TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Operating Systems Group

PRAXISBEISPIEL UNIX

MICHAEL ROITZSCH

Vorkenntnisse

Grundlagen (laut Modulbeschreibung):

Fähigkeiten in der Rechnerarchitektur und -organisation, der imperativen Programmierung (z.B. C oder Java), Stochastik (Zufallsgrößen und -verteilung) und ein Grundverständnis von Programmverifikation …

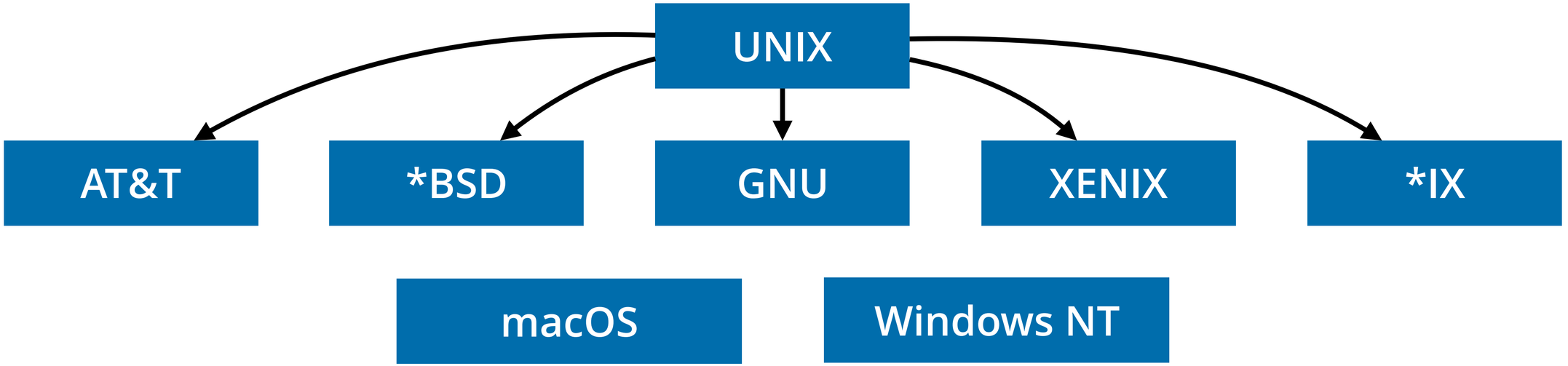
Wer hat schonmal …

- Ein Unix-basiertes System benutzt?
- Ein Programm geschrieben?
- Ein Programm in einer „Systemsprache“ geschrieben?
- Mit der Konsole/Shell gearbeitet?
- Mit Assembler-Code gearbeitet?

Unix Story

1964	MULTICS	wichtige Ideen, aber Fehlschlag
1971	Ken Thompson	UNICS auf PDP-7 (First Edition)
1973	Dennis Ritchie + KT	C, Neuimplementierung von Unix v4 in C
1974	Thompson + Ritchie	„The Unix Time-Sharing System“
1975		Unix v6, weite Verbreitung
1977	Richard Miller	Portierung auf Interdata 7/32 und 8/32 (32 Bit)
1979		Bourne-Shell, Portable C Compiler (PCC)
198x		virtueller Speicher, Netzwerke
1983	Richard Stallman	GNU Project
1985	Richard Stallman	Free Software Foundation (FSF)

Unix Wars



1986	IEEE	Portable Operating System Interface (POSIX)
1987	Tanenbaum	Minix
1989	NeXT	NeXTSTEP
199x	Torvalds et al.	Linux
2001	Apple	Mac OS X

Und was ist mit Windows?

- Vorlesung behandelt Grundlagen
  - bereits seit Unix etabliert
  - in Windows (wahrscheinlich) ähnlich umgesetzt
- Windows ungeeignet für Forschung (Software nicht einsehbar)
- heute: Windows Subsystem for Linux (WSL)
  - Linux-Kern als Teil von Windows

Exkurs: Freie Software

Gegenentwurf: Proprietäre Software

4 Freiheiten

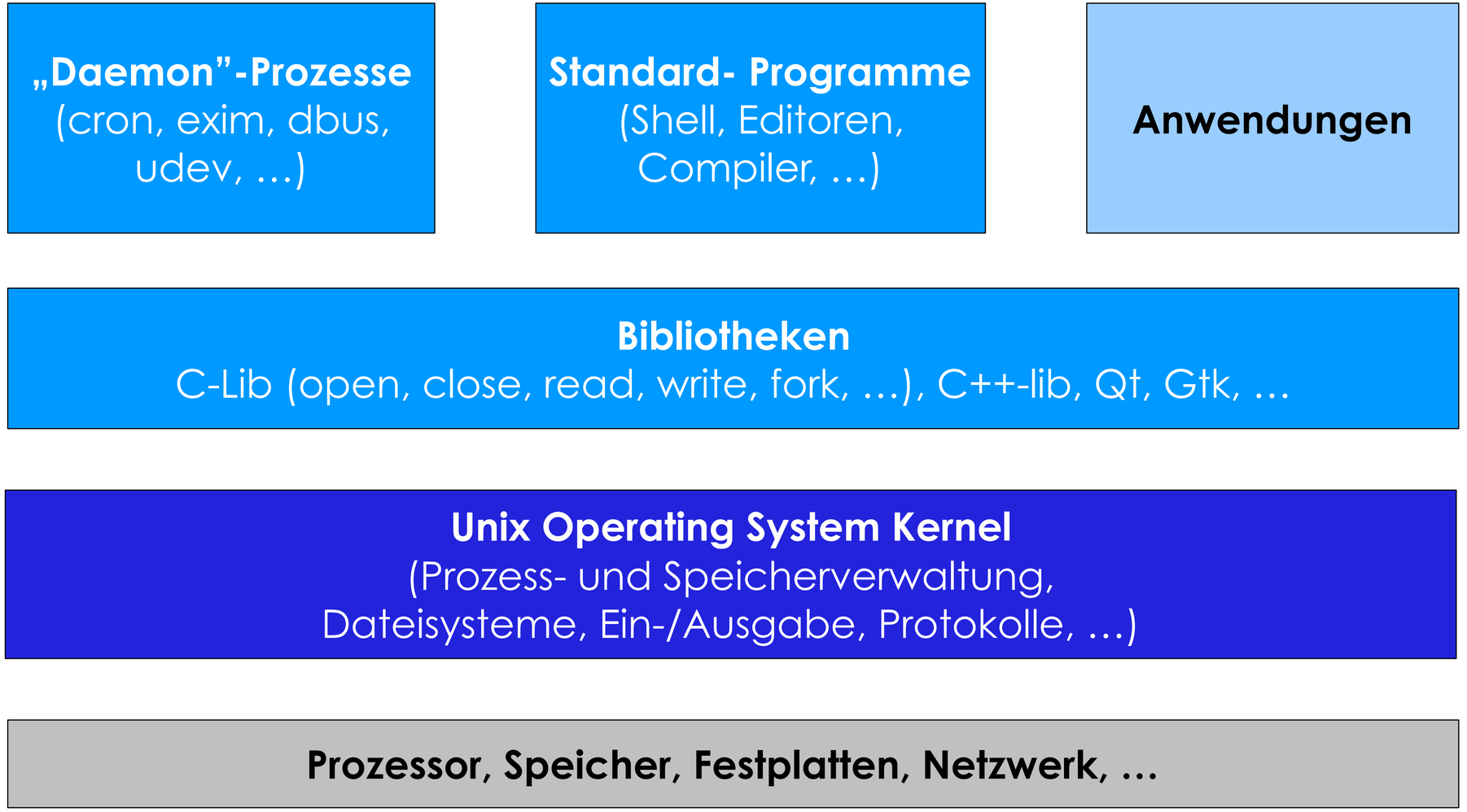
Quelle: fsfe.org

- unlimited use for any purpose
- study how the program works and understand it
- share copies of the software
- improve the program and distribute the improvements

Bekannte Lizenzen

- GPL (mit Copyleft): Linux, GNU-Software, ...
- BSD: *BSD, Chrome, Nginx, vi, ...

Grobstruktur Unix



Ausgewählte Shell-Befehle

`pwd`	Aktuelles Verzeichnis ausgeben (print working directory)
`ls`	Verzeichnisinhalt auflisten (list)
`mkdir`	Verzeichnis erstellen (make directory)
`cd`	Verzeichnis wechseln (change directory)
`cp`	Datei kopieren (copy)
`mv`	Datei umbenennen (move)
`rm`	Datei löschen (remove)
`chmod`	Dateirechte ändern (change modifiers)
`ln`	Verknüpfung zu Datei erzeugen (link)
`cat`	Dateien aneinanderhängen und ausgeben (concatenate)
`less`	Seitenweise Ausgabe/„Pager“ (vgl. more)
`ps`	Prozessliste (process status)
`man`	Anleitungen lesen (browse manual pages)

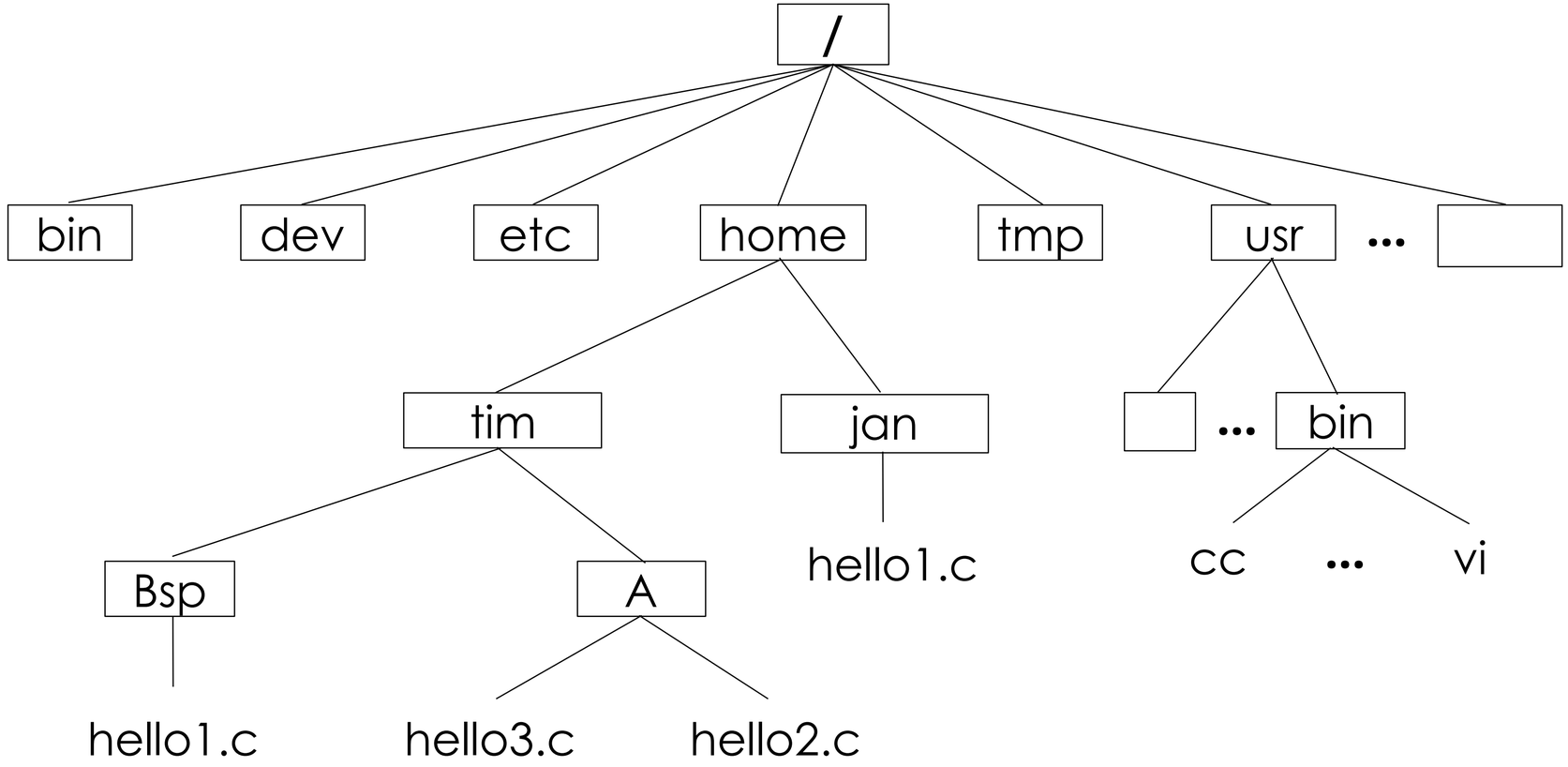
Programmentwicklung

hello1.c

```
#include <stdio.h>                          // Präprozessor-Direktive
int main(int argc, char *argv[])            // Eintrittspunkt
{
   printf("Hello World\n");
   return 0;                                // exit-Status
}                                           // (0: erfolgreich)
```

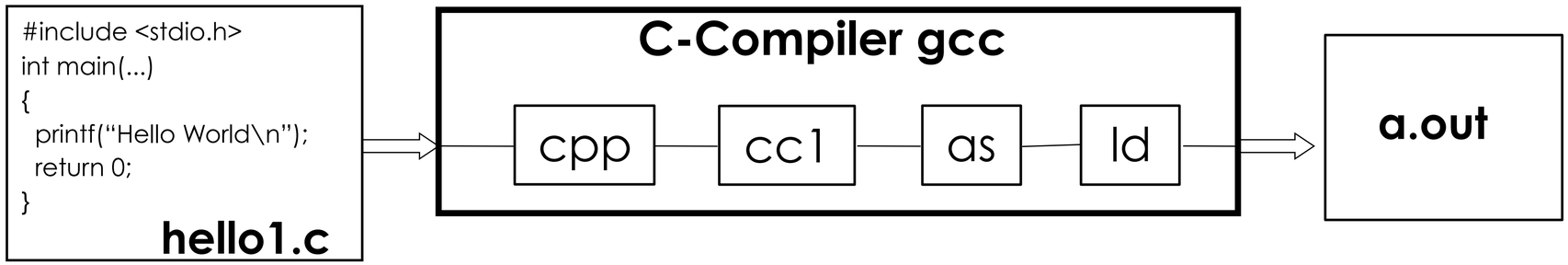
- **mkdir Bsp**
- **cp /home/jan/hello1.c Bsp**
- **cd Bsp**
- **ls -l**
- **chmod g+r hello1.c**
- **cat hello1.c**

Verzeichnisstruktur



Schritte des C-Compilers

- Präprozessieren: .c → .i C ohne Makros
- Kompilieren: .i → .s Assembler-Quelltext
- Assemblieren: .s → .o Objektdatei
- Linken: .o → a.out Programm
- .c, .i und .s sind Text Texteditor, z. B. **`vi`**
- .o ist Maschinencode **`nm`**, **`objdump`**, **`objcopy`**
- a.out ist ausführbar **`ldd`**, **`nm`**, **`readelf`**, **`gdb`**



Schnittstelle zum Betriebssystem

Systemaufrufe

hello2.c

```
#include <unistd.h>
int main(int argc, char *argv[])
{
   write(1, "Hello World\n", 12)
          ↑                    ↑
    STDOUT_FILENO        HELLO_LENGTH
   return 0;
}
```

- **cc -o hello2 hello2.c**
- **./hello2**

Systemaufrufe unter Linux

- Parameter werden in Registern übergeben
  - **`rax`** = Nummer des Systemaufrufs (0 – …)
- Systemaufrufe haben 0 bis 6 Parameter
  - **`rdi`** = 1. Parameter
  - **`rsi`** = 2. Parameter
  - **`rdx`** = 3. Parameter
  - **`r10`** = 4. Parameter
  - **`r8`** = 5. Parameter
  - **`r9`** = 6. Parameter
- Übergang in den Kern durch spezielle Instruktionen:
  `int 0x80`, **`sysenter`**, **`syscall`**
- Rückgabewert wird in Register **`rax`** übergeben
- Standard-C-Bibliothek enthält Hüllfunktionen (Wrapper)

Prozess-Struktur

hello3.c

```
#include <stdio.h>
int main(void)
{
   printf("Bitte Enter-Taste drücken...\n");
   int err = getchar();
   if (err < 0) {
      perror("getchar");
   }
   return 0;
}
```

Prozess-Struktur

- `./hello3 &`

  `Bitte Enter-Taste drücken...`
- `ps -l`

```
UID      PID   PPID   PRI   CMD
1026     331   330    75    bash
1026     433   331    76    hello3
1026     453   331    77    ps
```