



National Textile University

Department of Computer Science

Subject:

Operating System

Submitted to:

Sir Nasir Mehmood

Submitted by:

Eman Marium Tariq Rao

Reg number:

23-NTU-CS-1150

Semester:

05

3. C Programs with Threads

Program 1: Creating a Simple Thread

```
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

void* thread_function(void* arg) {
    (void)arg; // unused here
    printf("Hello from the new thread!\n");
    /* pthread_self() returns a pthread_t; cast to unsigned long for printing */
    printf("New thread ID: %lu\n", (unsigned long)pthread_self());
    return NULL;
}

int main(void) {
    pthread_t thread_id;
    int rc;

    printf("Main thread starting...\n");
    printf("Main thread ID: %lu\n", (unsigned long)pthread_self());

    /* Create the new thread:
        - &thread_id: where the created thread's ID is stored
        - NULL: default thread attributes
        - thread_function: function to run in new thread
        - NULL: argument passed to thread_function */
    rc = pthread_create(&thread_id, NULL, thread_function, NULL);
    if (rc != 0) {
```

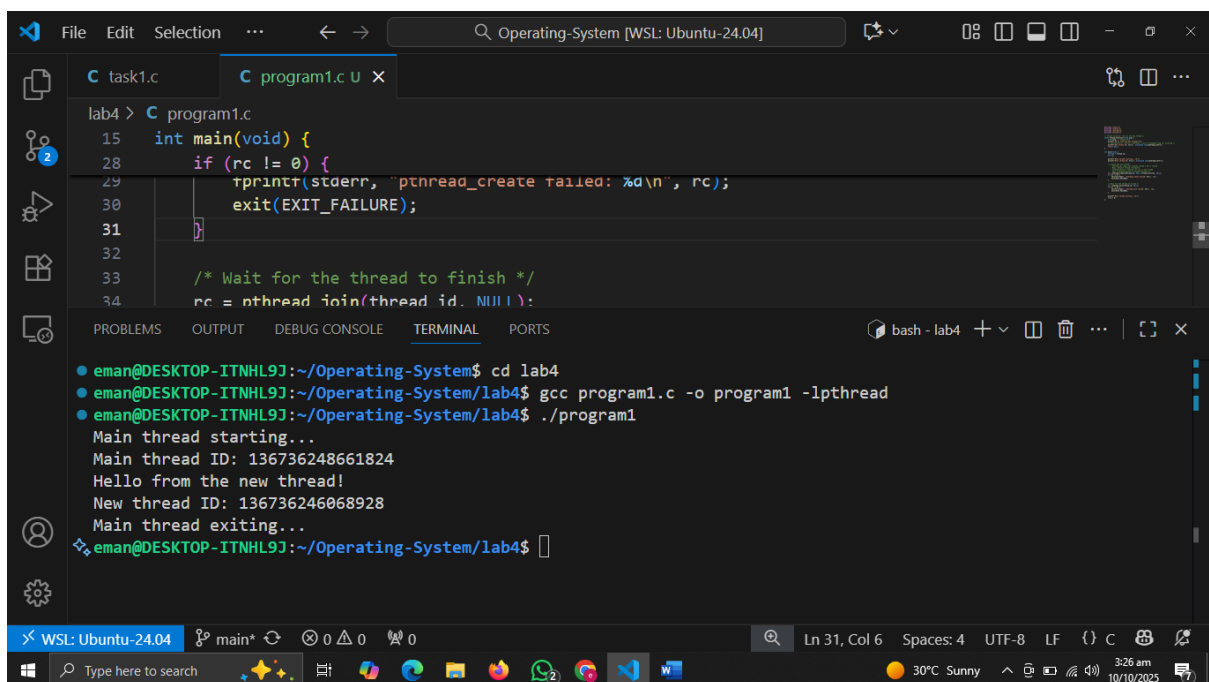
```

    fprintf(stderr, "pthread_create failed: %d\n", rc);
    exit(EXIT_FAILURE);
}

/* Wait for the thread to finish */
rc = pthread_join(thread_id, NULL);
if (rc != 0) {
    fprintf(stderr, "pthread_join failed: %d\n", rc);
    exit(EXIT_FAILURE);
}

printf("Main thread exiting...\n");
return 0;
}

```



The screenshot shows a Visual Studio Code editor window with a file named `program1.c` open. The code in the editor is a C program that demonstrates thread creation and joining. The terminal window at the bottom shows the execution of the program, including the compilation with `gcc` and the output of the program, which prints the main thread ID, the new thread ID, and the main thread exiting message.

```

lab4 > C program1.c
15  int main(void) {
28      if (rc != 0) {
29          fprintf(stderr, "pthread_create failed: %d\n", rc);
30          exit(EXIT_FAILURE);
31      }
32
33      /* Wait for the thread to finish */
34      rc = pthread_join(thread_id, NULL);

```

```

bash - lab4
● eman@DESKTOP-ITNHL9J:~/Operating-System$ cd lab4
● eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ gcc program1.c -o program1 -lpthread
● eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ ./program1
Main thread starting...
Main thread ID: 136736248661824
Hello from the new thread!
New thread ID: 136736246068928
Main thread exiting...
● eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$

```

Program 2: Passing Arguments to Threads

```
#include <stdio.h>

#include <pthread.h>

// Thread function — runs in the new thread
void* print_number(void* arg) {
    // Convert the void pointer back to an int pointer, then get the value
    int num = *(int*)arg;
    printf("Thread received number: %d\n", num);
    printf("Square: %d\n", num * num);
    return NULL;
}

int main() {
    pthread_t thread_id;
    int number = 42;

    printf("Creating thread with argument: %d\n", number);

    // Create thread and pass address of 'number'
    pthread_create(&thread_id, NULL, print_number, &number);

    // Wait for the thread to finish
    pthread_join(thread_id, NULL);

    printf("Main thread done.\n");
    return 0;
}
```

The screenshot shows a Visual Studio Code editor window with the title bar 'Operating-System [WSL: Ubuntu-24.04]'. The editor has three tabs: 'task1.c', 'program1.c U', and 'program2.c U'. The 'program2.c' tab is active, showing the following code:

```
lab4 > C program2.c
13 int main() {
14     int number = 42;
15     pthread_t thread_id;
16     printf("Creating thread with argument: %d\n", number);
17
18     // Create thread and pass address of 'number'
19     pthread_create(&thread_id, NULL, print_number, &number);
20
21     // Wait for thread to finish
22     pthread_join(thread_id, NULL);
23
24     printf("Main thread done.\n");
25     return 0;
26 }
```

The bottom panel shows the 'TERMINAL' output:

```
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ touch program2.c
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ gcc program2.c -o program2 -lpthread
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ ./program2
Creating thread with argument: 42
Thread received number: 42
Square: 1764
Main thread done.
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$
```

The status bar at the bottom indicates 'WSL: Ubuntu-24.04', 'main*' (with a refresh icon), 'Ln 28, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and 'C'.

Program 3: Passing Multiple Data

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

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

```
typedef struct {
```

```
    int id;
```

```
    char* message;
```

```
} ThreadData;
```

```
// Thread function — receives and prints the data
```

```
void* printData(void* arg) {
```

```
    // Convert void* back to ThreadData*
```

```
    ThreadData* data = (ThreadData*)arg;
```

```
    printf("Thread %d says: %s\n", data->id, data->message);
```

```
    return NULL;
```

```
}
```

```

int main() {

    pthread_t t1, t2;


    // Initialize data for each thread
    ThreadData data1 = {1, "Hello"};
    ThreadData data2 = {2, "World"};


    // Create two threads and pass data to each
    pthread_create(&t1, NULL, printData, &data1);
    pthread_create(&t2, NULL, printData, &data2);

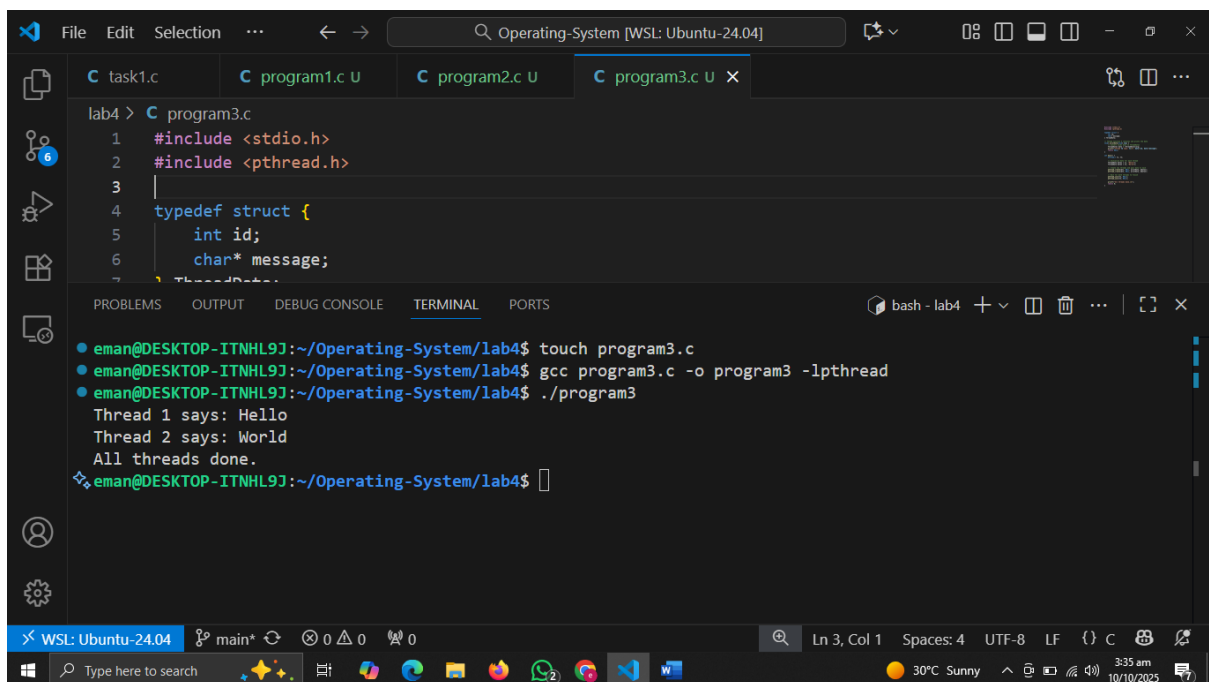

    // Wait for both threads to finish
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);


    printf("All threads done.\n");

    return 0;

}

```



The screenshot shows a Windows Subsystem for Linux (WSL) terminal window. The terminal displays the following commands and output:

```

lab4 > C program3.c
1  #include <stdio.h>
2  #include <pthread.h>
3
4  typedef struct {
5      int id;
6      char* message;
7  } ThreadData;
8
9  void* printData(void* arg) {
10     ThreadData* data = (ThreadData*) arg;
11     printf("Thread %d says: %s\n", data->id, data->message);
12 }
13
14 int main() {
15     pthread_t t1, t2;
16     ThreadData data1 = {1, "Hello"};
17     ThreadData data2 = {2, "World"};
18
19     pthread_create(&t1, NULL, printData, &data1);
20     pthread_create(&t2, NULL, printData, &data2);
21
22     pthread_join(t1, NULL);
23     pthread_join(t2, NULL);
24
25     printf("All threads done.\n");
26
27     return 0;
28 }

```

The terminal output shows the successful compilation and execution of the program:

```

eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ touch program3.c
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ gcc program3.c -o program3 -lpthread
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ ./program3
Thread 1 says: Hello
Thread 2 says: World
All threads done.
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$

```

The terminal window also shows the file explorer on the left with the following files:

- task1.c
- program1.c U
- program2.c U
- program3.c U X

The status bar at the bottom of the terminal window indicates the current file is "main*", the cursor is at "Ln 3, Col 1", and the encoding is "UTF-8". The system tray at the bottom shows the date and time as "3:35 am 10/10/2025".

Program 4: Multiple Threads

```
#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]; // Array to store 5 thread IDs
    int thread_args[5]; // Arguments for each thread

    // Create 5 threads
    for (int i = 0; i < 5; i++) {
        thread_args[i] = i + 1; // Thread number starts from 1
        printf("Main: Creating thread %d\n", i + 1);
        pthread_create(&threads[i], NULL, worker_thread, &thread_args[i]);
    }

    // Wait for all threads to complete
    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;
}

```

```

eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ touch program4.c
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ gcc program4.c -o program4 -lpthread
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ ./program4
Main: Creating thread 1
Main: Creating thread 2
Thread 1: Starting work...
Main: Creating thread 3
Thread 2: Starting work...
Main: Creating thread 4
Thread 3: Starting work...
Main: Creating thread 5
Thread 4: Starting work...
Thread 5: Starting work...
Thread 2: Work completed!
Thread 3: Work completed!
Thread 4: Work completed!
Thread 5: Work completed!
Thread 1: Work completed!
Main: Thread 1 has finished
Main: Thread 2 has finished
Main: Thread 3 has finished
Main: Thread 4 has finished
Main: Thread 5 has finished
All threads completed!
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$

```

Program 5: Thread Return Values

```

#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;

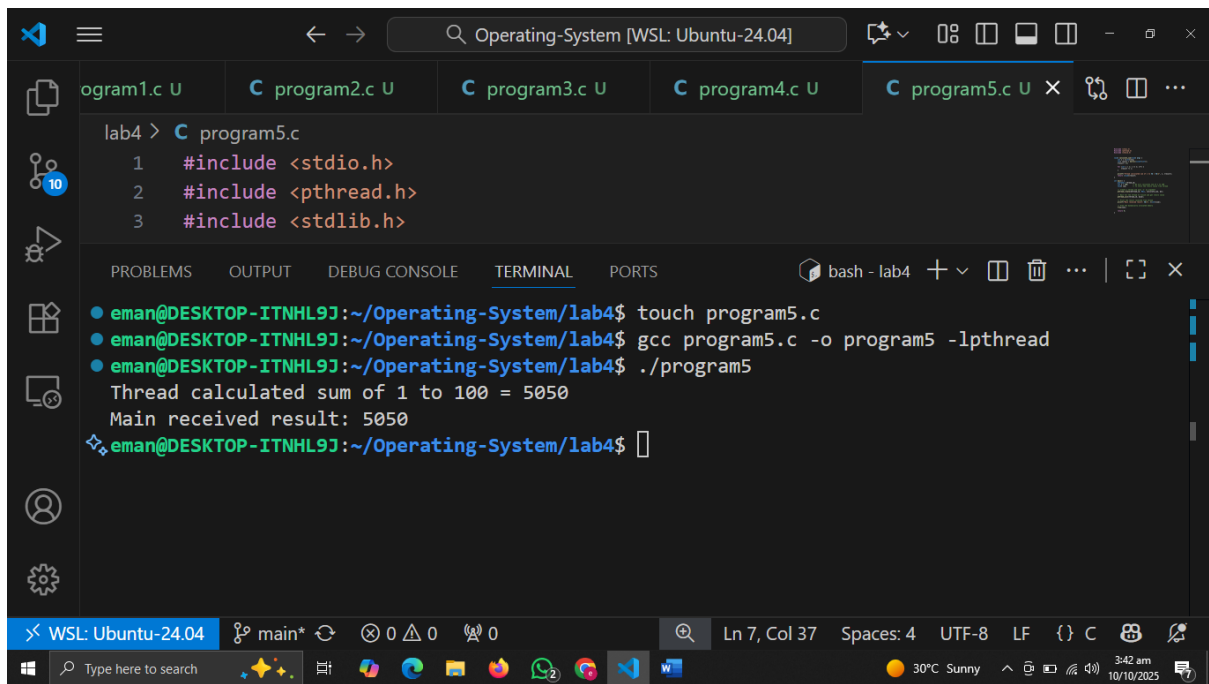
    for (int i = 1; i <= n; i++) {
        *result += i;
    }
}

```



```
    printf("Thread calculated sum of 1 to %d = %d\n", n, *result);  
    return (void*)result;  
}
```

```
int main() {  
    pthread_t thread_id;  
    int n = 100;    // We will calculate sum of 1 to 100  
    void* sum;      // To store the return value from thread  
  
    // Create a thread and pass 'n' as argument  
    pthread_create(&thread_id, NULL, calculate_sum, &n);  
  
    // Wait for the thread to finish and get return value  
    pthread_join(thread_id, &sum);  
  
    // Print the result received from thread  
    printf("Main received result: %d\n", *(int*)sum);  
  
    // Free the dynamically allocated memory  
    free(sum);  
  
    return 0;  
}
```



```
lab4 > C program5.c
1  #include <stdio.h>
2  #include <pthread.h>
3  #include <stdlib.h>

eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ touch program5.c
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ gcc program5.c -o program5 -lpthread
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ ./program5
Thread calculated sum of 1 to 100 = 5050
Main received result: 5050
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$
```

5. Hands-on Practice Exercises

Exercise 1: Thread Basics

Write a program that:

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

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

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

```
#include <unistd.h> // For sleep()
```

```
// Thread function
```

```
void* print_message(void* arg) {
```

```
    char* message = (char*)arg; // Receive string argument
```

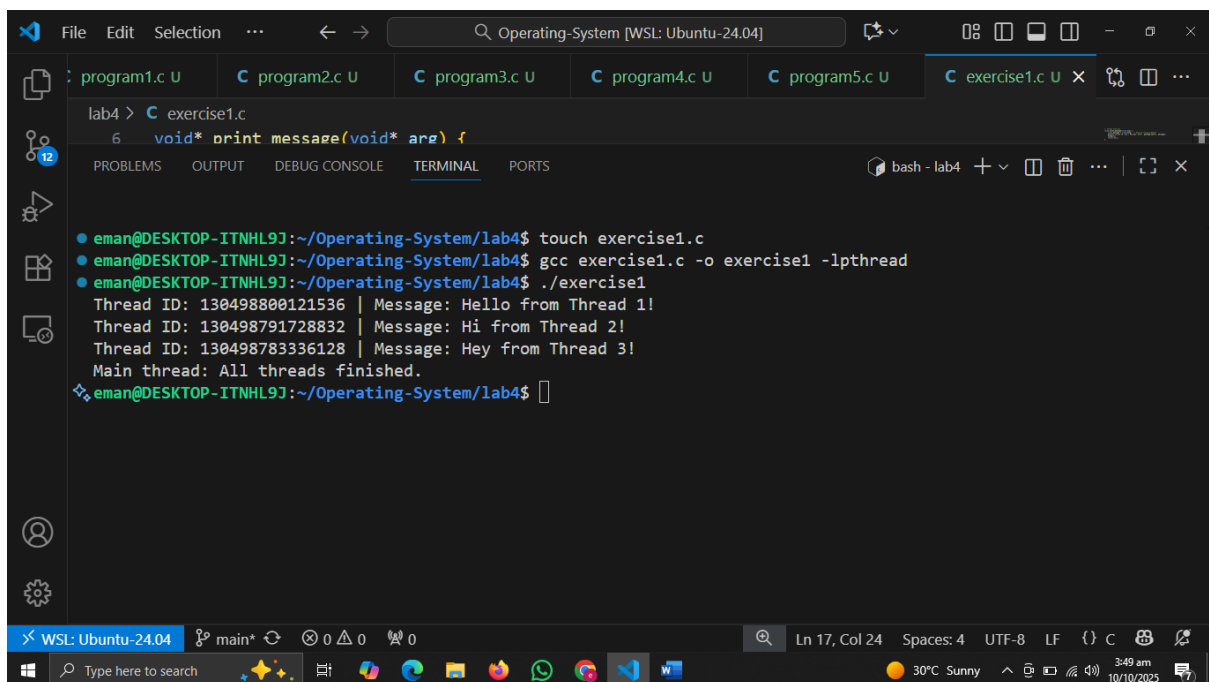
```
    printf("Thread ID: %lu | Message: %s\n", pthread_self(), message);
```

```
    sleep(1);
```

```
    return NULL;
```

```
}
```

```
int main() {  
    pthread_t threads[3];  
    char* messages[3] = {"Hello from Thread 1!", "Hi from Thread 2!", "Hey from Thread 3!"};  
  
    // Create 3 threads  
    for (int i = 0; i < 3; i++) {  
        pthread_create(&threads[i], NULL, print_message, messages[i]);  
    }  
  
    // Wait for all threads to finish  
    for (int i = 0; i < 3; i++) {  
        pthread_join(threads[i], NULL);  
    }  
    printf("Main thread: All threads finished.\n");  
    return 0;  
}
```



The screenshot shows a Windows Subsystem for Linux (WSL) terminal window titled "Operating-System [WSL: Ubuntu-24.04]". The terminal displays the execution of a C program named "exercise1.c". The program creates three threads, each printing a message: "Hello from Thread 1!", "Hi from Thread 2!", and "Hey from Thread 3!". The main thread then prints "Main thread: All threads finished." before returning 0. The terminal output shows the thread IDs and messages for each thread, confirming that all threads completed successfully.

```
lab4 > C exercise1.c  
6 void* print_message(void* arg) {  
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ touch exercise1.c  
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ gcc exercise1.c -o exercise1 -lpthread  
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ ./exercise1  
Thread ID: 130498800121536 | Message: Hello from Thread 1!  
Thread ID: 130498791728832 | Message: Hi from Thread 2!  
Thread ID: 130498783336128 | Message: Hey from Thread 3!  
Main thread: All threads finished.  
eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$
```

Exercise 2: Prime Number Checker

Write a program that:

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

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

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

```
#include <stdlib.h>
```

```
#include <stdbool.h>
```

```
void* check_prime(void* arg) {
```

```
    int n = *(int*)arg;
```

```
    bool* is_prime = malloc(sizeof(bool));
```

```
    *is_prime = true;
```

```
    if (n <= 1)
```

```
        *is_prime = false;
```

```
    else {
```

```
        for (int i = 2; i * i <= n; i++) {
```

```
            if (n % i == 0) {
```

```
                *is_prime = false;
```

```
                break;
```

```
            }
```

```
        }
```

```
    }
```

```
    printf("Thread checked number: %d\n", n);
```

```

    return (void*)is_prime; // Return pointer to result
}

int main() {
    pthread_t thread_id;
    int num;
    void* result;

    printf("Enter a number: ");
    scanf("%d", &num);
    // Create thread to check prime
    pthread_create(&thread_id, NULL, check_prime, &num);

    // Wait for thread to finish and get result
    pthread_join(thread_id, &result);
    bool is_prime = *(bool*)result;
    if (is_prime)
        printf("%d is a prime number.\n", num);
    else
        printf("%d is NOT a prime number.\n", num);
    free(result);
    return 0;
}

```

The screenshot shows a Visual Studio Code editor window with the title bar "Operating-System [WSL: Ubuntu-24.04]". The editor has several tabs open: "program3.c U", "C program4.c U", "C program5.c U", "C exercise1.c U", and "C exercise2.c U". The active tab is "C exercise2.c U", which contains the following code:

```
lab4 > C exercise2.c
1 #include <stdio.h>
```

Below the editor, the "TERMINAL" panel is active, showing the following commands and output:

```
● eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ touch exercise2.c
● eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ gcc exercise2.c -o exercise2 -lpthread
● eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$ ./exercise2
Enter a number: 2
Thread checked number: 2
2 is a prime number.
● eman@DESKTOP-ITNHL9J:~/Operating-System/lab4$
```

The bottom status bar of the editor shows "WSL: Ubuntu-24.04", "main*", "Ln 50, Col 1", "Spaces: 4", "UTF-8", "LF", and "C". The Windows taskbar at the bottom displays the search bar, task view, and various application icons, along with the system clock showing "3:52 am 10/10/2025".