



# HỆ ĐIỀU HÀNH

## CHƯƠNG 4: ĐỊNH THỜI CPU (PHẦN 1)

Định thời CPU là hoạt động quan trọng của thành phần quản lý tiến trình và có ảnh hưởng rất lớn đến hiệu suất máy tính cũng như trải nghiệm của người dùng. Trong chương này, người học được trình bày về mục đích và các tiêu chuẩn định thời, cũng như các chiến lược định thời CPU cơ bản.



## MỤC TIÊU

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



# NỘI DUNG

1. Các khái niệm cơ bản về định thời
2. Các loại định thời
3. Các tiêu chuẩn định thời CPU
4. Các giải thuật định thời
  - First-Come, First-Served (FCFS)
  - Shortest Job First (SJF)
  - Shortest Remaining Time First (SRTF)
  - Priority Scheduling



# CÁC KHÁI NIỆM CƠ BẢN VỀ ĐỊNH THỜI

01.



# 1. Các khái niệm cơ bản về định thời

- Trong các hệ thống đa nhiệm (multitasking), đơn vị xử lý
  - Cho phép thực thi đồng thời nhiều chương trình để làm tăng hiệu suất hệ thống (*Cho phép nhiều chương trình được nạp vào bộ nhớ*).
  - Tại mỗi thời điểm, chỉ có một tiến trình được thực thi.
- Cần phải giải quyết vấn đề phân chia, lựa chọn tiến trình thực thi để đạt được hiệu quả cao nhất.
- Cần có những phương pháp chọn lựa phù hợp.

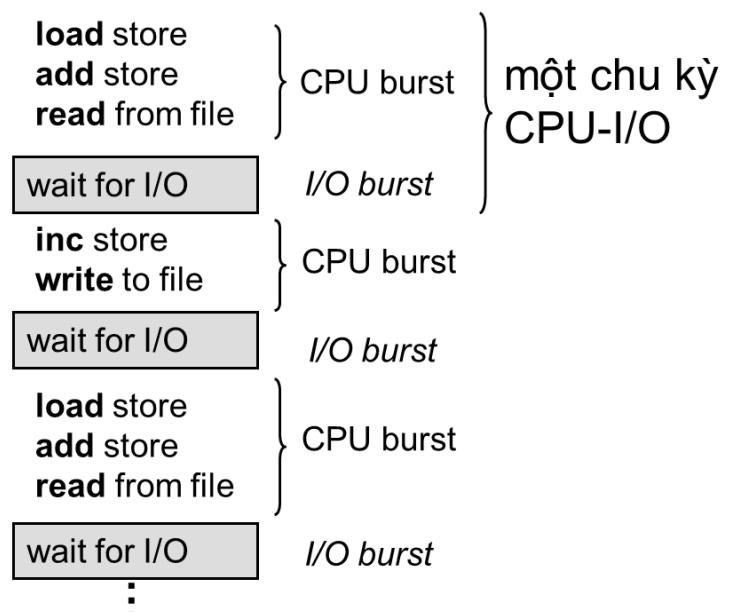
**Định thời** là chiến lược lựa chọn tiến trình phù hợp để được thực thi sao cho đạt được hiệu quả cao nhất.



# 1. Các khái niệm cơ bản về định thời

## Chu kỳ CPU-I/O

- **Service time** là thời gian một tiến trình cần CPU trong một chu kỳ CPU - I/O (hay còn gọi là **burst time**).
- Tiến trình có *service time* lớn được gọi là các **tiến trình hướng CPU** (CPU-bound process).



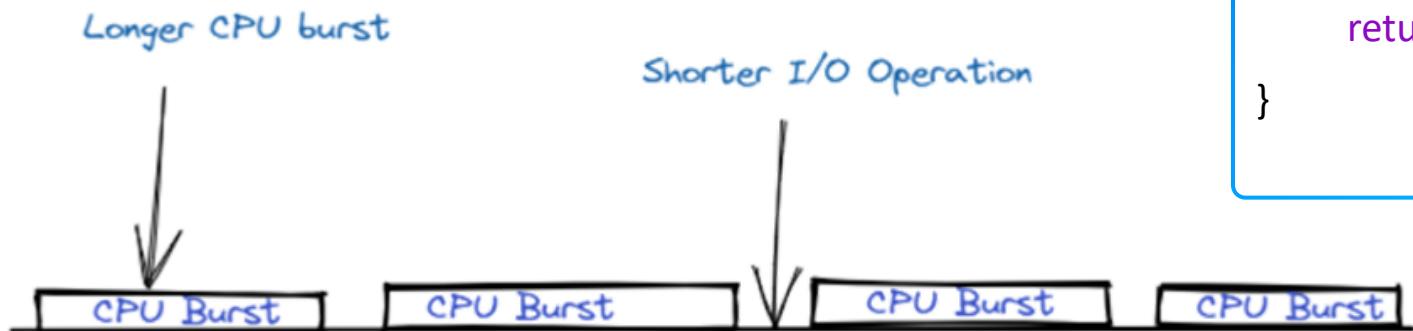
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



# 1. Các khái niệm cơ bản về định thời

## Tiến trình hướng CPU (CPU-bound)

- Tiến trình yêu cầu thời gian thực thi trên CPU nhiều.
- Thời gian hoàn thành chương trình phụ thuộc vào tốc độ thực thi của CPU.



```
#include <stdio.h>

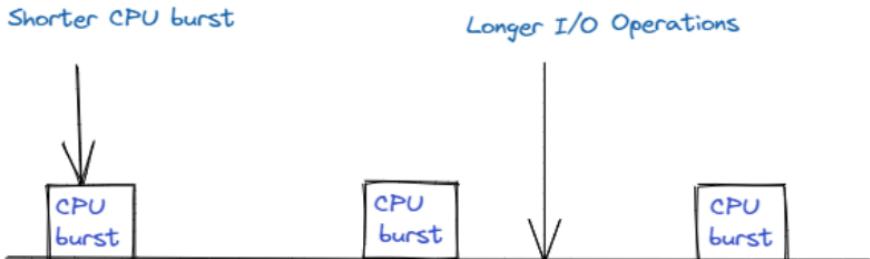
int main() {
    long long start = 1, end = 1000000,
    total = 0;
    for (long long i = start; i <= end;      i++){
        total += i;
    }
    printf("Sum of numbers from %lld to %lld is %lld\n",
start, end, total);
    return 0;
}
```



# 1. Các khái niệm cơ bản về định thời

## Tiến trình hướng I/O (I/O-bound)

- Tiến trình yêu cầu thời gian thực thi trên ngoại vi nhiều hơn.
- Thời gian hoàn thành chương trình phụ thuộc chu kỳ đợi cho các thao tác nhập/xuất.



```
#include <stdio.h>

int main(){
    FILE *fp;
    char filename[] = "example.txt";
    int total = 0, ch;
    fp = fopen(filename, "r");
    if (fp == NULL){
        printf("Failed to open file %s\n", filename);
        return 1;
    }
    while ((ch = fgetc(fp)) != EOF){
        total++;
    }
    fclose(fp);
    printf("Total number of characters in file %s is %d\n", filename,
    total);
    return 0;
}
```

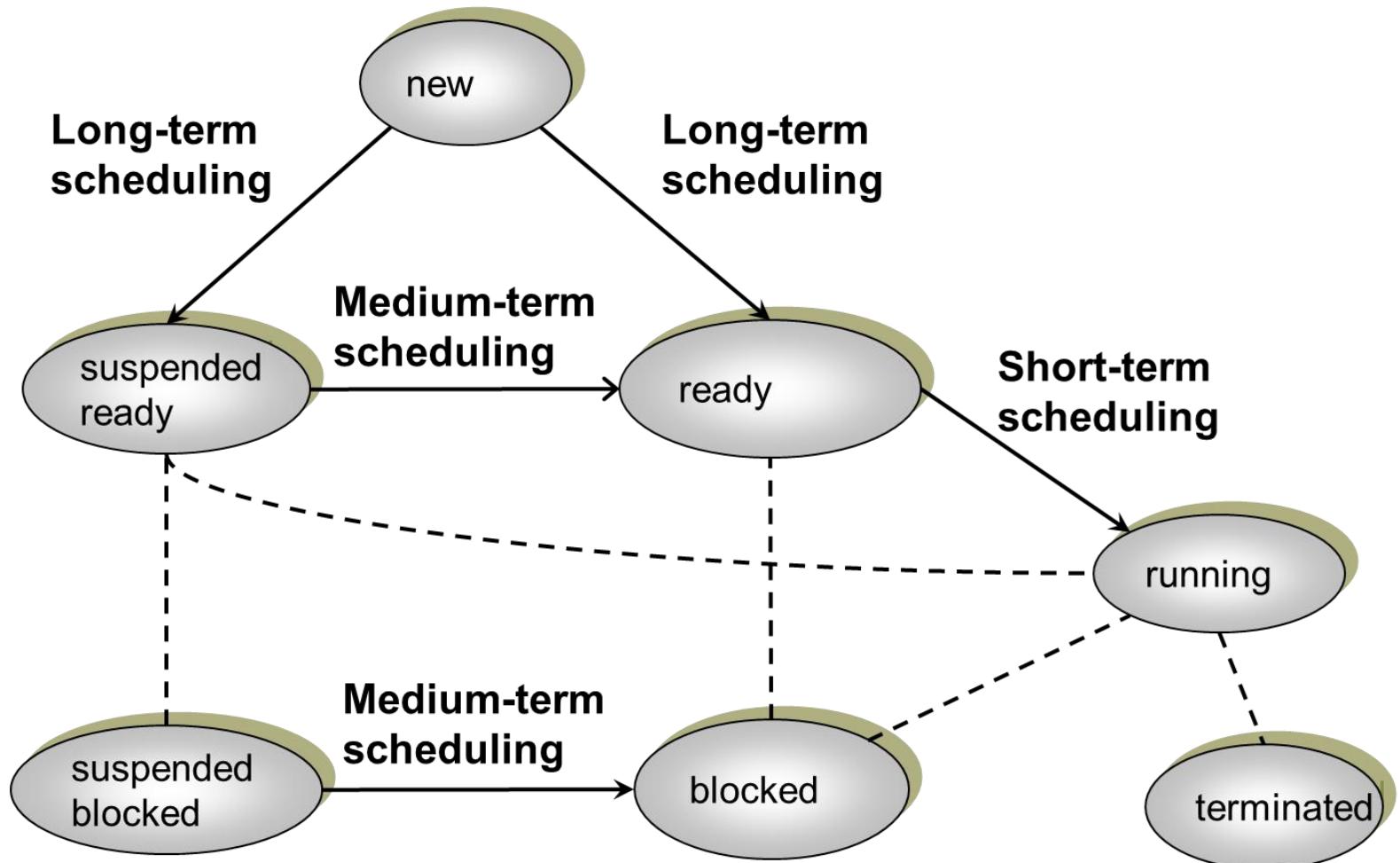


# CÁC LOẠI ĐỊNH THỜI

02.



## 2. Các loại định thời





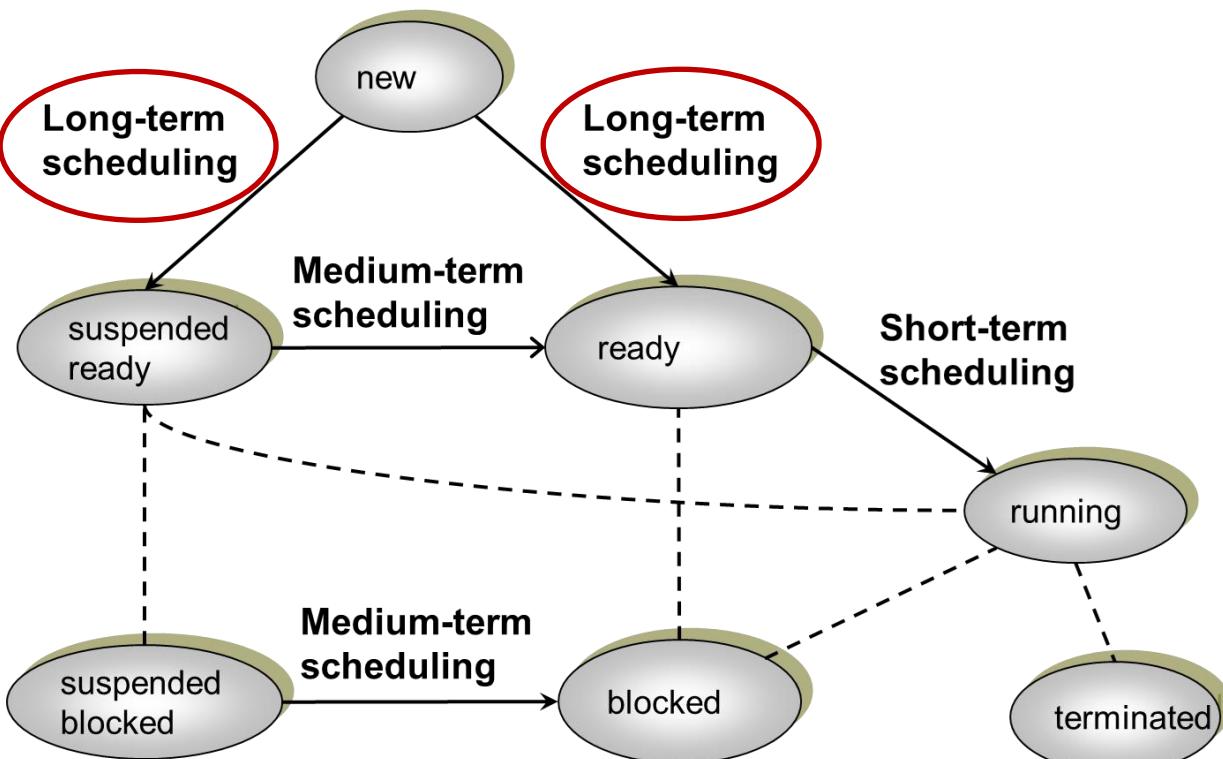
# CÁC LOẠI ĐỊNH THỜI

## 2.1. Định thời dài (Long-term scheduling)

02.

## 2.1. Định thời dài (Long-term Scheduling)

- Xác định chương trình nào được chấp nhận nạp vào hệ thống để thực thi.  
→ **Điều khiển mức độ đa chương của hệ thống.**
- Định thời dài thường cố gắng duy trì xen lẩn giữa tiến trình hướng CPU (CPU-bound process) và tiến trình hướng I/O (I/O-bound process).





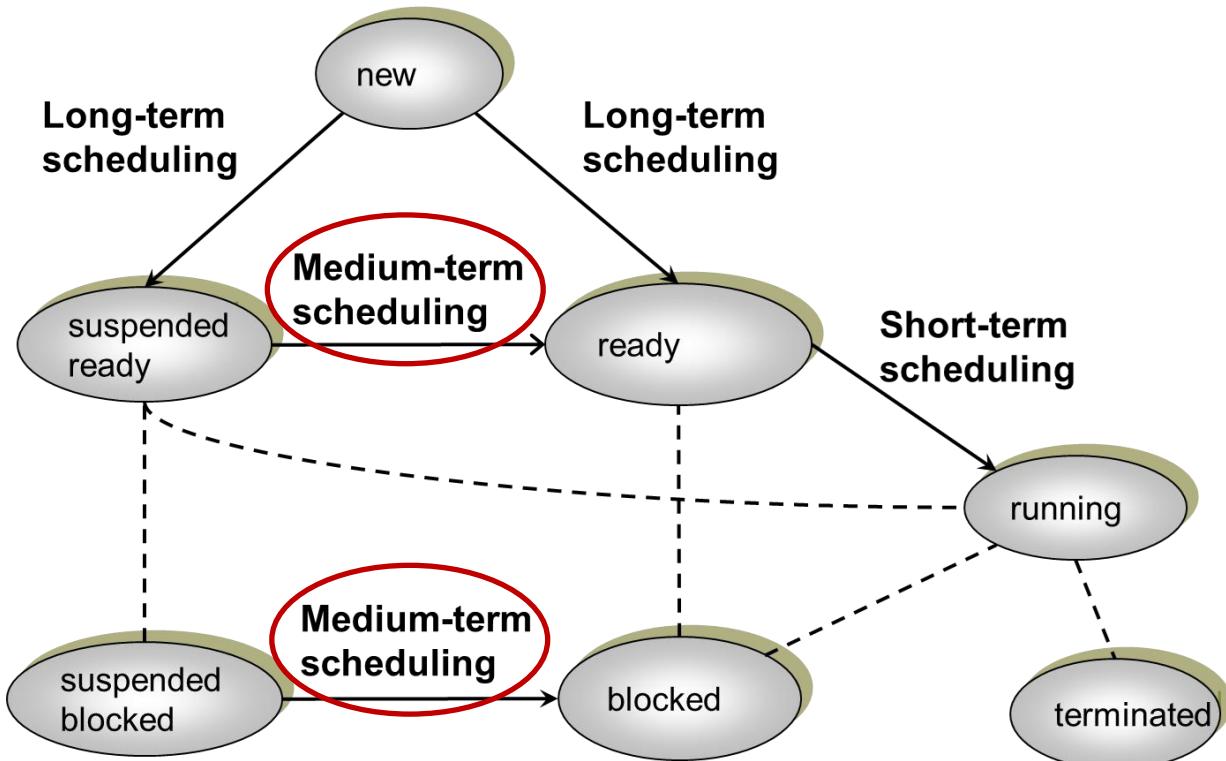
# CÁC LOẠI ĐỊNH THỜI

## 2.2. Định thời vừa (Medium-term scheduling)

02.

## 2.2. Định thời vừa (Medium-term scheduling)

- Định thời vừa quyết định tiến trình nào được đưa vào (swap in) và đưa ra khỏi (swap out) bộ nhớ chính trong quá trình thực thi của hệ thống.
- Được thực hiện bởi thành phần quản lý bộ nhớ (và sẽ được thảo luận ở chương về quản lý bộ nhớ).





# CÁC LOẠI ĐỊNH THỜI

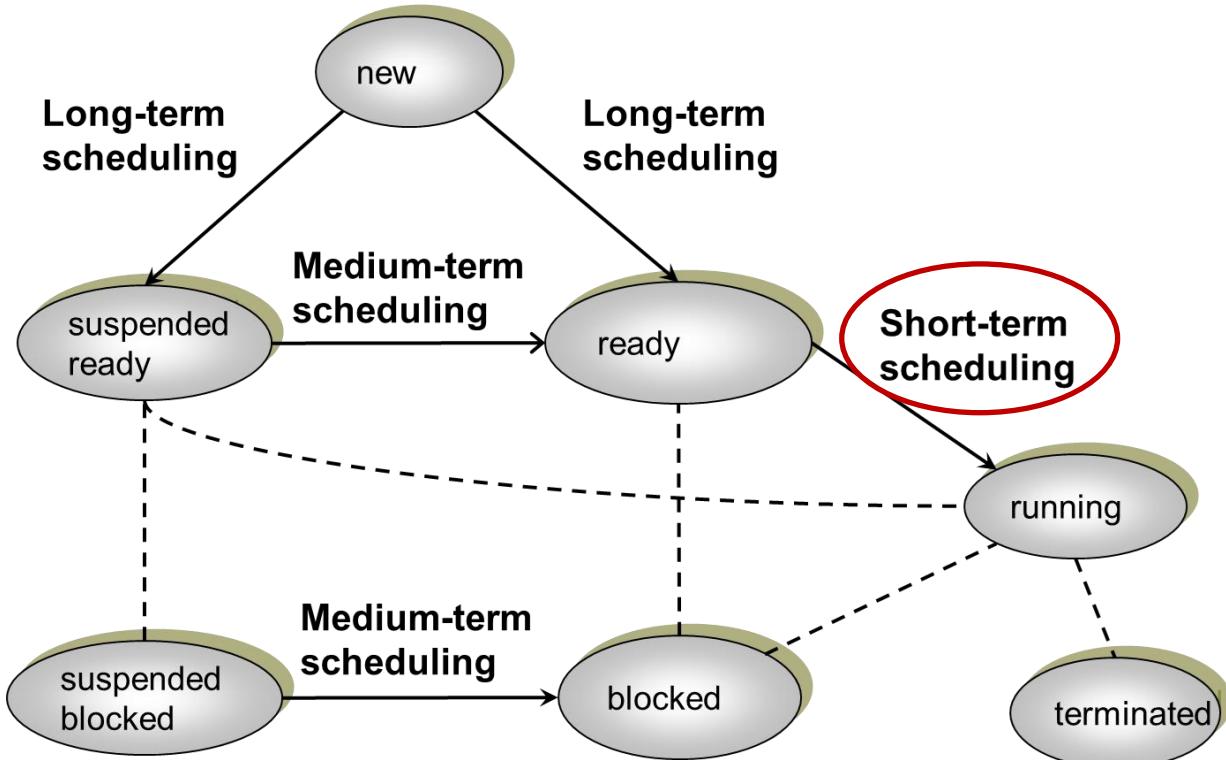
## 2.3. Định thời ngắn (Short-term scheduling)

02.



## 2.3. Định thời ngắn (Short-term scheduling)

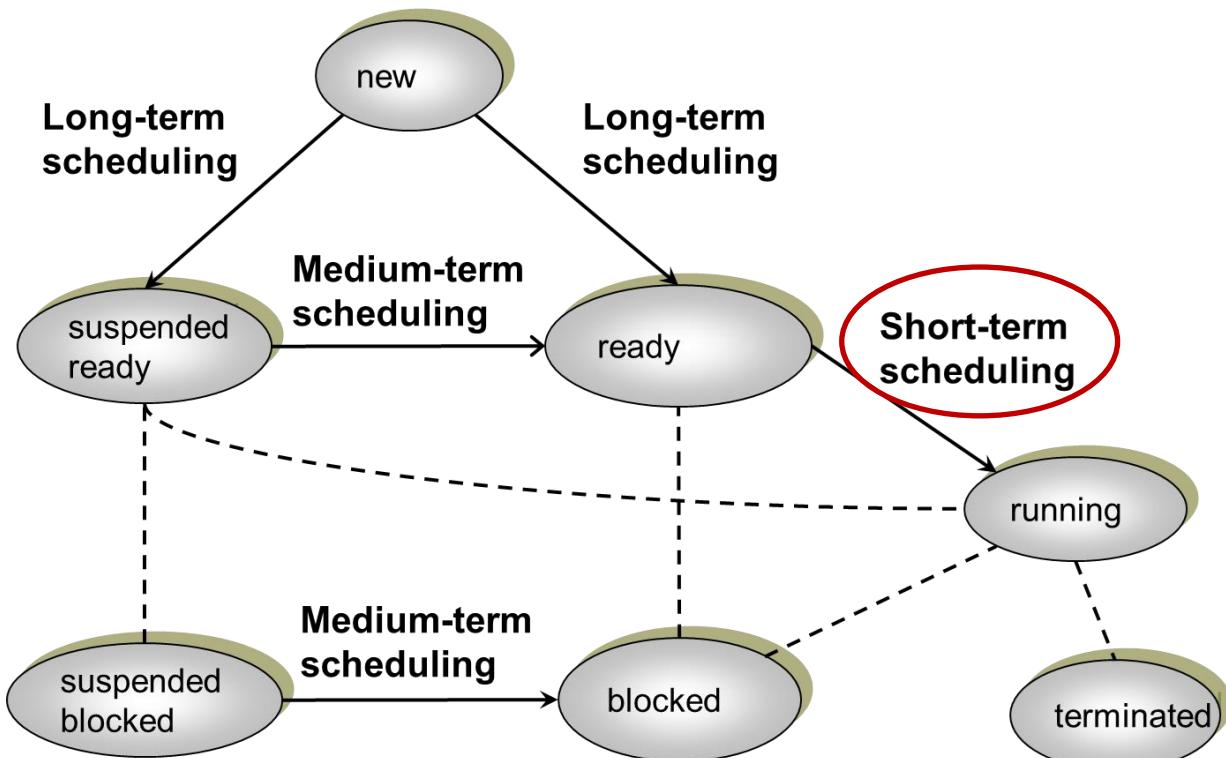
- Còn được gọi là **định thời CPU** (**CPU scheduling**).
- Xác định tiến trình nào trong **hàng đợi sẵn sàng (ready queue)** sẽ được chiếm CPU để thực thi kế tiếp.
- Đối với hệ thống hỗ trợ nhân đa luồng (multithreaded kernel), việc định thời CPU là do OS chọn **kernel thread** được chiếm CPU.





## 2.3. Định thời ngắn (Short-term scheduling)

- Bộ định thời ngắn được gọi khi có một trong các sự kiện/interrupt sau xảy ra:
  - Ngắt thời gian (clock interrupt)
  - Ngắt ngoại vi (I/O interrupt)
  - Lời gọi hệ thống (operating system call)
  - Tín hiệu đồng bộ hóa (Sẽ trao đổi sau ở Chương 5)





## 2.3. Định thời ngắn (Short-term scheduling)

### Bộ định thời ngắn (Short-term scheduler)

- Bộ định thời sẽ chuyển quyền điều khiển CPU về cho tiến trình được chọn.
- Quá trình chuyển đổi 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 (sử dụng thông tin địa chỉ tại program counter trong PCB).
- Công việc này gây ra phí tổn
  - **Dispatch latency**: thời gian mà bộ định thời dừng một tiến trình và khởi động một tiến trình khác.



# CÁC TIÊU CHUẨN ĐỊNH THỜI CPU

03.



### 3. Các tiêu chuẩn định thời CPU

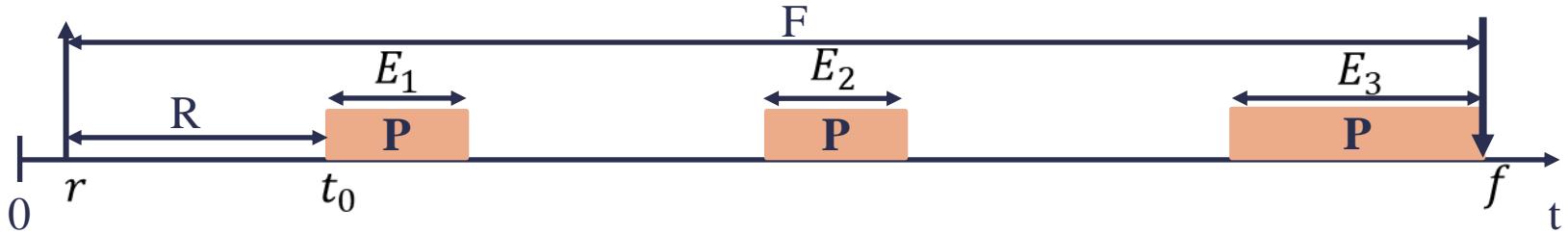
#### Hướng người dùng (user-oriented)

- **Thời gian đáp ứng (Response time)**: khoảng thời gian từ lúc tiến trình gửi yêu cầu thực thi đến khi yêu cầu được đáp ứng lần đầu tiên (trong các hệ thống time-sharing, interactive system) → **cực tiểu**
- **Thời gian hoàn thành (Turnaround time)**: khoảng thời gian từ lúc một tiến trình được nạp vào hệ thống đến khi tiến trình đó kết thúc → **cực tiểu**
- **Thời gian đợi (Waiting time)**: tổng thời gian một tiến trình đợi trong ready queue → **cực tiểu**



### 3. Các tiêu chuẩn định thời CPU

#### Cách xác định các thông số định thời



Giả sử :

- Quá trình thực thi một tiến trình P gồm nhiều phần.
- $r$  là thời điểm xuất hiện của P trong hệ thống (Arrival Time/Release Time).
- $t_0$  là thời điểm P được thực thi lần đầu tiên.
- $f$  là thời điểm tiến trình P hoàn thành việc thực thi (Finishing Time).

Gọi  $R$ ,  $F$ , và  $W$  lần lượt là thời gian đáp ứng, thời gian hoàn thành và thời gian đợi của tiến trình P. Khi đó:

$$R = t_0 - r$$

$$F = f - r$$

$$W = f - r - E$$

Trong đó  $E$  là thời gian yêu cầu của P để thực thi trên CPU (hay CPU Burst)

$$E = E_1 + E_2 + E_3$$



### 3. Các tiêu chuẩn định thời CPU

#### Hướng hệ thống (System oriented)

- **Hiệu năng sử dụng CPU** (*processor utilization*): định thời sao cho CPU càng bận càng tốt → **cực đại**
- **Tính công bằng** (*fairness*): tất cả tiến trình phải được **đối xử như nhau**.
- **Thông lượng** (*throughput*): số tiến trình hoàn tất công việc trong một đơn vị thời gian → **cực đại**



# CÁC GIẢI THUẬT ĐỊNH THỜI

## 4.1. Giải thuật định thời

04.



## 4.1. Giải thuật định thời

Một giải thuật định thời thông thường bao gồm hai yếu tố:

- **Hàm chọn lựa (selection function)**: mô tả cách thức (căn cứ) để chọn tiến trình nào trong *ready queue* được thực thi (Các hàm chọn lựa thường được xây dựng dựa trên độ ưu tiên, yêu cầu về tài nguyên, đặc điểm thực thi của tiến trình,...).
- **Chế độ quyết định (decision mode)**: quyết định thời điểm thực hiện hàm chọn lựa để định thời.



## 4.1. Giải thuật định thời

### Các chế độ quyết định

Có hai chế độ quyết định thường được áp dụng:

- **Không trưng dụng (Non-preemptive)**

- Khi ở trạng thái running, tiến trình sẽ thực thi cho đến khi kết thúc hoặc bị ngắt (blocked) do yêu cầu I/O.

- **Trưng dụng (Preemptive)**

- Tiến trình đang thực thi (ở trạng thái running) có thể bị ngắt giữa chừng và chuyển về trạng thái ready.
- Chi phí cao hơn chế độ không trưng dụng nhưng đánh đổi lại bằng thời gian đáp ứng tốt hơn vì không có trường hợp một tiến trình độc chiếm CPU quá lâu.



# 4.1. Giải thuật định thời

## Thời điểm thực thi hàm chọn lựa

- Hàm chọn lựa được thực thi vào các thời điểm sau:
  - (1) Có tiến trình chuyển từ trạng thái **running** sang **waiting**.
  - (2) Có tiến trình chuyển từ trạng thái **running** sang **ready**.
  - (3) Có tiến trình chuyển từ trạng thái **waiting**, **new** sang **ready**.
  - (4) Kết thúc thực thi của một tiến trình.

→ (1) và (4) không cần lựa chọn loại định thời, (2) và (3) cần.

- Việc thực thi hàm chọn lựa trong trường hợp (1) và (4) không phụ thuộc vào loại giải thuật định thời và thường áp dụng chế độ không trưng dụng.
- Ngược lại, trường hợp (2) và (3) phụ thuộc vào loại giải thuật định thời và thường áp dụng chế độ trưng dụng.

→ Thực hiện theo cơ chế nào khó hơn?  
Tại sao?



# 4.1. Giải thuật định thời

## 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

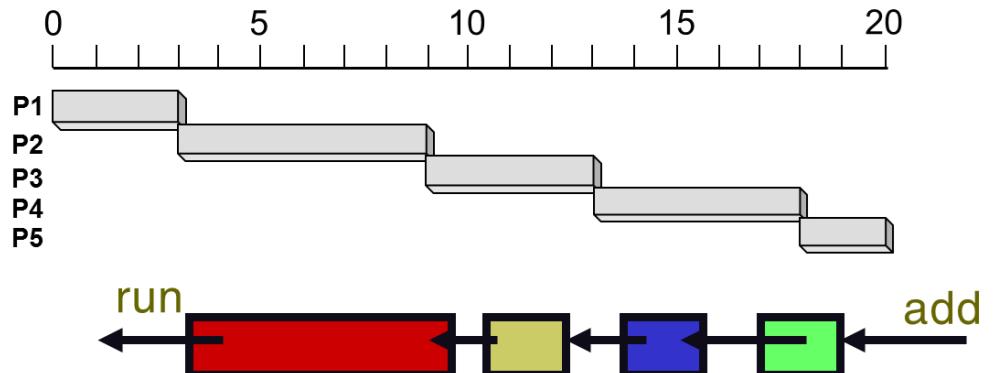
## 4.2. First-Come, First-Served (FCFS)

04.



## 4.2. First-Come, First-Served (FCFS)

- Hàm lựa chọn:
  - Tiến trình nào yêu cầu CPU trước sẽ được cấp phát CPU trước.
  - Tiến trình sẽ thực thi đến khi kết thúc hoặc bị blocked do I/O.
- Chế độ quyết định: không trung dung (non-preemptive).
- Hiện thực: sử dụng hàng đợi FIFO (FIFO queues)
  - Tiến trình mới xuất hiện được thêm vào cuối hàng đợi.
  - Tiến trình được lựa chọn để xử lý được lấy từ đầu của hàng đợi.

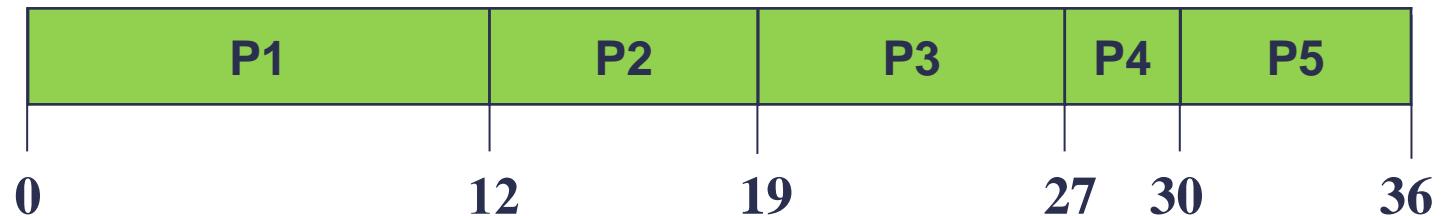




## 4.2. First-Come, First-Served (FCFS)

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

### Giản đồ Gantt



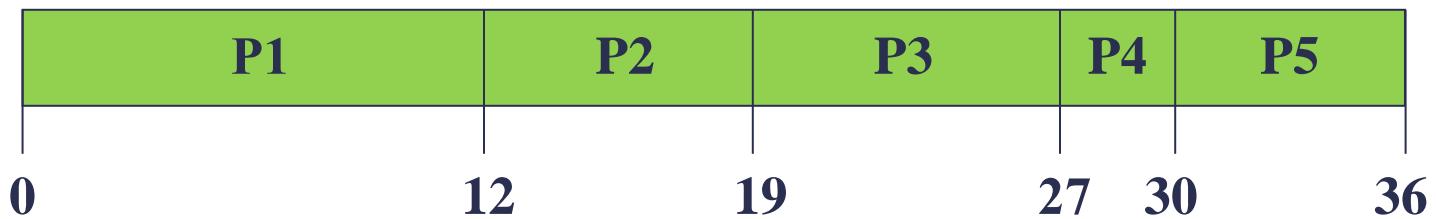


## 4.2. First-Come, First-Served (FCFS)

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 = 10, P3 = 14, P4 = 18, P5 = 18
- Thời gian đáp ứng trung bình:  
 $(0 + 10 + 14 + 18 + 18)/5 = 12$

### Giản đồ Gantt



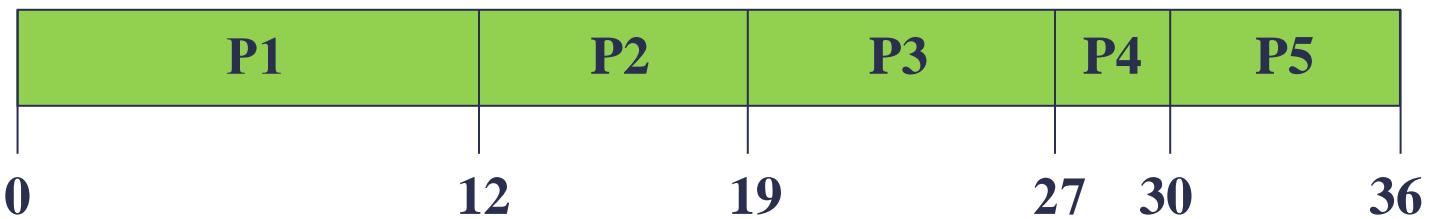


## 4.2. First-Come, First-Served (FCFS)

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

- Thời gian chờ:
  - P1 = 0, P2 = 10, P3 = 14, P4 = 18, P5 = 18
- Thời gian chờ trung bình:  
 $(0 + 10 + 14 + 18 + 18)/5 = 12$

### Giản đồ Gantt



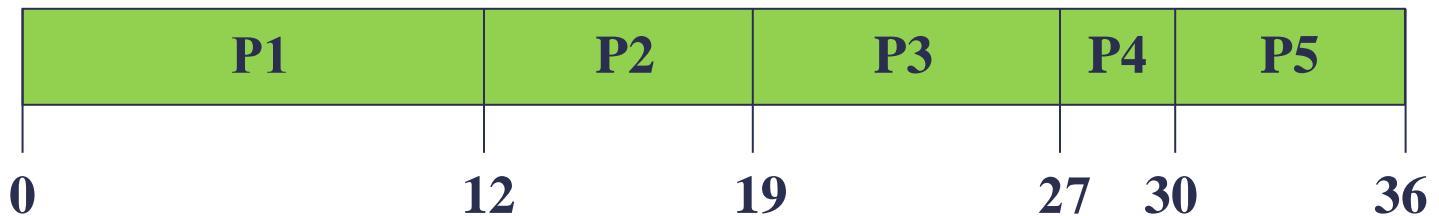


## 4.2. First-Come, First-Served (FCFS)

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

- Thời gian hoàn thành:
  - P1 = 12, P2 = 17, P3 = 22, P4 = 21, P5 = 24
  - Thời gian hoàn thành trung bình: $(12 + 17 + 22 + 21 + 24)/5 = 19.2$

### Giản đồ Gantt





# CÁC GIẢI THUẬT ĐỊNH THỜI

## 4.3. Shortest-Job-First (SJF)

04.



## 4.3. Shortest-Job-First (SJF)

- Hàm chọn lựa: tiến trình có thời gian yêu cầu thực thi (CPU burst) ngắn nhất sẽ được chọn.
  - Khi CPU trống, HĐH sẽ chọn tiến trình có *CPU Burst* ngắn nhất để được thực thi tiếp theo.
  - Giải thuật này sử dụng chiều dài thời gian thực thi của tiến trình làm căn cứ để chọn lựa.
- SJF có thể được hiện thực với cả hai chiến lược: trưng dụng và không trưng dụng.



## 4.3. Shortest-Job-First (SJF)

- SJF ở chế độ không trung dung:
  - Hàm chọn lựa được thực thi khi CPU trống.
  - Khi tiến trình được cấp CPU thì sẽ thực thi cho đến khi kết thúc.
  - Khi một tiến trình kết thúc, một tiến trình khác có thời gian thực thi ngắn nhất sẽ được chọn.

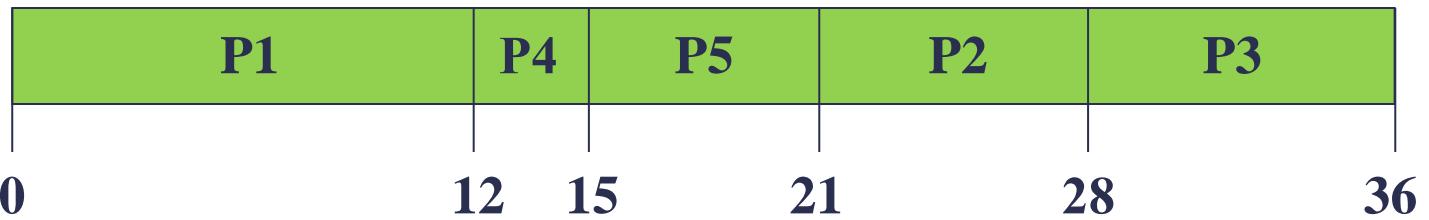


# Ví dụ: SJF ở chế độ không trung dung

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

- Thời gian đáp ứng:
  - $P1 = 0, P2 = 19, P3 = 23, P4 = 3, P5 = 3$
  - Thời gian đáp ứng trung bình: $(0 + 19 + 23 + 3 + 3)/5 = 9.6$

## Giản đồ Gantt



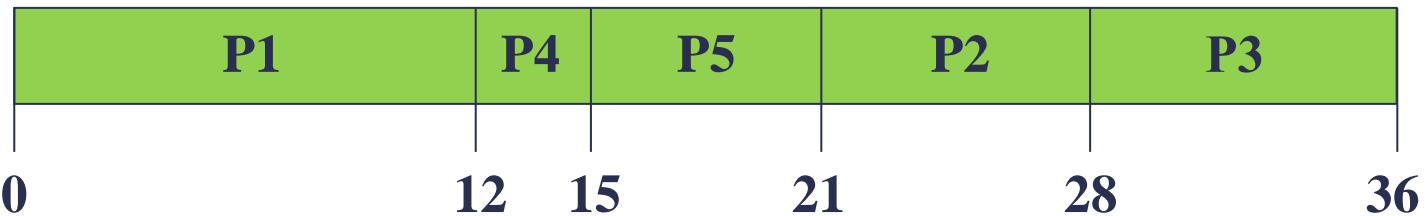


# Ví dụ: SJF ở chế độ không trung dung

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

- Thời gian chờ:
  - $P1 = 0, P2 = 19, P3 = 23, P4 = 3,$   
 $P5 = 3$
  - Thời gian chờ trung bình:  
 $(0 + 19 + 23 + 3 + 3)/5 = 9.6$

## Giản đồ Gantt



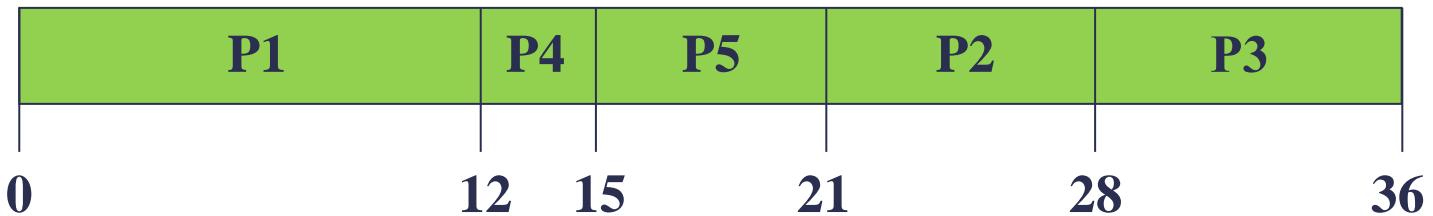


# Ví dụ: SJF ở chế độ không trung dung

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

- Thời gian hoàn thành:
  - P1 = 12, P2 = 26, P3 = 31, P4 = 6, P5 = 9
  - Thời gian hoàn thành trung bình:  $(12 + 26 + 31 + 6 + 9)/5 = 16.8$

## Giản đồ Gantt





## 4.3. Shortest-Job-First (SJF)

- **SJF ở chế độ trung dung**

- Hàm chọn lựa được thực thi khi có tiến trình mới xuất hiện hoặc có tiến trình kết thúc.
- Khi có tiến trình mới xuất hiện với **CPU-burst nhỏ hơn thời gian yêu cầu còn lại** (remaining time) của tiến trình đang thực thi, **tiến trình mới sẽ được chọn và tiến trình đang thực thi sẽ bị dừng lại**.
- Khi một tiến trình kết thúc, một tiến trình khác có CPU-burst (hoặc thời gian yêu cầu còn lại) nhỏ nhất sẽ được chọn tiếp theo.
- **SJF ở chế độ trung dung còn được gọi là Shortest-Remaining-Time-First (SRTF).**

**SJF là tối ưu về thời gian đợi: có thời gian chờ đợi trung bình ngắn nhất với một tập tiến trình cho trước.**



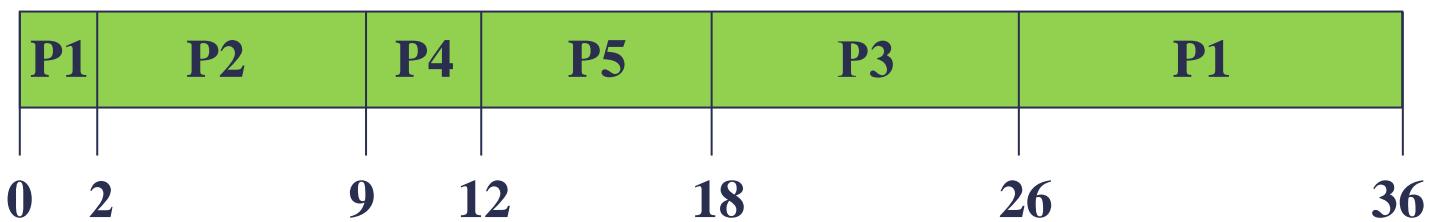
## 4.3. Shortest-Job-First (SJF)

### SJF trưng dụng

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: $(0 + 0 + 13 + 0 + 0)/5 = 2.6$

### Giản đồ Gantt





## 4.3. Shortest-Job-First (SJF)

### SJF trưng dụng

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

- Thời gian chờ:
  - P1 = 24, P2 = 0, P3 = 13, P4 = 0, P5 = 0
- Thời gian chờ trung bình:
$$(24 + 0 + 13 + 0 + 0)/5 = 7.4$$

### Giản đồ Gantt





## 4.3. Shortest-Job-First (SJF)

### SJF trưng dụng

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

- Thời gian hoàn thành:
  - $P1 = 36, P2 = 7, P3 = 21, P4 = 3, P5 = 6$
  - Thời gian hoàn thành trung bình: $(36 + 7 + 21 + 3 + 6)/5 = 14.6$

### Giản đồ Gantt





## 4.3. Shortest-Job-First (SJF)

### Nhận xét về giải thuật SJF

- Có thể xảy ra tình trạng “đói” tài nguyên (**starvation**) đối với các tiến trình có *CPU-burst* lớn nếu có nhiều tiến trình với *CPU-burst* nhỏ (liên tục) xuất hiện trong hệ thống.
- Cơ chế không trưng dụng không phù hợp cho hệ thống time sharing (interactive).
- Giải thuật SJF ngầm định rằng độ ưu tiên được xác định dựa theo độ dài *CPU-burst*.
  - Các tiến trình hướng CPU (CPU-bound) có độ ưu tiên thấp hơn so với tiến trình hướng I/O (I/O-bound).
  - Tuy nhiên, khi một tiến trình hướng CPU được thực thi thì nó độc chiếm CPU cho đến khi kết thúc.



## 4.3. Shortest-Job-First (SJF)

### Nhận xét về giải thuật SJF

- **Ưu điểm:** SJF tối ưu trong việc giảm thời gian đợi trung bình.
- **Hạn chế:** Cần phải ước lượng lượng thời gian cần CPU tiếp theo của tiến trình.

→ Giải pháp cho vấn đề này?



## 4.3. Shortest-Job-First (SJF)

### Nhận xét về giải thuật SJF

- Thời gian sử dụng CPU chính là độ dài của CPU burst:
  - Trung bình tất cả các CPU Burst đo được trong quá khứ.
  - Nhưng thông thường những CPU Burst càng mới càng phản ánh đúng hành của tiến trình trong tương lai.
- Một kỹ thuật thường dùng là sử dụng trung bình hàm mũ (exponential averaging):

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n, 0 \leq \alpha \leq 1$$

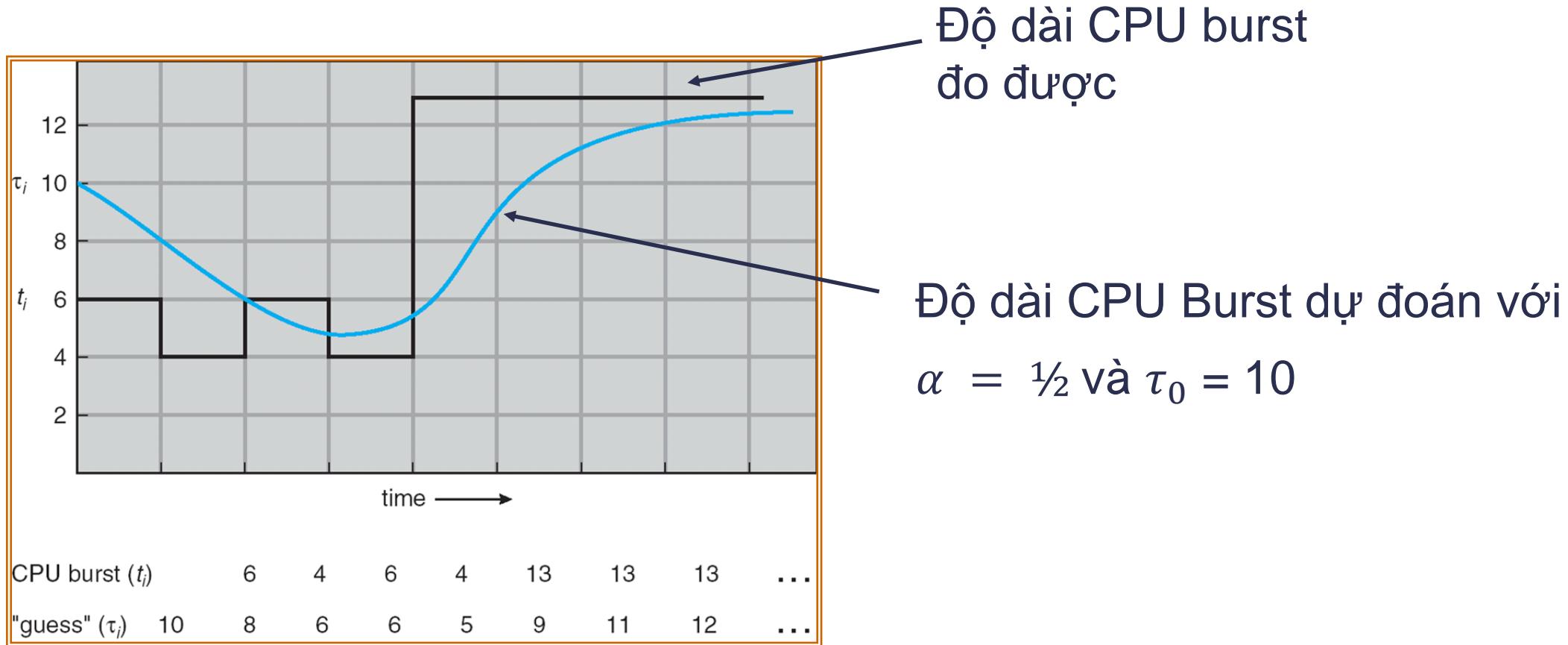
$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\alpha t_{n-1} + \dots + (1 - \alpha)^j \alpha t_{n-j} + \dots + (1 - \alpha)^{n+1}\tau_0, 0 \leq \alpha \leq 1$$

Nếu chọn  $\alpha = 1/2$  thì có nghĩa là trị đo được  $t_n$  và trị dự đoán  $\tau_n$  được xem quan trọng như nhau.



## 4.3. Shortest-Job-First (SJF)

### Dự đoán thời gian sử dụng CPU





# CÁC GIẢI THUẬT ĐỊNH THỜI

## 4.4. Định thời theo độ ưu tiên - Priority Scheduling

04.



## 4.4. Định thời theo độ ưu tiên – Priority Scheduling

- Mỗi tiến trình sẽ được gán một độ ưu tiên (thường biểu diễn bởi một con số).
- CPU sẽ được cấp cho tiến trình có độ ưu tiên cao nhất theo các giá trị số được gán (có thể theo thứ tự tăng dần hay giảm dần).
- Định thời sử dụng độ ưu tiên có thể:
  - Preemptive
  - Non-preemptive



## 4.4. Định thời theo độ ưu tiên – Priority Scheduling

### Cách gán độ ưu tiên cho tiến trình

- SJF là một giải thuật định thời sử dụng độ ưu tiên được xác định dựa vào thời gian sử dụng CPU (giá trị được ước lượng).
- Ngoài ra, việc gán độ ưu tiên còn có thể dựa vào:
  - Yêu cầu về bộ nhớ.
  - Số lượng file được mở.
  - Tỉ lệ thời gian dùng cho I/O trên thời gian sử dụng CPU.
  - Các yêu cầu bên ngoài ví dụ như: số tiền người dùng trả khi thực thi công việc.



## 4.4. Định thời theo độ ưu tiên – Priority Scheduling

### Hạn chế của định thời theo độ ưu tiên

- Vấn đề trì hoãn vô hạn định: tiến trình có độ ưu tiên thấp có thể không bao giờ được thực thi (do có những tiến trình độ ưu tiên cao hơn liên tục xuất hiện).
- Giải pháp: làm mới (aging) – độ ưu tiên của tiến trình sẽ tăng theo thời gian.



## 4.4. Định thời theo độ ưu tiên – Priority Scheduling

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 chờ?
- Thời gian đáp ứng?
- Thời gian hoàn thành?

### Giản đồ Gantt (Non-preemptive)





# Tóm tắt lại nội dung buổi học

- Các khái niệm cơ bản về định thời
- Các bộ định thời
- Các tiêu chuẩn định thời CPU
- Các giải thuật định thời
  - First-Come, First-Served (FCFS)
  - Shortest Job First (SJF)
  - Shortest Remaining Time First (SRTF)
  - Priority Scheduling



# Bài tập 1

- Sử dụng các giải thuật FCFS, SJF, SRTF, Priority để tính các giá trị thời gian đợi, thời gian đáp ứng và thời gian hoàn thành trung bình.

Process	Arrival Time	Burst Time	Priority
P1	0	4	3
P2	5	1	2
P3	3	8	1
P4	10	2	4
P5	8	7	3



# Bài tập 2

- Sử dụng các giải thuật FCFS, SJF, SRTF, Priority để tính các giá trị thời gian đợi, thời gian đáp ứng và thời gian hoàn thành trung bình.

Thread	Priority	Burst	Arrival
$P_1$	40	20	0
$P_2$	30	25	25
$P_3$	30	25	30
$P_4$	35	15	60
$P_5$	5	10	100
$P_6$	10	10	105



# THẢO LUẬN



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