

Name: Hồ Hồng Hà

ID: 20520480

Class:IT007.M13.1

OPERATING SYSTEM LAB 5'S REPORT

SUMMARY

Task		Status	Page
Section 5.4	Bài 1	Hoàn thành	2
	Bài 2	Done	4
	Bài 3	Done	6
	Bài 4	Done	8

Self-scores: 5

Note: Export file to **PDF and name the file by following format:
LAB X – <Student ID>.pdf*

Section 1.5

1. Task name 1

2. Task name 2

3. Task name 3

...

Bài 1: Hiện thực hóa mô hình trong ví dụ **5.3.1.2**, tuy nhiên thay bằng điều kiện sau:

$\text{sells} \leq \text{products} \leq \text{sells} + [2 \text{ số cuối của MSSV} + 10]$

Source code:

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

int sells=0,products=0;
sem_t sem1,sem2;
void *processA(void* mess)
{
    while(1){
        sem_wait(&sem1);
        sells++;
        printf("SELLs = %d\n",sells);
        sem_post(&sem2);
    }
}

void *processB(void* mess)
{
    while(1){
        sem_wait(&sem2);
        products++;
        printf("PRODUCTs = %d\n",products);
        sem_post(&sem1);
    }
}

int main()
{
    sem_init(&sem1,0,0);
    sem_init(&sem2,0,80+10);
    pthread_t pA,pB;
    pthread_create(
```

```

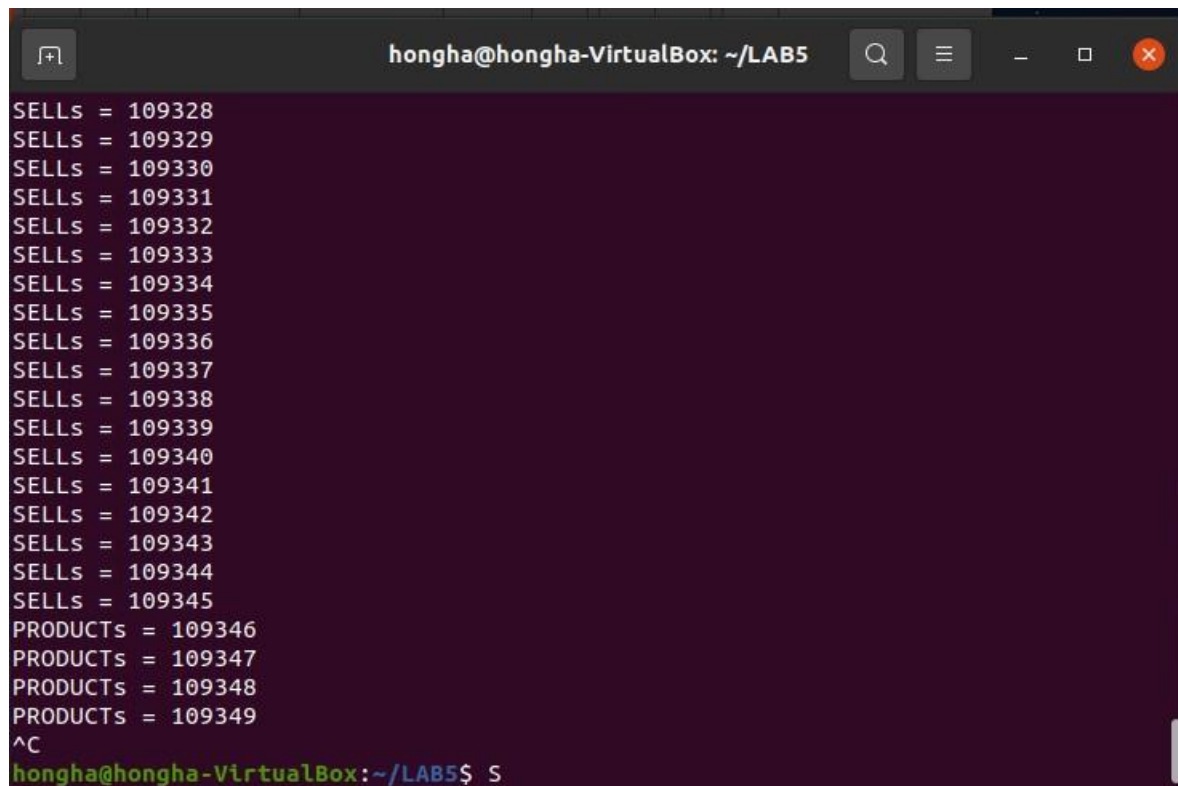
        &pA,
        NULL,
        &processA,
        NULL
    );

    pthread_create(
        &pB,
        NULL,
        &processB,
        NULL
    );

    while(1){ }
    return 0;
}

```

Kết quả chạy thử:



```

hongha@hongha-VirtualBox: ~/LAB5
SELLs = 109328
SELLs = 109329
SELLs = 109330
SELLs = 109331
SELLs = 109332
SELLs = 109333
SELLs = 109334
SELLs = 109335
SELLs = 109336
SELLs = 109337
SELLs = 109338
SELLs = 109339
SELLs = 109340
SELLs = 109341
SELLs = 109342
SELLs = 109343
SELLs = 109344
SELLs = 109345
PRODUCTs = 109346
PRODUCTs = 109347
PRODUCTs = 109348
PRODUCTs = 109349
^C
hongha@hongha-VirtualBox:~/LAB5$ S

```

Bài 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”.

Source code:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<pthread.h>
#include<semaphore.h>
sem_t sem;
pthread_mutex_t mutex;
int n;
int i = 0;
int static dem=0;
int a[1000];
void* process1()
{
    while (1)
    {
        pthread_mutex_unlock(&mutex);
        sem_wait(&sem);
        a[i++] = rand() % (n - 1);
        dem++;
        printf("\n[PUSH] Number of elements in array a: %d", dem);
        pthread_mutex_lock(&mutex);
    }
}
void* process2()
{
    int j;
    while (1)
    {
        pthread_mutex_lock(&mutex);

        if (dem==0)
        {
            printf("\n[POP] Nothing in array a");
        }
        else
```

```

    {
        dem--;
        for (j = 0; j < dem; j++)
        {
            a[j] = a[j + 1];
        }

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

    pthread_mutex_unlock(&mutex);
}

void main()
{
    printf("\nEnter n: ");
    scanf("%d",&n);
    sem_init(&sem, 0, n);
    pthread_mutex_init (&mutex,NULL);
    pthread_t p1, p2;
    pthread_create(&p1, NULL, process1, NULL);
    pthread_create(&p2, NULL, process2, NULL);
    while(1);
}

```

Kết quả chạy thử:

```

Enter n: 10

[PUSH] Number of elements in array a: 1
[PUSH] Number of elements in array a: 2
[PUSH] Number of elements in array a: 3
[PUSH] Number of elements in array a: 4
[PUSH] Number of elements in array a: 5
[PUSH] Number of elements in array a: 6
[PUSH] Number of elements in array a: 7
[PUSH] Number of elements in array a: 8
[PUSH] Number of elements in array a: 9
[POP] Number of elements in array a: 8
[POP] Number of elements in array a: 7
[POP] Number of elements in array a: 6
[POP] Number of elements in array a: 5
[POP] Number of elements in array a: 4
[POP] Number of elements in array a: 3
[POP] Number of elements in array a: 2
[POP] Number of elements in array a: 1
[POP] Number of elements in array a: 0
[POP] Nothing in array a

```

Bài 3:

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

int x = 0;	
PROCESS A	PROCESS B
processA() { while(1){ x = x + 1; if (x == 20) x = 0; print(x); } }	processB() { while(1){ x = x + 1; if (x == 20) x = 0; print(x); } }

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

Source code:

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

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

```
int x = 0;
```

```
void* processA()
```

```
{
```

```
    while (1)
```

```
    {
```

```
        x = x + 1;
```

```
        if (x == 20)
```

```
        {
```

```
            x = 0;
```

```
        }
```

```
        printf("PA: x = %d\n", x);
```

```
    }
```

```
}
```

```
void* processB()
```

```
{
```

```
    while (1)
```

```
    {
```

```
        x = x + 1;
```

```
        if (x == 20)
```

```
{
```

```
    x = 0;
```

```
    }
```

```
    printf("PB: x = %d\n", x);
```

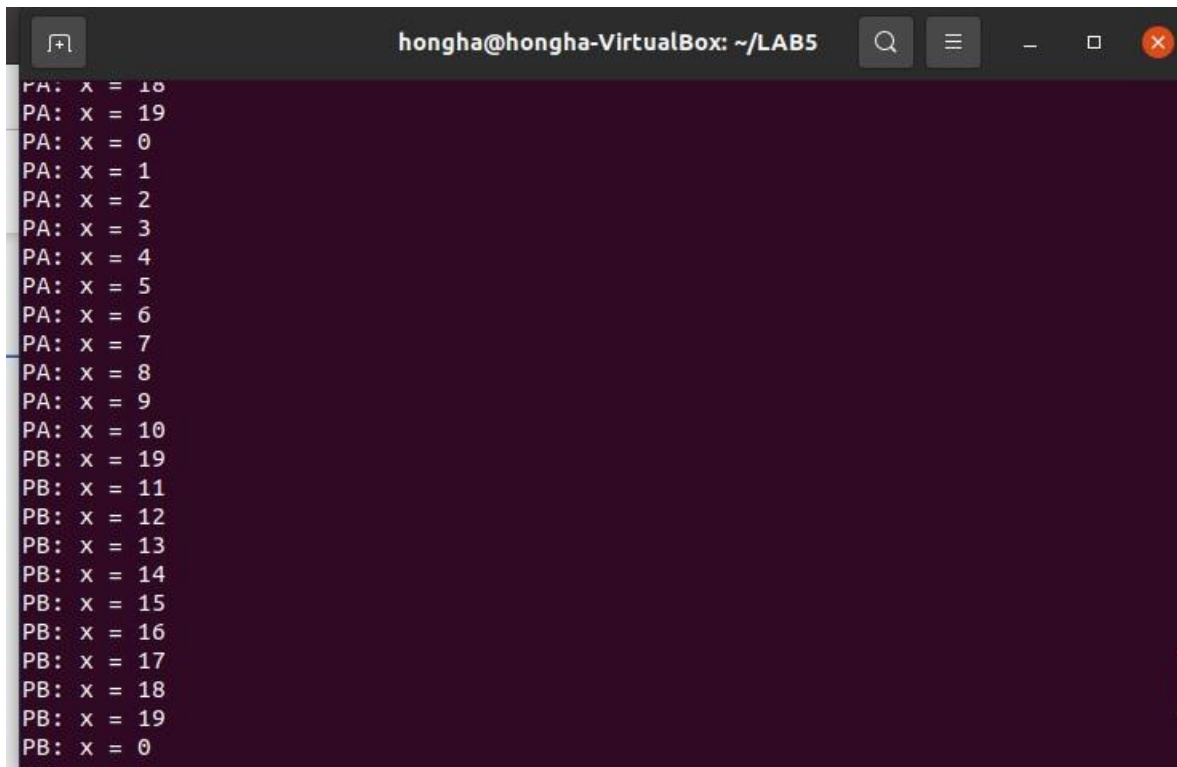
```
}
```

```

}
void main()
{
    pthread_t pA, pB;
    pthread_create(&pA, NULL, &processA, NULL);
    pthread_create(&pB, NULL, &processB, NULL);
    while (1);
}

```

Kết quả chạy thử:



```

PA: x = 10
PA: x = 19
PA: x = 0
PA: x = 1
PA: x = 2
PA: x = 3
PA: x = 4
PA: x = 5
PA: x = 6
PA: x = 7
PA: x = 8
PA: x = 9
PA: x = 10
PB: x = 19
PB: x = 11
PB: x = 12
PB: x = 13
PB: x = 14
PB: x = 15
PB: x = 16
PB: x = 17
PB: x = 18
PB: x = 19
PB: x = 0

```

Nhận xét: Kết quả chạy theo đoạn code trên là bất hợp lý. Vì đây có 2 processA và processB in ra màn hình giá trị từ 0 đến 19 theo thứ tự. Mà theo kết quả trên thì khi chương chuyển từ processA qua processB thì kết quả in ra không còn theo đúng thứ tự.

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

Source code

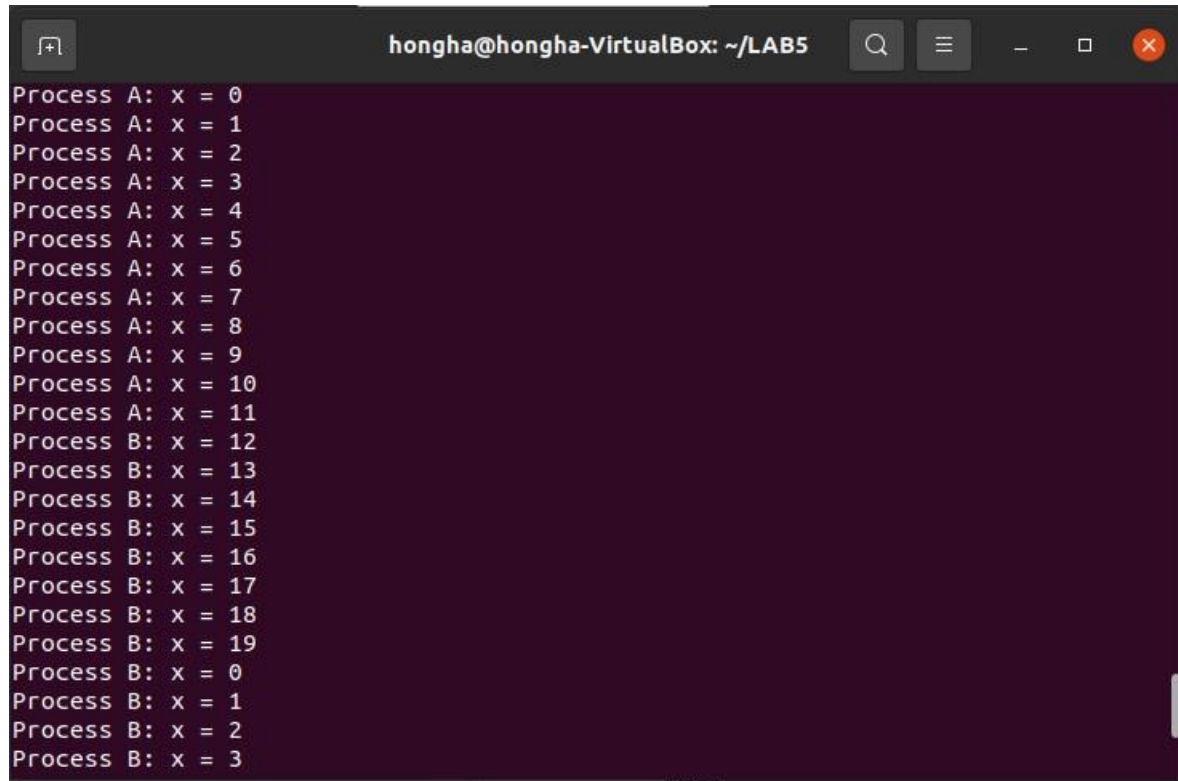
```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

int x = 0;
pthread_mutex_t mutex;
void* processA()
{
    while (1)
    {
        pthread_mutex_lock(&mutex);
        x++;
        if (x == 20)
        {
            x = 0;
        }
        printf("Process A: x = %d\n", x);
        pthread_mutex_unlock(&mutex);
    }
}
void* processB()
{
    while (1)
    {
        pthread_mutex_lock(&mutex);
        x++;
        if (x == 20)
        {
            x = 0;
        }
        printf("Process B: x = %d\n", x);
        pthread_mutex_unlock(&mutex);
    }
}
void main()
{
    pthread_mutex_init(&mutex, NULL);
    pthread_t pA, pB;
```



```
pthread_create(&pA, NULL, &processA, NULL);  
pthread_create(&pB, NULL, &processB, NULL);  
while (1);  
}
```

Kết quả chạy thử:



```
hongha@hongha-VirtualBox: ~/LAB5  
Process A: x = 0  
Process A: x = 1  
Process A: x = 2  
Process A: x = 3  
Process A: x = 4  
Process A: x = 5  
Process A: x = 6  
Process A: x = 7  
Process A: x = 8  
Process A: x = 9  
Process A: x = 10  
Process A: x = 11  
Process B: x = 12  
Process B: x = 13  
Process B: x = 14  
Process B: x = 15  
Process B: x = 16  
Process B: x = 17  
Process B: x = 18  
Process B: x = 19  
Process B: x = 0  
Process B: x = 1  
Process B: x = 2  
Process B: x = 3
```