main.c

Share | Run

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
void* print_message(void* arg) {
    char* message = (char*)arg;
    for (int i = 0; i < 5; i++) {
        printf("%s: %d\n", message, i);
        sleep(1);
    }
    return NULL;
}

int main() {
    pthread_t thread1, thread2;

    pthread_create(&thread1, NULL, print_message, "Thread 1");
    pthread_create(&thread2, NULL, print_message, "Thread 2");
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    printf("Both threads finished execution.\n");
    return 0;
}
```

Output | Clear

```
Thread 1: 0
Thread 2: 0
Thread 2: 1
Thread 1: 1
Thread 2: 2
Thread 1: 2
Thread 2: 3
Thread 1: 3
Thread 2: 4
Thread 1: 4
Both threads finished execution.


=== Code Execution Successful ===
```

main.c

Share Run

Output Clear

```c
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#define N 5

sem_t C[N];
void* P(void* i) {
    int id = *(int*)i;
    sem_wait(&C[id]); sem_wait(&C[(id + 1) % N]);
    printf("Philosopher %d is eating\n", id);
    sem_post(&C[id]); sem_post(&C[(id + 1) % N]);
    return NULL;
}
int main() {
    pthread_t T[N]; int ID[N];
    for (int i = 0; i < N; i++) sem_init(&C[i], 0, 1);
    for (int i = 0; i < N; i++) { ID[i] = i; pthread_create(&T[i], NULL, P,
        &ID[i]); }
    for (int i = 0; i < N; i++) pthread_join(T[i], NULL);
    return 0;
}

```

```
Philosopher 0 is eating
Philosopher 1 is eating
Philosopher 3 is eating
Philosopher 2 is eating
Philosopher 4 is eating


=== Code Execution Successful ===
```