

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**







**HỆ ĐIỀU HÀNH**

**LỚP: IT007.O18**

**Tên: Lê Minh Nhựt**

**MSSV: 22521060**



**THỰC HÀNH LAB 5**

1. Hiện thực hóa mô hình trong ví dụ **5.3.1.2**, tuy nhiên thay bằng điều kiện sau: **sells <= products <= sells + [2 số cuối của MSSV + 10]**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

sem\_t sem;

int sells=0, products=0;

void\* PROCESSA()

{

    while(1)

    {

        sem\_wait(&sem);

        sells++;

        printf("sells = %d\n", sells);

        sleep(2);

    }

}

void\* PROCESSB()

{

    while(1)

    {

        if (products <= sells + 60 + 10) {

            products++;

            printf("products = %d\n", products);

            sem\_post(&sem);

            sleep(1);

        }

    }

}

void main()

{

    sem\_init(&sem,0,0);

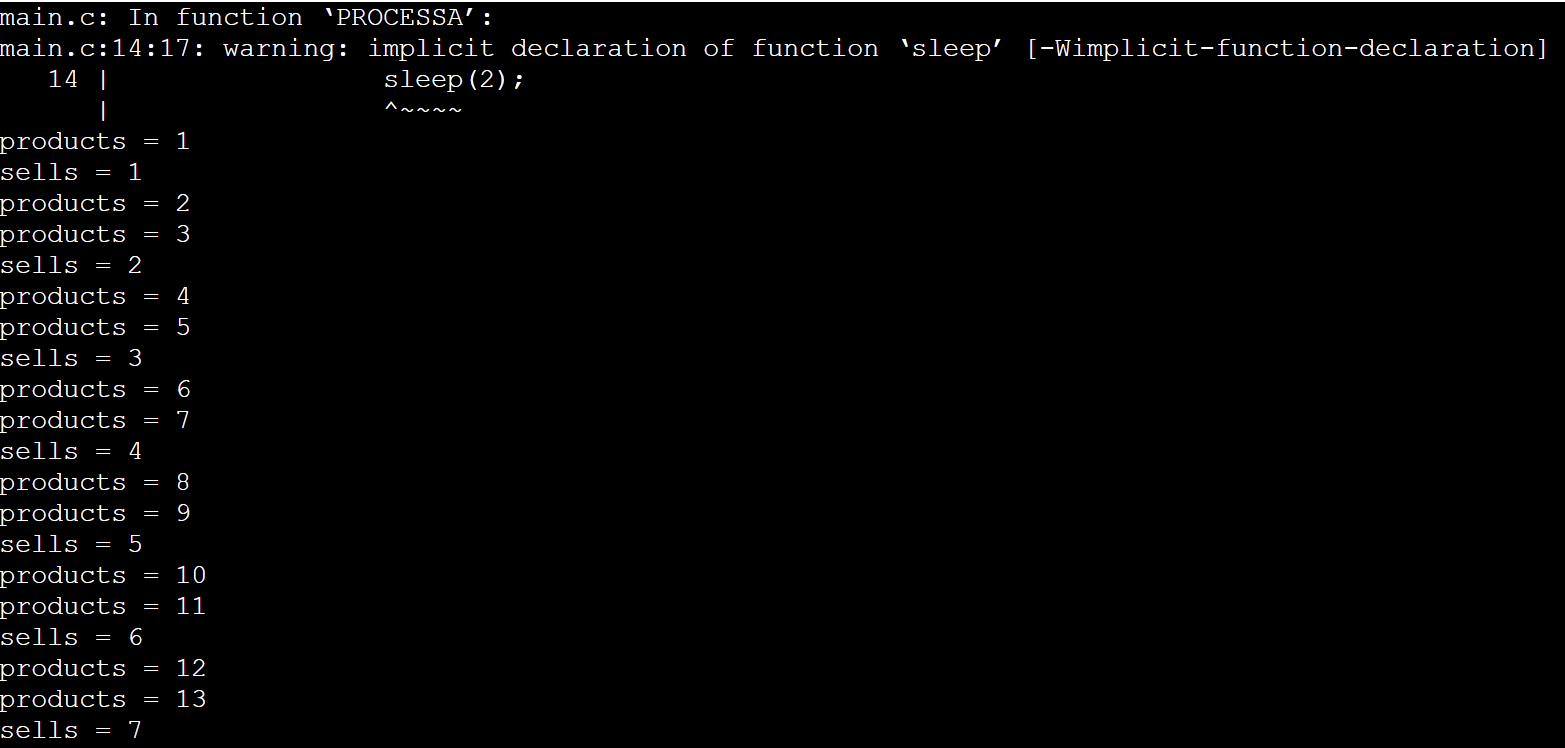
    pthread\_t th1, th2;

    pthread\_create(&th1, NULL, &PROCESSA, NULL);

    pthread\_create(&th2, NULL, &PROCESSB, NULL);

    while(1);

}



**2.** Cho một mảng a được khai báo như một mảng số nguyên có thể chứa n phần tử, a được khai báo như một biến toàn cục. Viết chương trình bao gồm 2 thread chạy song song:

Một thread làm nhiệm vụ sinh ra một số nguyên ngẫu nhiên sau đó bỏ vào a. Sau đó đếm và xuất ra số phần tử của a có được ngay sau khi thêm vào.

Thread còn lại lấy ra một phần tử trong a (phần tử bất kỳ, phụ thuộc vào người lập trình). Sau đó đếm và xuất ra số phần tử của a có được ngay sau khi lấy ra, nếu không có phần tử nào trong a thì xuất ra màn hình “Nothing in array a”.

Chạy thử và tìm ra lỗi khi chạy chương trình trên khi chưa được đồng bộ. Thực hiện đồng bộ hóa với semaphore.

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<pthread.h>

#include<semaphore.h>

sem\_t sem1, sem2;

int n;

int i = 0;

static int dem = 0;

int a[100000];

void\* PROCESS1()

{

    while (1)

    {

        if (dem < n) {

            a[i++] = rand() % (n - 1);

            dem++;

            printf("\n[PUSH] Number of elements in array a: %2d", dem);

        }

        int time\_sleep = rand() % 2 + 1;

        sleep(time\_sleep);

        sem\_post(&sem1);

    }

}

void\* PROCESS2()

{

    int j, b;

    while (1)

    {

        sem\_wait(&sem1);

        //if (dem <= n) {

            if (dem == 0)

            {

                printf("\n[POP] Nothing in array a");

            }

            else

            {

                dem--;

                b = a[0];

                for (j = 0; j < dem; j++)

                {

                    a[j] = a[j + 1];

                }

                printf("\n[POP] Number of elements in array a: %2d", dem);

            }

        //}

        int time\_sleep = rand() % 2 + 1;

        sleep(time\_sleep);

    }

}

void main()

{

    sem\_init(&sem1, 1, 0);

    sem\_init(&sem2, 0, 0);

    printf("\nEnter n: ");

    scanf("%d",&n);

    pthread\_t th1, th2;

    pthread\_create(&th1, NULL, PROCESS1, NULL);

    pthread\_create(&th2, NULL, PROCESS2, NULL);

    while(1);

}

**3.**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

int x = 0;

void\* A()

{

    while (1)

    {

        x = x + 1;

        if (x == 20)

        {

            x = 0;

        }

        printf("PA: x = %d\n", x);

    }

}

void\* B()

{

    while (1)

    {

        x = x + 1;

        if (x == 20)

        {

            x = 0;

        }

        printf("PB: x = %d\n", x);

    }

    sleep(1);

}

void main()

{

    pthread\_t th1, th2;

    pthread\_create(&th1, NULL, &A, NULL);

    pthread\_create(&th2, NULL, &B, NULL);;

    while (1);

}

1. Đồng bộ với mutex để sửa lỗi bất hợp lý trong kết quả của mô hình Bài 3.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

sem\_t sem\_1, sem\_2;

int x = 0;

pthread\_mutex\_t mutex;

void\* PROCESS1()

{

  while (1)

  {

    pthread\_mutex\_lock(&mutex);

    x++;

    if (x == 20)

    {

      x = 0;

    }

    printf("PA: x = %d\n", x);

    pthread\_mutex\_unlock(&mutex);

  }

}

void\* PROCESS2()

{

  while (1)

  {

    pthread\_mutex\_lock(&mutex);

    x++;

    if (x == 20)

    {

      x = 0;

    }

    printf("PB: x = %d\n", x);

    pthread\_mutex\_unlock(&mutex);

  }

}

void main()

{

  pthread\_mutex\_init(&mutex, NULL);

  pthread\_t th1, th2;

  pthread\_create(&th1, NULL, &PROCESS1, NULL);

  pthread\_create(&th2, NULL, &PROCESS2, NULL);

  while (1);

}

**Bài tập:**

Biến ans được tính từ các biến x1, x2, x3, x4, x5, x6 như sau:

w = x1 \* x2; (a)

v = x3 \* x4; (b)

y = v \* x5; (c)

z = v \* x6; (d)

y = w \* y; (e)

z = w \* z; (f)

ans = y + z; (g)

Giả sử các lệnh từ (a) → (g) nằm trên các thread chạy song song với nhau. Hãy lập trình mô phỏng và đồng bộ trên C trong hệ điều hành Linux theo thứ tự sau:(c), (d) chỉ được thực hiện sau khi v được tính, (e) chỉ được thực hiện sau khi w và y được tính (g) chỉ được thực hiện sau khi y và z được tính.

**\*Code:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

sem\_t p1\_5, p1\_6, p2\_3, p2\_4, p3\_5, p4\_6, p5\_7, p6\_7;

int x1 = 1;

int x2 = 2;

int x3 = 3;

int x4 = 4;

int x5 = 5;

int x6 = 6;

int w, v, z, y, x;

int ans = 0;

void\* PROCESS1()

{

    w = x1 \* x2;

    printf("w = %d\n", w);

    sem\_post(&p1\_5);

    sem\_post(&p1\_6);

    sleep(1);

}

void\* PROCESS2()

{

    v = x3 \* x4;

    printf("v = %d\n", v);

    sem\_post(&p2\_3);

    sem\_post(&p2\_4);

    sleep(1);

}

void\* PROCESS3()

{

    sem\_wait(&p2\_3);

    printf("y = %d\n", y);

    y = v \* x5;

    sem\_post(&p3\_5);

    sleep(1);

}

void \*PROCESS4()

{

    sem\_wait(&p2\_4);

    printf("z = %d\n", z);

    z = v \* x6;

    sem\_post(&p4\_6);

    sleep(1);

}

void \*PROCESS5()

{

    sem\_wait(&p1\_5);

    sem\_wait(&p3\_5);

    y = w \* y;

    printf("y = %d\n", y);

    sem\_post(&p5\_7);

    sleep(1);

}

void \*PROCESS6()

{

    sem\_wait(&p1\_6);

    sem\_wait(&p4\_6);

    z = w \* z;

    printf("z = %d\n", z);

    sem\_post(&p6\_7);

    sleep(1);

}

void\* PROCESS7()

{

    sem\_wait(&p5\_7);

    sem\_wait(&p6\_7);

    ans = y + z;

    printf("ans = %d\n", ans);

    sleep(1);

}

void main()

{

    sem\_init(&p1\_5, 0, 1);

    sem\_init(&p1\_6, 0, 0);

    sem\_init(&p2\_3, 0, 0);

    sem\_init(&p2\_4, 0, 0);

    sem\_init(&p3\_5, 0, 0);

    sem\_init(&p4\_6, 0, 0);

    sem\_init(&p5\_7, 0, 0);

    sem\_init(&p6\_7, 0, 0);

    pthread\_t th1, th2, th3, th4, th5, th6, th7;

    pthread\_create(&th1, NULL, &PROCESS1, NULL);

    pthread\_create(&th2, NULL, &PROCESS2, NULL);

    pthread\_create(&th3, NULL, &PROCESS3, NULL);

    pthread\_create(&th4, NULL, &PROCESS4, NULL);

    pthread\_create(&th5, NULL, &PROCESS5, NULL);

    pthread\_create(&th6, NULL, &PROCESS6, NULL);

    pthread\_create(&th7, NULL, &PROCESS7, NULL);

    while (1);

}

**\*Kết quả chương trình:**

