Operating Systems (A) (Honor Track)

Lecture 22: OS Design Issues

Yao Guo (郭耀)

Peking University

Fall 2021

Acknowledgements: Prof. Xiangqun Chen & Tao Wang at PKU and Prof. Yuanyuan Zhou at UCSD

This Lecture

OS Design Issues

Design Goals

- The designers must have a clear idea of what they want.

- For general-purpose operating systems:
  - Define abstractions
  - Provide primitive operations
  - Ensure isolation
  - Manage the hardware

Why Is It Hard to Design an OS?

1. OSes have become extremely large programs
2. OSes have to deal with concurrency
3. OSes have to deal with potentially hostile users
4. Many users do want to share some of their information and resources with selected other users
5. OSes live for a very long time
6. OS designers really do not have a good idea of how their systems will be used (generality)
7. Modern OSes are generally designed to be portable
8. OSes need to be backward compatible with some previous OS

Interface Design

- Interface: abstractions
  - E.g., system calls, POSIX, device drivers
- Guiding Principles
  - Principle 1: Simplicity
  - Principle 2: Completeness
  - Principle 3: Efficiency

Interface Design: Simplicity

- *Perfection is reached not when there is no longer anything to add, but when there is no longer anything to take away.*
  - Antoine de St. Exupery, French writer

- Less is better than more, at least in the operating system itself.

- The KISS principle: Keep It Simple, Stupid

Interface Design: Completeness

- *Everything should be as simple as possible, but no simpler.*
  - Albert Einstein
- *First, it is important to emphasize the value of simplicity and elegance, for complexity has a way of compounding difficulties and as we have seen, creating mistakes.* ***My definition of elegance is the achievement of a given functionality with a minimum of mechanism and a maximum of clarity.***
  - *Fernando Corbato, 1991 Turing Award winner*
- The key idea here is *minimum of mechanism*.

Interface Design: Efficiency

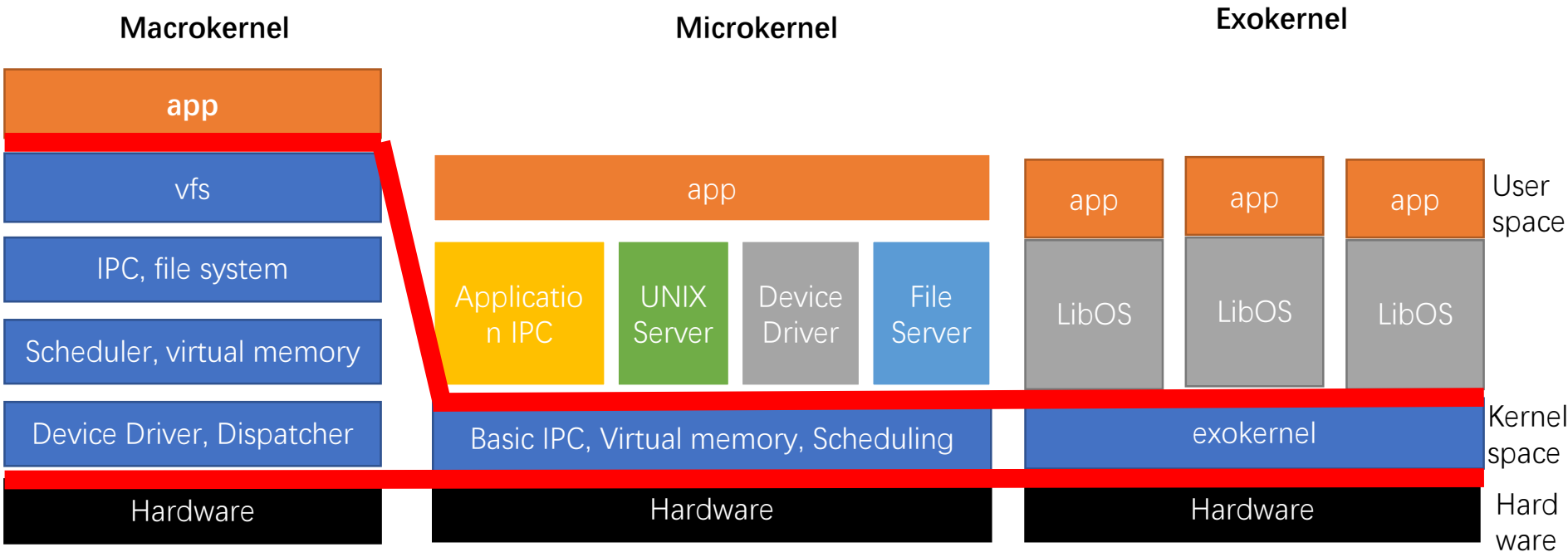
- Efficiency of implementation

- If a feature or system call cannot be implemented efficiently, it is probably not worth having

- It should also be intuitively obvious to the programmer about how much a system call costs.

System Structure

- OSes are typically using a layered structure
- Different designs are used in terms of how many functionalities are put in the kernel.



Performance

- Space-Time Trade-offs
- Caching
  - Buffering
- Exploiting Locality
- Optimize the Common Case

Project Management

- Programmers are perpetual optimists.
- *The Mythical Man Month*, Fred Brooks (Turing Award winner)
  - Adding manpower to a late software project makes it later.
  - No Silver Bullet
- Team Structure

Title	Duties
Chief programmer	Performs the architectural design and writes the code
Copilot	Helps the chief programmer and serves as a sounding board
Administrator	Manages the people, budget, space, equipment, reporting, etc.
Editor	Edits the documentation, which must be written by the chief programmer
Secretaries	The administrator and editor each need a secretary
Program clerk	Maintains the code and documentation archives
Toolsmith	Provides any tools the chief programmer needs
Tester	Tests the chief programmer's code
Language lawyer	Part timer who can advise the chief programmer on the language

Figure 12-10. Mills' proposal for populating a 10-person chief programmer team.

Trends in OS Design

- Virtualization and the Cloud
- Manycore Chips
- Large-Address-Space Operating Systems
- Seamless Data Access
- Battery-Powered Computers
- Embedded Systems

This Lecture

OS Security

The Secure Environment

- Goals and threats

Goal	Threat
Confidentiality	Exposure of data
Integrity	Tampering with data
Availability	Denial of service

- Attackers, intruders, or adversaries
  - Theft, hacktivism, vandalism, terrorism, cyberwarfare, espionage, spam, extortion, fraud, etc.

OS Security

- Can we build secure systems?
  - ‘‘In theory, yes.’’
  - In principle, software can be free of bugs and we can even verify that it is secure—as long as that software is not too large or complicated.
- Why is it not done?
  - The only known way to build a secure system is to keep it simple.
  - Features are the enemy of security.

Trusted Computing Base

- Trusted systems: systems that have formally stated security requirements and meet these requirements.
- Key component: a minimal TCB (Trusted Computing Base) consisting of the hardware and software necessary for enforcing all the security rules.
- The TCB consists of
  - most of the hardware (except I/O devices that do not affect security)
  - a portion of the operating system kernel, and
  - most or all of the user programs that have superuser power (e.g., SETUID root programs in UNIX)
- An important part of the TCB is the reference monitor
  - monitors all system calls involving security

Access Control

- Access Control Mechanisms
  - Access Control List (ACL)
  - Capabilities

- Different Access Control Models
  - MAC (Mandatory Access Control)
  - RBAC (Role-Based Access Control)
  - ABAC (Attribute-Based Access Control)

Cryptography

- Secret-Key Cryptography
- Public-Key Cryptography
  - Encrypt with Public key, decrypt with private key
  - Encrypt with private key, decrypt with public key
- RSA: the most popular public-key system
  - Math background: multiplying really big numbers is much easier for a computer to do than factoring really big numbers, especially when all arithmetic is done using modulo arithmetic and all the numbers involved have hundreds of digits

Exploiting Methods

- Buffer Overflow Attacks
- Format String Attacks
- Dangling Pointers
- Null Pointer Dereference Attacks
- Integer Overflow Attacks
- Command Injection Attacks
- TOCTOU (Time of Check to Time of Use)

Malware

- Trojan horses
- Viruses
- Worms
- Spyware
- Rootkits

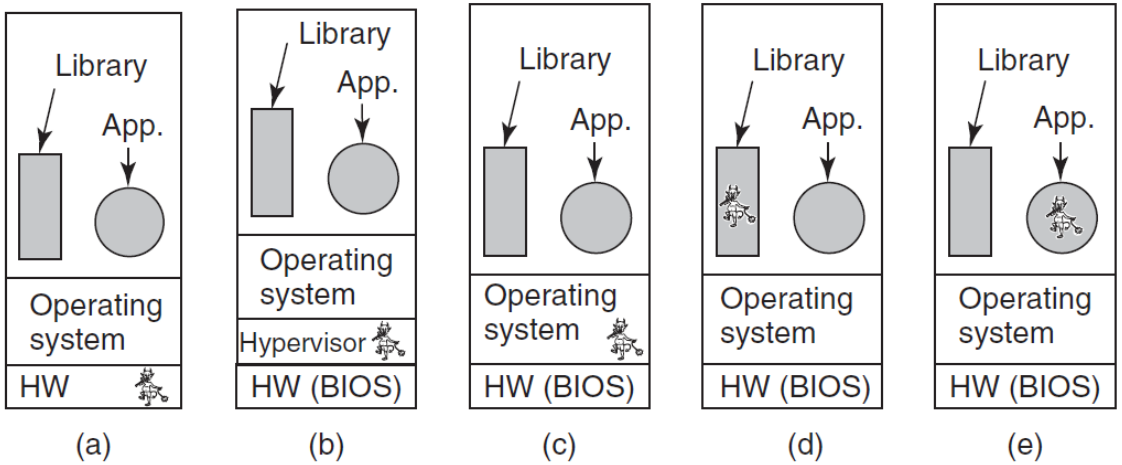


Figure 9-31. Five places a rootkit can hide.

Defense

- Firewalls
- Anti-virus
- Code signing
- Model-based intrusion detection
- Encapsulating Mobile Code

Research on Security

- Current trends
  - Cloud security
  - Mobile security
  - Blockchain-related
  - IoT security
- Four top conferences
  - S&P (Oakland)
  - CCS
  - Usenix Security
  - NDSS

Summary

- OS design issues
- OS Security

- Next time
  - Lab quiz
  - Course Review