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:

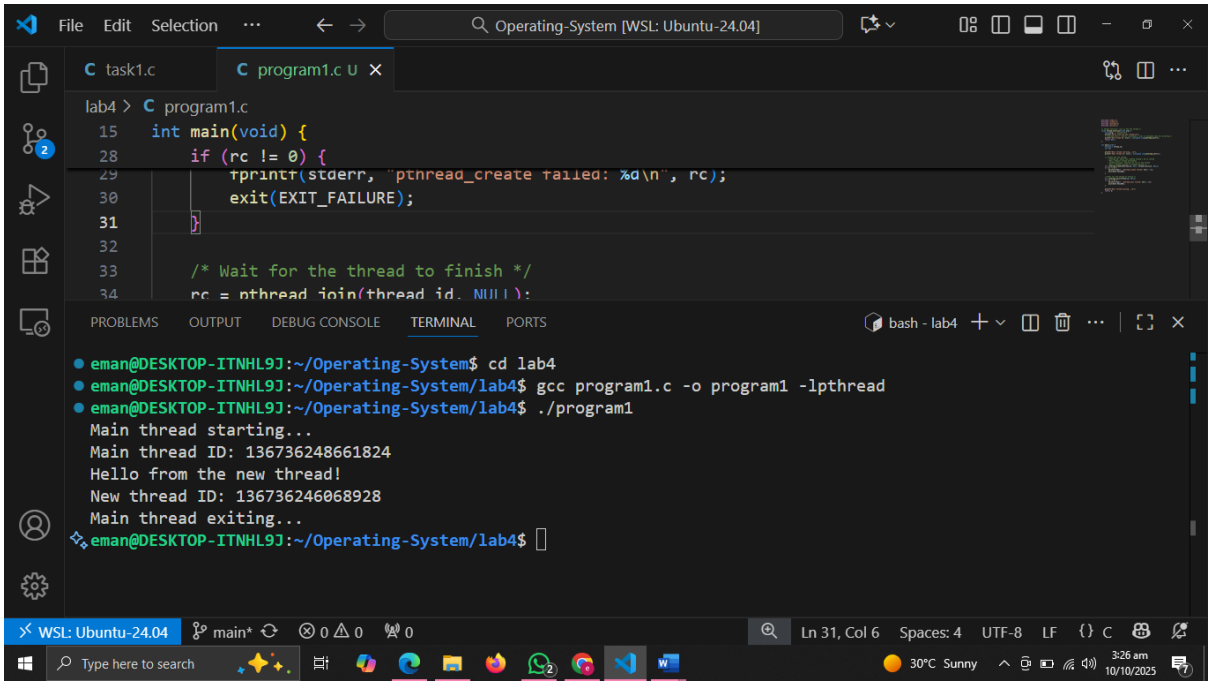
05

3. C Programs with Threads

Program 1: Creating a Simple Thread

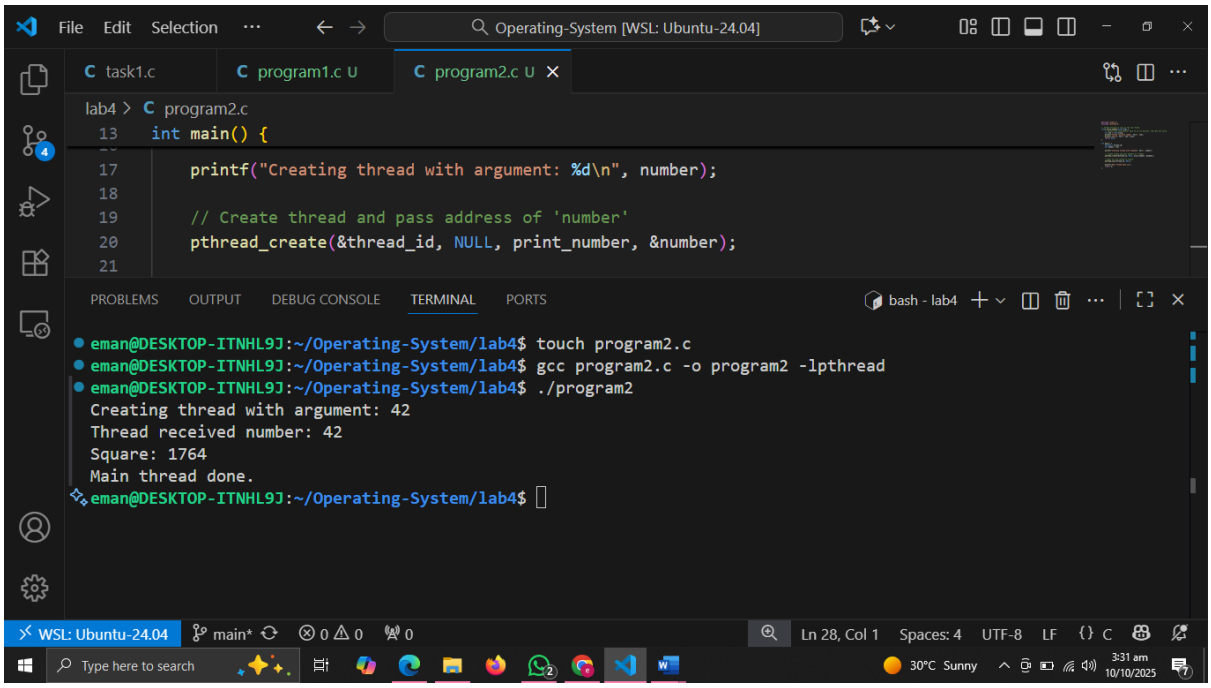
```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

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;
}
```



Program 2: Passing Arguments to Threads

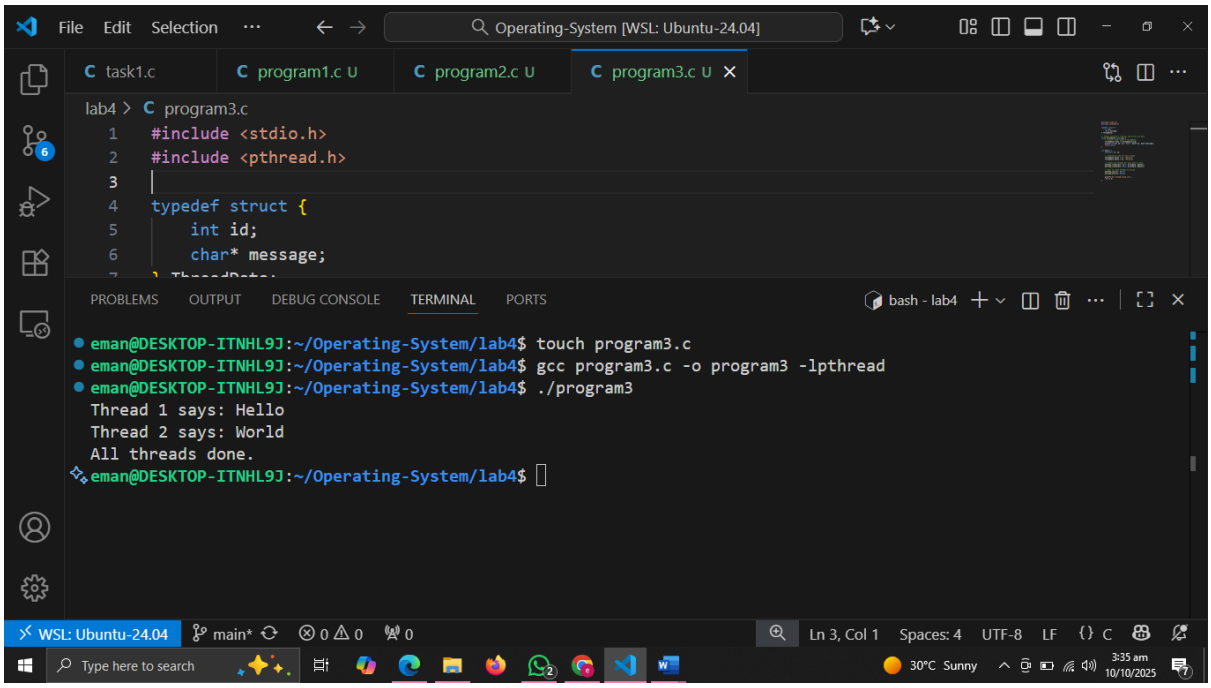
```c
#include <stdio.h>
#include <pthread.h>

// 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;
}
```



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;

}
```

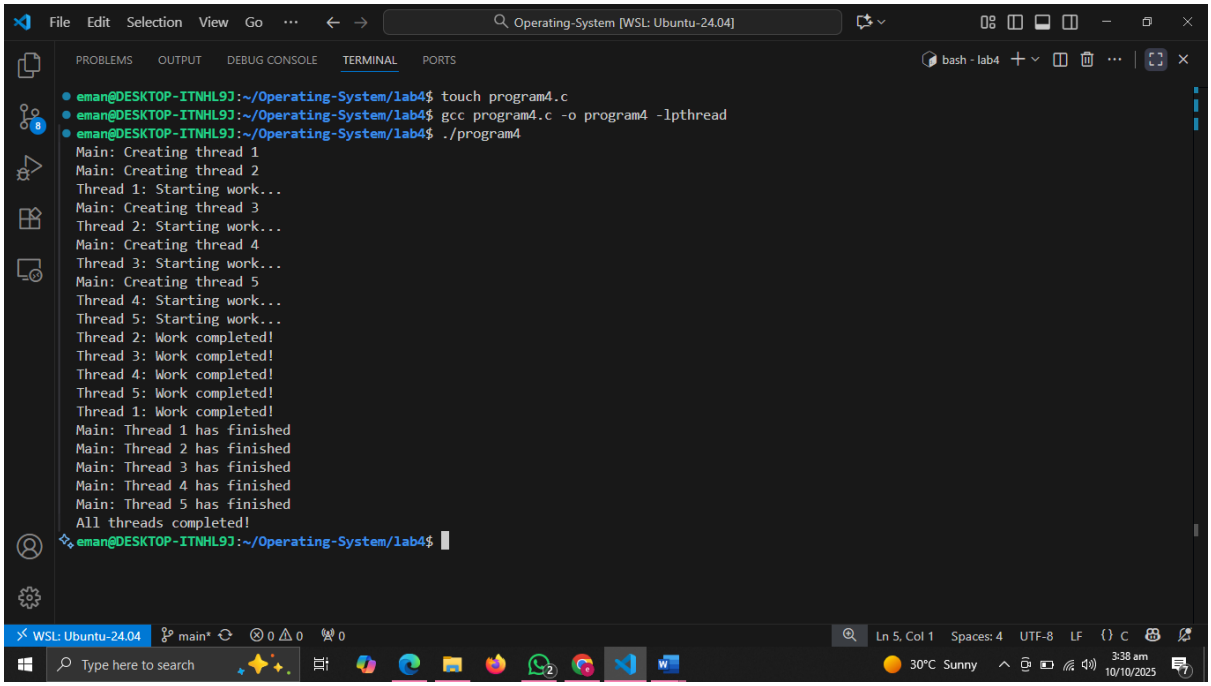
```
int main() {
    pthread_t t1, t2;

    // 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;
}
```



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);
```

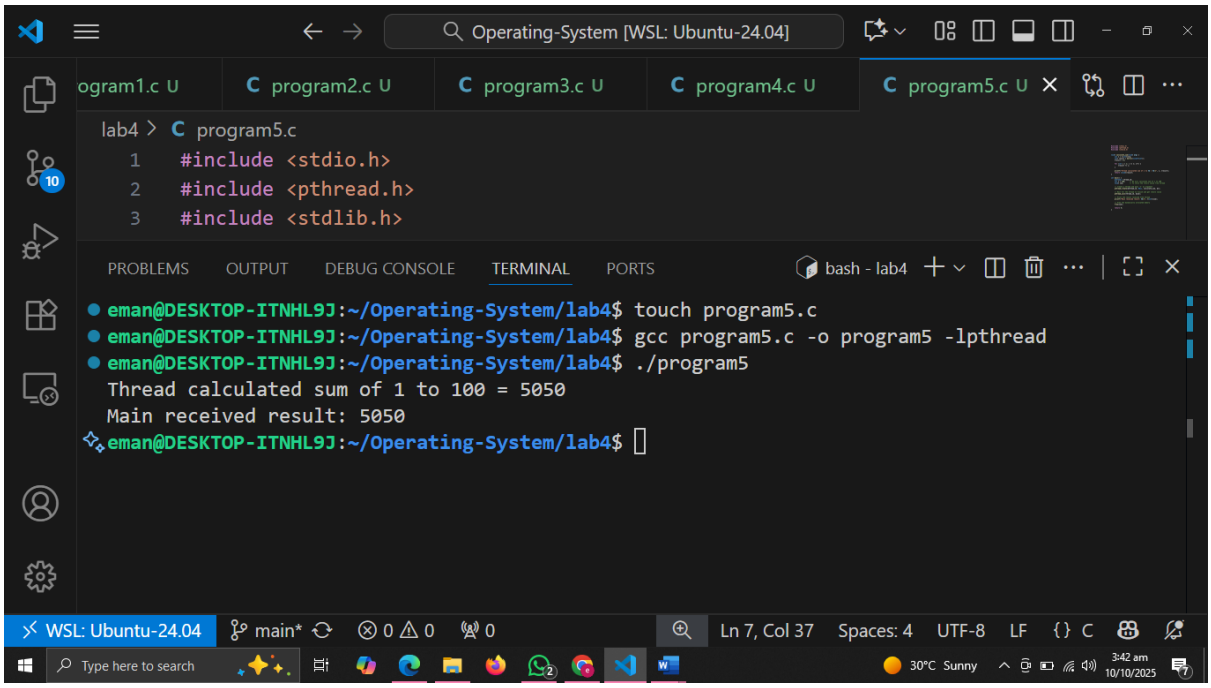
```
        printf("Main: Thread %d has finished\n", i + 1);
    }
    printf("All threads completed!\n");
    return 0;
}
```



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;
}
```



5. Hands-on Practice Exercises

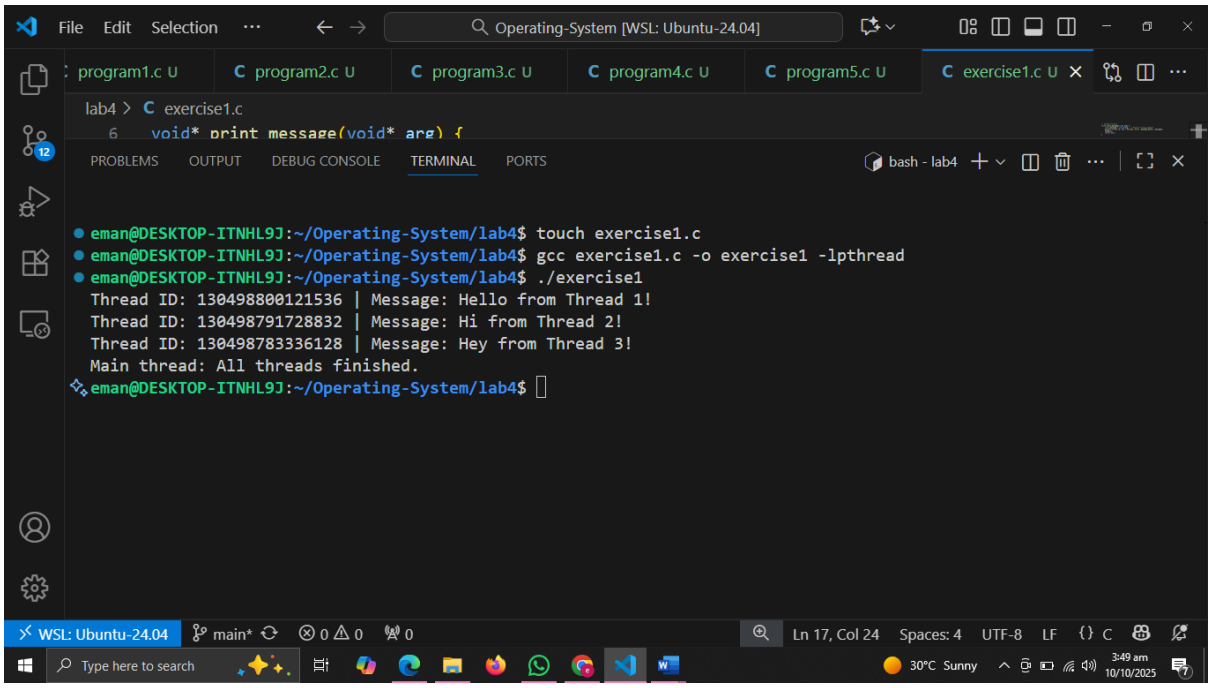
Exercise 1: Thread Basics

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;
}
```



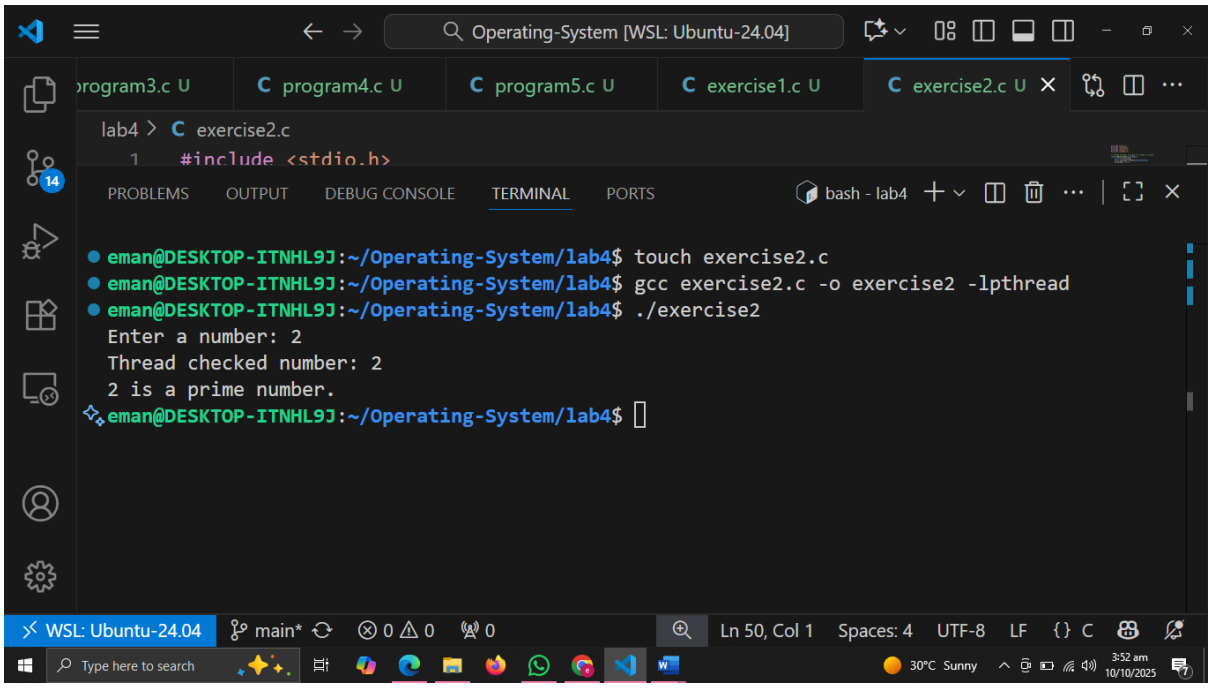
Exercise 2: Prime Number Checker

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;
}
```


Operating-System [WSL: Ubuntu-24.04]
program3.c U
program4.c U
program5.c U
exercise1.c U
exercise2.c U
lab4 > exercise2.c
#include <stdio.h>
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
bash - lab4
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$
WSL: Ubuntu-24.04
main*
Ln 50, Col 1
Spaces: 4
UTF-8
LF
C
Type here to search
30°C Sunny
3:52 am
10/10/2025