topic: OS organization

Isolation
Kernel/user mode
System call

OS

HW

xv6
Strawman design: no OS

Enforced multiplexing

No strong isolation
Unix interface

abstract the HW resources
processes instead CPU
exec instead of memory
files instead of disk block
OS should be defensive

App cannot crash the OS
App cannot break out of its isolation

Strong isolation between apps + OS

Typical: HW support @ virtual memory
User/kernel mode

Set up page table

Disabling clock interrupts

Privileged instructions

Unprivileged instructions

Add branch

Jr
Cpus provide virtual memory

page table: virtual addr -> physical.

process has its own page table

memory isolation
Entering kernel

ecall

for i < U

ecall syscall

for i < K

syscall

write

write
Kernel = trusted computing base (TCB)
- Kernel must have no bugs
- Kernel must treat processes as malicious

→ Security
monolithic kernel design

OS layers → tight integration → performance
Kernel design:
- echo
- sh
- msg

FS

VM

Kernel is small → fewer bugs

Performance
Kernel
PROC.C \rightarrow gcc \rightarrow PAX.S \rightarrow ASSEMBLER
\rightarrow PROC.O
\rightarrow d \rightarrow KERNEl
PIPE.C \rightarrow GCC \rightarrow PIPE.S \rightarrow AS \rightarrow PIPE O
\rightarrow KERNEl}\asm
Figure 1: FU540-C000 top-level block diagram.
```c
for (int i = 0; i < n; i++) {
    Read instruction
    Deco;le instruction
    Execute instruction: `add a0, 7`
}
```