CS314 : Operating Systems Lab

Lab 10

Sourabh Bhosale(200010004)
Dibyashu Kashyap(200010013)

April 5, 2023

1 Page Replacement policies

1.1 File Creation

You can start by creating the file as an immediate one. When a file grows beyond 32B, then you can make it a regular file.

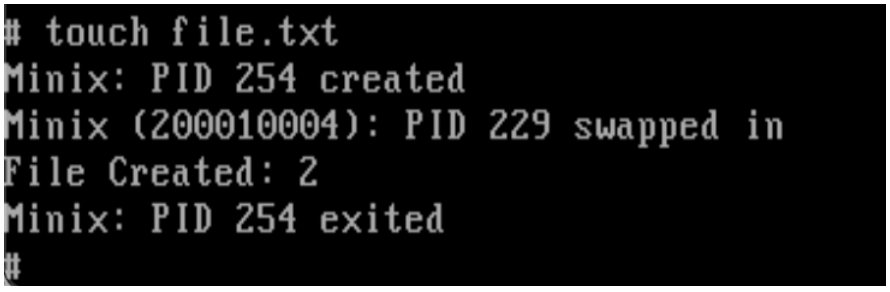

Figure 1: File Creation

For this, the changes were made in the common_open() function inside the file `open.c` present at location: minix/servers/vfs/open.c

From figure 1, we can see that we created a new file using "touch file.txt" command.

1.2 File Read

If it is an immediate file, you can respond with the inode structure contents. If not, you can follow the default behavior of looking up zones.
Firstly, we have "file.txt" which is not an immediate file.

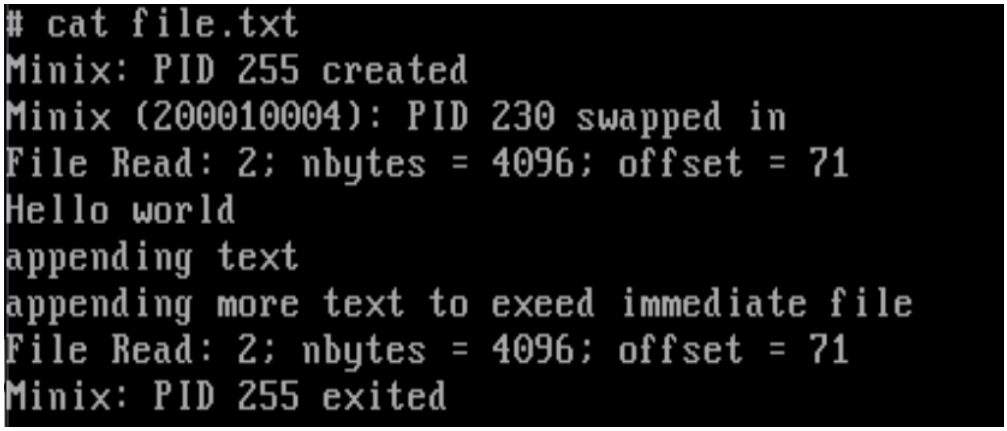

Figure 2: File Read

From figure 2, we can see that we read the file using "cat file.txt" command. Since this is not an immediate file, required output is shown according to default behavior like File Read: _ ; nbytes = _ ; offset = _ etc.

Now, we also have "newfile.txt" which is an immediate file.

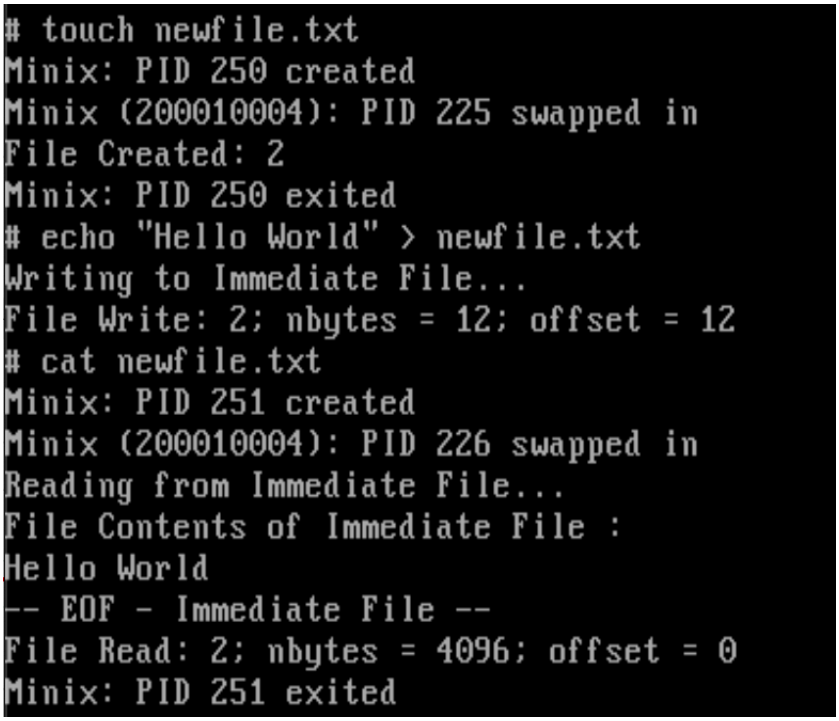

Figure 3: File Read

From figure 3, we can see that we created new file "newfile.txt" and read the file using "cat newfile.txt" command. Since this is an immediate file, required output is shown like Reading from immediate file, File contents of immediate file, EOF - immediate file etc.

For file read part, the changes were made in the fs_readwrite() function inside the file `read.c` present at location: minix/fs/mfs/read.c and in the read_write() function inside the file `read.c` present at location: minix/servers/vfs/read.c

1.3 File Write

Similar to read, You must take care to ensure that if you want to write to the inode structure, then the new file size is still within 32B. When a regular file shrinks to less than 32 bytes, there is no need to come back to immediate mode.

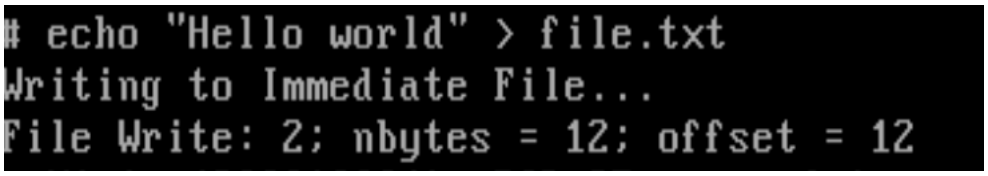

Figure 4: File write

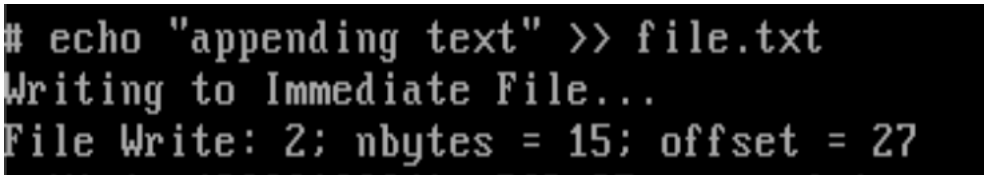

Figure 5: File write

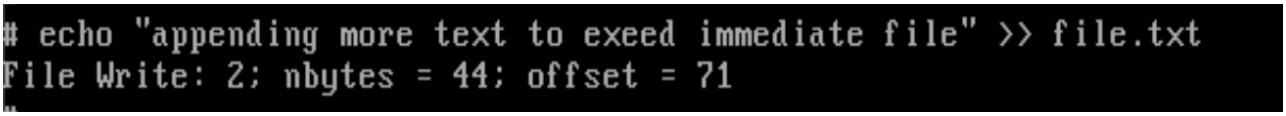

Figure 6: File write

For this, the changes were made in the fs_readwrite() function inside the file `read.c` present at location: minix/fs/mfs/read.c and in the read_write() function inside the file `read.c` present at location: minix/servers/vfs/read.c

From figure 4 and figure 5, we can see that we wrote into the file.txt using "echo "Hello World" > file.txt" command and later appended the text to it using "echo "appending text" > file.txt", even after that is still remains immediate file as its still less than 32B, so it displays the output like Writing to Immediate File... But if append more text to it (refer figure 6) and its size becomes more that 32B, it will be regular file, so in that case it displays default output like File Write : _ ; nbytes = _ ; offset = _

1.4 File Delete

Deleting immediate files does not require any handling of zones.

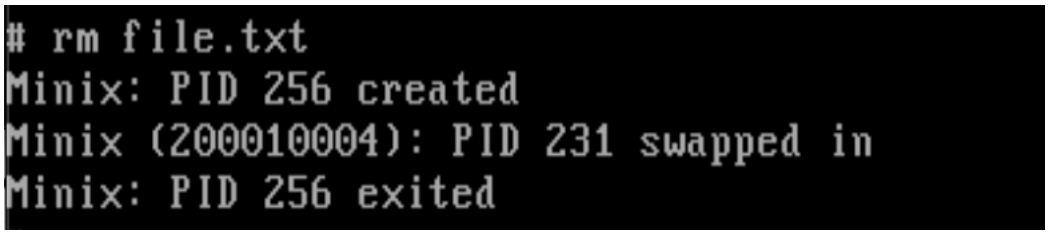

Figure 7: File Delete

As assignment does not require any handling of zones we doesn't have to make any changes anywhere. If we want to show output like File Deleted … then changes could have been made into do_unlink() function inside the file `link.c` present at location: minix/servers/vfs/link.c just like previous assignment.
From figure 7, we can see that we deleted a file using "rm file.txt" command.