



# **National Textile University**

## **Department of Computer Science**

Subject:

**Operating System**

---

Submitted to:

**Sir Nasir**

---

Submitted by:

**Esha Mubashir Khan**

---

Reg number:

**23-NTU-CS-1151**

---

Lab no: 4

---

Semester: 5<sup>th</sup>

---

### 3. C Programs with Threads

#### Program 1: Creating a Simple Thread

```
#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

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());

    pthread_create(&thread_id, NULL, thread_function, NULL);

    pthread_join(thread_id, NULL);

    printf("Main thread exiting...\n");

    return 0;

}
```

The screenshot shows the Visual Studio Code interface with a C file named `thread1.c` open. The code defines a `thread_function` and a `main` function that creates and joins a thread. The terminal at the bottom shows the compilation and execution of the program, displaying the output of the `printf` statements.

```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <unistd.h>
4 void* thread_function(void* arg) {
5     printf("Hello from the new thread!\n");
6     printf("Thread ID: %lu\n", pthread_self());
7     return NULL;
8 }
9 int main() {
10     pthread_t thread_id;
11     printf("Main thread starting...\n");
12     printf("Main Thread ID: %lu\n", pthread_self());
13     pthread_create(&thread_id, NULL, thread_function, NULL);
14     pthread_join(thread_id, NULL);
15     printf("Main thread exiting...\n");
16     return 0;
17 }
```

```
esha@ESHA-DELL:~/1151-lab4$ gcc thread1.c -o thread1 -lpthread
esha@ESHA-DELL:~/1151-lab4$ ./thread1
Main thread starting...
Main Thread ID: 138966501074752
Hello from the new thread!
Thread ID: 138966498014912
Main thread exiting...
esha@ESHA-DELL:~/1151-lab4$
```

## Program 2: Passing Arguments to Threads

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

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

```
void* print_number(void* arg) {
```

```
    int num = *(int*)arg;
```

```
    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);
```

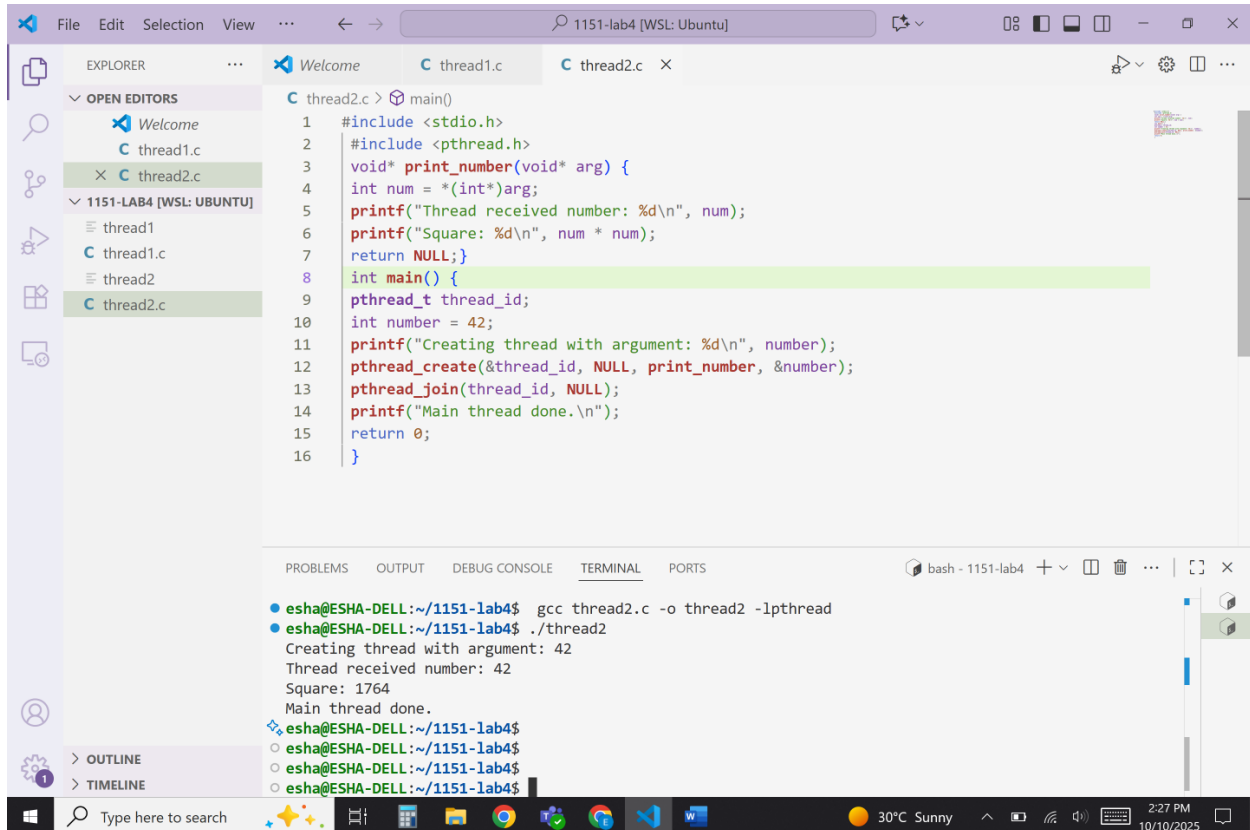
```
    pthread_create(&thread_id, NULL, print_number, &number);
```

```
    pthread_join(thread_id, NULL);
```

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

```
return 0;
```

```
}
```



The screenshot shows the Visual Studio Code editor with a C file named `thread2.c` open. The code defines a `print_number` function and a `main` function that creates a thread, prints the received number and its square, and then prints "Main thread done." before returning 0. The terminal at the bottom shows the compilation and execution of the program, with output matching the code's logic.

```
C thread2.c > main()
1  #include <stdio.h>
2  #include <pthread.h>
3  void* print_number(void* arg) {
4      int num = *(int*)arg;
5      printf("Thread received number: %d\n", num);
6      printf("Square: %d\n", num * num);
7      return NULL;
8  int main() {
9      pthread_t thread_id;
10     int number = 42;
11     printf("Creating thread with argument: %d\n", number);
12     pthread_create(&thread_id, NULL, print_number, &number);
13     pthread_join(thread_id, NULL);
14     printf("Main thread done.\n");
15     return 0;
16 }
```

```
bash - 1151-lab4
● esha@ESHA-DELL:~/1151-lab4$ gcc thread2.c -o thread2 -lpthread
● esha@ESHA-DELL:~/1151-lab4$ ./thread2
Creating thread with argument: 42
Thread received number: 42
Square: 1764
Main thread done.
● esha@ESHA-DELL:~/1151-lab4$
○ esha@ESHA-DELL:~/1151-lab4$
○ esha@ESHA-DELL:~/1151-lab4$
○ esha@ESHA-DELL:~/1151-lab4$
```

## 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, t2;

ThreadData data1 = {1, "Hello"};

ThreadData data2 = {2, "World"};

pthread_create(&t1, NULL, printData, &data1);

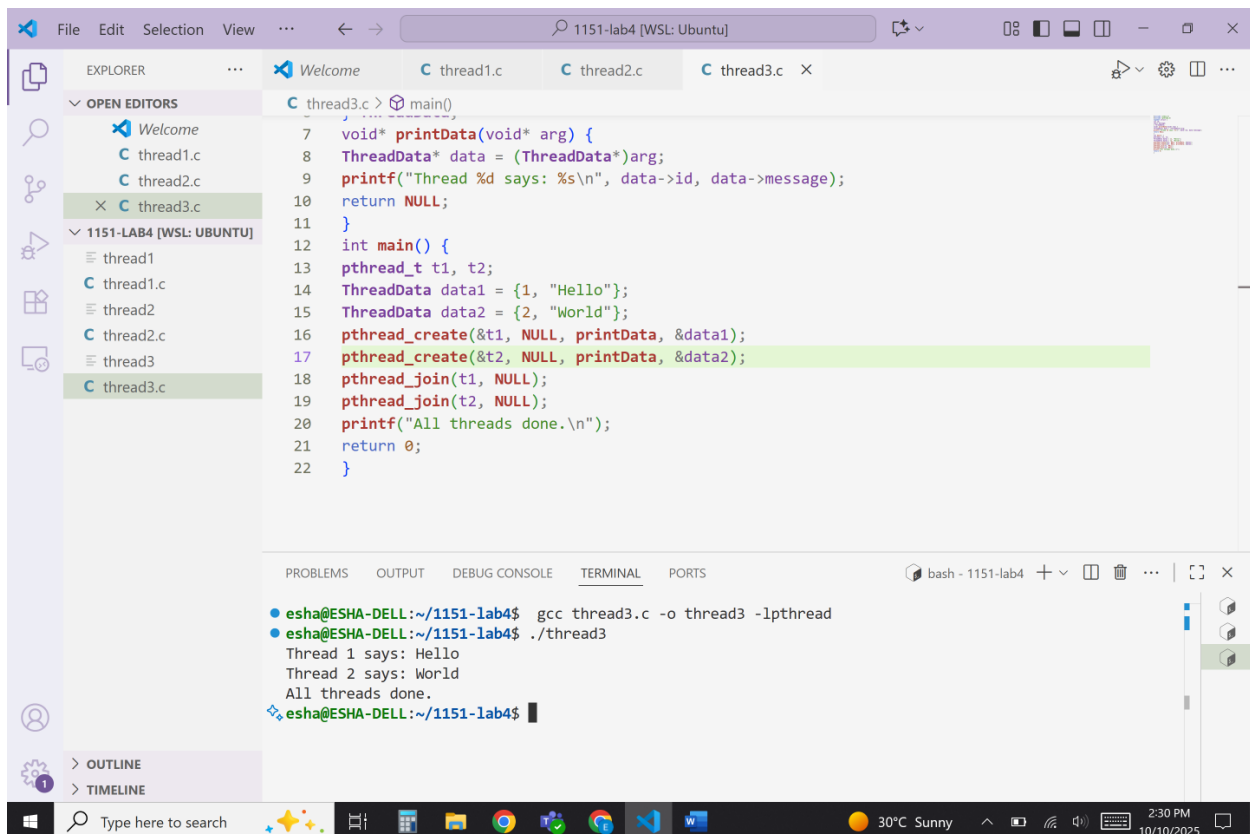
pthread_create(&t2, NULL, printData, &data2);

pthread_join(t1, NULL);

pthread_join(t2, NULL);

printf("All threads done.\n");

return 0; }
```



```
File Edit Selection View ... 1151-lab4 [WSL: Ubuntu]

EXPLORER
  OPEN EDITORS
    Welcome
    thread1.c
    thread2.c
    thread3.c
  1151-LAB4 [WSL: UBUNTU]
    thread1
    thread1.c
    thread2
    thread2.c
    thread3
    thread3.c

C thread3.c > main()
7 void* printData(void* arg) {
8   ThreadData* data = (ThreadData*)arg;
9   printf("Thread %d says: %s\n", data->id, data->message);
10  return NULL;
11 }
12 int main() {
13   pthread_t t1, t2;
14   ThreadData data1 = {1, "Hello"};
15   ThreadData data2 = {2, "World"};
16   pthread_create(&t1, NULL, printData, &data1);
17   pthread_create(&t2, NULL, printData, &data2);
18   pthread_join(t1, NULL);
19   pthread_join(t2, NULL);
20   printf("All threads done.\n");
21   return 0;
22 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - 1151-lab4
esha@ESHA-DELL:~/1151-lab4$ gcc thread3.c -o thread3 -lpthread
esha@ESHA-DELL:~/1151-lab4$ ./thread3
Thread 1 says: Hello
Thread 2 says: World
All threads done.
esha@ESHA-DELL:~/1151-lab4$
```