Rechnerorganisation

Martin Steffen

Wintersemester 2000/01

Abschnitt I

Funktionsweise auf Aufbau von Rechnern

Inhalt: Geschichtliches · von Neumann Architektur · CPU · Speicher und Speichergrößen

Geschichte des Computers

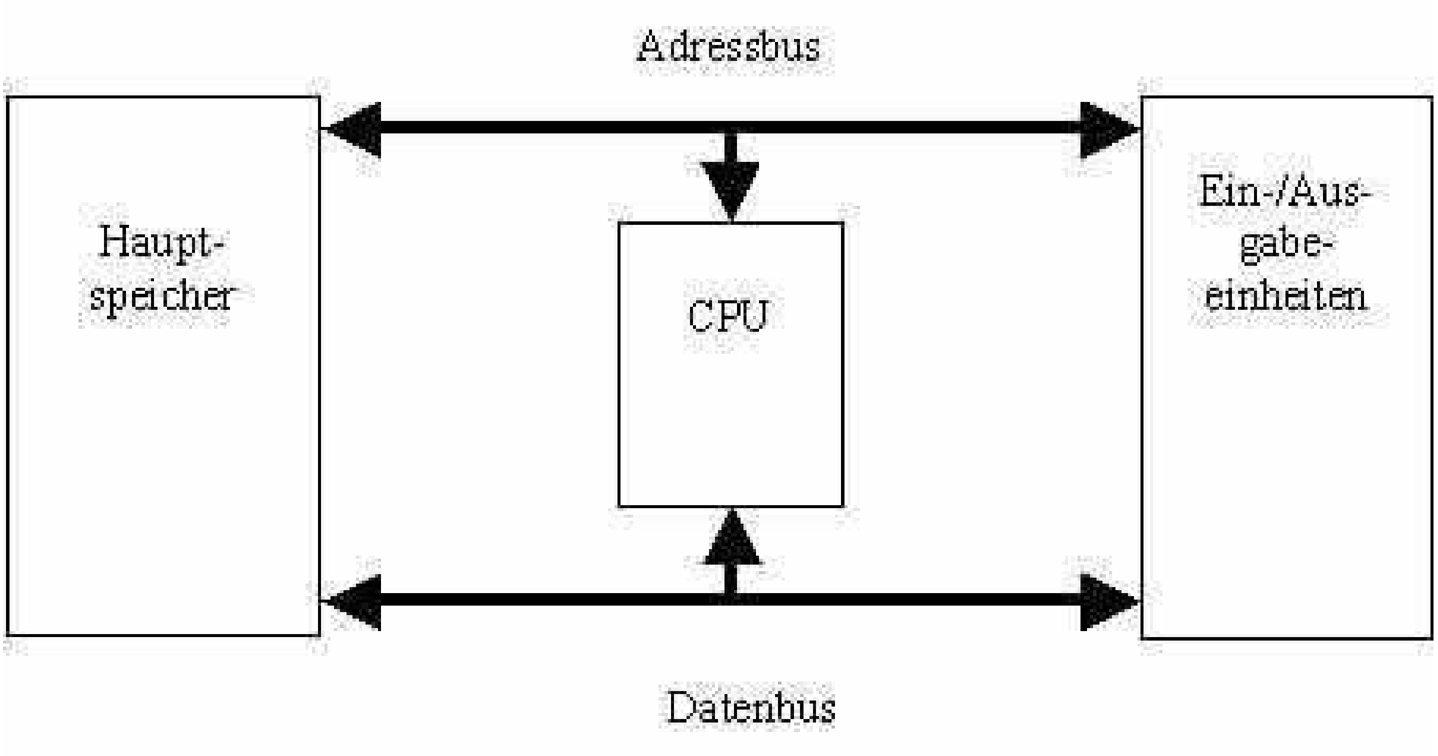
- lange Vorgeschichte von Rechenhilfen, aber keine "Computer" im modernen Sinne
- Charles Babbage (1791-1871): *Differential Engine*, *Analytical Engine* (nicht realisiert). Ada Lovelace: Gedankenspielereien zu "Software"
- Konrad Zuse: 1936 Z1: mechanisches, *programm*gesteuertes Rechnengerät
- Howard H. Aiken: Mark I (1944): elekromechanischer Rechner
- 1946: ENIAC: Erster "elektronischer Rechner (Röhren)
- John von Neumann: 1946: Prinzip der frei-programmierbaren Rechners
- Alan Turing: Theoretische Grundlagen (Turingmaschine), Rechenmachinen zum Entschlüsseln
- ...

Prinzipien der Von Neumann-Architektur

Prinzipien des Rechners wurden bereits früh formuliert: John von Neumann:

Ein Computer besteht aus Steuerwerk, Rechenwerk, Speicher, und den Ein-/Ausgabegeräten	Steuerwerk = CPU. Rechenwerk = ALU, Croprozessoren, verschiedene Speicherebenen.
Die Struktur des Computers muß unabhängig vom zu bearbeiteten Problem sein	
Im Speicher befinden sich Daten und das Programm	Heute meist mehrere Programme gleichzeitig
Der Speicher ist in Zellen gleicher Größe unterteilt und fortlaufend nummeriert	gilt im wesentl. unverändert (virtuelle Adressierung)
Das Programm besteht aus einer Folge von Befehlen, die nacheinander aufgeführt sind. Die Ausführung eines Sprunges kann von gespeicherten Werten abhängig gemacht werden	
Die Maschine benutzt Binärcodes.	

Schematische Architektur



CPU

- CPU
  - Central processing unit: “Prozessor”= da wo die Arbeit gemacht wird
  - Integrierte Schaltungen = “Chip”
  - Taktrate = Anzahl der elementaren Operationen pro Sekunde (gemessen in Hz, heutzutage eher MHz und bereits GHz)
- Bekannte Hersteller: Intel, AMD
- Beachte: Taktrate bestimmt nicht alleine die Geschwindigkeit des Rechner
- Ko-Prozessoren für Spezialaufgaben.

Speicherhierarchie (abstrakt)

- Speicher = Hardware zum Aufbewahren von Daten
- Es gibt nicht einen monolithischen Speicher, sondern eine Hierarchie, unterschieden nach
  - Speicherkapazität
  - Geschwindigkeit
  - Preis
  - „Nähe“ zur CPU
  - flüchtig oder dauerhaft
  - Zugriffsarten
    - nur lesbar (ROM), les- und schreibbar
    - Wahlfreier Zugriff (RAM), sequentieller Zugriff (Magnetband)
  - physikalische Speicherprinzipien

Speicherhierarchie

- Speicherstufen, aufsteigend nach Größe:
  - Register: sehr klein/schnell, flüchtig
  - Cache:[1] flüchtig
  - Hauptspeicher: flüchtig, Speicher in dem der Prozessor die für das laufende Programm benötigten Daten hält (oder zumindest zu halten versucht . . .)
  - Festplattenspeicher: Permanent, enthält das Dateisystem (siehe Abschitt III)
  - Hintergrundspeicher (zum Archivieren, d.h., permanent): Bänder, Zip, . . .
- Sonderspeicher:
  - Videospeicher (flüchtig),
  - Floppy: am Aussterben
  - CD-Rom, beschreibbare CD, . . .

[1] manchmal mehrstufig

Bits und Bytes

- Datenmenge werden in Bits[2] gemessen, beziehungsweise in 8er Gruppen von bits = Byte

$$
\begin{array}{rcl}
\text{1 bit} & = & \text{“}\textit{binary digit}\text{”} = 0 \text{ oder } 1 \\
\text{Byte} & = & \text{8 Bit} \\
\text{1 KByte} & = & \text{1024 Bytes (ca. 1000 bytes)} \\
\text{1 Megabyte (MB)} & = & \text{1 048 576 Bytes (ca. 1 Mio bytes)} \\
\text{1 Gigabyte (GB)} & = & \text{1 073 741 824 Bytes} \\
 & \vdots &
\end{array}
$$

[2] Das Binärsystem, also das 2er System, ist schon recht als, es wurde 1679 Leibnitz von Leibniz erfunden.

Speicherkapazitäten

Medium	Kapazität (ca.)
1 Schreibmaschinenseite	2 KB
Diskette (“Floppy”)	1 MB
Hauptspeicher	100MB
Festplatte	1 GB
Magnetband	2 GB und mehr
Audio-CD/CD-Rom	ca. 1GByte
DVD	20 Gbyte
Menschl. Gehirn	10^{15} Bytes (geschätzt)

Hard/Software im System

- Hardware: alles was man anfassen kann, der Rest = die Programme[3]
- grob: folgende Schichten im einem Rechner

Anwenderprogramme	Fenstersystem	...
Compiler	Editoren	Kommandointerpreter
Betriebssystem		
Maschinensprache		
Mikroprogrammierebene		
Physikalische Hardware		

[3]Die Grenzen sind nicht ganz so eindeutig

Abschnitt II

Betriebssystem

Inhalt: Aufgaben · Geschichte · Begriffsklärung

Aufgaben des Betriebssystems

- Resourcenverwaltung
- Resource =
  - Zeit (Stichwort Multitasking, Multiuserbetrieb)
  - Speicherverbrauch (Festplatten, Hauptspeicher)
  - Zuteilung und Verwaltung von I/O-Resourcen (Drucker, Tastatur, Bildschirm, Maus, (Daten-)Netzanschlüsse . . .)
- Verbergen der Hardware (Abstraktion) inbesondere Dateisystem
- Schutz
- Plattform/einheitliche Schnittstelle für Anwenderprogramme

Geschichte der Betriebssysteme

- bis ca. 1955: keinerlei Betriebssysteme
- 1955-65: Batch-Systeme (HW: Transistoren, Lochkarten/Magnetbändern als E/A)
- 65-80: Multiprogramming und Time-sharing-Betrieb (Interaktiv) auf Großrechnern (IBM360), Ursprung von Un*x: ca. 1970[4]
- Nach 1980:
  - Aufstieg des PC: DOS, Windows95 und seine Nachfolger[5]
  - Verbreitung von lokalen Netzen (verteilte Betriebssysteme), Internet
- Zukunft: eingebettete Systeme? Netcomputer? ...

[4] Man schreibt oft Un*x anstelle Unix, denn Unix ist ein geschütztes Markenzeichen von Bell Corp. Mit Un*x meint man alle Unixe — oder Unices, wie der Lateiner sagt — z.B. Berkely, BSD, Solaris, System V, HPUX, und auch die verschiedenen Linux-Abkömmlinge.

[5] Natürlich gibt's auch noch MAC-OS, und OS-2 &c

Multitasking/Multiuser/Timesharing

- Heutzutage: Betriebssystem abeitet viele Dinge (scheinbar) gleichzeitig ab[6]

- Illusion der Gleichzeitigkeit durch schnelles/geschicktes Hinundherjonglieren der zu bearbeitenden Aufgaben (Multitasking)

- inbesondere langsam: die menschlichen Reaktionen (Tastatur, Maus) und überhaupt E/A: ⇒ interaktives Verhalten für mehrere Benutzer möglich

- Mehrbenuzterbetrieb: erhöhte Anforderungen an Sicherheit (z.B. Trennung der Speicherbereiche etc.)

[6] Nur scheinbar, wenn man nur *einen* Prozessor besitzt. Letzten Endes ist es (fast) nur eine von-Neumann Maschine, und von Neumann hat u.A. gesagt: ein Befehl nach dem anderen.

Abschnitt III

Dateisystem

Inhalt: Dateien und Dateiformate · Organisation · spezielle Verzeichnisse · Navigation durch die Struktur · Veränderung der Verzeichnisstruktur · Rechte und Privilegien

Dateien

- Datei (engl. *file*) = zusammenhängende Menge gespeicherter Daten
- mögliche Inhalte: Text, Programm, Programmcode, Musik, Bilder, ...
- Typ der Datei bestimmt Bearbeitungsart (z.B. Musikdateien wird man nicht anschauen, sondern abspielen/anhören)
- Typ: gekennzeichnet per Konvention durch bestimmte Dateiextensionen[7]
- z.B., `datei.txt` kennzeichnet eine Textdatei, weitere Beispiel siehe Zusammenstellung auf der nächsten Folie.
- Extensionen können kombiniert werden `datei.ps.gz`

[7] In Dateibrowsern in graphischen Oberflächen werden, gemäß den Extensionen, die verschiedenen Typen durch mehr oder minder selbsterklärende Bildchen/Icons symbolisiert.

Dateiformate

Extension	Kürzel	Erklärung	Viewer/Bearbeitungsprogramm
*.txt	ASCII	einfachstes Textformat	jeder Editor
*.html, *.htm	hypertext markup lang.	WWW-Dokumente	Browser/Editor
*.ps	Postscript	Dokumente, wird von Druckern verarbeitet	ghostview, gv
*.pdf	port. document format	Dokumente	acroread
*.tex	LaTeX (oder TeX)		
*.dvi	"device independant"	Ergebnis eines LaTeX-Textes	xdvi
*.jpg, *.gif		zwei bekannte Bildformate, es gibt noch viele mehr	xv
*.fig		Zeicheneditorformat	xfig
*.gz *.Z		komprimierte Datei (mit Hilfe von gzip oder compress	dekomprimieren: gunzip
*.tar	"tape archive"	Archiv	tar

Tabelle 1: Häufige Dateiformate und ihre Extensionen

Organisation der Dateien: der Verzeichnisbaum

- hierarchische Strukturierung als Baum von Verzeichnissen
- Verzeichnis:[8] enthält Menge von Dateien und (Unter)-Verzeichnissen
- Verzeichnis $\approx$ Spezialform einer Datei[9]
- Pfad:
  - "Weg" durch den Baum $\Rightarrow$Lokalisierung von Dateien/Verzeichnissen
  - Trenner der Verzeichnisse: durch Schrägstrich (*"slash"*):[10] /
- Wurzel des Baumes (*"root"*-Verzeichnis), gekennzeichnet durch:

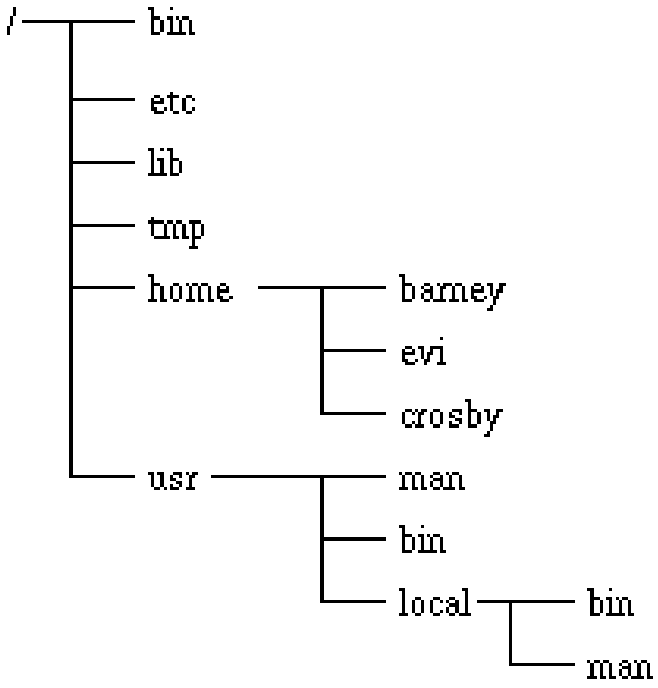
/

[8] Englisch: *directory*, beim Macintosh auch als Ordner bekannt

[9] der Vollständigkeit halber: Un*x kennt vier prinzipielle (aus Sicht des Betriebssystems unterschiedliche) Klassen von Dateien: normale Dateien, Verzeichnisse, Pipes und Devices. Wenn wir von „Datei" reden, werden wir im folgenden Datei im engeren Sinne meinen, keine Verzeichnisse etc.

[10] Bei DOS wird der umgekehrte Schrägstrich ("*backslash*") zum selben Zweck verwendet.

Typische Struktur eine Verzeichnisbaumes



Homeverzeichnis

- jede Benutzer: Portion des Plattenplatzes[11] = sein Teilbaum des Gesamtbaumes $\Rightarrow$

- Einstiegspunkt = Heimatverzeichnis (“*home*” oder “*home-directory*”)

- Beispiel: User `kurs001` $\Rightarrow$

  /home/kurs001/

  bezeichnet das entsprechende Home,

  `/home/kurs001/texte/` und `/home/kurs001/texte/seminararbeit.ps`

  ein Unterverzeichnis (vermutlich mit verschiedenen Texten) bzw. eine Datei, vermutlich die Postscriptdatei einer Seminararbeit

[11] Der einem von der Administration zugeteilte Anteil des Plattenplatzes heißt Quota, mit dem Befehl `quota` kann man sich über die Größe informieren.

Arbeitsverzeichnis

- aktuelles Verzeichnis (wo man "sich gerade gefindet"): Arbeitsverzeichnis (*working directory*)
- Default/nach dem Einloggen: = Homeverzeichnis
- Anzeigen des Arbeitsverzeichnisses mittels

```
pwd
```

Absolute und relative Pfadnamen

- 1. absolut = Pfad von der Wurzel aus
  2. relativ = Pfad vom Arbeitsverzeichnis aus

- Spezialverzeichnisse:

Abkürzung	Bedeutung
`.`	„Hier“, Arbeitsverzeichnis
`..`	„eins hoch“, übergeordnetes Verzeichnis
`~benutzer`	Homeverzeichnis von `benuzer` (i.d.R.= `/home/benutzer`)

- 1. absolut

  `/home/kurs001/texte/`

  2. relativ (ohne / zu Beginn). Angenommen, man befindet sich in `/home/kurs001/Mail`, dann bezeichnet

  `../texte/`

  das absolute Verzeichnis `/home/kurs001/texte`

Einfache Orientierung & Navigation

- Wo bin ich?: `pwd`[12] zeigt den absoluten Pfad des Arbeitsverzeichnisses
- Inhalt eines Verzeichnisses?: `ls <verzeichnis>`[13]
- Abkürzung: nur `ls`: listet das aktuelle Arbeitsverzeichnis
- Wechsel des Verzeichnisse: `cd`[14]

```
cd <verzeichnis>
```

  – Abkürzung: `cd` = `cd ~` = in's Homeverzeichnis wechseln

[12] *print working directory*

[13] `ls` = *list*. Es entricht dem DOS-Kommando `dir`. Der Befehl hat eine Reihe von Optionen, mit der man weitere Informationen (Zugriffsrechte, Dateigröße, Besitzer ...) bekommen kann. Man kann den Befehl auch auf ein einzelne Dateien, nicht nur Verzeichnisse anwenden.

[14] *change directory*. Genaue Verwendung bitte selbst nachschlagen.

Veränderung der Dateistruktur

- Anlegen von neuen Dateien: meist mittels allgemeiner/spezieller Editoren
- Auf dem Übungszettel: ein paar einfache Möglichkeiten zur Dateierzeugung
- Achtung! Nach dem Löschen sind die Daten weg

Befehl	Beispiel	Erklärung
`cp`		Kopieren
	`cp <datei1> <datei2>`	
	`cp <datei1> <verzeichnis>`	
`mv`		*"move"*
	`mv <datei1> <datei2>`	
	`mv <datei1> <verzeichnis>`	
`rm`	`rm <datei>`	*"remove"*: Datei löschen
`rmdir`	`rmdir <verzeichnis>`	*"remove directory"*: Verzeichnis löschen
	`mv <datei1> <verzeichnis>`	
`mkdir`	`mkdir <verzeichnis>`	*"make directory"*: neues Verzeichnis anlegen

Wildcards

- Hilfreiche Notation:
- auf Häufigsten gebraucht: $*$ = "Wildcard" = stellvertretend für beliebiges
- Zur Bezeichnung von Dateien
  - `ls a*`: Liste alle Dateien die mit a beginnen
  - `rm *.ps *.dvi` entferne alle postscript und alle dvi-Dateien

Un*x-Rechte

- drei Dinge, die man mit Dateien machen kann
  1. Lesen (r für *read*) Anschauen, kopieren
  2. Schreiben (w für *write*): Verändern, Überschreiben
  3. Ausführen (x für *execute*): Programme aufrufen/laufen lassen
- Verzeichnis = Sonderform einer Datei ⇒
  - Lesen: Inhalt mittel `ls` anzeigen lassen
  - Schreiben = Verzeichnisstrukur ändern d.h., neue Dateien/Unterverzeichnisse anlegen/löschen
  - Ausführen = in das Verzeichnis wechseln (mittels `cd`)[15]

[15] Beachte: Lesen eines Verzeichnisses und in das Verzeichnis wechseln sind verschiedene Dinge.

Benutzerklassen und Un*x-Gruppen

- drei hierarchiche Klassen von Benutzern

  1. Benutzer `u` *user*
  2. Gruppe `g` *group*
  3. Alle `o` *others*

- Jeder Benutzer gehört einer oder mehrerer Gruppen an, die seine Privilegien festlegen

- z.B. `user`, `student`, `mitarbeiter`, verschiedene administrative Gruppen

- Jede Datei/Verzeichnis hat Zugriffsprivilegien für Lesen/Schreiben/Ausführen getrennt für jede der drei Benutzer

- zwei wichtige Attribute einer Datei: Besitzer (*owner*) + Gruppe

Un*x-Gruppen (Forts.)

- Anschauen mittels `ls -lg` (g für „Gruppe“, `l` für „lang“, mit `ls -lag` bekommt man auch noch sog. unsichtbare Dateien angezeigt, a für „alle“)

```
[ms@sokrates] > ls -lag
total 1023
drwxr-xr-x  47 ms      mwpr      2560 Oct  8 11:40 .
dr-xr-xr-x   9 root    root         9 Oct  8 13:44 ..
drwxrwxr-x  11 ms      mwpr       512 Jun  2 07:29 Lehrstuhl
drwx------   5 ms      mwpr      4096 Oct  8 08:51 Mail
drwxr-xr-x   3 ms      mwpr       512 Aug 11 07:50 bin
drwxr-x---   8 ms      mwpr       512 Aug 17 08:59 papers
drwx------  14 ms      mwpr      1024 Oct  4 09:38 priv
```

- Ändern mittels `chgrp` und `chown`[16]
- jeder Benutzer ist für das Setzen der Rechte selbst verantwortlich
- für Fortgeschrittene: Automatisierung mittels `umask`-Kommando

[16] für *change group* und *change owner*

Abschnitt IV

Interaktion

Kommandointerpreter: die Shell

- wichtige System-Schnittstelle: Kommandointerpreter, Shell: interaktive Befehlseingabe
- In X-Menues: meist als Xterm benannt[17]
- es gibt eine Reihe von Un*x-Shells:[18] Bourne shell (der Urvater), C-Shell, tcshell, Korn shell. Bei uns *Bash* (“*bourne-again shell*”)
- Kommando: Textueller Befehl an das System, allgemeines Format

```
<commando> <optionen> <file>
```

- Optionen sind oft optional, verfügbare Optionen hängen vom Befehl ab (die *manpages* geben jeweils Auskunft)

[17] X-Terminal = ein Fensterrahmen um einen Kommandointerpreter herum

[18] Die Unterschiede sind für uns unerheblich. Der Benutzer merkt Unterschiede zwichen den Shells nur am Ausmaß der Bequemlichkeit. In diesem Sinne gibt es bequeme Shells, d.h. solche, bei denen man alte Befehle wiederholen kann und edieren etc., und unbequeme, bei denen man das nicht kann.

Umgebung

- Benutzerinteraktion unter bestimmten Voreinstellungen (Defaultdrucker? Lieblingseditor? Windowsystem? ...)

Umgebung (*environment*)

- Umgebung
  - von Systemadministration (hoffentlich) sinnvoll vorbelegt
  - von Benutzer anpassbar/erweiterbar
- Umgebungsvariable:
  - anzeigbar mittels `env`
  - verändern[19] mittels `export <VARIABLE>=<WERT>`
- wichtige Umgebungsvariable: `PATH`

[19] Umgebungsvariablen sind unabhängig von der Wahl der Shell, aber die Syntax `export .. = ..` ist bash-spezifisch.

Der Unix-PATH

- Annahme: ausführbare Datei ("Programm"):

  `/usr/local/games/bin/tetris`

⇒ Eingabe von `/usr/local/games/bin/tetris` führt es aus

⇒ Auf Dauer unpraktisch ⇒wichtige Umgebungsvariable `PATH`

- Aufruf eines Programmes in einem der dort erwähnten Verzeichnisse[20] Verzeichnisse: der Pfad kann weggelassen werden ⇒Aufruf nur mit tetris

- welches Programm wird aufgerufen: das, welches mit passendem Namen als erstes im `PATH` gefunden wird (Ausprobieren `which <programm>`)

[20] der Wert von `PATH` ist eine Liste von Verzeichnissen, sie sind durch ein : jeweils getrennt.

Sonstiges zur Shell-Anpassung (speziell Bash)

- bei gutverwalteten Systemen: vernünftige Voreinstellungen
- vom Benutzer änderbar
- dauerhafte Änderungen z.B. in `~benutzer/.bashrc`[21]
- Nützliche, einfache Sache: sog. alias
- Beispiel: `alias zeigemiralles="ls -lag"`
- Falls man das dauerhaft will: einfügen der Zeile ins `~benutzer/.bashrc`

[21] Viele Tools haben bestimmte Steuerdateien, oft Verborgene Dateien, das sind die, die mit einem Punkt beginnen und die man mit einem einfachen `ls` nicht zu Gesicht bekommt. Oft (durchaus nicht immer) enden sie mit dem Kürzel `rc`, was für *resource* steht. Wen's interessiert kann sich das `.bashrc` anschauen.

Abschnitt V

Sonstiges

Inhalt: Hilfe im System · Disketten · Drucken

Das wichtigste überhaupt: Hilfe im System

- primäre Hilfe zu einzelnen Befehlen: Manual-Pages:

  `man <command>` oder `xman`

  die *manpages* zeigen *kurz* die Verwendung des jeweiligen Befehl (Liste der Optionen) und oft ein paar Standardbeispiele

- Info-System im emacs oder xemacs[22]
  - Info-Information i.d.R. ausführlicher als *manpages*,
  - Manche (größere) Pakete verwenden Info-pages als primäre Nachschlagequelle

- Internet
  - bei lokale Problemen: FAQs des Instituts

[22] Emacs/Xemacs steht erst übermorgen auf dem Kursprogramm. Die Info-Seiten bekommt man auch mit `xinfo`.

Diskettenbenutzung

- bequeme Sammlung von Programmen zur Diskettenbenuztung: mtools
- verwendet DOS-Dateiformat ⇒brauchbar zum Datentransport zwischen[23]DOS↔ Un*x

`mformat a:`	Floppy formatieren (d.h., inbesondere auch löschen!!)
`mcopy datei.txt a:`	auf Floppy kopieren
`mcopy a:* .`	alles von Floppy ins System kopieren
`mdel a:datei.txt`	Datei auf Floppy löschen
`mdir a:`	Verzeichnis (directory) anzeigen
`eject`	Auswurf der Disk

[23]Es gibt (neben unterschiedlichen Filesystemen) einen Unterschied zwischen Textdateien in beiden Systemen: die Darstellung der "*Neuen Zeile*". Unter Un*x erkennt man eine DOS-generierte Textdatei daran, daß lauter `^M` durch die Datei gesprenkelt sind. `dos2unix` und `unix2dos` sind zwei Konvertierungskommandos, die man in dem Zusammenhang brauchen kann, falls einem der Editor die Angelegenheit nicht ohnehin ausbügeln hilft. Alternativ ist die Option `-t` von `mcopy` hilfreich.

Drucken

- Arten von Drucker: Tintenstrahl/Laser
- Leider: im Kurs drucken nicht möglich, keine Übungen dazu ...
- Voreingestellt pro Rechner (i.d.R): Standarddrucker (=der nächste erreichbare)
- Druckkommando: `lpr <datei>`[24]
- Druckbar: Text (="Ascii"), Postscript
- viele Tools: print-Button
- Vorsicht bei Folien: falsche Kombination von Folien/Drucker kann den Drucker beschädigen

[24] `lp` ist eine Alternative

Literatur

[AL92] Paul W. Abrahams and Bruce R. Larson. *Unix for the Impatient*. Addison-Wesley, 1992.

[et.92] Daniel Gilly et. al. *Unix in a Nutshell (System V Edition)*. O'Reilly & Associates, Inc., 1992.

[Hek97] Jessica Perry Hekman. *Linux in a Nutshell (A Desktop Quick Reference)*. O'Reilly & Associates, Inc., 1997.

[Reg99] Regionales Rechenzentrum für Niedersachsen, Universität Hannover. *Unix: Eine Einführung in die Benutzung*, 13th edition, November 1999.

[TSP96] Grace Todino, John Strang, and Jerry Peep. *Unix. Ein praktischer Einstieg*. Nutshell Handbooks. O'Reilly & Associates, Inc., 1996.