**Assignment 1**

1. Write a program to create five threads in C.
2. Program to print “Hello World” using thread in C
3. Program that computes the square roots of the integers from 0 to 99 in a separate thread and returns an array of doubles containing the \* results. In the meantime the main thread should display a short message to the user and then display the results of the computation
4. Implement a linked list as a Parallel program (based on Pthreads) with Mutex locks for the entire linked list Implementation should support Search( ), Insert( ), and Delete( ) functions.
5. Program to computes the total of the values of the matrix using thread and mutex lock on global variable total

Ans 1:

// WAP to create 5 threads in C

#include<stdio.h>

#include<pthread.h>

void \*t1()

{

printf("1\n");

}

void \*t2()

{

printf("2\n");

}

void \*t3()

{

printf("3\n");

}

void \*t4()

{

printf("4\n");

}

void \*t5()

{

printf("5\n");

}

int main()

{

pthread\_t tid1,tid2,tid3,tid4,tid5;

pthread\_create(&tid1,NULL,t1,NULL);

pthread\_create(&tid2,NULL,t2,NULL);

pthread\_create(&tid3,NULL,t3,NULL);

pthread\_create(&tid4,NULL,t4,NULL);

pthread\_create(&tid5,NULL,t5,NULL);

// printf("main\n");

pthread\_join(tid1,NULL);

pthread\_join(tid2,NULL);

pthread\_join(tid3,NULL);

pthread\_join(tid4,NULL);

pthread\_join(tid5,NULL);

return 0;

}

Ans 2:

// WAP to print "Hello World" using threads

#include<stdio.h>

#include<pthread.h>

void \*f1()

{

printf("Hello World\n");

}

int main()

{

pthread\_t p1;

pthread\_create(&p1,NULL,f1,NULL);

pthread\_join(p1,NULL);

return 0;

}

Ans 3:

/\*

WAP that computes the square roots of the integers from 0 to 99 in a separate thread

and returns an array of doubles containing the \* results.

In the meantime the main thread should display a short message to the user

and then display the results of the computation

\*/

#include<stdio.h>

#include<stdlib.h>

#include<pthread.h>

#include<math.h>

#define N 100

double\* arr;

void \*root()

{

arr = (double\*)malloc(sizeof(double)\*N);

int i;

for(i=0;i<N;i++)

{

arr[i]=(double)sqrt(i);

}

return (void\*)arr;

}

int main()

{

pthread\_t calc;

double \*roots;

void \*rootX;

pthread\_create(&calc,NULL,root,NULL);

printf("Computing Square roots of numbers from 0 to %d...\n",N);

pthread\_join(calc,&rootX);

roots=(double\*)rootX;

int i;

for(i=0;i<N;i++)

{

printf("Square root of %d is %lf\n",i,(double)roots[i]);

}

return 0;

}

Ans 4:

/\*

Implement a linked list as a Parallel program (based on Pthreads)

with Mutex locks for the entire linked list Implementation should

support Search( ), Insert( ), and Delete( ) functions.

\*/

#include<stdio.h>

#include<pthread.h>

#include<stdlib.h>

#define MAXVAL 10

#define CALLS 20

pthread\_mutex\_t m;

struct node

{

int value;

struct node \*next;

};

struct node \*root=NULL;

struct node \*create(int value)

{

struct node \*temp;

temp=(struct node\*)malloc(sizeof(struct node));

temp->value=value;

temp->next=NULL;

return temp;

}

void insert(int value)

{

struct node \*temp=create(value);

temp->next=root;

root=temp;

printf("\tElement %d inserted.\n",value);

}

int search(int value)

{

struct node \*temp=NULL;

temp=(struct node\*)malloc(sizeof(struct node));

temp=root;

while(temp!=NULL)

{

if(temp->value == value)

{

printf("\tElement %d found!\n",value);

return 1;

}

temp=temp->next;

}

printf("\tElement %d not found!\n",value);

return 0;

}

void delete(int value)

{

if(search(value)==0)

{

printf("\tElement %d can not be deleted.\n",value);

}

else

{

struct node \*temp=NULL;

temp=(struct node\*)malloc(sizeof(struct node));

struct node \*prev=NULL;

temp=(struct node\*)malloc(sizeof(struct node));

if(root->value==value)

{

root=root->next;

}

else

{

prev=root;

temp=prev->next;

while(temp->value!=value)

{

prev=temp;

temp=temp->next;

}

prev->next=temp->next;

}

printf("\tElement %d deleted.\n",value);

}

}

void choose(int choice, int value)

{

//pthread\_t calc;

switch(choice)

{

case 0:

// pthread\_create(&calc,NULL,insert,NULL);

insert(value);

break;

case 1:

// pthread\_create(&calc,NULL,delete,NULL);

delete(value);

break;

case 2:

// pthread\_create(&calc,NULL,search,NULL);

search(value);

break;

default:

printf("\tInvalid selection\n");

}

//p\_thread.join(calc,NULL);

}

void\* thread()

{

choose(rand() % 3,rand() % MAXVAL);

}

int main()

{

int i=0;

srand(time(0));

for(i=0; i<CALLS;i++)

{

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

pthread\_mutex\_lock(&m);

pthread\_t calc;

pthread\_create(&calc,NULL,&thread,NULL);

pthread\_join(calc,NULL);

pthread\_mutex\_unlock(&m);

}

return 0;

}

Ans 5:

/\*

Program to computes the total of the values of the matrix

using thread and mutex lock on global variable total

\*/

#include<stdio.h>

#include<pthread.h>

#include<stdlib.h>

#define ROW 10

#define COL 10

#define MAX\_NUM 100

int arr[ROW][COL];

long long int sum=0;

pthread\_mutex\_t m;

int i;

void \*add()

{

int j;

pthread\_mutex\_lock(&m);

int id=i;

pthread\_mutex\_unlock(&m);

long long int row\_sum=0;

for(j=0;j<COL;j++)

{

row\_sum+=arr[id][j];

}

pthread\_mutex\_lock(&m);

sum+=row\_sum;

pthread\_mutex\_unlock(&m);

}

int main()

{

int j;

srand(time(0));

for(i=0; i<ROW; i++)

for(j=0; j<COL; j++)

arr[i][j]=rand()%MAX\_NUM;

// printf((j==COL-1)?"%d\n":"%d ", arr[i][j]=rand()%MAX\_NUM);

pthread\_mutex\_init(&m,NULL);

pthread\_t t[ROW];

for(i=0; i<ROW; i++)

pthread\_create(&t[i],NULL,add,NULL);

for(i=0; i<ROW; i++)

pthread\_join(t[i],NULL);

printf("Sum: %lld\n", sum);

return 0;

}