priority than user processes. Consider each process priority to be from 1 to 3. Use priority scheduling for the processes in each queue

EX.NO: 14 DISK SCHEDULING ALGORITHMS

AIM:

Write a C program to simulate disk scheduling algorithms

a) FCFS b) SCAN c) C-SCAN

DESCRIPTION

One of the responsibilities of the operating system is to use the hardware efficiently. For the disk drives, meeting this responsibility entails having fast access time and large disk bandwidth. Both the access time and the bandwidth can be improved by managing the order in which disk I/O requests are serviced which is called as disk scheduling. The simplest form of disk scheduling is, of course, the first-come, first-served (FCFS) algorithm. This algorithm is intrinsically fair, but it generally does not provide the fastest service. In the SCAN algorithm, the disk arm starts at one end, and moves towards the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk. C-SCAN is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip

PROGRAM

FCFS DISK SCHEDULING ALGORITHM

```
#include<stdio.h>
 main()
{
        int t[20], n, I, j, tohm[20], tot=0;
        float avhm;
        clrscr();
        printf("enter the no.of tracks");
        scanf("%d",&n);
        printf("enter the tracks to be traversed");
        for(i=2;i<n+2;i++)
```

```
            scanf("%d",&t*i+);
        for(i=1;i<n+1;i++)
        {
            tohm[i]=t[i+1]-t[i];
            if(tohm[i]<0)
                tohm[i]=tohm[i]*(-1);
        }
        for(i=1;i<n+1;i++)
            tot+=tohm[i];
        avhm=(float)tot/n;
        printf("Tracks traversed\tDifference between tracks\n"); for(i=1;i<n+1;i++)
            printf("%d\t\t\t%d\n",t*i+,tohm*i+);
         printf("\nAverage header movements:%f",avhm);
        getch();
}
```

OUTPUT

Enter no.of tracks:9

Enter track position:55 58 60 70 18 90 150 160 184

Tracks traversed	Difference between tracks
55	45
58	3
60	2
70	10
18	52
90	72
150	60
160	10
184	24

Average header movements:30.888889

SCAN DISK SCHEDULING ALGORITHM

```
#include<stdio.h>
 main()
{
        int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0;
        clrscr();
        printf("enter the no of tracks to be traveresed");
        scanf("%d'",&n);
        printf("enter the position of head");
         scanf("%d",&h);
        t[0]=0;t[1]=h;
        printf("enter the tracks");
        for(i=2;i<n+2;i++)
                scanf("%d",&t[i]);
        for(i=0;i<n+2;i++)
        {
                for(j=0;j<(n+2)-i-1;j++)
                {
                        if(t[j]>t[j+1])
                        {
                                temp=t[j];
                                t[j]=t[j+1];
                                t[j+1]=temp;
                        }
                }
        }
        for(i=0;i<n+2;i++)
                if(t[i]==h)
```

```
                    j=i;k=i;
        p=0;
        while(t[j]!=0)
        {
                atr[p]=t[j];
                 j--;
                p++;
        }
        atr[p]=t[j];
         for(p=k+1;p<n+2;p++,k++)
                atr[p]=t[k+1];
        for(j=0;j<n+1;j++)
        {
                if(atr[j]>atr[j+1])
                        d[j]=atr[j]-atr[j+1];
                else
                        d[j]=atr[j+1]-atr[j];
                sum+=d[j];
        }
        printf("\nAverage header movements:%f",(float)sum/n);
        getch();
}
```

OUTPUT

```
Enter no.of tracks:9
Enter track position:55   58   60   70   18   90   150   160   184
Tracks traversed    Difference between tracks
150    50
160    10
1841   24
90     94
```

70 20

60 10

58 2

55 3

18 37

Average header movements: 27.77

C-SCAN DISK SCHEDULING ALGORITHM

```
#include<stdio.h>
main()
{
        int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0;
        clrscr();
        printf("enter the no of tracks to be traveresed");
        scanf("%d'",&n);
        printf("enter the position of head");
        scanf("%d",&h);
        t[0]=0;t[1]=h;
        printf("enter total tracks");
        scanf("%d",&tot);
        t[2]=tot-1;
        printf("enter the tracks");
        for(i=3;i<=n+2;i++)
                scanf("%d",&t[i]);
        for(i=0;i<=n+2;i++)
                for(j=0;j<=(n+2)-i-1;j++)
                        if(t[j]>t[j+1])
                        {
                                temp=t[j];
                                t[j]=t[j+1];
```

```
                        t[j+1]=temp;
                }
for(i=0;i<=n+2;i++)
        if(t[i]==h)
                j=i;break;
 p=0;
while(t[j]!=tot-1)
{
        atr[p]=t[j];
        j++;
        p++;
}
atr[p]=t[j];
 p++;
i=0;
while(p!=(n+3) && t[i]!=t[h])
{
        atr[p]=t[i];
         i++;
        p++;
}
for(j=0;j<n+2;j++)
{
if(atr[j]>atr[j+1])
        d[j]=atr[j]-atr[j+1];
else
        d[j]=atr[j+1]-atr[j];
sum+=d[j];
}
printf("total header movements%d",sum);
```

```
    printf("avg is %f",(float)sum/n);
    getch();
}
```

OUTPUT

Enter the track position: 55 58 60 70 18 90 150 160 184

Enter starting position : 100

Tracks traversed	Difference Between tracks
150	50
160	10
184	24
18	240
55	37
58	3
60	2
70	10
90	29

Average seek time : 35.7777779

RESULT:

Thus the program to implement disk Scheduling algorithm has been executed and verified

VIVA QUESTIONS:

1. What is disk scheduling?
2. List the different disk scheduling algorithms?
3. Define the terms – disk seek time, disk access time and rotational latency?
4. Define sequential file allocation?
5. What is the use of indexed file allocation?
6. What is the advantages if linked allocation?
7. What is the advantage of C-SCAN algorithm over SCAN algorithm?
8. Which disk scheduling algorithm has highest rotational latency? Why?