東南大學
SOUTHEAST UNIVERSITY

OPERATING SYSTEM CONCEPTS

Chapter 1. Introduction

A/Prof. Kai Dong

Contents

1. What Operating Systems Do
2. Computer-System Organization
3. Computer-System Architecture
4. Operating-System Structure
5. Operating-System Operations
6. Operating-System Functions
7. Kernel Data Structures
8. Open-Source Operating Systems

Warm-up

Discussion

- What Operating System(s) are you familiar with?
- What is the best Operating System in your opinion? And for what features?

Objectives

- To describe the basic organization of computer systems.
- To explain the evolution of operating system structures.
- To provide a grand tour of the major components of operating systems.
- To give an overview of the many types of computing environments.
- To explore several open-source operating systems.

Contents

1. What Operating Systems Do
2. Computer-System Organization
3. Computer-System Architecture
4. Operating-System Structure
5. Operating-System Operations
6. Operating-System Functions
7. Kernel Data Structures
8. Open-Source Operating Systems

What Operating Systems Do

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  - Execute user programs and make solving user problems easier.
  - Make the computer system convenient to use.
  - Use the computer hardware in an efficient manner.

What Operating Systems Do

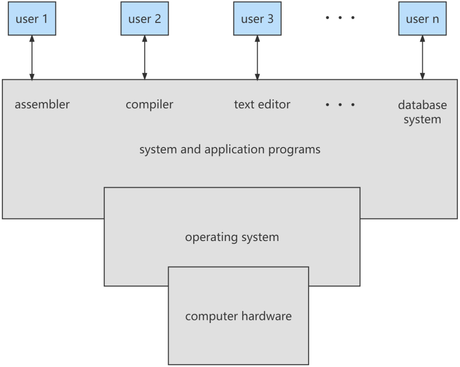
Computer System Structure

Computer system can be divided into four components:

- **Hardware** — provides basic computing resources.
  - CPU, memory, I/O devices
- **Operating system** — controls and coordinates use of hardware among various applications and users
- **Application programs** — define the ways in which the system resources are used to solve the computing problems of the users.
  - Word processors, compilers, web browsers, database systems, video games
- **Users**
  - People, machines, other computers

What Operating Systems Do

Computer System Structure (contd.)



- We can explore operating systems from two viewpoints: that of the user and that of the system.

What Operating Systems Do

User View

- Want convenience, ease of use and good performance.
- Don't care about resource utilization.
  - But *shared computer* such as *mainframe* or *minicomputer* must keep all users happy.
  - Users of dedicate systems such as *workstations* have dedicated resources but frequently use shared resources from *servers*.
  - Handheld computers are resource poor, optimized for usability and battery life.
  - Some computers have little or no user interface, such as embedded computers in devices and automobiles.

What Operating Systems Do

System View

- OS is a resource allocator.
  - Manages all resources.
  - Decides between conflicting requests for efficient and fair resource use.
- OS is a control program.
  - Controls execution of programs to prevent errors and improper use of the computer.

What Operating Systems Do

Defining Operating Systems

- No universally accepted definition.
- "Everything a vendor ships when you order an operating system"
  - is a good approximation,
  - but varies wildly.
- "The one program running at all times on the computer"
  - defines the kernel,
  - everything else is either
    - a system program (ships with the operating system) , or
    - an application program.

Contents

1. What Operating Systems Do
2. Computer-System Organization
3. Computer-System Architecture
4. Operating-System Structure
5. Operating-System Operations
6. Operating-System Functions
7. Kernel Data Structures
8. Open-Source Operating Systems

Computer-System Organization

Computer Startup

- Bootstrap program is loaded at power-up or reboot.
  - Typically stored in ROM or EPROM, generally known as firmware.
  - Initializes all aspects of system.
  - Loads operating system kernel and starts execution.
- Think of:
  - What happens when a program runs / How instructions are executed?
  - Millions/billions of times a second,
    - the processor fetches an instruction from memory,
    - decodes it,
    - executes it,
    - moves on to the next till end.

Computer-System Organization

Von Neumann Model

- We have just described the basics of the Von Neumann model of computing.
- "⋯ an electronic digital computer with parts consisting of
  - a processing unit containing an arithmetic logic unit and processor registers;
  - a control unit containing an instruction register and program counter;
  - a memory to store both data and instructions;
  - external mass storage; and
  - input and output mechanisms."

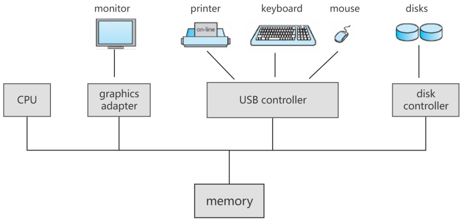
  — Wikipedia "Von Neumann architecture".

Computer-System Organization

Von Neumann Model (contd.)

The above Von Neumann Model defines/describes:

- How is a computer-system organized
  - One or more CPUs, device controllers connect through common bus providing access to shared memory.
- How is a computer-system operating
  - Concurrent execution of CPUs and devices competing for memory cycles.



Computer-System Organization

Computer-System Operation

- I/O devices and the CPU can execute concurrently.
- But how?
  - Each device controller is in charge of a particular device type.
  - Each device controller has a local buffer.
  - CPU moves data from/to main memory to/from local buffers.
  - I/O is from the device to local buffer of controller.
- By now communication is enabled, but how to cooperate?
  - Device controller informs CPU that it has finished its operation by causing an **interrupt**.

Computer-System Organization

Common Functions of Interrupts

- An interrupt is an input signal to the processor indicating an event that needs immediate attention.
- Interrupt transfers **control** to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction.
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request.
- An operating system is **interrupt driven**.

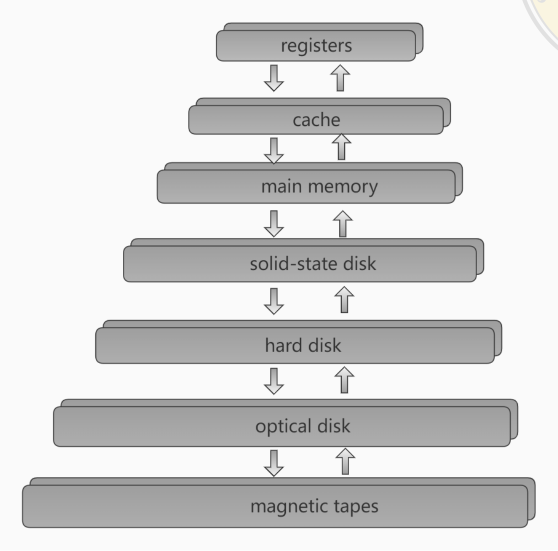
Computer-System Organization

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
  - polling, or
  - vectored interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt.

Computer-System Organization

Storage-Device Hierarchy



Computer-System Organization

Storage Structure

- Main memory — only large storage media that the CPU can access directly.
  - Random access.
  - Typically volatile.
- Secondary storage — extension of main memory that provides large nonvolatile storage capacity.
- Hard disks — rigid metal or glass platters covered with magnetic recording material.
  - Disk surface is logically divided into tracks, which are subdivided into sectors.
  - The disk controller determines the logical interaction between the device and the computer.
- Solid-state disks — faster than hard disks, nonvolatile.
  - Various technologies.
  - Becoming more popular.

Computer-System Organization

I/O Structure — No Interrupts

- After I/O starts, control returns to user program only upon I/O completion
  - Wait instruction idles the CPU.
  - Wait loop (contention for memory access).
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing.

Computer-System Organization

I/O Structure — Interrupt Driven

- After I/O starts, control returns to user program without waiting for I/O completion
  - **System call** — request to the OS to allow user to wait for I/O completion.
  - Device-status table contains entry for each I/O device indicating its type, address, and state.
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt.
- Who returns the control?
  - Device Driver for each device controller to manage I/O. Provides uniform interface between controller and kernel.

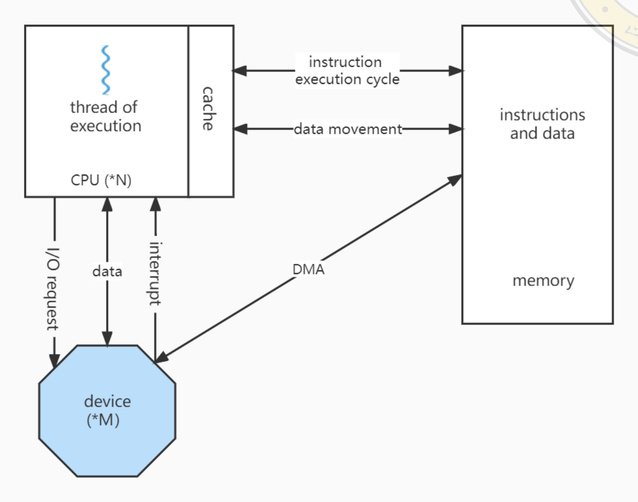
Computer-System Organization

I/O Structure — Direct Memory Access

- **Direct Memory Access** Structure
  - Used for high-speed I/O devices able to transmit information at close to memory speeds.
  - Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
  - Only one interrupt is generated per block, rather than the one interrupt per byte.

Computer-System Organization

How a Modern Computer Works



Contents

1. What Operating Systems Do
2. Computer-System Organization
3. Computer-System Architecture
4. Operating-System Structure
5. Operating-System Operations
6. Operating-System Functions
7. Kernel Data Structures
8. Open-Source Operating Systems

Computer-System Architecture

- Most systems use a single general-purpose processor until 10 years ago.
  - Most systems have special-purpose processors as well.
- Multiprocessor systems growing in use and importance.
  - Also known as parallel systems, tightly-coupled systems.
  - Advantages include:
    1. Increased throughput.
    2. Economy of scale.
    3. Increased reliability — graceful degradation or fault tolerance.
  - Two types:
    1. **Asymmetric Multiprocessing** — each processor is assigned a specific task.
    2. **Symmetric Multiprocessing** — each processor performs all tasks.

Computer-System Architecture

A Dual-Core Design

- Multi-chip and multicore.
  - Why multicore?
  - On-chip communication is faster, uses significantly less power.
- Blade Servers: all in one chassis.
  - Chassis containing multiple separate systems.

Computer-System Architecture

Clustered Systems

- Like multiprocessor systems, but multiple systems working together.
- Loosely-coupled systems.
  - Usually sharing storage via a storage-area network (SAN).
  - Provides a high-availability service which survives failures.
    - Asymmetric clustering has one machine in hot-standby mode.
    - Symmetric clustering has multiple nodes running applications, monitoring each other.
  - Some clusters are for high-performance computing (HPC).
    - Applications must be written to use parallelization.
  - Some have distributed lock manager (DLM) to avoid conflicting operations.

Contents

1. What Operating Systems Do
2. Computer-System Organization
3. Computer-System Architecture
4. Operating-System Structure
5. Operating-System Operations
6. Operating-System Functions
7. Kernel Data Structures
8. Open-Source Operating Systems

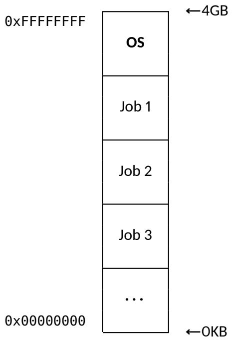
Operating-System Structure

Some History

- No operating system. (1946 $\sim$ 1955, ENIAC)
- A typical simple Batch system (since 1950s)
  - bring cards to IBM 1401
  - IBM 1401 read cards to tape
  - put tape on IBM 7094 which does computing
  - put tape on IBM 1401 which prints output

Operating-System Structure

Some History (contd.)



- **Multiprogramming** (Batch system) needed for efficiency. (since 1960s)
  - Single user cannot keep CPU and I/O devices busy at all times.
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute.
  - A subset of total jobs in system is kept in memory.
  - One job selected and run via **job scheduling** (**long term scheduling**).
  - When it has to wait (for I/O for example), OS switches to another job.

Operating-System Structure

Some History (contd.)

- **Timesharing** (multitasking, since 1960s) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing.
  - **Response time** should be < 1 second.
  - Each user has at least one program executing in memory — **process**.
  - If several jobs ready to run at the same time — **CPU scheduling** (**short term scheduling**).
  - If processes don't fit in memory, **swapping** moves them in and out to run.
  - **Virtual memory** allows execution of processes not completely in memory.

In Class Exercise

Multiprogramming

- Consider a multiprogramming environment, two programs *A* and *B* share the system simultaneously, and run as follows.
  - For every 35 min, A runs on CPU for 15 min, then waits for I/O for 20 min.
  - For every 20 min, B runs on CPU for 10 min, then waits for I/O for 10 min.

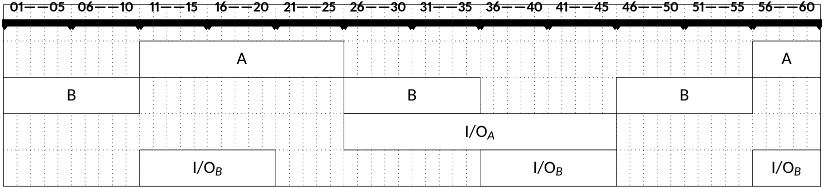
  Suppose *B* runs first, I/O_A and I/O_B are different devices, and the time of switching between *A* and *B* can be ignored. Draw the timeline for these two programs, and calculate CPU utilization within 60 minutes.
- What if I/O_A and I/O_B are the same device?

In Class Exercise

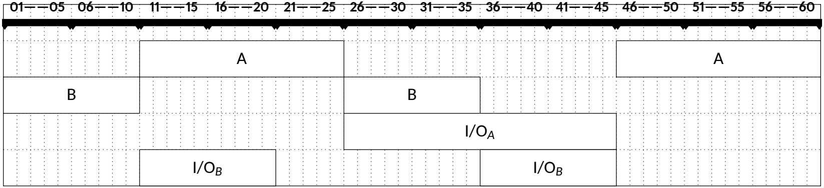
Key to Multiprogramming

Key to Q1: *CPU utilization* $= 50/60 = 83.33\%$

v1:



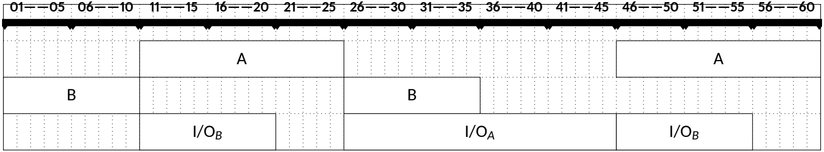
v2:



In Class Exercise

Key to Multiprogramming

Key to Q2: $\textit{CPU utilization} = 50/60 = 83.33\%$



Contents

1. What Operating Systems Do
2. Computer-System Organization
3. Computer-System Architecture
4. Operating-System Structure
5. Operating-System Operations
6. Operating-System Functions
7. Kernel Data Structures
8. Open-Source Operating Systems

Operating-System Operations

Interrupt Driven (hardware and software)

- Hardware interrupt by one of the devices.
- Software interrupt (**exception** or **trap**):
  - Software error (e.g., division by zero).
  - Request for operating system service.
  - Other process problems include infinite loop, processes modifying each other or the operating system.

Operating-System Operations

Dual Mode

- **Dual-mode** operation allows OS to protect itself and other system components.
  - **User mode** and **kernel mode**.
  - Mode bit provided by hardware.
    - Provides ability to distinguish when system is running user code or kernel code.
    - Some instructions designated as **privileged**, only executable in kernel mode.
    - **System call** changes mode to kernel, return from call resets it to user.
- Increasingly CPUs support multi-mode operations.
  - i.e., virtual machine manager (VMM) mode for guest VMs.

Operating-System Operations

Timer

- **Timer** to prevent infinite loop / process hogging resources.
  - Timer is set to interrupt the computer after some time period.
  - Keep a counter that is decremented by the physical clock.
  - Operating system set the counter (privileged instruction).
  - When counter zero generate an interrupt.
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time.

Operating-System Operations

What Happens When the Computer System is Booting?

OS @boot (kernel mode)	**Hardware**
initialize trap table	
	remember addresses of ...
	system call handler
	timer handler
	illegal instruction handler
start interrupt timer	
	start timer; interrupt after *X* ms

Operating-System Operations

What Happens When the Computer System is Running?

OS @run (kernel mode)	**Hardware**	**Program (user mode)**
to start process *A*:		
return-from-trap (into *A*)		
	move to *user mode*	
		process *A* runs:
		fetch instruction
		execute instruction
		...
	timer interrupt	
	move to *kernel mode*	
	jump to interrupt handler	

Contents

1. What Operating Systems Do
2. Computer-System Organization
3. Computer-System Architecture
4. Operating-System Structure
5. Operating-System Operations
6. Operating-System Functions
7. Kernel Data Structures
8. Open-Source Operating Systems

Operating-System Functions

- User interface, in Chapter 2
- CPU management (or Process management), in Chapter 3 ~ 7
  - Interrupt handling and management
  - CPU scheduling
  - Context switch (save & restore)
- Memory management, in Chapter 8 ~ 9
  - Physical memory allocation and deallocation
  - Address translation (logical $\Longleftrightarrow$ physical)
  - Memory virtualization
- Storage management, in Chapter 10 ~ 12
  - Storage space allocation
  - File & directory management
- Device management, in Chapter 13
- Protection and security, in Chapter 14 ~ 15

Process/Memory/Storage Management

Too Many Terms?

- Just remember three key ideas for now:
  - **Virtualization**: The OS takes a physical resource (such as the processor, or memory, or a disk) and transforms it into a more general, powerful, and easy-to-use virtual form of itself.
  - **Concurrency**: Many problems arise, and must be addressed, when working on many things at once (i.e., concurrently) in the same program.
  - **Persistence**: The OS makes information persist, despite computer crashes, disk failures, or power outages.

Contents

1. What Operating Systems Do
2. Computer-System Organization
3. Computer-System Architecture
4. Operating-System Structure
5. Operating-System Operations
6. Operating-System Functions
7. Kernel Data Structures
8. Open-Source Operating Systems

Kernel Data Structures

You Should Have Known the Concepts of ...

- arrays, lists, stacks, and queues,
- trees,
- hash functions and maps,
- bitmaps.

Contents

1. What Operating Systems Do
2. Computer-System Organization
3. Computer-System Architecture
4. Operating-System Structure
5. Operating-System Operations
6. Operating-System Functions
7. Kernel Data Structures
8. Open-Source Operating Systems

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary closed-source.
- Counter to the copy protection and Digital Rights Management (DRM) movement.
- Started by Free Software Foundation (FSF), which has “copyleft” GNU Public License (GPL).
- Examples include GNU/Linux and BSD UNIX (including core of Mac OS X), and many more.
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - available @ `http://www.virtualbox.com`) .
  - Use to run guest operating systems for exploration.

openEuler Operating System

- Xv6
  - Xv6 documentation (PDF): `link`
  - Xv6 codes (PDF): `link`
  - Xv6 structure (mind): `link`
- EulerOS
  - A commercial Linux distribution developed by Huawei based on CentOS source code for enterprise applications.
  - Available @ `https://openeuler.org/en/`