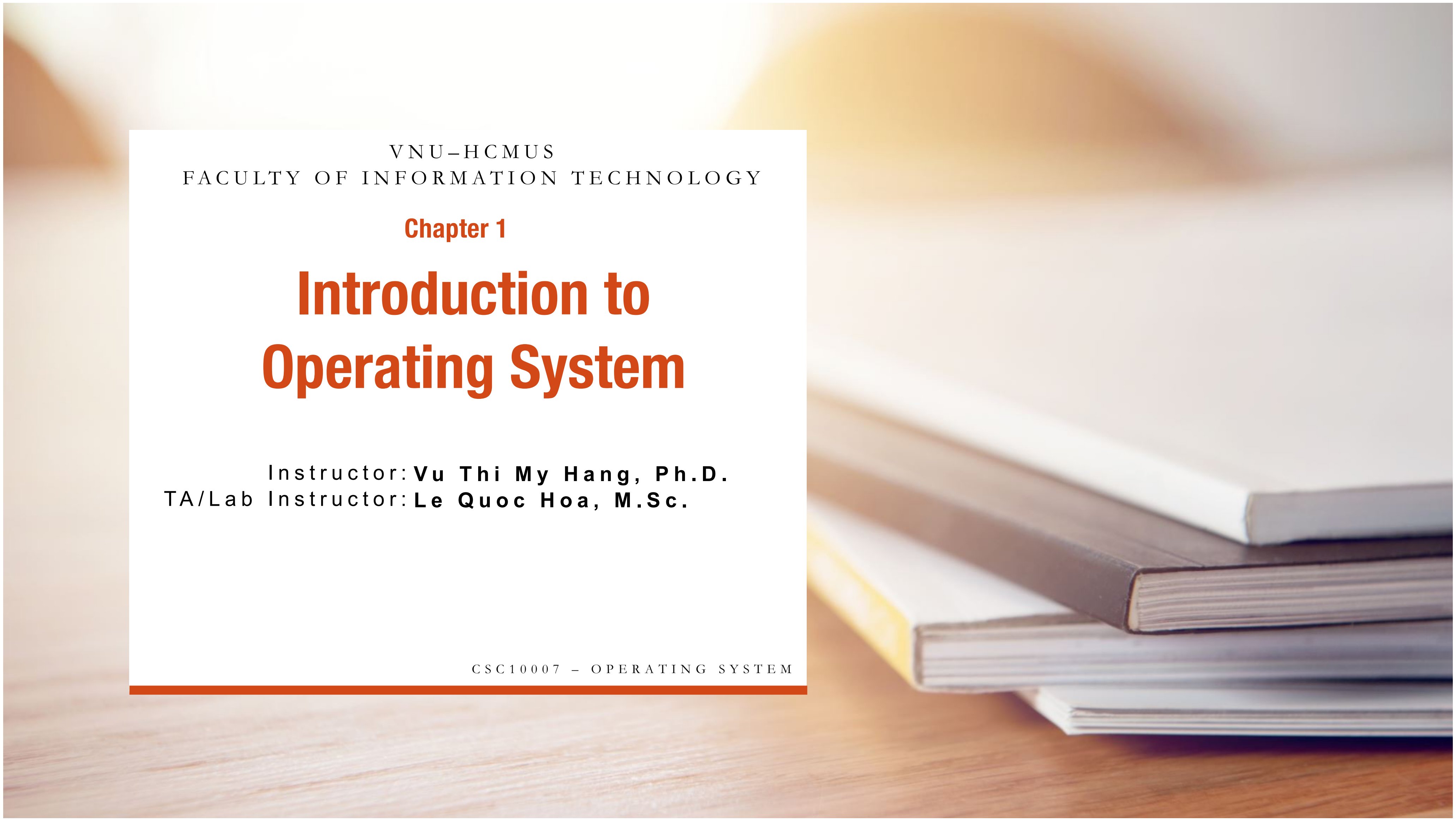


VNU–HCMUS

FACULTY OF INFORMATION TECHNOLOGY

Chapter 1

# Introduction to Operating System

Instructor: **Vu Thi My Hang, Ph.D.**

TA/Lab Instructor: **Le Quoc Hoa, M.Sc.**

## Plan

- OS: WHAT and WHY?
- Classification
- Main components
- OS Structure

## Plan

- **OS: WHAT and WHY?**
- Classification
- Main components
- OS Structure

WHAT and WHY?

## Operating System Definition

Windows Linux Ubuntu Mac OS/iOS Android

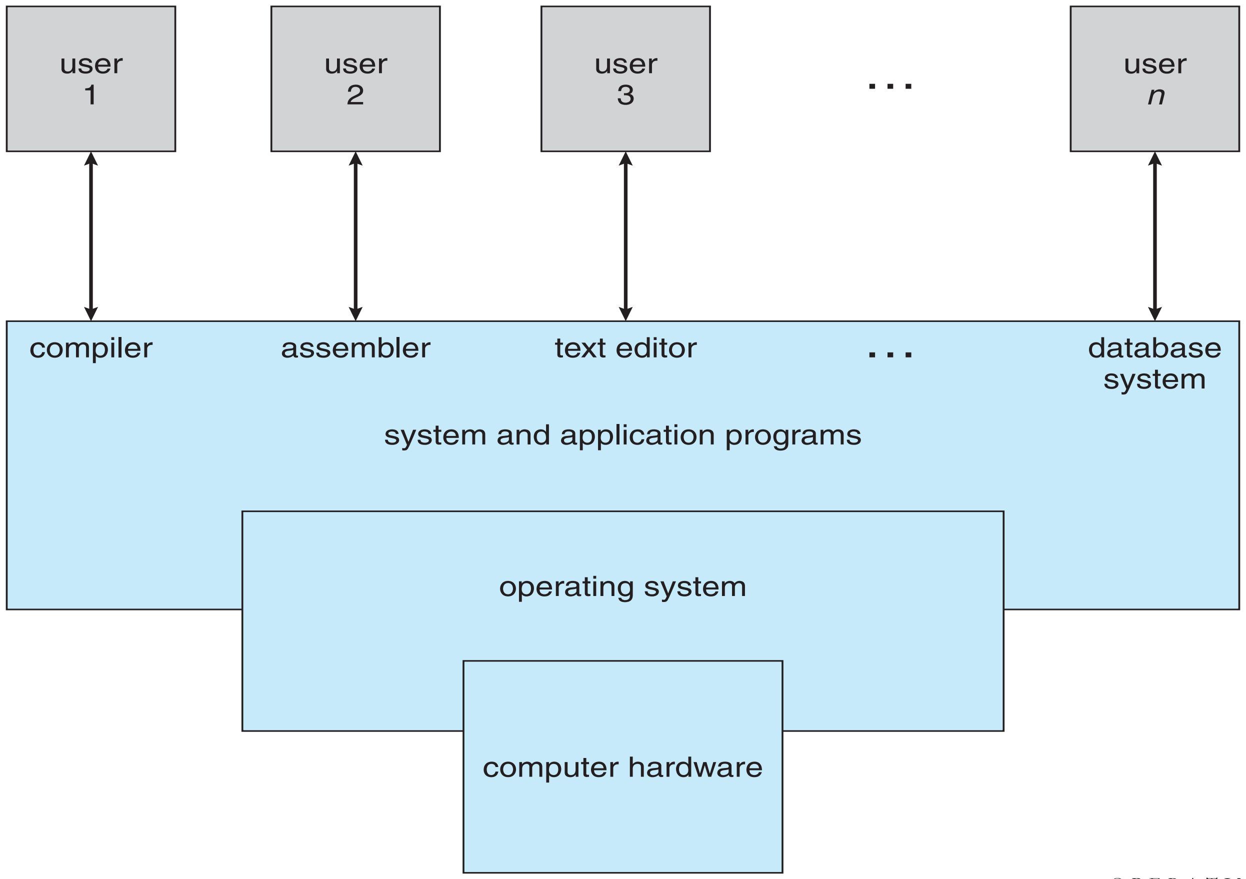
**A system program** (Phần mềm hệ thống)

- Provides an easy way to execute application programs, i.e., user programs solving user problems (e.g., word processor, web browser)

  (Phần mềm ứng dụng)
- Acts as an interface between application programs and computer hardware (e.g., CPU, memory, I/O devices)

WHAT and WHY?

## Computer System Overview

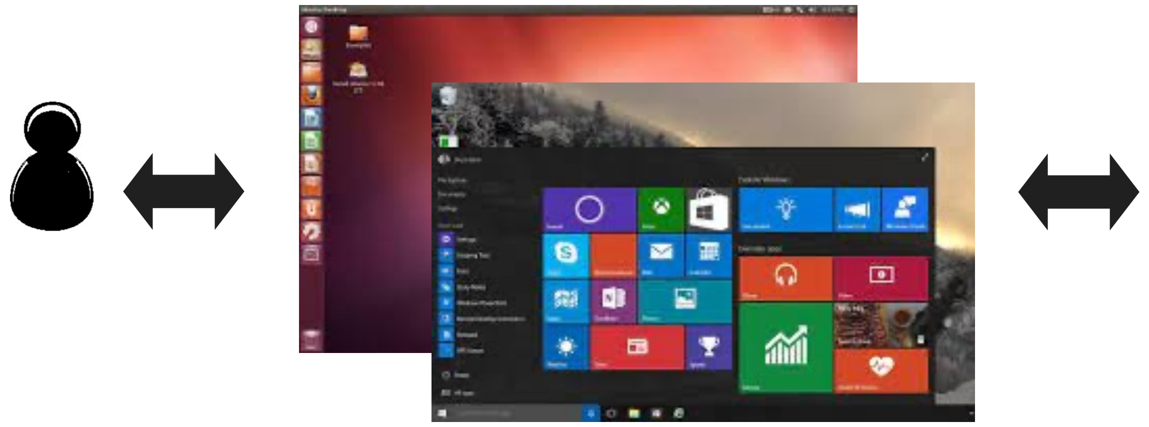


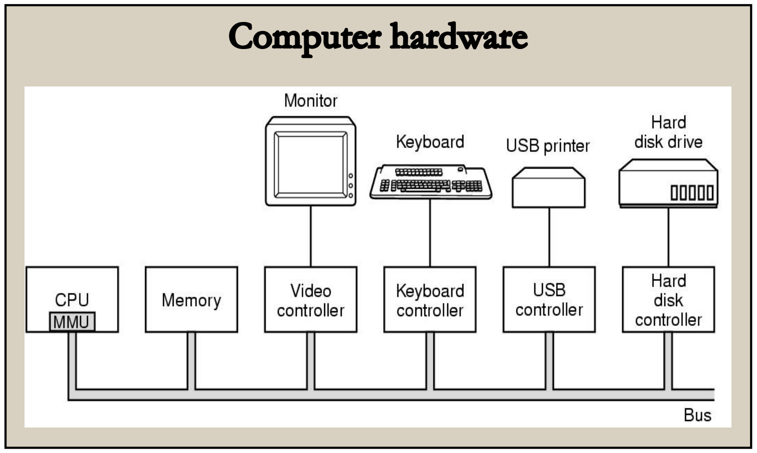
WHAT and WHY?

## Role of an Operating System

### Virtual machine (user view) (Giả lập máy tính mở rộng/máy tính ảo)

- Provides a uniform abstraction of computer hardware
- Provides abstract concepts (e.g. file) for using computers easier
  - ✓ What if users open myfile.docx?

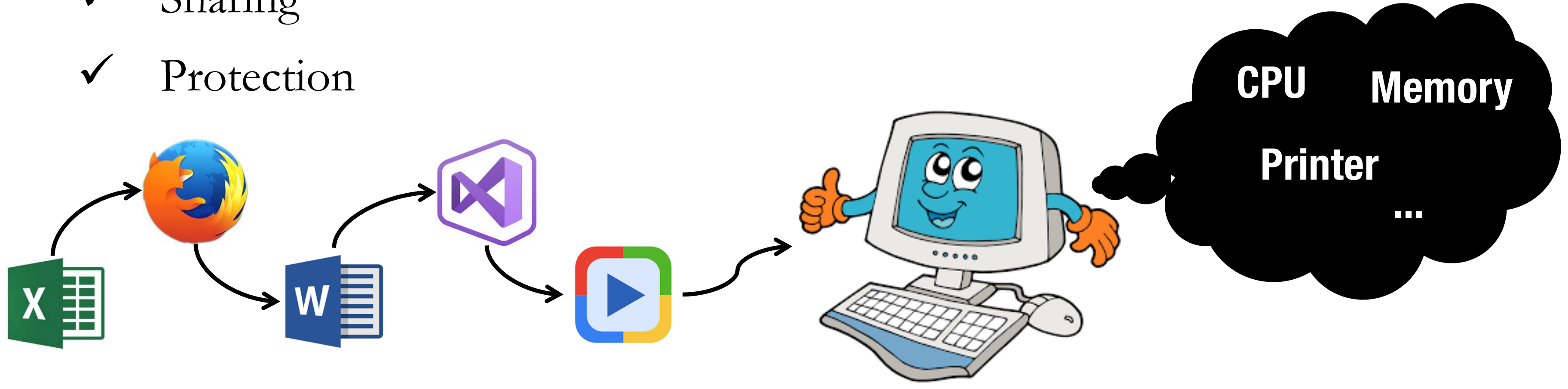




## Role of an Operating System

### Resource manager/allocator (system view) (Quản lý tài nguyên)

- Manages resources (CPU, memory, ...) among various programs in a fair, efficient, and safe manner
  - ✓ Allocation/Desallocation
  - ✓ Sharing
  - ✓ Protection



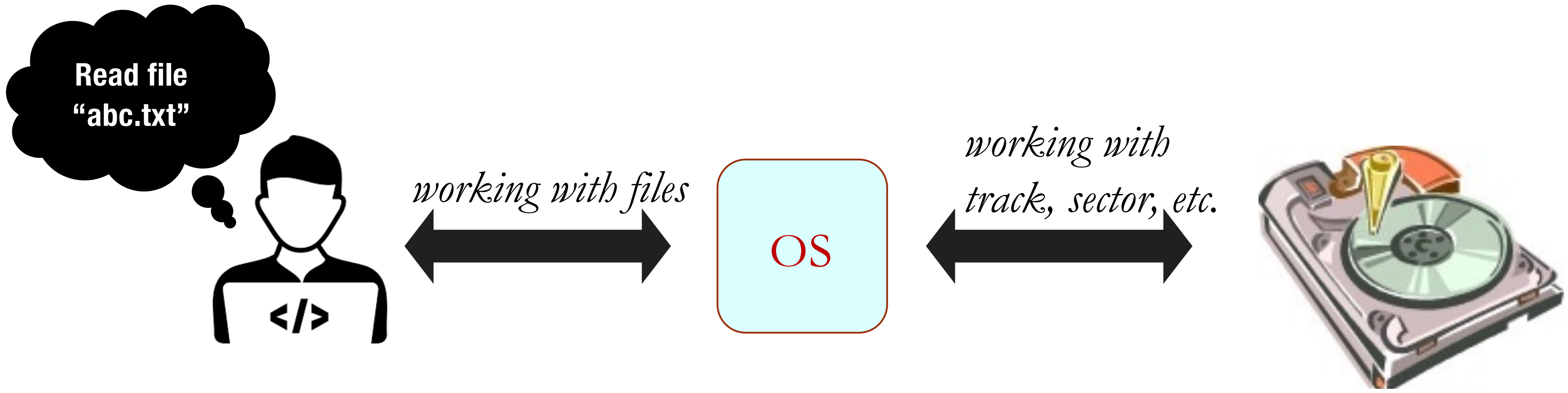
WHAT and WHY?

## Role of an Operating System

(Hỗ trợ giao tiếp với máy tính thông qua các lời gọi hệ thống)

### Set of utilities, i.e. system calls (programmer view)

- Facilitates and simplifies application programming
  - ✓ Write a program to read files from disk?



## Plan

- OS: WHAT and WHY?
- **Classification**
- Main components
- OS Structure

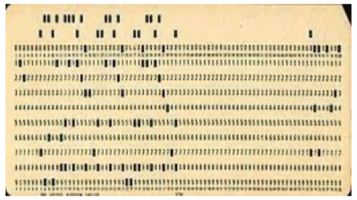
Classification

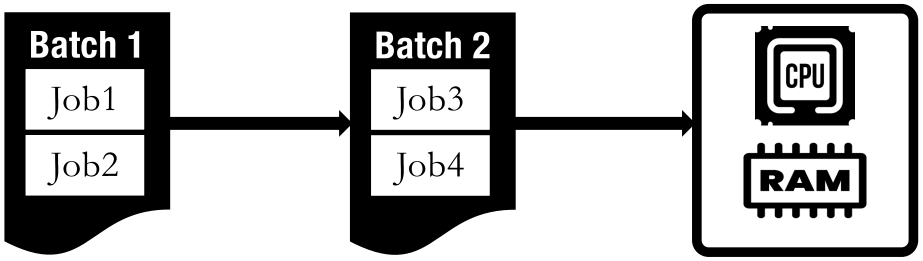
## Batch OS (HỆ ĐIỀU HÀNH XỬ LÝ THEO LÔ)

- A job/program (i.e., set of instructions, data)
  - ✓ Stored on punched cards, which are read by card readers
- A batch (i.e., set of similar jobs)
  - ✓ Executed one by one without user interaction directly

☹ CPU is idle while the current job is waiting for I/O to complete

☹ Lack of user interaction


Punched card



Batch processing

Classification

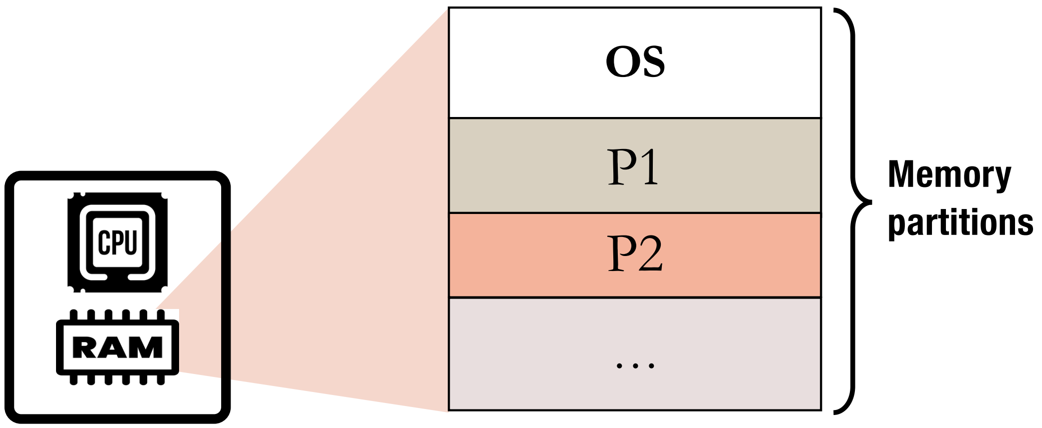
## Multiprogramming OS (HỆ ĐIỀU HÀNH ĐA CHƯƠNG)

- Multiple programs are kept in memory simultaneously
- While the current program is waiting for an I/O operation, the CPU switches to another

☺ Maximize CPU utilization

☹ CPU scheduling?

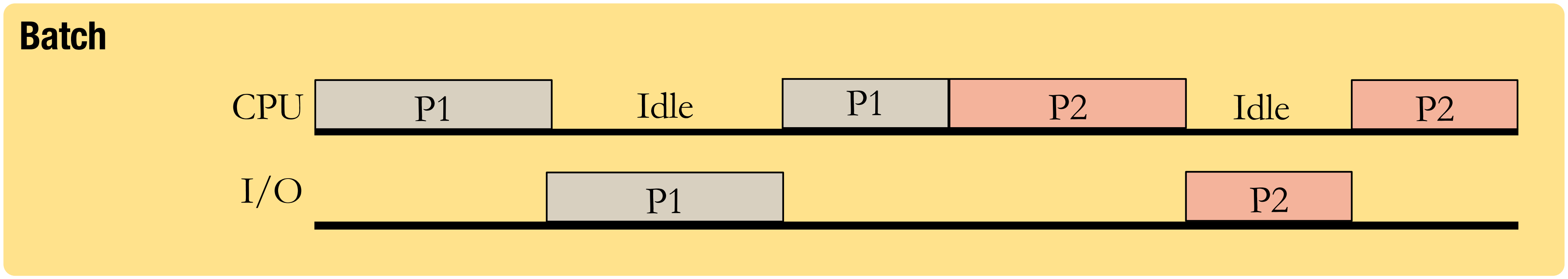
☹ Memory protection?

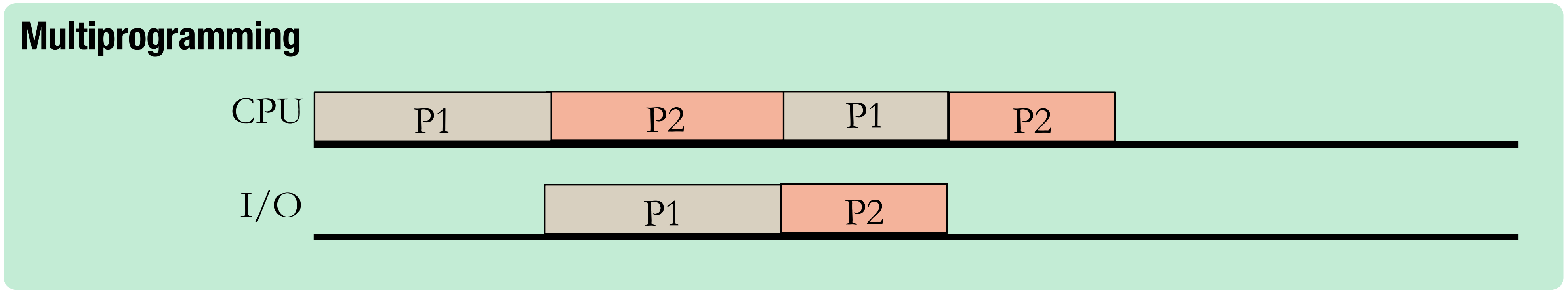


Classification

## Multiprogramming OS

**Batch system vs. Multiprogramming system**





Classification

## Timesharing OS (Multitasking OS)

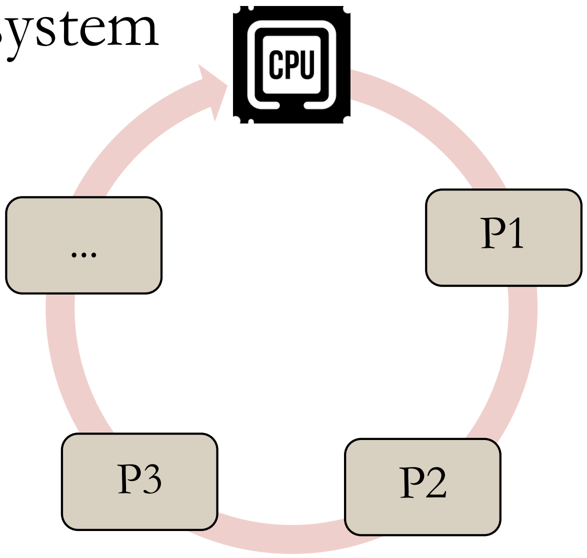
(HỆ ĐIỀU HÀNH CHIA SẺ THỜI GIAN / ĐA NHIỆM)

- An extension of the multiprogramming system
- Switches CPU among various programs
  - ✓ Each program utilizes CPU in a short quantum of time (time slice)
- Requires direct interaction users/systems

☺ Quick response to users

- ✓ Fast enough to give an illusion of **pseudo-parallelism**

☹ More complex CPU scheduling



Classification

## Parallel OS (Multiprocessing OS)

(HỆ ĐIỀU HÀNH SONG SONG / ĐA XỬ LÝ)

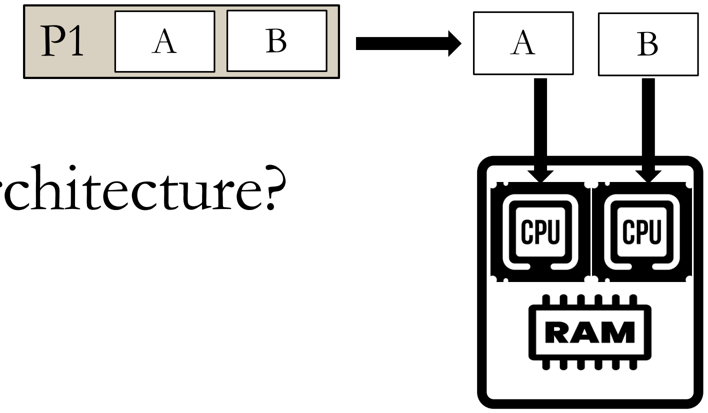
- Runs on multiprocessor computers with shared memory
- Divides a program into multiple activities/jobs, which could be executed on different CPUs simultaneously to speed up the execution

☺ Computing power increase

☺ Reliability

☹ Support of hardware and system architecture?

☹ Parallel computing techniques?



Classification

## Distributed OS (HỆ ĐIỀU HÀNH PHÂN TÁN)

- Consists of independent systems linked via network
- Need for a distributed OS
  - ✓ Resource sharing
  - ✓ Job collaboration
  - ✓ Computing power increase

Classification

## Real-time OS (RTOS) (HỆ ĐIỀU HÀNH XỬ LÝ **THỜI GIAN THỰC)**

- Tasks must complete within time constraint
- Hard real-time system: time constraint must be respected
  - ✓ Time delay causes system failure (i.e., tasks MUST be done on time or fail)
  - ✓ E.g., industrial control systems, traffic control system
- Soft real-time system
  - Time delay accepted occasionally
  - E.g., multimedia system

☹ Limited – Complex - Expensive

Classification

## Embedded OS (HỆ ĐIỀU HÀNH NHÚNG)

- Installed on phones, PDAs, and other devices (not a computer)
- Designed for specific purpose
- May or may not have user interface

☹ Limited resources (low CPU, small memory, no disk, ...)

☹ Complex algorithm

Smartphone

Smartwatch

Vacuum-cleaning robot

iPod

## Plan

- OS: WHAT and WHY?
- Classification
- **Main components**
- OS Structure

(QUẢN LÝ TIẾN TRÌNH VÀ TIỂU TRÌNH/LUỒNG)

## Process (and Thread) Management

- Process and Thread operations (e.g., create, destroy)
  - ✓ Process: a program in execution (Tiến trình)
  - ✓ Thread: a lightweight process, supported in most modern OS (Tiểu trình/Luồng)
- Interprocess Communication (IPC) (Cơ chế liên lạc)
  - ✓ For exchanging data or collaborating to solve a task ...
- CPU Scheduling (Điều phối CPU)
  - ✓ How to allocate CPU among many processes fairly and efficiently?
- Synchronization (Đồng bộ hoá)
  - ✓ What if various processes access a common resource at the same time?

Main components (QUẢN LÝ BỘ NHỚ)

## Memory Management

- In a multiprogramming system, many programs to be executed (i.e., process) are resident in memory simultaneously.
- OS deals with:
  - ✓ Memory allocation, deallocation, and protection (against invalid access)
  - ✓ Virtual memory management
    - o To have more space ...
    - o Programs loaded into memory partially ...

Main components (QUẢN LÝ TẬP TIN VÀ Ổ ĐĨA)

## File & Disk Management

- A file is an abstract concept provided by OS to store collection of data on disk
- OS deals with:
  - ✓ File organization
  - ✓ File allocation/deallocation
  - ✓ File storage on disk

Main components (QUẢN LÝ HỆ THỐNG NHẬP/XUẤT)

## I/O Management

- OS acts as an intermediary between I/O request and physical devices (e.g., mouse, keyboard, screen, printer)
- OS deals with:
  - ✓ I/O hardware communication (e.g., device controller, DMA, polling, interrupt I/O)
  - ✓ I/O software, which allows accessing and managing I/O operations (e.g., device driver, interrupt handler)

Main components

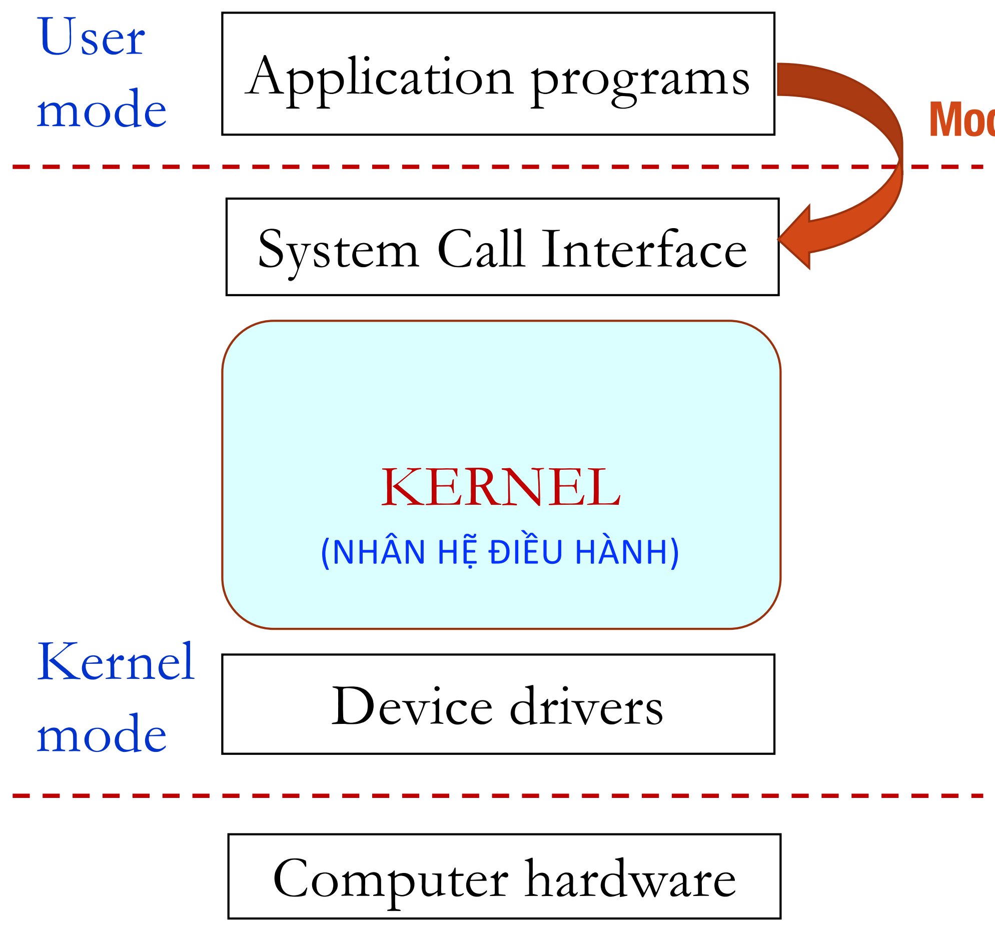
## System Call (LỜI GỌI HỆ THỐNG)

### (System) Programming Interface

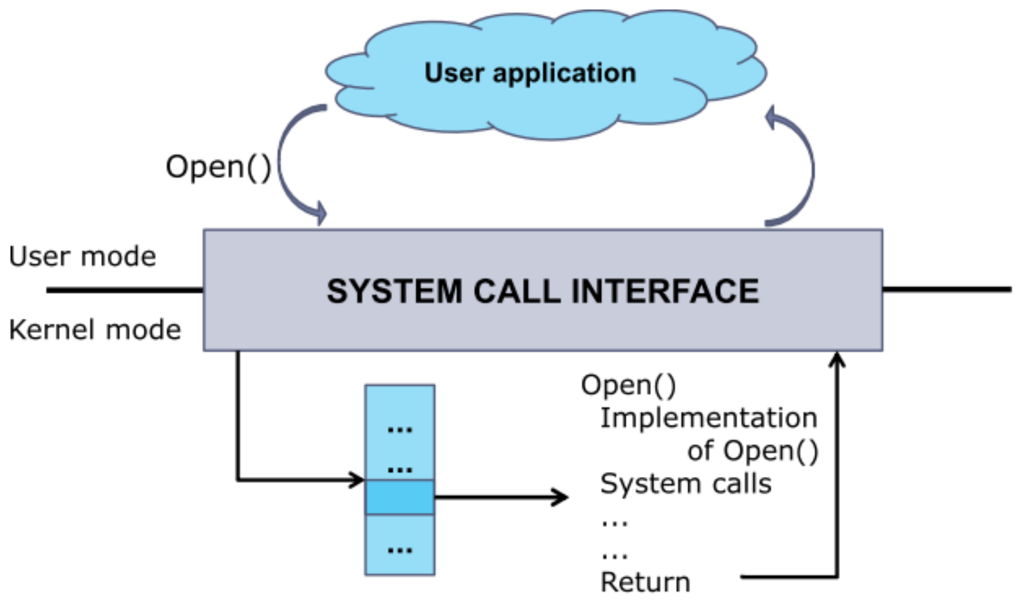
- Provided by OS to facilitate application programming
  - ✓ Win32 API for Windows
  - ✓ POSIX API for UNIX, Linux, and Mac OS
  - ✓ Java API for JVM
- Mostly called by programs via Application Programming Interface (API), i.e., utilities provided by programming language

Main components

## System Call

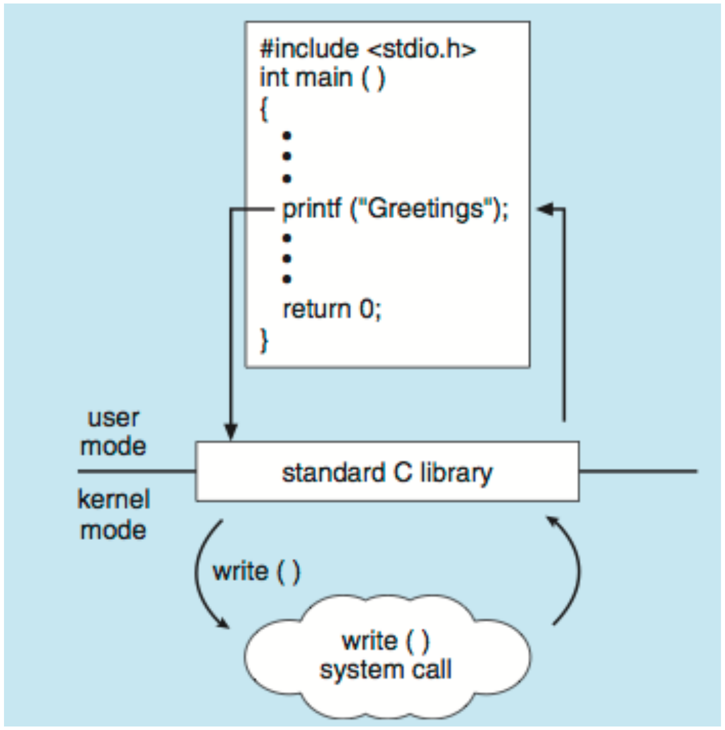


**Mode transition made via a system call**



Main components

## System Call


```
#include <stdio.h>
int main ( )
{
  •
  •
  •
  printf ("Greetings");
  •
  •
  •
  return 0;
}
```

C program calls “printf” function, which invokes “write” system call

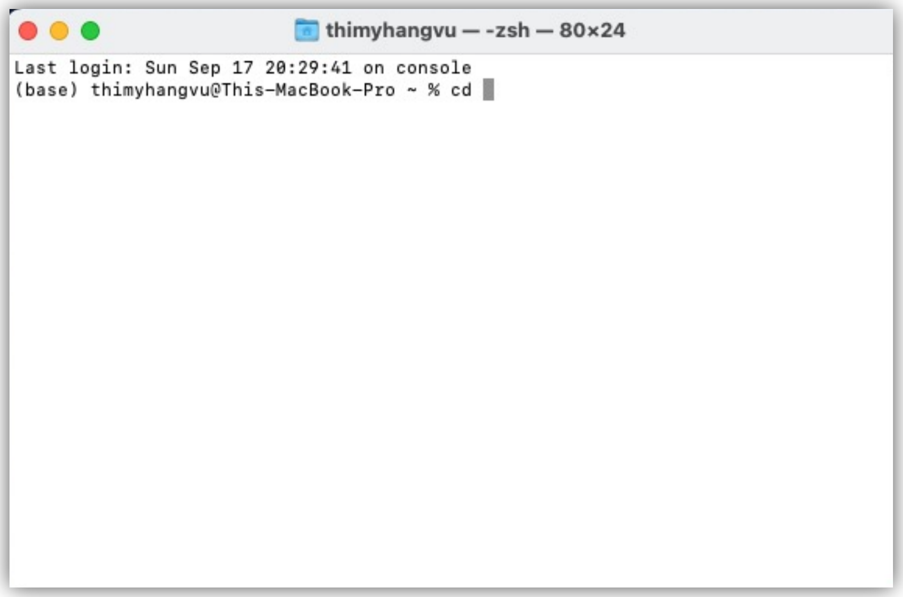
| | Windows | Unix |
|---|---|---|
| Process Control | CreateProcess() ExitProcess() WaitForSingleObject() | fork() exit() wait() |
| File Manipulation | CreateFile() ReadFile() WriteFile() CloseHandle() | open() read() write() close() |
| Device Manipulation | SetConsoleMode() ReadConsole() WriteConsole() | ioctl() read() write() |
| Information Maintenance | GetCurrentProcessID() SetTimer() Sleep() | getpid() alarm() sleep() |
| Communication | CreatePipe() CreateFileMapping() MapViewOfFile() | pipe() shmget() mmap() |
| Protection | SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup() | chmod() umask() chown() |

Windows and Unix System Calls

Main components

## SHELL or Command Interpreter (HỆ THỐNG CƠ CHẾ DÒNG LỆNH)

- Text-based interface for interacting with the operating system.



```
Last login: Sun Sep 17 20:29:41 on console
(base) thimyhangvu@This-MacBook-Pro ~ % cd
```

Main components (HỆ THỐNG BẢO VỆ VÀ BẢO MẬT)

## Protection and Security

- Computer resources (e.g., files, software, hardware) must be protected against insecure access.
  - ✓ Users/programs must have the right to manipulate system objects (e.g., files) or system components (software, hardware).

*Self-learning*

## Plan

- OS: WHAT and WHY?
- Classification
- Main components
- **OS Structure**

OS structure (KIẾN TRÚC ĐƠN GIẢN)

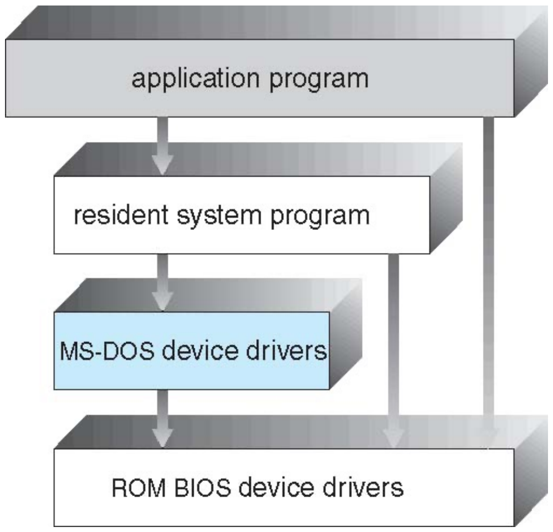
## Simple structure

- OS components are not well separated (e.g., MS-DOS)
- Application programs can access computer hardware directly

☺ Simple and easy to extend

☹ Vulnerable

  ✓ What if an application program fails?



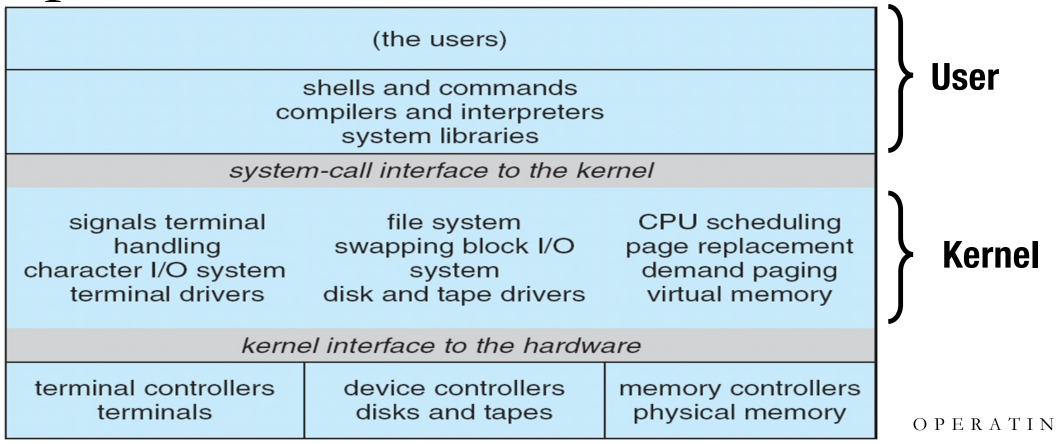
(KIẾN TRÚC MỘT KHỐI)

## Monolithic structure

- OS components are combined into one single module (e.g., traditional UNIX systems)
- Each procedure (function) can call any other procedures

☺ Simple and fast

☹ Difficult to implement and maintain



OS structure (KIẾN TRÚC PHÂN TẦNG)

## Layered structure

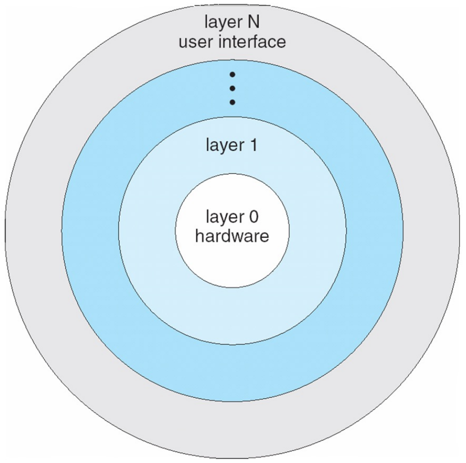
- OS components are structured into layers (e.g., MULTICS)
- Each layer uses services (operations, functions) prepared by its lower adjacent layer
  - ✓ layer N calls services defined in layer N-1

☺ Easier to implement and maintain than monolithic structure

☹ How to group services into layers? based on functionality?

☹ When invoking a system call, a user program may need to pass through many layers

→ efficiency?



(VI NHÂN)

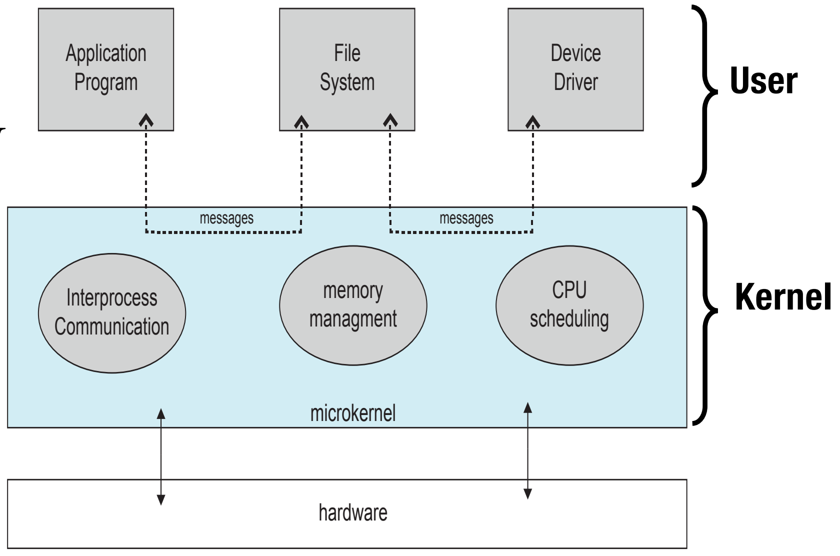
## Microkernel structure

- Kernel only stores essential components of OS (e.g., Mach)
- Other components are implemented as user programs

☺ Easy to manage, extend and build new architecture

☺ Security and Reliability

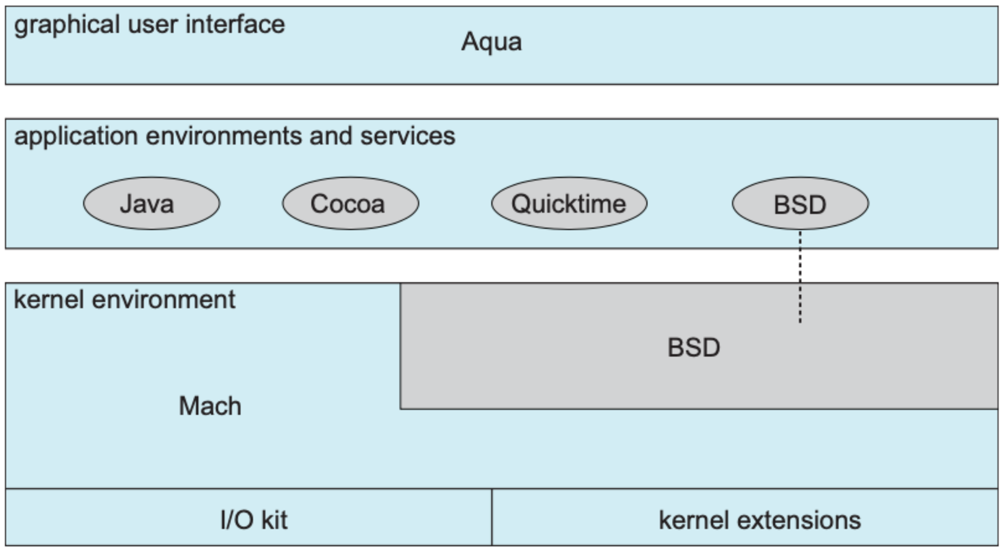
☹ Performance problem occasionally



(KIẾN TRÚC LAI)

## Hybrid structure

- A combination of various structures to address performance, security and usability
- Common design in many modern OS (e.g., Linux, Mac OS X, Windows, Android)

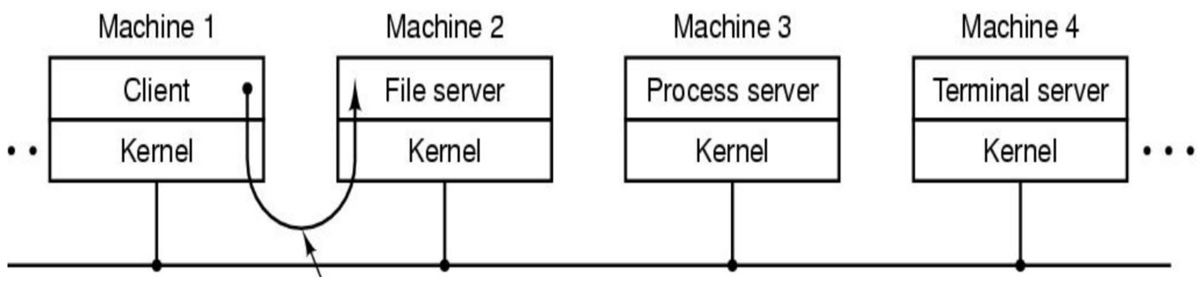


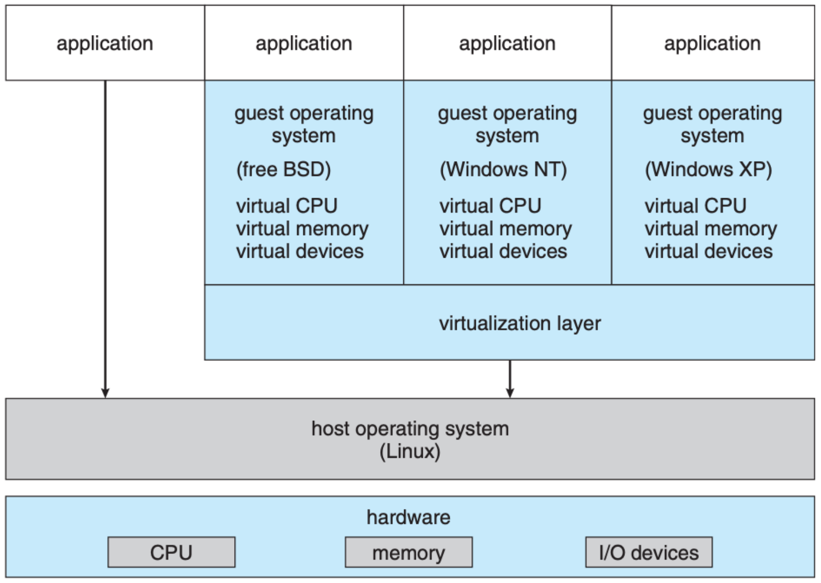
Hybrid system (Mac OS X)

OS structure

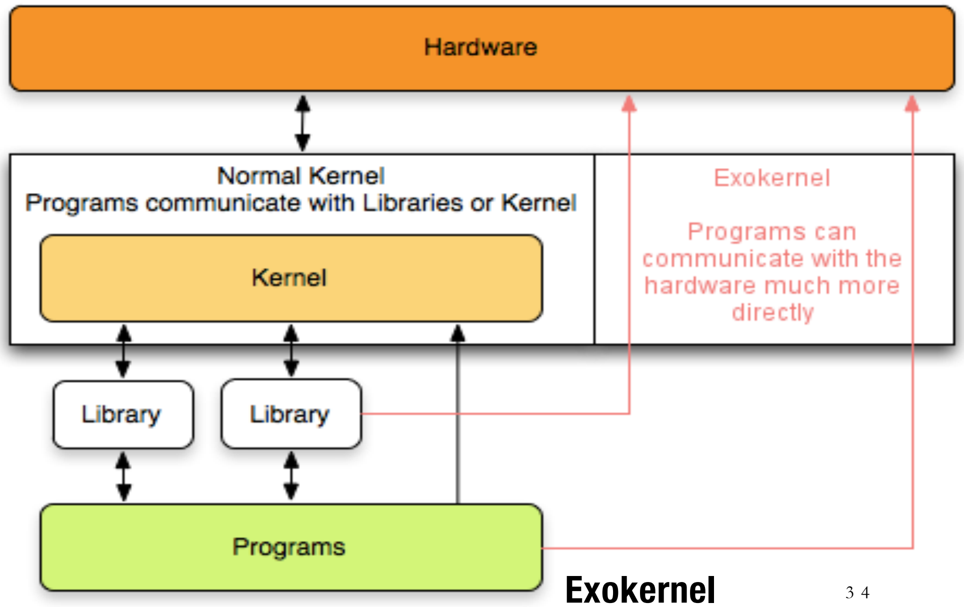
## And other ...

Client-Server





Virtual machine


Hardware
Normal Kernel
Programs communicate with Libraries or Kernel
Kernel
Exokernel
Programs can communicate with the hardware much more directly
Library
Library
Programs

Exokernel

## References

Operating System Concepts (8th Edition), *Silberschatz and Galvin*

Modern Operating Systems (4th Edition), *Andrew S. Tanenbaum*

Giáo trình Hệ điều hành, *HCMUS-FIT*