Experiment 10: Disk Scheduling

Aim: To implement Disk Scheduling Algorithms (FCFS, STF, SCAN, C-LOOK)

Theory: Disk scheduling is schedule I/O requests arriving for the disk. It is important because:-

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk head movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

There are some important point in Disk scheduling:-

- **Seek Time:** Seek time is the time taken to locate the disk head to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- **Disk Access Time:** Disk Access Time is=(Seek+ Rotational+ transfer time)

There are some Disk scheduling algorithms:-

FCFS:

Code:

```
#include<stdio.h>
#include<stdlib.h>

void main()
{
   int rq[20], dist=0, min,n,initial,i;
   printf("enter initial position: ");
   scanf("%d",&initial);
   printf("enter number of requests: ");
   scanf("%d",&n);
   printf("enter requests: ");
   for(i=0;i<n;i++){
      scanf("%d",&rq[i]);
   }
   for(i=0;i<n;i++)
   {
      dist=dist+abs(rq[i]-initial);
      initial=rq[i];
   }
   printf("Total head moment is %d",dist);
   printf("\nRequests resolved in following order: ");
   for(i=0;i<n;i++){
      printf("%d\t",rq[i]);
   }
}
```

Output:

```
enter initial position: 50
enter number of requests: 8
enter requests: 95 180 34 119 11 123 62 64
Total head moment is 644
Requests resolved in following order: 95        180     34      119     11      123     62      64
```

SSTF:

Code:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
   int rq[20], dist=0, min,n,initial,i,count=0,r[20];
   printf("enter initial position: ");
   scanf("%d",&initial);
   printf("enter number of requests: ");
   scanf("%d",&n);
   printf("enter requests: ");
   for(i=0;i<n;i++){
      scanf("%d",&rq[i]);
      r[i]=0;
   }
   while(count!=n)
   {
      int min=1000,d,index;
      for(i=0;i<n;i++)
      {
        d=abs(rq[i]-initial);
        if(min>d)
        {
           min=d;
           index=i;
           r[count] = rq[i];
        }
      }
      dist=dist+min;
      initial=rq[index];
      rq[index]=1000;
      count++;
   }
   printf("Total head movement is %d",dist);
   printf("\nRequests resolved in following order: ");
   for(i=0;i<n;i++){
      printf("%d\t",r[i]);
   }
}
```

Output:

```
enter initial position: 50
enter number of requests: 8
enter requests: 95 180 34 119 11 123 62 64
Total head movement is 236
Requests resolved in following order: 62        64      34      11      95      119     123     180
```

Scan:

Code:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
   int rq[20], r[20], dist=0, min,n,initial,i,j,size,count=0,index,temp;
   printf("enter initial position: ");
   scanf("%d",&initial);
    printf("Enter total disk size: ");
   scanf("%d",&size);
   printf("enter number of requests: ");
   scanf("%d",&n);
   printf("enter requests: ");
   for(i=0;i<n;i++){
      scanf("%d",&rq[i]);
   }
    for(i=0;i<n;i++)
   {
      for(j=0;j<n-i-1;j++)
      {
         if(rq[j]>rq[j+1])
         {
            temp=rq[j];
            rq[j]=rq[j+1];
            rq[j+1]=temp;
         }
      }
   }
   for(i=0;i<n;i++)
   {
      if(initial<rq[i])
      {
         index=i;
         break;
```

```
    }
  }

    for(i=index;i<n;i++)
    {
       dist=dist+abs(rq[i]-initial);
       initial=rq[i];
       r[count] = rq[i];
       count++;
    }
    //  last movement for max size
    dist=dist+abs(size-rq[i-1]-1);
    initial = size-1;
    r[count] = size - 1;
    count++;
    for(i=index-1;i>=0;i--)
    {
        dist=dist+abs(rq[i]-initial);
        initial=rq[i];
        r[count] = rq[i];
        count++;
    }

  printf("Total head moment is %d",dist);
  printf("\nRequests resolved in following order: ");
  for(i=0;i<n;i++){
    printf("%d\t",rq[i]);
  }
}
```

Output:

```
enter initial position: 50
Enter total disk size: 200
enter number of requests: 8
enter requests: 95 180 34 119 11 123 62 64
Total head moment is 337
Requests resolved in following order: 11        34      62      64      95      119     123     180
```

C-LOOK:

Code:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
   int rq[20], r[20], dist=0, min,n,initial,i,j,size,count=0,index,temp;
   printf("enter initial position: ");
   scanf("%d",&initial);
    printf("Enter total disk size: ");
   scanf("%d",&size);
   printf("enter number of requests: ");
   scanf("%d",&n);
   printf("enter requests: ");
   for(i=0;i<n;i++){
      scanf("%d",&rq[i]);
   }
    for(i=0;i<n;i++)
   {
      for(j=0;j<n-i-1;j++)
      {
         if(rq[j]>rq[j+1])
         {
            temp=rq[j];
            rq[j]=rq[j+1];
            rq[j+1]=temp;
         }

      }
   }
   for(i=0;i<n;i++)
   {
      if(initial<rq[i])
      {
         index=i;
         break;
      }
   }

      for(i=index;i<n;i++)
```

```
    {
       dist=dist+abs(rq[i]-initial);
       initial=rq[i];
       r[count] = rq[i];
       count++;
    }
    for(i=0;i<index;i++)
    {
        dist=dist+abs(rq[i]-initial);
        initial=rq[i];
        r[count] = rq[i];
        count++;
    }

  printf("Total head moment is %d",dist);
  printf("\nRequests resolved in following order: ");
  for(i=0;i<n;i++){
     printf("%d\t",rq[i]);
  }
}
```

Output:

```
enter initial position: 50
Enter total disk size: 200
enter number of requests: 8
enter requests: 95 180 34 119 11 123 62 64
Total head moment is 322
Requests resolved in following order: 11        34      62      64      95      119     123     180
```

Conclusion: Thus we have successfully implemented some Disk Scheduling Algorithms.