



TRAINING GIỮA KỲ II NĂM HỌC 2021-2022

HỆ ĐIỀU HÀNH

TRAINER

Nguyễn Nam Hải
Lê Thị Thu Huyền

 Email : bht.ktmt@gmail.com

 Fanpage: www.facebook.com/bht.ktmt

Fanpage Ban Học Tập
Khoa Kỹ Thuật Máy Tính 
www.facebook.com/bht.ktmt



NỘI DUNG

CHƯƠNG 1

**TỔNG QUAN
HỆ ĐIỀU
HÀNH**



CHƯƠNG 2

**CẤU TRÚC
HỆ ĐIỀU
HÀNH**



CHƯƠNG 3

**TIẾN
TRÌNH**



CHƯƠNG 4

**ĐỊNH THỜI
CPU**



CHƯƠNG 1

TỔNG QUAN VỀ

HỆ ĐIỀU HÀNH

TỔNG QUAN

SỰ CẦN THIẾT CỦA HỆ ĐIỀU HÀNH

CHỨC NĂNG CỦA HỆ ĐIỀU HÀNH

PHÂN LOẠI HỆ ĐIỀU HÀNH





TỔNG QUAN

Hệ điều hành là gì?

Chương trình trung gian giữa **phần cứng** máy tính & **người dùng**
Điều khiển và phối hợp việc sử dụng phần cứng
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 hiệu quả



SỰ CẦN THIẾT CỦA HỆ ĐIỀU HÀNH

- Quản lý phần cứng máy tính
- Cung cấp giao diện 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



CÁC CHỨC NĂNG CHÍNH CỦA HỆ ĐIỀU HÀNH



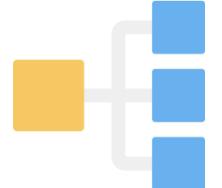
Phân chia thời gian xử lý
và định thời CPU



Quản lý tài nguyên hệ
thống (thiết bị I/O, bộ
nhớ, file chứa dữ liệu, ...)



Duy trì sự nhất quán của
hệ thống, kiểm soát lỗi
và phục hồi khi có lỗi



Phân phối và đồng bộ hoạt
động giữa các processes



Kiểm soát truy cập,
bảo vệ hệ thống



Cung cấp giao diện làm việc



PHÂN LOẠI HỆ ĐIỀU HÀNH

Về loại máy tính



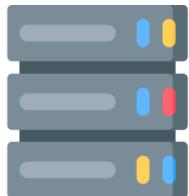
máy MainFrame



máy tính cá nhân
(Desktop, Laptop)



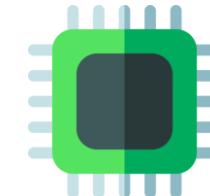
cho máy chuyên biệt
(Car, TV)



dành cho
máy Server



dành cho máy PDA
(Phone, Tablet)

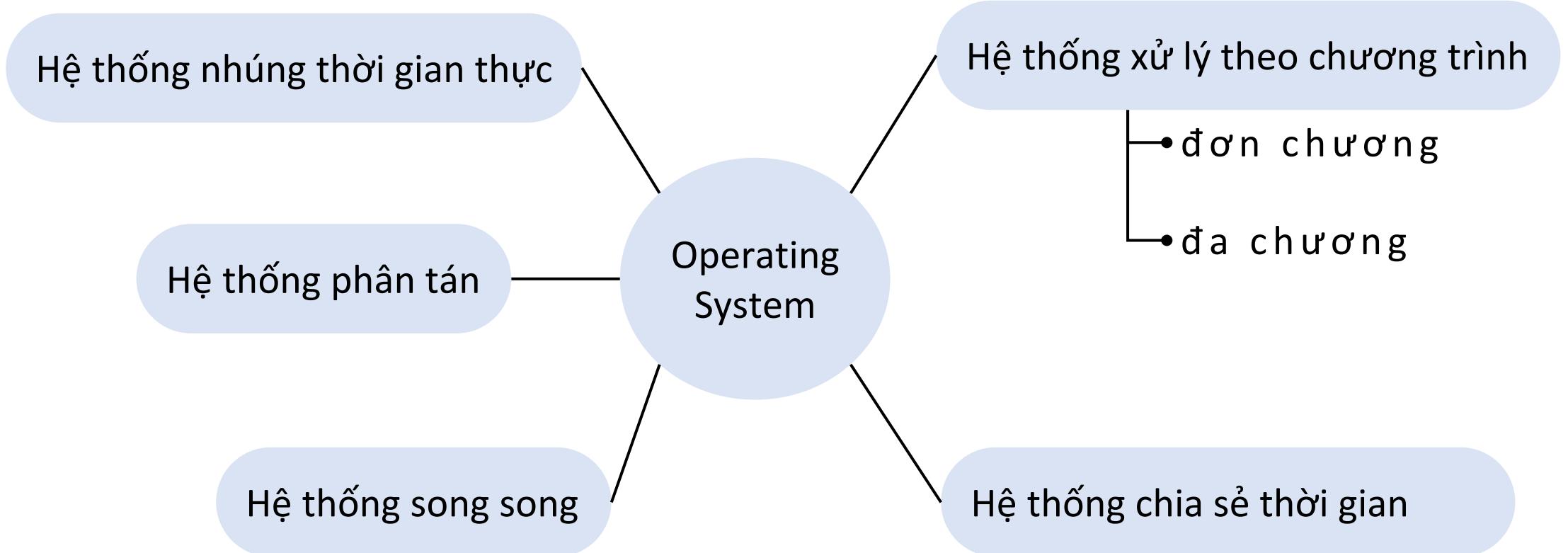


Cho thiết bị nhúng
(RTOS)



PHÂN LOẠI HỆ ĐIỀU HÀNH

Về hình thức xử lý

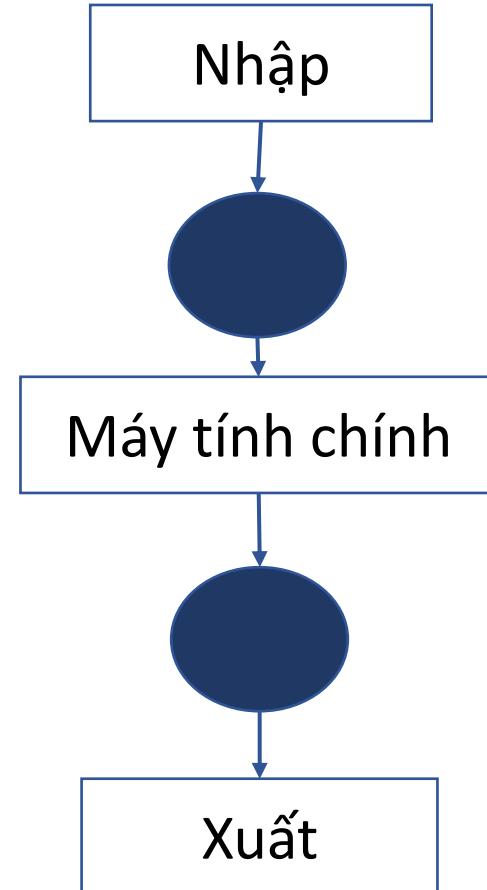




PHÂN LOẠI DƯỚI GÓC ĐỘ HÌNH THỨC XỬ LÝ

Hệ thống đơn chương

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

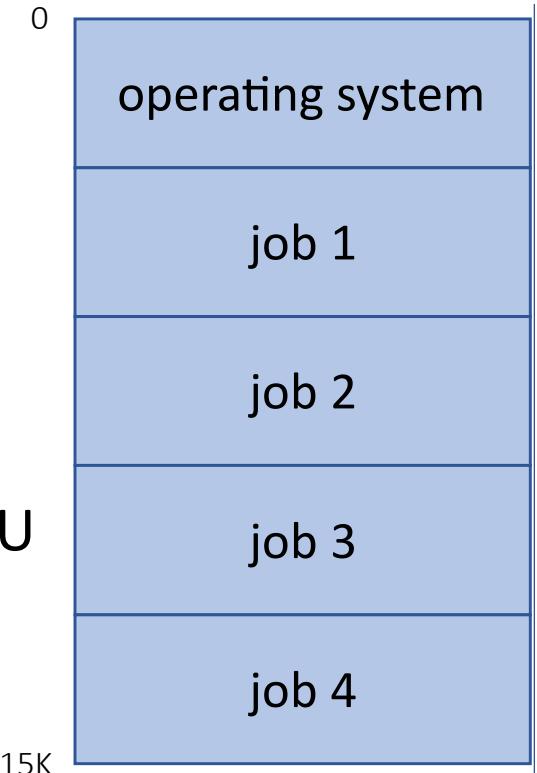
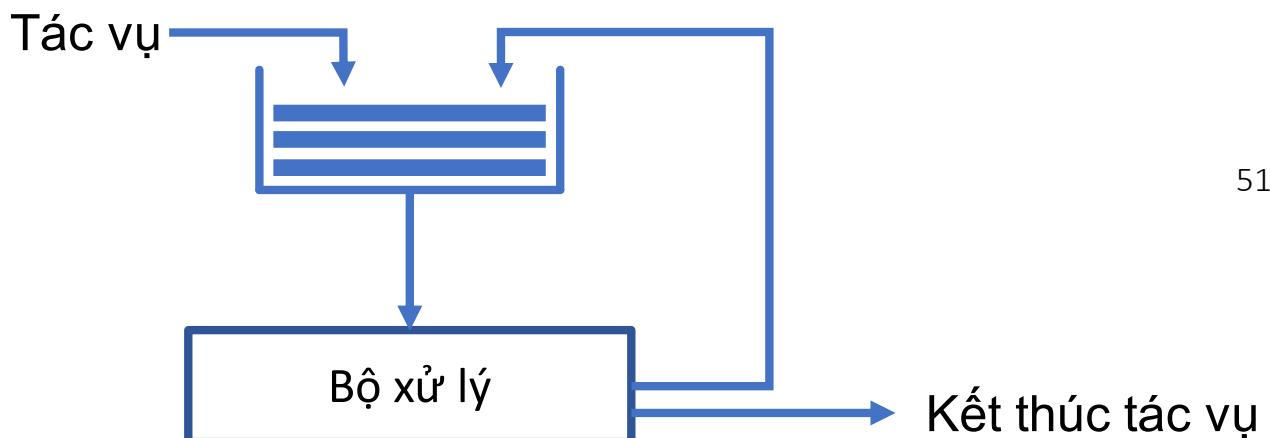




PHÂN LOẠI DƯỚI GÓC ĐỘ HÌNH THỨC XỬ LÝ

Hệ thống đa chương

- Nhiều công việc nạp đồng thời vào bộ nhớ chính
- Khi 1 tiến trình thực hiện I/O, 1 tiến trình khác được thực thi
- Tận dụng thời gian rảnh, tăng hiệu suất sử dụng CPU





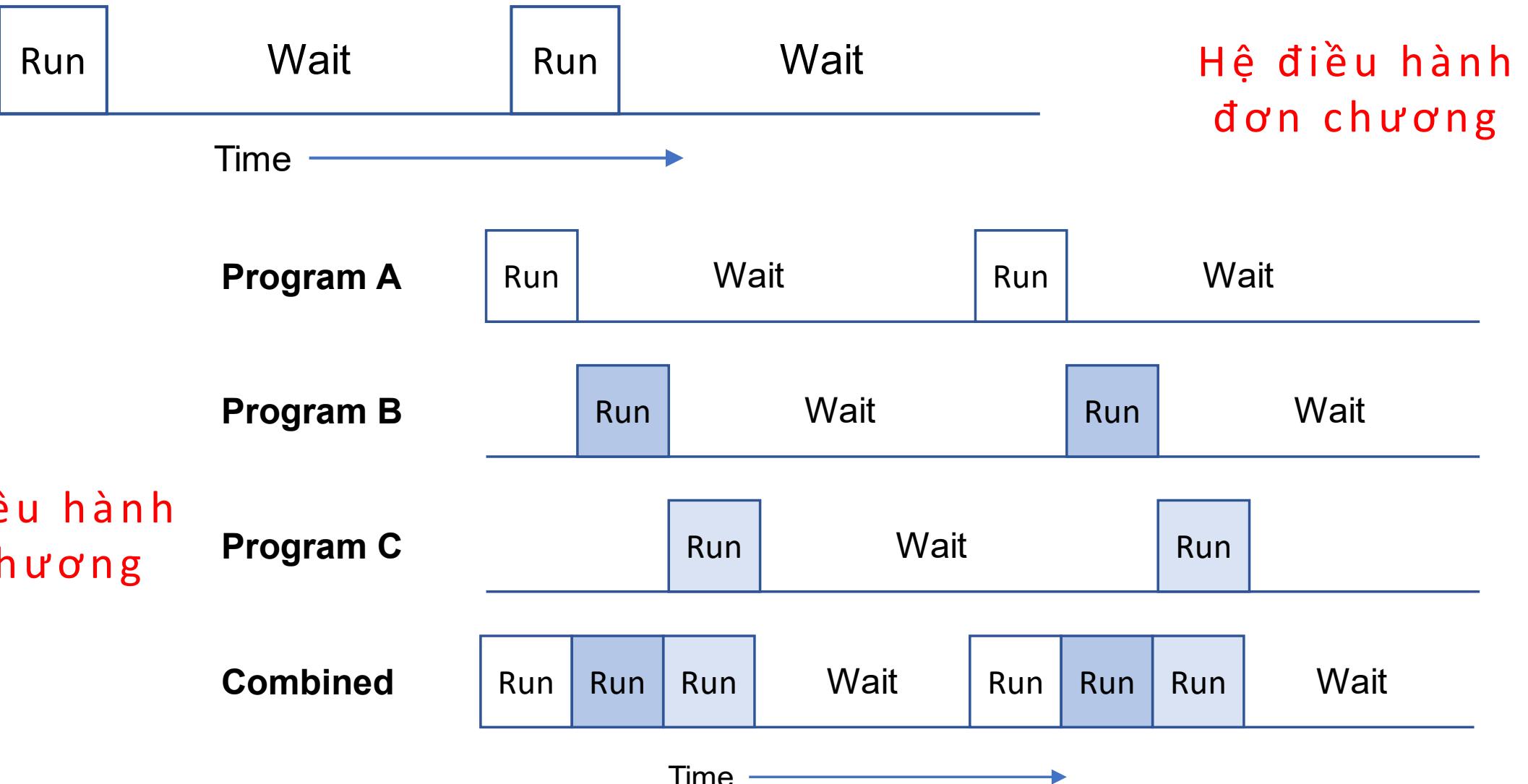
PHÂN LOẠI DƯỚI GÓC ĐỘ HÌNH THỨC XỬ LÝ

Hệ thống đa chương: yêu cầu đối với hệ điều hành

- Định thời công việc (job scheduling): chọn job trong job pool trên đĩa và nạp nó vào bộ nhớ để thực thi
- Quản lý bộ nhớ (memory management)
- Định thời CPU
- Cấp phát tài nguyên (đĩa, máy in, ...)
- Bảo vệ



PHÂN LOẠI DƯỚI GÓC ĐỘ HÌNH THỨC XỬ LÝ



Hệ điều hành
đa chương



PHÂN LOẠI DƯỚI GÓC ĐỘ HÌNH THỨC XỬ LÝ

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

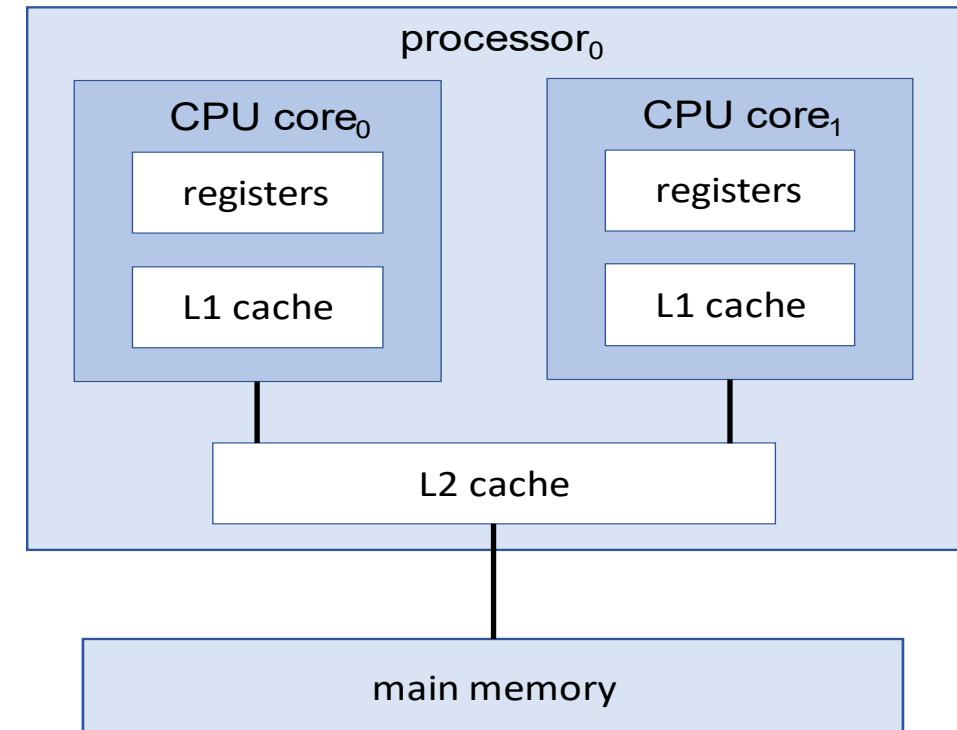
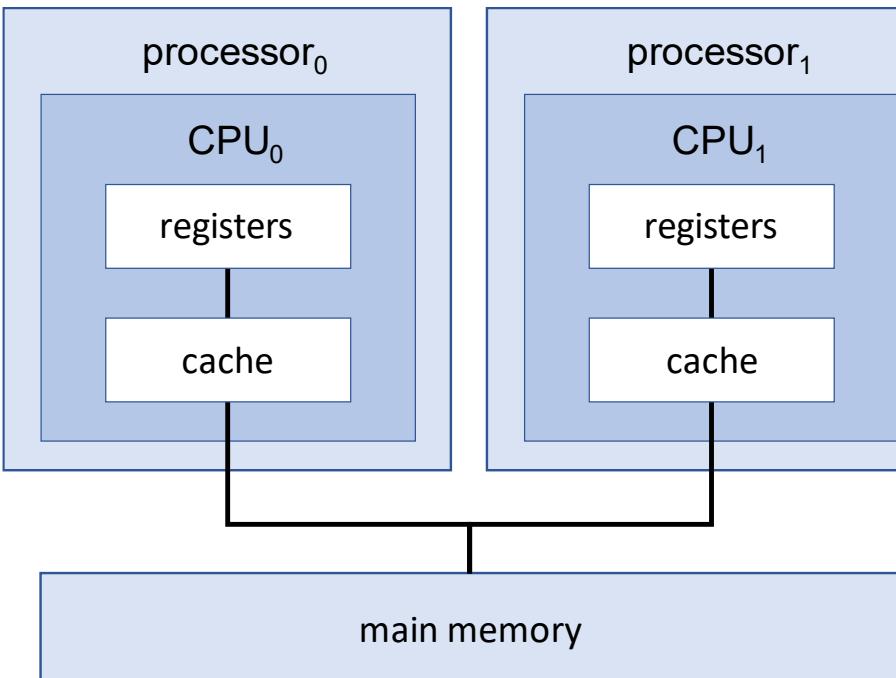
Ngoài ra:

- 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



Hệ thống song song

- Hai hoặc n bộ xử lý cùng chia sẻ một bộ nhớ
- Master/Slave: 1 bộ xử lý chính kiểm soát một số bộ xử lý I/O





Hệ thống song song

Phân loại

Đa xử lý đối xứng

- Mỗi processor vận hành 1 bản sao hệ điều hành giống nhau
- Copy dữ liệu cho nhau khi cần
- Windows NT, Solaris 5.0, Digital UNIX, OS/2, Linux

Đa xử lý bất đối xứng

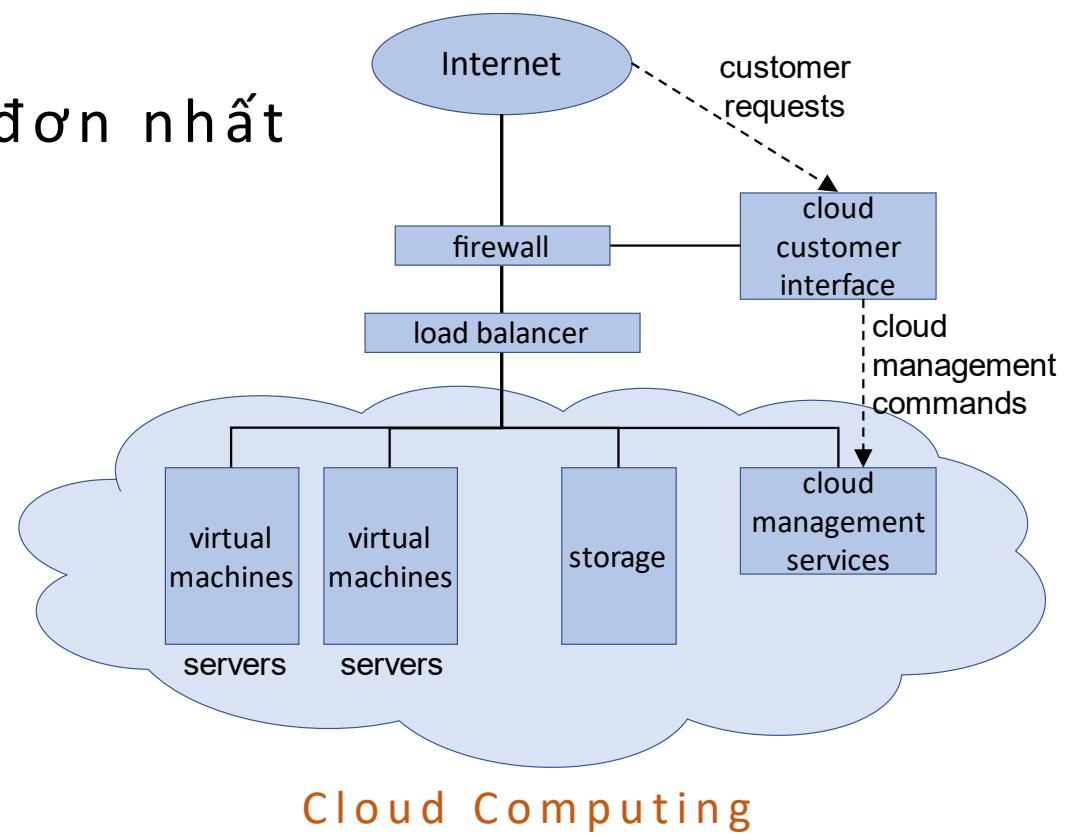
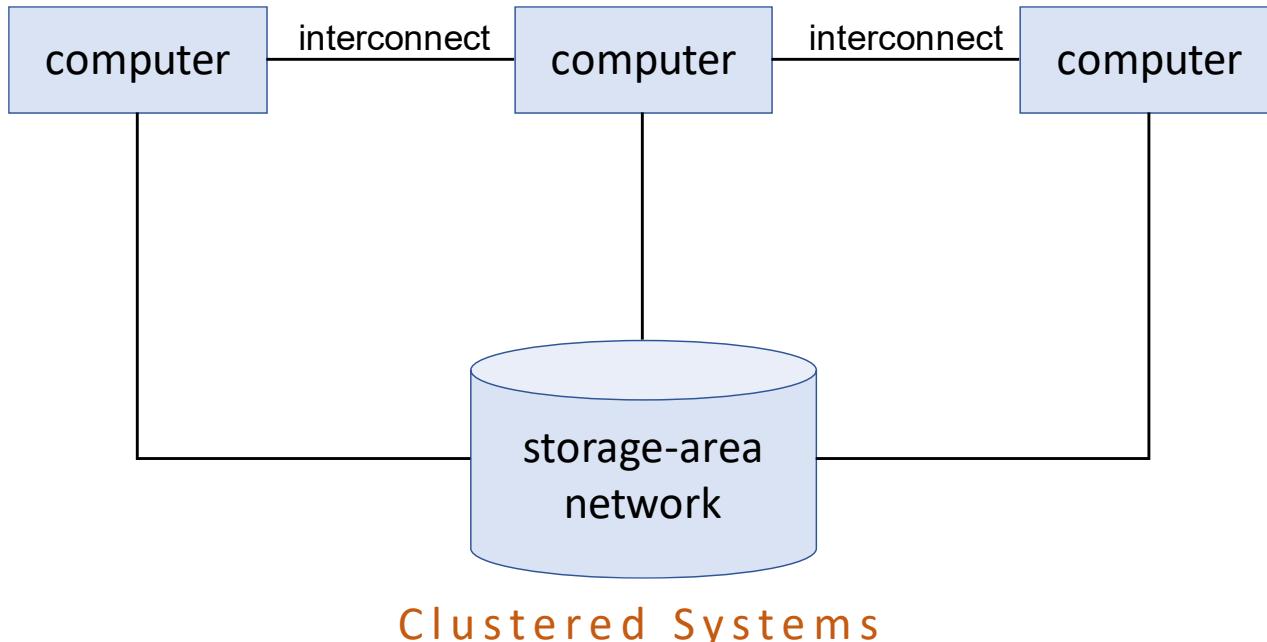
- Mỗi processor thực thi 1 công việc khác nhau
- Master processor định thời & phân chia công việc cho slave
- SunOS 4.0



PHÂN LOẠI DƯỚI GÓC ĐỘ HÌNH THỨC XỬ LÝ

Hệ thống phân tán

- Mỗi processor có bộ nhớ riêng, giao tiếp với nhau qua các kênh mạng, bus tốc độ cao
- Người dùng chỉ thấy một hệ thống đơn nhất



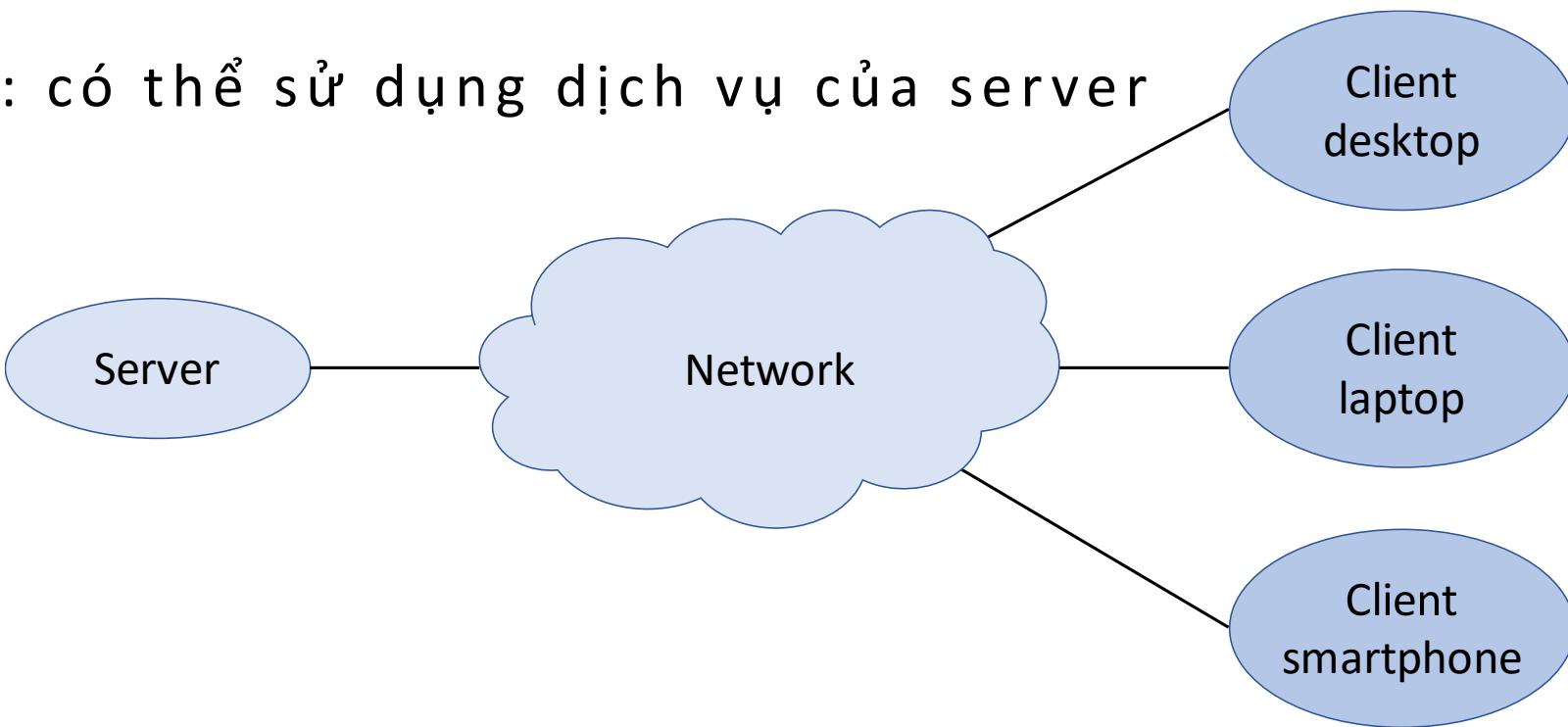


PHÂN LOẠI DƯỚI GÓC ĐỘ HÌNH THỨC XỬ LÝ

Các mô hình hệ thống phân tán

- Client-server

- Server: cung cấp dịch vụ
- Client: có thể sử dụng dịch vụ của server



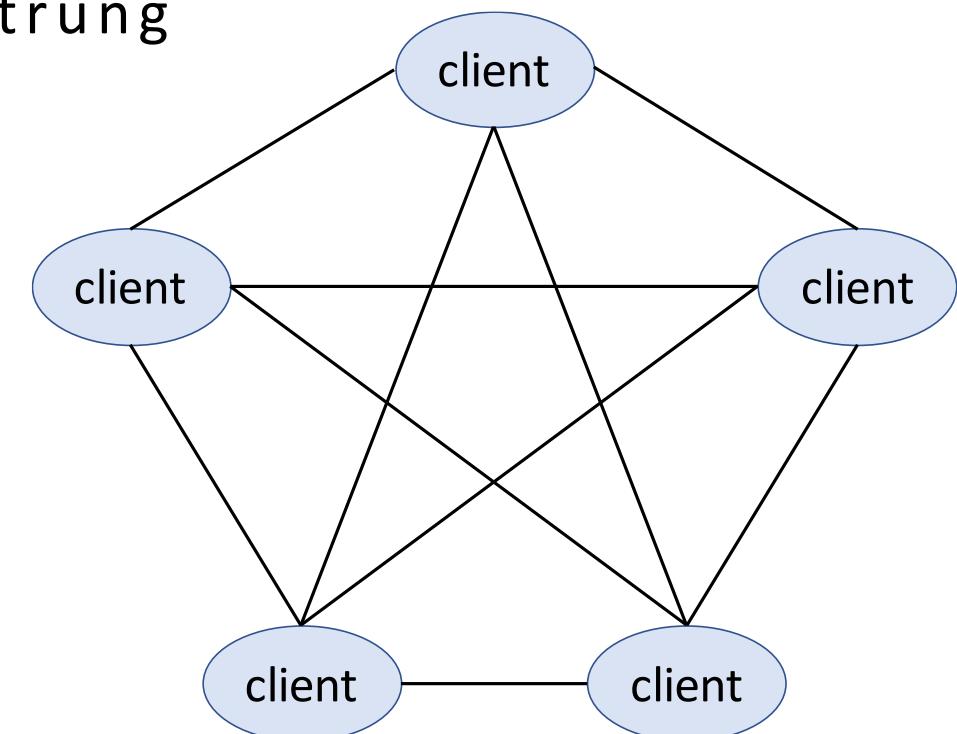


PHÂN LOẠI DƯỚI GÓC ĐỘ HÌNH THỨC XỬ LÝ

Các mô hình hệ thống phân tán

- Peer-to-Peer (P2P)

- Các peer (máy tính trong hệ thống) đều ngang hàng nhau
- Không có cơ sở dữ liệu tập trung
- Các peer là tự trị
- Ví dụ: Gnutella





PHÂN LOẠI DƯỚI GÓC ĐỘ HÌNH THỨC XỬ LÝ

Các mô hình hệ thống phân tán

- Peer-to-Peer (P2P)
 - └ Ưu điểm ┌ Chia sẻ tài nguyên (máy in, ...)
 ├ Phân chia công việc → Tốc độ tính toán cao
 ├ Độ an toàn cao
 ├ Độ sẵn sàng cao
 └ Truyền thông tin dễ dàng (file, mail, ...)



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, y học, quân sự, ...

• 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, ...



CÂU HỎI TRẮC NGHIỆM

Câu 1: Đâu không phải là một đặc điểm của hệ thống phân tán

- A. Người dùng chỉ nhìn thấy 1 hệ thống đơn nhất
- B. Độ sẵn sàng cao vì các dịch vụ của hệ thống được cấp phát liên tục cho dù có phần cứng bị hỏng
- C. Mỗi processor có bộ nhớ riêng, giao tiếp với nhau qua các kênh nối như mạng, bus có tốc độ cao
- D. Master processor định thời và phân công cho slave processor



CÂU HỎI TRẮC NGHIỆM

Câu 2: Hệ thống song song được phân loại như thế nào?

- A. Đa xử lý đối xứng và bất đối xứng
- B. Đơn chương và đa chương
- C. Client-server và peer-to-peer
- D. Hard real-time và soft real-time

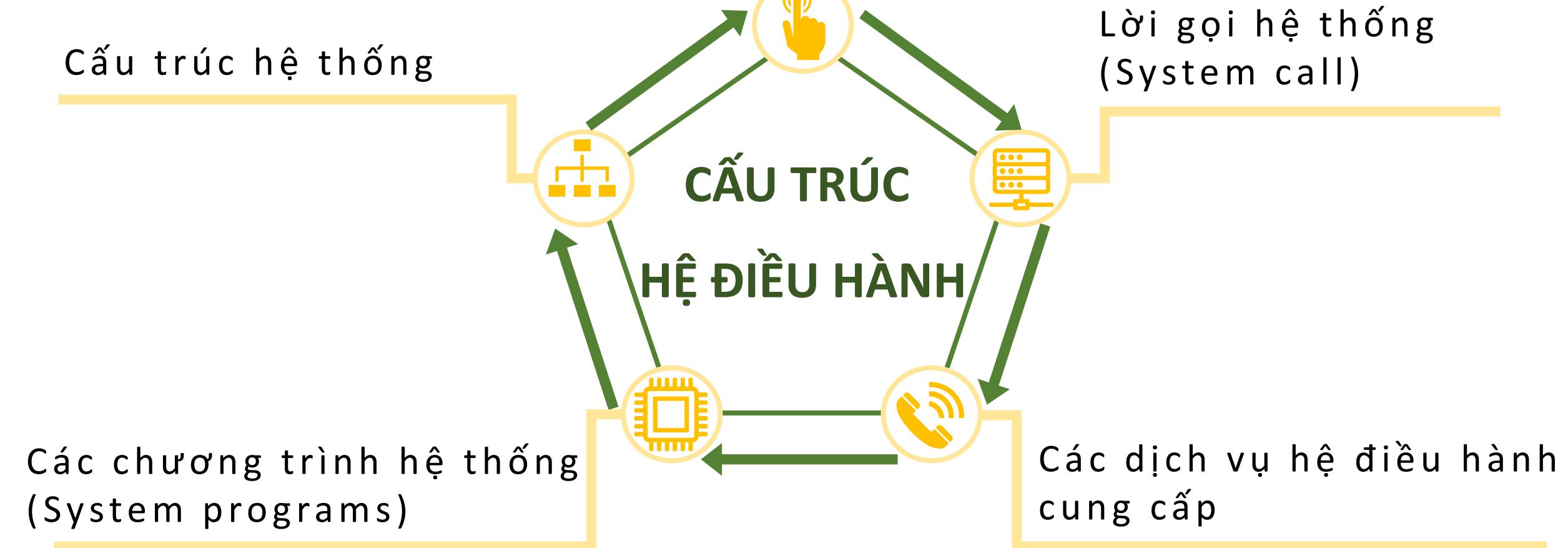
CHƯƠNG 2

CẤU TRÚC HỆ ĐIỀU HÀNH





Các thành phần của hệ điều hành





CÁC THÀNH PHẦN CỦA HỆ ĐIỀU HÀNH





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

Tạo và hủy tiến trình

Tạm dừng/ thực thi tiếp tiến trình

Nhiệm vụ chính

Cung cấp các cơ chế

Đồng bộ hoạt động các tiến trình

Giao tiếp giữa các tiến trình

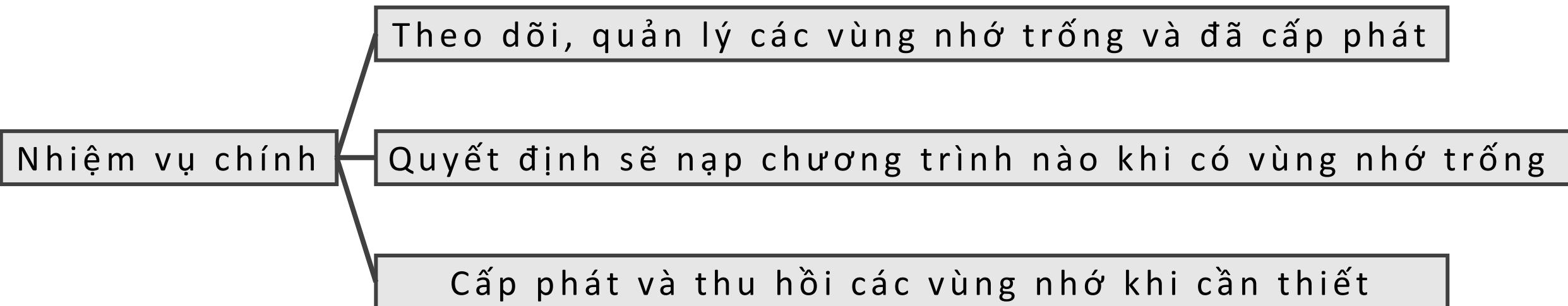
Khống chế tắc nghẽn



CÁC THÀNH PHẦN CỦA HỆ ĐIỀU HÀNH

Quản lý bộ nhớ chính

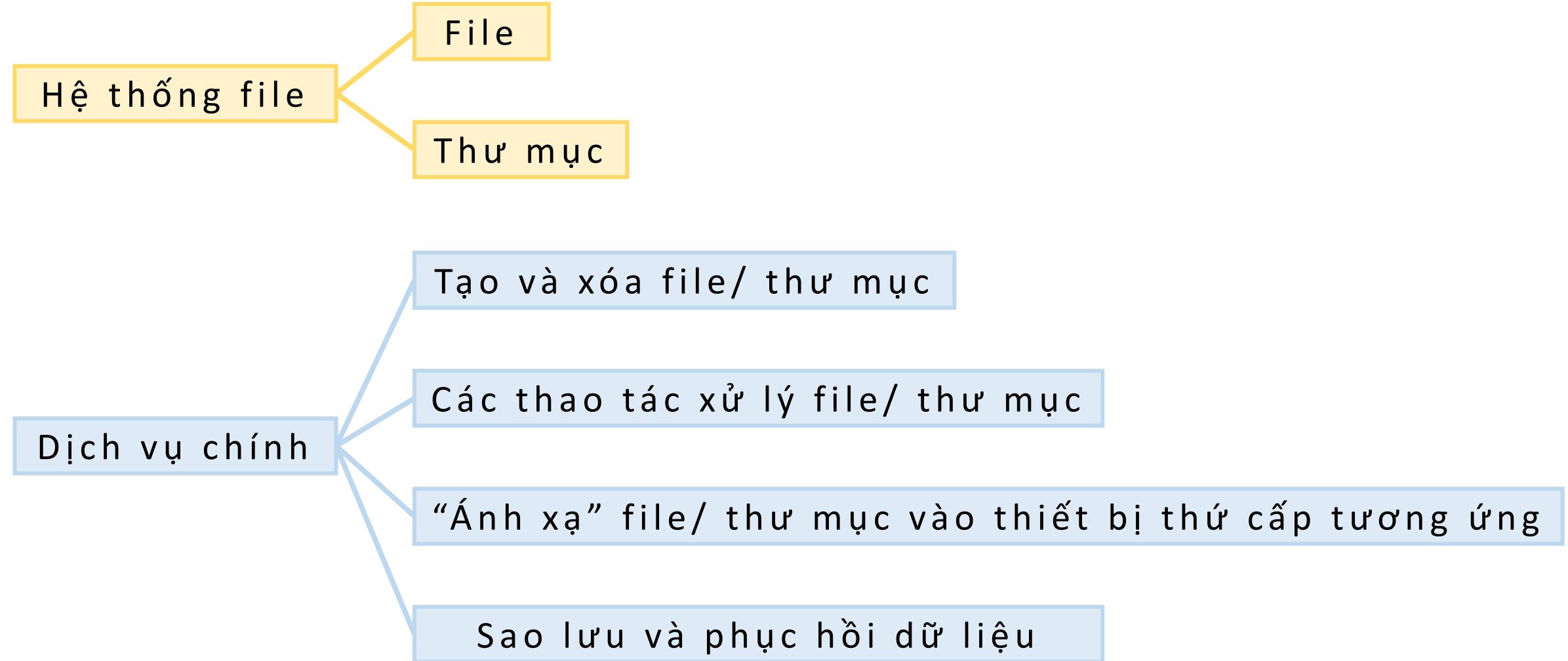
- Bộ nhớ chính: trung tâm của các thao tác, xử lý
- Hệ điều hành quản lý bộ nhớ thích hợp -> **nâng cao hiệu suất** sử dụng CPU





CÁC THÀNH PHẦN CỦA HỆ ĐIỀU HÀNH

Quản lý file

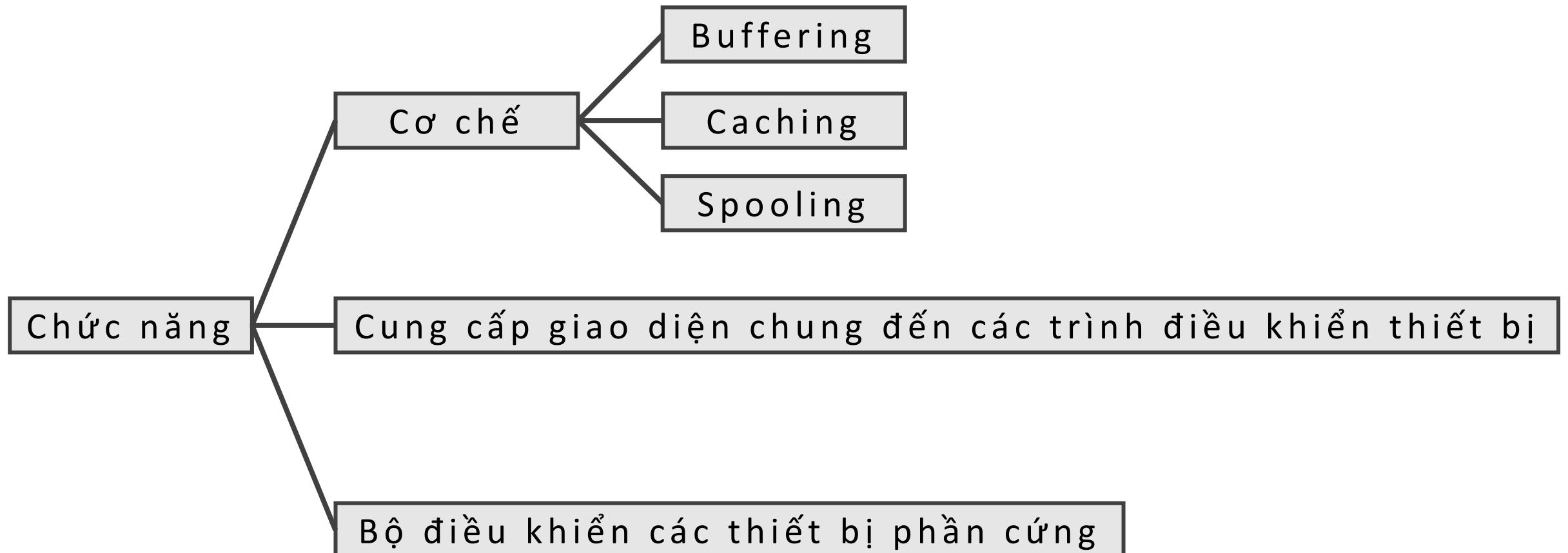




CÁC THÀNH PHẦN CỦA HỆ ĐIỀU HÀNH

Quản lý hệ thống I/O

Giúp che dấu sự khác biệt của các thiết bị I/O trước người dùng





CÁC THÀNH PHẦN CỦA HỆ ĐIỀU HÀNH

Quản lý hệ thống lưu trữ thứ cấp

Bộ nhớ chính

Kích thước **nhỏ**

Là môi trường chứa thông tin **không** bền vững
=> Cần hệ thống lưu trữ **thứ cấp** để lưu trữ
bền vững các dữ liệu, chương trình

Phương tiện lưu trữ thông dụng là đĩa từ, đĩa quang

Nhiệm vụ của hệ điều hành trong quản lý đĩa:

- Quản lý không gian trống trên đĩa (free space management)
- Cấp phát không gian lưu trữ (storage allocation)
- Định thời hoạt động cho đĩa (disk scheduling)

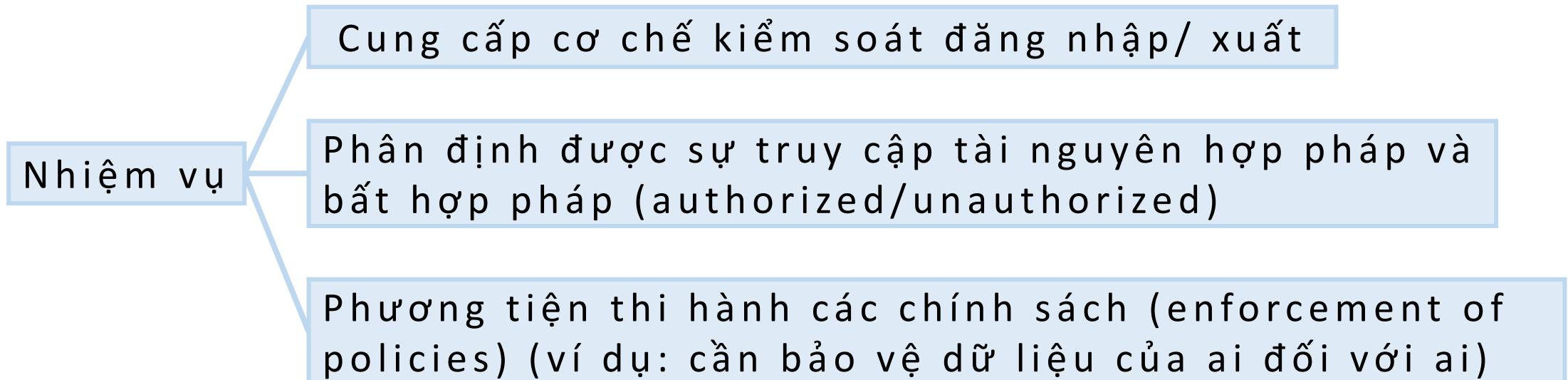


CÁC THÀNH PHẦN CỦA HỆ ĐIỀU HÀNH

Hệ thống bảo vệ

Trong hệ thống cho phép nhiều user hay nhiều process diễn ra đồng thời:

- ⌚ Kiểm soát tiến trình người dùng đăng nhập/ xuất và sử dụng hệ thống
- ⌚ Kiểm soát việc truy cập các tài nguyên trong hệ thống
- 🛡 Bảo đảm những user/process **chỉ được phép sử dụng các tài nguyên dành cho nó**





CÁC THÀNH PHẦN CỦA HỆ ĐIỀU HÀNH

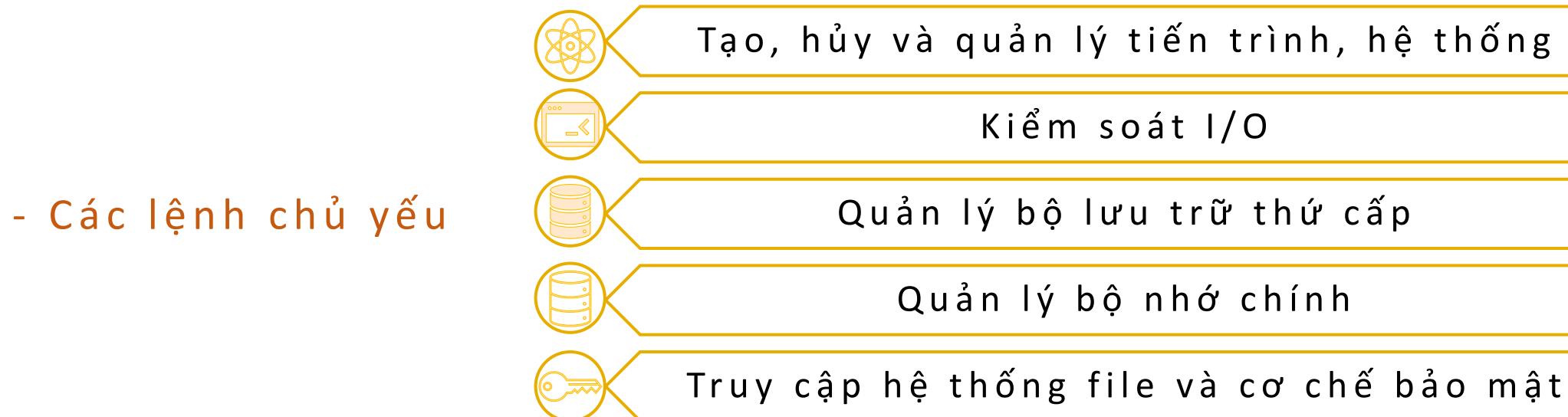
Hệ thống thông dịch lệnh

Là giao diện chủ yếu giữa **người dùng** và OS

Ví dụ: shell, mouse-based window-and-menu

Khi user login

- Command line interpreter (shell) chạy, chờ nhận lệnh từ người dùng, thực thi lệnh và trả kết quả về.
- Các lệnh -> bộ điều khiển lệnh -> hệ điều hành



- Các lệnh chủ yếu



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

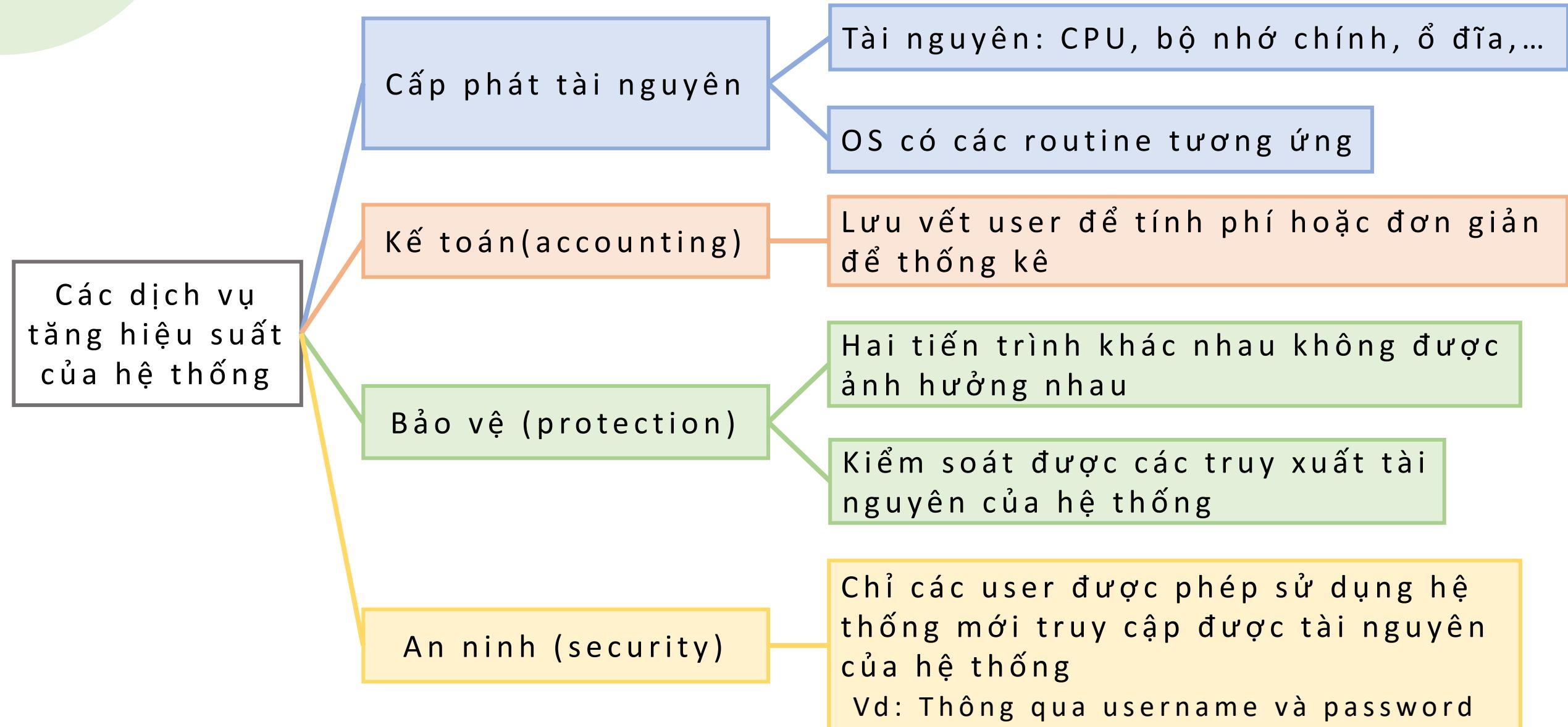
Chia sẻ bộ nhớ (Shared memory)
Chuyển thông điệp (Message passing)

Phát hiện lỗi

Trong CPU, bộ nhớ, trên thiết bị I/O (đữ liệu hư, hết giấy,...)
Do chương trình: chia cho 0, truy cập đến địa chỉ bộ nhớ không cho phép.



CÁC DỊCH VỤ HỆ ĐIỀU HÀNH CUNG CẤP





LỜI GỌI HỆ THỐNG

- **Giao tiếp** giữa tiến trình và hệ điều hành
- **Cung cấp giao diện** giữa tiến trình và hệ điều hành
 - Ví dụ: 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
 - Ví dụ: Windows API, thư viện GNU C/C++ như glibc, glibc++, ...



LỜI GỌI HỆ THỐNG

Ba phương pháp
truyền tham số
khi sử dụng
system call

Qua thanh
ghi

Qua một vùng
nhớ mà **địa chỉ**
vùng nhớ được
gửi đến HDH qua
thanh ghi

Qua stack



CÁC CHƯƠNG TRÌNH HỆ THỐNG

Quản lý file: create, delete, rename, list



Thông tin trạng thái: date, time, dung lượng bộ nhớ



Soạn thảo file



Chương trình hệ thống
(system program)

⚠ Phân biệt với application program ⚡



Giao tiếp: email, talk, web browser

Hỗ trợ lập trình: compiler, assembler, interpreter



Nạp, thực thi, tìm lỗi chương trình:
loader, debugger



Người dùng làm việc thông qua system program
(không làm việc “trực tiếp” với system call)



CẤU TRÚC HỆ THỐNG

Hệ điều hành là một chương trình lớn

Có nhiều dạng cấu trúc khác nhau:

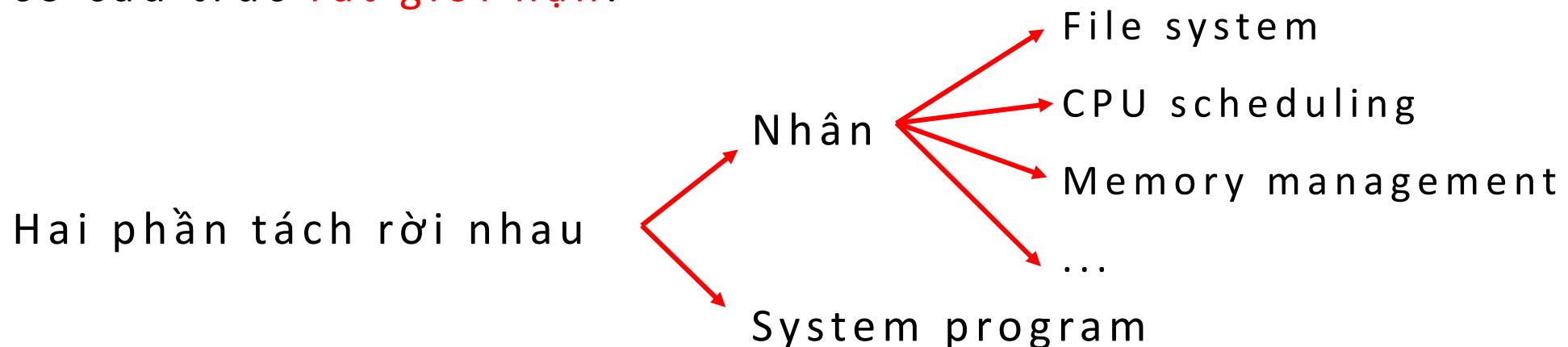
- Cấu trúc Monolithic - Original UNIX
- Cấu trúc Layered Approach
- Cấu trúc Microkernels
- Cấu trúc Modules
- Cấu trúc Hybrid Systems



CẤU TRÚC HỆ THỐNG

Cấu trúc Monolithic - Original UNIX

UNIX – do giới hạn về chức năng phần cứng nên Original UNIX cũng có cấu trúc **rất giới hạn**.



Cấu trúc hệ thống Linux

Linux dựa theo cấu trúc monolithic được thiết kế theo dạng **mô đun**



CẤU TRÚC HỆ THỐNG

Cấu trúc Layered Approach

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.

Ví dụ: Hệ điều hành THE

Hệ điều hành được
chia thành nhiều lớp

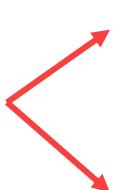


CẤU TRÚC HỆ THỐNG

Cấu trúc Microkernels

- Phân chia module theo microkernel (CMU Mach OS, 1980)
- Chuyển một số chức năng của OS từ **kernel space** sang **user space**
- Thu gọn kernel => microkernel

Chức năng tối thiểu



Quản lý tiến trình,

Bộ nhớ và cơ chế giao tiếp
giữa các tiến trình

- Giao tiếp giữa các user module qua cơ chế truyền thông điệp



CẤU TRÚC HỆ THỐNG

Cấu trúc Modules

Sử dụng cách tiếp cận
hướng đối tượng

Nhiều hệ điều hành
hiện đại sử dụng
loadable kernel
modules

Mỗi core thành phần là
tách biệt nhau

Mỗi module như
là một phần của nhân

Trao đổi thông tin qua
các interfaces

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

Ví dụ: Linux, Solaris



CẤU TRÚC HỆ THỐNG

Cấu trúc Hybrid Systems

Cấu trúc lai: Sự kết hợp để giải quyết hiệu suất, bảo mật, nhu cầu sử dụng

Nhân Linux và Solaris: cấu trúc không gian địa chỉ kernel + monolithic + modules

Nhân Windows: cấu trúc liền khối + cấu trúc vi nhân cho các hệ thống khác nhau

HDH hiện đại không dùng cấu trúc thuần mà lai giữa các cấu trúc với nhau



CẤU TRÚC HỆ THỐNG

Cấu trúc Android

Được phát triển bởi OpenHandset Alliance (Google)

Dựa trên nhân Linux

Môi trường chạy gồm các thư viện API và máy ảo

Thư viện Frameworks cho web browser Database Multimedia ...

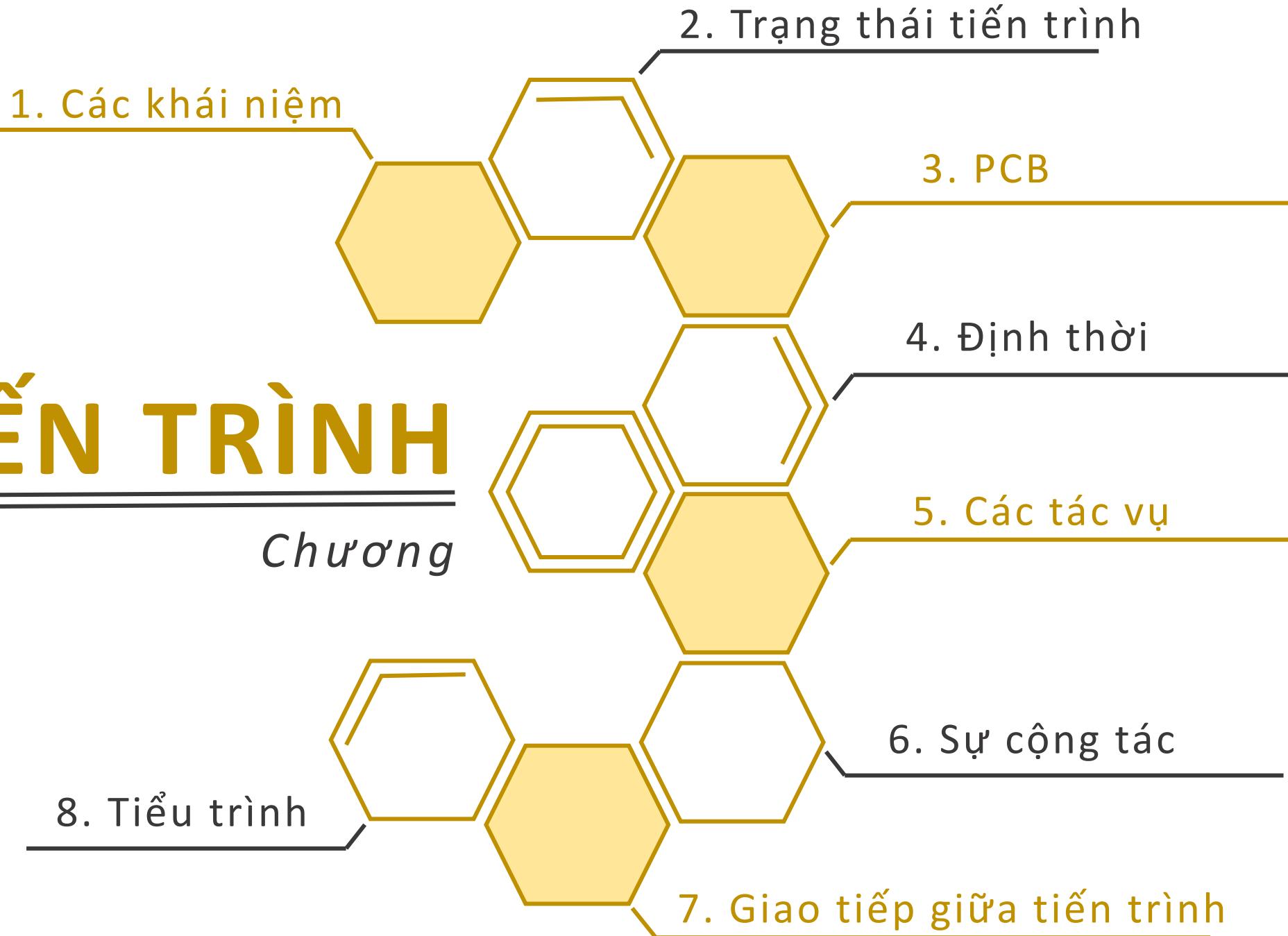
CHƯƠNG 3

TIẾN TRÌNH



TIẾN TRÌNH

Chương

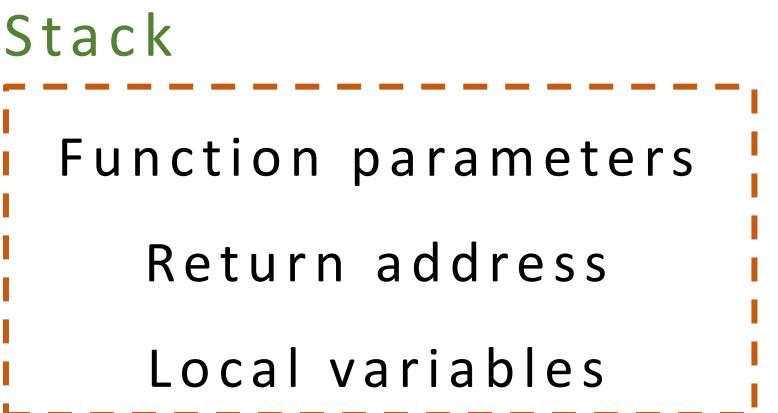




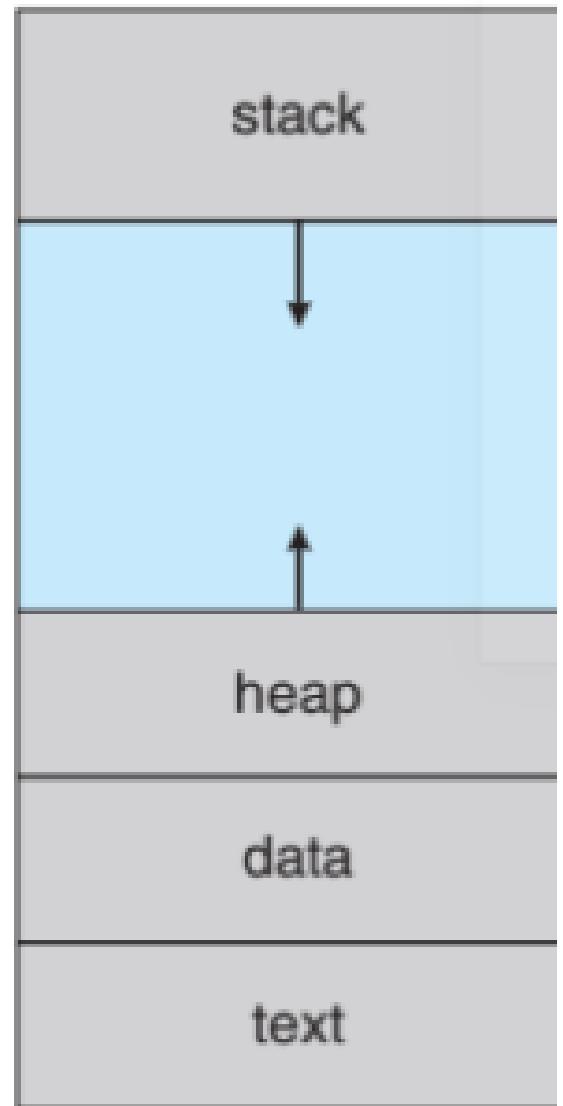
CÁC KHÁI NIỆM CƠ BẢN

- G
- Chương trình: thực thể bị động lưu trên đĩa
 - Tiến trình: { thực thể chủ động
 chương trình đang thực thi

khi file thực
thi được nạp
-> bộ nhớ



max





CÁC KHÁI NIỆM CƠ BẢN

Các bước khởi tạo tiến trình

1.

Cấp định dạng
duy nhất cho
tiến trình

2.

Cấp không
gian để nạp
tiến trình

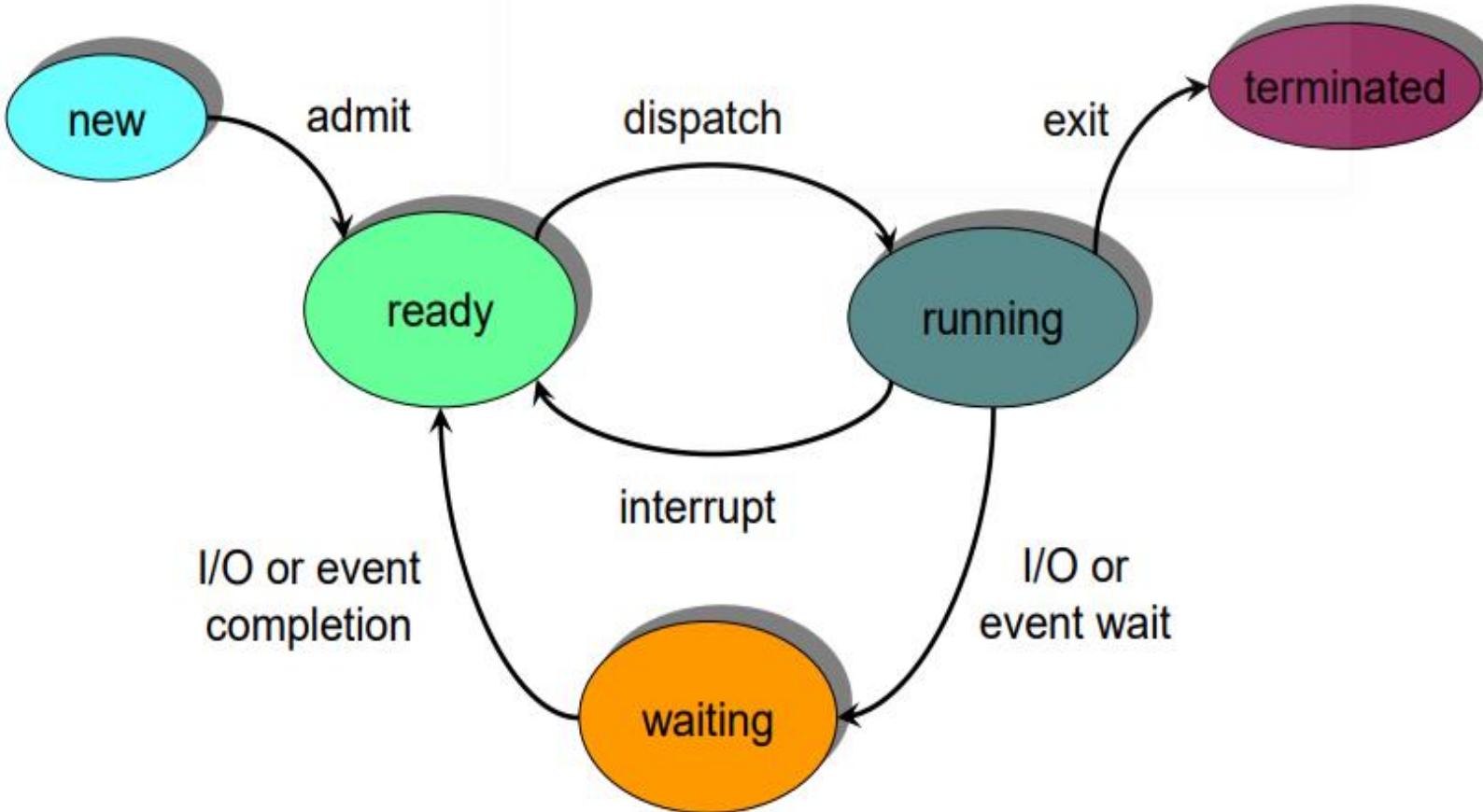
3.

Khởi tạo PCB

4.

Thêm các mối
liên hệ cần
thiết

TRẠNG THÁI TIẾN TRÌNH



Chuyển đổi giữa các trạng thái của tiến trình



TRẠNG THÁI TIẾN TRÌNH

Cho chương trình sau:

```
#include <stdio.h>

void main() {
    printf("Ban hoc tap khoa Ky Thuat May
Tinh");
    printf("chuc cac ban thi tot!");
    exit(0);
}
```



Tiến trình khi chạy từ chương trình trên đã trải qua những trạng thái nào?



new – ready – running
– waiting – ready –
running – waiting –
ready – running –
terminated



TRẠNG THÁI TIẾN TRÌNH

```
int main (int argc, char ** argv)
{
    Int a, b,i;
    For( i =4 ; i <= 15 ; i++) {
        if (i % 3==0)
        {
            Prinf(" Hello");
        }
        else {
            a++;
        }
    }
}
```



new – ready – **running** –
waiting – ready – **running** –
terminated

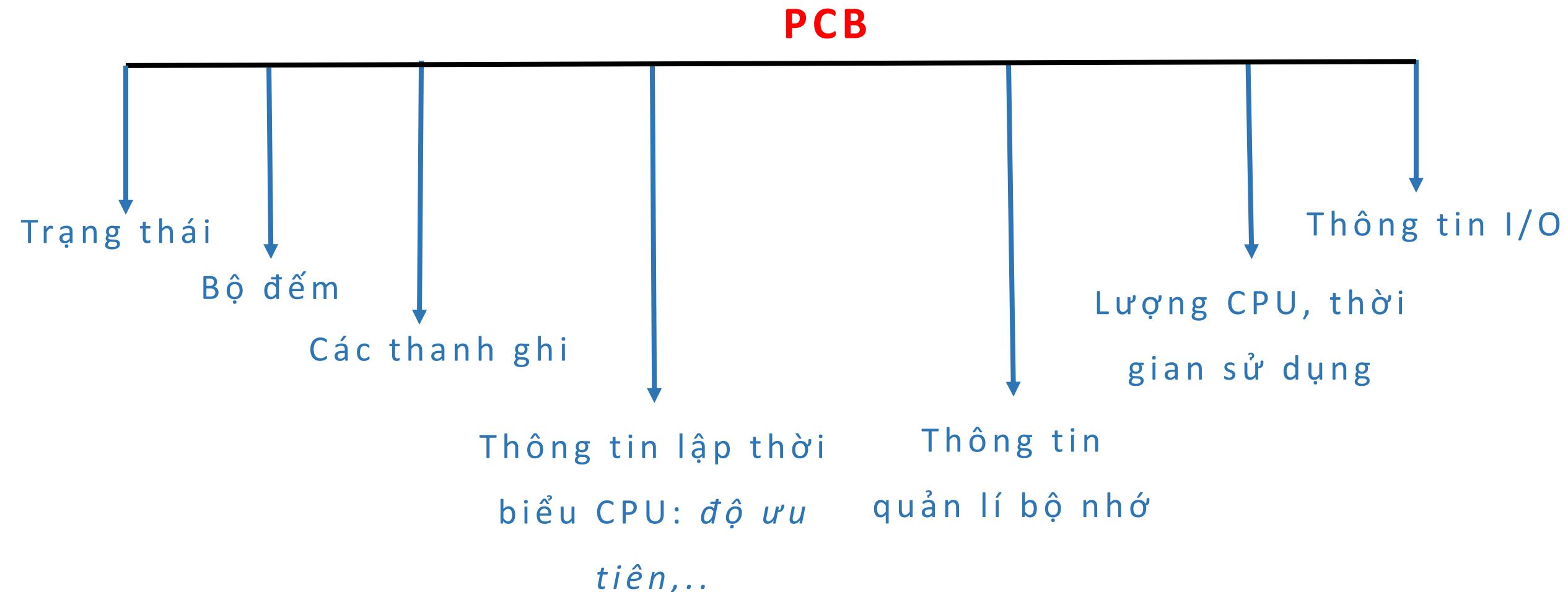


Khi kết thúc tiến trình, chương trình đã nằm trong
hàng đợi running bao nhiêu lần?



PROCESS CONTROL BLOCK

Mỗi tiến trình được cấp 1 PCB





ĐỊNH THỜI TIẾN TRÌNH

Định thời

Đa chương

Chia thời

Nhiều tiến trình chạy tại 1 thời điểm

Mục tiêu: tận dụng tối đa CPU

User tương tác với mỗi chương trình
đang thực thi

Mục tiêu: tối thiểu thời gian đáp ứng

Scheduling

Long-term | Xác định chương trình được vào thực thi (new -> ready)

Medium-term | Xác định tiến trình được đưa vào / đưa ra vùng nhớ chính

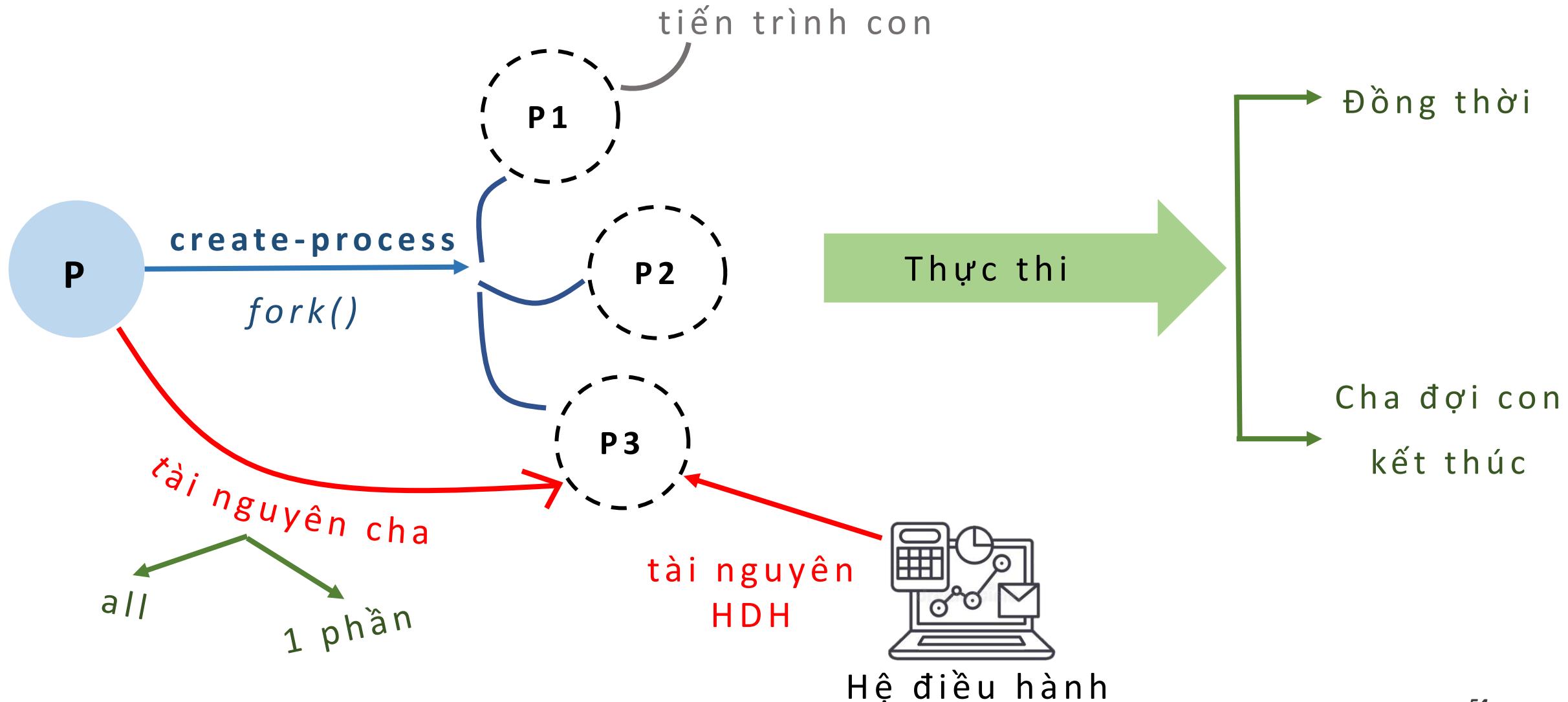
Short-term
(CPU Scheduler) | Xác định tiến trình tiếp theo trong ready queue được chiếm
CPU để thực thi



CÁC TÁC VỤ ĐỐI VỚI TIẾN TRÌNH

Tạo tiến trình mới

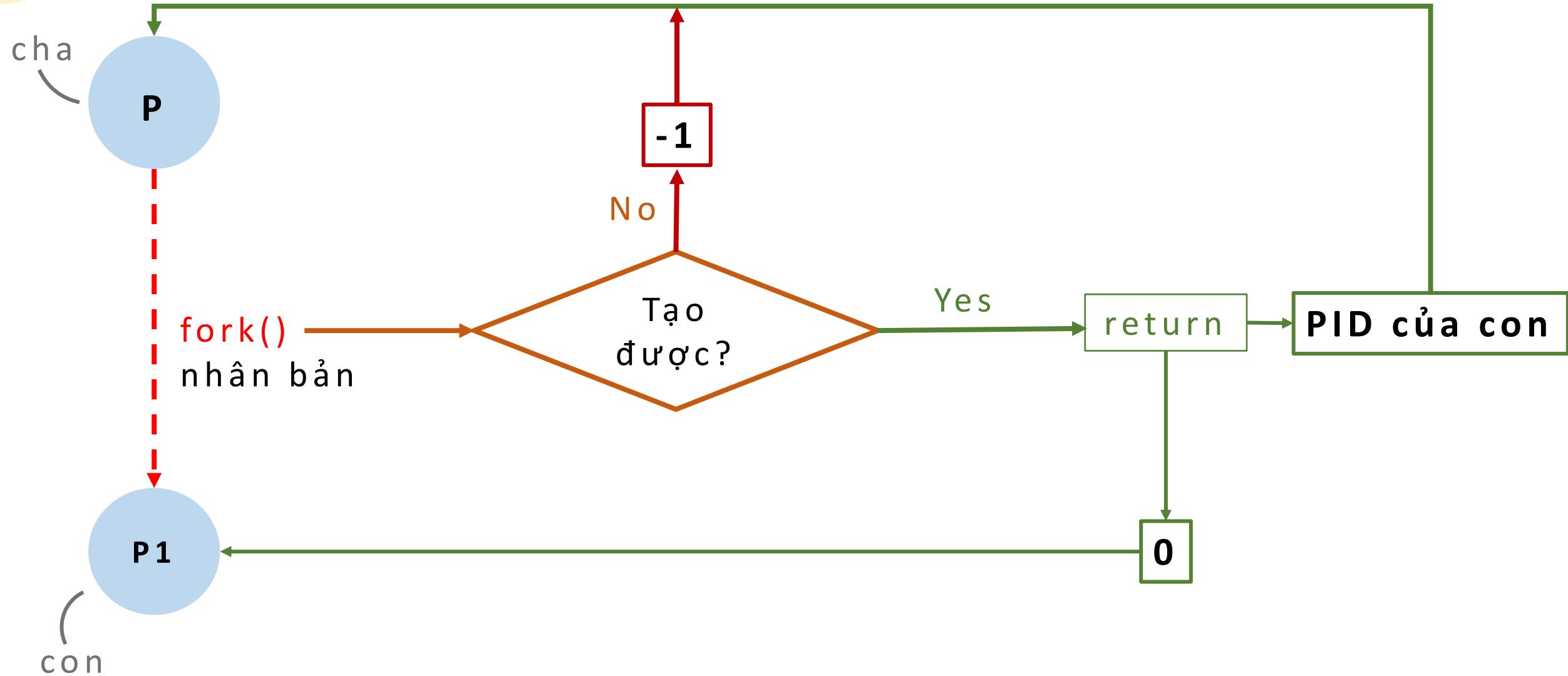
Kết thúc tiến trình





CÁC TÁC VỤ ĐỐI VỚI TIẾN TRÌNH

Ví dụ trong Linux/Unix





Ví dụ 1:

Có bao nhiêu chữ “hello” được sinh ra?

```
#include <stdio.h>
#include <sys/types.h>
int main()
{
    fork();
    fork();
    fork();
    printf("hello\n");
    return 0;
}
```

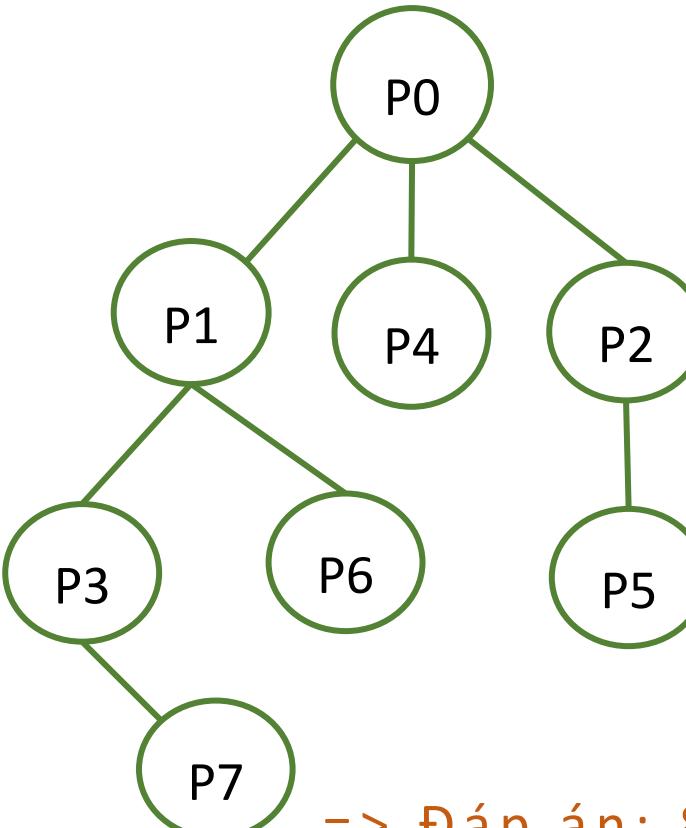
Số chữ “hello”
được in ra = số
tiến trình được
tạo ra (gồm cả
tiến trình cha)

Công Thức

Tổng số Process= 2^n

$n = \text{số lần gọi fork()}$

Cây tiến trình



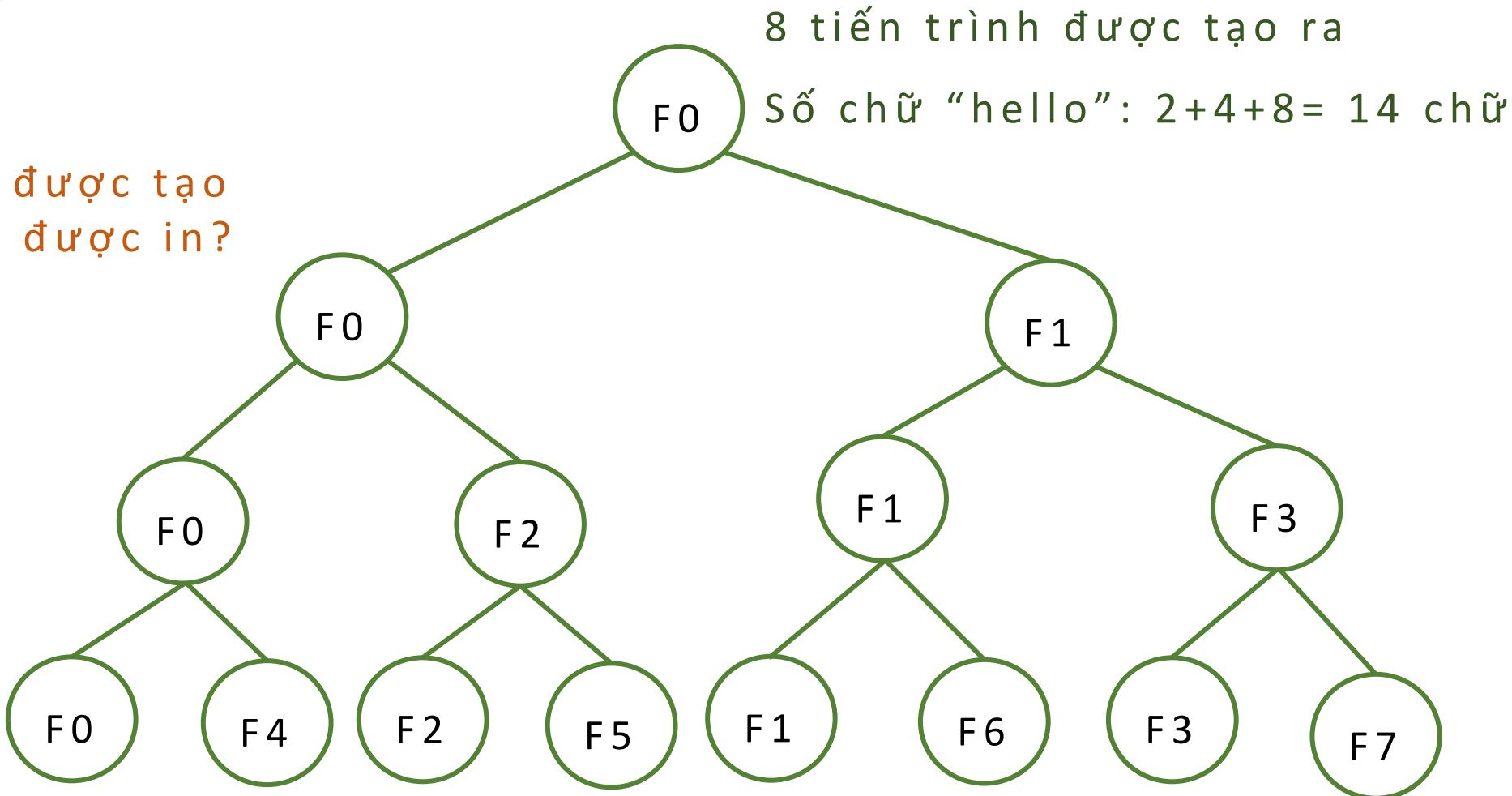
=> Đáp án: 8 chữ hello



Ví dụ 2: Hàm fork() trong vòng lặp

Tìm số tiến trình được tạo và số chữ “hello” được in?

```
#include <stdio.h>
#include <unistd.h>
int main()
{
    int i;
    for (i = 0; i < 3; i++){
        fork();
        printf("hello\n");
    }
    return 0;
}
```



8 tiến trình được tạo ra

Số chữ “hello”: $2+4+8= 14$ chữ

Công thức cấp số nhân:
 $\text{Tổng} = 2^1 + 2^2 + \dots + 2^n = u_1(q^n - 1)/(q-1)$



Ví dụ 3:

Cho đoạn chương trình sau:

```
if (fork() == 0) {  
    a = a + 5;  
    printf("%d, %d\n", a, &a);  
}  
else {  
    a = a - 5;  
    printf("%d, %d\n", a, &a);  
}
```

u, v là các giá trị được in bởi tiến trình cha
x, y là các giá trị được in bởi tiến trình con

Điều nào sau đây là ĐÚNG?

- (A) $u = x + 10$ và $v = y$ (C) $u + 10 = x$ và $v = y$
(B) $u = x + 10$ và $v! = y$ (D) $u + 10 = x$ và $v! = y$

fork () trả về 0 cho tiến trình
con và PID của con cho cha.

Ta có: $x = a + 5$; $u = a - 5$;
 $\Rightarrow x = u + 10$

Địa chỉ thực của 'a' trong cha
và con phải khác nhau. Nhưng
chương trình này truy cập các
địa chỉ ảo. con nhận được một
bản sao chính xác của cha và
địa chỉ ảo của 'a' không thay
đổi $\Rightarrow v=y$



CÁC TÁC VỤ ĐỐI VỚI TIẾN TRÌNH

Tạo tiến trình mới

Kết thúc tiến trình

tự kết thúc

thực hiện đến lệnh cuối

yêu cầu hệ điều hành xoá bằng lệnh exit()

Kết thúc
tiến trình

do tiến trình
khác có quyền
kết thúc nó

Gọi system routine abort

(tham số là pid của tiến trình cần kết thúc)

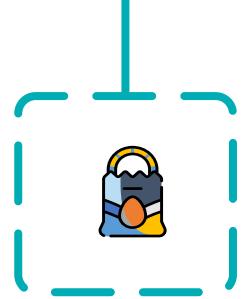
Tiến trình cha kết thúc

=> các tiến trình con của nó kết thúc

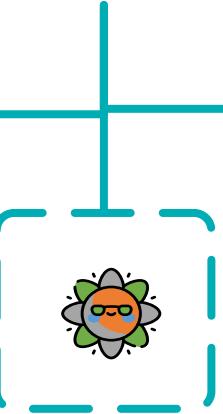


CỘNG TÁC GIỮA CÁC TIẾN TRÌNH

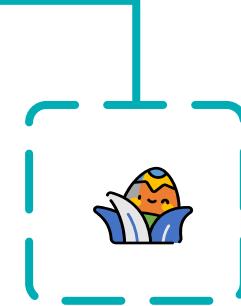
Khi thực thi, các tiến trình có thể cộng tác để hoàn thành công việc, nhằm



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



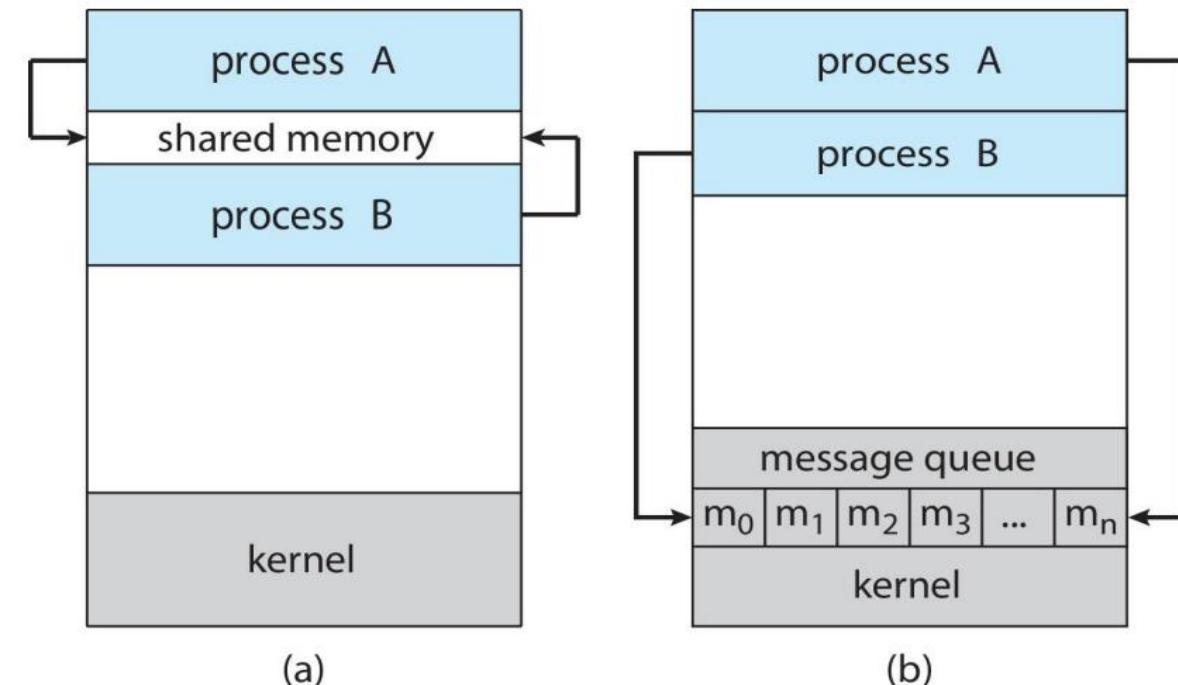
GIAO TIẾP LIÊN TIẾN TRÌNH

Hai mô hình IPC

(a) Shared memory.

(b) Message passing.

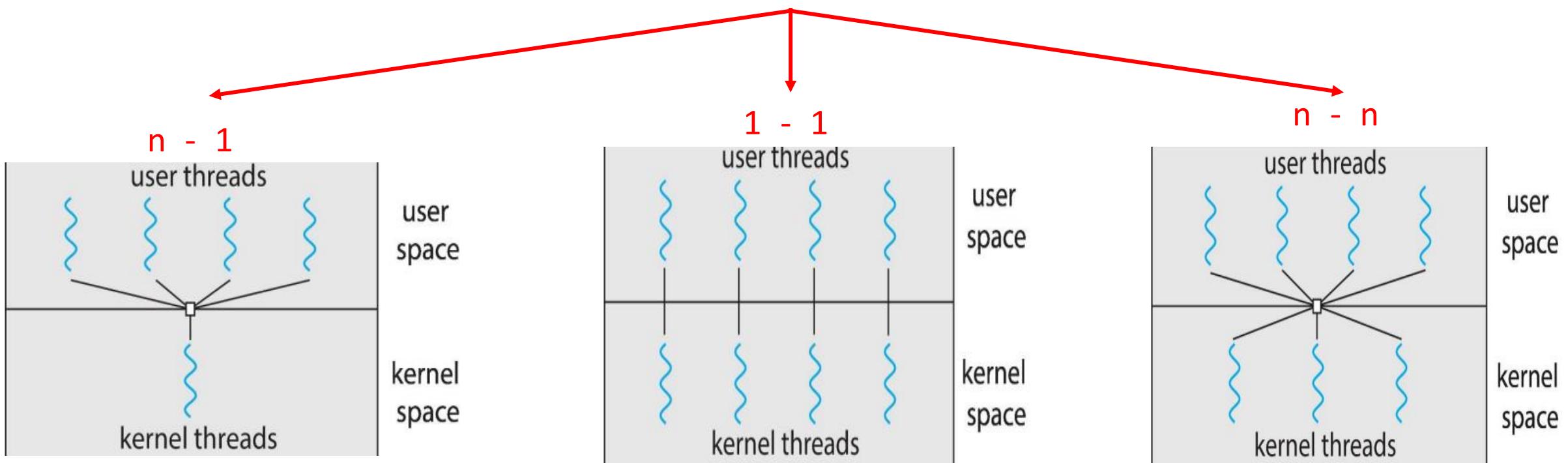
Giao tiếp liên tiến trình
dùng cơ chế IPC





đơn vị cơ bản sử dụng CPU
thread ID, PC, Registers, Stack, chia sẻ chung code, data, files }
Tiểu trình

Các mô hình đa tiểu trình





CÂU HỎI TRẮC NGHIỆM

Câu 1: Khi tiến trình được nạp vào bộ nhớ, stack section của nó không chứa thành phần:

- A. Biến cục bộ
- B. Địa chỉ trả về
- C. Biến toàn cục**
- D. Tham số truyền cho hàm

Câu 2: Trong mô hình đa tiểu trình, các tiểu trình bên trong tiến trình P có thể chia sẻ chung thành phần nào của P?

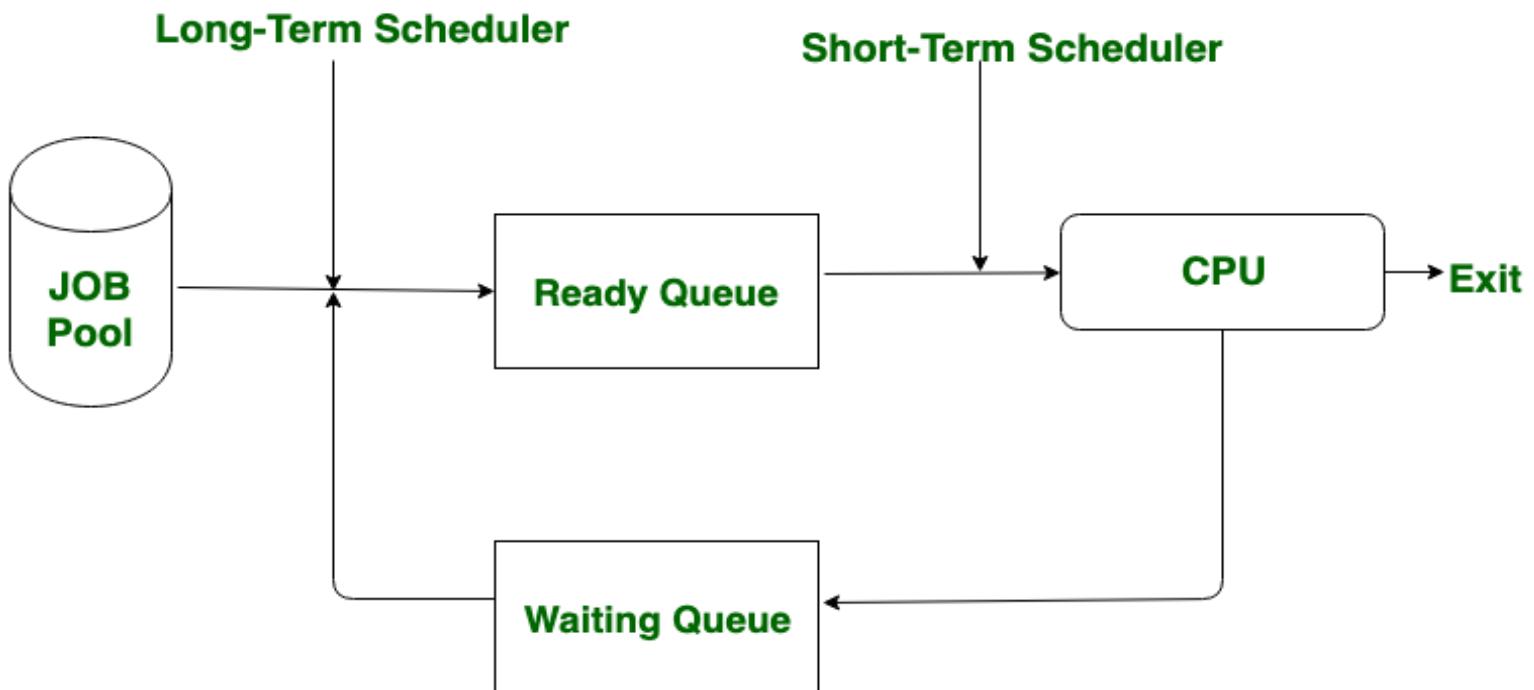
- A. Cả 3 thành phần trên
- B. Biến toàn cục**
- C. Bộ nhớ stack
- D. Thanh ghi



CÂU HỎI TRẮC NGHIỆM

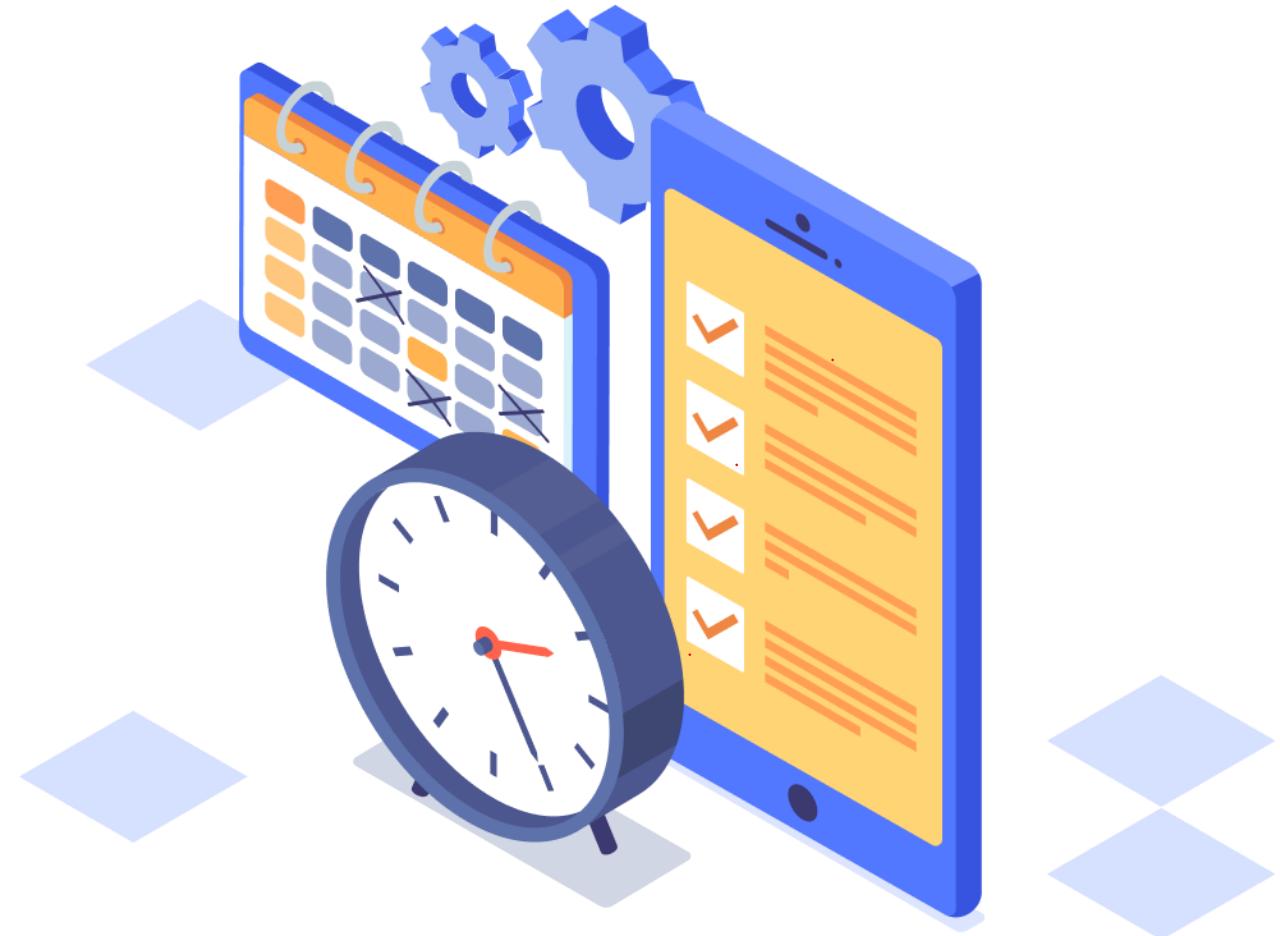
Câu 3: Bộ định thời cho phép process từ trạng thái new ->ready?

- A. Long-term scheduler
- B. Tất cả đều đúng
- C. Medium-term scheduler
- D. Short-term scheduler



CHƯƠNG 4

ĐỊNH THỜI CPU



Những vấn đề cần nắm



- Biết được các khái niệm cơ bản về định thời
- Hiểu được các tiêu chuẩn định thời CPU
- Hiểu được các giải thuật định thời
- Vận dụng các giải thuật định thời để làm bài tập và mô phỏng

NỘI DUNG

ĐỊNH THỜI CPU

Các khái niệm cơ bản về định thời

Các bộ định thời

Các tiêu chuẩn định thời CPU

Các giải thuật định thời

- pFirst-Come, First-Served (FCFS)
- pShortest Job First (SJF)
- pShortest Remaining Time First (SRTF)
- pPriority Scheduling



CÁC KHÁI NIỆM CƠ BẢN

Trong các hệ thống multitasking:

- Thực thi nhiều chương trình đồng thời làm tăng hiệu suất hệ thống
- Tại mỗi thời điểm, chỉ có 1 process được thực thi

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

➡ **Chiến lược định thời CPU**

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



Chương 4. Định thời CPU

CÁC BỘ ĐỊNH THỜI

Phân loại

Long-term

Xác định chương trình được vào thực thi (new -> ready)

Điều khiển mức độ multiprogramming của hệ thống

Xen lấn CPUboud và I/O-bound process

Medium-term

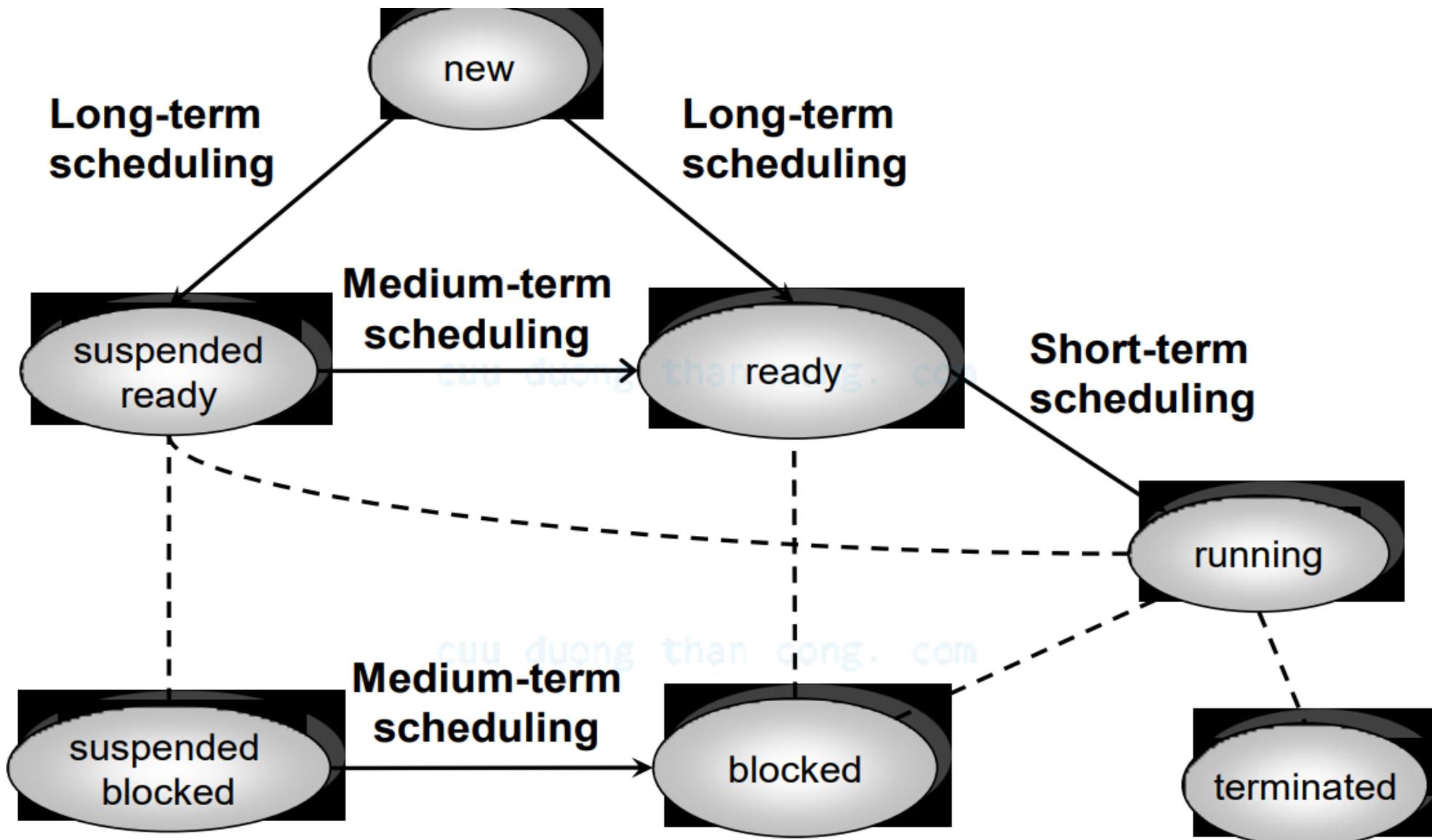
Xác định tiến trình được đưa vào / đưa ra vùng nhớ chính

Thực hiện và thảo luận ở phần quản lý bộ nhớ

Short-term

Xác định tiến trình trong ready queue được chiếm CPU

CÁC BỘ ĐỊNH THỜI





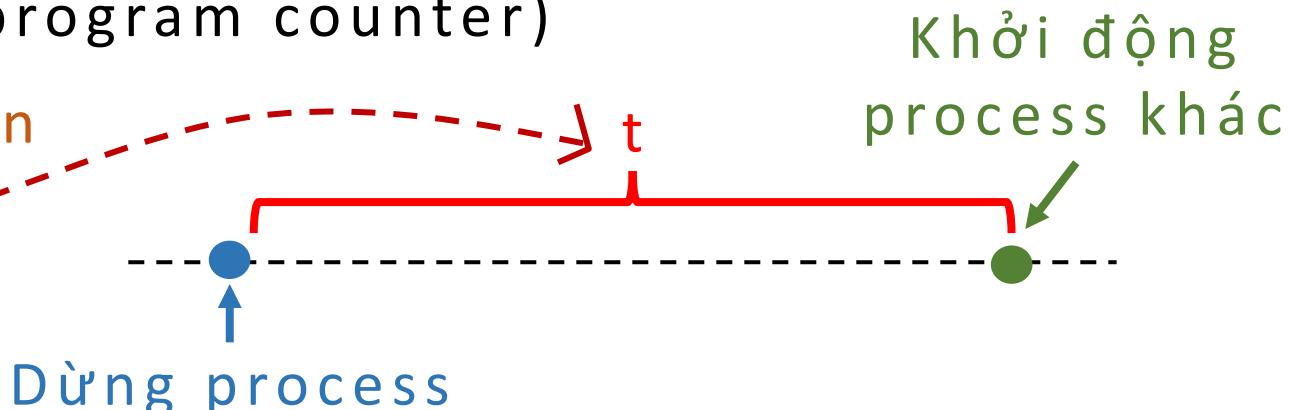
BỘ ĐỊNH THỜI

- Bộ định thời chuyển quyền điều khiển CPU -> process được chọn
Bao gồm:

- Chuyển ngữ cảnh (sử dụng thông tin ngữ cảnh trong PCB)
- Chuyển chế độ người dùng
- Nhảy đến vị trí thích hợp trong chương trình ứng dụng để khởi động lại chương trình (program counter)

- Công việc này gây ra phí tổn

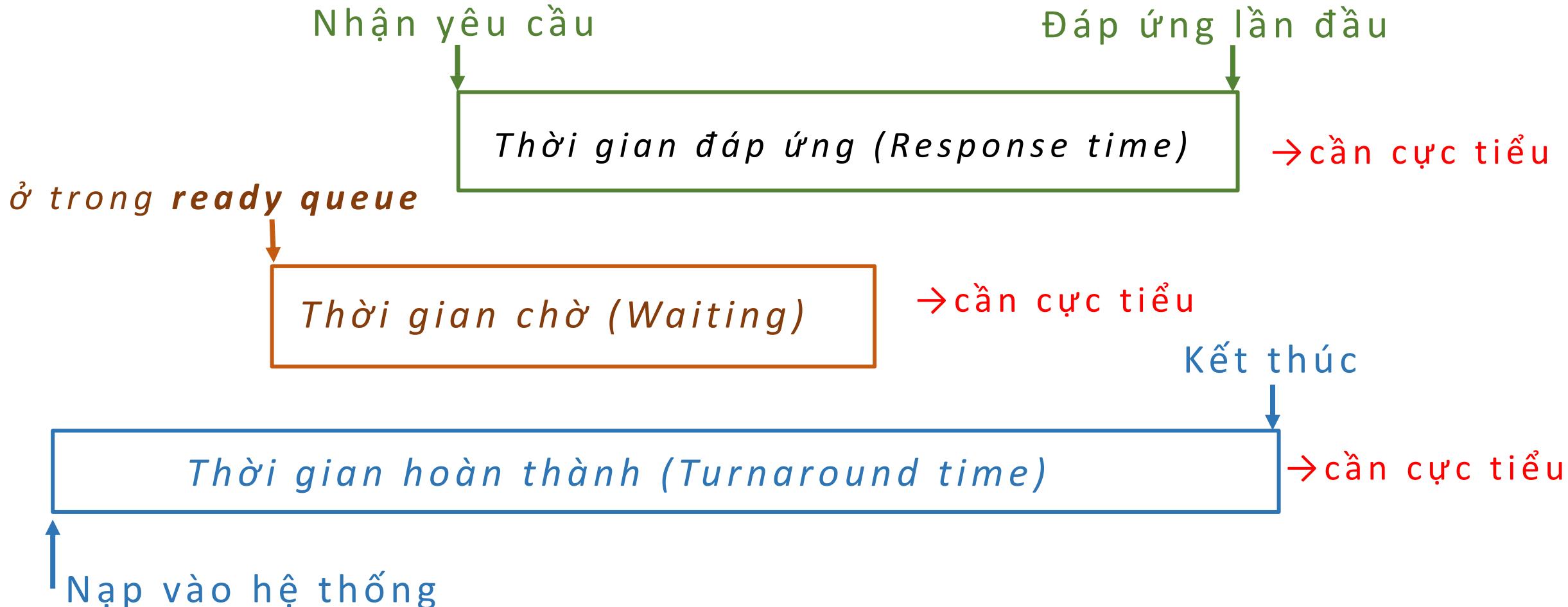
- Dispatch latency:





CÁC TIÊU CHUẨN ĐỊNH THỜI CPU

Hướng người dùng (User-oriented)





CÁC TIÊU CHUẨN ĐỊNH THỜI CPU

Hướng hệ thống (System-oriented)

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

CÁC GIẢI THUẬT ĐỊNH THỜI

Hai yếu tố của giải thuật định thời

■ *Hàm chọn lựa:* chọn process trong ready queue được thực thi (dựa trên độ ưu tiên, yêu cầu tài nguyên, đặc điểm thực thi,...)

■ *Chế độ quyết định:* chọn thời điểm dùng hàm chọn lựa

Trung dụng
(Preemptive)

Không trung dụng
(Non-preemptive)





CÁC GIẢI THUẬT ĐỊNH THỜI

Chế độ quyết định (decision mode)

Không trung dụng (Non-preemptive):

- Ở trạng thái running: thực thi đến khi kết thúc hoặc bị blocked

Trung dụng (Preemptive):

- Ở trạng thái running: có thể bị ngắt và chuyển về ready
- Chi phí cao hơn non-preemptive nhưng $T_{đáp ứng}$ tốt hơn

Vì ngăn 1 process độc
chiếm CPU quá lâu



CÁC GIẢI THUẬT ĐỊNH THỜI

Chế độ quyết định (decision mode)

■ Hàm định thời được thực hiện khi

- (1) Từ trạng thái running -> waiting
 - (2) Từ trạng thái running -> ready
 - (3) Từ trạng thái waiting / new -> ready
 - (4) Kết thúc thực thi
- (1) & (4) không cần lựa chọn loại định thời biểu, (2) & (3) cần

■ Trường hợp 1, 4: non-preemptive

■ Trường hợp 2, 3: preemptive



CÁC GIẢI THUẬT ĐỊNH THỜI

First-Come,
First-Served
(FCFS)

Shortest Job
First (SJF)

Shortest
Remaining
Time First
(SRTF)

Round-Robin (RR)

Priority
Scheduling

Highest
Response
Ratio Next
(HRRN)

Multilevel
Queue

Multilevel
Feedback
Queue

CÁC GIẢI THUẬT ĐỊNH THỜI

Giải thuật	Chế độ		Đặc điểm
	Non-prem	Prem	
FCFS	X		Vào trước cấp trước Không nhường Dùng FIFO
SJF	X		Ngắn nhất trước Không nhường Liên quan chiều dài thời gian dùng CPU cho lần tiếp theo
SRJF		X	Ngắn nhất trước Nhường khi P mới trong queue có $t' < t$ còn lại

CÁC GIẢI THUẬT ĐỊNH THỜI

Giải thuật	Chế độ		Đặc điểm
	Non-prem	Prem	
RR	X		<p>Độ ưu tiên ngang nhau</p> <p>Cấp trong khoảng thời gian q hết q thì bị đoạt quyền và về cuối queue</p> <p>T chờ lớn, T đáp ứng nhỏ</p> <p>Không có P nào đợi quá $(n-1).q$ thời gian</p>
Priority	X	X	<p>Cấp theo độ ưu tiên</p> <p>Có thể bị trì hoãn vô hạn</p>
Highest response	X		<p>Chọn T đáp ứng lớn nhất</p> <p>Ratio = (tg chờ + expected service)/expected service</p> <p>Giúp tránh chờ vô hạn</p>

***Độ ưu tiên** phụ thuộc: yêu cầu bộ nhớ, số file được mở, tỉ lệ thời gian



CÁC GIẢI THUẬT ĐỊNH THỜI

Giải thuật	Chế độ		Đặc điểm
	Non-prem	Prem	
Multi-level queue	X		<p>chia nhiều queue</p> <p>P không thể nhảy sang queue khác</p> <p>mỗi queue dùng giải thuật khác nhau</p> <p>1 thời điểm chỉ có 1 P thực thi</p> <p>=> các queue k thể chạy song song</p> <p>HDH định thời các queue bằng time slide (chọn queue theo độ ưu tiên hoặc mỗi queue được cấp 1 khoảng thời gian t)</p>
Multi-level feedback queue		X	<p>cho phép P nhảy sang queue khác</p> <p>phân loại P bằng CPU-burst</p>

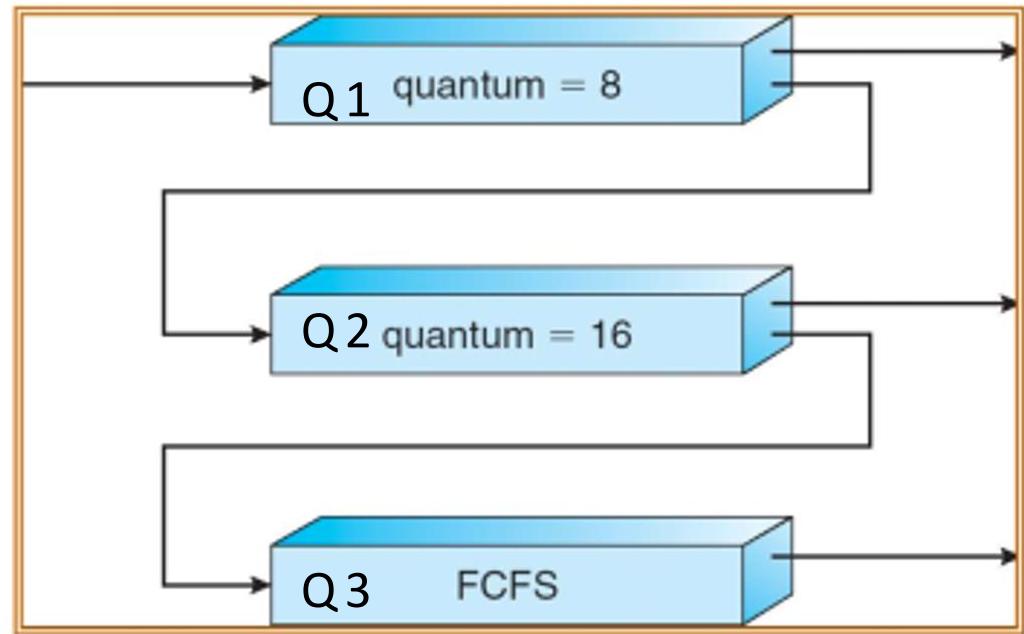
CÁC GIẢI THUẬT ĐỊNH THỜI

Ví dụ

Hệ thống sử dụng Multilevel feedback queue gồm 3 queue.

Tại t0: A vào hệ thống, thời gian thực thi A: tA=10ms.

Hỏi: từ khi vào hệ thống, A sẽ nằm ở queue nào?

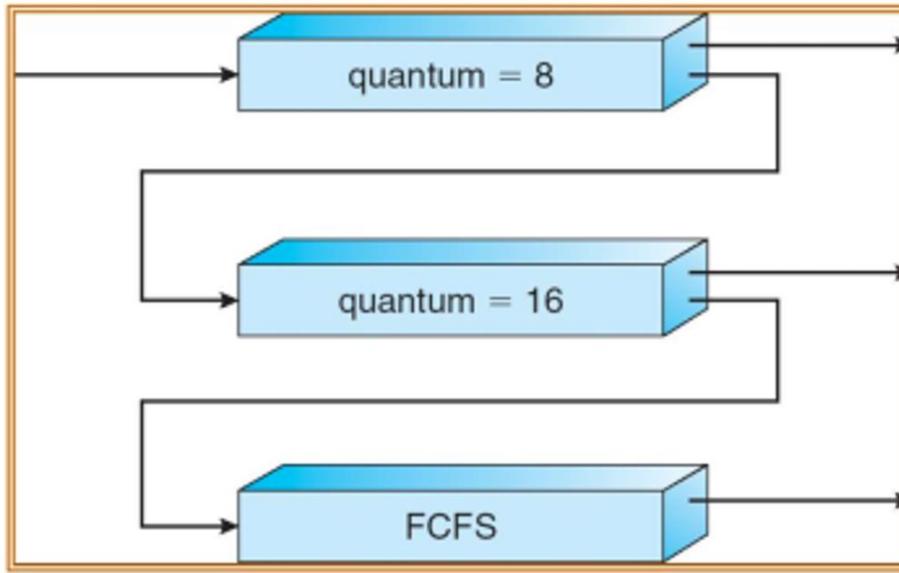


CÁC GIẢI THUẬT ĐỊNH THỜI

Trả lời: A nằm ở Q1 và Q2

Giải thích:

- Ban đầu: A vào Q1
- $t_A > \text{quantum time}(Q1) \rightarrow$ ngắt quãng \rightarrow quay về Ready queue
- Lặp vô hạn tại Ready queue \rightarrow queue dần đầy \rightarrow A được đẩy sang Q2
- $t_A < \text{quantum time}(Q2) \rightarrow$ không quay về Ready queue
- A được thực thi xong và kết thúc tại Q2



Ví dụ



Một số lý thuyết khác

Giải thuật RR:

Time slice ngắn: đáp ứng nhanh

→ Vấn đề: có nhiều chuyển ngữ cảnh. Phí tổn cao.

Time slice dài: throughput tốt hơn nhưng $T_{đáp\ ứng}$ lớn

→ Nếu time slice quá lớn, RR trở thành FCFS

✓ Nhược điểm: Các process dạng CPU-bound vẫn còn được “ưu tiên”

CÁC GIẢI THUẬT ĐỊNH THỜI

Giải thuật định thời nào

là tốt nhất?



So sánh các giải thuật

Phụ thuộc các yếu tố sau:

- Tần suất tải việc
- Sự hỗ trợ của phần cứng
- Tiêu chuẩn định thời: response time, hiệu suất CPU, throughput,...
- Phương pháp định lượng so sánh



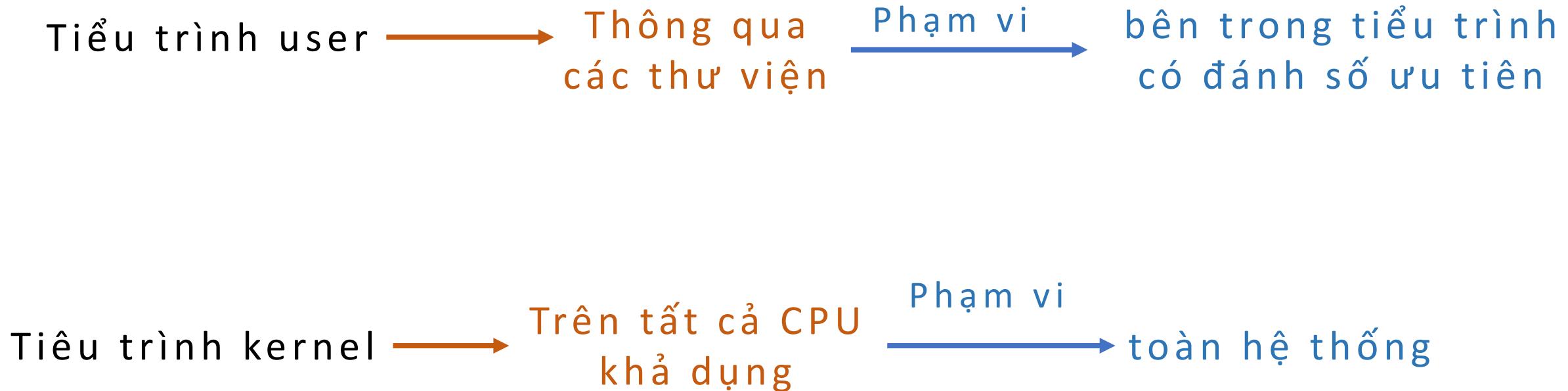
GIỚI THIỆU CÁC BỘ ĐỊNH THỜI

- 1. Định thời tiểu trình
- 2. Định thời đa bộ xử lý
- 3. Định thời thời gian thực
- 4. Định thời hệ điều hành
 - Linux
 - Windows
 - Solaris



GIỚI THIỆU CÁC BỘ ĐỊNH THỜI

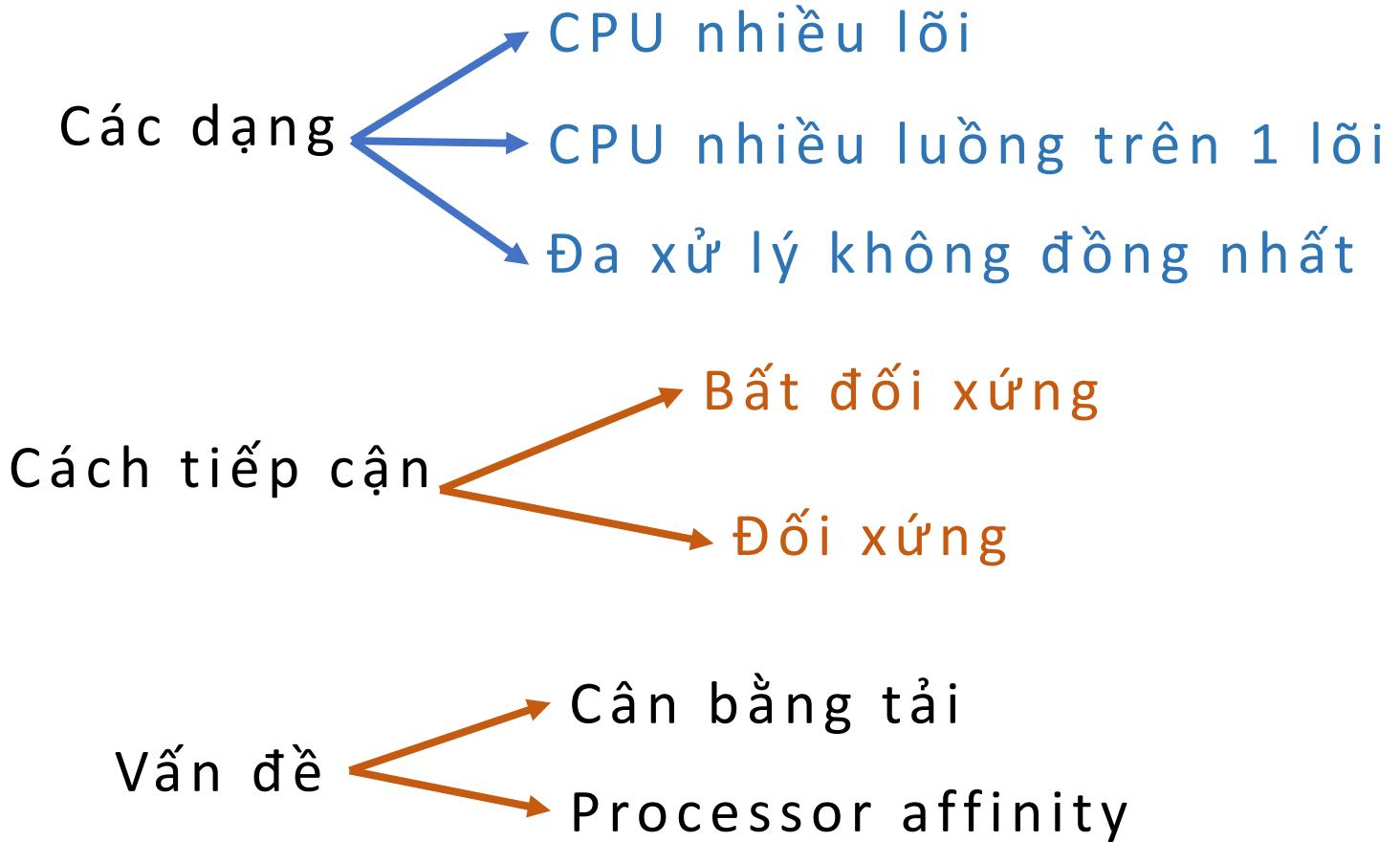
- ✓ Có trên các HDH hiện đại



1. Định thời tiểu trình

GIỚI THIỆU CÁC BỘ ĐỊNH THỜI

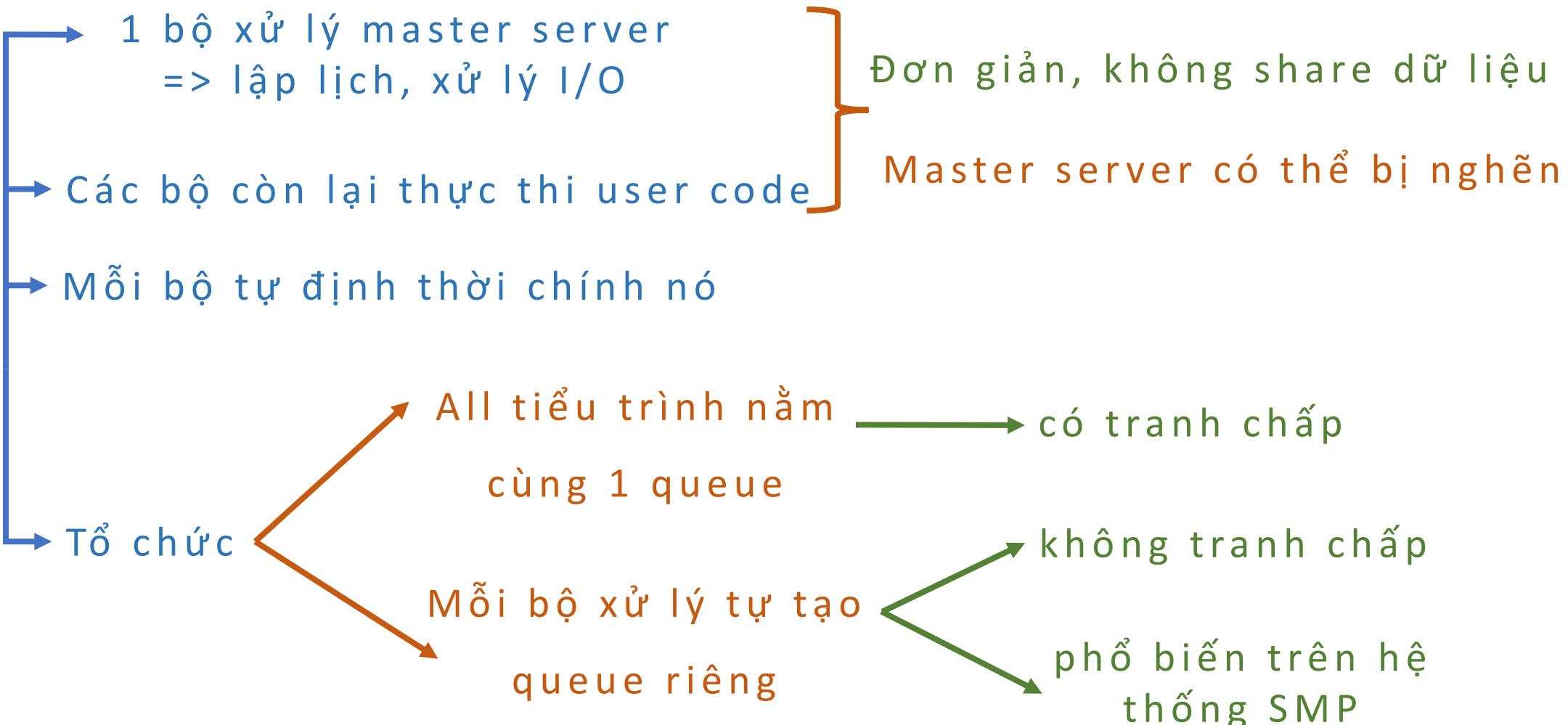
2. Định thời đa bộ xử lý





GIỚI THIỆU CÁC BỘ ĐỊNH THỜI

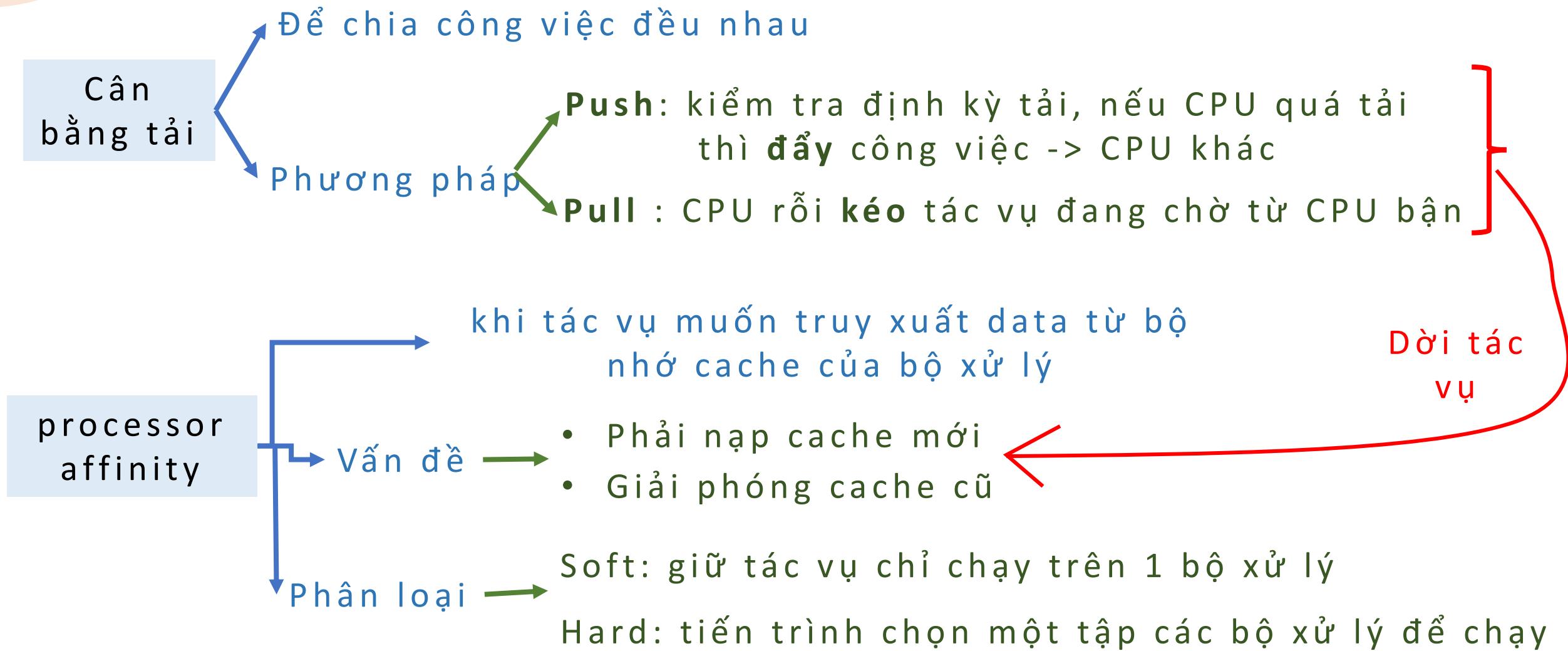
2. Định thời đa bộ xử lý

Bất đối
xứng



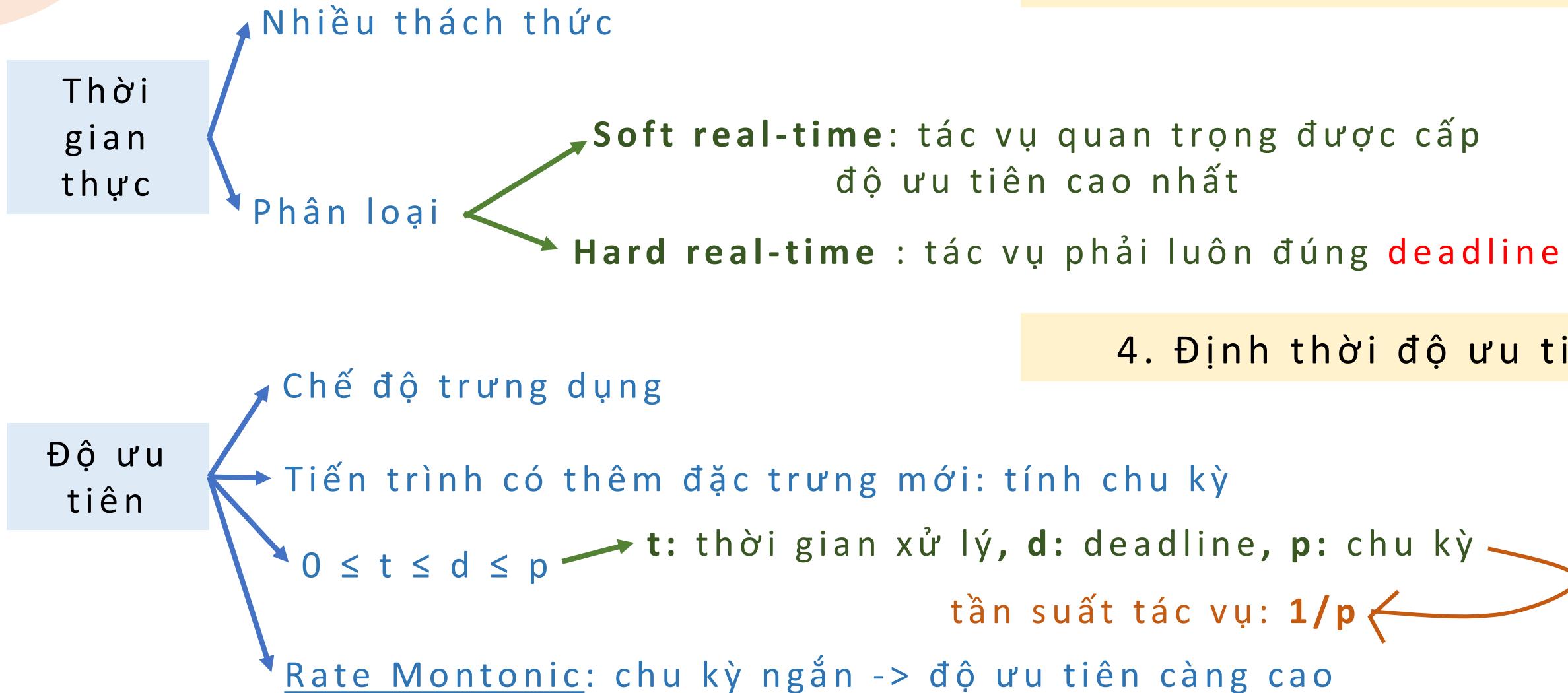
GIỚI THIỆU CÁC BỘ ĐỊNH THỜI

2. Định thời đa bộ xử lý





GIỚI THIỆU CÁC BỘ ĐỊNH THỜI



3. Định thời thời gian thực

4. Định thời độ ưu tiên



GIỚI THIỆU CÁC BỘ ĐỊNH THỜI

5. Định thời hệ điều hành

Linux

Định thời UNIX tiêu chuẩn

Hỗ trợ kém hệ thống đa bộ xử lý

Hiệu năng kém nếu nhiều tiến trình

Nhân Linux 2.5

Phiên bản trước

Định thời theo độ ưu tiên, thời gian = hằng số, có trưng dụng

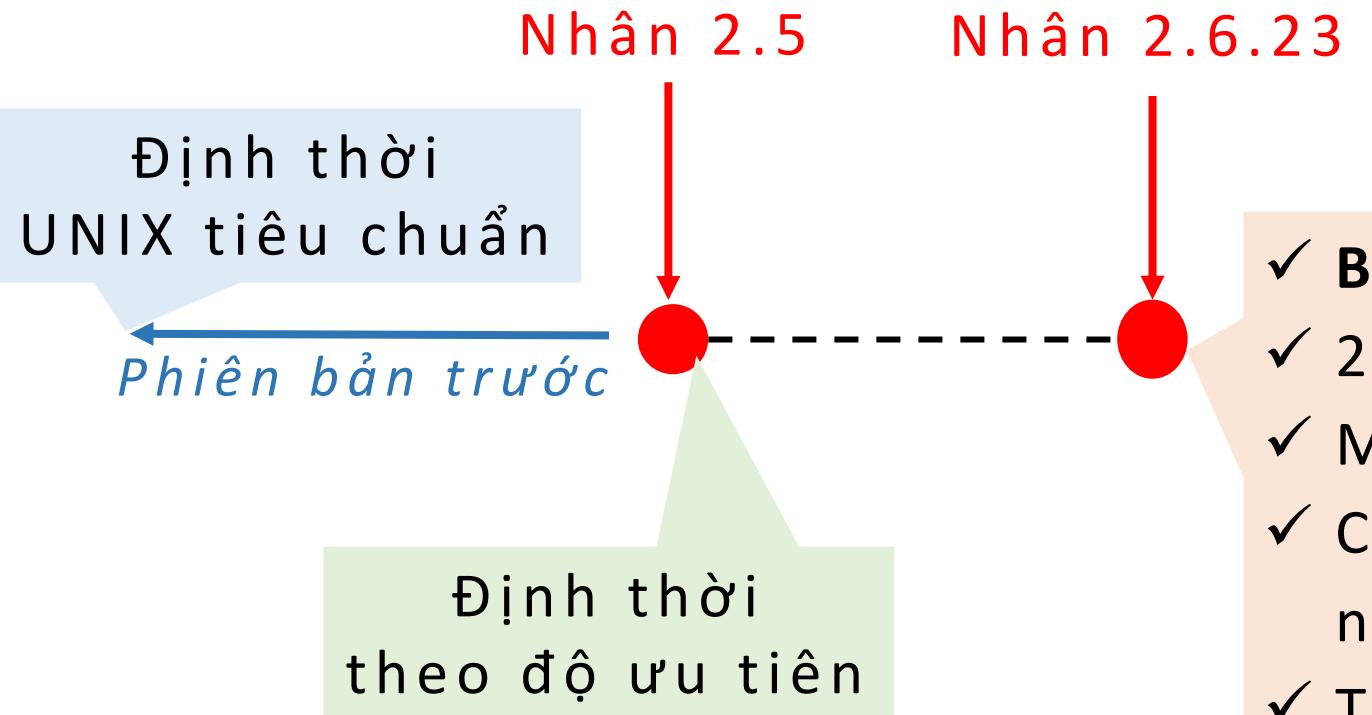
Hoạt động tốt với các hệ thống SMP

Đáp ứng kém với các tiến trình interactive.

GIỚI THIỆU CÁC BỘ ĐỊNH THỜI

5. Định thời hệ điều hành

Linux



- ✓ **Bộ định thời CFS (theo lớp)**
- ✓ 2 lớp có sẵn: default, real-time
- ✓ Mỗi lớp được gán một độ ưu tiên
- ✓ Chọn tác vụ có độ ưu tiên cao nhất của lớp ưu tiên cao nhất
- ✓ Thời gian sử dụng CPU: dựa trên chỉ số nice ∈ (-20;19)



GIỚI THIỆU CÁC BỘ ĐỊNH THỜI

5. Định thời hệ điều hành

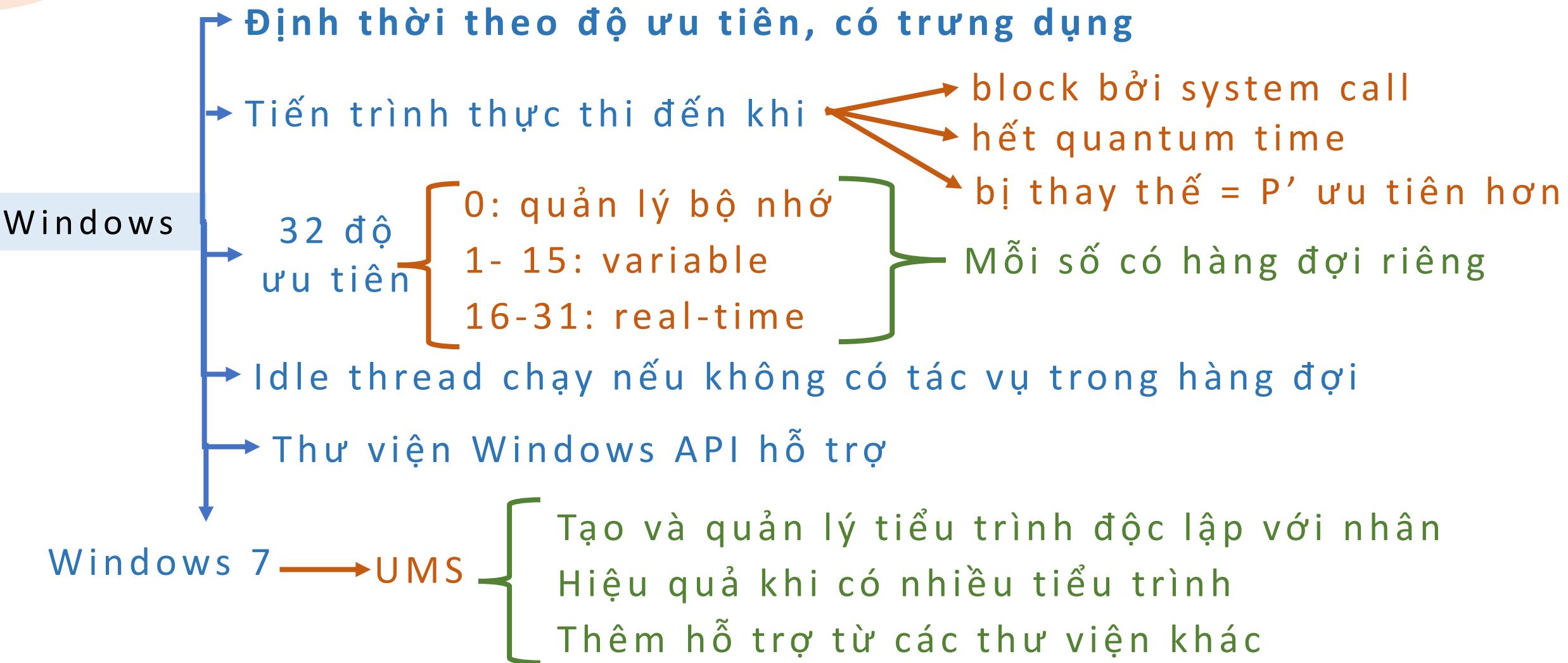
Android

- ✓ **Bộ định thời của Linux**
- ✓ Độ ưu tiên được phân chia theo nhóm của các tiến trình
- ✓ Thu hồi tài nguyên = kill tiến trình dựa trên độ ưu tiên



GIỚI THIỆU CÁC BỘ ĐỊNH THỜI

5. Định thời hệ điều hành



GIỚI THIỆU CÁC BỘ ĐỊNH THỜI

5. Định thời hệ điều hành

Solaris

Định thời theo độ ưu tiên

→ 6 lớp
(Độ ưu tiên
khác nhau)

Time sharing – mặc định
Interactive
Real time
System
Fair Share
Fixed priority

giải thuật định thời

MFQ

→ Độ ưu tiên càng lớn => time slice càng nhỏ

Chọn độ ưu
tiên cao nhất

→ Chuyển độ ưu tiên theo lớp -> độ ưu tiên toàn cục

→ Tiến trình thực thi đến khi

block bởi system call
hết quantum time
bị thay thế = P' ưu tiên hơn

→ Các tiến trình cùng độ ưu tiên: dùng hàng đợi round-robin



TRAINING GIỮA KỲ II NĂM HỌC 2021-2022

HỆ ĐIỀU HÀNH
CẢM ƠN BẠN ĐÃ THAM GIA!

 Email : bht.ktmt@gmail.com

 Fanpage: www.facebook.com/bht.ktmt

ĐIỂM DANH
LẤY ĐIỂM RÈN LUYỆN NÀO

