

Bài ôn tập: Đồng bộ hoá tiến trình

- Câu 1: 2 nhu cầu trao đổi thông tin của tiến trình nhằm :
 - a. Chia sẻ tài nguyên chung, Phối hợp hoạt động
 - b. Xử lý song song, Phối hợp hoạt động
 - c. Bảo đảm độc lập, Thông báo lỗi

Đáp án : a

Bài ôn tập 3: Đồng bộ hoá tiến trình

- Câu 2: Race Condition là
 - a. Kết quả thực hiện tiến trình phụ thuộc vào kết quả điều phối
 - b. Hiện tượng các tiến trình chia sẻ tài nguyên chung
 - c. Kết quả tiến trình thực hiện luôn luôn sai

Đáp án : a

Bài ôn tập 3: Đồng bộ hoá tiến trình

- Câu 3: Critical section là
 - a. Tài nguyên dùng chung giữa các tiến trình
 - b. Cơ chế bảo vệ tài nguyên dùng chung
 - c. Đoạn chương trình có khả năng gây ra hiện tượng race condition
 - d. Đoạn chương trình có truy cập tài nguyên dùng chung

Đáp án: c

Bài ôn tập 3 : Đồng bộ hoá tiến trình

- Câu 4 : 2 nhu cầu đồng bộ tiến trình là :
 - a. Hò hẹn , Phối hợp hoạt động
 - b. Trao đổi thông tin, Phối hợp hoạt động
 - c. Độc quyền truy xuất , Giải quyết tranh chấp
 - d. Không có câu nào đúng

Đáp án : d

Bài ôn tập 3: Đồng bộ hoá tiến trình

- Câu 5: Cho biết các điều kiện cho một giải pháp đồng bộ tốt

Đáp án:

- **Mutual Exclusion**: Không có hai tiến trình cùng ở trong miền găng cùng lúc
- **Progress**: Một tiến trình tạm dừng bên ngoài miền găng không được ngăn cản các tiến trình khác vào miền găng
- **Bounded Waiting**: Không có tiến trình nào phải chờ vô hạn để được vào miền găng.
 - Không có giả thiết nào đặt ra cho sự liên hệ về tốc độ của các tiến trình, cũng như về số lượng bộ xử lý trong hệ thống

Câu 6 : Xét giải pháp phần mềm do Dekker đề nghị để tổ chức truy xét độc quyền cho hai tiến trình . Hai tiến trình P0, P1 chia sẻ các biến sau :

```
var flag : array [0..1] of boolean; (khởi động là false)
    turn : 0..1;
```

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

```
repeat
flag[i] := true;
while flag[j] do
if turn = j then
                begin
                    flag[i] := false;
                    while turn = j do ;
                    flag[i] := true;
                end;
    critical_section();
turn := j;
flag[i] := false;
non_critical_section();
until false;
```

Giải pháp này có phải là một giải pháp đúng thỏa mãn 4 yêu cầu không ?



Câu 6: Đáp án

- Đúng.
 - Giải pháp này bảo đảm yêu cầu độc quyền truy xuất vì khi cả 2 tiến trình P_i và P_j đồng thời quan tâm đến việc vào miền găng ($flag[i]=true$ và $flag[j]=true$) thì chỉ có một tiến trình được vào miền găng tùy theo giá trị của $turn$.
 - Nếu tiến trình P_i đang xử lý `Non_criticalsection`, thì trước đó $flag[i]$ đã được gán giá trị `false`, do vậy không ngăn cản P_j quay lại `criticalsection`

Câu 7: Xét giải pháp đồng bộ hoá 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 ();  
}
```

Đây có phải là một giải pháp bảo đảm được độc quyền truy xuất không ?

Đáp án :

- Không. Xét tình huống khi $flag[0] = 1; turn = 0 \Rightarrow P_0$ vào CS, nếu lúc đó $flag[1] = 1$, P_1 có thể gán $turn = 1$ và vào luôn CS !

Câu 8: 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;
```

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



Câu 8: Đáp án

```
while (TRUE)
{
    key = TRUE;
    while ( key = TRUE)
    swap(lock,key);
    critical-section ();
    lock = false;
    Noncritical-section();
}
```

■ Câu 9: Xét hai tiến trình sau :

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

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

- a. Đồng bộ hoá xử lý của hai tiến trình trên, sử dụng hai semaphore tổng quát, sao cho tại bất kỳ thời điểm nào cũng có $nb < na \leq nb + 10$
- b. Nếu giảm điều kiện chỉ là $na \leq nb + 10$, giải pháp của bạn sẽ được sửa chữa như thế nào ?
- c. Giải pháp của bạn có còn đúng nếu có nhiều tiến trình loại A và B cùng thực hiện?

Câu 9: Đáp án

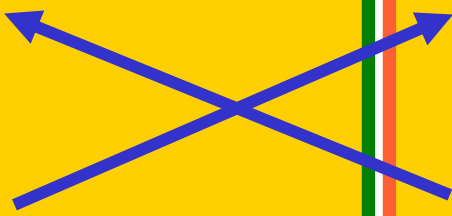
Đáp án : semaphore a = 0 ; b = 10 ;

Process A()

```
{  
  int item;  
  while (TRUE)  
  {  
    down(b);  
    na = na + 1;  
    up(a);  
  }  
}
```

Process B()

```
{  
  int item;  
  while (TRUE)  
  {  
    down(a);  
    nb = nb + 1;  
    up(b);  
  }  
}
```



■ Câu 10:

Một biến X được chia sẻ bởi hai 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 trị X có thể vượt quá 20. Cần sửa chữa đoạn chương trình trên như thế nào để bảo đảm X không vượt quá 20 ?

Câu 10: Đáp án

Đáp án :

```
Semaphore mutex = 1;
do
{
    down(mutex);
    X = X + 1;
    if ( X == 20) X = 0;
    up(mutex);
}while ( TRUE );
```

■ Câu 11 :

Xét hai tiến trình xử lý đoạn chương trình sau :

process P1 { A1 ; A2 }

process P2 { B1 ; B2 }

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

Câu 11: Đáp án

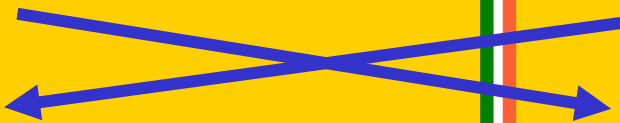
Đáp án : semaphore ab = 0 ; ba = 0 ;

Process A()

```
{  
  A1;  
  up(ba);  
  down(ab);  
  A2;  
}
```

Process B()

```
{  
  B1;  
  up(ab);  
  down(ba);  
  B2;  
}
```



■ Câu 12:

Tổng quát hoá câu hỏi 8) cho các tiến trình xử lý đoạn chương trình sau :

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

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

Đồng bộ hoá hoạt động của hai tiến trình này sao cho cả với k bất kỳ ($2 \leq k \leq 100$), 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.

Câu 12: Đáp án

Đáp án : semaphore ab = 1 ; ba = 1 ;

Process A()

```
{  
  for ( i = 1; i<=100; i++)  
  {  
    down(ab);  
    Ai;  
    up(ba);  
  }  
}
```

Process B()

```
{  
  for ( i = 1; i<=100; i++)  
  {  
    down(ba);  
    Bi;  
    up(ab);  
  }  
}
```

■ Câu 13:

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

$$w := x1 * x2$$

$$v := x3 * x4$$

$$y := v * x5$$

$$z := v * x6$$

$$y := w * y$$

$$z := w * z$$

$$\text{ans} := y + z$$

Đáp án :

```
process P1
{ w := x1 * x2;
  up(s15);
  up(s16);
}
process P2
{ v := x3 * x4;
  up(s23);
  up(s24);
}
process P3
{
  down(s23);
  y := v * x5;
  up(s35);
}
process P4
{
  down(s24);
  z := v * x;
  up(s46);
}
process P5
{
  down(s15);
  down(s35);
  y := w * y;
  up(s57);
}
process P6
{
  down(s16);
  down(s46);
  z := w * z;
  up(s67);
}
process P7
{
  down(s57);
  down(s67);
  ans := y + z;
}
```

```
process P5
{
  down(s15);
  down(s35);
  y := w * y;
  up(s57);
}
process P6
{
  down(s16);
  down(s46);
  z := w * z;
  up(s67);
}
process P7
{
  down(s57);
  down(s67);
  ans := y + z;
}
```



Câu 14:

Cho mảng sau: `int x[20];`

Sử dụng cơ chế đồng bộ hoá là semaphore để viết code cho 3 threads B,C,D cùng thực hiện đồng thời các thao tác trên mảng x thoả mãn các yêu cầu sau:

- a.** B tính tổng giá trị các phần tử mảng x có chỉ số chẵn.
- b.** C tính tổng giá trị các phần tử mảng x có chỉ số lẻ.
- c.** D tính tổng giá trị tất cả các phần tử của mảng x, dựa trên kết quả trả về của B và C.
- d.** Các threads được khởi động cùng lúc.
- e.** Các threads kết thúc khi xong công việc của mình, không cần chờ lẫn nhau.
- f.** Phải khai thác tối đa khả năng xử lý song song, chia sẻ tài nguyên dùng chung của các threads.

Câu 14 Cách 1:

Đáp án : semaphore overB = 0, overC = 0 ;
Integer sumB, sumC = 0;

Process B()

```
{  
    for(i=0;i<9,i++){  
        sumB +=x[2*i];  
    }  
    up(overB);  
}
```

Process C()

```
{  
    for(i=0;i<9,i++){  
        sumC +=x[2*i+1];  
    }  
    up(overC);  
}
```

Process D()

```
{  
    down(overB);  
    down(overC);  
    sum=sumB+sumC;  
}
```

Câu 14 Cách 2:

Đáp án : semaphore over=0;
Integer sumB, sumC = 0;

Process B()

```
{  
    for(i=0;i<9,i++){  
        sumB +=x[2*i];  
    }  
    up(over);  
}
```

Process C()

```
{  
    for(i=0;i<9,i++){  
        sumC +=x[2*i+1];  
    }  
    up(over);  
}
```

Process D()

```
{  
    down(over);  
    down(over);  
    sum=sumB+sumC;  
}
```

Câu 14 Cách 3:

```
Đáp án : semaphore mutex=1;  
Integer sum=0;
```

Process B()

```
{  
    for(i=0;i<9,i++){  
        down(mutex);  
        sum +=x[2*i];  
        up(mutex);  
    }  
}
```

Process C()

```
{  
    for(i=0;i<9,i++){  
        down(mutex);  
        sum +=x[2*i+1];  
        up(mutex);  
    }  
}
```

Process D()

```
{  
}
```


Câu 14 Cách 3:

Đáp án : semaphore mutex=1, over=0;
Integer sum=0;

Process B()

```
{  
    for(i=0;i<9,i++) {  
        down(mutex);  
        sum +=x[2*i];  
        up(mutex);  
    }  
    up(over);  
}
```

Process C()

```
{  
    for(i=0;i<9,i++){  
        down(mutex);  
        sum +=x[2*i+1];  
        up(mutex);  
    }  
    up(over);  
}
```

Process D()

```
{  
    down(over);  
    down(over);  
}
```



Câu 15:

Một hãng sản xuất xe ô tô có các bộ phận hoạt động song song:

+ Bộ phận sản xuất khung xe:

```
MakeChassis() { //Sản xuất ra một khung xe  
    Produce_chassis();  
}
```

+ Bộ phận sản xuất bánh xe:

```
MakeTire() { //Sản xuất ra một bánh xe  
    Produce_tire();  
}
```

+ Bộ phận lắp ráp: Sau khi có được 1 khung xe và 4 bánh xe thì tiến hành lắp ráp 4 bánh xe này vào khung xe:

```
Assemble(){ //Gắn 4 bánh xe vào khung xe  
    Put_4_tires_to_chassis();  
}
```

Hãy đồng bộ hoạt động của các bộ phận trên thoả các nguyên tắc sau:

Tại mỗi thời điểm chỉ cho phép sản xuất ra 1 khung xe. Cần chờ có đủ 4 bánh xe để gắn vào khung xe hiện tại này trước khi sản xuất ra một khung xe mới.

Câu 15: Đáp án

```
Semaphore chassis=0, tire=0, wait=1;
```

```
Make_Chassis()
```

```
{  
  down(wait);  
  produce_chas();  
  up(chassis);  
}
```

```
Make_Tire()
```

```
{  
  produce_Tire()  
  up(tire);  
}
```

```
Assemble()
```

```
{  
  down(tire);  
  down(chassis);  
  down(chassis);  
  down(chassis);  
  Put_4_tires_to_chassis();  
  up(wait);  
}
```

Câu 16:

Trong giai đoạn thử nghiệm, hầm đường bộ qua đèo Hải Vân chỉ cho phép các phương tiện lưu thông qua hầm với số lượng hạn chế và với những điều kiện nghiêm ngặt. Khi một phương tiện đến đầu hầm sẽ gọi hàm `EnterTunnel(direction)` để kiểm tra điều kiện vào hầm. Khi đã qua hầm sẽ gọi hàm `ExitTunnel(direction)` để báo hiệu kết thúc và rời hầm.

Giả sử hoạt động của mỗi một phương tiện được mô tả bằng tiến trình `Car()` sau đây:

```
Car(direction)                                     //Direction xác định hướng
di chuyển của phương tiện
{
    RuntoTunnel();                                  //Phương tiện di chuyển về phía hầm
    EnterTunnel(direction);                         //Đi vào hầm theo hướng direction
    PassTunnel();                                    //Qua hầm
    ExitTunnel(direction);                          //Rời khỏi hầm theo hướng
direction.
}
```

Hãy viết lại các hàm `EnterTunnel(direction)` và `ExitTunnel(direction)` kiểm soát giao thông qua hầm sao cho:

- Tại mỗi thời điểm chỉ cho phép tối đa 3 phương tiện lưu thông qua hầm theo hướng bất kỳ.
- Tại mỗi thời điểm chỉ cho phép tối đa 3 phương tiện lưu thông cùng hướng qua hầm.

Câu 16.a: Đáp án

```
Semaphore max=3;
```

```
EnterTunnel(direction)
```

```
{  
    down(max);  
    ...  
}
```

```
ExitTunnel()
```

```
{  
    ...  
    up(max);  
}
```