

Name: Lê Hoàng Khanh

ID: 20521448

Class: IT007.M12.MTCL.1

## OPERATING SYSTEM LAB 5'S REPORT

### SUMMARY

Task		Status	Page
5.4/Hướng dẫn thực hành	Câu 1	Done	2
	Câu 2	Done	3
	Câu 3	Done	6
	Câu 4	Done	9
5.5/	Câu 1	Done	12

**Self-scores:**

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

## 5.4/ Hướng dẫn thực hành

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

Code:

Code	Ý Nghĩa
<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;pthread.h&gt; #include &lt;unistd.h&gt; #include &lt;semaphore.h&gt;  sem_t sem; sem_t sem1; int sells= 0; int products=0; void* PROCESSA() {     while(1)     {         sem_wait(&amp;sem);         sells++;         printf("sells = %d\n", sells);         sem_post(&amp;sem1);         sleep(1);     } } void* PROCESSB() {     while(1)     {         sem_wait(&amp;sem1);         products++;         printf("products = %d\n", products);</pre>	<p>Khái báo các thư viện</p> <p>Khai báo các biến semaphore</p> <p>Khai báo giá trị ban đầu cho sell, product</p> <p>Viết code cho tiểu trình a</p> <p>Ngăn tiểu trình chạy nếu sem.value=0</p> <p>Tăng giá trị sem1.value lên 1 đơn vị</p> <p>Viết code cho tiểu trình b</p> <p>Ngăn tiểu trình chạy nếu sem1.value=0</p>

<pre> sem_post(&amp;sem); //sleep(1);  } } void main() {     sem_init(&amp;sem,0,0);     sem_init(&amp;sem1,0,58);     pthread_t th1, th2;     pthread_create(&amp;th1, NULL, &amp;PROCESSA, NULL);     pthread_create(&amp;th2, NULL, &amp;PROCESSB, NULL);     while(1); } </pre>	<p>Tăng giá trị sem.value lên 1 đơn vị</p> <p>sem.value = 0 sem1.value= 10 +48</p> <p>Tạo các tiểu trình chạy song song với nhau</p>
---	--

Thực thi:

```

khanhek@khanhek-VirtualBox: ~$ sells
sells = 2
sells = 3
sells = 4

khanhek@khanhek-VirtualBox: ~/Lab5
khanhek@khanhek-VirtualBox:~/Lab5$ ./Cau1

```

**Lưu ý:** Chương trình đã được bổ sung thêm code để xuất giá trị lên 2 terminal khác nhau

- 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

Code:

4

<pre> }  void* PROCESS2() {     int j, b;     while (1)     {         sem_wait(&amp;sem1);         //if (dem &lt;= n) {         if (dem == 0)         {             printf("\nThread 2: Nothing in array a");         }         else         {             dem--;             b = a[0];             for (j = 0; j &lt; dem; j++)             {                 a[j] = a[j + 1];             }             printf("\nThread 2: Sophan tu co trong a la %2d", dem);         }         sem_post(&amp;sem1);         sleep(1);     } }  void main() </pre>	<p>Viết code cho tiểu trình 2</p> <p>Ngăn tiểu trình chạy nếu sem1.value=0</p> <p>Tăng giá trị sem1.value lên 1 đơn vị</p>
--	--



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>
---	---

Code:

Code	Ý nghĩa
<pre> #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;pthread.h&gt; #include &lt;semaphore.h&gt; int x = 0; void* PROCESS_A() {     while (1) </pre>	<p>Khai báo các thư viện cần thiết</p> <p>Viết code cho tiểu trình a</p>

<pre> {     x = x + 1;     if (x == 20)     {         x = 0;     }     printf("PROCESS A: x = %d\n", x); } } void* PROCESS_B() {     while (1)     {         x = x + 1;         if (x == 20)         {             x = 0;         }         printf("PROCESS B: x = %d\n", x);     } } void main() {     pthread_t th1, th2;     pthread_create(&amp;th1, NULL, &amp;PROCESS_A, NULL);     pthread_create(&amp;th2, NULL, &amp;PROCESS_B, NULL);     while (1); } </pre>	<p>Viết code cho tiểu trình b</p> <p>Tạo các tiểu trình chạy song song</p>
---	--

Thực thi:



- Có xuất hiện lỗi đồng bộ khi tiểu trình a chuyển sang tiểu trình b

```

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 = 17
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

```

- Có xuất hiện lỗi đồng bộ khi tiểu trình b chuyển sang tiểu trình a

```

PROCESS B: x = 6
PROCESS B: x = 7
PROCESS B: x = 8
PROCESS B: x = 9
PROCESS B: x = 10
PROCESS B: x = 11
PROCESS B: x = 12
PROCESS B: x = 13
PROCESS A: x = 4
PROCESS A: x = 14
PROCESS A: x = 15
PROCESS A: x = 16
PROCESS A: x = 17
PROCESS A: x = 18
PROCESS A: x = 19
PROCESS A: x = 0
PROCESS A: x = 1
PROCESS A: x = 2

```

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

Code:

Code	Ý nghĩa
<pre> #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;pthread.h&gt; #include &lt;semaphore.h&gt; sem_t sem_1, sem_2; int x = 0; pthread_mutex_t mutex; void* PROCESS_A() { </pre>	<p>Khai báo các thư viện cần thiết</p> <p>Khai báo các biến semaphore sem_1, sem_2</p> <p>Khai báo biến mutex</p> <p>Viết code cho tiểu trình A</p>

<pre> while (1) {     pthread_mutex_lock(&amp;mutex);     x++;     if (x == 20)     {         x = 0;     }     printf("PROCESS A: x = %d\n", x);      pthread_mutex_unlock(&amp;mutex);  }  void* PROCESS_B() {     while (1)     {         pthread_mutex_lock(&amp;mutex);         x++;         if (x == 20)         {             x = 0;         }         printf("PROCESS B: x = %d\n", x);          pthread_mutex_unlock(&amp;mutex);     } }  void main() { </pre>	<p>Hàm sẽ bị khóa tiểu trình A nếu mutex đã bị khóa bởi tiểu trình B</p> <p>Mở khóa cho khóa mutex</p> <p>Viết code cho tiểu trình B</p> <p>Hàm sẽ bị khóa tiểu trình B nếu mutex đã bị khóa bởi tiểu trình A</p> <p>Mở khóa cho khóa mutex</p>
---	---

<pre>pthread_mutex_init(&amp;mutex, NULL); pthread_t th1, th2; pthread_create(&amp;th1, NULL, &amp;PROCESS_A, NULL); pthread_create(&amp;th2, NULL, &amp;PROCESS_B, NULL); while (1); }</pre>	<p>Khởi tạo giá trị khóa mutex với giá trị mặc định</p> <p>Tạo các tiểu trình chạy song song</p>
---	--

Thực thi:

- Không có xuất hiện lỗi đồng bộ khi tiểu trình A chuyển sang tiểu trình B

```
PROCESS A: x = 16
PROCESS A: x = 17
PROCESS A: x = 18
PROCESS A: x = 19
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 B: x = 6
PROCESS B: x = 7
PROCESS B: x = 8
PROCESS B: x = 9
PROCESS B: x = 10
PROCESS B: x = 11
PROCESS B: x = 12
PROCESS B: x = 13
PROCESS B: x = 14
PROCESS B: x = 15
```

- Không có xuất hiện lỗi đồng bộ khi tiểu trình B chuyển sang tiểu trình A

```
PROCESS B: 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 A: x = 18
PROCESS A: x = 19
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
```

## 5.5/Bài tập ôn tập

1. 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:

13

✚ (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:

Code	Ý nghĩa
<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;pthread.h&gt; #include &lt;semaphore.h&gt;  sem_t p1_5, p1_6, p2_3, p2_4, p3_5, p4_6, p5_7, p6_7; int x1, x2, x3, x4, x5, x6; int w, v, z, y, x; int ans = 0;</pre>	<p>Khai báo các thư viện cần thiết</p>         <p>Khai báo các biến semaphore Khai báo các biến toàn cục</p>

<pre> void* PROCESS1() {     w = x1 * x2;     printf("Process a: w = %d\n", w);     sem_post(&amp;p1_5);     sem_post(&amp;p1_6);     sleep(1); }  void* PROCESS2() {     v = x3 * x4;     printf("Process b: v = %d\n", v);     sem_post(&amp;p2_3);     sem_post(&amp;p2_4);     sleep(1); }  void* PROCESS3() {     sem_wait(&amp;p2_3);     y = v * x5;     printf("Process c: y = %d\n", y);     sem_post(&amp;p3_5);     sleep(1); }  void *PROCESS4() {     sem_wait(&amp;p2_4);     z = v * x6;     printf("Process d: z = %d\n", z);     sem_post(&amp;p4_6);     sleep(1); } </pre>	<p>Viết code cho tiểu trình 1</p> <p>Tăng giá trị p1_5.value lên 1 đơn vị Tăng giá trị p1_6.value lên 1 đơn vị</p> <p>Viết code cho tiểu trình 2</p> <p>Tăng giá trị p2_3.value lên 1 đơn vị Tăng giá trị p2_4.value lên 1 đơn vị</p> <p>Viết code cho tiểu trình 3</p> <p>Ngăn tiểu trình chạy nếu p2_3.value=0</p> <p>Tăng giá trị p3_5.value lên 1 đơn vị</p> <p>Viết code cho tiểu trình 4</p> <p>Ngăn tiểu trình chạy nếu p2_4.value=0</p> <p>Tăng giá trị p4_6.value lên 1 đơn vị</p>
---	---

<pre> } void *PROCESS5() {     sem_wait(&amp;p1_5);     sem_wait(&amp;p3_5);     y = w * y;     printf("Process e: y = %d\n", y);     sem_post(&amp;p5_7);     sleep(1); } void *PROCESS6() {     sem_wait(&amp;p1_6);     sem_wait(&amp;p4_6);     z = w * z;     printf("Process f: z = %d\n", z);     sem_post(&amp;p6_7);     sleep(1); } void* PROCESS7() {     sem_wait(&amp;p5_7);     sem_wait(&amp;p6_7);     ans = y + z;     printf("Process g: ans = %d\n", ans);     sleep(1); } void main() {     printf("Moi nhap cac gia gia tri tu x1 den x6: ");     scanf("%d%d%d%d%d%d",&amp;x1,&amp;x2,&amp;x3,&amp;x4,&amp;x5,&amp;x6);     sem_init(&amp;p1_5, 0, 1); </pre>	<p>Viết code cho tiểu trình 5</p> <p>Ngăn tiểu trình chạy nếu p1_5.value=0 Ngăn tiểu trình chạy nếu p3_5.value=0</p> <p>Tăng giá trị p5_7.value lên 1 đơn vị</p> <p>Viết code cho tiểu trình 6</p> <p>Ngăn tiểu trình chạy nếu p1_6.value=0 Ngăn tiểu trình chạy nếu p4_6.value=0</p> <p>Tăng giá trị p6_7.value lên 1 đơn vị</p> <p>Viết code cho tiểu trình 7</p> <p>Ngăn tiểu trình chạy nếu p5_7.value=0 Ngăn tiểu trình chạy nếu p6_7.value=0</p> <p>Nhập giá trị cho biến toàn cục x1 đến x6 p1_5.value=1</p>
---	---

<pre> sem_init(&amp;p1_6, 0, 0); sem_init(&amp;p2_3, 0, 0); sem_init(&amp;p2_4, 0, 0); sem_init(&amp;p3_5, 0, 0); sem_init(&amp;p4_6, 0, 0); sem_init(&amp;p5_7, 0, 0); sem_init(&amp;p6_7, 0, 0);  pthread_t th1, th2, th3, th4, th5, th6, th7; pthread_create(&amp;th1, NULL, &amp;PROCESS1, NULL); pthread_create(&amp;th2, NULL, &amp;PROCESS2, NULL); pthread_create(&amp;th3, NULL, &amp;PROCESS3, NULL); thread_create(&amp;th4, NULL, &amp;PROCESS4, NULL); pthread_create(&amp;th5, NULL, &amp;PROCESS5, NULL); pthread_create(&amp;th6, NULL, &amp;PROCESS6, NULL); pthread_create(&amp;th7, NULL, &amp;PROCESS7, NULL);  while (1);  } </pre>	<pre> p1_6.value=0 p2_3.value=0 p2_4.value=0 p3_5.value=0 p4_6.value=0 p5_7.value=0 p6_7.value=0 </pre> <p>Tạo các tiểu trình chạy song song</p>
--	--

Thực thi:

```

khanhek@khanhek-VirtualBox:~/Lab5$ ./Cau5
Moi nhap cac gia gia tri tu x1 den x6: 1 2 3 4 5 6
Process a: w = 2
Process b: v = 12
Process c: y = 60
Process d: z = 72
Process e: y = 120
Process f: z = 144
Process g: ans = 264

```