller übrigen Benutzer. Die für diese drei Gruppierungen (Benutzerin, Gruppe, Rest der Welt) gesetzten Rechte werden vom Kommando `l` beim Auflisten eines Verzeichnisinhaltes mitangezeigt. Die folgende Zeile gibt ein Beispiel:

```
-rw-r-----   1 liebersa student       570 Aug 21 17:52 Ueb1-Anleitung
```

Im ersten Block der Ausgabe sind die Zugriffsrechte aufgelistet, wobei das führende `-` anzeigt, daß es sich um eine Datei handelt (im Falle eines Verzeichnisses stünde an dieser Stelle ein `d` für *directory*). Die restlichen Zeichen des ersten Blockes bilden drei Teilblöcke für die Rechte von Benutzerin, Gruppe und Rest der Welt. Die Teilblöcke haben die Form `rwx` mit `r` für Leserecht (*read*), `w` für Schreibrecht (*write*) und `x` für Ausführrecht (*execute*). Ist ein Buchstabe durch ein `-` ersetzt, so fehlt der entsprechenden Gruppierung dieses Recht. Im angegebenen Beispiel hat die Benutzerin `liebersa` Lese- und Schreibrecht für die Datei `Ueb1-Anleitung`. Benutzerinnen, die der Gruppe `student` angehören, dürfen die Datei lesen, aber nicht verändern. Alle anderen Benutzerinnen dürfen die Datei weder lesen noch verändern.

Die übrigen Informationen der obigen Anzeige betreffen die Anzahl der enthaltenen Unterverzeichnisse (bei Verzeichnissen), die Benutzerin, der die Datei gehört, die Gruppe, der

im zweiten Teilblock Rechte eingeräumt werden, die Größe der Datei in Bytes, das Datum der letzten Änderung und den Namen der Datei.

Um herauszufinden, welchen Benutzergruppen eine Benutzerin momentan angehört, kann man das Kommando `groups` verwenden:

```
% groups liebersa
liebersa : student pad_S98 inf2_S98
```

Zum Ändern von Zugriffsrechten dient das Kommando `chmod`. Diesem wird mitgeteilt

- auf welche Gruppierung sich die Änderung bezieht, wobei `u` für die Benutzerin (*user*), `g` für die Gruppe (*group*), `o` für den Rest der Welt (*others*) und `a` für alle drei zusammen (*all*) steht,
- ob die Rechte gesetzt oder weggenommen werden sollen (`+` für Setzen, `-` für Wegnehmen),
- welche Rechte betroffen sind (`r`, `w`, `x`) und
- für welche Dateien/Verzeichnisse die Änderungen wirksam werden sollen.

Zum Beispiel bewirkt die Eingabe von `chmod o+r Ueb1-Anleitung`, daß dem Rest der Welt Leserechte für die Datei `Ueb1-Anleitung` eingeräumt werden.

3 Anzeigen und Bearbeiten von Texten

Dateien, die lesbaren Text enthalten, können mit dem Kommando `cat` ausgegeben werden. Nach Eingabe von `cat` *datei* wird der Inhalt der Datei *datei* im Eingabefenster ausgegeben, wobei mit der Ausgabe nicht am unteren Fensterrand angehalten wird, sondern der Text nach oben durchläuft. Dadurch ist bei längeren Dateien letztlich nur das Ende des Textes zu sehen.

Komfortabler arbeiten die Kommandos `more` *datei* und `less` *datei*, bei denen die Ausgabe am unteren Fensterende stoppt und die Möglichkeit besteht, seitenweise vor- und zurückzublättern (s.u.).

Um Texte nicht nur anzeigen, sondern auch bearbeiten (*editieren*) zu können, stehen spezielle Programme, sogenannte *Texteditoren*, zur Verfügung. Es gibt eine Vielzahl solcher Editoren, von denen u.a. die im folgenden beschriebenen auf den Rechnern des Bereichs Informatik zur Verfügung stehen.

3.1 More und Less

Die Kommandos `more` *datei* und `less` *datei* zeigen die angegebene Datei seitenweise an. Eine Seite ist dabei gerade so groß wie das Fenster, in dem das Kommando aufgerufen wurde. Am unteren Fensterrand wird die Ausgabe unterbrochen und es sind u.a. folgende Eingaben möglich:

`Spacebar`	Eine Seite weiterblättern
`b`	Eine Seite zurückblättern
`/`*string*	Datei nach einer Zeichenkette durchsuchen
`q`	Ausgabe beenden („quit“)

Mit dem Kommando `less` kann man zusätzlich noch unter Benutzung der Pfeiltasten zeilenweise vor- und zurückblättern. I.allg. ist `less` also nützlicher als `more`. Am Ende der anzuzeigenden Datei beendet `more` sich aber von selbst, während man `less` auf jeden Fall mit `q` beenden muß. Je nach Situation ist das eine oder das andere sinnvoller.

3.2 Textedit

Dieser Editor wird mit dem Kommando `textedit` aufgerufen (besser noch mit `textedit &`, siehe Abschnitte 4 und 5) und erscheint in einem eigenen Fenster (wie in Abbildung 2).

Die aktuelle Textposition wird durch ein kleines Dreieck markiert und kann mit den Pfeiltasten, den darüber befindlichen Editiertasten `Home` usw. und der Maus verändert werden.

Die wichtigsten Funktionen zur Anzeige und Bearbeitung von Texten enthält das Menü `File`, das man — genau wie das Dreiecksmenü des Window-Managers `olwm` — mit der rechten Maustaste benutzt:

- `Open...`: Laden einer Datei zur Anzeige im Textfenster. Es erscheint ein weiteres Fenster, in dem die Datei ausgewählt werden kann. In diesem Fenster kann auch der Pfad zu einem anderen Verzeichnis angegeben werden, so daß auch Dateien editiert werden können, die sich nicht im aktuellen Verzeichnis befinden. Im Zusammenhang mit `textedit` wird auch die Bezeichnung *Folder* für Verzeichnis verwendet.

- `Save`: Eine bearbeitete Datei wird unter ihrem bereits vorhandenen Namen wieder abgespeichert. Erst nach dem Abspeichern sind die im `textedit`-Fenster vorgenommenen Änderungen in der Datei wirksam. Wird der Texteditor beendet, ohne daß die bearbeitete Datei abgespeichert wurde, werden alle seit dem letzten Abspeichern gemachten Änderungen verworfen. Beim Abspeichern bleibt die ursprüngliche Datei (ohne die letzten Änderungen) mit einem an den Namen angehängten `%` als Sicherheitskopie erhalten.

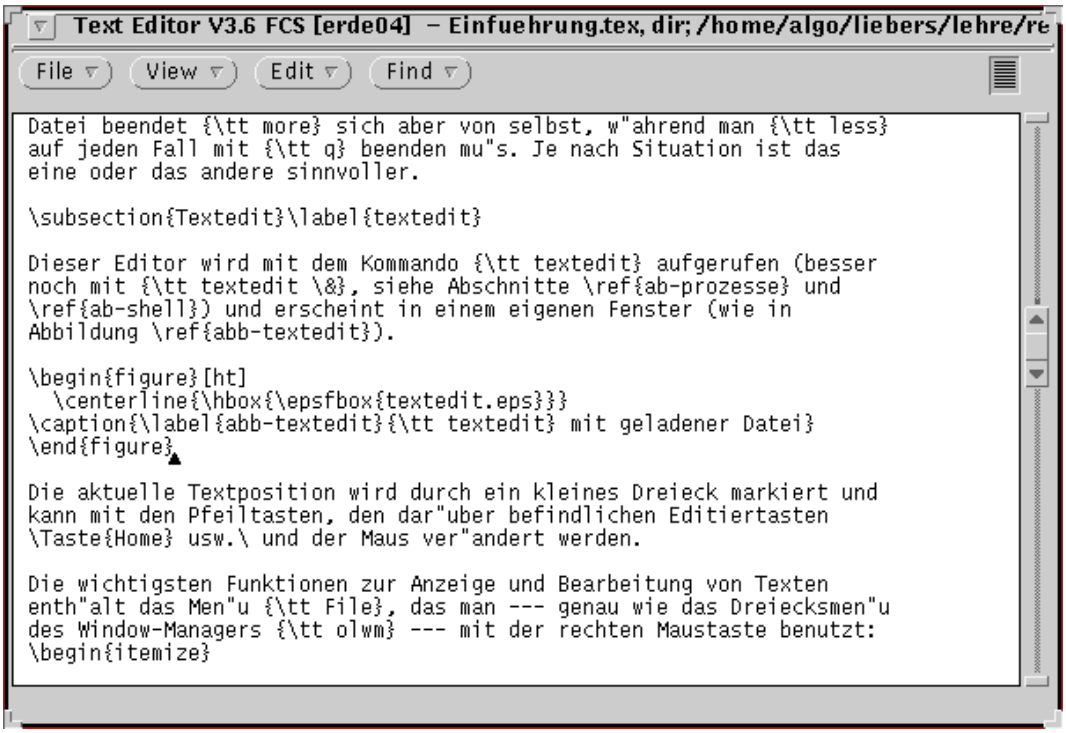

Abbildung 2: `textedit` mit geladener Datei. Die Datei zeigt übrigens einen Ausschnitt aus dem Text für die vorliegende Rechnereinführung. Es wird das Textsatzsystem LaTeX verwendet, in das [GH97] eine gute Einführung darstellt.

- `Save As...`: Die geänderte oder neu eingegebene Datei kann unter einem neuen Namen abgespeichert werden, der wie beim Öffnen in einem zusätzlich erscheinenden Fenster bestimmt wird.
- `Include...`: An die aktuelle Textpostion wird eine auszuwählende Textdatei eingefügt. Dabei ändert sich der Name der momentan bearbeiteten Datei nicht, sondern der Inhalt der ausgewählten Textdatei wird in die momentan bearbeitete Datei hineinkopiert. Die hineinkopierte Datei bleibt unverändert bestehen.
- `Empty Document`: Der Inhalt des Textfensters wird gelöscht und ein neuer Text kann eingegeben werden. Der neue Text gehört zunächst zu keiner Datei, bis er z.B. mittels `Save As...` abgespeichert wird.

Um den Texteditor zu beenden, ist, wie in Abschnitt 1.3 beschrieben, der Menüpunkt `Quit` aus dem vom Window Manager `olwm` zur Verfügung gestellten Dreiecksmenü auszuwählen.

3.3 Emacs

Der Editor `emacs` bietet viel mehr Möglichkeiten als der Editor `textedit`, ist jedoch komplizierter in der Bedienung. Er wird mit dem Kommando `emacs &` gestartet und erscheint ebenfalls in einem eigenen Fenster (siehe Abb. 3). Am oberen Rand des Fensters befinden sich Menüs, wobei wie beim `textedit` das Menü `File` die wichtigsten Menüpunkte enthält. Im Unterschied zum `textedit` werden die `emacs`-Menüs mit der *linken* Maustaste bedient.

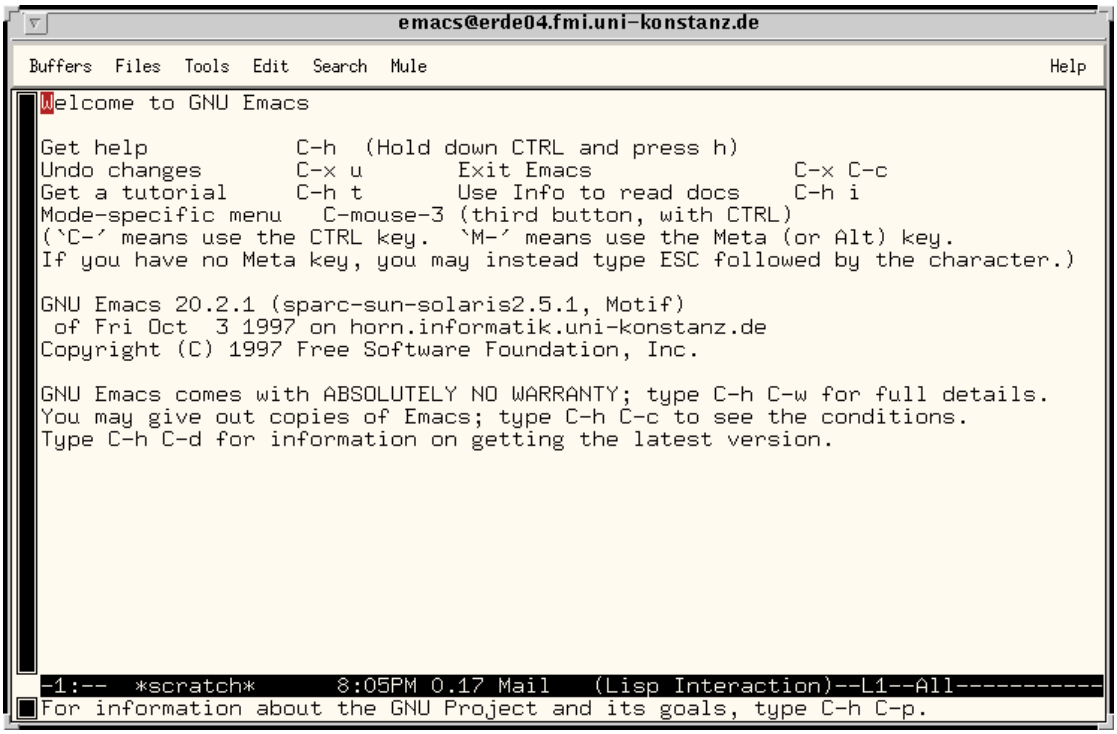

Abbildung 3: `emacs` nach dem Aufrufen

Zur Einarbeitung in den `emacs` ist es empfehlenswert, den *Tutorial* genannten Emacs-Lehrgang durchzuarbeiten. Dazu aktiviert man im Menü `Help` den Menüpunkt `Emacs Tutorial`.

Erwähnenswert ist noch das Menü `Buffers`. Wenn man nacheinander mehrere Dateien in den Emacs geladen hat, können diese Dateien abwechselnd bearbeitet werden, indem man die jeweils gewünschte Datei im Menü `Buffers` auswählt. Dieses „abwechselnde“ Bearbeiten verschiedener Dateien, ohne die Dateien jedes Mal erneut laden zu müssen, ist nur eines von vielen Konzepten, die `emacs` unterstützt und die über die Funktionalität von `textedit` weit hinausgehen. Interessierten Leserinnen sei das Buch *Learning GNU Emacs* aus der Reihe der Nutshell-Handbücher bei O'Reilly & Associates empfohlen [CRR96].

3.4 Ausdrucken

Das Ausdrucken einer z.B. mit einem Editor erstellten Datei ist mit dem Kommando `lp` möglich:

```
% lp datei
```

Soll sowohl die Vorder- als auch die Rückseite des Papiers bedruckt werden, verwendet man stattdessen

```
% lpv datei
```

Dieses Kommando druckt sowohl *Textdateien* (Texte also, die man z.B. mit dem Kommando `less` lesen kann), als auch PostScript-Dateien[7].

4 Prozesse

Betriebssysteme aus der Unix-Familie erlauben die „gleichzeitige“ Ausführung verschiedener Kommandos, Programme oder Unterprogramme, die in diesem Zusammenhang allgemeiner als Prozesse bezeichnet werden. Das Programm `olwm` (der Window-Manager) oder auch jedes mit `xterm` überschriebene Fenster stellt einen solchen Prozeß dar. Wird ein Editor gestartet, so stellt auch dieser Editor einen weiteren Prozeß dar. Die Prozesse laufen nicht wirklich gleichzeitig ab, aber durch die Schnelligkeit der Rechner und das i.allg. geschickt organisierte abwechselnde Bearbeiten der verschiedenen Prozesse durch das Betriebssystem kommt es dem Benutzer so vor.

Normalerweise geht mit dem Aufruf eines Prozesses die Kontrolle vom Eingabefenster (dem `xterm`) auf diesen über, so daß man in dem Fenster keine weiteren Kommandos aufrufen kann, bis der gestartete Prozeß (d.h. das gestartete Kommando) wieder beendet ist. Um jedoch mit weiteren Eingaben nach dem Aufruf (beispielsweise des Texteditors oder eines

[7] PostScript ist eine Seitenbeschreibungssprache, mit der man neben Texten vor allem auch Graphiken darstellen kann. Auch PostScript-Dateien kann man sich mit dem Kommando `less` ansehen, dabei bekommt man aber nur die PostScript-Befehle selber und nicht die Texte oder Graphiken, die durch diese Befehle erzeugt werden sollen, zu sehen. Beim Ausdrucken ist es wichtig, daß der Drucker erkennt, daß es sich um ein PostScript-Dokument handelt. Ist dies nicht der Fall, wird der mit `less` lesbare Inhalt der PostScript-Datei ausgedruckt, und das bedeutet i.allg. dutzende Seiten Papierverschwendung. Der Drucker erkennt nur dann, daß er den Inhalt einer Datei als Folge von PostScript-Befehlen interpretieren soll, wenn die Datei mit den zwei Zeichen `%!` beginnt. Bei PostScript-Dateien, die mit Programmen wie `dvips`, `a2ps`, `2up` etc. erstellt worden sind, ist das i.allg. gewährleistet (`dvips` ist im Zusammenhang mit LaTeX ein wichtiges Kommando, und mit `a2ps` bzw. `2up` kann man eine Textdatei bzw. eine PostScript-Datei z.B. so umformatieren, daß zwei Seiten nebeneinander auf eine Seite Papier gedruckt werden). Die Namen von PostScript-Dateien enden üblicherweise auf `.ps`.

weiteren Eingabefensters) nicht bis zur Beendigung des neuen Prozesses warten zu müssen, kann dem Aufruf ein Kaufmanns-Und („`&`“) angehängt werden, das den neuen Prozeß als sogenannten *Hintergrundprozeß* kennzeichnet. Man probiere zur Übung das Kommando `xterm` einmal mit und einmal ohne nachgestelltem „`&`“ aus, und versuche, jeweils weitere Eingaben im Aufruffenster (vor und nach dem Schließen des neuen Fensters) zu tippen[8].

Die derzeit auf dem Rechner laufenden Prozesse können mit dem Kommando `psu` angezeigt werden. Die Ausgabe besteht aus einer Liste aller Prozesse des aufrufenden Benutzers unter Angabe seiner Benutzerkennung, einer Prozeßnummer, der Aufrufzeit und einiger weiterer Angaben.

Eine Liste aller auf dem Rechner laufenden Prozesse (mit der Zuordnung zu den Benutzern) liefert das Kommando `psa`[9].

5 Die Shell

Eine sogenannte *Shell* ist ein Programm, das es dem Benutzer gestattet, Kommandos einzugeben, und das diese Kommandos dann startet. Im allgemeinen wird beim Einloggen mindestens eine solche Shell gestartet. In jedem mit `xterm` überschriebenen Fenster können Kommandos eingegeben werden, d.h. in jedem dieser Fenster läuft ständig eine solche Shell. Der Benutzer erkennt an der i.allg. mit einem Prozentzeichen oder einem Dollarzeichen endenden Eingabeaufforderung, daß die Shell auf die Eingabe eines Kommandos wartet.

So wie es verschiedene Editorprogramme gibt, gibt es auch unterschiedliche Shellprogramme. Eines davon ist die `csh` (C-Shell[10]). Eine Erweiterung der `csh` ist die im Bereich Informatik oft verwendete `tcsh`. Ein anderes Shell-Programm heißt `sh` (die sogenannte Bourne-Shell[11]). Die `bash` ist eine Erweiterung der `sh`, die ebenfalls im Bereich Informatik Verwendung findet. Im folgenden wird ein Ausschnitt der Funktionalität von `tcsh` und `bash` beschrieben.

Weiterführende (recht umfangreiche) Dokumentation findet man in den mit dem Kommando `man` abrufbaren Beschreibungen zu `csh` und `tcsh` bzw. zu `sh` und `bash`, sowie in den Büchern *A Student's Guide to Unix* [Hah93] und *UNIX in a Nutshell* [Gs].

Prompt

Die Eingabeaufforderung endet bei der `tcsh` i.allg. mit „%“ bzw. bei der `bash` mit „$“. Sie zeigt dem Benutzer an, daß er ein Kommando eingeben kann. In den Beispielen in dieser Einführung wird i.allg. eine `tcsh` angenommen.

[8] Die Fähigkeit, Prozesse „im Hintergrund“ zu starten, gehört zu den Eigenschaften der jeweiligen Shell. Siehe dazu Abschnitt 5.

[9] `psu` und `psa` selbst sind keine Programme, sondern Abkürzungen für längere Kommandos, die für Benutzer des Bereichs Informatik definiert wurden. Vgl. Abschnitt 5.

[10] Das „C“ in C-Shell kommt vom Namen der Programmiersprache C.

[11] Benannt nach ihrem Autor, S.R. Bourne.

Verzeichnis `~`*user*

Das Heimatverzeichnis eines Benutzers mit der Benutzerkennung *user* kann mit der Pfadangabe `~`*user* angesprochen werden.

```
% cd ~liebersa
```

bewirkt, daß das Heimatverzeichnis der Benutzerin mit der Benutzerkennung `liebersa` zum aktuellen Verzeichnis wird.

Umlenken der Ausgabe eines Kommandos

Mit dem Sonderzeichen „`>`“ kann die Ausgabe eines Kommandos in eine Datei „umgelenkt“ werden, statt auf dem Bildschirm zu erscheinen. Bsp.:

```
% cal 1995 > /tmp/cal_out
```

bewirkt, daß die Datei `/tmp/cal_out` (die im Falle der `tcsh` vorher noch nicht existieren darf) angelegt wird, und daß die Ausgabe des Kommandos `cal 1995` in dieser Datei abgespeichert wird. Existiert eine Datei bereits, so kann man mit den Sonderzeichen „`>>`“ die Ausgabe eines Kommandos an diese Datei anhängen. Bsp.:

```
% cal 1996 >> /tmp/cal_out
```

bewirkt, daß die Ausgabe des Kommandos `cal 1996` an die im vorigen Beispiel erzeugte Datei `/tmp/cal_out` angehängt wird. Diese Datei enthält also anschließend die Kalender für die Jahre 1995 und 1996.

Pipes

Es ist möglich, Kommandos mit dem Sonderzeichen „`|`“ „zusammenzustecken“, d.h. die Ausgabe eines Kommandos als Eingabe für ein weiteres Kommando zu verwenden (pipe – engl. Rohr, durch Röhren leiten). Bsp.:

```
% sysinfo | less
```

bewirkt, daß die Ausgabe des Kommandos `sysinfo` nicht auf dem Bildschirm angezeigt wird, sondern als Eingabe für das Kommando `less` verwendet wird. `less` zeigt diese Eingabe dann seitenweise an. Eine Kommandofolge, die (falls die Datei `/tmp/sysinfo_out` vorher nicht existiert) dasselbe bewirken würde, wäre z.B.

```
% sysinfo > /tmp/sysinfo_out
% less /tmp/sysinfo_out
% rm /tmp/sysinfo_out
```

Wildcards

Bei der Angabe von Datei- oder Verzeichnisnamen als Argumente für Kommandos haben die Sonderzeichen „`?`“ und „`*`“, die *Platzhalter* (engl. *Wildcards*), folgende

Bedeutung: „?“ steht für genau ein Zeichen, und „*“ steht für eine beliebige (auch leere) Folge von Zeichen.

Beispiel:

```
% cp * /tmp
```

bewirkt, daß alle Dateien im aktuellen Verzeichnis in das Verzeichnis `/tmp` kopiert werden (dabei sind jedoch Dateien, deren Namen mit einem Punkt beginnen, wieder ausgenommen, vgl. auch Abschnitt 2.1). Weitere Beispiele, gefolgt von den jeweiligen Ausgaben:

```
% cd /home/student
% ls -d b*l
bendel    berchtol  buerkel
% ls -d schmidt?
schmidta  schmidte  schmidtn  schmidts  schmidtt
```

Wildcards müssen immer dann mit besonderer Vorsicht verwendet werden, wenn das Kommando nicht rückgängig zu machende Veränderungen bewirkt (wie z.B. `rm *`).

Aliases

Für häufig verwendete Kommandos oder Kommandofolgen können Abkürzungen (sogenannte Aliases) definiert werden. So ist das in Abschnitt 4 eingeführte Kommando `psa` kein Programm, sondern ein Alias. `psa` steht für „`ps -ef | more`“, `ps` selbst ist dagegen ein Programm. Mit dem Kommando `alias` kann man alle definierten Abkürzungen anzeigen lassen.

Variable

Die `tcsh` und die `bash` unterhalten eine Liste von Steuerparametern, die das Verhalten der Shell beeinflussen. Mit dem Kommando `echo $`*Variablenname* kann man sich den Wert einer Variablen anschauen. Einer Variablen kann ein Wert zugewiesen werden, muß aber nicht. Ausgewählte Variablen und ihre Bedeutung:

`rmstar`
: Ist diese Variable gesetzt, so startet die `tcsh` nach Eingabe des Kommandos `rm *` dieses Kommando erst, nachdem der Benutzer eine entsprechende Frage, ob er wirklich alle Dateien löschen wolle, bejaht hat. Es kommt dabei nur darauf an, ob die Variable gesetzt ist oder nicht. Ihr Wert bleibt unberücksichtigt.

`shell`
: Enthält diese Variable den Wert `/usr/local/bin/tcsh`, so ist das momentan laufende Shellprogramm eine `tcsh`.

`BASH`
: Enthält diese Variable den Wert `/usr/local/bin/bash`, so ist das momentan laufende Shellprogramm eine `bash`.

`HOME`

Diese Variable enthält den Namen des Heimatverzeichnisses als Wert.

`PATH`

Diese Variable enthält die Namen der Verzeichnisse, in denen die Shell die vom Benutzer eingegebenden Kommandos als Dateien sucht.

Mit dem Kommando `which` *kommandoname* kann man herausfinden, in welchem dieser Verzeichnisse ein bestimmtes Kommando gefunden wird. Handelt es sich bei *kommandoname* um ein Alias, so zeigt `which` dies bei Verwendung der `tcsh` auch an.

In den folgenden Beispielen existieren die Variablen `rmstar` und `HOME`, während die Variable `blablu` nicht existiert. `rmstar` wurde kein Wert zugewiesen. `HOME` hat dagegen einen Wert:

```
% cd /tmp
% echo $rmstar
% rm *
Do you really want to delete all files? [n/y] n
% echo $HOME
/home/student/liebersa
% echo $blablu
blablu: Undefined variable.
% which date
/usr/local/bin/date
% which psa
psa:     aliased to ps -ef | more
```

Hintergrundbearbeitung

Wird ein Prozeß gestartet, der ein eigenes Fenster aufmacht (wie `xterm` oder `emacs`), so kann man mit dem Sonderzeichen „`&`“ bewirken, daß die `tcsh` bzw. die `bash` nicht auf die Beendigung dieses Prozesses wartet, sondern sofort weitere Eingaben akzeptiert (vgl. Abschnitt 4).

History

Die `tcsh` und die `bash` „merken“ sich die vom Benutzer eingegebenen Kommandos (bis zu einer maximalen Anzahl). Diese Kommandos kann man wiederholt ausführen lassen:

```
% !!
```

bewirkt, daß das zuletzt eingegebene Kommando wiederholt wird.

```
% !nummer
```

bewirkt, daß das Kommando mit der angegebenen Nummer wiederholt wird (die Nummer eines Kommandos ist i.allg. an der Eingabeaufforderung ablesbar).

```
% history
```

bewirkt, daß alle früheren Kommandos, die die Shell sich gemerkt hat, zusammen mit ihrer Nummer angezeigt werden.

Außerdem kann man mit den Pfeiltasten in der Liste der gemerkten Kommandos hin- und herlaufen. Hat man auf diese Art das gewünschte Kommando zur Anzeige gebracht, kann man es ggf. noch verändern und dann wie gewohnt durch Drücken der Taste `Return` abschicken.

File Name Completion

Wenn man als Argument für ein Kommando einen Datei- oder Verzeichnisnamen angibt, braucht man den Namen u.U. nicht vollständig zu tippen. Stattdessen beginnt man, den Namen einzugeben, und sobald der Name durch die bereits eingegebenen Anfangsbuchstaben eindeutig bestimmt ist, läßt man ihn durch Drücken der Taste `Tab` automatisch vervollständigen. Beispiel:

```
% less ~liebersa/Ueb1Tab
```

ergibt

```
% less ~liebersa/Ueb1-Anleitung
```

Built-in Commands

Die `tcsh` und die `bash` kennen einige „Kommandos“, bei denen es sich nicht um Programme handelt, die sich in einem der im Wert der Variablen `PATH` aufgezählten Verzeichnisse befinden, sondern um Befehle an die `tcsh` (bzw. an die `bash`) selbst. Diese „built-in“-Kommandos starten auch keinen eigenen Prozeß. Beispiele sind die Kommandos `alias`, `echo` und `history`.

Konfigurationsdateien

Das Verhalten der `tcsh` und der `bash` kann auf vielfältige Weise den Wünschen des Benutzers angepaßt werden, indem bestimmte Variablen gesetzt werden oder nicht, indem der Wert bestimmter Variablen verändert wird, oder indem Aliase gesetzt werden. Für all diese Setzungen gibt es im Heimatverzeichnis die Datei `.cshrc` (für die `tcsh`) bzw. `.bashrc` (für die `bash`). Die darin enthaltenen Kommandos werden automatisch abgearbeitet, wenn eine `tcsh` bzw. eine `bash` gestartet wird. Dies geschieht z.B. immer, wenn ein neues `xterm`-Fenster aufgemacht wird.

Wenn für neue Benutzer im Bereich Informatik ein Arbeitsbereich angelegt wird, werden Konfigurationsdateien in das Heimatverzeichnis kopiert, die z.B. einige Alias-Setzungen wie `psa` enthalten. Will ein Benutzer diese Setzungen ergänzen, so sollte dies in der Datei `.cshrc.`*user* (bzw. `.bashrc.`*user*) und nicht in der Datei `.cshrc` (bzw. `.bashrc`) geschehen. Die Datei `.cshrc.liebersa` könnte also z.B. weitere Alias-Setzungen für die Benutzerin aus Abschnitt 2.1 enthalten.

6 EMail

Jede Benutzerin hat die Möglichkeit, einer oder mehreren anderen Benutzerinnen (die ihre Benutzerkennung auch auf anderen Rechnersystemen als dem des Bereichs Informatik haben können) auf elektronischem Wege Mitteilungen (sogenannte *EMails*, electronic mails, elektronische Post) zukommen zu lassen. Wer eine Zugangsberechtigung auf Rechenrn des Bereichs Informatik hat, sollte regelmäßig seine neu eingetroffenen EMails lesen, denn wichtige Informationen von den Systemadministratorinnen, von Übungsgruppenleiterinnen oder auch von Seiten der Fachschaft werden häufig per EMail bekanntgegeben.

Jede Benutzerin hat (mindestens) eine EMail-Adresse. Für Mitteilungen an Benutzerinnen im Bereich Informatik genügt die Benutzerkennung als EMail-Adresse. Für Benutzerinnen außerhalb des Bereichs Informatik muß man zusätzlich den Namen des Rechnernetzes kennen, in dem die Adressatin einer EMail ihre Benutzerkennung hat.

Die Adresse der Benutzerin Annegret Liebers aus Abschnitt 2.1 ist also `liebersa` (für Benutzerinnen innerhalb des Bereichs Informatik) bzw.

```
liebersa@fmi.uni-konstanz.de
```

(für Benutzerinnen außerhalb des Bereichs Informatik). Die vollständige EMail-Adresse einer Benutzerin ist oft so aufgebaut, daß man an ihr ablesen kann, an welchem geographischen Ort und in welcher Organisationseinheit sich das Rechnernetz befindet, an dem die Benutzerin arbeitet. Ganz rechts steht dabei oft eine Landeskennung (`de` steht für Deutschland).

Das Programm `elm` ist eines von vielen, mit dem das Schreiben und Lesen von EMail möglich ist. Ein anderes, ebenfalls auf den Rechenrn des Bereichs Informatik installiertes, ist z.B. `mailtool`. Wir gehen im folgenden exemplarisch auf das Programm `elm` ein.

`elm` wird *auf keinen Fall* als Hintergrundprozeß aufgerufen, und es ist wichtig, daß man das Programm `elm` wieder verlassen hat, bevor man sich ausloggt. Wenn man `elm` zum allerersten Mal aufruft, wird man gefragt, ob `elm` ein Unterverzeichnis namens `.elm` und eines namens `Mail` anlegen darf. Man antwortet beide Male mit `y`. Das Fenster, in dem `elm` gestartet wurde, sieht nun ähnlich aus wie in Abb. 4. Jede EMail, die für die Benutzerin angekommen ist, wird in einer Zeile angezeigt. In Abb. 4 sind vier EMails zu sehen. Das `N` ganz links vor der ersten EMail steht für *new*, d.h. die EMail wurde noch nicht gelesen. Das Datum zeigt an, an welchem Tag die EMail angekommen ist. Außerdem werden jeweils die Absenderin der EMail und die Anzahl der in der EMail enthaltenen Zeilen angegeben. Weiterhin kann eine EMail ein sogenanntes *Subject* haben. Das Subject ist eine Betreff-Zeile oder auch Überschrift. Die in Abb. 4 gezeigte Liste der momentan vorhandenen EMails wird im Zusammenhang mit dem Programm `elm` auch mit *Index* bezeichnet.

Um eine EMail zu schreiben, wählt man aus dem angezeigten Menü die Funktion `mail` aus. Man tippt dazu jedoch nur den Anfangsbuchstaben `m` (*ohne* Drücken der Taste `Return`).

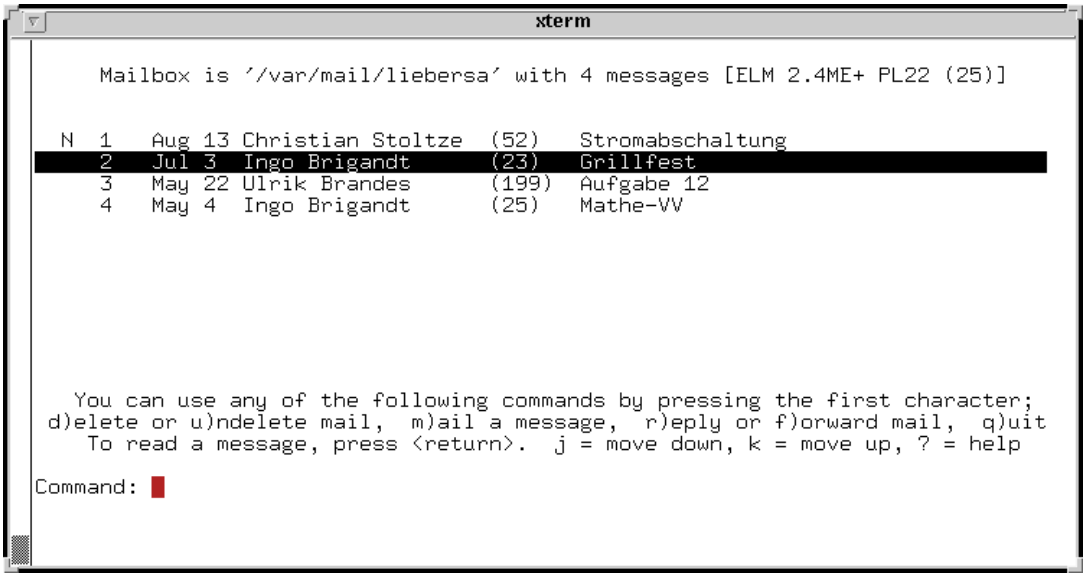

Abbildung 4: Programm `elm` zum Bearbeiten von EMail

Dann beantwortet man die von `elm` gestellten Fragen nach der Adressatin (`Send the message to:`), dem Subject und nach weiteren Adressatinnen, die eine Kopie erhalten sollen (`Copies to:`). Um die EMail zu erzeugen, legt `elm` nun eine Datei an, ruft einen Editor auf, in den die Datei geladen wird, und wartet solange, bis die Benutzerin die EMail geschrieben und den Editor beendet hat (dabei ist es wichtig, daß die angelegte Datei im Editor abgespeichert wurde, bevor der Editor beendet wird). Der aufgerufene Editor ist standardmäßig der `vi`. Man kann `elm` aber so konfigurieren (s.u.), daß ein anderer Editor verwendet wird.

Nach Beenden des Editors bietet `elm` der Benutzerin ein weiteres Menü an, aus dem sie i.allg. `s` für `send` (wieder ohne `Return`) auswählen wird. Daraufhin schickt `elm` die EMail ab. Die zuvor angelegte Datei wird automatisch von `elm` gelöscht. Anstatt die EMail sofort mittels `send` abzuschicken, kann sie aber auch noch einmal verändert werden (`edit message`), es können Adressatinnen hinzugefügt oder entfernt werden (`edit headers`), oder die EMail kann sogar ganz verworfen werden (`forget it`).

Um eine der angekommenen EMails zu lesen, bewegt man den schwarzen Balken mit Hilfe der Pfeiltasten[12] auf die Zeile mit der EMail, die man lesen möchte. Drückt man dann die Leertaste, so wird die aktuelle EMail unter Benutzung des Programms `less` seitenweise angezeigt. Hat man die EMail gelesen (und das Programm `less` wie gewohnt mit `q` beendet), kommt man durch Tippen von `i` (ohne `Return`) wieder zurück zum Index.

Alternativ zur Benutzung der Pfeiltasten kann man eine angezeigte EMail als aktuelle EMail auswählen, indem man die Nummer der EMail eintippt und die Eingabe mit der Taste `Return` beendet.

[12] Das Programm `elm` unterstützt die Mausbenutzung nicht.

EMails kann man außerdem löschen (`delete`), beantworten (`reply`), an andere Benutzerinnen weiterschicken (`forward`), ausdrucken (`print`) u.v.a.m. Um das Programm `elm` zu verlassen, tippt man `q` (für `quit`). Hat man EMails mit `d` (für `delete`) gelöscht, so wird die Benutzerin jetzt gefragt, ob diese EMails tatsächlich gelöscht werden sollen.

Alle für eine Benutzerin namens *user* ankommenden EMails werden in der Datei `/var/mail/`*user* gespeichert (vgl. Abb. 4, oberste Zeile). Alle abgeschickten EMails können vom Programm `elm` im Heimatverzeichnis der Benutzerin in der Datei `~/Mail/sent` gespeichert werden (s.u.). Erhält man EMails, die man nach dem Lesen gerne aufheben möchte, kann man sie in der Datei `~/Mail/received` im Heimatverzeichnis ablegen (d.h. man löscht sie nicht, und beantwortet beim Verlassen von `elm` die Frage „`Move read messages to "received" folder? (y/n)`“ mit `y`).

Die Dateien `sent` und `received` enthalten die abgeschickten bzw. abgelegten EMails in demselben Format, in dem angekommene EMails sonst in `/var/mail/`*user* stehen, und können ebenfalls mit dem Programm `elm` bearbeitet werden. `elm` bietet dafür die Funktion `change folder` an. Auf die Frage `Change to which folder:` antwortet man mit !`Return` für die sogenannte *incoming mailbox* `/var/mail/`*user*, mit `>``Return` für die Datei `~/Mail/received` mit den aufgehobenen EMails, bzw. mit `<``Return` für die Datei `~/Mail/sent` mit den abgeschickten EMails.

Zahlreiche Aspekte des Verhaltens von `elm` können über die Konfigurationsdatei `.elm/elmrc` im Heimatverzeichnis der Benutzerin gesteuert werden. Um zu erreichen, daß diese Datei existiert, wählt die Benutzerin aus dem Hauptmenü durch Tippen des Buchstabens `o` die (standardmäßig nicht angezeigte) Funktion `options` aus, und in dem dann angezeigten Menü erst `>` zum Anlegen der Datei `elmrc` und dann `i` für die Rückkehr zum Index. Sie verläßt nun `elm` und verändert die Inhalte der Datei `.elm/elmrc` mit Hilfe eines Editors: Das Einfügen der Zeile `copy = ON` sorgt z.B. dafür, daß beim nächsten Aufruf von `elm` von jeder abgeschickten EMail eine Kopie aufgehoben wird, und das Einfügen der Zeile `editor = emacs` stellt den beim Schreiben von EMails benutzten Editor von `vi` auf `emacs` um.

Wenn eine neue EMail ankommt, oder wenn beim Einloggen noch nicht gelesene EMails für die Benutzerin vorliegen, wird dies durch das mit `xbiff` überschriebene Fenster angezeigt, indem das Fenster statt eines weißen einen schwarzen Hintergrund erhält.

EMail ist nur eine der Kommunikations- und Informationsmöglichkeiten, die das Internet, an das auch die Rechner des Bereichs Informatik angeschlossen sind, bietet. Interessierten Leserinnen sei das Buch *The Whole Internet* aus der Reihe der Nutshell-Handbücher bei O'Reilly & Associates empfohlen [Kro].

Zum Schluß noch ein Aufruf zur Zurückhaltung: Genauso wie Massenwurfsendungen bei der „normalen“ Post i.allg. verpönt sind, gehört es unter EMail-Benutzerinnen zum guten Ton, andere Benutzerinnen nicht mit unerwünschten EMails, insbesondere mit kommerzieller Werbung, zu überfluten.

7 Referenz

Die nachfolgende Übersicht faßt die in den vorstehenden Abschnitten aufgetretenen Befehle und Meta-Zeichen zusammen. Mögliche Parameter sind *kursiv* angegeben und sinngemäß zu ersetzen, z.B. steht `cal` *jahr* für Eingaben der Form `cal 1994` oder `cal 1995`.

Kommandos, die i.allg. als Hintergrundprozesse gestartet werden sollten, sind im folgenden bereits mit dem anzuhängenden „`&`“ angegeben.

7.1 Allgemeine Kommandos

`cal (calendar)`	
`cal`	Ausgabe eines Kalenders für den aktuellen Monat.
`cal` *jahr*	Ausgabe eines Kalenders für das Jahr *jahr*.
`cal` *monat jahr*	Ausgabe eines Kalenders für den Monat *monat* im Jahr *jahr*. Man beachte die unterschiedlichen Ausgaben, die `cal 1995` und `cal 95` bewirken.
`date`	
`date`	Ausgabe von heutigem Datum und aktueller Uhrzeit.
`man (manual)`	
`man` *kommando*	Erläuterungen über Syntax und Wirkungsweise eines Kommandos. Die sogenannte `man-Seite` oder `man-page` wird unter Verwendung des Kommandos `less` angezeigt.
`xterm (X-terminal)`	
`xterm &`	Öffnen eines weiteren Fensters zur Kommando-Eingabe.
`sysinfo (system information)`	
`sysinfo`	Ausgaben von Informationen zum Modell und zur Hardware-Ausstattung der benutzten Workstation.

7.2 Das Dateisystem

`ls (list)`

`ls`	Auflistung aller nicht mit einem Punkt beginnenden Dateien des aktuellen Verzeichnisses.
`ls -a`	Auflistung aller Dateien (auch der mit einem Punkt beginnenden) des aktuellen Verzeichnisses.
`ls` *verzeichnis*	Auflistung der nicht mit einem Punkt beginnenden Dateien des angegebenen Verzeichnisses.
`ls -a` *verzeichnis*	Auflistung aller (auch der mit einem Punkt beginnenden) Dateien des angegebenen Verzeichnisses.
`ls -d` *verzeichnis*	Auflistung des Verzeichnisses ohne die darin enthaltenen Dateien, das heißt, daß nur die Informationen für das Verzeichnis selber angezeigt werden.

`l (list)`

`l`	Auflistung aller (auch der mit einem Punkt beginnenden) Dateien des aktuellen Verzeichnisses mit Angabe weiterer Informationen, insbesondere der Zugriffsrechte und des Zeitpunkts der letzten Änderung (*modification time*). Wenn die Ausgabe länger ist als die Größe des aktuellen `xterm`, so wird sie unter Verwendung des Kommandos `more` seitenweise angezeigt. `l` ist ein Alias.
`l` *verzeichnis*	Wie `l`, aber Auflistung aller Dateien des angegebenen Verzeichnisses.

`touch`

`touch` *datei*	Anlegen einer leeren Datei oder Setzen der *modification time* einer existierenden Datei auf den momentanen Zeitpunkt. Die *modification time* wird z.B. vom Kommando `l` angezeigt.

`cp (copy)`

`cp` *quelle ziel*	Kopieren der Datei mit Namen *quelle* in die Datei *ziel*. Eine eventuell vorhandene alte Datei *ziel* wird überschrieben. In beiden Parametern sind Pfadangaben erlaubt. *quelle* muß dabei den Namen genau einer Datei bezeichnen, während *ziel* auch einfach der Name eines Verzeichnisses sein darf. In diesem Fall wird die Datei *quelle* unter Beibehaltung ihres Namens in das Verzeichnis *ziel* kopiert.
`cp` *datei1 ... verzeichnis*	Kopieren mehrerer Dateien in das Verzeichnis *verzeichnis*. Vgl. S. 7.

`mv (move)`

`mv` *quelle ziel*	Verlagern einer Datei. Auch hier sind Pfadangaben möglich, ohne sie wird die Datei lediglich umbenannt.
`mv` *datei1 ... verzeichnis*	Verlagern mehrerer Dateien in das Verzeichnis *verzeichnis*.

`rm (remove)`

`rm` *datei*	Löschen einer Datei.

`pwd (print working directory)`

`pwd`	Ausgabe des Pfades zum aktuellen Verzeichnis. Vgl. S. 6.

`cd (change directory)`

`cd`	Wechsel in das Heimatverzeichnis.
`cd` *verzeichnis*	Wechsel in das angegebene Verzeichnis.
`cd ..`	Wechsel in das nächsthöhere Verzeichnis. Vgl. S. 6.

`mkdir (make directory)`

`mkdir` *verzeichnis*	Anlegen eines neuen Unterverzeichnisses.

`rmdir (remove directory)`

`rmdir` *verzeichnis*	Löschen eines Verzeichnisses. Das Verzeichnis muß dazu leer sein.

7.3 Anzeigen und Bearbeiten von Texten

`cat (concatenate)`

`cat` *datei*	Durchlaufende Ausgabe des Inhalts einer Textdatei.
`cat` *datei1 datei2 ...*	Aneinandergereihte Ausgabe des Inhalts der angegebenen Dateien. Vgl. S. 9.

`more`

`more` *datei*	Seitenweise Ausgabe des Inhalts einer Textdatei. Vgl. S. 10.

`less`

`less` *datei*	Ähnlich wie `more`. Vgl. S. 10.

`textedit`

`textedit &`	Starten des Texteditors `textedit` als Hintergrundprozeß.
`textedit` *datei* `&`	Starten des Texteditors `textedit` als Hintergrundprozeß mit gleichzeitigem Laden einer Textdatei. Vgl. S. 10.

`emacs`

`emacs &`	Starten des Texteditors `emacs` als Hintergrundprozeß. Vgl. S. 12.

`lp (line printer)`

`lp datei`	Ausdrucken einer Textdatei oder einer PostScript-Datei. Vgl. S. 13.

7.4 Prozesse

`& (and)`

kommando `&`	Starten eines neuen Prozesses, der das angegebene Kommando im Hintergrund ausführt. Vgl. S. 14.

`who`

`who`	Ausgabe der Kennungen aller auf dem Rechner eingeloggten Benutzer.

`psu (processes of user)`

`psu`	Ausgabe aller eigenen Prozesse auf diesem Rechner. `psu` ist ein Alias.

`psa (processes of all users)`

`psa`	Ausgabe aller Prozesse auf diesem Rechner mit Angabe der Benutzer, denen sie gehören. `psa` ist ein Alias.

7.5 Die Shell

Vergleiche auch S. 14 ff.

~ (home directory)

`~`	Bezeichnet das eigene Heimatverzeichnis.
`~`*user*	Bezeichnet das Heimatverzeichnis der angegebenen Benutzerin.

> >> (output redirection)

kommando `>` *datei*	Die Ausgabe des angegebenen Kommandos wird nicht auf dem Bildschirm angezeigt, sondern in der Datei *datei* gespeichert (in die Datei *datei* umgelenkt).
kommando `>>` *datei*	Wie bei „`>`“, aber die Ausgabe wird an den bereits existierenden Inhalt der Datei *datei* angehängt.

| (pipe)

kommando1 `|` *kommando2*	Die Ausgabe von *kommando1* wird nicht auf dem Bildschirm angezeigt, sondern als Eingabe für das Kommando *kommando2* benutzt.

*** ? (wildcards)**

`*`	Der Stern als Teil von Pfadangaben steht für eine beliebige (u.U. auch leere) Folge von Zeichen.
`?`	Das Fragezeichen als Teil von Pfadangaben steht für genau ein Zeichen.

alias

`alias`	Ausgabe aller definierten Abkürzungen.

echo

`echo $`*variablenname*	Ausgabe des Wertes der Shellvariablen *variablenname* bzw. einer Fehlermeldung, falls die Variable nicht definiert ist.

!! !*nummer* history

`!!`	Wiederholung des zuletzt ausgeführten Kommandos.
`!`*nummer*	Wiederholung des Kommandos mit der angegebenen Nummer.
`history`	Ausgabe aller Kommandos, die mit `!`*nummer* wieder abgerufen werden können.
Pfeiltasten	Durch Betätigen der Pfeiltasten können in der „History“ enthaltene Kommandos wieder angezeigt, nach Belieben verändert und nach Bedarf durch Drücken der Taste `Return` abgesetzt werden.

`Tab` `(file name completion)`

kommando unvollständigepfadangabe`Tab`

Vervollständigen der unvollständigen Pfadangabe (soweit das eindeutig geht).

7.6 EMail

`finger`

`finger` *name*

Angaben über Benutzer *name*, wobei *name* entweder der Vorname oder der Nachname oder die Benutzerkennung sein kann. Auf dem Rechnernetz der Informatik kann die Benutzerkennung als EMail-Adresse verwendet werden.

`elm (electronic mail)`

`elm`

Aufruf des Programms zur Verwaltung der elektronischen Post. Vgl. S. 19.

Inhaltsverzeichnis