main.c

Share Run

```c
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
void* print_numbers(void* arg) {
    int i;
    for (i = 1; i <= 5; i++) {
        printf("Thread 1: %d\n", i);
        sleep(1);}
    return NULL;}
void* print_letters(void* arg) {
    char ch;
    for (ch = 'A'; ch <= 'E'; ch++) {
        printf("Thread 2: %c\n", ch);
        sleep(1);}
    return NULL;}
int main() {
    pthread_t thread1, thread2;
    if (pthread_create(&thread1, NULL, print_numbers, NULL)) {
        fprintf(stderr, "Error creating thread 1\n");
        return 1;}
    if (pthread_create(&thread2, NULL, print_letters, NULL)) {
        fprintf(stderr, "Error creating thread 2\n");
        return 1;}
    if (pthread_join(thread1, NULL)) {
        fprintf(stderr, "Error joining thread 1\n");
        return 1;}
    if (pthread_join(thread2, NULL)) {
        fprintf(stderr, "Error joining thread 2\n");
        return 1;}
    printf("Both threads finished execution\n");

```

Output

Clear

```
Thread 1: 1
Thread 2: A
Thread 1: 2
Thread 2: B
Thread 1: 3
Thread 2: C
Thread 1: 4
Thread 2: D
Thread 1: 5
Thread 2: E
Both threads finished execution


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

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_PHILOSOPHERS 4
#define NUM_EATS 2
pthread_mutex_t forks[NUM_PHILOSOPHERS];
void think(int philosopher) {
    printf("Philosopher %d is thinking.\n", philosopher);
    sleep(rand() % 3 + 1);}
void eat(int philosopher) {
    printf("Philosopher %d is eating.\n", philosopher);
    sleep(rand() % 3 + 1);}
void* philosopher_action(void* arg) {
    int philosopher = *((int*)arg);
    for (int i = 0; i < NUM_EATS; i++) {
        think(philosopher);
        pthread_mutex_lock(&forks[philosopher]);
        pthread_mutex_lock(&forks[(philosopher + 1) % NUM_PHILOSOPHERS]);
        eat(philosopher);
        pthread_mutex_unlock(&forks[(philosopher + 1) % NUM_PHILOSOPHERS]);
        pthread_mutex_unlock(&forks[philosopher]);}
    return NULL;}
int main() {
    pthread_t threads[NUM_PHILOSOPHERS];
    int philosophers[NUM_PHILOSOPHERS];
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_mutex_init(&forks[i], NULL);}
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        philosophers[i] = i;
        if (pthread_create(&threads[i], NULL, philosopher_action, &philosophers[i]) != 0) {
            perror("Failed to create thread");
            return 1;}}
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_join(threads[i], NULL);}
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_mutex_destroy(&forks[i]);}
    return 0;}
```

Output

```
Philosopher 0 is thinking.
Philosopher 1 is thinking.
Philosopher 2 is thinking.
Philosopher 3 is thinking.
Philosopher 2 is eating.
Philosopher 0 is eating.
Philosopher 0 is thinking.
Philosopher 2 is thinking.
Philosopher 1 is eating.
Philosopher 3 is eating.
Philosopher 1 is thinking.
Philosopher 3 is thinking.
Philosopher 0 is eating.
Philosopher 2 is eating.
Philosopher 3 is eating.
Philosopher 1 is eating.


--- Code Execution Successful ---
```