|  |
| --- |
| Name: Phạm Mai Dung  ID: 19520477  Class: IT007.L21.1 |

OPERATING SYSTEM  
LAB 4’S REPORT

**SUMMARY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | | **Status** | **Page** |
| Section 1.5 | Ex 1 | Done | 2 – 7 |
| Ex 2 | Done | 8 – 13 |
| Ex 3 | Done | 14 – 18 |
|  |  |  |
|  |  |  |

**Self-scrores: 9**

*\*Note: Export file to* ***PDF*** *and name the file by following format:* ***LAB X – <Student ID>.pdf***

# Section 1.5

## Task name 1: Chương trình mô phỏng giải thuật SJF:

* Diagram, schematic

  Description automatically generatedLưu đồ thuật toán:

Hình 1.1 \_ lưu đồ giải thuật SJF

* Source code:

*/\*######################################*

*# University of Information Technology #*

*# IT007 Operating System #*

*# Pham Mai Dung, 19520477 #*

*# File: sjf.cpp #*

*######################################\*/*

*#include <iostream>*

*using namespace std;*

*struct Process*

*{*

*float pn = 0, arr = 0, burst = 0, start = 0, response = 0, wait = 0, tat = 0, finish = 0;*

*};*

*void swap(float& a, float& b) {*

*float temp;*

*temp = a;*

*a = b;*

*b = temp;*

*}*

*void sortAscendingBurstTime(Process p[], int m, int n) {*

*for (int i = m; i < n - 1; i++) {*

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

*if (p[i].burst > p[j].burst) {*

*swap(p[i].pn, p[j].pn);*

*swap(p[i].arr, p[j].arr);*

*swap(p[i].burst, p[j].burst);*

*}*

*}*

*}*

*}*

*void sortAscendingArrivalTime(Process p[], int n)*

*{*

*for (int i = 0; i < n - 1; i++)*

*{*

*for (int j = i + 1; j < n; j++)*

*{*

*if (p[i].arr > p[j].arr)*

*{*

*swap(p[i].pn, p[j].pn);*

*swap(p[i].arr, p[j].arr);*

*swap(p[i].burst, p[j].burst);*

*}*

*else if (p[i].arr == p[j].arr && p[i].burst > p[j].burst)*

*{*

*swap(p[i].pn, p[j].pn);*

*swap(p[i].arr, p[j].arr);*

*swap(p[i].burst, p[j].burst);*

*}*

*}*

*}*

*}*

*int main()*

*{*

*Process\* p;*

*int num;*

*cout << "Enter the number of processes: ";*

*cin >> num;*

*p = new Process[num];*

*float totalBurst = 0;*

*for (int i = 0; i < num; i++)*

*{*

*cout << "Enter Process Name: ";*

*cin >> p[i].pn;*

*cout << "Enter Arrival Time: ";*

*cin >> p[i].arr;*

*cout << "Enter Burst Time: ";*

*cin >> p[i].burst;*

*}*

*sortAscendingArrivalTime(p, num);*

*float presentTime = 0; int temp;*

*for (int i = 0; i < num; i++)*

*{*

*if (i == 0)*

*{*

*p[i].start = p[i].arr;*

*p[i].wait = p[i].start - p[i].arr;*

*p[i].response = p[i].wait;*

*p[i].finish = p[i].start + p[i].burst;*

*p[i].tat = p[i].finish - p[i].arr;*

*presentTime = p[i].finish;*

*}*

*else*

*{*

*while (p[i].arr > presentTime)*

*{*

*presentTime++;*

*}*

*for (int j = i; j <= num; j++)*

*{*

*if (p[j].arr <= presentTime)*

*temp = j;*

*}*

*sortAscendingBurstTime(p, i, temp);*

*p[i].start = presentTime;*

*p[i].wait = p[i].start - p[i].arr;*

*p[i].response = p[i].wait;*

*p[i].finish = p[i].start + p[i].burst;*

*p[i].tat = p[i].finish - p[i].arr;*

*presentTime = p[i].finish;*

*}*

*}*

*float totalwt = 0, totaltat = 0;*

*cout << "PName | Arrival | Burst | Start | Response | Waiting | TAT | FinishTime\n";*

*for (int i = 0; i < num; i++)*

*{*

*cout << p[i].pn << "\t\t" << p[i].arr << "\t" << p[i].burst << "\t" << p[i].start << "\t" << p[i].response << "\t" << p[i].wait << "\t" << p[i].tat << "\t" << p[i].finish << "\n";*

*totalwt += p[i].wait;*

*totaltat += p[i].tat;*

*}*

*cout << "total Waiting Time: " << totalwt << endl;*

*cout << "Average Waiting Time: " << (float)totalwt / num << endl;*

*cout << "total TurnAround Time: " << totaltat << endl;*

*cout << "Average TurnAround Time: " << (float)totaltat / num << endl;*

*return 0;*

}

* Giải thích:
* Khai báo một mảng cấu trúc process có n phần tử chứa thông tin của process.
* Viết hàm swap(float& a, float& b) để hoán vị 2 số a, b nhằm phục vụ các hàm sắp xếp.
* Hàm sortAscendingBurstTime() để sắp xếp các process theo thứ tự tăng dần burst time.
* Hàm sortAscendingArrivalTime() để sắp xếp các process theo thứ tự tăng dần arrival time, sắp xếp khi process vào không theo thứ tự.
* Sau khi nhập thông tin processes và sắp xếp theo arrival time, lúc này các phần tử trong mảng đã được gán theo thứ tự sắp xếp, process thứ 0 sẽ được thực thi (tính các giá trị theo yêu cầu) đầu tiên. Sau khi nó chạy xong, cập nhật biến temp = phần tử cuối cùng, sắp xếp các process theo burst time tăng dần từ phần tử i đến temp, rồi cho các process còn lại thực thi lần lượt. Xuất ra các thông tin process (arrival, burst, response, waiting, turn around, finish time); tính waiting time và turn around time trung bình, sau đó xuất ra màn hình, ta được kết quả như hình dưới.
* Source code được viết bằng c++, do đó em đã cài ide g++ và thực thi chương trình bằng lệnh *g++ sjf.cpp -o sjf.*
* Text

  Description automatically generatedKết quả:

Hình 1.2 \_ Trường hợp không có process nào và trường hợp process đầu tiên đến lúc 0

Text

Description automatically generatedText

Description automatically generatedHình 1.3 \_ Có 5 process, trong đó process đầu tiên arrival != 0, có 2 process arrival time bằng nhau và 2 process có burst time bằng nhau.

Hình 1.4 \_ Trường hợp có thời điểm 1 tiến process kết thúc mà chưa có process nào vào hàng đợi

## Task name 2: Chương trình mô phỏng giải thuật SRTF:

* Lưu đồ thuật toán:

Diagram, schematic

Description automatically generated

Hình 2.1 \_ Lưu đồ giải thuật SRTF

* Source code:

*/\*######################################*

*# University of Information Technology #*

*# IT007 Operating System #*

*# Pham Mai Dung, 19520477 #*

*# File: srtf.cpp #*

*######################################\*/*

*#include <iostream>*

*using namespace std;*

*struct Process*

*{*

*float pn = 0, arr = 0, burst = 0, response = 0, wait = 0, tat = 0, burstTemp = 0, finish = 0;*

*};*

*void swap(float& a, float& b) {*

*float temp;*

*temp = a;*

*a = b;*

*b = temp;*

*}*

*void sortAscendingBurstTime(Process p[], int m, int n) {*

*for (int i = m; i < n; i++) {*

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

*if (p[i].burst > p[j].burst) {*

*swap(p[i].pn, p[j].pn);*

*swap(p[i].arr, p[j].arr);*

*swap(p[i].burst, p[j].burst);*

*swap(p[i].burstTemp, p[j].burstTemp);*

*}*

*}*

*}*

*}*

*void sortAscendingArrivalTime(Process p[], int n)*

*{*

*for (int i = 0; i < n - 1; i++)*

*{*

*for (int j = i + 1; j < n; j++)*

*{*

*if (p[i].arr > p[j].arr)*

*{*

*swap(p[i].pn, p[j].pn);*

*swap(p[i].arr, p[j].arr);*

*swap(p[i].burst, p[j].burst);*

*swap(p[i].burstTemp, p[j].burstTemp);*

*}*

*}*

*}*

*}*

*int main() {*

*Process\* p;*

*int num;*

*cout << "Enter the number of processes: ";*

*cin >> num;*

*p = new Process[num];*

*float totalBurst = 0;*

*for (int i = 0; i < num; i++)*

*{*

*cout << "Enter Process Name: ";*

*cin >> p[i].pn;*

*cout << "Enter Arrival Time: ";*

*cin >> p[i].arr;*

*cout << "Enter Burst Time: ";*

*cin >> p[i].burst;*

*totalBurst += p[i].burst;*

*p[i].burstTemp = p[i].burst;*

*}*

*sortAscendingArrivalTime(p, num);*

*float presentTime = 0;*

*int FirstIndex = -1, LastIndex = -1;*

*while (presentTime != totalBurst + 1)*

*{*

*while (LastIndex < num - 1)*

*{*

*if (p[LastIndex + 1].arr <= presentTime)*

*LastIndex++;*

*else break;*

*}*

*if (FirstIndex >= LastIndex)*

*{*

*presentTime++;*

*totalBurst++;*

*}*

*else*

*{*

*sortAscendingBurstTime(p, FirstIndex + 1, LastIndex);*

*if (p[FirstIndex + 1].burst == 0)*

*{*

*FirstIndex++;*

*p[FirstIndex].tat = presentTime - p[FirstIndex].arr;*

*p[FirstIndex].finish = presentTime;*

*p[FirstIndex].wait = p[FirstIndex].finish - p[FirstIndex].arr - p[FirstIndex].burstTemp;*

*}*

*else*

*{*

*if (p[FirstIndex + 1].burst == p[FirstIndex + 1].burstTemp)*

*p[FirstIndex + 1].response = presentTime - p[FirstIndex + 1].arr;*

*p[FirstIndex + 1].burst--;*

*presentTime++;*

*}*

*}*

*}*

*float totalwt = 0, totaltat = 0;*

*cout << "PName | Arrival | Burst | Response | Waiting | TAT | FinishTime\n";*

*for (int i = 0; i < num; i++)*

*{*

*cout << p[i].pn << "\t\t" << p[i].arr << "\t" << p[i].burstTemp << "\t" << p[i].response << "\t" << p[i].wait << "\t" << p[i].tat << "\t" << p[i].finish << "\n";*

*totalwt += p[i].wait;*

*totaltat += p[i].tat;*

*}*

*cout << "total Waiting Time: " << totalwt << endl;*

*cout << "Average Waiting Time: " << (float)totalwt / num << endl;*

*cout << "total TurnAround Time: " << totaltat << endl;*

*cout << "Average TurnAround Time: " << (float)totaltat / num << endl;*

*return 0;*

*}*

* Giải thích:
* Khai báo một mảng cấu trúc process có n phần tử chứa thông tin của process.
* Viết hàm swap(float& a, float& b) để hoán vị 2 số a, b nhằm phục vụ các hàm sắp xếp.
* Hàm sortAscendingBurstTime() để sắp xếp các process theo thứ tự tăng dần burst time.
* Hàm sortAscendingArrivalTime() để sắp xếp các process theo thứ tự tăng dần arrival time, sắp xếp khi process vào không theo thứ tự.
* Sau khi nhập thông tin processes và sắp xếp theo arrival time, lúc này các phần tử trong mảng đã được gán theo thứ tự sắp xếp.
* Khai báo các biến: totalBurst\_tính tổng burst time của các process; burstTemp\_lưu giá trị của burst trong quá trình thực thi sẽ bị thay đổi (burst--); FirstIndex và LastIndex\_ lưu vị trí đầu cuối hàng đợi; presentTime\_ thời gian hoạt động, khi ==totalBurst là các process được thực thi hết.
* Khi presentTime != totalBurst, cập nhật LastIndex == vị trí cuối hàng đợi, sắp xếp mảng process tăng dần từ FirstIndex+1 đến LastIndex. Nếu FI >= LI\_ không có process nào trong hàng đợi, cập nhật totalBurst và presentTime tăng đến khi có process vào hàng đợi. Nếu có process trong hàng đợi, hàm sắp xếp tăng dần bursttime và thực thi process có burst nhỏ nhất (cập nhật các giá trị response, tat, waiting,..) và cập nhật tiếp thời điểm tiếp theo (burst process đang chạy sẽ giảm dần, presentTime tăng dần) và quay lại vòng lặp.
* Tính waiting và turn around time trung bình, xuất ra các thông tin theo yêu cầu.
* Thực thi chương trình bằng lệnh *g++ srtf.cpp -o srtf.*
* Kết quả:

Text

Description automatically generatedHình 2.2\_ Trường hợp không có process nào và trường hợp process đầu tiên đến lúc 0.

Text

Description automatically generated

Hình 2.3 \_ Kết quả mô phỏng SRTF với 5 processes

## Task name 3: Chương trình mô phỏng giải thuật RR:

* Lưu đồ giải thuật:

Diagram

Description automatically generated

Hình 3.1 \_ Lưu đồ giải thuật Round Robin.

* Source code:

*/\*######################################*

*# University of Information Technology #*

*# IT007 Operating System #*

*# Pham Mai Dung, 19520477 #*

*# File: roundRobin.cpp #*

*######################################\*/*

*#include <iostream>*

*using namespace std;*

*struct Process*

*{*

*float pn = 0, burst = 0, response = 0, wait = 0, tat = 0, burstTemp = 0, finish = 0;*

*};*

*int main()*

*{*

*Process\* p;*

*int num;*

*float qTime;*

*cout << "Nhap so tien trinh: ";*

*cin >> num;*

*p = new Process[num];*

*float totalBurst = 0;*

*for (int i = 0; i < num; i++)*

*{*

*cout << "Process Name: ";*

*cin >> p[i].pn;*

*cout << "Burst Time: ";*

*cin >> p[i].burst;*

*totalBurst += p[i].burst;*

*p[i].burstTemp = p[i].burst;*

*}*

*cout << "Nhap Quantum Time: ";*

*cin >> qTime;*

*cout << "PName" << "\t" << "Time" << "\t" << "Status" << endl;*

*float presentTime = 0;*

*int index = 0;*

*while (presentTime != totalBurst)*

*{*

*if (p[index].burst == 0)*

*{*

*index++;*

*if (index == num)*

*index = 0;*

*}*

*else*

*{*

*if (p[index].burst > qTime)*

*{*

*if (p[index].burst == p[index].burstTemp)*

*{*

*p[index].response = presentTime;*

*}*

*cout << p[index].pn << "\t" << presentTime << "\t" << " Start \n";*

*presentTime = presentTime + qTime;*

*cout << p[index].pn << "\t" << presentTime << "\t" << " Stop \n";*

*p[index].burst = p[index].burst - qTime;*

*index++;*

*if (index == num)*

*index = 0;*

*}*

*else*

*{*

*if (p[index].burst == p[index].burstTemp)*

*{*

*p[index].response = presentTime;*

*}*

*cout << p[index].pn << "\t" << presentTime << "\t" << " Start \n";*

*presentTime = presentTime + p[index].burst;*

*cout << p[index].pn << "\t" << presentTime << "\t" << " Stop \n";*

*p[index].burst = 0;*

*p[index].finish = presentTime;*

*p[index].wait = p[index].finish - p[index].burstTemp;*

*p[index].tat = p[index].finish - p[index].response;*

*index++;*

*if (index == num) index = 0;*

*}*

*}*

*}*

*float totalwt = 0, totaltat = 0;*

*for (int i = 0; i < num; i++)*

*{*

*totaltat += p[i].tat;*

*totalwt += p[i].wait;*

*}*

*cout << "total Waiting Time: " << totalwt << endl;*

*cout << "Average Waiting Time: " << (float)totalwt / num << endl;*

*cout << "total TurnAround Time: " << totaltat << endl;*

*cout << "Average TurnAround Time: " << (float)totaltat / num << endl;*

*return 0;*

*}*

* Giải thích:
* Khai báo một mảng cấu trúc process có n phần tử chứa thông tin của process.
* Nhập thông tin các process, tính totalBurst, cập nhật burstTemp, nhập quantumTime.
* Nếu p[i].burst != 0, burst > quantumTime thì presentTime = presentTime + Quantum và burst = burst – quantumTime, ngược lại thì Time = Time + Burst, Burst = 0 => xuất ra trạng thái của p[i] tại thời điểm presnentTime.
* Index tăng dần đến num và lặp lại đến khi các process được thực thi hết.
* Tính waiting và turn around time trung bình, xuất ra các thông tin.
* Thực thi chương trình bằng lệnh *g++ srtf.cpp -o srtf.*
* Text

  Description automatically generatedKết quả:

Hình 3.2 \_ Nhập các process, quantumTime

Text

Description automatically generated

Hình 3.3 \_ Kết quả mô phỏng round robin theo các dữ liệu vừa nhập trên