



National Textile University
Department of Computer Science

Subject:

Operating System

Submitted to:

Dr. Nasir Mehmood

Submitted by:

Dawar Abbas

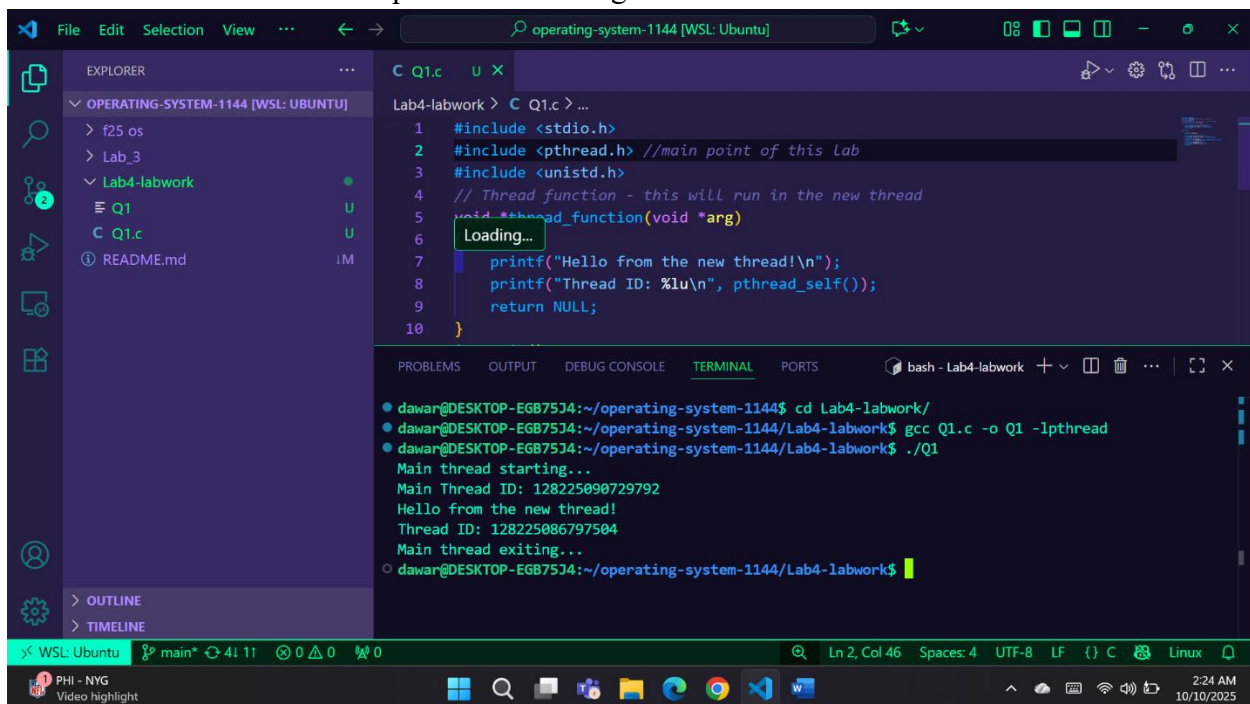
Reg number:

Lab no. :
4

Semester:
5th

Lab 4- Lab work

In the first program we created a thread and first we printed the main “thread id” and then we created a new thread and also passed it some arguments:



The screenshot displays the Visual Studio Code interface with a C program in the editor and its execution output in the terminal. The Explorer pane on the left shows the file structure of the 'Lab4-labwork' directory, including 'Q1.c' and 'README.md'. The editor shows the following C code:

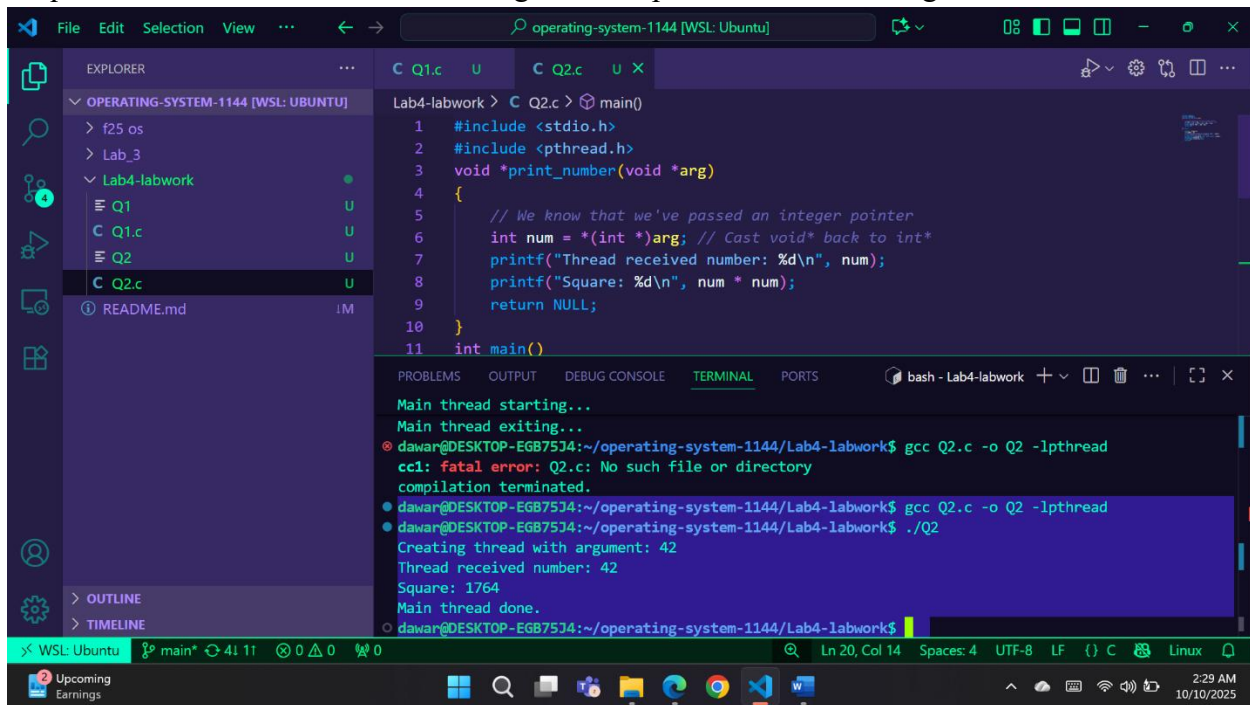
```
1 #include <stdio.h>
2 #include <pthread.h> //main point of this lab
3 #include <unistd.h>
4 // Thread function - this will run in the new thread
5 void *thread_function(void *arg)
6 {
7     printf("Hello from the new thread!\n");
8     printf("Thread ID: %lu\n", pthread_self());
9     return NULL;
10 }
```

The terminal output shows the execution of the program:

```
dawar@DESKTOP-EG875J4:~/operating-system-1144$ cd Lab4-labwork/
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork$ gcc Q1.c -o Q1 -lpthread
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork$ ./Q1
Main thread starting...
Main Thread ID: 128225090729792
Hello from the new thread!
Thread ID: 128225086797504
Main thread exiting...
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork$
```

The status bar at the bottom indicates the current file is 'main.c' at line 2, column 46, with 4 spaces, UTF-8 encoding, and LF line endings. The system tray shows the time as 2:24 AM on 10/10/2025.

In question 2 we have used the threading and also passed the number arguments:

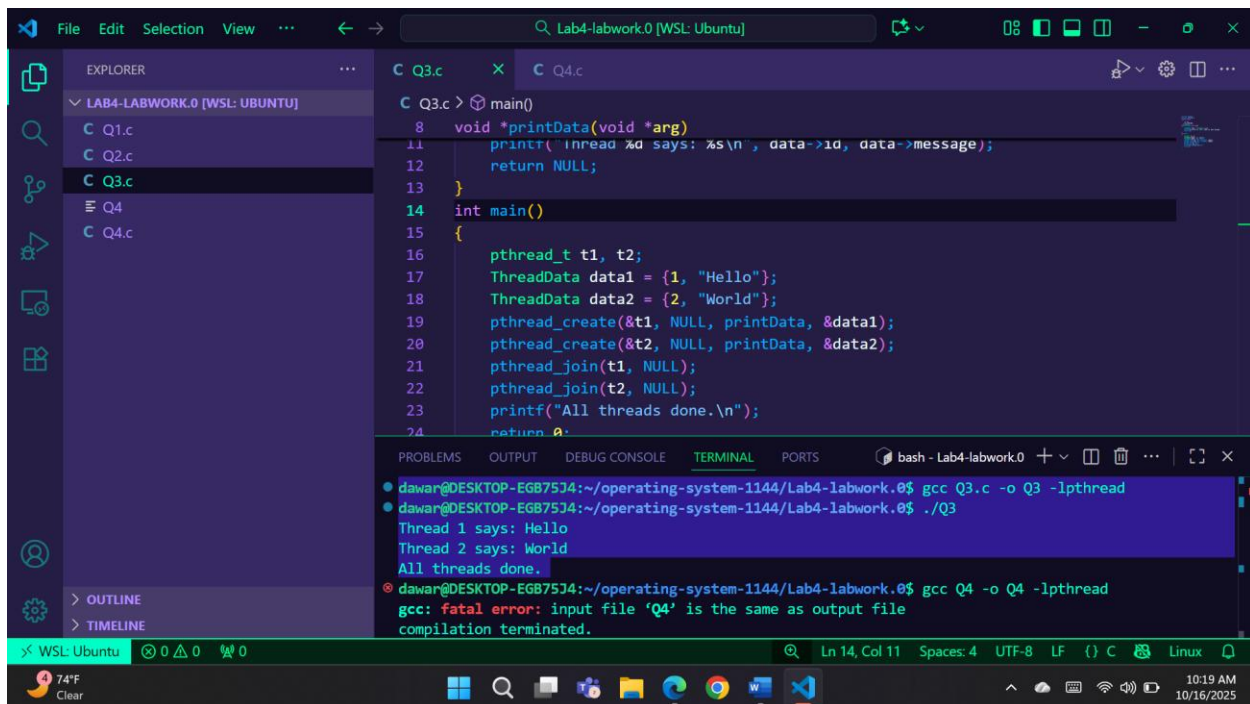


The screenshot shows the Visual Studio Code interface with the Explorer pane on the left displaying the file structure of 'OPERATING-SYSTEM-1144 [WSL: UBUNTU]'. The file 'Q2.c' is selected. The main editor displays the code for 'Q2.c', which includes `<stdio.h>` and `<pthread.h>`. It defines a function `*print_number(void *arg)` that casts the argument to an integer, prints it, prints its square, and returns NULL. The `main()` function calls `pthread_create` to start a new thread with the argument 42. The TERMINAL pane at the bottom shows the execution of `gcc Q2.c -o Q2 -lpthread` and the output of the program: 'Thread received number: 42' and 'Square: 1764'.

```
1 #include <stdio.h>
2 #include <pthread.h>
3 void *print_number(void *arg)
4 {
5     // We know that we've passed an integer pointer
6     int num = *(int *)arg; // Cast void* back to int*
7     printf("Thread received number: %d\n", num);
8     printf("Square: %d\n", num * num);
9     return NULL;
10 }
11 int main()
12 {
13     pthread_t t1;
14     pthread_create(&t1, NULL, print_number, (void *)42);
15     pthread_join(t1, NULL);
16     printf("Main thread done.\n");
17     return 0;
18 }
```

```
bash - Lab4-labwork
Main thread starting...
Main thread exiting...
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork$ gcc Q2.c -o Q2 -lpthread
cc1: fatal error: Q2.c: No such file or directory
compilation terminated.
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork$ gcc Q2.c -o Q2 -lpthread
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork$ ./Q2
Creating thread with argument: 42
Thread received number: 42
Square: 1764
Main thread done.
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork$
```

Program 3 is about passing values or parameters to the threads.



The screenshot shows the Visual Studio Code interface with the Explorer pane on the left displaying the file structure of 'LAB4-LABWORK.0 [WSL: UBUNTU]'. The file 'Q3.c' is selected. The main editor displays the code for 'Q3.c', which includes `<stdio.h>` and `<pthread.h>`. It defines a function `*printData(void *arg)` that prints the data and message. The `main()` function creates two threads, `t1` and `t2`, with data `{1, "Hello"}` and `{2, "World"}` respectively. It joins both threads and prints 'All threads done.'. The TERMINAL pane at the bottom shows the execution of `gcc Q3.c -o Q3 -lpthread` and the output of the program: 'Thread 1 says: Hello' and 'Thread 2 says: World'.

```
8 void *printData(void *arg)
9 {
10     printf("Thread %d says: %s\n", data->id, data->message);
11     return NULL;
12 }
13
14 int main()
15 {
16     pthread_t t1, t2;
17     ThreadData data1 = {1, "Hello"};
18     ThreadData data2 = {2, "World"};
19     pthread_create(&t1, NULL, printData, &data1);
20     pthread_create(&t2, NULL, printData, &data2);
21     pthread_join(t1, NULL);
22     pthread_join(t2, NULL);
23     printf("All threads done.\n");
24     return 0;
25 }
```

```
bash - Lab4-labwork.0
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ gcc Q3.c -o Q3 -lpthread
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ ./Q3
Thread 1 says: Hello
Thread 2 says: World
All threads done.
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ gcc Q4 -o Q4 -lpthread
gcc: fatal error: input file 'Q4' is the same as output file
compilation terminated.
```

The 4 program is about creating multiple threads and is also about that main thread has to wait until all threads finish their job.

```
File Edit Selection View ... Lab4-labwork.0 [WSL: Ubuntu]
EXPLORER LAB4-LABWORK.0 [WSL: UBUNTU]
  C Q1.c
  C Q2.c
  C Q3.c
  Q4
  C Q4.c
OUTLINE
TIMELINE

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ gcc Q4 -o Q4 -lpthread
compilation terminated.
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ gcc Q4.c -o Q4 -lpthread
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ ./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 1: Work completed!
Thread 2: Work completed!
Thread 3: Work completed!
Thread 4: Work completed!
Thread 5: Work completed!
Main: Thread 1 has finished
Main: Thread 2 has finished
Main: Thread 3 has finished
Main: Thread 4 has finished
Main: Thread 5 has finished
All threads completed!
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ ./Q4
Main: Creating thread 1
```

The 5th program describes how a thread can return value.

```
File Edit Selection View ... Lab4-labwork.0 [WSL: Ubuntu]
EXPLORER LAB4-LABWORK.0 [WSL: UBUNTU]
  C Q1.c
  C Q2.c
  C Q3.c
  C Q4.c
  Q5
  C Q5.c
OUTLINE
TIMELINE

C Q5.c calculate_sum(void *)
18 pthread_t thread_id;
19 int n = 100;
20 void *sum;
21 pthread_create(&thread_id, NULL, calculate_sum, &n);
22 // Get the return value from thread
23 pthread_join(thread_id, &sum);
24 printf("Main received result: %d\n", *(int *)sum);
25 free(sum); // Don't forget to free allocated memory
26 return 0;
27 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ ./Q4
Main: Thread 4 has finished
Main: Thread 5 has finished
All threads completed!
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ gcc Q5.c -o Q5 -lpthread
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ ./Q5
Thread calculated sum of 1 to 100 = 5050
Main received result: 5050
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$
```

In practice task 1 we have created 3 threads and printed their id and unique message. We also have used the “join” to wait for threads to complete before termination of process.

The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying a project named 'LAB4-LABWORK.0 [WSL: UBUNTU]'. The file 'Q6.c' is selected. The main editor displays the code for 'Q6.c', which includes headers for `<stdio.h>`, `<pthread.h>`, and `<unistd.h>`. It defines a `ThreadData` struct with `int id` and `char *message`. A `printData` function is defined to print the thread's ID and message. The `main` function creates an array of `pthread_t` threads and calls `pthread_t threads[3];`. The terminal at the bottom shows the execution of `./Q6`, resulting in the following output:

```
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ ./Q6
Thread 1 says: Hello
Thread 3 says: from here
Thread 2 says: World
All threads completed!
```

In the last program we checked whether a number is prime or not using thread and passing data to that thread and also return value from that thread in main.

The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying a project named 'LAB4-LABWORK.0 [WSL: UBUNTU]'. The file 'Q7.c' is selected. The main editor displays the code for 'Q7.c', which includes headers for `<stdio.h>`, `<pthread.h>`, and `<unistd.h>`. It defines a `checkPrimeNumber` function that takes a thread ID, a pointer to a `pthread_t` variable, and a pointer to an `int` variable. The `main` function creates a thread and calls `pthread_create` and `pthread_join` to pass data and retrieve the result. The terminal at the bottom shows the execution of `gcc Q7.c -o Q7 -lpthread` and `./Q7`, resulting in the following output:

```
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ gcc Q7.c -o Q7 -lpthread
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ ./Q7
17 is not a prime number and result is 0
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ gcc Q7.c -o Q7 -lpthread
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ ./Q7
17 is a prime number and result is 1
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ gcc Q7.c -o Q7 -lpthread
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$ ./Q7
17 is a prime number
dawar@DESKTOP-EG875J4:~/operating-system-1144/Lab4-labwork.0$
```