CS3026 Assessment 2016_2017

Friday 25 November 2016

Stefan Rudvin

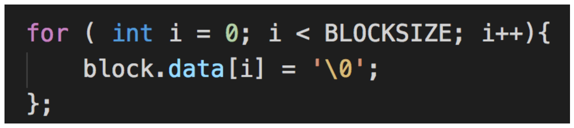
51549217

CGS D3-D1

To run this section, run ‘make’ in your console at the root of the CGS_D3_D1 folder. Then run ‘./shell’ to run the code and show its output. To view the hexdump, run ‘hexdump -C virtualdiskD3_D1’.

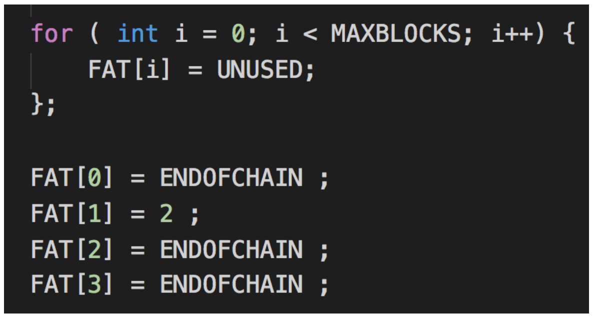
Format()

- Initializes the first four blocks of the virtual disk with the root block, FAT blocks and directory block



```
for ( int i = 0; i < BLOCKSIZE; i++){
    block.data[i] = '\0';
};
```

- Initialize an empty block by writing ‘/0’ to its every data position
- Copy string ‘CS3026 Operating Systems Assessment’ to the data section and write the block to index 0 of the virtual disk
- Prepare the FAT by making every element of the array UNUSED, then set the initial values from 0-3 which correspond to the root block, two FAT blocks and the root directory block

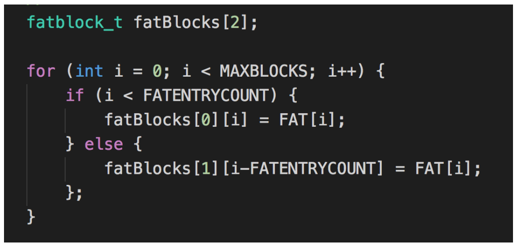


```
for ( int i = 0; i < MAXBLOCKS; i++) {
    FAT[i] = UNUSED;
};

FAT[0] = ENDOFCHAIN ;
FAT[1] = 2 ;
FAT[2] = ENDOFCHAIN ;
FAT[3] = ENDOFCHAIN ;
```

- Call copyFat(), explained below
- Prepare the root directory by writing ‘/0’ to every data position in a block
- As the block is a directory, set ‘isdir’ to 1, and initialize every entrylist entry to unused since the directory is empty.
- Set the rootDirIndex to 3, as we know it will be at this position throughout runtime
- Write the root directory into its corresponding block in the virtual disk

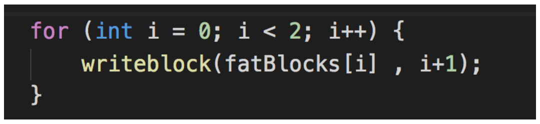
copyFat()

- Copies the global FAT object to the blocks 1 and 2 in the virtual disk.
- The FAT structure will be divided into two blocks, so create an array of two fatblock_t structures, write the first half of FAT to the first block and the second half to the second block

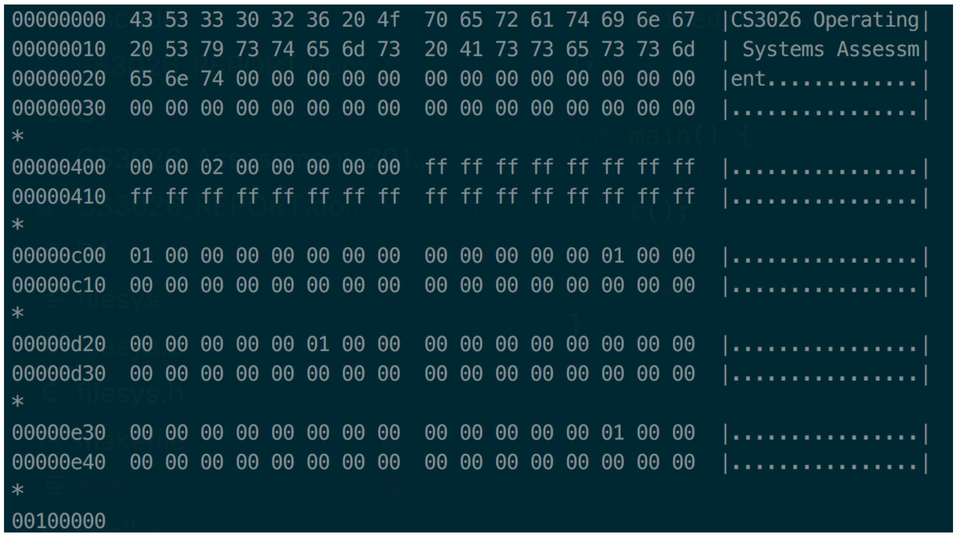


```
fatblock_t fatBlocks[2];

for (int i = 0; i < MAXBLOCKS; i++) {
    if (i < FATENTRYCOUNT) {
        fatBlocks[0][i] = FAT[i];
    } else {
        fatBlocks[1][i-FATENTRYCOUNT] = FAT[i];
    };
}
```

- Finally, cycle through the two blocks and write both of them to the virtual disk



```
for (int i = 0; i < 2; i++) {
    writeblock(fatBlocks[i] , i+1);
}
```



```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 00 00 00 00 00  00 00 00 00 00 00 00 00  |ent.............|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  ff ff ff ff ff ff ff ff  |................|
00000410  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00100000
```

hexdump -C virtualdiskD3_D1 output

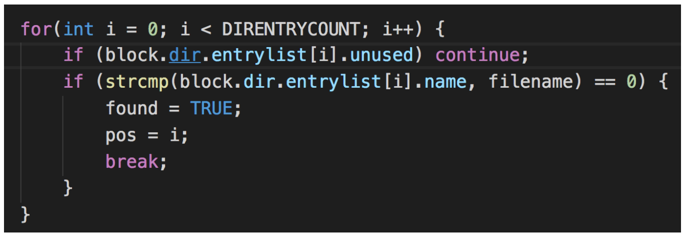
CGS C3-C1

To run this section, run 'make' in your console in the root of the 'CGS_C3_C1' folder. Then run './shell' to run the code and show its output. To view the hexdump, run 'hexdump -C virtualdiskC3_C1'.

Note: The additions in the beginning of myfopen() of part A3-A1 will be discussed in the corresponding section.

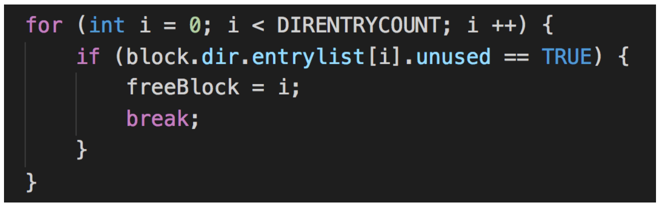
myfopen()

- Initialize MyFILE * openFile with malloc()
- Set local variables that are used by the function – found, pos, I and freeBlock
- Set block to the current directory block specified by the global variable currentDirIndex
- Loop through the entryLists of the directory block and see if the file already exists. If yes, set found to true and save the position of the entrylist



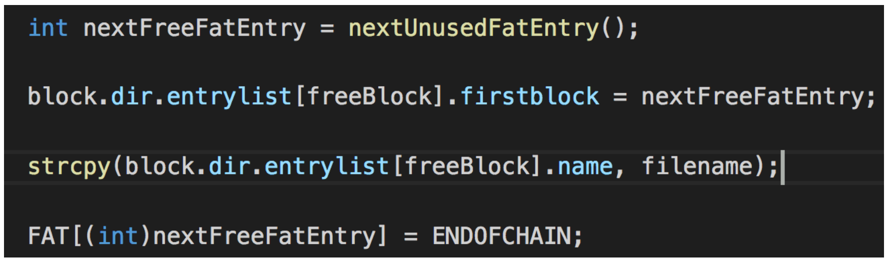
```
for(int i = 0; i < DIRENTRYCOUNT; i++) {
    if (block.dir.entrylist[i].unused) continue;
    if (strcmp(block.dir.entrylist[i].name, filename) == 0) {
        found = TRUE;
        pos = i;
        break;
    }
}
```

- If found, set the variables of openFile to correspond with the existing file
- If not found, loop through the entrylist of the current directory and find an unused block



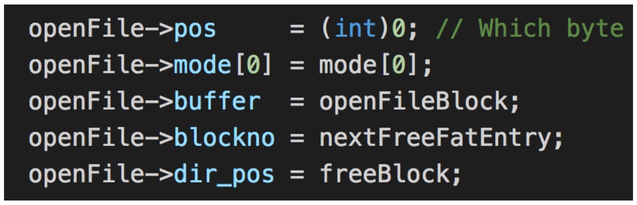
```
for (int i = 0; i < DIRENTRYCOUNT; i ++) {
    if (block.dir.entrylist[i].unused == TRUE) {
        freeBlock = i;
        break;
    }
}
```

- Set the found unused entry's 'unused' to TRUE since it is in use
- Find the next unused FAT entry, described later in this section
- Set the index of the unused FAT to ENDOFCHAIN, since it will be in use
- Set the block and name of the entry in the directory to the unused FAT index and the filename respectively



```
int nextFreeFatEntry = nextUnusedFatEntry();

block.dir.entrylist[freeBlock].firstblock = nextFreeFatEntry;

strcpy(block.dir.entrylist[freeBlock].name, filename);

FAT[(int)nextFreeFatEntry] = ENDOFCHAIN;
```

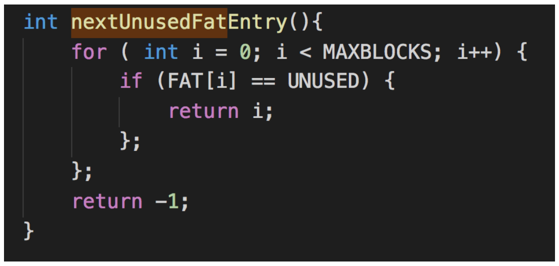
- Save the fat and directory to the virtual disk
- Add all of the gathered information to the openFile
- Return openFile



```
openFile->pos     = (int)0; // Which byte
openFile->mode[0] = mode[0];
openFile->buffer  = openFileBlock;
openFile->blockno = nextFreeFatEntry;
openFile->dir_pos = freeBlock;
```

nextUnusedFatEntry()

- Loop through all index of the FAT and return an index that is unused

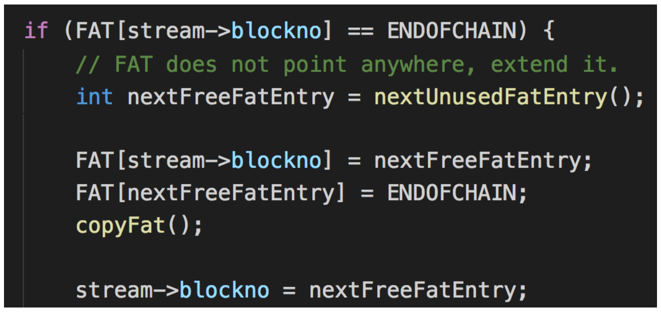


```
int nextUnusedFatEntry(){
    for ( int i = 0; i < MAXBLOCKS; i++) {
        if (FAT[i] == UNUSED) {
            return i;
        };
    };
    return -1;
}
```

Next, I created a char array of a repeating sequence of the alphabet. Each index was individually sent to myfputc(). Then I closed the file with myfclose, and then opened it again. Then I read the file letter by letter, output the results to the console and wrote them to the file ‘testfileC3_C1_copy.txt’.

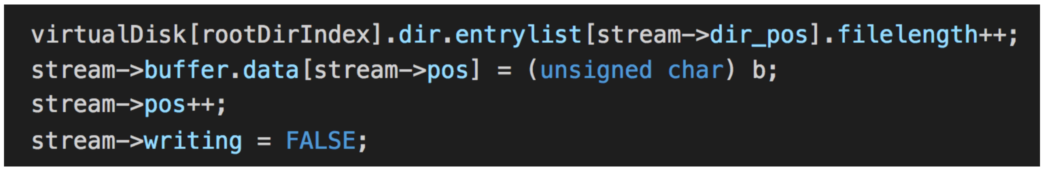
Myfputc()

- Set writing of the stream to true
- Check if the pointer is too high for the current block
- If so, write the current buffer to the virtual disk, and point to the next FAT entry
- If the next FAT entry does not exist i.e. current FAT entry == ENDOFCHAIN, create a new fat entry by finding a free fat entry and pointing the existing entry to it. Then set the block of the stream to the new fat entry



```
if (FAT[stream->blockno] == ENDOFCHAIN) {
    // FAT does not point anywhere, extend it.
    int nextFreeFatEntry = nextUnusedFatEntry();

    FAT[stream->blockno] = nextFreeFatEntry;
    FAT[nextFreeFatEntry] = ENDOFCHAIN;
    copyFat();

    stream->blockno = nextFreeFatEntry;
```

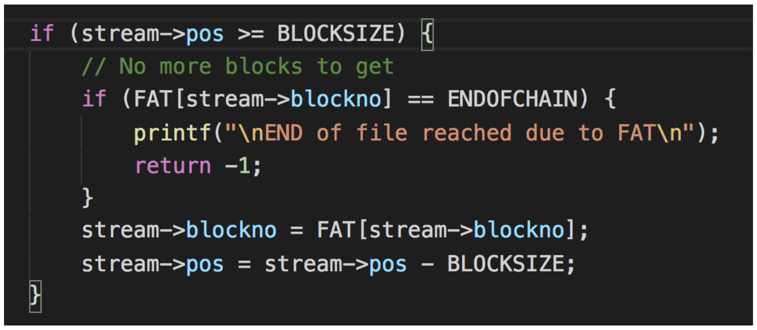
- Flush out buffer
- Increase the filelength of the file with the added ‘dir_pos’ value to find which the position of the file in the current directory
- Write the value to the buffer at the current position
- Increment the pos counter and stop writing to the file



```c
virtualDisk[rootDirIndex].dir.entrylist[stream->dir_pos].filelength++;
stream->buffer.data[stream->pos] = (unsigned char) b;
stream->pos++;
stream->writing = FALSE;
```

myfgetc()

- Initialize the length of the file from its directory entry
- If the position is above the current length, move to the next block with the FAT table if it exists. If yes, adjust the block and position.

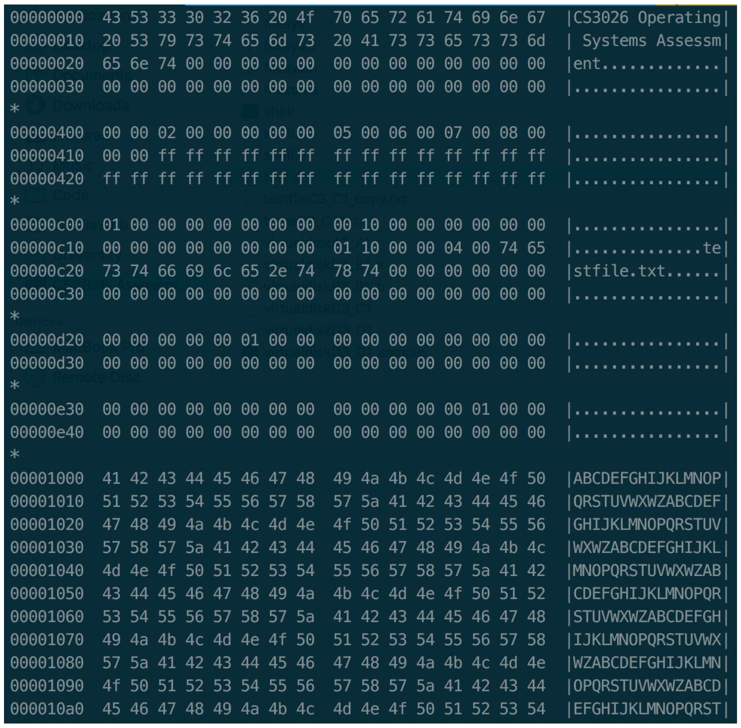


```c
if (stream->pos >= BLOCKSIZE) {
    // No more blocks to get
    if (FAT[stream->blockno] == ENDOFCHAIN) {
        printf("\nEND of file reached due to FAT\n");
        return -1;
    }
    stream->blockno = FAT[stream->blockno];
    stream->pos = stream->pos - BLOCKSIZE;
}
```

- Make sure the current character is being read has not exceeded the file size in the directory entry, not EOF
- Then write the character to the buffer, increment the position and length of the entry in the directory
- Return the character

myfclose()

- Write the current buffer to the virtual disk
- Free the stream pointer



```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 00 00 00 00 00  00 00 00 00 00 00 00 00  |ent.............|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  05 00 06 00 07 00 08 00  |................|
00000410  00 00 ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000420  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 10 00 00 00 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  01 10 00 00 04 00 74 65  |..............te|
00000c20  73 74 66 69 6c 65 2e 74  78 74 00 00 00 00 00 00  |stfile.txt......|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  41 42 43 44 45 46 47 48  49 4a 4b 4c 4d 4e 4f 50  |ABCDEFGHIJKLMNOP|
00001010  51 52 53 54 55 56 57 58  57 5a 41 42 43 44 45 46  |QRSTUVWXWZABCDEF|
00001020  47 48 49 4a 4b 4c 4d 4e  4f 50 51 52 53 54 55 56  |GHIJKLMNOPQRSTUV|
00001030  57 58 57 5a 41 42 43 44  45 46 47 48 49 4a 4b 4c  |WXWZABCDEFGHIJKL|
00001040  4d 4e 4f 50 51 52 53 54  55 56 57 58 57 5a 41 42  |MNOPQRSTUVWXWZAB|
00001050  43 44 45 46 47 48 49 4a  4b 4c 4d 4e 4f 50 51 52  |CDEFGHIJKLMNOPQR|
00001060  53 54 55 56 57 58 57 5a  41 42 43 44 45 46 47 48  |STUVWXWZABCDEFGH|
00001070  49 4a 4b 4c 4d 4e 4f 50  51 52 53 54 55 56 57 58  |IJKLMNOPQRSTUVWX|
00001080  57 5a 41 42 43 44 45 46  47 48 49 4a 4b 4c 4d 4e  |WZABCDEFGHIJKLMN|
00001090  4f 50 51 52 53 54 55 56  57 58 57 5a 41 42 43 44  |OPQRSTUVWXWZABCD|
000010a0  45 46 47 48 49 4a 4b 4c  4d 4e 4f 50 51 52 53 54  |EFGHIJKLMNOPQRST|
```

hexdump -C virtualdiskC3_C1 output

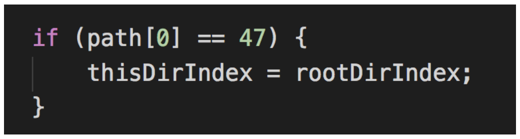
CGS B3-B1

To run this section, run ‘make’ in your console in the root of the ‘CGS_B3_B1’ folder. Then run ‘./shell’ to run the code and show its output. To view the hexdump, run ‘hexdump -C virtualdiskB3_B1_a’ or ‘hexdump -C virtualdiskB3_B1_b’.

Create a new path with mymkdir, get the contents of a subpath, print it out. Change directory with mychdir, create a file in current directory with myfopen and list and print contents of subpath.

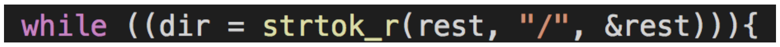
mymkdir()

- If path starts with ‘/’, start at root dir. Otherwise start at relative current directory with currentDirIndex



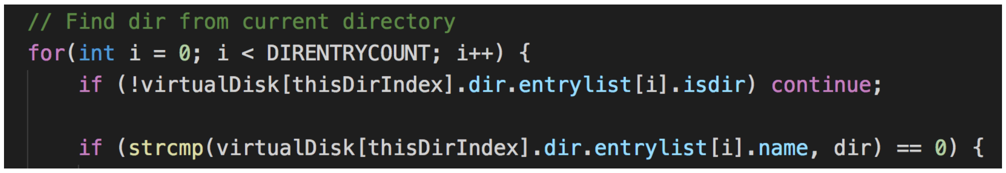
```
if (path[0] == 47) {
    thisDirIndex = rootDirIndex;
}
```

- Loop through directories in path with strtok_r



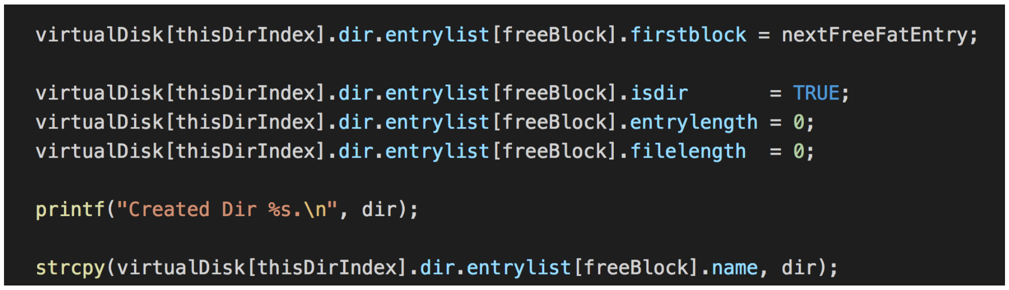
```
while ((dir = strtok_r(rest, "/", &rest))){
```

- Loop through directory entries in current directory and see if their name matches dir



```
// Find dir from current directory
for(int i = 0; i < DIRENTRYCOUNT; i++) {
    if (!virtualDisk[thisDirIndex].dir.entrylist[i].isdir) continue;

    if (strcmp(virtualDisk[thisDirIndex].dir.entrylist[i].name, dir) == 0) {
```

- If yes, change current directory
- If no, find a free block in the virtual disk, set its directory to unused
- Set other variables to it such as block number, name, length

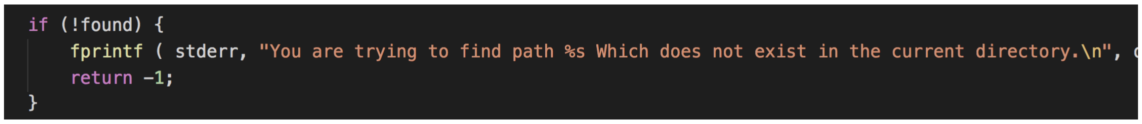


```
virtualDisk[thisDirIndex].dir.entrylist[freeBlock].firstblock = nextFreeFatEntry;

virtualDisk[thisDirIndex].dir.entrylist[freeBlock].isdir       = TRUE;
virtualDisk[thisDirIndex].dir.entrylist[freeBlock].entrylength = 0;
virtualDisk[thisDirIndex].dir.entrylist[freeBlock].filelength  = 0;

printf("Created Dir %s.\n", dir);

strcpy(virtualDisk[thisDirIndex].dir.entrylist[freeBlock].name, dir);
```

- Set FAT to ENDOFCHAIN
- Write FAT and directory to virtual disk
- Create a new blank disk by cycling ‘\0’
- Write new block to disk

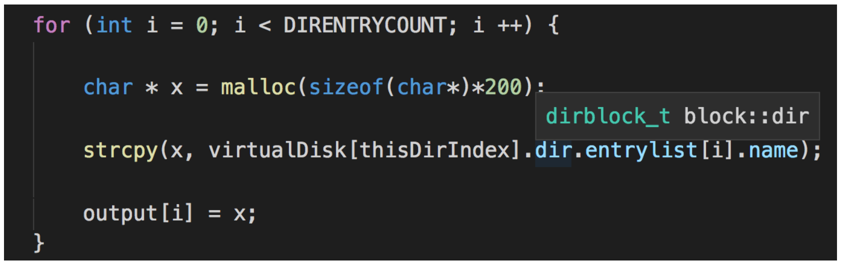
mylistdir()

- Go through same process as in mymkdir, except do not create new directories
- If directory does not exist, return error

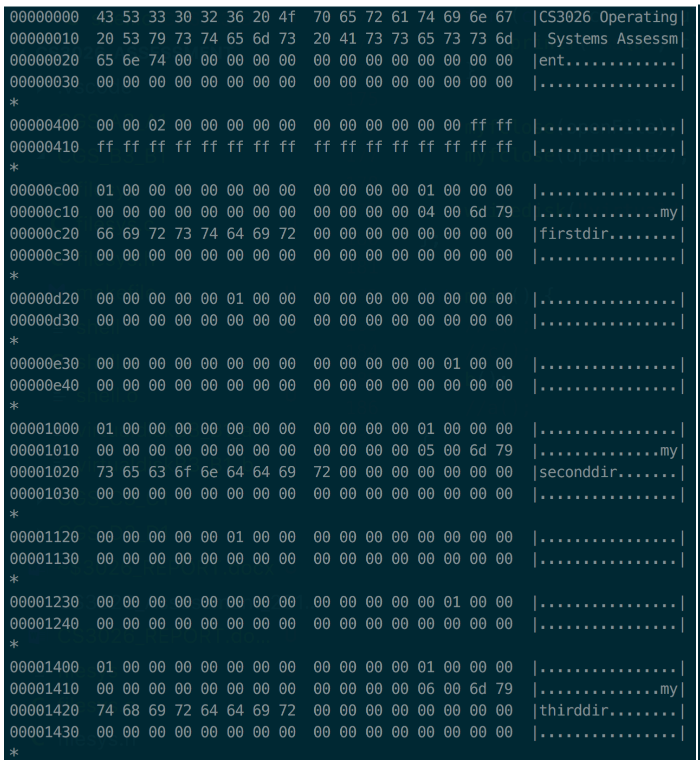


```
if (!found) {
    fprintf ( stderr, "You are trying to find path %s Which does not exist in the current directory.\n",
    return -1;
}
```

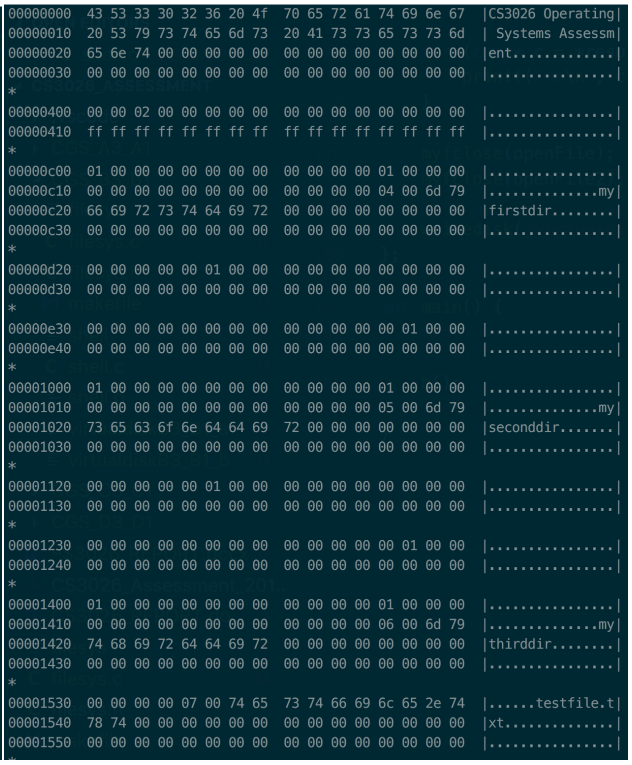
- Once all directories are processed, loop through entries in current directory and add them to the output



```
for (int i = 0; i < DIRENTRYCOUNT; i ++) {

    char * x = malloc(sizeof(char*)*200);

    strcpy(x, virtualDisk[thisDirIndex].dir.entrylist[i].name);

    output[i] = x;
}
```



```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 00 00 00 00 00  00 00 00 00 00 00 00 00  |ent.............|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  00 00 00 00 00 00 ff ff  |................|
00000410  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 04 00 6d 79  |..............my|
00000c20  66 69 72 73 74 64 69 72  00 00 00 00 00 00 00 00  |firstdir........|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001010  00 00 00 00 00 00 00 00  00 00 00 00 05 00 6d 79  |..............my|
00001020  73 65 63 6f 6e 64 64 69  72 00 00 00 00 00 00 00  |seconddir.......|
00001030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001120  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001230  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001410  00 00 00 00 00 00 00 00  00 00 00 00 06 00 6d 79  |..............my|
00001420  74 68 69 72 64 64 69 72  00 00 00 00 00 00 00 00  |thirddir........|
00001430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```



```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 00 00 00 00 00  00 00 00 00 00 00 00 00  |ent.............|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000410  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 04 00 6d 79  |..............my|
00000c20  66 69 72 73 74 64 69 72  00 00 00 00 00 00 00 00  |firstdir........|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001010  00 00 00 00 00 00 00 00  00 00 00 00 05 00 6d 79  |..............my|
00001020  73 65 63 6f 6e 64 64 69  72 00 00 00 00 00 00 00  |seconddir.......|
00001030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001120  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001230  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001410  00 00 00 00 00 00 00 00  00 00 00 00 06 00 6d 79  |..............my|
00001420  74 68 69 72 64 64 69 72  00 00 00 00 00 00 00 00  |thirddir........|
00001430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001530  00 00 00 00 07 00 74 65  73 74 66 69 6c 65 2e 74  |......testfile.t|
00001540  78 74 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |xt..............|
00001550  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

hexdump -C virtualdiskB3_B1_a and virtualdiskB3_B1_b outputs

CGS A5-A1

To run this section, run ‘make’ in your console in the root of the ‘CGS_A3_A1’ folder. Then run ‘./shell’ to run the code and show its output. To view the hexdump, run ‘hexdump -C virtualdiskA3_A1’.

For this section, I implemented all of the required functions and modified myfopen() accordingly. Also, directories can be created relative to the current directory like explained in section B3-B1. However, I did not add the two default directory entries (. and ..) due to time limitations.

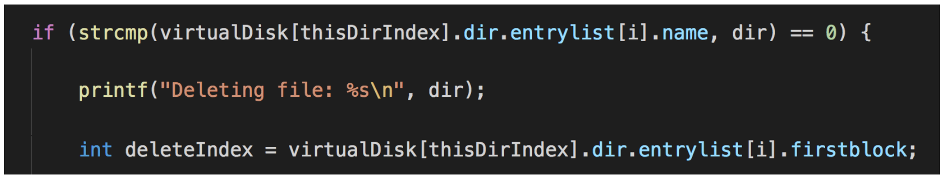
The function a() in shell.c goes through a process outlining each method and what has been implemented. The resulting hexdump is also provided in the end of this section after the process of creation and deletion has completed.

mychdir()

- Go through same process as in mylistdir, except that once all directories have been processed, set the current directory

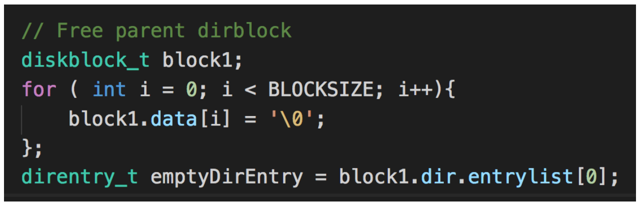
myremove()

- Go through same process of going through the path as in previous method
- If the entry is not a directory and its filename matches the name, the file to be deleted has been found



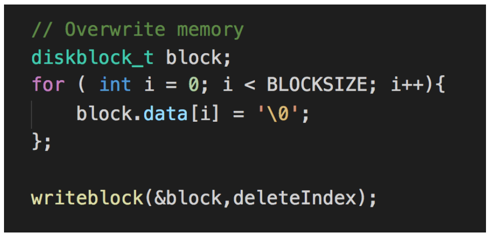
```c
if (strcmp(virtualDisk[thisDirIndex].dir.entrylist[i].name, dir) == 0) {

    printf("Deleting file: %s\n", dir);

    int deleteIndex = virtualDisk[thisDirIndex].dir.entrylist[i].firstblock;
```

- Find the block index of the file
- Set its FAT to UNUSED and save FAT to the virtual disk
- Set the directory entry to unused
- Create a new empty block, set it to the directory entry



```c
// Free parent dirblock
diskblock_t block1;
for ( int i = 0; i < BLOCKSIZE; i++){
    block1.data[i] = '\0';
};
direntry_t emptyDirEntry = block1.dir.entrylist[0];
```

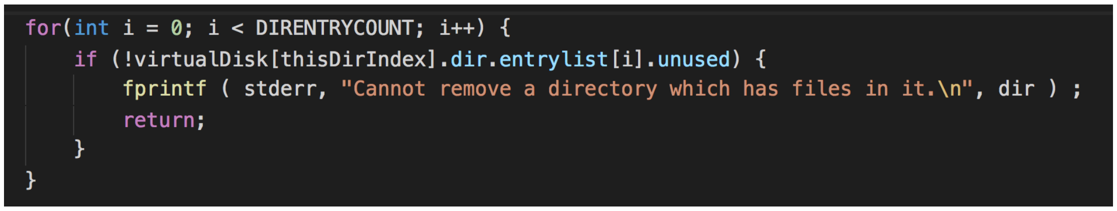
- Create another empty block and set it to the block index and write it to the virtual disk



```
// Overwrite memory
diskblock_t block;
for ( int i = 0; i < BLOCKSIZE; i++){
    block.data[i] = '\0';
};

writeblock(&block,deleteIndex);
```

myrmdir()

- Go through same process of going through the path as in previous methods
- Check that directory is empty



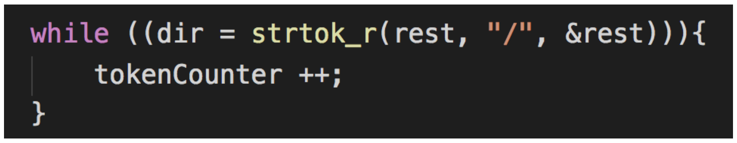
```
for(int i = 0; i < DIRENTRYCOUNT; i++) {
    if (!virtualDisk[thisDirIndex].dir.entrylist[i].unused) {
        fprintf ( stderr, "Cannot remove a directory which has files in it.\n", dir ) ;
        return;
    }
}
```

- Set the FAT block and directory entry to unused and copy both to virtual disk
- Create a new empty block and write it to the disk at the block index if the directory

myfopen()

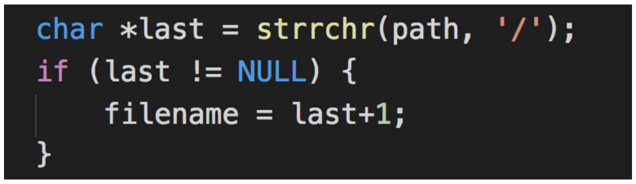
Section before actually writing file:

- Loop through path and find number of tokens

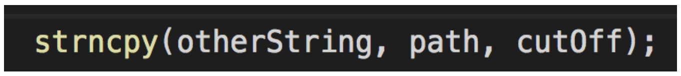


```
while ((dir = strtok_r(rest, "/", &rest))){
    tokenCounter ++;
}
```

- If it is above 1, set the filename to the last token and the path to the filename without the last token

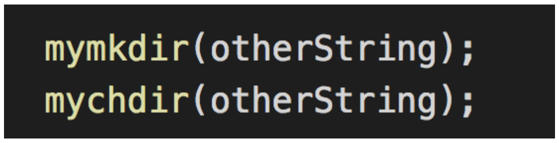


```
char *last = strrchr(path, '/');
if (last != NULL) {
    filename = last+1;
}
```



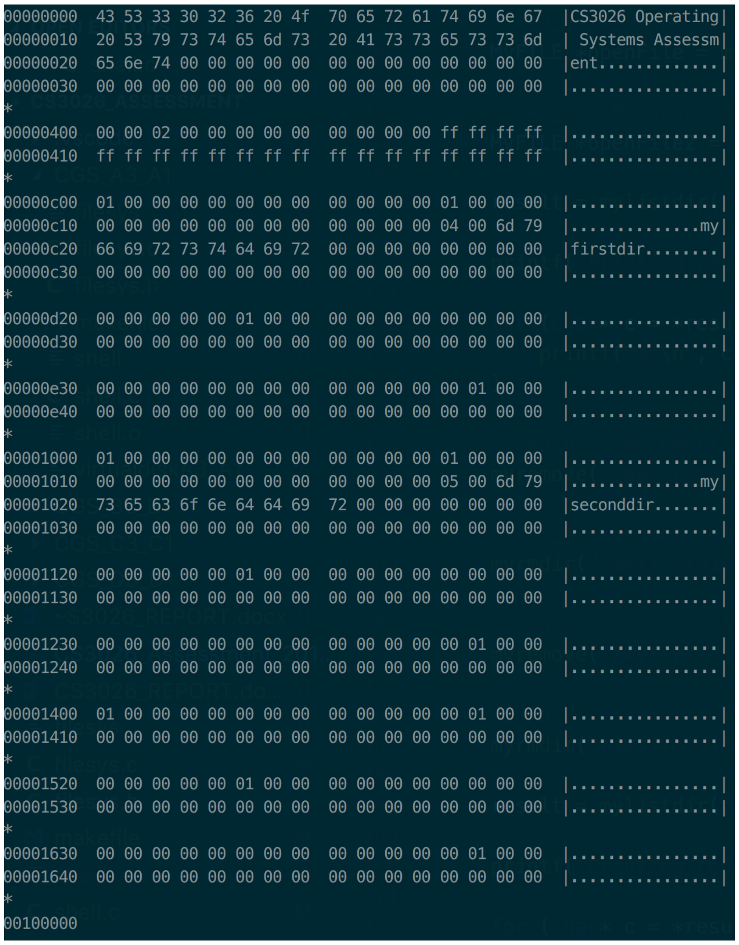
```
strncpy(otherString, path, cutOff);
```

- Make the path with mymkdir()
- Move to the path with mychdir()



```
mymkdir(otherString);
mychdir(otherString);
```

- Continue with the method as shown in section CGS_C3_C1



```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 00 00 00 00 00  00 00 00 00 00 00 00 00  |ent.............|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  00 00 00 00 ff ff ff ff  |................|
00000410  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 04 00 6d 79  |..............my|
00000c20  66 69 72 73 74 64 69 72  00 00 00 00 00 00 00 00  |firstdir........|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001010  00 00 00 00 00 00 00 00  00 00 00 00 05 00 6d 79  |..............my|
00001020  73 65 63 6f 6e 64 64 69  72 00 00 00 00 00 00 00  |seconddir.......|
00001030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001120  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001230  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  01 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001410  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001520  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001530  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001630  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001640  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00100000
```

hexdump -C virtualdiskA3_A1 output after series of operations in shell.c

// EOF