ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN

HỆ ĐIỀU HÀNH
IT007.O21.LT

BÁO CÁO
BÁO CÁO THỰC HÀNH LAB 6

Sinh viên thực hiện:

Trần Duy Tân - 22550020

Giảng viên:

ThS. Đỗ Trí Nhựt

Thành phố Hồ Chí Minh, năm 2024

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

Bài làm:
Để thao tác nhanh ví dụ bên dưới các biến được gắn là: x1=1, x2=2, x3=3, x4=4, x5=5, x6=6
Theo như đề bài chương trình có thể chạy được nhiều trường hợp khác nhau, sau đây là 4 kết quả sẻ xảy ra:

Kết quả 1:
Trường hợp thread g sẻ chạy ngay lập tức khi z và y đã được tính:

```
Thread a was running!
Thread b was running!
Thread c was running!
Thread d was running!
Thread g was running!
ans = 132
Thread f was running!
Thread e was running!


...Program finished with exit code 0
Press ENTER to exit console.
```

Kết quả 2:
Trường hợp thread g sẻ sau z, y đã được tính và y được tính lại bởi thread e

```
Thread a was running!
Thread b was running!
Thread c was running!
Thread e was running!
Thread d was running!
Thread g was running!
ans = 192
Thread f was running!


...Program finished with exit code 0
Press ENTER to exit console.
```

Kết quả 3:
Trường hợp thread g sẻ sau z, y đã được tính và z được tính lại bởi thread f

```
input
Thread b was running!
Thread d was running!
Thread a was running!
Thread c was running!
Thread f was running!
Thread g was running!
ans = 204
Thread e was running!


...Program finished with exit code 0
Press ENTER to exit console.
```

Kết quả 4:
Trường hợp thread g sẻ chạy sau tất cả các thread:

```
input
Thread a was running!
Thread b was running!
Thread c was running!
Thread e was running!
Thread d was running!
Thread f was running!
Thread g was running!
ans = 264


...Program finished with exit code 0
Press ENTER to exit console.
```

Code demo

```c
#include <stdio.h>
#include <pthread.h>

#define NUM_THREADS 7

int x1 = 1, x2 = 2, x3 = 3, x4 = 4, x5 = 5, x6 = 6;
int w = 0, v = 0, y = 0, z = 0;
int ans;

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;

void *thread_a(void *arg) {
    pthread_mutex_lock(&mutex);
    printf("Thread a was running!\n");
    w = x1 * x2;
    pthread_cond_broadcast(&cond);
```

```
    pthread_mutex_unlock(&mutex);
    return NULL;
}

void *thread_b(void *arg) {
    pthread_mutex_lock(&mutex);
    printf("Thread b was running!\n");
    v = x3 * x4;
    pthread_cond_broadcast(&cond);
    pthread_mutex_unlock(&mutex);
    return NULL;
}

void *thread_c(void *arg) {
    pthread_mutex_lock(&mutex);
    while (v == 0) pthread_cond_wait(&cond, &mutex);
    printf("Thread c was running!\n");
    y = v * x5;
    pthread_cond_broadcast(&cond);
    pthread_mutex_unlock(&mutex);
    return NULL;
}

void *thread_d(void *arg) {
    pthread_mutex_lock(&mutex);
    while (v == 0) pthread_cond_wait(&cond, &mutex);
    printf("Thread d was running!\n");
    z = v * x6;
    pthread_cond_broadcast(&cond);
    pthread_mutex_unlock(&mutex);
    return NULL;
}

void *thread_e(void *arg) {
    pthread_mutex_lock(&mutex);
    while (w == 0 || y == 0) pthread_cond_wait(&cond, &mutex);
    printf("Thread e was running!\n");
    y = w * y;
    pthread_cond_broadcast(&cond);
    pthread_mutex_unlock(&mutex);
    return NULL;
}

void *thread_f(void *arg) {
    pthread_mutex_lock(&mutex);
    while (w == 0 || z == 0) pthread_cond_wait(&cond, &mutex);
    printf("Thread f was running!\n");
    z = w * z;
```

```c
    pthread_cond_broadcast(&cond);
    pthread_mutex_unlock(&mutex);
    return NULL;
}

void *thread_g(void *arg) {
    pthread_mutex_lock(&mutex);
    while (z == 0 || y == 0) pthread_cond_wait(&cond, &mutex);
    printf("Thread g was running!\n");
    ans = y + z;
    pthread_mutex_unlock(&mutex);
    printf("ans = %d\n", ans);
    return NULL;
}

int main() {
    int i;
    pthread_t threads[NUM_THREADS];

    // Create threads
    pthread_create(&threads[0], NULL, thread_a, NULL);
    pthread_create(&threads[1], NULL, thread_b, NULL);
    pthread_create(&threads[2], NULL, thread_c, NULL);
    pthread_create(&threads[3], NULL, thread_d, NULL);
    pthread_create(&threads[4], NULL, thread_e, NULL);
    pthread_create(&threads[5], NULL, thread_f, NULL);
    pthread_create(&threads[6], NULL, thread_g, NULL);

    // Wait for threads to finish
    for (i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    return 0;
}
```