Systems 3

OS Structure

Marcel Waldvogel

(Handout)

Department of Computer and Information Science
University of Konstanz

Winter 2019/2020

Photo by Kelly Sikkema on Unsplash

Chapter Goals

- How does the OS (kernel) differ from user space?
- How to switch between privileged/unprivileged mode?
- How to interface with the kernel?
- What to put into the kernel?

OS as Black Box

- System calls as API
- Scheduling
- Memory Management
- Resource Management

Central Processing Unit (simplified)

What is a CPU and how does it work?

- Arithmetic Logic Unit (ALU)
- Processor registers
- Control Unit

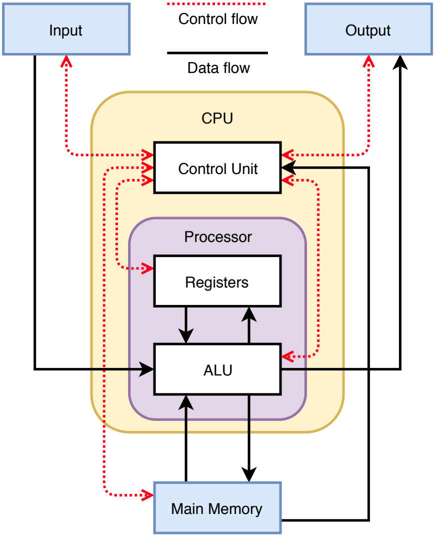


Figure: Lambtron (CC BY-SA 4.0) modified by Klaus Herberth

Instruction Cycle (simplified)

The classic RISC pipeline consists of the following operations:

- Fetch
- Decode
- Execute

Question: How does the CPU find the next instruction?

Solution: Program counter

Heads up

In a few weeks, we will look into abusing the instruction cycle.

Instructions

r3=r2+r1	ADD r3, r2, r1
add(a, b)	PUSH a PUSH b CALL add ADD sp, sizeof(a)+sizeof(b)
while (c) x();	loop: CALL x TEST c BNE label

Permissions

Problem

If every program can execute every instruction, how can we prevent processes interfering with each other?

Solution

Have some privileged instructions, which only the kernel can execute.

Which ones?

1. Access to/manipulating hardware
2. Manipulating CPU state (outside of ‘user-visible’ registers), e.g., modify memory mapping, privilege level
3. Access memory not assigned to current process (kernel data structures, memory of other processes)

Privileges

Definitions

User level Access rights of a user process (no access to privileged instructions)

Kernel level Access rights of the kernel (access to privileged instructions)

User space Memory visible to user process

Kernel space Memory visible to kernel

Space/level distinction not always strict.

Switching between User and Kernel

Switching steps

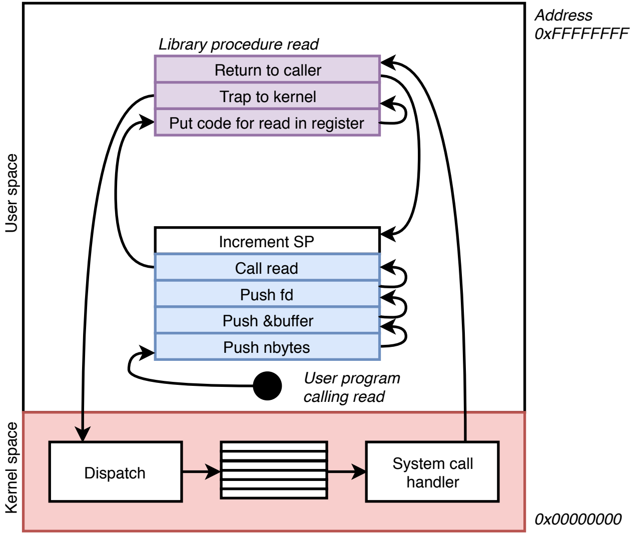
1. Address space
2. Privilege mode (bit in processor status register)
3. Program counter (to a kernel-defined address only)

Requesting a switch

- Interrupt
- Trap (illegal instruction, illegal access, syscall, ...)

Return under kernel control

Making a System Call



What should be part of the Kernel?

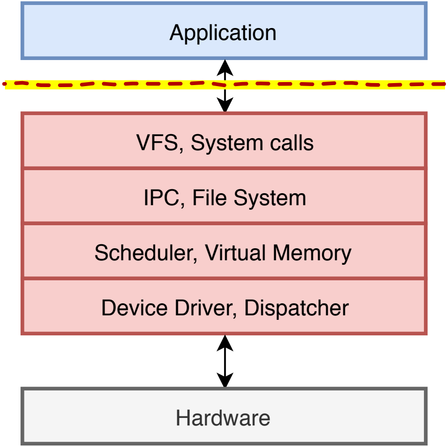
Criteria for inclusion

1. Direct hardware manipulation (I/O, CPU configuration)
2. Direct triggering by interrupt
3. Communication between processes

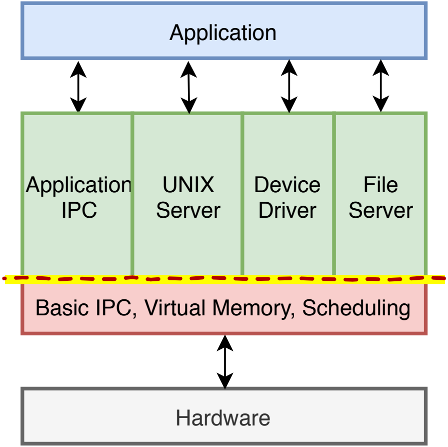
Criteria for exclusion

1. Complex operation (error-prone)
2. Debuggable
3. Updateable
4. Restartable

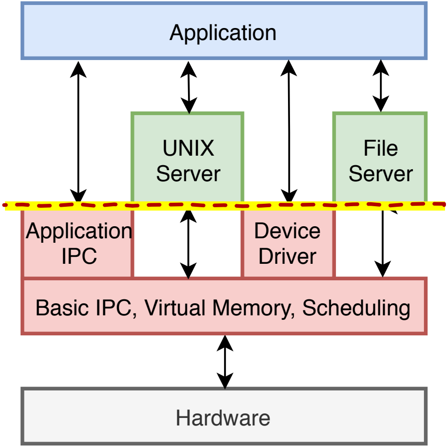
Monolithic Kernel



Microkernel

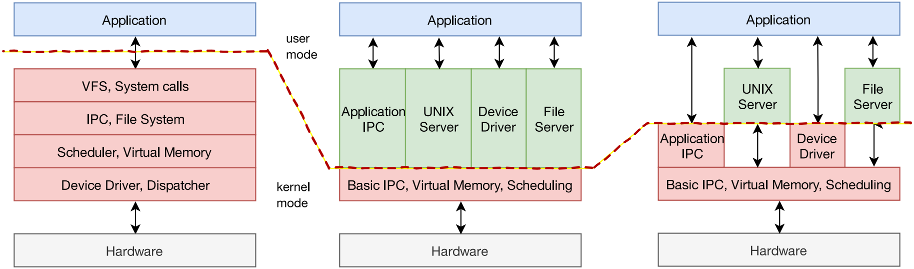


Hybrid kernels



Comparison

Criteria	Microkernel	Monolithic
Size	small	large
Speed	slower	fast
Extensible	yes	no
Reliability	high	lower[1]
Interface	messages	functions



Adapted from https://www.geeksforgeeks.org/monolithic-kernel-and-key-differences-from-microkernel/ and https://techdifferences.com/difference-between-microkernel-and-monolithic-kernel.html.

[1] Crash of a service leads to system crash