

Netzwerke – Übung WiSe 2018/19

Grundlagen *nix & Shell

Benjamin.Troester@HTW-Berlin.de

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

Benjamin Tröster

Road-Map

1 Laborhardware

- Laborrechner & Infrastruktur
- Raspberry Pi

2 Unixoide Betriebssysteme

- Historisches zu Unix
- Linux
- Aufgaben des OS
- Architektur (monolithischer Kernel)

- Aufbau Filesystem
- User, Gruppen
- Nutzerrechte

3 Shell

- Einführung
- Unix-Philosophie

4 Systemcalls

5 Shell 101

Laborrechner & Infrastruktur

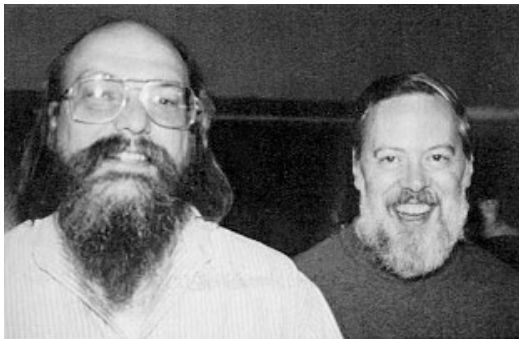
- WH C 625
- 21 Dell Optiplex
 - Intel Core i5-7500 – 4 Cores/ 4 Threads, 3.40 - 3.80 GHz
 - 16 GB RAM – 2 × 8 GB @2400MHz DDR4 Memory
 - 256 GB SATA-SSD
 - Ubuntu 18.04 / Windows 10
- GBit-Local Area Network (LAN)
- GBit-Wide Area Network (WAN) ins Deutsches Forschungsnetz (DFN) – IPv4 Only

Raspberry Pi

- Raspberry Pi 3 Model B – System on a chip (SoC)
<https://www.raspberrypi.org/>
- Architektur: ARM Cortex (64-Bit) Broadcom BCM2837
- Quad Core: 4 × ARM Cortex-A53 @ 1.2GHz
- 1GB LPDDR2 (900 MHz)
- 10/100 Ethernet, 2.4GHz 802.11n wireless
- 4 USB2 ports
- Raspbian 9 Stretch – Debian Fork
<https://raspbian.org/>

Historisches zu Unix

- Eigentlich von Uniplexed Information and Computing Service (UNICS) – Anspielung auf Multics ¹
- 1969 entwickelt in den Bell Laboratories
- Bekannte Vertreter:
 - Berkeley Software Distribution (BSD), SunOS/Solaris, Minix
 - https://de.wikipedia.org/wiki/Berkeley_Software_Distribution



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



Linux

- 1991 im Usenet ² veröffentlicht von Linus Torvalds
- Linux im wesentlichen Kernel (Betriebssystemkern)
+ GNU-Tools (Compiler, Debugger etc.)
- Distributionen nutzen (angepassten) Linux Kernel
+ (eigene) Standardsoftware – wie Paketmanager
etc.
- Bekannte Linux Distributionen:
 - Slackware, Red Hat, Debian, Gentoo, Arch Linux
 - <https://de.wikipedia.org/wiki/Linux>
 - <https://www.kernel.org/>



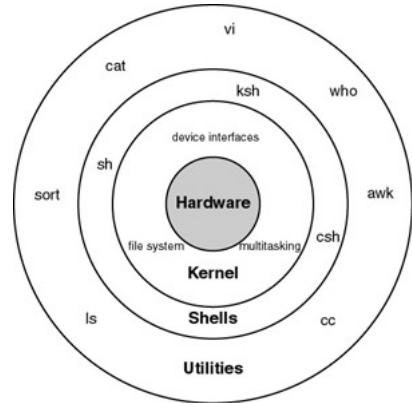
²<https://de.wikipedia.org/wiki/Usenet>

Hauptaufgaben des Betriebssystems

- Bereitstellen einer virtuellen Maschine
https://de.wikipedia.org/wiki/Virtuelle_Maschine
 - als Abstraktion des Rechnersystems
- Verwaltung der 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 ...

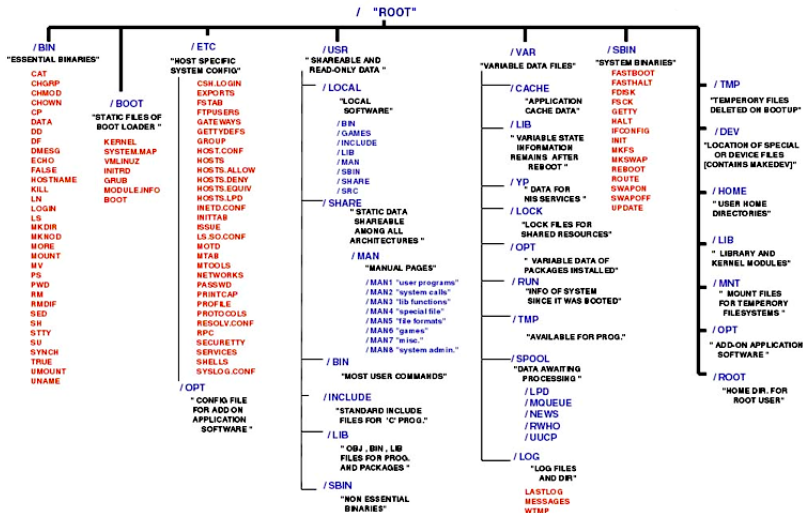
Aufbau eines Betriebssystems: Ringmodell

- 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 des Betriebssystems



Dateisystem

- 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:
 - UFS – Unix File System
 - FAT – File Allocation Table
 - NTFS – New Technology File System
 - ZFS – Zettabyte File System
 - CEPH



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 – Konfiguration, Shared-Software
 - /usr/local – lokale Software
 - /usr/share – statische Daten
 - /usr/bin – User-Land-Software/ Commands
 - /usr/include – Standard-Bibliotheken für C/C++
 - /usr/lib – Bibliotheken für Programmiersprachen
 - /usr/sbin – andere Binaries

- /var – Variable Daten
 - /var/log – zentrale Log-File-Stelle
- /sbin – System Binaries
- /tmp – Temporäre Dateien
- /dev – Geräte
- /home – User-Bereich
- /lib – Bibliotheken & Kernel-Module
- /opt – zusätzliche Software – Add Ons
- /root – Verzeichnis vom Root-User

Ausführlicher: [https:](https://en.wikipedia.org/wiki/Unix_filesystem#Conventional_directory_layout)

[//en.wikipedia.org/wiki/Unix_filesystem#Conventional_directory_layout](https://en.wikipedia.org/wiki/Unix_filesystem#Conventional_directory_layout)

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,...
- Raspberry Pi:
 - Benutzername: pi
 - Gruppen: users, wheel,...

Um die Zugriffsrechte der jeweiligen Nutzer zu regeln bietet Linux/ Unix ein Berechtigungsmodell

Abbildung der Nutzer, Gruppen auf Zugriffsmöglichkeiten der Dateien

- 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

Zahlensystem:

- Dezimalsystem – Basis 10
 - Werte 0 - 9
- Dualsystem/ Binärsystem – Basis 2
 - Werte 0 oder 1
 - Bit-Darstellung in der Informatik/ Rechnertechnik
- Oktalsystem – Basis 8
 - Werte 0 - 7
 - Für Darstellung der Zahlen 0 - 7 \rightarrow 3 Bit notwendig, $2^3 = 8$

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
 - ...

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
-rwxr-xr-x 1 root sudo 0 Sep 29 13:54 stop_redis_daemon.sh
```

Shell

- Textbasierte Schnittstelle
- Schnittstelle zwischen OS-Kernel & OS-Util. & User
- Shell ist ein Kommando-Interpreter → führt Schrittweise Befehle aus
 - Kommandos können auch Binärdatei aufgerufen werden
 - Kommandos können direkt aufgerufen werden
 - Aufruf von Systemcalls möglich → Administration des Systems

Unix-Philosophie

Nach Douglas McIlroy:

- 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: Einfach Programme die zusammenarbeiten können, sodass komplexere Probleme lösbar sind









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
- ...

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
 - Bsp.: anlegen von Dateien auf SSD, Verbindung des Browsers zu einer Webseite ...
- Daemons – Hintergrunddienste
- Stellen Dienste des BS bereit, auf die Programme zugreifen können
 - Bsp: Netzwerkdienste, Sockets ...




Shell 101

- Auf den Laborrechner:  (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

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
- 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 cat id_rsa.pub >> authorized_keys  
3 head < index.html
```

- 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