**Bahria University, Lahore Campus**

Department of Computer Sciences

Lab Journal 11

**(spring 2024)**

|  |  |  |
| --- | --- | --- |
| Course: | **Operating System Lab** | Date: 05/30/24 |
| Course Code: | CSL - 320 | Max Marks: 20 |
| Faculty’s Name: | Abdullah |  |

Name:\_AFFAN AHMAD\_\_\_\_ Enroll No: \_\_03-134221-003\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Objective(s) :

Understanding of threads, creation of threads, passing arguments to threads and to joining threads.

## Lab Tasks :

**Task 1:** Create a simple program for thread creation and termination. The created thread should display “I am child thread”.

### Task 2: Write a program that creates a number of threads and each thread should print “Hello World!” along with its number passed as argument in a thread.

**Task 3:** Write the output of the program for the difference between processes and threads. Check the values of Global and Local variables in threads and processes.

**Task 4.1:** Write a program to create two threads. One should take input from user and stores thefactorial of that input. Other should take two variable base and power, calculate power.

**Task 4.2:** Create ten threads. A global variable is declared sum. As each thread is created a function is called to calculate the sum of number from 1-10.

**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 5 |  |  |
| 2. | 5 |  |  |
| 3. | 5 |  |  |
| 4. | 5 |  |  |
| **Total** | **20** |  | **Signature** |

**Note : Attempt all tasks and get them checked by your Lab. Instructor**.

# Lab 11: Threads

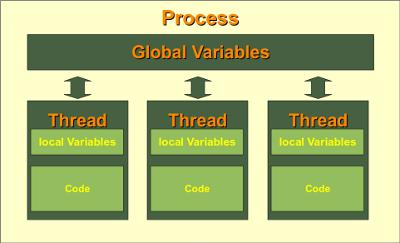
**Objective(s):**

* + To understand threads.
  + To create threads.
  + To pass arguments to threads.
  + To join threads.

**Tool(s) used:**

Ubuntu, VIM Editor

**What are Threads?** Threads are lightweight processes which execute in the address spaceof the process.



All threads can access the global variables but the access to the local variables of each thread is limited to its own thread.

**Library**: We will be using **pthread.h** in this lab.

**Man Pages of linux:** To read the complete details of any function of this library you can seethe man pages of linux. Just type **man** and the name of that function in the shell. e.g. man pthread\_create(). You will see the prototype of the function pthread\_create() as below

## 

**Creating a thread:**

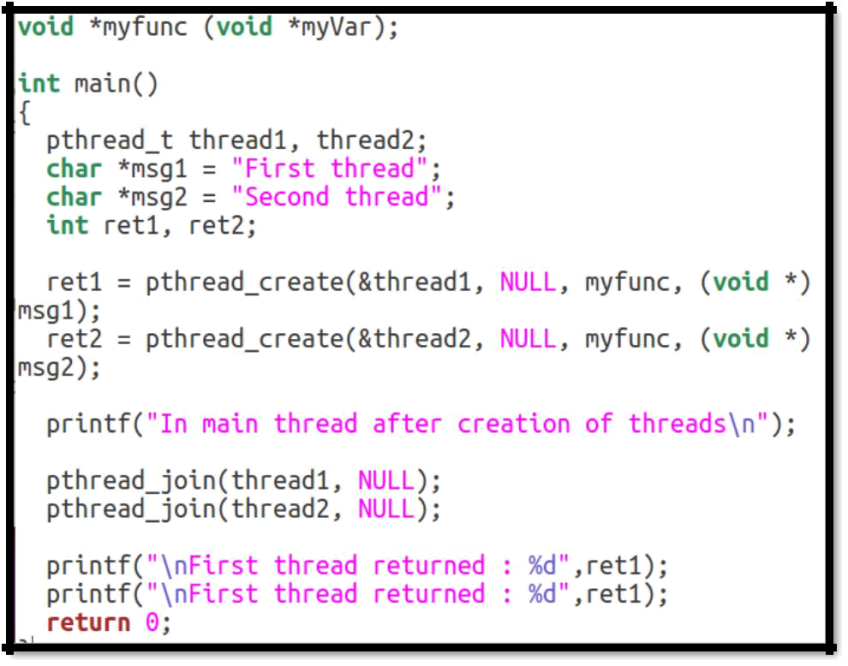
As you can see in the prototype above that the create function takes 4 arguments.

1. Id of the thread which you will declare before calling this function.
2. Attribute of the thread. This is the behavior of the thread which you can set or keep it null for default behavior.
3. The function/routine which will invoke when this thread will be created.
4. Arguments which you want to pass to that function which will be invoked upon creation of thread. Make sure to type cast the argument as void pointer before sending it, because you can only send the pointer of type void here. Later on in the function you can type cast back to your required type.

By default, a joinable thread is created. You must join it later with the main thread. The function used to join the thread is pthread\_join(). Main thread waits at this function call, stops its execution and resumes the execution until the thread is joined.

**Return Value:**

It returns 0 on success and error number on error.

**Sample Program:**

### Task 1 Create a simple program for thread creation and termination. The created thread should display “I am child thread”.

**Program**

**#include <iostream>**

**using namespace std;**

**#include <thread>**

**void childThreadFunction() {**

**cout << "I am child thread" << endl;**

**}**

**int main() {**

**thread childThread(childThreadFunction);**

**childThread.join();**

**cout << "Child thread has finished execution." << endl;**

**return 0;**

**}**

**OUTPUT**

## 

### Task 2 Write a program that creates a number of threads, and each thread should print “Hello World!” along with its number passed as argument in a thread.

**Program**

**#include <iostream>**

**using namespace std;**

**#include <thread>**

**#include <vector>**

**void printHelloWorld(int threadNumber) {**

**cout << "Hello World! from thread " << threadNumber << endl;**

**}**

**int main() {**

**const int numThreads = 10;**

**vector<thread> threads;**

**for (int i = 0; i < numThreads; ++i) {**

**threads.push\_back(thread(printHelloWorld, i + 1));**

**}**

**for (auto& thread : threads) {**

**thread.join();**

**}**

**cout << "All threads have finished execution." << endl;**

**return 0;**

**}**

**OUTPUT**



### Task 3 Write the output of the program for the difference between processes and threads. Check the values of Global and Local variables in threads and processes.

**Program**

#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/types.h>

#include <wait.h>

int global\_var=10;

void \*thread\_func(void \*var);

int main(){

int local\_var=20;

pid\_t id;

pthread\_t thread1,thread2;

printf("Create two threads to see what content they share\n");

pthread\_create(&thread1,NULL,thread\_func,NULL);

pthread\_create(&thread2,NULL,thread\_func,NULL);

pthread\_join(thread1,NULL);

pthread\_join(thread2,NULL);

printf("\nAfter thread global var = %d",global\_var);

printf("\n");

printf("\nBefore Fork :\nGlobal Variable:%d \nLocal Variable:%d",global\_var,local\_var);

id=fork();

if(id==0){

printf("\nIn child: \nGlobal Variable=%d \nLocal Variable=%d",global\_var,local\_var);

local\_var=133; global\_var=100;

printf("\nChild set: Global variable=%d \nLocal Variable=%d",global\_var,local\_var);

}

else{

wait(NULL);

printf("\nIn parent: \nGlobal Variable:%d \nLocal Variable=%d",global\_var,local\_var);

}

return 0; }

void \*thread\_func(void \*var){

int local\_var=200;

printf("\nThread :\n Global Variable :%d\n Local Variable:%d ",global\_var,local\_var);

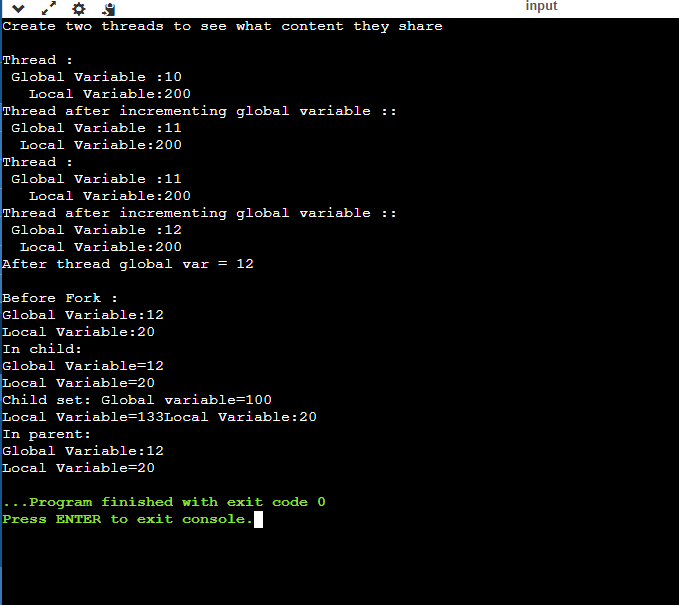
global\_var++;

printf("\nThread after incrementing global variable ::\n Global Variable :%d \n Local Variable:%d ",global\_var,local\_var);

pthread\_exit(0);

}

**OUTPUT**



**Task 4:** Write a program to create two threads. One should take input from user and stores thefactorial of that input. Other should take two variable base andpower, calculate power.

**#include <iostream>**

**using namespace std;**

**#include <thread>**

**#include <cmath>**

**void calculateFactorial() {**

**int number;**

**cout << "Enter a number to calculate its factorial: ";**

**cin >> number;**

**long long factorial = 1;**

**for (int i = 1; i <= number; ++i) {**

**factorial \*= i;**

**}**

**cout << "Factorial of " << number << " is " << factorial << endl;**

**}**

**void calculatePower() {**

**double base;**

**int exponent;**

**cout << "Enter base: ";**

**cin >> base;**

**cout << "Enter exponent: ";**

**cin >> exponent;**

**double result = pow(base, exponent);**

**cout << base << " raised to the power of " << exponent << " is " << result << endl;**

**}**

**int main() {**

**thread factorialThread(calculateFactorial);**

**thread powerThread(calculatePower);**

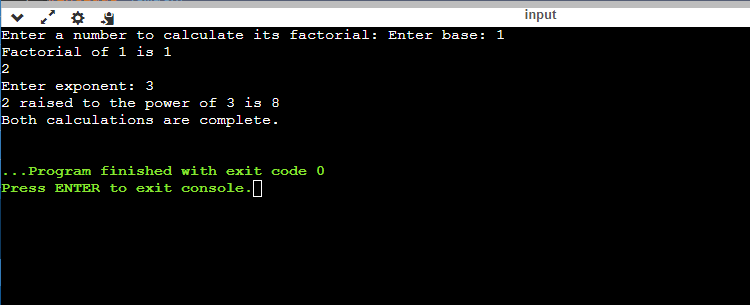
**factorialThread.join();**

**powerThread.join();**

**cout << "Both calculations are complete." << endl;**

**return 0;**

**}**



**Task 4.1** Create ten threads. A global variable is declared sum. As each thread is created a function is called to calculate the sum of number from 1-10.

#include <iostream>

using namespace std;

#include <thread>

#include <vector>

#include <mutex>

int sum = 0;

mutex sumMutex;

void calculateAndAddSum() {

int localSum = 0;

for (int i = 1; i <= 10; ++i) {

localSum += i;

}

lock\_guard<mutex> guard(sumMutex);

sum += localSum;

}

int main() {

const int numThreads = 10;

vector<thread> threads;

for (int i = 0; i < numThreads; ++i) {

threads.push\_back(thread(calculateAndAddSum));

}

for (auto& thread : threads) {

thread.join();

}

cout << "The final sum is: " << sum << endl;

return 0;

}

