CHAPTER 5

# Khi nào thì xảy ra tranh chấp?

* Race condition xảy ra khi nhiều process truy xuất vào thao tác đồng thời lên dữ liệu chia sẻ.

# Vấn đề vùng tranh chấp (critical section) là gì?

* Critical section problems là vấn đề về tìm cách thiết kế một giao thức nào đó để cách process có thể phối hợp với nhau hoàn thành nhiệm vụ của nó.

# Có những yêu cầu nào cho lời giải của bài toán vùng tranh chấp?

* Multual excution: khi một process P đang thực thi trong vùng tranh chấp (CS) của nó thì không có một process Q nào thực thi trong CS của Q.
* Progress: một tiến trình tạm dừng bên ngoài CS không được ngăn chặn các tiến trình khác vào CS.
* Bounded Waiting: mỗi process chỉ phải chờ để được vào CS trong một khoảng thời gian có hạn. Không được xảy ra tình trạng đói tài nguyên.

# Có mấy loại giải pháp đồng bộ? Kể tên và trình bày đặc điểm của các loại giải pháp đó?

* Có 2 nhóm giải pháp chính: Busy waiting, Sleep and Wakeup.
* Busy Waiting
  + - Tiếp tục tiêu thụ CPU trong khi chờ đợi vào miền găng (thông qua việc kiểm tra điều kiện vào CS liên tục).
    - Không đòi hỏi sự trợ giúp của hệ điều hành.
* Sleep and Wakeup
  + - Từ bỏ CPU khi chưa vào được CS.
    - Cần sự hổ trợ của hệ điều hành (để đánh thức process và đưa process vào trạng thái blocked).

# Phân tích và đánh giá ưu, nhược điểm của các giải pháp đồng bộ busy waiting (cả phần cứng và phần mềm)?

# 10. (Bài giải)

* Xét 2 process P0 và P1 chạy song song.
  + - Khi P0 chạy, i = 0, j = 1, flag [0] == TRUE, turn = 0.
    - Khi P1 chạy, I = 1, j = 0, flag [1] == TRUE, turn = 1.
* Xét điều kiện *while (turn == j && flag[j]==TRUE)*
  + - Tại P0, turn(0) == 1 (không thỏa), flag [1] == TRUE (thỏa), thoát loop vào miền găng.
    - Tại P1, turn(1) == 0 (không thỏa), flag [0] == TRUE (thỏa), thoát loop vào miền găng.
* Không thỏa Multual Excution.

# 11. (Bài giải)

* Chỉ thị swap có thể tổ chức truy xuất độc quyền.

Void Swap (bool a, bool b) {

bool temp;

temp = a;

a = b;

b = temp;

}

# 12. (Bài giải)

1. Semaphore\_1 = 0;

Semaphore\_2 = 10;

Void processA ()

{

While (1)

{

wait (Semaphore\_2);

na = na + 1;

signal (Semaphore\_1);

}

}

Void processB ()

{

While (1)

{

wait (Semaphore\_1);

nb = nb + 1;

signal(semaphore\_2);

}

}

1. Semaphore\_2 = 10;

Void processA ()

{

While (1)

{

wait(Semaphore\_2);

na = na + 1;

}

}

Void processB ()

{

While (1)

{

nb = nb + 1;

signal(semaphore\_2);

}

1. Đúng.

# 14. (Bài giải)

Semaphore\_2 = 19;

Void processA ()

{

While (true)

{

wait(Semaphore\_2);

x = x + 1;

if (x == 20) x = 0;

}

}

Void processB ()

{

While (1)

{

x = x + 1;

if (x == 20) x = 0;

signal(semaphore\_2);

}