Name: K Srikanth Roll Number: 17ETCS002124

Laboratory 3

Title of the Laboratory Exercise: Programs based on multithreaded programming

1. Introduction and Purpose of Experiment

   Multithreading is the ability of a processor or a single core in a multi-core processor to execute multiple threads concurrently, supported by the operating system. By solving students will be able to manipulate multiple threads in a program.

2. Aim and Objectives

   Aim

   - To develop programs using multiple threads.

   Objectives

   At the end of this lab, the student will be able to

   - Identify multiple tasks
   - Use threads constructs for creating threads
   - Apply threads for different/multiple tasks

3. Experimental Procedure

   i. Analyse the problem statement

   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code

   iii. Implement the algorithm in C language

   iv. Compile the C program

   v. Test the implemented program

   vi. Document the Results

   vii. Analyse and discuss the outcomes of your experiment

4. Questions

   Create multithreaded programs to implement the following

   a) Display “Hello World” message by 3 different threads
   b) Create two threads;

- Thread1 adds marks (out of 10) of student1 from subject1 to subject5, Thread2 adds marks of student2 from subject1 to subject 5. Main process takes the sum returned from the Thread1 and Thread2, decides who scored more marks and displays student with its highest score.

5. Calculations/Computations/Algorithms

Question 1 .

1. Start

2. Declare a call-back function, printstatement

a. Print Hello world

3. Declare a main Function

i. Declare p_thread array thread[3]
ii. For i:=0 to 2 do
1. Rc:=create a p_thread pointing to printstatement function
2. If Rc>0, then output “Error occurred”

4. Stop

Question 2

1. Start

2. Declare a call back function, marks

a. returns the total marks of 5 subjects

3**. Declare a main Function**

a. Declare two threads , thread 1 and thread 2

b. Declare pointer variables to store data of students.

c. Input Student 1 and Student 2 Marks

d. Create thread process for student 1 with thread 1 and marks function

e. Join thread 1 with pointer s1

f. Create thread process for student 2 with thread 2 and marks function

g. Join thread 2 with pointer s2

h. Compare the sum of s1 and s2

i. Display the highest marks among s1 and s2

4. Stop

Name: K Srikanth Roll Number: 17ETCS002124

Question 1

Code

```c
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
# define Threads_Num 3

// K Srikanth 17ETCS002124

void * printstatement(void * tid)
{
    int id;
    id = (int)tid;

   printf("Hello World !! \nIf you are wondering i was created by thread %d.\n",id);
   pthread_exit(NULL);
}

int main()
{
    int error;
   pthread_t threads[Threads_Num];
    for (int i = 0; i < Threads_Num; i++)
    {
        printf("Creating a thread in main %d \n",i);
        error = pthread_create(&threads[i],NULL,printstatement,(void*) i);
        if(error){
            printf("Opps ... !! There was a error while creating thread CODE : %d \n",error);
            exit(-1);
        }
    }

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

Figure 1 C Program to print hello world with three different threads

Question 2

Code

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

// K Srikanth 17ETCS002124

void *marks(int *arg)
{
    int sum = 0;
    for (int i = 0; i < 5; i++)
    {
        sum = sum + arg[i];
    }
    pthread_exit(sum);
}

int main()
{
    pthread_t thread1, thread2;
    void *s1, *s2;
    int i, marks1[5], marks2[5];

    printf("Enter the Marks of Studnet 1 : \n");
    printf("**********************************\n");
    for (i = 0; i < 5; i++)
    {
        printf("Enter the marks of Subject %d \n", (i + 1));
        scanf("%d", &marks1[i]);
    }
    printf("\n");
    printf("Enter the Marks of Studnet 2 : \n");
    printf("**********************************\n");
```

Figure 2 C Program to find the highest score between student 1 and student 2 using threads

Continued Code….Q2

```c
    for (i = 0; i < 5; i++)
    {
        printf("Enter the marks of Subject %d \n", (i + 1));
        scanf("%d", &marks1[i]);
    }
    printf("\n");
    printf("Enter the Marks of Studnet 2 : \n");
    printf("**********************************\n");
    for (i = 0; i < 5; i++)
    {
        printf("Enter the marks of Subject %d \n", (i + 1));
        scanf("%d", &marks2[i]);
    }

    pthread_create(&thread1, NULL, marks, (void *)marks1);
    pthread_join(thread1, &s1);
    pthread_create(&thread2, NULL, marks, (void *)marks2);
    pthread_join(thread1, &s2);
    printf("\n");

    printf("Total Marks of Student 1 is %d \n", s1);
    printf("Total Marks of Student 2 is %d \n", s2);
    printf("\n");
    if (s1 > s2)
    {
        printf("Student 1 has the Highest Marks. \n");
    }
    else
    {
        printf("Student 2 has the Highest Marks. \n");
    }

    return 0;
}
```

Figure 3 Continued.. C Program to find the highest score between student 1 and student 2 using threads

6. Presentation of Results

Question 1

```
Creating a thread in main 0
Creating a thread in main 1
Creating a thread in main 2
Hello World !!
If you are wondering i was created by thread 0.
Hello World !!
If you are wondering i was created by thread 1.
Hello World !!
If you are wondering i was created by thread 2.
Program ended with exit code: 0
```

Figure 4 C Program output for question 1

Question 2

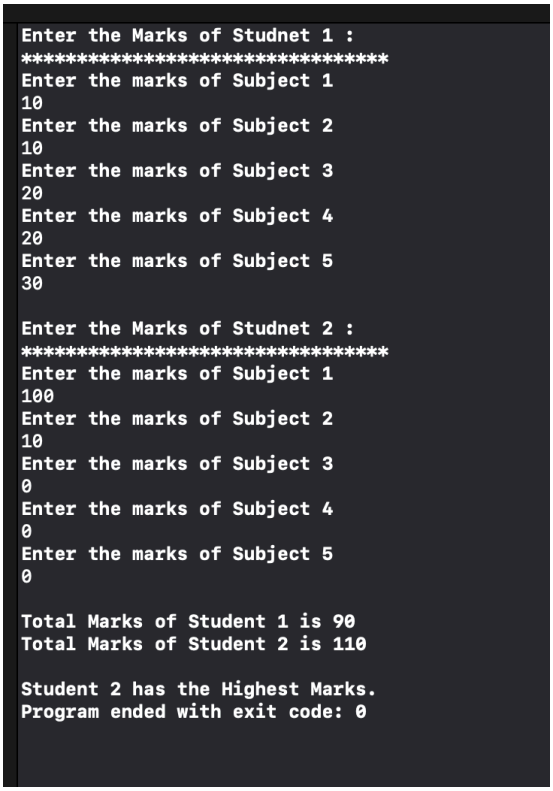

Figure 5 C Program output for question 2

7. Analysis and Discussions

A thread is a semi-process that has its own stack, and executes a given piece of code. Unlike a real process, the thread normally shares its memory with other threads. pthread is short for POSIX thread.

- In main() we declare a variable called thread_id, which is of type pthread_t. This is basically an integer used to identify the thread in the system. After declaring thread_id, we call the pthread_create function to create a real, living thread.

- pthread_create() gets 4 arguments The first argument is a pointer to thread_id, used by pthread_create() to supply the program with the thread's identifier. The second argument is used to set some attributes for the new thread. In our case we supplied a NULL pointer to tell pthread_create() to use the default values.

- Notice that printstatement() accepts a void * as an argument and also returns a void * as a return value.

- To pass our thread an arbitrary argument, We use the fourth argument to the pthread_create() call. If we do not want to pass any data to the new thread, we set the fourth argument to NULL.

- pthread_create() returns zero on success and a non-zero value on failure.

- After pthread_create() successfully returns, the program will consist of two threads. This is because the main program is also a thread and it executes the code in the main() function in parallel to the thread it creates.

- The call to pthread_exit causes the current thread to exit and free any thread-specific resources it is taking.

8. Conclusions

Threads can be implemented in four different stages, Define thread reference variables, create an entry point of thread, run the thread and join everything back up. Threads run as the groups sharing OS resources, data section, code section, files etc. even though they have separate program counter.

9. Comments

1. Limitations of Experiments

Threads cannot run independently as the group of threads share OS resources, data section, code section, files etc even though they have separate program counter.

2. Limitations of Results

Since program 3a hasn't implemented pthread_join function, the order of output is not maintained.

3. Learning happened

Threads, Multithread processing, differences and similarities between threads and processes

4. Recommendations

If implemented correctly, threads have some advantages over processes Compared to the standard fork (), threads carry a lot less overhead