National Textile University

Department of Computer Science

Subject:
Operating System

Submitted to:
Sir Nasir

Submitted by:
Ghulam Mohyuddin

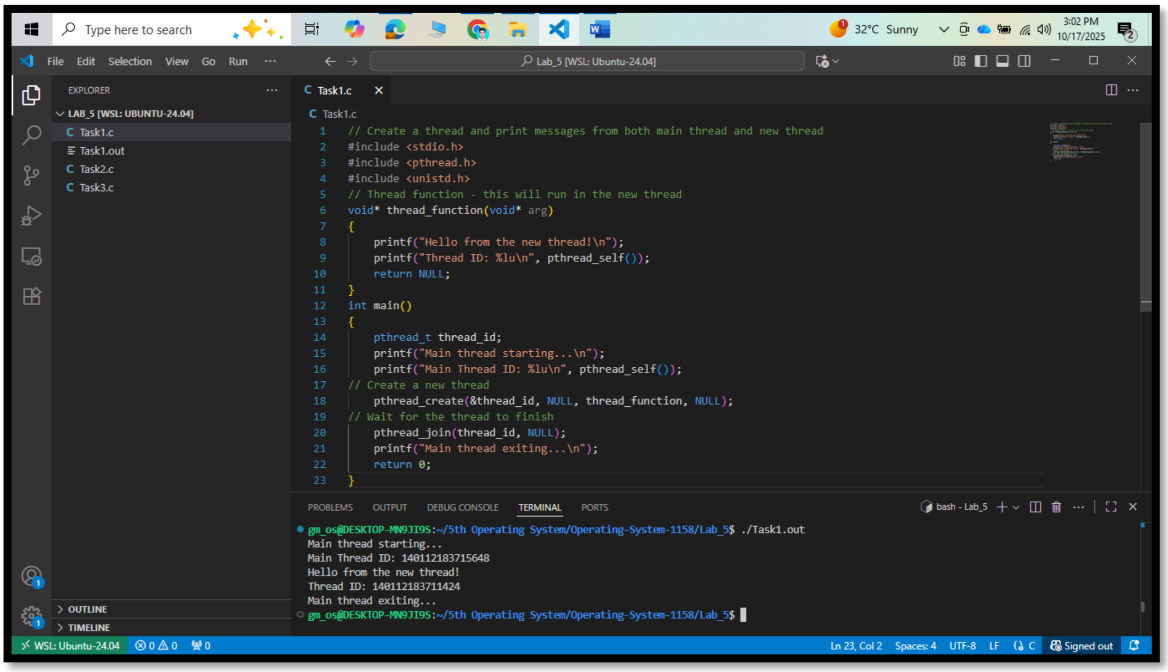
Reg number:
23-NTU-CS-1158

Lab number:
5th

Semester:
5th

Task 3: C Programs with Threads

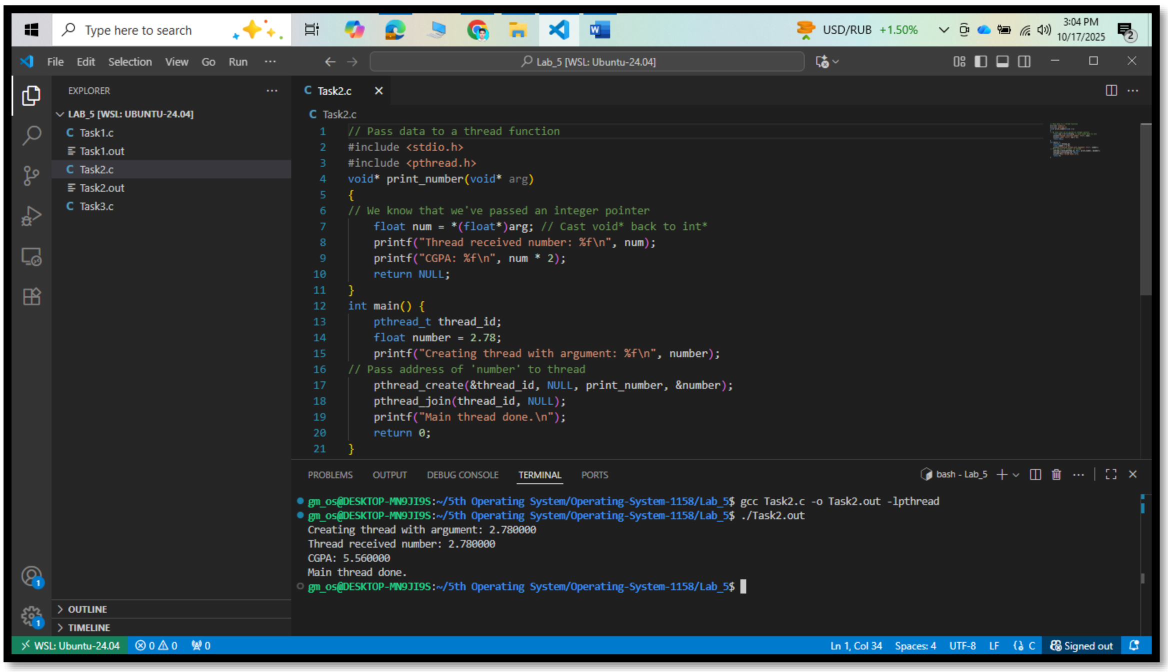
Program 1: Creating a Simple Thread



```c
// Create a thread and print messages from both main thread and new thread
#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);
// Wait for the thread to finish
    pthread_join(thread_id, NULL);
    printf("Main thread exiting...\n");
    return 0;
}
```

```
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ ./Task1.out
Main thread starting...
Main Thread ID: 140112183715648
Hello from the new thread!
Thread ID: 140112183711424
Main thread exiting...
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$
```

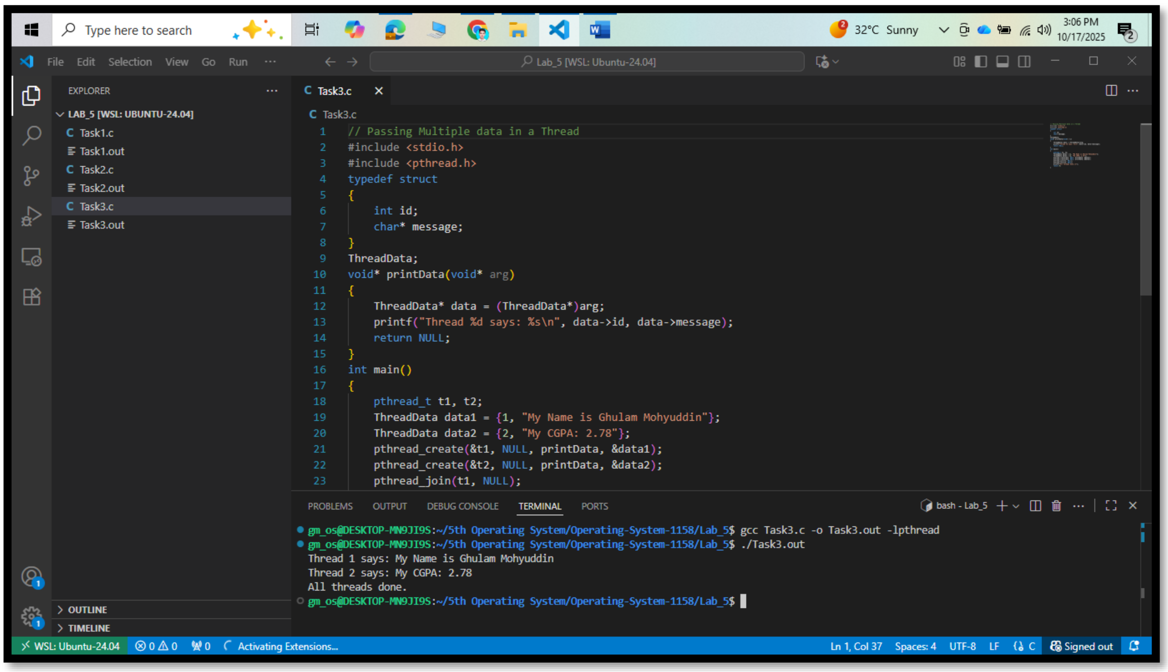
Program 2: Passing Arguments to Thread



```c
// Pass data to a thread function
#include <stdio.h>
#include <pthread.h>
void* print_number(void* arg)
{
// We know that we've passed an integer pointer
    float num = *(float*)arg; // Cast void* back to int*
    printf("Thread received number: %f\n", num);
    printf("CGPA: %f\n", num * 2);
    return NULL;
}
int main() {
    pthread_t thread_id;
    float number = 2.78;
    printf("Creating thread with argument: %f\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;
}
```

```
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ gcc Task2.c -o Task2.out -lpthread
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ ./Task2.out
Creating thread with argument: 2.780000
Thread received number: 2.780000
CGPA: 5.560000
Main thread done.
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$
```

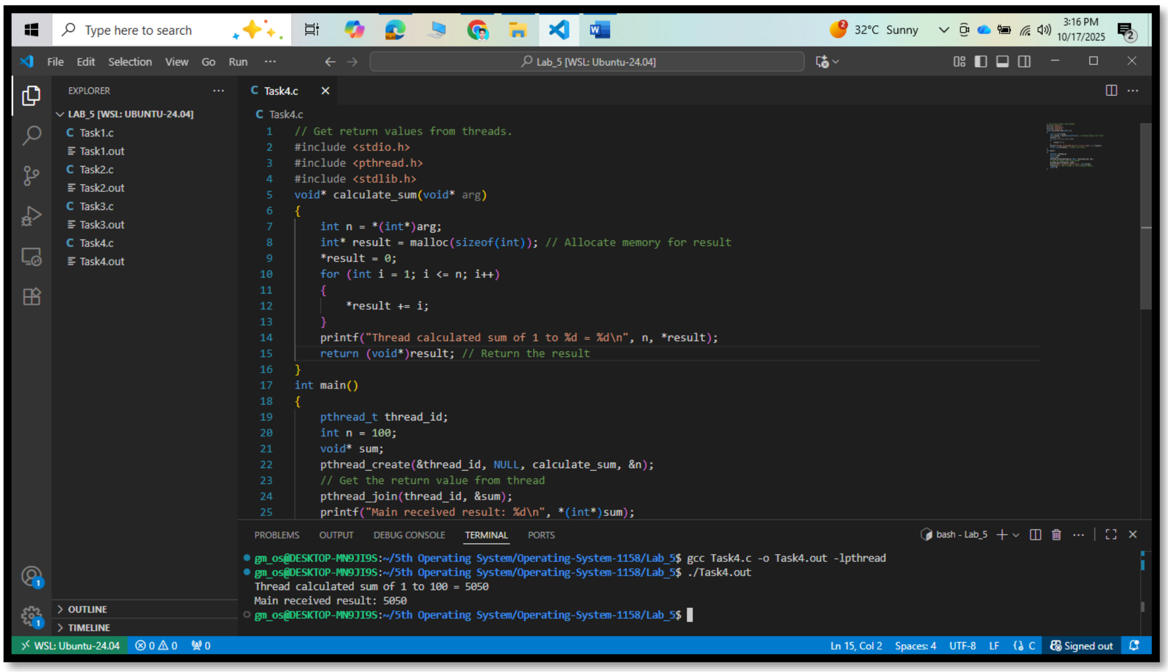
Program 3: Passing Multiple Data



```c
// Passing Multiple data in a Thread
#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, "My Name is Ghulam Mohyuddin"};
    ThreadData data2 = {2, "My CGPA: 2.78"};
    pthread_create(&t1, NULL, printData, &data1);
    pthread_create(&t2, NULL, printData, &data2);
    pthread_join(t1, NULL);
```

```
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ gcc Task3.c -o Task3.out -lpthread
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ ./Task3.out
Thread 1 says: My Name is Ghulam Mohyuddin
Thread 2 says: My CGPA: 2.78
All threads done.
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$
```

Program 4: Thread Return Values

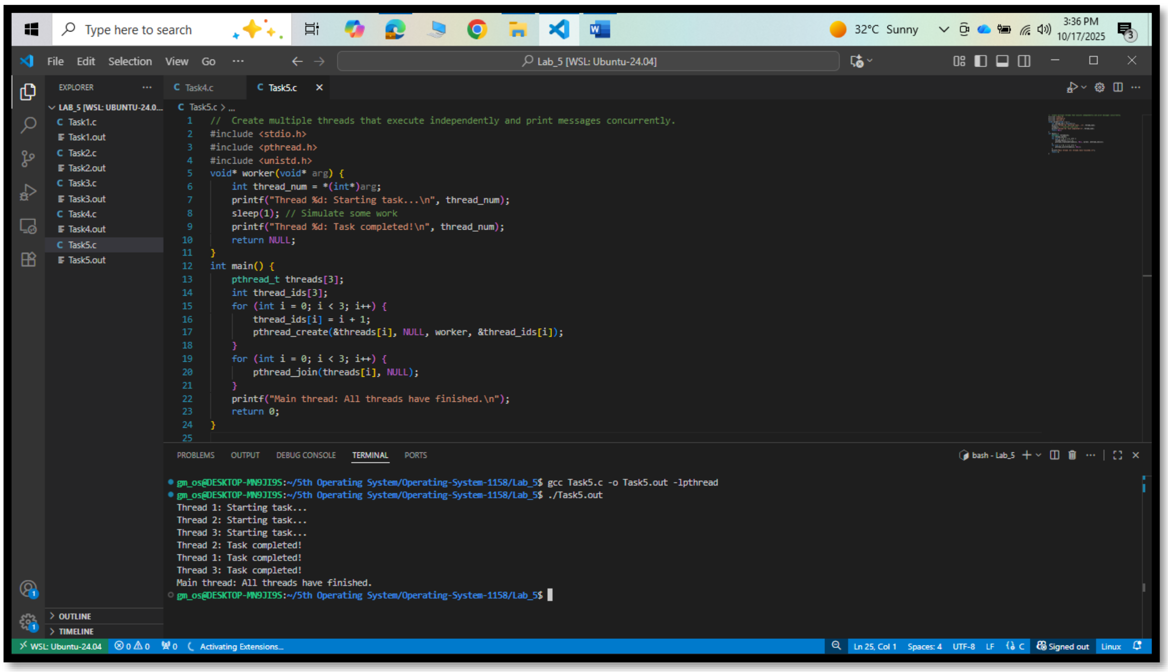


```c
// Get return values from threads.
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
void* calculate_sum(void* arg)
{
    int n = *(int*)arg;
    int* result = malloc(sizeof(int)); // Allocate memory for result
    *result = 0;
    for (int i = 1; i <= n; i++)
    {
        *result +=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

```
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ gcc Task4.c -o Task4.out -lpthread
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ ./Task4.out
Thread calculated sum of 1 to 100 = 5050
Main received result: 5050
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$
```

Task 4: Basic Multithreading

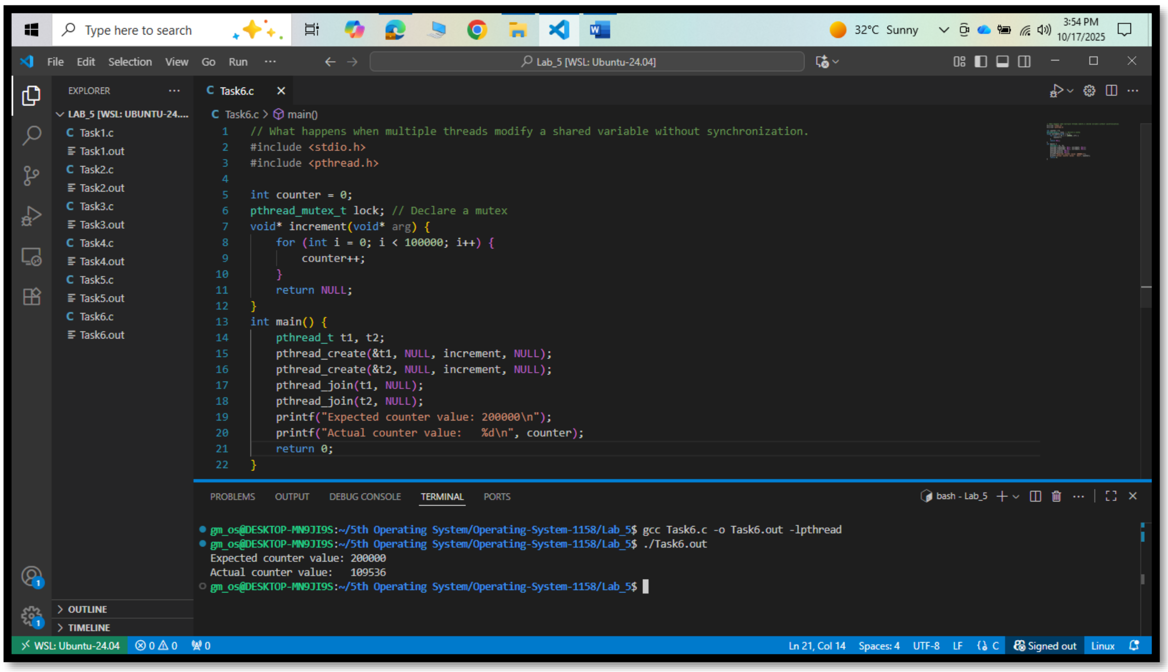
Program 1: Creating and Running Multiple Threads



```c
//  Create multiple threads that execute independently and print messages concurrently.
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
void* worker(void* arg) {
    int thread_num = *(int*)arg;
    printf("Thread %d: Starting task...\n", thread_num);
    sleep(1); // Simulate some work
    printf("Thread %d: Task completed!\n", thread_num);
    return NULL;
}
int main() {
    pthread_t threads[3];
    int thread_ids[3];
    for (int i = 0; i < 3; i++) {
        thread_ids[i] = i + 1;
        pthread_create(&threads[i], NULL, worker, &thread_ids[i]);
    }
    for (int i = 0; i < 3; i++) {
        pthread_join(threads[i], NULL);
    }
    printf("Main thread: All threads have finished.\n");
    return 0;
}
```

```
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ gcc Task5.c -o Task5.out -lpthread
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ ./Task5.out
Thread 1: Starting task...
Thread 2: Starting task...
Thread 3: Starting task...
Thread 2: Task completed!
Thread 1: Task completed!
Thread 3: Task completed!
Main thread: All threads have finished.
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$
```

Program 2: Demonstrating a Race Condition



```c
// What happens when multiple threads modify a shared variable without synchronization.
#include <stdio.h>
#include <pthread.h>

int counter = 0;
pthread_mutex_t lock; // Declare a mutex
void* increment(void* arg) {
    for (int i = 0; i < 100000; i++) {
        counter++;
    }
    return NULL;
}
int main() {
    pthread_t t1, t2;
    pthread_create(&t1, NULL, increment, NULL);
    pthread_create(&t2, NULL, increment, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    printf("Expected counter value: 200000\n");
    printf("Actual counter value:   %d\n", counter);
    return 0;
}
```

```
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ gcc Task6.c -o Task6.out -lpthread
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$ ./Task6.out
Expected counter value: 200000
Actual counter value:   109536
gm_os@DESKTOP-MN9JI9S:~/5th Operating System/Operating-System-1158/Lab_5$
```