MÔN HỌC: HỆ ĐIỀU HÀNH
CÂU HỎI VÀ BÀI TẬP CHƯƠNG 5

1. Khi nào thì xảy ra tranh chấp?
2. Vấn đề vùng tranh chấp (critical section) là gì?
3. Có những yêu cầu nào dành cho lời giải của bài toán vùng tranh chấp?
4. Phân tích các giải pháp đồng bộ dựa trên ngắt (giải pháp phần mềm)? Các giải pháp này có những vấn đề gì?
5. Trình bày các giải pháp đồng bộ dựa trên phần cứng?
6. Mutex lock là gì? Đặc điểm và cách sử dụng của mutex lock?
7. Semaphore là gì? Đặc điểm của semaphore? Cách thức hiện thực semaphore? Có mấy loại semaphore? Khi sử dụng semaphore cần lưu ý những vấn đề gì?
8. Monitor là gì?
9. Đặc điểm và yêu cầu đồng bộ của các bài toán đồng bộ kinh điển?
10. (Bài tập mẫu) Xét giải pháp phần mềm do Dekker đề nghi để tổ chức truy xuất độc quyền cho 2 tiến trình. Hai tiến trình P0 và P1 chia sẻ các biến sau:

```
boolean flag[2]; /* initially false */
int turn;
```

Cấu trúc một tiến trình Pi (với i = 0 hay 1 và j là tiến trình còn lại) như sau:

```
while (true) {
      flag[i] = true;

      while (flag[j]) {
            if (turn == j) {
                  flag[i] = false;
                  while (turn == j)
                        ; /* do nothing */
                  flag[i] = true;
            }
      }
            /* critical section */
      turn = j;
      flag[i] = false;
            /* remainder section */
}
```

Giải pháp này có thỏa 3 yêu cầu trong viêc giải quyết tranh chấp không?

Trả lời:

Giải pháp này thỏa 3 yêu cầu trong giải quyết tranh chấp vì:

- Loại trừ tương hỗ: Tiến trình Pi chỉ có thể vào vùng tranh chấp khi flag[j] = false. Giả sử P0 đang ở trong vùng tranh chấp, tức là flag[0] = true và flag[1] = false. Khi đó P1 không thể vào vùng tranh chấp (do bị chặn bởi lệnh while (flag[j])). Tương tự cho tình huống P1 vào vùng tranh chấp trước.
- Progress: Giá trị của biến turn chỉ có thể thay đổi ở cuối vùng tranh chấp. Giả sử chỉ có 1 tiến trình Pi muốn vào vùng tranh chấp. Lúc này, flag[j] = false và tiến trình Pi sẽ được vào vùng tranh chấp ngay lập tức. Xét trường hợp cả 2 tiến trình đều muốn vào vùng tranh chấp và giá trị của turn đang là 0. Cả flag[0] và flag[1] đều bằng true. Khi đó, P0 sẽ được vào vùng tranh chấp, bởi tiến trình P1 sẽ thay đổi flag[1] = false (lệnh kiểm tra điều kiện if (turn == j) chỉ đúng với P1). Tương tự cho trường hợp turn = 1.
- Chờ đợi giới hạn: Pi chờ đợi lâu nhất là sau 1 lần Pj vào vùng tranh chấp (flag[j] = false sau khi Pj ra khỏi vùng tranh chấp). Tương tự cho trường hợp Pj chờ Pi.

11. Xét giải pháp đồng bộ hóa sau:

```
while (TRUE) {
      int j = 1-i;
      flag[i]= TRUE;
      turn = i;
```

```
      while (turn == j && flag[j]==TRUE);
      critical-section ();
      flag[i] = FALSE;
      Noncritical-section ();
}
```

Giải pháp này có thỏa yêu cầu độc quyền truy xuất không?

12. Giả sử một máy tính không có chỉ thị TSL, nhưng có chỉ thị Swap có khả năng hoán đổi nội dung của hai từ nhớ chỉ bằng một thao tác không thể phân chia:

```
procedure Swap(var a,b: boolean){
      var temp : boolean;
      begin
            temp := a;
            a:= b;
            b:= temp;
      end;
}
```

Sử dụng chỉ thị này có thể tổ chức truy xuất độc quyền không? Nếu có, xây dựng cấu trúc chương trình tương ứng.

13. Xét hai tiến trình sau:

process A {while (TRUE) na = na +1;}

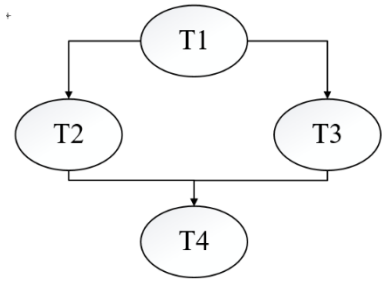
process B {while (TRUE) nb = nb +1;}

a. Đồng bộ hóa xử lý của 2 tiến trình trên, sử dụng 2 semaphore tổng quát, sao cho tại bất kỳ thời điểm nào cũng có nb <= na <= nb +10.

b. Nếu giảm điều kiện chỉ còn là na <= nb +10, cần sửa chữa giải pháp trên như thế nào?

c. Giải pháp trên còn đúng nếu có nhiều tiến trình loại A và B cùng thực hiện?

14. (Bài tập mẫu) Xét một hệ thống có 4 tiểu trình T1, T2, T3, T4. Quan hệ giữa các tiểu trình này được biểu diễn như sơ đồ bên dưới, với mũi tên từ tiểu trình (Tx) sang tiểu trình (Ty) có nghĩa là tiểu trình Tx phải kết thúc quá trình hoạt động của nó trước khi tiểu trình Ty bắt đầu thực thi. Giả sử tất cả các tiểu trình đã được khởi tạo và sẵn sàng để thực thi. Hãy sử dụng semaphore để đồng bộ hoạt động của các tiểu trình sao cho đúng với sơ đồ đã cho.



Trả lời:

Khai báo và khởi tạo các semaphore:

init(sem1,0); //khởi tạo semaphore sem1 có giá trị bằng 0

init(sem2,0); //khởi tạo semaphore sem2 có giá trị bằng 0

void T1(void)
{
 	
//T1 thực thi
 	
signal(sem1)
signal(sem1)
}
	void T2(void)
	{
	
	wait(sem1)
	
	//T2 thực thi
	
	signal(sem2)
	
	}
		void T3(void)
		{
		
		wait(sem1)
		
		//T3 thực thi
		
		signal(sem2)
		
		}
			void T4(void)
			{
			
			wait(sem2)
			wait(sem2)
			//T4 thực thi
			
			}

15. Một biến X được chia sẻ bởi 2 tiến trình cùng thực hiện đoạn code sau:

```
do
      X = X + 1;
      if (X == 20) X = 0;
while (TRUE);
```

Bắt đầu với giá trị X = 0, chứng tỏ rằng giá tri X có thể vượt quá 20. Cần sửa chữa đoạn chương trình trên như thế nào để đảm bảo X không vượt quá 20?

16. Xét 2 tiến trình xử lý đoạn chương trình sau:

process P1 {A1 ; A2 }

process P2 {B1 ; B2 }

Đồng bộ hóa hoạt động của 2 tiến trình này sao cho cả A1 và B1 đều hoàn tất trước khi A2 và B2 bắt đầu.

17. Tổng quát hóa bài tập 14 cho các tiến trình có đoạn chương trình sau:

process P1 { for (i = 1; i <= 100; i ++) A_i }

process P2 { for (j = 1; j <= 100; j ++) B_j }

Đồng bộ hóa hoạt động của 2 tiến trình này sao cho với k bất kỳ (2<=k<=100), A_k chỉ có thể bắt đầu khi $B_{(k-1)}$ đã kết thúc và B_k chỉ có thể bắt đầu khi $A_{(k-1)}$ đã kết thúc.

18. Sử dụng semaphore để viết lại chương trình sau theo mô hình xử lý đồng hành:

w := x1 * x2

v := x3 * x4

y := v * x5

z := v * x6

x := w * y

z := w * z

ans := y + z