Operating System

1

Textbook:

Abraham Silberschatz, *Peter Baer Galvin*, *Greg Gagne*,
Operating System Concepts, John Wiley & Sons, Inc.

Rough grade breakdown:

average of 4 Midterm (50%) + Final (50%)

midterm 1 (week 5: 26/02/2024 -02/03/2024)
midterm 2 (week 8), midterm 3 (week 11), midterm 4 (week 14).
final exam (according to the school's exam schedule)

All tests are online quizzes on https://utex.hcmute.edu.vn/

2

Contents:

Chapter 1. Introduction

To provide a grand tour of the major operating systems components

To provide coverage of basic computer system organization.

Chapter 2: Operating System Structures

To describe the service an operating system provides to users, processes, and other system.

To discuss the various way of structuring an OS.

To explain how OS are installed and customized and how they boot.

3

Contents:

Chapter 3. Processes

To introduce the notion of a process - a program in execution, which form the basis of all computation

To describe the various features of processes, including scheduling, creation and termination, and communication.

To introduction a fundamental unit of CPU utilization that forms the basis of multi-threaded computer systems

To examine issues related to multi-threaded programming.

4

Contents:

Chapter 4. CPU scheduling

To introduce CPU scheduling, which is the basis for multi-programmed operating systems

To describe various CPU-scheduling algorithms

To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system.

5

Contents:

Chapter 5. Process Synchronization

To introduce the critical-section problem, whose solutions can be used to ensure the consistency of shared data.

To present both software and hardware solutions of the critical-section problem.

To introduce the concept of an atomic transaction and describe mechanisms to ensure atomicity

Chapter 6. Deadlocks

To develop a description of deadlocks, which prevent sets of concurrent processes from completing their tasks

To present a number of different methods for preventing or avoiding deadlocks in a computer system

6

Contents:

Chapter 7: Memory Management - Main Memory

To provide a detailed description of various ways of organizing memory hardware

To discuss various memory-management techniques, including paging and segmentation

To provide a detailed description of the Intel Pentium, which supports both pure segmentation and segmentation with paging

7

Contents:

Chapter 8. Virtual memory - Mass Storage

To describe the benefits of a virtual memory system

To explain the concepts of demand paging, page-replacement algorithms, and allocation of page frames

To discuss the principle of the working-set model

Disk Performance - Hardware performance parameters, Queuing Theory.

Chapter 9: File System

Explain the function of file systems.

Describe the interfaces to file systems.

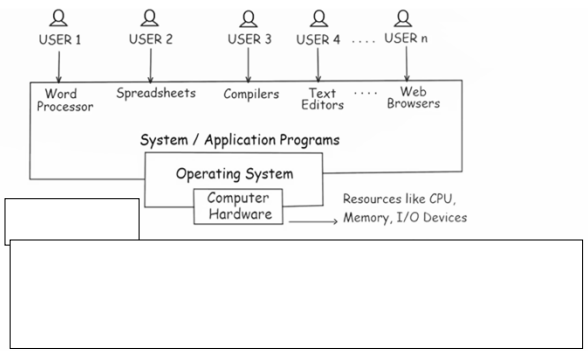
8

Introduction to Operating System

Introduction to OPERATING SYSTEM

→ An Operating System (OS) is a program that manages the computer hardware.
→ It also provides a basis for Application Programs and acts as an intermediary between computer User and computer Hardware.

10



11

Types of OS:
→ Batch OS
→ Time sharing OS
→ Distributed OS
→ Network OS
→ Real Time OS
→ Multi Programming/ Processing/ Tasking OS

Goals of OS:
i) Convenience
ii) Efficiency
iii) Both

Functions of OS:
→ It is an interface between User & Hardware
→ Allocation of Resources
→ Management of Memory, Security, etc.

12

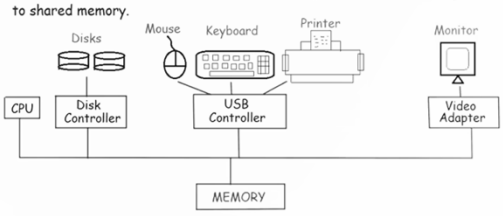
Basics of OS

13

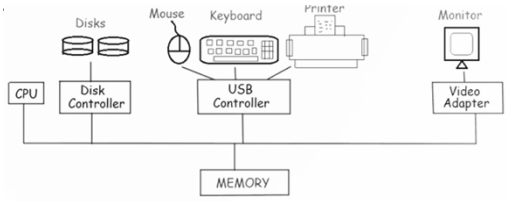
Basics of Operating System (Computer System Operation)

Some basic knowledge of the structure of Computer System is required to understand how Operating Systems work.

→ A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory.



14



→ Each device controller is in charge of a specific type of device.

→ The CPU and the device controllers can execute concurrently, competing for memory cycles.

→ To ensure orderly access to the shared memory, a memory controller is provided whose function is to synchronize access to the memory

15

Some important terms:

1) **Bootstrap Program:** → The initial program that runs when a computer is powered up or rebooted.
→ It is stored in the ROM.
→ It must know how to load the OS and start executing that system.
→ It must locate and load into memory the OS Kernel.

2) **Interrupt:** → The occurence of an event is usually signalled by an Interrupt from Hardware or Software.
→ Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by the way of the system bus.

3) **System Call (Monitor call):** → Software may trigger an interrupt by executing a special operation called System Call.

16

When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location.

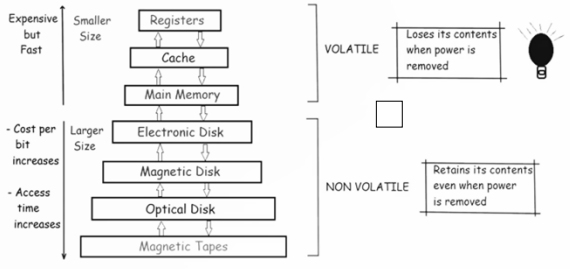
→ The fixed location usually contains the starting address where the Service Routine of the interrupt is located.

The Interrupt Service Routine executes.

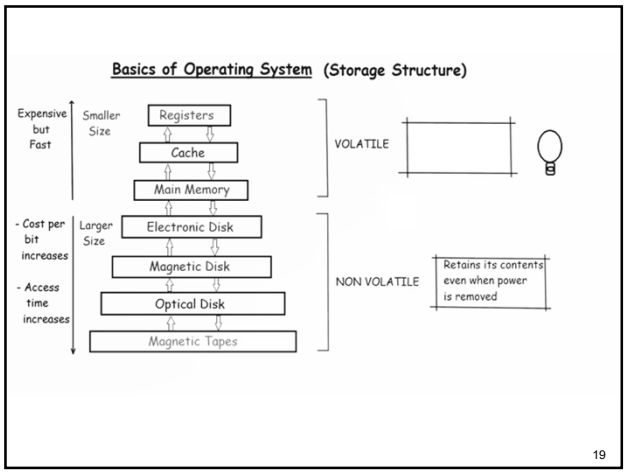
On completion, the CPU resumes the interrupted computation.

17

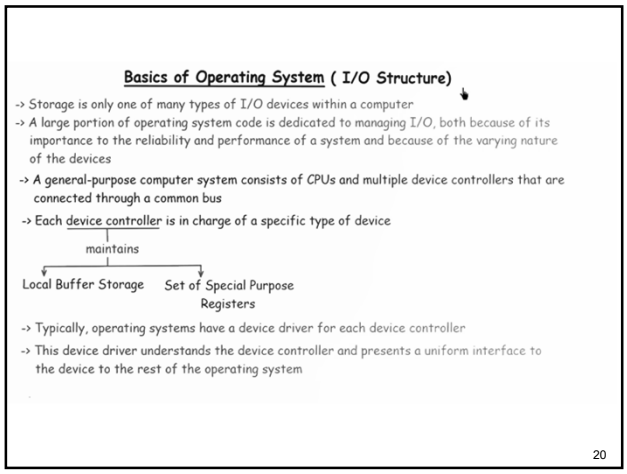
Basics of Operating System (Storage Structure)



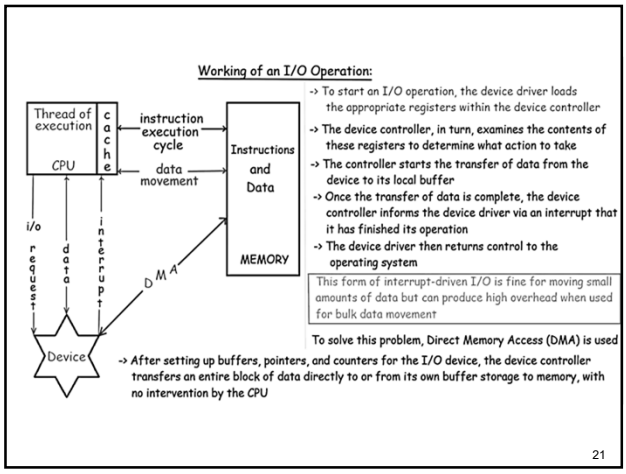
18


Basics of Operating System (Storage Structure)
Expensive but Fast
Smaller Size
Registers
Cache
Main Memory
VOLATILE
- Cost per bit increases
- Access time increases
Larger Size
Electronic Disk
Magnetic Disk
Optical Disk
Magnetic Tapes
NON VOLATILE
Retains its contents even when power is removed
19

19


Basics of Operating System (I/O Structure)
-> Storage is only one of many types of I/O devices within a computer
-> A large portion of operating system code is dedicated to managing I/O, both because of its importance to the reliability and performance of a system and because of the varying nature of the devices
-> A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus
-> Each device controller is in charge of a specific type of device
maintains
Local Buffer Storage
Set of Special Purpose Registers
-> Typically, operating systems have a device driver for each device controller
-> This device driver understands the device controller and presents a uniform interface to the device to the rest of the operating system
20

20


Working of an I/O Operation:
Thread of execution
CPU
cache
instruction execution cycle
data movement
Instructions and Data
MEMORY
i/o request
data
interrupt
DMA
Device
-> To start an I/O operation, the device driver loads the appropriate registers within the device controller
-> The device controller, in turn, examines the contents of these registers to determine what action to take
-> The controller starts the transfer of data from the device to its local buffer
-> Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation
-> The device driver then returns control to the operating system
This form of interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement
To solve this problem, Direct Memory Access (DMA) is used
-> After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU
21

21