Chương 1 : Tổng quan về HĐH

Hệ điều hành là Chương trình trung gian giữa phần cứng máy tính và người sử dụng, có chức năng điều khiển và phối hợp việc sử dụng phần cứng và cung cấp các dịch vụ cơ bản cho các ứng dụng.

■ **Mục tiêu**

- Giúp người dùng dễ dàng sử dụng hệ thống.

Quản lý và cấp phát tài nguyên hệ thống một cách hiệu quả

■ **Vai trò(roles)**:

- Virtual machine(user view): ảo hóa phần cứng máy tính và cung cấp các khái niệm trừu tượng giúp dễ dàng thao tác với phần cứng. OS cung cấp một giao diện để người dùng tương tác với máy tính, thường qua giao diện dòng lệnh (CLI) hoặc giao diện đồ họa (GUI).
- Resource Manager: Quản lý tài nguyên (CPU, Manager,..) giữa các chương trình khác nhau trong sự quản lý fair, effective, safe (cấp phát, thu hồi, chia sẻ, bảo vệ).
- Hỗ trợ giao tiếp với máy tính thông qua các lời gọi hệ thống: Tạo điều kiện và đơn giản hóa việc lập trình ứng dụng. Thực thi và quản lý phần mềm

Phân loại:

■ Batch OS (hđh xử lý theo lô/ đơn chương)

- Job/Program/ tệp lệnh được lưu trữ bằng thẻ.
- Tác vụ thi hành tuần tự, bộ giám sát thường trực

A batch executed one by one without user interaction directly

Nhược: CPU is rảnh while the current job is waiting for I/O to complete, Lack of user interaction

■ Multiprogramming OS (Đa chương)

• Multiple programs are kept in memory simultaneously[đồng thời]

• While the current program is waiting for an I/O operation, the CPU switches to another

yêu cầu: job scheduling, memory management, CPU scheduling, cấp phát tài nguyên, protect

Ưu: tận dụng thời gian rảnh, tăng hiệu suất CPU

Nhược:

- CPU scheduling: starvation, context switching->overhead
- Memory protection: Fragmentation,

■ Time-Sharing OS/Multitasking OS: hệ thống chia sẻ thời gian

- Một phiên bản mở rộng của multiprogramming system
- Switches CPU among various programs (Each program sử dụng CPU in a short quantum of time (time slice))
- Requires direct interaction users/systems

* **Yêu cầu đối với OS trong hệ thống time-sharing**

Job scheduling, memory management (Virtual memory)

Quản lý các quá trình (process management)

-Định thời CPU -Đồng bộ các quá trình (synchronization)

-Giao tiếp giữa các quá trình (process communication) -Tránh deadlock

Quản lý hệ thống file, hệ thống lưu trữ

Cấp phát hợp lý các tài nguyên

Bảo vệ (protection

Ưu: Quick response to users (nhanh đủ để cảm thấy về sự song song giả giữa các tiến trình

Nhược: More complex CPU scheduling

■ Parallel OS (Multiprocessing OS)

• Chạy trên máy tính đa bộ xử lý(multiprocessor) có bộ nhớ dùng chung
• Divides a program into multiple activities/jobs, which could be executed on different CPUs simultaneously[đồng thời] để tăng tốc độ thực hiện.
Ưu :Computing power increase, Reliability
Nhược:
- Support for hardware and system architecture?
- Parallel computing techniques?
■ Distributed OS Hệ điều hành phân tán
- Bao gồm nhiều hệ thống độc lập được liên kết với nhau thông qua Internet
- Những điều cần có Distributed OS: Chia sẻ tài nguyên, Job collaboration, Computing power increase, high reliability, high availability
■ Real-time OS (HĐH xử lý thời gian thực)
- Nhiệm vụ phải hoàn thành trong thời gian giới hạn
- Hệ thống thời gian thực cứng: phải tôn trọng ràng buộc về thời gian (industrial control systems, traffic control system)
-Hệ thống thời gian thực mềm: Thỉnh thoảng chấp nhận độ trễ thời gian(multimedia system)
Nhược điểm: Limited – Complex - Expensive
■ Embedded OS(HĐH nhúng) smartphone, smartwatch, robot lau nhà, ipod
- Được cài đặt trên điện thoại, PDA và các thiết bị khác (không phải máy tính)
- Được thiết kế cho mục đích cụ thể
- Có thể có hoặc không có giao diện người dùng
Hạn chế: Giới hạn tài nguyên (CPU thấp, bộ nhớ nhỏ, ko ổ đĩa,..)
Thuật toán phức tạp
■■ **Những thành phần chính của HĐ**H:
■ Process and Thread Management:
- Tiến trình là chương trình đang thực thi . Luồng/Tiểu trình là một quy trình nhẹ, được hỗ trợ trong hầu hết các hệ điều hành hiện đại.
- Cơ chế liên lạc (IPC) : Để trao đổi dữ liệu hoặc cộng tác để solve a task
- CPU Scheduling: Cấp phát tài nguyên giữa các processes fairly and efficiently
- Synchronization(đồng bộ hóa) Điều gì xảy ra nếu nhiều tiến trình truy cập tài nguyên cùng lúc.
■ Memory management (quản lý bộ nhớ): Trong một hệ thống đa chương trình, nhiều chương trình được thực thi (tức là tiến trình) được lưu trữ đồng thời trong bộ nhớ.
- Hệ điều hành xử lý: Cấp phát, giải phóng và bảo vệ bộ nhớ (chống truy cập không hợp lệ)
Quản lý bộ nhớ ảo: Để có nhiều không gian hơn. Các chương trình được tải vào bộ nhớ một phần
■ File & Disk Management
-Tệp là một khái niệm trừu tượng được hệ điều hành cung cấp để lưu trữ bộ sưu tập dữ liệu trên đĩa
- OS deals with: File organization, File allocation/deallocation, File storage on disk
■ I/O Management
- HĐH đóng vai trò trung gian giữa yêu cầu I/O và các thiết bị vật lý (ví dụ: chuột, bàn phím, màn hình, máy in)
-OS deals with: I/O hardware communication((e.g., device controller, DMA, polling, interrupt I/O software, which allows accessing and managing I/O operations (e.g., device driver, interrupt handler)

■ System Call: Dùng để giao tiếp giữa tiến trình và hệ điều hành. Cung cấp giao diện giữa tiến trình và hệ điều hành
- Được cung cấp bởi hệ điều hành để hỗ trợ lập trình ứng dụng
-Chủ yếu được gọi bởi các (API), tức là các tiện ích được cung cấp bởi ngôn ngữ lập trình
3 phương thức truyền tham số khi sử dụng SC: qua thanh ghi, vùng nhớ, stack
■ SHELL or Command Interpreter: Giao diện dựa trên văn bản để tương tác với hệ điều hành
■Protection and Security: Tài nguyên máy tính (ví dụ: tệp, phần mềm, phần cứng) phải được bảo vệ khỏi truy cập không an toàn.
- Người dùng/chương trình phải có quyền thao tác với các đối tượng hệ thống (ví dụ: tệp) hoặc các thành phần hệ thống (phần mềm, phần cứng)
■ ■**OS structure**
■ Simple structure
- Các thành phần OS không được tách biệt rõ ràng (ví dụ: MS-DOS)
- Các chương trình ứng dụng có thể truy cập trực tiếp vào phần cứng máy tính: không có user mode hay kernel mode.
Ưu: Đơn giản và dễ mở rộng
Nhược Dễ bị tổn thương
- Điều gì sẽ xảy ra nếu một chương trình ứng dụng bị lỗi?
+ Crash hoặc Hàng Hệ Thống, mất dữ liệu, Vấn Đề Bảo Mật:, Tác Động đến Các Tiến Trình Khác,Khó Khăn Trong Việc Điều Tra và Sửa Lỗi
■ Monolithic structure(Kiến trúc một khối) :
- Các thành phần hđh được kết hợp thành một mô-đun duy nhất (ví dụ: hệ thống UNIX truyền thống) . Có mode user , an toàn do đã có phân lớp.
- Mỗi thủ tục (chức năng) có thể gọi bất kỳ thủ tục nào khác
- Ưu: Đơn giản và nhanh chóng
- Nhược: Khó triển khai và bảo trì (phải xử lý nguyên cái layer)
[UIT} UNIX gồm 2 phần: Nhân (cung cấp file system, CPU scheduling, memory management, và một số chức năng khác) và System Program
■ Layered structure(Kiến trúc phân tầng): Các thành phần hđh được cấu trúc thành các lớp (ví dụ: MULTICS) lớp cuối cùng là hardware, trên cùng là user
- Mỗi lớp sử dụng các dịch vụ (thao tác, chức năng) được chuẩn bị bởi lớp liền kề bên dưới của nó (lớp N gọi các dịch vụ được xác định trong lớp N-1)
- Ưu: Dễ dàng thực hiện và bảo trì hơn cấu trúc nguyên khối
- Khó khăn: Làm cách nào để nhóm các dịch vụ thành các lớp? dựa trên chức năng?
+ Khi gọi một lời gọi hệ thống, một chương trình người dùng có thể cần phải trải qua nhiều lớp à, hiệu quả không ?
■Microkernel structure (Vi nhân) Kernel chỉ lưu trữ các thành phần thiết yếu của HĐH (ví dụ: Mach). Thu gọn kernel => microkernel, microkernel chỉ bao gồm các chức năng tối thiểu như quản lý tiến trình, bộ nhớ và cơ chế giao tiếp giữa các tiến trình
- Các thành phần khác được triển khai dưới dạng chương trình người dùng
Ưu: Dễ dàng quản lý, mở rộng và xây dựng kiến trúc mới, Bảo mật và Độ tin cậy
Nhược: Đôi khi có vấn đề về hiệu suất
■Hybrid structure(Kiến trúc lai): - Sự kết hợp của nhiều cấu trúc khác nhau để giải quyết vấn đề hiệu suất, bảo mật và khả năng sử dụng
- Thiết kế phổ biến trong nhiều hệ điều hành hiện đại (ví dụ: Linux, Mac OS X, Windows, Android)

■ Other: Client-server, virtual machine, exokernel

Chương 2: Process & Inter-process communication

■■ **Tiến trình (process)** là Một chương trình đang thực thi

■ **Chương trình** là thực thể bị động lưu trên đĩa (tập tin thực thi - executable file); tiến trình là thực thể chủ động. Chương trình trở thành tiến trình khi một tập tin thực thi được nạp vào bộ nhớ.

Tiểu trình là một đơn vị cơ bản sử dụng CPU gồm: Thread ID, PC, Registers, Stack và chia sẻ chung code, data, resources (files)

Process control block(PCB) Các tính năng của quy trình: Cấu trúc dữ liệu cụ thể:- Chứa thông tin quy trình. - Nằm trong không gian địa chỉ hệ điều hành và chỉ có thể truy cập được ở chế độ kernel

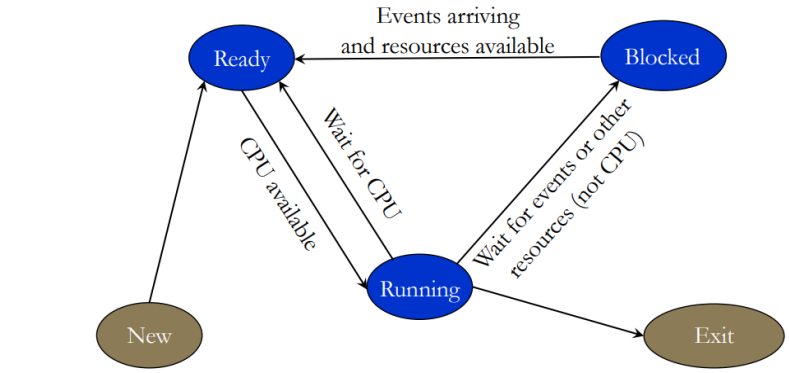
Trạng thái của triến trình: New(Được tạo lập), Ready(nằm trong hàng đợi chờ được thực thi), Running(đang được thực thi), Blocked(Khi đang running nhưng phải đợi tài nguyên khác, hoàn tất thì quay về Ready), Exit(thu hồi tài nguyên)

Vai trò của PCB và xác thực phát biểu:

- PCB (Process Control Block) chứa thông tin cần thiết để quản lý tiến trình như trạng thái tiến trình, thông tin thanh ghi, thông tin lập lịch và thông tin về quản lý bộ nhớ.
- Phát biểu "A PCB of a process resides[nằm] in memory zone reserved[dành riêng] for this process" là sai. PCB thường được lưu trữ trong một khu vực bộ nhớ được quản lý bởi hệ điều hành, không phải trong không gian bộ nhớ được phân bổ cho tiến trình mà nó mô tả.

Process features

Process state (Trạng thái của tiến trình)



Process identifier	A unique number (at a time t) to identifier a process
Process state	Current state of the process (running, ready, blocked)
Program counter	Indicate the address of the next instruction to be executed
CPU registers	Information for reloading the process correctly
CPU scheduling information	Process priority, pointers to scheduling queues, ...
Memory management information	Information for locating the process in memory
Accounting information	Information for statistics and system performance
I/O status information	Information about I/O devices, files, ... allocated to the process

■**Yêu cầu của HĐH đối với quản lý tiến trình:**

- Hỗ trợ sự thực thi luân phiên giữa nhiều tiến trình
- Hiệu suất sử dụng CPU
- Thời gian đáp ứng

+Phân phối tài nguyên hệ thống hợp lý

+Tránh deadlock, trì hoãn vô hạn định

+ Cung cấp cơ chế giao tiếp và đồng bộ hoạt động các tiến trình

+ Cung cấp cơ chế hỗ trợ user tạo/kết thúc tiến trình

■ ■ **process creation:**

■ Một tiến trình cha tạo ra các tiến trình con, do đó, các tiến trình này có thể tạo ra các tiến trình con của chính chúng: System calls: fork (Unix), CreateProcess (Windows)

■ Một tiến trình con có thể: - là bản sao của tiến trình cha (tức là có cùng không gian bộ nhớ với tiến trình cha) - tải một chương trình khác để thực thi: System call: exec (Unix)

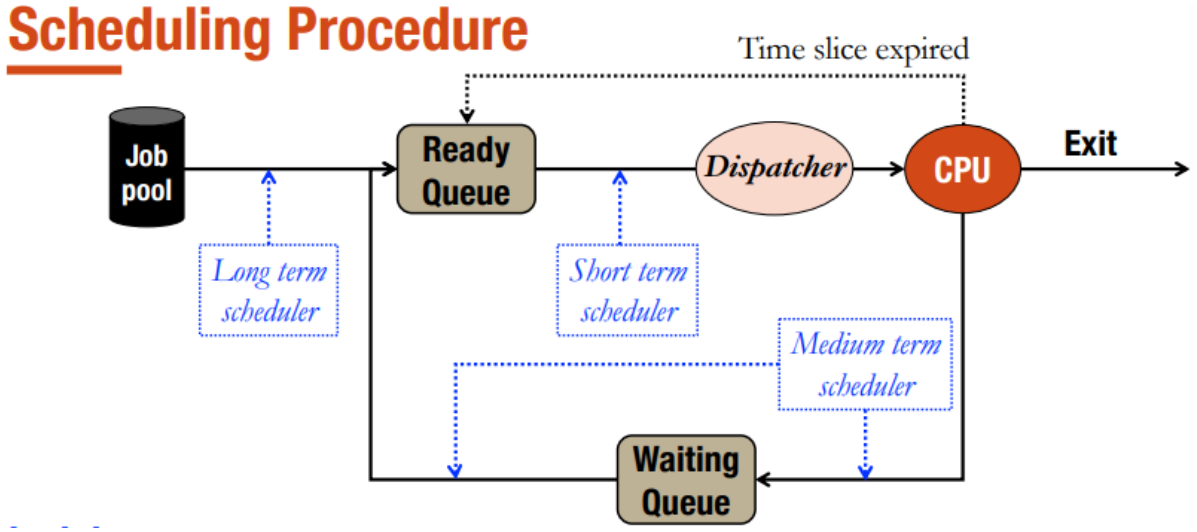
■ Khi một quy trình mới được tạo: - Nó sẽ được cấp một mã định danh quy trình (pid) và các

tài nguyên khác (không gian nhớ)
- Một PCB chứa thông tin của quy trình mới sẽ được tạo và đặt trong Bảng quy trình
■ ■ **process termination:**
■ Một tiến trình có thể chấm dứt khi kết thúc quá trình thực thi của nó hoặc bị chấm dứt bởi các tiến trình mẹ hoặc hệ điều hành của nó System call: exit (Unix), ExitProcess (Windows)
■ **Quá trình zombie:** Một con chấm dứt nhưng PCB của nó vẫn chưa được phát hành- > Cha mẹ phải "đợi" con mình hoàn thành. Để giải quyết, tiến trình cha cần gọi System call: wait (Unix), WaitForSingleObject (Windows) để thu thập thông tin và giải phóng tài nguyên.
■ Những tiến trình mồ côi: TT cho kết thúc nhưng children vẫn hiện hữu ->các tiến trình mồ côi -> được gán cho "tiến trình gốc" (thường là init) để quản lý việc giải phóng tài nguyên khi nó kết thúc.
- Chấm dứt xếp tầng
■ ■ **Cơ chế liên lạc/ giao tiếp giữa các tiến trình**: IPC provides the way in which processes communicate to each other.
■ **Pipe** : Truyền thông một chiều (Sending data in a one-way direction).Consists of two processes: writer and reader. Two kinds of pipes:
- Các đường ống thông thường (i.e các đường ống ẩn danh): hai quy trình giao tiếp phải có mối quan hệ cha-con
- Các đường ống được đặt tên:hai quá trình giao tiếp không cần phải có mối quan hệ cha con
■ **Shared Memory**: Các tiến trình thực hiện giao tiếp thông qua bộ nhớ dùng chung
- Bộ nhớ dùng chung nằm trong không gian địa chỉ của tiến trình đã tạo ra nó
- Nói chung, việc chia sẻ bộ nhớ cần có cơ chế đồng bộ hóa
■ **Socket**: Communication endpoints between processes on distant machines(Based on client-server mode)
• A socket is identified by an IP address and a port number
- IP address: nhận dạng máy tính qua mạng
- Port number: nhận dạng chương trình trên máy tính

Connectionless (UDP) socket
- ✓ No connection required
- ✓ Data transmitted as datagrams
- ✓ Loss data possible
- ✓ Fast
- ✓ Less reliable than TCP socket

• Connection-oriented (TCP) socket
- ✓ A logical connection required during data transfer
- ✓ Data transmitted as streams of bytes
- ✓ Re transmission
- ✓ Slower than UDP socket
- ✓ Reliable

CPU – I/O Burst Cycle (CHu kỳ CPU và Chu kỳ Nhập/Xuất)
Chương 3: CPU Scheduling
• Quá trình thực hiện xen kẽ giữa:: CPU Burst thực hiện các phép tính. I/O Burst performing I/O operations
■ Scheduler(Bộ đình thời): Bộ lập lịch là một thành phần hệ điều hành liên quan đến các hoạt động lập lịch, bao gồm:
- Long term scheduler (Job scheduler): chọn các công việc từ nhóm công việc và tải chúng vào bộ nhớ (hàng đợi sẵn sàng) để thực thi
- Short term scheduler (CPU scheduler): chọn một quy trình để được thực thi trong số các quy trình đã sẵn sàng
- Medium-term scheduler: xử lý trao đổi trong và trao đổi các tiến trình



■ Dispatcher (Bộ điều phối): Phân bổ CPU cho quy trình được chọn bởi Bộ lập lịch ngắn hạn
■ Scheduling criteria(Tiêu chí lập kế hoạch): Định thời CPU (CPU scheduling) là một quá trình quan trọng trong hệ điều hành nhằm phân phối thời gian CPU một cách công bằng và hiệu quả giữa các tiến trình hoặc các tác vụ khác nhau. Việc định thời này giúp tối ưu hóa việc sử dụng CPU, cải thiện thời gian phản hồi, giảm thời gian chờ và tăng hiệu suất tổng thể của hệ thống.
• Fairness and efficiency (Công bằng và Hiệu quả)
• Maximize CPU utilization (Tối đa hoá sử dụng CPU)
• Maximize throughput (Cực đại hoá thông lượng)
- Throughput là số tiến trình được hoàn thành trong một đơn vị thời gian
• Giảm thiểu: turnaround time, waiting time, and response time
- Turnaround time là khoảng thời gian từ khi gửi quy trình đến khi hoàn thành quy trình
-Waiting time là khoảng thời gian ready queue
- Response time là khoảng thời gian từ thời điểm gửi quy trình đến thời điểm tạo ra phản hồi đầu tiên
■**Scheduling Types**
- In **Non-Preemptive scheduling** (Độc quyền): tiến trình đang chạy giữ CPU cho đến khi nó kết thúc hoặc chuyển sang trạng thái chờ (bằng cách hoàn thành I/O hoặc một sự kiện)
- In **Preemptive scheduling** (i.e., cooperative scheduling):Quá trình đang chạy có thể bị hệ điều hành tạm dừng do có quá trình có mức độ ưu tiên cao hơn đến hoặc phần thời gian của nó đã hết hạn
■**Chuyển ngữ cảnh(Context Switch)**:Quy trình chuyển đổi ngữ cảnh là quá trình lưu trữ trạng thái của một tiến trình đang thực thi để sau đó hệ điều hành có thể nạp và tiếp tục một tiến trình khác. Điều này bao gồm lưu các thanh ghi CPU, bộ đếm chương trình và thông tin quản lý tài nguyên.
■■ **Thuật toán CPU scheduling**
■First-Come, First-Served (FCFS) Scheduling
• Nonpreemptive scheduling
• Pick the first process standing in ready queue for execution
Ưu điểm: đơn giản, hiệu quả thấp
Nhược điểm: Không quan tâm tới mức độ ưu tiên
■Round-Robin (RR) Scheduling

• Preemptive scheduling
    - Especially used in **timesharing systems**
• Pick the first process standing in ready queue for execution within a given time quantum q (time slice). After the time quantum, if the active process has not finished yet, it will move to the end of the ready queue for the next execution
Ưu điểm: Công bằng, giảm waiting time của CPU-burst processes ngắn
Nhược điểm: ko độ ưu tiên, vấn đề xác định time quantum
■ Shortest-Job-First (SJF) Scheduling
• Nonpreemptive scheduling
  -> Preemptive version: Shortest-Remaining-Time-Next (SRTN) Scheduling
• Pick the process having the smallest CPU-bursts time
Ưu điểm: Đơn giản, giản TT và WT của những tiến trình ngắn, tăng thông lượng
Nhược điểm : Starvation
■ Priority Scheduling
• preemptive and nonpreemptive scheduling (Especially used in timesharing systems)
• Each process is associated with a priority
• Pick the process having the highest priority in the ready queue
- Preemptive version: the running process will be stopped immediately
- Non-preemptive version: the running process finishes its task then the highest priority process will be allocated the CPU
Ưu điểm: Có quan tâm đến độ ưu tiên
Nhược điểm : Starvation
■ Multilevel Priority Queue Scheduling: Separate the ready queue into multiple priority queues
• Each priority queue is associated with a priority level
  -> Each queue has its scheduling algorithms
• Each process is assigned to one queue
• Pick processes presenting in higher priority queue first
  ->Process selection is based on the scheduling algorithm applied to this queue
■Problem of Starvation
Những thuật toán có thể dẫn đến tình trạng đói CPU là: SJF, SRTN, Priority (Preemptive và Non-preemptive).
- Nguyên nhân : Một tiến trình có độ ưu tiên thấp có thể bị khóa vô thời hạn bởi vì những tiến trình có mức độ ưu tiên cao hơn giữ CPU.
- Giải pháp: Aging Là kỹ thuật tăng dần mức độ ưu tiên của các tiến trình đang đợi trong hệ thống trong một thời gian dài.

■Multilevel Priority Queue: Điều phối với hàng đợi đa cấp, chia ready queue thành nhiều hàng đợi với độ ưu tiên khác nhau. Mỗi queue sẽ sử dụng 1 thuật toán phù hợp. Mỗi tiến trình được gắn với 1 queue. Chọn process ở hàng đợi có độ ưu tiên cao hơn trước và việc chọn process nào trong queue phụ thuộc vào thuật toán của queue.

- Khác: Guaranteed scheduling, Lottery scheduling, Fair-Share scheduling
- Context Switch:
- Sự khác biệt giữa đa chương trình (multiprogramming), đa nhiệm (multitasking) và đa xử lý (multiprocessing):

+ Đa chương trình là khả năng của một hệ điều hành để giữ nhiều chương trình trong bộ nhớ đồng thời, cho phép chuyển đổi giữa chúng để tối ưu hóa việc sử dụng CPU.
+ Đa nhiệm là khả năng thực hiện nhiều công việc (tasks) cùng một lúc. Hệ điều hành chia nhỏ thời gian CPU và gán cho các tiến trình khác nhau một cách nhanh chóng, tạo ảo giác rằng mọi tiến trình đều đang chạy đồng thời.
+ Đa xử lý là việc sử dụng nhiều CPU trong một hệ thống máy tính để thực hiện công việc. Các CPU có thể chạy cùng một hoặc khác chương trình đồng thời, tăng cường hiệu suất xử l