**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP NMHĐH**

**HK2, 2021-2022**

**Lab 4**

**Nhóm:** 05 **Tổ:** 01

Thành Viên Nhóm:

1: Nguyễn Trần Quang Huy (MSSV: 52000668)

**Muc lục**

[A. PHẦN THỰC HÀNH 3](#_Toc98914391)

[Ví Dụ 1: 3](#_Toc98914392)

[1.1: Code Chương Trình 3](#_Toc98914393)

[1.2: Kết Quả Demo 3](#_Toc98914394)

[Ví Dụ 2: 5](#_Toc98914395)

[2.1: Code chương trình 5](#_Toc98914396)

[2.2 Kết quả Demo 6](#_Toc98914397)

[Ví Dụ 3: 6](#_Toc98914398)

[3.1 Code chương trình 6](#_Toc98914399)

[3.2 Kết quả demo 7](#_Toc98914400)

[Ví Dụ 4: 8](#_Toc98914401)

[4.1 Code chương trình 8](#_Toc98914402)

[4.2 Kết quả demo 10](#_Toc98914403)

[Ví Dụ 5: 11](#_Toc98914404)

[5.1 Code chương trình 11](#_Toc98914405)

[5.2 Kết quả demo 13](#_Toc98914406)

[B. PHẦN BÀI TẬP 14](#_Toc98914407)

[BÀI 1: 14](#_Toc98914408)

[A: Code Chương Trình: 14](#_Toc98914409)

[B: Kết Quả Demo: 16](#_Toc98914410)

[BÀI 2: 16](#_Toc98914411)

[A: Code Chương Trình: 16](#_Toc98914412)

[B: Kết Quả Demo: 18](#_Toc98914413)

[BÀI 3: 20](#_Toc98914414)

[A: Code Chương Trình: 20](#_Toc98914415)

[B: Kết Quả Demo: 23](#_Toc98914416)

[BÀI 4: 23](#_Toc98914417)

[A: Code Chương Trình: 23](#_Toc98914418)

[B: Kết Quả Demo: 25](#_Toc98914419)

[BÀI 5: 26](#_Toc98914420)

[A: Code Chương Trình: 26](#_Toc98914421)

[Kết Quả Demo: 28](#_Toc98914422)

[BÀI 6: 29](#_Toc98914423)

[A: Code Chương Trình: 29](#_Toc98914424)

[B: Kết Quả Demo: 33](#_Toc98914425)

[KẾT LUẬN 35](#_Toc98914426)

[KẾT QUẢ LÀM VIỆC CÁC THÀNH VIÊN TRONG NHÓM 36](#_Toc98914427)

# A. PHẦN THỰC HÀNH

# Ví Dụ 1:

Về hàm tạo thread và hàm pthread\_create()

## 1.1: Code Chương Trình

#include <pthread.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

void \* thr1(void \* ar) {

printf("This is thread % d\n", \*((int \* ) ar));

sleep(2);

}

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

int i;

int num = atoi(argv[1]);

pthread\_t tid[num]; //tạo ra mảng chứa num thread

for (i = 0; i < num; i++) { // chạy vòng lập để tạo ra thread bằng cách dùng hàm pthread\_create

pthread\_create( & tid[i], NULL, thr1, (void \* ) & tid[i]);

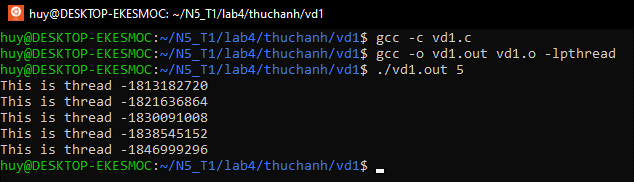
}

sleep(3);

return (0);

}

## 1.2: Kết Quả Demo



Hãy biên dịch, chạy thử, bỏ hàm sleep() ở dòng 17, hoặc tăng giá trị đối số sleep() và giải thích kết quả chạy cho từng tình huống.

Text

Description automatically generated

Trường hợp bỏ sleep() => số thread tạo ra không bằng chỉ số nhập vào theo em là do trường trình tạo không kịp ạ.

Text

Description automatically generated

Trường hợp tăng sleep() => số thread tạo ra bằng so với chỉ số nhập vào và thời gian chờ tổng thể > 10s 1 ít theo em do có thêm thời gian sleep nên số thread tạo ra đủ số lượng ạ.

Còn theo thầy giải thích là không đảm bảo rằng các tiểu trình con bắt theo kịp, điều đó phụ thuộc vào bộ lập lịch của CPU.

# Ví Dụ 2:

Truyền dữ liệu vào thread đã tạo

## 2.1: Code chương trình

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

struct arr {

int n;

int a[10];

};

void \* thr1(void \* ar) {

int count;

struct arr \* ap = (struct arr \* ) ar;

for (count = 0; count < ap -> n; count++)

printf("%d\t", ap -> a[count]);

printf("\n");

}

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

struct arr ar;

ar.n = 5;

int i;

for (i = 0; i < ar.n; i++)

ar.a[i] = i + 1;

pthread\_t tid;

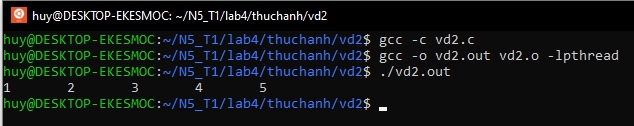
pthread\_create( & tid, NULL, & thr1, & ar);

sleep(2);

return 0;

}

## 2.2 Kết quả Demo



# Ví Dụ 3:

Về hàm pthread\_join

## 3.1 Code chương trình

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

void \* thr1(void \* ar) {

int count;

printf("This is thread %d\n", \*((int \* ) ar));

sleep(2);

}

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

int i;

pthread\_t tid[3];

int status, \* pstatus = & status;

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

pthread\_create( & tid[i], NULL, thr1, (void \* ) & tid[i]);

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

if (pthread\_join(tid[i], (void \*\* ) pstatus) > 0) {

printf("pthread\_join for thread %d failure\n", (int) tid[i]);

}

printf("pthread\_waited of %d OK, return code: %d\n", (int) tid[i], status);

sleep(1);

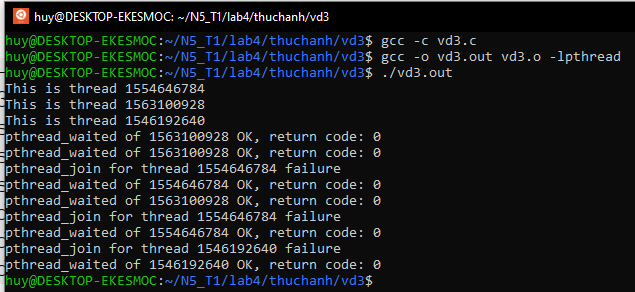
}

sleep(1);

return 0;

}

## 3.2 Kết quả demo



# Ví Dụ 4:

tổng hợp các ví dụ đã nêu.

- Hàm thr1 sử dụng để khởi tạo mảng

- Hàm thr2 sử dụng để tính tổng các phần tử của mảng.

- Hàm thr3 sử dụng để ghi mảng ra file

## 4.1 Code chương trình

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

struct arr {

int n;

int a[10];

};

struct file {

struct arr ar;

char \* filename;

};

static int sum = 0;

void \* thr1(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

ap -> n = 3;

int i = 0;

for (i = 0; i < ap -> n; i++)

ap -> a[i] = i + 1;

}

void \* thr2(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

int i, s = 0;

for (i = 0; i < ap -> n; i++)

s = s + ap -> a[i];

sum = s;

}

void \* thr3(void \* ar) {

struct file \* fi = (struct file \* ) ar;

FILE \* out;

int count;

out = fopen(fi -> filename, "wb");

fprintf(out, "number element or array: %d\n", fi -> ar.n);

for (count = 0; count < fi -> ar.n; count++) {

fprintf(out, "%d\t", fi -> ar.a[count]);

}

fprintf(out, "\n");

fprintf(out, "sum=%d\n", sum);

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

fclose(out);

}

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

int i;

pthread\_t tid[3];

struct arr ar;

int status, \* pstatus = & status;

pthread\_create( & tid[0], NULL, thr1, (void \* ) & ar);

sleep(1);

if (pthread\_join(tid[0], (void \*\* ) pstatus) == 0) {

pthread\_create( & tid[1], NULL, thr2, (void \* ) & ar);

if (pthread\_create( & tid[1], NULL, thr2, (void \* ) & ar) == 0) {

struct file arf;

arf.ar = ar;

arf.filename = argv[1];

pthread\_create( & tid[2], NULL, thr3, (void \* ) & arf);

}

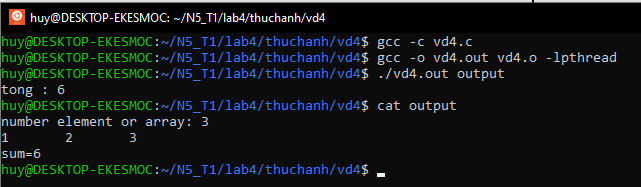
}

sleep(2);

return 0;

}

## 4.2 Kết quả demo



# Ví Dụ 5:

Tạo ra 2 thread đọc và ghi file

## 5.1 Code chương trình

// C program to Open a File,

// Write in it, And Close the File#include <stdio.h>

#include <string.h>

#include <stdio.h>

int main() {

// Declare the file pointer

FILE \* filePointer;

// Get the data to be written in file

char dataToBeWritten[50] = "GeeksforGeeks-A Computer Science Portal for Geeks ";

// Open the existing file GfgTest.c using fopen()

// in write mode using "w" attribute

filePointer = fopen("GfgTest.c", "w");

// Check if this filePointer is null

// which maybe if the file does not exist

if (filePointer == NULL) {

printf("GfgTest.c file failed to open.");

} else {

printf("The file is now opened.\n");

// Write the dataToBeWritten into the file

if (strlen(dataToBeWritten) > 0) {

// writing in the file using fputs()

fputs(dataToBeWritten, filePointer);

fputs("\n", filePointer);

}

// Closing the file using fclose()

fclose(filePointer);

printf("Data successfully written in file GfgTest.c\n");

printf("The file is now closed.");

}

return 0;

}

// C program to Open a File,

// Read from it, And Close the File

#include <stdio.h>

#include <string.h>

int main() {

// Declare the file pointer

FILE \* filePointer;

// Declare the variable for the data to be read

char dataToBeRead[50];

// Open the existing file GfgTest.c using fopen()

// in read mode using "r" attribute

filePointer = fopen("GfgTest.c", "r");

// Check if this filePointer is null

// which maybe if the file does not exist

if (filePointer == NULL) {

printf("GfgTest.c file failed to open.");

} else {

printf("The file is now opened.\n");

// Read the dataToBeRead from the file

// using fgets() method

while (fgets(dataToBeRead, 50, filePointer) !=

NULL) {

// Print the dataToBeRead

printf("%s", dataToBeRead);

}

// Closing the file using fclose()fclose(filePointer) ;

printf("Data successfully read from file GfgTest.c\n");

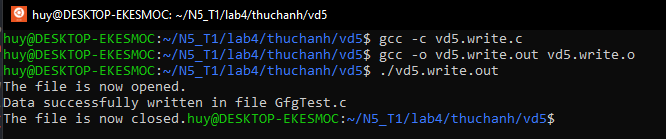
printf("The file is now closed.");

}

return 0;

}

## 5.2 Kết quả demo



Write to file

Text

Description automatically generated

Read from file

# B. PHẦN BÀI TẬP

# BÀI 1:

(Bài tập 4.22) Viết chương trình đa luồng tính toán các giá trị thống kê khác nhau từ một danh sách các số được truyền vào thông qua đối số của dòng lệnh. Chương trình sau đó sẽ tạo ba tiểu trình tính toán riêng biệt. Một tiểu trình sẽ xác định trung bình cộng của các số, tiểu trình thứ hai sẽ xác định giá trị lớn nhất và tiểu trình thứ ba sẽ xác định giá trị nhỏ nhất.

## A: Code Chương Trình:

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

struct arr {

int n;

int a[20];

};

static float avg = 0;

static int max = 0;

static int min = 0;

void \* thr1(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

int i = 0;

float sum = 0;

for (i = 0; i < ap -> n; i++) {

sum += ap -> a[i];

}

avg = sum / ap -> n;

}

void \* thr2(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

int i = 0;

max = ap -> a[0];

for (i = 1; i < ap -> n; i++) {

if (max < ap -> a[i]) {

max = ap -> a[i];

}

}

}

void \* thr3(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

int i = 0;

min = ap -> a[0];

for (i = 1; i < ap -> n; i++) {

if (min > ap -> a[i]) {

min = ap -> a[i];

}

}

}

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

int i;

pthread\_t tid[3];

struct arr ar;

ar.n = argc - 1;

for (int i = 0; i < argc - 1; i++) {

ar.a[i] = atoi(argv[i + 1]);

}

int status, \* pstatus = & status;

pthread\_create( & tid[0], NULL, thr1, (void \* ) & ar);

pthread\_join(tid[0], NULL);

pthread\_create( & tid[1], NULL, thr2, (void \* ) & ar);

pthread\_join(tid[1], NULL);

pthread\_create( & tid[2], NULL, thr3, (void \* ) & ar);

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

printf("so lon nhat: %d\n", max);

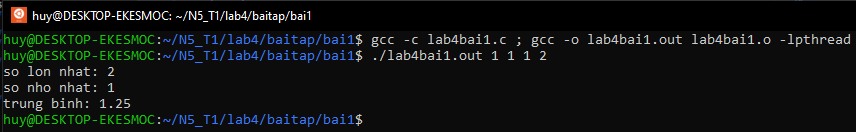
printf("so nho nhat: %d\n", min);

printf("trung binh: %.2f\n", avg);

return 0;

}

## B: Kết Quả Demo:



# BÀI 2:

(Bài tập 4.23) Viết chương trình đa luồng để xuất ra số nguyên tố. Người dùng chạy chương trình và nhập vào một số nguyên thông qua đối số tại dòng lệnh. Chương trình sau đó sẽ tạo ra một tiến trình riêng biệt xuất ra tất cả các số nguyên tố nhỏ hơn hoặc bằng số được nhập bởi người dùng.

## A: Code Chương Trình:

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

struct arr {

int n;

};

int isPrime(int num) {

if (num < 2) {

return 0;

}

if (num == 2) {

return 1;

}

for (int i = 2; i < num; i++) {

if (num % i == 0) {

return 0;

}

}

return 1;

}

void \* thr1(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

for (int i = 0; i <= ap -> n; i++) {

if (isPrime(i)) {

printf("%d,\t", i);

}

}

printf("\n");

}

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

int n = atoi(argv[1]);

struct arr ar;

ar.n = n;

pthread\_t tid;

pthread\_create( & tid, NULL, thr1, (void \* ) & ar);

pthread\_join(tid, NULL);

return 0;

}

## B: Kết Quả Demo:

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated with medium confidence

# BÀI 3:

Chỉnh sửa lại bài 1 sao cho thay vì xuất kết quả ra màn hình thì kết quả sẽ được ghi vào tập tin result.txt ở cùng thư mục với chương trình chạy.

## A: Code Chương Trình:

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

struct arr {

int n;

int a[20];

};

struct file {

struct arr ar;

char \* filename;

};

static float avg = 0;

static int max = 0;

static int min = 0;

void \* thr1(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

int i = 0;

float sum = 0;

for (i = 0; i < ap -> n; i++) {

sum += ap -> a[i];

}

avg = sum / ap -> n;

}

void \* thr2(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

int i = 0;

max = ap -> a[0];

for (i = 1; i < ap -> n; i++) {

if (max < ap -> a[i]) {

max = ap -> a[i];

}

}

}

void \* thr3(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

int i = 0;

min = ap -> a[0];

for (i = 1; i < ap -> n; i++) {

if (min > ap -> a[i]) {

min = ap -> a[i];

}

}

}

void \* thr4(void \* ar) {

struct file \* fi = (struct file \* ) ar;

FILE \* out;

int count;

out = fopen(fi -> filename, "wb");

fprintf(out, "so lon nhat: %d\n", max);

fprintf(out, "so nho nhat: %d\n", min);

fprintf(out, "trung binh: %.2f\n", avg);

fclose(out);

}

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

int i;

pthread\_t tid[4];

struct arr ar;

ar.n = argc - 1;

for (int i = 0; i < argc - 1; i++) {

ar.a[i] = atoi(argv[i + 1]);

}

int status, \* pstatus = & status;

pthread\_create( & tid[0], NULL, thr1, (void \* ) & ar);

pthread\_join(tid[0], NULL);

pthread\_create( & tid[1], NULL, thr2, (void \* ) & ar);

pthread\_join(tid[1], NULL);

pthread\_create( & tid[2], NULL, thr3, (void \* ) & ar);

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

struct file arf;

arf.ar = ar;

arf.filename = "result.txt";

pthread\_create( & tid[3], NULL, thr4, (void \* ) & arf);

pthread\_join(tid[3], NULL);

printf("so lon nhat: %d\n", max);

printf("so nho nhat: %d\n", min);

printf("trung binh: %.2f\n", avg);

return 0;

}

## B: Kết Quả Demo:

A screenshot of a computer

Description automatically generated

# BÀI 4:

Viết chương trình để sao chép dữ liệu từ file nguồn vào file đích. Với tên file nguồn và file đích là đối số đầu vào. Đếm số kí tự đã sao chép. Giả sử chương trình chỉ chạy trên tập tin text.

## A: Code Chương Trình:

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

struct arr {

int n;

char s[1000];

};

struct file {

struct arr ar;

char \* filename;

};

void \* thrRead(void \* ar) {

struct file \* fi = (struct file \* ) ar;

FILE \* in ;

int count; in = fopen(fi -> filename, "rb");

char ch;

int i = 0;

do {

ch = fgetc( in );

fi -> ar.s[i++] = ch;

// printf("%c", ch);

} while (ch != EOF);

fi -> ar.n = i;

// printf("\n");

fclose( in );

}

void \* thrWrite(void \* ar) {

struct file \* fi = (struct file \* ) ar;

FILE \* out;

int count;

out = fopen(fi -> filename, "wb");

for (int i = 0; i < fi -> ar.n; i++) {

fprintf(out, "%c", fi -> ar.s[i]);

}

printf("Da sao chep thanh cong %d ki tu.\n", fi -> ar.n);

fclose(out);

}

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

pthread\_t tid[4];

int status, \* pstatus = & status;

struct file arf;

// arf.ar = ar;

arf.filename = argv[1];

pthread\_create( & tid[0], NULL, thrRead, (void \* ) & arf);

pthread\_join(tid[0], NULL);

// struct file arf;

// arf.ar = ar;

arf.filename = argv[2];

pthread\_create( & tid[1], NULL, thrWrite, (void \* ) & arf);

pthread\_join(tid[1], NULL);

return 0;

}

## B: Kết Quả Demo:

Text

Description automatically generated

# BÀI 5:

Viết chương trình tạo ra 3 thread thực hiện các công việc sau:

* Thread thứ nhất nhận đối số truyền từ môi trường (argv[1]), kiểm tra nếu số này lớn 0, tính giai thừa của số này (có thể sử dụng struct hoặc biến toàn cục).
* Thread thứ hai chờ thread thứ nhất hoàn tất, thực hiện việc tính tổng các số chẳn từ nhỏ hơn kết quả giai thừa thread thứ nhất tính được.
* Thread thứ ba chờ thread thứ 2 hoàn tất, ghi kết quả vào file là đối số thứ 2 từ biến môi trường (argv[2]). Nội dung ghi vào gồm: dòng thứ nhất lưu giá trị của argv[1]; dòng thứ 2 lưu kết quả của giai thừa; dòng thứ 3 lưu tổng các số chẵn nhỏ hơn giai thừa.

## A: Code Chương Trình:

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

struct arr {

int n;

int a[10];

};

struct file {

struct arr ar;

char \* filename;

};

static int n = 0;

static long loser = 1;

static long sum = 0;

void \* thr1(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

if (ap -> n <= 0) {

printf("can nhap n > 0\n");

return 0;

}

for (int i = 1; i <= ap -> n; i++) {

loser \*= i;

}

}

void \* thr2(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

for (long i = 1; i <= loser; i++) {

if (i % 2 == 0) {

sum += i;

}

}

}

void \* thr3(void \* ar) {

struct file \* fi = (struct file \* ) ar;

FILE \* out;

int count;

out = fopen(fi -> filename, "w");

fprintf(out, "n = %d\n", fi -> ar.n);

fprintf(out, "%d! = %ld\n", fi -> ar.n, loser);

fprintf(out, "sum = %ld\n", sum);

fclose(out);

}

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

int i;

pthread\_t tid[3];

struct arr ar;

ar.n = atoi(argv[1]);

int status, \* pstatus = & status;

pthread\_create( & tid[0], NULL, thr1, (void \* ) & ar);

// pthread\_join(tid[0], NULL);

if (pthread\_join(tid[0], (void \*\* ) pstatus) == 0) {

if (pthread\_create( & tid[1], NULL, thr2, (void \* ) & ar) == 0) {

struct file arf;

arf.ar = ar;

arf.filename = argv[2];

pthread\_create( & tid[2], NULL, thr3, (void \* ) & arf);

}

}

printf("n = %d\n", ar.n);

printf("%d! = %ld\n", ar.n, loser);

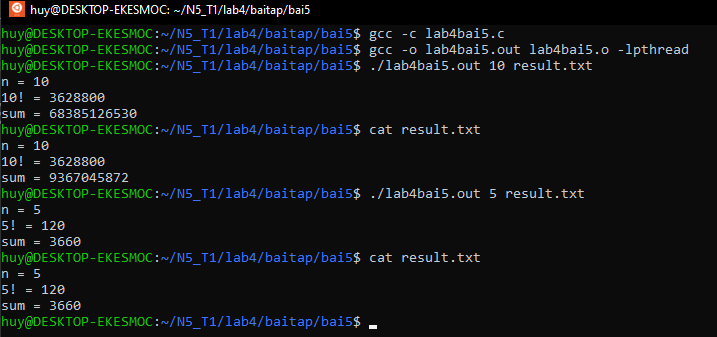
printf("sum = %ld\n", sum);

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

return 0;

}

## Kết Quả Demo:



# BÀI 6:

Cho một tập tin có cấu trúc sau:

* Dòng đầu tiên chứa số phần tử mảng
* Dòng còn lại chứa các phần tử là số nguyên

Viết chương trình gồm các thread thực hiện các công việc sau:

* Thread thứ nhất đọc file đầu vào là đối số thứ nhất từ biến môi trường
* Thread thứ hai tính tổng các số nguyên tố trong mảng
* Thread thứ ba tính sắp xếp mảng tăng dần
* Thread thứ tư thực hiện việc ghi file result. Nội dung file đầu vào và đầu ra như sau:

## A: Code Chương Trình:

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

struct arr {

int n;

int a[100];

};

struct file {

struct arr ar;

char \* filename;

};

static int size = 0;

static int arr[100];

static int arrSorted[100];

static long sum = 0;

void \* thr1(void \* ar) {

struct file \* fi = (struct file \* ) ar;

FILE \* in ;

int count; in = fopen(fi -> filename, "r");

fscanf( in , "%d", & (fi -> ar.n));

for (int i = 0; i < fi -> ar.n; i++) {

int temp = 0;

fscanf( in , "%d", & temp);

fi -> ar.a[i] = temp;

}

fclose( in );

}

int isPrime(int num) {

if (num < 2) {

return 0;

}

if (num == 2) {

return 1;

}

for (int i = 2; i < num; i++) {

if (num % i == 0) {

return 0;

}

}

return 1;

}

void \* thr2(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

int j = 0;

for (int i = 0; i < ap -> n; i++) {

if (isPrime(ap -> a[i])) {

sum += ap -> a[i];

arrSorted[j] = ap -> a[i];

arr[j++] = ap -> a[i];

}

}

size = j;

}

void \* thr3(void \* ar) {

struct arr \* ap = (struct arr \* ) ar;

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

for (int j = i + 1; j < size; j++) {

if (arrSorted[i] > arrSorted[j]) {

int temp = arrSorted[i];

arrSorted[i] = arrSorted[j];

arrSorted[j] = temp;

}

}

}

}

void \* thr4(void \* ar) {

struct file \* fi = (struct file \* ) ar;

FILE \* out;

int count;

out = fopen(fi -> filename, "w");

fprintf(out, "So phan tu mang: %d\n", fi -> ar.n);

for (int i = 0; i < fi -> ar.n; i++) {

fprintf(out, "%d ", fi -> ar.a[i]);

}

fprintf(out, "\n");

fprintf(out, "Mang cac so nguyen to:\n");

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

fprintf(out, "%d ", arr[i]);

}

fprintf(out, "\n");

fprintf(out, "Tong cac so nguyen to: %ld\n", sum);

fprintf(out, "Mang cac so nguyen to da duoc sap xep: \n");

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

fprintf(out, "%d ", arrSorted[i]);

}

fprintf(out, "\n");

fclose(out);

}

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

int i;

pthread\_t tid[4];

struct arr ar;

ar.n = atoi(argv[1]);

struct file arf;

arf.ar = ar;

arf.filename = argv[1];

int status, \* pstatus = & status;

pthread\_create( & tid[0], NULL, thr1, (void \* ) & arf);

if (pthread\_join(tid[0], (void \*\* ) pstatus) == 0) { // nếu đọc file thành công thì vô tính tiếp

if (pthread\_create( & tid[1], NULL, thr2, (void \* ) & arf) == 0) { // tính tổng các số nguyên tố trong mảng

if (pthread\_create( & tid[2], NULL, thr3, (void \* ) & arf) == 0) { // sắp xếp mảng số nguyên tố

arf.filename = "result.txt";

pthread\_create( & tid[3], NULL, thr4, (void \* ) & arf); // ghi ra file

}

}

}

pthread\_join(tid[3], NULL);

return 0;

}

* Thread thứ hai tính tổng các số nguyên tố trong mảng
* Thread thứ ba tính sắp xếp mảng tăng dần

Thread thứ tư thực hiện việc ghi file result. Nội dung file đầu vào và đầu ra như sau

## B: Kết Quả Demo:

A screenshot of a computer

Description automatically generated

# KẾT LUẬN

Sau khi học và hoàn thành phần **LAB 4** nhóm thu được kết sau:

* Cách tạo các tác vụ chạy song song với nhau
* Cách tạo các tác vụ chạy gối nhau

# KẾT QUẢ LÀM VIỆC CÁC THÀNH VIÊN TRONG NHÓM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HỌ TÊN SV | MÃ SV | VAI TRÒ | CÔNG VIỆC ĐƯỢC PHÂN CÔNG | HT |
| Nguyễn Trần Quang Huy | 52000668 | TN | Làm cả phần thực hành và bài tập | 100% |
|  |  |  |  |  |

CHÚ THÍCH:

TN: TRƯỞNG NHÓM

TV: THÀNH VIÊN

HT: MỨC ĐỘ HOÀN THÀNH CÔNG VIỆC ĐƯỢC GIAO