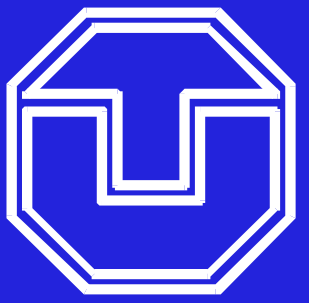
# Einführung

## Betriebssysteme und Sicherheit

Prof. Hermann Härtig
TU Dresden



# Inhalte

- Betriebssysteme
  - Prozesse und Threads
  - Scheduling
  - Speicherverwaltung
  - Dateisysteme
  - Verteilte Systeme
- Sicherheit (27.11. – 21.12.)
  - Anforderungen
  - Modellierung
  - Mechanismen

# Vorkenntnisse

Wer hat schonmal …

- Einen Computer verwendet?
- Einen Desktop/Laptop verwendet?
- Microsoft Windows benutzt?
- Ein Unix-basiertes System benutzt?
- Ein Programm in C/C++ geschrieben?
- Mit der Konsole/Shell gearbeitet?
- Mit Assembler-Code gearbeitet?

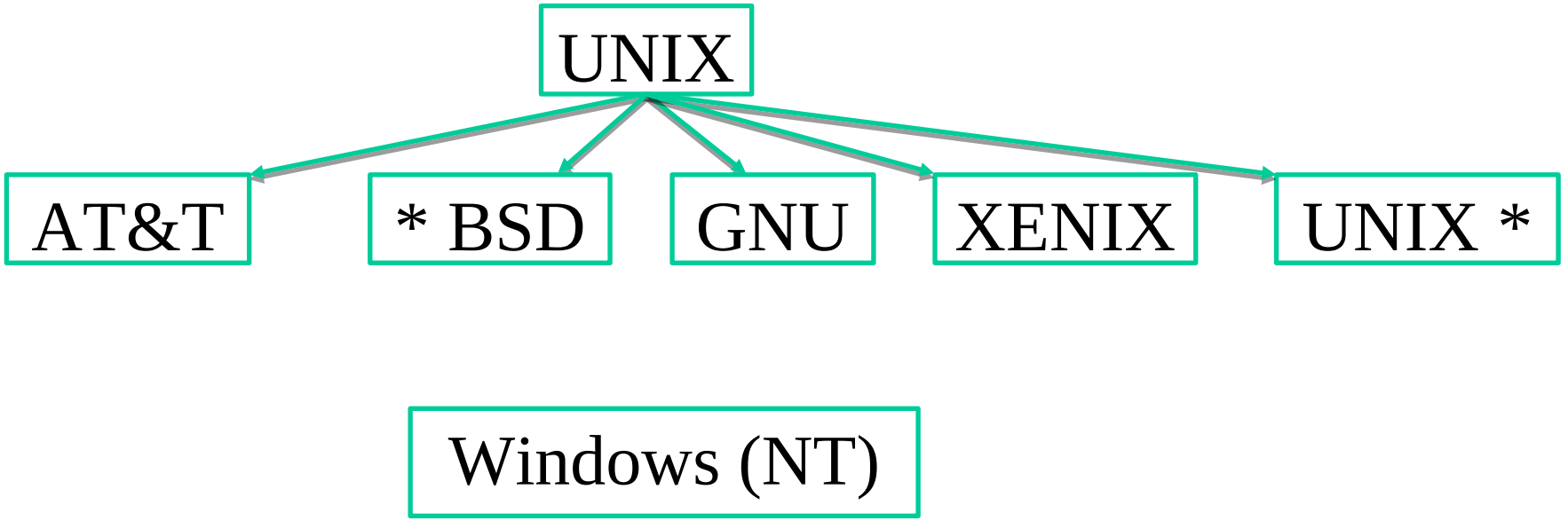
Nötige Grundlagen (Modulbeschreibung):

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

# 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 |
| 199x | Torvalds et al. | Linux |

# Und was ist mit Windows?

- Vorlesung behandelt Grundlagen
  - bereits seit Unix etabliert
  - in Windows (wahrscheinlich) ähnlich umgesetzt
- Windows ungeeignet für Forschung (proprietäre Software)
- „Windows Subsystem for Linux“ (WSL)

# Exkurs: Freie Software

- Gegenentwurf: Proprietäre Software
- 4 Freiheiten:

Quelle: https://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

**„Daemon”-Prozesse**
(cron, exim, dbus, udev, ...)

**Standard-Programme**
(X11, Shell, Editoren, Compiler, ...)

**Anwendungen**

**Bibliotheken**
C-Lib (open, close, read, write, fork, ...), X11-lib, ...

**Unix Operating System Kernel**
(Prozess- und Speicherverwaltung, Dateisysteme, Ein-/Ausgabe, Protokolle, ...)

Prozessor, Speicher, Festplatten, Netzwerk, ...

# 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) |

***Syntax:*** <Befehl> <Optionen> <Argumente> (Bsp.: **`ls -a .`**)

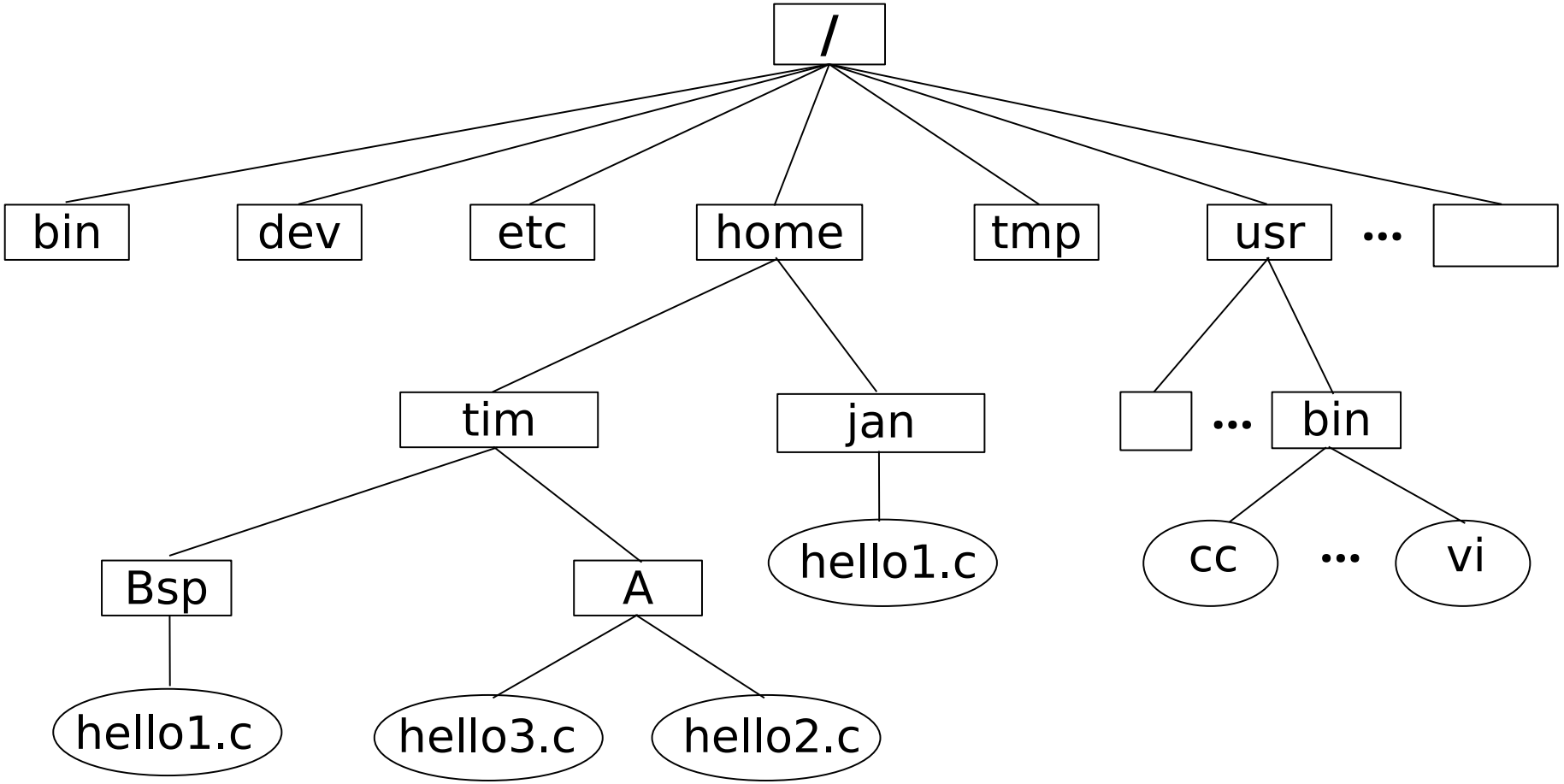
# Programmentwicklung

**hello1.c**

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

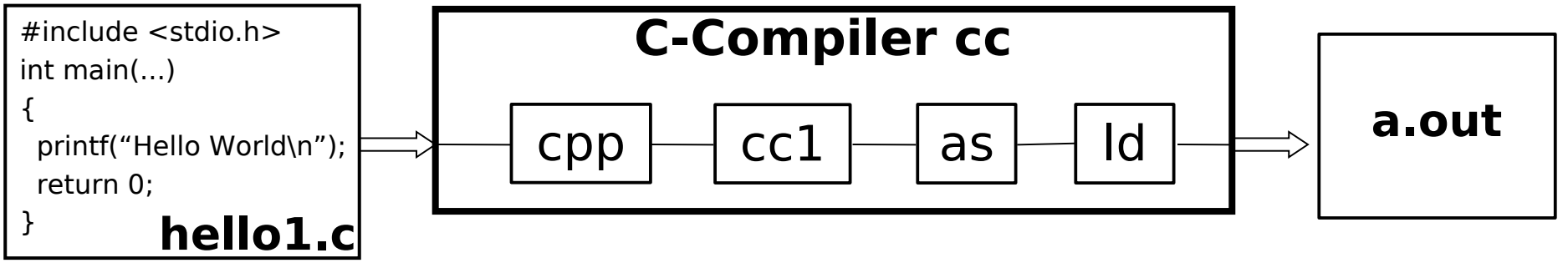
```
$ mkdir Bsp
$ cp /home/jan/hello1.c Bsp
$ cd Bsp
$ ls -l
$ chmod g+r hello1.c
$ cat hello1.c
```

# Verzeichnisstruktur



# Schritte des C-Compilers

- cc ist ein „Frontend“ für folgende Teile:
  - Präprozessor `cpp`: .c ⇒ .i — C ohne Makros
  - Compiler `cc1`: (.i) .c ⇒ .s — Assemblerquelltext
  - Assembler `as`: .s ⇒ .o — Objektdatei
  - Linker `ld`: .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 unter Linux *(siehe `/usr/include/asm/unistd.h`)*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| fork | getegid | profil | writev | ftruncate64 | get_thread_area | migrate_pages | open_by_handle_at |
| read | acct | statfs | getsid | stat64 | io_setup | openat | clock_adjtime |
| write | umount2 | fstatfs | fdatasync | lstat64 | io_destroy | mkdirat | syncfs |
| open | lock | ioperm | _sysctl | fstat64 | io_getevents | mknodat | sendmmsg |
| close | ioctl | socketcall | mlock | lchown32 | io_submit | fchownat | setns |
| waitpid | fcntl | syslog | munlock | getuid32 | io_cancel | futimesat | process_vm_readv |
| creat | mpx | setitimer | mlockall | getgid32 | fadvise64 | fstatat64 | process_vm_writev |
| link | setpgid | getitimer | munlockall | geteuid32 | exit_group | unlinkat | kcmp |
| unlink | ulimit | stat | sched_setparam | getegid32 | lookup_dcookie | renameat | finit_module |
| execve | oldolduname | lstat | sched_getparam | setreuid32 | epoll_create | linkat | sched_setattr |
| chdir | umask | fstat | sched_setscheduler | setregid32 | epoll_ctl | symlinkat | sched_getattr |
| time | chroot | olduname | sched_getscheduler | getgroups32 | epoll_wait | readlinkat | renameat2 |
| mknod | ustat | iopl | sched_yield | setgroups32 | remap_file_pages | fchmodat | seccomp |
| chmod | dup2 | vhangup | sched_get_priority_max | fchown32 | set_tid_address | faccessat | getrandom |
| lchown | getppid | idle | sched_get_priority_min | setresuid32 | timer_create | pselect6 | memfd_create |
| break | getpgrp | vm86old | sched_rr_get_interval | getresuid32 | timer_settime | ppoll | bpf |
| oldstat | setsid | wait4 | nanosleep | setresgid32 | timer_gettime | unshare | execveat |
| lseek | sigaction | swapoff | mremap | getresgid32 | timer_getoverrun | set_robust_list | socket |
| getpid | sgetmask | sysinfo | setresuid | chown32 | timer_delete | get_robust_list | socketpair |
| mount | ssetmask | ipc | getresuid | setuid32 | clock_settime | splice | bind |
| umount | setreuid | fsync | vm86 | setgid32 | clock_gettime | sync_file_range | connect |
| setuid | setregid | sigreturn | query_module | setfsuid32 | clock_getres | tee | listen |
| getuid | sigsuspend | clone | poll | setfsgid32 | clock_nanosleep | vmsplice | accept4 |
| stime | sigpending | setdomainname | nfsservctl | pivot_root | statfs64 | move_pages | getsockopt |
| ptrace | sethostname | uname | setresgid | mincore | fstatfs64 | getcpu | setsockopt |
| alarm | setrlimit | modify_ldt | getresgid | madvise | tgkill | epoll_pwait | getsockname |
| oldfstat | getrlimit | adjtimex | prctl | getdents64 | utimes | utimensat | getpeername |
| pause | getrusage | mprotect | rt_sigreturn | fcntl64 | fadvise64_64 | signalfd | sendto |
| utime | gettimeofday | sigprocmask | rt_sigaction | gettid | vserver | timerfd_create | sendmsg |
| stty | settimeofday | create_module | rt_sigprocmask | readahead | mbind | eventfd | recvfrom |
| gtty | getgroups | init_module | rt_sigpending | setxattr | get_mempolicy | fallocate | recvmsg |
| access | setgroups | delete_module | rt_sigtimedwait | lsetxattr | set_mempolicy | timerfd_settime | shutdown |
| nice | select | get_kernel_syms | rt_sigqueueinfo | fsetxattr | mq_open | timerfd_gettime | userfaultfd |
| ftime | symlink | quotactl | rt_sigsuspend | getxattr | mq_unlink | signalfd4 | membarrier |
| sync | oldlstat | getpgid | pread64 | lgetxattr | mq_timedsend | eventfd2 | mlock2 |
| kill | readlink | fchdir | pwrite64 | fgetxattr | mq_timedreceive | epoll_create1 | copy_file_range |
| rename | uselib | bdflush | chown | listxattr | mq_notify | dup3 | preadv2 |
| mkdir | swapon | sysfs | getcwd | llistxattr | mq_getsetattr | pipe2 | pwritev2 |
| rmdir | reboot | personality | capget | flistxattr | kexec_load | inotify_init1 | pkey_mprotect |
| dup | readdir | afs_syscall | capset | removexattr | waitid | preadv | pkey_alloc |
| pipe | mmap | setfsuid | sigaltstack | lremovexattr | add_key | pwritev | pkey_free |
| times | munmap | setfsgid | sendfile | fremovexattr | request_key | rt_tgsigqueueinfo | statx |
| prof | truncate | _llseek | getpmsg | tkill | keyctl | perf_event_open | arch_prctl |
| brk | ftruncate | getdents | putpmsg | sendfile64 | ioprio_set | recvmmsg | io_pgetevents |
| setgid | fchmod | _newselect | vfork | futex | ioprio_get | fanotify_init | rseq |
| getgid | fchown | flock | ugetrlimit | sched_setaffinity | inotify_init | fanotify_mark | |

*Kernel 2.2 / 2.4 / 2.6 / 2.6.28 / 3.0 / 4.18 ⇒ 190 / 237 / 273 / 331 / 346 / 383 Systemaufrufe*

# Programmentwicklung

**hello2.c**

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

```
$ cc hello2.c
$ ./hello2
```

# Systemaufrufe unter Linux

- Parameter werden in Registern übergeben

  **`eax`** = Nummer des Systemaufrufs (0 – ...)

- Systemaufrufe haben 0 bis 5 Parameter

  **`ebx`** = 1. Parameter
  **`ecx`** = 2. Parameter
  **`edx`** = 3. Parameter
  **`esi`** = 4. Parameter
  **`edi`** = 5. Parameter

- Übergang in den Kern durch Software-Interrupt „**`int 0x80`**“, Instruktion „**`sysenter`**“ bzw. „**`syscall`**“ oder spez. Speicherzugriff

- Rückgabewert wird in Register eax übergeben

- Standard-C-Bibliothek enthält Hüllfunktionen („Wrapper“)

  z. B.: #include <unistd.h>

  ssize_t **write**(int fd, const void *buf, size_t count);

  ebx ecx edx

# Prozess-Struktur

**hello3.c**

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

# Prozess-Struktur

```
$ ./hello3 &
[1] 433
$ 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 |

# Organisatorisches

- Webseite: www.inf.tu-dresden.de/index.php?node_id=1312
- Mailingliste: bs2018@os.inf.tu-dresden.de
- Organisation: jan.bierbaum@os.inf.tu-dresden.de
- Übungen
  - Aufgaben auf Webseite
  - Einschreibungen über jExam ab heute Nachmittag
  - eine englische Übungsgruppe
  - zweite Vorlesungswoche: praktische Übung/Konsultation
- Klausur
  - Inhalt: Vorlesung und Übungsstoff (Fokus auf Übung)
  - Verständnis und Anwendung, nicht reine Wiedergabe des Stoffs
  - Hilfsmittel: einfacher Taschenrechner und „Spickzettel“
  - voraussichtlich **keine** Wiederholungsprüfung im SS 2019