Hệ điều hành

I. Chương 1: Tổng quan về hệ điều hành

1. Tổng quan cơ bản
2. Phân loại hệ điều hành

II. Chương 2: Cấu trúc hệ điều hành

1. Các thành phần của hệ điều hành
2. Các dịch vụ hệ điều hành cung cấp
3. Lời gọi hệ thống
4. Các chương trình hệ thống
5. Cấu trúc hệ thống

III. Chương 3: Tiến trình

1. Các khái niệm cơ bản
2. Trạng thái tiến trình
3. Khối điều khiển tiến trình (PCB)
4. Định thời tiến trình
5. Các tác vụ đối với tiến trình

IV. Chương 4: Định thời CPU

1. Các khái niệm và các loại bộ định thời
2. Các tiêu chuẩn định thời CPU
3. Các yếu tố của giải thuật định thời
4. Các giải thuật định thời

CHƯƠNG I

Tổng quan về hệ điều hành

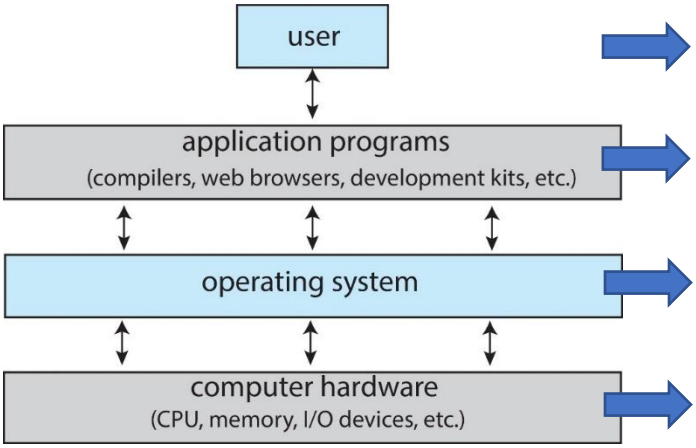
I.1 Tổng quan cơ bản

Hệ điều hành:

- **Khái niệm:** 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.
- **Lợi ích:**
  - Quản lý phần cứng máy tính
  - Cung cấp giao diện cho người dùng
  - Là nơi để người dùng cài đặt các chương trình ứng dụng
  - Kết nối các thiết bị phần cứng với nhau
  - Tương tác giữa các chương trình với nhau và với phần cứng
- **Chức năng:**
  - Phân chia thời gian xử lý và định thời CPU
  - Phối hợp và đồng bộ hoạt động giữa các processes (coordination & synchronization)
  - Quản lý tài nguyên hệ thống (Thiết bị I/O, bộ nhớ, file chứa dữ liệu,..)
  - Kiểm soát truy cập, bảo vệ hệ thống
  - Duy trì sự nhất quán (integrity) của hệ thống, kiểm soát lỗi và phục hồi hệ thống khi có lỗi (error recovery)
  - Cung cấp giao diện làm việc cho users

⇨ *Nói một cách dễ hiểu, hệ điều hành như một người quản lý, nó quản lý toàn bộ các thành phần của máy tính để mà chúng ta có thể dễ dàng sử dụng chúng.*

Cấu trúc hệ thống máy tính



Users(people, machines, other computers)

Sử dụng hệ thống tài nguyên để giải quyết một bài toán tính toán nào đó của người sử dụng
VD: compliers, database systems, video games,..

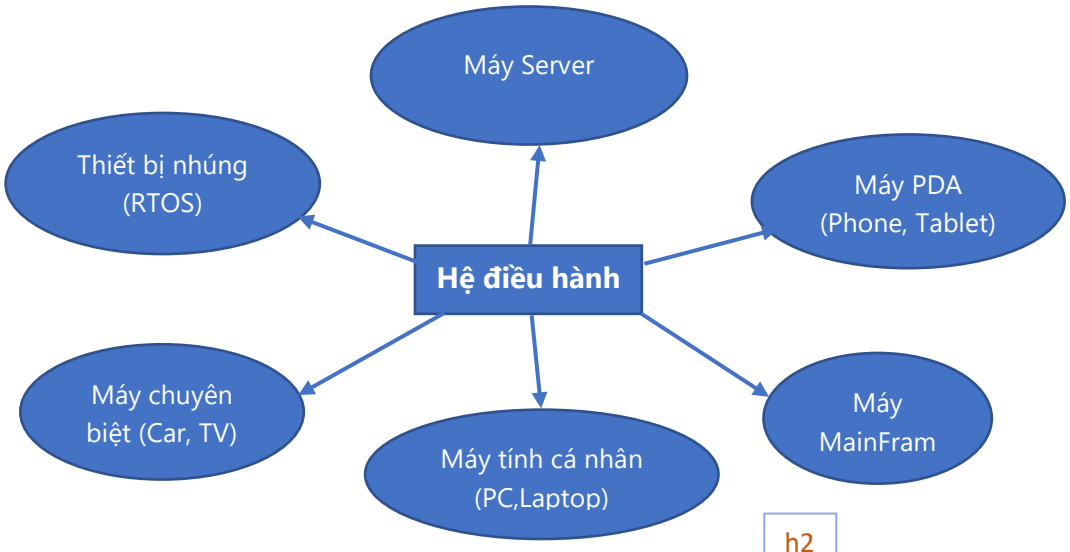
Hệ điều hành
Phân phối tài nguyên điều khiển và phối hợp các hoạt động của các chương trình trong hệ thống

Bao gồm các tài nguyên cơ bản của máy tính như CPU, bộ nhớ, các thiết bị I/O

h1

I.2 Phân loại hệ điều hành

Dưới góc độ loại máy tính

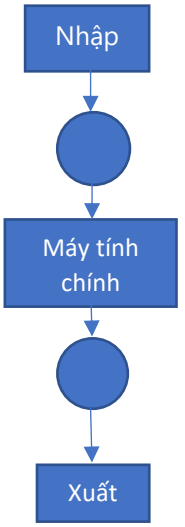


h2

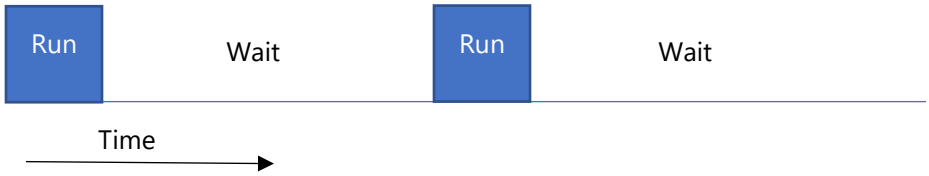
Dưới góc độ hình thức xử lý

❖ Hệ thống xử lý theo chương trình

- *Hệ thống đơn chương (uniprograming OS)*
  - Tác vụ được thi hành tuần tự
  - Bộ giám sát thường trực
  - CPU và các thao tác nhập xuất (*h3*)
    - Xử lý offline
    - Đồng bộ hóa các thao tác bên ngoài – Spooling (Simultaneous Peripheral Operation On Line)



h3

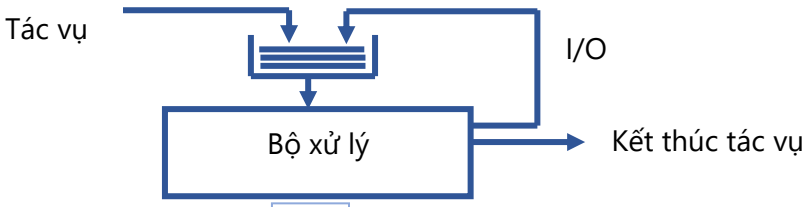


h4. Hệ điều hành đơn chương

- *Hệ thống đa chương (multiprogramming OS)*
  - Nhiều công việc được nạp đồng thời vào bộ nhớ chính(*h5*)
  - Khi một tiến trình thực hiện I/O, một tiến trình khác được thực thi (*h6*)

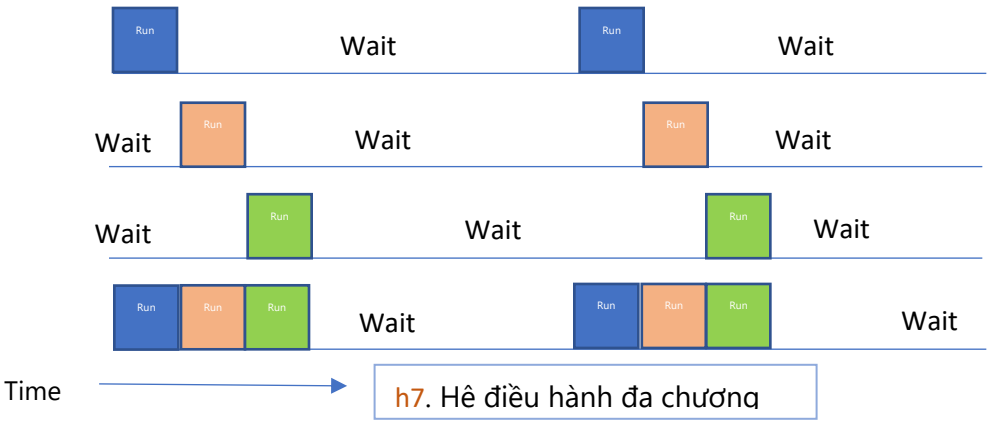
⇨ Tận dụng được thời gian rảnh, tăng hiệu suất sử dụng CPU

h5



h6

- Yêu cầu đối với hệ điều hành:
  - Định thời công việc
  - Quản lý bộ nhớ
  - Định thời CPU
  - Cấp phát tài nguyên (đĩa,máy in,..)
  - Bảo vệ



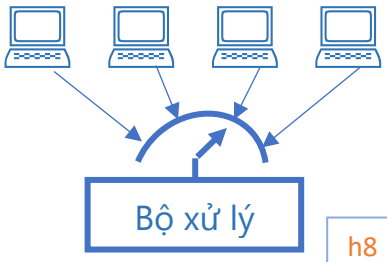
h7. Hệ điều hành đa chương

❖ **Hệ thống chia sẻ thời gian**

⇨ *Mở rộng của hệ thống đa chương*

- Mỗi công việc chạy trong khoảng thời gian nhất định
- **Các yêu cầu**: tương tự hệ thống đa chương

  Ngoài ra,
  - Quản lý các quá trình
    - Đồng bộ giữa các công việc
    - Giao tiếp giữa các công việc
    - Tránh deadlock
  - Quản lý hệ thống file, hệ thống lưu trữ



h8

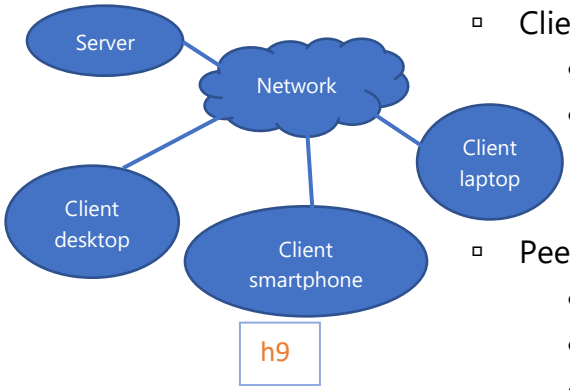
❖ **Hệ thống song song**

- Nhiều bộ xử lý cùng chia sẻ một bộ nhớ
- Master/Slave: một bộ xử lý chính kiểm soát một bộ xử lý I/O
- Ưu điểm:
  - Năng suất cao (do nhiều công việc được xử lý đồng thời)
  - Sự hỏng hóc của một bộ xử lý không ảnh hưởng đến toàn bộ hệ thống
- Gồm có đa xử lý đối xứng và bất đối xứng

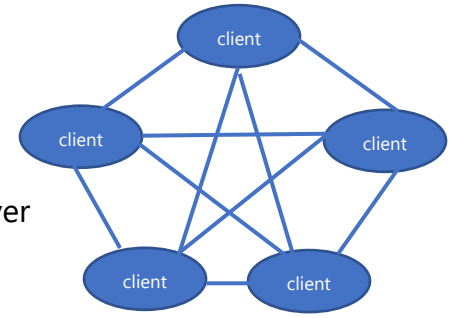
Đa xử lý đối xứng	Đa xử lý bất đối xứng
▫ Mỗi processor vận hành một bản sao hệ điều hành giống nhau ▫ Copy dữ liệu cho nhau khi cần ▫ VD: Windows NT, Solaris 5.0, Digital UNIX, OS/2, Linux	▫ Mỗi processor thực thi một công việc khác nhau ▫ Bộ xử lý chính định thời và phân công việc cho các bộ xử lý khác ▫ VD: SunOS 4.0

❖ **Hệ thống phân tán**

- Mỗi processor có bộ nhớ riêng, giao tiếp với nhau qua các kênh nối như mạng, bus tốc độ cao
- Người dùng chỉ thấy một hệ thống đơn nhất
- Các mô hình hệ thống phân tán:
  - Client-server(***h9***)
    - Server: cung cấp dịch vụ
    - Client: có thể sử dụng dịch vụ của server



h9



h10

  - Peer-to-peer(***h10***)
    - Các peer(máy tính trong hệ thống) đều ngang hang
    - Không có cơ sở dữ liệu tập trung
    - Các peer là tự trị
    - VD: Gnutella

❖ **Hệ thống nhúng thời gian thực**

- Hệ thống cho kết quả chính xác trong thời gian nhanh nhất
- Điều khiển công nghiệp, thử nghiệm khoa học, quân sự,..
- Ràng buộc về thời gian: hard và soft real-time
  - Hard real-time
    - Yêu cầu thời gian xử lý, đáp ứng nghiêm ngặt
    - Giới hạn bộ nhớ
  - Soft real-time
    - Công việc thực hiện theo độ ưu tiên
    - Dùng trong lĩnh vực multimedia, virtual reality,..

Chương II

Cấu trúc hệ điều hành

II.1 Các thành phần của hệ điều hành

- **Quản lý tiến trình**
  - Để hoàn thành công việc, một tiến trình cần:
    - CPU
    - Bộ nhớ
    - File
    - Thiết bị I/O,..
  - Các nhiệm vụ chính:
    - Các nhiệm vụ chính:
      - Tạo và hủy tiến trình
      - Tạm dừng/ thực thi tiếp tiến trình
      - Cung cấp các cơ chế:
        - Đồng bộ hoạt động các tiến trình
        - Giao tiếp giữa các tiến trình
        - Khống chế tắc nghẽn
- **Quản lý bộ nhớ chính**
  - Bộ nhớ chính là trung tâm của các thao tác, xử lý
  - Hệ điều hành cần quản lý bộ nhớ đã cấp phát cho mỗi tiến trình để tránh xung đột
  - Nhiệm vụ:
    - Quản lý các vùng nhớ trống và đã cấp phát
    - Quyết định sẽ nạp chương trình nào khi có vùng nhớ trống
    - Cấp phát bộ nhớ và thu hồi bộ nhớ
- **Quản lý file**
  - Hệ thống file:
    - File
    - Thư mục
  - Dịch vụ chính:
    - Tạo và xóa file/ thư mục
    - Các thao tác xử lý file/ thư mục
    - "Ánh xạ" file/ thư mục vào thiết bị thứ cấp tương ứng
    - Sao lưu và phục hồi dữ liệu

- **Quản lý hệ thống I/O**
  - Che dấu khác biệt của các thiết bị I/O trước người dùng
  - Chức năng:
    - Cơ chế: buffering, caching, spooling
    - Cung cấp giao diện chung đến các trình điều khiển thiết bị
    - Bộ điều khiển các thiết bị phần cứng
- **Quản lý hệ thống lưu trữ thứ cấp**
  - Lưu trữ bền vững các dữ liệu, chương trình (*vì bộ nhớ chính: kích thước nhỏ & môi trường chứa thông tin không bền vững*)
  - Phương tiện lưu trữ thường là đĩa từ,đĩa quang(HDD,SDD)
  - Nhiệm vụ hệ điều hành trong quản lý đĩa:
    - Quản lý không gian trống trên đĩa
    - Cấp phát không gian lưu trữ
    - Định thời hoạt động cho đĩa
- **Hệ thống bảo vệ**
  - Nhiệm vụ:
    - Cung cấp cơ chế kiểm soát đăng nhập/xuất
    - Phân định được sự truy cập tài nguyên hợp pháp và bất hợp pháp
    - Phương tiện thi hành các chính sách
- **Hệ thống thông dịch lệnh**
  - Là giao diện chủ yếu giữa người dùng và OS
    - VD: shell, mouse-based window-and-menu
  - Khi user login
    - Command line interpreter (shell) chạy, chờ nhận kết quả từ người dùng, thực thi lệnh và trả kết quả về
    - Các lệnh => Bộ điều khiển lệnh => Hệ điều hành
    - Các lệnh chủ yếu:
      - Tạo, hủy và quản lý tiến trình, hệ thống
      - Kiểm soát I/O
      - Quản lý bộ lưu trữ thứ cấp
      - Quản lý bộ nhớ chính
      - Truy cập hệ thống file và cơ chế bảo mật

II.2 Các dịch vụ hệ điều hành cung cấp

- Thực thi chương trình
- Thực hiện các thao tác I/O theo yêu cầu của chương trình

- Các thao tác trên hệ thống file
- Trao đổi thông tin giữa các tiến trình qua:
  - Chia sẻ bộ nhớ
  - Chuyển thông điệp
- Phát hiện lỗi
  - Trong CPU, bộ nhớ, trên thiết bị I/O (dư liệu hư, hết giấy,..)
  - Do chương trình: chia cho 0, truy cập đến địa chỉ bộ nhớ không cho phép
- Ngoài ra còn có các dịch vụ tăng hiệu suất của hệ thống:
  - Cấp phát tài nguyên
    - Tài nguyên: CPU, bộ nhớ chính, ổ đĩa,…
    - OS có các routine tương ứng
  - Kế toán
    - Nhằm lưu vết user để tính phí hoặc đơn giản để thống kê
  - Bảo vệ
    - Hai tiến trình khác nhau không được ảnh hưởng nhau
    - Kiểm soát được các truy xuất tài nguyên của hệ thống
  - An ninh
    - Chỉ các user được pháp sử dụng hệ thống mới truy cập được tài nguyên của hệ thống
      - Thông qua uername & password

II.3 Lời gọi hệ thống

- Giao tiếp giữa tiến trình và hệ điều hành
- Cung cấp giao diện giữa tiến trình và hệ điều hành
  - VD: open, read, write file
- Thông thường ở dạng thư viện nhị phân hoặc các lệnh hợp ngữ
- Trong các ngôn ngữ lập trình cấp cao, một số thư viện lập trình được xây dựng dựa trên các thư viện hệ thống
  - VD: Windows API, thư viện GNU C/C+như glibc,..
- 3 phương pháp truyền tham số khi sử dụng system call:
  - Qua thanh ghi
  - Qua một vùng nhớ, địa chỉ của vùng nhớ được gửi đến hệ điều hành qua thanh ghi
  - Qua stack

II.4 Các chương trình hệ thống

- Người dùng chủ yếu làm việc thông qua các system program (không làm việc "trực tiếp" với các system call)
- Các chương trình hệ thống (system program # application program) gồm:
  - Quản lý hệ thống file: create, delete, rename, list
  - Thông tin trạng thái: date,time,dung lượng bộ nhớ trống
  - Soạn thảo file: file editor
  - Hỗ trợ ngôn ngữ lập trình: compiler, assembler, interpreter
  - Nạp, thực thi, giúp tìm lỗi chương trình: loader, debugger
  - Giao tiếp: email,talk, web browser,..

II.5 Cấu trúc hệ thống

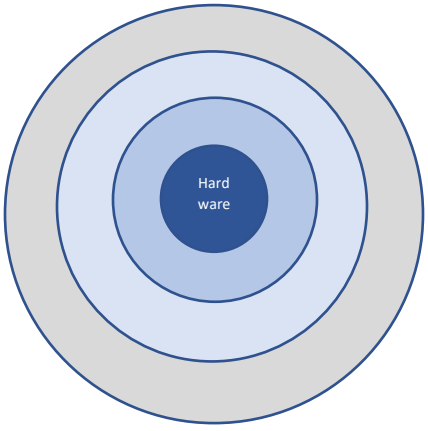
Hệ điều hành là một chương trình lớn, có nhiều dang cấu trúc:

- Cấu trúc Monolithic-Original UNIX
  - UNIX giới hạn về chức năng phần cứng nên Original UNIX cũng có cấu trúc rất giới hạn
  - UNIX gồm 2 phần tách rời:
    - Nhân
      - Cung cấp file system, CPU scheduling,..
    - System program

⇨ LINUX dựa theo cấu trúc monolithic được thiết kế theo dạng mô đun

- Cấu trúc Layered Approach
  - Hệ điều hành được chia thành nhiều lớp:
    - Lớp dưới cùng: hardware
    - Lớp trên cùng là giao tiếp với user
    - Lớp trên chỉ phụ thuộc lớp dưới
    - Một lớp chỉ có thể gọi các hàm của lớp dưới và các hàm của nó được gọi bởi lớp trên
    - VD: Hệ điều hành THE
  - Mỗi lớp tương đương một đối tương trừu tượng: cấu trúc dữ liệu + thao tác
  - Lợi ích của phân lớp:
    - Gỡ rối (debugger)
    - Kiểm tra hệ thống
    - Thay đổi chức năng

h11



- ❖ Cấu trúc Microkernels
  - Phân chia module theo microkernel
  - Chuyển một số chức năng của OS từ kernel space sang user space
  - Thu gọn kernel => microkernel
  - Microkernels có chức năng tối thiểu:
    - Quản lý tiến trình, bộ nhớ
    - Cơ chế giao tiếp giữa các tiến trình
  - Giao tiếp giữa các user module qua cơ chế truyền thông điệp
- ❖ Cấu trúc Modules
  - Nhiều hệ điều hành hiện đại triển khai các loadable kernel modules(LKMs)
    - Sử dụng cách tiếp cận hướng đối tượng
    - Mỗi core thành phần là tách biệt nhau
    - Trao đổi thông qua các interfaces
    - Mỗi module như là một phần của nhân

⇨ Cấu trúc Modules giống với cấu trúc Layer nhưng phức tạp hơn

- VD: Linux, Solaris,.

- ❖ Cấu trúc Hybrid System
  - Hầu hết các hệ điều hành hiện đại không theo một cấu trúc thuần túy nào mà lai giữa các cấu trúc với nhau
    - Cấu trúc lai là sự kết hợp nhiều cách tiếp cận để giải quyết các nhu cầu về hiệu suất, bảo mật, nhu cầu sử dụng
    - Nhân Linux và Solaris theo cấu trúc kết hợp không gian địa chỉ kernel, cấu trúc monolithic và modules
    - Nhân Windows hầu như theo cấu trúc liền khối, cộng với cấu trúc vi nhân cho các hệ thống cá nhân khác nhau

Chương III
Tiến trình

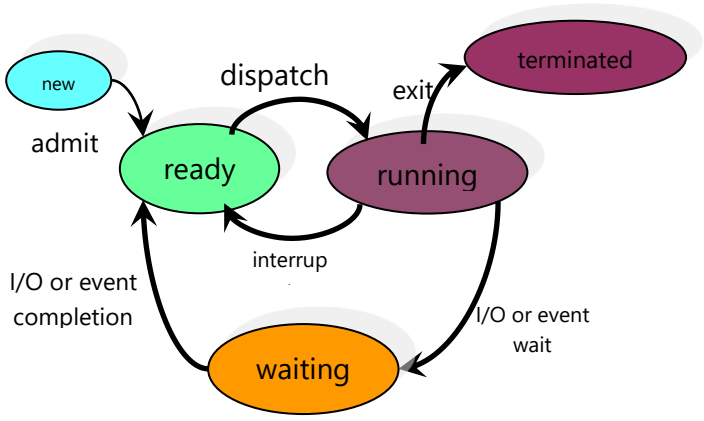
III.1 Các khái niệm cơ bản

- **Khái niệm**
  - Tiến trình(process) là một chương trình đang thực thi
    - Tiến trình bao gồm:
      - Text section(program code)
      - Data section (chứa global variables)
      - Program counter, processor registers
      - Heap section (chứa bộ nhớ cấp phát động)
      - Stack section (chứa dữ liệu tạm thời)
        - Function parameters
        - Return address
        - Local variables
  - Chương trình là thực thể bị động lưu trên đĩa (tập tin thực thi), tiến trình là thực thể chủ động

  ⇨ Chương tình trở thành tiến trình khi một tập tin thực thi được nạp vào bộ nhớ
- **Các bước khởi tạo tiến trình**
  - Cấp phát một định danh duy nhất cho tiến trình
  - Cấp phát không gian nhớ để nạp tiến trình
  - Khởi tạo khối dữ liệu PCB(Process Control Block) cho tiến trình
  - Thiết lập các mối liên hệ cần thiết
    - VD: sắp PCB vào hàng đợi đinh thời,..

III.2 Trạng thái tiến trình

- New : tiến trình vừa được tạo
- Ready: tiến trình đã có đủ tài nguyên, chỉ còn cần CPU
- Running: các lệnh của tiến trình đang được thực thi
- Waiting: hay là blocked, tiến trình đợi I/O hoàn tất, tín hiệu
- Terminated: tiến trình đã kết thúc



h12

Chương trình	Trạng thái trải qua

#include <stdio.h>
Void main() {
Printf("BHT CNPM");
Printf("Thi tot nhe ^^");
Exit(0);
}
	- New
	- Ready - Running
	-waiting (chờ I/O khi gọi printf)
	- ready - running
	- waiting(chờ I/O khi gọi printf)
	- ready – running
	- terminated

h13

III.3 Khối điều khiển tiến trình (PCB)

- **Khái niệm**
  - PCB(Process Control Block) là một trong các cấu trúc dữ liệu quan trọng nhất của hệ điều hành
    - Mỗi tiến trình trong hệ thống đều được cấp phát một PCB
- **Cấu tạo**
  - PCB bao gồm:
    - Trạng thái tiến trình: new, ready,..
    - Bộ đếm chương trình
    - Các thanh ghi
    - Thông tin lập thời biểu CPU: bộ ưu tiên,..
    - Thông tin quản lý bộ nhớ
    - Thông tin: lượng CPU, thời gian sử dụng
    - Thông tin trạng thái I/O

Pointer	Process State
Process number	
Process counter	
registers	
Memory limits	
List of open files	
...	

h14

III.4 Định thời tiến trình

- **Định thời**

- Đa chương
  - Nhiều tiến trình chạy tại 1 thời điểm
  - Mục tiêu: tận dụng tối đa CPU
- Chia thời
  - User tương tác với mỗi chương trình đang thực thi
  - Mục tiêu: tối thiểu thời gian đáp ứng

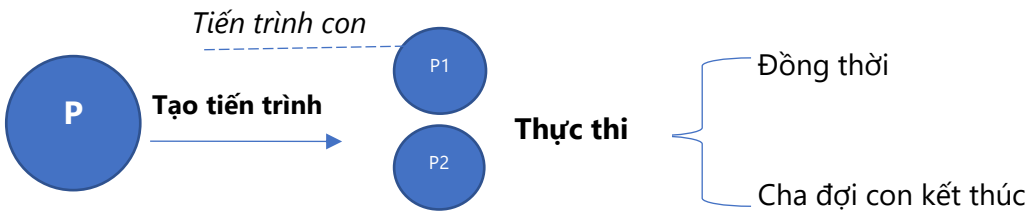
Scheduling

- Long-term scheduling (Job scheduler)
  - Xác định chương trình nào được chấp nhận nạp vào hệ thống để thực thi (new -> ready)
- Medium-term scheduling
  - Xác định tiến trình nào được đưa vào (swap in) và đưa ra (swap out) khỏi vùng nhớ chính
- Short-term scheduling
  - Xác định tiến trình nào trong ready queue sẽ được chiếm CPU để thực thi kế tiếp

III.5 Các tác vụ đối với tiến trình

Tạo tiến trình mới

- Tiến trình con nhận tài nguyên từ HĐH hoặc từ tiến trình cha
- Chia sẻ tài nguyên của tiến trình cha
  - Tiến trinh cha và con chia sẻ mọi tài nguyên
  - Tiến trình con chia sẻ một phần tài nguyên của cha
- Trình thực thi
  - Tiến trình cha và con thực thi đồng thời (concurrently)
  - Tiến trình cha đợi đến khi các tiến trình con kết thúc



h15

Kết thúc tiến trình

- Tiến trình tự kết thúc
  - Tiến trình kết thúc khi thực thi lệnh cuối và gọi system routine exit

- ❖ Tiến trình kết thúc do tiến trình khác(có đủ quyền, vd: tiến trình cha của nó)
  - Gọi system routine abort với tham số là pid(process identifier) của tiến trình cần được kết thúc

⇨ Hệ điều hành thu hồi tất cả các tài nguyên của tiến trình kết thúc(vùng nhớ, I/O buffer,..)

Cộng tác giữa các tiến trình

- ❖ Trong tiến trình thực thi, các tiến trình có thể cộng tác để hoàn thành công việc, nhằm:
  - Chia sẻ dữ liệu
  - Tăng tốc tính toán
  - Thực hiện một công việc chung

⇨ *Sự cộng tác yêu cầu hệ điều hành hỗ trợ cơ chế giao tiếp và cơ chế đồng bộ hoạt động của các tiến trình*

Chương IV

Định thời CPU

IV.1 Các khái niệm và các loại bộ định thời

Vấn đề

- ❖ Trong hệ thống multitasking
  - Thực thi nhiều chương trình đồng thời làm tăng hiệu suất hệ thống
  - Tại mỗi thời điểm,chỉ có một process được thực thi

⇨ Cần phải giải quyết vấn đề phân chia, lựa chọn process thực thi sao cho được hiệu quả nhất

⇨ Chiến lược định thời CPU

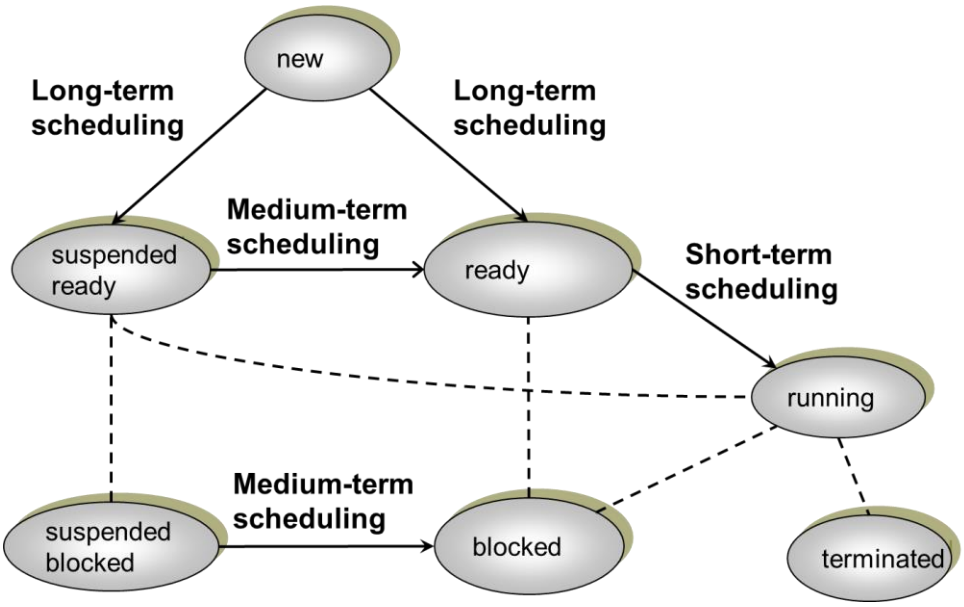
Khái niệm

- ❖ Định thời CPU
  - Chọn một process(từ ready queue) thực thi
  - Với một multithreaded kernel, việc định thời CPU là do OS chọn kernel thread được chiếm CPU

Các loại bộ định thời

- ❖ Long-term scheduling: Xác định chương trình nào được chấp nhận nạp vào hệ thống để thực thi

- Medium-term scheduling: Process nào được đưa vào (swap in) và đưa ra (swap out) khỏi vùng nhớ chính
- Short-term scheduling: Xác định process nào trong ready queue sẽ được chiếm CPU để thực thi kế tiếp



h16

IV.2 Các tiêu chuẩn định thời CPU

Hướng người dùng (User-oriented)

- Thời gian đáp ứng (response time): Khoảng thời gian process nhận yêu cầu đến khi yêu cầu đầu tiên được đáp ứng (time-sharing, interactive system) => cực tiểu
- Thời gian quay vòng (hoàn thành) (Turnaround time): Khoảng thời gian từ lúc một process được nạp vào hệ thống đến khi process đó kết thúc => cực tiểu
- Thời gian chờ (Waiting time): Tổng thời gian một process đợi trong ready queue => cực tiểu

Hướng hệ thống (System-oriented)

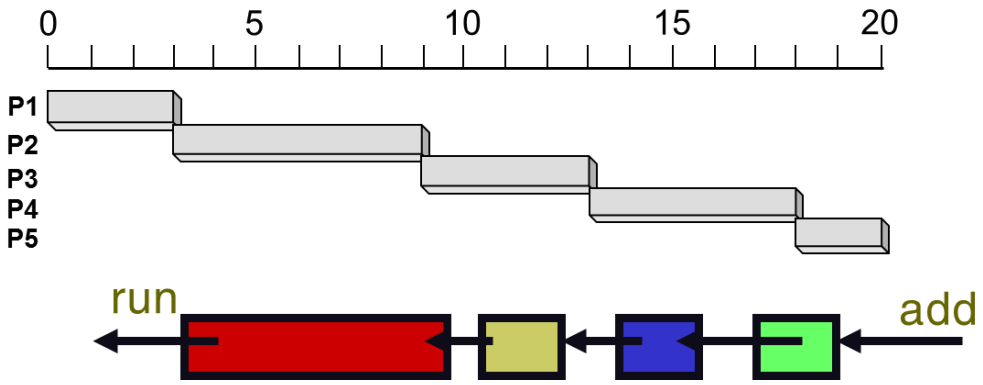
- Sử dụng CPU (processor utilization): định thời sao cho CPU càng bận càng tốt => cực đại
- Công bằng (fairness): Tất cả process phải được đối cử như nhau
- Thông lượng (throughput): Số process hoàn tất công việc trong một đơn vị thời gian => cực đại

IV.3 Các yếu tố của giải thuật định thời

- **Hàm chọn lựa (selection function)**
  - Dùng để chọn process nào trong ready queue được thực thi(thường dựa trên độ ưu tiên,yêu cầu tài nguyên, đặc điểm thực thi của process,..)
- **Chế độ quyết định (decision mode)**
  - Chọn thời điểm thực hiện hàm chọn lựa để định thời
  - Hai chế độ:
    - Không trưng dụng (Non-preemptive)
      - Khi ở trạng thái running, process sẽ thực thi cho đến khi kết thúc hoặc bị blocked do yêu cầu I/O
    - Trưng dụng (Preemptive)
      - Process đang thực thi (running) có thể bị ngắt nửa chừng và chuyển về trạng thái ready
      - Chi phí cao hơn non-preemptive nhưng đánh đổi lại bằng thời gian đáp ứng tốt hơn vì không có trường hợp một process độc chiếm CPU quá lâu

IV.4 Các giải thuật định thời

- **First-Come, First-Served (FCFS)**
  - Hàm chọn lựa
    - Tiến trình nào yêu cầu CPU trước sẽ được cấp phát CPU trước
    - Process sẽ thực thi đến khi kết thúc hoặc bị blocked do I/O
  - Chế độ quyết định: non-preemptive algorithm
  - Hiện thực: sử dụng hàng đợi FIFO
    - Tiến trình đi vào được thêm vào cuối hàng đợi
    - Tiến trình được lựa chọn để xử lý được lấy từ đầu của queues



h17

- **Shortest Job First (SJF)**
  - Hàm chọn lựa: Lựa chọn process có thời gian thực thi ngắn nhất trước
  - Chế độ quyết định
    - Scheme 1: Non-preemptive
      - Khi CPU được trao cho quá trinh nó không nhường cho đến khi nó kết thúc chu kỳ xử lý của nó
    - Scheme 2: Preemptive (Shortest-remainning-time-First)
      - Nếu một tiến trình mới được đưa vòa có chiều dài sử dụng CPU cho lần tiếp theo nhỏ hơn thời gian còn lại của tiến trình đang xử lý => Dừng hoạt động tiến trình hiện hành
  - Ưu điểm: Thời gian chờ đợi trung bình giảm
  - Nhược điểm: Process lớn sẽ đói (starvation) khi có nhiều process nhỏ
- **Priority Scheduling**
  - Hàm chọn lựa: Mỗi process sẽ có một độ ưu tiên,CPU sẽ được cấp cho process có độ ưu tiên cao nhất
  - Chế độ quyết định:
    - Preemptive
    - Non-preemptive
  - Ưu điểm: Các process quan trọng được thực thi trước
  - Nhược điểm: Trì hoãn vô hạn định cho các process có độ ưu tiên thấp

  ⇨ Giải pháp là tăng độ ưu tiên theo thời gian
- **Round-Robin (RR)**
  - Hàm lựa chọn:

- Mỗi process được cấp cho một định mức thời gian (quantum time q € [10-100ms])
- Sau khoảng thời gian đó thì process bị đoạt quyền và trở về cuối hàng đợi ready
- Hiệu suất:
  - Nếu q lớn: RR => FCFS
  - Nếu q quá nhỏ: tốn chi phí chuyển ngữ cảnh

❖ Ưu điểm: Thời gian đáp ứng nhanh
❖ Nhược điểm: Các process dạng CPU-bound (hướng xử lý) vẫn được "ưu tiên" và thời gian chờ đợi thường quá dài
❖ Nếu có n process trong hàng đợi ready, quantum time là q thì mỗi process lấy 1/n thời gian CPU thoe từng khối có kích thước lớn nhất là q
- Không có process nào chờ lâu hơn (n-1)*q đ/vị thời gian

❖ RR sử dụng giả thiết ngầm là tất cả các process đều có tầm quan trọng ngang nhau
- Không thể sử dụng RR nếu muốn các process có độ ưu tiên khác nhau

Highest Respone Ratio Next (HRRN)

❖ Hàm lựa chọn
- Chọn các process có tỉ lệ phản hồi (response ratio-RR) cao nhất
- Các process ngắn được ưu tiên hơn (vì service time nhỏ)

❖ Công thức
- $RR = \frac{\text{time spent waiting} + \text{expected service time}}{\text{expected service time}}$

Multilevel Queue

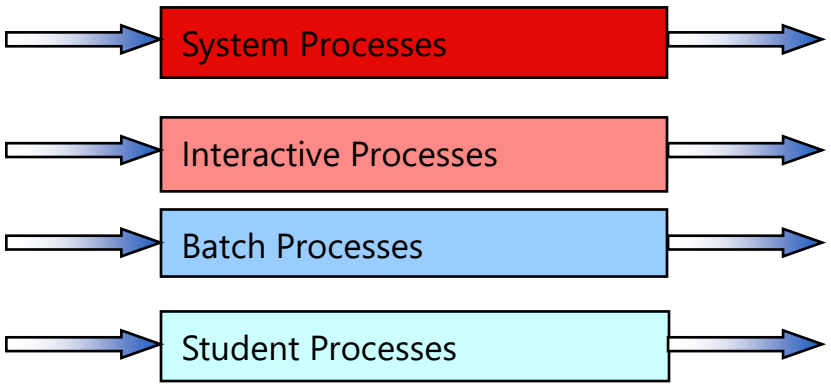
❖ Hàng đợi ready được chia thành các hàng đợi riêng biệt dựa vào:
- Đặc điểm và yêu cầu định thời của process
- Foreground và background process,..

❖ Process được gán cố định vào một hàng đợi, mỗi hàng đợi sử dụng giải thuật định thời riêng
❖ Qúa trình được chạy ở chế độ giao tiếp (foreground hay interactive process) được ưu tiên hơn so với quá trình chạy nền (background process)
❖ Hệ điều hành cần phải định thời cho các hàng đợi
- Fixed priority scheduling: phục vụ từ hàng đợi có độ ưu tiên cao đến thấp

- Vấn đề: Có thể gây ra tình trạng starvation (đói tài nguyên)

- Time slice: Mỗi hàng đợi được nhận một khoảng thời gian chiếm CPU và phân phối cho các process trong hàng đợi khoảng thời gian đó
  - VD: 80% cho hàng đợi foreground định thời bằng Round Robin và 20% cho hàng đợi background định thời bằng giải thuật FCFS

Độ ưu tiên cao nhất



h18

Độ ưu tiên thấp nhất

❖ Vấn đề của multilevel queue:
- Process không thể chuyển từ hàng đợi này sang hàng đợi khác

⇨ Cơ chế feedback

Multilevel Feedback Queue

❖ Cơ chế feedback:
- Các process có thể di chuyển qua lại giữa các hàng đợi
- Nếu một quá trình dùng quá nhiều thời gian CPU thì nó bị di chuyển đến hàng đợi có độ ưu tiên thấp
- Ngược lại, khi chờ lâu trong hàng đợi có độ ưu tiên thấp quá lâu thì nó có thể được di chuyển sang hàng đợi có độ ưu tiên cao hơn

❖ Ưu và nhược điểm:
- Thuật toán định thời phổ biến và phức tạp nhất

- Những quá trình nằm trong khoảng thời gian t nào đó sẽ được đáp ứng nhanh

❖ Yêu cầu cần giải quyết"

- Số lượng hàng đợi bao nhiêu là thích hợp ?
- Dùng giải thuật nào ở mỗi hàng đợi ?
- Làm sao để xác định thời điểm để chuyển một process đến hàng đợi cao hoặc thấp hơn ?
- Khi process yêu cầu được xử lý thì hàng đợi nào là hợp lý nhất ?