Kernels Make The OS

`https://l.mafiasi.de/kernel-kbs`

Bene 17ostendo
Finn 17sell

15. Januar 2024

Well actually, Linux is just the Kernel. What you're talking about is the GNU/Linux Operating System...

Warum eigentlich einen Kernel entwickeln?

Linux Syscalls

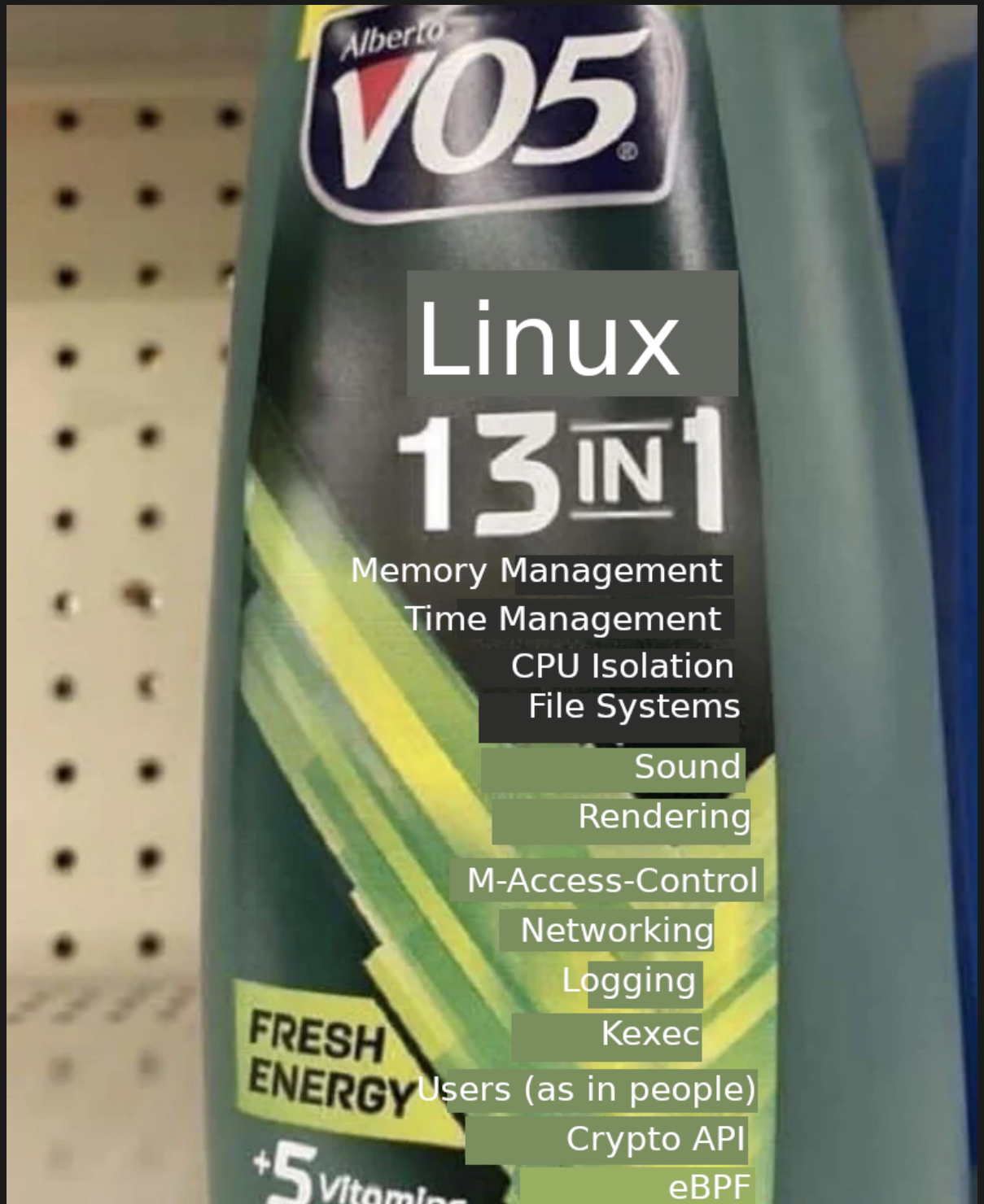
read write open close stat fstat lstat poll lseek
mmap mprotect munmap brk rt_sigaction
rt_sigprocmask rt_sigreturn ioctl pread64 pwrite64
readv writev access pipe select sched_yield mremap
msync mincore madvise shmget shmat shmctl dup
dup2 pause nanosleep getitimer alarm setitimer
getpid sendfile socket connect accept sendto
recvfrom sendmsg recvmsg shutdown bind
getsockname getpeername socketpair setsockopt
getsockopt clone fork vfork execve exit wait4 kill

clone fork vfork execve exit wait4 kill uname semget semop semctl shmdt msgget msgsnd msgrcv msgctl fcntl flock fsync fdatasync truncate ftruncate getdents getcwd chdir fchdir rename mkdir rmdir creat link unlink symlink readlink chmod fchmod chown fchown lchown umask gettimeofday getrlimit getrusage sysinfo times ptrace getuid syslog getgid setuid setgid geteuid getegid setpgid getppid getpgrp setsid setreuid setregid getgroups setgroups setresuid getresuid setresgid getresgid getpgid setfsuid setfsgid getsid capget capset rt_sigpending rt_sigtimedwait rt_sigqueueinfo rt_sigsuspend sigaltstack utime mknod uselib ustat statfs fstatfs sysfs getpriority setpriority sched_setparam sched_getparam sched_setscheduler sched_getscheduler sched_get_priority_max sched_get_priority_min sched_rr_get_interval mlock munlock munlockall vhangup modify_ldt pivot_root _sysctl prctl arch_prctl adjtimex setrlimit chroot sync acct settimeofday mount umount2 swapon swapoff reboot sethostname setdomainname iopl ioperm create_module delete_module get_kernel_syms nfsservctl readahead setxattr lsetxattr fsetxattr getxattr lgetxattr fgetxattr listxattr llistxattr flistxattr removexattr lremovexattr fremovexattr tkill time futex sched_setaffinity sched_getaffinity set_thread_area io_destroy io_getevents io_submit io_cancel get_thread_area epoll_ctl_old getdents64 set_tid_address restart_syscall semtimedop fadvise64 timer_create timer_settime timer_gettime timer_getoverrun timer_delete clock_settime clock_gettime clock_getres clock_nanosleep exit_group epoll_wait epoll_ctl tgkill utimes vserver set_mempolicy get_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 newfstatat unlinkat renameat linkat readlinkat fchmodat faccessat pselect6 ppoll unshare set_robust_list get_robust_list splice tee sync_file_range vmsplice move_pages utimensat epoll_pwait signalfd timerfd_create eventfd fallocate timerfd_settime timerfd_gettime accept4 signalfd4 eventfd2 epoll_create1 dup3 pipe2 inotify_init1 preadv pwritev rt_tgsigqueueinfo perf_event_open recvmmsg fanotify_init fanotify_mark prlimit64 name_to_handle_at open_by_handle_at clock_adjtime syncfs sendmmsg setns getcpu process_vm_readv process_vm_writev kcmp finit_module sched_setattr sched_getattr renameat2 seccomp getrandom memfd_create kexec_file_load bpf execveat userfaultfd membarrier mlock2 copy_file_range preadv2 pwritev2 pkey_mprotect pkey_alloc pkey_free statx io_pgetevents rseq 't numbers 387 through 423, add new calls after the last " pidfd_send_signal io_uring_setup io_uring_enter io_uring_register open_tree move_mount fsopen fsconfig fsmount fspick pidfd_open clone3 close_range openat2 pidfd_getfd faccessat2 process_madvise epoll_pwait2 mount_setattr landlock_create_ruleset landlock_add_rule landlock_restrict_self memfd_secret process_mrelease futex_waitv set_mempolicy_home_node cachestat fchmodat2 map_shadow_stack futex_wake futex_wait futex_requeue statmount listmount lsm_get_self_attr lsm_set_self_attr lsm_list_modules rt_sigaction rt_sigreturn ioctl readv writev recvfrom sendmsg recvmsg execve ptrace rt_sigpending rt_sigtimedwait rt_sigqueueinfo sigaltstack timer_create mq_notify kexec_load waitid set_robust_list get_robust_list vmsplice move_pages preadv pwritev rt_tgsigqueueinfo recvmmsg sendmmsg process_vm_readv process_vm_writev setsockopt getsockopt io_setup io_submit execveat preadv2 pwritev2

Linux Kernel Features

- Dateisystem (-e)
- Nutzer (-Rechte)
- PAM
- Netzwerk
- Firewall x2

und vieles mehr...


Alberto
VO5
Linux
13 IN 1
Memory Management
Time Management
CPU Isolation
File Systems
Sound
Rendering
M-Access-Control
Networking
Logging
Kexec
Users (as in people)
Crypto API
eBPF
FRESH
ENERGY
+5 Vitamin

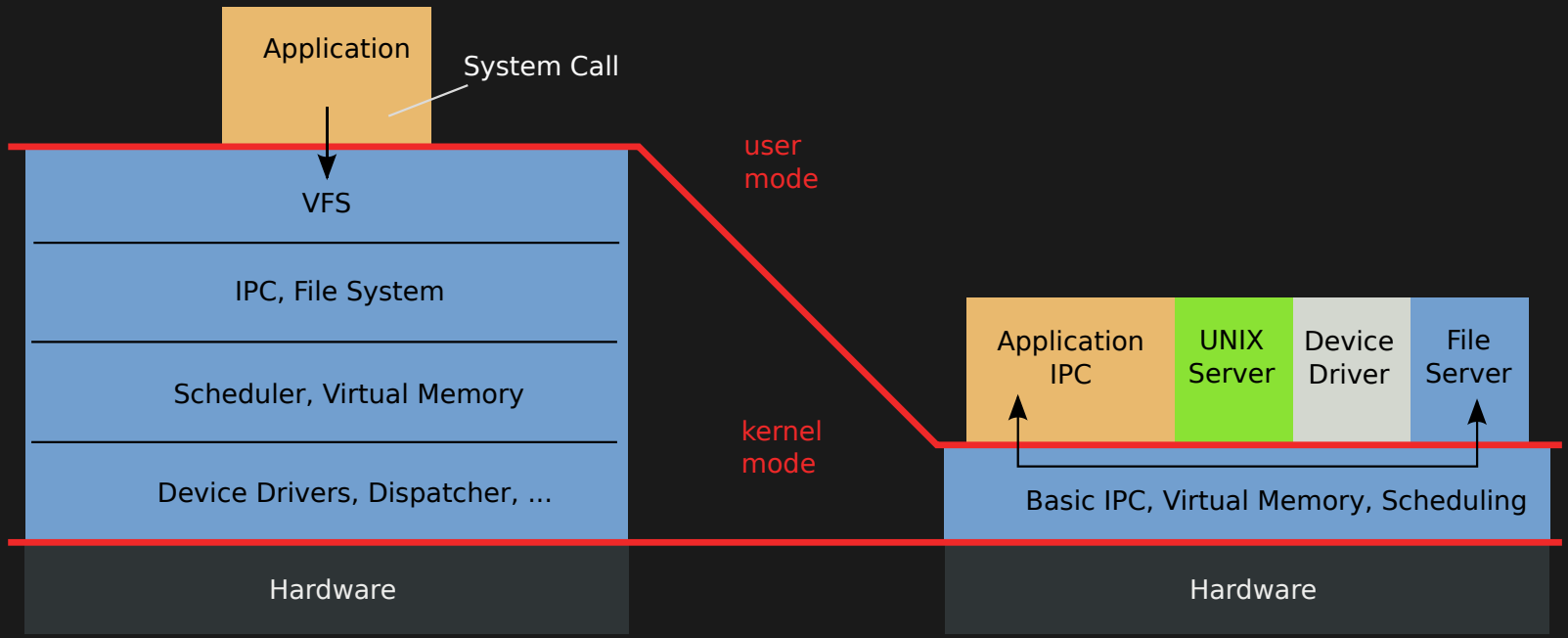
Micro vs. Monolithic

- Principle of Least Privilege
- Principle of Least Scope

Ein Kernel **sollte** irgendwann *fertig* sein!

Monolithic Kernel
based Operating System

Microkernel
based Operating System



Delegation!

Syscall Security

Permissions?

→ Reicht nicht!

→ Mandatory-Access-Control auch nicht! (AppArmor, seccomp, pledge)

→ DenyListing: Alles doof...

Capability Token, das Resource identifiziert
und Zugriff erlaubt[1]

[1] Think: File Descriptor

Everything is a File

Everything is a Capability

Intro to Kernel Development

Was ist nochmal ein Computer?

+ CPU
+ RAM
+ Diverse Geräte
+ ROM

Bootstrapping

1. Gerät lädt Firmware vom ROM
2. Firmware lädt Bootloader
3. Bootloader lädt Kernel
4. Kernel lädt OS

- **Ein** Computer vs **mehrere** Programme
- **Ein** Computer vs **mehrere** Benutzer

Der Kernel macht aus einem physikalischem Computer mehrere logische Computer.

Isolation

Multiplexing!

- Rechenzeit
- Prozessorzustand
- Speicherplatz
- Gerätezugriff

Hardwareunterstützung

- Privilegierter CPU-Modus
- Speichersegmentierung
- Virtuelle Addressierung
- Preemption
- Timer

Kernel

- Gott-Zugriff
- konfiguriert Speicherisolierung
- wechselt regelmäßig Prozesse
- Syscalls

not_kernel.rs

```
import std;
malloc()
println!("Hi")
let x = new MyObject()
let y = List::new()
```

kernel.rs

```
# boot
todo!()

while true:
    todo!()
```

Time-Management

kernel.rs

```
# boot
todo!()

while True:
    proc = schedule()
    exec_user_proc(proc)
```

RISC-V CPU State

Regs 32x 64bit Register

Float Regs 32x 64bit IEEE Floating Point

PC Aktuelle Stelle im Code

Switch to Userspace

```
func exec_user_proc(proc):
    cpu.regs = proc.load_state()
    cpu.pc = proc.pc
    cpu.mode = "user"
    unreachable!()
```

Problem: Gierige Prozesse!

$\rightarrow$ Regelmäßige Unterbrechungen!

$\rightarrow$ Interrupts!

RISC-V Privileged Registers

ie Interrupt Enabled

Timer, External, Software

ip Interrupt Pending

tvec Interrupt Handler Address

```
# boot
todo!()

while True:
    proc = schedule()
    setup_timer(5ms)                    <-- new
    interrupt = exec_user_proc(proc)
    handle_interrupt(interrupt)  <-- new
```


HEY RUST BASED KERNEL WHY DO YOU ALWAYS WEAR THAT MASK?
RUST BASED KERNEL
ASSEMBLY TRAP HANDLER
LET'S KEEP THIS ON.

RISC-V Privileged Registers. Cont

scratch Scratch

epc Exception Program Counter

cause Trap Cause

```
# boot
setup_trap_handler()          <— new

while True:
  proc = schedule()
  setup_timer(5ms)
  exec_user_proc(proc)
  label!("loop_resume")
```

```
#[asm]
func trap_handler():
    proc = cpu.scratch
    proc.save_state(cpu.regs)
    proc.pc = cpu.epc
    goto!("loop_resume")
```

Multiplexing!

- Rechenzeit
- Prozessorzustand
- Speicherplatz
- Gerätezugriff

Memory-Management

Segmentation (-fault)

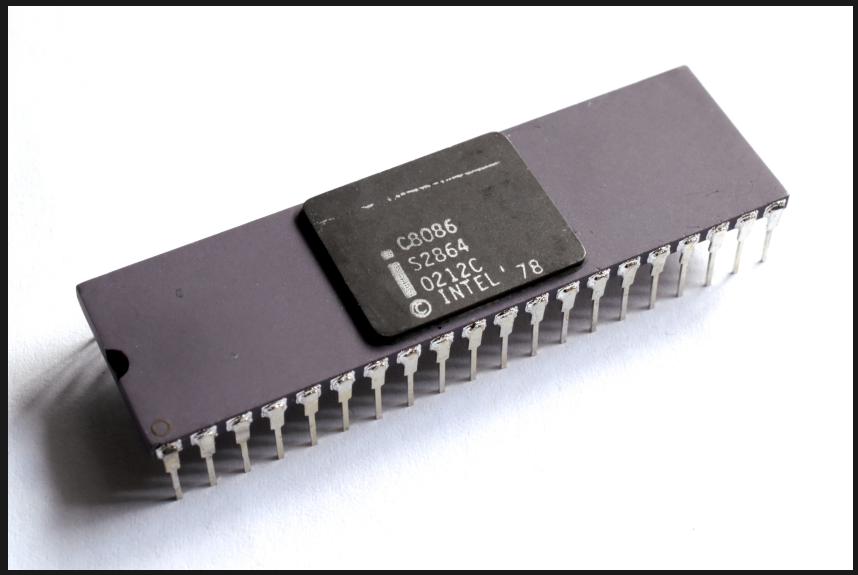


Abbildung 1: Intel iAPX 86

+ Introduces Memory Access Rights
- Fixed number of memory segments
- Fragmentation

RISC-V Privileged Registers. Cont

atp Address Translation and Protection

status Verschiedene Status Bits

Address Translation

Sv39 page table entry.

RISC-V Virtual Addressing

Sv39 virtual address.

Sv39 physical address.

```
func exec_user_proc(proc):
  cpu.regs = proc.load_state()
  cpu.atp = proc.pagetable        <-- new
  cpu.pc = proc.pc
  cpu.mode = "user"
  unreachable!()
```

Multiplexing!

- Rechenzeit
- Prozessorzustand
- Speicherplatz
- Gerätezugriff

Devices

Geräte sind memory-mapped

$\rightarrow$ Memory können wir schon!

Multiplexing!

- Rechenzeit
- Prozessorzustand
- Speicherplatz
- Gerätezugriff

Kernel fertig?