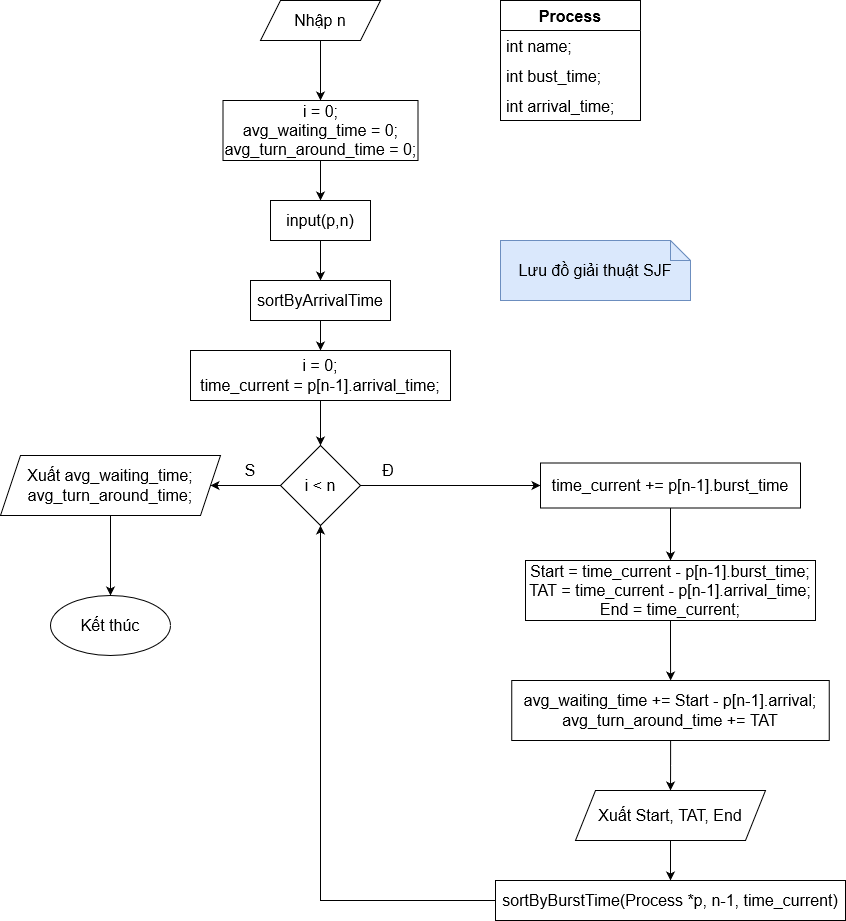
|  |
| --- |
| Họ và tên Sinh viên: Nguyễn Huỳnh Chí  Mã số sinh viên: 110122001  Mã lớp: DA22TTA  Ngày thực hiện: 9/5/2024 |

Bài đánh giá quá trình 1

## Task name 1: Viết chương trình mô phỏng giải thuật SJF.

Lưu đồ giải thuật SJF



Hình: Lưu đồ giải thuật SJF

* + Giải thích:
    - Đầu tiên ta sẽ tạo ra một struct tên process với 3 thông tin cơ bản như trên. Sau đó chúng ta khai báo thêm 2 biến toàn cục là biến tổng thời gian đợi và thời gian thực hiện trong hệ thống.
    - Tiến hành nhập n là số process, Sau đó dùng hàm Input để nhập các thông tin của các process.
    - Sắp xếp lại các tiến trình bằng hàm SortByArrivalTime. Sau đó khai báo thêm biến time\_current = thời gian vào của tiến trình có arrival\_time bé nhất.
    - Cho các tiến trình vào vòng lặp lấy ra phần tử ngoài cùng lúc này tiến trình đầu tiên được thực thi, time\_current lúc này đã được cộng thêm busrt\_time của tiến trình đó lúc này time\_current là thời gian kết thúc của tiến trình trong vòng lặp.
    - Tiến hành tính toán các thời gian Star, TAT, End.
    - Sắp xếp lại các tiến trình còn lại dựa vào hàm sortByBurstTime và lặp lại đối với các tiến trình còn lại.

Code của giải thuật (1 điểm)

Sinh viên hiện thực code C trong file với tên như sau: **<MSSV>\_SJF.cpp**

Ví dụ: Sinh viên Nguyễn Văn ABC có MSSV là 123456789

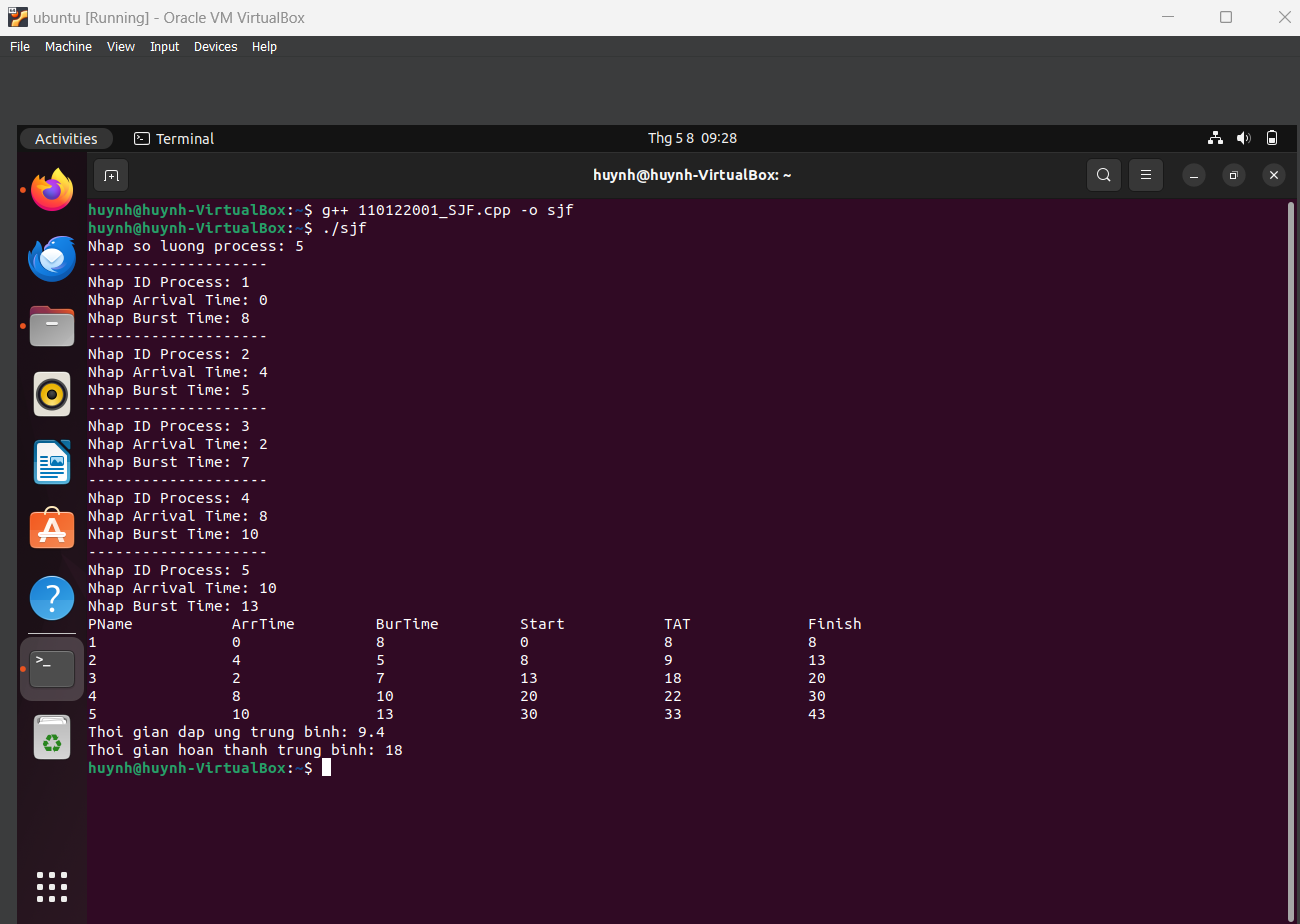
thì tên file là: 123456789\_SJF.cpp

Test case (2 điểm)

* + Ví dụ 1:

|  |  |  |
| --- | --- | --- |
| Process 1 | Arrival Time | Burst Time |
| P1 | 0 | 8 |
| P2 | 4 | 5 |
| P3 | 2 | 7 |
| P4 | 8 | 10 |
| P5 | 10 | 13 |

* + - Kết quả khi chạy code (0.5đ)



* + - Kết quả khi giải tay (0.5đ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P5 |
|  |  |  |  |  |

-Giản đồ Gantt:

0 8 13 20 30 43

-Thời gian chờ:

P1=0, P2=4, P3=11, P4=12, P5=20

Thời gian chờ trung bình: (0+4+11+12+20)/5 = 9.4

-Thời gian đáp ứng:

P1=0, P2=4, P3=11, P4=12, P5=20

Thời gian đáp ứng trung bình: (0+4+11+12+20)/5 = 9.4

-Thời gian hoàn thành:

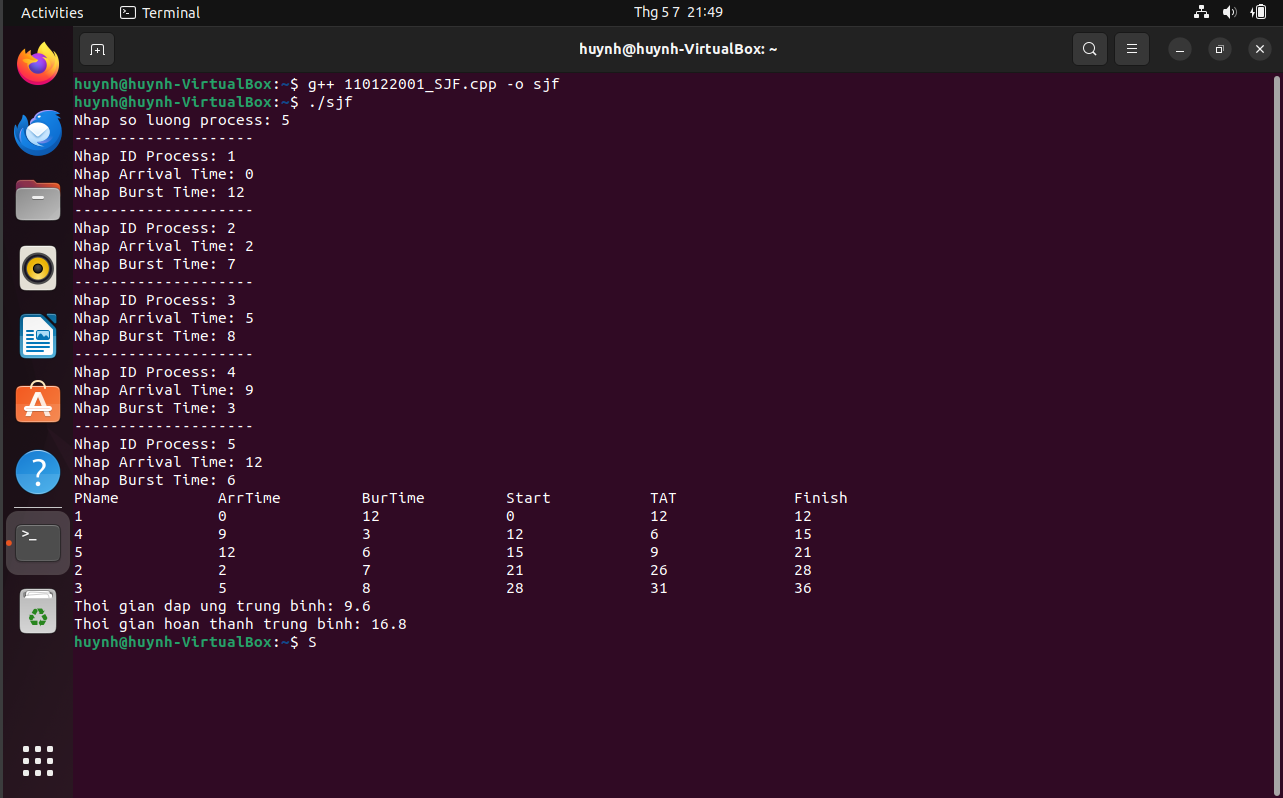
P1=8, P2=9, P3=18, P4=22, P5=33

Thời gian chờ trung bình: (8+9+18+22+33)/5 = 18

* + Ví dụ 2:

|  |  |  |
| --- | --- | --- |
| Process | Arriva Time | Burst Time |
| P1 | 0 | 12 |
| P2 | 2 | 7 |
| P3 | 5 | 8 |
| P4 | 9 | 3 |
| P5 | 12 | 6 |

* + - Kết quả khi chạy code (0.5đ)



* + - Kết quả khi giải tay (0.5đ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P4 | P5 | P2 | P3 |
|  |  |  |  |  |

-Giản đồ Gantt:

0 12 15 21 28 36

-Thời gian chờ:

P1=0, P2=19, P3=23, P4=3, P5=3

Thời gian chờ trung bình: (0+19+23+3+3)/5 = 9.6

-Thời gian đáp ứng:

P1=0, P2=19, P3=23, P4=3, P5=3

Thời gian đáp ứng trung bình: (0+19+23+3+3)/5 = 9.6

-Thời gian hoàn thành:

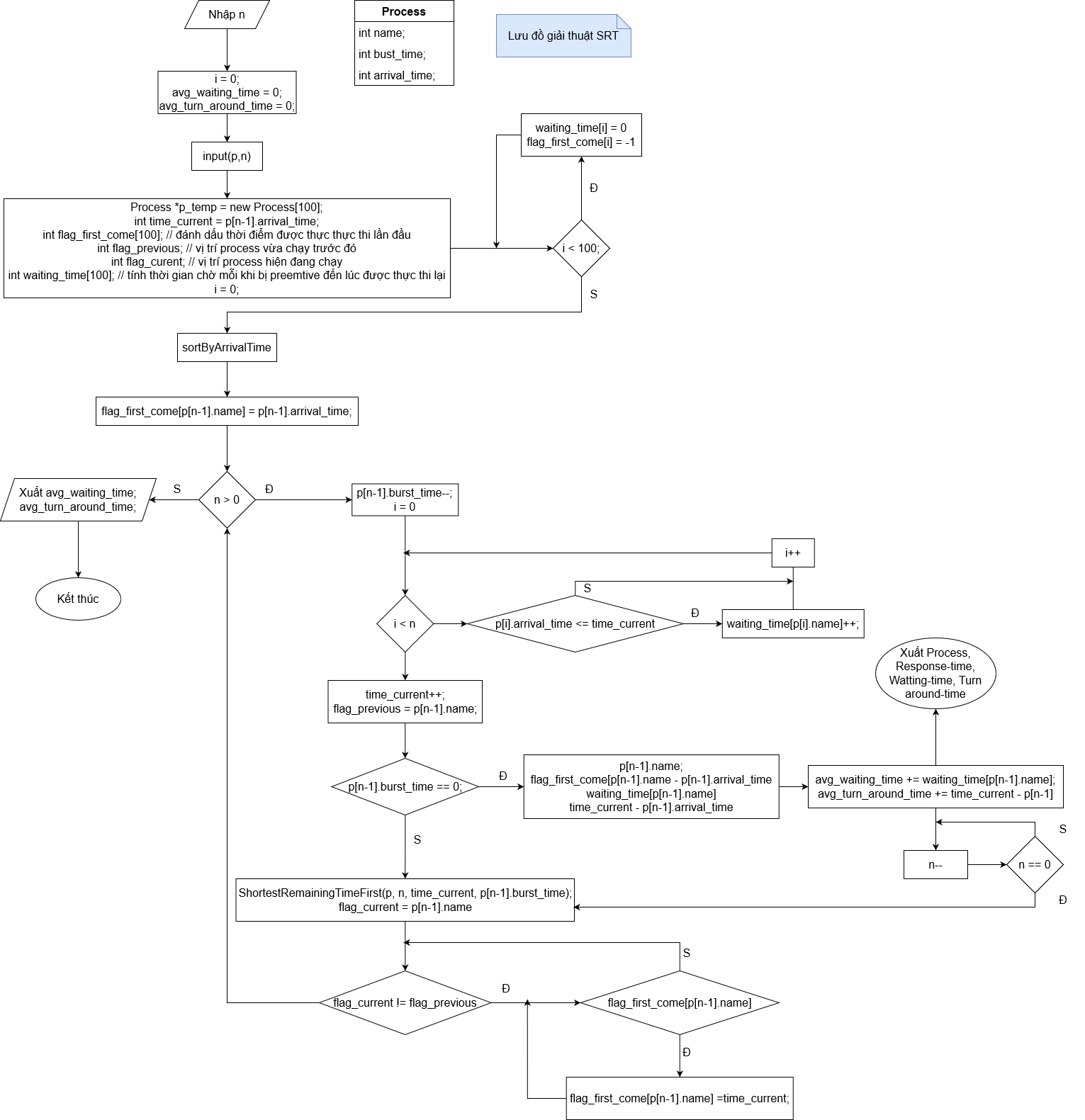
P1=12, P2=26, P3=31, P4=6, P5=9

Thời gian chờ trung bình: (12+26+31+6+9)/5 = 16.8

## Task name 2: Viết chương trình mô phỏng giải thuật SRT.

Lưu đồ giải thuật SRT

* + Giải thích:
    - Các bước đầu sẽ là tạo struct và tiến hành nhập các process tương tự như giải thuật SJF.
    - Sau đó ta sẽ có các biến như là time\_current là timeline của chương trình, flag\_first\_com là list đánh dấu các thời điểm thực thi lần đầu.
    - flag\_previous: Vị trí của process vừa chạy trước đó, lag\_current: vị trí của tiến trình dang chạy; waitting\_time: là thời gian chờ mỗi khi bị preemtive đến lúc được thực thi lại.
    - Ta chạy hàm for cho các mảng waiting\_time và flag\_first\_come để đánh dấu. -1 là chỉ truy cập 1 lần.
    - Sau đó sử dụng hàm SortByArrivalTime để sort tiến trinh.
    - Duyệt từ cuối lên. Ta xép từ từ chậm rãi. Hàm for đầu tiên có tác dụng là tăng waiting time khi processs đã đến hàng đượi mà chưa được thực thi.
    - Tăng timeline lên dần, và lưu tên process sắp rời đi.
    - Với hàm if tiếp theo là nếu đã thực thi hết, không còn burst thì xuất trạng thái. Và ta tính các thông tin Start, TAT, End và cộng dồn thời gian chờ và thời gian hoàn thành. Sau đó giảm n-- để thu hẹp các tiến trình. Khi nào n = 0 thì thoát vòng lặp.
    - Dùng Hàm ShortestRemainingTimeFirst(p, n, time\_current, p[n-1] .burst\_time) để chọn ra các tiến trình có burst < burst còn lại của p[flag\_current].
    - Hàm if ở cuối có nghĩa là nếu xảy ra trường hợp chuyển ngữ cảnh thì thì thời điểm đánh dấu sẽ bằng timeline chương trình.



Hình: Lưu đồ giải thuật SRT

Code giải thuật SRT (1đ)

Sinh viên hiện thực code C trong file với tên như sau: <MSSV>\_SRT.cpp

Ví dụ: Sinh viên Nguyễn Văn ABC có MSSV là 123456789

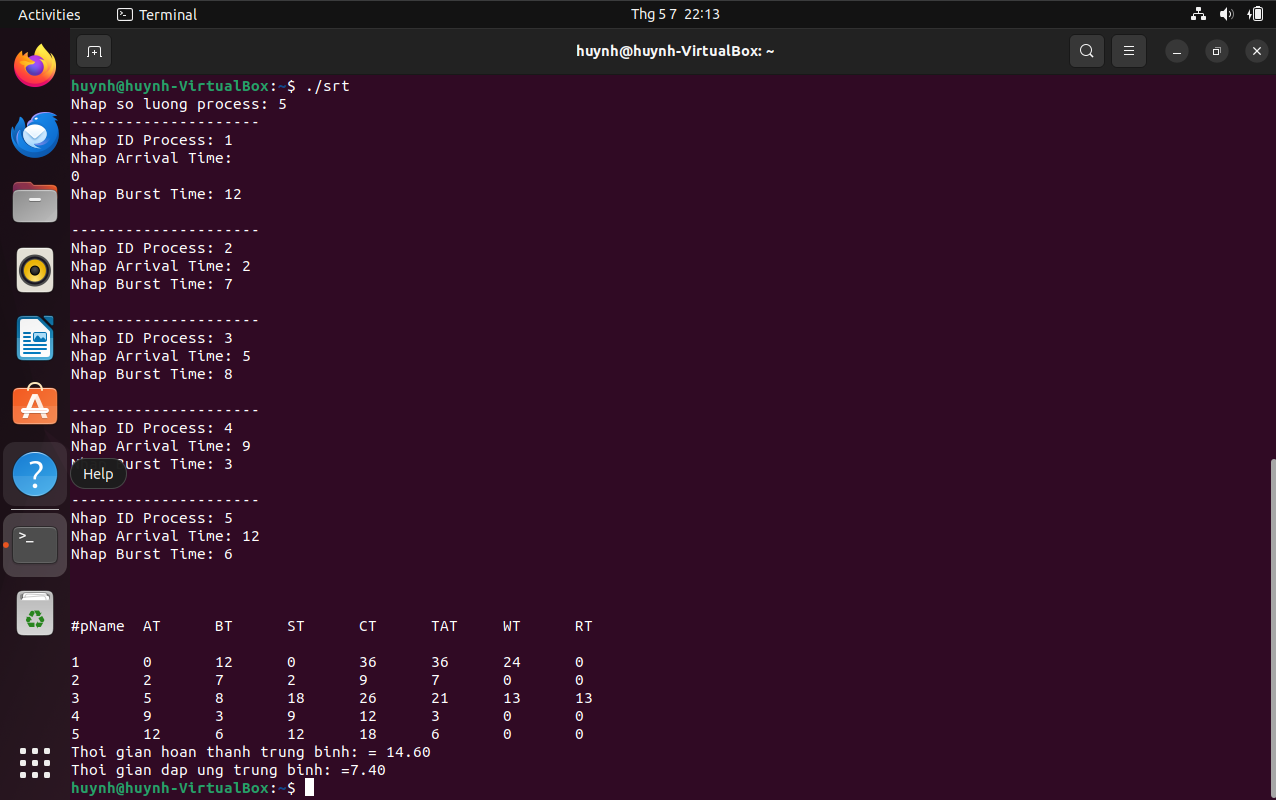
thì tên file là: 123456789\_SRT.cpp

Test case(2đ)

* + Ví dụ 1:

|  |  |  |
| --- | --- | --- |
| Process 1 | Arrival Time | Burst Time |
| P1 | 0 | 12 |
| P2 | 2 | 7 |
| P3 | 5 | 8 |
| P4 | 9 | 3 |
| P5 | 12 | 6 |

* + - Kết quả khi chạy code: (0.5đ)



* + - Kết quả khi giải tay: (0.5đ)

-Giản đồ Gantt:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P2 | P4 | P5 | P3 | P1 |
|  |  |  |  |  |  |

0 2 9 12 18 26 36

-Thời gian chờ:

P1=24, P2=0, P3=13, P4=0, P5=0

Thời gian chờ trung bình: (24+0+13+0+0)/5 = 7.4

-Thời gian đáp ứng:

P1=0, P2=0, P3=13, P4=0, P5=0

Thời gian đáp ứng trung bình: (0+0+13+0+0)/5 = 2.6

-Thời gian hoàn thành:

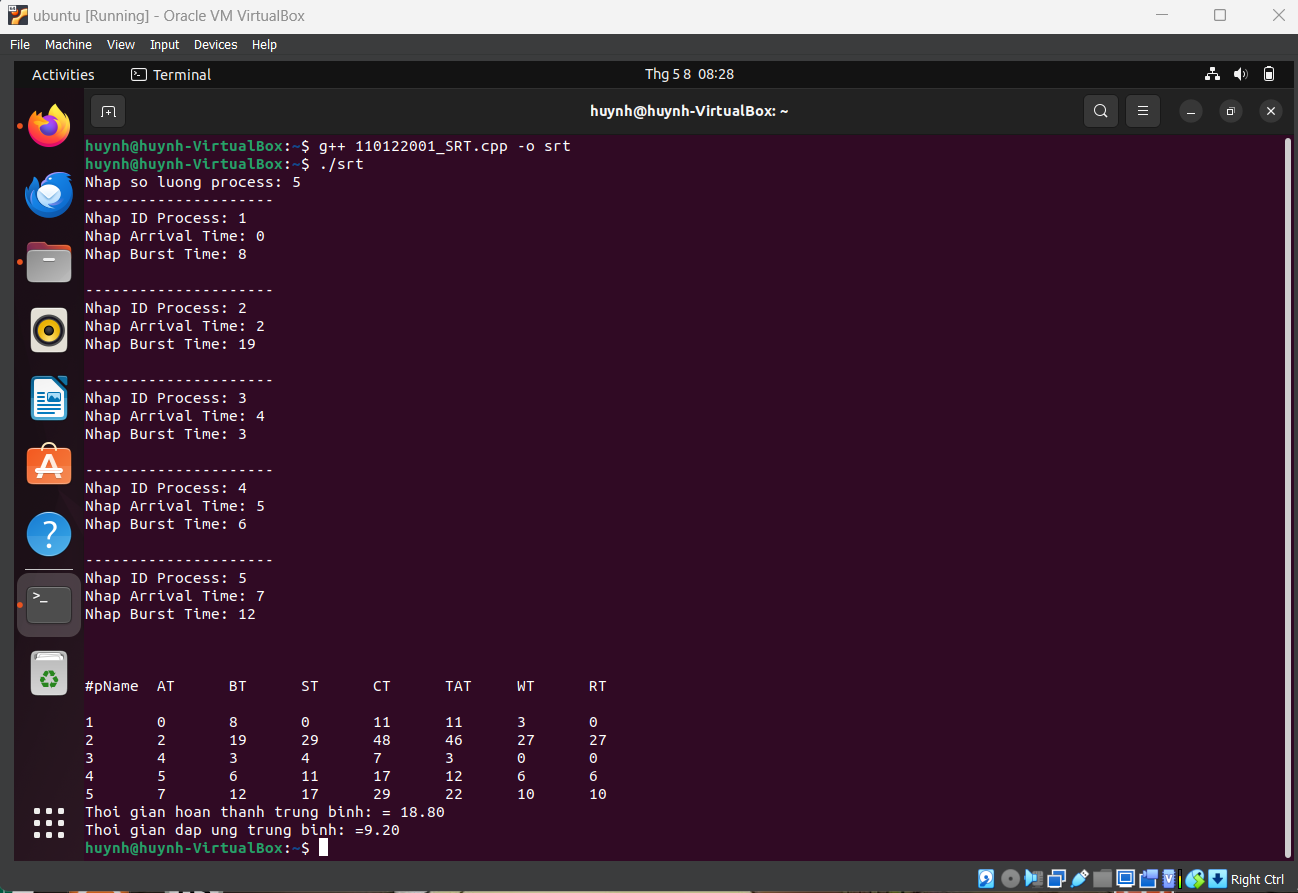
P1=36, P2=7, P3=21, P4=3, P5=6

Thời gian chờ trung bình: (36+7+21+3+6)/5 = 14.6

* + Ví dụ 2:

|  |  |  |
| --- | --- | --- |
| Process 1 | Arrival Time | Burst Time |
| P1 | 0 | 8 |
| P2 | 2 | 19 |
| P3 | 4 | 3 |
| P4 | 5 | 6 |
| P5 | 7 | 12 |

* + - Kết quả khi chạy code: (0.5đ)



* + - Kết quả khi giải tay: (0.5đ)

-Giản đồ Gantt:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P3 | P1 | P4 | P5 | P2 |
|  |  |  |  |  |  |

0 4 7 11 17 29 48

-Thời gian chờ:

P1=3, P2=27, P3=0, P4=6, P5=10

Thời gian chờ trung bình: (3+27+0+6+10)/5 = 9.2

-Thời gian đáp ứng:

P1=3, P2=27, P3=0, P4=6, P5=10

Thời gian đáp ứng trung bình: (3+27+0+6+10)/5 = 9.2

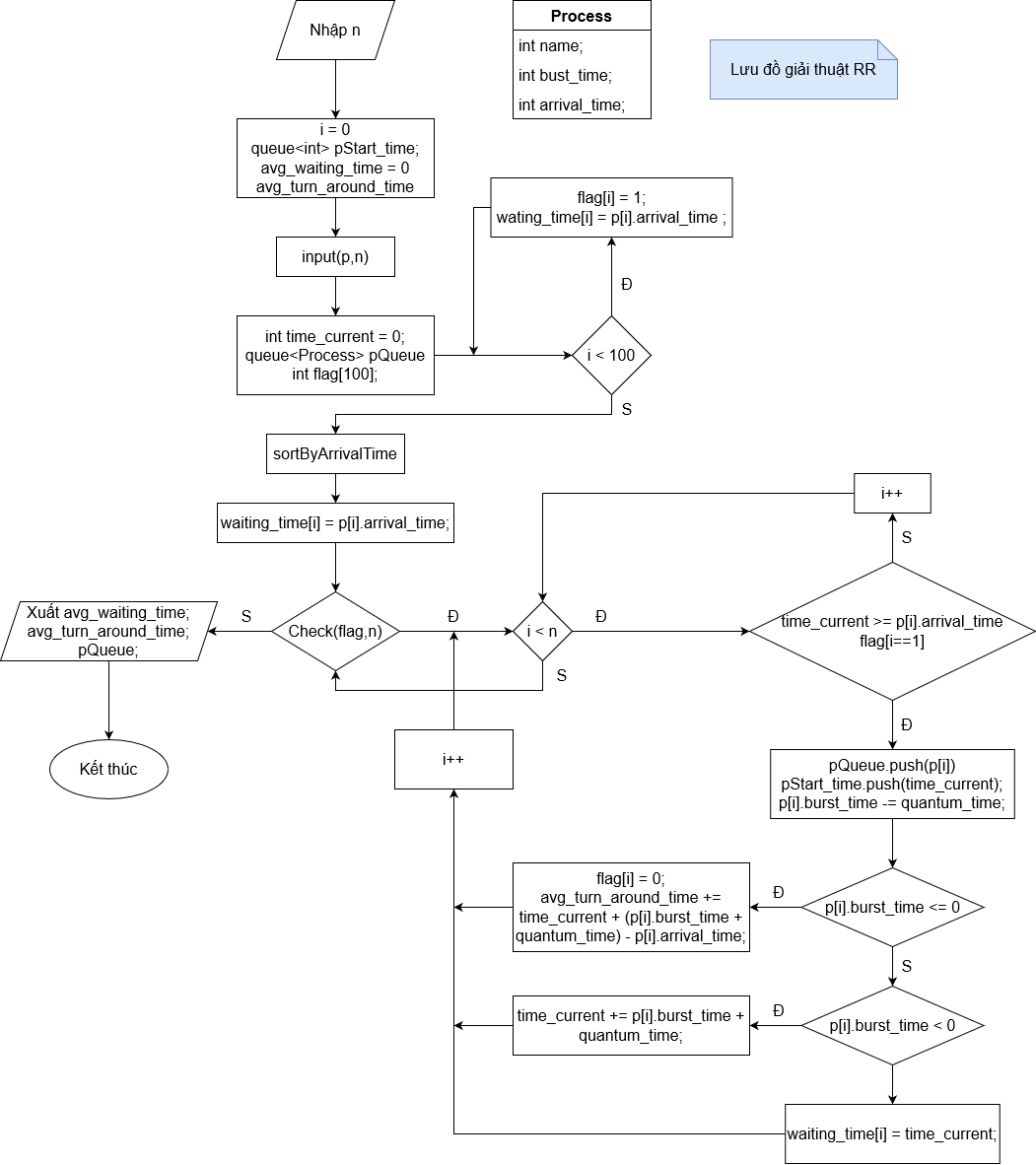
-Thời gian hoàn thành:

P1=11, P2=46, P3=3, P4=12, P5=22

Thời gian chờ trung bình: (11+46+3+12+22)/5 = 18.8

## Task name 3

* 1. Giải thuật RR
  + **Giải thích:**
    - Tương tự như các giải thuật trên ta sẽ tiến hành tạo struct và tiến hành nhập các thông tin. Và nhập quantumtime.
    - Tạo list flag có tác dụng kiểm tra xem busrt\_time của tiến trình còn hay không. Ban đầu ta sẽ gắn hết bằng 1. Và waiting\_time sẽ bằng thời gian đến.
    - Tiến hành sort theo arrival time. Và tạo ra 1 list các tiến trình rỗng khác
    - Dùng hàm check để kiểm tra xem còn tiến trình nào vẫn còn burst\_time hay không.
    - Duyệt qua lần lượt tất cả các process. Tìn trình nào đã đến và còn burst time thì được xét. Ta sẽ tạo bản sao và được gắn vào list qQueue và các thông tin khác sẽ được lưu. Sau đó dảm burstTime đi với số lượng = quantum\_time
    - Còn nếu burst\_time == 0 thì cho flag[i] =0 và tính thời gian đợi và thười gian hoàng thành. Nếu bursttime < 0 thì thời timeline lúc này sẽ được cộng thêm burst\_time và quantum\_time. Các trường hợp khác thì time\_current là timeline lúc này được cộng thêm quantum\_time.
    - Tiến hành với các tiến trình khác và kiểm tra day flag còn phần tử nào bằng 1 hay không. Nếu hết rỏi thì xuất ra qQueue. Từ đó sẽ lấy được thông tin cụ thể.



Hình: Lưu đồ giải thuật RR

* 1. Code giải thuật RR (1đ)

Sinh viên hiện thực code C trong file với tên như sau: <MSSV>\_RR.cpp

Ví dụ: Sinh viên Nguyễn Văn ABC có MSSV là 123456789

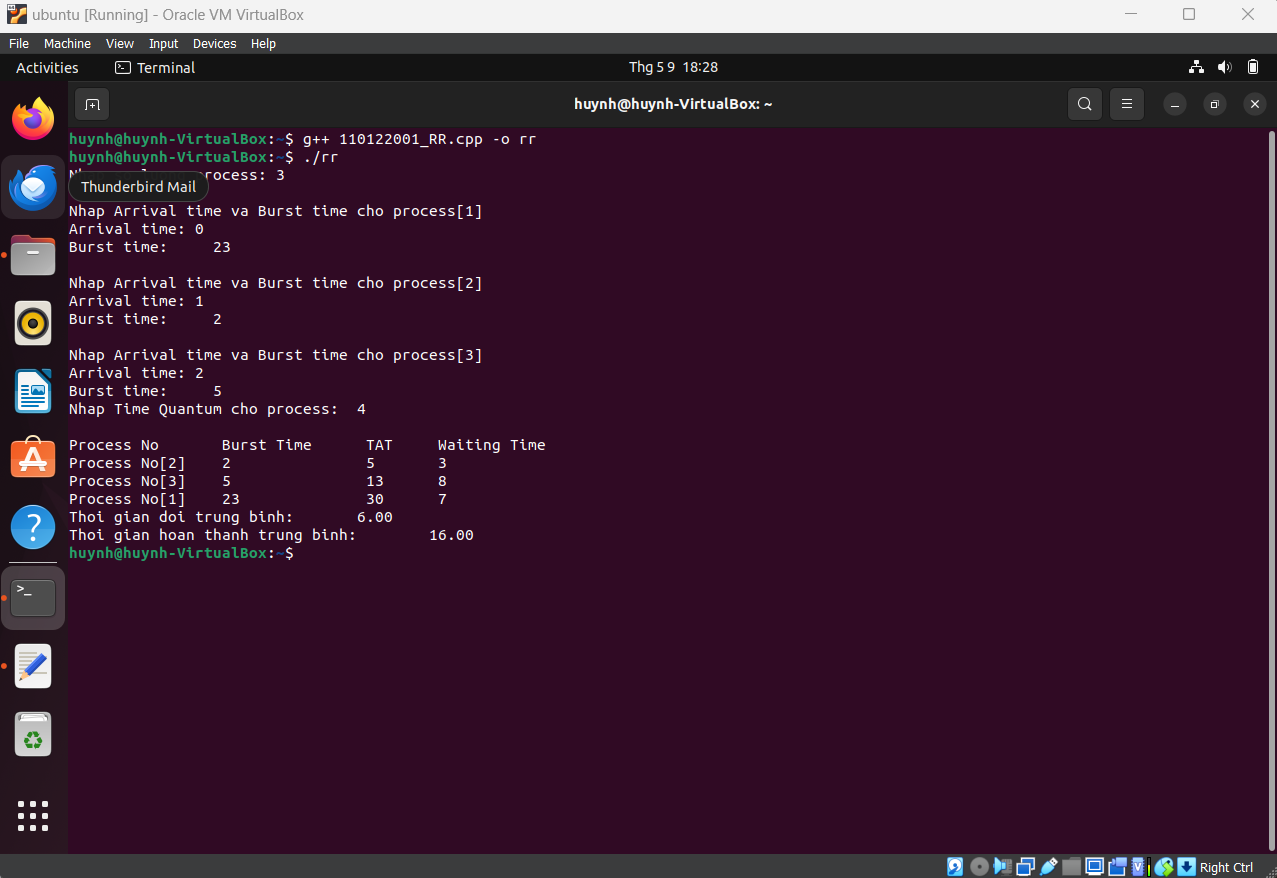
thì tên file là: 123456789\_RR.cpp

* 1. Test case (3đ)
  + Ví dụ 1:

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 23 |
| P2 | 1 | 2 |
| P3 | 2 | 5 |

Q=4

* Kết quả khi chạy code:



* Kết quả khi giải tay:

-Giản đồ Gantt:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P1 | P3 | P1 | P1 | P1 | P1 |
|  |  |  |  |  |  |  |  |  |

0 4 6 10 14 15 19 23 27 30

-Thời gian chờ:

P1= 7, P2= 3, P3= 8

Thời gian chờ trung bình: (7+3+8)/3=6

-Thời gian hoàn thành :

P1= 30, P2= 5, P3= 13

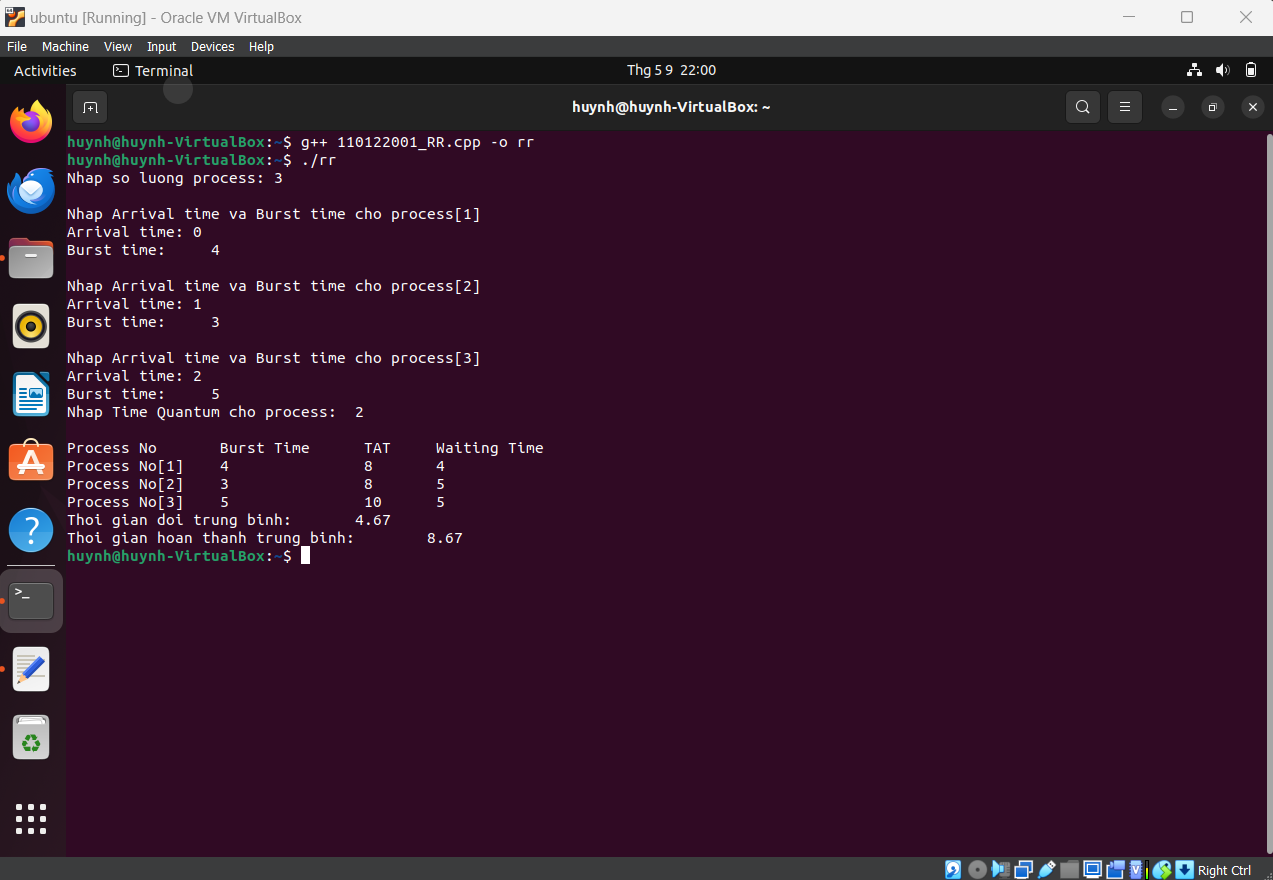
Thời gian hoàn thành trung bình: (30+5+13)/3=16

* + Ví dụ 2:

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 4 |
| P2 | 1 | 3 |
| P3 | 2 | 5 |

Q=2

* Kết quả khi chạy code:



* Kết quả khi giải tay:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P1 | P2 | P3 | P3 |
|  |  |  |  |  |  |  |

0 2 4 6 8 9 11 12

-Thời gian chờ:

P1= 4, P2= 5, P3= 5

Thời gian chờ trung bình: (4+5+5)/3=4.67

-Thời gian hoàn thành :

P1= 8, P2= 8, P3= 10

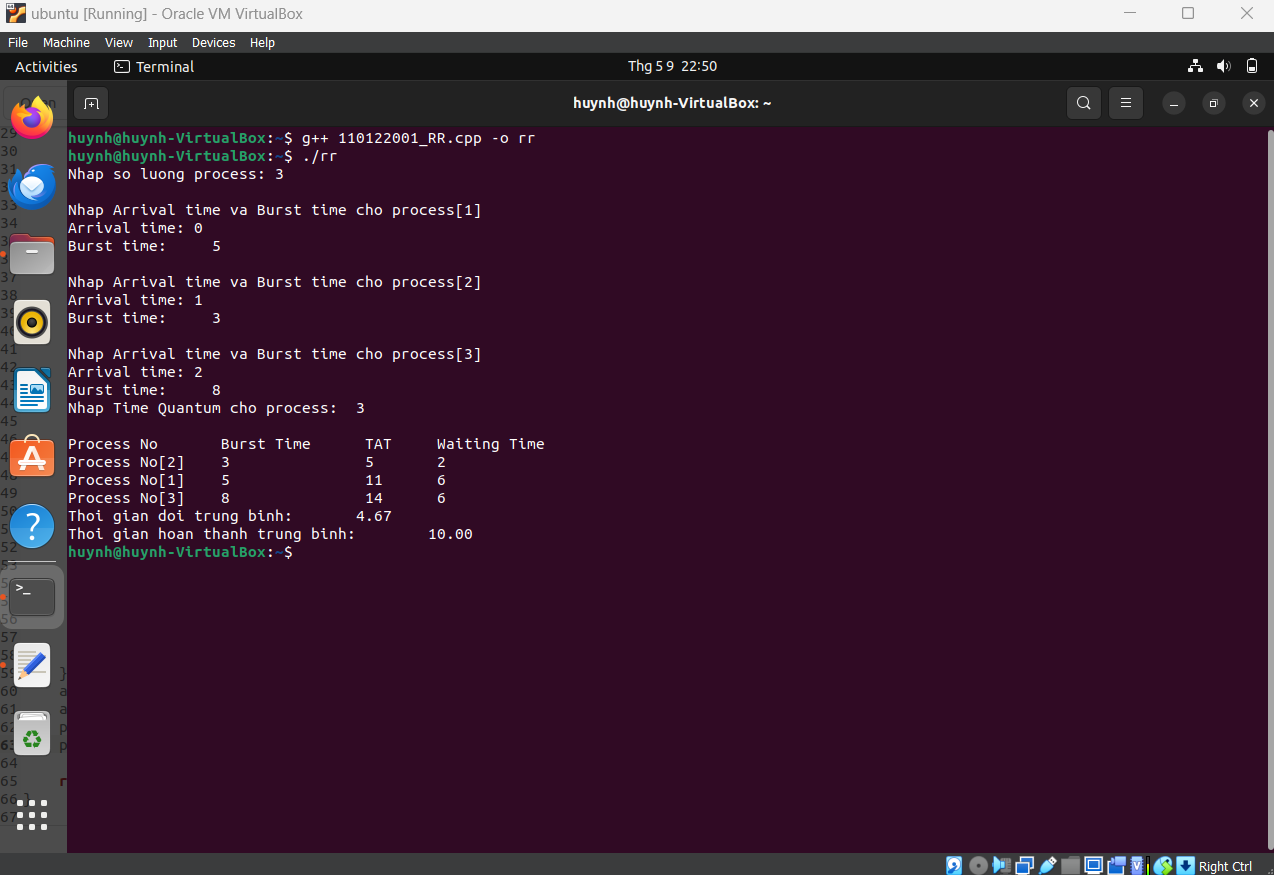
Thời gian hoàn thành trung bình: (8+8+10)/3=8.67

* + Ví dụ 3:

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 5 |
| P2 | 1 | 3 |
| P3 | 2 | 8 |

Q=2

* Kết quả khi chạy code:



* Kết quả khi giải tay:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P1 | P3 | P3 |
|  |  |  |  |  |  |

0 3 6 9 11 14 16

-Thời gian chờ:

P1= 2, P2= 6, P3= 6

Thời gian chờ trung bình: (2+6+6)/3=4.67

-Thời gian hoàn thành :

P1= 11, P2= 5, P3= 14

Thời gian hoàn thành trung bình: (11+5+14)/3=10

Lưu ý: sinh viên nộp tổng cộng là 4 file:

1. File word (là file này) với tên file <MSSV>\_BaoCaoQT1.docx
2. <MSSV>\_SJF.cpp
3. <MSSV>\_SRT.cpp
4. <MSSV>\_RR.cpp