RIPHAH INTERNATIONAL UNIVERSITY, ISLAMABAD

Lab # 14

Bachelors of Computer Science – 6th Semester

Subject: OS

Submitted to: Ms. Kausar

Submitted by: Ayesha Noor _ 41379

Date of Submission: 29- Nov -2024

Lab Tasks:

Question: 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>  //Header file for sleep(). man 3 sleep for details.
#include <pthread.h>

// A normal C function that is executed as a thread
// when its name is specified in pthread_create()
void *myThreadFun(void *vargp)
{
    sleep(1);
    printf("Printing GeeksQuiz from Thread \n");
    return NULL;
}

int main()
{
    pthread_t thread_id;
    printf("Before Thread\n");
    pthread_create(&thread_id, NULL, myThreadFun, NULL);
    pthread_join(thread_id, NULL);
    printf("After Thread\n");
    exit(0);
}
```

Solution:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> // Header file for sleep(). man 3 sleep for details.
#include <pthread.h>

// A normal C function that is executed as a thread
// when its name is specified in pthread_create()
void *myThreadFun(void *vargp)
{
    sleep(1);
    printf("Printing GeeksQuiz from Thread \n");
    return NULL;
}

int main()
{
    pthread_t thread_id;
    printf("Before Thread\n");
    pthread_create(&thread_id, NULL, myThreadFun, NULL);
    pthread_join(thread_id, NULL);
    printf("After Thread\n");
    exit(0);
}
```

Libraries

- **stdio.h**: For input and output (e.g., printf).
- **stdlib.h**: For utility functions like exit().
- **unistd.h**: For system calls like sleep().
- **pthread.h**: For thread creation and management (e.g., pthread_create, pthread_join).

myThreadFun Function

- This is the thread's execution function. It:
    - Pauses execution for 1 second (sleep(1)).
    - Prints "Printing GeeksQuiz from Thread".
    - Returns NULL (since the function has no return value).

main Function

- Creates a thread using pthread_create.
- Uses pthread_join to wait for the thread to complete.
- Prints messages before and after the thread is executed.

Output:

- Output will be:

```
Before Thread
Printing GeeksQuiz from Thread
After Thread
```

Question: 2

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

// Let us create a global variable to change it in threads
int g = 0;

// The function to be executed by all threads
void *myThreadFun(void *vargp)
{
    // Store the value argument passed to this thread
    int *myid = (int *)vargp;

    // Let us create a static variable to observe its changes
    static int s = 0;

    // Change static and global variables
    ++s; ++g;

    // Print the argument, static and global variables
    printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);
}

int main()
{
    int i;
    pthread_t tid;

    // Let us create three threads
    for (i = 0; i < 3; i++)
        pthread_create(&tid, NULL, myThreadFun, (void *)&tid);

    pthread_exit(NULL);
    return 0;
}
```

Solution:

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

// Let us create a global variable to change it in threads
int g = 0;

// The function to be executed by all threads
void *myThreadFun(void *vargp)
{
    // Store the value argument passed to this thread
    int *myid = (int *)vargp;

    // Let us create a static variable to observe its changes
    static int s = 0;

    // Change static and global variables
    ++s;
    ++g;

    // Print the argument, static and global variables
    printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, s, g);
    return NULL;
}
```

```
int main()
{
    int i;
    pthread_t tid;

    // Let us create three threads
    for (i = 0; i < 3; i++)
        pthread_create(&tid, NULL, myThreadFun, (void *)&tid);

    pthread_exit(NULL);
    return 0;
}
```

Global Variable

- g is a global variable accessible by all threads.

Static Variable in Thread Function

- s is static, so it retains its value across multiple calls.

myThreadFun Function

- Takes a thread ID as input (vargp).
- Increments the static (s) and global (g) variables.
- Prints the thread's ID along with the updated values of s and g.

main Function

- Creates three threads in a loop.
- Each thread calls myThreadFun with its ID.

Step-by-Step Explanation:

Thread Initialization

- Three threads are created using a loop.
- Each thread is passed its index (i) as an argument.

Shared Variable Behavior

- Static variable s keeps its value across threads because it's shared within the function's scope.
- Global variable g is shared across the entire program and increments with every thread.

Output:

```
Thread ID: 139679658698496, Static: 1, Global: 1
Thread ID: 139679658698496, Static: 2, Global: 2
Thread ID: 139679658698496, Static: 3, Global: 3
```

- Each thread prints its ID, the value of s, and the global g.
- Output order may vary due to thread scheduling.

Question: 3

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

void * workerThreadFunc(void * tid){
  long * myID = (long *) tid;
  printf("HELLO WORLD! THIS IS THREAD %ld\n",*myID);
}

int main(){

  pthread_t tid0;
  pthread_create(&tid0,NULL,workerThreadFunc,(void *)&tid0);

  pthread_exit(NULL);
  return 0;
}
```

Solution:

Creating a New Thread:

- We create a new thread to do work in parallel with the main program.
- The new thread runs a function (workerThreadFunc()) that prints a message with the thread's ID.

Key Code Parts

- pthread_create(&tid0, NULL, workerThreadFunc, (void*)&tid0);: This starts a new thread and passes the thread ID to it.
- pthread_exit(NULL);: Makes the main program wait for the new thread to finish before ending.

Worker Thread

The new thread prints "HELLO WORLD! THIS IS THREAD X", where X is the thread's unique ID.

Why pthread_exit()?

It ensures the main program waits for the thread to finish, preventing it from ending early.

Output

```
HELLO WORLD! THIS IS THREAD 139679658698496
```

Where <Thread_ID> is the unique identifier for each thread. Since you're creating only one thread in this example, it would look like:

```
HELLO WORLD! THIS IS THREAD 139679658698496
```

Question: 4

Define posix thread and its working in your own words?

Solution:

A **POSIX thread (pthread)** is a unit of execution within a program that follows the POSIX standard for multi-threading, allowing multiple tasks to run concurrently while sharing resources like memory.

Key Points:

- **Thread Creation**: pthread_create() creates a thread, specifying the thread's ID, function to execute, and any arguments.
- **Thread Execution**: Each thread executes the specified function, with its own unique ID, but shares memory with other threads.

- **Synchronization**: Threads use mechanisms like **mutexes** to avoid conflicts when accessing shared resources.
- **Thread Joining**: pthread_join() makes a thread wait for another thread to finish before continuing.
- **Thread Termination**: A thread can end by returning from its function or calling pthread_exit().