

Họ và tên: Nguyễn Thị Minh Châu – MSSV: 21520645

Lớp: IT007.N25

## BÁO CÁO THỰC HÀNH LAB 5 HỆ ĐIỀU HÀNH

1.

Hiện thực hóa mô hình trong ví dụ 5.3.1.2, tuy nhiên thay bằng điều kiện sau:  $sells \leq products \leq sells + [2 \text{ số cuối của MSSV} + 10]$

```
1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <semaphore.h>
5 #include <pthread.h>
6 sem_t sem1, sem2;
7 int sells = 0, products;
8
9 void *processA()
10 {
11     while (1)
12     {
13         sem_wait(&sem1);
14         sells++;
15         sem_post(&sem2);
16         printf("sells: sells = %d, products = %d, %d \n", sells, products, products - sells);
17     }
18 }
19
20 void *processB()
21 {
22     while (1)
23     {
24         sem_wait(&sem2);
25         products++;
26         sem_post(&sem1);
27         printf("products: sells = %d, products = %d, %d \n", sells, products, products - sells);
28     }
29 }
30 int main()
31 {
32     sem_init(&sem1, 0, 0);
33     sem_init(&sem2, 0, 55);
34     pthread_t pA, pB;
35     pthread_create(&pA, NULL, processA, NULL);
36     pthread_create(&pB, NULL, processB, NULL);
37     pthread_join(pA, NULL);
38     pthread_join(pB, NULL);
39     return 0;
40 }
```

Đây là chương trình

old configure file with --disable-gvfs metadata.

zeri@LAPTOP-HQ4PM7S6:~\$ ./bail

```
products: sells = 0, products = 1, 1
products: sells = 1, products = 2, 1
products: sells = 1, products = 3, 2
products: sells = 1, products = 4, 3
products: sells = 1, products = 5, 4
products: sells = 1, products = 6, 5
products: sells = 1, products = 7, 6
products: sells = 1, products = 8, 7
products: sells = 1, products = 9, 8
products: sells = 1, products = 10, 9
products: sells = 1, products = 11, 10
products: sells = 1, products = 12, 11
sells: sells = 1, products = 1, 0
sells: sells = 2, products = 13, 11
sells: sells = 3, products = 13, 10
sells: sells = 4, products = 13, 9
sells: sells = 5, products = 13, 8
sells: sells = 6, products = 13, 7
sells: sells = 7, products = 13, 6
sells: sells = 8, products = 13, 5
sells: sells = 9, products = 13, 4
sells: sells = 10, products = 13, 3
sells: sells = 11, products = 13, 2
sells: sells = 12, products = 13, 1
sells: sells = 13, products = 13, 0
products: sells = 1, products = 13, 12
products: sells = 13, products = 14, 1
products: sells = 13, products = 15, 2
products: sells = 13, products = 16, 3
products: sells = 13, products = 17, 4
products: sells = 13, products = 18, 5
```

```
products: sells = 13, products = 20, 7
products: sells = 13, products = 21, 8
products: sells = 13, products = 22, 9
products: sells = 13, products = 23, 10
products: sells = 14, products = 24, 10
products: sells = 14, products = 25, 11
products: sells = 14, products = 26, 12
products: sells = 14, products = 27, 13
products: sells = 14, products = 28, 14
products: sells = 14, products = 29, 15
products: sells = 14, products = 30, 16
products: sells = 14, products = 31, 17
products: sells = 14, products = 32, 18
products: sells = 14, products = 33, 19
products: sells = 14, products = 34, 20
products: sells = 14, products = 35, 21
products: sells = 14, products = 36, 22
products: sells = 14, products = 37, 23
products: sells = 14, products = 38, 24
products: sells = 14, products = 39, 25
products: sells = 14, products = 40, 26
products: sells = 14, products = 41, 27
products: sells = 14, products = 42, 28
products: sells = 14, products = 43, 29
products: sells = 14, products = 44, 30
products: sells = 14, products = 45, 31
products: sells = 14, products = 46, 32
products: sells = 14, products = 47, 33
products: sells = 14, products = 48, 34
products: sells = 14, products = 49, 35
products: sells = 14, products = 50, 36
products: sells = 14, products = 51, 37
products: sells = 14, products = 52, 38
products: sells = 14, products = 53, 39
products: sells = 14, products = 54, 40
```

```
products: sells = 14, products = 56, 42
products: sells = 14, products = 57, 43
products: sells = 14, products = 58, 44
products: sells = 14, products = 59, 45
products: sells = 14, products = 60, 46
products: sells = 14, products = 61, 47
products: sells = 14, products = 62, 48
products: sells = 14, products = 63, 49
products: sells = 14, products = 64, 50
products: sells = 14, products = 65, 51
products: sells = 14, products = 66, 52
products: sells = 14, products = 67, 53
products: sells = 14, products = 68, 54
products: sells = 14, products = 69, 55
```

Kết quả chạy chương trình

- **Giải thích:** sem1 đóng vai trò là điều kiện ( $\text{sells} \leq \text{products}$ ), sem2 đóng vai trò là điều kiện ( $\text{products} \leq \text{sells} + 55$ ). Khi chương trình được nạp, giả sử ProcessA được nạp trước. Khi đó A sẽ bị khóa lại với bởi biến sem1 (khởi tạo = 0). Vì vậy ProcessB sẽ được chạy. Products và sem1 (bởi hàm sem\_post) sẽ được tăng lên cho đến khi sem2=0 (được giảm bởi hàm sem\_wait). Khi đó ProcessA sẽ được chạy. Sells và sem2 (bởi hàm sem\_post) sẽ tăng cho đến khi sem1=0 (được giảm bởi hàm sem\_wait). Hai tiến trình này sẽ được lặp đi lặp lại trong 1 vòng lặp vô hạn.

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.

Open          bai2.c    ~    Save

```
1 #include <stdio.h>
2 #include <semaphore.h>
3 #include <pthread.h>
4 #include <time.h>
5 #include <stdlib.h>
6
7 int *arr;
8 int currentSize = 0;
9 int capacitySize = 0;
10 int Put(int value)
11 {
12     arr[currentSize] = value;
13     return ++currentSize;
14 }
15
16 int NumberGenerator()
17 {
18     srand(time(NULL));
19     return rand();
20 }
21
22
23 int Take(int pos)
24 {
25     if (pos == 0)
26     {
27         for(int i = pos; i < currentSize - 1; i++)
28         {
29             arr[i] = arr[i + 1];
30         }
31         return --currentSize;
32     }
33     else
34     {
35         for (int i = pos - 1; i < currentSize - 1 ; i++)
36         {
37             arr[i] = arr[i + 1];
38         }
39         return --currentSize;
40     }
41 }
42
43 }
44
45 void *ThreadA(void *mess)
46 {
47     while(1)
48     {
49         printf("Mang sau khi them: %i\n", Put(NumberGenerator()));
50     }
51 }
52 }
53
```

```



50
57 void *ThreadB(void *mess)
58 {
59     while(1)
60     {
61         int pos = rand() % currentSize;
62         int newSize = Take(pos);
63
64         if (newSize == 0)
65         {
66             printf("Nothing in array a \n");
67         }
68         else
69         {
70             printf("So phan tu trong mang a: %i\n", newSize);
71         }
72     }
73 }
74
75 int main()
76 {
77     printf("Nhap kich thuoc cua mang: ");
78     scanf("%d", &capacitySize);
79     arr = (int*)malloc(capacitySize*sizeof(int));
80
81     pthread_t pA, pB;
82     pthread_create(&pA, NULL, &ThreadA, NULL);
83     pthread_create(&pB, NULL, &ThreadB, NULL);
84     while(1)
85     {
86
87         return 0;
88
89     }
90

```

Đây là chương trình khi chưa đồng bộ

```
zeri@LAPTOP-HQ4PM7S6:~$ gcc bai2.c -o bai2 -lpthread -lrt
zeri@LAPTOP-HQ4PM7S6:~$ ./bai2
Nhap kích thước của mảng: 1000
Mảng sau khi thêm: 1
Mảng sau khi thêm: 1
Nothing in array a
Số phần tử trong mảng a: 1
Nothing in array a
Mảng sau khi thêm: 2
Mảng sau khi thêm: 1
Mảng sau khi thêm: 2
Mảng sau khi thêm: 3
Mảng sau khi thêm: 4
Mảng sau khi thêm: 5
Mảng sau khi thêm: 6
Mảng sau khi thêm: 7
Mảng sau khi thêm: 8
Mảng sau khi thêm: 9
Mảng sau khi thêm: 10
Mảng sau khi thêm: 11
Mảng sau khi thêm: 12
Mảng sau khi thêm: 13
Mảng sau khi thêm: 14
Mảng sau khi thêm: 15
Mảng sau khi thêm: 16
Mảng sau khi thêm: 17
Mảng sau khi thêm: 18
Mảng sau khi thêm: 19
Mảng sau khi thêm: 20
Mảng sau khi thêm: 21
Mảng sau khi thêm: 22
Mảng sau khi thêm: 23
Mảng sau khi thêm: 24
Mảng sau khi thêm: 25
Mảng sau khi thêm: 26
```

Lỗi xuất hiện khi chạy chương trình chưa đồng bộ

Open    ▾        \*bai22.c    Save    

```
1 #include <stdio.h>
2 #include <semaphore.h>
3 #include <pthread.h>
4 #include <time.h>
5 #include <stdlib.h>
6
7 int n;
8 int i;
9 int *a;
10 sem_t sem1, sem2;
11
12 int Generator()
13 {
14     srand(time(NULL));
15     return rand();
16 }
17
18 int Them(int value)
19 {
20     a[i] = value;
21     return ++i;
22 }
23
24 int Lay(int pos)
25 {
26     if(pos == 0)
27     {
28         for(int j = pos; j < i - 1; j++)
29         {
30             a[j] = a[j + 1];
31         }
32         return --i;
33     }
34     else
35     {
36         for(int j = pos - 1; j < i - 1; j++)
37         {
38             a[j] = a[j+1];
39         }
40         return --i;
41     }
42 }
43
44 void *Thread1(void *mess)
45 {
46     while(1)
47     {
48         sem_wait(&sem2);
49         printf("Kich thuoc mang sau khi them: %i\n", Them(Generator()));
50         sem_post(&sem1);
51     }
52 }
```



```

54 void *Thread2(void *mess)
55 {
56     while(1)
57     {
58         sem_wait(&sem1);
59         int pos = Generator() % i;
60         int ni = Lay(pos);
61         if (ni == 0)
62         {
63             printf("Nothing in array a\n");
64         }
65         else
66         {
67             printf("Kich thuoc mang sau khi lay: %i\n", ni);
68         }
69     }
70     sem_post(&sem2);
71 }
72
73 }
74 }
75
76 int main()
77 {
78     printf("Nhap kich thuoc mang: ");
79     scanf("%d", &n);
80     a = (int *)malloc(n * sizeof(int));
81
82     sem_init(&sem2, 0, n);
83     sem_init(&sem1, 0, 0);
84     pthread_t pA, pB;
85     pthread_create(&pB, NULL, &Thread2, NULL);
86     pthread_create(&pA, NULL, &Thread1, NULL);
87     while(1) {}
88     return 0;
89 }
90 }
91

```

Đây là chương trình sau khi đồng bộ

```
zeri@LAPTOP-HQ4PM7S6: ~  
zeri@LAPTOP-HQ4PM7S6:~$ ./bai22  
Nhap kich thuoc mang: 10  
Kich thuoc mang sau khi them: 1  
Kich thuoc mang sau khi them: 2  
Kich thuoc mang sau khi them: 3  
Kich thuoc mang sau khi them: 4  
Kich thuoc mang sau khi them: 5  
Kich thuoc mang sau khi them: 6  
Kich thuoc mang sau khi them: 7  
Kich thuoc mang sau khi them: 8  
Kich thuoc mang sau khi them: 9  
Kich thuoc mang sau khi them: 10  
Kich thuoc mang sau khi lay: 9  
Kich thuoc mang sau khi lay: 8  
Kich thuoc mang sau khi lay: 8  
Kich thuoc mang sau khi lay: 7  
Kich thuoc mang sau khi lay: 6  
Kich thuoc mang sau khi lay: 5  
Kich thuoc mang sau khi lay: 4  
Kich thuoc mang sau khi lay: 3  
Kich thuoc mang sau khi lay: 2  
Kich thuoc mang sau khi lay: 1  
Kich thuoc mang sau khi them: 9  
Kich thuoc mang sau khi them: 2  
Kich thuoc mang sau khi them: 2  
Kich thuoc mang sau khi lay: 1  
Kich thuoc mang sau khi them: 3  
Kich thuoc mang sau khi them: 3  
Kich thuoc mang sau khi them: 4  
Kich thuoc mang sau khi them: 5  
Kich thuoc mang sau khi them: 6  
Kich thuoc mang sau khi them: 7  
Kich thuoc mang sau khi them: 8  
Kich thuoc mang sau khi them: 9  
Kich thuoc mang sau khi lay: 2
```

Kết quả sau khi chạy chương trình đã đồng bộ

3.

3. Cho 2 process A và B chạy song song như sau:

int x = 0;	
<b>PROCESS A</b>	<b>PROCESS B</b>
processA() {	processB() {

12

<pre>while(1){     x = x + 1;     if (x == 20)         x = 0;     print(x); }</pre>	<pre>while(1){     x = x + 1;     if (x == 20)         x = 0;     print(x); }</pre>
---	---

Hiện thực mô hình trên C trong hệ điều hành Linux và  
nhận xét kết quả.

```
Open  ▾  [icon]  *bai3.c
~

1 #include <stdio.h>
2 #include <semaphore.h>
3 #include <pthread.h>
4
5 int x = 0;
6
7 void *ProcessA(void* mess)
8 {
9     while(1)
10    {
11        x = x + 1;
12        if (x==20)
13            x = 0 ;
14        printf("x(pA) = %d\n",x);
15    }
16 }
17
18 void *ProcessB(void* mess)
19 {
20     while(1)
21    {
22        x = x + 1;
23        if (x==20)
24            x = 0 ;
25        printf("x(pB) = %d\n",x);
26    }
27 }
28
29 int main()
30 {
31     pthread_t pA, pB;
32     pthread_create(&pA, NULL, &ProcessA, NULL);
33     pthread_create(&pB, NULL, &ProcessB, NULL);
34     while(1) {}
35     return 0;
36 }
37 }
```

Đây là chương trình

```
x(pB) = 2
x(pB) = 3
x(pB) = 4
x(pB) = 5
x(pB) = 6
x(pB) = 7
x(pB) = 8
x(pB) = 9
x(pB) = 10
x(pB) = 11
x(pB) = 12
x(pB) = 13
x(pB) = 14
x(pB) = 15
x(pA) = 13
x(pA) = 17
x(pA) = 18
x(pA) = 19
x(pA) = 0
x(pA) = 1
x(pA) = 2
x(pA) = 3
x(pA) = 4
x(pA) = 5
x(pA) = 6
x(pA) = 7
x(pA) = 8
x(pA) = 9
x(pA) = 10
x(pA) = 11
x(pA) = 12
x(pA) = 13
x(pA) = 14
x(pA) = 15
x(pA) = 16
```

Đây là kết quả chạy chương trình

- **Nhận xét:** Lỗi xuất hiện khi chưa được đồng bộ. Process A đã lấy giá trị của biến x ở thời điểm Process B đang tính toán x=15, do đó sau khi Process B kết thúc quá trình chạy thì Process A lập tức in ra kết quả đã tính trước đó.

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

```
Open [v] [+]
1 #include <stdio.h>
2 #include <semaphore.h>
3 #include <pthread.h>
4
5 int x = 0;
6 pthread_mutex_t mutex;
7
8 void *ProcessA(void* mess)
9 {
10     while(1)
11     {
12         pthread_mutex_lock(&mutex);
13         x = x + 1;
14         if (x==20)
15             x = 0 ;
16         printf("x(pA) = %d\n",x);
17         pthread_mutex_unlock(&mutex);
18     }
19 }
20
21 void *ProcessB(void* mess)
22 {
23     while(1)
24     {
25         pthread_mutex_lock(&mutex);
26         x = x + 1;
27         if (x==20)
28             x = 0 ;
29         printf("x(pB) = %d\n",x);
30         pthread_mutex_unlock(&mutex);
31     }
32 }
33
34 int main()
35 {
36     pthread_t pA, pB;
37     pthread_mutex_init(&mutex, NULL);
38     pthread_create(&pA, NULL, &ProcessA, NULL);
39     pthread_create(&pB, NULL, &ProcessB, NULL);
40     while(1) {}
41     return 0;
42 }
43 }
```

Đây là chương trình sau khi đồng bộ với mutex

```
zeri@LAPTOP-HQ4PM7S6: ~ × + v
x(pB) = 8
x(pB) = 9
x(pB) = 10
x(pB) = 11
x(pB) = 12
x(pB) = 13
x(pB) = 14
x(pB) = 15
x(pB) = 16
x(pB) = 17
x(pB) = 18
x(pA) = 19
x(pA) = 0
x(pA) = 1
x(pA) = 2
x(pA) = 3
x(pA) = 4
x(pA) = 5
x(pA) = 6
x(pA) = 7
x(pA) = 8
x(pA) = 9
x(pA) = 10
x(pA) = 11
x(pA) = 12
x(pA) = 13
x(pA) = 14
x(pA) = 15
x(pA) = 16
x(pA) = 17
x(pA) = 18
x(pA) = 19
x(pA) = 0
x(pA) = 1
x(pA) = 2
```

Kết quả chạy chương trình

- **Nhận xét:** biến mutex đóng vai trò là chìa khóa đóng mở vùng tranh chấp, tránh tình trạng process lấy giá trị của biến trong vùng tranh chấp khi 1 process đang chạy làm dữ liệu trở nên không đồng nhất.