



# National Textile University

## **Department of Computer Science**

Subject: Operating System

---

Submitted to: Sir Nasir

---

Submitted by: Maha

---

Reg. number: 23-NTU-CS-1170

---

Lab 4 (Home task)

---

Semester: 5<sup>th</sup>

## Lab 4 Home Task

---

Question 1-5 in class task

### 5. Hands-on Practice Exercises

#### Exercise 1: Thread Basics

Write a program that:

1. Creates 3 threads
2. Each thread prints its thread ID and a unique message
3. Main thread waits for all threads to complete

Code:

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

void* print_message(void* arg) {
    int thread_num = *(int*)arg;
    pthread_t tid = pthread_self();

    printf("[Thread %d] ID: %lu - Hello from thread %d!\n", thread_num, tid,
thread_num);

    sleep(1);
    printf("[Thread %d] Finished!\n", thread_num);
    return NULL;
}
```

```
int main() {
    pthread_t threads[3];
    int thread_nums[3] = {1, 2, 3};

    printf("Main: creating 3 threads...\n");

    for (int i = 0; i < 3; i++) {
        pthread_create(&threads[i], NULL, print_message, &thread_nums[i]);
        printf("Main: started thread %d\n", i + 1);
    }

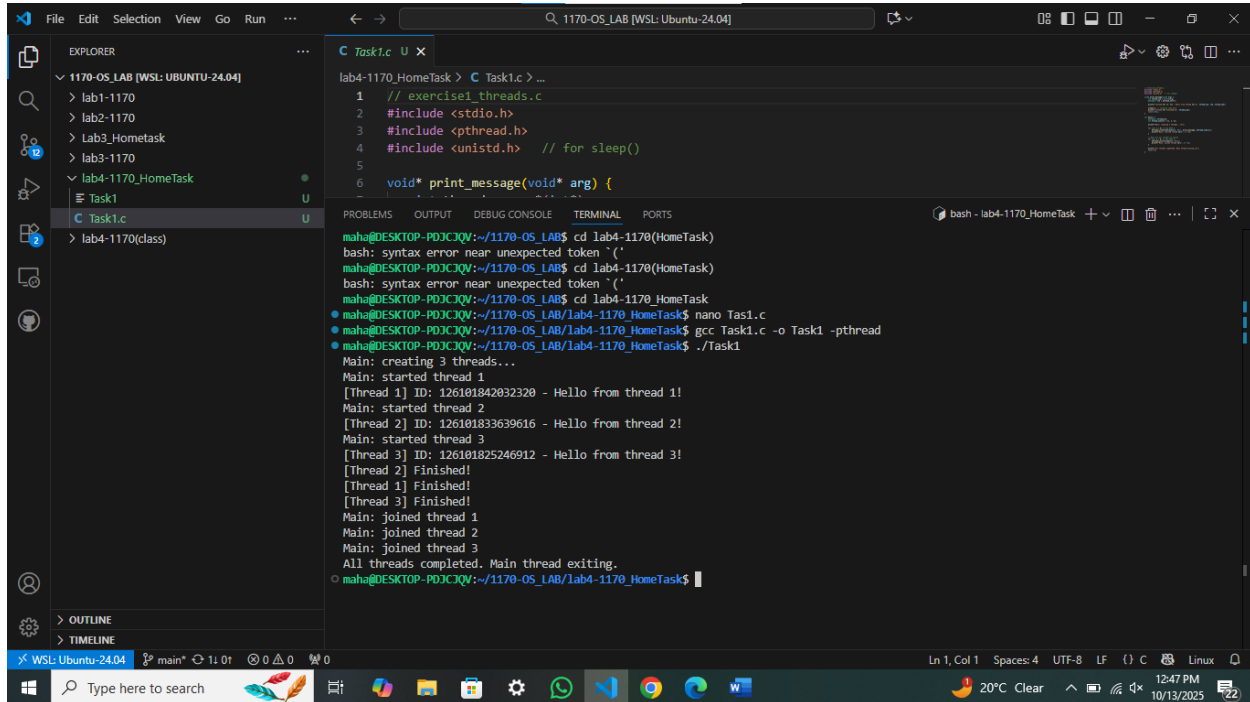
    for (int i = 0; i < 3; i++) {
        pthread_join(threads[i], NULL);
        printf("Main: joined thread %d\n", i + 1);
    }

    printf("All threads completed. Main thread exiting.\n");
    return 0;
}
```

Execution:

- Command:
  - Gcc task1.c -o task1 -pthread
  - ./task1

## Terminal:



```
lab4-1170_HomeTask > C Task1.c > ...
1 // exercise1_threads.c
2 #include <stdio.h>
3 #include <pthread.h>
4 #include <unistd.h> // for sleep()
5
6 void* print_message(void* arg) {
    ...
}

maha@DESKTOP-PD3CJQV:~/1170-OS_LAB$ cd lab4-1170(HomeTask)
bash: syntax error near unexpected token `('
maha@DESKTOP-PD3CJQV:~/1170-OS_LAB$ cd lab4-1170(HomeTask)
bash: syntax error near unexpected token `('
maha@DESKTOP-PD3CJQV:~/1170-OS_LAB$ cd lab4-1170_HomeTask
maha@DESKTOP-PD3CJQV:~/1170-OS_LAB/lab4-1170_HomeTask$ nano Tas1.c
maha@DESKTOP-PD3CJQV:~/1170-OS_LAB/lab4-1170_HomeTask$ gcc Task1.c -o Task1 -pthread
maha@DESKTOP-PD3CJQV:~/1170-OS_LAB/lab4-1170_HomeTask$ ./Task1
Main: creating 3 threads...
Main: started thread 1
[Thread 1] ID: 126101842032320 - Hello from thread 1!
Main: started thread 2
[Thread 2] ID: 126101833639616 - Hello from thread 2!
Main: started thread 3
[Thread 3] ID: 126101825246912 - Hello from thread 3!
[Thread 2] Finished!
[Thread 1] Finished!
[Thread 3] Finished!
Main: joined thread 1
Main: joined thread 2
Main: joined thread 3
All threads completed. Main thread exiting.
maha@DESKTOP-PD3CJQV:~/1170-OS_LAB/lab4-1170_HomeTask$
```

## Exercise 2: Prime Number Checker

Write a program that:

1. Takes a number as input
2. Creates a thread that checks if the number is prime
3. Returns the result to the main thread
4. Main thread prints whether the number is prime or not

Code:

```
#include <stdio.h>

#include <pthread.h>

#include <stdbool.h>

#include <math.h>

typedef struct {
```

```
    int number;
    bool is_prime;
} PrimeData;

void* check_prime(void* arg) {
    PrimeData* data = (PrimeData*)arg;
    int n = data->number;

    if (n <= 1) {
        data->is_prime = false;
        return NULL;
    }

    data->is_prime = true;
    for (int i = 2; i <= sqrt(n); i++) {
        if (n % i == 0) {
            data->is_prime = false;
            break;
        }
    }

    return NULL;
}
```

```
int main() {  
    PrimeData data;  
    pthread_t thread;  
  
    printf("Enter a number to check for prime: ");  
    scanf("%d", &data.number);  
  
    pthread_create(&thread, NULL, check_prime, &data);  
    pthread_join(thread, NULL);  
  
    if (data.is_prime)  
        printf("%d is a prime number.\n", data.number);  
    else  
        printf("%d is NOT a prime number.\n", data.number);  
  
    return 0;  
}
```

Execution:

- Command:
  - Gcc task2.c -o task2 -pthread
  - ./task2

Terminal:

