BAN HỌC TẬP KHOA CÔNG NGHỆ PHẦN MỀM
CHUỖI TRAINING GIỮA HỌC KÌ II NĂM HỌC 2021-2022

ĐOÀN KHOA
CÔNG NGHỆ PHẦN MỀM
BAN HỌC TẬP

Sharing is learning

Ban học tập

Khoa Công Nghệ Phần Mềm
Trường ĐH Công Nghệ Thông Tin
ĐHQG Hồ Chí Minh

Email / Group

bht.cnpm.uit@gmail.com
fb.com/groups/bht.cnpm.uit
https://www.facebook.com/bhtcnpm

Training

Hệ điều hành

Hệ điều hành

Thời gian training: 19h30 ngày 07/04/2022

Trainer: Nguyễn Minh Hiếu - 20520183

Nguyễn Khắc Thái - 20521888

Đoàn Ngọc Như Quỳnh - 20520732

ĐOÀN KHOA CÔNG NGHỆ PHẦN MỀM

BAN HỌC TẬP

Sharing is learning

NỘI DUNG TRAINING

ĐOÀN KHOA CÔNG NGHỆ PHẦN MỀM
BAN HỌC TẬP
Sharing is learning

CHƯƠNG 1: TỔNG QUAN VỀ HỆ ĐIỀU HÀNH

CHƯƠNG 2: CẤU TRÚC HỆ ĐIỀU HÀNH

CHƯƠNG 3: QUẢN LÝ TIẾN TRÌNH

CHƯƠNG 4: ĐỊNH THỜI CPU

NỘI DUNG TRAINING

ĐOÀN KHOA CÔNG NGHỆ PHẦN MỀM
BAN HỌC TẬP
Sharing is learning

CHƯƠNG 1: TỔNG QUAN VỀ HỆ ĐIỀU HÀNH

I. Tổng quan cơ bản:

1. Khái niệm, mục tiêu:
2. Cấu trúc hệ thống máy tính:
3. Sự cần thiết của hệ điều hành:
4. Các chức năng chính của hệ điều hành:

I. Tổng quan cơ bản

ĐOÀN KHOA CÔNG NGHỆ PHẦN MỀM
BAN HỌC TẬP
Sharing is learning

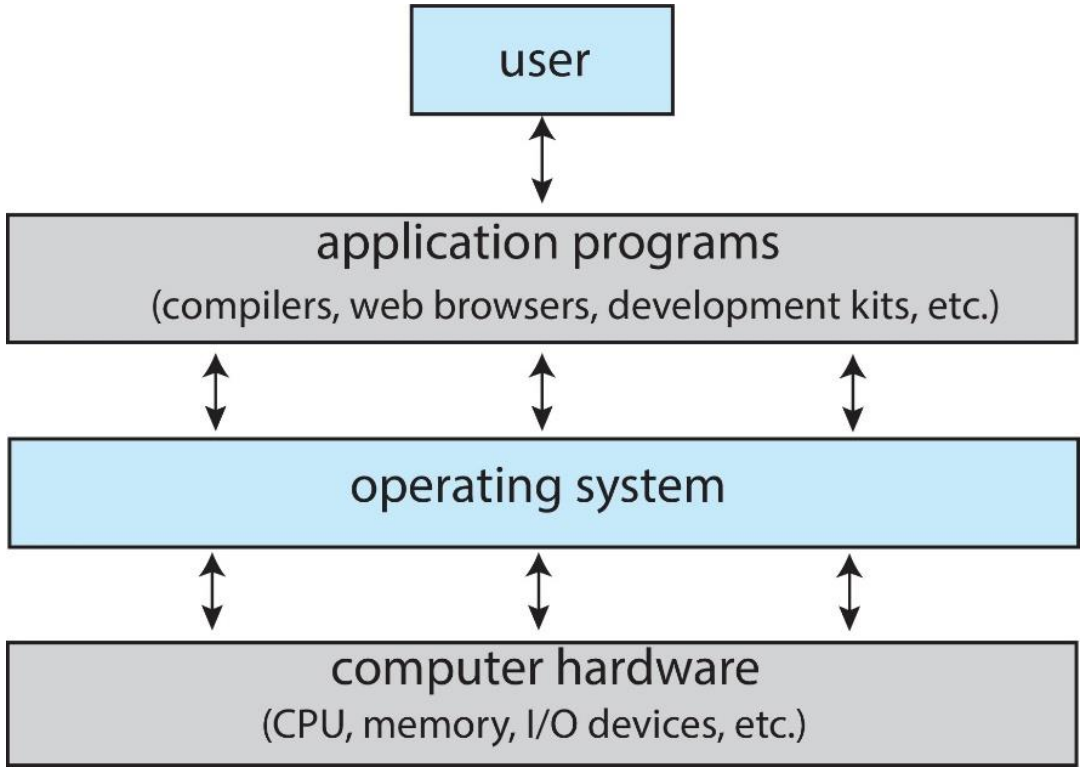
1. Khái niệm và mục tiêu:

- **Hệ điều hành**: chương trình trung gian giữa phần cứng máy tính và người sử dụng, điều khiển và phối hợp việc sử dụng các phần cứng, cung cấp dịch vụ cơ bản cho các ứng dụng.

- **Mục tiêu:**
  - Giúp người dùng dễ dàng sử dụng hệ thống.
  - Quản lý tài nguyên hiệu quả.

I. Tổng quan cơ bản

2. Cấu trúc hệ thống:



Users (people, machines, other computers)

Chương trình ứng dụng (application programs): Sử dụng hệ thống tài nguyên để giải quyết một bài toán tính toán nào đó của người sử dụng.
Ví dụ: compilers, database systems, video games, business programs

Hệ điều hành (operating system): Phân phối tài nguyên, điều khiển và phối hợp các hoạt động của các chương trình trong hệ thống.

Phần cứng (hardware): Bao gồm các tài nguyên cơ bản của máy tính như CPU, bộ nhớ, các thiết bị I/O

I. Tổng quan cơ bản

ĐOÀN KHOA CÔNG NGHỆ PHẦN MỀM
BAN HỌC TẬP
Sharing is learning

3. Sự cần thiết của hệ điều hành:

- Quản lý phần cứng máy tính.
- Kết nối các thiết bị phần cứng với nhau.
- Tương tác giữa các chương trình với nhau và với phần cứng.
- Là nơi để người dùng cài đặt các chương trình ứng dụng.
- Cung cấp giao diện cho người dùng.

I. Tổng quan cơ bản

ĐOÀN KHOA CÔNG NGHỆ PHẦN MỀM
BAN HỌC TẬP
Sharing is learning

4. Một số chức năng chính của hệ điều hành:

- Phân chia thời gian xử lý và định thời CPU.
- Phối hợp và đồng bộ giữa các processes.
- Quản lý tài nguyên hệ thống (thiết bị I/O, bộ nhớ,...).
- Kiểm soát truy cập, bảo vệ hệ thống.
- Duy trì sự nhất quán của hệ thống, kiểm soát lỗi và phục hồi.
- Cung cấp giao diện làm việc cho người dùng.

NỘI DUNG TRAINING

CHƯƠNG 1: TỔNG QUAN VỀ HỆ ĐIỀU HÀNH

II. Phân loại hệ điều hành

1. Góc độ loại máy tính:
2. Góc độ hình thức xử lý:

II. Phân loại hệ điều hành

1. Loại máy tính

Hệ điều hành
- Máy MainFrame
- Máy Server
- Máy tính cá nhân (PC, Laptop)
- Máy PDA (Phone, Tablet)
- Máy chuyên biệt (Car, TV)
- Thiết bị nhúng (RTOS)

II. Phân loại hệ điều hành

2. Hình thức xử lý

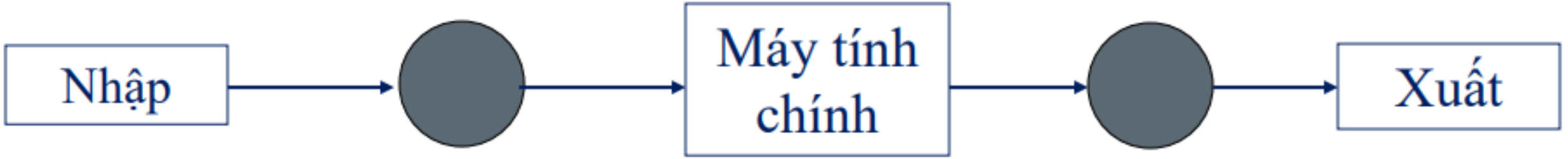
Hệ thống
- Xử lý theo chương trình (đơn hoặc đa chương)
- Chia sẻ thời gian – đa nhiệm
- Song song – đa xử lý
- Phân tán
- Nhúng thời gian thực

II. Phân loại hệ điều hành

2.1. Xử lý theo chương trinh:

a. Hệ thống đơn chương:

- Tác vụ được thi hành tuần tự.
- Bộ giám sát thường trực.
- VD: **DOS**



II. Phân loại hệ điều hành

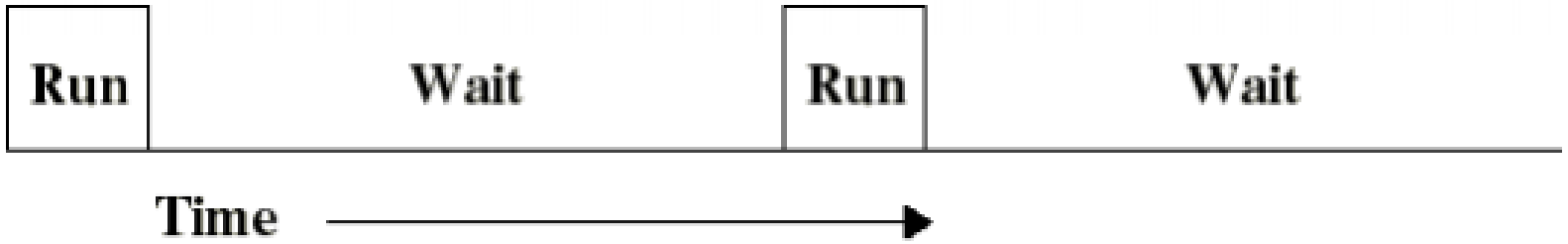
b. Hệ thống đa chương

- Nhiều công việc được nạp đồng thời vào bộ nhớ chính.
- Khi một tiến trình thực hiện I/O, một tiến trình khác được thực thi.

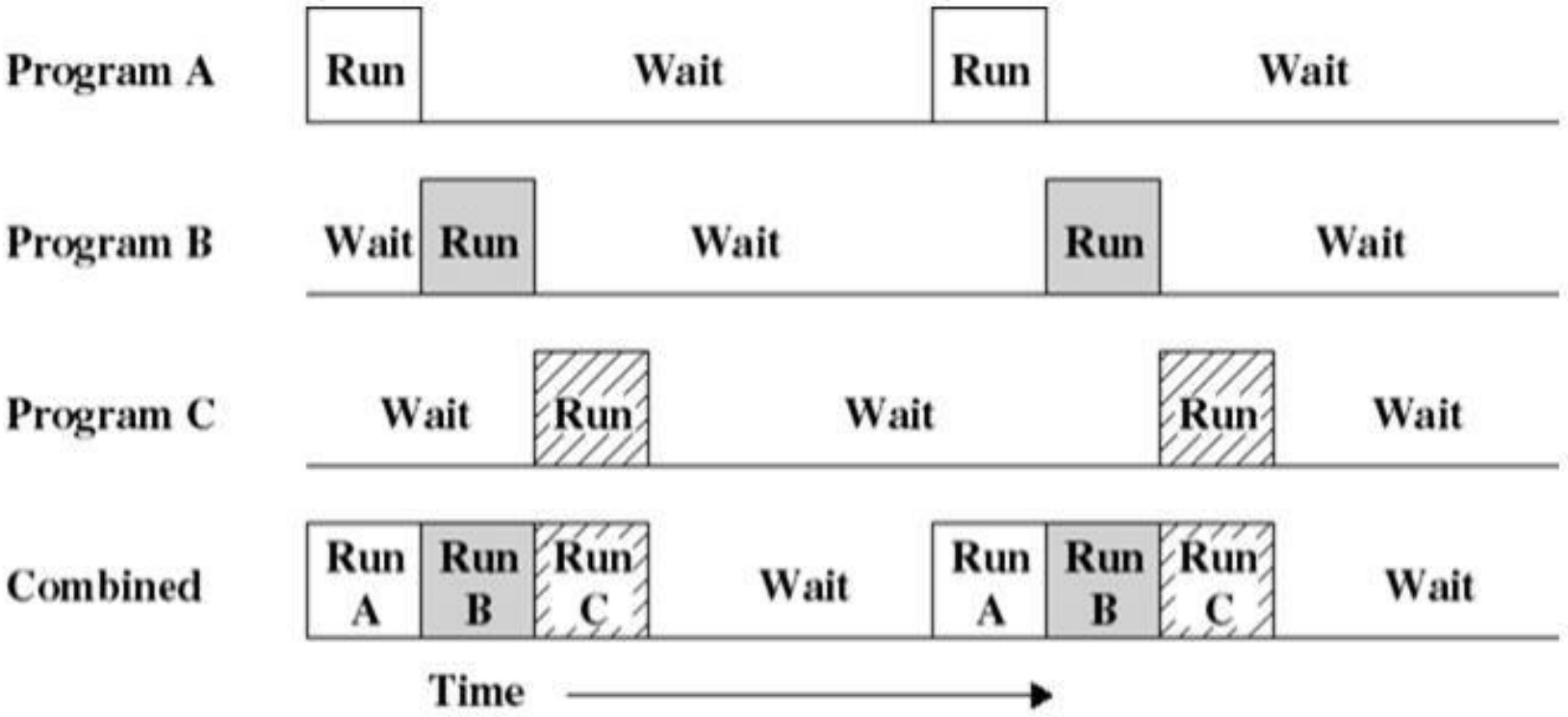
→ Tận dụng thời gian rảnh, tăng hiệu suất CPU

- Lập lịch CPU: chọn công việc để thực hiện, tránh 1 công việc chờ đợi quá lâu.
- Quản lý bộ nhớ: vùng nhớ đã cấp phát hay chưa, thu hồi bộ nhớ…
- Cấp phát tài nguyên: đĩa, máy in…
- Bảo vệ.

II. Phân loại hệ điều hành



Đơn chương



Đa chương

II. Phân loại hệ điều hành

2.2. Hệ thống chia sẻ thời gian – đa nhiệm

- Mở rộng của *hệ thống đa chương*.
- Mỗi công việc chạy trong khoảng t/g nhất định.
- Các yêu cầu tương tự *hệ thống đa chương*.
- Đồng bộ giữa các công việc.
- Giao tiếp giữa các công việc.
- Tránh deadlock.

II. Phân loại hệ điều hành

2.3. Hệ thống song song (đa xử lý)

- Các bộ xử lý chia sẻ bộ nhớ với nhau.
- Các công việc được các bộ xử lý thực hiện đồng thời.
- Gồm có đa xử lý đối xứng và bất đối xứng.

- Năng suất cao (do công việc được xử lý đồng thời).
- Sự hỏng hóc của một bộ xử lý không ảnh hưởng đến toàn bộ hệ thống.

II. Phân loại hệ điều hành

2.3. Hệ thống song song(tt)

- Đa xử lý đối xứng: các bộ xử lý ngang cấp, chạy với một bản sao của hệ điều hành.
- VD: **Windows NT, Solaris 5.0, Digital UNIX, OS/2, Linux**
- Đa xử lý bất đối xứng: có bộ xử lý chính (master processor) định thời và phân công việc cho các bộ xử lý khác (slave processors).
- VD: **SunOS 4.0**

II. Phân loại hệ điều hành

2.4. Hệ thống phân tán

- Tương tự hệ thống xử lý song song.
- Các bộ xử lý có bộ nhớ riêng.
- Người dùng chỉ thấy một hệ thống đơn nhất.
- Phân làm 2 loại:
  - Client – server
  - Peer to Peer

II. Phân loại hệ điều hành

2.4. Hệ thống phân tán (tt)

- 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, ...).

II. Phân loại hệ điều hành

2.5. 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 khoa, quân sự…
- Hard real – time:
  - Yêu cầu thời gian xử lý, đáp ứng nghiêm ngặt.
  - Giới hạn về độ nhớ.
- Soft real – time:
  - Các công việc thực hiện theo độ ưu tiên.
  - Dùng trong lĩnh vực multimedia, virtual reality, …

NỘI DUNG TRAINING

CHƯƠNG 2: CẤU TRÚC HỆ ĐIỀU HÀNH

I. Các thành phần của hệ điều hành:

1. Quản lý tiến trình
2. Quản lý bộ nhớ chính
3. Quản lý file
4. Quản lý hệ thống I/O
5. Quản lý hệ thống lưu trữ thứ cấp
6. Hệ thống bảo vệ
7. Hệ thống thông dịch lệnh

I. Các thành phần của HDH

1. Quản lý tiến trình

- Tiến trình là một chương trình đang thi hành.
- Tại một thời điểm có thể có nhiều tiến trình.
- Tiến trình có thể tạo ra các tiến trình con ➔ cây tiến trình.
- HĐH có cung cấp các cơ chế đồng bộ, giao tiếp giữa các tiến trình, khống chế tắc nghẽn.
- Trao đổi thông tin giữa các tiến trình qua hai cách:
  - Chia sẻ bộ nhớ.
  - Chuyển thông điệp.

I. Các thành phần của HĐH

2. Quản lý bộ nhớ chính

- HĐH 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 chọn tiến trình nào để nạp vào chỗ trống của bộ nhớ chính.
  - Cấp phát bộ nhớ và thu hồi bộ nhớ.

I. Các thành phần của HĐH

3. Quản lý file

- Các dịch vụ:
  - 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.

I. Các thành phần của HĐH

4. Quản lý thiết bị I/O

- Che dấu sự 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.

I. Các thành phần của HĐH

5. Quản lý hệ thống lưu trữ thứ cấp

- Phương tiện lưu trữ thường là đĩa từ đĩa quang (HDD, SSD, ...)
- Nhiệm vụ của HĐH:
  - Quản lý không gian trống.
  - Cấp phát không gian lưu trữ.
  - Định thời hoạt động.

I. Các thành phần của HĐH

6. 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 (enforcement of policies) (ví dụ: cần bảo vệ dữ liệu của ai đối với ai).

I. Các thành phần của HĐH

7. Hệ thống thông dịch lệnh

- Là giao diện cơ bản để người dùng giao tiếp với HĐH (shell, ...).
- 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. 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.
- Phát hiện lỗi.

II. Các dịch vụ HĐH cung cấp

- Cấp phát tài nguyên (resource allocation)
- Kế toán (accounting):
  - Nhằm lưu vết user để tính phí hoặc đơn giản để thống kê.
- Bảo vệ (protection):
  - Hai tiến trình khác nhau không được ảnh hưởng nhau.
  - Kiểm soát được các truy xuất tài nguyên của hệ thống.
- An ninh (security):
  - Chỉ các user được phép sử dụng hệ thống mới truy cập được tài nguyên của hệ thống (vd: thông qua username và password).

III. Lời gọi hệ thống:

- Dùng để giao tiếp giữa tiến trình và HĐH.
- Trong các ngôn ngữ lập trình cấp cao, một số thư viện lập trình được xây dựng dựa trên các thư viện hệ thống.

III. Lời gọi hệ thống

- Ba 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.

IV. 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).
- Không phải là application program.
- Gồm:
  - Soạn thảo file: file editor.
  - Thông tin trạng thái: như date, time, dung lượng bộ nhớ trống.
  - ...

V. Cấu trúc hệ thống:

- Hệ điều hành là một chương trình lớn
- Nó có nhiều dạng cấu trúc khác nhau:

1. Cấu trúc Monolithic - Original UNIX
2. Cấu trúc Layered Approach (hệ điều hanh THE)
3. Cấu trúc Microkernels
4. Cấu trúc Modules
5. Cấu trúc Hybrid Systems

1. Cấu trúc Monolithic – Original Unix

- Cấu trúc đơn giản (monolithic) MS-DOS: khi thiết kế, do giới hạn về dung lượng bộ nhớ nên không phân chia thành các module và chưa phân chia rõ chức năng giữa các phần của hệ thống.

- UNIX: gồm hai phần tách rời nhau
  - Nhân (cung cấp file system, CPU scheduling, memory management, và một số chức năng khác)
  - System program

2. Cấu trúc Layered Approach

+) HĐH được chia thành nhiều lớp (layer).

- Lớp dưới cùng: hardware
- Lớp trên cùng là giao tiếp với user
- Lớp trên chỉ phụ thuộc lớp dưới

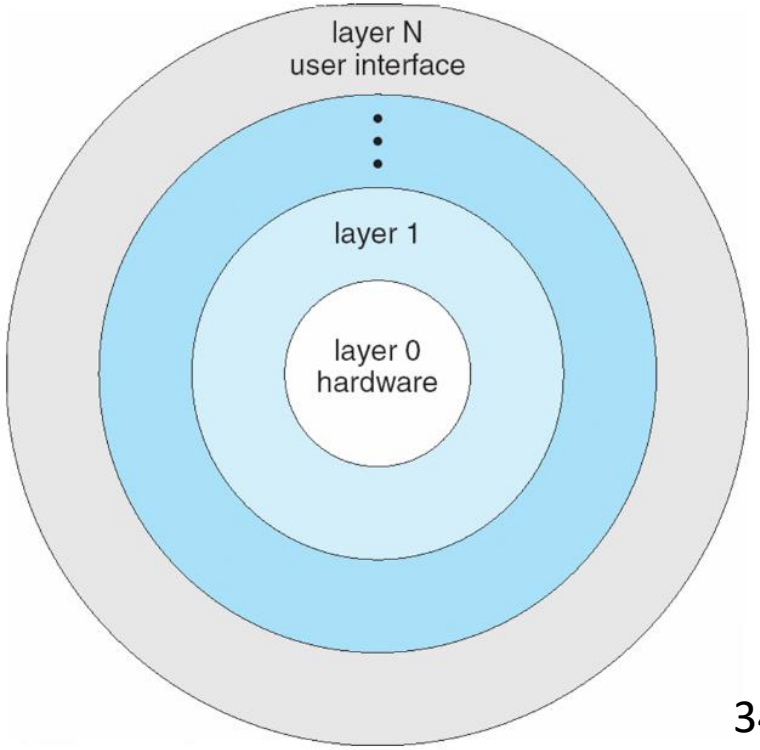
Ví dụ: Hệ điều hành THE (Technische Hogeschool Eindhoven)

+) Mỗi lớp tương đương một đối tượng trừu tượng: cấu trúc dữ liệu + thao tác

+) Phân lớp có lợi ích gì?

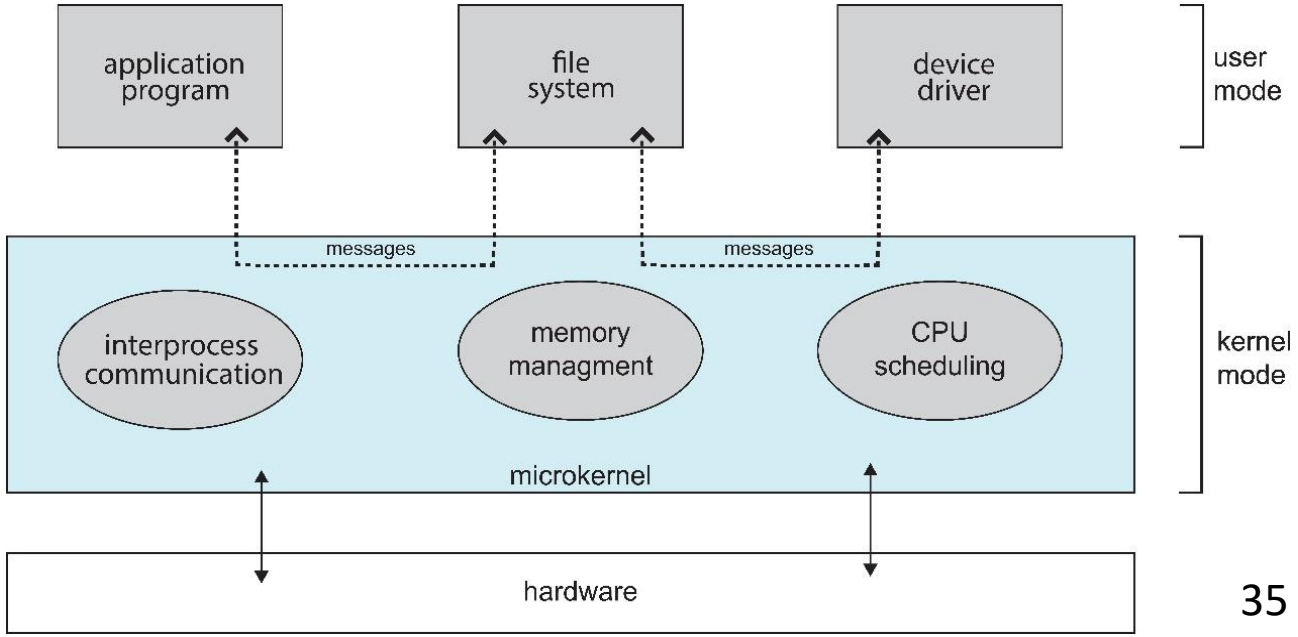
- Gỡ rối (debugger)
- Kiểm tra hệ thống
- Thay đổi chức năng

Lớp 5	user program
Lớp 4	Tạo buffer cho thiết bị I/O
Lớp 3	Device driver thao tác màn hình
Lớp 2	Quản lý bộ nhớ
Lớp 1	Lập lịch CPU
Lớp 0	Phần cứng

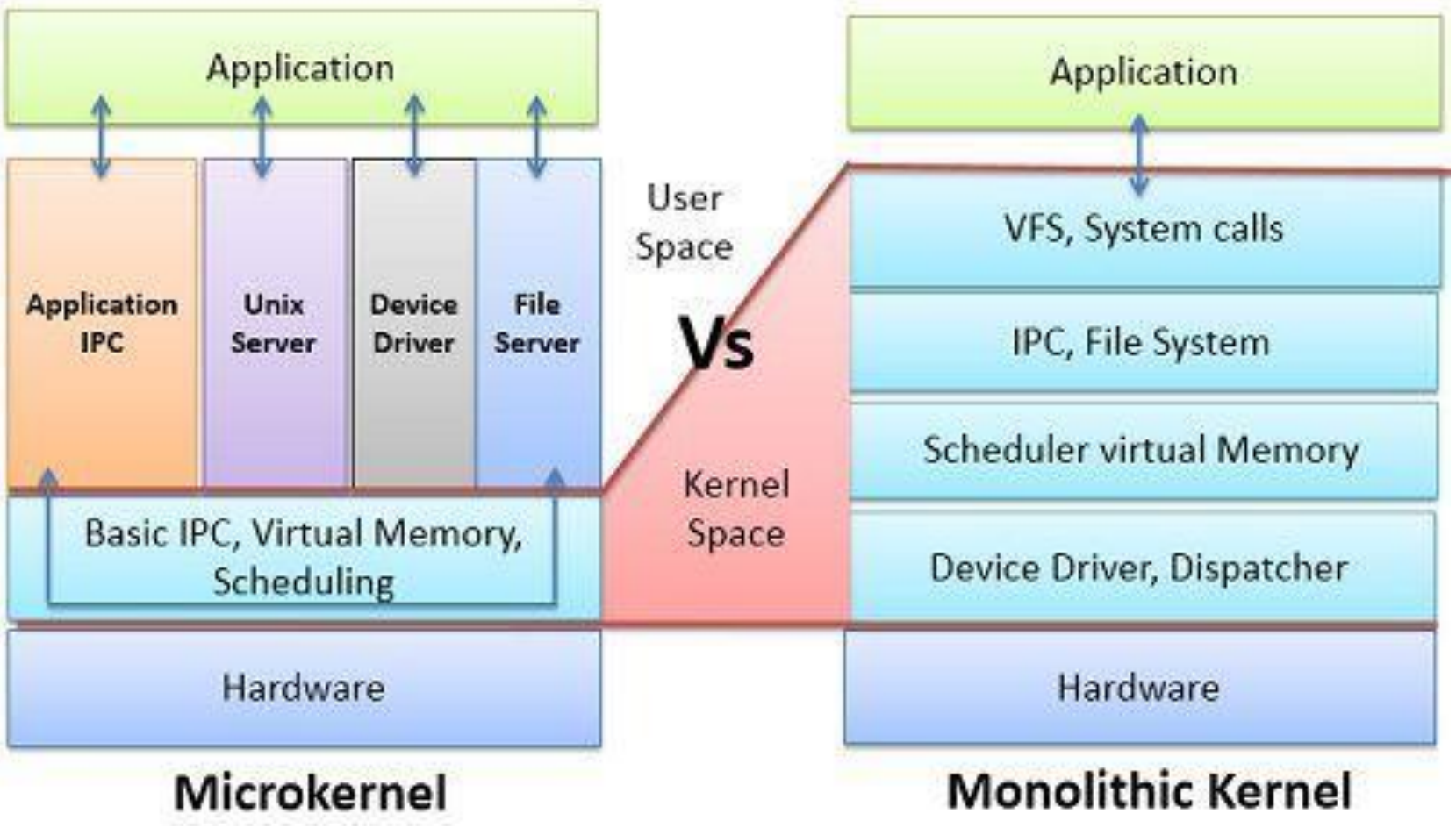


3. Cấu trúc Microkernel

- Phân chia module theo microkernel.
- Chuyển một số chức năng của OS từ kernel space sang user space
- 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, giao tiếp giữa các user module qua cơ chế truyền thông điệp.



3. Cấu trúc Microkernel



3. Cấu trúc Microkernel

Một số HĐH hiện đại sử dụng vi nhân:

- Mach
- Tru64 UNIX (Digital UNIX trước đây)
- Apple MacOS Server
- QNX
- Windows NT

4. 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

Nhìn chung, cấu trúc Modules giống với cấu trúc Layer nhưng phức tạp hơn

- Ví dụ: Linux, Solaris

5. Cấu trúc Hybrid Systems

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 3: TIẾN TRÌNH

I. Các khái niệm cơ bản
II. Trạng thái tiến trình
III. PCB
IV. Định thời tiến trinh
V. Các tác vụ đối với tiến trình

I. Các khái niệm cơ bản

- Tiến trình (process) là một chương trình đang thực thi

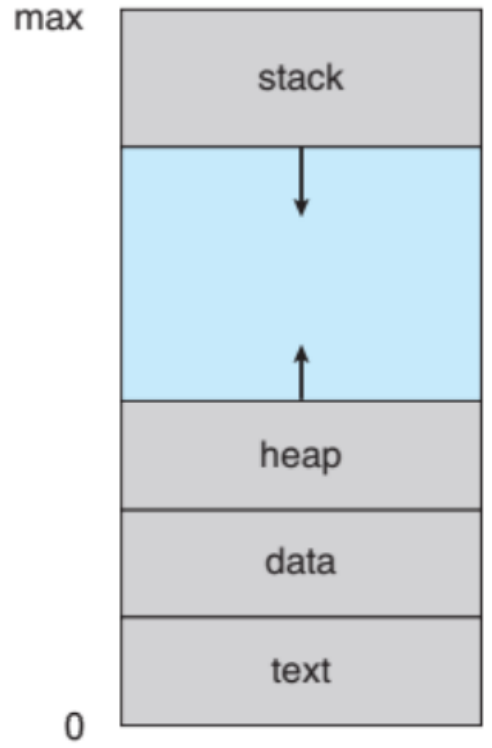
- Chương trình là thực thể bị động lưu trên đĩa (tập tin thực thi - executable file), tiến trình là thực thể chủ động.

-Chương trình trở thành tiến trình khi một tập tin thực thi được nạp vào bộ nhớ.

I. Các khái niệm cơ bản

Một tiến trình bao gồm:

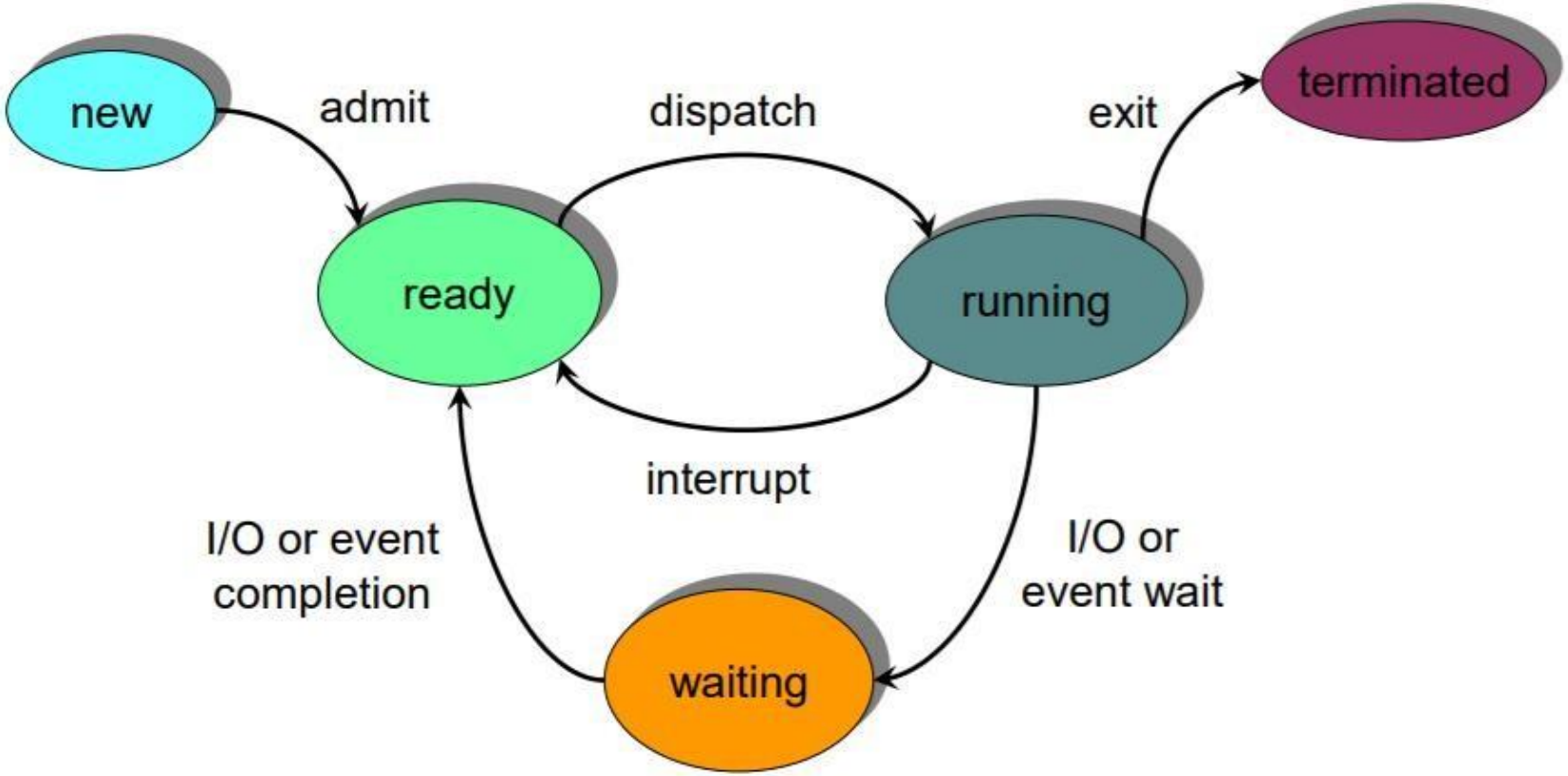
- Text section (program code)
- Data section (chứa global variables)
- Program counter, processor registers
- Heap section (chứa bộ nhớ cấp phát động)
- Stack section (chứa dữ liệu tạm thời)



II. Trạng thái tiến trình

- **new:** tiến trình vừa được tạo
- **ready:** đã có đủ tài nguyên, chỉ còn cần CPU
- **running:** các lệnh đang được thực thi
- **waiting (blocked):** đợi I/O hoàn tất, tín hiệu
- **terminated:** tiến trình đã kết thúc

II. Trạng thái tiến trình



II. Trạng thái tiến trình

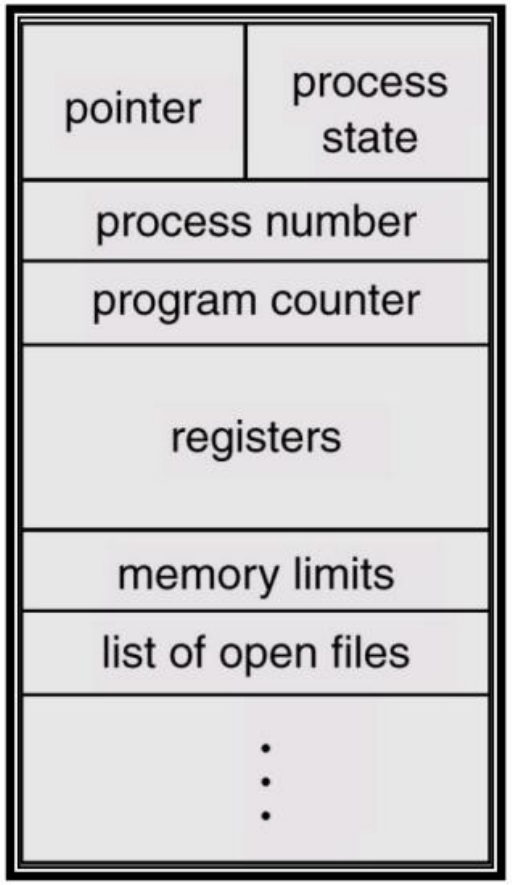
```
#include <stdio.h>
void main() {
    printf("Ban hoc tap Doan khoa Cong nghe Phan mem ");
    printf("chuc cac ban thi tot!");
    exit(0);
}
```

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

III. Process Control Block

- Mỗi tiến trình trong hệ thống đều được cấp phát một Process Control Block (PCB)
- PCB gồm:

- Trạng thái tiến trình: new, ready, running,...
- Bộ đếm chương trình
- Các thanh ghi
- Thông tin lập thời biểu CPU: độ ưu tiên,...
- Thông tin quản lý bộ nhớ
- Thông tin: lượng CPU, thời gian sử dụng,
- Thông tin trạng thái I/O



IV. Định thời tiến trình

Tại sao phải định thời?

- Đa chương
  - Có vài tiến trình chạy tại các 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

IV. Định thời tiến trình

- **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

V. Các tác vụ đối với tiến trình

Có 2 tác vụ chính: Tạo tiến trình mới và kết thúc tiến trình.

❖ Tạo tiến trình mới:

- Một tiến trình có thể tạo nhiều tiến trình mới thông qua một lời gọi hệ thống create-process (Vd: Hàm fork() trong Unix).
- Tiến trình được tạo là tiến trình con của tiến trình tạo (tiến trình cha).
  - Quan hệ cha-con định nghĩa một cây tiến trình.

V. Các tác vụ đối với tiến trình

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

V. Các tác vụ đối với tiến trình

❖ Kết thúc tiến trình:

- Tiến trình **tự kết thúc**: khi thực hiện lệnh cuối cùng của nó và yêu cầu hệ điều hành xoá bằng lệnh gọi hệ thống (system call) exit().

- Tiến trình kết thúc do **tiến trình khác** có quyền **kết thúc** nó:
  - Gọi system routine abort với tham số là pid của tiến trình cần được kết thúc.
  - Tiến trình cha kết thúc và kéo theo tất cả tiến trình con của nó đều kết thúc.

V. Các tác vụ đối với tiến trình

- fork() tạo ra tiến trình mới bằng cách nhân bản (duplicate) tiến trình gọi hàm này.
- Tiến trình ban đầu gọi là tiến trình cha (parent process)
- Tiến trình được nhân bản ra được gọi là tiến trình con (child process), là một bản sao giống với tiến trình cha tạo ra nó (**kể cả trạng thái thực thi**)

V. Các tác vụ đối với tiến trình

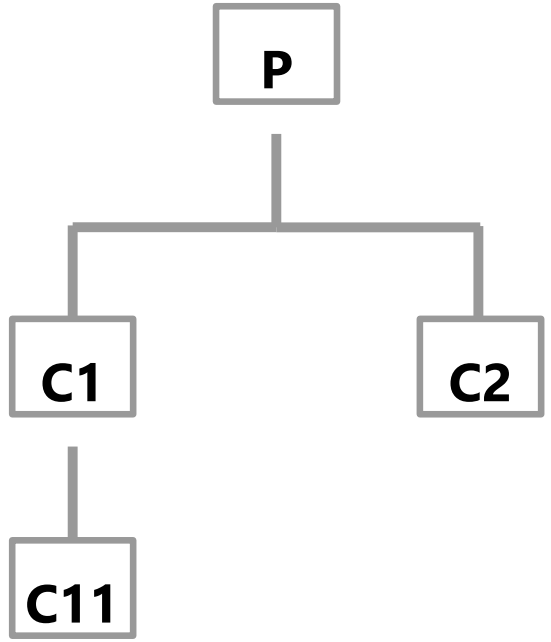
fork() sẽ trả về :

- Cho parent process:
  - PID của tiến trình con nếu tạo được tiến trình con
  - Trả về -1 nếu quá trình tạo tiến trình con bị lỗi
- Cho child process: 0 (Số 0)

TIẾN TRÌNH

Vẽ cây tiến trình và cho biết output

```
1   #include <stdio.h>
2   void main() {
3       printf("hi ");
4       int pid = fork();
5       if (pid > 0) {
6           fork();
7           printf("hello ");
8       } else {
9           fork();
10      }
11      printf("bye ");
    }
```




```
hi bye bye hello bye hello bye
```

TIẾN TRÌNH

Cho biết output và trạng thái tiến trình

```
void main() {
    int a, i = 2;
    while(i <= 5) {
        i++;
        if (i % 2 == 0) {
            a = i * 3;
            printf("Gia tri a = %d", &a);
        } else {
            a = i * 5;
            printf("Gia tri a = %d", &a);
        }
    }
    exit(0);
}
```

Output:

Gia tri a = 15

Gia tri a = 12

Gia tri a = 25

Gia tri a = 18

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

CHƯƠNG 4: ĐỊNH THỜI CPU

I. Các khái niệm và các loại bộ định thời
II. Các tiêu chuẩn định thời CPU
III. Các yếu tố của giải thuật định thời
IV. Các giải thuật định thời

ĐỊNH THỜI CPU

I. Các khái niệm và các loại bộ định thời

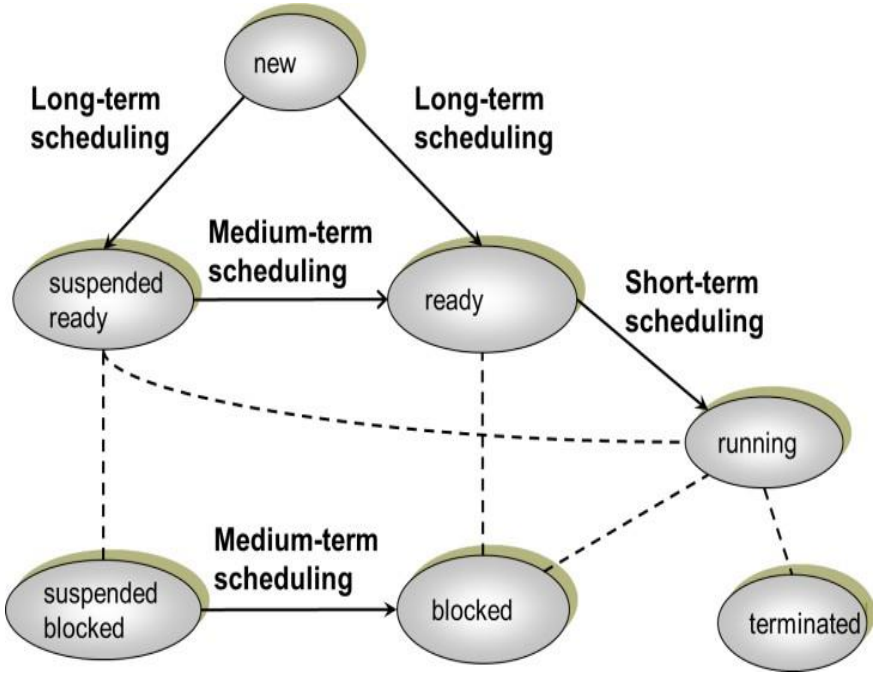
- Trong hệ thống multitasking tại mỗi thời điểm, một nhân chỉ thực thi được một process => Định thời CPU

- **Các 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



II. Các tiêu chuẩn định thời CPU

- **Hướng người dùng (User-oriented)**

+ Thời gian đáp ứng (Response time): Thời gian process nạp vào hệ thống -> yêu cầu đầu tiên được đáp ứng

+ Thời gian hoàn thành (Turnaround time): Thời gian process nạp vào hệ thống -> process kết thúc.

+ Thời gian chờ (Waiting time): Tổng thời gian một process đợi trong ready queue => Cực tiểu

- **Hướng hệ thống (System-oriented)**

+ Sử dụng CPU (process ultilization): CPU càng bận càng tốt

+ Công bằng (fairness): các process phải được đối xử như nhau

+ Thông lượng (throughput): số process được hoàn tất trong một đơn vị thời gian là cực đại

ĐỊNH THỜI CPU

ĐOÀN KHOA CÔNG NGHỆ PHẦN MỀM
BAN HỌC TẬP
Sharing is learning

III. Các yếu tố của giải thuật định thời

- **Hàm lựa chọn (selection function):** chọn process nào trong ready queue được thực thi

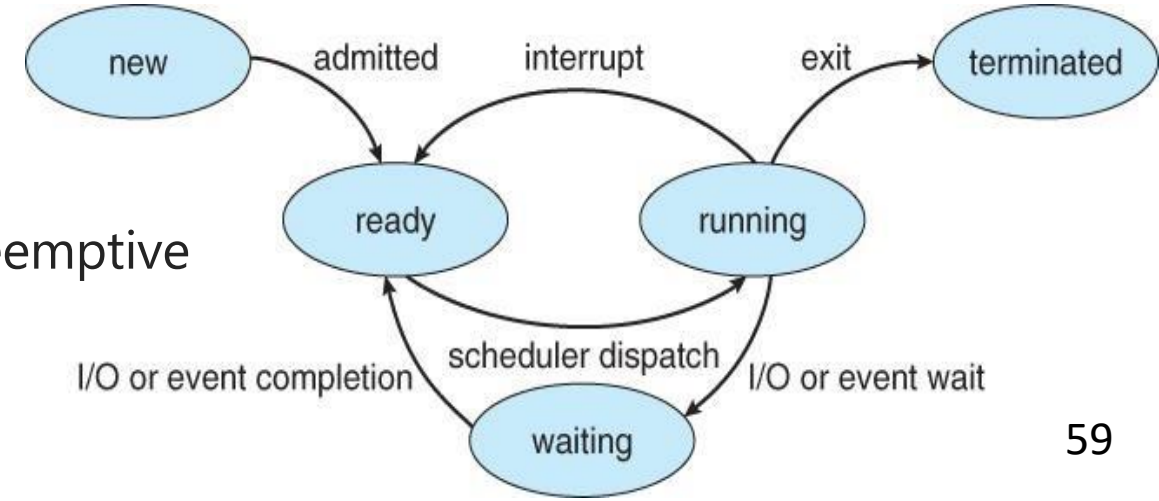
- **Chế độ quyết định (decision mode):** chọn thời điểm thực hiện hàm chọn lựa

+ Chế độ không trưng dụng (non-preemptive): Khi running, process sẽ thực thi cho đến khi kết thúc hoặc bị blocked do yêu cầu I/O

+ Chế độ trưng dụng (preemptive): Khi running, process có thể bị ngắt nửa chừng chuyển về trạng thái ready

- **Ví dụ:**

Running -> ready (do interrupted): định thời preemptive



ĐỊNH THỜI CPU

IV. Các giải thuật định thời

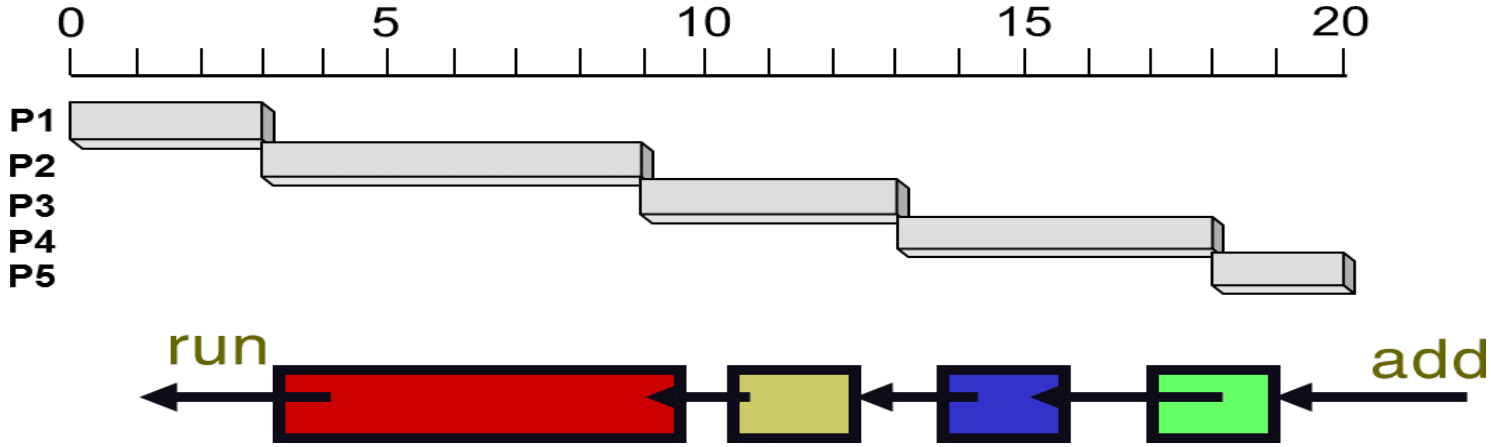
1. First-Come, First-Served (FCFS)
2. Shortest Job First (SJF)
3. Shortest Remaining Time First (SRTF)
4. Priority Scheduling
5. Round-Robin (RR)
6. Highest Response Ratio Next (HRRN)
7. Multilevel Queue
8. Multilevel Feedback Queue

ĐỊNH THỜI CPU

1. First-come, First-served (FCFS)

- **Hàm lựa chọn:** 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 (FIFO queues)
  - Tiến trình đi vào được thêm vào cuối hàng đợi
  - Tiến trình được lựa chọn để xử lý được lấy từ đầu queues



ĐỊNH THỜI CPU

1. First-come, First-served (FCFS)

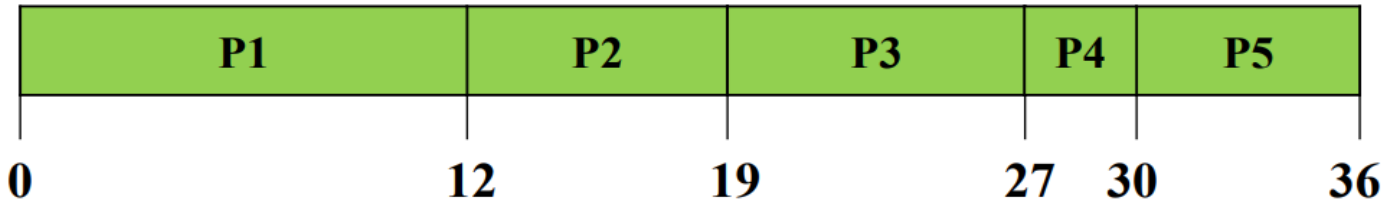
Ví dụ: Cho 5 tiến trình P1, P2, P3, P4, P5 với thời gian vào Ready Queues và thời gian cần CPU tương ứng như bảng sau:

Process	Arrival Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

ĐỊNH THỜI CPU

1. First-come, First-served (FCFS)

Giản đồ Gantt



Thời gian đáp ứng:

P1 = 0, P2 = 10, P3 = 14, P4 = 18, P5 = 18

Thời gian đáp ứng trung bình: (0 + 10 + 14 + 18 + 18)/5 = 12

Thời gian chờ:

P1 = 0, P2 = 10, P3 = 14, P4 = 18, P5 = 18

Thời gian chờ trung bình: (0 + 10 + 14 + 18 + 18)/5 = 12

Thời gian hoàn thành:

P1 = 12, P2 = 17, P3 = 22, P4 = 21, P5 = 24

Thời gian hoàn thành trung bình: (12 + 17 + 22 + 21 + 24)/5 = 19.2

Process	Arrival Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

ĐỊNH THỜI CPU

2. Shortest Job First (SJF)

- **Hàm lựa chọn:** 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 trình nó không nhường cho đến khi kết thúc chu kỳ xử lý của nó

  + Scheme 2: Preemptive **(Shortest-Remaining-Time-First)**
    Nếu một tiến trình mới được đưa vào 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.

3. Non-Preemptive SJF

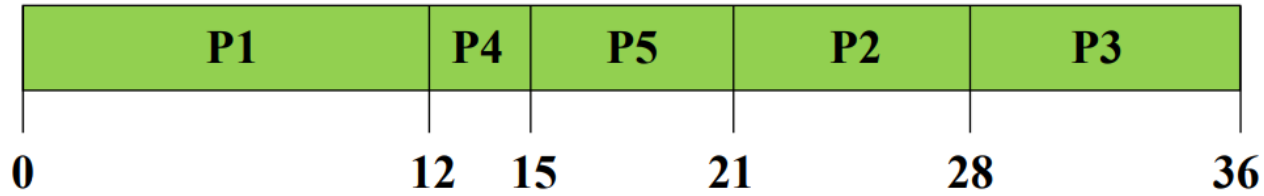
Ví dụ: Cho 5 tiến trình P1, P2, P3, P4, P5 với thời gian vào Ready Queues và thời gian cần CPU tương ứng như bảng sau:

Process	Arrival Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

ĐỊNH THỜI CPU

3. Non-Preemptive SJF

Giản đồ Gantt



Thời gian đáp ứng:

P1 = 0, P2 = 19, P3 = 23, P4 = 3, P5 = 3

Thời gian đáp ứng trung bình: (0 + 19 + 23 + 3 + 3)/5 = 9.6

Thời gian chờ:

P1 = 0, P2 = 19, P3 = 23, P4 = 3, P5 = 3

Thời gian chờ trung bình: (0 + 19 + 23 + 3 + 3)/5 = 9.6

Thời gian hoàn thành:

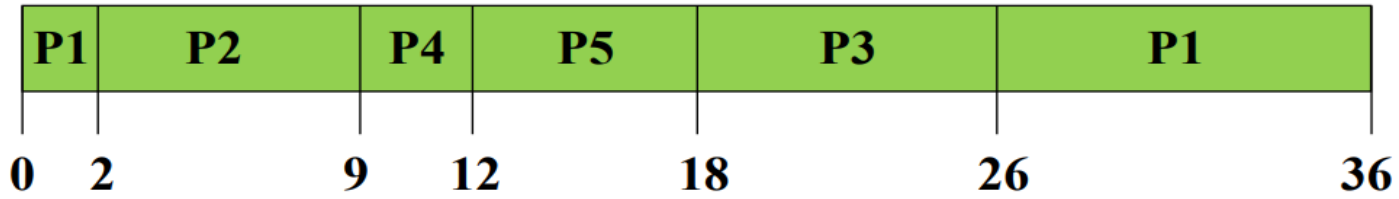
P1 = 12, P2 = 26, P3 = 31, P4 = 6, P5 = 9

Thời gian hoàn thành trung bình: (12 + 26 + 31 + 6 + 9)/5 = 16.8

Process	Arrival Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

3. Preemptive SJF (SRTF)

Giản đồ Gantt



Thời gian đáp ứng:

P1 = 0, P2 = 0, P3 = 13, P4 = 0, P5 = 0

Thời gian đáp ứng trung bình: (0 + 0 + 13 + 0 + 0)/5 = 2.6

Thời gian chờ:

P1 = 0 + 24, P2 = 0, P3 = 13, P4 = 0, P5 = 0

Thời gian chờ trung bình: (24 + 0 + 13 + 0 + 0)/5 = 7.4

Thời gian hoàn thành:

P1 = 36, P2 = 7, P3 = 21, P4 = 3, P5 = 6

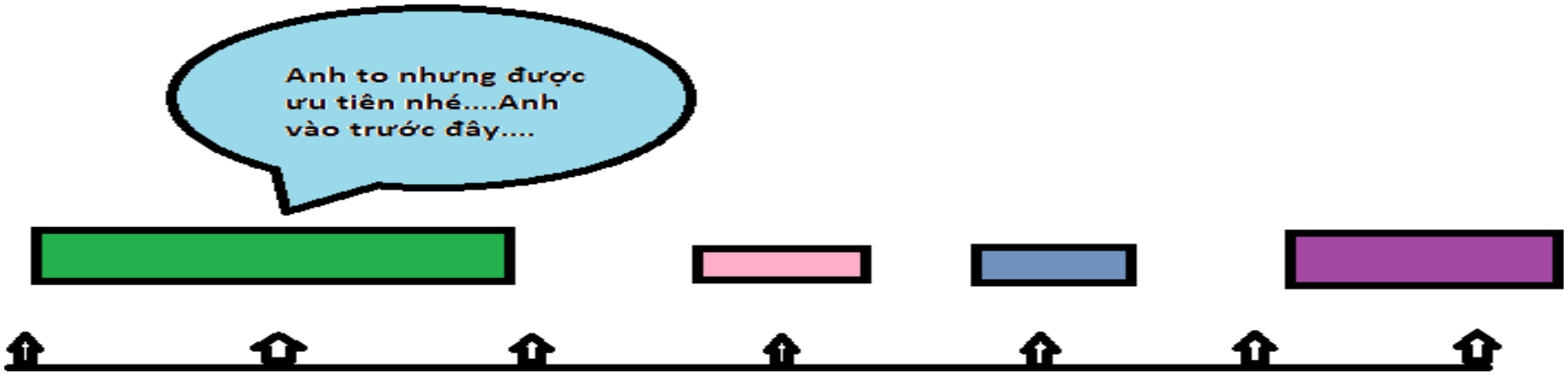
Thời gian hoàn thành trung bình: (36 + 7 + 21 + 3 + 6)/5 = 14.6

Process	Arrival Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

ĐỊNH THỜI CPU

4. Priority Scheduling

- **Hàm lựa chọn:** Mỗi process sẽ có một độ ưu tiên, CPU sẽ được cấp cho process có độ ưu tiên cao nhất.
- **Chế độ quyết định:** + Non-preemptive
  + 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 (aging)



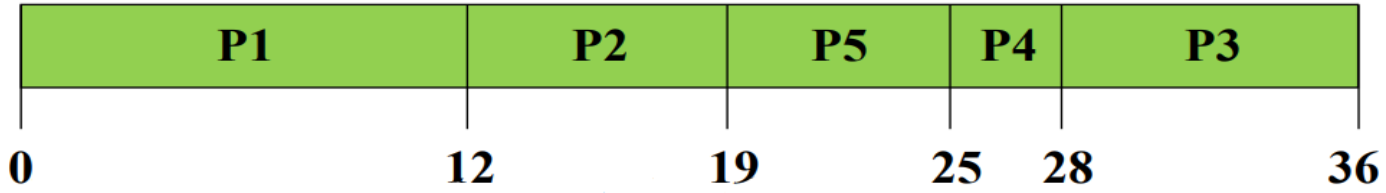
4. Priority Scheduling

Ví dụ: Cho 5 tiến trình P1, P2, P3, P4, P5 với thời gian vào Ready Queues và thời gian cần CPU tương ứng như bảng sau:

Process	Arrival Time	Burst Time	Priority
P1	0	12	2
P2	2	7	1
P3	5	8	5
P4	9	3	4
P5	12	6	3

4. Non-Preemptive Priority Scheduling

Giản đồ Gantt



Process	Arrival Time	Burst Time	Priority
P1	0	12	2
P2	2	7	1
P3	5	8	5
P4	9	3	4
P5	12	6	3

Thời gian đáp ứng:

P1 = 0, P2 = 10, P3 = 23, P4 = 16, P5 = 7

Thời gian đáp ứng trung bình: (0 + 10 + 23 + 16 + 7)/5 = 11.2

Thời gian chờ:

P1 = 0, P2 = 10, P3 = 23, P4 = 16, P5 = 7

Thời gian chờ trung bình: (0 + 10 + 23 + 16 + 7)/5 = 11.2

Thời gian hoàn thành:

P1 = 12, P2 = 17, P3 = 31, P4 = 19, P5 = 13

Thời gian hoàn thành trung bình: (12 + 17 + 31 + 19 + 13)/5 = 18.4

5. 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.
+ Khi q lớn RR -> FCFS; 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

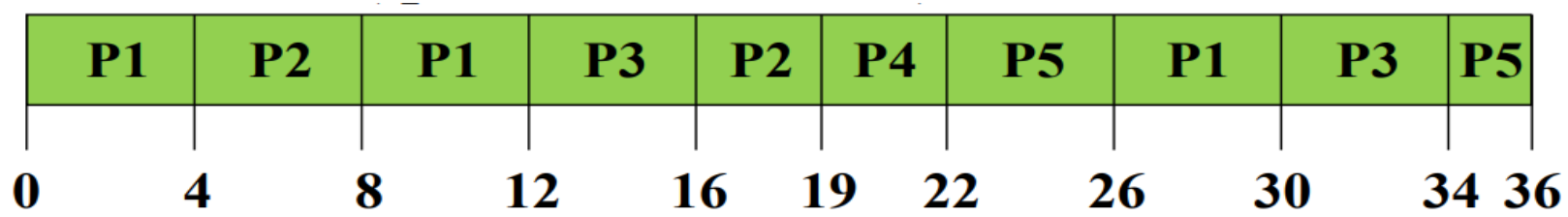
- Nếu có n process trong hàng đợi ready, quantum time là q → mỗi process lấy 1/n thời gian CPU theo từng khối có kích thước lớn nhất là q
  - Không có process nào chờ lâu hơn (n-1)*q đơn vị thời gian

- 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

ĐỊNH THỜI CPU

5. Round Robin (RR)

Giản đồ Gantt (quantum time = 4)



Thời gian đáp ứng:

P1 = 0, P2 = 2, P3 = 7, P4 = 10, P5 = 10

Thời gian đáp ứng trung bình: (0 + 2 + 7 + 10 + 10)/5 = 5.8

Thời gian chờ:

P1 = 0 + 4 + 14, P2 = 2 + 8, P3 = 7 + 14, P4 = 10, P5 = 10 + 8

Thời gian chờ trung bình: (18 + 10 + 21 + 10 + 18)/5 = 15.4

Thời gian hoàn thành:

P1 = 30, P2 = 17, P3 = 29, P4 = 13, P5 = 24

Thời gian hoàn thành trung bình: (30 + 17 + 29 + 13 + 24)/5 = 22.6

Process	Arrival Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

6. Highest Response Ratio Next

Hàm lựa chọn:

+ Chọn process có tỉ lệ phản hồi (response ratio - RR) cao nhất.
+ Các process ngắn được ưu tiên hơn (vì service time nhỏ)

Công thức:

$$RR = \frac{\text{time spent waiting} + \text{expected service time}}{\text{expected service time}}$$

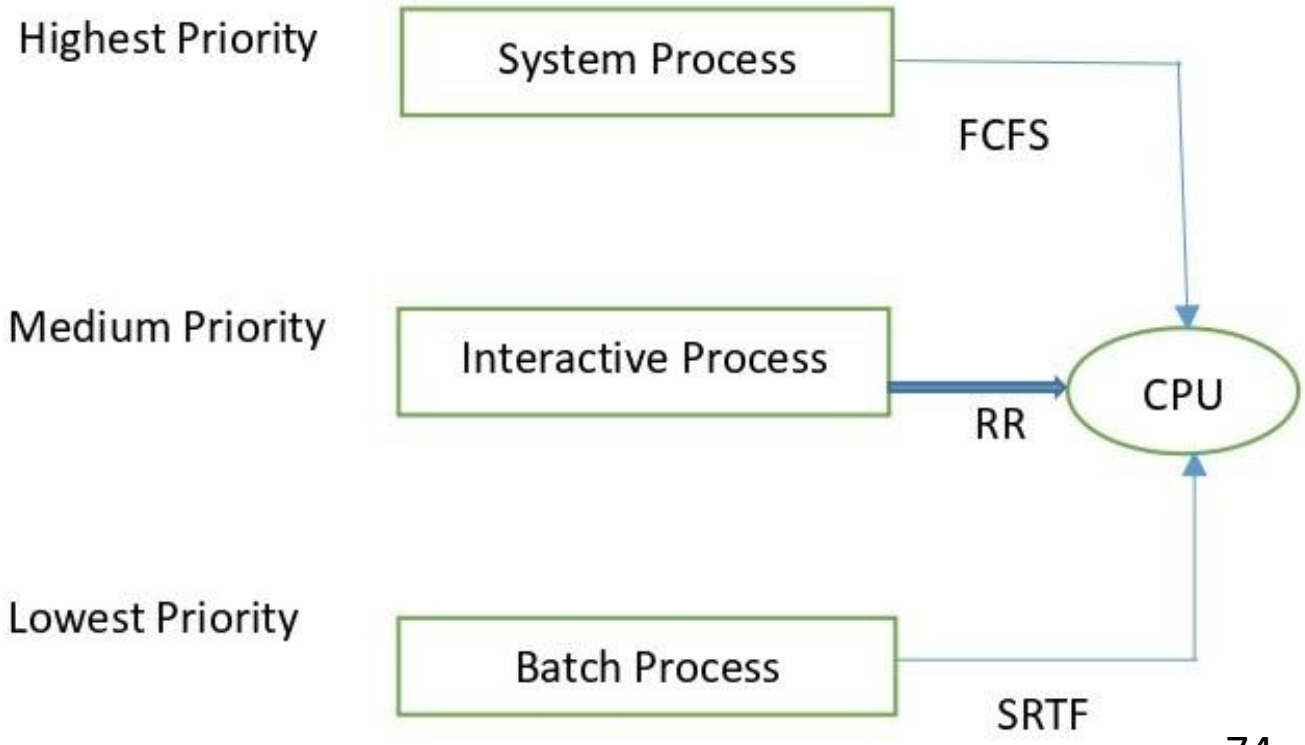
7. Multilevel Queue Scheduling

Hàng đợi ready được chia thành các hàng đợi riêng biệt dựa vào các tiêu chuẩn như:

- 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

+ Quá trình được chạy ở chế độ giao tiếp (foreground hay interactive process) được ưu tiên hơn so với quá trình chạy nền (background process)



7. Multilevel Queue Scheduling

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

  Ví dụ: 80% cho hàng đợi foreground định thời bằng Round Robin và 20% cho hàng đợi background định thời bằng giải thuật FCFS

8. Multilevel Feedback Queue

Vấn đề của multilevel queue:

Process không thể chuyển từ hàng đợi này sang hàng đợi khác

→ **Giải pháp:** Cơ chế feedback

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.

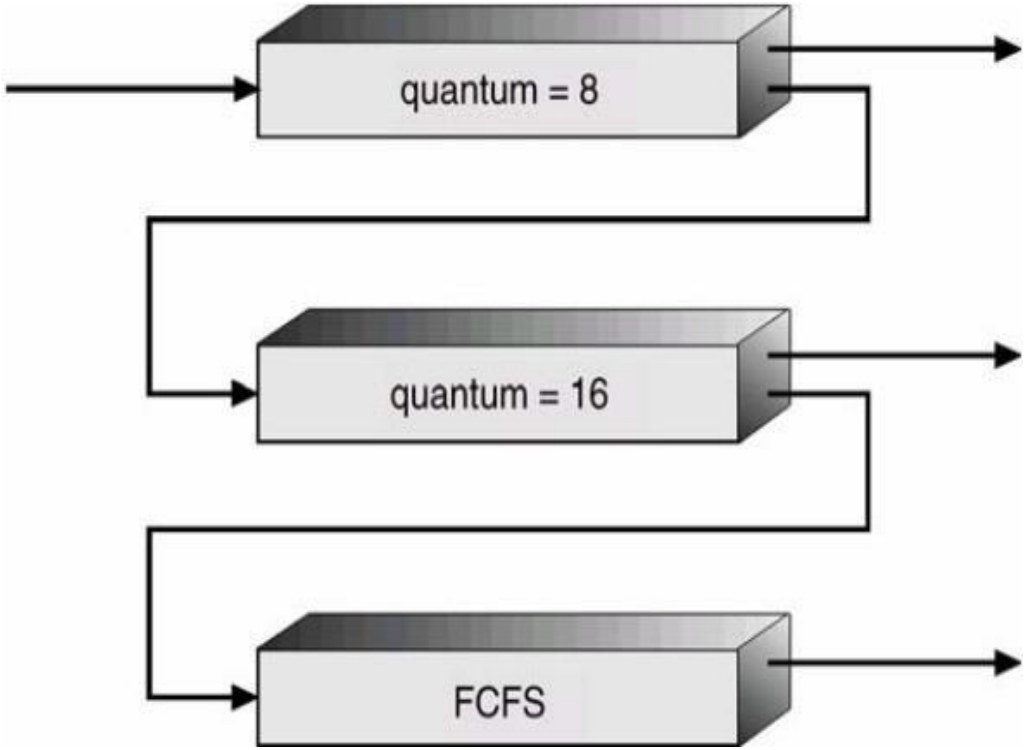
8. Multilevel Feedback Queue

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?



ĐỊNH THỜI CPU

Bài tập

Cho 5 tiến trình P1, P2, P3, P4, P5 với thời gian vào Ready Queue và thời gian cần CPU tương ứng Như bảng sau:

Process	Arrival Time	Burst - time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

Vẽ giản đồ Gantt. Tính thời gian đáp ứng, chờ đợi, hoàn thành trung bình cho từng giải thuật:

1. Round Robin, quantum time = 4

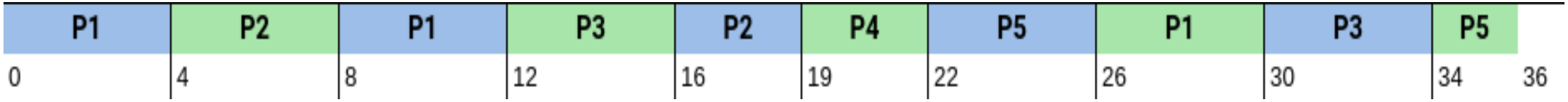
2. SRTF

ĐỊNH THỜI CPU

ĐOÀN KHOA CÔNG NGHỆ PHẦN MỀM
BAN HỌC TẬP
Sharing is learning

Bài tập

1. Round Robin với quantum time = 4



Process	Arrival Time	Burst - time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

Thời gian đáp ứng:

P1 = 0, P2 = 2, P3 = 7, P4 = 10, P5 = 10

=> Thời gian đáp ứng trung bình: 5.8

Thời gian chờ:

P1 = 4 + 14, P2 = 2 + 8, P3 = 7 + 14, P4 = 10, P5 = 10 + 8

=> Thời gian chờ trung bình: 15.4

Thời gian hoàn thành:

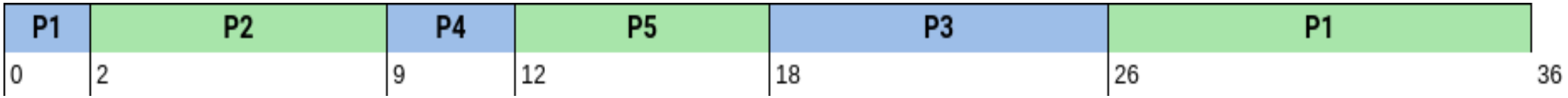
P1 = 30, P2 = 17, P3 = 29, P4 = 13, P5 = 24

=> Thời gian hoàn thành trung bình: 22.6

ĐỊNH THỜI CPU

Bài tập

2. Shortest Remaining Time First (SRTF)



Process	Arrival time	Burst - time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

Thời gian đáp ứng:

P1 = 0, P2 = 0, P3 = 13, P4 = 0, P5 = 0

=> Thời gian đáp ứng trung bình: 2.6

Thời gian chờ:

P1 = 24, P2 = 0, P3 = 13, P4 = 0, P5 = 0

=> Thời gian chờ trung bình: 7.4

Thời gian hoàn thành:

P1 = 36, P2 = 7, P3 = 21, P4 = 3, P5 = 6

=> Thời gian hoàn thành trung bình: 14.6

BAN HỌC TẬP KHOA CÔNG NGHỆ PHẦN MỀM
CHUỖI TRAINING GIỮA HỌC KÌ 1 NĂM HỌC 2020 - 2021

Sharing is learning

HẾT

CẢM ƠN CÁC BẠN ĐÃ THEO DÕI.
CHÚC CÁC BẠN CÓ KẾT QUẢ THI THẬT TỐT!

Ban học tập
Khoa Công Nghệ Phần Mềm
Trường ĐH Công Nghệ Thông Tin
ĐHQG Hồ Chí Minh

Email / Group
bht.cnpm.uit@gmail.com
fb.com/groups/bht.cnpm.uit