Anuska Pant
CSCE 611 - OPERATING SYSTEMS
Machine Problem 7: Vanilla File System

Introduction
In this machine problem you will implement a simple file system. Files support sequential access only.

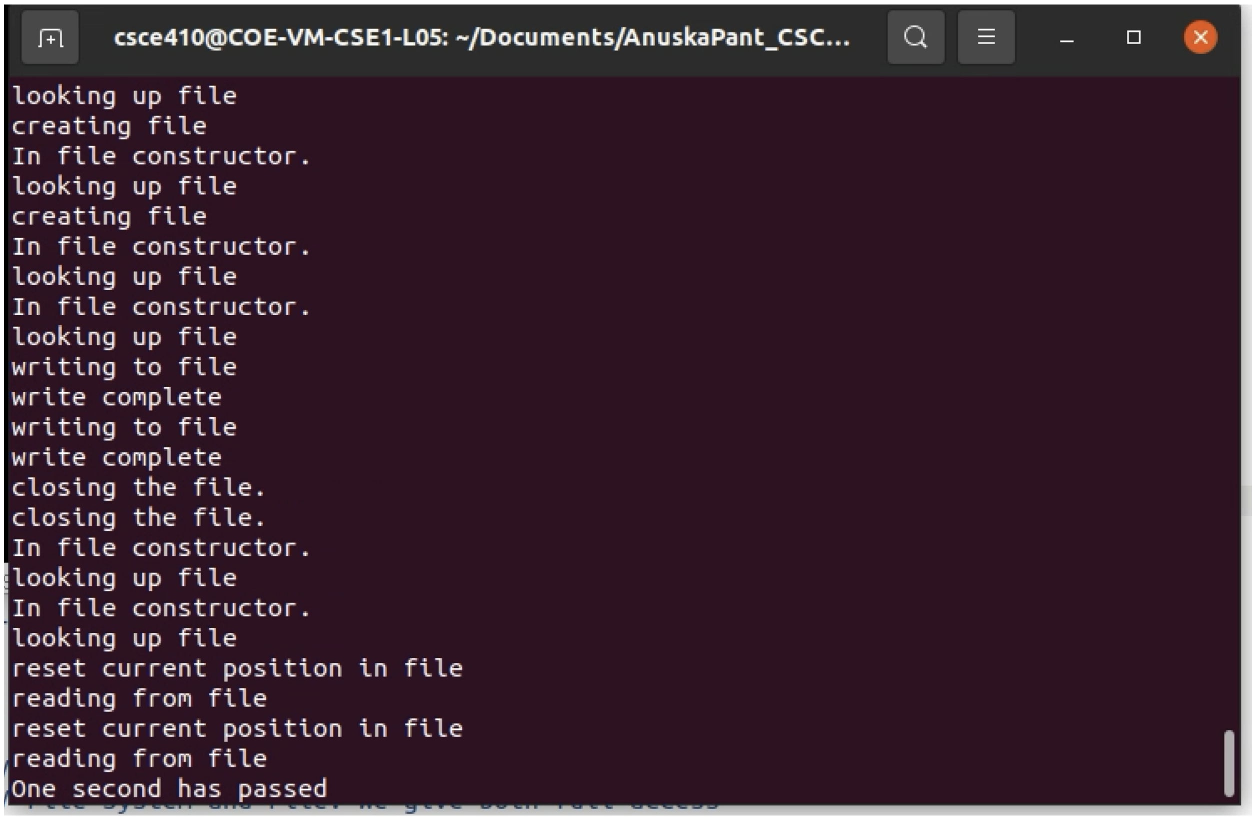
Approach:
Here we use two classes together to implement the vanilla file system, the File System and File class. The file system is responsible for assigning file name space to files. It also handles the free block management and allocation of files. The File contains functions for reading a file, writing a file and resetting the file point to the start so we can read again after the write operation. We initialize the disk with three blocks, where one is the super block and contains details about the file system such as the number of files and size. The second block is the directory block which is used to map the inodes with a file descriptor and the third block is the free block and rest of the blocks are used as data blocks. Each disk block is 512 bytes. After initializing all the blocks we track the first freeblock and initialize the rest of the blocks as free indicating the disk is empty.
The file system contains a linked list of files allocated in the disk and updates the value of free block number to next free block. Reading and writing a file involves accessing the startblock information and its size and reading from the disk, while writing involves writing to the disk and increasing the size. Similarly when a file is deleted, its entry in the linked list is removed, and its blocks are freed and the free block number is correspondingly updated. The next sections talk about the various interfaces in File System and File class.

FILESYSTEM CLASS:
Mount: This function reads the disk and initializes the FILE SYSTEM variables with the content in the disk and mainly allocates the disk object as the FILE SYSTEM's disk.
CreateFile: This function creates a file, with the _file_id, ie. It first extracts the available free block in the File System, uses it as the Info block for the file, and from the last 4 bytes of this info block, it gets the next sequentially available free block. These two blocks are assigned as the starting info block and starting data block for the new file and are used to create a File object which holds these parameters local to its object which represents each file. Further, we also maintain a linked list of the files that are created in the File system and allocate this file object in the Linkedlist too.
AddToInode: The File Node is of file_blockNode type which contains parameters such as file ID, file, next pointer.
LookupFile: This function is used to Lookup if a file with the given FILE ID exists, in the File system. We do this by checking for the file ID in the Linked List of the files we store to represent

the file List in the file system. If we find the node with the input file ID, we return the file corresponding to that node.
DeleteFile: In this function, we delete the file with the input file ID, i.e. we deallocate the memory allocated to it as info and data blocks and also remove its entry from the file linked list.

FILE CLASS:
Read: This function reads _n bytes from the given file based on the current position and the position of bytes being read everytime.
Write: This function writes _n bytes to the file in the disk. While writing , again it maintains a set of counters to track the current position in the file etc.
Reset: This function resets the current position of the file to the start of the file.
Rewrite: This function erases the contents of the file and updates the freeblock value and the start block value of the file.
EoF: This function is used to tell if the end of the file is reached.
The implementation of the file system was tested in kernel.C and the code didn't assert.

Screenshot of the output is below.



```
looking up file
creating file
In file constructor.
looking up file
creating file
In file constructor.
looking up file
In file constructor.
looking up file
writing to file
write complete
writing to file
write complete
closing the file.
closing the file.
In file constructor.
looking up file
In file constructor.
looking up file
reset current position in file
reading from file
reset current position in file
reading from file
One second has passed
```