National University of Computer and Emerging Sciences

Lab Manual 10
(Operating Systems)

Department of Computer Science
FAST-NU, Lahore

Threads

OBJECTIVE: To understand and learn about threads and their implementation in programs

Pthreads:

pthread_create(pthread_t*,NULL,void*,void*)

- First Parameter is pointer of threaded it should be different for all threads.
- Second Parameter is used to change stack size of thread. Null means use default size.
- Third parameter is address of function which we are going to use as thread.
- Forth parameter is argument to function.

pthread_join(ipthread_t,void**)

- Pthread join is used in main program to wait for the end of a particular thread.
- First parameter is ThreadID of particular thread.
- Second Parameter is used to catch return value from thread.

Thread Library:

- POSIX Pthreads
- Two general stratergies for creating multiple threads.
  a) Asynchronous threading:
    - Parent and child threads run independently of each other
    - Typically little data sharing between threads
  b) Synchronous threading:
    - Parent thread waits for all of its children to terminate
    - Children threads run concurrently
    - Significant data sharing

Getting Thread's ID in itself:

pthread_self()

Pthreads Header File:

- **pthread.h**
- Each thread has a set of attributes, including stack size and scheduling information ·
- In a Pthreads program, separate threads begin execution in a specified function //runner()
- When a program begins
  - A single thread of control begins in main()

➢ main() creates a second thread that begins control in the runner() function

➢ Both threads share the global data

Compiling multithreads C program:
Gcc –lpthread main.c –o main

Example:

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

int var = 0;

void *worker(void *param)
{
    var = param;
    pthread_exit(0);
}

int main()
{
    printf("[Before Thread Execution] Var = %d\n", var);

    int value = 10;
    pthread_t threadID;
    pthread_create(&threadID, NULL, worker, (void *)value);

    pthread_join(threadID, NULL);
    printf("[After Thread Execution] Var = %d\n", var);
    return 0;
}
```

```
irtiza@Irtiza:/mnt/c/Users/m7irt/OneDrive/Desktop$ ./a.out
[Before Thread Execution] Var = 0
[After Thread Execution] Var = 10
irtiza@Irtiza:/mnt/c/Users/m7irt/OneDrive/Desktop$
```

Example: Sending and Receiving an array

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

void *worker(void *params)
{
    int *numbers = (int *)params;
    numbers[0] += 10;
    numbers[1] += 10;
    printf("New Thread -\t Value[0] - %d Value[1] = %d\n", numbers[0], numbers[1]);
    pthread_exit(numbers);
}

int main()
{
    int values[] = {10, 20};
    pthread_t threadID;

    pthread_create(&threadID, NULL, worker, (void *)values);

    int *updated_values;
    pthread_join(threadID, (void **)&updated_values);

    printf("Main Thread -\t Value[0] = %d Value[1] = %d\n", updated_values[0], updated_values[1]);
    return 0;
}
```

```
irtiza@Irtiza:/mnt/c/Users/m7irt/OneDrive/Desktop$ ./a.out
New Thread -     Value[0] - 20 Value[1] = 30
Main Thread -    Value[0] = 20 Value[1] = 30
irtiza@Irtiza:/mnt/c/Users/m7irt/OneDrive/Desktop$
```

Task1:

Write a program that creates two synchronous threads and pass the file names f1.txt and f2.txt as a parameter to each thread function and it will remove all the duplicate and negative values from the file and then return the unique and non-negative values it to the main thread and will not display it. The mainthreadwillthencomputetheaverageoftheuniqueintegersandwillwritetheoutputtoaseparate file output.txt.

Task2:

Write a program twoMany.c that will create a number N of threads specified in the command line, each of which prints out a hello message and its own thread ID. To see how the execution of the threads interleaves, make the main thread sleep for 1 second for every 4 or 5threadsit creates. The output of your code should be similar to:(hint: pthread_self() function to get thread id and pthread_exit() to exit the thread execution)