

# Netzwerke – Übung SoSe 2020

Grundlagen \*nix & Shell

Benjamin.Troester@HTW-Berlin.de

PGP: ADE1 3997 3D5D B25D 3F8F 0A51 A03A 3A24 978D D673

Benjamin Tröster

# Kurze Einführung zu Betriebssysteme

- Wir arbeiten das Semester über mit Debian – Linux-Derivat
- Warum? Guter Support der Tools, Großteil der Netzwerkinfrastruktur basiert auf Linux/UNIX
- Warum dann eine Einführung zu Betriebssysteme? Hintergrundwissen!
- Kurze Einführung zu den Themen
  - Betriebssysteme
  - Shell

# Road-Map

## 1 Historisches zu Unix

- Linux

## 2 Unixoide Betriebssysteme

- Aufgaben des OS'
- Architektur (monolithischer Kernel)
- Dateisysteme
- User & Gruppen

## ■ Ziffer- und Positionssysteme

- DAC

## 3 Systemcalls & Daemons

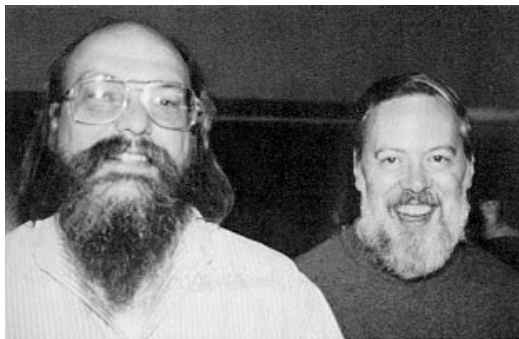
## 4 Unix-Philosophie

## 5 Shell

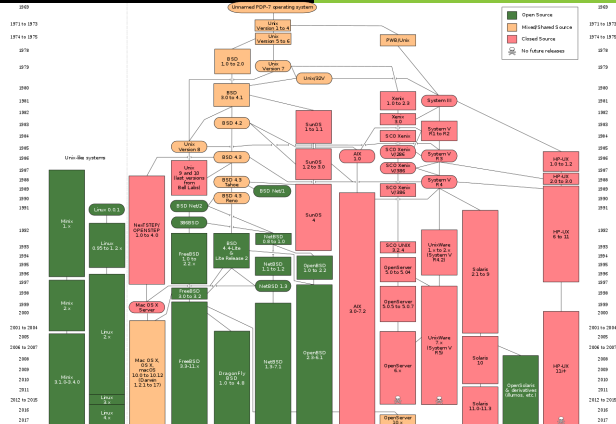
- Einführung
- Shell 101
- Shell Input/Output

# Historisches zu Unix

- Eigentlich von Uniplexed Information and Computing Service (UNICS) – Anspielung auf Multics <sup>1</sup>
- 1969 entwickelt in den Bell Laboratories
- Bekannte Vertreter:
  - Berkeley Software Distribution (BSD), SunOS/Solaris, Minix, OpenBSD, IllumOS, FreeBSD
  - [https://de.wikipedia.org/wiki/Berkeley\\_Software\\_Distribution](https://de.wikipedia.org/wiki/Berkeley_Software_Distribution)



<sup>1</sup><https://de.wikipedia.org/wiki/Multics>



# Linux

- 1991 im Usenet <sup>2</sup> veröffentlicht von Linus Torvalds
- Linux im wesentlichen Kernel (Betriebssystemkern)  
+ GNU-Tools (Compiler, Debugger etc.)
  - Kernel übernimmt die elementarsten Aufgaben im BS (s. [https://en.wikipedia.org/wiki/Kernel\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Kernel_(operating_system)))
- Distributionen nutzen (angepassten) Linux Kernel  
+ (eigene) Standardsoftware – Paketmanager etc.
- Bekannte Linux Distributionen:
  - Slackware, Red Hat, Debian, Gentoo, Arch
  - Mehr unter: <https://www.kernel.org/>

<sup>2</sup><https://de.wikipedia.org/wiki/Usenet>



## 1 Historisches zu Unix

## 2 Unixoide Betriebssysteme

- Aufgaben des OS'
- Architektur (monolithischer Kernel)
- Dateisysteme
- User & Gruppen
- Ziffer- und Positionssysteme
- DAC

## 3 Systemcalls & Daemons

## 4 Unix-Philosophie

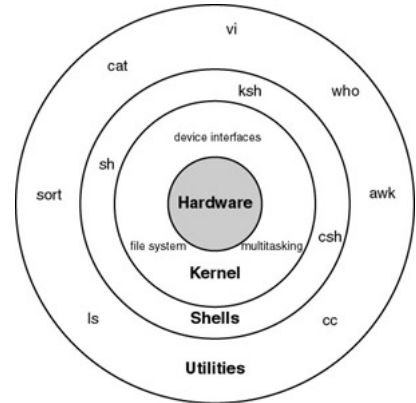
# Hauptaufgaben des Betriebssystems

- Bereitstellen einer virtuellen Maschine  
[https://de.wikipedia.org/wiki/Virtuelle\\_Maschine](https://de.wikipedia.org/wiki/Virtuelle_Maschine)
  - als Abstraktion des Rechnersystems
- Verwaltung und Operationen auf den Ressourcen
  - physische Ressourcen
  - logische Ressourcen
- Adaption der Rechnerstruktur für Nutzeranforderungen
- Legt die Grundlage für geregelten, nebenläufigen Ablauf der Aktivitäten
- Verwaltung der Daten & Ressourcen
- Unterstützung bei Fehlern & Ausfällen ...



# Aufbau eines Betriebssystems: Ringmodell

- Hardware
  - CPU, RAM, Mainboard ...
- Kernel – Betriebssystemkern
  - Gerätetreiber, Dateisystem, Prozesssteuerung, Systemaufrufe ...
- Shell – Schnittstelle zwischen Nutzer & Diensten des Betriebssystems
  - Command Line Interface (CLI) oder Graphical User Interface (GUI)
  - Interpretiert & bearbeitet Eingaben des Nutzers
- Anwendungsprogramme
  - Standardsoftware & 3rd-Party-Software

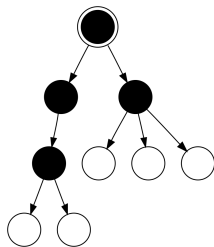


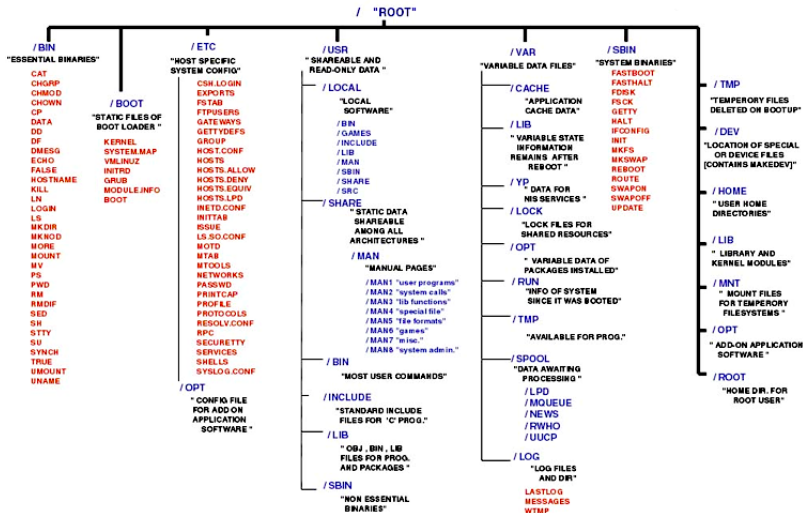
# Dateisysteme

- Dateisystem ist die Abstraktion der eigentlichen physischen Ressource (HDDs, SSDs,...)
- Dateien sind logische Ressourcen → Kollektion von logischen Dateneinheiten – Records
- Dateisysteme richten sich (wie Betriebssystem) nach den Anforderungen
- Beispiele:
  - FAT – File Allocation Table
  - NTFS – New Technology File System
  - UFS – Unix File System
  - ZFS – Zettabyte File System

# Bäume

- Mathematische Struktur – spezieller, zusammenhängender, azyklischer Graph (Graphentheorie)
- In Unix-Dateisystem: Es gibt ein Wurzelement: "/" – sprich: *root*
- Alle anderen Einträge (Ordner, Dateien etc.) sind dem Wurzelement untergeordnet





## Dateisystem cnt.

In Linux/Unix ist grundsätzlich alles eine Datei!

**Baumstruktur – statt separate Massenspeicher**

Exemplarisch:

- / – Wurzelverzeichnis
- /bin – wichtigste Programme in Binäreformat
- /boot – Boot-Loader
- /etc – System Konfiguration
  - /usr/local – lokale Software
  - /usr/bin – User-Land-Software
  - /usr/include – Standard-Bibliotheken für C/C++
  - /usr/lib – Bibliotheken für Programmiersprachen

■ ...

Linux/Unix sind Mehrbenutzersysteme, d.h. mehrere Nutzer können simultan auf einem Rechner arbeiten

- Zuordnung der Nutzer zu User & Group
- Regelt Zugangskontrolle im System auf
  - Dateien, Ordner & Peripheriegeräte
- Unterschiedliche Nutzer/Gruppen → unterschiedliche Rechte
- Im Labor:
  - Benutzername: Matrikelnummer
  - Gruppen: student, domain, users,...
- Virtual Machine (Debian):
  - Benutzername: student
  - Gruppen: student, users, wireshark,...

## Ziffer- und Positionssysteme

- Dezimalsystem – Basis 10
  - Werte 0 - 9
  - Beispiel:  $42_{10}$
- Dualsystem/ Binärsystem – Basis 2
  - Werte 0 oder 1
  - Bit-Darstellung in der Informatik/ Rechnertechnik
  - Beispiel:  $42_{10} = 0010\ 1010_2$
- Oktalsystem – Basis 8
  - Werte 0 - 7
  - Für Darstellung der Zahlen 0 - 7  $\rightarrow$  3 Bit notwendig,  $2^3 = 8$
  - Beispiel:  $42_{10} = 52_8 = 0010\ 1010_2$
- Sehr schöne Aufarbeitung, Kapitel 1.1.2ff:  
<http://numerik.mi.fu-berlin.de/mattheon-G8/comaBuch.pdf>

# Discretionary Access Control

Zugriff auf Dateien allein anhand der Identität.

Ressourcenzugriff haben Eigentümer & Gruppe – regeln Abbildung auf Nutzer

- Grundsätzlich in drei Kategorien:
  - Owner – regelt Berechtigung des Eigentümers
  - Group – regelt Berechtigung der Gruppe
  - Other (world) – regelt Berechtigung aller anderen Nutzer
- Unix Zugriffsmodi:
  - read (r) – Lesezugriff
  - write (w) – Schreibzugriff
  - execute (x) – Ausführzugriff



# Discretionary Access Control

Darstellung im System via Oktalzahlen:

- Zuordnung der Berechtigung r,w,x bestimmten Werten
  - Lesen (r)  $\rightarrow 4_8$  oder  $100_2$
  - Schreiben (w)  $\rightarrow 2_8$  oder  $010_2$
  - Ausführen (x)  $\rightarrow 1_8$  oder  $001_2$
  - None  $\rightarrow 0_8$  oder  $000_2$
- Zusammensetzen der Oktalwerte ergibt Zugriffsrechte:
  - Lesen, schreiben und ausführen  $\rightarrow 7_8$  oder  $111_2$
  - Lesen und Schreiben  $\rightarrow 6_8$  oder  $110_2$
  - Lesen und Ausführen  $\rightarrow 5_8$  oder  $101_2$
  - ...

# Discretionary Access Control

## Zusammensetzung der Berechtigung

- 3er-Oktett gibt Zugriffsmodalitäten an

- 1 user r,w,x – erstes Oktett
- 2 group r,w,x – zweites Oktett
- 3 other r,w,x – drittes Oktett

```
benjamin@node01 [13:55:38] [~]
-> % ls -l
total 2944
drwxr-xr-x 3 benjamin benjamin 4096 Apr 28 11:15 ~
-rw-r--r-- 1 benjamin benjamin 18 Apr 28 12:28 dump.rdb
-rw-r--r-- 1 benjamin benjamin 2958334 Apr 28 11:15 gnode.jar
-rw-r--r-- 1 benjamin benjamin 182 Apr 28 11:12 grischa.conf
drwxr-xr-x 2 benjamin benjamin 4096 May 1 23:11 log_redis
-rw-r--r-- 1 benjamin benjamin 32835 Apr 28 12:25 redis.conf
-rwxr-xr-x 1 benjamin benjamin 0 Sep 29 13:53 start.sh
```

# Systemcalls & Daemons – !Short Version

- Systemcalls – Methode für Anwendungsprogramme, um Funktionalitäten des BS' nutzen zu können
- Systemcalls vollführen Wechsel von Anwenderebene auf BS-Kern
- Übergabe der Kontrolle von Anwender an das Betriebssystem
  - Bspw.: anlegen von Dateien auf SSD, Verbindung des Browsers zu einer Webseite etc.
- Daemons – Hintergrunddienste
- Stellen Dienste des BS bereit, auf die Programme zugreifen können
  - Bsp: Netzwerkdienste, Sockets ...

- 1 Historisches zu Unix
- 2 Unixoide Betriebssysteme
- 3 Systemcalls & Daemons
- 4 Unix-Philosophie**
- 5 Shell

# Unix-Philosophie

Nach Douglas McIlroy <sup>3</sup> <sup>4</sup>

- Schreibe Computerprogramme so, dass sie nur eine Aufgabe erledigen und diese gut machen.
- Schreibe Programme so, dass sie zusammenarbeiten.
- Schreibe Programme so, dass sie Textströme verarbeiten, denn das ist eine universelle Schnittstelle.

**Bottom-Line:** Baue Programme derart das Interoperabilität besteht, sodass komplexere Probleme lösbar sind!

<sup>3</sup>[https://en.wikipedia.org/wiki/Douglas\\_McIlroy](https://en.wikipedia.org/wiki/Douglas_McIlroy)

<sup>4</sup>[https://homepage.cs.uri.edu/~thenry/resources/unix\\_art/ch01s06.html](https://homepage.cs.uri.edu/~thenry/resources/unix_art/ch01s06.html)

- 1 Historisches zu Unix
- 2 Unixoide Betriebssysteme
- 3 Systemcalls & Daemons
- 4 Unix-Philosophie
- 5 Shell
  - Einführung
  - Shell 101
  - Shell Input/Output

# Einführung Shell









- Textbasierte Schnittstelle
- Schnittstelle zwischen BS-Kern, Werkzeugen des BS' & User
- Shell ist ein Kommando-Interpreter → führt Schrittweise Befehle aus
  - Kommandos sind oft Binärdateien, Programmskripte
  - Kommandos können direkt aufgerufen werden
  - Aufruf von Systemcalls möglich → Administration des Systems

# Shells

- Ur-Shell: Thompson Shell – OSH
- SH – Bourne Shell
- BASH – Bourne Again Shell
- CSH – C Shell
- TCSH – TENEX C Shell
- KSH – Korn Shell
- ZSH – Zhong Shao Shell
- ...






# Shell 101

- In der VM:  (Windowstaste) und dann einfach Terminal eingeben
- Alternativ:  +  +  (Str + Alt + t)
- Abbrechen eines Kommandos:  +  (Str + c)
- Beenden/Schließen des Terminals:  +  (Str + d) oder einfach *exit* eintippen

# Shell Input/Output

- Kommandozeile hat drei Standardkanäle:
  - 1 Standardinput (stdin) – Eingabe von Daten
  - 2 Standardoutput (stdout) – Ausgabe von Daten
  - 3 Standarderror (stderr) – Ausgabekanal im Fall von Fehlern
- Ausgabe von Tools zumeist auf stdout
- Ein- & Ausgabe kann jedoch auch umgelenkt werden
- Ausgabe kann somit in Datei geschrieben bzw. aus Datei gelesen werden
- Verbinden von Kommandos durch *Piping*
  - Ausgabe eines Programms wird Eingabe des anderen Programms
- Schauen Sie sich die Links im Moodle-Kurs an!

# Input/Output Redirection

- Umlenken der Ausgabe in eine Datei: 
  - Lenkt Ausgabe in eine Datei
  - Datei wird dabei vollständig neu geschrieben
  - Alter Inhalt geht verloren
- Anfügen einer Datei: 
  - Hängt Ausgabe an das Ende der Datei
- Umlenken der Eingabe aus einer Datei: 
  - Programm erhält zeilenweise Eingabe aus der Ressource

# Piping

- Verbinden mehrerer Kommandos durch eine *Pipe*
- Pipe: Datenstruktur – Folgt dem First-In-First-Out-Prinzip (FIFO)
- Wirkt wie ein Puffer, eingehende Daten werden gepuffert und bei Bedarf wieder ausgegeben
- Piping ermöglicht es Programme zu verbinden
- Ausgabestrom eines Programms wird Eingabestrom des nächsten Programms
- **Vorteil:** Einfache Programme können zu mächtigeren Programmen zusammengesetzt werden
- Folgt der Unix-Philosophie

# Beispiele

```
1 date > foo.txt
2 echo "student name" >> studi_list_1.csv
3 head < studi_list_1.csv
4 sort studi_list_1.csv studi_list_2.csv | uniq -u > clean_list.csv
```

- BSD** Berkeley Software Distribution
- CLI** Command Line Interface
- DFN** Deutsches Forschungsnetz
- GUI** Graphical User Interface
- GW** Gateway
- LAN** Local Area Network
- MOCO** Mobile Computing
- SoC** System on a chip
- WAN** Wide Area Network
- UNICS** Uniplexed Information and Computing Service