

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



G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 3

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, ...)

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 4

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
- Performerter
- Wesentlich leichter zu administrieren (wenn die Lernkurve erst einmal durchschritten ist)
- Auf allen Plattformen verfügbar

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 5

Plattformen

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



G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 6

Die Erfinder

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



Ken Thompson and Dennis Ritchie
 Your new heroes

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 7

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, ...

The diagram shows a central brown circle labeled 'Kernel'. It is surrounded by two green semi-circular shapes. The left one is labeled 'Utilities' and the right one is labeled 'Shell'.

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 8

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
 - ...

The diagram shows a central brown circle labeled 'Kernel'. It is surrounded by four green rectangular boxes. The top one is 'Processes (time sharing, protected address space)'. The left one is 'Interprocess comm. (signals, pipes sockets, ...)'. The right one is 'Virtual memory (swapping, paging, mapping)'. The bottom one is 'The filesystem (files, directories, devices, pipes, namespace, ...)'.

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 9

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

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 10

File Tree

Files/directories werden in einem Baum organisiert

The diagram shows a tree structure starting from a root directory '/'. It branches into 'bin' (commands), 'dev' (devices), 'usr', 'etc' (boot and config files), and 'home' (homes of users). Under 'usr', there are 'bin' (man pages), 'man', and 'local'. Under 'home', there are 'bob', 'alice', and 'zach'. Under 'zach', there are 'bin' and 'lehre'. Under 'lehre', there are 'VR' and 'Programmierung'.

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 11

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

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 12

Beispiele:

The diagram shows the same file tree as in slide 11. It highlights specific paths: '/bin/ls' pointing to the 'ls' file in the 'bin' directory under 'usr'; '/bin/cp' pointing to the 'cp' file in the 'bin' directory under 'usr'; and '/home/zach/lehre/Programmierung' pointing to the 'Programmierung' file in the 'lehre' directory under 'zach'.

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unix/Linux, 13

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**

```

graph TD
  root[" / "] --> usr["usr"]
  root --> etc["etc"]
  root --> home["home"]
  home --> bob["bob"]
  home --> alice["alice"]
  home --> zach["zach"]
  zach --> bin["bin"]
  zach --> lehre["lehre"]
  bin --> VR["VR"]
  bin --> Programmierung["Programmierung"]
  
```

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 14

Spezielle Verzeichnisse

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

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 15

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`

```

graph TD
  zach["zach"] --> pics["pics"]
  zach --> music["music"]
  pics --> cobain_pics["cobain.jpg"]
  music --> Nirvana["Nirvana"]
  Nirvana --> cobain_music["cobain.jpg"]
  
```

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 16

Andere Platten

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

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 17

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: `~`

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 18

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

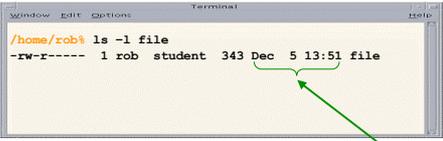
```

/home/rob% ls -l file
-rw-r--r-- 1 rob student 343 Dec 5 13:51 file
  
```

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 20

Weitere File-Attribute

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

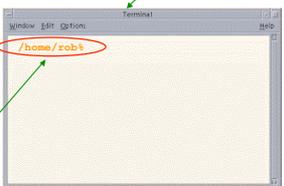


- Größe, Links, ...

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 22

Erstes Einloggen

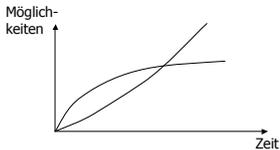
- Wie bekommt man eine Shell?
 - An der "Konsole" ("console")
 - Remote (ssh, rlogin, telnet)
- Login/passwd sind case-sensitive!
- Wieviele Shells kann man haben?
 - Beliebig viele ...
- Das Prompt



G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 25

Das User-Interface

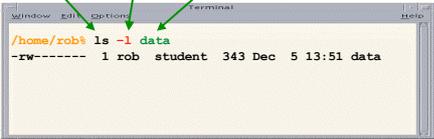
- Ist immer noch die Kommandozeile (CLI = command line interface)
- Gibt inzwischen zwar auch Windows-Look-Alikes
- Für Programmierer ist CLI sehr viel effizienter!
- Lernkurve ist natürlich länger ("steiler")



G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 26

Aufbau einer Kommandozeile

Kommando Optionen Parameter



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

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 27

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

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 28

Kommandowiederholung

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

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

G. Zachmann Grundlagen der Programmierung in C - WS 05/06 Einführung in Unity/Linux, 29