Chapter 1. Introduction

- What Operating Systems Do
- Computer-System Organization
- Operating-System Structure
- Operating-System Operations
Objectives

- To provide a grand tour of the major operating systems components
- To provide coverage of basic computer system organization

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
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

Four Components of a Computer System
To understand more fully the operating systems role, we next explore operating systems from two viewpoints:

Operating system's viewpoints

User view

The user's view of the computer varies according to the interface being used

System view

From the computer's point of view, the operating system is the program most intimately involved with the hardware

User view

Users set in front of a PC:
• want convenience, ease of use,
• Don’t care about resource utilization

operating system is designed mostly for with some attention paid to performance and none paid to resource utilization

User sits at a terminal connected to a mainframe or minicomputers:
• share resources and may exchange information

operating system is designed to maximize resource utilization, and keep all users happy
Users set at workstations connected to networks of other workstations and servers:

- have dedicated **resources** at their disposal
- share resources such as **networking** and **servers-file**, compute, and print servers

operating system is designed to compromise between **individual usability** and **resource utilization**

User of Handheld computers:

- handheld computers are resource poor

operating system is designed for **individual usability**, but performance per unit of **battery life** is important

Some computers have little or no user interface, such as embedded computers in devices and automobiles

Operating systems are designed primarily to run without user intervention
What Operating Systems Do

In this context, we can view an operating system as a (Operating System Definition):

<table>
<thead>
<tr>
<th>Resource allocator:</th>
<th>Control program:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on.</td>
<td>- It manages the execution of user programs to prevent errors and improper use of the computer.</td>
</tr>
<tr>
<td>- The operating system acts as the manager of these resources</td>
<td>- It is especially concerned with the operation and control of I/O devices.</td>
</tr>
<tr>
<td>- Decides between conflicting requests for efficient and fair resource use</td>
<td></td>
</tr>
</tbody>
</table>

Operating System Definition

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly (vary greatly across systems)
    - Some systems take up less than 1 megabyte of space and lack even a full-screen editor,
    - whereas others require gigabytes of space and are entirely based on graphical windowing systems
- "The one program running at all times on the computer" is the kernel. Everything else is either a system program (ships with the operating system) or an application program (all programs not associated with the operation of the system)
Computer Startup

- When the computer is **powered up** or **rebooted**, it needs to have an initial program (**bootstrap program**) to run.
- **Bootstrap program** is loaded at power-up or reboot.
- Typically stored in **ROM** (read-only memory) or **EEPROM** (electrically erasable programmable read-only memory), generally known as **firmware**.
  - Initializes all aspects of system (from CPU registers to device controllers to memory contents).
  - Loads operating system kernel and starts execution.
- OS then starts executing the first process, and waits for some event to occur.
- Event is usually signaled by an **interrupt** from either the hardware or the software.
  - Hardware may trigger an interrupt at any time by sending a **signal** to the CPU, usually by way of the system bus.
  - Software may trigger an interrupt by executing a special operation called a **system call** (also, called **monitor call**).

Computer System Organization

- Computer-system operation
  - A modern general-purpose computer system consists of:
    - One or more CPUs, and a number of device controllers connect through common bus providing access to shared memory.
    - Each device controller is in charge of a particular device type.
Computer-System Operation

- I/O devices and the CPU can execute concurrently
- 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
- Device controller informs CPU that it has finished its operation by causing an interrupt

Common Functions of Interrupts

- 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
- Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt
- A trap is a software-generated interrupt caused either by an error or a user request
- An operating system is interrupt driven
- 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
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 interrupt:**
    - is a specific type of **I/O interrupt** that notifies the part of the computer containing the I/O interface that a device is ready to be read or otherwise handled but does not indicate which device.
    - The interrupt controller must poll (send a signal out to) each device to determine which one made the request.
  - **Vectored interrupt system**
    - an interrupt signal that includes the identity of the device sending the interrupt signal.
  - Separate segments of code determine what action should be taken for each type of interrupt.

Interrupt Timeline

1. I/O device informs CPU that it has finished its operation by causing an **interrupt**
2. When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location.
3. The fixed location usually contains the starting address where the service routine for the interrupt is located.
4. The interrupt service routine executes; on completion, the CPU resumes the interrupted computation.
I/O Structure

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

- After I/O starts, control returns to user program without waiting for I/O completion
  - **System call** – request to the operating system 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
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt

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
**Operating System Structure**

- **Multiprogramming (Batch system)** needed for efficiency
  - 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**
  - When it has to wait (for I/O for example), OS switches to another job
  - **Multiprogramming increases CPU utilization**, but they do not provide for user interaction with the computer system.

- **Timesharing (multitasking)** 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**
  - 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
Operating-System Operations

- **Dual-mode** operation allows OS to protect itself and other system components (it used to ensure the proper execution of the OS)
  - **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

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time
End of Chapter 1