DSE 3151

OPERATING SYSTEMS

V Semester BTech DSE

Dr. Poornima P Kundapur
Associate Professor

poornima.girish@manipal.edu
99808 53396

DSE 3153 OPERATING SYSTEMS

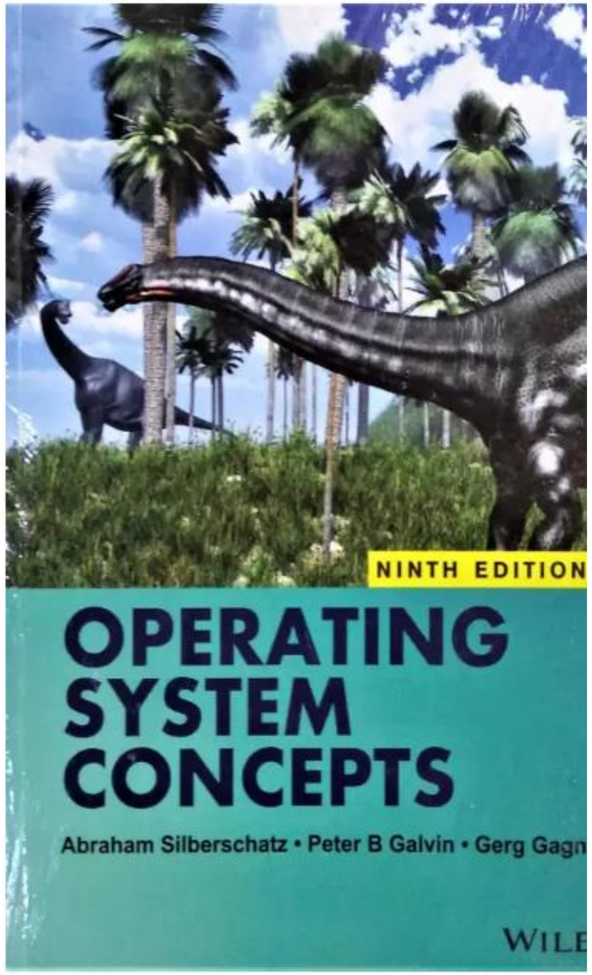
Operating System Structure and Operations, Process Management, **Memory Management**, Storage Management, Operating System Services, User Operating System Interfaces, Types of **System Calls,** System Programs, Operating System Structure, System Boot, Overview, Process Scheduling, Operations on Processes, **Inter-process Communication,** Multithreaded Models, Thread Libraries, **Scheduling Algorithms**, Thread Scheduling, Linux scheduling, Critical Section Problem, Peterson's Solution, **Synchronization Hardware,** Semaphores, Logical Versus Physical Address Space, Segmentation, Contiguous **Memory Allocation**, Paging, Structure of Page Table, Segmentation, Demand Paging, Copy-On-Write, **Page Replacement**, Allocation of Frames, Thrashing, Disk Scheduling, Swap-Space Management, System Model, Deadlock: Deadlock prevention, Avoidance, Detection, Recovery, **File Concept**, Protection.

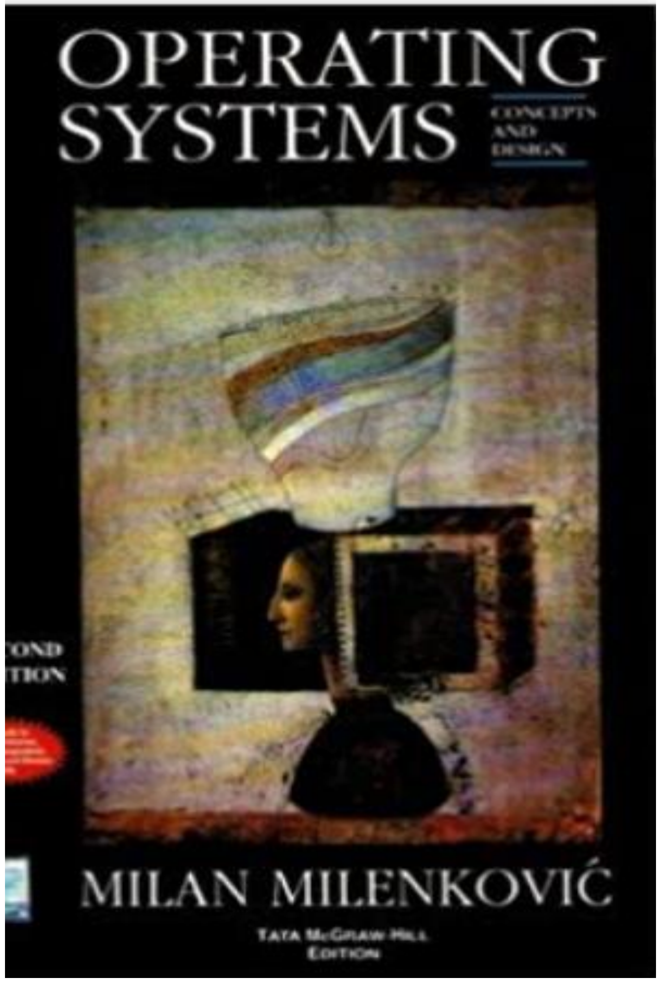
Multicore systems

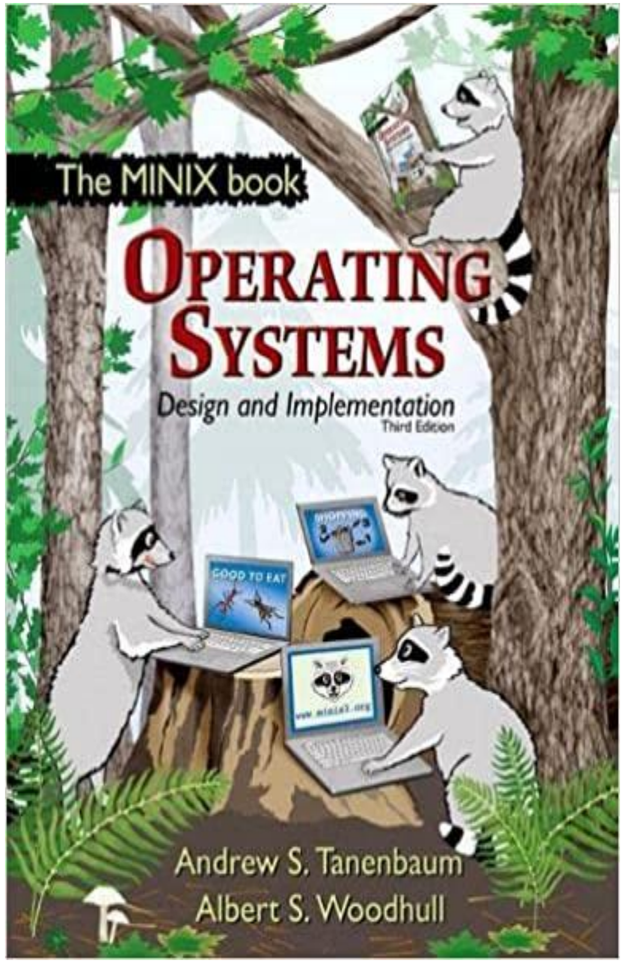
Mobile computing

Virtualization

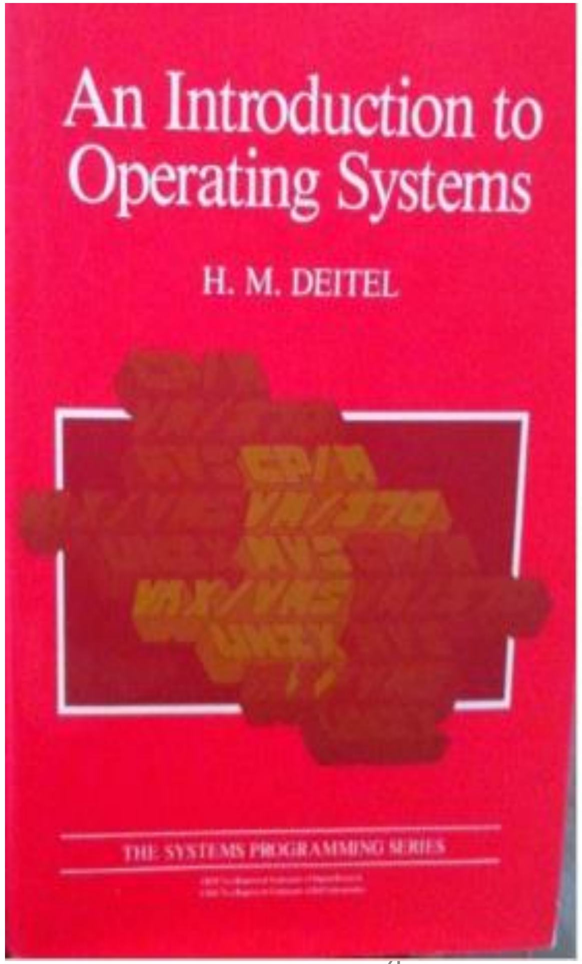
Reference Books







https://www.minix3.org/



https://www.youtube.com/watch?v=MG29rUtvNXg

Class Engagement Rules.

- **Be on time** to class. No Late entry.
- **Late entry**: No attendance but you can sit through the class.
- Carry a notebook for all classes, helps in your revision.
- 75% isn’t what you signed up for, so let's try to work above that.
- Questions are allowed, there is no silly question.
- Problem solving is an essential part of learning, engage in it.
- Let the learning happen.

Course Outcomes

At the end of the course, you should be able to

- Understand Operating System components, structure and the process concept
- Analyze the requirement of threads and process scheduling.
- Explore process synchronization, deadlocks avoidance, detection and recovery algorithms.
- Evaluate different memory management strategies.
- Analyze file Systems, secondary storage management, protection and Linux system.

Chapter 1: Introduction

Objectives of this chapter.

- To describe the basic organization of computer systems
- 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

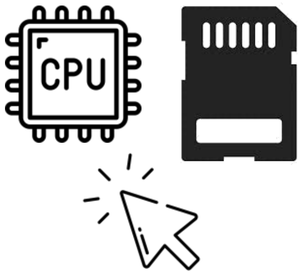
Computer System Structure

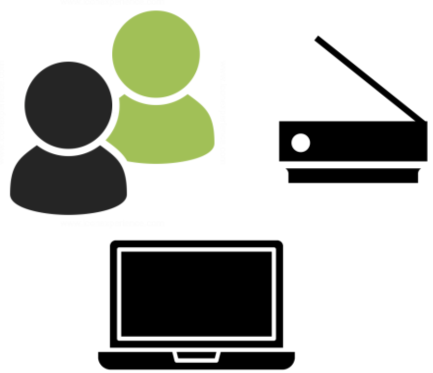
Hardware

Operating system

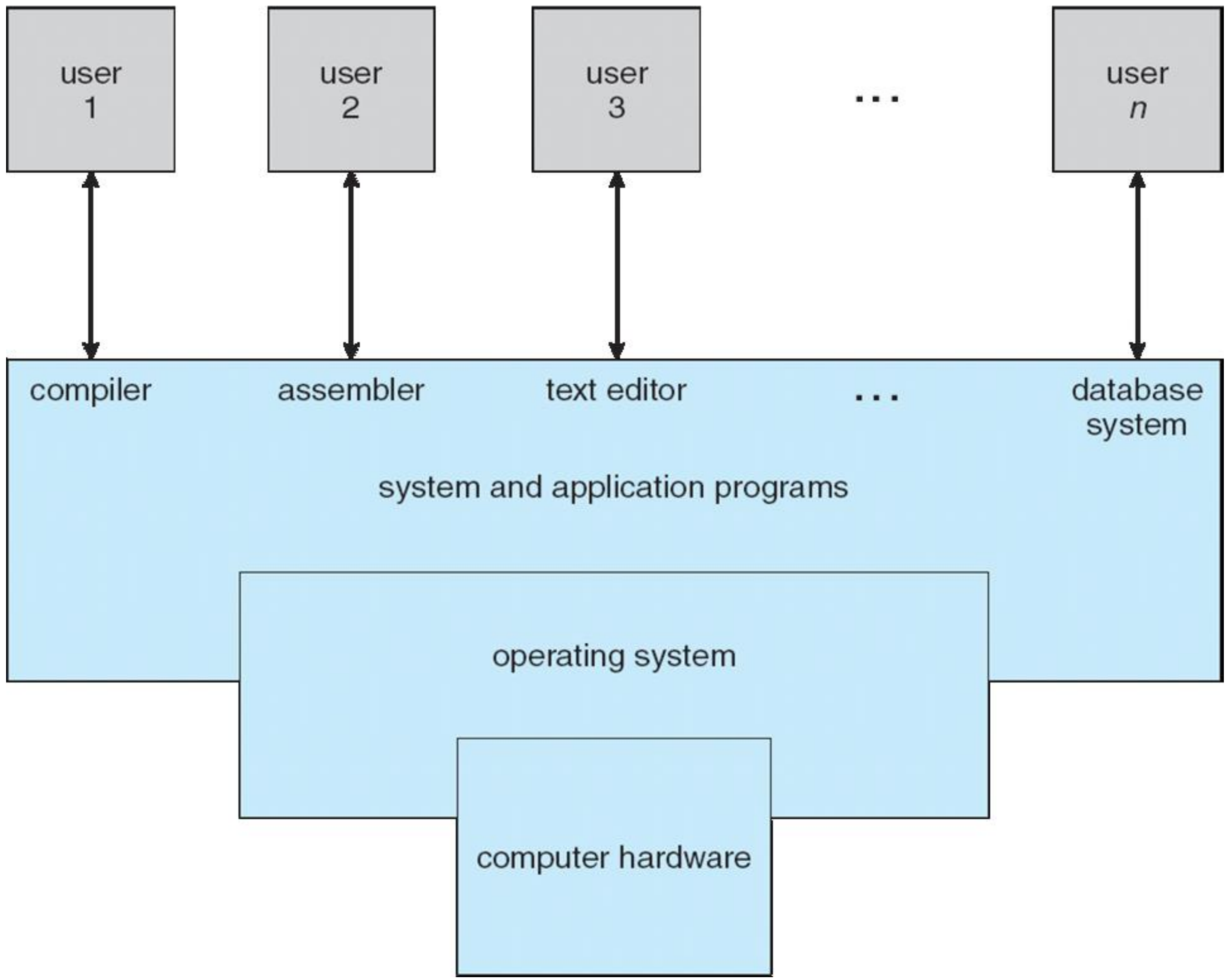
Application programs

Users





Four Components of a Computer System



What Operating Systems Do?

- Depends on the point of view:
  - **Users:** Convenience, ease of use and good performance and don’t care about resource utilization
  - **Shared computer** (mainframe or minicomputer) must keep all users happy
  - **Users of dedicate systems** (workstations) have dedicated resources but frequently use shared resources from servers
  - **Handheld computers:** Resource poor, optimized for usability and battery life
  - **Embedded computers: (**devices and automobiles) have little or no user interface.

Definition of an Operating System.

- An Operating System (OS) is a software that acts as an **interface between** computer hardware components and the user.
- 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

Definition of an Operating System.

- 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

Definition of an Operating System.

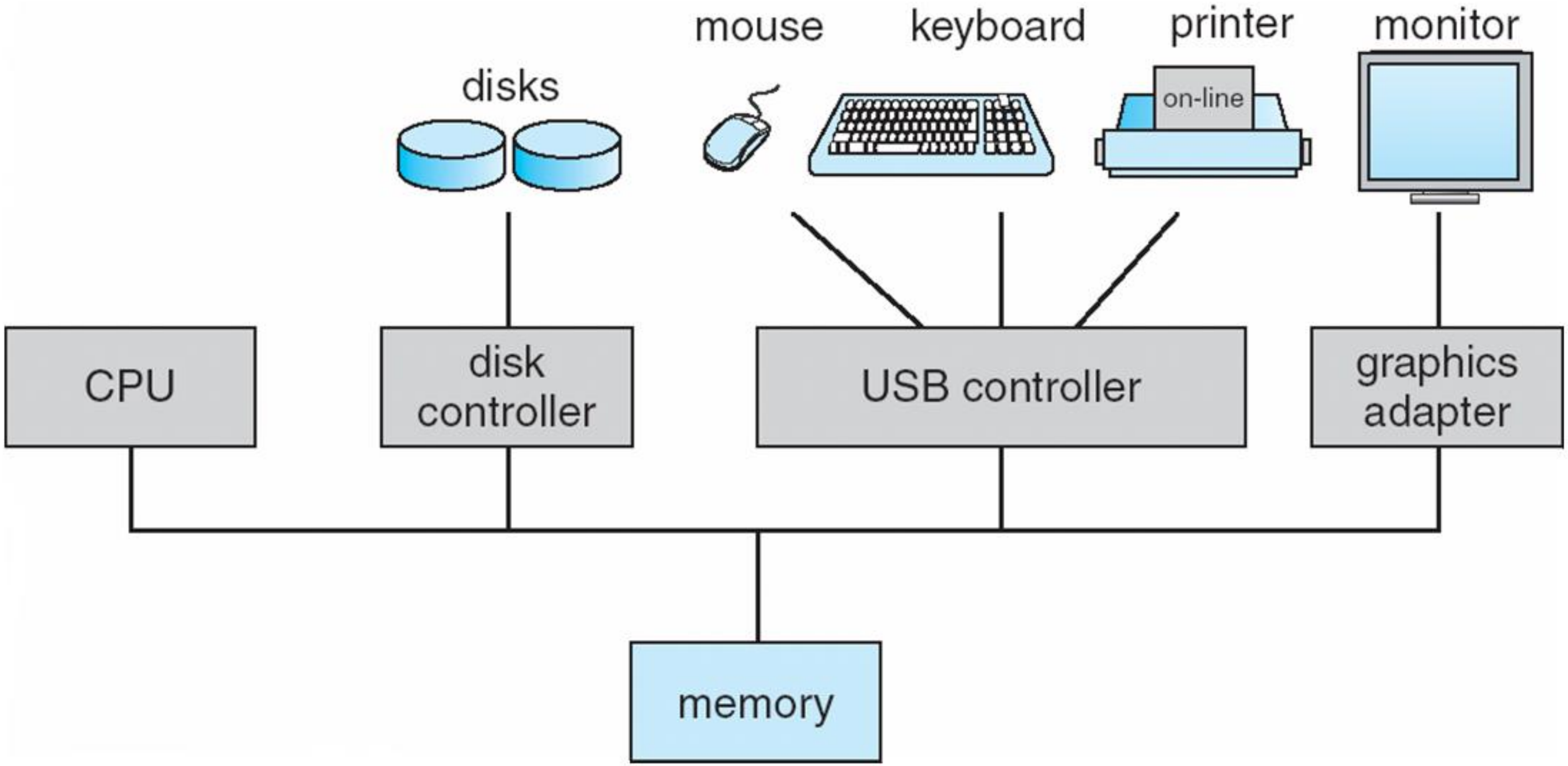
- No universally accepted definition
- "**Everything a vendor ships when you order an operating system" is a good approximation**
- But varies wildly
  - "The one program running at all times on the computer" is **the kernel.**
  - Everything else is either
    - a system program (ships with the operating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

Definition of a “reliable” Operating System.

- An operating system is said to be reliable when a typical user has never experienced even a single failure in his or her lifetime and does not know anybody who has ever experienced a failure.

Andrew S. Tanenbaum: MINIX 3

Computer System organization.



Storage definitions

- Bit: 0 or 1 (basic unit)
- Nibble: 4 bit aggregation
- Byte: 8 bit octet
- Kilobyte: 1024 bytes (KB)
- Megabyte: 1024^2 bytes (MB)
- Gigabyte:1024^3 bytes (GB)
- Terabyte: 1024^4 bytes (GB)
- Petabyte: 1024^5 bytes (GB)
- **Network measurements all in bits**

After the break.

Operating System Structure

Multiprogramming

Timesharing

Multiprogramming (Batch system)

Job pool

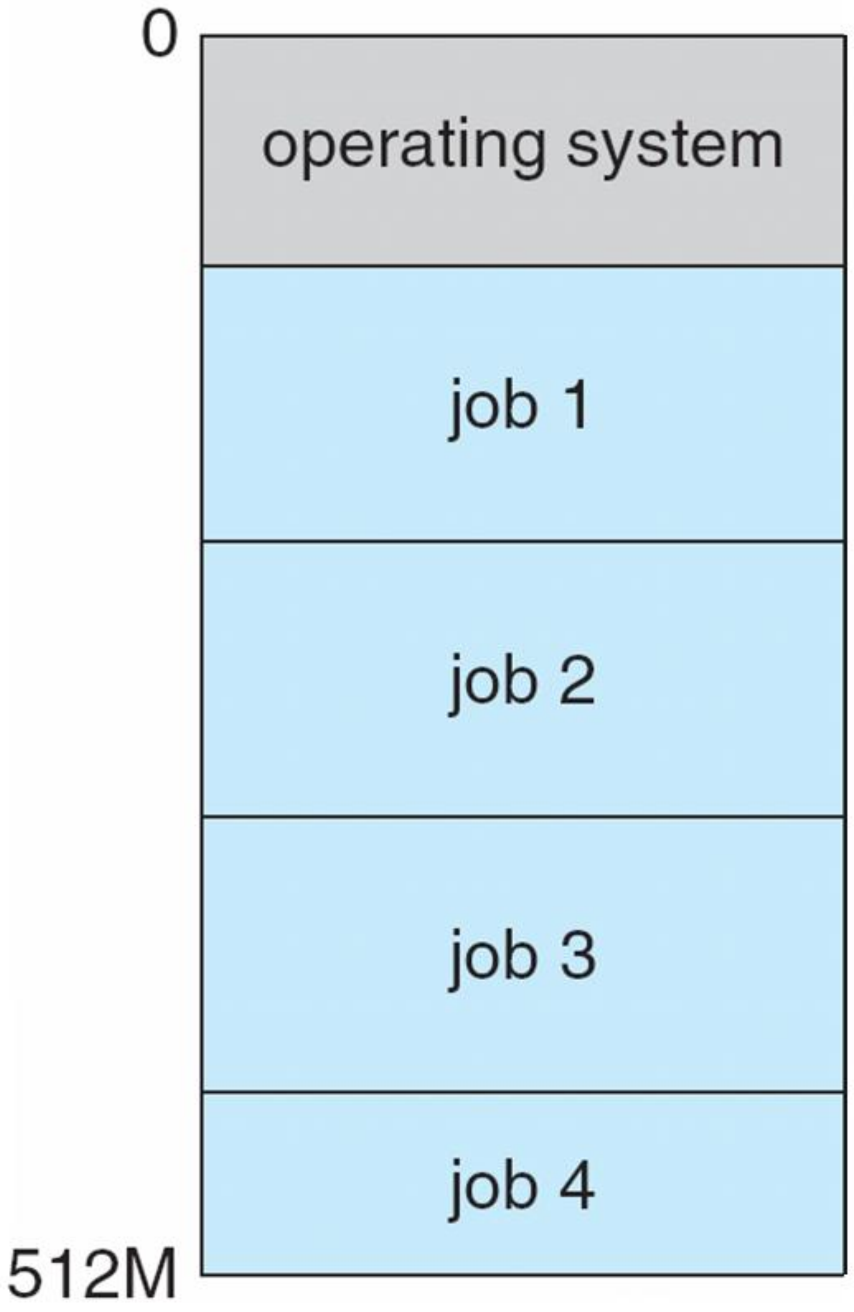
- Needed for efficiency
- Single program ***cannot keep CPU or I/O devices*** busy at all times
- Single users have multiple programs running
- Increases CPU utilization
  - Organizes **jobs (code & data)** so CPU always has one to execute

Multiprogramming (Jobs)

- How does it work?
- OS keeps several jobs in **memory** simultaneously
- *Main memory (RAM)* is **too small** to accommodate all jobs, the jobs are kept initially on the disk in the **job pool.**
- The set of jobs in memory can be a **subset of the jobs** kept in the job pool.
- One job selected and run via job scheduling
- When it has to wait (for I/O for example), OS switches to another job

Multiprogramming (Jobs)

Job pool
consists of **all processes** residing on **disk** awaiting allocation of main memory.



Timesharing (multitasking).

Logical extension

- CPU executes **multiple jobs** by ***switching*** among them, but the switches occur so **frequently** that the **users can interact** with ***each program*** while it is running
- *Creating* **interactive** computing
- Response time should be **< 1 second**

Timesharing (multitasking).

- **Allows many users** to **share the computer** **simultaneously**
- **Each action** or command tends to be **short**, only a **little CPU** time is ***needed for each user.***
- As the **system switches rapidly** from **one user** to the **next**, each **user is given the impression** that the entire computer system is **dedicated** to his use, even though it is being shared among many users.

Timesharing (Process).

- Each user has **at least one separate program** in memory.
- A program loaded into memory and executing is called a **process**

CPU scheduling

Multiprogramming

Timesharing (Job scheduling)

If **several jobs are ready** to be brought into memory, and if there is ***not enough room*** for all of them, then the **system must choose** among them.

Making this decision involves **job scheduling** (Chapter 6)

Timesharing (Memory Management)

When the operating system **selects a job** from the job pool,

Loads that job into **memory** for **execution**

Several programs in memory at the ***same time*** requires some form of **memory management** (Chapters 8 and 9)

Timesharing (CPU Scheduling)

If **several jobs** are **ready to run** at the **same time,** system *must choose which job* will run first.

Making this decision is called **CPU scheduling** (*Chapter 6*)

Time sharing (Virtual Memory)

Reasonable **response time**

- Processes are swapped in and out of main memory to the disk.
- **Virtual memory:** *a technique that allows the execution of a process that is not completely in memory* (Chapter 9).
- Why virtual-memory scheme?
  - Enables users to run programs that are larger than actual physical memory.
  - Abstracts main memory into a large, uniform array of storage, separating logical memory as viewed by the user from physical memory.
  - Frees programmers from concern over memory-storage limitations.

Time sharing (File System Management)

- **File system** resides on a collection of disks: Disk management (Chapter 10).
- To ensure orderly execution, the system must provide mechanisms for **job synchronization and communication** (Chapter 5)
- Ensure that jobs do not get stuck in a **deadlock**, forever waiting for one another (Chapter 7).

Modern 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

Modern Operating-System Operations.

More **subtle errors can occur** in a **multiprogramming system,** where one erroneous program might modify another program, the data of another program, or even the operating system itself

A **properly designed operating system** must ensure that an **incorrect (or malicious) program** cannot cause other programs to execute incorrectly.

Dual-Mode and Multimode Operation.

In order to ensure the proper execution of the operating system, we must be able to **distinguish** between the execution of **operating-system code** and **user-defined code**

Modes of Operation.

User mode

Kernel mode

Supervisor mode, system mode, or privileged mode

Modes of Operation.

A bit, called the **mode bit,** indicates the current mode: **kernel (0)** or **user (1).**

Mode bit distinguishes between a task that is executed on behalf of the operating system and one that is executed on behalf of the user.

When the **computer system is executing on behalf of a user** application, the system is in **user mode**.

When a **user application requests a service** from the operating system (via a **system call**), the system must transition from user to **kernel mode** to fulfil the request.

Dual-Mode and Multimode Operation.

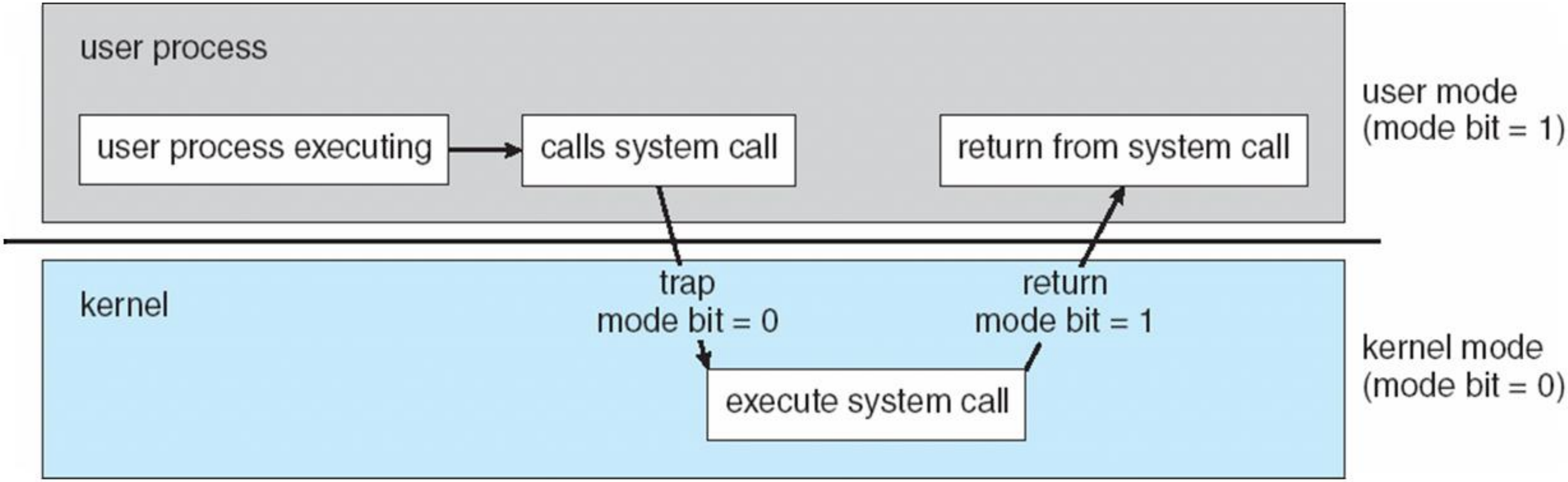


Figure 1.10 Transition from user to kernel mode

Dual-Mode and Multimode Operation

- System boot time: **Hardware** starts in **kernel mode**.
- OS then loaded and starts **user applications** in **user mode**.
- Trap or interrupt occurs:
  - Hardware switches from user mode to kernel mode (state of mode bit – 0)
- Whenever the OS gains control of the computer, it is in **kernel mode.**
- System always switches to user mode (setting mode bit to 1) before passing control to a user program.

Dual-Mode Operation? How does it help?

- **Means for protecting the OS** from errant users—and errant users from one another.
- How?
  - By designating some of the machine instructions that may cause harm as **privileged instructions**.
- Hardware allows **privileged instructions** to be executed only **in kernel mode**.
- If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute *(illegal & traps it)*

So, what is a privileged instruction?

The instruction to switch to kernel mode is an example of a **privileged instruction.**

I/O control

Interrupt management

Timer management

Virtual Memory Manager

AFTER LAST CLASS.

Timer.

- **Does not allow** a user program to get stuck **in an infinite loop** or to **fail to call <u>system services</u>** and **never return control** to the operating system.
- Can be **set to interrupt** the computer after a specified period.

Fixed

(1/60 second)

Variable

(1 millisecond to 1 second)
Implemented by a fixed-rate clock and a counter

Who/What sets the counter?

Timer.

- **OS sets the counter**. Every time the clock ticks, the counter is decremented. When the counter reaches 0, an interrupt occurs.
- For instance, a ***10-bit counter*** with a 1-millisecond clock allows interrupts at intervals from 1 millisecond to 1,024 milliseconds, in steps of 1 millisecond.

Process Management (1)

- A **program does nothing** *unless its instructions* are **executed by a CPU.**
- **A program in execution,** as mentioned earlier many times, **is a process**.
  - *Compiler is a process.*
  - A *word-processing program* on a PC is a process.
  - A *system task*, such as sending output to a printer, can also be a process (or at least part of one).

Process Management (2)

- A **process needs certain resources** (CPU time, memory, files, and I/O devices) to accomplish its task.
  - Given to the process when it is **created or allocated** to it while it is running.
- **A program by itself is not a process**.
- A **program is a passive entity**, like the contents of a file stored on disk.
- A **process is an active entity.** A single-threaded process has one program counter specifying the next instruction to execute (Chapter 4)

Process Management (3)

- A **process** is the unit of work in a system.
- A **system consists of a collection of processes:**

Operating system processes

(those that execute system code)

User processes

(those that execute user code)

- All **these processes execute concurrently** by multiplexing on a single CPU, for example.

Process Management (4)

- OS is responsible for the **following activi**ties in connection with process management:
  - **Scheduling processe**s and **threads** on the CPUs
  - **Creating and deleting** both *user and system processes*
  - **Suspending and resuming** processes
  - Providing mechanisms for **process synchronization**
  - Providing mechanisms for **process communication**

MEMORY MANAGEMENT (1)

- **Main memory** is central to the operation of a modern computer system.
- To **execute a program** all (or part) of the instructions must be in ***memory***
- All (or part) of the **data** **that is needed by the *program*** must be in memory.
- Memory management determines what is in memory and when.
- Optimizing CPU utilization and computer response to users

MEMORY MANAGEMENT (1)

- OS is responsible for the **following activities** in connection with **memory management**:
  - **Keeping track** of which parts of **memory** are **currently being used** and **who** is using them
  - **Deciding which processes** (or parts of processes) and data to **move into and out of memory**
  - **Allocating** and **deallocating memory space** as needed

STORAGE MANAGEMENT.

- OS provides **uniform, logical view** of information storage
  - Abstracts physical properties to logical storage unit: **file**
  - Each **medium** is **controlled by device** (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

FILE SYSTEM **MANAGEMENT**.

- **Files** usually organized into **directories/folder**
- **Access control** on most systems to determine who can access what
- **OS activities** include
  - **Creating and deleting** files and directories
  - **Primitives to manipulate** files and directories
  - **Mapping files** onto secondary storage
  - **Backup files** onto stable (non-volatile) storage media

MASS STORAGE MANAGEMENT(1)

- **Disks** used to **store data** does not fit in main memory or data that must be kept for a “long” periods of time.
- Proper management is of central importance
- Entire **speed of computer operation** hinges on disk subsystem and its algorithms
- **OS activities**
  - Free-space management
  - Storage allocation
  - Disk scheduling

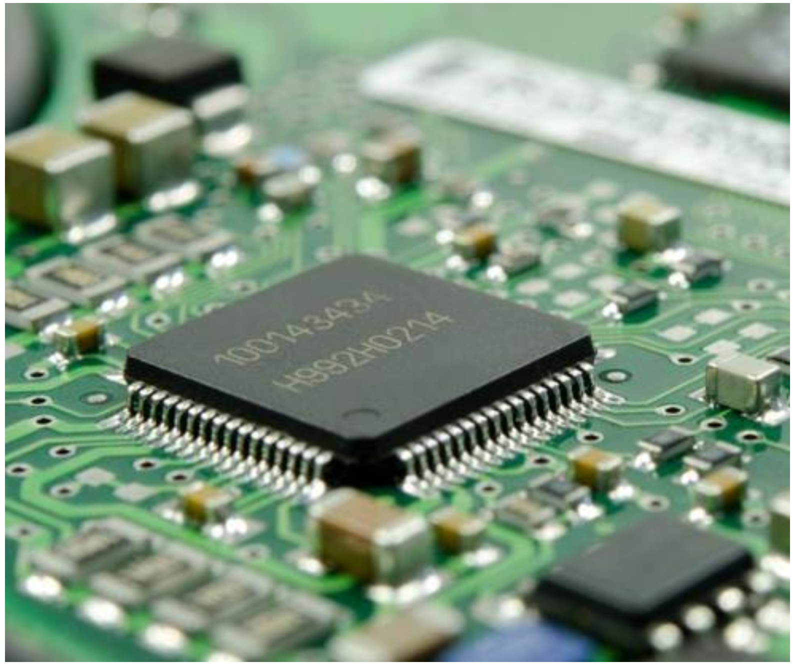
MASS STORAGE MANAGEMENT(2)

- Some **storage** need **not be fast**
  - **Tertiary storage** *(not crucial to system performance)* includes:
    - optical storage, magnetic tape
  - Still must be managed – by OS or applications
  - Varies between:
    - WORM: write-once, read-many-times, and
    - RW: read-write

Performance of Various Levels of Storage.

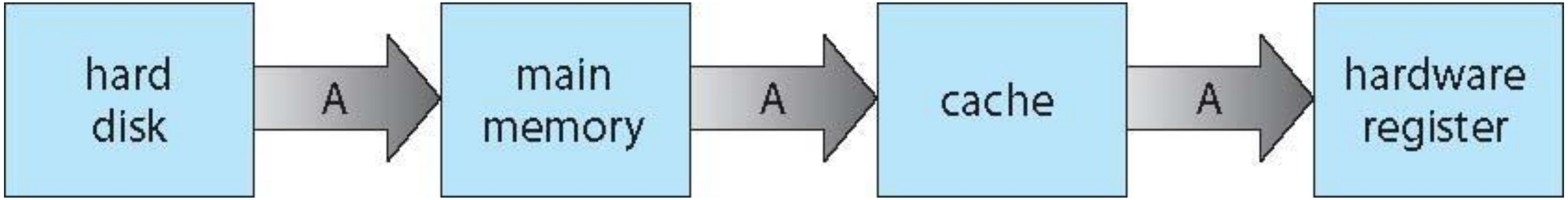
Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Caching.



- Information is normally kept in some storage system (*main memory*).
- As it is used, it is copied into a faster storage system *(the cache)* on a temporary basis.
- When a particular piece of **information is needed**, **cache** is checked. **If found**, ***information is directly used from the cache***.
- **If not,** ***information from the source is used*** and ***a copy*** is **placed in the cache** under assumption it will be needed again.

Migration of data “A” from Disk to Register.



- **Multitasking environments** **uses most recent value**, no matter where it is stored in the storage hierarchy
- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex, several copies of a datum can exist.

Migration of data "A" from Disk to Register.

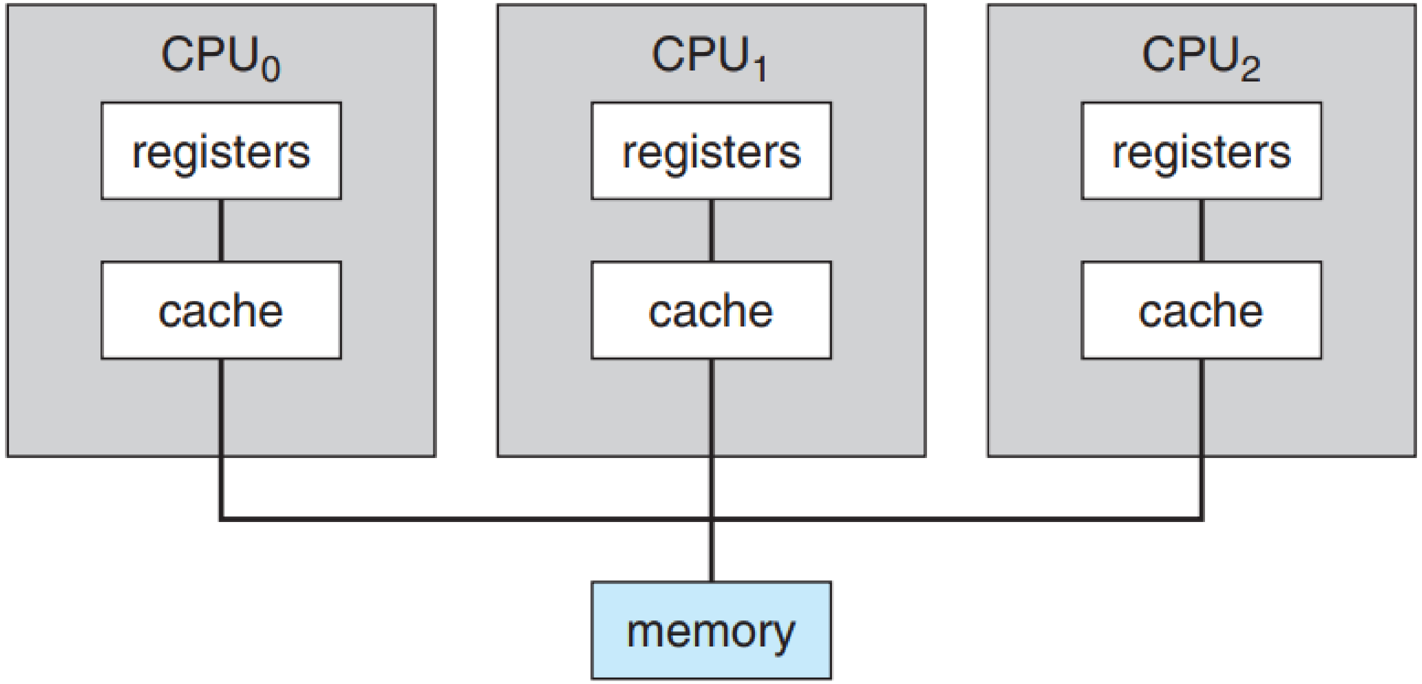


Figure 1.6 Symmetric multiprocessing architecture.

- A **copy of A** may exist simultaneously in several caches.
- Since the various ***CPUs can all execute in parallel***, an **update to value of A** in one cache is ***immediately reflected in all other caches*** where A resides.
- This situation is called **cache coherency**, and it is usually a hardware issue.

I/O Subsystems.

- One purpose of OS is to **hide peculiarities** of hardware devices from the user
- **I/O subsystem** responsible for:
  - **Memory management of I/O** including **buffering** (storing data temporarily while it is being transferred), **caching**, **spooling** (overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

PROTECTION AND SECURITY

Protection	SECURITY
Any mechanism for controlling access of processes or users to resources defined by the OS	Defense of the system against internal and external attacks
• Memory-addressing hardware ensures that a process can execute only within its own address space. • The timer ensures that no process can gain control of the CPU without eventually relinquishing control.	Huge range, including denial-of-service, worms, viruses, identity theft, theft of service

PROTECTION AND SECURITY

- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - **User ID** then associated with **all files, processes of that user** to determine access control
  - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights

End of Chapter 1

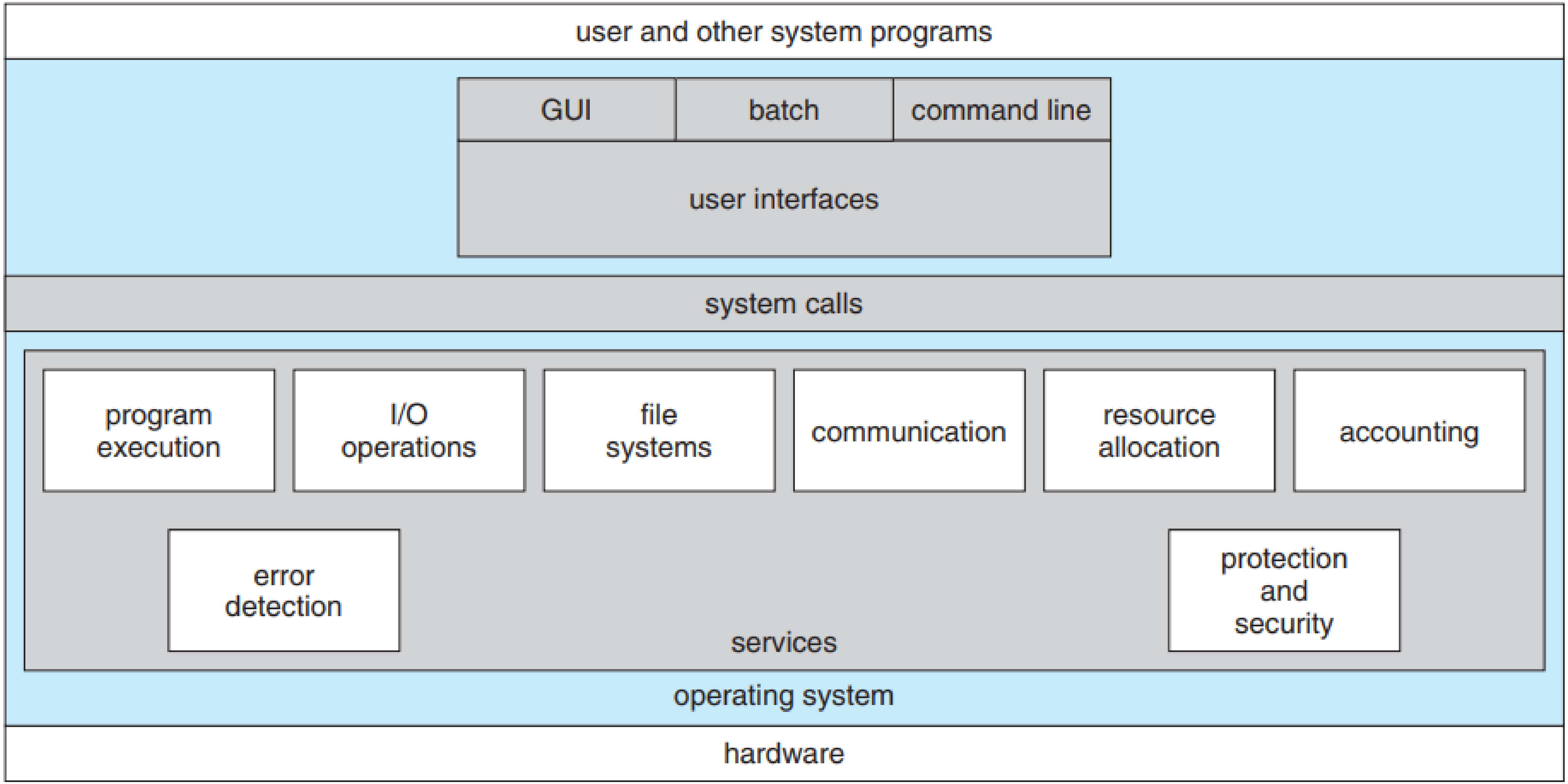


Figure 2.1 A view of operating system services.