



# National Textile University

*Department of Computer Science*

**Subject:**

Operating system

**Submitted To:**

Sir Nasir

**Submitted By:**

Noor ul Ain

**Registration No:**

23-NTU-CS-1221

**Lab No:**

4 -Home task

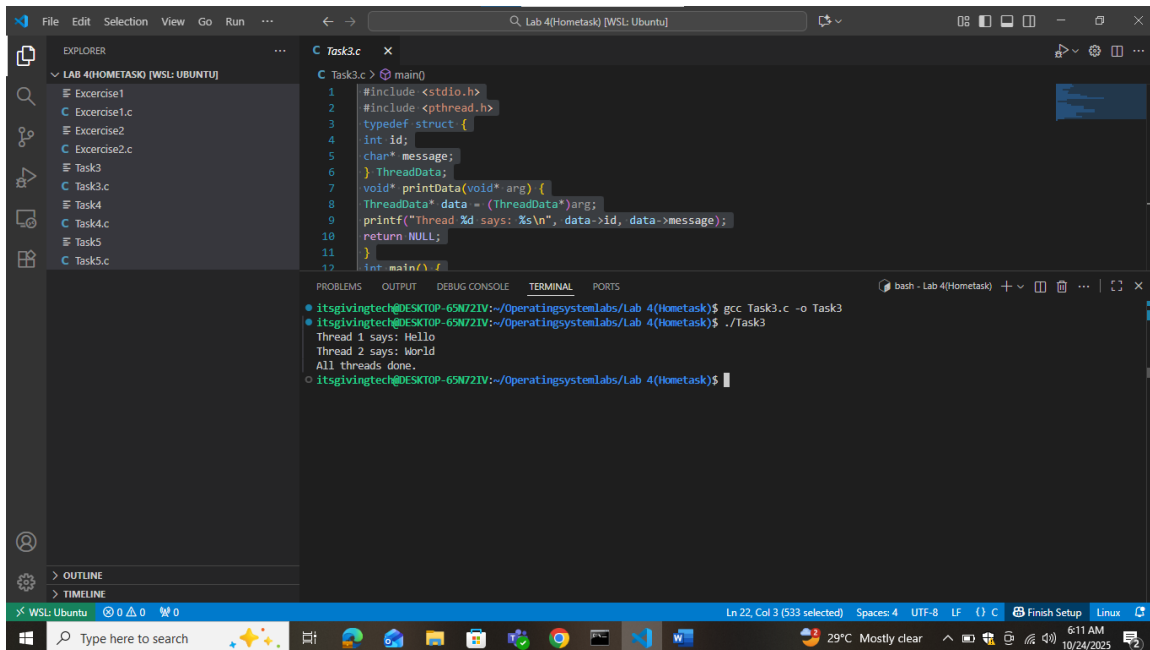
**Semester:**

## Task 3:

### Program 3: Passing Multiple Data

```
#include <stdio.h>
#include <pthread.h>
typedef struct {
    int id;
    char* message;
} ThreadData;
void* printData(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    printf("Thread %d says: %s\n", data->id, data->message);
    return NULL;
}
int main() {
    pthread_t t1, t2;
    ThreadData data1 = {1, "Hello"};
    ThreadData data2 = {2, "World"};
    pthread_create(&t1, NULL, printData, &data1);
    pthread_create(&t2, NULL, printData, &data2);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    printf("All threads done.\n");
    return 0;
}
```

Output:



The screenshot shows a code editor with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a directory named 'LAB 4(HOMETASK)' with files 'Exercise1.c', 'Exercise2.c', 'Task3.c', 'Task4.c', and 'Task5.c'. The code editor shows the C program from the previous block. The terminal shows the output of the program:

```
itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 4(Hometask)$ gcc Task3.c -o Task3
itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 4(Hometask)$ ./Task3
Thread 1 says: Hello
Thread 2 says: World
All threads done.
itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 4(Hometask)$
```

## Task 4:

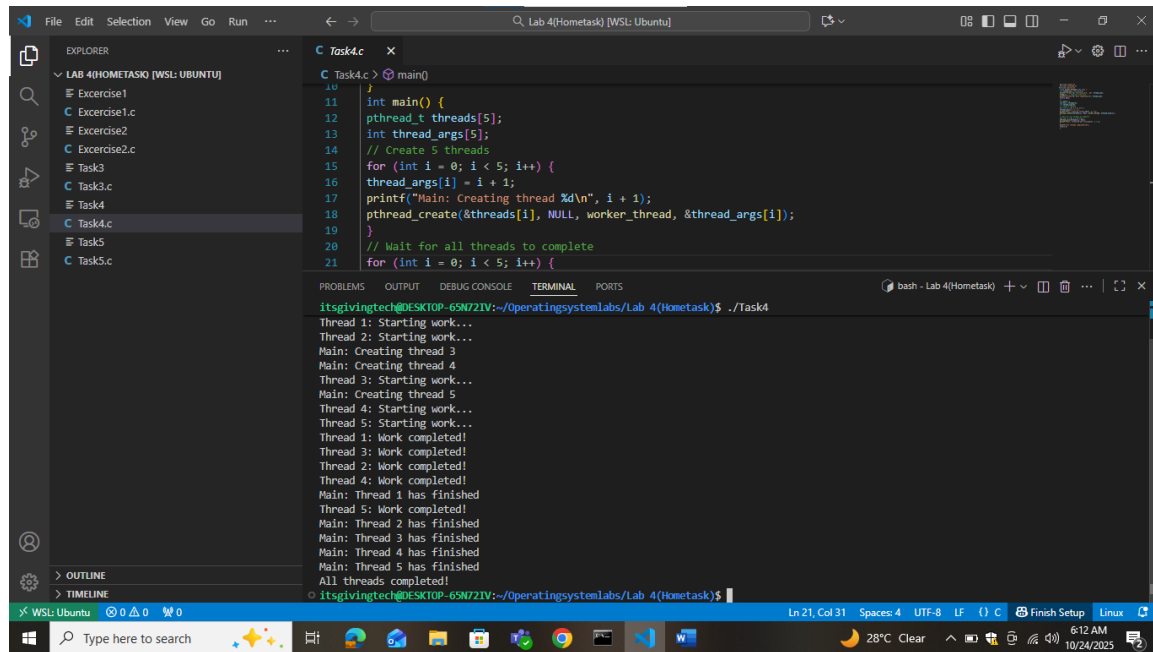
### Program 4: Multiple Threads

#### Code:

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
void* worker_thread(void* arg) {
    int thread_num = *(int*)arg;
    printf("Thread %d: Starting work...\n", thread_num);
    sleep(1);
    printf("Thread %d: Work completed!\n", thread_num);
    return NULL;
}
int main() {
    pthread_t threads[5];
    int thread_args[5];
    // Create 5 threads
    for (int i = 0; i < 5; i++) {
        thread_args[i] = i + 1;
        printf("Main: Creating thread %d\n", i + 1);
        pthread_create(&threads[i], NULL, worker_thread, &thread_args[i]);
    }

    for (int i = 0; i < 5; i++) {
        pthread_join(threads[i], NULL);
        printf("Main: Thread %d has finished\n", i + 1);
    }
    printf("All threads completed!\n");
    return 0;
}
```

## Output:



The screenshot shows the Visual Studio Code interface with a C file named `Task4.c` open. The code defines a `main` function that creates 5 threads, each performing a simple calculation. The output window shows the execution results, including thread creation and completion messages.

```
10 }
11 int main() {
12     pthread_t threads[5];
13     int thread_args[5];
14     // Create 5 threads
15     for (int i = 0; i < 5; i++) {
16         thread_args[i] = i + 1;
17         printf("Main: Creating thread %d\n", i + 1);
18         pthread_create(&threads[i], NULL, worker_thread, &thread_args[i]);
19     }
20     // Wait for all threads to complete
21     for (int i = 0; i < 5; i++) {
```

Output:

```
itsgivingtech@DESKTOP-69W72IV:~/OperatingSystemLabs/Lab 4(Hometask)$ ./Task4
Thread 1: Starting work...
Thread 2: Starting work...
Main: Creating thread 3
Main: Creating thread 4
Thread 3: Starting work...
Main: Creating thread 5
Thread 4: Starting work...
Thread 5: Starting work...
Thread 1: Work completed!
Thread 3: Work completed!
Thread 2: Work completed!
Thread 4: Work completed!
Main: Thread 1 has finished
Thread 5: Work completed!
Main: Thread 2 has finished
Main: Thread 3 has finished
Main: Thread 4 has finished
Main: Thread 5 has finished
All threads completed!
```

## Task 5:

### Program 5: Thread Return Values

## Code:

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
void* calculate_sum(void* arg) {
    int n = *(int*)arg;
    int* result = malloc(sizeof(int));
    *result = 0;
    for (int i = 1; i <= n; i++) {
        *result += i;
    }
    printf("Thread calculated sum of 1 to %d = %d\n", n, *result);
    return (void*)result;
}
int main() {
    pthread_t thread_id;
    int n = 100;
    void* sum;
    pthread_create(&thread_id, NULL, calculate_sum, &n);
    pthread_join(thread_id, &sum);
    printf("Main received result: %d\n", *(int*)sum);
}
```

```

free(sum);
return 0;
}

```

Output:

The screenshot shows a Visual Studio Code editor window titled 'Lab 4(Hometask) [WSL: Ubuntu]'. The Explorer panel on the left shows a file tree with 'LAB 4(HOMETASK) [WSL: UBUNTU]' containing 'Exercise1', 'Exercise1.c', 'Exercise2', 'Exercise2.c', 'Task3', 'Task3.c', 'Task4', 'Task4.c', 'Task5', and 'Task5.c'. The main editor area shows the code for 'Task5.c':

```

1 #include <stdio.h>
2 #include <pthread.h>
3 #include <stdlib.h>
4 void* calculate_sum(void* arg) {
5     int n = *(int*)arg;
6     int* result = malloc(sizeof(int)); // Allocate memory for result
7     *result = 0;
8     for (int i = 1; i <= n; i++) {
9         *result += i;
10    }
11    printf("Thread calculated sum of 1 to %d = %d\n", n, *result);
12    return (void*)result; // Return the result

```

The bottom panel shows the 'TERMINAL' output:

```

itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 4(Hometask)$ gcc Task5.c -o Task5
itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 4(Hometask)$ ./Task5
Thread calculated sum of 1 to 100 = 5050
Main received result: 5050
itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 4(Hometask)$

```

## Exercise 1:

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>

typedef struct {
    int id;
    char *message;
} ThreadData;

void* printData(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    printf("Thread %d says: %s\n", data->id, data->message);
    return NULL;
}

```

```

int main() {
    pthread_t threads[3];
    ThreadData data[3] = {
        {1, "Hello from thread 1!"},
        {2, "Greetings from thread 2!"},
        {3, "Hi there from thread 3!"}
    };

    for (int i = 0; i < 3; i++) {
        pthread_create(&threads[i], NULL, printData, &data[i]);
    }
    for (int i = 0; i < 3; i++) {
        pthread_join(threads[i], NULL);
    }

    printf("All threads have finished.\n");
    return 0;
}

```

## Output:

The screenshot shows a Visual Studio Code editor window with a C program in a file named `Exercise1.c`. The program creates three threads, each printing a message. The terminal output shows the execution of the program, including the compilation of `Task5.c` and the execution of `Exercise1`. The output of the program is:

```

Thread 1 says: Hello from thread 1!
Thread 2 says: Greetings from thread 2!
Thread 3 says: Hi there from thread 3!
All threads have finished.

```

The terminal also shows the command prompt and the execution of the program:

```

itsgivingtech@DESKTOP-69N72IV:~/OperatingSystemLabs/Lab 4(Homework)$ gcc Task5.c -o Task5
itsgivingtech@DESKTOP-69N72IV:~/OperatingSystemLabs/Lab 4(Homework)$ ./Task5
Main received result: 5850
itsgivingtech@DESKTOP-69N72IV:~/OperatingSystemLabs/Lab 4(Homework)$ ./Exercise1

```

## **Exercise 2:**

### **Prime Number Checker:**

- 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>

typedef struct {
    int number;
    bool isPrime;
} ThreadData;

void* checkPrime(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    int n = data->number;

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

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

    return NULL;
}

int main() {
    pthread_t thread;
```

```

    ThreadData data;

    printf("Enter a number: ");
    scanf("%d", &data.number);

    pthread_create(&thread, NULL, checkPrime, &data);

    pthread_join(thread, NULL);

    if (data.isPrime)
        printf("%d is a prime number.\n", data.number);
    else
        printf("%d is not a prime number.\n", data.number);

    return 0;
}

```

Output:

The screenshot shows a Visual Studio Code editor window with a file explorer on the left displaying a project structure under 'LAB 4(HOMETASK) [WSL: UBUNTU]'. The main editor area shows a C file named 'Exercise2.c' with the following code:

```

32 int main() {
40
41     pthread_create(&thread, NULL, checkPrime, &data);
42
43     pthread_join(thread, NULL);
44
45
46     if (data.isPrime)
47         printf("%d is a prime number.\n", data.number);
48     else
49         printf("%d is not a prime number.\n", data.number);
50

```

Below the code editor is a terminal window titled 'bash - Lab 4(Hometask)'. It shows the execution of the program:

```

itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 4(Hometask)$ ./Exercise2
Enter a number: 4
4 is not a prime number.
itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 4(Hometask)$

```

The status bar at the bottom indicates the file is 'Ln 53, Col 1 (874 selected)' in 'UTF-8' encoding, and the system clock shows '6:17 AM 10/24/2025'.