



```
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;
33 }
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - lab 01 + □ ☰ ... | ☰ x

● abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ gcc Q1.c -o q1
● abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q1
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
○ abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$
```

DESCRIPTION:

1,0



```
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, 1, 0); // Binary semaphore initialized to 0
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;
33 }
```

A screenshot of a terminal window within a dark-themed IDE interface. The terminal tab is active, showing the following text:

```
18 | sleep(1);  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
abdel@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ gcc Q1.c -o q1  
abdel@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q1  
Thread 1: Waiting...  
Thread 2: Waiting...
```



```
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, 1, 0); // Binary semaphore initialized to 0
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;
33 }
```

The screenshot shows a terminal window with the following text output:

```
● abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ gcc Q1.c -o q1
○ abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q1
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 1: Waiting...
```



```
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 0
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;
33 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - lab 01 + × ☰

● abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ gcc Q1.c -o q1
● abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q1
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 2: Waiting...
Thread 2: In critical section | Counter = 5
Thread 1: Waiting...
Thread 1: In critical section | Counter = 6
Thread 2: Waiting...
Thread 2: In critical section | Counter = 7
Thread 1: Waiting...
Thread 1: 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
○ abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$
```

Task2:

```
● ○ ●
```

```
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 void* thread_function1(void* arg) {
23     int id = *(int*)arg;
24     for (int i = 0; i < 5; i++) {
25        printf("Thread %d: Waiting...\n", id);
26        //sem_wait(&mutex); // Acquire
27        // Critical section
28        counter--;
29        printf("Thread %d: In critical section | Counter = %d\n", id,
30        counter);
31        sleep(1);
32        sem_post(&mutex); // Release
33        sleep(1);
34    }
35    return NULL;
36 }
37 int main() {
38     sem_init(&mutex, 0, 1); // Binary semaphore initialized to 0
39     pthread_t t1, t2;
40     int id1 = 1, id2 = 2;
41     pthread_create(&t1, NULL, thread_function, &id1);
42     pthread_create(&t2, NULL, thread_function1, &id2);
43     pthread_join(t1, NULL);
44     pthread_join(t2, NULL);
45     printf("Final Counter Value: %d\n", counter);
46     sem_destroy(&mutex);
47     return 0;
48 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - lab 01 + x

● abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ gcc Q2.c -o q2
● abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q2
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Final Counter Value: 0
○ abdul@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ 
```



The screenshot shows a terminal window with the following details:

- Header:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS.
- Title Bar:** bash - lab 01
- Content:** The terminal displays the output of a C program. It shows two threads (Thread 1 and Thread 2) interacting with a shared counter. The threads alternately enter a critical section (indicated by '| Counter = 1') and wait (indicated by 'Waiting...'). The counter value fluctuates between -1, 0, and 1. The final counter value is printed as 0.

```
abdel@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$ ./q2
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = 0
Thread 1: Waiting...
Thread 1: In critical section | Counter = 1
Thread 2: Waiting...
Thread 2: In critical section | Counter = -1
Thread 1: Waiting...
Thread 1: In critical section | Counter = 0
Thread 2: Waiting...
Thread 2: In critical section | Counter = -1
Thread 1: Waiting...
Thread 1: In critical section | Counter = 0
Thread 2: Waiting...
Thread 2: In critical section | Counter = -1
Thread 1: Waiting...
Thread 1: In critical section | Counter = 0
Final Counter Value: 0
abdel@DESKTOP-N6RB9UV:~/Operating System/After mid/lab 01$
```

| Mutex | Semaphore |
|---|---|
| In mutex one thread can access the resource | In semaphore multiple threads can access the resources based on the count |
| Works like a lock | Works like a counter |
| has only two states : locked/unlocked | has a value that will be 0 and 1 |

| Mutex | Semaphore |
|--------------------------------------|---|
| Used for mutual exclusion | Used for controlling access to limited resources |
| Only the owner can release it | Anyone can signal (increase) it |