**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP NMHĐH**

**HK2, 2021-2022**

**Lab 5**

**Nhóm:** 05 **Tổ:** 01

Thành Viên Nhóm:

1: Nguyễn Trần Quang Huy (MSSV: 52000668)

**Muc lục**

[A. PHẦN THỰC HÀNH 4](#_Toc100869422)

[B. PHẦN BÀI TẬP 4](#_Toc100869423)

[BÀI 1.1: 4](#_Toc100869424)

[A: Code Chương Trình: 4](#_Toc100869425)

[B: Kết Quả Demo: 5](#_Toc100869426)

[BÀI 1.2: 6](#_Toc100869427)

[A: Code Chương Trình: 6](#_Toc100869428)

[B: Kết Quả Demo: 8](#_Toc100869429)

[BÀI 1.3: 8](#_Toc100869430)

[A: Code Chương Trình: 8](#_Toc100869431)

[B: Kết Quả Demo: 10](#_Toc100869432)

[BÀI 1.4: 11](#_Toc100869433)

[A: Code Chương Trình: 11](#_Toc100869434)

[B: Kết Quả Demo: 12](#_Toc100869435)

[BÀI 1.5: 13](#_Toc100869436)

[A: Code Chương Trình: 13](#_Toc100869437)

[Kết Quả Demo: 14](#_Toc100869438)

[BÀI 2: 15](#_Toc100869439)

[A: Code Chương Trình: 15](#_Toc100869440)

[B: Kết Quả Demo: 17](#_Toc100869441)

[BÀI 3: 17](#_Toc100869442)

[A: Code Chương Trình: 17](#_Toc100869443)

[B: Kết Quả Demo: 23](#_Toc100869444)

[KẾT LUẬN 24](#_Toc100869445)

[KẾT QUẢ LÀM VIỆC CÁC THÀNH VIÊN TRONG NHÓM 25](#_Toc100869446)

# A. PHẦN THỰC HÀNH

# B. PHẦN BÀI TẬP

# BÀI 1.1:

Tính thời gian quay vòng trung bình, thời gian chờ trung bình cho thuật toán FCFS

## A: Code Chương Trình:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "task.h"

#include "cpu.h"

#include "schedulers.h"

#define SIZE 100

static int i = 0;

Task task[SIZE];

void add(char \* name, int priority, int arrival, int burst) {

task[i].name = name;

task[i].priority = priority;

task[i].arrival = arrival;

task[i].burst = burst;

i++;

}

void schedule() {

// yc 1.2

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

for (int k = h + 1; k < i; k++) {

if (task[h].arrival > task[k].arrival) {

Task temp = task[h];

task[h] = task[k];

task[k] = temp;

}

}

}

int j = 0;

int time = 0;

float arround = 0;

float aver1;

// yc 1.1

float wait = 0;

float aver2;

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

run( & task[j], time, task[j].burst);

time += task[j].burst;

arround += time - task[j].arrival;

wait += time - task[j].burst;

}

aver1 = arround / 8;

aver2 = wait / 8;

printf("Thoi gian xoay vong trung binh cua cac tien trinh la: %.2f \n", aver1);

printf("Thoi gian cho trung binh cua cac tien trinh la: %.2f \n", aver2);

}

## B: Kết Quả Demo:

Text

Description automatically generated

# BÀI 1.2:

Giả sử thời gian đến của mỗi tiến trình khác nhau và khác 0, tính thời gian quay vòng trung bình, thời gian chờ trung bình cho thuật toán FCFS

## A: Code Chương Trình:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "task.h"

#include "cpu.h"

#include "schedulers.h"

#define SIZE 100

static int i = 0;

Task task[SIZE];

void add(char \* name, int priority, int arrival, int burst) {

task[i].name = name;

task[i].priority = priority;

task[i].arrival = arrival;

task[i].burst = burst;

i++;

}

void schedule() {

// yc 1.2

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

for (int k = h + 1; k < i; k++) {

if (task[h].arrival > task[k].arrival) {

Task temp = task[h];

task[h] = task[k];

task[k] = temp;

}

}

}

int j = 0;

int time = 0;

float arround = 0;

float aver1;

// yc 1.1

float wait = 0;

float aver2;

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

run( & task[j], time, task[j].burst);

time += task[j].burst;

arround += time - task[j].arrival;

wait += time - task[j].burst;

}

aver1 = arround / 8;

aver2 = wait / 8;

printf("Thoi gian xoay vong trung binh cua cac tien trinh la: %.2f \n", aver1);

printf("Thoi gian cho trung binh cua cac tien trinh la: %.2f \n", aver2);

}

## B: Kết Quả Demo:

Text

Description automatically generated

# BÀI 1.3:

Hiện thực tập tin schedule\_priority.c

## A: Code Chương Trình:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "task.h"

#include "cpu.h"

#include "schedulers.h"

#define MIN\_PRIORITY 1

#define MAX\_PRIORITY 10

#define SIZE 100

static int i = 0;

Task task[SIZE];

void add(char \* name, int priority, int arrival, int burst) {

task[i].name = name;

task[i].priority = priority;

task[i].arrival = arrival;

task[i].burst = burst;

i++;

}

void schedule() {

// yc 1.2

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

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

if (task[k].priority > task[k + 1].priority) {

Task temp = task[k + 1];

task[k + 1] = task[k];

task[k] = temp;

}

}

}

int j = 0;

int time = 0;

float arround = 0;

float aver1;

// yc 1.1

float wait = 0;

float aver2;

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

run( & task[j], time, task[j].burst);

time += task[j].burst;

arround += time - task[j].arrival;

wait += time - task[j].burst;

}

aver1 = arround / 8;

aver2 = wait / 8;

printf("Thoi gian xoay vong trung binh cua cac tien trinh la: %.2f \n", aver1);

printf("Thoi gian cho trung binh cua cac tien trinh la: %.2f \n", aver2);

}

## B: Kết Quả Demo:

Text

Description automatically generated

# BÀI 1.4:

Hiện thực tập tin schedule\_sjf.c

## A: Code Chương Trình:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "task.h"

#include "cpu.h"

#include "schedulers.h"

#define MIN\_PRIORITY 1

#define MAX\_PRIORITY 10

#define SIZE 100

static int i = 0;

Task task[SIZE];

void add(char \* name, int priority, int arrival, int burst) {

task[i].name = name;

task[i].priority = priority;

task[i].arrival = arrival;

task[i].burst = burst;

i++;

}

void schedule() {

// yc 1.2

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

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

if (task[k].burst > task[k + 1].burst) {

Task temp = task[k + 1];

task[k + 1] = task[k];

task[k] = temp;

}

}

}

int j = 0;

int time = 0;

float arround = 0;

float aver1;

// yc 1.1

float wait = 0;

float aver2;

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

run( & task[j], time, task[j].burst);

time += task[j].burst;

arround += time - task[j].arrival;

wait += time - task[j].burst;

}

aver1 = arround / 8;

aver2 = wait / 8;

printf("Thoi gian xoay vong trung binh cua cac tien trinh la: %.2f \n", aver1);

printf("Thoi gian cho trung binh cua cac tien trinh la: %.2f \n", aver2);

}

## B: Kết Quả Demo:

Text

Description automatically generated

# BÀI 1.5:

Hiện thực tập tin schedule\_rr.c

## A: Code Chương Trình:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "task.h"

#include "cpu.h"

#include "schedulers.h"

#define MIN\_PRIORITY 1

#define MAX\_PRIORITY 10

#define SIZE 100

static int i = 0;

Task task[SIZE];

void add(char \* name, int priority, int arrival, int burst) {

task[i].name = name;

task[i].priority = priority;

task[i].arrival = arrival;

task[i].burst = burst;

i++;

}

void schedule() {

int j = 0;

int time = 0;

int timeQuantum = 10;

while (i >= 0) {

if (task[j].burst <= timeQuantum) {

run( & task[j], time, task[j].burst);

time += task[j].burst;

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

task[h] = task[h + 1];

}

i--;

} else {

run( & task[j], time, timeQuantum);

time += timeQuantum;

task[j].burst -= timeQuantum;

}

}

}

## Kết Quả Demo:

Graphical user interface, text

Description automatically generated

# BÀI 2:

Ước lượng giá trị số PI

## A: Code Chương Trình:

#include <pthread.h>

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

static int counter = 0;

static int m;

void \* thr(void \* ar) {

FILE \* out\_file = fopen("m\_point.txt", "a");

int max = 2;

int min = -1;

for (int index = 0; index < m; index++) {

float x = (((float) rand() / (float)(RAND\_MAX)) \* max) + min;

float y = (((float) rand() / (float)(RAND\_MAX)) \* max) + min;

fprintf(out\_file, "(%2.2f, %2.2f)\n", x, y);

float d = sqrt(x \* x + y \* y);

if (d <= 1.0) {

counter += 1;

}

}

}

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

int i;

int status, \* pstatus = & status;

int n = atoi(argv[1]);

m = atoi(argv[2]);

pthread\_t tid[n];

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

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

sleep(1);

}

printf("%d\n", counter);

printf("pi = %.6f\n", 4.0 \* counter / (n \* m));

return 0;

}

## B: Kết Quả Demo:

Text

Description automatically generated

# BÀI 3:

Ứng dụng sắp xếp bằng tiểu trình (merge sort)

## A: Code Chương Trình:

#include <stdio.h>

#include <pthread.h>

#include <time.h>

#include <stdlib.h>

int opt\_a;

int opt\_t;

int opt\_r;

int MAX;

int THREAD\_MAX;

int \* a;

struct tsk {

int tsk\_no;

int tsk\_low;

int tsk\_high;

};

void merge(int low, int mid, int high) {

int n1 = mid - low + 1;

int n2 = high - mid;

int \* left = malloc(n1 \* sizeof(int));

int \* right = malloc(n2 \* sizeof(int));

int i;

int j;

for (i = 0; i < n1; i++)

left[i] = a[i + low];

for (i = 0; i < n2; i++)

right[i] = a[i + mid + 1];

int k = low;

i = j = 0;

while (i < n1 && j < n2) {

if (left[i] <= right[j])

a[k++] = left[i++];

else

a[k++] = right[j++];

}

while (i < n1)

a[k++] = left[i++];

while (j < n2)

a[k++] = right[j++];

free(left);

free(right);

}

void merge\_sort(int low, int high) {

int mid = low + (high - low) / 2;

if (low < high) {

merge\_sort(low, mid);

merge\_sort(mid + 1, high);

merge(low, mid, high);

}

}

void \*

merge\_sort123(void \* arg) {

struct tsk \* tsk = arg;

int low;

int high;

low = tsk -> tsk\_low;

high = tsk -> tsk\_high;

int mid = low + (high - low) / 2;

if (low < high) {

merge\_sort(low, mid);

merge\_sort(mid + 1, high);

merge(low, mid, high);

}

return 0;

}

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

clock\_t start, end, duration;

start = clock();

char \* cp;

struct tsk \* tsk;

MAX = 20;

THREAD\_MAX = 4;

opt\_a = 1;

a = malloc(sizeof(int) \* MAX);

if (opt\_t)

printf("ORIG:");

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

if (opt\_r)

a[i] = MAX - i;

else

a[i] = rand() % 100;

if (opt\_t)

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

}

if (opt\_t)

printf("\n");

printf("\n\nInput:");

for (int i = 0; i < MAX; i++)

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

printf("\n");

pthread\_t threads[THREAD\_MAX];

struct tsk tsklist[THREAD\_MAX];

int len = MAX / THREAD\_MAX;

int low = 0;

for (int i = 0; i < THREAD\_MAX; i++, low += len) {

tsk = & tsklist[i];

tsk -> tsk\_no = i;

if (opt\_a) {

tsk -> tsk\_low = low;

tsk -> tsk\_high = low + len - 1;

if (i == (THREAD\_MAX - 1))

tsk -> tsk\_high = MAX - 1;

} else {

tsk -> tsk\_low = i \* (MAX / THREAD\_MAX);

tsk -> tsk\_high = (i + 1) \* (MAX / THREAD\_MAX) - 1;

}

if (opt\_t)

printf("RANGE %d: %d %d\n", i, tsk -> tsk\_low, tsk -> tsk\_high);

}

for (int i = 0; i < THREAD\_MAX; i++) {

tsk = & tsklist[i];

pthread\_create( & threads[i], NULL, merge\_sort123, tsk);

}

for (int i = 0; i < THREAD\_MAX; i++)

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

merge(0, (MAX / 2 - 1) / 2, MAX / 2 - 1);

merge(MAX / 2, MAX / 2 + (MAX - 1 - MAX / 2) / 2, MAX - 1);

merge(0, (MAX - 1) / 2, MAX - 1);

printf("\n\nOutput:");

for (int i = 0; i < MAX; i++)

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

printf("\n");

end = clock();

duration = (end - start);

printf("Time taken : %f seconds\n", (float) duration / CLOCKS\_PER\_SEC);

return 0;

}

## B: Kết Quả Demo:

A screenshot of a computer

Description automatically generated with medium confidence

# KẾT LUẬN

Sau khi học và hoàn thành phần **LAB 5** nhóm thu được kết sau:

* Các viết các giải thuật fcfs, sjf, priority, rr, …
* Tính ước lượng số Pi
* Học được thuật toán merge sort và ứng dụng của lập trình multi-thread

# 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