**Operating Systems – COC 3071L**

**SE 5th A – Fall 2025**

# Lab 4: Introduction to Threads

## 1. Introduction to Threads

**1.1 What is a Thread?**

A **thread** is the smallest unit of execution within a process.

A **process** can have multiple threads running concurrently All threads within a process share:

Memory space (code, data, heap)

File descriptors

Process ID

Each thread has its own:

Thread ID (TID)

Stack

Program counter

Register set

**Real-world analogy:**

**Process** = A restaurant kitchen

**Threads** = Multiple cooks working together in the same kitchen, sharing ingredients and equipment

### 1.2 Threads vs Processes – Quick Comparison

|  |  |  |
| --- | --- | --- |
| **Feature** | **Process** | **Thread** |
| Memory | Separate memory space | Shared memory space |
| Creation | Expensive (fork) | Lightweight (pthread\_create) |
| Communication | IPC needed (pipes, etc.) | Direct (shared variables) |
| Context Switch | Slower | Faster |
| Independence | Fully independent | Dependent on parent process |

**When to use threads?**

When tasks need to share data frequently

For parallel execution within the same application

When you need lightweight concurrency

## 2. POSIX Threads (pthreads) Library

In Linux, we use the **POSIX threads (pthreads)** library for thread programming.

### 2.1 Compilation Requirements

When compiling programs with threads, you **must** link the pthread library:



The



flag links the pthread library.



## 3. C Programs with Threads

### Program 1: Creating a Simple Thread

**Objective:** Create a thread and print messages from both main thread and new thread.





**pthread\_t thread\_id**

This creates a **variable** to hold the thread’s ID (like a file descriptor or process ID).

It’s just a handle the OS uses to manage the thread.

#### pthread\_create(&thread\_id, NULL, thread\_function, NULL)`

Let’s decode the four parameters:

**Parameter**

**Type**

**Meaning**

&thread

pthread\_t\*

Where the new thread ID will be st

ored

NULL

pthread\_attr\_t\*

Thread attributes (priority, stack size, etc.)

—

NULL

means default

myThread

void\* (\*start\_routine) (void\*)

Function to run in the new thread

NULL

void\*

Pointer passed to the function for data



→ Waits for thr

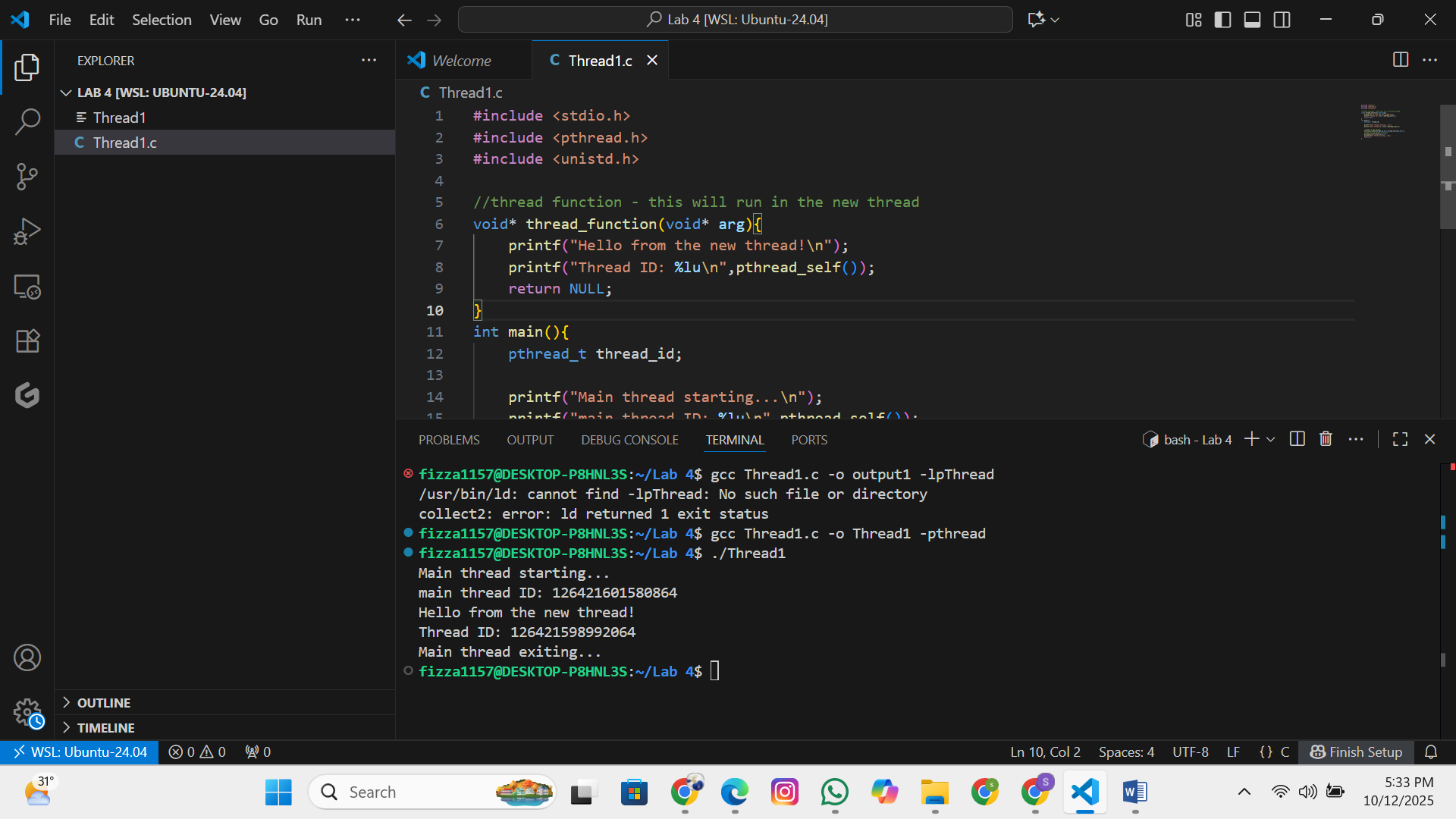
ead to finish (like



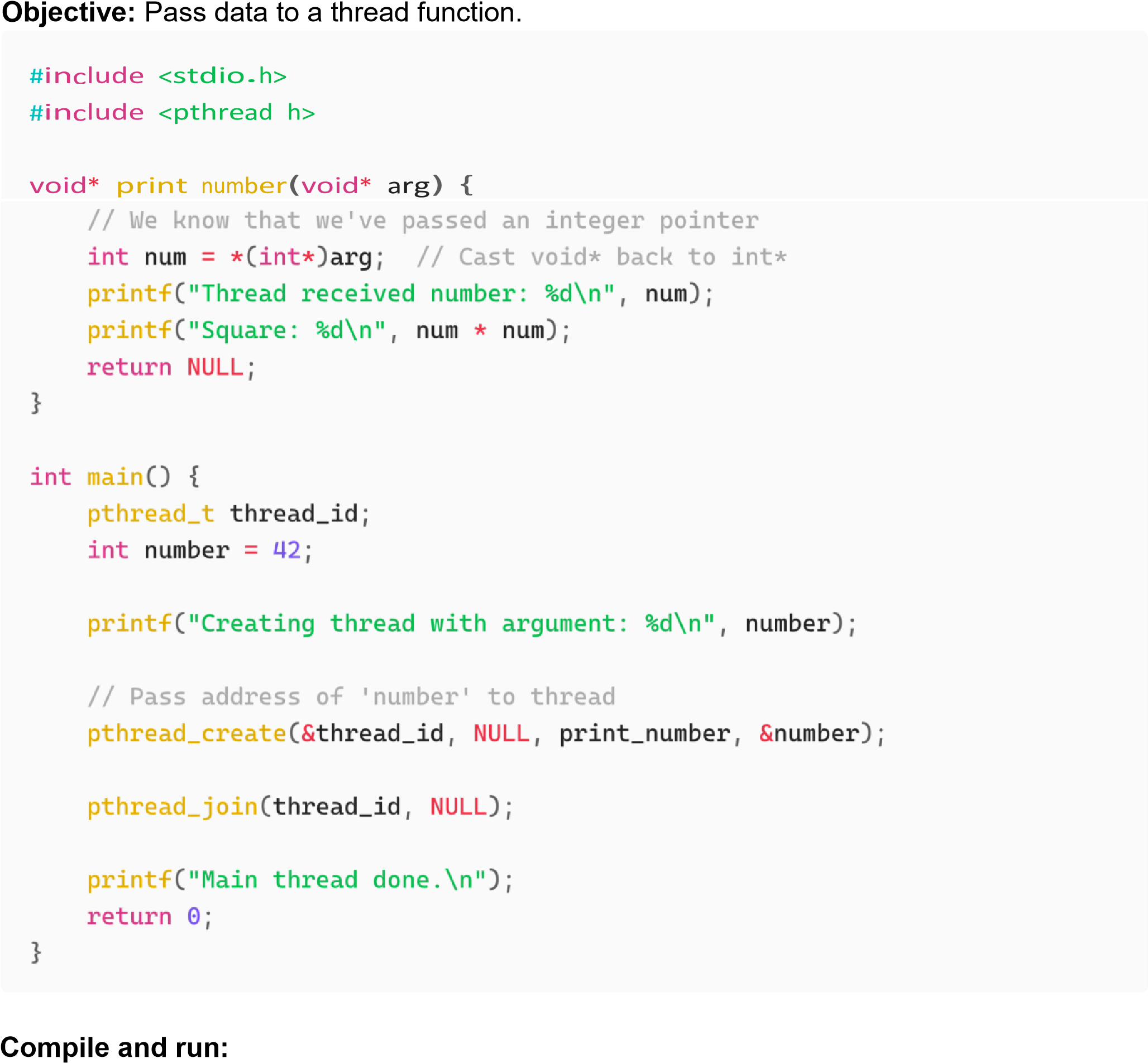
for processes)



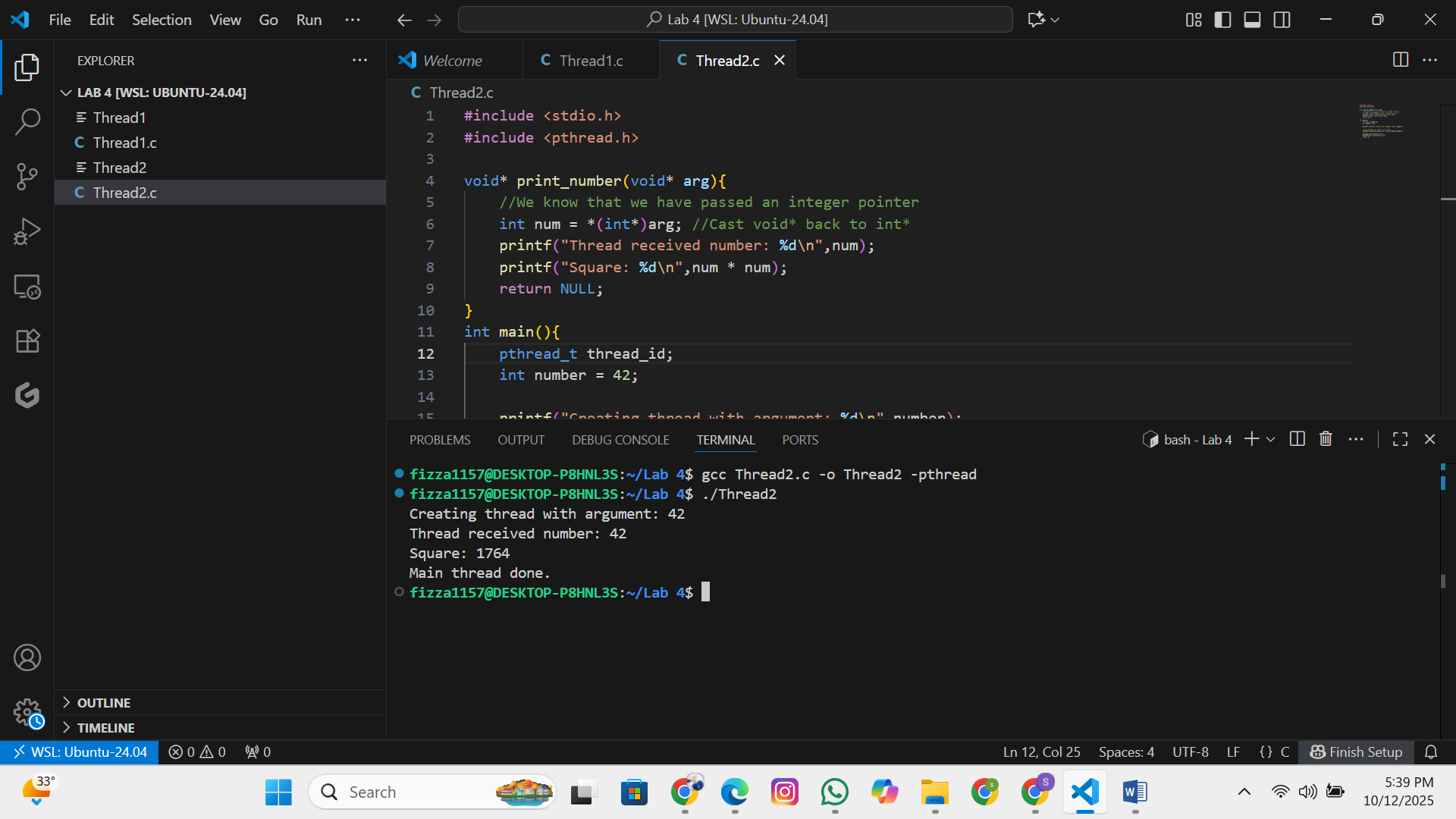
→ Returns the thread ID of calling thread



### Program 2: Passing Arguments to Threads







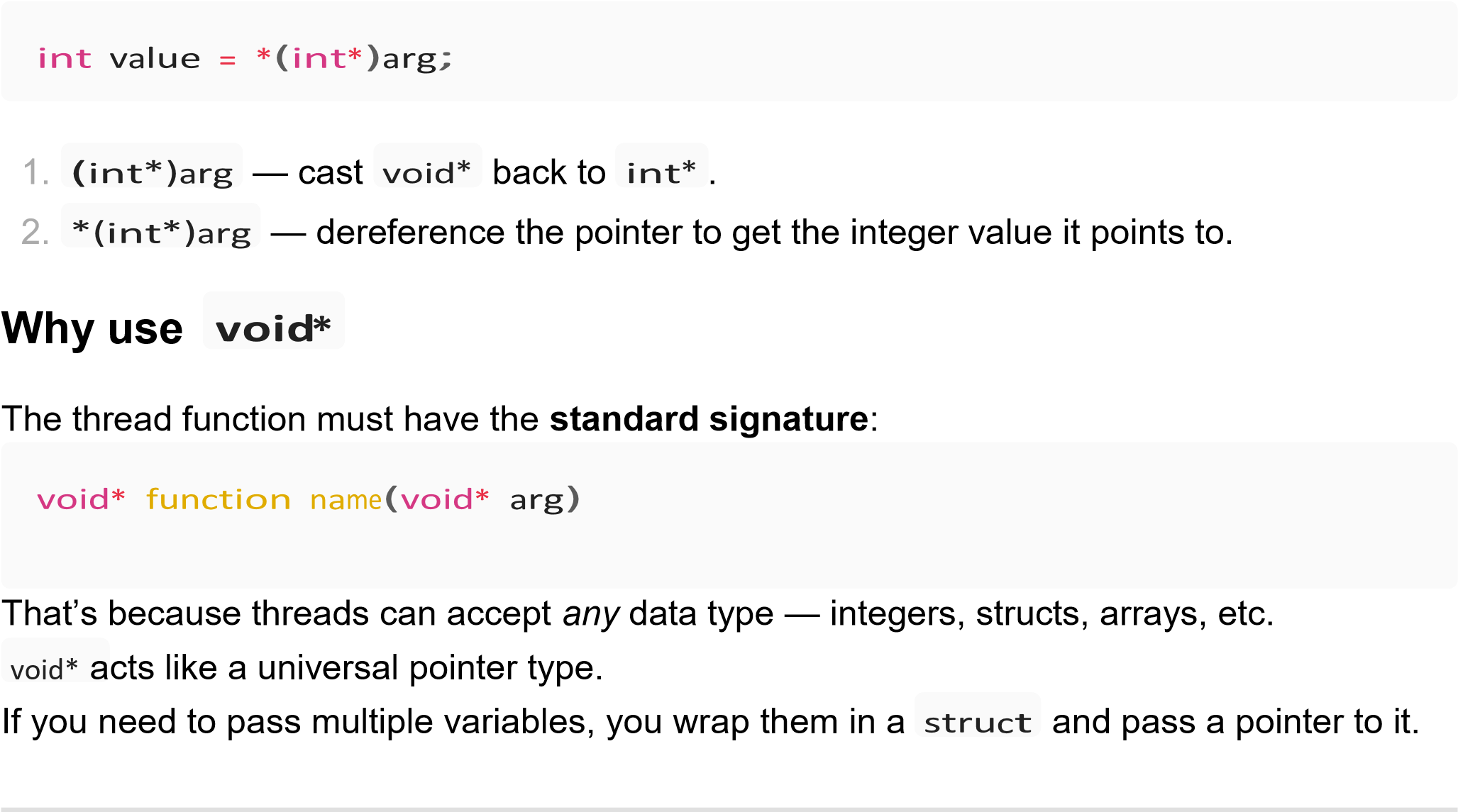
**Important Notes:**

The 4th argument of pthread\_create() is passed to the thread function

It's a void\* pointer, so you can pass any data type

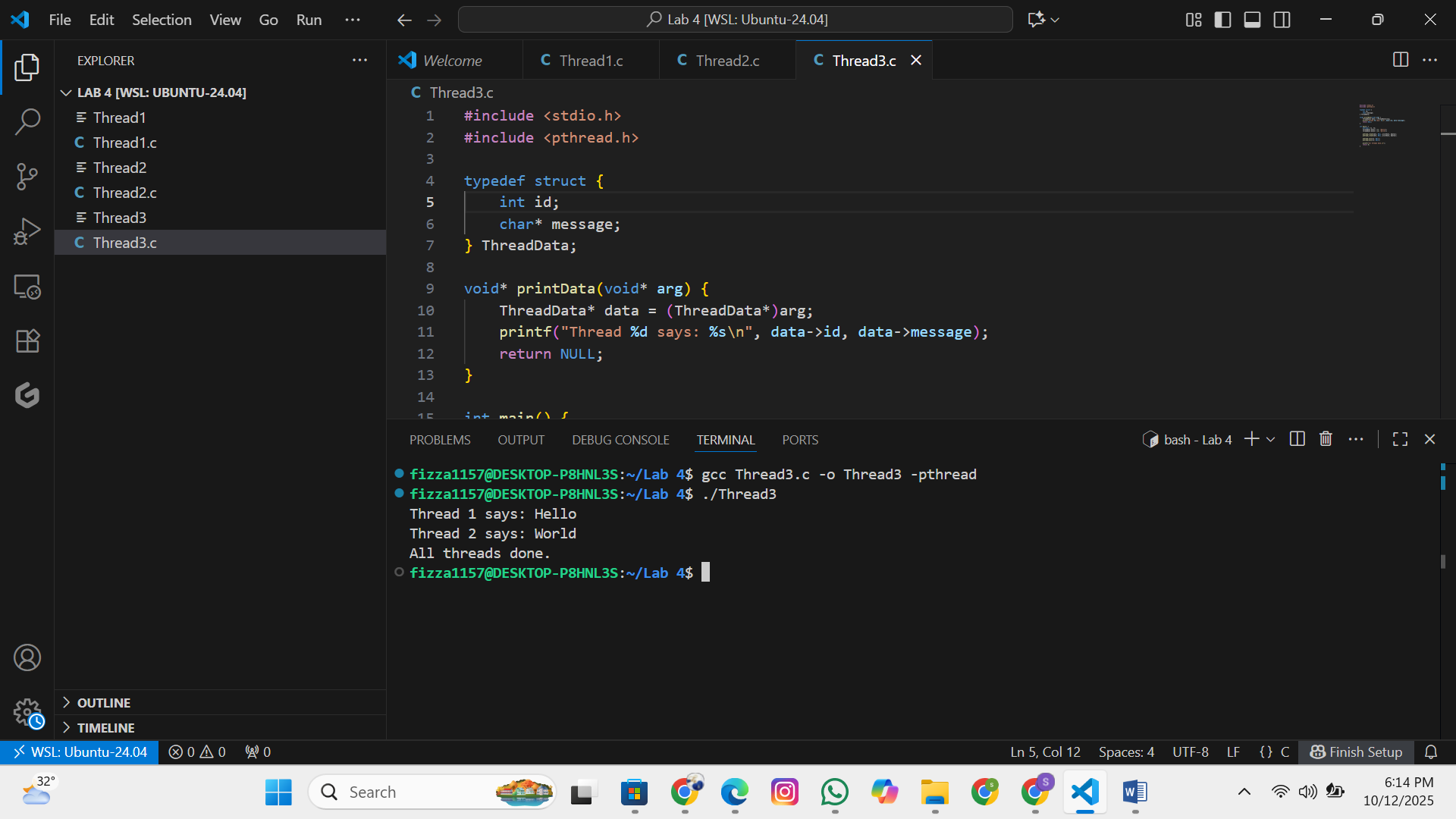
Remember to cast it properly inside the thread function

Here’s what happens step by step:



### Program 3: Passing Multiple Data





### Program 4: Multiple Threads

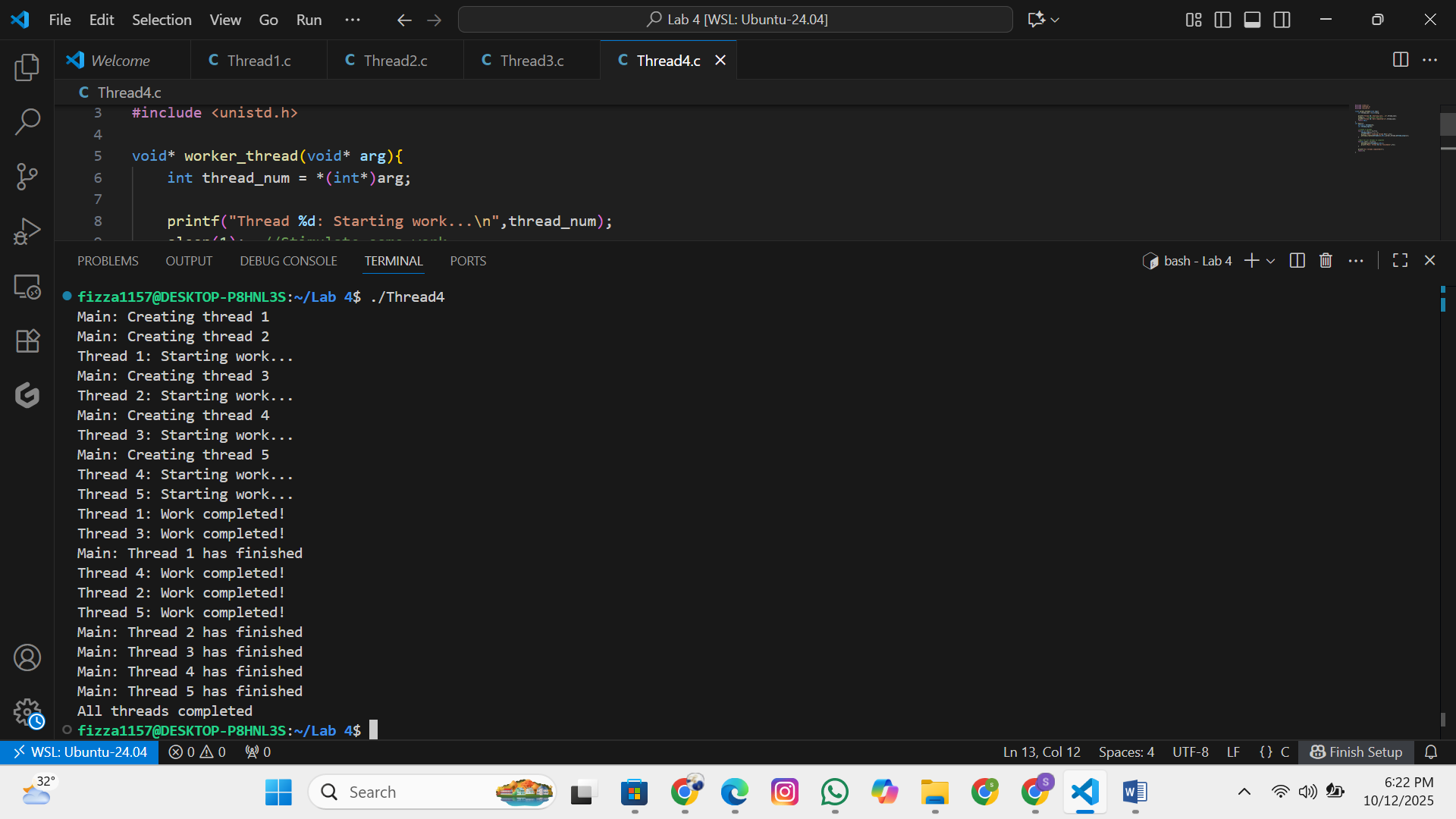
**Objective:** Create multiple threads executing the same function.





**Compile and run:**





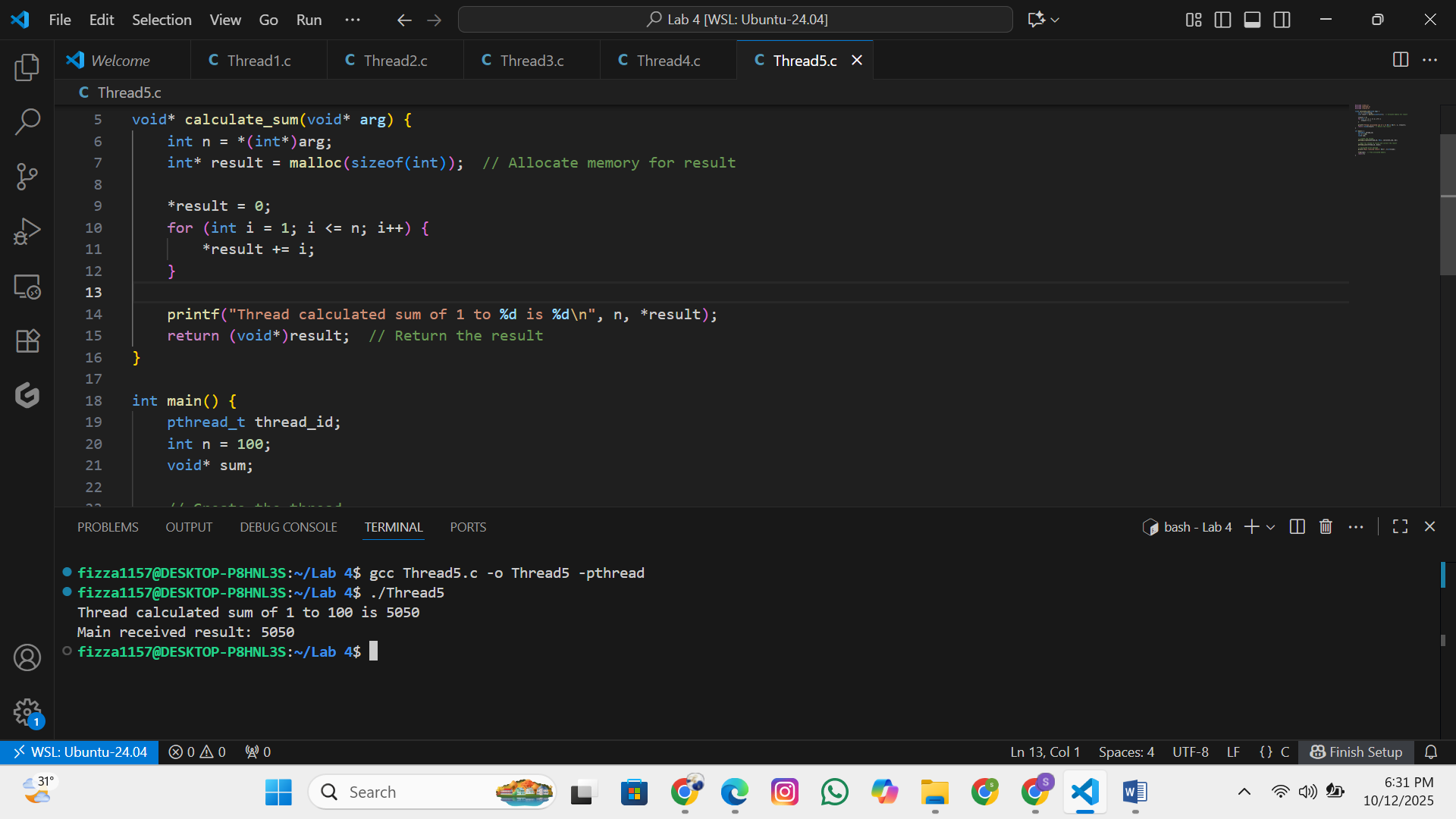
### Program 5: Thread Return Values

**Objective:** Get return values from threads.



**Compile and run:**





**Key Points:**

Thread functions return void\*

Use pthread\_join() to retrieve the return value

Remember to free any dynamically allocated

memory

## 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

### 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