

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH



BÁO CÁO LAB 4
MÔN THỰC HÀNH HỆ ĐIỀU HÀNH – IT007

HỌ VÀ TÊN: NGUYỄN NAM HẢI

MSSV: 20520171

LỚP: IT007.M11.MTCL.1

GIẢNG VIÊN HƯỚNG DẪN:

NGUYỄN DUY XUÂN BÁCH

TP. HỒ CHÍ MINH – Tháng 11 năm 2021

MỤC LỤC

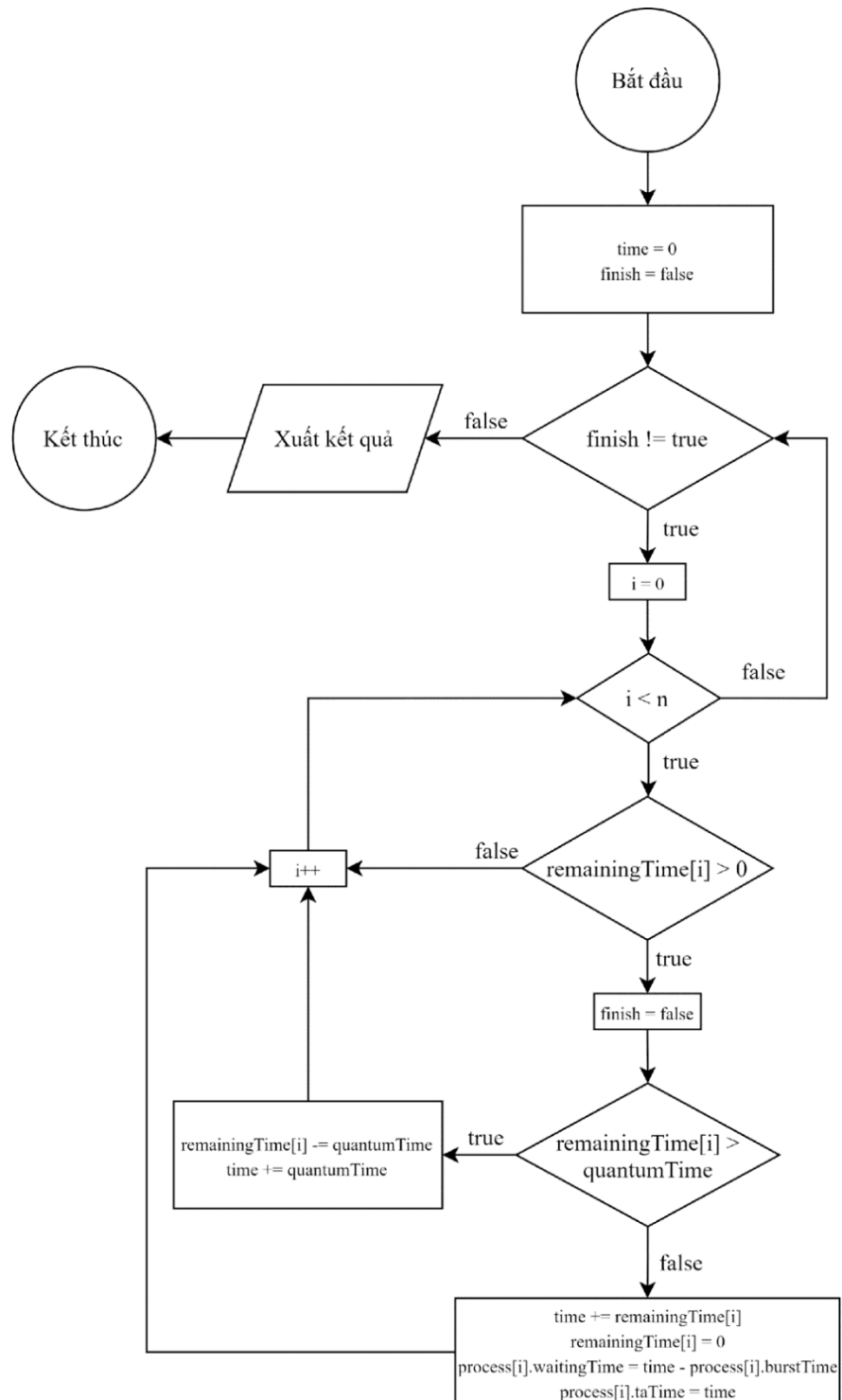
4.3 Chuẩn bị	3
4.3.5 Câu hỏi chuẩn bị	3
4.4 Thực hành	7
4.5 Bài tập	9

4.3 Chuẩn bị

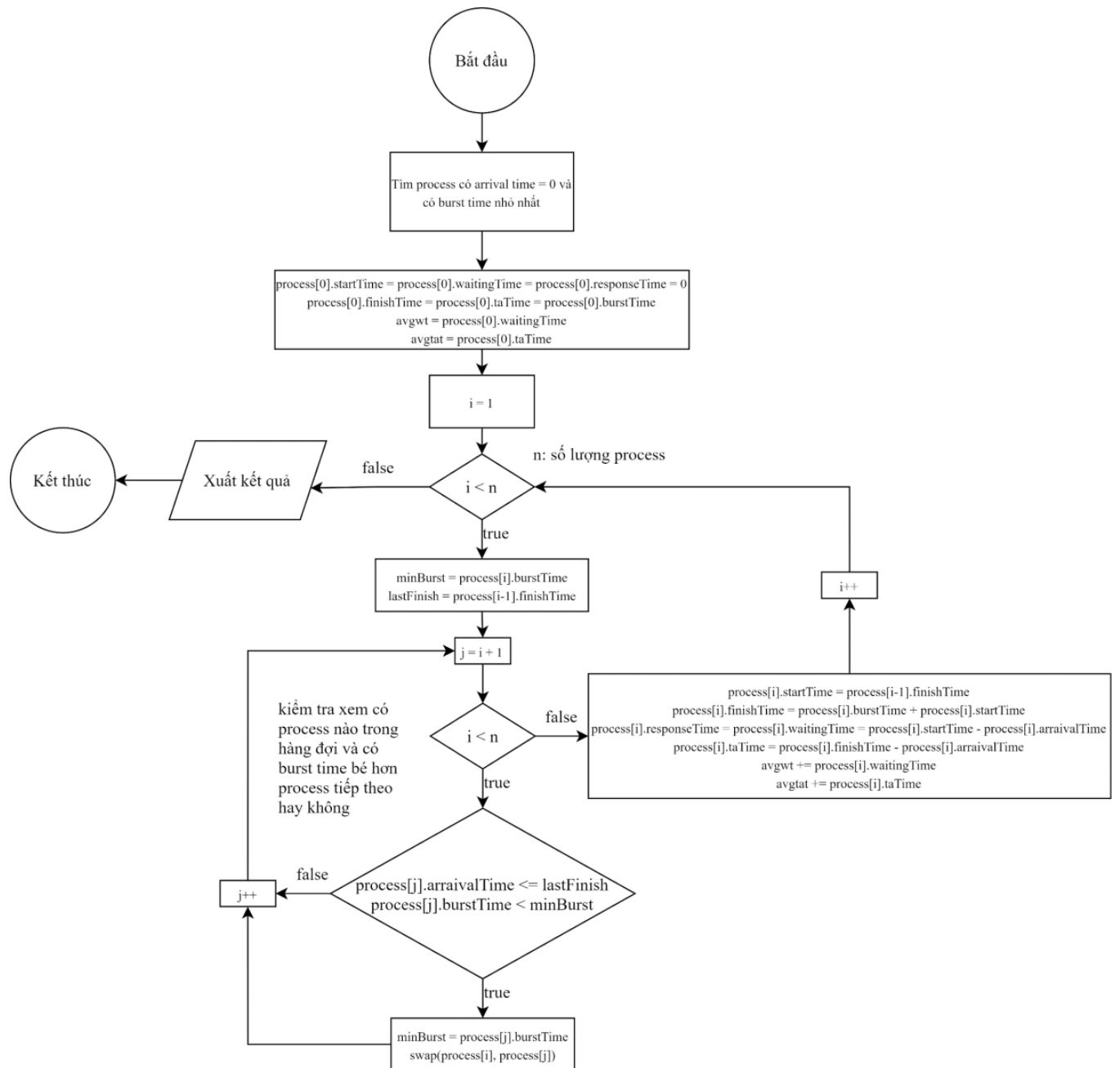
4.3.5 Câu hỏi chuẩn bị

1. Vẽ sơ đồ giải thuật của các giải thuật lập lịch tiến trình:

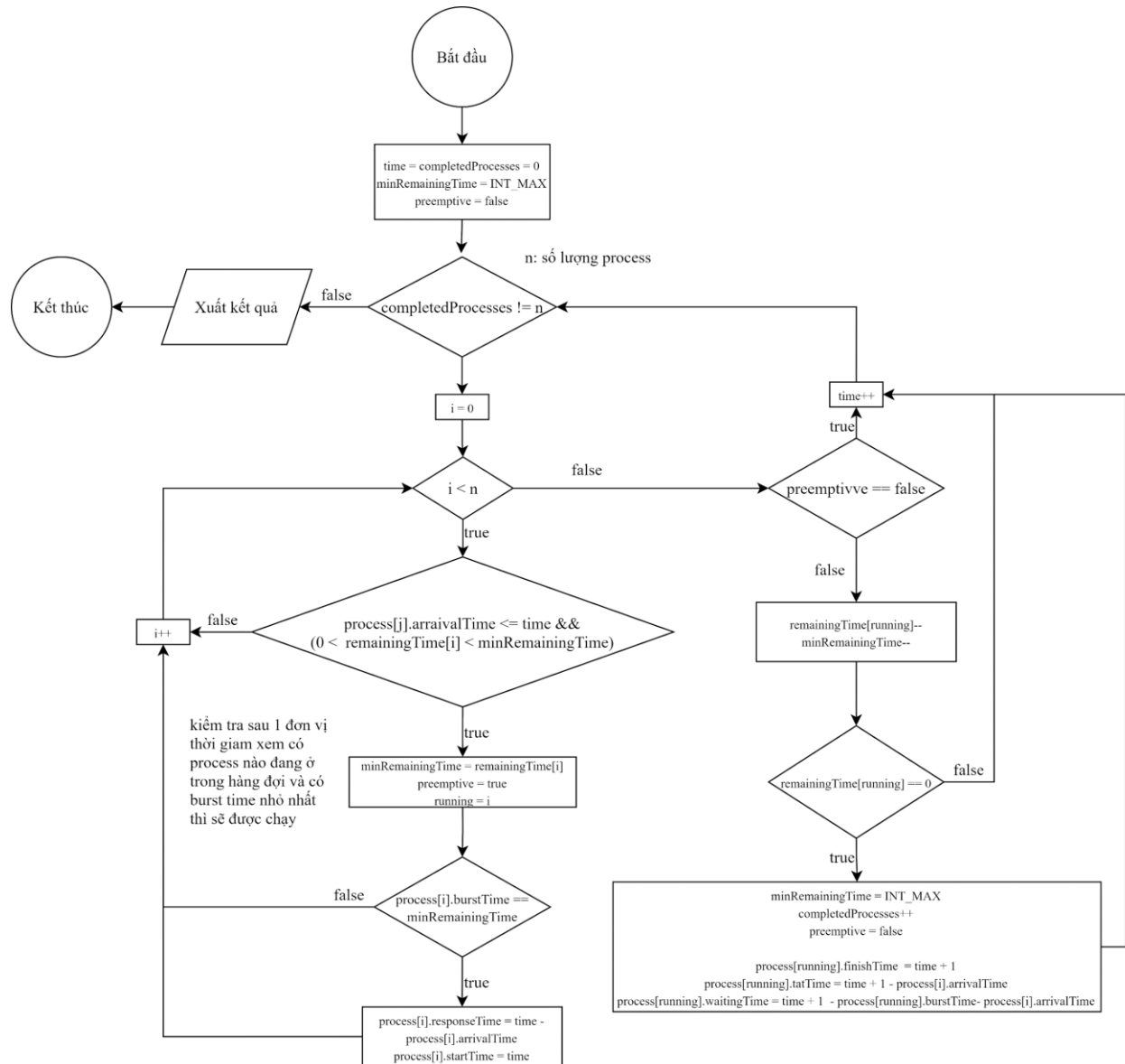
- Lưu đồ thuật toán RR ():



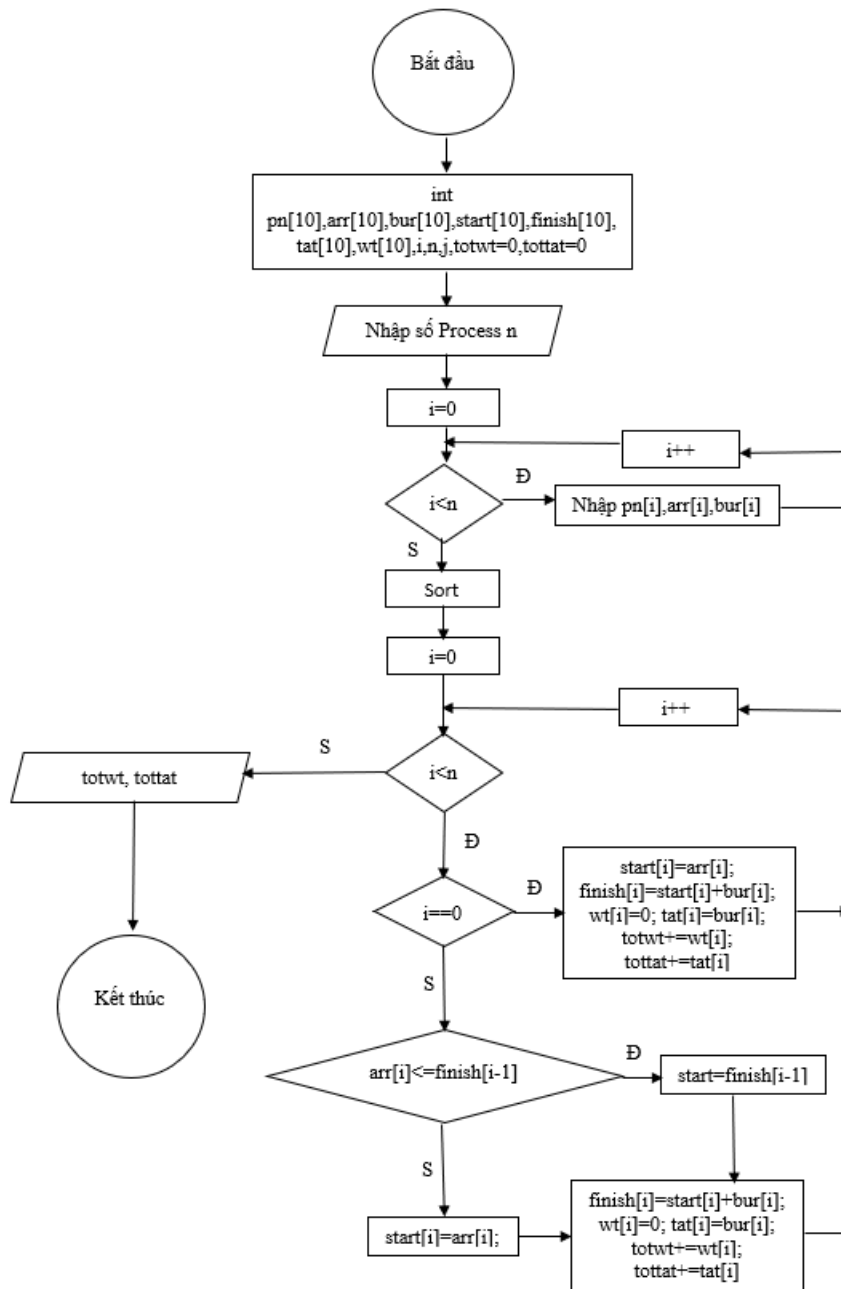
- Lưu đồ thuật toán SJF:



- Lưu đồ thuật toán SRT:



-Lưu đồ thuật toán FCFS:



2. Giải thích các thuật ngữ sau:

TT	Thuật ngữ	Mô Tả
1	Arrival time	Thời gian tới
2	Burst time	Thời gian để hoàn thành
3	Quantum time (timeslice)	<ul style="list-style-type: none">- Mỗi process nhận được một đơn vị nhỏ thời gian CPU, thông thường từ 10-100 msec để thực thi –- Sau khoảng thời gian đó, process bị đoạt quyền và trở về cuối hàng đợi ready –- Nếu có n process trong hàng đợi ready và quantum time = q thì không có process nào phải chờ đợi quá $(n - 1)q$ đơn vị thời gian
4	Response time	Thời gian đáp ứng: <ul style="list-style-type: none">- Thời gian từ lúc có yêu cầu của người dùng (user request) đến khi có đáp ứng đầu tiên- Thường là vấn đề với các I/O-bound process
5	Waiting time	Thời gian chờ: <ul style="list-style-type: none">- Thời gian một process ở trong hàng đợi ready
6	Turnaround time	Thời gian quay vòng
7	Average waiting time	Thời gian đợi trung bình
8	Average turn-around time	Thời gian quay vòng trung bình

4.4 Thực hành

fcfs.c:

The screenshot shows a VMware Workstation interface with a Ubuntu 64-bit virtual machine. The terminal window is open, displaying a C program for process scheduling. The program includes headers, defines arrays for process details, and implements a main function with nested loops for scheduling logic. The output shows process details for 10 processes.

```
#include<stdio.h>

void main(){
    int pn[10];
    int arr[10], bur[10], star[10], finish[10], tat[10], wt[10], i, n;
    int totwt=0, tottat=0;

    printf("Enter the number of processes:");
    scanf("%d",&n);

    for(i=0;&n;i++) {
        printf("Enter the Process Name, Arrival Time & Burst Time:");
        scanf("%s%d%d",&pn[i],&arr[i],&bur[i]);
    }

    for(i=0;&n;i++) {
        if(i==0) {
            star[i]=arr[i];
            wt[i]=star[i]-arr[i];
            finish[i]=star[i]+bur[i];
            tat[i]=finish[i]-arr[i];
        } else{
            star[i]=finish[i-1];
            wt[i]=star[i]-arr[i];
            finish[i]=star[i]+bur[i];
            tat[i]=finish[i]-arr[i];
        }
    }

    printf("\nPidName Arrtime Burtine Start TAT Finish");
    for(i=0;&n;i++) {
        printf("\n%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t",pn[i],arr[i],bur[i],star[i],tat[i],finish[i],
        totwt=wt[i];
        tottat=tat[i];
    }
}
```

The output of the program shows the following details for 10 processes:

```

PidName Arrtime Burtine Start TAT Finish
0 0 1 0 0 1
1 1 4 1 0 5
2 2 1 5 3 6
3 3 1 6 3 7
4 4 1 7 3 8
5 5 1 8 3 9
6 6 1 9 3 10
7 7 1 10 3 11
8 8 1 11 3 12
9 9 1 12 3 13

```

KQ:

```
hainn_vit@hainn:~/Desktop/Lab/Lab4$ gcc fcfs.c -o fcfs
hainn_vit@hainn:~/Desktop/Lab/Lab4$ ./fcfs
Enter the number of processes:3
Enter the Process Name, Arrival Time & Burst Time:0 0 2
Enter the Process Name, Arrival Time & Burst Time:1 1 3
Enter the Process Name, Arrival Time & Burst Time:2 2 4

PName Arrrtime Burtime Start TAT Finish
0          0          2          0          2          2
1          1          3          2          4          5
2          2          4          5          7          9
```

Bổ sung code:

```
float avewt, avetat;
avewt = (totwt*1.0)/n;
avetat = (tottat)/n;
printf("\nAverage waiting time: %0.3f", avewt);
printf("\nAverage turnaround: %0.3f", avetat);
```

Biên dịch và KQ:

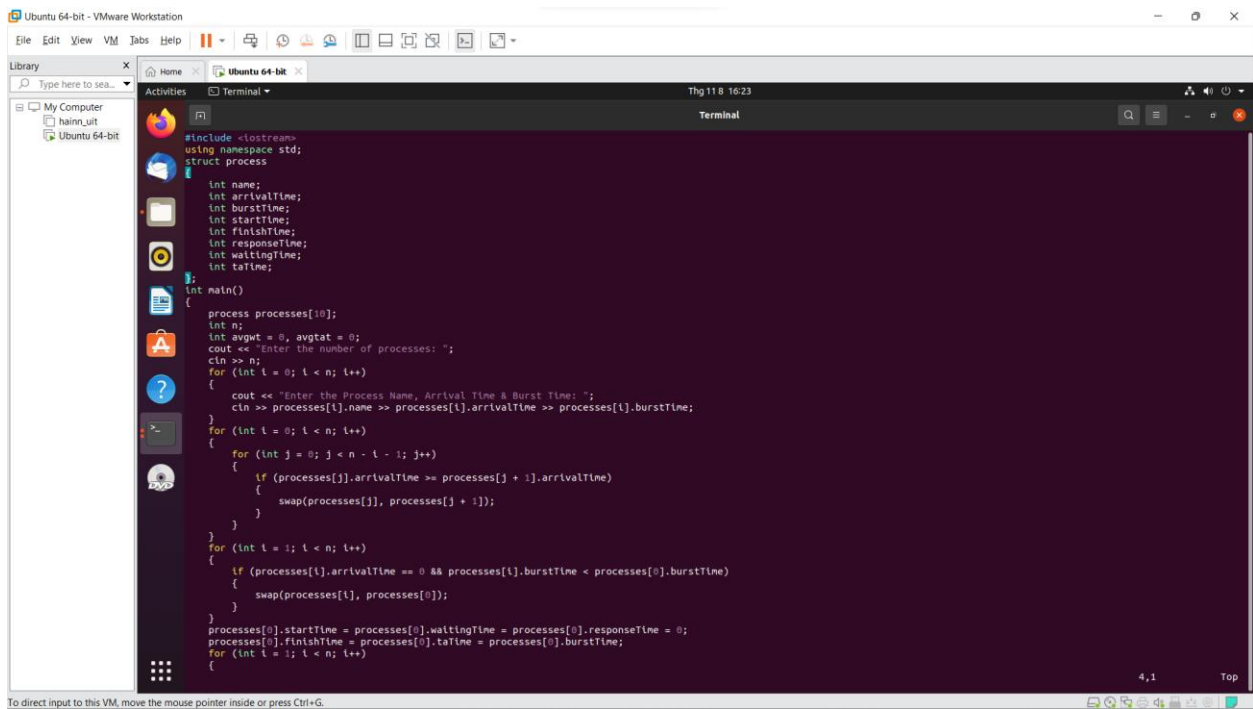

```
hainn_uit@hainn:~/Desktop/Lab/Lab4$ gcc fcfs.c -o fcfs
hainn_uit@hainn:~/Desktop/Lab/Lab4$ ./fcfs
Enter the number of processes:3
Enter the Process Name, Arrival Time & Burst Time:0 0 2
Enter the Process Name, Arrival Time & Burst Time:1 1 3
Enter the Process Name, Arrival Time & Burst Time:2 2 4

PName Arrrtime Burtime Start TAT Finish
0      0          2      2      0      2
1      1          3      3      2      5
2      2          4      4      5      9

Average waiting time: 1.333
Average turnaround: 4.000
```

4.5 Bài tập

- Viết chương trình mô phỏng giải thuật SJF với các yêu cầu sau:
 - Code:



```
#include <iostream>
using namespace std;
struct process
{
    int name;
    int arrivalTime;
    int burstTime;
    int startTime;
    int finishTime;
    int responseTime;
    int waitingTime;
    int tatTime;
};

int main()
{
    process processes[10];
    int n;
    int avgwt = 0, avgtat = 0;
    cout << "Enter the number of processes: ";
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cout << "Enter the Process Name, Arrival Time & Burst Time: ";
        cin >> processes[i].name >> processes[i].arrivalTime >> processes[i].burstTime;
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            if (processes[j].arrivalTime >= processes[j + 1].arrivalTime)
            {
                swap(processes[j], processes[j + 1]);
            }
        }
    }
    for (int i = 1; i < n; i++)
    {
        if (processes[i].arrivalTime == 0 && processes[i].burstTime < processes[0].burstTime)
        {
            swap(processes[i], processes[0]);
        }
    }
    processes[0].startTime = processes[0].waitingTime = processes[0].responseTime = 0;
    processes[0].finishTime = processes[0].tatTime = processes[0].burstTime;
    for (int i = 1; i < n; i++)
    {
        processes[i].waitingTime = processes[i-1].tatTime;
        processes[i].startTime = processes[i-1].finishTime;
        processes[i].finishTime = processes[i].startTime + processes[i].burstTime;
        processes[i].tatTime = processes[i].finishTime - processes[i].arrivalTime;
        avgwt += processes[i].waitingTime;
        avgtat += processes[i].tatTime;
    }
    cout << "Average waiting time: " << avgwt / (n - 1) << endl;
    cout << "Average turnaround: " << avgtat / (n - 1) << endl;
}
```

```

if (processes[i].arrivalTime == 0 && processes[i].burstTime < processes[0].burstTime)
{
    swap(processes[i], processes[0]);
}
processes[0].startTime = processes[0].waitingTime = processes[0].responseTime = 0;
processes[0].finishTime = processes[0].taTime = processes[0].burstTime;
for (int i = 1; i < n; i++)
{
    int minBurst = processes[i].burstTime;
    int lastFinish = processes[i - 1].finishTime;
    for (int j = i + 1; j < n; j++)
    {
        if (processes[j].arrivalTime <= lastFinish && processes[j].burstTime < minBurst)
        {
            minBurst = processes[j].burstTime;
            swap(processes[i], processes[j]);
        }
    }
    processes[i].startTime = processes[i - 1].finishTime;
    processes[i].finishTime = processes[i].burstTime + processes[i].startTime;
    processes[i].waitingTime = processes[i].startTime - processes[i].arrivalTime;
    processes[i].responseTime = processes[i].waitingTime;
    processes[i].taTime = processes[i].finishTime - processes[i].arrivalTime;
}
cout << "PName\tArrTime\tBurstTime\tStartTime\tFinsh\tResponse Time\tWaiting Time\tTurnaround Time\n";
for (int i = 0; i < n; i++)
{
    cout << processes[i].name << "\t" << processes[i].arrivalTime << "\t" << processes[i].burstTime << "\t\t" << processes[i].startTime << "\t" << processes[i].finishTime << "\t" << processes[i].responseTime << "\t\t" << processes[i].waitingTime << "\t\t" << processes[i].taTime << "\n";
    avgwt += processes[i].waitingTime;
    avgtat += processes[i].taTime;
}
cout << "Average waiting time: " << (float)avgwt / (float)n << endl;
cout << "Average turnaround time: " << (float)avgtat / (float)n << endl;
return 0;

```

- Demo:

```

hainn_uit@hainn: ~/Desktop/Lab/Lab4
hainn_uit@hainn:~/Desktop/Lab/Lab4$ ./sjf
Enter the number of processes: 5
Enter the Process Name, Arrival Time & Burst Time: 1 0 12
Enter the Process Name, Arrival Time & Burst Time: 2 2 7
Enter the Process Name, Arrival Time & Burst Time: 3 5 8
Enter the Process Name, Arrival Time & Burst Time: 4 9 3
Enter the Process Name, Arrival Time & Burst Time: 5 12 6
PName  ArrTime BurstTime      Start   Finsh   Response Time   Waiting Time
Turnaround Time
1      0      12      0       12      0              0
4      9      3       12      15      3              3
5      12     6       15      21      3              3
2      2      7       21      28      19             19
3      5      8       28      36      23             23
Average waiting time: 9.6
Average turnaround time: 16.8
hainn_uit@hainn:~/Desktop/Lab/Lab4$

```

2. Viết chương trình mô phỏng giải thuật SRT với các yêu cầu sau:

- Code:

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help

Library

Home x Ubuntu 64-bit x

Activities Terminal

Thg 11 8 16:24

```
#include <iostream>
using namespace std;
struct process
{
    int name;
    int arrivalTime;
    int burstTime;
    int startTime;
    int finishTime;
    int responseTime;
    int waitingTime;
    int tailTime;
};
int main()
{
    process processes[10];
    int remainingTime[10];
    int n;
    int avgwt = 0, avgtat = 0;
    cout << "Enter the number of processes: ";
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cout << "Enter the Process Name, Arrival Time & Burst Time: ";
        cin >> processes[i].name >> processes[i].arrivalTime >> processes[i].burstTime;
        remainingTime[i] = processes[i].burstTime;
    }
    int time = 0, completedProcesses = 0;
    int minRemainingTime = INT_MAX;
    int running;
    bool preemptive = false;
    while (completedProcesses != n)
    {
        for (int i = 0; i < n; i++)
        {
            if (processes[i].arrivalTime <= time &&
                (0 < remainingTime[i] && remainingTime[i] < minRemainingTime))
            {
                minRemainingTime = remainingTime[i];
                preemptive = true;
                running = i;
                if (processes[i].burstTime == minRemainingTime)
                {
                    processes[i].responseTime = time - processes[i].arrivalTime;
                    processes[i].startTime = time;
                }
            }
        }
        if (!preemptive)
        {
            time++;
            continue;
        }
        minRemainingTime = --remainingTime[running];
        if (remainingTime[running] == 0)
        {
            minRemainingTime = INT_MAX;
            completedProcesses++;
            preemptive = false;
            processes[running].finishTime = time + 1;
            processes[running].tailTime = processes[running].finishTime - processes[running].arrivalTime;
            processes[running].waitingTime = processes[running].finishTime -
                processes[running].burstTime - processes[running].arrivalTime;
        }
        time++;
    }
    cout << "PName\tArrTime\tBurstTime\tStartTime\tFinsh\tResponse Time\tWaiting Time\tTurnaround Time\n";
    for (int i = 0; i < n; i++)
    {
        cout << processes[i].name << "\t" << processes[i].arrivalTime << "\t" << processes[i].burstTime
            << "\t" << processes[i].startTime << "\t" << processes[i].finishTime << "\t" << processes[i].responseTime << "\t" << processes[i].waitingTime << "\t" << processes
            [i].tailTime << "\n";
        avgwt += processes[i].waitingTime;
        avgtat += processes[i].tailTime;
    }
    cout << "Average waiting time: " << (float)avgwt / (float)n << endl;
    cout << "Average turnaround time: " << (float)avgtat / (float)n << endl;
    return 0;
}
```

1,1 Top

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help

Library

Home x Ubuntu 64-bit x

Activities Terminal

Thg 11 8 16:24

```
if (processes[i].arrivalTime <= time &&
    (0 < remainingTime[i] && remainingTime[i] < minRemainingTime))
{
    minRemainingTime = remainingTime[i];
    preemptive = true;
    running = i;
    if (processes[i].burstTime == minRemainingTime)
    {
        processes[i].responseTime = time - processes[i].arrivalTime;
        processes[i].startTime = time;
    }
}
if (!preemptive)
{
    time++;
    continue;
}
minRemainingTime = --remainingTime[running];
if (remainingTime[running] == 0)
{
    minRemainingTime = INT_MAX;
    completedProcesses++;
    preemptive = false;
    processes[running].finishTime = time + 1;
    processes[running].tailTime = processes[running].finishTime - processes[running].arrivalTime;
    processes[running].waitingTime = processes[running].finishTime -
        processes[running].burstTime - processes[running].arrivalTime;
}
time++;
}
cout << "PName\tArrTime\tBurstTime\tStartTime\tFinsh\tResponse Time\tWaiting Time\tTurnaround Time\n";
for (int i = 0; i < n; i++)
{
    cout << processes[i].name << "\t" << processes[i].arrivalTime << "\t" << processes[i].burstTime
        << "\t" << processes[i].startTime << "\t" << processes[i].finishTime << "\t" << processes[i].responseTime << "\t" << processes[i].waitingTime << "\t" << processes
        [i].tailTime << "\n";
    avgwt += processes[i].waitingTime;
    avgtat += processes[i].tailTime;
}
cout << "Average waiting time: " << (float)avgwt / (float)n << endl;
cout << "Average turnaround time: " << (float)avgtat / (float)n << endl;
return 0;
}
```

88,0-1 Bot

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

- Demo:

```
hainn_uit@hainn: ~/Desktop/Lab/Lab4
hainn_uit@hainn:~/Desktop/Lab/Lab4$ g++ srt.cpp -o srt
hainn_uit@hainn:~/Desktop/Lab/Lab4$ ./srt
Enter the number of processes: 5
Enter the Process Name, Arrival Time & Burst Time: 1 0 12
Enter the Process Name, Arrival Time & Burst Time: 2 2 7
Enter the Process Name, Arrival Time & Burst Time: 3 5 8
Enter the Process Name, Arrival Time & Burst Time: 4 9 3
Enter the Process Name, Arrival Time & Burst Time: 5 12 6
PName ArrTime BurstTime Start Finish Response Time Waiting Time Turnaround Time
1 0 12 0 36 0 24 36
2 2 7 2 9 0 0 7
3 5 8 18 26 13 13 21
4 9 3 9 12 0 0 3
5 12 6 12 18 0 0 6
Average waiting time: 7.4
Average turnaround time: 14.6
hainn_uit@hainn:~/Desktop/Lab/Lab4$
```

- Viết chương trình mô phỏng giải thuật RR với các yêu cầu sau (giả sử tất cả các tiến trình đều có arrival time là 0):

- Code:

```
Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
hainn_uit
Ubuntu 64-bit
Activities
Terminal
Thg 11 8 16:31
#include <iostream>
using namespace std;
struct process
{
    int name;
    int burstTime;
    int waitingTime;
    int taTime;
};
int main()
{
    process processes[10];
    int remainingTime[10];
    int n, quantumTime;
    int avgwt = 0, avgtat = 0;
    cout << "Enter the number of processes: ";
    cin >> n;
    cout << "Enter the quantum time: ";
    cin >> quantumTime;
    for (int i = 0; i < n; i++)
    {
        cout << "Enter the Process Name, Burst Time: ";
        cin >> processes[i].name >> processes[i].burstTime;
        remainingTime[i] = processes[i].burstTime;
    }
    int time = 0;
    bool finish = false;
    while (!finish)
    {
        finish = true;
        for (int i = 0; i < n; i++)
        {
            if (remainingTime[i] > 0)
            {
                finish = false;
                if (remainingTime[i] > quantumTime)
                {
                    remainingTime[i] -= quantumTime;
                    time += quantumTime;
                }
                else
                {
                    time += remainingTime[i];
                    remainingTime[i] = 0;
                    processes[i].waitingTime = time - processes[i].burstTime;
                    processes[i].taTime = time;
                }
            }
        }
    }
}
```

```
cin >> quantumTime;
for (int i = 0; i < n; i++)
{
    cout << "Enter the Process Name, Burst Time: ";
    cin >> processes[i].name >> processes[i].burstTime;
    remainingTime[i] = processes[i].burstTime;
}

int time = 0;
bool finish = false;
while (!finish)
{
    finish = true;
    for (int i = 0; i < n; i++)
    {
        if (remainingTime[i] > 0)
        {
            finish = false;
            if (remainingTime[i] > quantumTime)
            {
                remainingTime[i] -= quantumTime;
                time += quantumTime;
            }
            else
            {
                time += remainingTime[i];
                remainingTime[i] = 0;
                processes[i].waitingTime = time - processes[i].burstTime;
                processes[i].turnaroundTime = time;
            }
        }
    }
}

cout << "PName\tBurstTime\tWaiting Time\tTurnaround Time\n";
for (int i = 0; i < n; i++)
{
    cout << processes[i].name << "\t" << processes[i].burstTime << "\t\t"
    << processes[i].waitingTime << "\t\t" << processes[i].turnaroundTime << "\n";
    avgwt += processes[i].waitingTime;
    avgtat += processes[i].turnaroundTime;
}

cout << "Average waiting time: " << (float)avgwt / (float)n << endl;
cout << "Average turnaround time: " << (float)avgtat / (float)n << endl;
return 0;
```

- Demo:

```
hainn_uit@hainn: ~/Desktop/Lab/Lab4
hainn_uit@hainn:~/Desktop/Lab/Lab4$ g++ rr.cpp -o rr
hainn_uit@hainn:~/Desktop/Lab/Lab4$ ./rr
Enter the number of processes: 5
Enter the quantum time: 4
Enter the Process Name, Burst Time: 1 12
Enter the Process Name, Burst Time: 2 7
Enter the Process Name, Burst Time: 3 8
Enter the Process Name, Burst Time: 4 3
Enter the Process Name, Burst Time: 5 6
PName    BurstTime    Waiting Time    Turnaround Time
1        12          24             36
2         7          19             26
3         8          22             30
4         3          12             15
5         6          26             32
Average waiting time: 20.6
Average turnaround time: 27.8
hainn_uit@hainn:~/Desktop/Lab/Lab4$
```