Operating systems

project nine

Made by:

SAMRIDHI KOHLI 169108125

SANCHITT GUPTA 169108126

IT C

Multithreaded Sorting Algorithm

Program Statement

**Project 9: Multithreaded Sorting Algorithm**

Write a multithreaded sorting program that works as follows:

A list of integers is divided into two smaller lists of equal size. Two separate threads *(sorting threads)* sort each sublist using a sorting algorithm of your choice. The two sublists are then merged by a third thread *(merging thread)* which merges the two sublists into a single sorted list.

Because global data are shared across all threads, perhaps the easiest way to set up the data is to create a global array. Each sorting thread will work on one half of this array. A second global array of the same size as the unsorted integer array will also be established. The merging thread will then merge the two sublists into this second array.

Algorithm

Step 1: Include all necessary header files

Step 2: Define SIZE and NUMBER\_OF\_THREADS with value 10 and 3 respectively

Step 3: Merger and Sorter functions are declared

Step 4: Two arrays list and result each of size 10 are defined

Step 5: A user defined structure is defined with members from\_index and to\_index both of int type and the object name is parameters

Step 6: The main function starts which creates a thread that takes data from the first position to the middle position of the array and is sorted by the sorter function *(bubble sort)*

Step 7: Another thread is created that then takes data after the middle position to the end of the array and is also sorted by the sorter function

Step 8: Both these threads are then joined with the main function in a loop

Step 9: Another thread is created that takes up data from these two sorted threads from first position to the end of the array and is merged by the merger function

Step 10: Both the merger and sorter functions are then defined both of which use bubble sort algorithm

Code

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#define SIZE 10

#define NUMBER\_OF\_THREADS 3

void \*sorter(void \*params);

void \*merger(void \*params);

int list[SIZE] = {7,12,19,3,18,4,2,6,15,8};

int result[SIZE];

typedef struct

{

int from\_index;

int to\_index;

}parameters;

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

{

int i;

pthread\_t workers[NUMBER\_OF\_THREADS];

parameters \*data=(parameters \*)malloc(sizeof(parameters));

data->from\_index=0;

data->to\_index=(SIZE/2)-1;

pthread\_create(&workers[0],0,sorter,data);

data=(parameters \*)malloc(sizeof(parameters));

data->from\_index=(SIZE/2);

data->to\_index=SIZE-1;

pthread\_create(&workers[1],0,sorter,data);

for(i=0;i<NUMBER\_OF\_THREADS-1;i++)

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

data=(parameters \*)malloc(sizeof(parameters));

data->from\_index=0;

data->to\_index=SIZE-1;

pthread\_create(&workers[2],0,merger,data);

pthread\_join(workers[2], NULL);

return 0;

}

**void \*sorter(void \*params)**

{

parameters \*p =(parameters \*)params;

int i,j,k;

int begin=p->from\_index

int end=p->to\_index+1;

printf("\nThe array recieved is:\n");

for(i=begin;i<end;i++)

{

printf("%d\t",list[i]);

}

printf("\n");

int temp=0;

for(i=begin;i<end;i++)

{

for(j=begin;j<end-1;j++)

{

if(list[j]>list[j+1])

{ temp = list[j];

list[j] = list[j+1];

list[j+1] = temp;

}

}

}

printf("\nThe sorted array:\n");

for(k=begin;k<end;k++)

{

printf("%d\t",list[k]);

}

for(i=begin;i<end;i++)

{

result[i] = list[i];

}

printf("\n");

pthread\_exit(NULL);

}

**void \*merger(void \*params)**

{

parameters \*p=(parameters \*)params;

int i,j,d;

int begin=p->from\_index;

int end=p->to\_index+1;

int temp=0;

for(i=begin;i<end;i++)

{

for(j=begin;j<end-1;j++)

{

if(result[j]>result[j+1])

{ temp= result[j];

result[j] = result[j+1];

result[j+1] = temp;

}

}

}

printf("\n\nFINAL RESULT IS:\n");

for(d=begin;d<end;d++)

{ printf("%d\t", result[d]);

}

pthread\_exit(NULL);

}

Output

