



HỆ ĐIỀU HÀNH

CHƯƠNG 3: TIẾN TRÌNH (PHẦN 1)

Trình bày các khái niệm cơ bản về tiến trình, các thông số của tiến trình, các khái niệm cơ bản về định thời tiến trình và giao tiếp giữa các tiến trình, và biết được các tác vụ cơ bản của một tiến trình



MỤC TIÊU

1. Hiểu được khái niệm và các trạng thái của tiến trình
2. Biết được các thông số của tiến trình
3. Biết được các khái niệm về định thời tiến trình



NỘI DUNG

1. Khái niệm cơ bản
2. Trạng thái tiến trình
3. Khối điều khiển tiến trình
4. Định thời tiến trình



KHÁI NIỆM CƠ BẢN

1



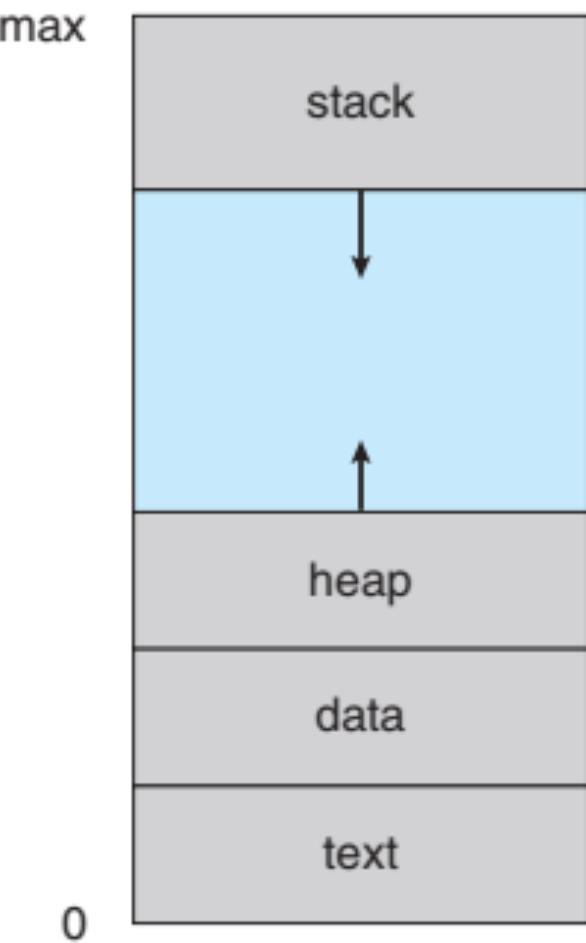
1. Khái niệm cơ bản

- Một hệ điều hành thực thi *chương trình* như là một *tiến trình*.
- Tiến trình (process) là gì?
 - **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.



1. 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)
 - Function parameters
 - Return address
 - Local variables

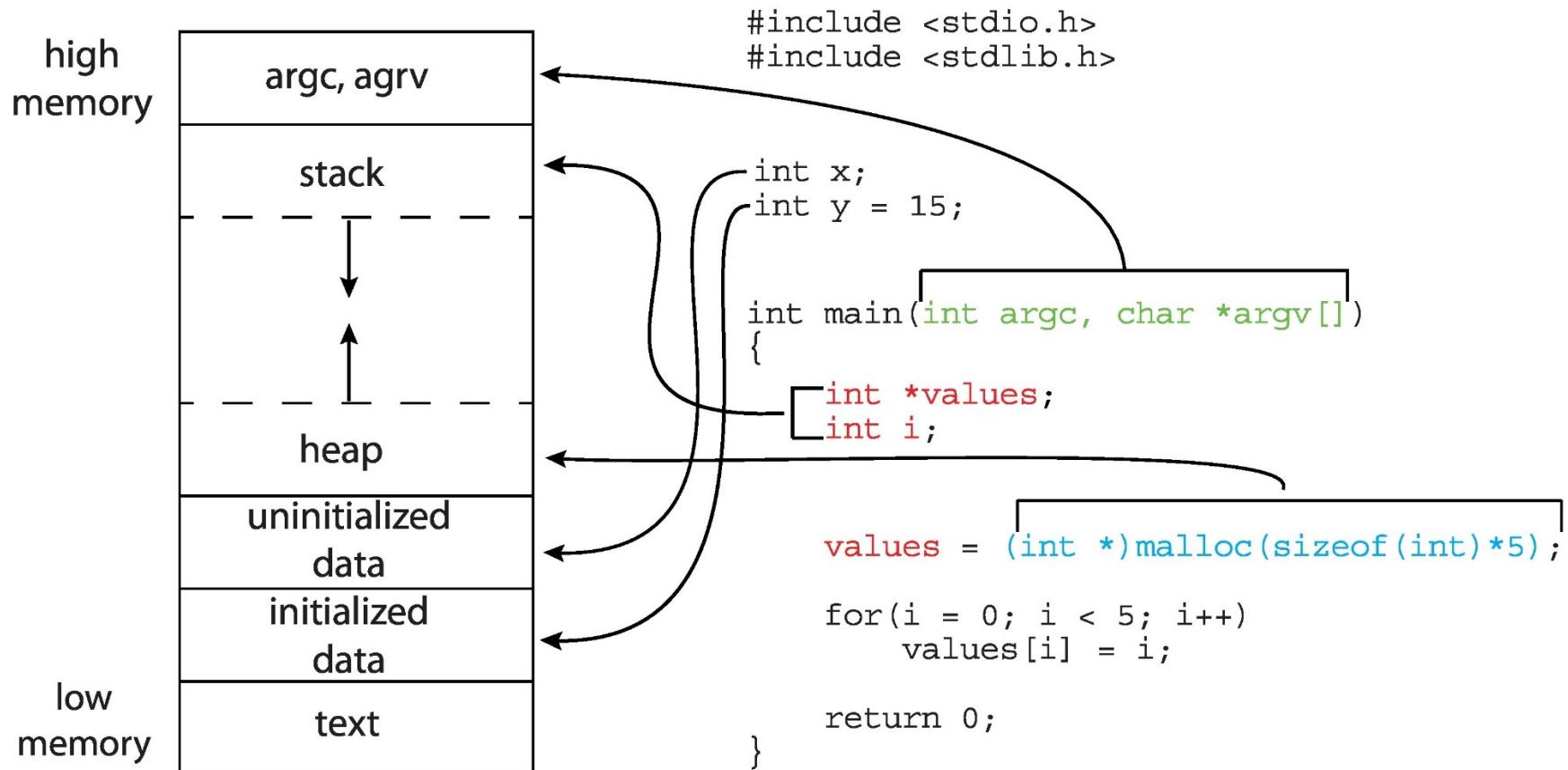


Layout của tiến trình trong bộ nhớ



1. Khái niệm cơ bản

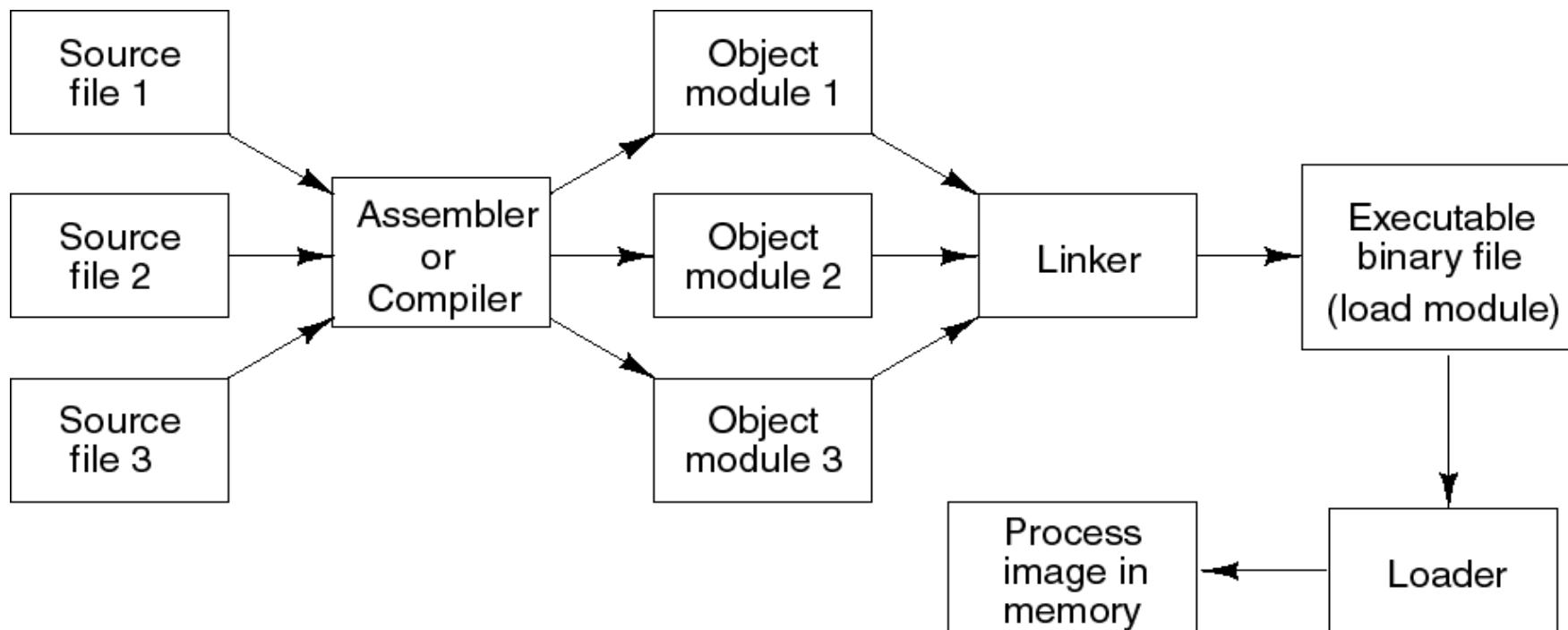
Layout bộ nhớ của một chương trình C





1. Khái niệm cơ bản

Các bước nạp chương trình vào bộ nhớ:





1. Khái niệm cơ bản

- 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 Process Control Block (PCB) cho tiến trình.
 - Thiết lập các mối liên hệ cần thiết (ví dụ: sắp PCB vào hàng đợi định thời, ...).



TRẠNG THÁI TIẾN TRÌNH

2

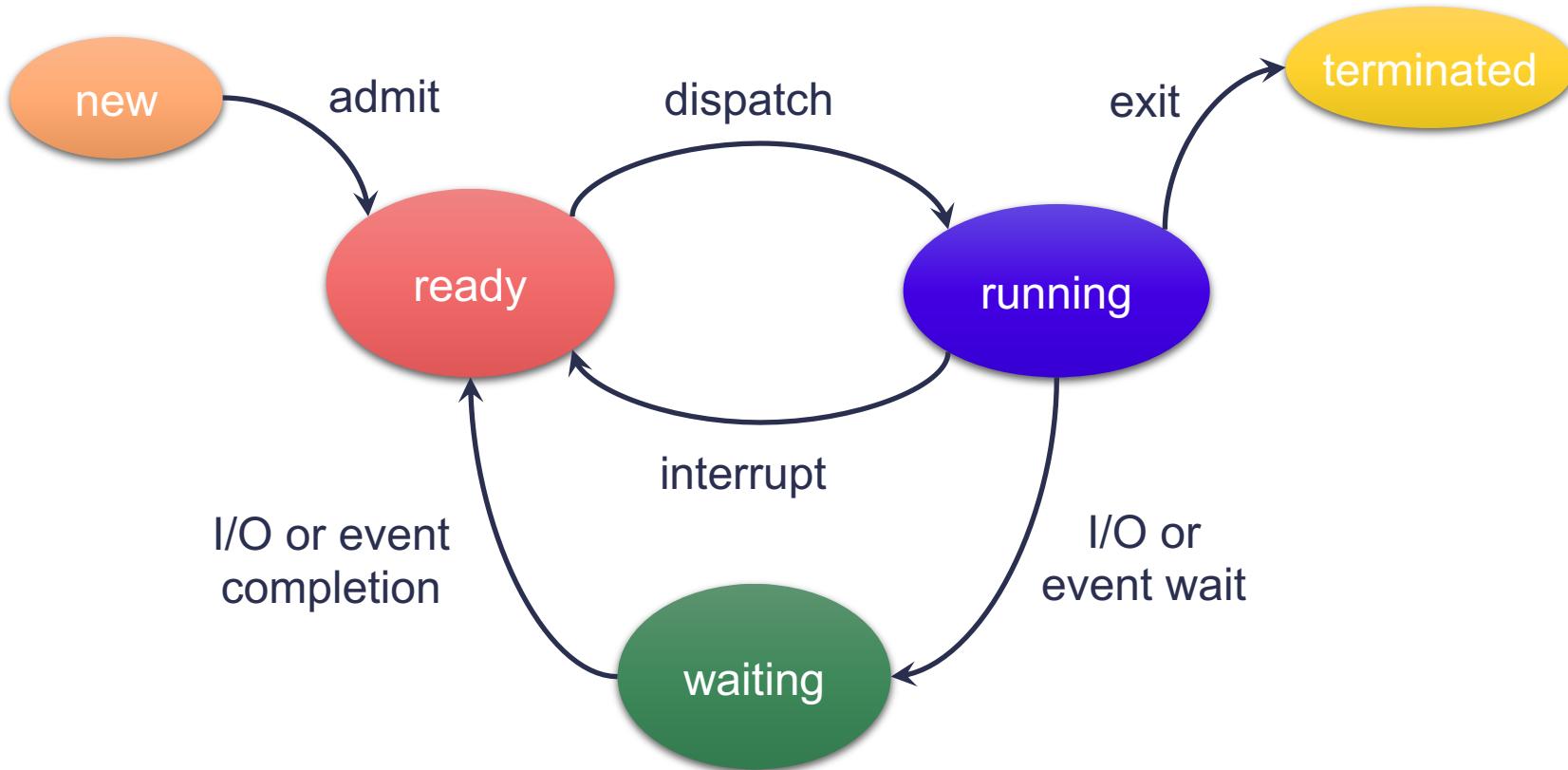


2. Trạng thái tiến trình

- **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 **blocked**): tiến trình đợi I/O hoàn tất, hoặc đợi tín hiệu
- **terminated**: tiến trình đã kết thúc



2. Trạng thái tiến trình



Chuyển đổi giữa các trạng thái của tiến trình



2. Trạng thái tiến trình

```
/* test.c */  
int main(int argc, char** argv)  
{  
    printf("Hello world\n");  
    exit(0);  
}
```

- Biên dịch chương trình trong Linux: **gcc test.c -o test**
- Thực thi chương trình test:
./test
- Trong hệ thống sẽ có một tiến trình test được tạo ra, thực thi và kết thúc.

- Chuỗi trạng thái của tiến trình test như sau (trường hợp tốt nhất):
 - new
 - ready
 - running
 - waiting (do chờ I/O khi gọi printf)
 - ready
 - running
 - terminated



2. Trạng thái tiến trình

```
int main (int argc, char** argv)
{
    int i = 2;
    while (i < 5)
    {
        i++;
        if (i % 2 == 0)
        {
            printf ("Hello");
            printf ("Hi");
        }
        else
        {
            printf ("Bye");
        }
    }
    exit (0);
}
```

- Hỏi sau khi kết thúc thì tiến trình khi chạy từ chương trình trên đã nằm trong hàng đợi waiting bao nhiêu lần?

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



PROCESS CONTROL BLOCK

3



3. Process Control Block

- Mỗi tiến trình trong hệ thống đều được cấp phát một **Process Control Block** (PCB).
 - PCB là một trong các cấu trúc dữ liệu quan trọng nhất của hệ điều hành.
- 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





4. ĐỊNH THỜI TIỀN TRÌNH

4



4. Định thời tiến trình

Yêu cầu đối với hệ điều hành về quản lý tiến trình

- Hỗ trợ sự thực thi luân phiên giữa nhiều tiến trình:
 - Hiệu suất sử dụng CPU
 - Thời gian đáp ứng
- Phân phối tài nguyên hệ thống hợp lý.
- Tránh deadlock, trì hoãn vô hạn định.
- Cung cấp cơ chế giao tiếp và đồng bộ hoạt động các tiến trình.
- Cung cấp cơ chế hỗ trợ user tạo/kết thúc tiến trình.



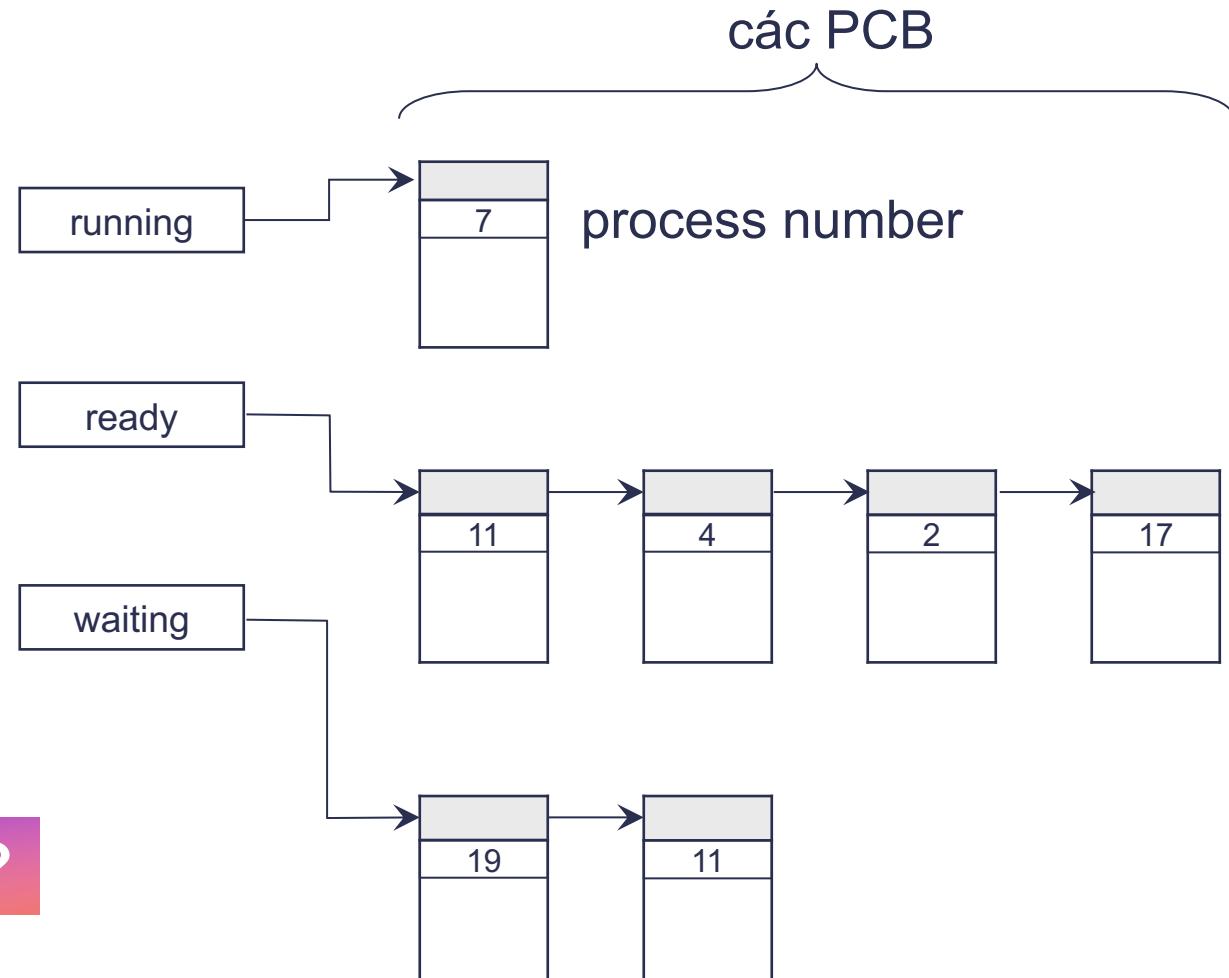
4. Định thời tiến trình

- 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.



4. Định thời tiến trình

Quản lý các tiến trình: các hàng đợi



Có trường hợp sai không?



4. ĐỊNH THỜI TIỀN TRÌNH

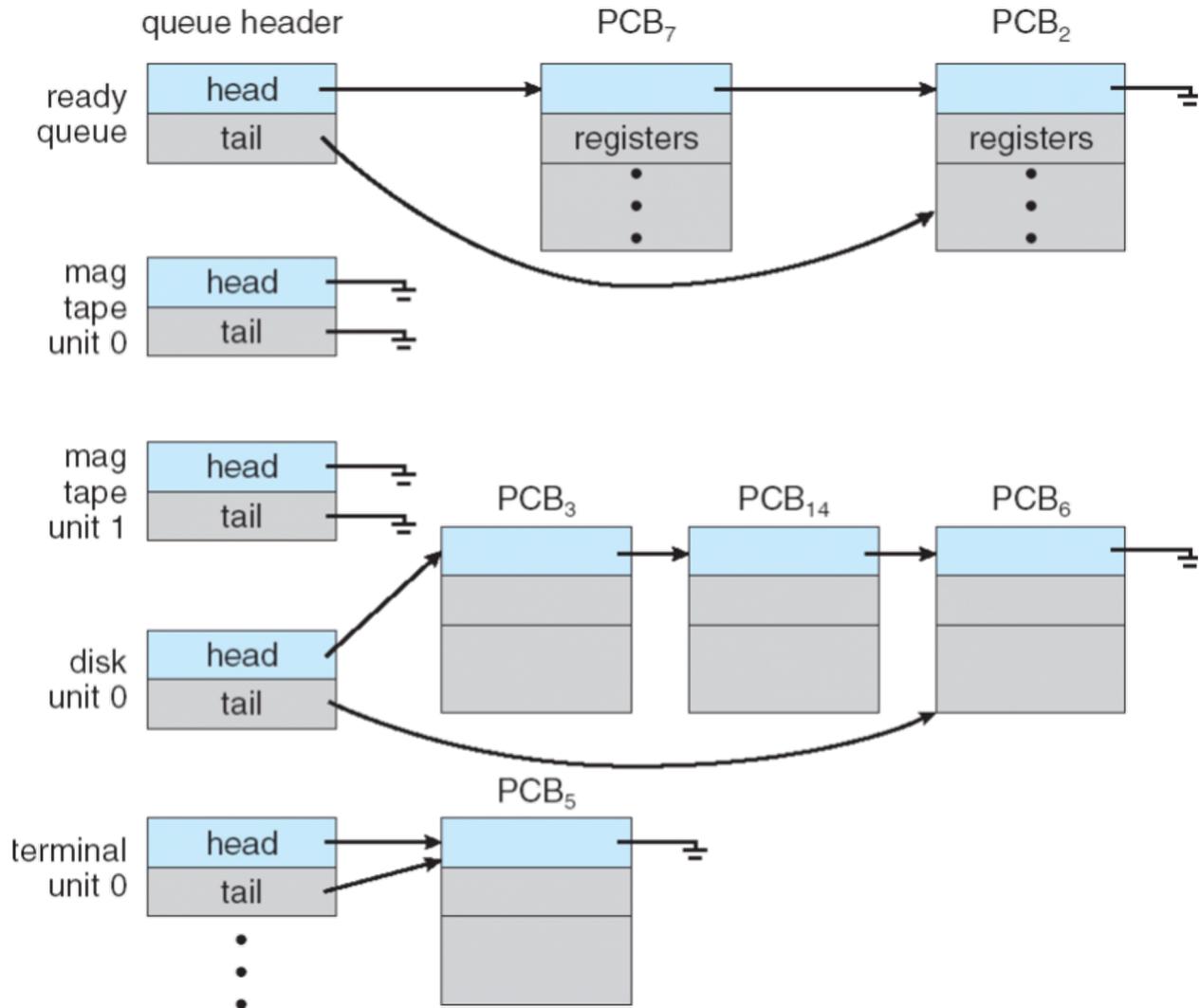
4.1. Các hàng đợi định thời

4



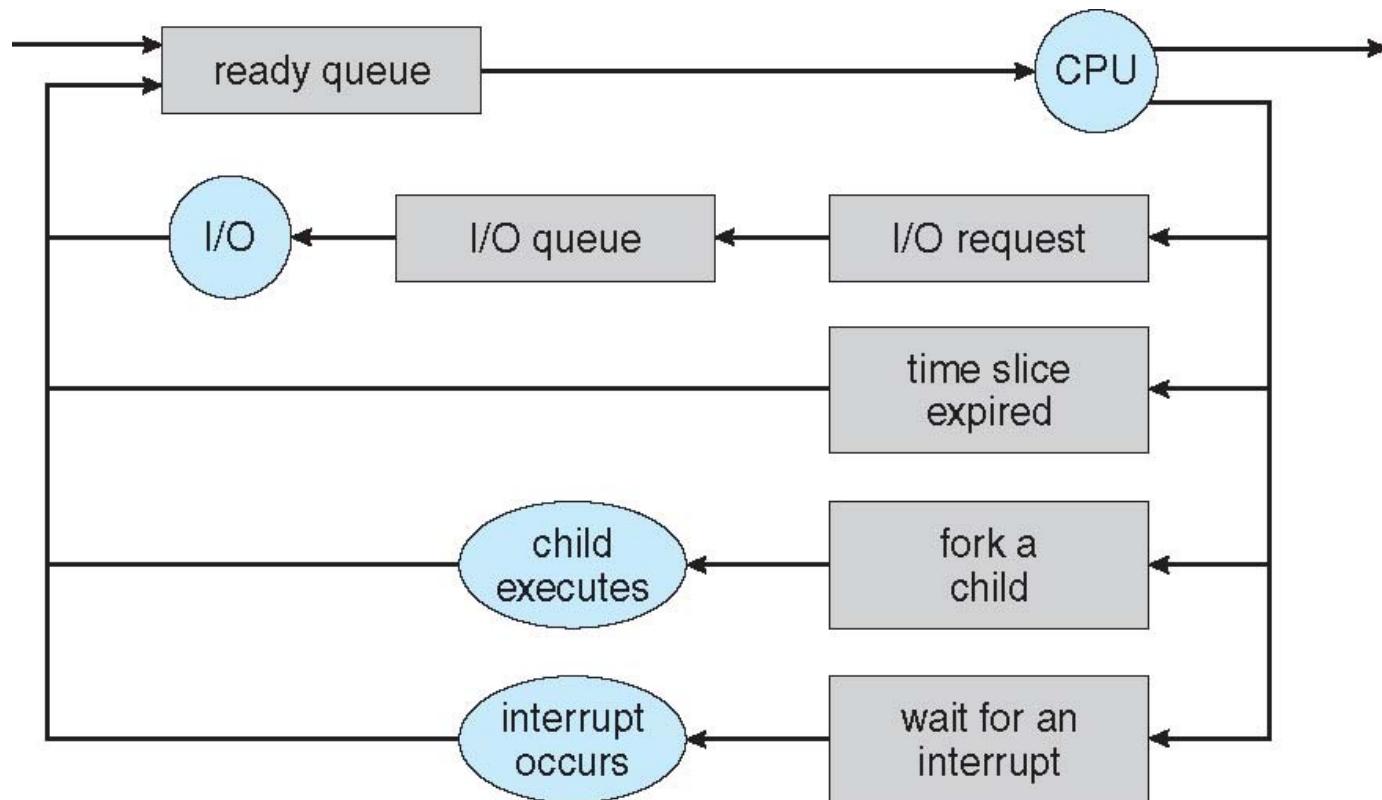
4.1. Các hàng đợi định thời

- Hàng đợi công việc - Job queue
- Hàng đợi sẵn sàng - Ready queue
- Hàng đợi thiết bị - Device queues
- ...





4.1. Các hàng đợi định thời



Lưu đồ hàng đợi của định thời tiến trình



4. ĐỊNH THỜI TIỀN TRÌNH

4.2. Bộ định thời

4



4.2. Bộ định thời

Phân loại bộ định thời

- **Bộ định thời công việc** (Job scheduler) hay **bộ định thời dài** (long-term scheduler).
- **Bộ định thời CPU** hay **bộ định thời ngắn**.

Phân loại tiến trình

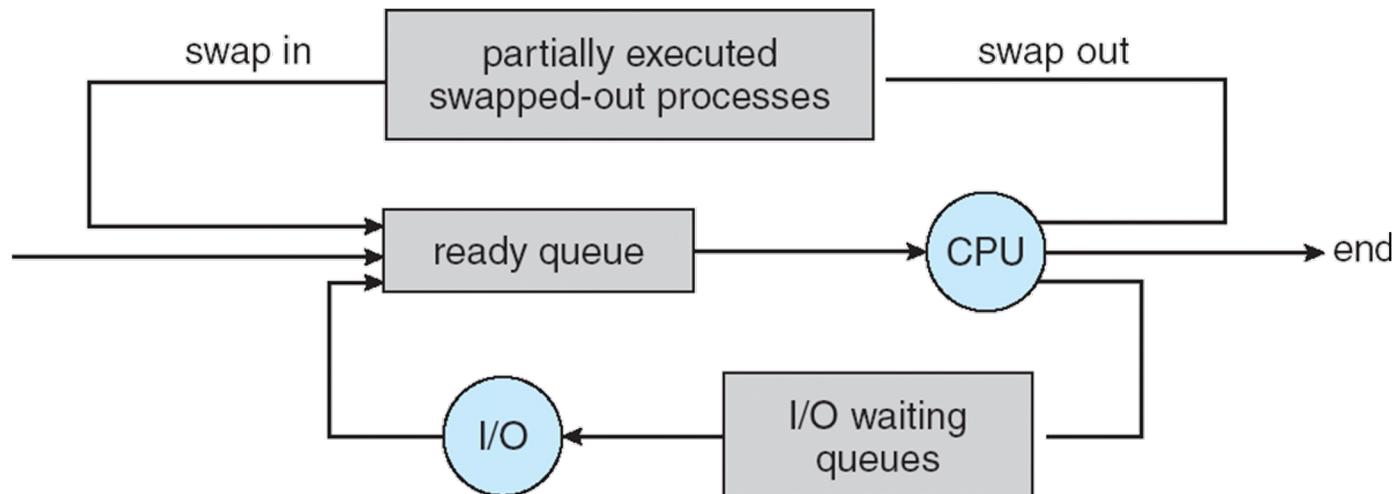
- Các tiến trình có thể mô tả như:
 - **tiến trình hướng I/O**
 - **tiến trình hướng CPU**
- Thời gian thực hiện khác nhau -> kết hợp hài hòa giữa chúng.



4.2. Bộ định thời

Bộ định thời trung gian/vừa

- Đôi khi hệ điều hành (như time-sharing system) có thêm **medium-term scheduling** để điều chỉnh mức độ đa chương của hệ thống.
- Medium-term scheduler:**
 - chuyển tiến trình từ bộ nhớ sang đĩa (swap out).
 - chuyển tiến trình từ đĩa vào bộ nhớ (swap in).



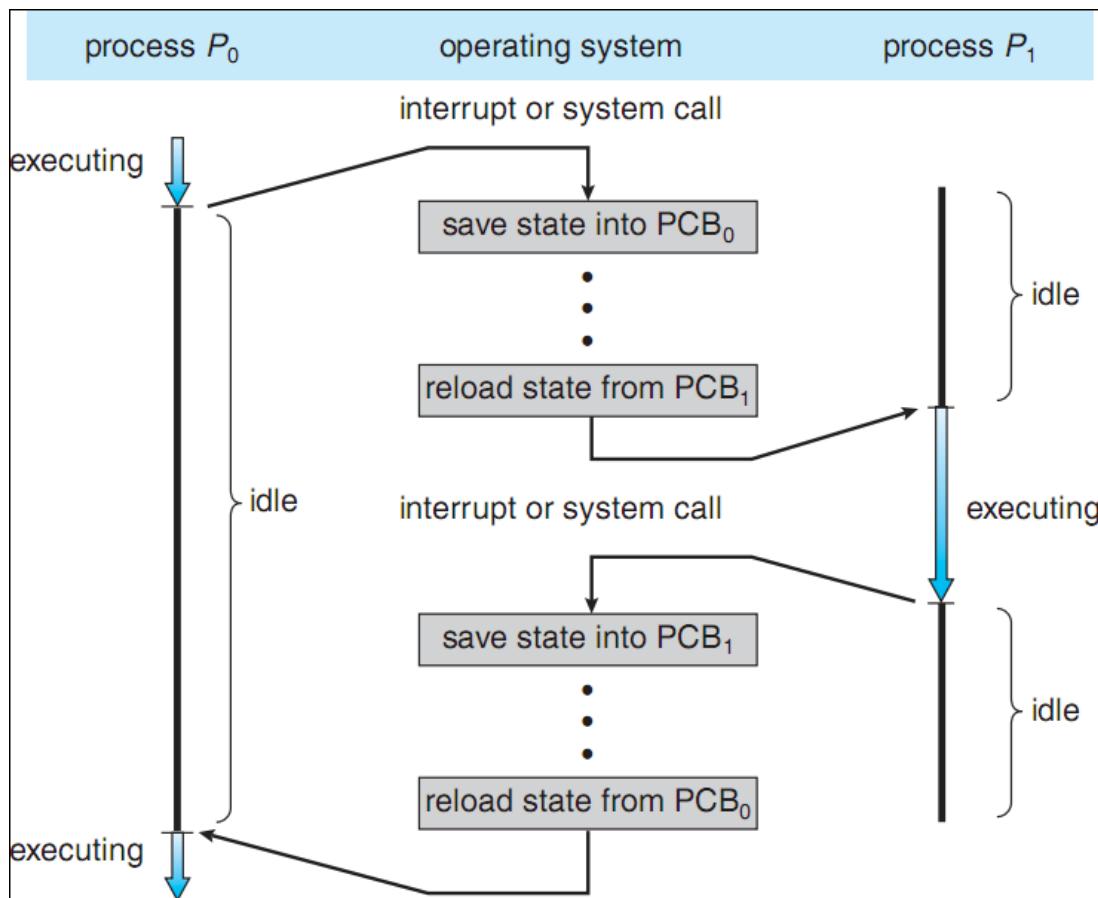


4.2. Bộ định thời

Chuyển ngữ cảnh (context switch)

Chuyển ngữ cảnh:

Quá trình CPU chuyển từ tiến trình này đến tiến trình khác.





Tóm tắt lại nội dung bài học

- Khái niệm cơ bản
- Trạng thái tiến trình
- Khối điều khiển tiến trình
- Định thời tiến trình



THẢO LUẬN



Thực hiện bởi Trường Đại học Công nghệ Thông tin, ĐHQG-HCM