



Lê Quang Khải's Blog

Giải bài tập chương 5–Hệ điều hành

NOVEMBER 17, 2010 / SCOTT LEE

Đây là phần giải bài tập chương 5 môn hệ điều hành do mình tự làm (thầy Khánh không giải. Mình sẽ chỉ viết bài giải từ bài 5 trở đi thôi. Từ bài 1 đến bài 4 không có gì để nói hết. Các bạn nào biết đáp án rồi mà phát hiện mình sai chỗ nào comment dùm nhen. Thanks các bạn nhiều.

Bài 5: bài giải tổng quát như sau (để bài phải sửa lại thành $nb \leq na \leq nb + 10$ mới có thể thực hiện được):

```
// Share data
Semaphore b, max;
b.value = 0;
max.value = 9; // if we set max = 10, then
                // at some point, a will be greater than b+10

// Process A
while (true)
{
    wait(max); // Check whether it has reached the max value or not
               // (na<=nb+10)
```

```

        na++; // do work
        signal(b); // increase b.value one unit. This makes process A
                    // always execute before process B,
                    // so nb <= na every time
    } // Process B
while (true)
{
    wait(b);
    nb++; // do work
    signal(max); // release one value of max's value, so that it make A
                  //be able to increase its
                  // value, and also ensure na<=nb+10
}

```

Còn dưới đây là code demo sử dụng C++:

```

1:  #include "stdio.h"
2:  #include "windows.h"
3:  #include "conio.h"
4:  #include "tchar.h"
5:  HANDLE maxSemaphore, mutex;
6:
7:  DWORD WINAPI ProcessA (LPVOID num)
8:  {
9:      int na=0, i=0; 10:      while (i<100)
11:      {
12:          if (WaitForSingleObject(maxSemaphore, INFINITE)==0)
13:          {
14:              na++;
15:              ReleaseSemaphore(mutex, 1, NULL);
16:              //printf ("Release mutex error!\n");
17:              printf ("na= %dn", na);
18:              i++;
19:          }
20:          else
21:              printf ("Wait for maxSemaphoren");
22:      }
23:      return 0;
24:  }
25:
26:  DWORD WINAPI ProcessB (LPVOID num)
27:  {
28:      int nb=0, i=0;
29:      while (i<100)
30:      {
31:          if (WaitForSingleObject(mutex, INFINITE)==0)
32:          {
33:              ReleaseSemaphore(maxSemaphore, 1, NULL);
34:              nb++;

```

```

35:         printf ("nb= %dn", nb);
36:         i++;
37:     }
38:     else
39:         printf ("Wait for mutexn");
40: }
41: return 0;
42: }
43:
44:
45: int main()
46: {
47:     HANDLE hThread[2];
48:     DWORD threadID[2];
49:     mutex = CreateSemaphore (NULL, 0, 1, NULL);
50:     if (mutex==NULL)
51:     {
52:         printf ("Create mutex error!\n");
53:         return -1;
54:     }
55:     maxSemaphore = CreateSemaphore(NULL, 9, 9, NULL);
56:     if (maxSemaphore==NULL)
57:     {
58:         printf ("Create maxSemaphore error!\n");
59:         return -1;
60:     }
61:     hThread[0] = CreateThread(NULL, // default security attributes
62:         0, // default stack size
63:         (LPTHREAD_START_ROUTINE)ProcessA, // procedure to execute
64:         NULL, // no argument
65:         0, // default creation flags
66:         &threadID[0]); // thread identifier
67:     if (hThread[0]==NULL)
68:     {
69:         printf ("Create thread A error: %dn", GetLastError());
70:         return -2;
71:     }
72:
73:     hThread[1] = CreateThread(NULL, // default security attributes
74:         0, // default stack size
75:         (LPTHREAD_START_ROUTINE)ProcessB, // procedure to execute
76:         NULL, // no argument
77:         0, // default creation flags
78:         &threadID[1]); // thread identifier
79:     if (hThread[1]==NULL)
80:     {
81:         printf ("Create thread B error: %dn", GetLastError());
82:         return -2;
83:     }
84:     WaitForMultipleObjects(2,hThread,FALSE,INFINITE);

```

```

85:     for (int i=0; i<2; i++)
86:     {
87:         CloseHandle(hThread[i]);
88:     }
89:     CloseHandle(maxSemaphore);
90:     CloseHandle(mutex);
91:     return 0;
92: }

```

Bài 6: Dễ dàng chứng minh với đoạn code đó thì X có thể vượt quá 20. Để đảm bảo X không vượt quá 20, ta chỉ cần sử dụng một trong các giải thuật định thời đã học.

Bài 7:

```

// share data
Semaphore s1, s2;
s1=0; s2=0;
// do work
A1();
signal(s1);
B1();
signal(s2);
wait(s2);
A2();
wait(s1);
B2();

```

Bài 8: With every k ($k \geq 1$), A_k just starts if only B_{k-1} has finished and B_k just starts if only A_{k-1} has finished.

```

// Share data
Semaphore S[100];
for (int i=0; i<100; i++)
    S[i].value = 0;
// Process A
for (int i=0; i<100; i+=2)
{
    execute Ai
    signal(S[i]);
    wait(S[i]+1);
    execute Ai+1
}

```

```

}
// Process B
for (int j=0; j<100; j+=2)
{
    execute Bj
    signal(S[j]+1);
    wait(S[j]);
    execute B[j]+1;
}

```

Bài 9: Sử dụng Semaphore để viết lại chương trình theo mô hình xử lý đồng hành.

Giải quyết đề bài như sau: (ứng với mỗi số được xem như một process hoặc thread).

```

w := x1 * x2 (1) // 1 starts before 5 and 6
v := x3 * x4 (2) // 2 starts before 3 and 4
y := v * x5 (3) // 3 starts after 2 and before 5
z := v * x6 (4) // 4 starts after 2 and before 6
y := w * y (5) // 5 starts after 1, 3 and before 7
z := w * z (6) // 6 starts after 1, 4 and before 7
ans := y + z (7) // 7 starts after 5 and 6

```

Ta sẽ tạo ra các semaphore gồm: s15, s16, s23, s24, s35, s46, s57, s67. Initial value của tất cả các semaphore đều là 0. Giải thích việc đặt tên như sau: process nào thực thi trước sẽ có số đại diện đứng trước, process thực thi sau sẽ có số đại diện đứng sau, ví dụ process 1 thực thi trước process 5 và process 6 nên sẽ có s15, s16.

```

// Process 1
w = x1 * x2;
signal (s15);
signal (s16);
// Process 2
v = x3 * x4;
signal(s23);
signal(s24);
// Process 3
wait(s23);
y = v * x5;
signal(s35);
// Process 4
wait(s24);

```

```
z = v * x6;  
signal(s46);  
// Process 5  
wait(s15);  
wait(s35);  
y = w * y;  
signal(s57);  
// Process 6  
wait(s16);  
wait(s46);  
z = w * z;  
signal(s67);  
// Process 7  
wait(s57);  
wait(s67);  
ans = y + z;
```

Ok, bài giải của mình. Các bạn xem cho ý kiến đúng sai thế nào nha. Thanks a lot .

Advertisements

Chia sẻ



Tweet



Share



Print



More



Like

Be the first to like this.

Các giai đoạn đầu tư, phát triển của một startup
In "Learning"

Câu hỏi ôn tập hệ điều hành - thầy Lương Ngọc Khánh
In "Learning"

Process Management - Part 1: Processes (Tiến trình)
In "Learning"

Các giai đoạn đầu tư, phát triển của một startup

Câu hỏi ôn tập hệ điều hành - thầy Lương Ngọc Khánh

Process Management - Part 1: Processes (Tiến trình)

Categories: [Learning](#), [Operating System](#)Tags: [Bài tập](#), [hệ điều hành](#)

Cơ bản về XAML-Part 2

Báo cáo tiểu luận-Hệ điều hành

3 Comments

**Chestnut**

DECEMBER 21, 2010 — 1:35 PM

REPLY

Thanks ban nhieu nha, tuyet voi!

Bạn giải thử bài 1 chương 5 xem, mình thấy code nó sai sai hay sao đó'!

Thanks bạn nhiều!

**Scott Lee** (Post author)

DECEMBER 21, 2010 — 2:45 PM

REPLY

Bài này có một chỗ lập lờ đó là không biết cái turn nó gán từ khi nào. Nếu bỏ qua và xem như biến giá trị của biến turn được gán vào thời điểm nào đó hợp lí (tùy đề bài) thì nó thỏa cả 4 tính chất. Nhưng ngược lại giả sử giá trị của biến turn được gán ngay sau `flag[i]=true` với giá trị là `turn:=i`; thì lúc này chương trình sẽ không thỏa mutual exclusion. Nói chung là do đề bài có vấn đề :))

**Trường**

NOVEMBER 15, 2012 — 6:25 PM

REPLY

cho mình hỏi ở bài 9 mấy cái biến x dùng để làm j thê z ??

0 Pingbacks

Leave a Reply

CATEGORIES

IT Knowledge (11)

Learning (32)

C# (15)

WPF (9)

Operating System (11)

Vài lời linh tinh (2)

FAVORITE

About me

Code Project

Facebook

Nhân Duyên's Blog

LƯU LẠI EMAIL

Để lại email tại đây, chúng tôi sẽ gửi thông báo cho bạn khi có bài viết mới

Join 402 other followers

Enter your email address

Đăng kí