



National Textile University

Department of Computer Science

Subject:

Operating System

Submitted to:

Sir Naseer

Submitted by:

Ayema

Reg number:

23-NTU-CS-1142

Assignment no. :

04

Semester: 5Th

Task 1: (Semaphore):

```
● ● ●  
1 #include <stdio.h>  
2 #include <pthread.h>  
3 #include <semaphore.h>  
4 #include <unistd.h>  
5 sem_t mutex; // Binary semaphore  
6 int counter = 0;  
7 void* thread_function(void* arg) {  
8     int id = *(int*)arg;  
9     for (int i = 0; i < 5; i++) {  
10         printf("Thread %d: Waiting...\n", id);  
11         sem_wait(&mutex); // Acquire  
12         // Critical section  
13         counter++;  
14         printf("Thread %d: In critical section | Counter = %d\n", id,  
15             counter);  
16         sleep(1);  
17         sem_post(&mutex); // Release  
18         sleep(1);  
19     }  
20     return NULL;  
21 }  
22 int main() {  
23     sem_init(&mutex, 0, 1); // Binary semaphore initialized to 1  
24     pthread_t t1, t2;  
25     int id1 = 1, id2 = 2;  
26     pthread_create(&t1, NULL, thread_function, &id1);  
27     pthread_create(&t2, NULL, thread_function, &id2);  
28     pthread_join(t1, NULL);  
29     pthread_join(t2, NULL);  
30     printf("Final Counter Value: %d\n", counter);  
31     sem_destroy(&mutex);  
32     return 0;}
```

The screenshot shows a Visual Studio Code (VS Code) interface running on a Windows host (WSL: Ubuntu-24.04). The left sidebar displays a file tree with several projects and files under 'OPERATING-SYSTEM-1 [WSL: UBUNTU-24.04]'. The current file is 'task1.c' in the 'task1' folder. The code implements a binary semaphore to synchronize two threads. The terminal window shows the execution of the program, which prints the value of the counter as it increments from 0 to 10. The status bar at the bottom indicates the terminal has 18 columns, 11 rows, and 4 spaces.

```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <semaphore.h>
4 #include <unistd.h>
5 sem_t mutex; // Binary semaphore
6 int counter = 0;
7 void* thread_function(void* arg) {
8     int id = *(int*)arg;
9     for (int i = 0; i < 5; i++) {
10         if (id == 1) {
11             sem_wait(mutex);
12             counter++;
13             sem_post(mutex);
14         }
15         else {
16             sem_wait(mutex);
17             counter--;
18             sem_post(mutex);
19         }
20     }
21 }
```

```
ayena@DESKTOP-1VVC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system-1/1142-lab9$ gcc task1.c -o task1.out -lpthread
collect2: error: ld returned 1 exit status
ayena@DESKTOP-1VVC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system-1/1142-lab9$ gcc task1.c -o task1.out -lpthread
ayena@DESKTOP-1VVC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system-1/1142-lab9$ ./task1.out
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 2
Thread 1: Waiting...
Thread 1: In critical section | Counter = 3
Thread 2: Waiting...
Thread 2: In critical section | Counter = 4
Thread 1: Waiting...
Thread 1: In critical section | Counter = 5
Thread 2: Waiting...
Thread 2: In critical section | Counter = 6
Thread 1: Waiting...
Thread 1: In critical section | Counter = 7
Thread 2: Waiting...
Thread 2: In critical section | Counter = 8
Thread 1: Waiting...
Thread 1: In critical section | Counter = 9
Thread 2: Waiting...
Thread 2: In critical section | Counter = 10
Final Counter Value: 10
```

Demonstration:

Removing Sem post → Woh critical meh hi rahay ga baheer Nahi ai ga sirf ek counter Chalay ga, threads stop forever.

- ayena@DESKTOP-1VVC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab9\$ gcc task1.c -o task1.out -lpthread
- ayena@DESKTOP-1VVC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab9\$./task1.out
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 1: Waiting...
□

Changing semaphore value from 1 to 0 → both threads enter, race condition. They blocked and waiting for each other.

```
ayema@DESKTOP-1WHC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system-1/1142-lab3$ ./task1.out  
Thread 1: Waiting...  
Thread 2: Waiting...
```

Removing sem_wait:

```

ayema@DESKTOP-1VVC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1$ cd 1142-lab9
ayema@DESKTOP-1VVC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab9$ gcc task1.c
ut -lpthread
ayema@DESKTOP-1VVC56:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab9$ ./task1.out
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 2
Thread 1: Waiting...
Thread 1: In critical section | Counter = 3
Thread 2: Waiting...
Thread 2: In critical section | Counter = 4
Thread 1: Waiting...
Thread 1: In critical section | Counter = 5
Thread 2: Waiting...
Thread 2: In critical section | Counter = 6
Thread 1: Waiting...
Thread 1: In critical section | Counter = 7
Thread 2: Waiting...
Thread 2: In critical section | Counter = 8
Thread 1: Waiting...
Thread 1: In critical section | Counter = 9
Thread 2: Waiting...
Thread 2: In critical section | Counter = 10
Final Counter Value: 10

```

Task2 : (Two Function)

```

1 #include <stdio.h>
2 #include <pthread.h>
3 #include <semaphore.h>
4 #include <unistd.h>
5
6 sem_t mutex; // Binary semaphore
7 int counter = 0;
8
9 // Thread that increments counter
10 void* increment_thread(void* arg) {
11     int id = *(int*)arg;
12
13     for (int i = 0; i < 5; i++) {
14         printf("Thread %d: Waiting to increment...\n", id);
15
16         sem_wait(&mutex); // acquire
17
18         counter++;
19         printf("Thread %d: Incremented | Counter = %d\n", id, counter);
20
21         sleep(1);
22         sem_post(&mutex); // release
23         sleep(1);
24     }
25     return NULL;
26 }
27
28 // Thread that decrements counter
29 void* decrement_thread(void* arg) {
30     int id = *(int*)arg;
31
32     for (int i = 0; i < 5; i++) {
33         printf("Thread %d: Waiting to decrement...\n", id);
34
35         sem_wait(&mutex); // acquire
36
37         counter--;
38         printf("Thread %d: Decrement | Counter = %d\n", id, counter);
39
40         sleep(1);
41         sem_post(&mutex); // release
42         sleep(1);
43     }
44     return NULL;
45 }
46
47 int main() {
48     sem_init(&mutex, 0, 1); // semaphore = 1
49
50     pthread_t t1, t2;
51     int id1 = 1, id2 = 2;
52
53     pthread_create(&t1, NULL, increment_thread, &id1);
54     pthread_create(&t2, NULL, decrement_thread, &id2);
55
56     pthread_join(t1, NULL);
57     pthread_join(t2, NULL);
58
59     printf("Final Counter Value: %d\n", counter);
60
61     sem_destroy(&mutex);
62     return 0;
63 }

```

```

File Edit Selection View Go Run Terminal Help < > Operating-system-1 [WSL: Ubuntu-24.04]
EXPLORER OPERATING-SYSTEM-1 [WSL: UBUNTU-24.04]
> 1142-assignment1
> 1142-evaluation
> 1142-lab1
> 1142-lab1(HW)
> 1142-lab2
> 1142-lab3
> 1142-lab3(HW)
> 1142-lab4
> 1142-lab5
> 1142-lab6
> 1142-lab9
C task1.c U task1.out C task2.c U x
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
ayem@DESKTOP-1VHCS6:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab9$ gcc task1.c -o task1.o
ut -Ipthread
11 |     int id = (int)arg;
task2.c: In function 'decrement_thread':
task2.c:30:14: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
30 |     int id = (int)arg;
|           ^
ayem@DESKTOP-1VHCS6:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab9$ gcc task2.c -o task2.o
ayem@DESKTOP-1VHCS6:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab9$ ./task2.out
Thread 1: Waiting to increment...
Thread 1: Incremented | Counter = 1
Thread 2: Waiting to decrement...
Thread 2: Decrement | Counter = 0
Thread 1: Waiting to increment...
Thread 1: Incremented | Counter = 1
Thread 2: Waiting to decrement...
Thread 2: Decrement | Counter = 0
Thread 1: Waiting to increment...
Thread 1: Incremented | Counter = 1
Thread 2: Waiting to decrement...
Thread 2: Decrement | Counter = 0
Thread 1: Waiting to increment...
Thread 1: Incremented | Counter = 1
Thread 2: Waiting to decrement...
Thread 2: Decrement | Counter = 0
Final Counter Value: 0
ayem@DESKTOP-1VHCS6:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab9$ ln 18, Col 15 Spaces: 4 UTF-8 LF C Finish Setup 21/11/2025

```

Description:

Changing semaphore value from 1 to 0 → both threads enter, race condition. They blocked and waiting for each other.

```

ayem@DESKTOP-1VHCS6:~/1142-lab3/Operating-system-1/1142-lab3/Operating-system/Operating-system/Operating-system-1/1142-lab9$ ./task2.out
Thread 1: Waiting to increment...
Thread 2: Waiting to decrement...

```

Task 3:

Comparison btw mutex and semaphore:

Feature	Mutex	Semaphore
Definition	Mutual exclusion object for one thread at a time	Signalling mechanism, can allow multiple threads
Count	Binary (locked/unlocked)	Can be counting (value > 1)
Ownership	Owned by the thread that locks it	Not owned by any thread; any thread can signal
Use case	Protect critical section for shared resources	Synchronize threads or processes, control access to N resources
Deadlock risk	Yes, if a thread never unlocks	Yes, if a semaphore is never posted
Performance	Lightweight, faster for thread-level locking	Slightly heavier, works across processes

Example	pthread_mutex_lock/unlock()	sem_wait/sem_post()
---------	-----------------------------	---------------------