



National Textile University

Department of Computer Science

Subject: Operating System

Submitted to: Sir Nasir

Submitted by: Maha

Reg. number: 23-NTU-CS-1170

Lab 4 (class task)

Semester: 5th

3. C Programs with Threads

Program 1: Creating a Simple Thread

Objective: Create a thread and print messages from both main thread and new thread.

Code:

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

// Thread function - this will run in the new thread
void* fun(void* arg) {
    printf("Lab 4 class task of threads \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, fun, NULL);

    // Wait for the thread to finish
    pthread_join(thread_id, NULL);
    printf("Main thread exiting...\n");
}
```

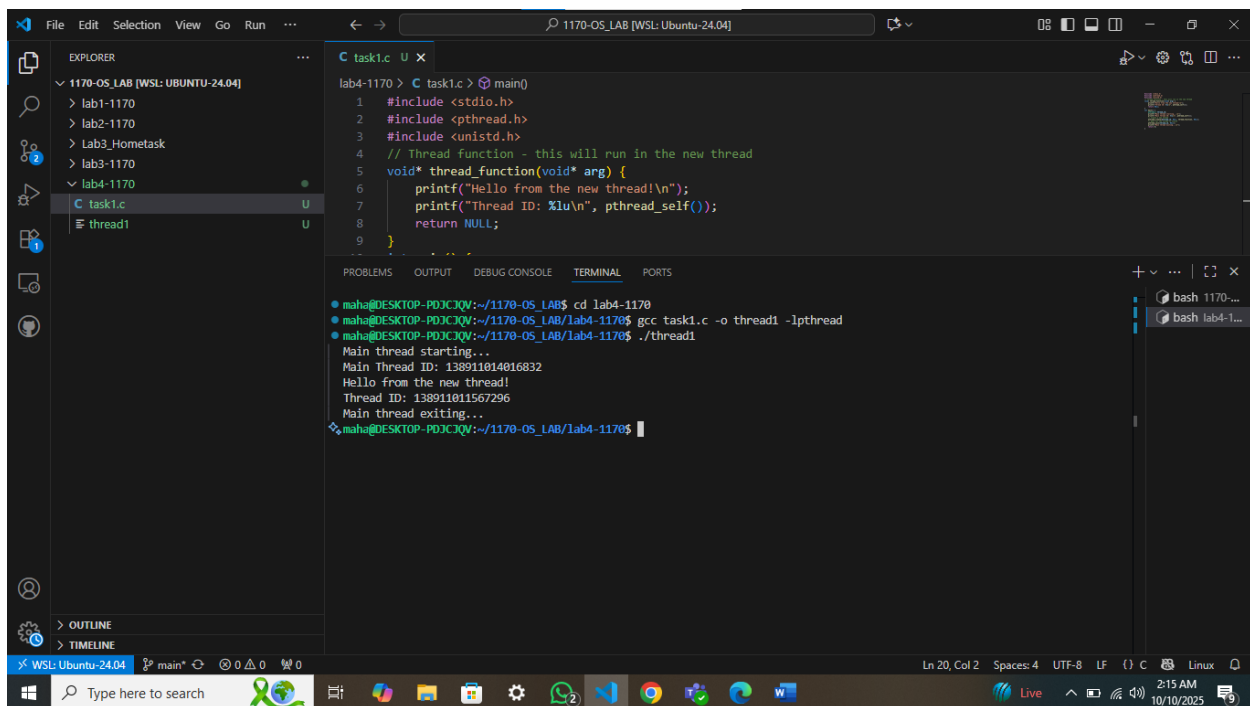
```
return 0;  
}
```

Compilation:

Compile and run:

```
gcc thread1.c -o thread1 -lpthread  
./thread1
```

Terminal:



The screenshot shows the Visual Studio Code interface with a C file named `task1.c` open. The code defines a thread function `thread_function` that prints a message and its thread ID. The terminal output shows the compilation and execution of the program, displaying the main thread ID and the new thread ID.

```
lab4-1170 > C task1.c > main()
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <unistd.h>
4 // Thread function - this will run in the new thread
5 void* thread_function(void* arg) {
6     printf("Hello from the new thread!\n");
7     printf("Thread ID: %lu\n", pthread_self());
8     return NULL;
9 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
maha@DESKTOP-PD3CJQV:~/1170-OS_LAB$ cd lab4-1170
maha@DESKTOP-PD3CJQV:~/1170-OS_LAB/lab4-1170$ gcc task1.c -o thread1 -lpthread
maha@DESKTOP-PD3CJQV:~/1170-OS_LAB/lab4-1170$ ./thread1
Main thread starting...
Main Thread ID: 138911014016832
Hello from the new thread!
Thread ID: 138911011567296
Main thread exiting...
```

Program 2: Passing Arguments to Threads

Objective: Pass data to a thread function.

Code:

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

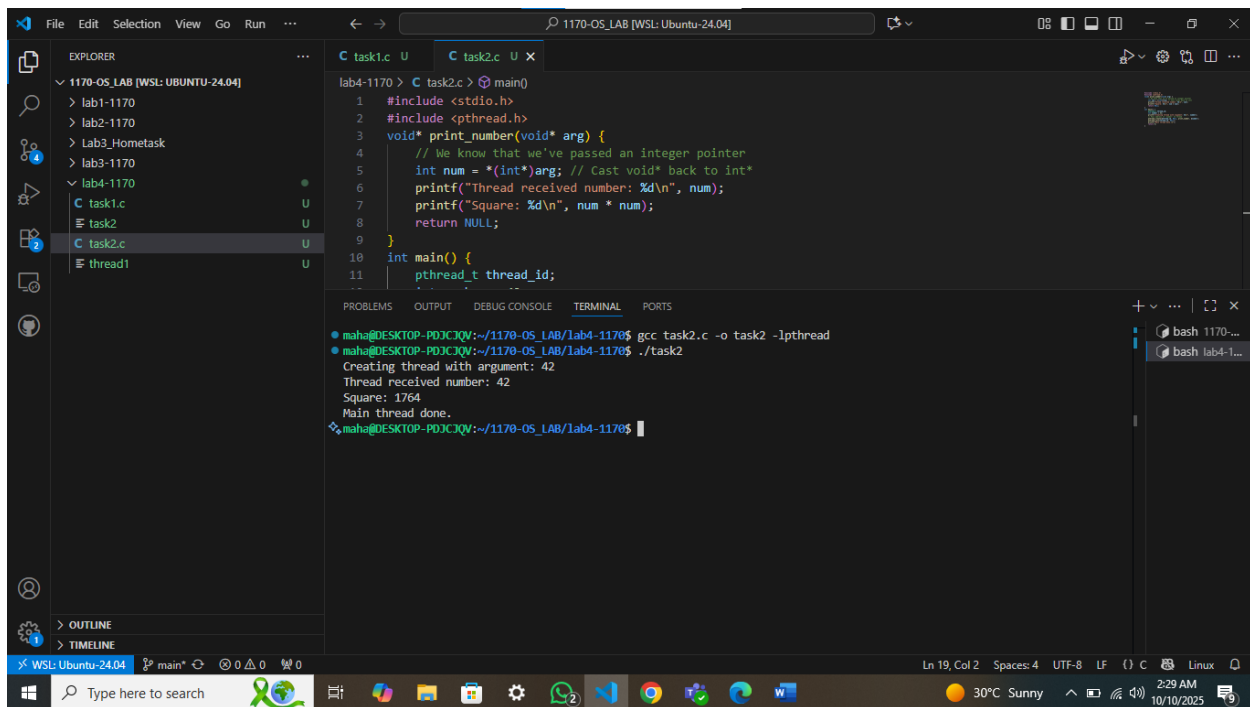
```
void* print_number(void* arg) {
    // We know that we've passed an integer pointer
    int num = *(int*)arg; // Cast void* back to int*
    printf("Thread received number: %d\n", num);
    printf("Square: %d\n", num * num);
    return NULL;
}

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;
}
```

Compile and run:

```
gcc thread2.c -o thread2 -lpthread
./thread2
```

Terminal:



Program 3: 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;
}

```

Terminal:

The screenshot shows the Visual Studio Code interface with a C program in 'task3.c' and its execution output in the terminal. The program creates two threads, 't1' and 't2', which print 'Hello' and 'World' respectively. The main thread then prints 'All threads done.' before returning 0.

```

lab4-1170 > C task3.c > main()
1  #include <stdio.h>
2  #include <pthread.h>
3
4  typedef struct {
5      int id;
6      char* message;
7  } ThreadData;
8
9  void* printData(void* arg) {
10     ThreadData* data = (ThreadData*)arg;
11     printf("Thread %d says: %s\n", data->id, data->message);
12 }

```

```

maha@DESKTOP-PDJCJQV:~/1170-OS_LAB/lab4-1170$ ./task3
Thread 1 says: Hello
Thread 2 says: World
All threads done.
maha@DESKTOP-PDJCJQV:~/1170-OS_LAB/lab4-1170$

```

Program 4: Multiple Threads

Objective: Create multiple threads executing the same function.

Code:

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

void* worker_thread(void* arg) {
    int thread_num = *(int*)arg;
    printf("Thread %d: Starting work...\n", thread_num);
    sleep(1); // Simulate some work
    printf("Thread %d: Work completed!\n", thread_num);
    return NULL;
}

int main() {
    pthread_t threads[5];
    int thread_args[5];
    // Create 5 threads
    for (int i = 0; i < 5; i++) {
        thread_args[i] = i + 1;
        printf("Main: Creating thread %d\n", i + 1);
        pthread_create(&threads[i], NULL, worker_thread, &thread_args[i]);
    }
    // Wait for all threads to complete
    for (int i = 0; i < 5; i++) {
```

```
pthread_join(threads[i], NULL);

printf("Main: Thread %d has finished\n", i + 1);

}

printf("All threads completed!\n");

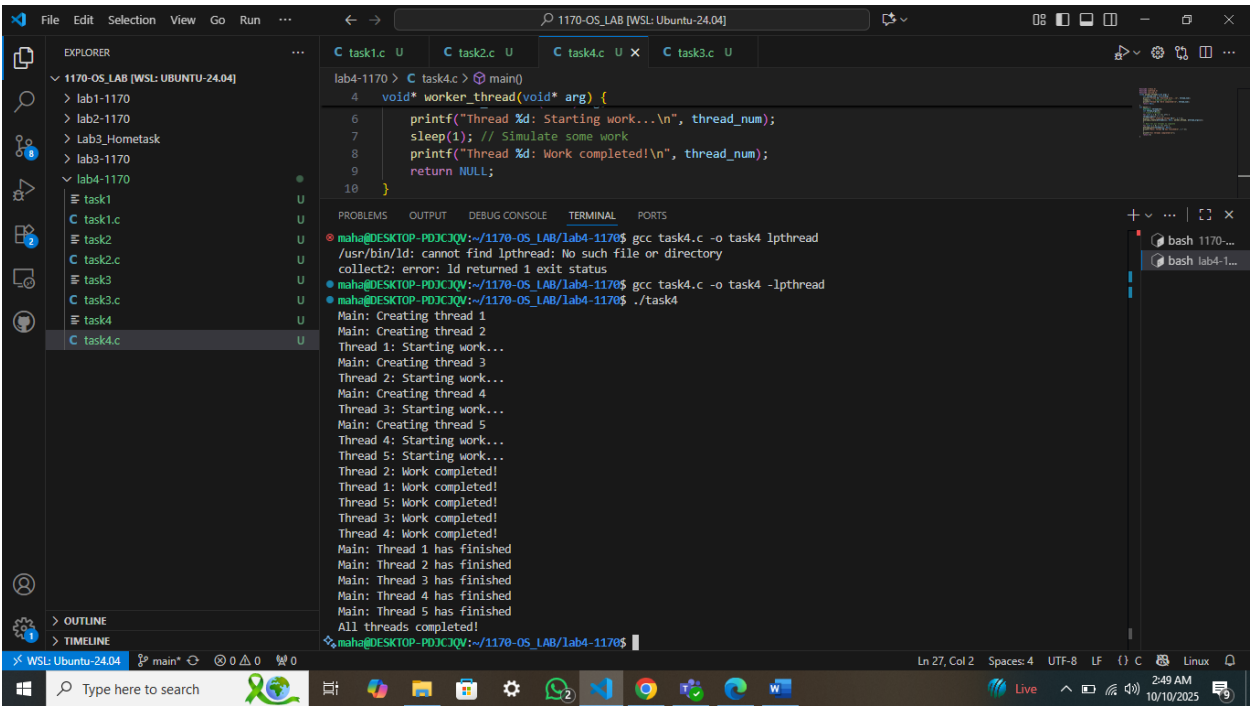
return 0;

}
```

Compile and run:

```
gcc thread3.c -o thread3 -lpthread
./thread3
```

Terminal:



Question 5:

Program 5: Thread Return Values

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;
}
```

}

Terminal:

The screenshot shows the Visual Studio Code interface with a C program being edited and executed. The Explorer panel on the left shows the project structure with folders 'lab1-1170', 'lab2-1170', 'lab3-Hometask', 'lab3-1170', and 'lab4-1170'. The 'lab4-1170' folder is expanded, showing files 'task1.c', 'task2.c', 'task3.c', 'task4.c', and 'task5.c'. The main editor displays the code for 'task5.c' with the following content:

```
lab4-1170 > C task5.c > main()
14 int main() {
22     free(sum); // Don't forget to free allocated memory
23     return 0;
24 }
```

The Output panel at the bottom shows the execution results:

```
maha@DESKTOP-PDJCJQV:~/1170-OS_LAB/lab4-1170$ gcc task5.c -o task5 -lpthread
maha@DESKTOP-PDJCJQV:~/1170-OS_LAB/lab4-1170$ ./task5
Thread calculated sum of 1 to 100 = 5050
Main received result: 5050
maha@DESKTOP-PDJCJQV:~/1170-OS_LAB/lab4-1170$
```

The status bar at the bottom indicates the current file is 'main.c' and the cursor is at line 24, column 3. The system clock shows 2:50 AM on 10/10/2025.