

Fallbeispiel Unix

Betriebssysteme

Hermann Härtig
TU Dresden



Wegweiser

Geschichte und Struktur von Unix

Vom Programm zum Prozess

Unix-Grundkonzepte

- Dateien
- Prozesse

Prozess-Kommunikation

- Signale
- Pipes
- Sockets

Rechte und Schutz

Unix-Story

196x	MULTICS (MIT)	wichtige Ideen, aber „Fehlschlag“
1971	Ken Thompson	„UNICS" auf PDP-7 (First Edition)
1973	Dennis Ritchie + KT	C, rewrite in C
1974	TR74	The Unix Time-Sharing System
1975		Sixth Edition, weite Verbreitung
1977	Richards	Portierung auf Interdata (32 Bit)
1979		Bourne-Shell, PCC
198x		virtueller Speicher, Netzwerke
1983	Stallman	GNU Project
1985	Stallman	Free Software Foundation
1986	IEEE	Posix
		“Unix wars“
199x	L. Torvalds et al.	Linux

Grobstruktur Unix

“Daemon” Processes

(cron, exim, dbus,
udevd, ...)

Standard Utility Programs

(X11, shell, editors,
compilers, etc.)

Anwen- dungen

Libraries

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

Unix Operating System Kernel

(process management, memory management,
file systems, I/O, protocols, etc.)

CPU, memory, disks, terminals, etc.

Wegweiser

Geschichte und Struktur von Unix

Vom Programm zum Prozess

Unix-Grundkonzepte

- Dateien
- Prozesse

Prozess-Kommunikation

- Signale
- Pipes
- Sockets

Rechte und Schutz

Ausgewählte Shell-Kommandos

pwd	print name of working directory
ls	list a directory
mkdir	make a directory
cd	change directory
mv	move (rename) files
rm	remove (delete) files
chmod	change file access permissions
cp	copy
ln	make links between files
cat	concatenate files and print on standard output
less	opposite of more
ps	process list
man	browse manual pages

***Syntax:* name options arguments**

Programmentwicklung

hello1.c

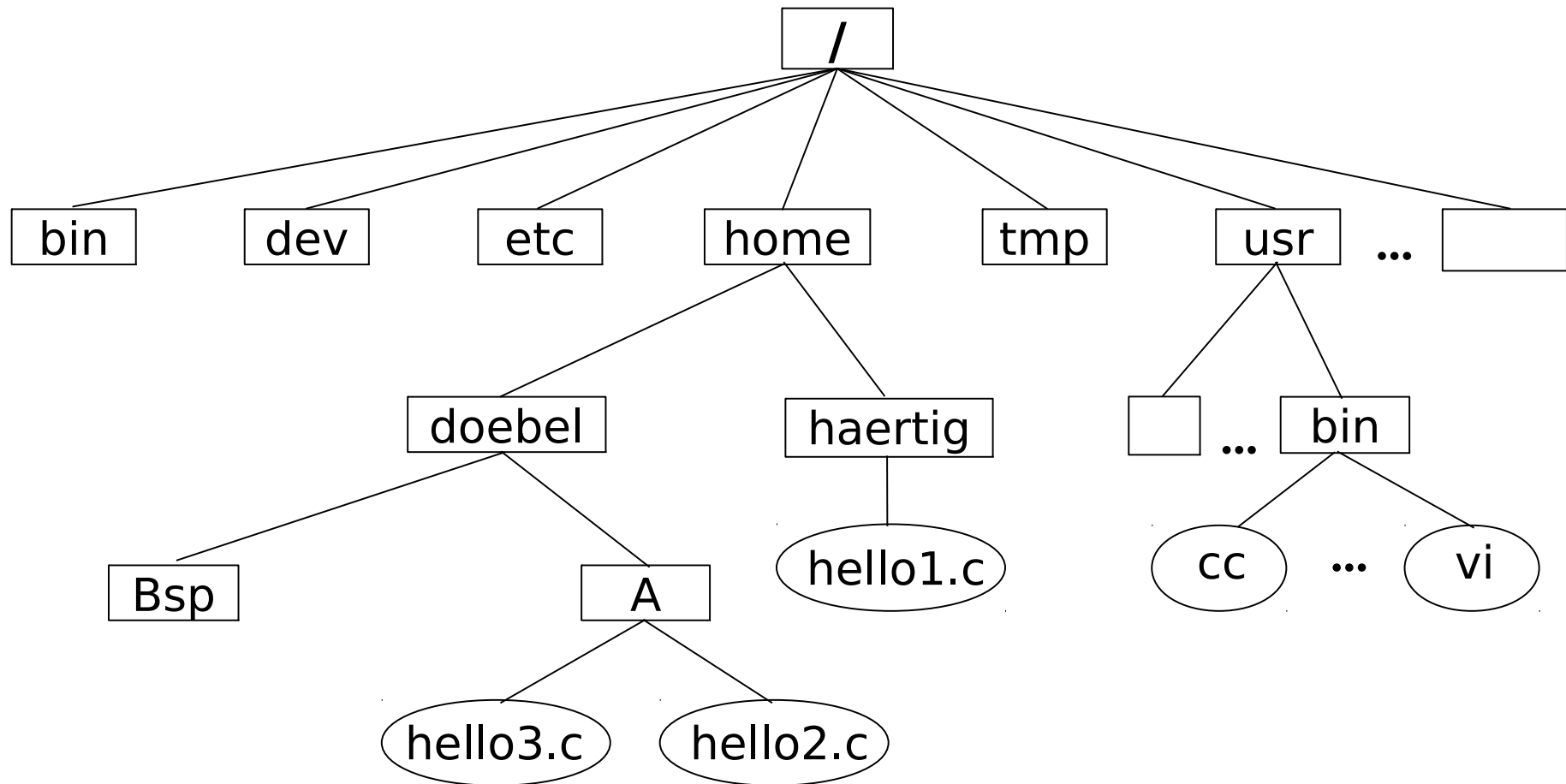
```
#include <stdio.h>
int main(int argc, char *argv[])
{
    printf("Hello World\n");
    return 0;
}
```

Präprozessor-Direktive
Eintrittspunkt

exit-Status
(0: erfolgreich)

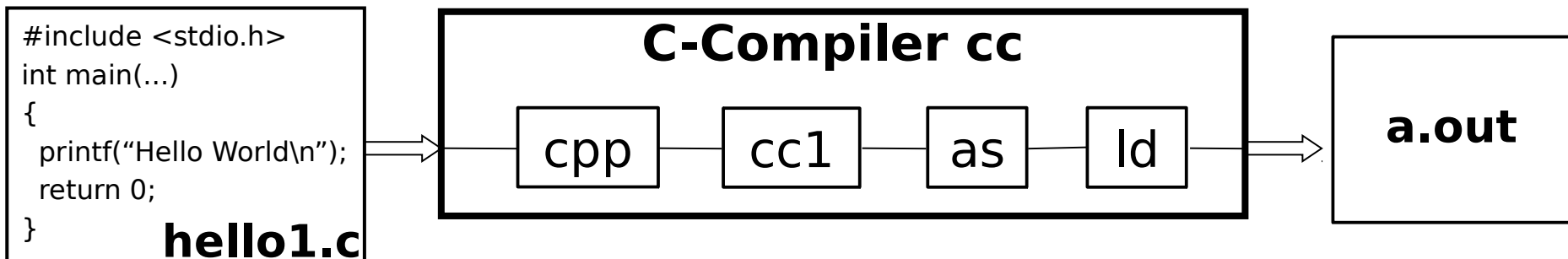
```
mkdir Bsp
cp /home/haertig/hello1.c Bsp
cd Bsp
ls -l
chmod g+r hello1.c
cat hello1.c
```

Verzeichnisstruktur



Schritte des C-Compilers

- cc ist ein „Frontend“ für folgende Teile:
 - Präprozessor cpp: `.c ⇒ .i` C ohne Makros
 - Compiler cc1: `(.i) .c ⇒ .s` Assembler Quelltext
 - Assembler as: `.s ⇒ .o` Objektdaten
 - Linker ld: `.o ⇒ a.out` Programm
- `.c`, `.i` und `.s` sind Text $\Rightarrow$ Texteditor, z. B. vi
- `.o` ist Maschinencode $\Rightarrow$ nm, objdump, objcopy
- `a.out` ist ausführbar $\Rightarrow$ ldd, nm, readelf, gdb



Schritte des C-Compilers

- Präprozessor:
`cc -E -o hello1.i hello1.c`
`less hello1.i`
- Compiler:
`cc -S -o hello1.s hello1.i`
- Assembler:
`cc -c -o hello1.o hello1.s`
`objdump -lSd hello1.o`
- Linker:
`cc -o hello1 hello1.o`

`hello1`

`ls -l h*`

Rolle des Betriebssystems

```
void _start()
{
    // setup
    status = main(argc, argv);
    int main(int argc, char **argv)
    {
        printf("Hello World\n");
        int printf(...) → int vprintf(...)
        {
            write(stdout, "Hello world\n", 12);
        }
        return 0;
    }
    exit(status);
}
```

libc.so.6

a.out

crt1.o

Schnittstelle zum Betriebssystem

346 Systemaufrufe unter Linux (siehe */usr/include/asm/unistd.h*)

Kernel 2.2/2.4/2.6/2.6.28/3.0: 190 ⇒ 237 ⇒ 273 ⇒ 331 ⇒ 346 Systemaufrufe

restart_syscall exit fork read write open close waitpid creat link unlink execve chdir time mknod chmod lchown break oldstat lseek
getpid mount umount setuid getuid stime ptrace alarm oldfstat pause utime stty gtty access nice ftime sync kill rename mkdir rmdir
dup pipe times prof brk setgid getgid signal geteuid getegid acct umount2 lock ioctl fcntl mpx setpgid ulimit oldolduname umask
chroot ustat dup2 getppid getpgrp setsid sigaction sgetmask ssetmask setreuid setregid sigsuspend sigpending sethostname
setrlimit getrlimit getrusage gettimeofday settimeofday getgroups setgroups select symlink oldlstat readlink uselib swapon reboot
readdir mmap munmap truncate ftruncate fchmod fchown getpriority setpriority profil statfs fstatfs ioperm socketcall syslog setitimer
getitimer stat lstat fstat olduname iopl vhangup idle vm86old wait4 swapoff sysinfo ipc fsync sigreturn clone setdomainname uname
modify_ldt adjtimex mprotect sigprocmask create_module init_module delete_module get_kernel_syms quotactl getpgid fchdir
bdflush sysfs personality afs_syscall setfsuid setfsgid _llseek getdents _newselect flock msync readv writev getsid fdasysync _sysctl
mlock munlock mlockall munlockall sched_setparam sched_getparam sched_setscheduler sched_getscheduler sched_yield
sched_get_priority_max sched_get_priority_min sched_rr_get_interval nanosleep mremap setresuid getresuid vm86 query_module
poll nfsservctl setresgid getresgid prctl rt_sigreturn rt_sigaction rt_sigprocmask rt_sigpending rt_sigtimedwait rt_sigqueueinfo
rt_sigsuspend pread64 pwrite64 chown getcwd capget capset sigaltstack sendfile getpmsg putpmsg vfork ugetrlimit mmap2
truncate64 ftruncate64 stat64 lstat64 fstat64 lchown32 getuid32 getgid32 geteuid32 getegid32 setreuid32 setregid32 getgroups32
setgroups32 fchown32 setresuid32 getresuid32 setresgid32 getresgid32 chown32 setuid32 setgid32 setfsuid32 setfsgid32
pivot_root mincore madvise madvise1 getdents64 fcntl64 gettid readahead setxattr lsetxattr fsetxattr getxattr lgetxattr fgetxattr
listxattr llistxattr flistxattr removexattr lremovexattr fremovexattr tkill sendfile64 futex sched_setaffinity sched_getaffinity
set_thread_area get_thread_area io_setup io_destroy io_getevents io_submit io_cancel fadvise64 exit_group lookup_dcookie
epoll_create epoll_ctl epoll_wait remap_file_pages set_tid_address timer_create timer_settime timer_gettime timer_getoverrun
timer_delete clock_settime clock_gettime clock_getres clock_nanosleep statfs64 fstatfs64 tkill utimes fadvise64_64 vserver mbind
get_mempolicy set_mempolicy mq_open mq_unlink mq_timedsend mq_timedreceive mq_notify mq_getsetattr kexec_load waitid
add_key request_key keyctl ioprio_set ioprio_get inotify_init inotify_add_watch inotify_rm_watch migrate_pages openat mkdirat
mknodat fchownat futimesat fstatat64 unlinkat renameat linkat symlinkat readlinkat fchmodat faccessat pselect6 ppoll unshare
set_robust_list get_robust_list splice sync_file_range tee vmsplice move_pages getcpu epoll_pwait utimensat signalfd
timerfd_create eventfd falloca timerfd_settime timerfd_gettime signalfd4 eventfd2 epoll_create1 dup3 pipe2 inotify_init1 preadv
pwritev rt_tsigqueueinfo perf_event_open recvmmsg fanotify_init fanotify_mark prlimit64 name_to_handle_at open_by_handle_at
clock_adjtime syncfs sendmmsg setns

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 Softwareinterrupt „int 0x80“, Instruktion „sysenter“ oder spez. Speicherzugriff
- Rückgabewert wird in Register eax übergeben
- Bibliothek libc 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