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:

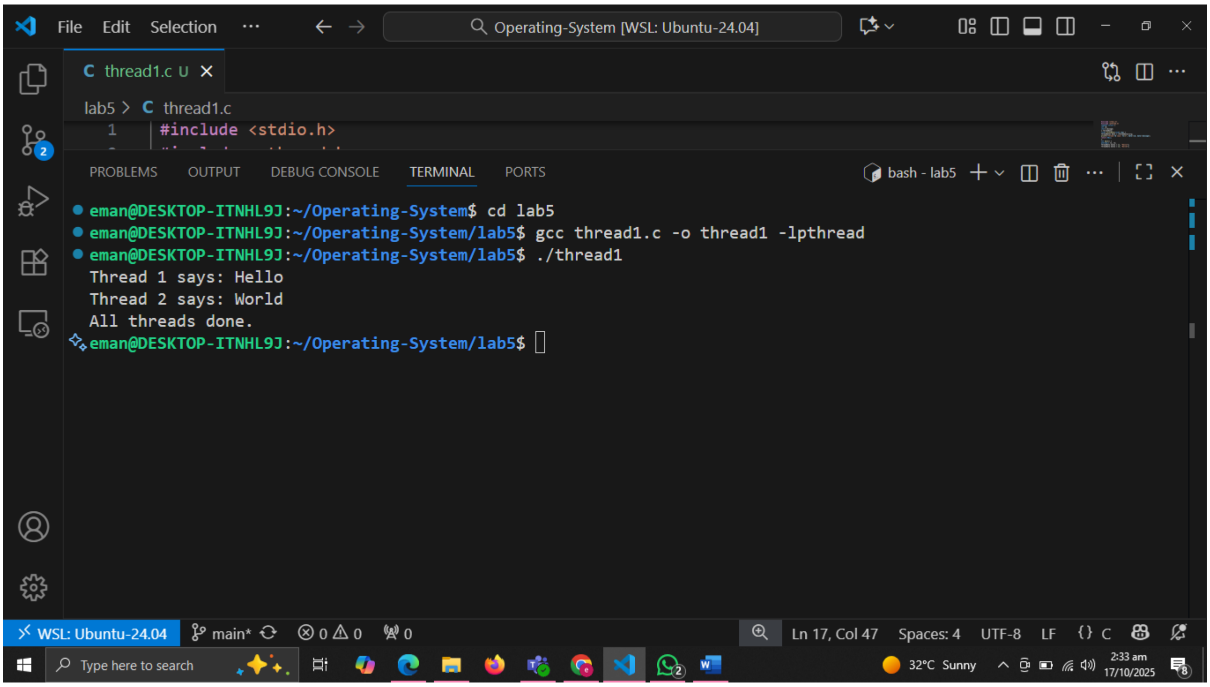
05

3. C Programs with Threads

Program 1: Creating a Simple Thread

Objective: Create a thread and print messages from both main thread and new 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, "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;
 }
```

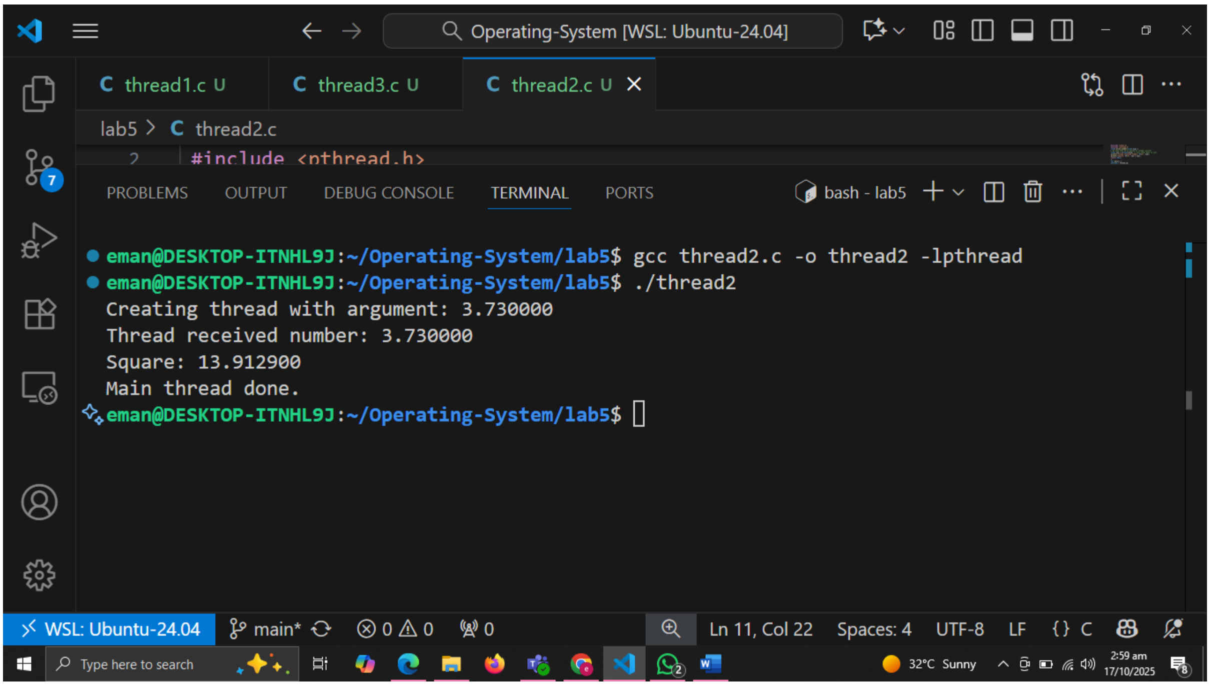


```
eman@DESKTOP-ITNHL9J:~/Operating-System$ cd lab5
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ gcc thread1.c -o thread1 -lpthread
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ ./thread1
Thread 1 says: Hello
Thread 2 says: World
All threads done.
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$
```

Program 2: Passing Arguments to Threads

Objective: Pass data to a thread function.

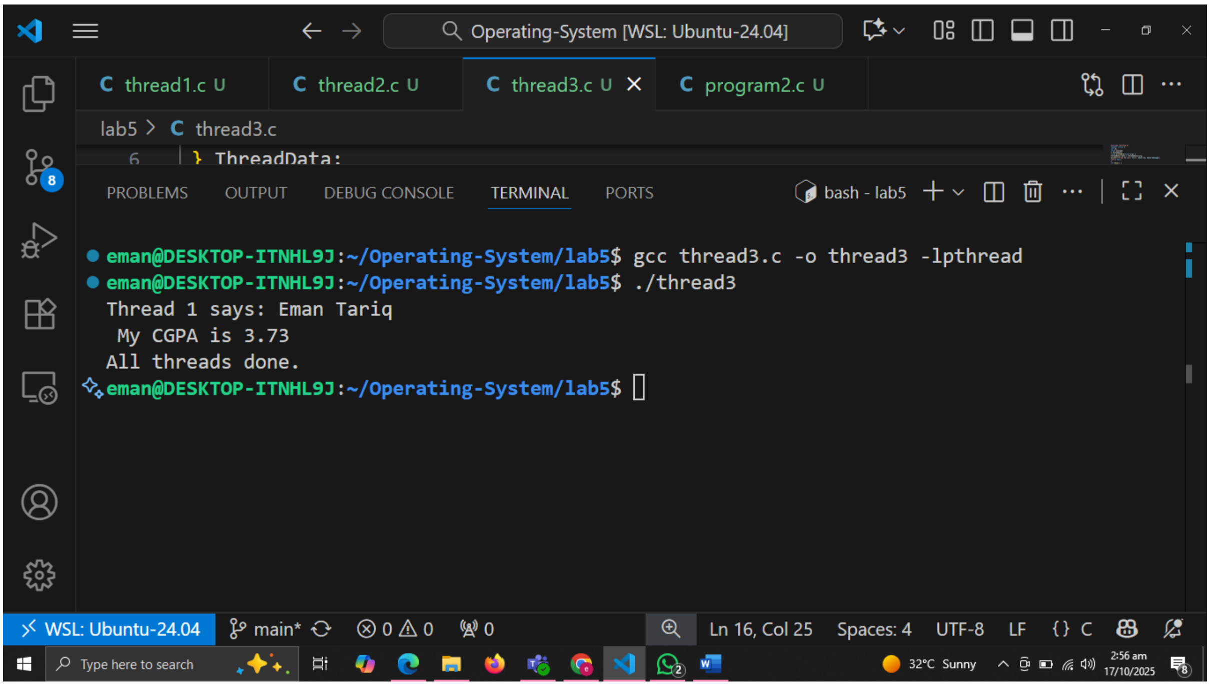
```
#include <stdio.h>
 #include <pthread.h>
 void* print_number(void* arg) {
// We know that we've passed an integer pointer
 float num = *(float*)arg; // Cast void* back to int*
 printf("Thread received number: %f\n", num);
 printf("Square: %f\n", num * num);
 return NULL;
 }
 int main() {
 pthread_t thread_id;
 float number = 3.73;
 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;
 }
```



```
● eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ gcc thread2.c -o thread2 -lpthread
● eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ ./thread2
  Creating thread with argument: 3.730000
  Thread received number: 3.730000
  Square: 13.912900
  Main thread done.
  eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ 
```

Program 3: Passing Multiple Data

```
#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;
ThreadData data1 = {1, "Eman Tariq \n My CGPA is 3.73"};
pthread_create(&t1, NULL, printData, &data1);
pthread_join(t1, NULL);
printf("All threads done.\n");
return 0;
}
```



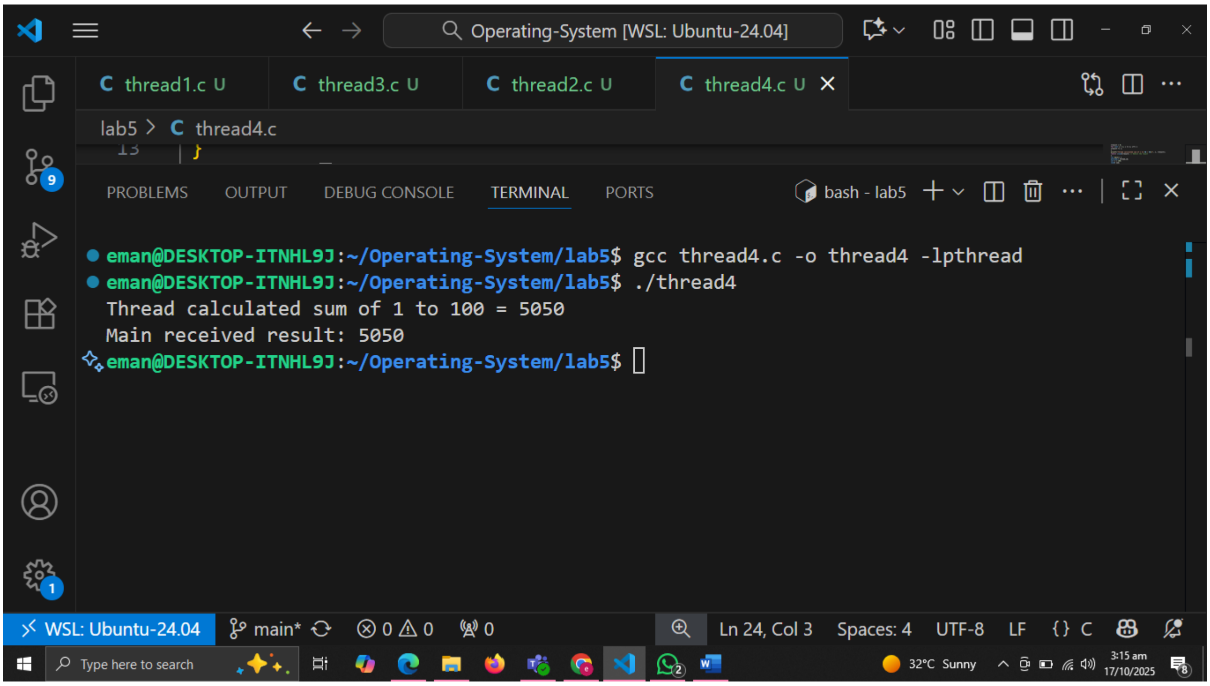
```
6   } ThreadData;
```

```
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ gcc thread3.c -o thread3 -lpthread
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ ./thread3
Thread 1 says: Eman Tariq
 My CGPA is 3.73
All threads done.
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$
```

Program 4: Thread Return Values

Objective: 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);
free(sum); // Don't forget to free allocated memory
return 0;
}
```



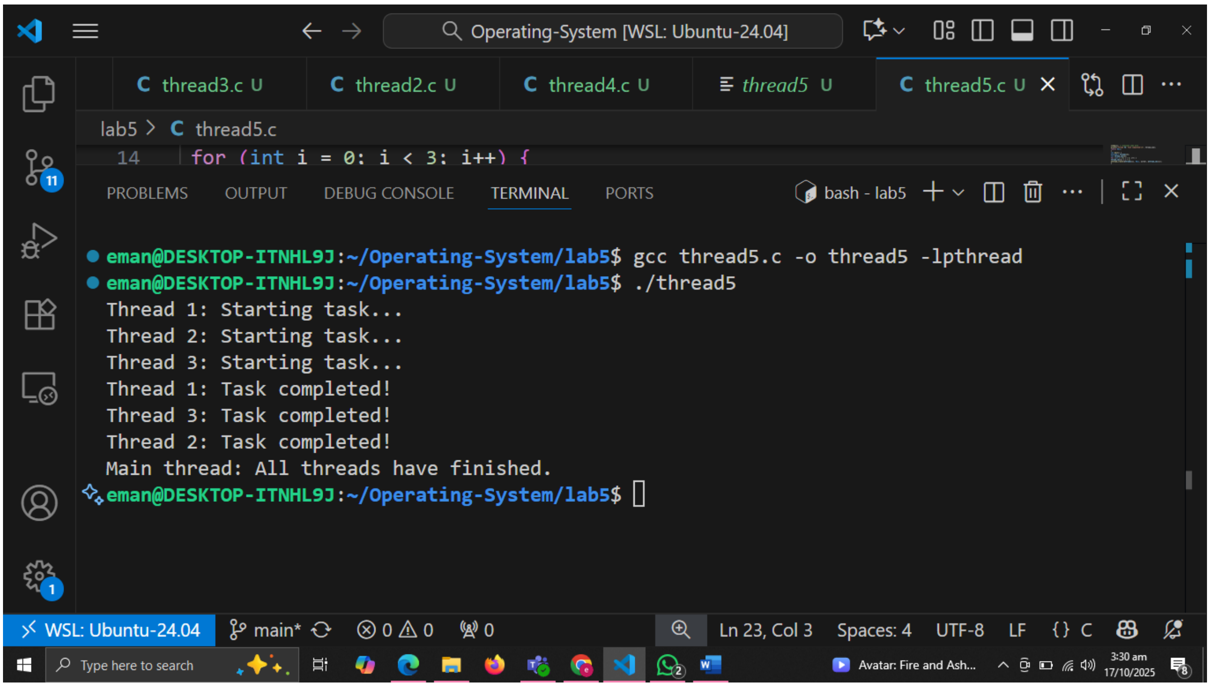
```
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ gcc thread4.c -o thread4 -lpthread
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ ./thread4
Thread calculated sum of 1 to 100 = 5050
Main received result: 5050
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ 
```

4. Basic Multithreading

Program 1: Creating and Running Multiple Threads

Objective: 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;
 }
```



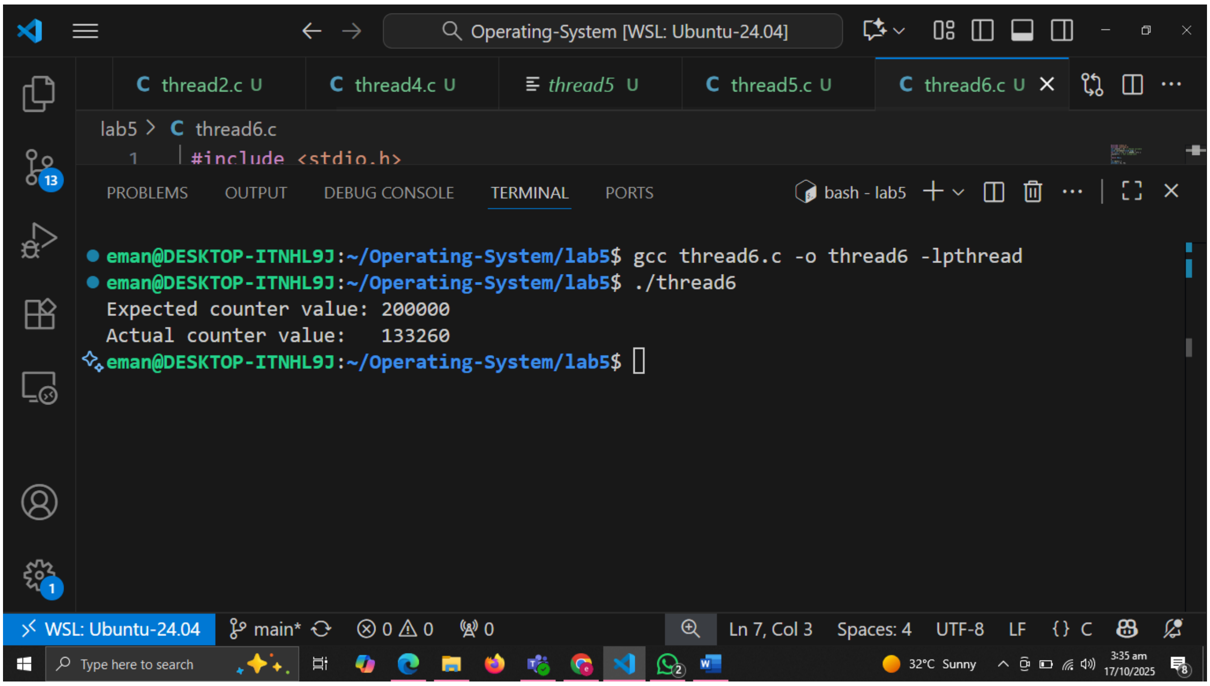
```
14    for (int i = 0; i < 3; i++) {
```

```
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ gcc thread5.c -o thread5 -lpthread
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ ./thread5
Thread 1: Starting task...
Thread 2: Starting task...
Thread 3: Starting task...
Thread 1: Task completed!
Thread 3: Task completed!
Thread 2: Task completed!
Main thread: All threads have finished.
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ 
```

Program 2: Demonstrating a Race Condition

Objective: What happens when multiple threads modify a shared variable without synchronization

```
#include <stdio.h>
 #include <pthread.h>
 int counter = 0; // Shared variable
 void* increment(void* arg) {
 for (int i = 0; i < 100000; i++) {
 counter++; // Not thread-safe
 }
 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;
 }
```



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

```
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ gcc thread6.c -o thread6 -lpthread
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ ./thread6
Expected counter value: 200000
Actual counter value:   133260
eman@DESKTOP-ITNHL9J:~/Operating-System/lab5$ 
```