National Textile University

Department of Computer Science

Subject: Operating System

Submitted to: Sir Nasir

Submitted by: Amina

Reg. number: 23-NTU-CS-1136

Lab no.: lab4

Semester:5th

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.

Real-world analogy:

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

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

Process = A restaurant kitchen

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

equipment

When to use threads?

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 -lpthread 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.

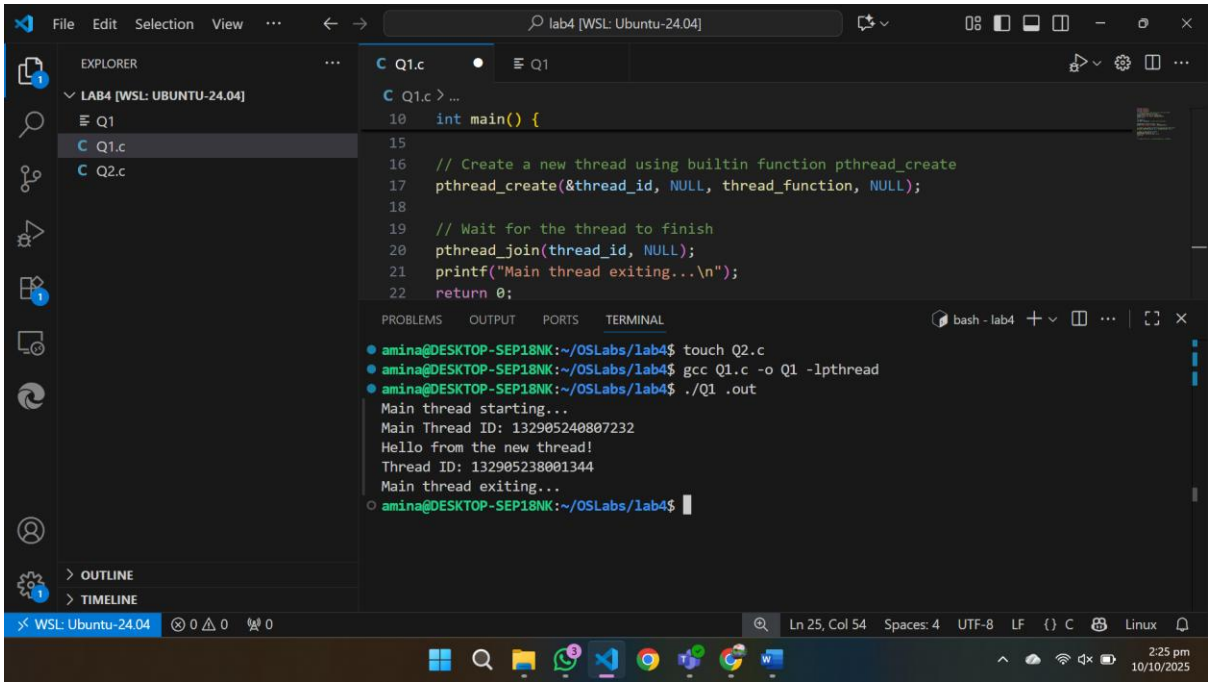
Objective: Create a thread and print messages from both main thread and new thread.

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

// Thread function - this will run in the new thread
void* thread_function(void* arg) {
    printf("Hello from the new thread!\n");
    printf("Thread ID: %lu\n", pthread_self());
    return NULL;
}

int main() {
    pthread_t thread_id;

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

    // Create a new thread
    pthread_create(&thread_id, NULL, thread_function, NULL);
```

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

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

Compile and run:

```
gcc thread1.c -o thread1 -lpthread
./thread1
```


File Edit Selection View
lab4 [WSL: Ubuntu-24.04]
EXPLORER
LAB4 [WSL: UBUNTU-24.04]
Q1
Q1.c
Q2.c
Q1.c
Q1
Q1.c > ...
10 int main() {
15
16 // Create a new thread using builtin function pthread_create
17 pthread_create(&thread_id, NULL, thread_function, NULL);
18
19 // Wait for the thread to finish
20 pthread_join(thread_id, NULL);
21 printf("Main thread exiting...\n");
22 return 0;
PROBLEMS OUTPUT PORTS TERMINAL
bash - lab4
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$ touch Q2.c
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$ gcc Q1.c -o Q1 -lpthread
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$ ./Q1 .out
Main thread starting...
Main Thread ID: 132905240807232
Hello from the new thread!
Thread ID: 132905238001344
Main thread exiting...
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$
OUTLINE
TIMELINE
WSL: Ubuntu-24.04
Ln 25, Col 54 Spaces: 4 UTF-8 LF C Linux
2:25 pm
10/10/2025

Q2 Passing Arguments to Threads:

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

void* print_number(void* arg) {
    // We know that we've passed an integer pointer
    int num = *(int*)arg;  // Cast void* back to int*
    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);

    // Pass address of 'number' to thread
    pthread_create(&thread_id, NULL, print_number, &number);

    pthread_join(thread_id, NULL);

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

Compile and run:

```
gcc thread2.c -o thread2 -lpthread
./thread2
```

Output:

```
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$ gcc Q2.c -o Q2 -lpthread
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$ ./Q2 .out
Creating thread with argument: 42
Thread received number: 42
Square: 1764
Main thread done.
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$
```

Q3 Passing Multiple Data:

```c
#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;
}
```

```
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$ ./Q3
Thread 1 says: Hello
Thread 2 says: World
All threads done.
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$
```

Program 3: Passing Multiple Threads

Objective: Create multiple threads executing the same function.

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

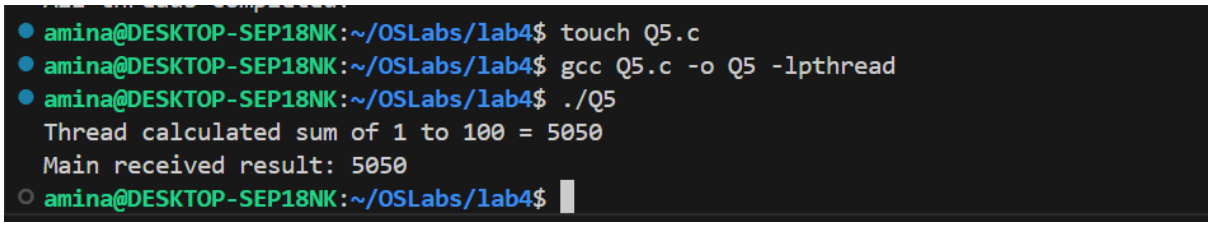
```c
#include <unistd.h>

void* worker_thread(void* arg) {
    int thread_num = *(int*)arg;

    printf("Thread %d: Starting work...\n", thread_num);
    sleep(1);  // Simulate some work
    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]);
    }

    // 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;
}
```

Compile and run:

```
gcc thread3.c -o thread3 -lpthread
./thread3
```

```
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$ gcc Q4.c -o Q4 -lpthread
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$ ./Q4
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 1: Work completed!
Main: Thread 1 has finished
Thread 5: Work completed!
Main: Thread 2 has finished
Thread 4: Work completed!
Thread 3: Work completed!
Main: Thread 3 has finished
Main: Thread 4 has finished
Main: Thread 5 has finished
All threads completed!
amina@DESKTOP-SEP18NK:~/OSLabs/lab4$
```

Program 5: Thread Return Values



5. Hands-on Practice Exercises

Exercise 1: Thread Basics

Write a program that:

Exercise 2: Prime Number Checker

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

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