REPORT – ASSIGNMENT 4

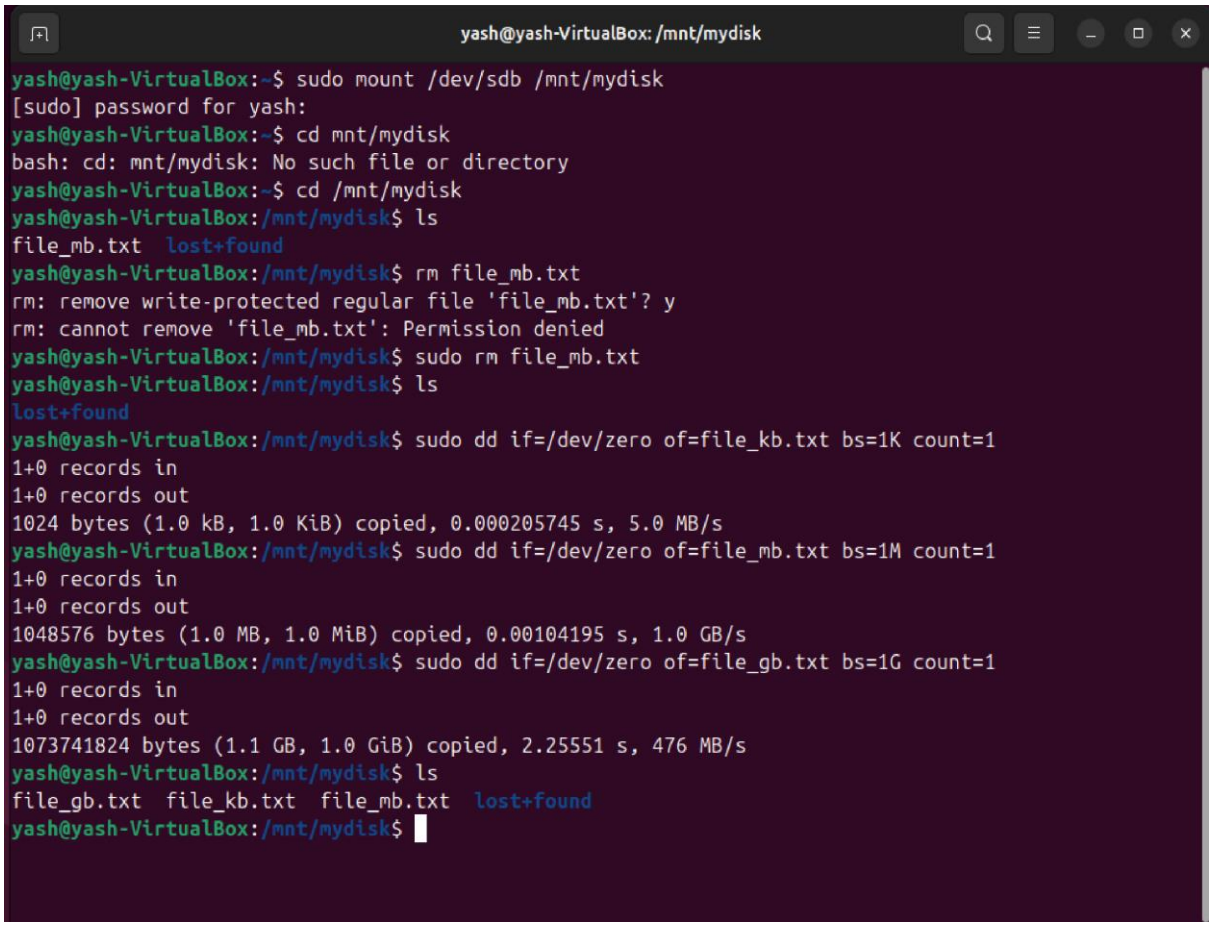
Task-1 – Recovery of files

Setup: I added a virtual hard disk of size 4GB to my Ubuntu VM. Then I used the command: *sudo fdisk -l /dev/sdb* to partition and use the disk for this assignment.

Next, I use the command: *sudo mkfs.ext2 /dev/sdb (my partition name)*, to format the partition with an ext2 file system. Then, I used the command: *sudo mkdir /mnt/mydisk* , to create a mount point.

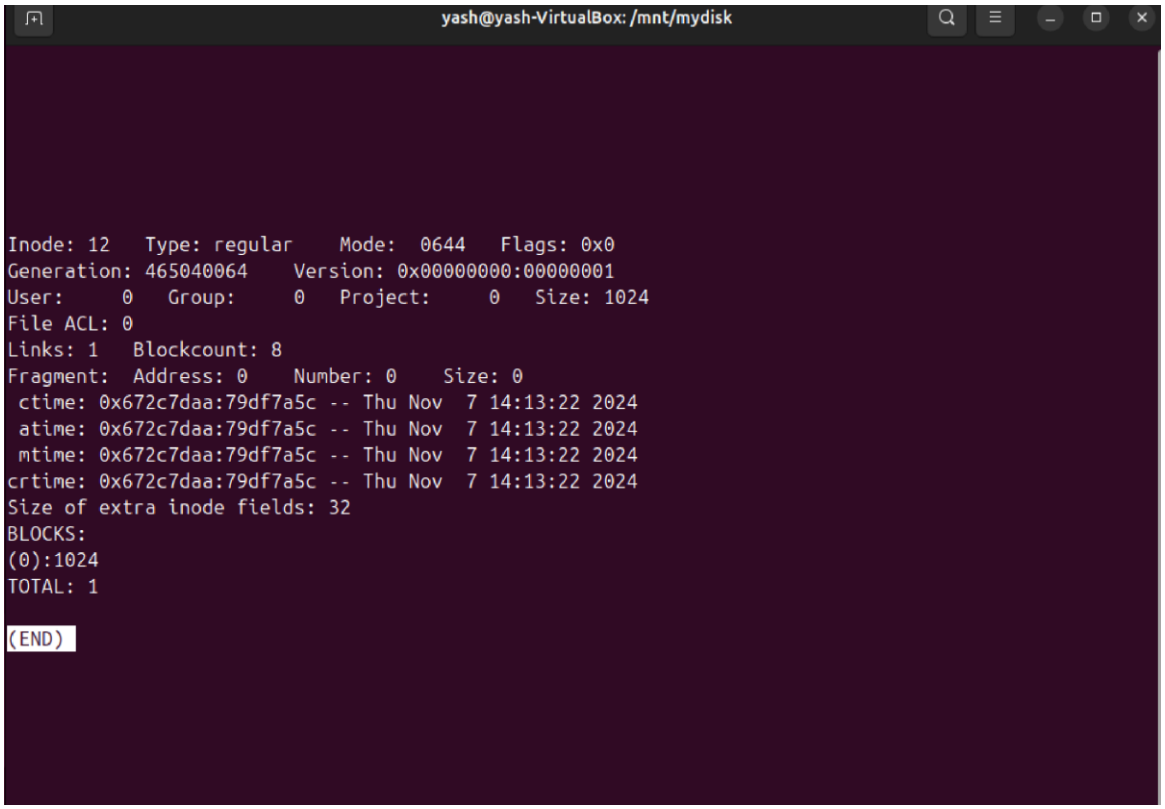
Method I – Using *undel* command

- First, I mount the file system and my change my directory to the mount point. I have created 3 files of size 1kb, 1mb, 1gb respectively using the *dd* command. We can see that the files have been created using the ls command.
  Note: */dev/zero* fills the file data with 0s. We can alternatively use */dev/urandom* for filling the files with random data.

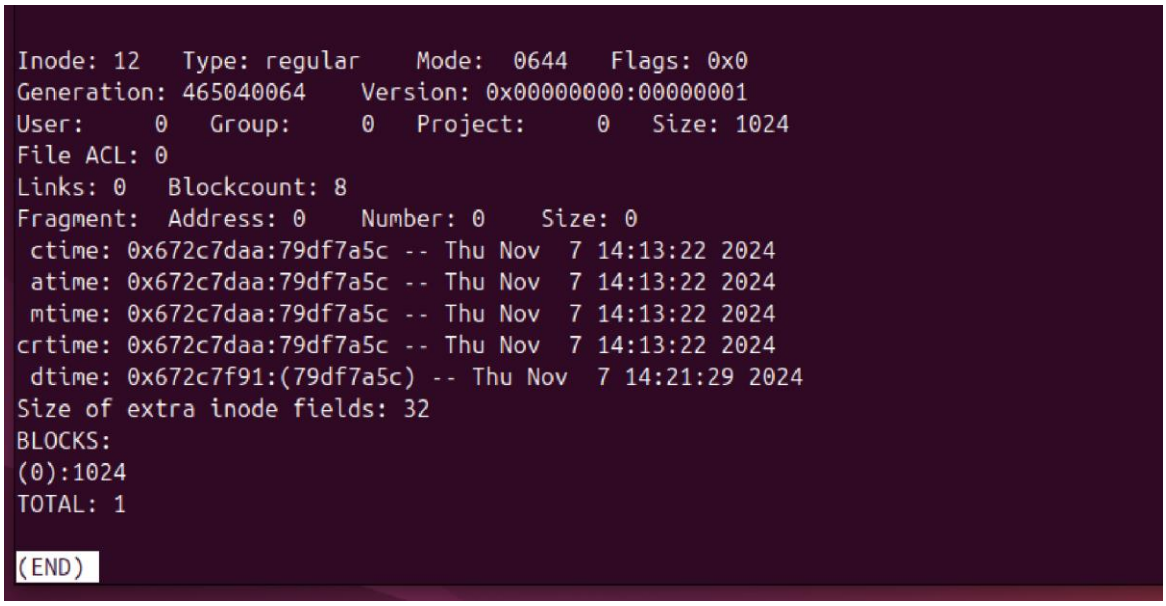


- We will look at the process followed for the 1kb file (the methodology for the other two files is identical).
  1. Using *ls -l* to know the the inode number corresponding to each file created. Inode number for the kb file is seen to be 12.
  2. We then run *debugfs* with write privileges on the filesystem.

3. Then we use *stat <inode number>* to get the information about the inode. The following is the image before deletion of the file. Notice the absence of dtime.



4. Then we remove the file using the *rm* command and check the inode information again. We can observe the deletion time (dtime) at this point.



5. Upon exiting *debugfs,* we can use *ls -i* to observe that the file has indeed been deleted.
6. We run *debugfs* again and we know use the following command: *undel <inode no> filename* to link the inode again to the specified filename. We perform *stat <12>* to check the inode information after recovery. Notice the absence of dtime again.

```
Inode: 12   Type: regular    Mode:  0644   Flags: 0x0
Generation: 465040064    Version: 0x00000000:00000001
User:     0   Group:     0   Project:     0   Size: 1024
File ACL: 0
Links: 1   Blockcount: 8
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x672c7daa:79df7a5c -- Thu Nov  7 14:13:22 2024
 atime: 0x672c7daa:79df7a5c -- Thu Nov  7 14:13:22 2024
 mtime: 0x672c7daa:79df7a5c -- Thu Nov  7 14:13:22 2024
crtime: 0x672c7daa:79df7a5c -- Thu Nov  7 14:13:22 2024
Size of extra inode fields: 32
BLOCKS:
(0):1024
TOTAL: 1

(END)
```

7. We then exit *debugfs* and we see that the file exists with the same size using *ls -i* and *ls -l.* The following snapshot captures the whole process for the 1 kb file.

```
yash@yash-VirtualBox:/mnt/mydisk$ sudo debugfs /dev/sdb
debugfs 1.47.0 (5-Feb-2023)
debugfs:  stat<12>
debugfs: Unknown request "stat<12>".  Type "?" for a request list.
debugfs:  stat <12>
debugfs:  rm file_kb.txt
rm: Filesystem opened read/only
debugfs:  stat <12>
debugfs:  quit
yash@yash-VirtualBox:/mnt/mydisk$ sudo debugfs -w /dev/sdb
debugfs 1.47.0 (5-Feb-2023)
debugfs:  stat <12>
debugfs:  rm file_kb.txt

debugfs:  stat <12>
debugfs:  quit
yash@yash-VirtualBox:/mnt/mydisk$ ls -i
14 file_gb.txt  13 file_mb.txt  11 lost+found
yash@yash-VirtualBox:/mnt/mydisk$ sudo debugfs -w /dev/sdb
debugfs 1.47.0 (5-Feb-2023)
debugfs:  undel <12> file_kb.txt
debugfs:  stat <12>
debugfs:  quit
yash@yash-VirtualBox:/mnt/mydisk$ ls -i
14 file_gb.txt  12 file_kb.txt  13 file_mb.txt  11 lost+found
yash@yash-VirtualBox:/mnt/mydisk$ ls -l
total 1050652
-rw-r--r-- 1 root root 1073741824 Nov  7 14:13 file_gb.txt
-rw-r--r-- 1 root root       1024 Nov  7 14:13 file_kb.txt
-rw-r--r-- 1 root root    1048576 Nov  7 14:13 file_mb.txt
drwx------ 2 root root      16384 Nov  7 12:29 lost+found
yash@yash-VirtualBox:/mnt/mydisk$
```

- We follow the same process for the 1mb and 1kb files.
- Attached screenshots show the process followed for the 1 MB file, and then the 1GB file.

```
yash@yash-VirtualBox:/mnt/mydisk$ sudo debugfs -w /dev/sdb
debugfs 1.47.0 (5-Feb-2023)
debugfs:  stat <13>
debugfs:  rm file_mb.txt

debugfs:  stat <13>
debugfs:  quit
yash@yash-VirtualBox:/mnt/mydisk$ ls -i
14 file_gb.txt  12 file_kb.txt  11 lost+found
yash@yash-VirtualBox:/mnt/mydisk$ sudo debugfs -w /dev/sdb
debugfs 1.47.0 (5-Feb-2023)
debugfs:  undel <13> file_mb.txt
debugfs:  stat <13>
debugfs:  quit
yash@yash-VirtualBox:/mnt/mydisk$ ls -i
14 file_gb.txt  12 file_kb.txt  13 file_mb.txt  11 lost+found
yash@yash-VirtualBox:/mnt/mydisk$ ls -l
total 1050652
-rw-r--r-- 1 root root 1073741824 Nov  7 14:13 file_gb.txt
-rw-r--r-- 1 root root       1024 Nov  7 14:13 file_kb.txt
-rw-r--r-- 1 root root    1048576 Nov  7 14:13 file_mb.txt
drwx------ 2 root root      16384 Nov  7 12:29 lost+found
yash@yash-VirtualBox:/mnt/mydisk$
```

Figure 2.1: Process for mb file

```
Inode: 13   Type: regular    Mode:  0644   Flags: 0x0
Generation: 1168179146    Version: 0x00000000:00000001
User:     0   Group:     0   Project:     0   Size: 1048576
File ACL: 0
Links: 1   Blockcount: 2056
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
 atime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
 mtime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
crtime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
Size of extra inode fields: 32
BLOCKS:
(0-11):1536-1547, (IND):777, (12-15):1548-1551, (16-63):816-863, (64-127):896-959, (128-255):2048-2175
TOTAL: 257

(END)
```

Figure 2.2: Stat before deletion.

```
Inode: 13   Type: regular    Mode:  0644   Flags: 0x0
Generation: 1168179146    Version: 0x00000000:00000001
User:     0   Group:     0   Project:     0   Size: 1048576
File ACL: 0
Links: 0   Blockcount: 2056
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
 atime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
 mtime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
crtime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
 dtime: 0x672c81b2:(61198c98) -- Thu Nov  7 14:30:34 2024
Size of extra inode fields: 32
BLOCKS:
(0-11):1536-1547, (IND):777, (12-15):1548-1551, (16-63):816-863, (64-127):896-959, (128-255):2048-2175
TOTAL: 257

(END)
```

Figure 2.3: Stat after deletion

```
Inode: 13   Type: regular    Mode:  0644   Flags: 0x0
Generation: 1168179146    Version: 0x00000000:00000001
User:     0   Group:     0   Project:     0   Size: 1048576
File ACL: 0
Links: 1   Blockcount: 2056
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
 atime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
 mtime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
crtime: 0x672c7dc3:61198c98 -- Thu Nov  7 14:13:47 2024
Size of extra inode fields: 32
BLOCKS:
(0-11):1536-1547, (IND):777, (12-15):1548-1551, (16-63):816-863, (64-127):896-959, (128-255):2048-2175
TOTAL: 257

(END)
```

Figure 2.4: Stat after recovery

```
yash@yash-VirtualBox:/mnt/mydisk$ ls -i
14 file_gb.txt  12 file_kb.txt  13 file_mb.txt  11 lost+found
yash@yash-VirtualBox:/mnt/mydisk$ sudo debugfs -w /dev/sdb
debugfs 1.47.0 (5-Feb-2023)
debugfs:  stat <14>
debugfs:  rm file_gb.txt

debugfs:  stat <14>
debugfs:  quit
yash@yash-VirtualBox:/mnt/mydisk$ ls -i
12 file_kb.txt  13 file_mb.txt  11 lost+found
yash@yash-VirtualBox:/mnt/mydisk$ sudo debugfs -w /dev/sdb
debugfs 1.47.0 (5-Feb-2023)
debugfs:  undel <14> file_gb.txt
debugfs:  stat <14>
debugfs:  quit
yash@yash-VirtualBox:/mnt/mydisk$ ls -i
14 file_gb.txt  12 file_kb.txt  13 file_mb.txt  11 lost+found
yash@yash-VirtualBox:/mnt/mydisk$ ls -l
total 1050652
-rw-r--r-- 1 root root 1073741824 Nov  7 14:13 file_gb.txt
-rw-r--r-- 1 root root       1024 Nov  7 14:13 file_kb.txt
-rw-r--r-- 1 root root    1048576 Nov  7 14:13 file_mb.txt
drwx------ 2 root root      16384 Nov  7 12:29 lost+found
yash@yash-VirtualBox:/mnt/mydisk$
```

Figure 3.1: Process for gb file

```
Inode: 14   Type: regular    Mode:  0644   Flags: 0x0
Generation: 2173767800    Version: 0x00000000:00000002
User:     0   Group:     0   Project:     0   Size: 1073741824
File ACL: 0
Links: 1   Blockcount: 2099208
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x672c7dcf:0fd115d0 -- Thu Nov  7 14:13:59 2024
 atime: 0x672c7dce:733c7f28 -- Thu Nov  7 14:13:58 2024
 mtime: 0x672c7dcf:0fd115d0 -- Thu Nov  7 14:13:59 2024
crtime: 0x672c7dce:733c7f28 -- Thu Nov  7 14:13:58 2024
Size of extra inode fields: 32
BLOCKS:
(0-11):1025-1036, (IND):778, (12-15):1037-1040, (16-31):880-895, (32-63):960-991, (64-127):2208-2271, (128-255):2304-243
1, (256-511):2560-2815, (512-1023):3072-3583, (1024-1035):4096-4107, (DIND):779, (IND):780, (1036-2047):4108-5119, (2048
-2059):6144-6155, (IND):781, (2060-3083):6156-7179, (IND):782, (3084-4107):7180-8203, (IND):783, (4108-5131):8204-9227,
(IND):784, (5132-6155):9228-10251, (IND):785, (6156-7179):10252-11275, (IND):786, (7180-8203):11276-12299, (IND):787, (8
204-9227):12300-13323, (IND):788, (9228-10251):13324-14347, (IND):789, (10252-11275):14348-15371, (IND):790, (11276-1229
9):15372-16395, (IND):791, (12300-13323):16396-17419, (IND):792, (13324-14347):17420-18443, (IND):793, (14348-15371):184
44-19467, (IND):794, (15372-16395):19468-20491, (IND):795, (16396-17419):20492-21515, (IND):796, (17420-18443):21516-225
39, (IND):797, (18444-19467):22540-23563, (IND):798, (19468-20491):23564-24587, (IND):799, (20492-21515):24588-25611, (I
ND):800, (21516-22539):25612-26635, (IND):801, (22540-23563):26636-27659, (IND):802, (23564-24587):27660-28683, (IND):80
3, (24588-25611):28684-29707, (IND):804, (25612-26635):29708-30731, (IND):805, (26636-27659):30732-31755, (IND):806, (27
660-28671):31756-32767, (28672-28683):34816-34827, (IND):807, (28684-29707):34828-35851, (IND):808, (29708-30731):35852-
36875, (IND):809, (30732-31755):36876-37899, (IND):810, (31756-32779):37900-38923, (IND):811, (32780-33803):38924-39947,
 (IND):812, (33804-34827):39948-40971, (IND):813, (34828-35851):40972-41995, (IND):814, (35852-36875):41996-43019, (IND)
:815, (36876-37899):43020-44043, (IND):992, (37900-38923):44044-45067, (IND):993, (38924-39947):45068-46091, (IND):994,
(39948-40971):46092-47115, (IND):995, (40972-41995):47116-48139, (IND):996, (41996-43019):48140-49163, (IND):997, (43020
-44043):49164-50187, (IND):998, (44044-45067):50188-51211, (IND):999, (45068-46091):51212-52235, (IND):1000, (46092-4711
5):52236-53259, (IND):1001, (47116-48139):53260-54283, (IND):1002, (48140-49163):54284-55307, (IND):1003, (49164-50187):
55308-56331, (IND):1004, (50188-51211):56332-57355, (IND):1005, (51212-52235):57356-58379, (IND):1006, (52236-53259):583
80-59403, (IND):1007, (53260-54283):59404-60427, (IND):1008, (54284-55307):60428-61451, (IND):1009, (55308-56331):61452-
:
```

Figure 3.2: Stat before deletion

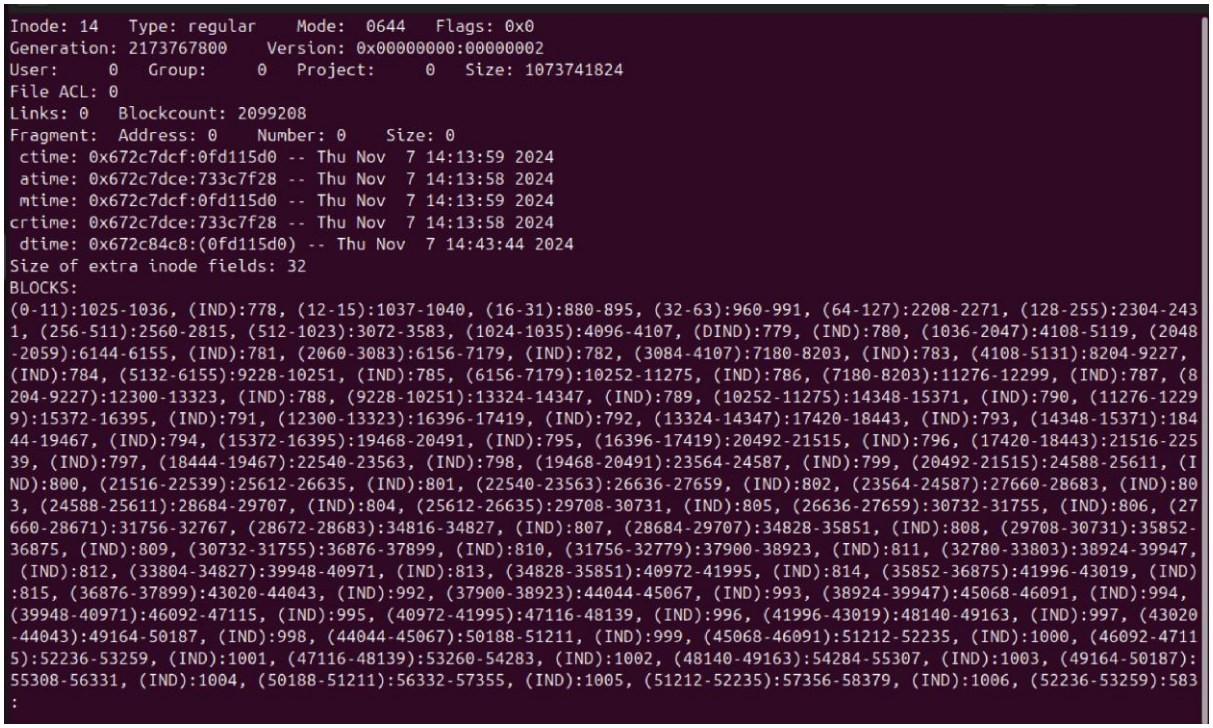


Figure 3.3: Stat after deletion

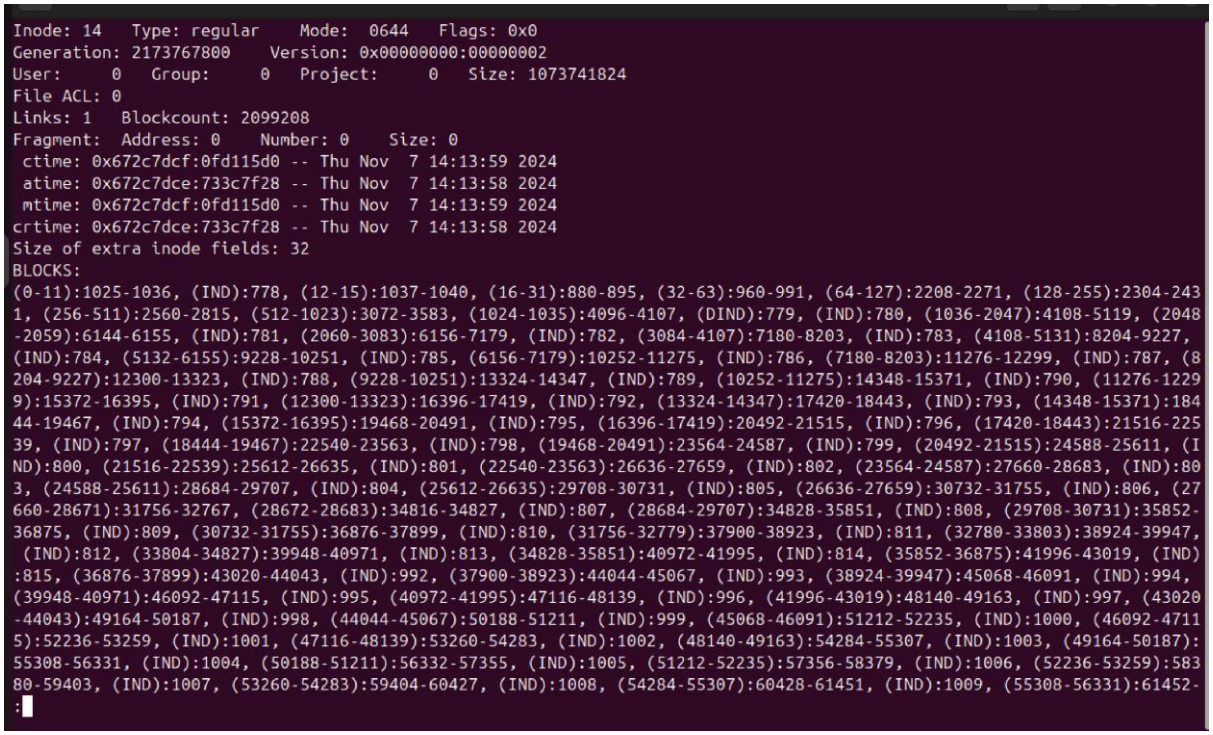


Figure 3.4: Stat after recovery

- After recovering the files, We now do a consistency check on the file system using the commands: *fsck and e2fsck*. It is necessary to unmount the file system before performing the mentioned consistency checks.
  Note: -y is used to automatically respond yes to all the prompts.

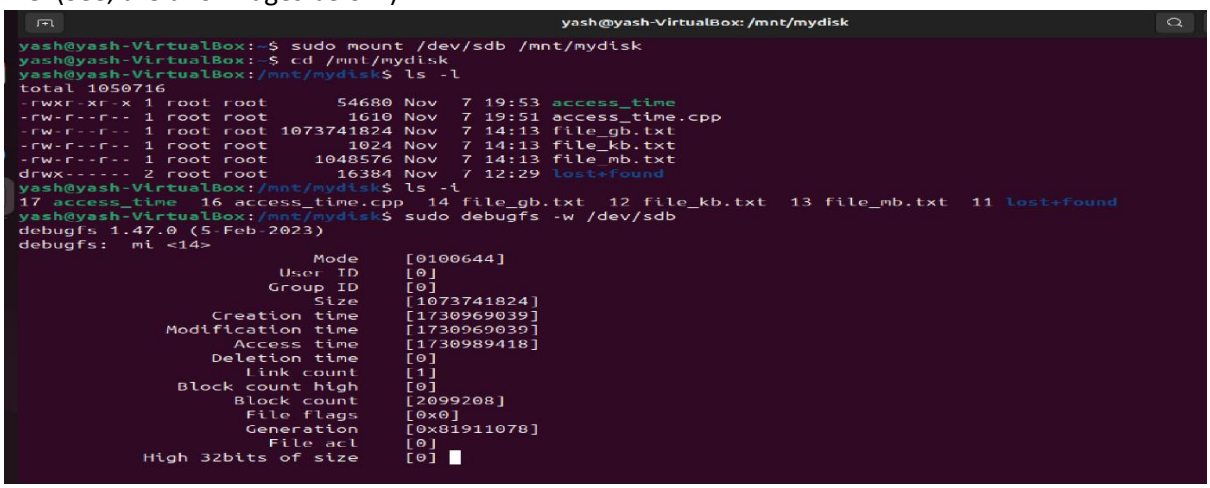
```
yash@yash-VirtualBox:/mnt/mydisk$ sudo umount /dev/sdb
umount: /mnt/mydisk: target is busy.
yash@yash-VirtualBox:/mnt/mydisk$ mount | grep /dev/sdb
/dev/sdb on /mnt/mydisk type ext2 (rw,relatime)
yash@yash-VirtualBox:/mnt/mydisk$ cd
yash@yash-VirtualBox:~$ sudo umount /dev/sdb
yash@yash-VirtualBox:~$ mount | grep /dev/sdb
yash@yash-VirtualBox:~$ sudo fsck /dev/sdb
fsck from util-linux 2.39.3
e2fsck 1.47.0 (5-Feb-2023)
/dev/sdb: clean, 14/262144 files, 281169/1048576 blocks
yash@yash-VirtualBox:~$ sudo e2fsck -y /dev/sdb
e2fsck 1.47.0 (5-Feb-2023)
/dev/sdb: clean, 14/262144 files, 281169/1048576 blocks
yash@yash-VirtualBox:~$
```

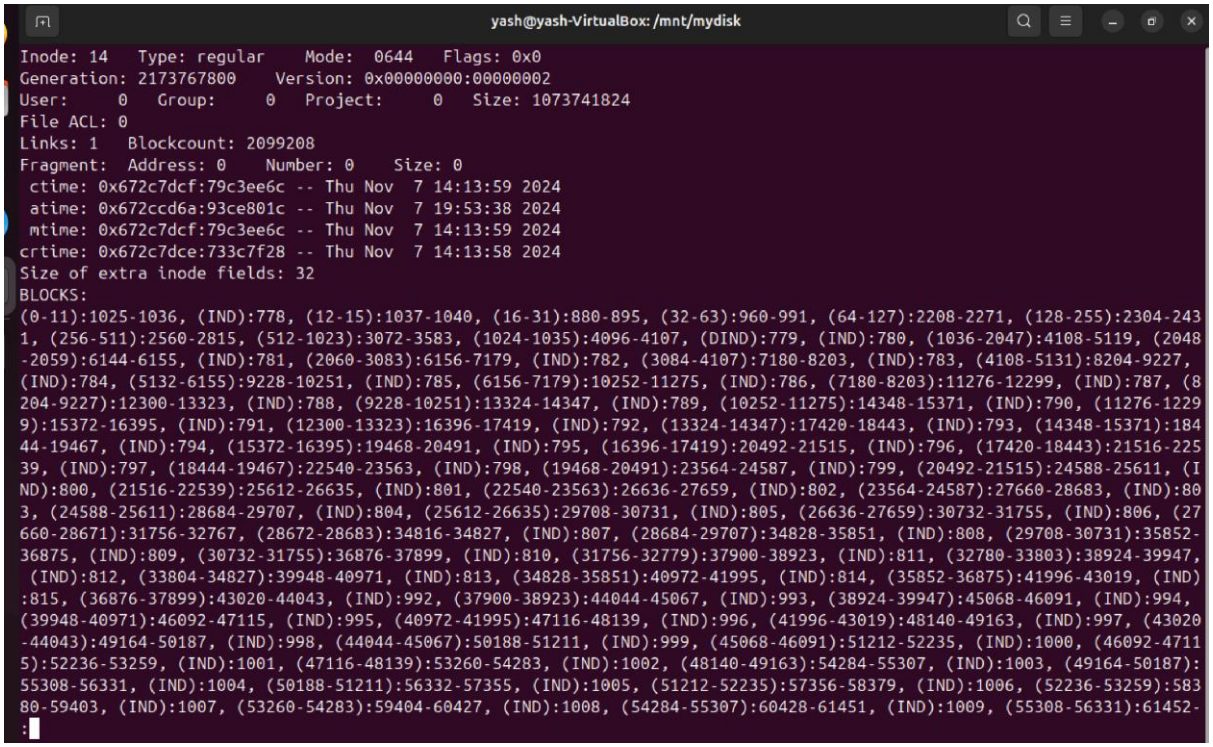
- It is recommended to reboot the system after these checks before mounting the file system for use again.

Method II – Using commands *mi and link*

Note: This method is only demonstrated with the 1GB file. The procedure for the files of the other sizes is the same.

- With the same setup and the file system mounted, we check the inode numbers of the current files. Observe that the inode number of the 1GB file is 14. Then, we run *debugfs command* in write mode. we first use the following commands:
  mi <inode number> - We use this to get all the initial information about the inode. No modifications are made yet.
  stat <inode number> - Just to check the how the information is displayed before deletion of the file. (See, the two images below.)

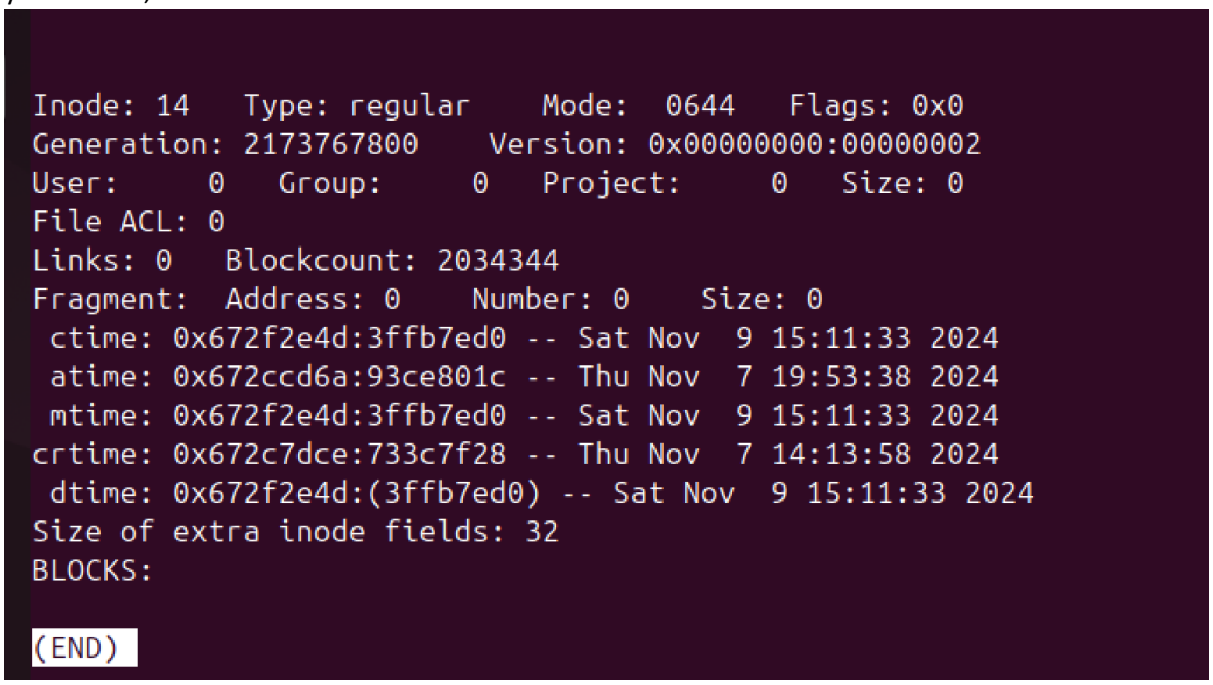




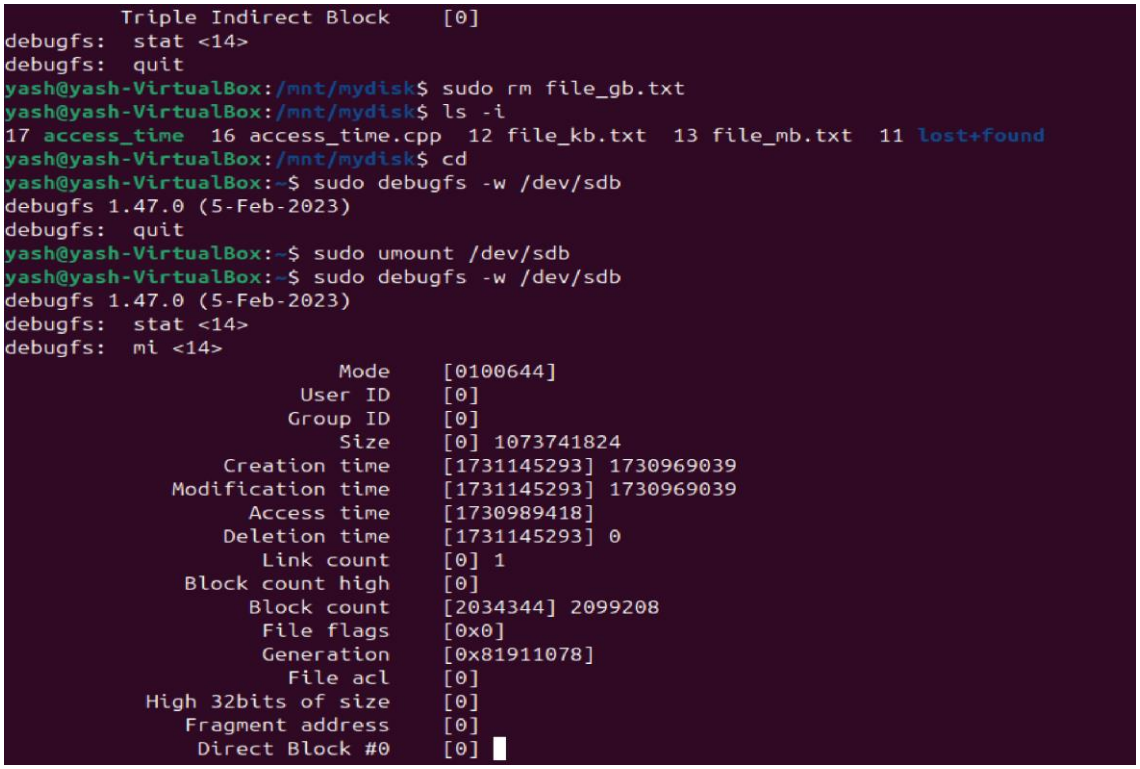
- Then, we remove the file. We use *ls -i* to check whether the file has been deleted. (See below)

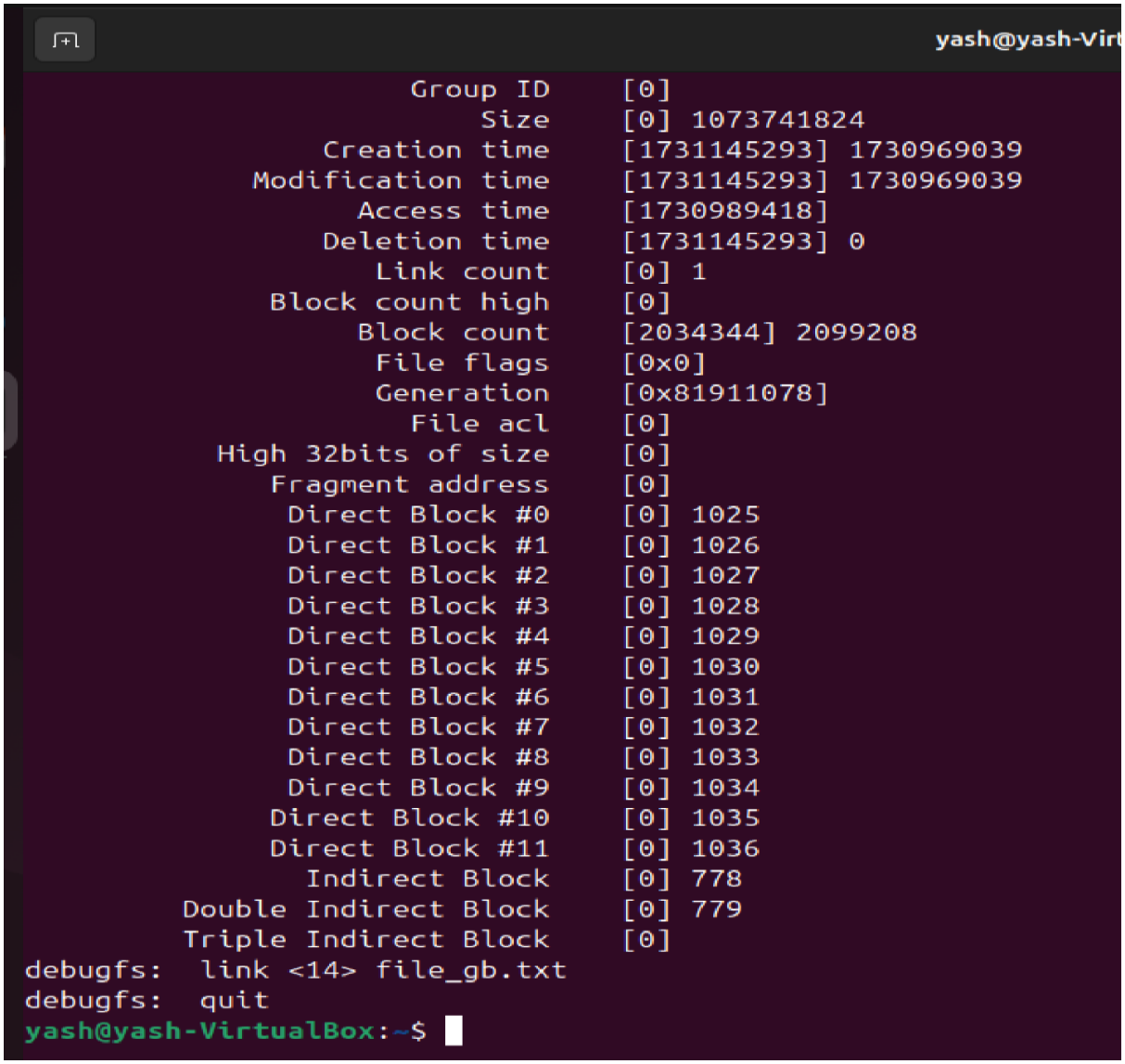
```
                    File acl    [0]
          High 32bits of size   [0]
             Fragment address   [0]
              Direct Block #0   [1025]
              Direct Block #1   [1026]
              Direct Block #2   [1027]
              Direct Block #3   [1028]
              Direct Block #4   [1029]
              Direct Block #5   [1030]
              Direct Block #6   [1031]
              Direct Block #7   [1032]
              Direct Block #8   [1033]
              Direct Block #9   [1034]
             Direct Block #10   [1035]
             Direct Block #11   [1036]
               Indirect Block   [778]
         Double Indirect Block  [779]
         Triple Indirect Block  [0]
debugfs:  stat <14>
debugfs:  quit
yash@yash-VirtualBox:/mnt/mydisk$ sudo rm file_gb.txt
yash@yash-VirtualBox:/mnt/mydisk$ ls -i
17 access_time  16 access_time.cpp  12 file_kb.txt  13 file_mb.txt  11 lost+found
yash@yash-VirtualBox:/mnt/mydisk$ cd
yash@yash-VirtualBox:~$ sudo debugfs -w /dev/sdb
debugfs 1.47.0 (5-Feb-2023)
debugfs:  quit
yash@yash-VirtualBox:~$ sudo umount /dev/sdb
yash@yash-VirtualBox:~$ sudo debugfs -w /dev/sdb
debugfs 1.47.0 (5-Feb-2023)
debugfs:  stat <14>
debugfs:
```

- We again use *stat <inode number> in debugfs* to check the inode information after deletion. As you can see, it shows no blocks attached.

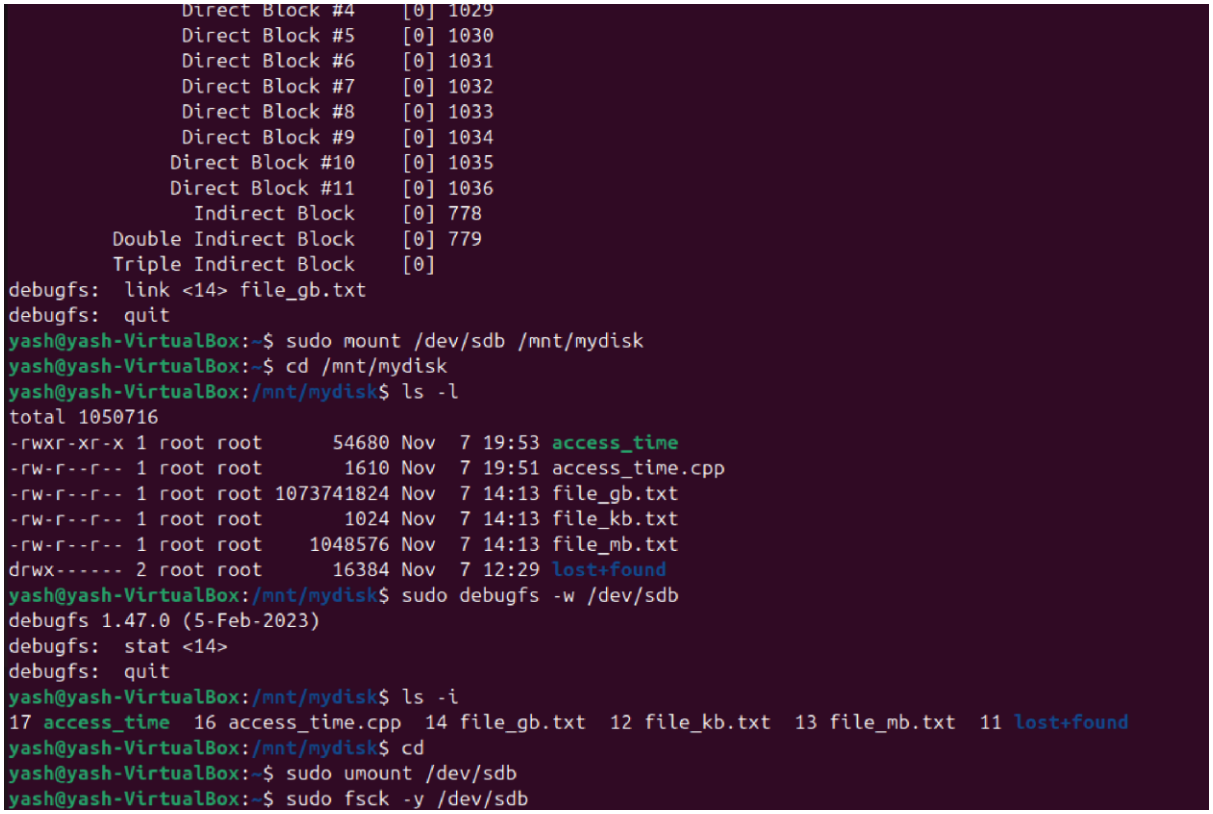


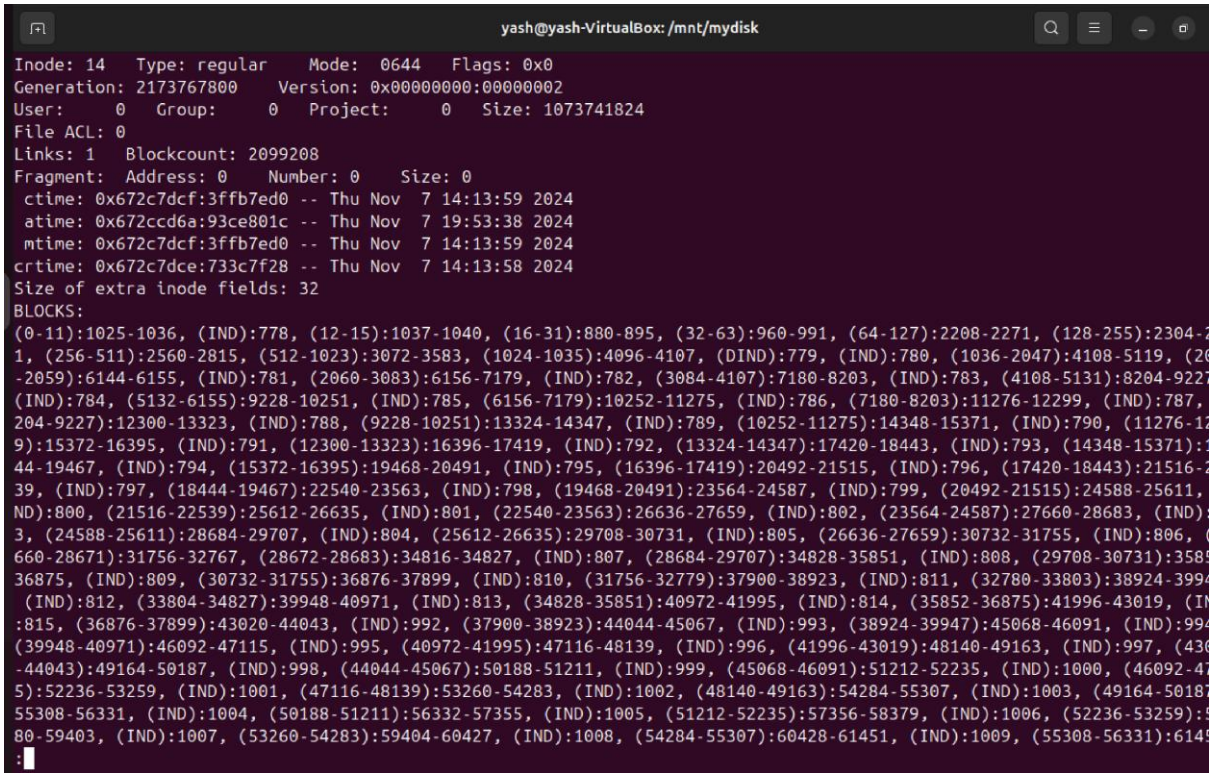
- Now, we unmount the file system and run the *debugfs command* on the filesystem. We use *mi <inode number>* to modify the inode information according what we recorded initially. We then link the inode to a filename using the command: *link <inode number> filename.*




yash@yash-Virt
Group ID [0]
Size [0] 1073741824
Creation time [1731145293] 1730969039
Modification time [1731145293] 1730969039
Access time [1730989418]
Deletion time [1731145293] 0
Link count [0] 1
Block count high [0]
Block count [2034344] 2099208
File flags [0x0]
Generation [0x81911078]
File acl [0]
High 32bits of size [0]
Fragment address [0]
Direct Block #0 [0] 1025
Direct Block #1 [0] 1026
Direct Block #2 [0] 1027
Direct Block #3 [0] 1028
Direct Block #4 [0] 1029
Direct Block #5 [0] 1030
Direct Block #6 [0] 1031
Direct Block #7 [0] 1032
Direct Block #8 [0] 1033
Direct Block #9 [0] 1034
Direct Block #10 [0] 1035
Direct Block #11 [0] 1036
Indirect Block [0] 778
Double Indirect Block [0] 779
Triple Indirect Block [0]
debugfs: link <14> file_gb.txt
debugfs: quit
yash@yash-VirtualBox:~$

- Finally, we mount the filesystem again to check whether the file has been recovered (We also use *stat command* again to see whether all the information has been restored correctly). We also run a consistency check using the *fsck* command.





Task-2- Analysis of File Access Times Across Different File Sizes

Overview of the Process

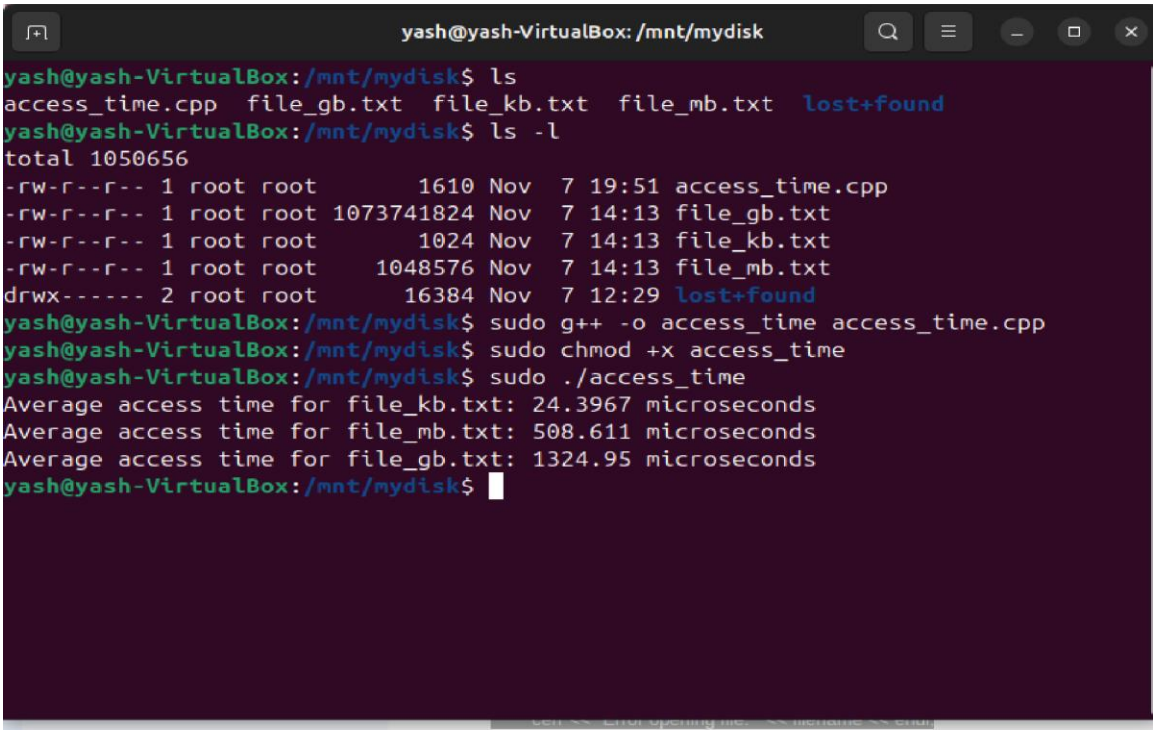
- **Purpose of the Code**: The included C++ program *access_time.cpp* measures the average access time required to read a single byte from random positions within the files of different sizes that we created. This task is designed to understand how file size impacts the efficiency of data retrieval operations.
- **How the code works**: The program uses the measure_access_time function to read a single byte from 100 random positions in specified files and measure the time taken for each read using high-resolution timers. It accomplishes this by opening files in binary mode, generating random positions with a uniformly distributed random number generator, and then seeking to these positions to read a byte. The average of these 100 measurements gives the access time for each file, highlighting the effect of file size on data retrieval efficiency.

Observations from the Terminal Output

- **File Creation**: Files named file_kb.txt, file_mb.txt, and file_gb.txt were successfully created with sizes of 1KB, 1MB, and 1GB respectively, as confirmed by the ls -l command output in the terminal.
- **Compilation and Execution**: The program access_time.cpp was compiled without errors, and the executable was run to measure the access times.
- **Measured Times**:
    - **1KB file**: The access time was approximately 24.3967 microseconds.
    - **1MB file**: The access time was significantly higher at about 508.611 microseconds.
    - **1GB file**: The largest file had the highest access time, averaging 1324.95 microseconds.

Conclusions Drawn

- **Impact of File Size on Access Time**: The results clearly indicate an increase in access time as the file size increases. This can be attributed to the larger file sizes potentially exceeding the system's cache or available memory, thereby requiring actual disk reads which are slower.
- **System Performance**: The system's ability to handle file operations efficiently is evident, although performance degrades with larger files. This degradation likely results from increased seek times and the mechanical limitations of disk-based storage (if not using SSD).



Sources and References

- Commands for creating the files using dd, and commands for debugfs have been referred from the man pages.
- For task-2, The lines for generating random numbers have been referred from the net. They have been used to get the best results possible.