**Threads Practice**

**1.Simple program:**

#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 \*j)

{

sleep(1);

printf("Printing string from Thread \n");

return NULL;

}

int main()

{

pthread\_t thread\_id;

printf("Before Thread\n");

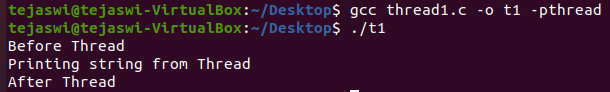
pthread\_create(&thread\_id, NULL, myThreadFun, NULL); //Create a thread

pthread\_join(thread\_id, NULL);

printf("After Thread\n");

exit(0);}

}

****

In the above program we simply create thread using a pthread\_create() and pass myThreadFun function into it which prints a statement and returns the control back to the main after execution thereby printing “After Thread” statement.

**2.With global and static variables:**

#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;

// 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++)

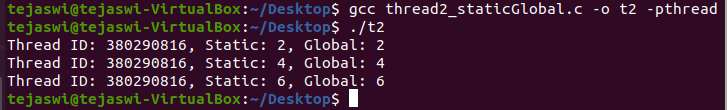
pthread\_create(&tid, NULL, myThreadFun, (void \*)&tid);

pthread\_join(tid,NULL);

pthread\_exit(NULL);

return 0;

}

In the above program, we test the scope of static and global variables in threads.We create a static variable and global variable and pre increment the variables in myThreadFun(). We create 3 threads and pass them to the function and finally join them using pthread\_join() and print the corresponding values of the variables.

**3.Threads:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

void \*SampleThread1(void \*vargp)

{ int i = 0;

printf("SampleThread(1) is running ... \n");

for(i = 0; i < 3; i++) {

sleep(1);

printf("timer running inside SampleThread(1) = %d\n", i);

}

printf("SampleThread(1) is exiting ... \n");

return NULL;

};

void \*SampleThread2(void \*vargp)

{

int i = 0;

printf("SampleThread(2) is running ... \n");

for(i = 0; i < 5; i++) {

sleep(1);

printf("timer running inside SampleThread(2) = %d\n", i);

}

printf("SampleThread(2) is exiting ... \n");

return NULL;

};

int main()

{int i = 0;

pthread\_t tid1, tid2;

pthread\_create(&tid1, NULL, SampleThread1, NULL);

pthread\_create(&tid2, NULL, SampleThread2, NULL);

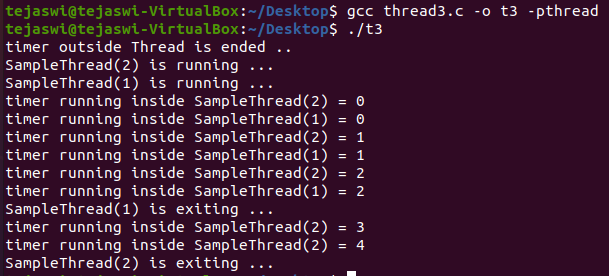
printf("timer outside Thread is ended ..\n");

pthread\_join(tid1, NULL);

pthread\_join(tid2, NULL);

exit(0);

}



SampleThread1(void \*vargp) is the function to be run as the first thread.It runs for about 3 seconds. SampleThread2(void \*vargp) is the function used to run as the second thread. It runs for about 5 seconds.The main runs the threads parallely.

**4. Passing 1 argument:**

#include <stdio.h>

#include <pthread.h>

void \* hello(void \*input) {

printf("%s\n", (char \*)input);

pthread\_exit(NULL);

}

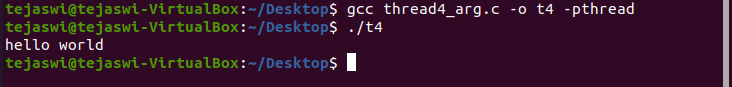
int main(void) {

pthread\_t tid;

pthread\_create(&tid, NULL, hello, "hello world");

pthread\_join(tid, NULL);

return 0;

****}

In the above program, we pass an argument from main, to a function and print it in the function.

**5. Structure:**

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

struct args {

char\* name;

int age;

};

void \*hello(void \*input) {

printf("name: %s\n", ((struct args\*)input)->name);

printf("age: %d\n", ((struct args\*)input)->age);

}

int main() {

struct args \*Allen = (struct args \*)malloc(sizeof(struct args));

char allen[] = "Allen";

Allen->name = allen;

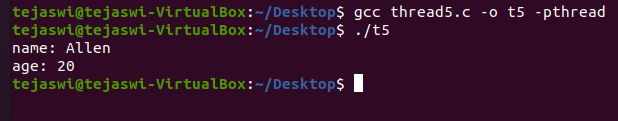
Allen->age = 20;

pthread\_t tid;

pthread\_create(&tid, NULL, hello, (void \*)Allen);

pthread\_join(tid, NULL);

return 0;

}

In the above program we pass more than 1 argument. A structure is used to pass multiple arguments. The values are printed.

**6. Arrays**

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#define NUM\_THREADS 8

char \*messages[NUM\_THREADS];

void \*PrintHello(void \*threadid)

{

int \*id\_ptr, taskid

sleep(1);

id\_ptr = (int \*) threadid;

taskid = \*id\_ptr;

printf("Thread %d: %s\n", taskid, messages[taskid]);

pthread\_exit(NULL);

}

int main(int argc, char \*argv[])

{

pthread\_t threads[NUM\_THREADS];

int \*taskids[NUM\_THREADS];

int rc, t;

messages[0] = "English: Hello World!";

messages[1] = "French: Bonjour, le monde!";

messages[2] = "Spanish: Hola al mundo";

messages[3] = "Klingon: Nuq neH!";

messages[4] = "German: Guten Tag, Welt!";

messages[5] = "Russian: Zdravstvytye, mir!";

messages[6] = "Japan: Sekai e konnichiwa!";

messages[7] = "Latin: Orbis, te saluto!";

for(t=0;t<NUM\_THREADS;t++)

{ taskids[t] = (int \*) malloc(sizeof(int));

\*taskids[t] = t;

printf("Creating thread %d\n", t);

rc = pthread\_create(&threads[t], NULL, PrintHello, (void \*) taskids[t] );

if (rc) {

printf("ERROR; return code from pthread\_create() is %d\n", rc);

exit(-1);

}

}

pthread\_exit(NULL);

}

A array of messages are printed using threads and pthread\_create is used to create threads.

**Questions:**

1. Write a C program to calculate the factorial value of a number by creating new thread. ( number should be passed through command line argument)

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

void\* fact(void\* input){

int n=input;

int f=1;

int i;

for(i=2;i<=n;i++){

f=f\*i;

}

printf("\nFactorial is %d\n",f);

pthread\_exit(NULL);

}

int main(){

pthread\_t tid;

int inp;

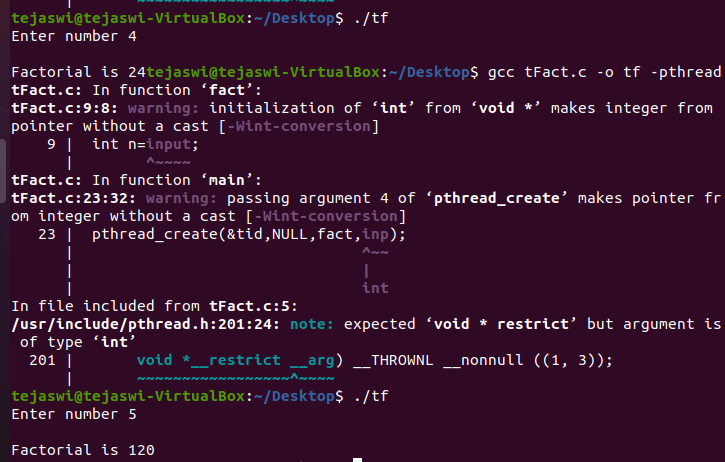
printf("Enter number ");

scanf("%d",&inp);

pthread\_create(&tid,NULL,fact,inp);

pthread\_join(tid,NULL);

return 0;

 }

2. Implement the C program in which main program accepts list of integers. Main program prints the sum of odd numbers from the list of integers. A thread is created and it has to calculate the sum of even numbers from the same list and should be printed by the main thread.

#include <stdio.h>

#include <pthread.h>

int size;

void \*factorial(void \*input){

int \*x = (int \*) input;

int sum = 0;

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

if (x[i] % 2 ==0){

sum = sum + x[i];

}

}

printf("\nEven sum : %d",sum);

pthread\_exit(NULL);

}

int main(void){

pthread\_t tid;

printf("\nEnter the size of the array : ");

scanf("%d",&size);

int inp[size];

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

scanf("%d",&inp[i]);

}

pthread\_create(&tid,NULL,factorial,(void\*)inp);

pthread\_join(tid,NULL);

int sum = 0;

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

if (inp[i] % 2 != 0){

sum = sum + inp[i];

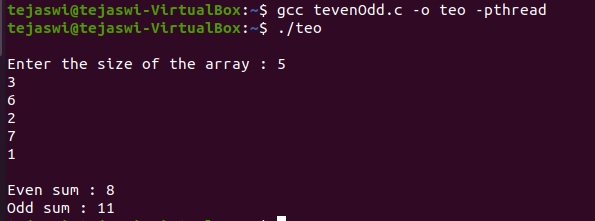
}

}

printf("\nOdd sum : %d \n",sum);

return 0;

}



3. Write a C program to display the student grade sheet. Thread1 should get the details of a student; Thread 2 should get the details of 5 subject marks. Main thread should calculate the grade point and prepare a grade sheet.

#include <stdio.h>

#include <pthread.h>

struct thread\_data {

char name[500];

char rollno[500];

} thread;

int marks[5];

void \*myThread(void)

{

printf("Enter the name : ");

scanf("%s",&thread.name);

printf("Enter your roll no: ");

scanf("%s",&thread.rollno);

pthread\_exit(NULL);

}

void \*myThread2(void){

printf("Enter the marks in 5 subjects \n");

for(int i = 0;i < 5;i++){

scanf("%d",&marks[i]);

}

pthread\_exit(NULL);

}

int main(){

pthread\_t tid1;

pthread\_create(&tid1, NULL, myThread,NULL);

pthread\_join(tid1, NULL);

pthread\_t tid2;

pthread\_create(&tid2, NULL, myThread2,NULL);

pthread\_join(tid2, NULL);

printf("The name of the student %s \n",thread\_data.name);

printf("The rollno of the student %s \n",thread\_data.rollno);

int sum,points;

sum = 0;

for (int i = 0;i < 5;i++){

points = 0;

if(marks[i] > 90){

sum = sum + 10;

points = points + 10;

}

else if(marks[i] > 80){

sum = sum + 9;

points = points + 9;

}

else if(marks[i] > 70){

sum = sum + 8;

points = points + 8;

}

else if(marks[i] > 60){

sum = sum + 7;

points = points + 7;

}

else if(marks[i] > 50){

sum = sum + 6;

points = points + 6;

}

else if(marks[i] > 40){

sum = sum + 5;

points = points + 5;

}

else if(marks[i] > 30){

sum = sum + 4;

points = points + 4;

}

else{

sum = sum + 3;

points = points + 3;

}

printf("subject %d : %d points \n ",i+1,points);

}

float point = sum/5;

printf("The grade points : %f \n" ,point);

return 0;

}

