CS3300 – Operating Systems

Assignment 4

REPORT

Name: Vedang Bahuguna

Roll Number: CS22B091

Task 1: Collect Inode and Block Numbers for the Information Stored

- To collect inodes, we can use the **ls -ail** command. The -i flag ensures that the inodes (first column) for all the concerned files are displayed along with the file names.

```
vedang@vedang-VirtualBox:/mnt/ext2dir$ ls -ail
total 51304
     2 drwxr-xr-x 3 root root     4096 Nov  6 22:31 .
393219 drwxr-xr-x 3 root root     4096 Nov  6 22:25 ..
    14 -rwxr-xr-x 1 root root    17184 Nov  6 22:31 a.out
    13 -rw-r--r-- 1 root root      197 Nov  6 22:31 gen.cpp
    11 drwx------ 2 root root    16384 Nov  6 22:25 lost+found
    15 -rw-r--r-- 1 root root 52428800 Nov  6 22:32 test
vedang@vedang-VirtualBox:/mnt/ext2dir$
```

Here, 15 is the inode of the test file.

- To collect the block numbers, we enter into the debugfs mode through the following command: **sudo debugfs -w ext2_image.img**. Once in the debugfs mode, we can find out the block numbers linked to the inode (the file) through: **blocks <inode>**. These are all the 102512 blocks of size 4096 bytes used by the file.

```
vedang@vedang-VirtualBox:/mnt/ext2dir$ sudo debugfs -w /ext2_image.img
debugfs 1.47.0 (5-Feb-2023)
debugfs:  blocks <15>
2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2048 2061 2062 2063 2064 1648 1649 1650
1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670
1671 1672 1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690
1691 1692 1693 1694 1695 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742
1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762
1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778 1779 1780 1781 1782
1783 1784 1785 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802
1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 1821 1822
1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842
1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862
1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882
1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902
1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 3712 3713 3714
3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730 3731 3732 3733 3734
3735 3736 3737 3738 3739 3740 3741 3742 3743 3744 3745 3746 3747 3748 3749 3750 3751 3752 3753 3754
3755 3756 3757 3758 3759 3760 3761 3762 3763 3764 3765 3766 3767 3768 3769 3770 3771 3772 3773 3774
3775 3776 3777 3778 3779 3780 3781 3782 3783 3784 3785 3786 3787 3788 3789 3790 3791 3792 3793 3794
3795 3796 3797 3798 3799 3800 3801 3802 3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813 3814
3815 3816 3817 3818 3819 3820 3821 3822 3823 3824 3825 3826 3827 3828 3829 3830 3831 3832 3833 3834
3835 3836 3837 3838 3839 3840 3841 3842 3843 3844 3845 3846 3847 3848 3849 3850 3851 3852 3853 3854
3855 3856 3857 3858 3859 3860 3861 3862 3863 3864 3865 3866 3867 3868 3869 3870 3871 3872 3873 3874
3875 3876 3877 3878 3879 3880 3881 3882 3883 3884 3885 3886 3887 3888 3889 3890 3891 3892 3893 3894
3895 3896 3897 3898 3899 3900 3901 3902 3903 3904 3905 3906 3907 3908 3909 3910 3911 3912 3913 3914
3915 3916 3917 3918 3919 3920 3921 3922 3923 3924 3925 3926 3927 3928 3929 3930 3931 3932 3933 3934
```

- Another way to get the Direct and Indirect blocks information is to use the **mi <inode>** query inside debugfs as follows:

```
vedang@vedang-VirtualBox:/mnt/ext2dir$ sudo debugfs -w /ext2_image.img
debugfs 1.47.0 (5-Feb-2023)
debugfs:  mi <15>
                          Mode    [0100644]
                       User ID    [0]
                      Group ID    [0]
                          Size    [52428800]
                 Creation time    [1730912556]
             Modification time    [1730912556]
                   Access time    [1730912512]
                 Deletion time    [0]
                    Link count    [1]
              Block count high    [0]
                   Block count    [102512]
                    File flags    [0x0]
                    Generation    [0xfd81cf83]
                      File acl    [0]
           High 32bits of size    [0]
              Fragment address    [0]
               Direct Block #0    [2049]
               Direct Block #1    [2050]
               Direct Block #2    [2051]
               Direct Block #3    [2052]
               Direct Block #4    [2053]
               Direct Block #5    [2054]
               Direct Block #6    [2055]
               Direct Block #7    [2056]
               Direct Block #8    [2057]
               Direct Block #9    [2058]
              Direct Block #10    [2059]
              Direct Block #11    [2060]
                Indirect Block    [2048]
         Double Indirect Block    [2560]
         Triple Indirect Block    [0]
debugfs:
```

Task 1: Deleting the file and reconstructing it such that it has the same inode number.

- Note the following steps are carried out in the **ext2 filesystem.** This was done so, as there are high chances of the copied file being corrupted, which in turn corrupts the entire OS, when the direct/indirect blocks were not restored properly. For that, a new disk image of size 200MB was created using the following steps:

1. Create a disk image in the root directory, which is going to emulate a physical device (such as a hard disk), with the following command:

   sudo dd if=/dev/zero of=ext2_image.img bs=1M count=200

```
vedang@vedang-VirtualBox:/$ sudo dd if=/dev/zero of=ext2_image.img bs=1M count=200
[sudo] password for vedang:
200+0 records in
200+0 records out
209715200 bytes (210 MB, 200 MiB) copied, 0.552932 s, 379 MB/s
vedang@vedang-VirtualBox:/$ ls
bin                etc             lib.usr-is-merged  proc            sbin.usr-is-merged  usr
bin.usr-is-merged  ext2_image.img  lost+found         retrieved_file  snap                var
boot               home            media              root            srv
cdrom              lib             mnt                run             sys
dev                lib64           opt                sbin            tmp
vedang@vedang-VirtualBox:/$
```

2. Create a new directory inside the empty mnt directory inside root: **sudo mkdir mnt/ext2dir**

3. Change the filesystem of the created image:

   sudo mkfs.ext2 ext2_image.img

4. Mount the image to the created directory. Thus, the newly created directory becomes the mount point for the disk image.

   sudo mount -o loop ext2_image.img mnt/ext2dir

```
vedang@vedang-VirtualBox:/$ sudo mkdir mnt/ext2dir
vedang@vedang-VirtualBox:/$ sudo mkfs.ext2 ext2_image.img
mke2fs 1.47.0 (5-Feb-2023)
Discarding device blocks: done
Creating filesystem with 51200 4k blocks and 51200 inodes
Filesystem UUID: c0638101-2cb4-4289-8baa-f937c9c605ba
Superblock backups stored on blocks:
        32768

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

vedang@vedang-VirtualBox:/$ sudo mount -o loop ext2_image.img mnt/ext2dir/
vedang@vedang-VirtualBox:/$
```

5. Create the required file of ranges KB, MB and GB. Here, I have demonstrated a 50MB file. The steps remain the same for any file size. I am also using a simple gen.cpp file that generates random characters and outputs it to a test file. The file can then be truncated accordingly to get 50MB.

6. Enter the debugfs mode as follows: **sudo debugfs -w /ext2_image.img**

7. In the debugfs mode, query:
mi <inode number of the file to be deleted> and store the metadata of the test file for future use. Mi stands for modify_inode and is used to modify the inode struct entries in an interactive mode.

```
vedang@vedang-VirtualBox:/$ ls
bin                 etc             lib.usr-is-merged  proc            sbin.usr-is-merged  usr
bin.usr-is-merged   ext2_image.img  lost+found         retrieved_file  snap                var
boot                home            media              root            srv
cdrom               lib             mnt                run             sys
dev                 lib64           opt                sbin            tmp
vedang@vedang-VirtualBox:/$ sudo debugfs -w /ext2_image.img
debugfs 1.47.0 (5-Feb-2023)
debugfs:  mi <15>
```

8. Delete the file: **rm test**. Now after deleting the file and before modifying the deleted inode, we have to UNMOUNT the disk image. This is done, so that the inode changes that are made are reflected as well (unmounting and working on essential sensitive data), and the OS does not interfere with inode that was recently freed in the image. To unmount the image: **sudo umount /mnt/ext2dir.**

```
vedang@vedang-VirtualBox:/$ sudo rm /mnt/ext2dir/test
vedang@vedang-VirtualBox:/$ sudo umount /mnt/ext2dir
```

9. Now, with the disk image unmounted, enter the debugfs mode again and access the metadata of the deleted inode using **mi <inode>**. Restore the essential pointers such as Size, Block Count, Deletion Time, Link Count, Direct Blocks and Indirect Blocks etc. using the data we earlier stored step 6. In this step, we are basically modifying the inode structure so that it again points to the same blocks, and is completely restored to how it was before the test file was removed. Moreover, this is possible as the blocks are NOT immediately overwritten in the hard disk. So, they retain the previous data until being linked to another inode where they might be overwritten.

10. Link the file: **link <inode> restored_file_50MB.** This query creates a new file “restored_file_50MB” with the inode number specified, and ensures that while doing so, the number of hard

links does not increase i.e. remains as 1 (as it should be). This is the recovered file that has the same inode as the deleted file.

```
vedang@vedang-VirtualBox:/$ sudo debugfs -w ext2_image.img
debugfs 1.47.0 (5-Feb-2023)
debugfs:  mi <15>
                          Mode    [0100644]
                       User ID    [0]
                      Group ID    [0]
                          Size    [0] 52428800
                 Creation time    [1730912918]
             Modification time    [1730912918]
                   Access time    [1730912904]
                 Deletion time    [1730912918] 0
                    Link count    [0] 1
              Block count high    [0]
                   Block count    [0] 102512
                    File flags    [0x0]
                    Generation    [0xfd81cf83]
                      File acl    [0]
           High 32bits of size    [0]
              Fragment address    [0]
               Direct Block #0    [0] 2049
               Direct Block #1    [0] 2050
               Direct Block #2    [0] 2051
               Direct Block #3    [0] 2052
               Direct Block #4    [0] 2053
               Direct Block #5    [0] 2054
               Direct Block #6    [0] 2055
               Direct Block #7    [0] 2056
               Direct Block #8    [0] 2057
               Direct Block #9    [0] 2058
              Direct Block #10    [0] 2059
              Direct Block #11    [0] 2060
                Indirect Block    [0] 2048
          Double Indirect Block    [0] 2560
          Triple Indirect Block    [0]
debugfs:  link <15> restored_file_50MB
debugfs:
```

Here, the values in [ ] corresponds to the deleted inode, and the ones written next to it are entered manually using the data stored in step 6.

11. Now, a very important step: Mounting the image back again, so that the changes made are visible. We earlier unmounted the disk image so as to avoid synchronization issues such as operating system toying with the recently freed inode.
sudo mount -o loop ext2_image.img /mnt/ext2dir

```
vedang@vedang-VirtualBox:/$ sudo mount -o loop ext2_image.img /mnt/ext2dir/
vedang@vedang-VirtualBox:/$ cd /mnt/ext2dir/
vedang@vedang-VirtualBox:/mnt/ext2dir$ ls -ail
total 102560
     2 drwxr-xr-x 3 root root     4096 Nov  6 22:38 .
393219 drwxr-xr-x 3 root root     4096 Nov  6 22:25 ..
    14 -rwxr-xr-x 1 root root    17184 Nov  6 22:31 a.out
    13 -rw-r--r-- 1 root root      197 Nov  6 22:31 gen.cpp
    11 drwx------ 2 root root    16384 Nov  6 22:25 lost+found
    15 -rw-r--r-- 1 root root 52428800 Nov  6 22:38 restored_file_50MB
    12 -rw-r--r-- 1 root root 52428800 Nov  6 22:38 test_cpy
vedang@vedang-VirtualBox:/mnt/ext2dir$
```

12. To check whether the restored file is exactly the same as the original file, I did also create a test_cpy file by **sudo cp test test_cpy** before deleting test. Then we can use **diff restored_file_50MB test_cpy** to check whether the 2 files are any different or not.

```
vedang@vedang-VirtualBox:/mnt/ext2dir$ ls -ail
total 102560
     2 drwxr-xr-x 3 root root     4096 Nov  6 22:38 .
393219 drwxr-xr-x 3 root root     4096 Nov  6 22:25 ..
    14 -rwxr-xr-x 1 root root    17184 Nov  6 22:31 a.out
    13 -rw-r--r-- 1 root root      197 Nov  6 22:31 gen.cpp
    11 drwx------ 2 root root    16384 Nov  6 22:25 lost+found
    15 -rw-r--r-- 1 root root 52428800 Nov  6 22:38 restored_file_50MB
    12 -rw-r--r-- 1 root root 52428800 Nov  6 22:38 test_cpy
vedang@vedang-VirtualBox:/mnt/ext2dir$ diff restored_file_50MB test_cpy
vedang@vedang-VirtualBox:/mnt/ext2dir$
```

This is one of the safest methods of recovering a deleted file and ensuring it has the same inode as before.

Task 1: Modifying file attributes by directly modifying the inode information.

- Firstly, unmount the disk image: **sudo umount /mnt/ext2dir**
- After this, enter into the debugfs mode with writable permissions: **sudo debugfs -w /ext2_image.img**

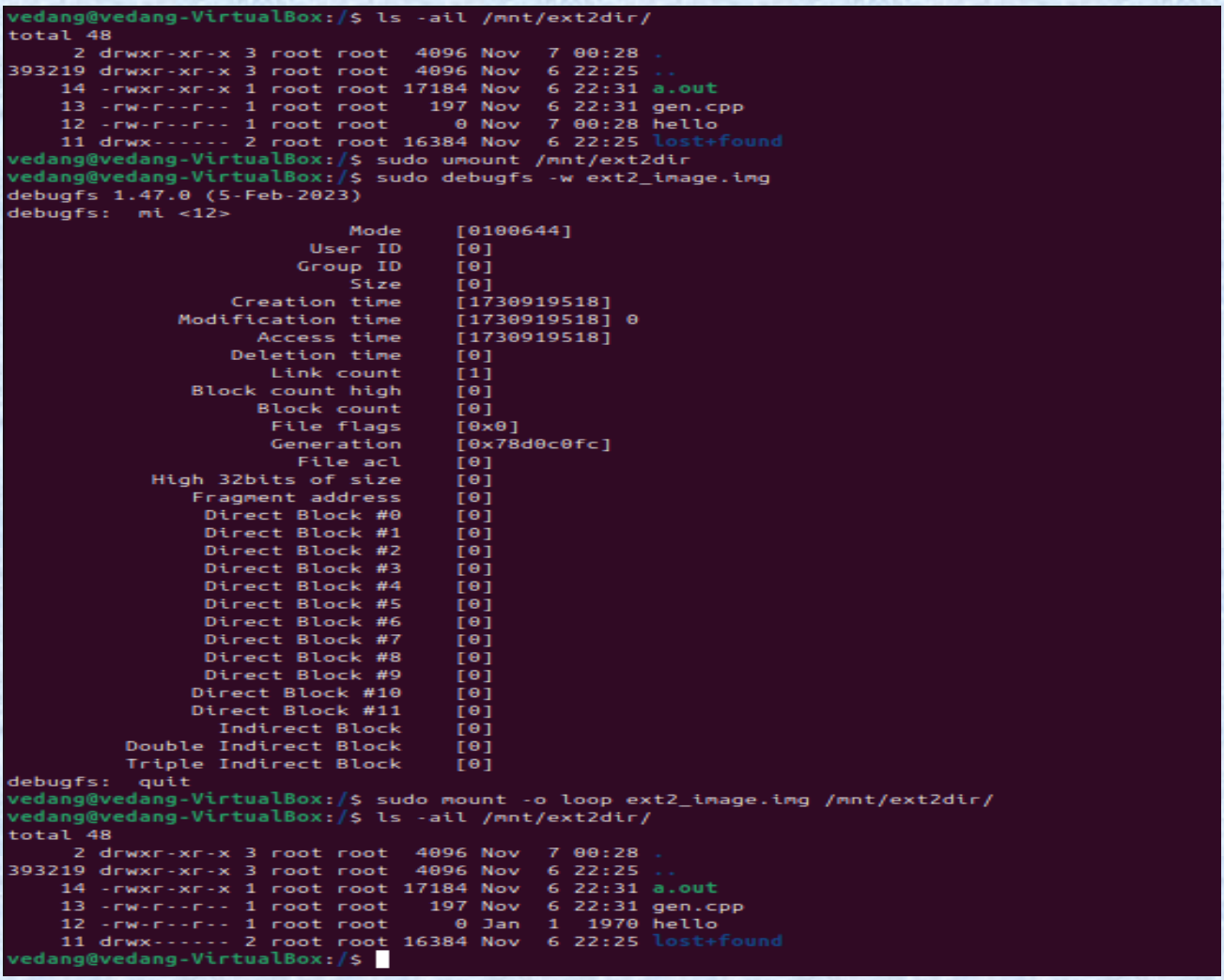
There are two ways now: We can either use the **mi <inode>** command or the **set_inode_field <inode> <field> <value>**

command to change the file attributes such as creation time, access time, modification time etc.

- **mi <inode>.** After this, set the modification time, for example, to 0. The following screenshots show the changes that are encountered.

  Clearly, the modification time for the file “hello” is set to Jan 1, 1970 (EPOCH).

- **set_inode_field <inode> <field> <value>.** The second screenshot corresponds to this command, and again changes the modification time of the file “hello” to Jan 1, 1970 (EPOCH).



mi command has been used here.

```
vedang@vedang-VirtualBox:/$ ls -ail mnt/ext2dir/
total 48
     2 drwxr-xr-x 3 root root  4096 Nov  7 00:39 .
393219 drwxr-xr-x 3 root root  4096 Nov  6 22:25 ..
    14 -rwxr-xr-x 1 root root 17184 Nov  6 22:31 a.out
    13 -rw-r--r-- 1 root root   197 Nov  6 22:31 gen.cpp
    15 -rw-r--r-- 1 root root     0 Nov  7 00:39 hello
    11 drwx------ 2 root root 16384 Nov  6 22:25 lost+found
vedang@vedang-VirtualBox:/$ sudo umount /mnt/ext2dir
vedang@vedang-VirtualBox:/$ sudo debugfs -w ext2_image.img
debugfs 1.47.0 (5-Feb-2023)
debugfs:  set_inode_field <15> mtime 0
debugfs:  quit
vedang@vedang-VirtualBox:/$ sudo mount -o loop ext2_image.img /mnt/ext2dir/
vedang@vedang-VirtualBox:/$ ls -ail mnt/ext2dir/
total 48
     2 drwxr-xr-x 3 root root  4096 Nov  7 00:39 .
393219 drwxr-xr-x 3 root root  4096 Nov  6 22:25 ..
    14 -rwxr-xr-x 1 root root 17184 Nov  6 22:31 a.out
    13 -rw-r--r-- 1 root root   197 Nov  6 22:31 gen.cpp
    15 -rw-r--r-- 1 root root     0 Jan  1  1970 hello
    11 drwx------ 2 root root 16384 Nov  6 22:25 lost+found
vedang@vedang-VirtualBox:/$
```

set_inode_field command has been used here.

Task 2: Random Access Times for the file.

The three files that are created are: test, test2, test3 with sizes 4KB, 1MB and 1GB respectively.

To measure the random-access times for the file, I am using the **time** and **dd** command as follows:

time dd if=file_4KB of=/dev/null bs=4K count=1 skip=$(RANDOM % ($(stat -c %s file) / 4096))) iflag=direct

- time – Used to measure how long, the command mentioned after it, takes to execute. It gives 3 fields: real, user and

system. Real indicates the total time from the start to the end, including any waiting time. User indicates the time that was spent in the user mode, such as executing the command. System indicates the time spent in the kernel or the supervisor mode (such as I/O). We are interested in the Real field.

- dd – This is used for low-level copying of data, from our file to a disk partition.

- if – The field that specifies the input file i.e. our file_4KB.

- of – Output file. Here, /dev/null is being since the contents that are dumped in it are lost, since we are not interested in the copied contents, but the time taken to copy.

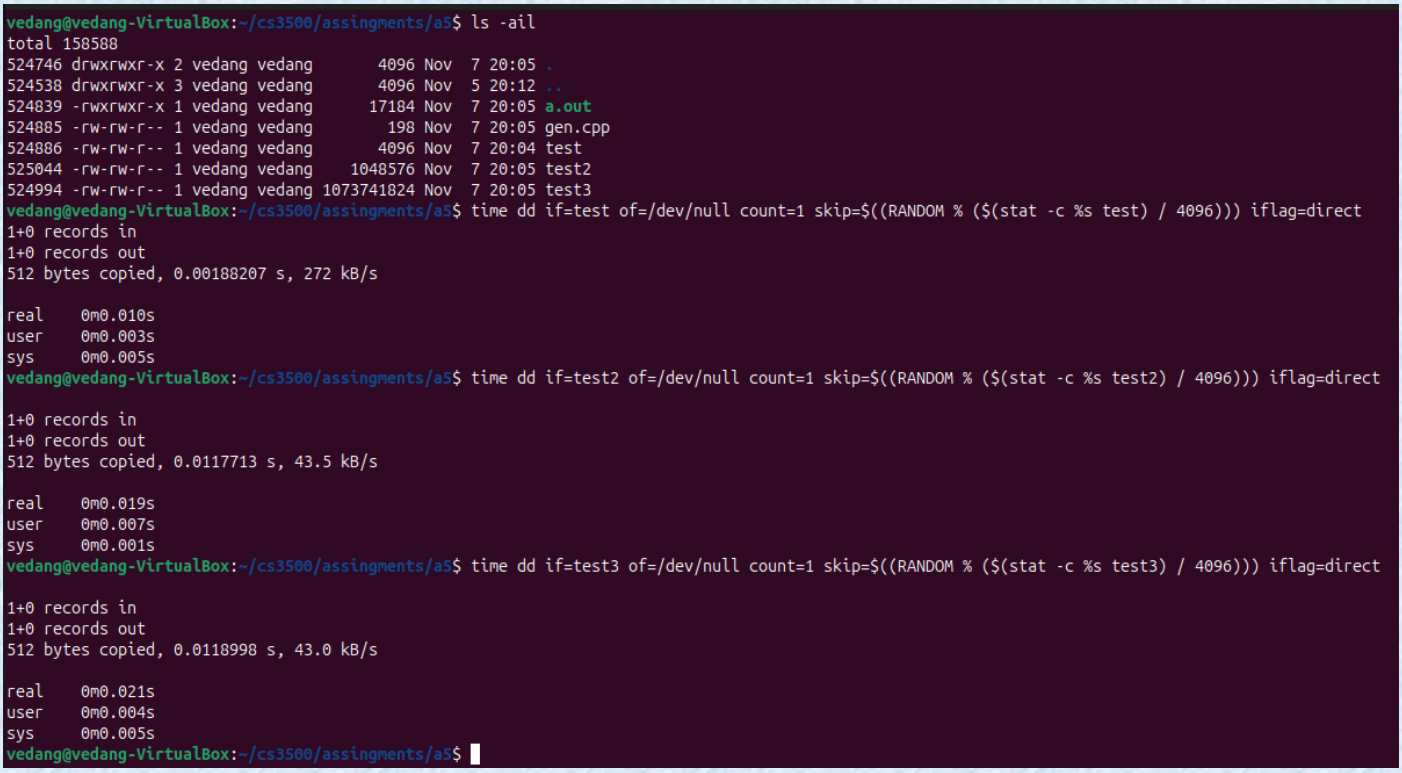
- bs – Block size, which is 4KB for my system.

- count – Specifies the number of blocks being accessed and copied. Here, I am copying only 1 block.

- skip – RANDOM is a built-in variable that returns a value from 0 to 32767. RANDOM % number of blocks, gives a random block number to be copied.

- stat -c %s <file> - Gives the filesize in bytes. Dividing this by block size = 4096 bytes gives the number of blocks.

- iflag=direct – Bypasses system cache and accesses the block directly from the hard disk.



The following are the access times for the files:

- test (4KB) – 0.010s
- test2 (1MB) – 0.019s
- test3 (1GB) – 0.021s

Here, even with caching disabled (iflag = direct), the primary reasons why the access times differ are: Disk seek time and Block read latency.

The slight increase in the access times is primarily due to the increased range of seeks with increase in file size. Since the larger file has more blocks, a random access may occasionally require slightly longer seeks. Moreover, for larger files, the filesystem may have to access additional metadata such as the

indirect blocks to locate the requested block. This, as well, contributes to a minor overhead.

However, since the block size is the same for all three cases (4KB) and we are measuring only a single random-access time, the values don't differ much. The dd command skips to a random location, resulting in roughly the same seek and read times for each file.