National Textile University

Department of Computer Science

Subject: Operating System

Submitted to: Sir Nasir

Submitted by: Amina

Reg. number: 23-NTU-CS-1136

Lab no.: lab5

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.

```
#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;
}
```

```
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$ gcc Task1.c -o task1 -lpthread
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$ ./task1
Main thread starting...
Main Thread ID: 123646009440064
Hello from the new thread!
Thread ID: 123646005737152
Main thread exiting...
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$
```

Program 2: Passing Arguments to Threads

Objective: Pass data to a thread function.

```
#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;

}
```

```
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$ gcc Task2.c -o task2 -lpthread
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$ ./task2
Creating thread with argument: 42
Thread received number: 42
Square: 1764
Main thread done.
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$
```

Task2CGPA:

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

```
#include <pthread.h>

void* print_number(void* arg)
{
    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 cgpa = 3.67;

    printf("Creating thread with argument: %f\n", cgpa);

    // Pass address of 'cgpa' to thread
    pthread_create(&thread_id, NULL, print_number, &cgpa);
    pthread_join(thread_id, NULL);

    printf("Main thread done.\n");
    return 0;
}
```

```
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$ ./taskcgpa
Creating thread with argument: 3.670000
Thread received number: 3.670000
Square: 13.468901
Main thread done.
```

Program 3: Passing Multiple Data

```
//Multiple Threads (CGPA +NAME)
#include <stdio.h>
#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, "My name is Amina and cpga is 3.67"};
pthread_create(&t1, NULL, printData, &data1);
pthread_join(t1, NULL);
printf("All threads done.\n");
return 0;
}
```

```
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$ ./task3
Thread 1.000000 says: My name is Amina and cpga is 3.67
All threads done.
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$
```

Program 4: Thread Return Values

Objective: Get return values from threads.

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

```c
#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;
}
```

```
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$ gcc Task5.c -o task5 -lpthread
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$ ./task5
Thread 1: Starting task...
Thread 3: Starting task...
Thread 2: Starting task...
Thread 3: Task completed!
Thread 2: Task completed!
Thread 1: Task completed!
Main thread: All threads have finished.
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$
```

Program 2: Demonstrating a Race Condition

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

```c
#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;
}
```

```
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$ ./task6
Expected counter value: 200000
Actual counter value: 151887
amina@DESKTOP-SEP18NK:~/OSLabs/lab5$
```