Lab 9: Disk Management Systems

System calls related to disk

df command:

The **'df'** (Disk Free) command in an inbuilt utility to find the available and the disk usage space on Linux servers/storage.

The following table provides an overview of the options of df command in Linux.

Options	Description
-a	-all: includes real files and virtual files like pseudo,pro, sysfs. lxc
-h	It prints the sizes in the human-readable format in power of 1024 (eg: 1K 1M 1G)
-H	--si : likewise '-h' but here in power of 1000
-i	-- inodes: correspondence the inode details
-k	--block-size=1K display the disk space
-l	--local display local file systems only
-m	--megabytes display the disk space
-t	--type=TYPE To filter a particular file system type
-T'	--print-type List the file system types
-x	--exclude-type=TYPE To exclude a particular file system type

1. **How to check the details of disk space used in each file system?**

```
# df
```

```
Output:

Filesystem     1K-blocks      Used Available Use% Mounted on
/dev/sda2      164962420 142892148  15301196  91% /
udev               10240         0     10240   0% /dev
tmpfs            3291620    329084   2962536  10% /run
tmpfs            8229048         0   8229048   0% /dev/shm
tmpfs               5120         0      5120   0% /run/lock
tmpfs            8229048         0   8229048   0% /sys/fs/cgroup
/dev/sda1          97167     76552     15598  84% /boot
tmpfs            1645812         0   1645812   0% /run/user/301703
tmpfs            1645812         0   1645812   0% /run/user/301677
tmpfs            1645812         0   1645812   0% /run/user/301483
```

Note: Using 'df' command without any option/parameter will display all the partitions and the usage information of the disk space. The result of the above command contains 6 columns which are explained here below:

```
Filesystem      -->   Mount Point Name
1K-blocks       -->   Available total space counted in 1kB (1000 bytes)
Used            -->   Used block size
Available       -->   Free blocks size
Use%            -->   Usage on percentage-wise
Mounted on      -->   Show the path of the mounted point

# df -k
```

Note: Even using the '-k' option also provides the same output as the default 'df' command. Both outputs provide the same data usage of file systems in block size which is measured in 1024 bytes.

2. **How to sum up the total of the disk space usage?**

```
# df -h --total
```

```
Output:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sda2       158G  137G   15G  91% /
udev             10M     0   10M   0% /dev
tmpfs           3.2G  322M  2.9G  11% /run
tmpfs           7.9G     0  7.9G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           7.9G     0  7.9G   0% /sys/fs/cgroup
/dev/sda1        95M   75M   16M  84% /boot
tmpfs           1.6G     0  1.6G   0% /run/user/301703
tmpfs           1.6G     0  1.6G   0% /run/user/301483
tmpfs           1.6G     0  1.6G   0% /run/user/301613
tmpfs           1.6G     0  1.6G   0% /run/user/301677
total           183G  137G   40G  78% -
```

Note: using '--total' along with '-h' will sum up the total disk usage of all the file systems

3. **How to list the inodes information of all file systems?**

```
# df -i
```

```
 Output:

Filesystem       Inodes   IUsed   IFree IUse% Mounted on
/dev/sda2      10240000 2491518 7748482   25% /
udev            2054985     305 2054680    1% /dev
tmpfs           2057262     767 2056495    1% /run
tmpfs           2057262       1 2057261    1% /dev/shm
tmpfs           2057262      11 2057251    1% /run/lock
tmpfs           2057262      13 2057249    1% /sys/fs/cgroup
/dev/sda1         25168     336   24832    2% /boot
tmpfs           2057262       4 2057258    1% /run/user/301703
tmpfs           2057262       4 2057258    1% /run/user/301483
tmpfs           2057262       4 2057258    1% /run/user/301613
tmpfs           2057262       4 2057258    1% /run/user/301677
```

Note: Using '-i' will list the information about the Inodes of all the filesystem.

Disk scheduling algorithms for the disk structure are used by operating systems to determine the order in which read and write operations are performed on a disk. The main goal of these algorithms is to minimize the disk head movements and optimize the overall disk performance.

Here are some common disk scheduling algorithms:

1. **First-Come, First-Served (FCFS):** In this algorithm, the disk requests are executed in the order they arrive. The disk head moves from its current position to the requested track, serving the requests sequentially. FCFS is simple but can result in poor performance due to the phenomenon called the "elevator effect."

2. **Shortest Seek Time First (SSTF):** This algorithm selects the request that requires the least disk head movement from the current position. It minimizes the seek time by serving the closest request first. SSTF provides better performance compared to FCFS, but it may cause starvation for requests located farther from the current position.

3. **SCAN:** Also known as the elevator algorithm, SCAN moves the disk head in one direction, serving requests along the way until it reaches the end of the disk. Then, it changes direction and serves requests in the opposite direction. SCAN provides a fair servicing order for all requests but may cause delays for requests located at the extremes.

4. **C-SCAN:** Similar to SCAN, C-SCAN moves the disk head in one direction, serving requests until the end of the disk is reached. However, instead of changing direction, it immediately jumps to the opposite end and starts servicing requests from there. This algorithm avoids the delays caused by SCAN for requests located at the extremes.

5. **LOOK:** LOOK is an improvement over SCAN. Instead of moving to the end of the disk, LOOK changes direction as soon as there are no pending requests in the current direction. This reduces the head movement and improves the average seek time.

6. **C-LOOK:** C-LOOK combines the advantages of C-SCAN and LOOK. It moves the disk head in one direction, serving requests until the last request in that direction. Then, it jumps to the

opposite end and starts servicing requests from there without reversing direction. C-LOOK reduces head movement and provides better performance than both SCAN and LOOK.

The First-Come, First-Served (FCFS) algorithm is one of the simplest disk scheduling algorithms. It operates on the principle that requests are serviced in the order they arrive. In the context of disk scheduling, FCFS refers to the order in which read and write requests are executed on the disk.

Sample Programming:

Simulating FCFS Disk Scheduling algorithm

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int queue[100],n,head,i,j,k,seek=0,diff;
        float avg;
        // clrscr();
        printf("*** FCFS Disk Scheduling Algorithm ***\n");
        printf("Enter the size of Queue\t");
        scanf("%d",&n);
        printf("Enter the Queue\t");
        for(i=1;i<=n;i++)
        {
                scanf("%d",&queue[i]);
        }
       printf("Enter the initial head position\t");
       scanf("%d",&head);
        queue[0]=head;
       printf("\n");
        for(j=0;j<=n-1;j++)
        {
                diff=abs(queue[j+1]-queue[j]);
                seek+=diff;
                printf("Move from %d to %d with Seek %d\n",queue[j],queue[j+1],diff);
        }
        printf("\nTotal Seek Time is %d\t",seek);
        avg=seek/(float)n;
       printf("\nAverage Seek Time is %f\t",avg);
        getch();
}
```

Output

```
*** FCFS Disk Scheduling Algorithm ***
Enter the size of Queue 8
Enter the Queue 98 183 37 122 14 124 65 67
Enter the initial head position 53

Move from 53 to 98 with Seek 45
Move from 98 to 183 with Seek 85
Move from 183 to 37 with Seek 146
Move from 37 to 122 with Seek 85
Move from 122 to 14 with Seek 108
Move from 14 to 124 with Seek 110
Move from 124 to 65 with Seek 59
Move from 65 to 67 with Seek 2

Total Seek Time is 640
Average Seek Time is 80.000000
```

Lab Exercises:

1. Write a multithreaded program that implements the banker's algorithm. Create n threads that request and release resources from the bank. The banker will grant the request only if it leaves the system in a safe state. You may write this program using pthreads. It is important that shared data be safe from concurrent access. To ensure safe access to shared data, you can use mutex locks, which are available in the pthreads libraries.

2. Simulate implementation of Disk Scheduling Algorithms: FCFS, SSTF using a structure DSA. An DSA contains the request ID, arrival timestamp, cylinder, address, and the ID of the process that posted the request.

```
struct DSA
 {
int request_id;
int arrival_time_stamp;
int cylinder;
int address;
int process_id;
}
```

3. A file system uses contiguous allocation of disk space to files. A few blocks on the disk are reserved as spare blocks. If some disk blocks is found to be bad, the file system allocates a spare disk block to it and notes the address of the bad block and its allocated spare block in a “bad blocks table”. This table is consulted while accessing the disk block. Simulate the same.

4. Display the list of devices connected to your system including the physical names and its instance number.