527326-4-6E AID: 258164 | 11/10/2020

**The following show the program plan of parallel merge sort:**

1. Create c program with name “main.c” and define NOTHREADS2 .
2. In this program, provide two sorted array and these can be merge through the created linked list for holding the results of merge sort.
3. Create struct node and then merge() function can merge the array using thread.
4. Compile and run the program using any online or offline compiler. Like “onlineGDB.com”

**The following show the program code of parallel merge sort:**

Main.c

#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

#define NOTHREADS 2

int a[] = {10, 18, 15, 12, 13, 6, 7, 11, 4, 19};

typedef struct node {

int i;

int j;

} NODE;

void merge(int i, int j)

{

int mid = (i+j)/2;

int ai = i;

int bi = mid+1;

int newa[j-i+1], newai = 0;

while(ai <= mid && bi <= j) {

if (a[ai] > a[bi])

newa[newai++] = a[bi++];

else

newa[newai++] = a[ai++];

}

while(ai <= mid) {

newa[newai++] = a[ai++];

}

while(bi <= j) {

newa[newai++] = a[bi++];

}

for (ai = 0; ai < (j-i+1) ; ai++)

a[i+ai] = newa[ai];

}

void \* mergesort(void \*a)

{

NODE \*p = (NODE \*)a;

NODE n1, n2;

int mid = (p->i+p->j)/2;

pthread\_t tid1, tid2;

int returns;

n1.i = p->i;

n1.j = mid;

n2.i = mid+1;

n2.j = p->j;

if (p->i >= p->j)

return 0;

returns = pthread\_create(&tid1, NULL, mergesort, &n1);

if (returns) {

printf("%d %s - unable to make thread - returns - %d\n", \_\_L\_\_, \_\_FUN\_\_, returns);

exit(1);

}

returns = pthread\_create(&tid2, NULL, mergesort, &n2);

if (returns) {

printf("%d %s - unable to make thread - returns - %d\n", \_\_L\_\_, \_\_FUN\_\_, returns);

exit(1);

}

pthread\_join(tid1, NULL);

pthread\_join(tid2, NULL);

merge(p->i, p->j);

pthread\_exit(NULL);

}

int main()

{

int i;

NODE m;

m.i = 0;

m.j = 9;

pthread\_t tid;

int returns;

returns=pthread\_create(&tid, NULL, mergesort, &m);

if (returns) {

printf("%d %s - unable to make thread - returns - %d\n", \_\_L\_\_, \_\_FUN\_\_, returns);

exit(1);

}

pthread\_join(tid, NULL);

for (i = 0; i < 10; i++)

printf ("%d ", a[i]);

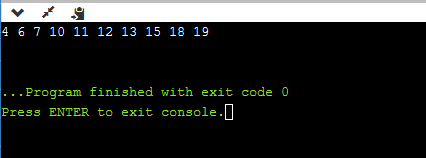
printf ("\n");

// pthread\_exit(NULL);

return 0;

}

**The output of code:**

****