Karlsruhe Institute of Technology

Operating Systems

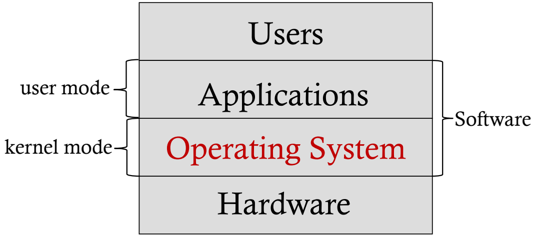
01. Introduction

Prof. Dr. Frank Bellosa | WT 2020/2021

What is an Operating System

Application programmers don't understand the hardware in detail

- Add **Operating System (OS)** layer between application and hardware to make system **easy to use**



- OS takes a **physical** resource (e.g., processor, DRAM memory, hard drive) and transforms it in a **virtual** form of itself (**virtualization**)

Resource Manager

- Hides hardware details behind interfaces (system calls)
  → acts as a standard library
- Provides **abstractions** for each resource
  - CPU: processes and / or threads
  - Memory: address space
  - Disk: files
- Makes hardware use **efficient** for applications (cost, time,energy)
- Provides **protection**
  - Prohibit processes changing/read the state of others (isolation)
    e.g., writing into other processes memory
  - Prevent exploitation of OS services

Sharing in Time & Space

- Multiplexes hardware to multiple running applications
  - Allows and controls shared use of a resource (e.g., CPU, volatile memory, persistent storage)
  - Fair assignment of resources (accounting & allocation)
- Controls the execution of applications (**control program**)
- Prevents improper use of the computer

Challenges

- What are the correct **abstractions**?
  How much of hardware should be exposed?
- What are the correct **mechanisms**?

  Mechanism: Implementation of what is done
  (e.g., the commands to switch off the back light of a display)

- What are the correct **policies**?

  Policy: The rules which decide when & what is done where & how fast
  (e.g., how often, how many resources are used, . . .)

Mechanisms can be reused even when the policy changes

Operating Systems: Piece I

Virtualization

- Make each application believe it has each resource to itself

  Examples:
  - CPU
    - Abstractions: process & thread
    - Policies: scheduling
  - Memory
    - Abstractions: virtual address spaces
    - Policies: paging

Operating Systems: Piece II

Concurrency

Events are occurring simultaneously and may interact with each other

- Hide concurrency of independent processes
- Manage concurrency with interacting processes
  - Provide abstractions (e.g., locks, condition variables, critical sections)
  - Ensure processes do not deadlock

Operating Systems: Piece III

Persistence

- Access information permanently
  - Lifetime of information exceeds lifetime of any one process
  - Machine may be rebooted
  - Machine may lose power or crash unexpectedly due to inconsistencies

- Provide abstraction so applications do not know how data is stored
  → e.g., logical view in form of files, directories, links (**file system**)
  - **Drivers** hide peculiarities of specific hardware devices
  - General interface abstracts physical properties to logical units
    (e.g. drive + CHS → logical block)

Operating Systems: Piece III

Persistence

- Correctness despite unexpected failures (transient, permanent)
- Performance: hide latency and limited bandwidth of storage devices (hard drive, solid sate disk)
  - **Buffering**: Store data temporarily in faster storage before it is being transferred
  - **Caching:** Store parts of data in faster storage for performance (for reuse)

Operating Systems: Advanced Topics

- Multiprocessors
- Persistent main memory (e.g., PC-RAM, STT-RAM)
- Integration of accelerators (GPU, FPGA, ...)
- Virtual machines
- Power management
- System security
- Networked & distributed systems
- Application support (graph processing, machine learning, ...)

Why Study Operating Systems?

- Build, modify, or administer an operating system or VMM (e.g., AUTOSAR, XX.OS, XEN, Firecracker VMM, Linux, Windows)
- Understand system performance
  - Tune workload performance
  - Behavior of OS impacts entire machine
  - Apply knowledge across many layers
    - Computer architecture
    - Programming languages
    - Data structures and algorithms
    - Software engineering
    - AI and ML

Why Study Operating Systems?

- Build, modify, or administer an operating system or VMM (e.g., AUTOSAR, XX.OS, XEN, Firecracker VMM, Linux, Windows)
- Understand system performance
  - Tune workload performance
  - Behavior of OS impacts entire machine
  - Apply knowledge across many layers
    - Computer architecture
    - Programming languages
    - Data structures and algorithms
    - Software engineering
    - AI and ML

Why Study Operating Systems?

- Build, modify, or administer an operating system or VMM (e.g., AUTOSAR, XX.OS, XEN, Firecracker VMM, Linux, Windows)
- Understand system performance
  - Tune workload performance
  - Behavior of OS impacts entire machine
  - Apply knowledge across many layers
    - Computer architecture
    - Programming languages
    - Data structures and algorithms
    - Software engineering
    - AI and ML

Why Study Operating Systems?

- Build, modify, or administer an operating system or VMM (e.g., AUTOSAR, XX.OS, XEN, Firecracker VMM, Linux, Windows)
- Understand system performance
  - Tune workload performance
  - Behavior of OS impacts entire machine
  - Apply knowledge across many layers
    - Computer architecture
    - Programming languages
    - Data structures and algorithms
    - Software engineering
    - AI and ML

Why Study Operating Systems?

- Build, modify, or administer an operating system or VMM (e.g., AUTOSAR, XX.OS, XEN, Firecracker VMM, Linux, Windows)
- Understand system performance
  - Tune workload performance
  - Behavior of OS impacts entire machine
  - Apply knowledge across many layers
    - Computer architecture
    - Programming languages
    - Data structures and algorithms
    - Software engineering
    - AI and ML

Why Study Operating Systems?

- Build, modify, or administer an operating system or VMM (e.g., AUTOSAR, XX.OS, XEN, Firecracker VMM, Linux, Windows)
- Understand system performance
  - Tune workload performance
  - Behavior of OS impacts entire machine
  - Apply knowledge across many layers
    - Computer architecture
    - Programming languages
    - Data structures and algorithms
    - Software engineering
    - AI and ML

Why Study Operating Systems?

- Build, modify, or administer an operating system or VMM (e.g., AUTOSAR, XX.OS, XEN, Firecracker VMM, Linux, Windows)
- Understand system performance
  - Tune workload performance
  - Behavior of OS impacts entire machine
  - Apply knowledge across many layers
    - Computer architecture
    - Programming languages
    - Data structures and algorithms
    - Software engineering
    - AI and ML