

## PROGRAM NUMBER :6

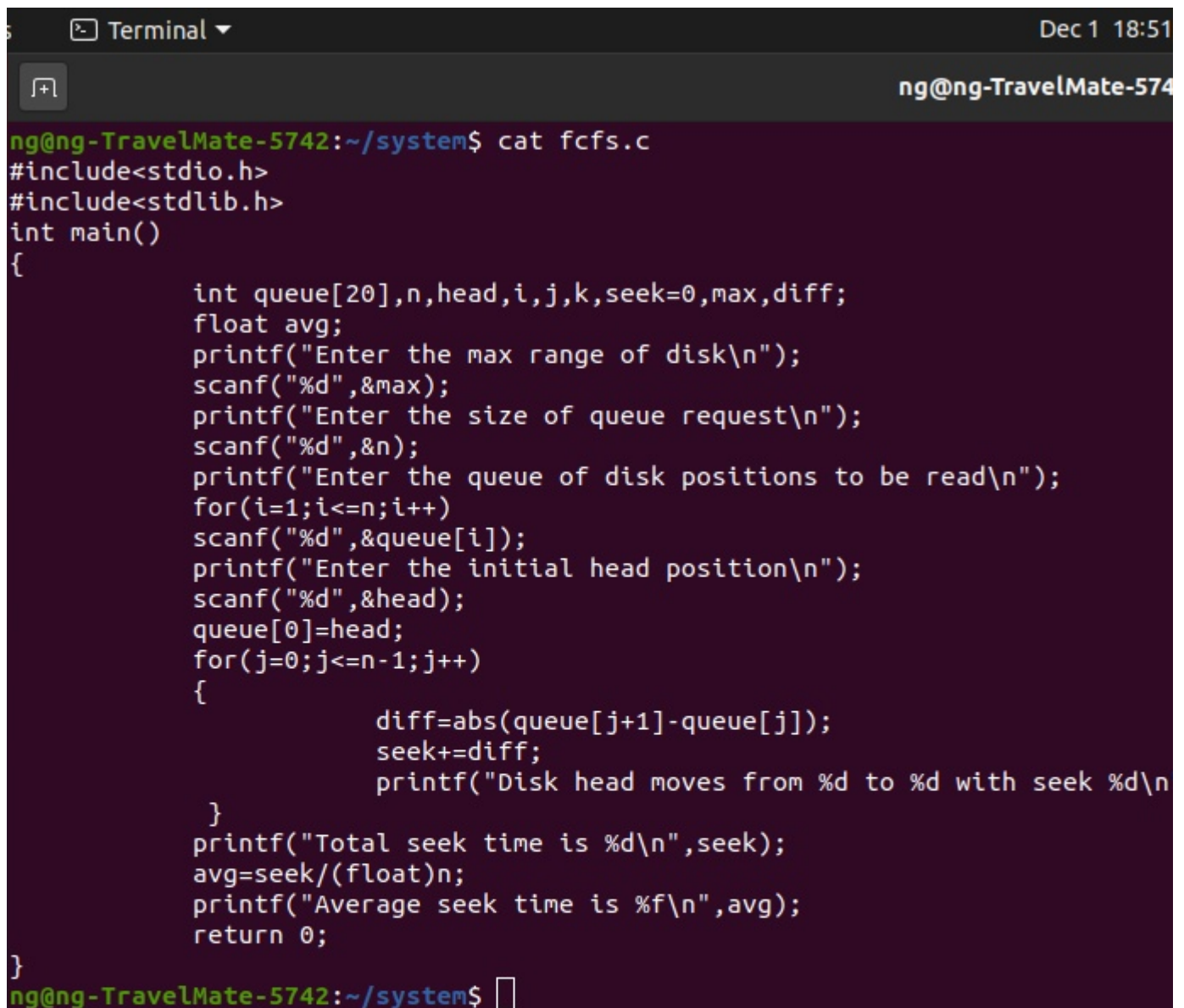
*AIM:*

Simulate the following disk scheduling algorithms.

- a) FCFS
- b) SCAN
- c) C-SCAN

*PROGRAM*

## a) FCFS



```
ng@ng-TravelMate-5742:~/system$ cat fcfs.c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int queue[20],n,head,i,j,k,seek=0,max,diff;
    float avg;
    printf("Enter the max range of disk\n");
    scanf("%d",&max);
    printf("Enter the size of queue request\n");
    scanf("%d",&n);
    printf("Enter the queue of disk positions to be read\n");
    for(i=1;i<=n;i++)
        scanf("%d",&queue[i]);
    printf("Enter the initial head position\n");
    scanf("%d",&head);
    queue[0]=head;
    for(j=0;j<=n-1;j++)
    {
        diff=abs(queue[j+1]-queue[j]);
        seek+=diff;
        printf("Disk head moves from %d to %d with seek %d\n",queue[j],queue[j+1],diff);
    }
    printf("Total seek time is %d\n",seek);
    avg=seek/(float)n;
    printf("Average seek time is %f\n",avg);
    return 0;
}
ng@ng-TravelMate-5742:~/system$
```

## OUTPUT

```
s Terminal ▾ Dec 1 18:50
ng@ng-TravelMate-574
ng@ng-TravelMate-5742:~/system$ gcc fcfs.c
ng@ng-TravelMate-5742:~/system$ ./a.out
Enter the max range of disk
200
Enter the size of queue request
8
Enter the queue of disk positions to be read
90 120 35 122 38 128 65 68
Enter the initial head position
50
Disk head moves from 50 to 90 with seek 40
Disk head moves from 90 to 120 with seek 30
Disk head moves from 120 to 35 with seek 85
Disk head moves from 35 to 122 with seek 87
Disk head moves from 122 to 38 with seek 84
Disk head moves from 38 to 128 with seek 90
Disk head moves from 128 to 65 with seek 63
Disk head moves from 65 to 68 with seek 3
Total seek time is 482
Average seek time is 60.250000
ng@ng-TravelMate-5742:~/system$
```

## PROGRAM

### b. SCAN

```
#include<stdio.h>
#include<stdlib.h>
void scan_algorithm(int left[], int right[], int count, int limit)
{
    int arr[20];
    int x = count - 1, y = count + 1, c = 0, d = 0, j;
    while(x > -1)
    {
        printf("\nX:\t%d", x);
        printf("\nLeft[X]:\t%d", left[x]);
        arr[d] = left[x];
        x--;
        d++;
    }
    arr[d] = 0;
    while(y < limit + 1)
    {
        arr[y] = right[c];
        c++;
        y++;
    }
    printf("\nScanning Order:\n");
    for(j = 0; j < limit + 1; j++)
    {
        printf("%d\n", arr[j]);
    }
}
```

```

    }
}

void division(int elements[], int limit, int disk_head)
{
    int count = 0, p, q, m, x;
    int left[20], right[20];
    for(count = 0; count < limit; count++)
    {
        if(elements[count] > disk_head)
        {
            printf("\nBreak Position:\t%d\n", elements[count]);
            break;
        }
    }
    printf("\nValue:\t%d\n", count);
    q = 1;
    p = 0;
    m = limit;
    left[0] = elements[0];
    printf("\nLeft:\t%d", left[0]);
    while(q < count)
    {
        printf("\nElement[l] value:\t%d" , elements[q]);
        left[q] = elements[q];
        printf("\nLeft:\t%d", left[q]);
        q++;
        printf("\nl:\t%d", q);
    }
    x = count;
    while(x < m)
    {
        right[p] = elements[x];
        printf("\nRight:\t%d", right[p]);
        printf("\nElement:\t%d", elements[x]);
        p++;
        x++;
    }
    scan_algorithm(left, right, count, limit);
}

void sorting(int elements[], int limit)
{
    int location, count, j, temp, small;
    for(count = 0; count < limit - 1; count++)
    {
        small = elements[count];
        location = count;
        for(j = count + 1; j < limit; j++)
        {
            if(small > elements[j])
            {
                small = elements[j];
                location = j;
            }
        }
    }
}

```



```

        temp = elements[location];
        elements[location] = elements[count];
        elements[count] = temp;
    }
}

int main()
{
    int count, disk_head, elements[20], limit;
    printf("Enter total number of locations:\t");
    scanf("%d", &limit);
    printf("\nEnter position of disk head:\t");
    scanf("%d", &disk_head);
    printf("\nEnter elements of disk head queue\n");
    for(count = 0; count < limit; count++)
    {
        printf("Element[%d]:\t", count + 1);
        scanf("%d", &elements[count]);
    }
    sorting(elements, limit);
    division(elements, limit, disk_head);
    return 0;
}

```

## OUTPUT

```

Enter total number of locations:      3

Enter position of disk head:      3

Enter elements of disk head queue
Element[1]:      1
Element[2]:      2
Element[3]:      3

Value:  3

Left:  1
Element[l] value:      2
Left:  2
l:      2
Element[l] value:      3
Left:  3
l:      3
X:      2
Left[X]:      3
X:      1
Left[X]:      2
X:      0
Left[X]:      1
Scanning Order:
3
2
1
0
ng@ng-TravelMate-5742:~/system$ 

```

# PROGRAM

## c) C-SCAN

```
ng@ng: /home/...$ cd /system; cat c-scan.c
#include<stdio.h>
int main()
{
    int queue[20],n,head,i,j,k,seek=0,max,diff,temp,queue1[20],queue2[20],temp1=0,temp2=0;
    float avg;
    printf("Enter the max range of disk\n");
    scanf("%d",&max);
    printf("Enter the initial head position\n");
    scanf("%d",&head);
    printf("Enter the size of queue request\n");
    scanf("%d",&n);
    printf("Enter the queue of disk positions to be read\n");
    for(i=1;i<=n;i++)
    {
        scanf("%d",&temp);
        if(temp>=head)
        {
            queue1[temp1]=temp;
            temp1++;
        }
        else
        {
            queue2[temp2]=temp;
            temp2++;
        }
    }
    for(i=0;i<temp1-1;i++)
    {
        for(j=i+1;j<temp1;j++)
        {
            if(queue1[i]>queue1[j])
            {
                temp=queue1[i];
                queue1[i]=queue1[j];
                queue1[j]=temp;
            }
        }
    }
    for(i=0;i<temp2-1;i++)
    {
        for(j=i+1;j<temp2;j++)
        {
            if(queue2[i]>queue2[j])
            {
                temp=queue2[i];
                queue2[i]=queue2[j];
                queue2[j]=temp;
            }
        }
    }
    for(i=0;i<temp1;i++)
    {
        printf("%d ",queue1[i]);
    }
    printf("\n");
    for(i=0;i<temp2;i++)
    {
        printf("%d ",queue2[i]);
    }
    printf("\n");
}
```

```

    for(i=1,j=0;j<temp1;i++,j++)
        queue[i]=queue1[j];
    queue[i]=max;
    queue[i+1]=0;
    for(i=temp1+3,j=0;j<temp2;i++,j++)
        queue[i]=queue2[j];
    queue[0]=head;
    for(j=0;j<=n+1;j++)
    {
        diff=abs(queue[j+1]-queue[j]);
        seek+=diff;
        printf("Disk head moves from %d to %d with seek %d\n",
            queue[j], queue[j+1], diff);
    }
    printf("Total seek time is %d\n",seek);
    avg=seek/(float)n;
    printf("Average seek time is %f\n",avg);
    return 0;
}

```

## OUTPUT

```

ng@ng-TravelMate-5742:~/system$ ./a.out
Enter the max range of disk
200
Enter the initial head position
50
Enter the size of queue request
8
Enter the queue of disk positions to be read
90 120 35 122 38 128 65 68
Disk head moves from 50 to 65 with seek 15
Disk head moves from 65 to 68 with seek 3
Disk head moves from 68 to 90 with seek 22
Disk head moves from 90 to 120 with seek 30
Disk head moves from 120 to 122 with seek 2
Disk head moves from 122 to 128 with seek 6
Disk head moves from 128 to 200 with seek 72
Disk head moves from 200 to 0 with seek 200
Disk head moves from 0 to 35 with seek 35
Disk head moves from 35 to 38 with seek 3
Total seek time is 388
Average seek time is 48.500000
ng@ng-TravelMate-5742:~/system$ 

```

## *RESULT*

Program is executed successfully and output is obtained.

BY NIVEA GIGEN  
S5 C  
CHN18CS092