

National Textile University

**Department of Computer Science**

Subject: Operating System

Submitted to: Sir Nasir Mahmood

Submitted by: Eisha Muzaffar

Reg. number: 23-NTU-CS-1147

Lab no: lab5

semester:5th



#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

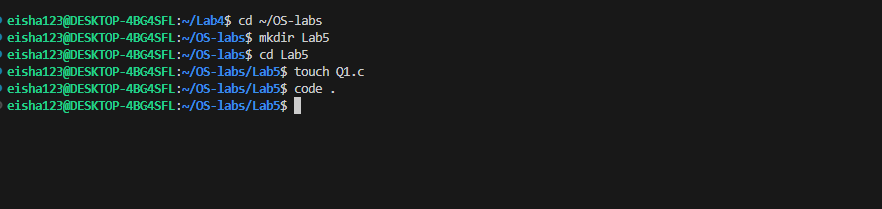
pthread\_create(&thread\_id, NULL, thread\_function, NULL);

pthread\_join(thread\_id, NULL);

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

return 0;

}



**Q2 CGPA:**  
#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;

float number = 3.5;

printf("Creating thread with argument: %d\n", number);

// Pass address of 'number' to thread

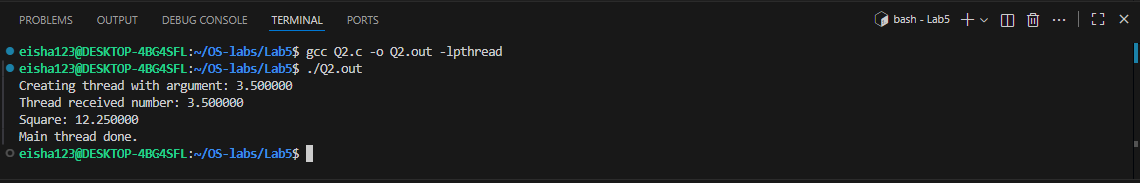
pthread\_create(&thread\_id, NULL, print\_number, &number);

pthread\_join(thread\_id, NULL);

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

return 0;

}  
**Terminal:**





#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);

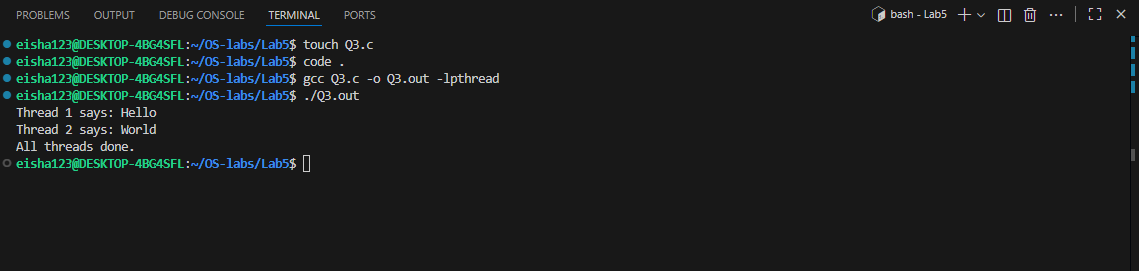
pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

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

return 0;

}

**Terminal:  
**

**Task 3.1 Name +CGPA:**

**//Multiple Threads (CGPA +NAME)**

#include <pthread.h>

typedef struct {

float id;

char\* message;

} ThreadData;

void\* printData(void\* arg) {

ThreadData\* data = (ThreadData\*)arg;

printf("Thread %f says: %s\n", data->id, data->message);

return NULL;

}

int main() {

pthread\_t t1;

ThreadData data1 = {1, "Eisha Muzaffar \n My CGPA is 3.5"};

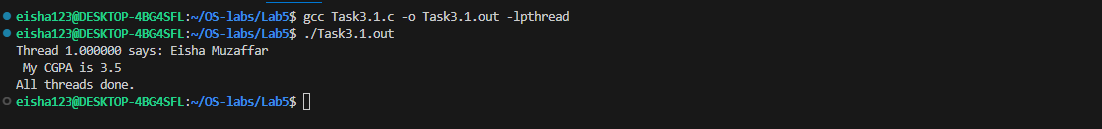
pthread\_create(&t1, NULL, printData, &data1);

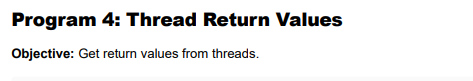
pthread\_join(t1, NULL);

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

return 0;

}

**Terminal:**

****

#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);

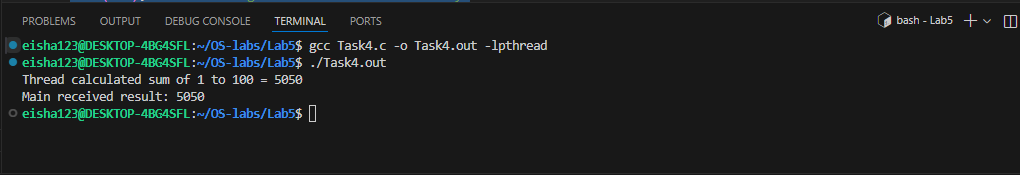
printf("Main received result: %d\n", \*(int\*)sum);

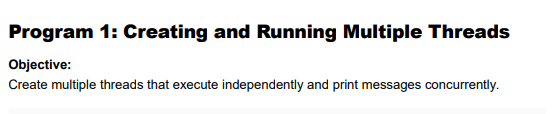
free(sum); // Don't forget to free allocated memory

return 0;

}

**Terminal:**

****

****

#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);

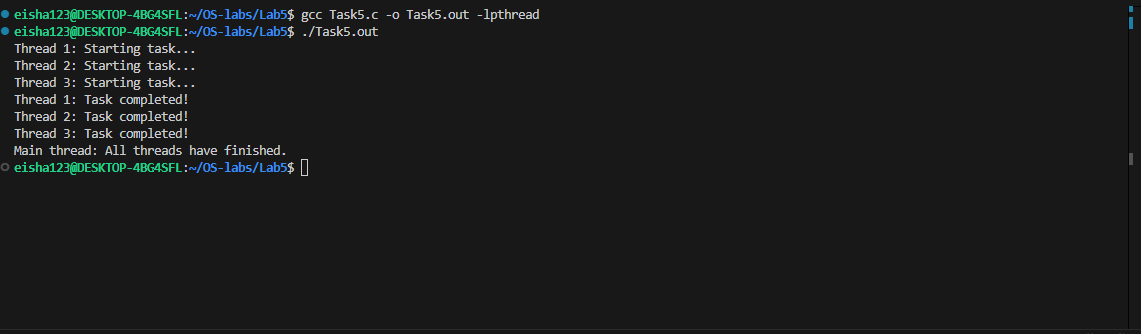
}

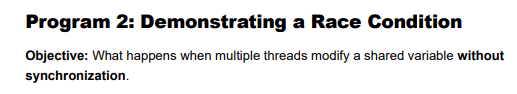
printf("Main thread: All threads have finished.\n");

return 0;

}

**Terminal:**





**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);

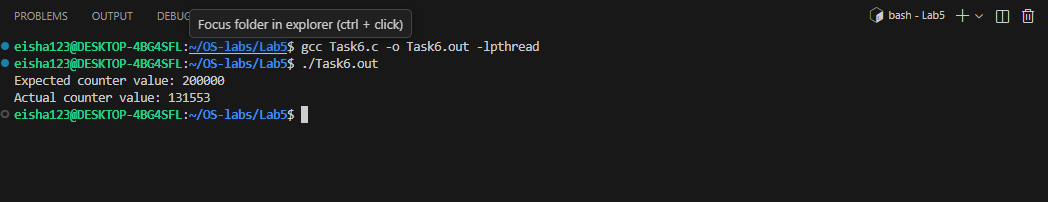
printf("Expected counter value: 200000\n");

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

return 0;

}

**Terminal:**

****