



**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 ...

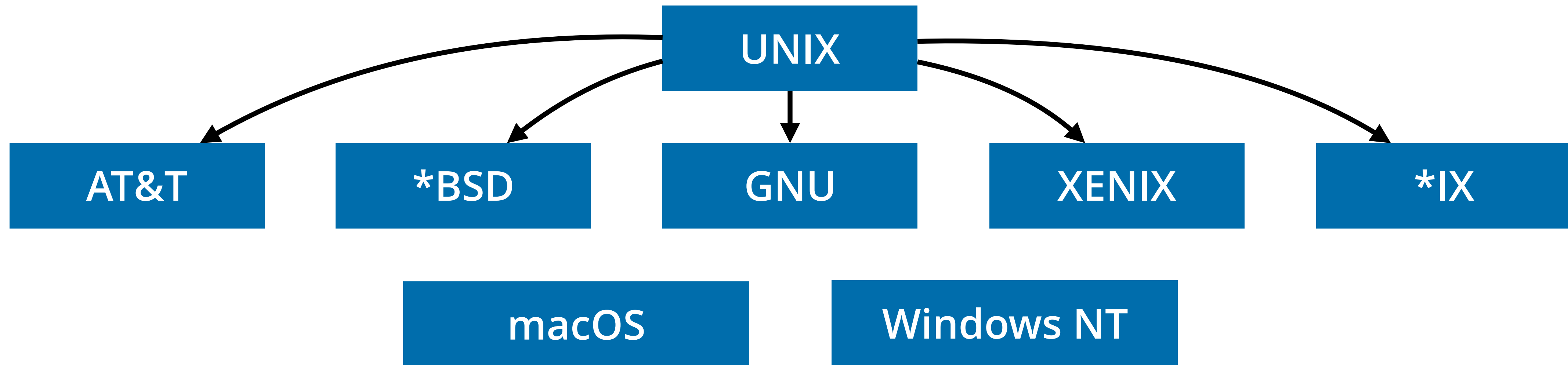
Wer hat schonmal ...

- 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

1987 Tanenbaum

1989 NeXT

199x Torvalds et al.

2001 Apple

Portable Operating System Interface (POSIX)

Minix

NeXTSTEP

Linux

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

- 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

Quelle: fsfe.org

Bekannte Lizenzen

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

Grobstruktur Unix

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

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

Anwendungen

Bibliotheken

C-Lib (open, close, read, write, fork, ...), C++-lib, Qt, Gtk, ...

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)

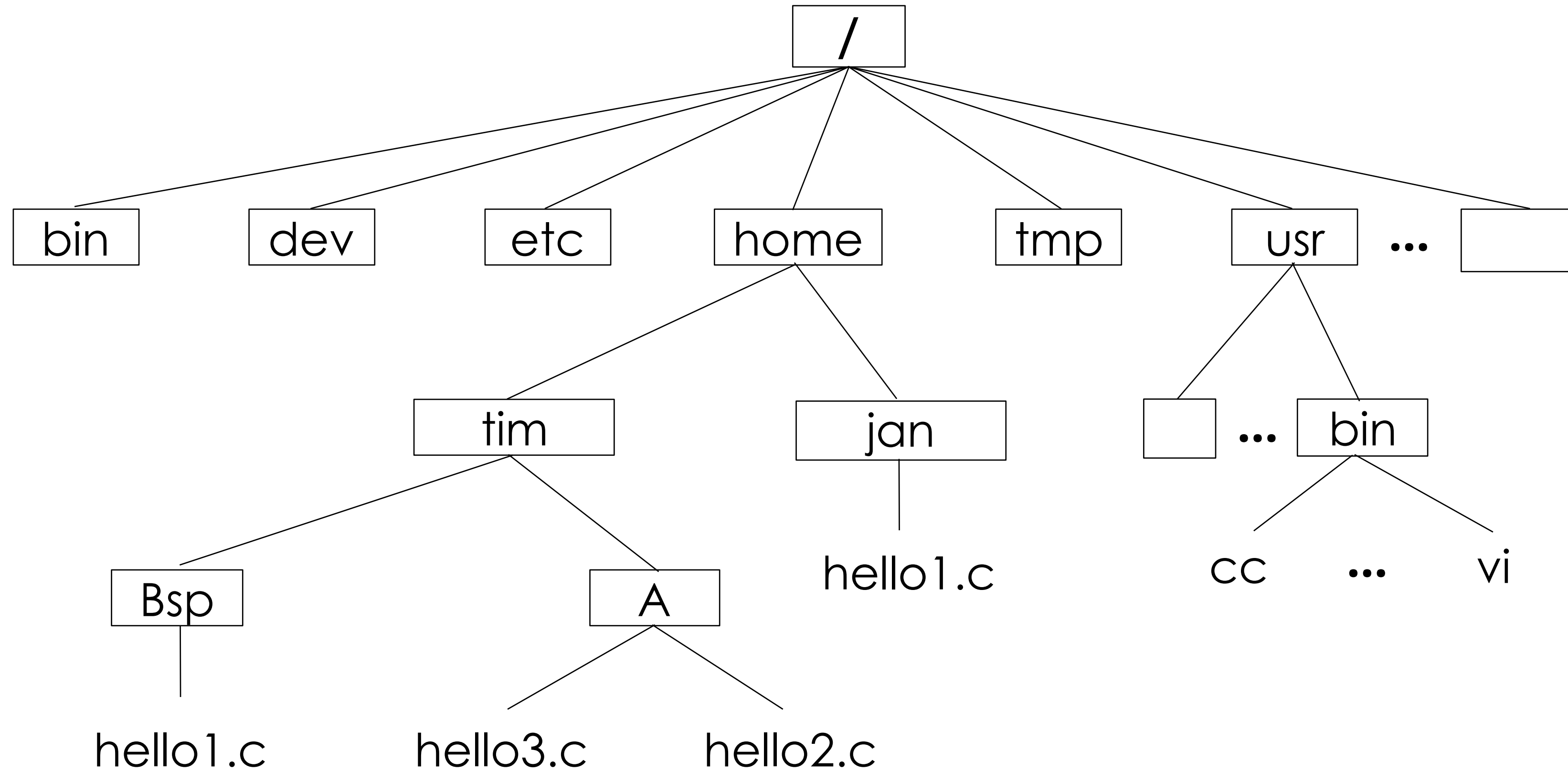
Programmentwicklung

hello1.c

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

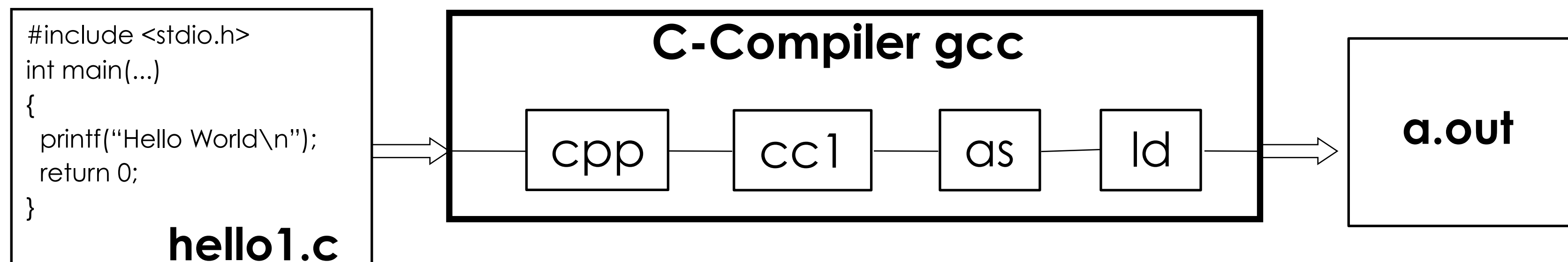
- ▶ **mkdir Bsp**
- ▶ **cp /home/jan/hello1.c Bsp**
- ▶ **cd Bsp**
- ▶ **ls -l**
- ▶ **chmod g+r hello1.c**
- ▶ **cat hello1.c**

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

fork	getegid	profil	wrlev	ltruncate64	get_thread_area	migrate_pages	open_by_handle_at
read	acct	stfs	getsid	stat64	to_set	penat	clock_adjtime
write	lock	stafs	fcntl64	fcntl64	io_submit	mknodat	syncfs
open	lock	ioperm	_sysctl	fstat64	io_getevents	fchownat	sendmsg
close	ioctl	socketcall	mlock	lchown32	io_cancel	futimesat	setns
waitpid	fcntl	syslog	munlock	getuid32	exit_group	fstatat64	process_vm_readv
creat	mpx	setitimer	mlockall	getgid32	lookup_dcookie	unlinkat	process_vm_writev
link	setpgid	getitimer	munlockall	geteuid32	epoll_create	renameat	kcmp
unlink	ulimit	stat	sched_setparam	getegid32	epoll_ctl	linkat	finit_module
execve	oldolduname	lstat	sched_getparam	setreuid32	epoll_wait	symlinkat	sched_setattr
chdir	umask	fstat	sched_setscheduler	setregid32	remap_file_pages	readlinkat	sched_getattr
time	chroot	olduname	sched_getscheduler	getgroups32	set_tid_address	fchmodat	renameat2
mknod	ustat	iopl	sched_yield	setgroups32	timer_create	facecessat	seccomp
chmod	dup2	vhangup	sched_get_priority_max	fchown32	timer_settime	pselect6	getrandom
lchown	getppid	idle	sched_get_priority_min	setresuid32	timer_gettime	ppoll	memfd_create
break	getpgrp	vm86old	sched_rr_get_interval	getresuid32	timer_getoverrun	unshare	bpf
oldstat	setsid	wait4	nanosleep	setresgid32	timer_delete	set_robust_list	execveat
lseek	sigaction	swapoff	mremap	getresgid32	clock_settime	get_robust_list	socket
getpid	sgetmask	sysinfo	setresuid	chown32	clock_gettime	splice	socketpair
mount	ssetmask	ipc	getresuid	setuid32	clock_getres	sync_file_range	bind
umount	setreuid	fsync	vm86	setgid32	clock_nanosleep	tee	connect
setuid	setregid	sigreturn	query_module	setfsuid32	statfs64	vmsplice	listen
getuid	sigsuspend	clone	poll	setfsgid32	fstatfs64	move_pages	accept4
stime	sigpending	setdomainname	nfsservctl	pivot_root	tgkill	getcpu	getsockopt
ptrace	sethostname	uname	setresgid	mincore	utimes	epoll_pwait	setsockopt
alarm	setrlimit	modify_ldt	getresgid	madvise	fcntl64	utimensat	getsockname
oldfstat	getrlimit	adjtimex	prctl	getdents64	gettid	signalfd	getpeername
pause	getrusage	mprotect	rt_sigreturn	fcntl64	readahead	timerfd_create	sendto
utime	gettimeofday	sigprocmask	rt_sigaction	gettid	setxattr	eventfd	sendmsg
stty	settimeofday	create_module	rt_sigprocmask	readahead	lsetxattr	fallocate	recvfrom
gtty	getgroups	init_module	rt_sigpending	setxattr	fsetxattr	timerfd_settime	recvmsg
access	setgroups	delete_module	rt_sigtimedwait	lsetxattr	getxattr	timerfd_gettime	shutdown
nice	select	get_kernel_syms	rt_sigqueueinfo	getxattr	lgetxattr	signalfd4	userfaultfd
ftime	symlink	quotactl	rt_sigsuspend	fgetxattr	listxattr	eventfd2	membarrier
sync	oldlstat	getpgid	pread64	listxattr	lsetxattr	epoll_create1	mlock2
kill	readlink	fchdir	pwrite64	lsetxattr	fgetxattr	dup3	copy_file_range
rename	uselib	bdflush	chown	lsetxattr	listxattr	pipe2	preadv2
mkdir	swapon	sysfs	getcwd	lsetxattr	lsetxattr	inotify_init1	pwritev2
rmdir	reboot	personality	capget	lsetxattr	lsetxattr	preadv	pkey_mprotect
dup	readdir	afs_syscall	capset	lsetxattr	lsetxattr	pwritev	pkey_alloc
pipe	mmap	setfsuid	sigaltstack	lsetxattr	lsetxattr	rt_tgsigqueueinfo	pkey_free
times	munmap	setfsgid	sendfile	lsetxattr	lsetxattr	perf_event_open	statx
prof	truncate	_llseek	getpmsg	lsetxattr	lsetxattr	keyctl	arch_prctl
brk	fcntl64	getdents	getpmsg	lsetxattr	lsetxattr	keyctl	arch_prctl
setgid	fcntl64	getdents64	getpmsg	lsetxattr	lsetxattr	keyctl	arch_prctl
setuid	fcntl64	getdents64	getpmsg	lsetxattr	lsetxattr	keyctl	arch_prctl
setgid	fcntl64	getdents64	getpmsg	lsetxattr	lsetxattr	keyctl	arch_prctl

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

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