

Experiment 9

Subject: CSL403 Operating System Lab

NAME: GINI CHACKO

ROLL: 8942

CLASS: SE COMPS B

Aim: Study Disk Management

Objectives: Demonstrate and analyze concepts of file management and I/O management techniques.

Problem Statement:

Implement Disk scheduling algorithms - FCFS, SCAN, C-SCAN

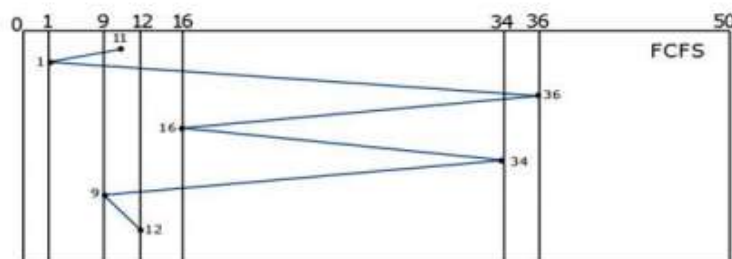
Given the tracks to be traversed by the disk head, find the order of traversal of cylinder.

Answer:

Consider an imaginary disk with 51 cylinders. A request comes in to read a block on cylinder 11. While the seek to cylinder 11 is in progress, new requests come in for cylinders 1, 36, 16, 34, 9, and 12, in that order. Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk scheduling Algorithms?

1. FCFS (First come first serve)
2. SCAN
3. C-SCAN

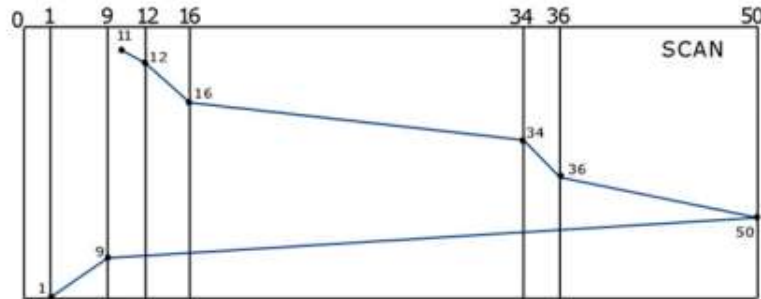
1. FCFS (First Come First Serve) Disk Scheduling:



- Here requests are served in the order of their arrival.
- In given example disk movement will be 11, 1, 36, 16, 34, 9 and 12 as first come first served.

- Total cylinder movement:
 $(11-1) + (36-1) + (36-16) + (34-16) + (34-9) + (9-12) = 111$ cylinders

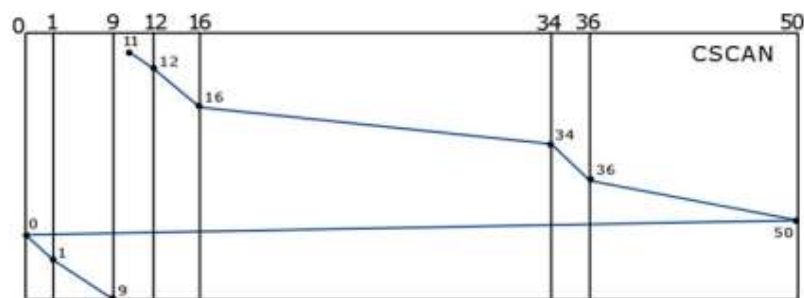
2. SCAN Disk Scheduling:



- From the current position disk arm starts in up direction and moves towards the end, serving all the pending requests until end.
- At that end arm direction is reversed (down) and moves towards the other end serving the pending requests on the way.
- As per SCAN request will be satisfied in order: 11, 12, 16, 34, 36, 50, 9, 1
- Total cylinder movement:

$$(12-11) + (16-12) + (34-16) + (36-34) + (50-36) + (50-9) + (9-1) = 88$$

3. C-SCAN Disk Scheduling:



- From the current position disk arm starts in up direction and moves towards the end, serving request until end.
- At the end the arm direction is reversed (down), and arm directly goes to other end and again continues moving in upward direction.
- As per SCAN request will be satisfied in order: 11, 12, 16, 34, 36, 50, 0, 1, 9
- Total cylinder movement:

$$(12-11) + (16-12) + (34-16) + (36-34) + (50-36) + (50-0) + (1-0) + (9-1) = 98$$

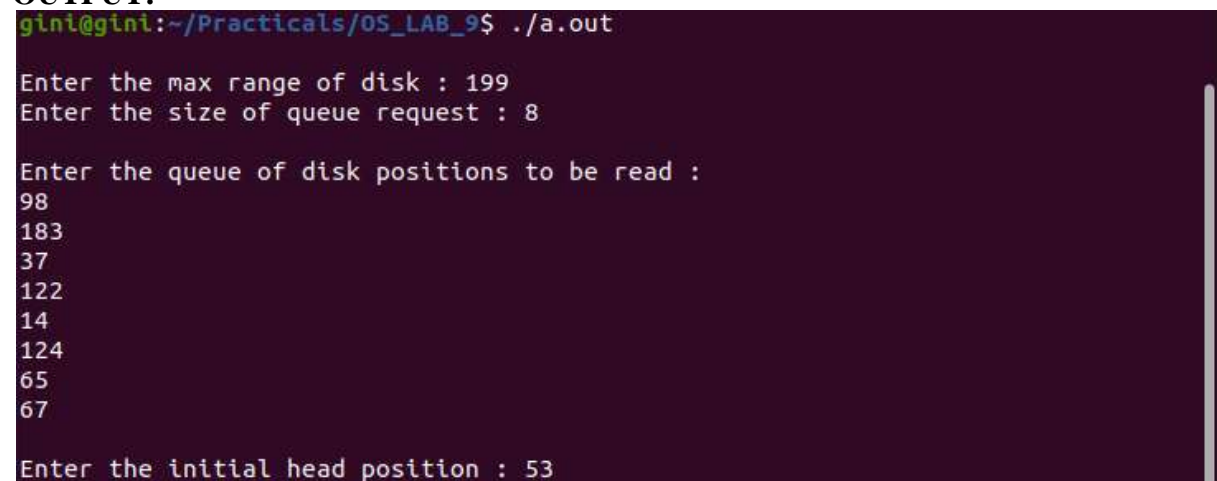
Program and Output Section:

A.] FCFS Disk Scheduling Algorithm

CODE:

```
#include<stdio.h>
int main()
{
    int queue[20],n,head,i,j,k,seek=0,max,diff;
    float avg;
    printf("\nEnter the max range of disk : ");
    scanf("%d",&max);
    printf("Enter the size of queue request : ");
    scanf("%d",&n);
    printf("\nEnter the queue of disk positions to be read : \n");
    for(i=1;i<=n;i++)
        scanf("%d",&queue[i]);
    printf("\nEnter the initial head position : ");
    scanf("%d",&head);
    queue[0]=head;
    printf("-----");
    for(j=0;j<=n-1;j++)
    {
        diff=abs(queue[j+1]-queue[j]);
        seek+=diff;
        printf("\nDisk head moves from %d to %d with seek
%d\n",queue[j],queue[j+1],diff);
    }
    printf("-----");
    printf("\nTotal seek time is %d\n",seek);
    avg=seek/(float)n;
    printf("-----");
    printf("\nAverage seek time is %f\n",avg);
    printf("-----");
    return 0;
}
```

OUTPUT:



```
gini@gini:~/Practicals/OS_LAB_9$ ./a.out
Enter the max range of disk : 199
Enter the size of queue request : 8
Enter the queue of disk positions to be read :
98
183
37
122
14
124
65
67
Enter the initial head position : 53
```

```

-----
Disk head moves from 53 to 98 with seek 45
Disk head moves from 98 to 183 with seek 85
Disk head moves from 183 to 37 with seek 146
Disk head moves from 37 to 122 with seek 85
Disk head moves from 122 to 14 with seek 108
Disk head moves from 14 to 124 with seek 110
Disk head moves from 124 to 65 with seek 59
Disk head moves from 65 to 67 with seek 2
-----
Total seek time is 640
-----
Average seek time is 80.000000
-----

```

B.] SCAN Disk Scheduling Algorithm

CODE:

```

#include<stdio.h>
int absoluteValue(int); // Declaring function absoluteValue

void main()
{
    int queue[25],n,headposition,i,j,k,seek=0, maxrange,
    difference,temp,queue1[20],queue2[20],temp1=0,temp2=0;
    float averageSeekTime;

    // Reading the maximum Range of the Disk.
    printf("Enter the maximum range of Disk : ");
    scanf("%d",&maxrange);

    // Reading the number of Queue Requests(Disk access requests)
    printf("Enter the number of queue requests : ");
    scanf("%d",&n);

    // Reading disk positions to be read in the order of arrival
    printf("\nEnter the queue of disk positions to be read : \n");
    for(i=1;i<=n;i++) // Note that i varies from 1 to n instead of 0 to n-1
    {
        scanf("%d",&temp); //Reading position value to a temporary variable

        //Now if the requested position is greater than current headposition,
        //then pushing that to array queue1
        if(temp>headposition)
        {
            queue1[temp1]=temp; //temp1 is the index variable of queue1[]

```

```

        temp1++; //incrementing temp1
    }
    else //else if temp < current headposition, then push to array queue2[]
    {
        queue2[temp2]=temp; //temp2 is the index variable of queue2[]
        temp2++;
    }
}

// Reading the initial head position.(ie. the starting point of execution)
printf("Enter the initial head position : ");
scanf("%d",&headposition);

//Now we have to sort the two arrays
//SORTING array queue1[] in ascending order
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;
        }
    }
}

//SORTING array queue2[] in descending order
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;
        }
    }
}

//Copying first array queue1[] into queue[]
for(i=1,j=0;j<temp1;i++,j++)
{
    queue[i]=queue1[j];
}

//Setting queue[i] to maxrange because the head goes to
//end of disk and comes back in scan Algorithm

```

```

queue[i]=maxrange;

//Copying second array queue2[] after that first one is copied, into queue[]
for(i=temp1+2,j=0;j<temp2;i++,j++)
{
    queue[i]=queue2[j];
}

//Setting queue[i] to 0. Because that is the innermost cylinder.
queue[i]=0;

//At this point, we have the queue[] with the requests in the
//correct order of execution as per scan algorithm.
//Now we have to set 0th index of queue[] to be the initial headposition.
queue[0]=headposition;

// Calculating SEEK TIME. seek is initially set to 0 in the declaration part.
printf("\n-----\n");
for(j=0; j<=n; j++) //Loop starts from headposition. (ie. 0th index of queue)
{
    // Finding the difference between next position and current position.
    difference = absoluteValue(queue[j+1]-queue[j]);

    // Adding difference to the current seek time value
    seek = seek + difference;

    // Displaying a message to show the movement of disk head
    printf("Disk head moves from position %d to %d with Seek %d \n", queue[j],
queue[j+1], difference);
}

// Calculating Average Seek time
averageSeekTime = seek/(float)n;

//Display Total and Average Seek Time(s)
printf("\n-----\n");
printf("Total Seek Time = %d\n", seek);
printf("\n-----\n");
printf("Average Seek Time = %f\n", averageSeekTime);
printf("\n-----\n");
}

// Defining function absoluteValue
int absoluteValue(int x)
{
    if(x>0)
    {
        return x;
    }
    else

```

```

    {
        return x*-1;
    }
}

```

OUTPUT:

```

gini@gini:~/Practicals/OS_LAB_9$ gcc scan.c
gini@gini:~/Practicals/OS_LAB_9$ ./a.out
Enter the maximum range of Disk : 199
Enter the number of queue requests : 8

Enter the queue of disk positions to be read :
98
183
37
122
14
124
65
67

Enter the initial head position : 53

-----
Disk head moves from position 53 to 14 with Seek 39
Disk head moves from position 14 to 37 with Seek 23
Disk head moves from position 37 to 65 with Seek 28
Disk head moves from position 65 to 67 with Seek 2
Disk head moves from position 67 to 98 with Seek 31
Disk head moves from position 98 to 122 with Seek 24
Disk head moves from position 122 to 124 with Seek 2
Disk head moves from position 124 to 183 with Seek 59
Disk head moves from position 183 to 199 with Seek 16

-----
Total Seek Time = 224

-----
Average Seek Time = 28.000000
-----

```

C.] C-SCAN Disk Scheduling Algorithm

CODE:

```

#include<stdio.h>
int absoluteValue(int); // Declaring function absoluteValue

void main()
{
    int queue[25],n,headposition,i,j,k,seek=0, maxrange,
    difference,temp,queue1[20],queue2[20],temp1=0,temp2=0;
    float averageSeekTime;

    // Reading the maximum Range of the Disk.
    printf("Enter the max range of Disk: ");
    scanf("%d",&maxrange);

```

```

// Reading the number of Queue Requests(Disk access requests)
printf("Enter the size of queue requests: ");
scanf("%d",&n);

// Reading disk positions to be read in the order of arrival
printf("\nEnter the queue of disk positions to be read : \n");
for(i=1;i<=n;i++) // Note that i varies from 1 to n instead of 0 to n-1
{
    scanf("%d",&temp); //Reading position value to a temporary variable

    //Now if the requested position is greater than current headposition,
    //then pushing that to array queue1
    if(temp>headposition)
    {
        queue1[temp1]=temp; //temp1 is the index variable of queue1[]
        temp1++; //incrementing temp1
    }
    else //else if temp < current headposition,then push to array queue2[]
    {
        queue2[temp2]=temp; //temp2 is the index variable of queue2[]
        temp2++;
    }
}

// Reading the initial head position.(ie. the starting point of execution)
printf("Enter the initial head position: ");
scanf("%d",&headposition);

//Now we have to sort the two arrays
//SORTING array queue1[] in ascending order
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;
        }
    }
}

//SORTING array queue2[] in ascending order
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;
    }
}

//Copying first array queue1[] into queue[]
for(i=1,j=0;j<temp1;i++,j++)
{
    queue[i]=queue1[j];
}

//Setting queue[i] to maxrange because the head goes to
//end of disk and comes back in scan Algorithm
queue[i]=maxrange;

//Moving Disk head to the inner most cylinder, because this is Circular Scan.
queue[i+1]=0;

//Copying second array queue2[] after that first one is copied, into queue[]
for(i=temp1+3,j=0;j<temp2;i++,j++)
{
    queue[i]=queue2[j];
}

//At this point, we have the queue[] with the requests in the
//correct order of execution as per C-SCAN algorithm.
//Now we have to set 0th index of queue[] to be the initial headposition.
queue[0]=headposition;

// Calculating SEEK TIME. seek is initially set to 0 in the declaration part.
printf("-----");
for(j=0; j<=n+1; j++) //Loop starts from headposition. (ie. 0th index of queue)
{
    // Finding the difference between next position and current position.
    difference = absoluteValue(queue[j+1]-queue[j]);

    // Adding difference to the current seek time value
    seek = seek + difference;

    // Displaying a message to show the movement of disk head
    printf("Disk head moves from position %d to %d with Seek %d \n",
        queue[j], queue[j+1], difference);
}

// Calculating Average Seek time
averageSeekTime = seek/(float)n;

//Display Total and Average Seek Time(s)

```

```

printf("-----");
printf("Total Seek Time= %d\n", seek);
printf("-----");
printf("Average Seek Time= %f\n", averageSeekTime);
printf("-----");
}

// Defining function absoluteValue
int absoluteValue(int x)
{
    if(x>0)
    {
        return x;
    }
    else
    {
        return x*-1;
    }
}

```

OUTPUT:

```

gini@gini:~/Practicals/OS_LAB_9$ gcc cscan.c
gini@gini:~/Practicals/OS_LAB_9$ ./a.out
Enter the max range of Disk : 199
Enter the size of queue requests : 8

Enter the queue of disk positions to be read :
98
183
37
122
14
124
65
67

Enter the initial head position : 53
-----
Disk head moves from position 53 to 14 with Seek 39
Disk head moves from position 14 to 37 with Seek 23
Disk head moves from position 37 to 65 with Seek 28
Disk head moves from position 65 to 67 with Seek 2
Disk head moves from position 67 to 98 with Seek 31
Disk head moves from position 98 to 122 with Seek 24
Disk head moves from position 122 to 124 with Seek 2
Disk head moves from position 124 to 183 with Seek 59
Disk head moves from position 183 to 199 with Seek 16
Disk head moves from position 199 to 0 with Seek 199
-----
Total Seek Time= 423
-----
Average Seek Time= 52.875000
-----

```