SYSTEM SOFTWARE LAB
EXPERIMENT 2

Amruth DD
Roll.NO:09
S5 CS

September 9, 2020

Contents

1 Aim

Simulate the following disk scheduling algorithms.

1. FCFS
2. SCAN
3. C-SCAN

2 Algorithm and Source Code

2.1 FCFS

2.1.1 Algorithm

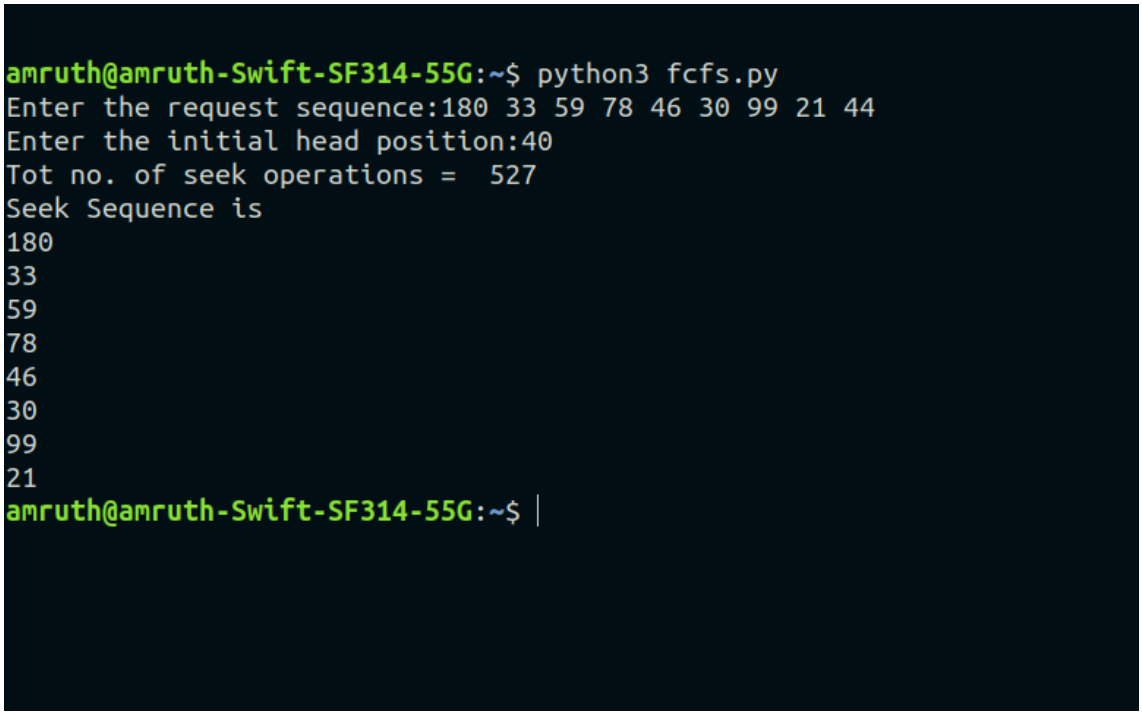
```
1. Let Request array represents an array storing indexes of tracks
that have been requested in ascending order of their time of
arrival head is the position of disk head .
2. Let us one by one take the tracks in default order and calculate
the absolute distance of the track from the head .
3. Increment the total seek count with this distance .
4. Currently serviced track position now becomes the new head
position .
5. Go to step 2 until all tracks in request array have not been
serviced
```

2.1.2 Source code

```
size = 8;
def FCFS(arr, head):
    seek_count = 0;
    distance, cur_track = 0, 0;
    for i in range(size):
        cur_track = arr[i];
        distance = abs(cur_track - head);
        seek_count += distance;
        head = cur_track;
    print("Total number of seek operations = ", seek_count);
    print("Seek Sequence is");
    for i in range(size):
    print(arr[i]);
if __name__ == '__main__':
    str_arr = input("Enter the request sequence:").split(' ')
    arr = [int(num) for num in str_arr]
    head = int(input("Enter the initial head position:"))
```

```
    FCFS(arr, head);
```

2.1.3 Output



2.2 SCAN

2.2.1 Algorithm

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival . head is the position of disk head .
2. Let direction represents whether the head is moving towards left or right .
3. In the direction in which head is moving service all tracks one by one .
4. Calculate the absolute distance of the track from the head .
5. Increment the total seek count with this distance .
6. Currently serviced track position now becomes the new head position .
7. Go to step 3 until we reach at one of the ends of the disk .
8. If we reach at the end of the disk reverse the direction and go to

step 2 until all tracks in request array have not been serviced.

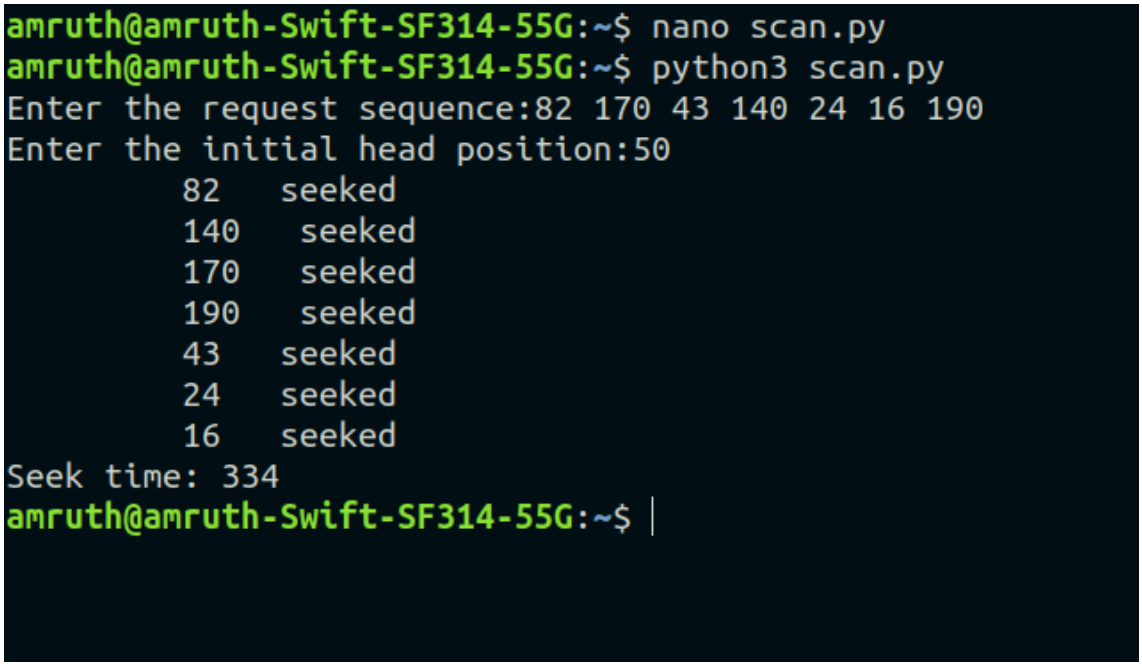
2.2.2 Source code

```
def SCAN(hp,reqs):
    requests = reqs
    pos = hp
    time = 0
    end=200
    start=0

    for i in range(pos,end+1):
        if i in requests:
            time+=abs(pos-i)
            pos=i
            print("    ",i,"  seeked")
            requests.remove(i)
    time+=abs(pos-end)
    pos=end

    for i in range(end,start-1,-1):
        if i in requests:
            time+=abs(pos-i)
            pos=i
            print("    ",i,"   seeked")
            requests.remove(i)
    print("Seek time:",time)
if __name__ == '__main__':
    str_arr = input("Enter the request sequence:").split(' ')
    arr = [int(num) for num in str_arr]
    head = int(input("Enter the initial head position:"))
    SCAN(head, arr)
```

2.2.3 Output



2.3 C-SCAN

2.3.1 Algorithm

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival head is the position of disk head .
2. The head services only in the right direction from 0 to size of the disk .
3. While moving in the left direction do not service any of the tracks .
4. When we reach at the beginning (left end) reverse the direction .
5. While moving in right direction it services all tracks one by one .
6. While moving in right direction calculate the absolute distance of the track from the head .
7. Increment the total seek count with this distance .
8. Currently serviced track position now becomes the new head position .
9. Go to step 6 until we reach at right end of the disk .

2.3.2 Source code

```
def C_SCAN(hp,reqs):
    requests = reqs
    pos = hp
    time = 0
    end=200
    start=0
        for i in range(pos,end+1):
            if i in requests:
                time+=abs(pos-i)
                pos=i
                print("     ",i,"   seeked")
                requests.remove(i)
    time+=abs(pos-end)
    pos=end
    for i in range(start,hp+1):
        if i in requests:
            time+=abs(pos-i)
            pos=i
            print("     ",i,"    seeked")
            requests.remove(i)
    print(time)
if __name__ == '__main__':
    str_arr = input("Enter the request sequence:").split(' ')
    arr = [int(num) for num in str_arr]
    head = int(input("Enter the initial head position:"))
    C_SCAN(head, arr)
```

2.3.3 Output

```
amruth@amruth-Swift-SF314-55G:~$ python3 cscan.py
Enter the request sequence:82 170 43 140 24 16 190
Enter the initial head position:50
          82   seeked
          140   seeked
          170   seeked
          190   seeked
          16   seeked
          24   seeked
          43   seeked
```

3 Result

Different disk scheduling algorithms performed.