

MC 214 LAB-8 REPORT

Nisarg Suthar

202003030

[23 Nov 2021]

EX-1: Instead of using one, two, or three-level indexed allocation schemes, Linux UFS uses a combined Indexed allocation scheme as shown below.

Assume that there are 10 direct pointers to data blocks, and 1 each of indirect, double indirect, and triple indirect pointers (a total of 13 pointers). There are 1K Bytes per block and block addresses are 32bit in size. Indirect pointers point to inode blocks also of 1KB size.

1. Compute the maximum size of the files that can be created using this structure.
2. For the same number of pointers (13) in the inode, compute the maximum file size for one, two, and three-level schemes.
3. Assuming the word read time of 10ms per 1KB, and seek time is 30ms(read pointer block) what is the time required to read files of sizes 4KB, 512KB, 16 MB, 1GB, and 512 GB respectively for all the above schemes.

ANSWER:

1. **Size of block** = 1 KB = 1024 bytes.

Size of address = 32 bits = 4bytes.

So, we can store $\frac{1024}{4} = 256$ addresses on one block.

So, the single indirect pointer points to a block of 256 pointers which point to data blocks. Similarly, the double indirect pointer points to a block of 256 pointers which in turn point to blocks of 256 pointers which point to data blocks. And similar for the triple indirect pointer.

Hence, the maximum size of file that can be created using this structure is:

$$\begin{aligned} & (No\ of\ DP + 256 * No\ of\ SIP + 256^2 * No\ of\ DIP + 256^3 * No\ of\ TIP) * block\ size \\ &= (10 + 256 * 1 + 65536 * 1 + 16777216 * 1) * 1\ KB \\ &= \mathbf{16843018\ KB} \\ &\approx 16.062\ GB \end{aligned}$$

2. (a) For **one-level** scheme having 13 pointers:

$$Max\ file\ size = 13 * Block\ Size$$

$$= 13 * 1\ KB = \mathbf{13\ KB}$$

- (b) For **two-level** scheme having 13 pointers:

$$\text{Max file size} = 13 * 256 * \text{block size}$$

$$= 13 * 256 * 1 \text{ KB} = \mathbf{3328 \text{ KB}}$$

(c) For **three-level** scheme having 13 pointers:

$$\text{Max file size} = 13 * 256^2 * 1 \text{ KB} = \mathbf{851968 \text{ KB}}$$

(d) For **four-level** scheme having 13 pointers:

$$\text{Max file size} = 13 * 256^3 * 1 \text{ KB} = \mathbf{2181038080 \text{ KB}}$$

3. TIME TAKEN TO READ FILES:

10 milliseconds per 1KB

Seek time = 30 milliseconds

Assumption: Reading the inode takes almost zero time.

(1) One-level scheme:

(a) 4 KB

Here we need to seek the required 4 blocks of 1 KB memory each and read them.

We directly get the address of blocks from the inode.

$$\therefore \text{Time taken} = (30 * 4 + 4 * 10) = 120 + 40 = 160 \text{ milliseconds.}$$

(b) 512 KB

Out of bounds.

(c) 16 MB

Out of bounds.

(d) 1 GB

Out of bounds.

(e) 512 GB

Out of bounds.

(2) Two-level scheme:

(a) 4KB

Here we seek a pointer block from the inode and then read it. Then we seek the required 4 data blocks and read them.

$$\therefore \text{Time taken} = (30 + 10 + 30 * 4 + 10 * 4) = 200 \text{ milliseconds.}$$

(b) 512 KB

$$\therefore \text{Time taken} = (30 + 30 * 256 + 256 * 10) * 2 = 20540 \text{ milliseconds.}$$

(c) 16 MB

$$16 \text{ MB} = 16 * 1024 \text{ KB} = 16384 \text{ KB}$$

\therefore Not possible.

(d) 1 GB

Out of bounds.

(e) 512 GB

Out of bounds.

(3) Three-level scheme:

(a) 4 KB

\therefore Time taken = $(30 + 10 + 30 + 10 + 30*4 + 10*4) = 240$ milliseconds.

(b) 512 KB

\therefore Time taken = $(30 + 10 + 30 + 30 + 10 + 10 + 30*256 + 30*256 + 10*256 + 10*256) = 206000$ milliseconds.

(c) 16 MB

16 MB = 16384 KB = $256*64$ KB.

\therefore Time taken = $(30 + 10 + 30*64 + 10*64 + 40 * 16384) = 657960$ ms.

(d) 1 GB

1 GB = 1024 MB = 1048576 KB.

\therefore *Not possible.*

(e) 512 GB

Out of bounds.

(4) Four-level scheme:

(a) 4 KB

\therefore Time taken = $(40 + 40 + 40 + 40*4) = 280$ milliseconds.

(b) 512 KB

\therefore Time taken = $(40 + 40 + 40*2 + 40 * 512) = 20640$ milliseconds.

(c) 16 MB

\therefore Time taken = $(40 + 40 + 40*64 + 40*16384) = 658000$ milliseconds.

(d) 1 GB

1 GB = 1048576 KB = $256*4096$ KB = $16*256*256$ KB.

\therefore Time taken = $(40 + 40*16 + 40*16*256 + 1048576*40) = 10650280$ ms.

(e) 512 GB

512 GB = $512 * 1$ GB = $512*16*256*256$ KB = $32*256*256*256$ KB

\therefore *Not possible.*

➤ The time taken for the Linux UFS is attached in the document below:

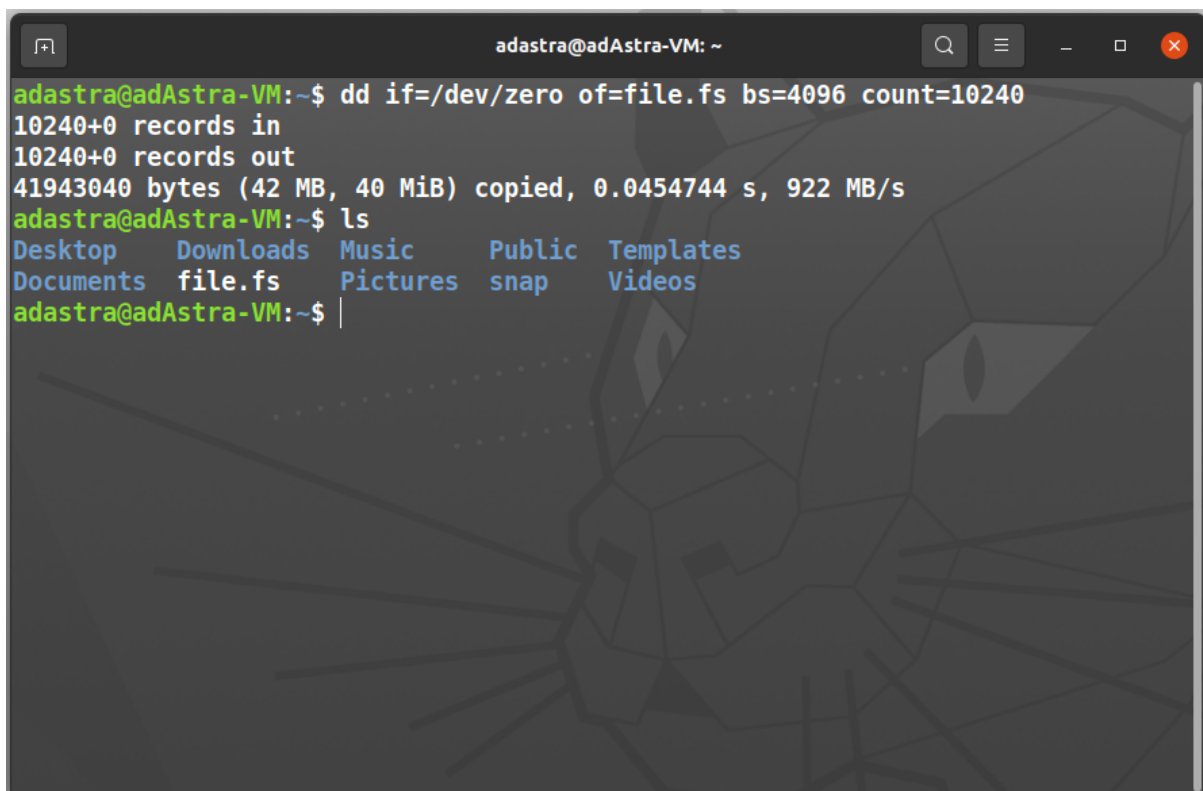
<https://docs.google.com/document/d/1FZU9hqYorRtGw5c1yRiulPW5krxkZSeMMbHT7GuCnwo/edit?usp=sharing>

EX-2: Implement a file-based file system using the commands explained.
(Attach screenshots of every command executed in the report)

- The file system should be of size 40 MB with 4096 block-size.
- Mount point: /mnt/tmp
- Create a few simple text files in this file system and list out the files.
- Display the stat of any one file (use stat command).
- Unmount the file system and list out the content under the mount point directory.

Here we are creating a file base file system.

First, we create a zero filled file with a specific size using **dd** command. Here we provide the input file argument as /dev/zero which creates a zero filled file. The output file argument is file.fs which is the file we will use to create the file system. Since we need the block size of 4 KB and the file system to be of 40 MB we provide the block size argument as 4096 and the count argument as 10240. (40 MiB = 40 * 1024 * 1024 bytes = 4096 * 10240 bytes).



```
adastra@adAstra-VM: ~  
adastra@adAstra-VM:~$ dd if=/dev/zero of=file.fs bs=4096 count=10240  
10240+0 records in  
10240+0 records out  
41943040 bytes (42 MB, 40 MiB) copied, 0.0454744 s, 922 MB/s  
adastra@adAstra-VM:~$ ls  
Desktop  Downloads  Music      Public    Templates  
Documents file.fs    Pictures  snap      Videos  
adastra@adAstra-VM:~$ |
```

After execution of any of the above command we will now have **file.fs** file available in our current working directory.

Next, we create a filesystem using **mkfs** command. Here I have used ext4 file system.

```
adastra@adAstra-VM: ~  
adastra@adAstra-VM:~$ mkfs.ext4 file.fs  
mke2fs 1.45.5 (07-Jan-2020)  
Discarding device blocks: done  
Creating filesystem with 10240 4k blocks and 10240 inodes  
  
Allocating group tables: done  
Writing inode tables: done  
Creating journal (1024 blocks): done  
Writing superblocks and filesystem accounting information: done  
  
adastra@adAstra-VM:~$ |
```

Now we create a mount point **/mnt/tmp** and mount our new filesystem.

```
adastra@adAstra-VM: ~  
adastra@adAstra-VM:~$ sudo mkdir /mnt/tmp  
adastra@adAstra-VM:~$ |
```

```
adastra@adAstra-VM: ~  
adastra@adAstra-VM:~$ sudo mount file.fs /mnt/tmp  
adastra@adAstra-VM:~$ mount | grep file.fs  
/home/adastra/file.fs on /mnt/tmp type ext4 (rw,relatime)  
adastra@adAstra-VM:~$ |
```

Now, we create some sample text files in our file system and list the stats of a file.

```
adastra@adAstra-VM: /mnt/tmp  
adastra@adAstra-VM:~$ cd /mnt/tmp  
adastra@adAstra-VM:/mnt/tmp$ sudo cat > sample1.txt  
Hello! This is a sample file.  
adastra@adAstra-VM:/mnt/tmp$ sudo touch sample2.txt  
adastra@adAstra-VM:/mnt/tmp$ sudo touch sample3.txt sample4.txt  
adastra@adAstra-VM:/mnt/tmp$ ls -l  
total 20  
drwx----- 2 root    root    16384 Nov 25 17:48 lost+found  
-rw-rw-r-- 1 adastra adastra   30 Nov 25 17:56 sample1.txt  
-rw-r--r-- 1 root    root      0 Nov 25 17:57 sample2.txt  
-rw-r--r-- 1 root    root      0 Nov 25 17:57 sample3.txt  
-rw-r--r-- 1 root    root      0 Nov 25 17:57 sample4.txt  
adastra@adAstra-VM:/mnt/tmp$ stat sample1.txt  
File: sample1.txt  
Size: 30             Blocks: 8           IO Block: 4096   regular file  
Device: 704h/1796d   Inode: 12          Links: 1  
Access: (0664/-rw-rw-r--)  Uid: ( 1000/ adastra)   Gid: ( 1000/ adastra)  
Access: 2021-11-25 17:56:31.000000000 +0530  
Modify: 2021-11-25 17:56:47.000000000 +0530  
Change: 2021-11-25 17:56:47.000000000 +0530  
Birth: -  
adastra@adAstra-VM:/mnt/tmp$ |
```

Finally, we unmount our file system using **umount** command. And if we now try to list the files under our file system, we don't get any output.



```
adastra@adAstra-VM: /  
adastra@adAstra-VM:~$ cd /  
adastra@adAstra-VM:/$ sudo umount /mnt/tmp  
adastra@adAstra-VM:/$ ls /mnt/tmp  
adastra@adAstra-VM:/$ ls -l /mnt/tmp  
total 0  
adastra@adAstra-VM:/$ |
```

The image shows a terminal window with a dark background and a faint, stylized graphic of a creature's face. The terminal displays a sequence of commands and their outputs. The commands are: `cd /`, `sudo umount /mnt/tmp`, `ls /mnt/tmp`, `ls -l /mnt/tmp`, and a prompt with a vertical bar. The outputs are: `total 0` and the prompt `adastra@adAstra-VM:/$ |`. The window title is `adastra@adAstra-VM: /`.