Abraham Schultz
Temple CIS 3207
Lab 4

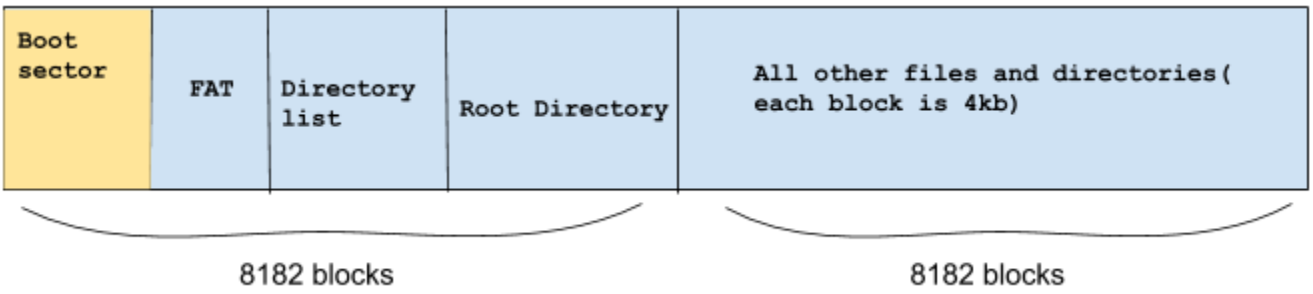
Virtual Disk Design Document

DESIGN OVERVIEW

This is my design for a virtual disk. This disk will reside on top of the existing linux file system within a single file.

I will use a FAT as the basis for my virtual file system.

1. The maximum file size will be 32 mb which will be all of the blocks. Each block is 4KB or 4096 bytes. There will be 16,384 blocks but only half of them need to be used for storing data files. This Means that we should have **64 MB total** for our virtual disk, half of that being allocated for storing files.



2. The FAT will act as one large central table to store information about the location of each new file we would like to add. It should be Essentially a list of linked lists. Where each linked list within is referencing the location of the blocks needed to store a particular file.

3. If we create a new file, first we will need to check if the size of that file is small enough to fit into one block. If so then we can assign just one block add add that block to a list on the FAT. Otherwise if the file requires more than one block we will need to break the file up into multiple blocks. The FAT will record these files as a linked list with the starting positions of each required block as an entry in that files linked list. In this way we can store a file in a non contiguous way taking advantage of gaps in the memory.

4. Every time a new file is created we will subtract one from the available disk count. We will also check for the current offset (that is the starting position in the disk memory to start writing to) we will then begin writing to the disk one block at a time assigning each block the starting position of the current offset making sure to check if the next block is used or not. If it is used then we assign the block and mark as used and update the current offset, otherwise we skip to the next block.

STRUCTS FOR PARTITIONS OF VIRTUAL DISK

The FAT Struct will look as follows it will be our main struct, which will keep track of files open and the blocks being used.

Struct FAT{

```
Int TotalBlocks = 16384

Int UnusedBlocks;  // start at total blocks and subtract from as we go

Vector <File > FAT // a linked list containing linked lists of blocks.
```

} // struct FAT

The file struct will be a linked list of blocks

Struct File{

```
vector <Block> blockList;      // this will contain the blocks holding the data

char  filename[15];            // name of file

int FileSize;         //  = blockList.size * BLOCK_SIZE;

int filePointer;      // pointer to current data byte

int filedes;          // file descriptor , -1 if closed should always be closed after unmount

int startingAddr;     // address of first block used by this file

int isDir;            // flag 0 if dir 1 if regular file

char  parent[15];             // parent of file or directory none if root.
```

} // end Struct File

The Block struct will be an element in the

Struct Block {

Char DATA [4000] // these file will only hold text for now. 4000 bytes = 4KB

Int addrNextBlock; // point to next block used -1 if not used

Int blocknum

} // end Struct Block

// used to initialize super block

PSEUDO CODE AND FUNCTIONS

// the main function will contain a central while loop that has a simple shell, i will probably reuse some code from lab 2.

#declare constants and macros IE size of disk ect

Main() {

intitlize ();

While (running == true) {

*Char * buff // hold user input*

Getline ();

Big switch case

Switch ()

Case open()

….

Rest of the functions listed below will each be there own case

Default

Case unrecognized command try again

}

} // end while

} // end main

int make_fs(char *disk_name); {

MakeDisk(disk_name)

//create new FAT instance

Struct FAT = new FAT();

FAT.totalBlocks = TOTALBLOCKS

Struct FILE rootDir; // create root directory on lowest memory blocks

***} //** end make_fs*

int mount_fs(char *disk_name); {

// check if disk exists

If it exists

Open disk (disk_name)

Else

Error message

// we need to make the selected virtual disk available to edit and read and write from

***} //** end mount*

int umount_fs(char *disk_name); {

Closedisk (disk_name)

***} //** end unmount*

int fs_open(char *name);{

For (each item in FAT)

{

If FAT(i).name = name{

// open file

Return 1;

}

}

Return -1 // only here if no file found

}

int fs_close(int fildes);

```
exist or is not open, the function returns -1.
int fs_create(char *name){
    File tmpFile = File;
    File.name = name;
    //first check if we have space
    If (FAT unusedBlocks > 0) {
        FAT.insert(name)

    } // else we have no space
    printf("not enough space on disk \n")
        };
int fs_delete(char *name);{
//check if file exists
For (each item in FAT)
    {
    If FAT(i).name = name{
        FAT.erase(i)
        Return 1;
        }
    }
    Return -1 // only here if no file found

}
int fs_mkdir(char *name);{
  // add new file to FAT with is dir bit set to 0
 // and filename set to name
}// end fs_mkdir
int fs_read(int fildes, void *buf, size_t nbyte);{
// wrapper function for one provided by professor
}
int fs_write(int fildes, void *buf, size_t nbyte);{
```

This function should write some data to the designated file

it will first check the nbyte size to calculate how many blocks we need

then we will check if there are enough open blocks

if not return 0

if there are we will then assign blocks to the file vector the starting from the lowest order blocks.

We then will write the data 1 block at a time using the block write function

}// end fs_write

int fs_get_filesize(int fildes); {

For (each element int FAT) {

// check if file exists

If (FAT(i) = fildes){

Return FAT(i).fileSize;

} // end if

} // end for

***}** // end fs_get_filesize*

int fs_lseek(int fildes, off_t offset);{

***}** //end fs_lseek*

int fs_truncate(int fildes, off_t length);{

***//**check if file exists*

// if file exists then check how big it its

// if it is bigger than the length we must free the extra blocks

// if file does not exist then return error message and -1

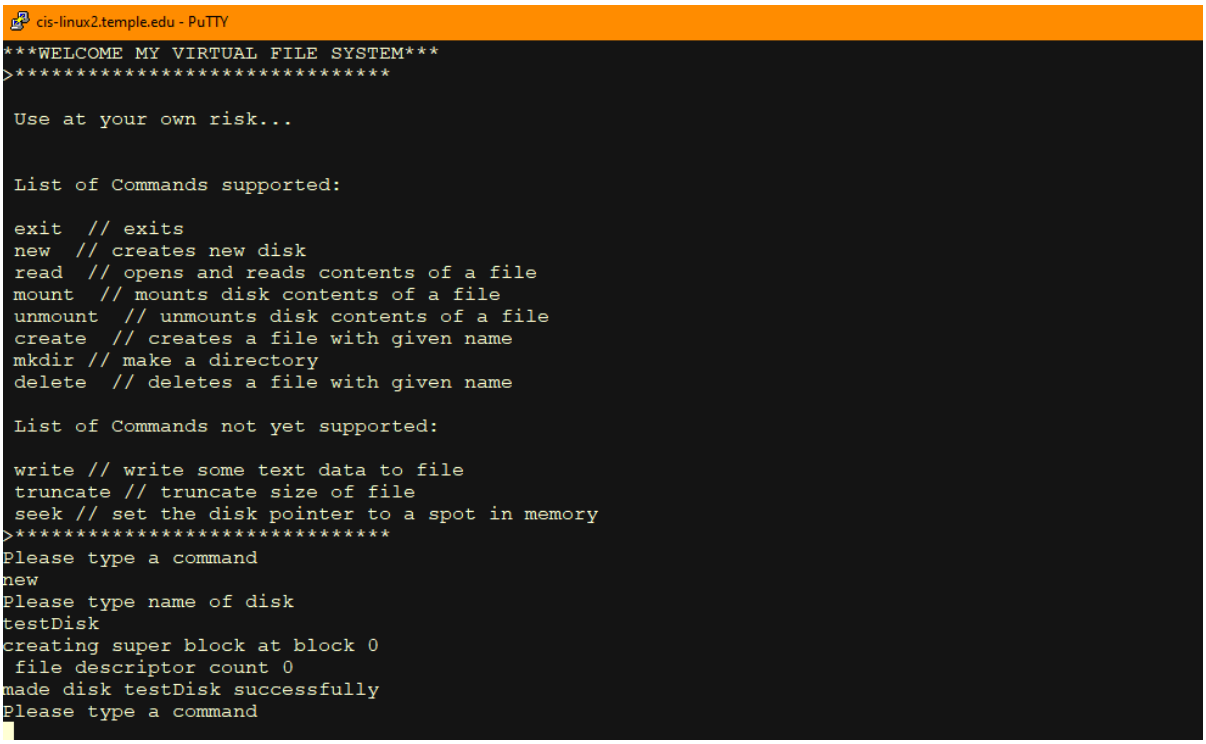
***}** //end fs_truncate*

UNIT TESTS

We should be concerned generally with creating and editing files. Additionally, another important feature is persistence, that is this virtual disk should be able to be run, save files, then exit. And when i return they should still be there.

Below are some tests that I ran to verify that the program is fulfilling the project requirements:

1. Run shell and accept command to create filesystem

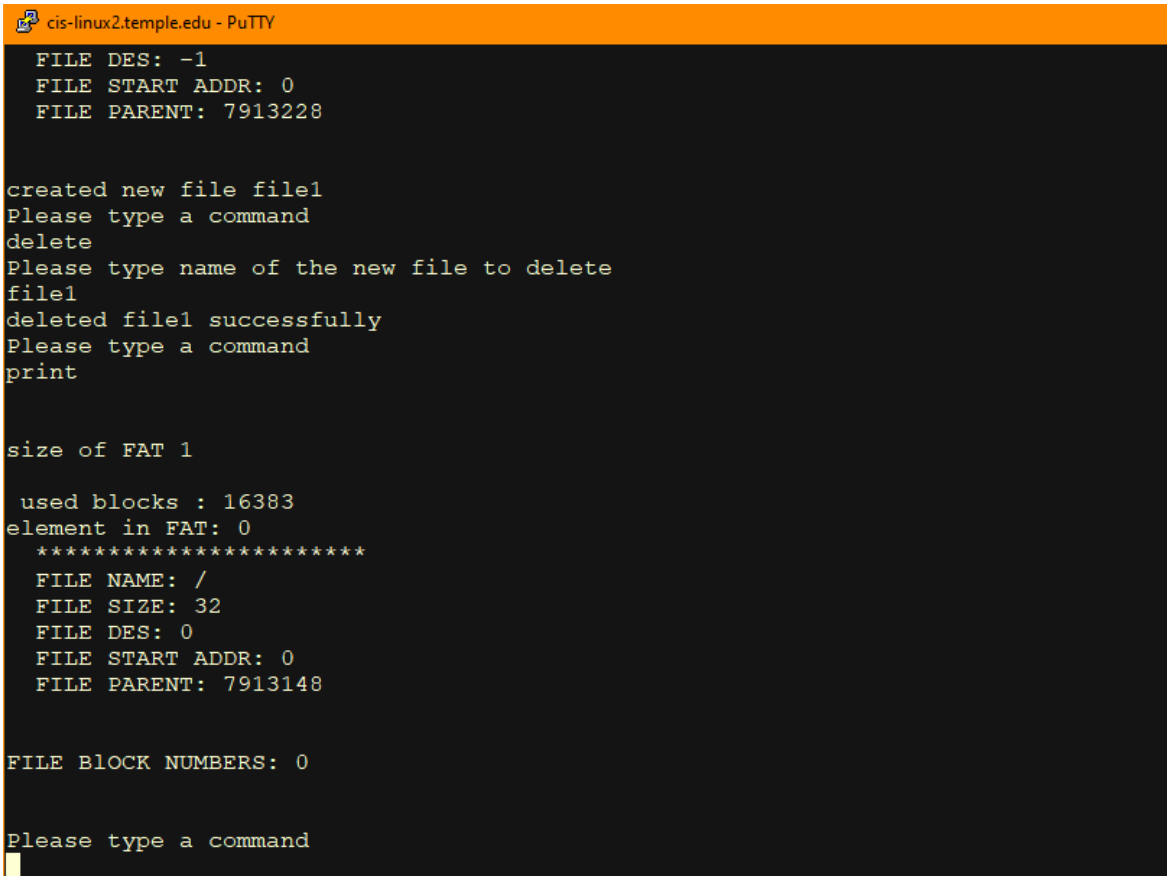


In the above test I make a new disk and run the shell. I use the hex dump on the new disk to confirm it's the correct size.

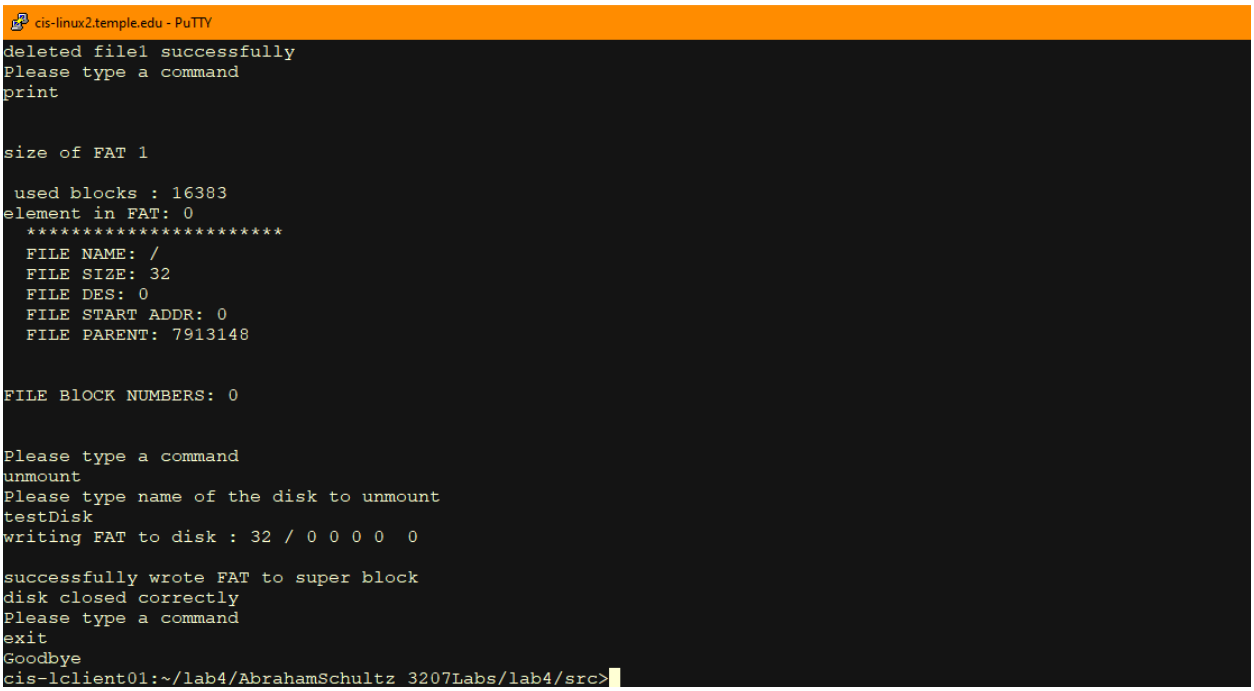
2. Accept command Mount file system. This only opens the disk as of now, it does not read the meta data from the FAT back in to memory.
3. Create file , the below image shows a file and directory being created. You can use the cd and ls commands to navigate.

```
/home/TU/tuk85386/lab4/AbrahamSchultz_3207Labs/lab4/src >>  new
Please type name of disk
testDisk
creating super block at block 0
 file descriptor count 1
made disk testDisk successfully
/home/TU/tuk85386/lab4/AbrahamSchultz_3207Labs/lab4/src >> / create
Please type name of the new file to make
file1
/ is the cwd
created new file file1
/home/TU/tuk85386/lab4/AbrahamSchultz_3207Labs/lab4/src >> / mkdir
Please type name of the new directory
folder
created new file folder
/home/TU/tuk85386/lab4/AbrahamSchultz_3207Labs/lab4/src >> / print


size of FAT 3

 used blocks : 16383
element in FAT: 0
  **********************
  FILE NAME: /
  FILE SIZE: 32
  FILE DES: 0
  FILE ISDIR: 0
  FILE START ADDR: 0
  FILE PARENT:


FILE BlOCK NUMBERS: 0


element in FAT: 1
  **********************
  FILE NAME: file1
  FILE SIZE: 0
  FILE DES: -1
  FILE ISDIR: 1
  FILE START ADDR: 0
  FILE PARENT: /


element in FAT: 2
  **********************
  FILE NAME: folder
  FILE SIZE: 0
  FILE DES: -1
  FILE ISDIR: 0
  FILE START ADDR: 0
  FILE PARENT: /
```

4. Accept command to Open file
   Open file works by assigning a new file descriptor to the file. It defaults to -1 when closed
5. Edit file
6. Delete file


cis-linux2.temple.edu - PuTTY
FILE DES: -1
FILE START ADDR: 0
FILE PARENT: 7913228
created new file file1
Please type a command
delete
Please type name of the new file to delete
file1
deleted file1 successfully
Please type a command
print
size of FAT 1
used blocks : 16383
element in FAT: 0
****
FILE NAME: /
FILE SIZE: 32
FILE DES: 0
FILE START ADDR: 0
FILE PARENT: 7913148
FILE BlOCK NUMBERS: 0
Please type a command

7. Exit


cis-linux2.temple.edu - PuTTY
deleted file1 successfully
Please type a command
print
size of FAT 1
used blocks : 16383
element in FAT: 0
****
FILE NAME: /
FILE SIZE: 32
FILE DES: 0
FILE START ADDR: 0
FILE PARENT: 7913148
FILE BlOCK NUMBERS: 0
Please type a command
unmount
Please type name of the disk to unmount
testDisk
writing FAT to disk : 32 / 0 0 0 0 0
successfully wrote FAT to super block
disk closed correctly
Please type a command
exit
Goodbye
cis-lclient01:~/lab4/AbrahamSchultz_3207Labs/lab4/src>

```
/home/TU/tuk85386/lab4/AbrahamSchultz_3207Labs/lab4/src >> folder unmount
Please type name of the disk to unmount
testDisk
writing FAT to disk : 32 / 0 0 0 0  0 -0 file1 0 -1 0 1 /  -0 folder 0 -1 0 0 /
 -0 file2 0 -1 0 1 folder  -
successfully wrote FAT to super block
disk closed correctly
/home/TU/tuk85386/lab4/AbrahamSchultz_3207Labs/lab4/src >> folder exit
Goodbye
writing FAT to disk : 32 / 0 0 0 0  0 -0 file1 0 -1 0 1 /  -0 folder 0 -1 0 0 /
 -0 file2 0 -1 0 1 folder  -
block_write: disk not active
successfully wrote FAT to super block
close_disk: no open disk
cis-lclient05:~/lab4/AbrahamSchultz_3207Labs/lab4/src>hd testDisk
00000000  33 32 20 2f 20 30 20 30  20 30 20 30 20 20 30 20  |32 / 0 0 0 0  0 |
00000010  2d 30 20 66 69 6c 65 31  20 30 20 2d 31 20 30 20  |-0 file1 0 -1 0 |
00000020  31 20 2f 20 20 2d 30 20  66 6f 6c 64 65 72 20 30  |1 /  -0 folder 0|
00000030  20 2d 31 20 30 20 30 20  2f 20 20 2d 30 20 66 69  | -1 0 0 /  -0 fi|
00000040  6c 65 32 20 30 20 2d 31  20 30 20 31 20 66 6f 6c  |le2 0 -1 0 1 fol|
00000050  64 65 72 20 20 2d 00 00  00 00 00 00 00 00 00 00  |der  -..........|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000080  00 00 00 00 00 00 00 00  e1 10 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
04000000
cis-lclient05:~/lab4/AbrahamSchultz_3207Labs/lab4/src>
```

Above is a hex dump of the metadata for files created in the FAT.