National Textile University

Department of Computer Science

Subject:

Operating System

Submitted to:

Sir Nasir Mehmood

Submitted by:

Akasha Fatima

Reg. number:

23-NTU-CS-FL-1132

Semester:

5th - A

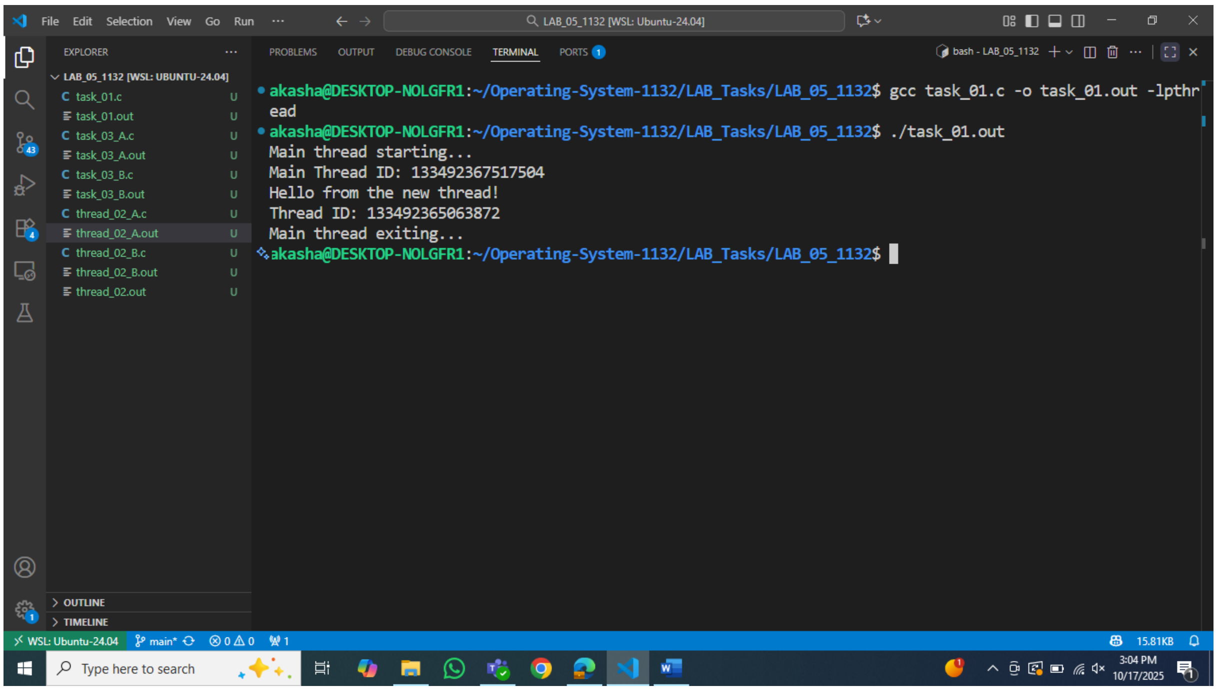
LAB_05

Task_01: Creating a simple thread

CODE:

```
// creating a simple thread

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

// Thread function - this will run in the new thread
void* thread_function(void* arg) {  // void* thread_function return type for
pthreads i.e., null
    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:



```
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ gcc task_01.c -o task_01.out -lpthr
ead
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ ./task_01.out
Main thread starting...
Main Thread ID: 133492367517504
Hello from the new thread!
Thread ID: 133492365063872
Main thread exiting...
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$
```

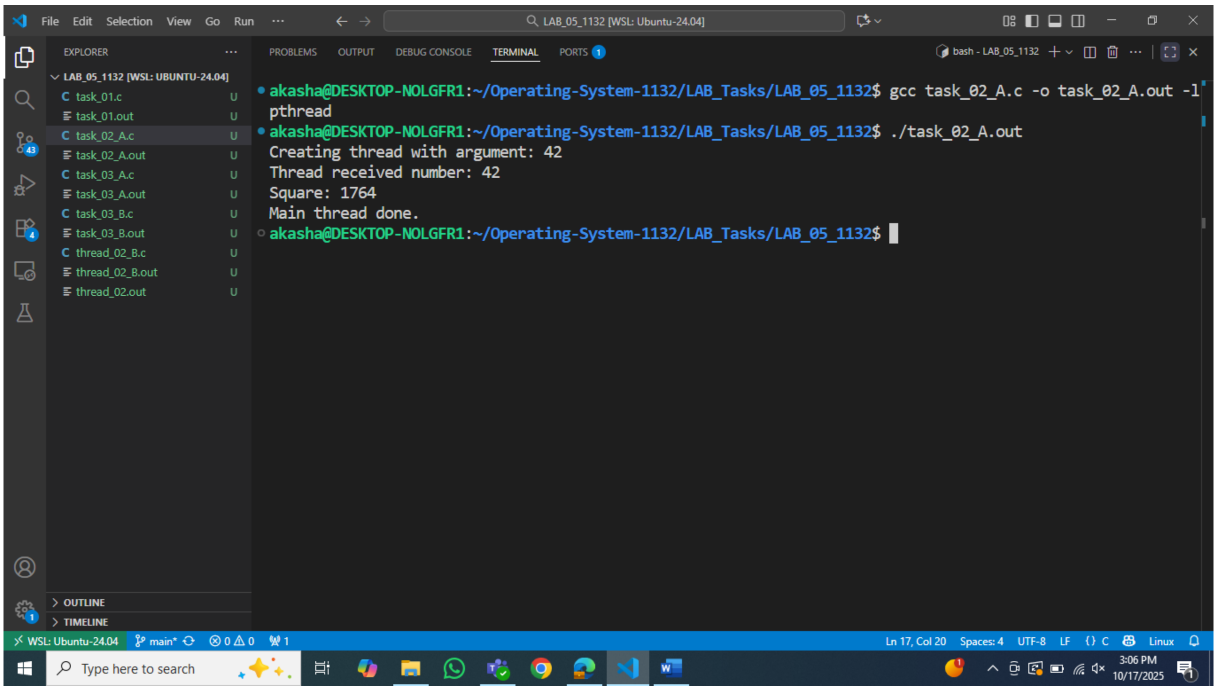
Task_02: Passing Arguments to Threads in C

CODE:

```
// Passing Arguments to Threads in C

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

int main() {
   pthread_t thread_id;
   float number = 3.79; // Example float number
   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");
```

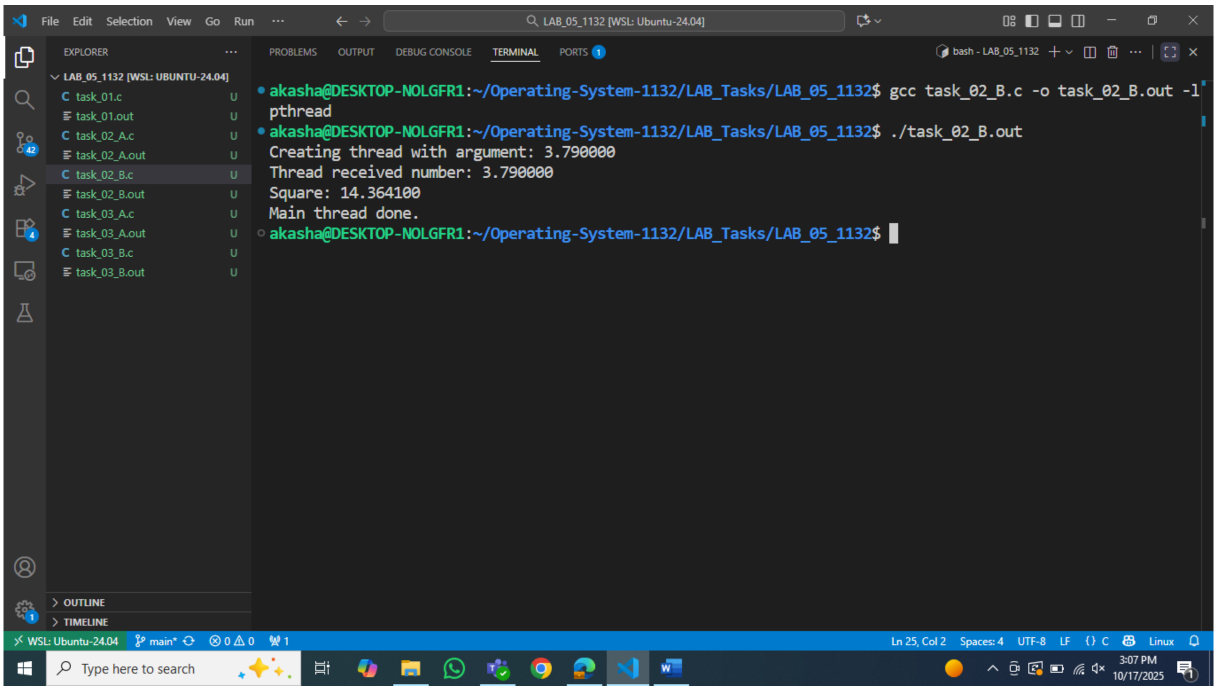
```
    return 0;
}
```

Output:



```
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ gcc task_02_A.c -o task_02_A.out -l
pthread
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ ./task_02_A.out
Creating thread with argument: 42
Thread received number: 42
Square: 1764
Main thread done.
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$
```

With CGPA:



```
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ gcc task_02_B.c -o task_02_B.out -l
pthread
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ ./task_02_B.out
Creating thread with argument: 3.790000
Thread received number: 3.790000
Square: 14.364100
Main thread done.
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$
```

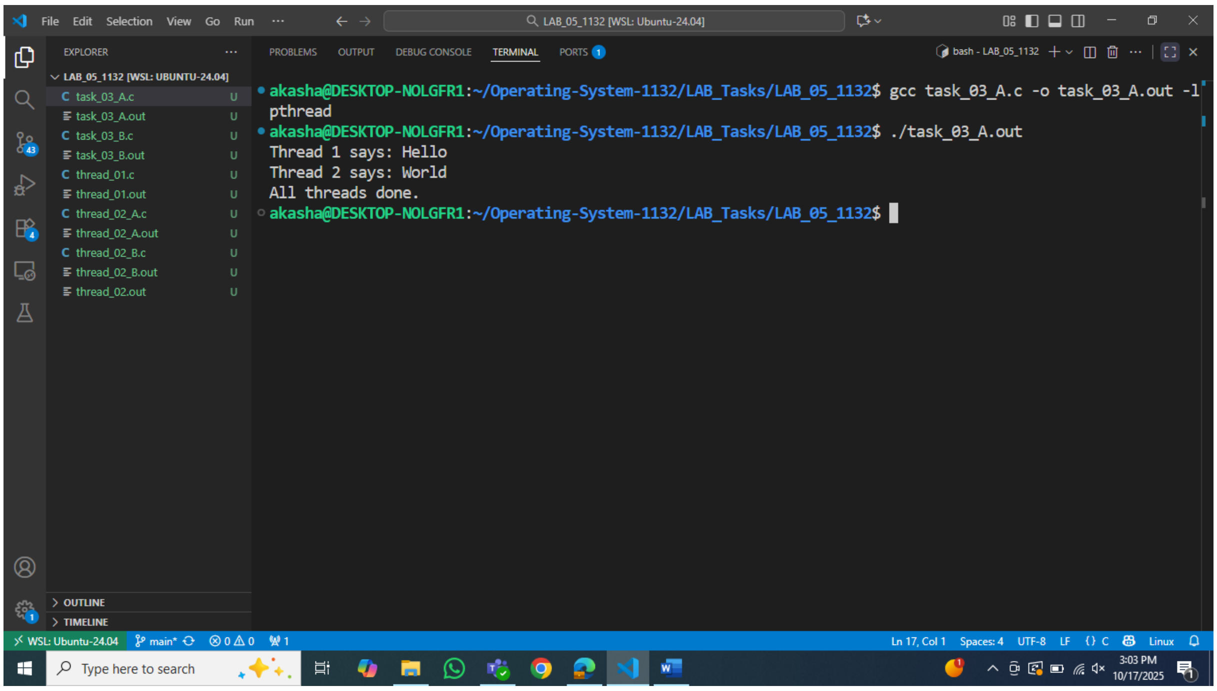
Task_03: Passing Multiple Arguments to Threads using Structs

CODE:

```
// Passing Multiple Arguments to Threads using Structs
```

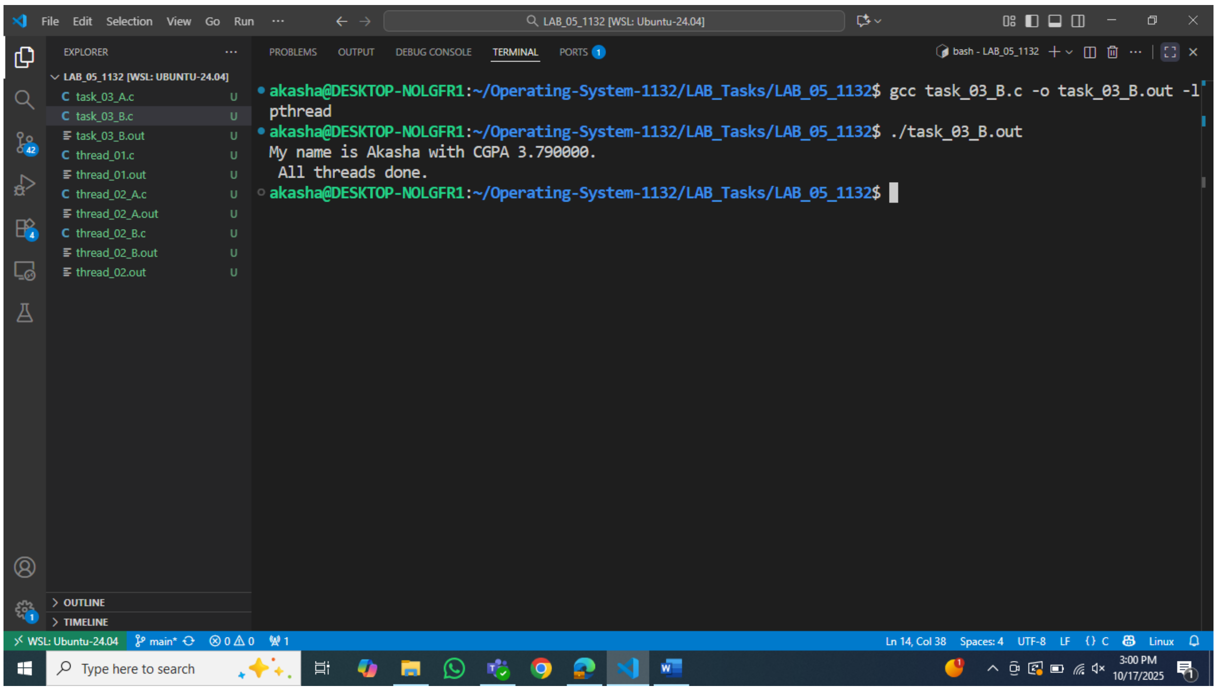
```c
#include <stdio.h>
#include <pthread.h>
// Define a struct to hold multiple arguments
typedef struct {
    char* message;
    float cgpa;
} ThreadData;
 // Thread function
void* printData(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    printf("My name is %s with CGPA %f.\n ", data->message, data->cgpa);
    return NULL;
}

int main() {
    pthread_t t1, t2;                    // Thread identifiers
    ThreadData data1 = {"Akasha", 3.79};
    pthread_create(&t1, NULL, printData, &data1);
    pthread_join(t1, NULL);
    printf("All threads done.\n");
    return 0;
}
```

Output:



```
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ gcc task_03_A.c -o task_03_A.out -l
pthread
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ ./task_03_A.out
Thread 1 says: Hello
Thread 2 says: World
All threads done.
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$
```

With Name and CGPA:



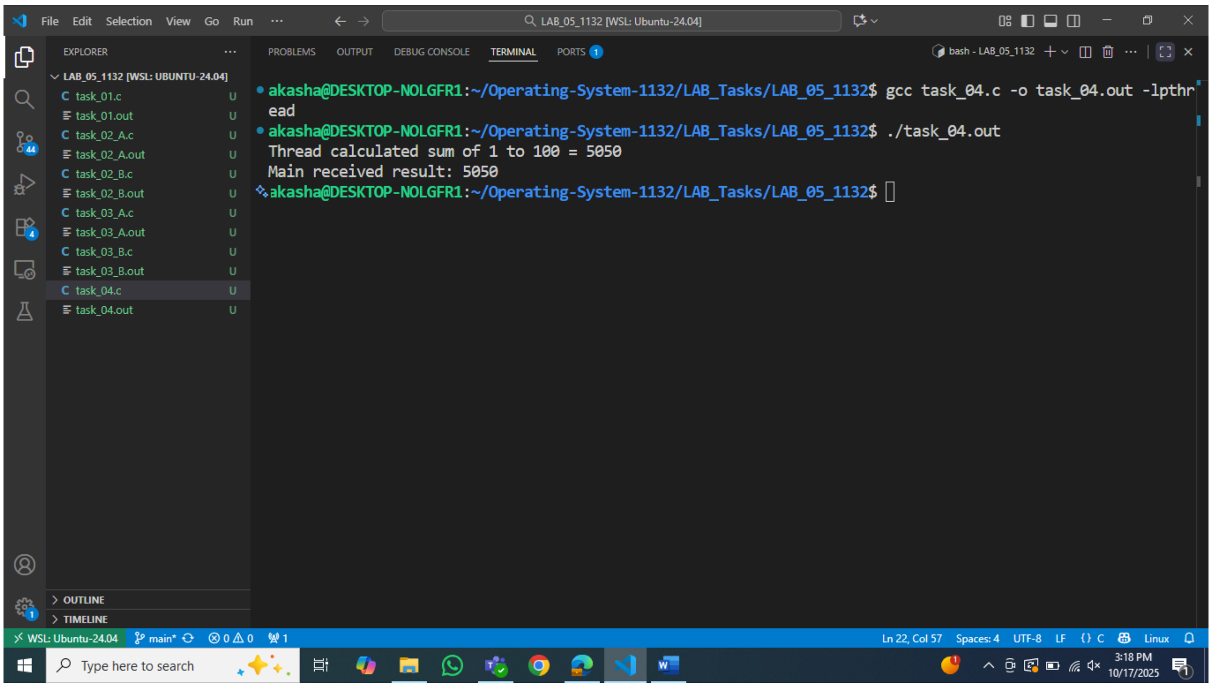
```
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ gcc task_03_B.c -o task_03_B.out -l
pthread
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ ./task_03_B.out
My name is Akasha with CGPA 3.790000.
 All threads done.
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$
```

Task_04: Threads Return Value

CODE:

```
// Threads return value
#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:



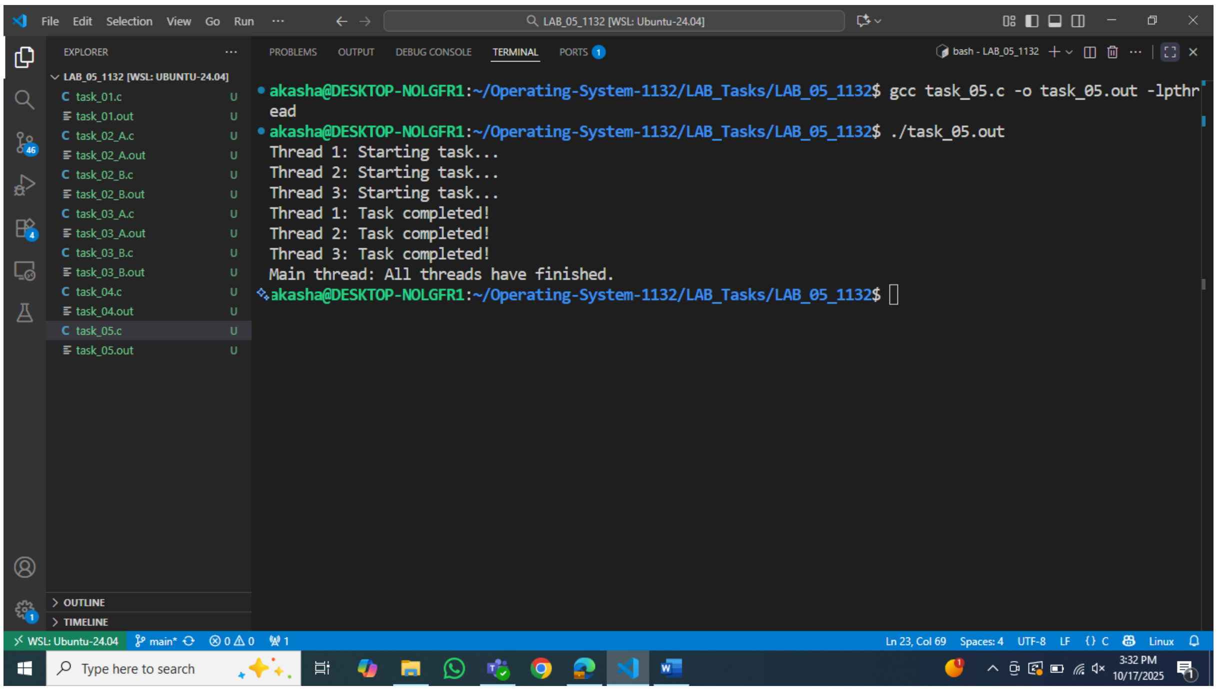
```
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ gcc task_04.c -o task_04.out -lpthr
ead
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ ./task_04.out
Thread calculated sum of 1 to 100 = 5050
Main received result: 5050
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$
```

Task_05: Creating and running multiple threads

CODE:

```
// creating and running multiple threads
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
void* worker(void* arg) {
    int thread_num = *(int*)arg;          // Get thread number
    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];        // Array to hold thread identifiers
int thread_ids[3];
for (int i = 0; i < 3; i++) {
    thread_ids[i] = i + 1;      // Thread numbers 1, 2, 3
    pthread_create(&threads[i], NULL, worker, &thread_ids[i]);// Create thread
}
for (int i = 0; i < 3; i++) {
    pthread_join(threads[i], NULL);     // Wait for thread to finish
}

printf("Main thread: All threads have finished.\n");
return 0;
}
```

Output:



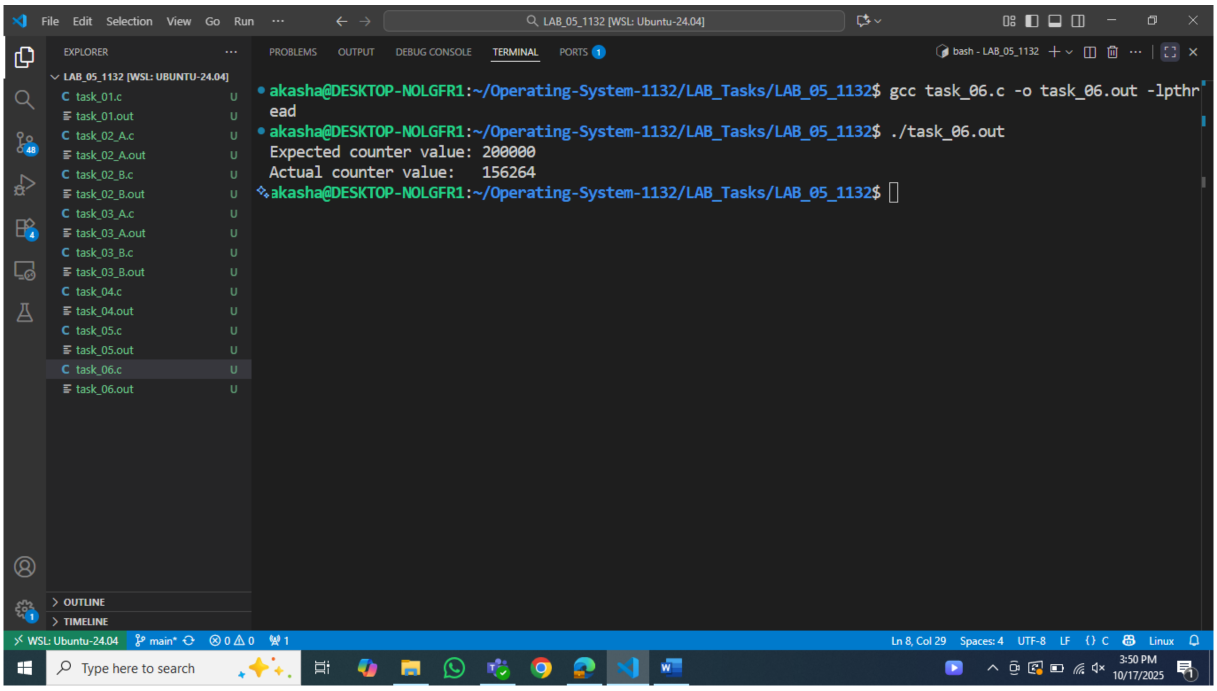
```
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ gcc task_05.c -o task_05.out -lpthr
ead
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ ./task_05.out
Thread 1: Starting task...
Thread 2: Starting task...
Thread 3: Starting task...
Thread 1: Task completed!
Thread 2: Task completed!
Thread 3: Task completed!
Main thread: All threads have finished.
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$
```

Task_06: Demostrating a race condition

CODE:

```
// demostrating a race condition
#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:



```
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ gcc task_06.c -o task_06.out -lpthr
ead
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ ./task_06.out
Expected counter value: 200000
Actual counter value:   156264
akasha@DESKTOP-NOLGFR1:~/Operating-System-1132/LAB_Tasks/LAB_05_1132$ 
```