Name : Kaushal vashisth Roll no: 18ETCS002147

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) Display thread IDs by each thread

c) Create three threads;

- Thread1 add marks out of 10 from subject1 to subject5 of student 1, Thread2 adds from Subject1 to subject 5 of student 2 and Thread3 takes the sum from the Thread1 and Thread2 and decides who scored more marks. Display the total marks of the student (highest score) in parent process.

  Instructions: Use a global variable to keep track of the sum from Thread1 and Thread2 and update it. Thread3 needs to wait until both Thread1 and Thread2 are done. Finally, parent process needs to wait until Thread3 is done. Get the return value from Thread3, and print the return value.

5. Calculations/Computations/Algorithms

Algo:-

STEP 1: Start
STEP 2: define NUM_THREADS ← 3
STEP 3: ret_code ← 0, malloc threads array
STEP 4: for i=0 to NUM_THREADS, do
    4.1: ret_code ← create thread that executes callback function (pthread_create)
    4.2: if ret_code is positive, then display error message and exit.
STEP 5: Stop

void* callback(void* args);
STEP 1: Start
STEP 2: display “hello world” and id
STEP 3: Stop

6. Presentation of Results:-

Program for part a and b:-

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

#define NUM_THREADS 3

void *callbackFunc(void *);

int main()
{
```

```
    int ret_code;
    pthread_t threads[NUM_THREADS];

    for (int i = 0; i < NUM_THREADS; i++)
    {
        ret_code = pthread_create(&threads[i], NULL, callbackFunc, NULL);
        if (ret_code)
        {
            printf("Error creating thread. error code: %d", ret_code);
            exit(-1);
        }
    }

    for (int i = 0; i < NUM_THREADS; i++)
        pthread_join(threads[i], NULL);
}

void *callbackFunc(void *args)
{
    printf("Hello World! id: %ld\n", pthread_self());
}
```

Output:-

```
Hello World! id: 25770041376
Hello World! id: 25770369520
Hello World! id: 25770140064
PS F:\RUAS\5 sem\Operating system\Os lab\lab4\programs>
```

Program for part c:-

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

int total1 = 0, total2 = 0, temp;
int m1[5], m2[5];

void* function1(void* tid) {
  printf("Thread: %d Calculates Marks Of Student One\n", tid+1);
  for (size_t i = 0; i < 5; i++) {
    total1 += m1[i];
  }
  pthread_exit(NULL);
```

```c
}

void* function2(void* tid) {
  printf("Thread: %d Calculates Marks Of Student Two\n", tid+1);
  for (size_t i = 0; i < 5; i++) {
    total2 += m2[i];
  }
  pthread_exit(NULL);
}

int main(int argc, char const *argv[]) {
  int status;
  void* exitstatus;
  pthread_t threads[3];
  printf("\n Student 1:-" );
  printf("\n\t Enter the marks: \n");
  for (size_t i = 0; i < 5; i++) {
    scanf("\t%d", &m1[i] );
  }
  printf("\nStudent 2:-" );
  printf("\n\t Enter the marks: \n");
  for (size_t i = 0; i < 5; i++) {
    scanf("\t%d", &m2[i] );
  }

  printf("\nSucessfully created Thread 1\n");
  status = pthread_create(&threads[0], NULL, function1, (void*)0);
  pthread_join(threads[0], &exitstatus);
  printf("Total Marks of student 1: %d\n",total1);

  printf("\n---Sucessfully created Thread 2---\n");
  status = pthread_create(&threads[1], NULL, function2, (void*)1);
  pthread_join(threads[1], &exitstatus);
  printf("Total Marks of student 2: %d\n\n",total2);

  printf("Main thread calcualatess highest marks\n" );
  if (total1==total2) {
    printf("Student 1 and Student 2 scored equal marks\n" );
  }
  else   if (total1 > total2) {
      printf("Student 1 Scored Higher marks than Student 2\n" );
      temp = total1;
      return temp;
  }
  else   if (total1 < total2) {
```

```
        printf("Student 2 Scored Higher marks than Student 1\n" );
        temp = total2;
        return temp;
    }

    return 0;
}
```

Output:-

```
 Student 1:-
         Enter the marks:
80 90 99 78 100

Student 2:-
         Enter the marks:
100 100 56 89 99

Sucessfully created Thread 1
Thread: 1 Calculates Marks Of Student One
Total Marks of student 1: 447

---Sucessfully created Thread 2---
Thread: 2 Calculates Marks Of Student Two
Total Marks of student 2: 444

Main thread calcualatess highest marks
Student 1 Scored Higher marks than Student 2
PS F:\RUAS\5 sem\Operating system\Os lab\lab4\programs>
```

7. Analysis and Discussions:-
   A thread is a path of execution within a process. A process can contain multiple threads.

   Advantages of Thread over Process:-
   1. Responsiveness: If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.
   2. Faster context switch: Context switch time between threads is lower compared to process context switch. Process context switching requires more overhead from the CPU.
   3. Effective utilization of multiprocessor system: If we have multiple threads in a single process, then we can schedule multiple threads on multiple processor. This will make process execution faster.
   4. Resource sharing: Resources like code, data, and files can be shared among all threads within a process.

Note: stack and registers can't be shared among the threads. Each thread has its own stack and registers.

8. Conclusions:-
   A thread is also known as lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. More advantages of multithreading are discussed below

9. Comments

   Limitations of Experiments:-
   Blocking: The major disadvantage if that if the kernel is single threaded, a system call of one thread will block the whole process and CPU may be idle during the blocking period.
   Security: Since there is, an extensive sharing among threads there is a potential problem of security.