Design Document Machine Problem 7:
Vanilla File System

Operating Systems : MP 7 **Rajendra Thottempudi**
UIN : 933004901

Design :

In this machine problem, we are basically asked to implement a simple file system. These files support sequential access only. This file system is to be implemented in classes FileSystem and File. All the basic functions required to manage a system of files are provided by the class File System and all the details related to the file are present and are managed by various methods in the File class. In particular, the methods corresponding to reading a file, writing a file, updating a file and other file operations are present in the file class. We also have an inode structure that we should use to manage the files on the disk. It contains the details of the file that it points to in particular, because we use id as an identifier for a file, we have the id in the inode to know the file that it is pointing to.

Let us look at the various methods in the file system :

1. Constructor : The constructor of the file system is used to initialize the various variables that we have in the file system such as the pointer to the simple disk in order to access it for any operation related to any of the files. We maintain a free list which contains the list of all the free blocks which are available and where can be used whenever we need to create a new file.

2. Mount : In this method, we assign the disk pointer which we later use to access the disk i.e. we mainly allocate the disk object as the File System's disk.

3. Create File : In this function, we first search if we have an existing file with that id and if it exists we throw an error stating that we already have a file with that id. If we don't have any such file, we then create a new file with this file id. We find a free block in the list of free blocks that we have - use that to hold this new file's information and then mark that block as used.

4. Lookup File : In this method, we tend to see if we have a file in our file system with an existing file id. This is done by first looking at the blocks that are used and checking for the file id in them to see if any of the used blocks that we have in our file system has the information related to this particular file id. If yes, then that indicates that we have the file in our file system and we return the file corresponding to that id. If no, we simply print that the file with that id doesn't exist in our file system and return null.

5. Delete File : In this particular method, we receive a file id as one of the parameters of this method and we are expected to delete the file with that particular id from our file system. We first check if such a file id exists or not. While checking for the file by going through the used blocks, if we do not find out we just print that the file with that id doesn't exist. If found, we delta that file by de-allocating the memory associated with it and resets the block assigned to the file id.

Now that we have the file system, we can modify the functions present in the File class.

Various methods in the implementation of Class File and its details are as follows :

1. Constructor : The constructor of the file class is used to initialize the values of the private variable of a file.

2. Read : In this method, we want to read the contents of a file. This is achieved by reading the required contents of the file and placing them in the buffer. If the read pointer reaches the end of the block, since one file can fit into one block, it indicates that the contents of the file are completed and we return.

3. Write : In this function, we want to update the contents of a file. We write the number of bytes requested into the buffer and to the disk. Once the writing is done, we reset the pointer to point to the beginning of the file.

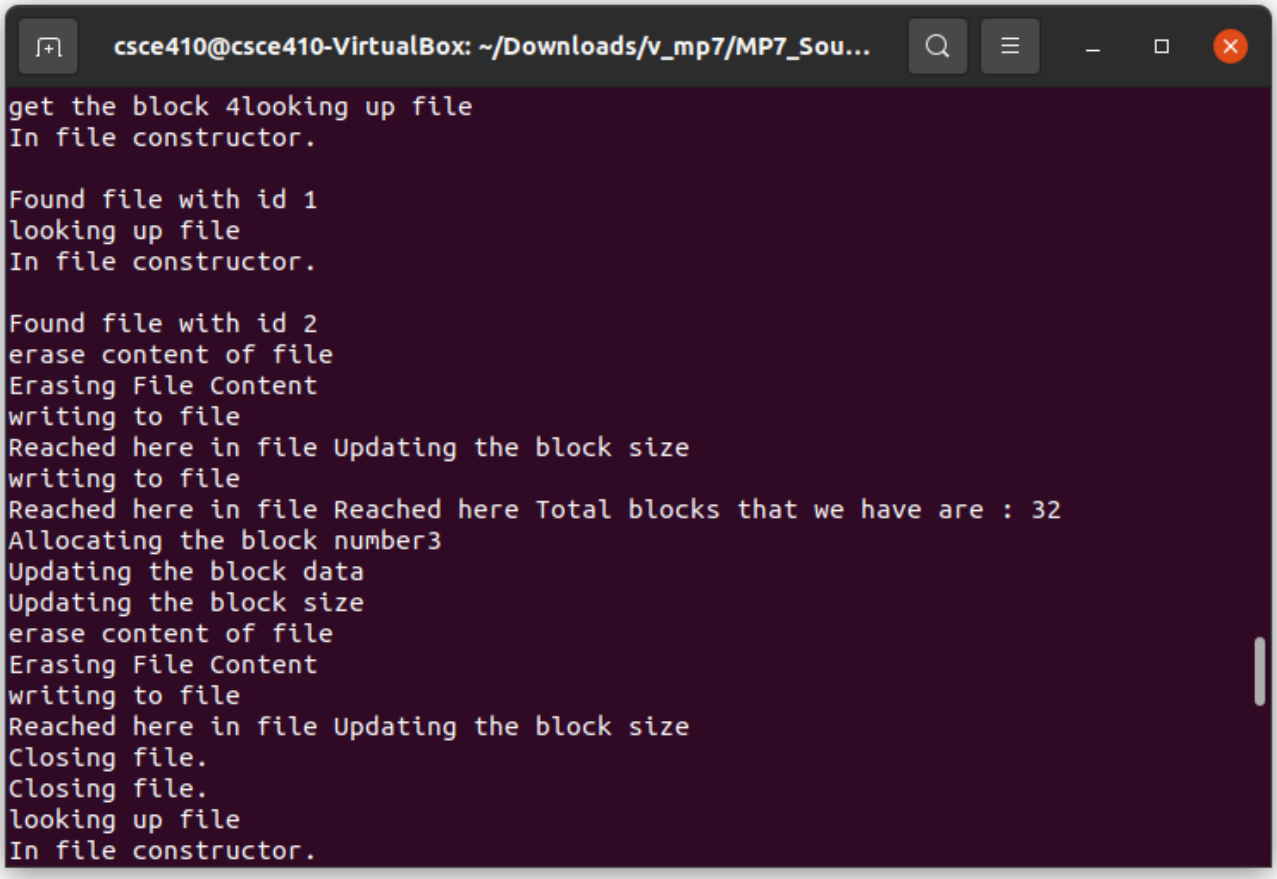
4. Reset : In this particular method, we simply put the pointer of the file to point to the beginning of the file.

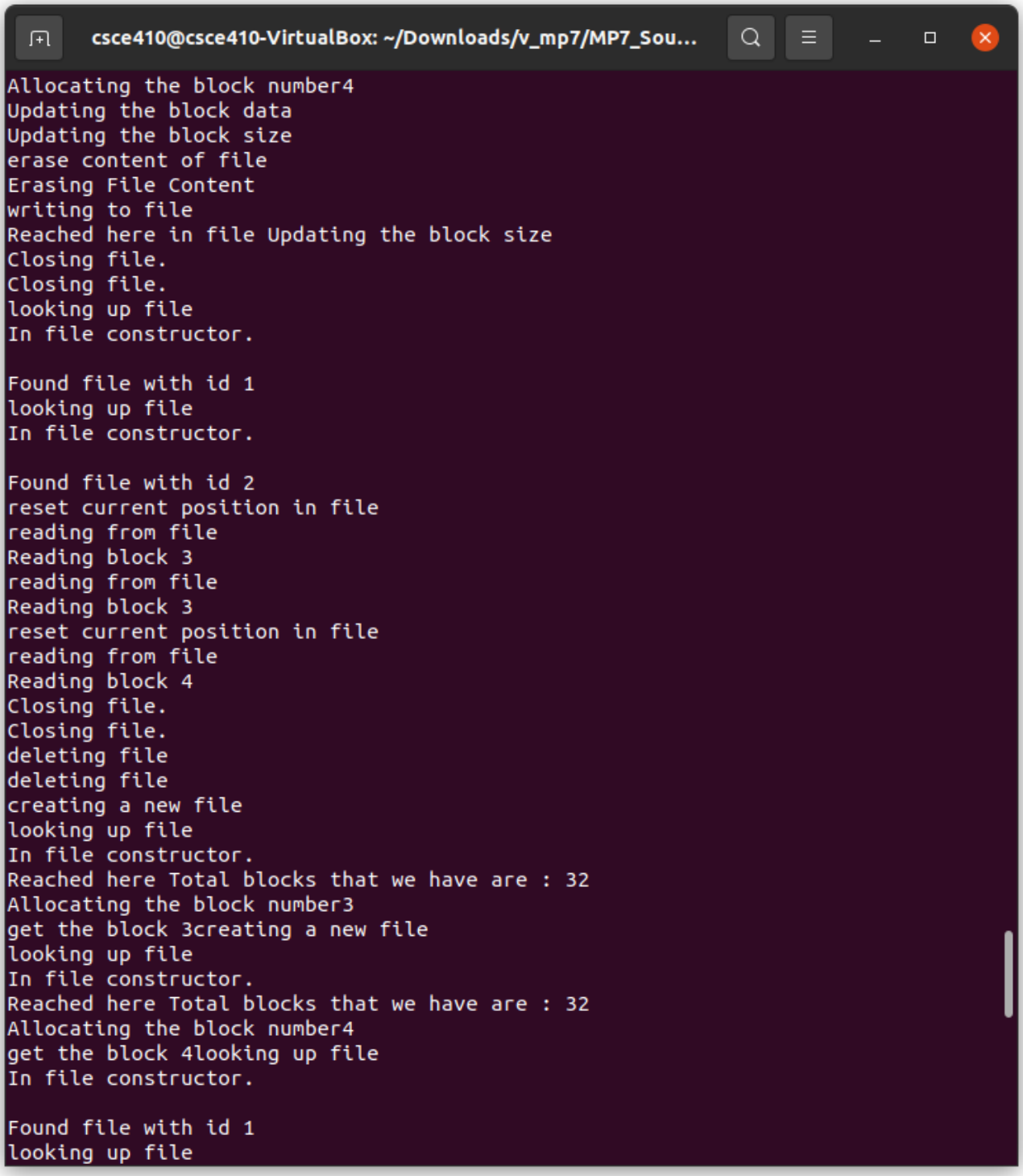
I also implemented certain utility api's in both the file and the file system classes to help me perform certain functions which have to be performed repeatedly in multiple places within the classes.

Bonus 1 :

Currently we are assuming that the maximum size of a file is of maximum one block (512 Byte) which was a major limitation of the file system described. If we want to extend this basic file system to allow for files that are up-to 64 kB long - we have to find a way to improve how we are currently storing the files to accommodate for the extra memory and for the continuity in read and write operations. One way to achieve this is to maintain a list of blocks instead of a single block to maintain the data of the file. If we do this, we then have to keep the size of the file and the pointer to the initial block of the file in the inode. Now, once we do this, we have to change the way read and write operations of the file are implemented. Instead of reading it from a single block, we have to iterate through the list of blocks to update the file pointer and it is the same when it comes to write operation except that in write we keep updating the data of the file.

Please find the screenshots attached below :


csce410@csce410-VirtualBox: ~/Downloads/v_mp7/MP7_Sou...
get the block 4looking up file
In file constructor.
Found file with id 1
looking up file
In file constructor.
Found file with id 2
erase content of file
Erasing File Content
writing to file
Reached here in file Updating the block size
writing to file
Reached here in file Reached here Total blocks that we have are : 32
Allocating the block number3
Updating the block data
Updating the block size
erase content of file
Erasing File Content
writing to file
Reached here in file Updating the block size
Closing file.
Closing file.
looking up file
In file constructor.


csce410@csce410-VirtualBox: ~/Downloads/v_mp7/MP7_Sou...
Allocating the block number4
Updating the block data
Updating the block size
erase content of file
Erasing File Content
writing to file
Reached here in file Updating the block size
Closing file.
Closing file.
looking up file
In file constructor.
Found file with id 1
looking up file
In file constructor.
Found file with id 2
reset current position in file
reading from file
Reading block 3
reading from file
Reading block 3
reset current position in file
reading from file
Reading block 4
Closing file.
Closing file.
deleting file
deleting file
creating a new file
looking up file
In file constructor.
Reached here Total blocks that we have are : 32
Allocating the block number3
get the block 3creating a new file
looking up file
In file constructor.
Reached here Total blocks that we have are : 32
Allocating the block number4
get the block 4looking up file
In file constructor.
Found file with id 1
looking up file