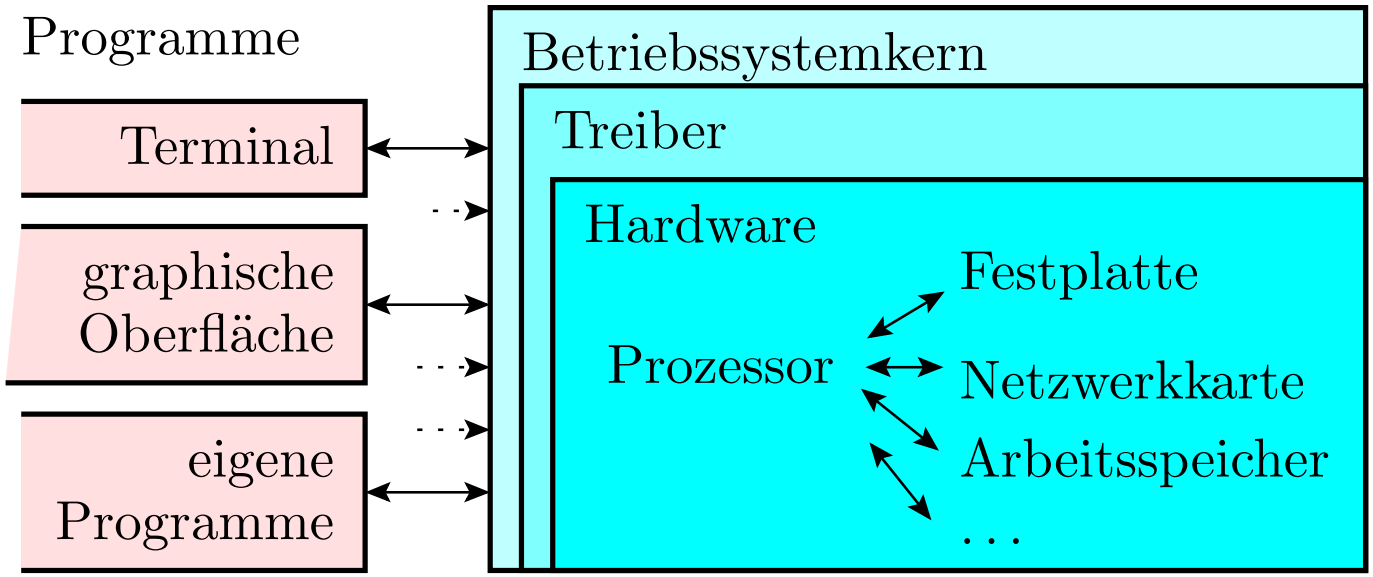
Informatik für Schüler, Foliensatz 1
Einführung bis ...

Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

19. August 2009

Der Rechner aus Programmsicht



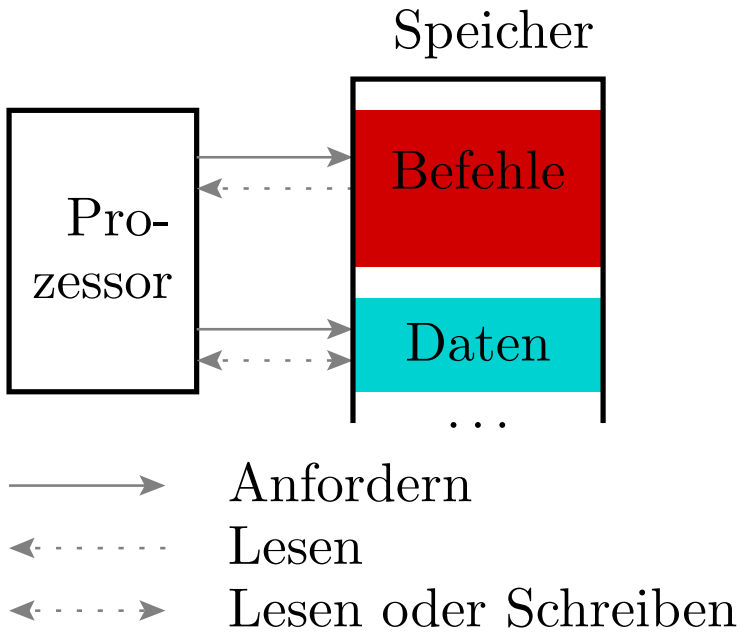
Hardware: so kompliziert und vielfältig, dass kaum ein Mensch den Durchblick hat

Treiber: Software, die den Zugriff auf unterschiedliche Hardware für das Betriebssystem vereinheitlicht

Betriebssystem (-Kern): spiegelt den anderen Programmen einen stark vereinfachten Rechner vor

Ein Programm sieht nur:

- einen Speicher, in dem die eigenen Befehle und die eigenen Daten stehen
- einen Prozessor, der die Befehle nacheinander abarbeitet, dabei Daten
  - liest
  - verarbeitet und
  - zurückspeichert



- alle anderen Aufgaben (Programme starten, Plattenzugriffe, Ein- und Ausgaben etc.) delegieren Programme an das Betriebssystem.

Was sieht der Programmierer

Dateien: Sortierkisten für Daten; Programme sind auch Daten
Prozesse: gestartete Programme, die etwas tun

Wie arbeitet man mit Dateien und Prozessen?

- Mit Hilfsprogrammen. Wir arbeiten unter Linux. Die folgenden der benötigten Programme werden aus dem Menü »Anwendungen« gestartet:
- Firefox (Untermenü Internet): Webbrowser zum Suchen und Lesen von Dokumenten, z.B. den Aufgabenblättern
- gedit (Untermenü Zubehör): Texteditor zum Schreiben der Programme
- Terminal (Untermenü Zubehör): textbasierter Kommandointerpreter, im Weiteren unser Hauptwerkzeug.

Programmiersprachen

- Programme werden in einer Programmiersprache geschrieben
- Jede Programmiersprache hat eine Syntax. Die Syntax beschreibt die formalen Regeln, etwa alles, was in Deutsch, Englisch etc. unter Rechtschreibung, Grammatik und Form fällt.
- Künstliche Sprachen sind viel einfacher als natürliche Sprachen; nur 10 bis 100 Vokabeln; kaum Ausnahmen; Schreibfehler erkennt meist der Übersetzer.

Metasprache

- Die Metasprache dient als Unterrichtshilfe und ist eine Sprache zur Beschreibung einer Sprache.
- Unsere Metasprache kommt mit folgenden Elementen aus:

`key`	Schlüsselwort, feststehender Name
`[...]`	optionaler Text, darf Null mal oder einmal enthalten sein
`{...}`	optionaler Text, darf beliebig oft enthalten sein
kursiv	Nicht-Terminalsymbol; Platzhalter für nach bestimmten Regeln zu bildende Texte
normal	Bezeichner; selbst gewählte Namen; Folgen aus Buchstaben und Ziffern; jeder Bezeichner darf nur einmal definiert sein

Terminalsprache (interaktive Übung)

- Wir werden den Rechner meist über ein Terminal bedienen.
- Jeder Befehl startet ein Programm; Befehlsaufbau:

  Befehl [*Optionen*] [*Argument1* [*Argument2*] ...] [&]

  Jede Befehlszeile mit »Enter« abschließen
  (Betätigen der Eingabetaste)

- Befehle mit abschließendem »&« werden als neue Prozesse gestartet, so dass das Terminal für weitere Eingaben bereitsteht
- Aufruf der Befehlsbeschreibung (Hilfe):

  man [*Befehl*]
  (man: Manual; mit Strg-q beenden)

- Start des Python-Interpreters

  `python [Programmdatei]`
  `(mit Strg-D beenden)`

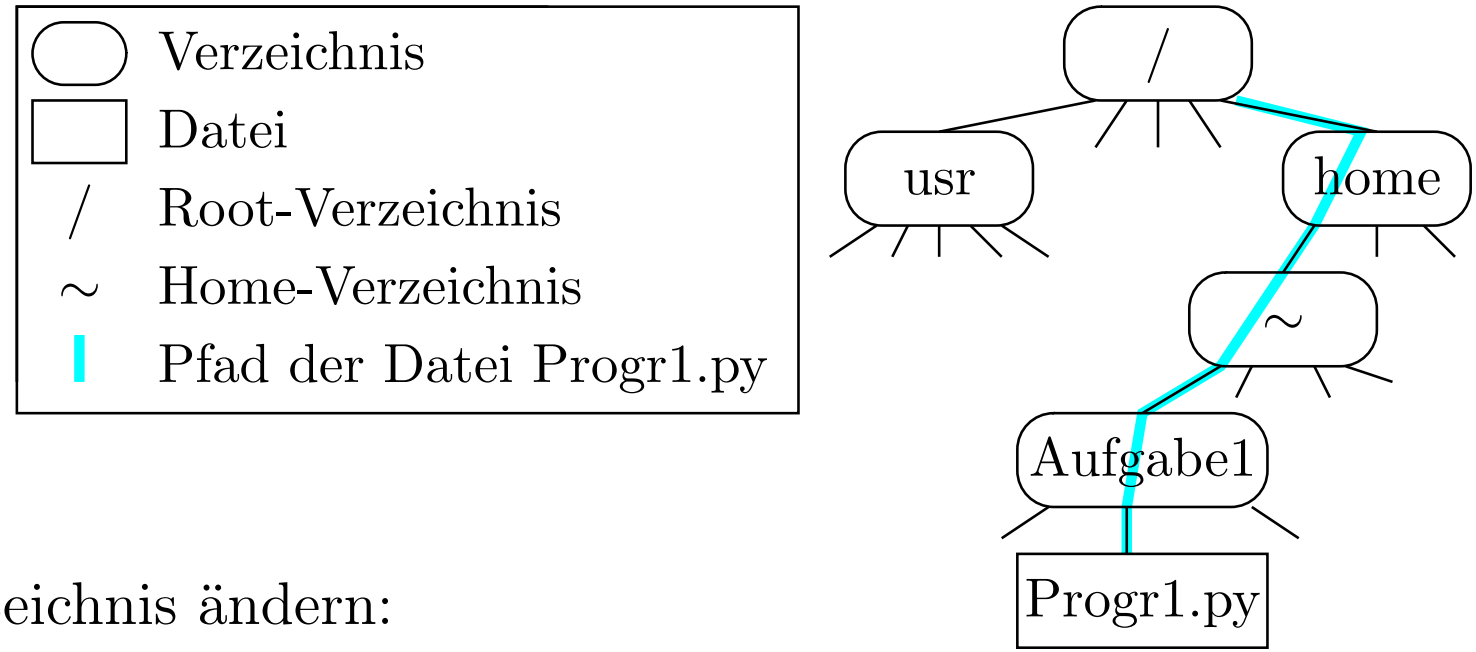
- weiteres Terminal öffnen

  `gnome-terminal &`

- Texteditor starten

  `gedit [Dateiname] &`

Verzeichnisse und Dateien (interaktive Übung)



- Verzeichnis ändern:

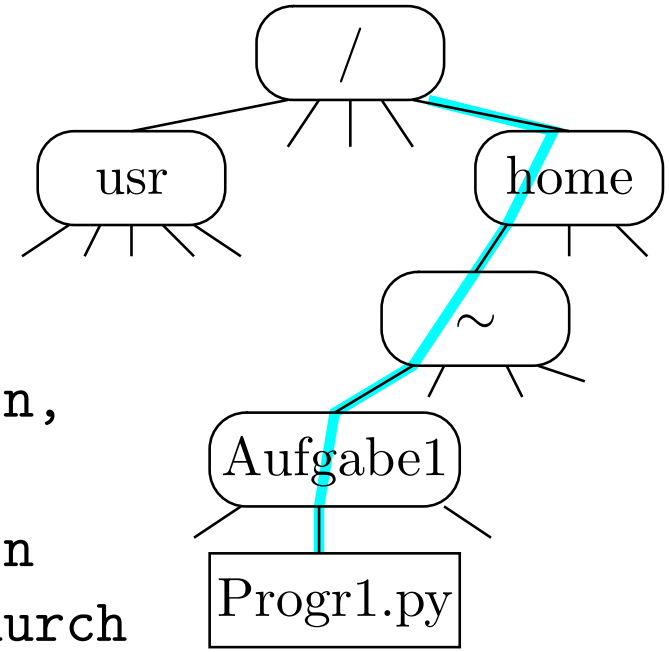
  `cd` *`Pfad`*

  (cd – change directory; »~« oder keine Verzeichnisangabe: wechseln ins Home-Verzeichnis; »..« ein Verzeichnis tiefer; »/« Root-Verzeichnis)

- Verzeichnisinhalt anzeigen (list)

  `ls [-l][-a] [Pfad][Datei]`

  - `-l: zusätzliche Anzeige der Attribute Besitzer, Gruppe, ...);`
  - `-a: auch versteckte Dateien, die mit ».« beginnen;`
  - `im Pfad und/oder Dateinamen können Teilzeichenketten durch das Zeichen »*« ersetzt werden.`



- ls A* Auflisten aller Dateien aus dem aktuellen Verzeichnis, die mit »A« beginnen
- ls *.py Auflisten aller Dateien aus dem aktuellen Verzeichnis, die mit ».py« enden.

- neues Verzeichnis erzeugen (make directory)

  `mkdir` *`Verzeichnisname`*

- Verzeichnis löschen (remove directory)

  `rmdir` *`Verzeichnisname`*

- Datei kopieren

  `cp [-r] [`*`Pfad`*`][`*`Quelldatei`*`] [`*`Pfad`*`][`*`Zieldatei`*`]`

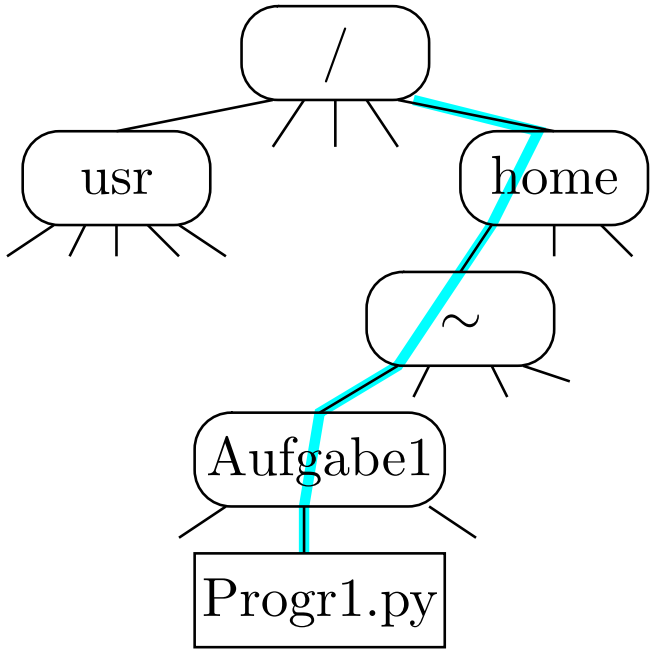
- Datei löschen (remove)

  `rm [-r] [`*`Pfad`*`][`*`Datei`*`]`

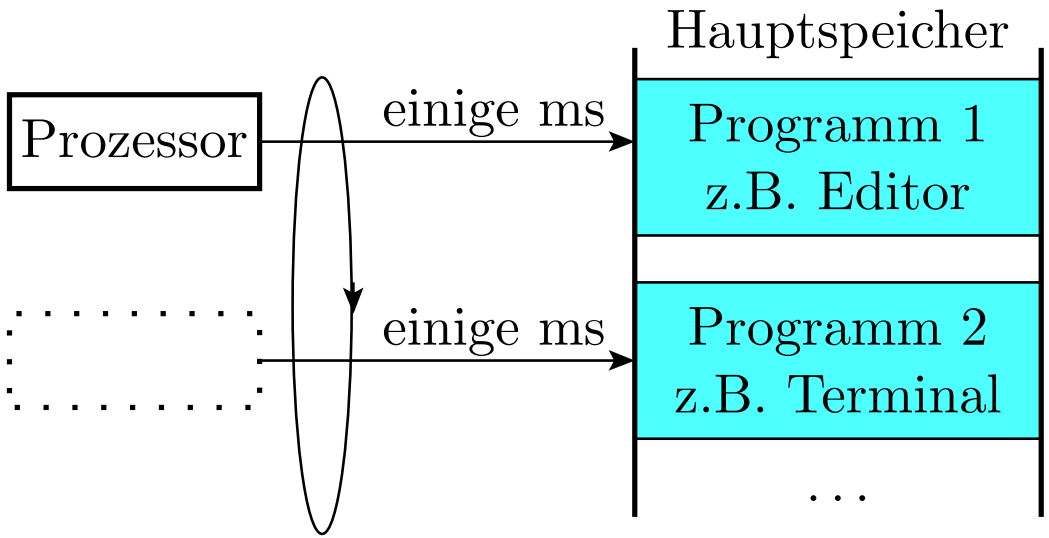
  - `-r` rekursiv, auch die Dateien in den Unterverzeichnissen
  - genauere Informationen mit dem `man`-Kommando erfragen

Experiment

- Welche Unterverzeichnisse gibt es im Verzeichnis /usr?
- Erzeugen Sie im eigenen Homeverzeichnis ein Unterverzeichnis »Aufgabe1«.
- Erzeugen Sie im Unterverzeichnis »Aufgabe1« mit dem Editor eine Textdatei mit dem Namen Progr1.py.



Prozesse



- Prozesse sind Programme, die in den Speicher geladen und zur Abarbeitung bereit sind
- Der Prozessor arbeitet reihum an jedem Programm einige Millisekunden weiter

- Anschauen der aktiven Prozesse

  ```
  ps [-A] Prozessnummer
  ```

  - mit »-A« alle Prozesse der Session
  - ohne »-A« nur die vom Terminal gestarten (Kind-) Prozesse)
- Starten eines neuen Prozesses

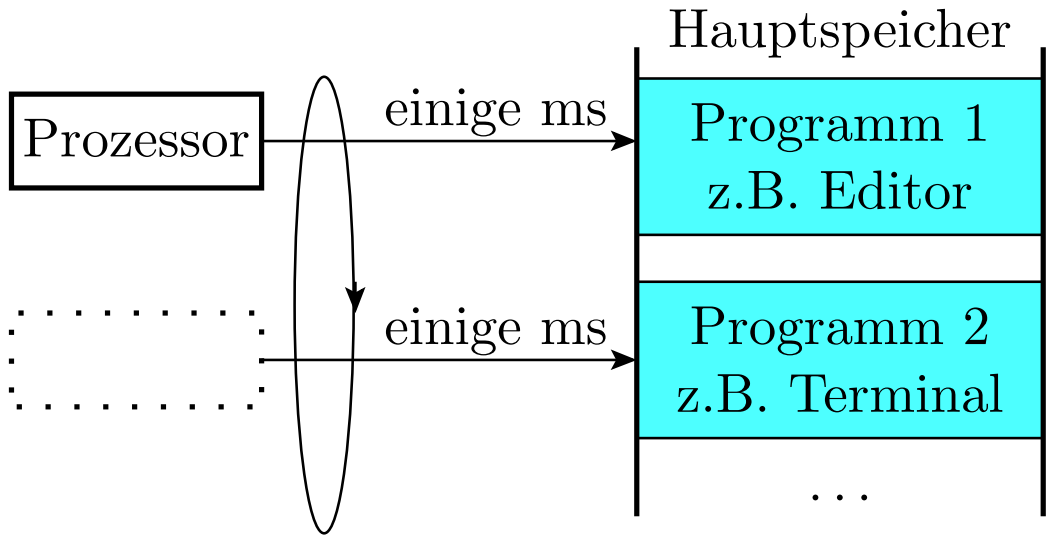
  ```
  Befehl [Optionen] [Argument1 [Argument2] ...] &
  ```

  - ohne »&« übernimmt das gestartete Programm die Prozessnummer des Terminalprozesses; das Terminal ist bis zu seiner Beendigung nicht mehr benutzbar)
- Abschießen eines Prozesses

  ```
  kill [-9] Prozessnummer
  ```

  - Prozessnummer mit ps ermitteln; »-9« Löschen ohne Rücksicht auf Verluste

Experiment



- Starten Sie aus dem Terminal den Editor (gedit) und mehrere Terminals (xterm oder gnome-terminal)
- ermitteln Sie mit ps die Prozessnummern
- Schießen Sie die gestarteten Prozesses nacheinander wieder ab.