Grundlagen der Programmierung in C

Einführung in Unix/Linux

Wintersemester 2005/2006
G. Zachmann
Clausthal University, Germany
zach@in.tu-clausthal.de

Was ist UNIX?

- Ein Betriebssystem
- Eine Sammlung von nützlichen Tools
- Eine (Computer-)Kultur

Wer braucht UNIX?

"Unix ist zwar ein Mainframe-Betriebssystem (und damit obsolet) hat aber noch viele Anhänger."
Windows MSCE-Training-Guide Windows 2000 Server
Kapitel 2.6.3 "Zusammenspiel mit UNIX", Verlag Markt & Technik

- Programmierer
- Web-Server
- Distributed Computing
- Wer braucht UNIX *nicht* (unbedingt) ?
  - Sekretärinnen
  - Büro- und Business-Software (Word, Buchhaltung, Powerpoint, Lagerhaltung, …)

Vorteile von UNIX

- Extrem ausgereift (besonders die kommerziellen Unices)
- Gut durchdachtes Konzept von Anfang an
  - "Alles ist ein File"
  - "Alles ist ein Prozeß"
- Von Anfang an Multi-User- und Multi-Task-fähig
- Relativ sicher
- Flexibler
- Performanter
- Wesentlich leichter zu administrieren (wenn die Lernkurve erst einmal durchschritten ist)
- Auf allen Plattformen verfügbar

Plattformen

- Sun (Solaris)
- HP (HP-UX)
- SGI (IRIX)
- IBM (AIX)
- Mac (OS-X)
- PC (Linux)
- PDA
- Set-top boxes
- Armbanduhr
- Auto
- …

http://www.linuxdevices.com/

http://www.research.ibm.com/WearableComputing/index.html

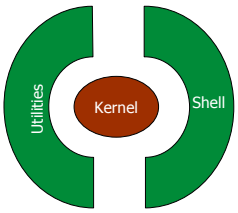
Die Erfinder

- Ca. 1970:
  - Haben UNIX und C erfunden!

Ken Thompson and Dennis Ritchie
Your new heroes

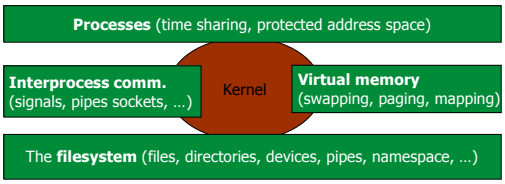
UNIX Komponenten

- Kernel: Herz des OS, managet Hardware
- Shell: eine Applikation, nimmt Kommandos entgegen und führt sie aus (CLI)
- Utilities: viele kleine (und große) Tools zur täglichen Arbeit, z.B. Files kopieren, ASCII-Texte editieren, …



UNIX-Konzepte

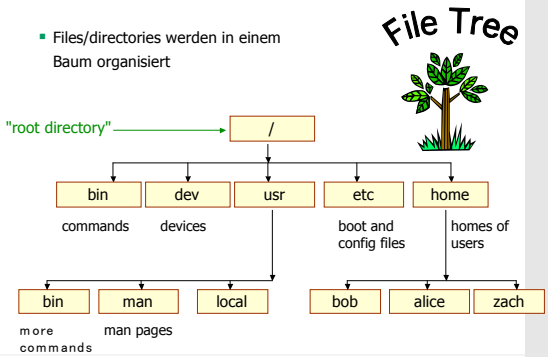
- Einige wenige Grundkonzepte:
  - Alles ist ein File (Programm, Daten, Speicher, …)
  - Alles ist ein Prozeß (OS, laufendes Programm, Editor, Shell, …)
  - Viele kleine Utilities, die kombiniert werden können
  - …



Das Filesystem

- Directories ("Folders") und Files
- File enthält sequentielle Folge von Zeichen (Bytes)
- Interpretation ist Sache des benutzenden Programms:
  - Text, Zahlen, Programm, Speicherauszug, …
- Jeder File hat einen Namen:
  - Case-sensitive! (UNIX allg.)
  - Länge typ. bis zu 1024
  - Können beliebige Zeichen enthalten – besser nur alphanumerische Zeichen und Underscore!
- Directory ("Verzeichnis"):
  - Enthält Name von File und Verweis darauf
  - Spezieller File

- Files/directories werden in einem Baum organisiert

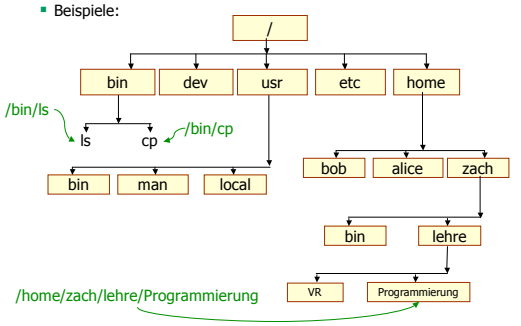


Eindeutigkeit

- Definition "Pfadname" (*pathname*) eines Files: Konkatenierung aller Verzeichnisnamen und des Filenamens auf dem Weg von der Wurzel bis zum File, getrennt durch /
- Eindeutigkeit:
  - Files im selben Verzeichnis müssen verschiedene Namen haben
  - Files in verschiedenen Directories dürfen gleiche Namen haben!

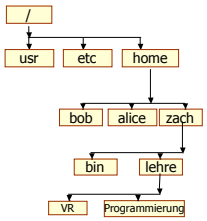
  →Eindeutigkeit von Pfadnamen garantiert

- Beispiele:



Absolute / relative Pfade

- Absolute Pfadnamen: starten mit /
- Relative Pfadnamen:
  - starten von einem anderen Dir aus
  - Sind also *relativ* zu diesem Dir
- Beispiele: der absolute Pfad `/home/zach/lehre/-Programmierung` von …
  - `home` aus = `zach/lehre/-Programmierung`
  - `zach` aus = `lehre/Programmierung`
  - `lehre` aus = `Programmierung`

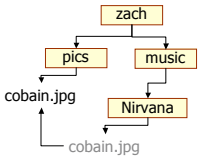


Spezielle Verzeichnisse

- `'.'` Bezeichnet das aktuelle Verzeichnis
  - Bsp.: `/bin/ls` = `/bin/./ls` = `/bin/./././ls` …
- '..' Bezeichnet das Vater-Verzeichnis (*parent diretory*)
  - Bsp.: `/usr/bin/w` = `/home/../usr/bin/w` = `/usr/man/../bin/w` …
- Wird besonders wichtig im Zusammenhang mit dem CWD (*current working directory*)

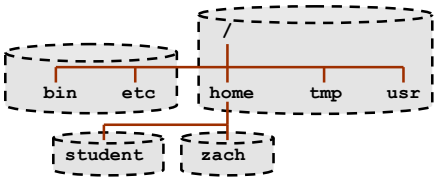
Symbolische Links

- Problem: File "gehört" genau einem Verzeichnis
  - Beispiel: File `/home/zach/pics/cobain.jpg` soll auch im Dir. `/home/zach/music/Nirvana` sichtbar sein …
- Lösung: *symbolic links* (*symlinks*)
  - Bsp.: `music/Nirvana/cobain.jpg` ist ein Symlink nach `../../pics/cobain.jpg`



Andere Platten

- Der Verzeichnisbaum enthält (i.A.) mehrere Platten!
- Einige davon sind auf anderen Rechnern (NFS)

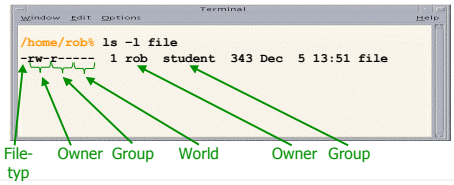


Home Sweet Home

- Jeder User hat ein *Home*
  - Z.B. `/home/zach`
  - Enthält normalerweise alle Daten des Users
  - Alle Konfigurationsfiles aller Programme ("Dot-Files", z.B. `.login`) (riesiger Vorteil gegenüber Registry!)
- Beim Einloggen "startet man im Home"
- Normalerweise auf einem Fileserver
- Ist auf jeder Maschine gleich zugreifbar
- Schreibweise: ~

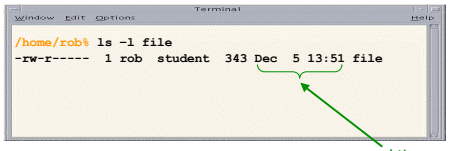
File Permissions

- 3 Personengruppen: Owner (=User), Group, World (Other)
- File gehört genau 1 User
- File ist assoziiert zu genau 1 Group
- Für jede 3 File-Permissions: read, write, execute



Weitere File-Attribute

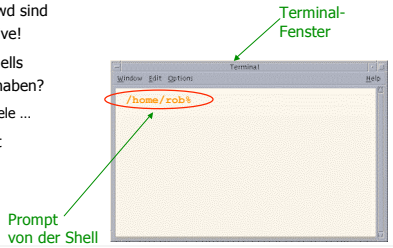
- Zeiten:
  - Modification (write): `ls -l`
  - Creation: `ls -lc`
  - Access (read): `ls -lu`



- Größe, Links, ...

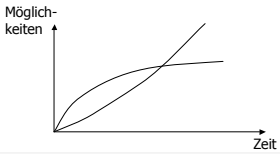
Erstes Einloggen

- Wie bekommt man eine Shell?
  - An der "Konsole" ("console")
  - Remote (ssh, rlogin, telnet)
- Login/passwd sind case-sensitive!
- Wieviele Shells kann man haben?
  - Beliebig viele ...
- Das Prompt

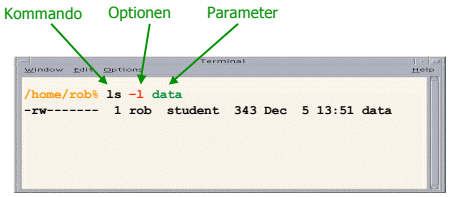


Das User-Interface

- Ist immer noch die Kommandozeile (CLI = command line interface)
- Gibt inzwischen zwar auch Windows-Look-Alikes
- Für Programmierer ist CLI sehr viel effizienter!
- Lernkurve ist natürlich länger ("steiler")



Aufbau einer Kommandozeile



- Optionen (options, flags): ändern Verhalten
- Parameter: i.a. Files, auf denen Kommando operiert

Editieren der Kommandozeile

- In der Zeile:

Taste	Funktion
Tab	File- / Command-Completion
Ctrl-B / Ctrl-F	Wortweise vor / zurück springen
Ctrl-W	Voriges Wort löschen
Ctrl-U / Ctrl-K	Zeile bis zum Anfang / Ende löschen
Ctrl-A / Ctrl-E	An Ende / Anfang springen

- In der History:

Taste	Funktion
Cursor-Up / -Down	In der History rauf / runter
Ctrl-P / Ctrl-N	Match in der History nach oben / unten suchen

Kommandowiederholung

Komando	Bedeutung
!!	Letztes Kommando wiederholen
!string	Kommando, das mit 'string' beginnt, wiederholen
!17	Kommando mit Nummer 17 i.d. History wiederholen
^a^b	Letztes Kommando wiederholen, dabei das erste Vorkommen von 'a' durch 'b' ersetzen

- History anzeigen: `history` (`alias h`)