**https://github.com/BrentLeeSF/DiscSchedulingAlgorithms/blob/master/diskAlgorithms.c?fbclid=IwAR2JaDbueKs-5X\_PQO06T-iKosx5a7brWDyf7Nk5TKuoKGTmwfj2kRs-g5o**

**Mục tiêu:**

So Sánh hiệu quả thuật toán lập lịch

FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK

**Chức năng chính:**

Mô phỏng 1 đĩa có 5000 cylinders và random 1000 cylinder cần đầu đọc đến. Với vị trí bắt đầu của đầu đọc cho trước, chương trình sẽ tính tổng số lượng di chuyển mà đầu đọc đã đi qua.

Giải thích thuật toán:

-FCFS(First Come First Server): head bắt đầu đi từ start value đến end qua tất cả request, nếu hết thì quay lại 0

-SSTF(Shortest Seek time first): head tìm 2 yêu cầu cạnh mình, so sánh xem cái nào ngắn nhất và đi đến đó

-SCAN: Từ điểm bắt đầu head đi đến cuối và từ điểm bắt đầu nó đi đến 0 qua tất cả request

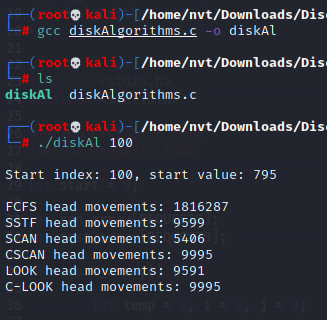
-C-SCAN: Giống Scan nhưng đi theo vòng tròn đến cuối rồi quay lại từ đầu

-LOOK: Head đi từ giá trị lớn hơn start value rồi đi đến giá trị lớn nhất sau đó từ giá trị nhỏ hơn value đi đến giá trị nhỏ nhất

-C-LOOK: Giống LOOK nhưng đi theo vòng tròn, đến cuối rồi quay về đầu

**Cài đặt và chạy:**  
 Cài đặt trên môi trường linux(không có lưu ý j đặc biệt):

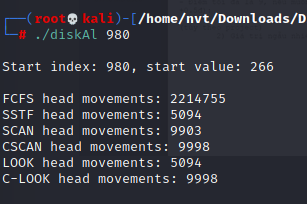
-Git clone <https://github.com/BrentLeeSF/DiscSchedulingAlgorithms.git>



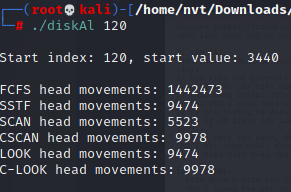
**Xử lí dữ liệu:**

Đầu vào : vị trí bắt đầu của head để đọc (số nguyên từ 0 -4999)

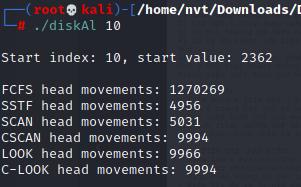
Đầu ra: số lượng di chuyển mà đầu đọc đã phải di chuyển để đọc hết tất cả 1000 yêu cầu



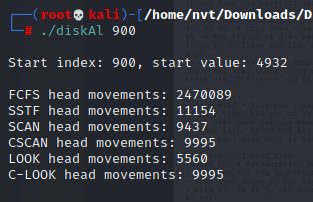
FCFS luôn chạy lâu nhất(Có thể lên đến O(4N))



CSCAN và C-LOOK luôn luôn sấp sỉ nhau



SSTF, SCAN và LOOK trong vài trường hợp có thể = 1 nửa CSCAN => tối ưu nhưng trong trường hợp xấu thì nhỏ hơn CSCAN 1 chút



SSTF trong trường hợp xâu có thể lớn hơn CSCAN

* Khuyên dùng SCAN hoặc LOOK
* Trong trường hợp tối ưu 2 thuật toán có thể duyệt những yêu cầu trong O(N)
* Trong trường hợp xấu thì 2 thuật toán có thể duyệt những yêu cầu trong O(2N)