



ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÁO CÁO

MÔN HỆ ĐIỀU HÀNH

| Chủ đề |

Process Scheduling

| Sinh viên |

Phạm Hoàng Nam Anh.....18120278

Thành phố Hồ Chí Minh – Tháng 12 năm 2021

MỤC LỤC

MỤC LỤC	2
GIỚI THIỆU	3
HƯỚNG DẪN SỬ DỤNG	5
Menu.....	5
First come, first served	6
Round Robin	7
Shortest job first	8
Tất cả thuật toán	9
CẤU TRÚC CHƯƠNG TRÌNH	10
Cấu trúc thư mục	10
Các phương thức dùng chung.....	11
First come, first served	12
Round Robin	13
Shortest job first	14
TÀI LIỆU THAM KHẢO	16

GIỚI THIỆU

Chương trình nhận file văn bản **Input.txt** chứa thông tin của các tiến trình cần điều phối (tập tin có định dạng:

Dòng đầu tiên lưu số lượng tiến trình và thời gian quantum của chiến lược Round Robin.

Mỗi dòng còn lại đều có cấu trúc: <Tên tiến trình> <thời điểm vào> <thời gian xử lý> [<độ ưu tiên>]).

Xác định kết quả điều phối CPU và các thông số tương ứng theo các chiến lược đã được trình bày (FCFS, RR, SJF,...), với kết quả ở mỗi chiến lược được lưu trên 01 tập tin văn bản có tên tương ứng.

Ví dụ về Input.txt:

Process	Arrival Time	CPU Burst	Priority	Quantum=4
P1	0	24	3	Thời gian xử lý của modul điều phối rất nhỏ và xem như là 0 (tức không xét), các tiến trình này cũng không rơi vào trạng thái Blocked
P2	1	5	2	
P3	2	3	1	

Ví dụ về FCFS.txt:

```

FCFS.txt - Notepad
File Edit Format View Help
0~P1~24~P2~29~P3~32
P1      TT = 24 WT = 0
P2      TT = 28 WT = 23
P3      TT = 30 WT = 27
Average:      TT = 27.3333      WT = 16.6667

```

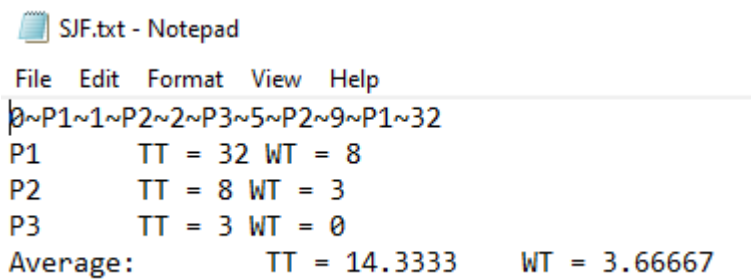
Ví dụ về RR.txt:

```

RR.txt - Notepad
File Edit Format View Help
0~P1~4~P2~8~P3~11~P1~15~P2~16~P1~32
P1      TT = 32 WT = 8
P2      TT = 15 WT = 10
P3      TT = 9  WT = 6
Average:      TT = 18.6667      WT = 8

```

Ví dụ về SJF.txt:

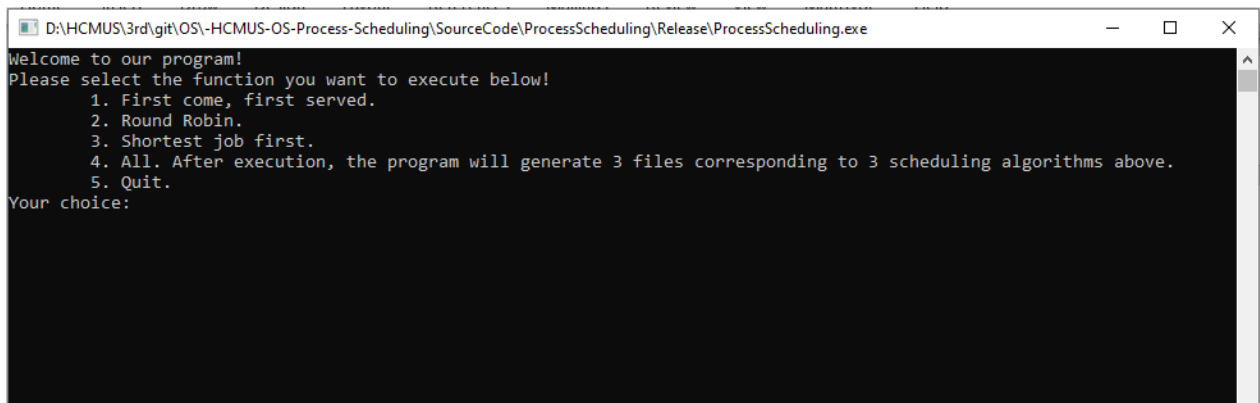


```
SJF.txt - Notepad
File Edit Format View Help
0~P1~1~P2~2~P3~5~P2~9~P1~32
P1      TT = 32 WT = 8
P2      TT = 8  WT = 3
P3      TT = 3  WT = 0
Average:      TT = 14.3333  WT = 3.66667
```

HƯỚNG DẪN SỬ DỤNG

Chạy file ProcessScheduling.exe theo đường dẫn SourceCode/ProcessScheduling/Release/ProcessScheduling.exe để mở chương trình.

Menu



```

D:\HCMUS\3rd\git\OS\HCMUS-OS-Process-Scheduling\SourceCode\ProcessScheduling\Release\ProcessScheduling.exe
Welcome to our program!
Please select the function you want to execute below!
1. First come, first served.
2. Round Robin.
3. Shortest job first.
4. All. After execution, the program will generate 3 files corresponding to 3 scheduling algorithms above.
5. Quit.
Your choice:

```

Đầu tiên vào menu chương trình như hình trên.

Chọn thuật toán điều phối tiến trình mong muốn:

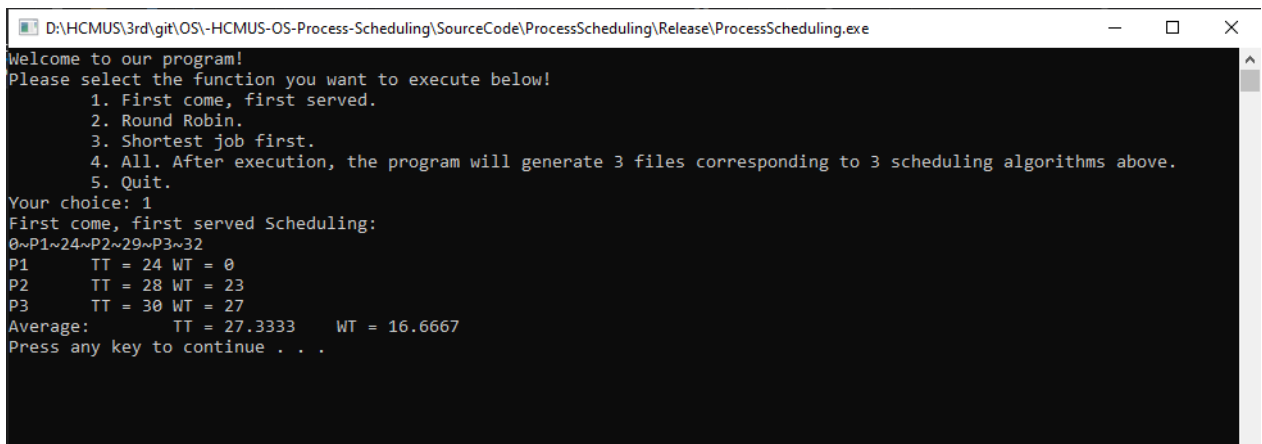
1. First come, first served.
2. Round Robin.
3. Shortest job first.
4. Thực hiện tất cả các thuật toán và in ra kết quả cùng một lúc.
5. Thoát chương trình.

First come, first served

Chọn chức năng 1.

Chương trình sẽ tiến hành tải lên tập tin Input.txt. Vui lòng đảm bảo rằng tập tin Input.txt nằm cùng thư mục với tập tin thực thi chương trình ProcessScheduling.exe.

Sau khi tải lên thành công tập tin Input.txt, chương trình sẽ thực hiện điều phối tiến trình theo thuật toán First come, first served và in ra kết quả thu được như hình dưới.



```
D:\HCMUS\3rd\git\OS\HCMUS-OS-Process-Scheduling\SourceCode\ProcessScheduling\Release\ProcessScheduling.exe
Welcome to our program!
Please select the function you want to execute below!
1. First come, first served.
2. Round Robin.
3. Shortest job first.
4. All. After execution, the program will generate 3 files corresponding to 3 scheduling algorithms above.
5. Quit.
Your choice: 1
First come, first served Scheduling:
0~P1~24~P2~29~P3~32
P1    TT = 24 WT = 0
P2    TT = 28 WT = 23
P3    TT = 30 WT = 27
Average:    TT = 27.3333    WT = 16.6667
Press any key to continue . . .
```

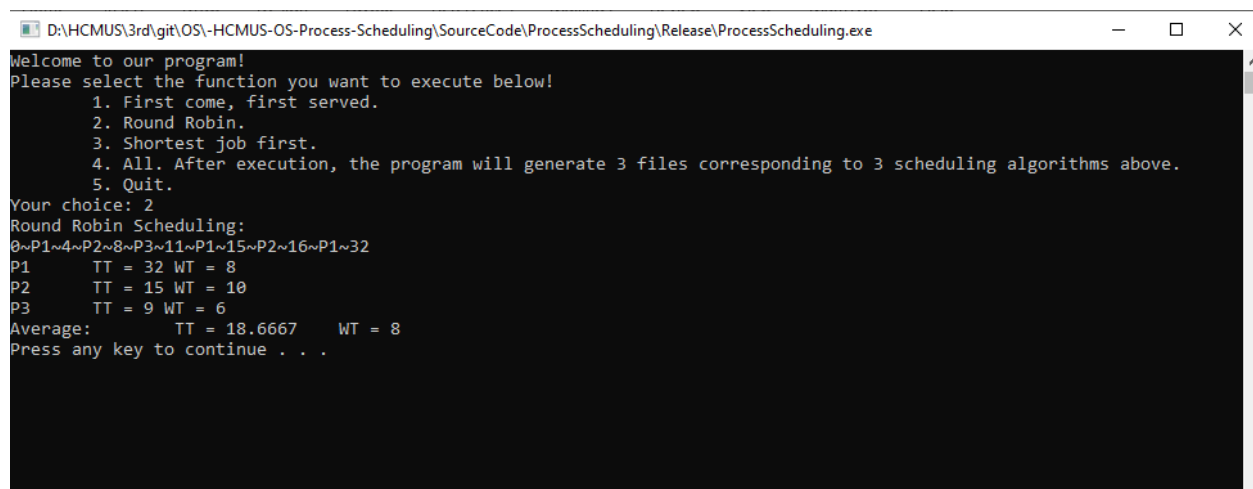
Ấn phím bất kỳ để quay về menu chính.

Round Robin

Chọn chức năng 2.

Chương trình sẽ tiến hành tải lên tập tin Input.txt. Vui lòng đảm bảo rằng tập tin Input.txt nằm cùng thư mục với tập tin thực thi chương trình ProcessScheduling.exe.

Sau khi tải lên thành công tập tin Input.txt, chương trình sẽ thực hiện điều phối tiến trình theo thuật toán Round Robin và in ra kết quả thu được như hình dưới.



```
D:\HCMUS\3rd\git\OS\HCMUS-OS-Process-Scheduling\SourceCode\ProcessScheduling\Release\ProcessScheduling.exe
Welcome to our program!
Please select the function you want to execute below!
1. First come, first served.
2. Round Robin.
3. Shortest job first.
4. All. After execution, the program will generate 3 files corresponding to 3 scheduling algorithms above.
5. Quit.
Your choice: 2
Round Robin Scheduling:
0~P1~4~P2~8~P3~11~P1~15~P2~16~P1~32
P1    TT = 32 WT = 8
P2    TT = 15 WT = 10
P3    TT = 9  WT = 6
Average:    TT = 18.6667    WT = 8
Press any key to continue . . .
```

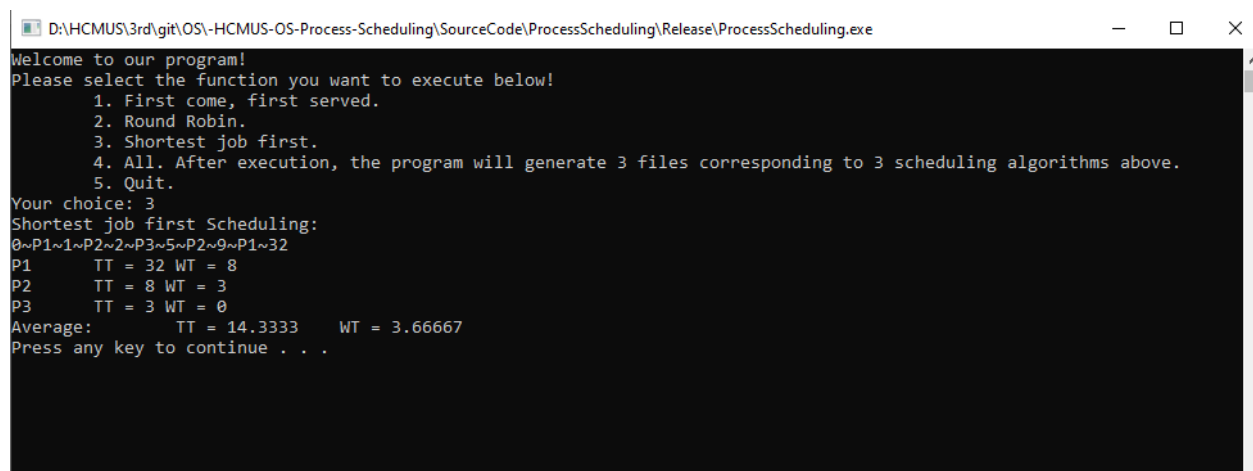
Ấn phím bất kỳ để quay về menu chính.

Shortest job first

Chọn chức năng 3.

Chương trình sẽ tiến hành tải lên tập tin Input.txt. Vui lòng đảm bảo rằng tập tin Input.txt nằm cùng thư mục với tập tin thực thi chương trình ProcessScheduling.exe.

Sau khi tải lên thành công tập tin Input.txt, chương trình sẽ thực hiện điều phối tiến trình theo thuật toán Shortest job first và in ra kết quả thu được như hình dưới.



```
D:\HCMUS\3rd\git\OS\HCMUS-OS-Process-Scheduling\SourceCode\ProcessScheduling\Release\ProcessScheduling.exe
Welcome to our program!
Please select the function you want to execute below!
1. First come, first served.
2. Round Robin.
3. Shortest job first.
4. All. After execution, the program will generate 3 files corresponding to 3 scheduling algorithms above.
5. Quit.
Your choice: 3
Shortest job first Scheduling:
0~P1~1~P2~2~P3~5~P2~9~P1~32
P1    TT = 32 WT = 8
P2    TT = 8  WT = 3
P3    TT = 3  WT = 0
Average:    TT = 14.3333    WT = 3.66667
Press any key to continue . . .
```

Ấn phím bất kỳ để quay về menu chính.

Tất cả thuật toán

Chọn chức năng 4.

Chương trình sẽ tiến hành tải lên tập tin Input.txt. Vui lòng đảm bảo rằng tập tin Input.txt nằm cùng thư mục với tập tin thực thi chương trình ProcessScheduling.exe.

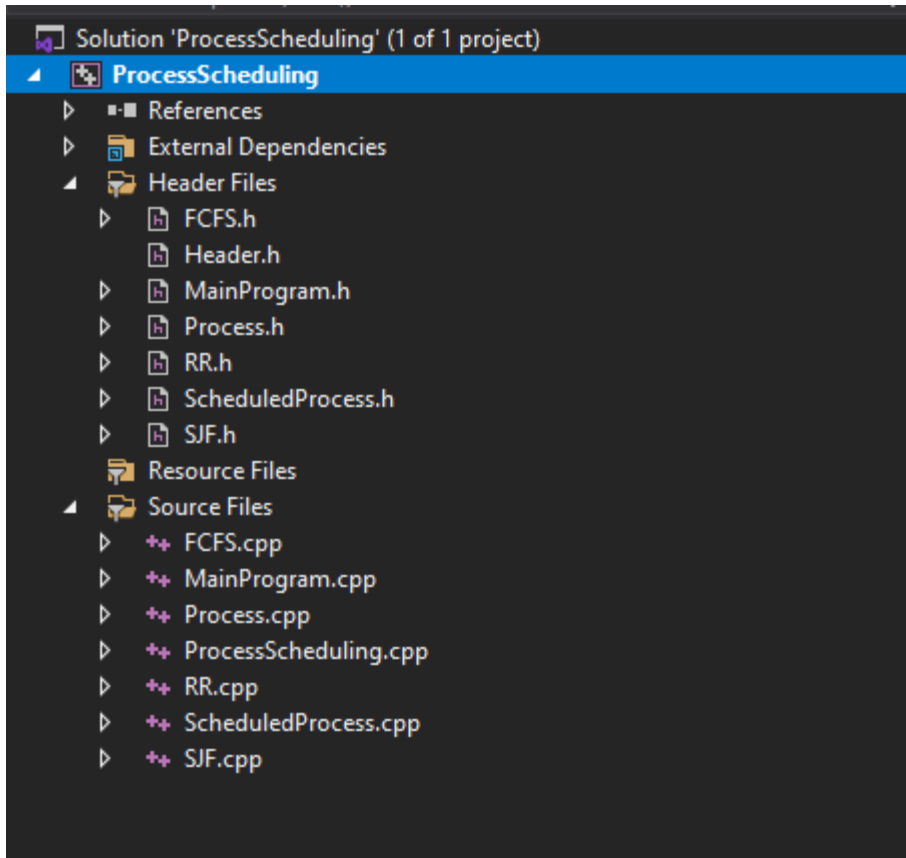
Sau khi tải lên thành công tập tin Input.txt, chương trình sẽ thực hiện điều phối tiến trình theo các thuật toán lần lượt là First come, first served, Round Robin, Shortest job first và in ra kết quả thu được như hình dưới.

```
D:\HCMUS\3rd\git\OS\HCMUS-OS-Process-Scheduling\SourceCode\ProcessScheduling\Release\ProcessScheduling.exe
Welcome to our program!
Please select the function you want to execute below!
1. First come, first served.
2. Round Robin.
3. Shortest job first.
4. All. After execution, the program will generate 3 files corresponding to 3 scheduling algorithms above.
5. Quit.
Your choice: 4
First come, first served Scheduling:
0~P1~24~P2~29~P3~32
P1    TT = 24 WT = 0
P2    TT = 28 WT = 23
P3    TT = 30 WT = 27
Average:    TT = 27.3333    WT = 16.6667
Round Robin Scheduling:
0~P1~4~P2~8~P3~11~P1~15~P2~16~P1~32
P1    TT = 32 WT = 8
P2    TT = 15 WT = 10
P3    TT = 9 WT = 6
Average:    TT = 18.6667    WT = 8
Shortest job first Scheduling:
0~P1~1~P2~2~P3~5~P2~9~P1~32
P1    TT = 32 WT = 8
P2    TT = 8 WT = 3
P3    TT = 3 WT = 0
Average:    TT = 14.3333    WT = 3.66667
Press any key to continue . . .
```

Ấn phím bất kỳ để quay về menu chính.

CẤU TRÚC CHƯƠNG TRÌNH

Cấu trúc thư mục



Chương trình có cấu trúc thư mục như trên.

Gồm các lớp:

- + FCFS: lưu trữ các phương thức sử dụng cho thuật toán First come, first served.
- + RR: lưu trữ các phương thức sử dụng cho thuật toán Round Robin.
- + SJF: lưu trữ các phương thức sử dụng cho thuật toán Shortest job first.
- + MainProgram: menu chương trình chính, điều phối luồng thực thi các tính năng.
- + Process: lưu trữ thông tin các tiến trình chưa được điều phối.
- + ScheduledProcess: kế thừa lớp Process, lưu trữ thông tin các tiến trình sau quá trình điều phối.

Các phương thức dùng chung

- Tính toán thời gian tồn tại của tiến trình: phương thức findTurnaroundTime.
Dựa trên nguyên tắc: $\text{turn_around_time} = \text{burst_time} + \text{waiting_time}$.
Điều này đúng với mọi thuật toán.

```
void ScheduledProcess::findTurnaroundTime(vector<ScheduledProcess>& processes, int n)
{
    for (int i = 0; i < n; i++)
        processes[i].turnaroundTime = processes[i].cpuBurst + processes[i].waitingTime;
}
```

First come, first served

Thuật toán:

Bước 1: Nhập các thông tin tiến trình kèm thời gian thực thi.

Bước 2: Thời gian chờ cho tiến trình đầu tiên là 0.

Bước 3: Tìm thời gian chờ (waiting time) cho từng tiến trình còn lại.

Thời gian chờ của tiến trình thứ i = Tổng thời gian thực thi của tiến trình $(0 \rightarrow i-1)$ - Thời gian tiến trình i vào hàng đợi.

```
void FCFS::fcfs_findWaitingTime(vector<ScheduledProcess>& processes, int n, string& chart)
{
    processes[0].waitingTime = 0;
    Process::updateChart(chart, processes[0], processes[0].cpuBurst);
    for (int i = 1; i < n; i++)
    {
        processes[i].waitingTime = ScheduledProcess::sumCPUBurst(processes, 0, i - 1) - processes[i].arrivalTime;
        Process::updateChart(chart, processes[i], ScheduledProcess::sumCPUBurst(processes, 0, i));
    }
}
```

Bước 4: Tìm thời gian tồn tại của từng tiến trình.

Thời gian tồn tại = Thời gian thực thi – Thời gian chờ.

Bước 5: Tính trung bình thời gian chờ và thời gian tồn tại của các tiến trình.

Round Robin

Thuật toán:

Bước 1: Tạo mảng **rem_bt[]** lưu trữ thời gian thực thi còn lại của từng tiến trình tương ứng.

```
// Make a copy of burst times bt[] to store remaining
// burst times.
vector<int> rem_bt(n);
for (int i = 0; i < n; i++)
    rem_bt[i] = processes[i].cpuBurst;
```

Bước 2: Thời gian chờ cho tiến trình đầu tiên là 0. Mốc thời gian **t** ban đầu là **t = 0**.

Bước 3: Duyệt các phần tử trong danh sách tiến trình đến khi không còn tiến trình nào cần thực hiện (nghĩa là mảng **rem_bt[]** có tất cả các phần tử bằng 0).

Bước 3.1: Nếu tiến trình còn cần thực thi.

Bước 3.1.1: Nếu thời gian còn lại lớn hơn quantum, thực thi tiến trình trong thời gian quantum (đồng thời tăng mốc thời gian **t** một khoảng quantum) và giảm thời gian thực thi còn lại một khoảng quantum.

Bước 3.1.2: Nếu thời gian còn lại nhỏ hơn hoặc bằng quantum, thực thi tiến trình trong thời gian thực thi còn lại và tính toán thời gian chờ.

Thời gian chờ = Mốc thời gian hiện tại **t** – Thời gian thực thi – Thời gian tiến trình vào hàng đợi.

```
if (rem_bt[i] > quantum)
{
    t += quantum;
    rem_bt[i] -= quantum;
}
else
{
    t = t + rem_bt[i];
    processes[i].waitingTime = t - processes[i].cpuBurst - processes[i].arrivalTime;
    rem_bt[i] = 0;
}
```

Bước 3.2: Nếu tiến trình không cần thực thi, duyệt đến tiến trình kế tiếp.

Bước 4: Tìm thời gian tồn tại của từng tiến trình.

Thời gian tồn tại = Thời gian thực thi – Thời gian chờ.

Bước 5: Tính trung bình thời gian chờ và thời gian tồn tại của các tiến trình.

Shortest job first

Thuật toán:

Bước 1: Tạo mảng **rt[]** lưu trữ thời gian thực thi còn lại của từng tiến trình tương ứng.

```
vector<int> rt(n);  
  
// Copy the burst time into rt[]  
for (int i = 0; i < n; i++)  
    rt[i] = processes[i].cpuBurst;
```

Bước 2: Khởi tạo một số giá trị: complete (số tiến trình hoàn thành), t (mốc thời gian), minm (thời gian thực thi còn lại ngắn nhất trong các tiến trình), shortest (chỉ số của tiến trình có thời gian thực thi còn lại ngắn nhất), finish_time (mốc thời gian tiến trình hoàn thành), check (biến xác định trạng thái sẽ tiếp tục chạy tiến trình hiện tại hay chuyển sang tiến trình mới).

```
int complete = 0, t = 0, minm = INT_MAX;  
int shortest = 0, finish_time;  
bool check = false;
```

Bước 3: Duyệt các phần tử trong danh sách tiến trình đến khi không còn tiến trình nào cần thực hiện (nghĩa là complete = n).

Bước 3.1: Tìm kiếm tiến trình có thời gian thực thi còn lại ngắn nhất từng đơn vị thời gian (giây).

Bước 3.2: Giảm thời gian thực thi còn lại của tiến trình tìm được đi 1.

Bước 3.3: Kiểm tra thời gian thực thi còn lại của tiến trình này.

Bước 3.3.1: Nếu thời gian thực thi còn lại bằng 0, tăng số lượng tiến trình hoàn thành, đánh dấu thời gian hoàn thành và tính toán thời gian chờ của tiến trình đó.

Thời gian chờ = Mốc thời gian hoàn thành – Thời gian thực thi – Thời gian tiến trình vào hàng đợi.

Bước 3.3.2: Nếu thời gian thực thi còn lại khác 0, tăng mốc thời gian và quay lại bước 3.

```
// If a process gets completely
// executed
if (rt[shortest] == 0)
{
    // Increment complete
    complete++;
    check = false;
    finish_time = t + 1;

    processes[shortest].waitingTime = finish_time - processes[shortest].cpuBurst - processes[shortest].arrivalTime;

    if (processes[shortest].waitingTime < 0)
        processes[shortest].waitingTime = 0;
}

t++;
```

Bước 4: Tìm thời gian tồn tại của từng tiến trình.

Thời gian tồn tại = Thời gian thực thi – Thời gian chờ.

Bước 5: Tính trung bình thời gian chờ và thời gian tồn tại của các tiến trình.

TÀI LIỆU THAM KHẢO

[1] <https://www.geeksforgeeks.org/>