



**National Textile University**  
**Department of Computer Science**

**Subject**  
**Operating System**  
**Submitted to:**

Sir Nasir Mehmood

---

**Submitted by:**

Kashmir Jamshaid

---

**Registration Number**  
**23-NTU-CS-1167**

---

**Lab No.**  
**05**

---

**Semester**  
**5th**

---

## Task 1:

**Objective: Pass data to a thread function.**

### Code :

```
#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

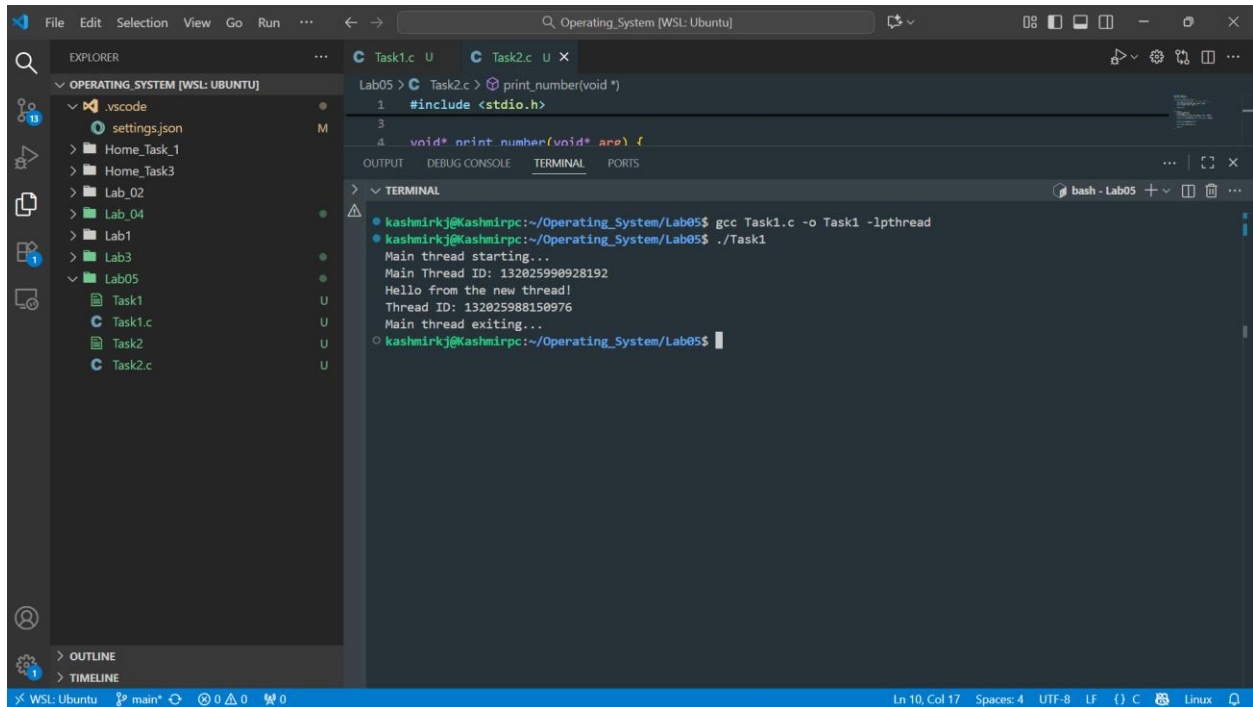
// Thread function - this will run in the new thread
void* thread_function(void* arg) {
    printf("Hello from the new thread!\n");
    printf("Thread ID: %lu\n", pthread_self());
    return NULL;
}

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;
}
```

## Output:



The screenshot shows a VS Code editor window titled 'Operating\_System [WSL: Ubuntu]'. The Explorer panel on the left shows a project structure with folders 'Home\_Task\_1', 'Home\_Task\_3', 'Lab\_02', 'Lab\_04', 'Lab1', 'Lab3', and 'Lab05'. Inside 'Lab05', there are files 'Task1', 'Task1.c', 'Task2', and 'Task2.c'. The main editor displays 'Task2.c' with the following code:

```
Lab05 > C Task2.c > print_number(void *)
1 #include <stdio.h>
3
4 void* print_number(void* arg) {
```

The TERMINAL panel at the bottom shows the execution output:

```
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ gcc Task1.c -o Task1 -lpthread
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ ./Task1
Main thread starting...
Main Thread ID: 132025990928192
Hello from the new thread!
Thread ID: 132025988150976
Main thread exiting...
kashmirkj@Kashmirpc:~/Operating_System/Lab05$
```

## Task2:

Pass data to a thread function.

### Code:

```
#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*
```

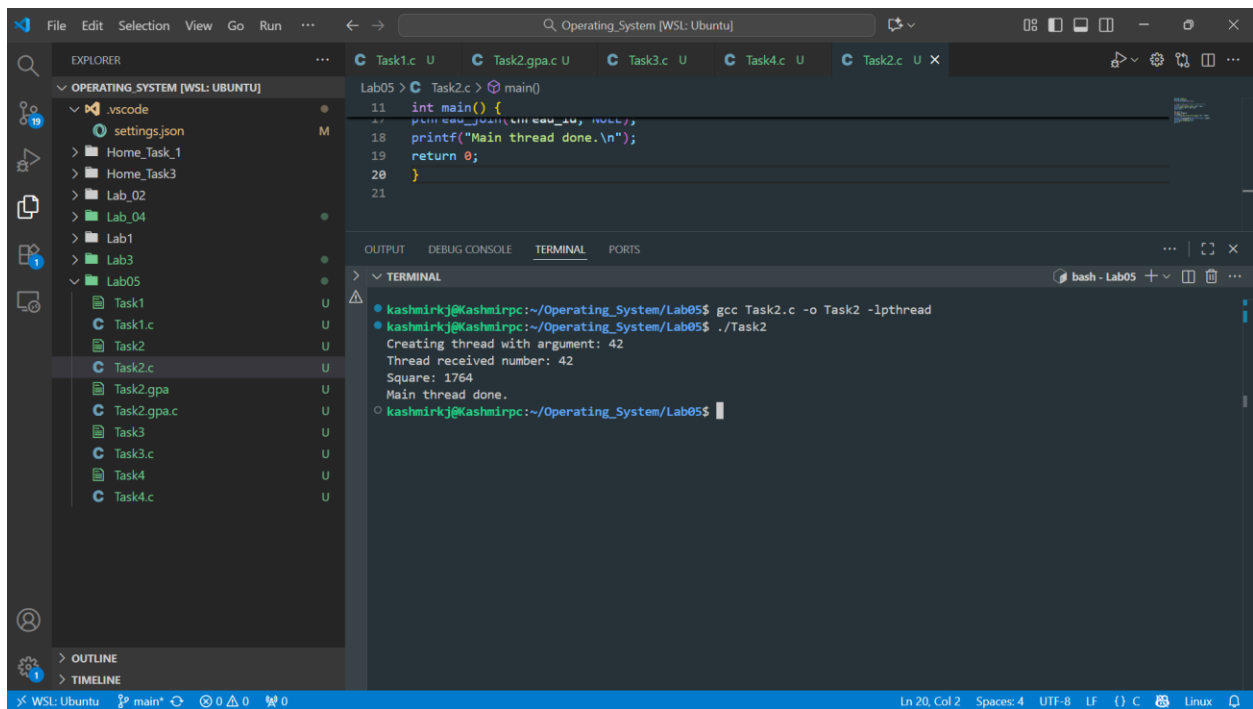
```
printf("Thread received number: %d\n", num);
```

```
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;  
}
```

**OutPut:**



## Task2.gpa :

### Code :

```
#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);
```

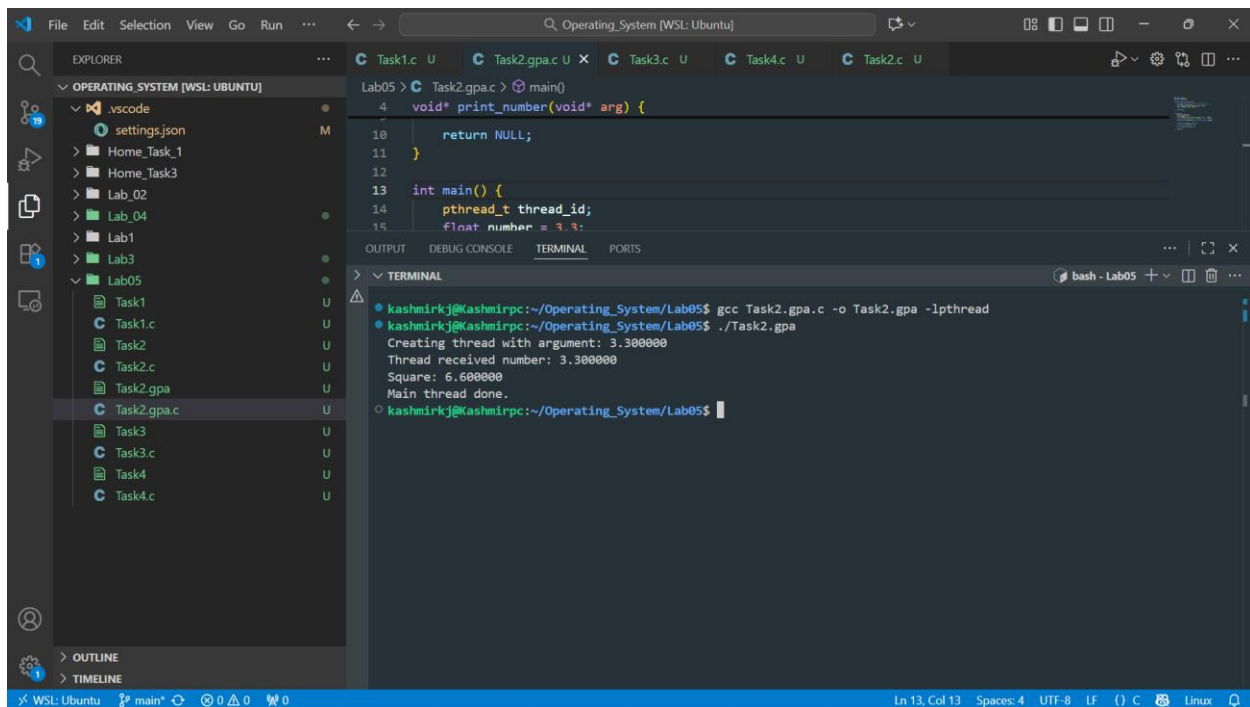
```
    printf("Square: %f\n", num * 2);
```

```
    return NULL;
```

```
}
```

```
int main() {  
    pthread_t thread_id;  
    float number = 3.3;  
    printf("Creating thread with argument: %f\n", number);  
    // Pass address of 'number' to thread  
    pthread_create(&thread_id, NULL, print_number, &number);  
  
    // Wait for the thread to finish  
    pthread_join(thread_id, NULL);  
  
    printf("Main thread done.\n");  
  
    return 0;  
}
```

**OutPut:**



## Task3

### Passing Multiple Data(My name && Gpa )

#### Code:

```

#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);
}

```

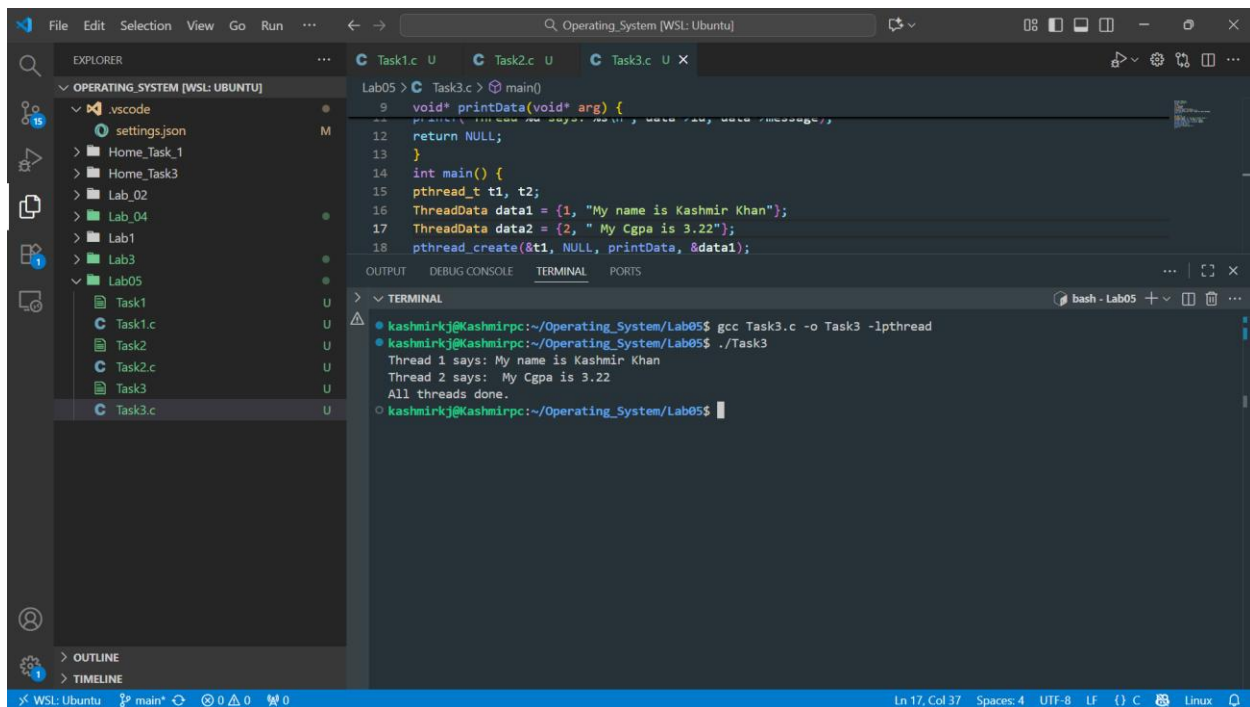
```
return NULL;

}

int main() {
pthread_t t1, t2;
ThreadData data1 = {1, "My name is Kashmir Khan"};
ThreadData data2 = {2, " My Cgpa is 3.22"};
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;
}
```

**Output:**





## Task3.a:

### Passing Multiple Data

#### Code:

```
#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);

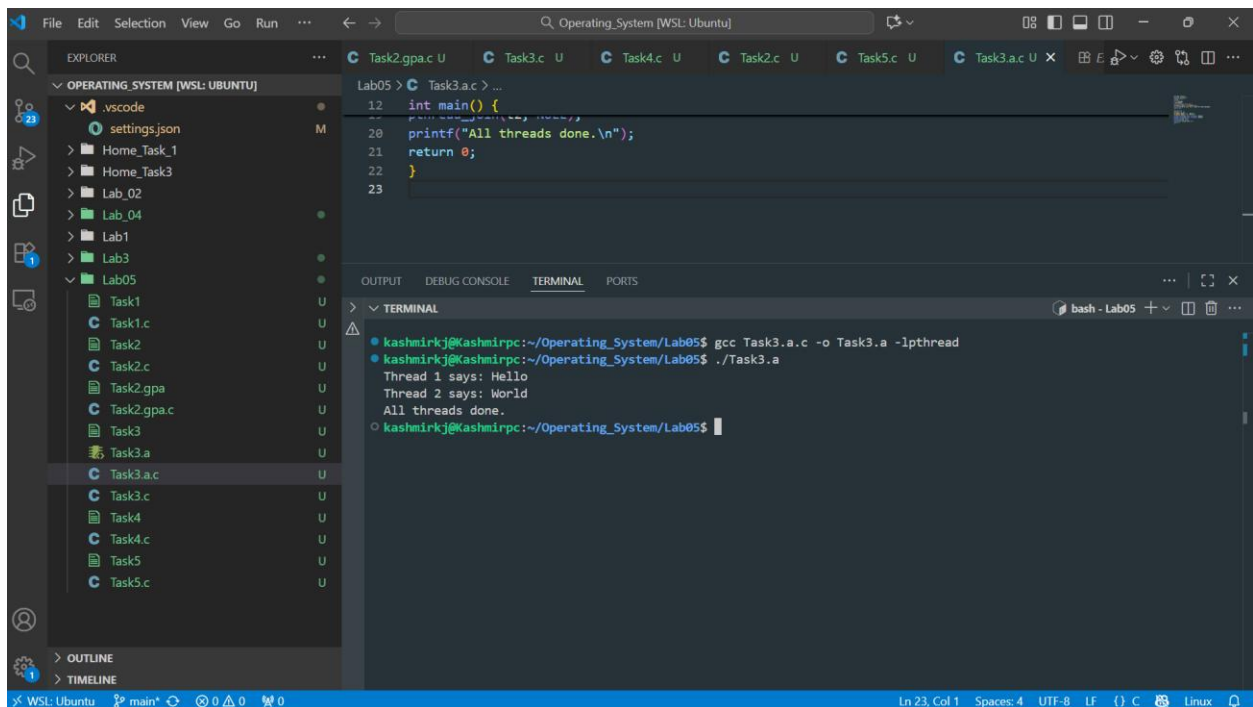
pthread_join(t1, NULL);
pthread_join(t2, NULL);

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

return 0;
}

```

## Output:



The screenshot shows the Visual Studio Code interface with a C program in the editor and its output in the terminal. The Explorer panel on the left shows the project structure, including a folder named 'Lab05' containing several files. The editor displays the source code of 'Task3.a.c', which is a C program using pthreads to create two threads that print 'Hello' and 'World' before printing 'All threads done.'. The terminal window at the bottom shows the execution of the program, with the output matching the expected behavior of the code.

```

Lab05 > C Task3.a.c > ...
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  }
23

OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
> TERMINAL
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ gcc Task3.a.c -o Task3.a -lpthread
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ ./Task3.a
Thread 1 says: Hello
Thread 2 says: World
All threads done.
kashmirkj@Kashmirpc:~/Operating_System/Lab05$

```

## Task4

### Code:

```
#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

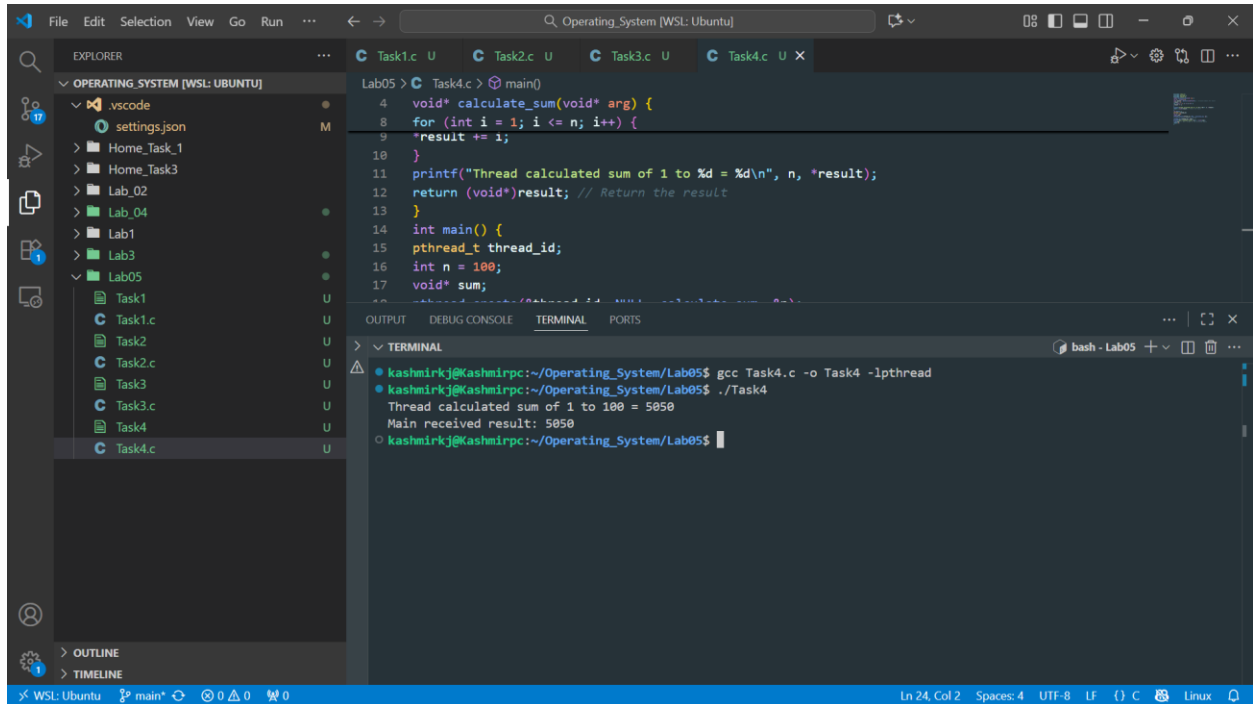
void* calculate_sum(void* arg) {
    int n = *(int*)arg;
    int* result = malloc(sizeof(int)); // Allocate memory for result
    *result = 0;
    for (int i = 1; i <= n; i++) {
        *result += i;
    }
    printf("Thread calculated sum of 1 to %d = %d\n", n, *result);
    return (void*)result; // 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;
```

```
}
```

## Output:



The screenshot shows a VS Code editor window titled 'Operating\_System [WSL: Ubuntu]'. The Explorer panel on the left shows a file tree for 'OPERATING\_SYSTEM [WSL: UBUNTU]' with files like '.vscode', 'settings.json', and several 'Lab' folders. The main editor area shows a C program 'Task4.c' with the following code:

```
Lab05 > C Task4.c > main()
4 void* calculate_sum(void* arg) {
8   for (int i = 1; i <= n; i++) {
9     *result += i;
10  }
11  printf("Thread calculated sum of 1 to %d = %d\n", n, *result);
12  return (void*)result; // Return the result
13 }
14 int main() {
15   pthread_t thread_id;
16   int n = 100;
17   void* sum;
18   pthread_create(&thread_id, NULL, calculate_sum, (void*)100);
19   pthread_join(thread_id, &sum);
20   printf("Main received result: %d\n", *(int*)sum);
21   return 0;
22 }
```

The bottom panel shows the 'TERMINAL' output:

```
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ gcc Task4.c -o Task4 -lpthread
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ ./Task4
Thread calculated sum of 1 to 100 = 5050
Main received result: 5050
kashmirkj@Kashmirpc:~/Operating_System/Lab05$
```

## Task5

### Program 1: Creating and Running Multiple Threads

#### Code:

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

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

```
#include <unistd.h>
```

```
void* worker(void* arg) {
```

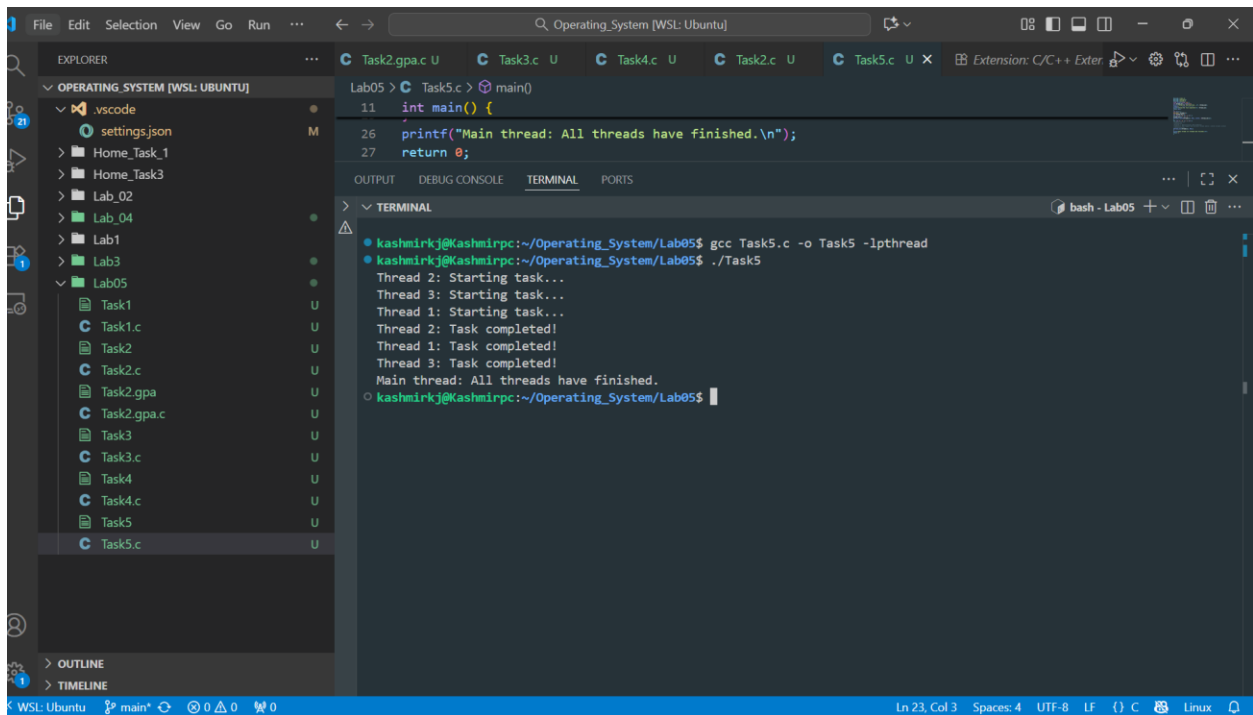
```
int thread_num = *(int*)arg;
```

```
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;  
}
```

**Output:**



## Task 6

What happens when multiple threads modify a shared variable without synchronization.

### Code:

```
#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);

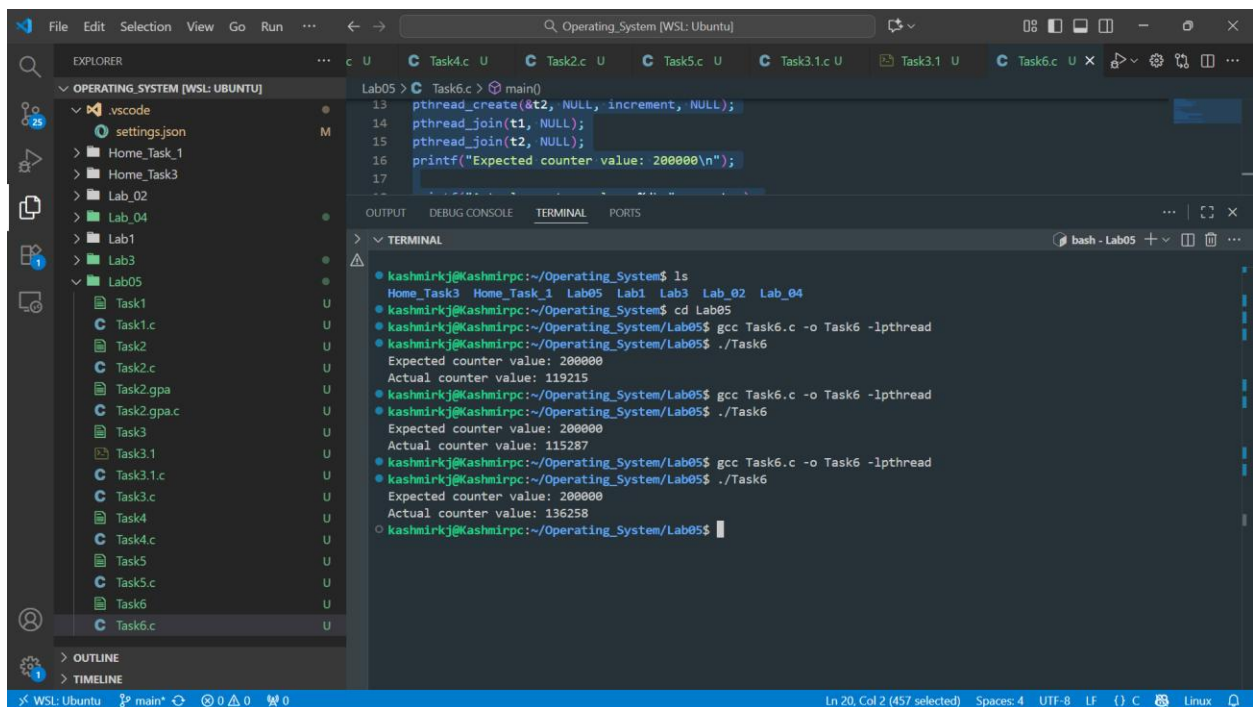
printf("Expected counter value: 200000\n");

printf("Actual counter value: %d\n", counter);

return 0;
}

```

## Output:



The screenshot shows a Visual Studio Code editor with a C program in the main editor and its output in the terminal. The program is a multi-threaded application that creates two threads, increments a counter, and prints the expected and actual values.

**Source Code (Task6.c):**

```

13 pthread_create(&t2, NULL, increment, NULL);
14 pthread_join(t1, NULL);
15 pthread_join(t2, NULL);
16 printf("Expected counter value: 200000\n");
17

```

**Terminal Output:**

```

kashmirkj@Kashmirpc:~/Operating_System$ ls
Home_Task3 Home_Task1 Lab05 Lab1 Lab3 Lab02 Lab04
kashmirkj@Kashmirpc:~/Operating_System$ cd Lab05
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ gcc Task6.c -o Task6 -lpthread
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ ./Task6
Expected counter value: 200000
Actual counter value: 119215
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ gcc Task6.c -o Task6 -lpthread
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ ./Task6
Expected counter value: 200000
Actual counter value: 115287
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ gcc Task6.c -o Task6 -lpthread
kashmirkj@Kashmirpc:~/Operating_System/Lab05$ ./Task6
Expected counter value: 200000
Actual counter value: 136258
kashmirkj@Kashmirpc:~/Operating_System/Lab05$

```

