

Name: Phạm Nguyễn Hải Anh

ID: 21520586

Class: IT007.ANTN.1

## OPERATING SYSTEM LAB X'S REPORT

### SUMMARY

Task		Status	Page
Section 5.4	Ex 1	Hoàn thành	2
	Ex 2	Hoàn thành	3
	Ex 3	Hoàn thành	5
	Ex 4	Hoàn thành	7
Section 5.5	...	Hoàn thành	9
	...		

Self-scores:

*\*Note: Export file to **PDF** and name the file by following format:  
**Student ID\_LABx.pdf***

## Section 5.4

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

File bt1.cpp:



```
GNU nano 4.8
#include <iostream>
#include <semaphore.h>
#include <pthread.h>
#include <unistd.h>

using namespace std;

sem_t sem;
int sell, product;

void *sellFunc(void *arg){
    while (true){
        sleep(1);
        sem_wait(&sem);
        sell++;
        cout << "Sell: " << sell << endl;
    }
}

void *productFunc (void *arg){
    while (true){
        if (product < sell + 86 + 10) {
            sleep (1);
            product++;
            cout << "Product: " << product << endl;
            sem_post(&sem);
        }
    }
}

int main(){
    sem_init (&sem, 0, 0);
    pthread_t thread;
    pthread_create (&thread, NULL, sellFunc, NULL);
    pthread_create (&thread, NULL, productFunc, NULL);
    pthread_join (thread, NULL);
    sem_destroy (&sem);
    return 0;
}
```

Run:

```

pnggha2@ansibletarget1:~/Lab5$ nano bt1.cpp
pnggha2@ansibletarget1:~/Lab5$ g++ bt1.cpp -o bt1.o -lpthread -lrt
pnggha2@ansibletarget1:~/Lab5$ ./bt1.o
Product: 1
Sell: 1
Product: 2
Sell: 2
Product: 3
Sell: 3
Product: 4
Sell: 4
Product: 5
Sell: 5
Product: 6
Sell: 6
Product: 7
Sell: 7
Product: 8
Sell: 8
Product: 9
Sell: 9
Product: 10
Sell: 10
Product: 11
Sell: 11
Product: 12
Sell: 12
Product: 13
Sell: 13
Product: 14
Sell: 14
Product: 15
Sell: 15
Product: 16
Sell: 16
Product: 17
Sell: 17
Product: 18
Sell: 18
Product: 19
Sell: 19
Product: 20
Sell: 20
Product: 21
Sell: 21
Product: 22
Sell: 22
Product: 23
Sell: 23
Product: 24
Sell: 24
^C

```

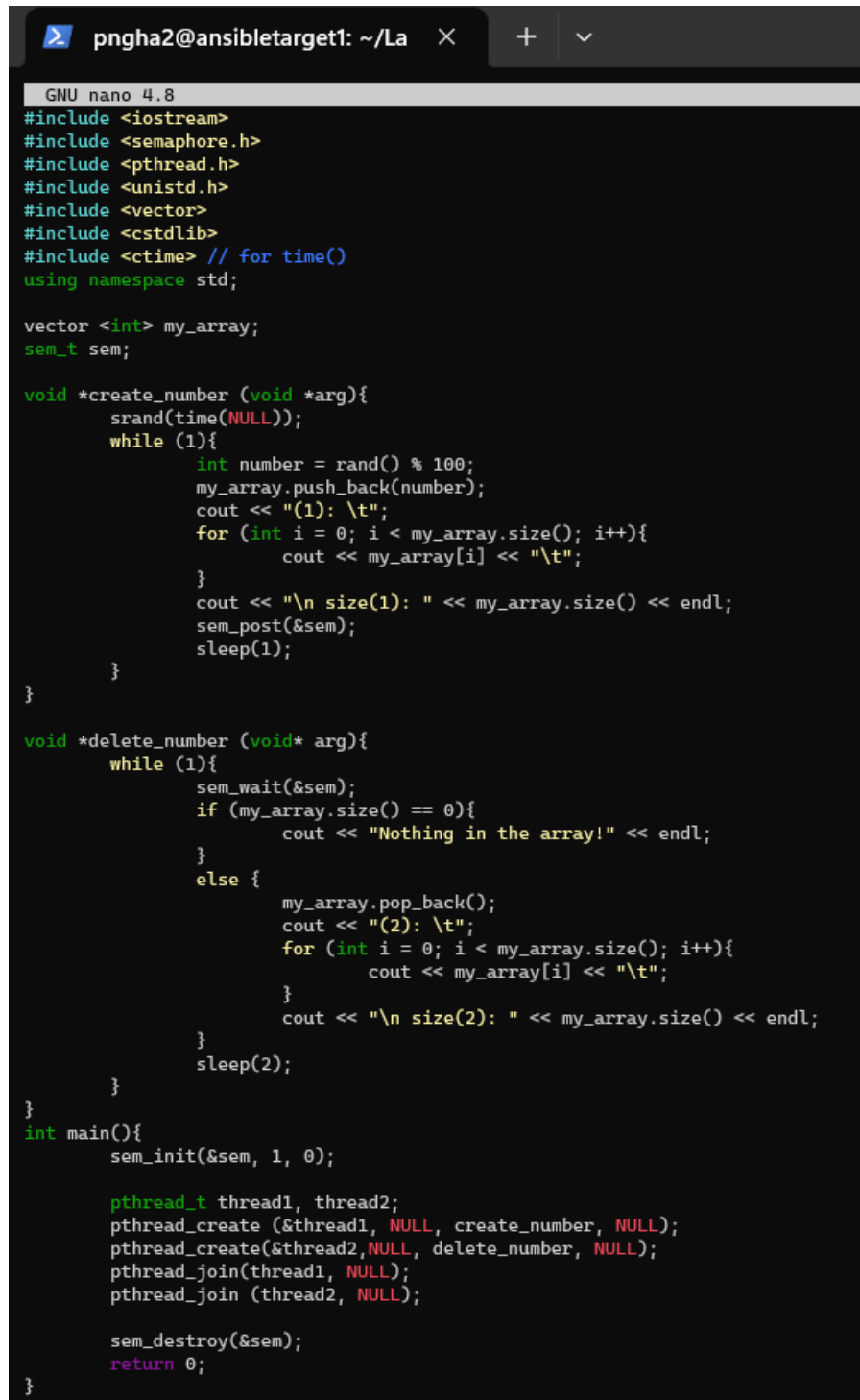
2. Cho một mảng  $a$  được khai báo như một mảng số nguyên có thể chứa  $n$  phần tử,  $a$  được khai báo như một biến toàn cục. Viết chương trình bao gồm 2 thread chạy song song:

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.

File bt2.cpp:



```
GNU nano 4.8
#include <iostream>
#include <semaphore.h>
#include <pthread.h>
#include <unistd.h>
#include <vector>
#include <cstdlib>
#include <ctime> // for time()
using namespace std;

vector<int> my_array;
sem_t sem;

void *create_number (void *arg){
    srand(time(NULL));
    while (1){
        int number = rand() % 100;
        my_array.push_back(number);
        cout << "(1): \t";
        for (int i = 0; i < my_array.size(); i++){
            cout << my_array[i] << "\t";
        }
        cout << "\n size(1): " << my_array.size() << endl;
        sem_post(&sem);
        sleep(1);
    }
}

void *delete_number (void* arg){
    while (1){
        sem_wait(&sem);
        if (my_array.size() == 0){
            cout << "Nothing in the array!" << endl;
        }
        else {
            my_array.pop_back();
            cout << "(2): \t";
            for (int i = 0; i < my_array.size(); i++){
                cout << my_array[i] << "\t";
            }
            cout << "\n size(2): " << my_array.size() << endl;
        }
        sleep(2);
    }
}

int main(){
    sem_init(&sem, 1, 0);

    pthread_t thread1, thread2;
    pthread_create (&thread1, NULL, create_number, NULL);
    pthread_create(&thread2, NULL, delete_number, NULL);
    pthread_join(thread1, NULL);
    pthread_join (thread2, NULL);

    sem_destroy(&sem);
    return 0;
}
```

Chạy chương trình:

```

pnggha2@ansibletarget1:~/Lab5$ g++ bt2.cpp -o bt2.o -lpthread -lrt
pnggha2@ansibletarget1:~/Lab5$ ./bt2.o
(1): 10
size(1): 1
(2):
size(2): 0
(1): 44
size(1): 1
(1): 44 94
size(1): 1
(2): 44
size(2): 1
(1): 44 13
size(1): 2
(2): 44
size(2): 1
(1): 44 25
size(1): 2
(1): 44 25 72
size(1): 3
(2): 44 25
size(2): 2
(1): 44 25 89
size(1): 3

```

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

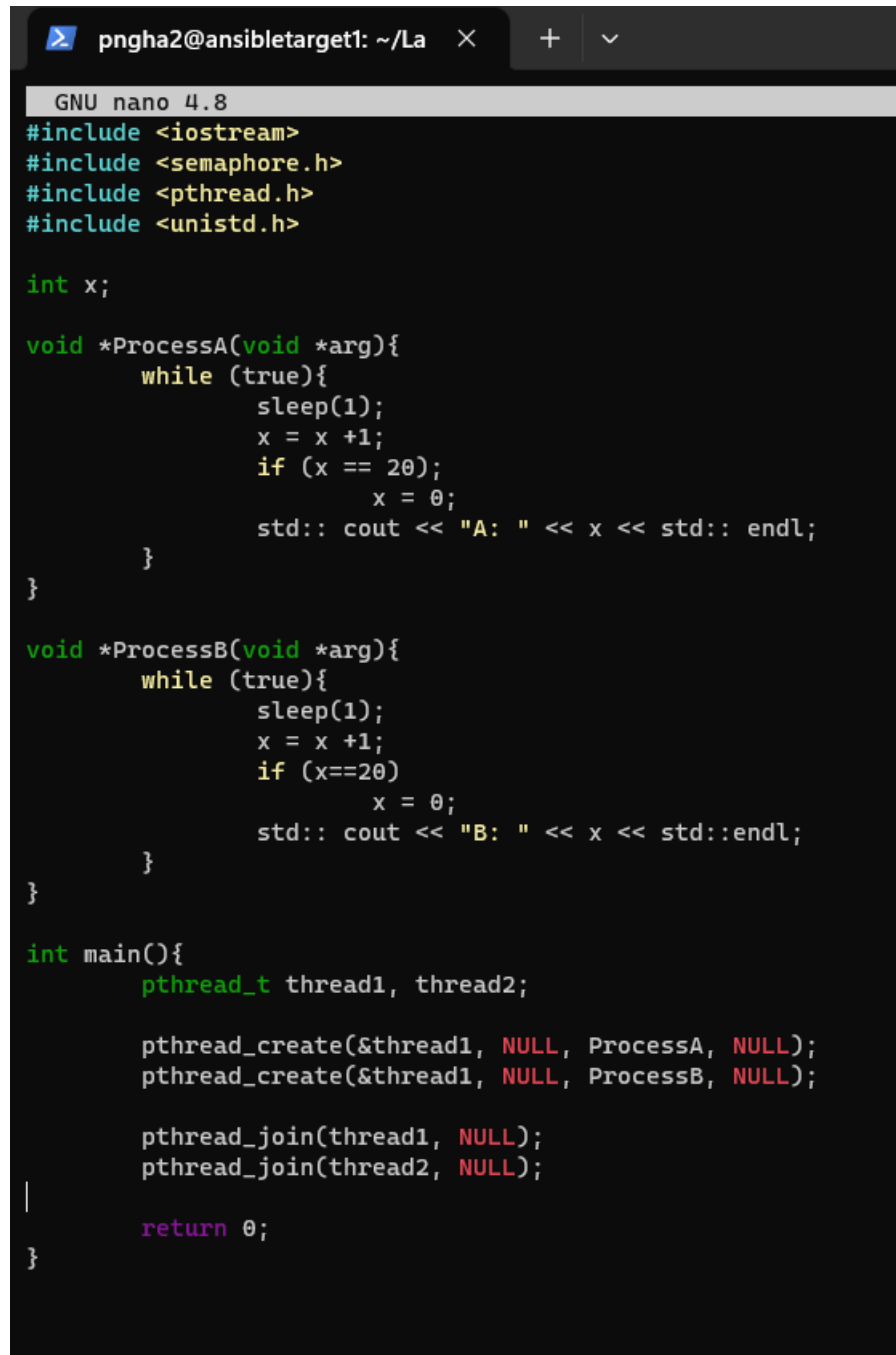
int x = 0;	
PROCESS A	PROCESS B
processA()	processB()
{	{

12

while(1){ x = x + 1; if (x == 20) x = 0; print(x); }	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:

File bt3.cpp:



```
GNU nano 4.8
#include <iostream>
#include <semaphore.h>
#include <pthread.h>
#include <unistd.h>

int x;

void *ProcessA(void *arg){
    while (true){
        sleep(1);
        x = x +1;
        if (x == 20);
            x = 0;
        std::cout << "A: " << x << std::endl;
    }
}

void *ProcessB(void *arg){
    while (true){
        sleep(1);
        x = x +1;
        if (x==20)
            x = 0;
        std::cout << "B: " << x << std::endl;
    }
}

int main(){
    pthread_t thread1, thread2;

    pthread_create(&thread1, NULL, ProcessA, NULL);
    pthread_create(&thread1, NULL, ProcessB, NULL);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    |
    return 0;
}
```

Chạy chương trình:

```

pnggha2@ansibletarget1:~/Lab5$ nano bt3.cpp
pnggha2@ansibletarget1:~/Lab5$ g++ bt3.cpp -o bt3.o -lpthread -lrt
pnggha2@ansibletarget1:~/Lab5$ ./bt3.o
A: 1
B: 1
A: 0
B: 1
B: 2
A: 0
A: B: 11

A: B: 10

A: B: 0
1
A: 0
B: 1
B: A: 0
0
B: 0
A: 0
A: B: 01

A: 0
B: 1
B: A: 11

B: 2A:
0
A: 0
B: 1
A: 0
B: 1

```

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

File bt4.cpp:

```
pngha2@ansibletarget1: ~/La × + v
GNU nano 4.8
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <iostream>
int x;
pthread_mutex_t mutex;
using namespace std;
void *ProcessA (void *arg){
    while (true){
        pthread_mutex_lock (&mutex);
        sleep(1);
        x = x +1;
        if (x == 20)
            x = 0;
        cout << "A: " << x << endl;
        pthread_mutex_unlock(&mutex);
    }
}

void *ProcessB (void *arg){
    while (true){
        pthread_mutex_lock(&mutex);
        sleep(1);
        x = x +1;
        if (x == 20){
            x = 0;
        }
        cout << "B: " << x << endl;
        pthread_mutex_unlock(&mutex);
    }
}

int main(){
    pthread_t thread1, thread2;
    pthread_mutex_init (&mutex, NULL);

    pthread_create (&thread1, NULL, ProcessA, NULL);
    pthread_create (&thread2, NULL, ProcessB, NULL);

    pthread_join(thread1,NULL);
    pthread_join(thread2,NULL);

    return 0;
}
```

Build file object:

```
g++ bt4.cpp -o bt4.o -lpthread -lrt
```

Chạy chương trình:



```
-bash: ./bt4: No such file or directory
pngha2@ansibletarget1:~/Lab5$ ./bt4.o
A: 1
A: 2
A: 3
A: 4
A: 5
A: 6
A: 7
A: 8
A: 9
A: 10
A: 11
```

5.5:

File bt5\_5.cpp:

```
pngha2@ansibletarget1: ~/La × + ▾
GNU nano 4.8
#include <stdio.h>
#include <pthread.h>

int x1, x2, x3, x4, x5, x6;

int w, v, y, z, ans;

pthread_mutex_t mutex; // Mutex for synchronization

void* compute_w(void* arg) {
    pthread_mutex_lock(&mutex);
    w = x1 * x2;
    pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
}

void* compute_v(void* arg) {
    pthread_mutex_lock(&mutex);
    v = x3 * x4;
    pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
}

void* compute_y(void* arg) {
    pthread_mutex_lock(&mutex);
    y = v * x5;
    pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
}

void* compute_z(void* arg) {
    pthread_mutex_lock(&mutex);
    z = v * x6;
    pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
}

int main() {
    x1 = 1;
    x2 = 2;
    x3 = 3;
    x4 = 4;
    x5 = 5;
    x6 = 6;

    pthread_mutex_init(&mutex, NULL);

    pthread_t thread_w, thread_v, thread_y, thread_z;

    pthread_create(&thread_w, NULL, compute_w, NULL);
    pthread_create(&thread_v, NULL, compute_v, NULL);

    pthread_join(thread_w, NULL);
    pthread_join(thread_v, NULL);

    pthread_create(&thread_y, NULL, compute_y, NULL);
    pthread_create(&thread_z, NULL, compute_z, NULL);

    pthread_join(thread_y, NULL);
    pthread_join(thread_z, NULL);

    pthread_mutex_lock(&mutex);
    y = w * y;
    z = w * z;
    ans = y + z;
    pthread_mutex_unlock(&mutex);
}
```

```
pthread_join(thread_y, NULL);  
pthread_join(thread_z, NULL);  
  
pthread_mutex_lock(&mutex);  
y = w * y;  
z = w * z;  
ans = y + z;  
pthread_mutex_unlock(&mutex);  
  
printf("ans = %d\n", ans);  
  
pthread_mutex_destroy(&mutex);  
  
return 0;  
}
```

Run:

```
pngha2@ansibletarget1:~/Lab5$ g++ bt5_5.cpp -o bt5_5.o -lpthread -lrt  
pngha2@ansibletarget1:~/Lab5$ ./bt5_5.o  
ans = 264
```