Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Technology

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

**Assignment-8**

|  |  |
| --- | --- |
| **Subject** | Operating System |
| **Name** | Ajinkya Walunj |
| **Class** | CS-A |
| **Roll No.** | 05 |

**Title**: I**mplement following Disk Scheduling Algorithms**

**1. FCFS**

**2. SSTF**

**3. SCAN**

**4. C SCAN**

1. **FCFS**

The FCFS disk scheduling algorithm has the advantage of being simple to implement and can guarantee that every I/O request will eventually be serviced. However, it can also result in poor performance, especially if there are long requests at the beginning of the queue. This is because requests at the beginning of the queue are serviced before those at the end, which may lead to higher average access times for requests at the end of the queue.

*Code :-*

#include<stdio.h>

#include<stdlib.h>

int no\_of\_tracks;

int main(){

    printf("Enter the number of request tracks: ");

    scanf("%d",&no\_of\_tracks);

    int sec\_track[no\_of\_tracks][2];

    for(int i=0;i<no\_of\_tracks;i++){

        printf("Enter the %d(th/st/nd/rd) track: ",i+1);

        scanf("%d",&sec\_track[i][0]);

    }

    int head;

    printf("Enter the head: ");

    scanf("%d",&head);

    for(int i=0;i<no\_of\_tracks;i++){

        sec\_track[i][1]=abs(head - sec\_track[i][0]);

        head=sec\_track[i][0];

    }

    int total\_seek\_time;

    for(int i=0;i<no\_of\_tracks;i++){

        total\_seek\_time+=sec\_track[i][1];

    }

    printf("Next track to be traversed  No. of tracks to be traversed\n");

    for(int i=0;i<no\_of\_tracks;i++){

        printf("\t\t%d\t\t",sec\_track[i][0]);

        printf("\t\t%d\t\t",sec\_track[i][1]);

        printf("\n");

    }

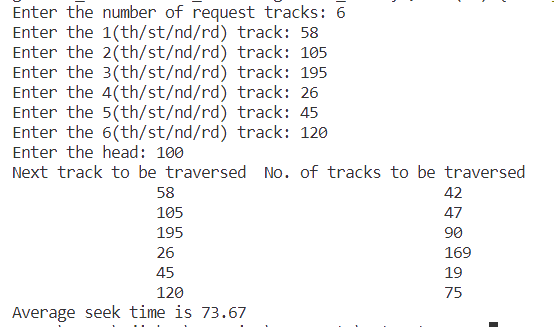
    float avg\_s\_time = (float)total\_seek\_time/no\_of\_tracks;

    printf("Average seek time is %.2f ",avg\_s\_time);

    return 0;

}

*Output :-*



1. **SSTF**

The SSTF disk scheduling algorithm works by selecting the I/O request that is closest to the current position of the disk head. In other words, it services the request that has the shortest seek time, i.e., the minimum distance between the current position of the disk head and the location of the next request.

*Code :-*

#include<stdio.h>

#include<stdlib.h>

int no\_of\_tracks;

int main(){

    printf("Enter the number of request tracks: ");

    scanf("%d",&no\_of\_tracks);

    int sec\_track[no\_of\_tracks][2];

    for(int i=0;i<no\_of\_tracks;i++){

        printf("Enter the %d(th/st/nd/rd) track: ",i+1);

        scanf("%d",&sec\_track[i][0]);

    }

    for(int i=0;i<no\_of\_tracks;i++){

        sec\_track[i][1]=0;

    }

    int head;

    printf("Enter the head: ");

    scanf("%d",&head);

    int temp[no\_of\_tracks][2];

    int j=0;

    for(int i=0;i<no\_of\_tracks;i++){

        temp[i][1]=0;

    }

    int headt = head;

    int flag;

    int count = 0;

    while(count<=no\_of\_tracks){

        int min = 9999;

        for(int i=0;i<no\_of\_tracks;i++){

            if(abs(sec\_track[i][0]-headt)<min && sec\_track[i][0]!=-1){

                min = abs(sec\_track[i][0]-headt);

                flag = i;

            }

        }

        count++;

        headt = sec\_track[flag][0];

        temp[j][0]=headt;

        sec\_track[flag][0]=-1;

        j++;

    }

    printf("Next track to be traversed  No. of tracks to be traversed\n");

    for(int i=0;i<no\_of\_tracks;i++){

        printf("\t\t%d\t\t",temp[i][0]);

        printf("\t\t%d\t\t",temp[i][1]);

        printf("\n");

    }

    for(int i=0;i<no\_of\_tracks;i++){

        temp[i][1]=abs(head - temp[i][0]);

        head=temp[i][0];

    }

    int total\_seek\_time=0;

    for(int i=0;i<no\_of\_tracks;i++){

        total\_seek\_time+=temp[i][1];

    }

    printf("Next track to be traversed  No. of tracks to be traversed\n");

    for(int i=0;i<no\_of\_tracks;i++){

        printf("\t\t%d\t\t",temp[i][0]);

        printf("\t\t%d\t\t",temp[i][1]);

        printf("\n");

    }

    printf("Total seek time: %d\n",total\_seek\_time);

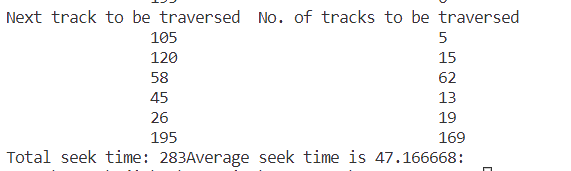
    float avg\_s\_time = (float)total\_seek\_time/(no\_of\_tracks);

    printf("Average seek time is %f: ",avg\_s\_time);

    return 0;

}

*Output :-*



1. **SCAN**

The SCAN algorithm works by servicing I/O requests in one direction until there are no more requests in that direction, and then it reverses direction and services the remaining requests in the opposite direction. The disk head moves in one direction only until it reaches the end of the disk and then reverses direction.

*Code:-*

#include<stdio.h>

#include<stdlib.h>

int no\_of\_tracks;

int main(){

    printf("Enter the number of request tracks: ");

    scanf("%d",&no\_of\_tracks);

    int sec\_track[no\_of\_tracks][2];

    for(int i=0;i<no\_of\_tracks;i++){

        printf("Enter the %d(th/st/nd/rd) track: ",i+1);

        scanf("%d",&sec\_track[i][0]);

    }

    for(int i=0;i<no\_of\_tracks;i++){

        sec\_track[i][1]=0;

    }

    int head;

    printf("Enter the head: ");

    scanf("%d",&head);

    for(int i=0;i<no\_of\_tracks-1;i++){

        for(int j=0;j<no\_of\_tracks-i-1;j++){

            if(sec\_track[j][0]<sec\_track[j+1][0]){

                int temp = sec\_track[j][0];

                sec\_track[j][0]=sec\_track[j+1][0];

                sec\_track[j+1][0]=temp;

            }

        }

    }

    int flag = 0;

    for(int i=0;i<no\_of\_tracks;i++){

        if(sec\_track[i][0]<head){

            flag = i;

            break;

        }

    }

    int l = 0;

    int r = flag-1;

    while(l<r){

        int temp = sec\_track[l][0];

        sec\_track[l][0]=sec\_track[r][0];

        sec\_track[r][0]=temp;

        l++;

        r--;

    }

    printf("Next track to be traversed  No. of tracks to be traversed\n");

    for(int i=0;i<no\_of\_tracks;i++){

        printf("\t\t%d\t\t",sec\_track[i][0]);

        printf("\t\t%d\t\t",sec\_track[i][1]);

        printf("\n");

    }

    for(int i=0;i<no\_of\_tracks;i++){

        sec\_track[i][1]=abs(head - sec\_track[i][0]);

        head=sec\_track[i][0];

    }

    int total\_seek\_time;

    for(int i=0;i<no\_of\_tracks;i++){

        total\_seek\_time+=sec\_track[i][1];

    }

    printf("Next track to be traversed  No. of tracks to be traversed\n");

    for(int i=0;i<no\_of\_tracks;i++){

        printf("\t\t%d\t\t",sec\_track[i][0]);

        printf("\t\t%d\t\t",sec\_track[i][1]);

        printf("\n");

    }

    printf("Total seek time: %d\n",total\_seek\_time);

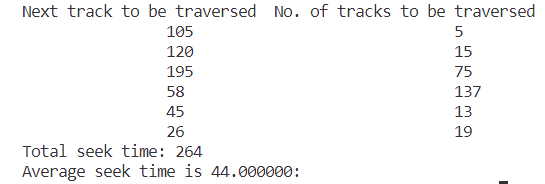
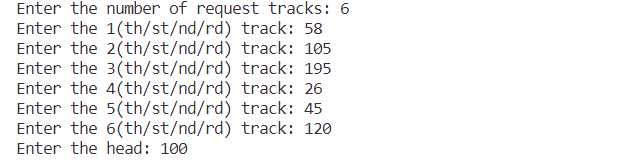
    float avg\_s\_time = (float)total\_seek\_time/no\_of\_tracks;

    printf("Average seek time is %f: ",avg\_s\_time);

    return 0;

}

*Output:-*

**

1. **CSCAN**

The C-SCAN algorithm works by servicing I/O requests in one direction only and when it reaches the end of the disk, it immediately returns to the beginning of the disk without servicing any requests on the way back. This results in a circular path of disk access.

*Code:-*

#include<stdio.h>

#include<stdlib.h>

int no\_of\_tracks;

int main(){

    printf("Enter the number of request tracks: ");

    scanf("%d",&no\_of\_tracks);

    int sec\_track[no\_of\_tracks][2];

    for(int i=0;i<no\_of\_tracks;i++){

        printf("Enter the %d(th/st/nd/rd) track: ",i+1);

        scanf("%d",&sec\_track[i][0]);

    }

    for(int i=0;i<no\_of\_tracks;i++){

        sec\_track[i][1]=0;

    }

    int head;

    printf("Enter the head: ");

    scanf("%d",&head);

    for(int i=0;i<no\_of\_tracks-1;i++){

        for(int j=0;j<no\_of\_tracks-i-1;j++){

            if(sec\_track[j][0]>sec\_track[j+1][0]){

                int temp = sec\_track[j][0];

                sec\_track[j][0]=sec\_track[j+1][0];

                sec\_track[j+1][0]=temp;

            }

        }

    }

    int flag = 0;

    for(int i=0;i<no\_of\_tracks;i++){

        if(sec\_track[i][0]>head){

            flag = i;

            break;

        }

    }

    int j = 0;

    while(flag<no\_of\_tracks){

        int temp=sec\_track[j][0];

        sec\_track[j][0] = sec\_track[flag][0];

        sec\_track[flag][0]=temp;

        flag++;

        j++;

    }

    printf("Next track to be traversed  No. of tracks to be traversed\n");

    for(int i=0;i<no\_of\_tracks;i++){

        printf("\t\t%d\t\t",sec\_track[i][0]);

        printf("\t\t%d\t\t",sec\_track[i][1]);

        printf("\n");

    }

    for(int i=0;i<no\_of\_tracks;i++){

        sec\_track[i][1]=abs(head - sec\_track[i][0]);

        head=sec\_track[i][0];

    }

    int total\_seek\_time;

    for(int i=0;i<no\_of\_tracks;i++){

        total\_seek\_time+=sec\_track[i][1];

    }

    printf("Next track to be traversed  No. of tracks to be traversed\n");

    for(int i=0;i<no\_of\_tracks;i++){

        printf("\t\t%d\t\t",sec\_track[i][0]);

        printf("\t\t%d\t\t",sec\_track[i][1]);

        printf("\n");

    }

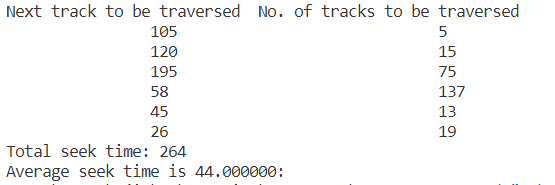
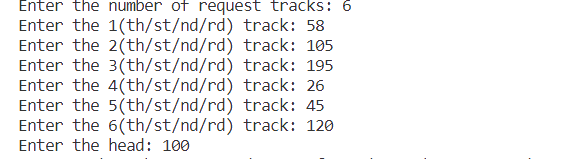
    float avg\_s\_time = (float)total\_seek\_time/no\_of\_tracks;

    printf("Average seek time is %f: ",avg\_s\_time);

    return 0;

}

*Output:-*

**