**CS3026 Assessment 2016\_2017**

**Friday 25 November 2016**

**Stefan Rudvin**

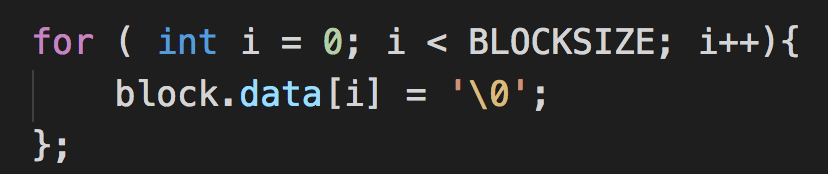
**51549217**

**CGS D3-D1**

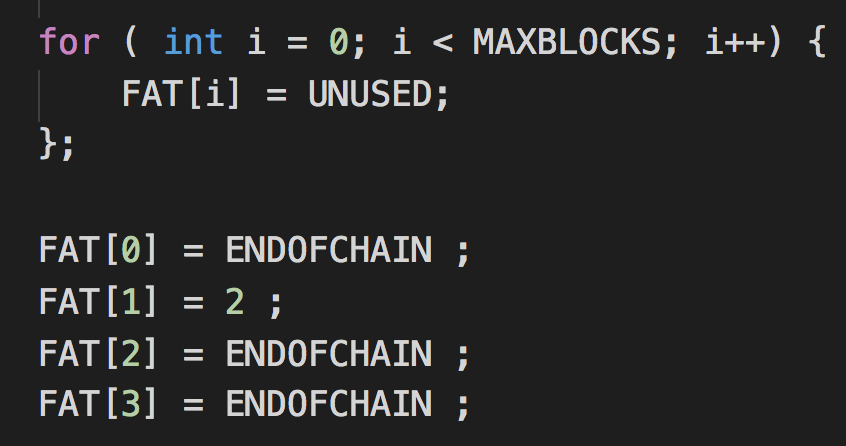
To run this section, run ‘make’ in your console at the root of the CGS\_D3\_D1 folder. Then run ‘./shell’ to run the code and show its output. To view the hexdump, run ‘hexdump -C virtualdiskD3\_D1’.

**Format()**

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



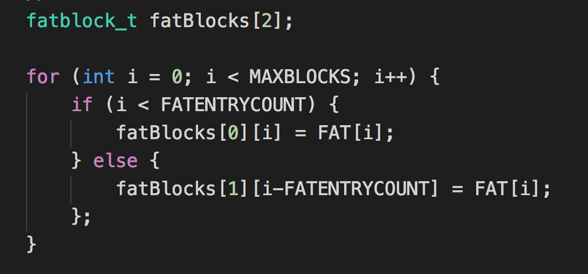
* Initialize an empty block by writing ‘/0’ to its every data position
* Copy string ‘CS3026 Operating Systems Assessment’ to the data section and write the block to index 0 of the virtual disk
* Prepare the FAT by making every element of the array UNUSED, then set the initial values from 0-3 which correspond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

****

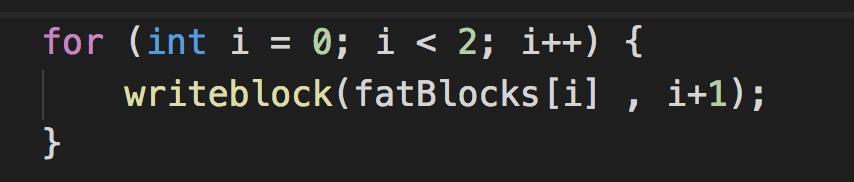
* Call copyFat(), explained below
* Prepare the root directory by writing ‘/0’ to every data position in a block
* As the block is a directory, set ‘isdir’ to 1, and initialize every entrylist entry to unused since the directory is empty.
* Set the rootDirIndex to 3, as we know it will be at this position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

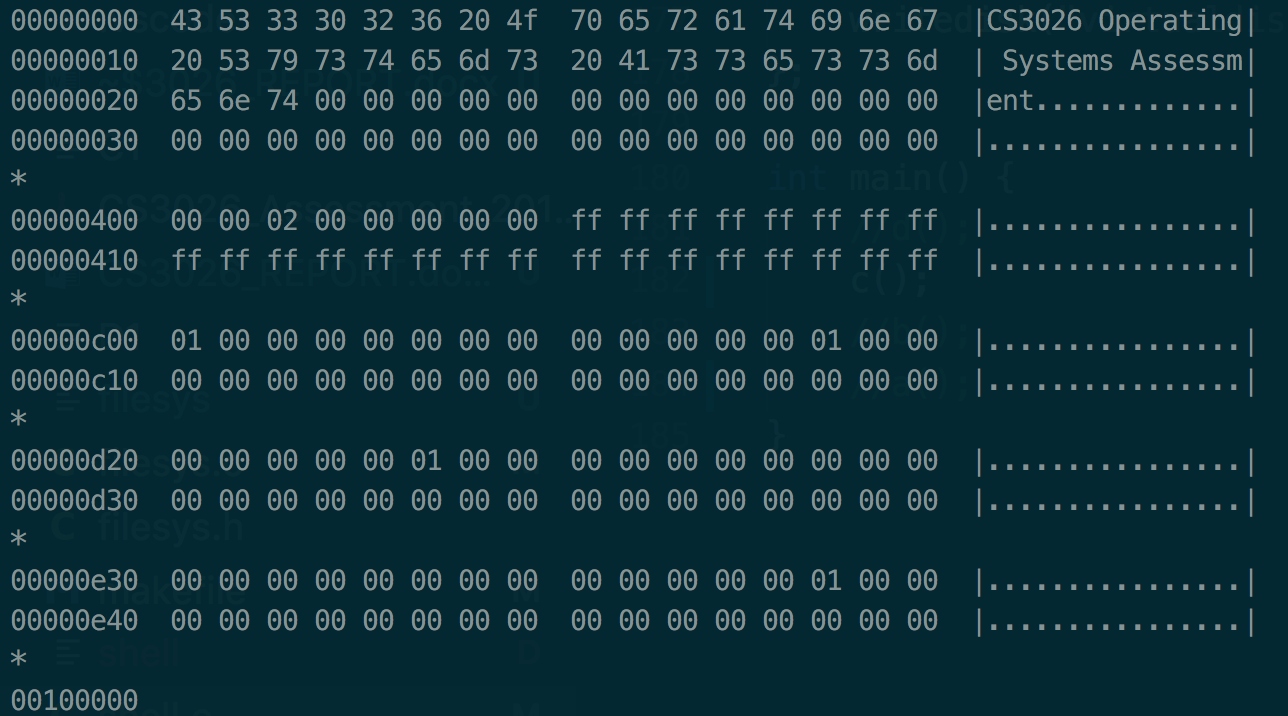
**copyFat()**

* 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



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





*hexdump -C virtualdiskD3\_D1 output*

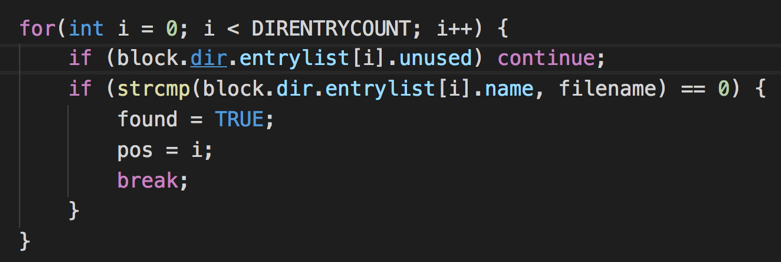
**CGS C3-C1**

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’.

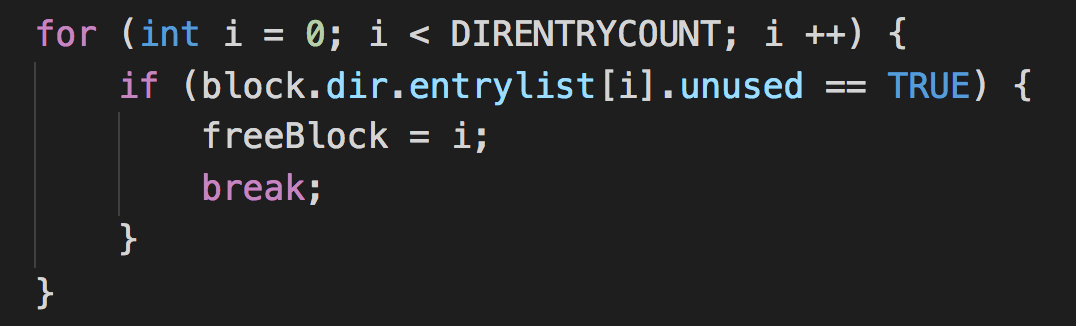
Note: The additions in the beginning of myfopen() of part A3-A1 will be discussed in the corresponding section.

**myfopen()**

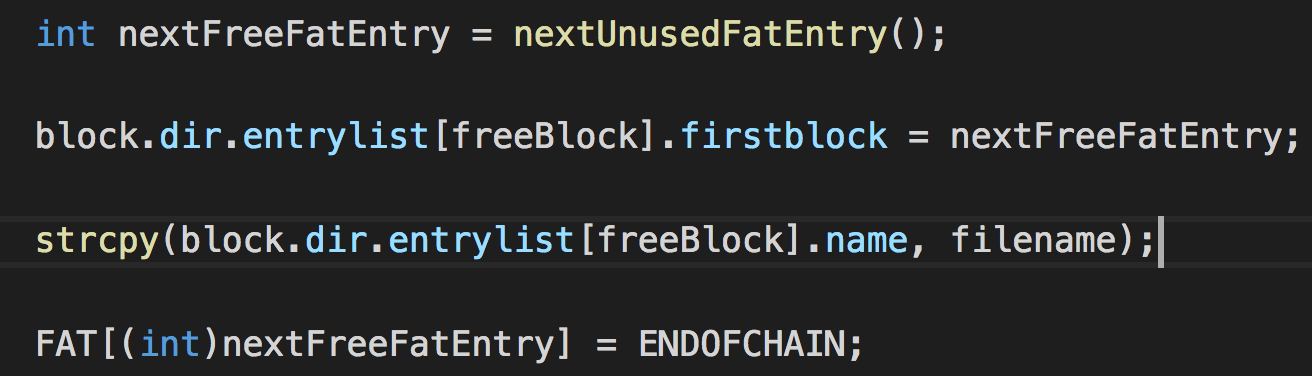
* Initialize MyFILE \* openFile with malloc()
* Set local variables that are used by the function – found, pos, I and freeBlock
* Set block to the current directory block specified by the global variable currentDirIndex
* Loop through the entryLists of the directory block and see if the file already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

****

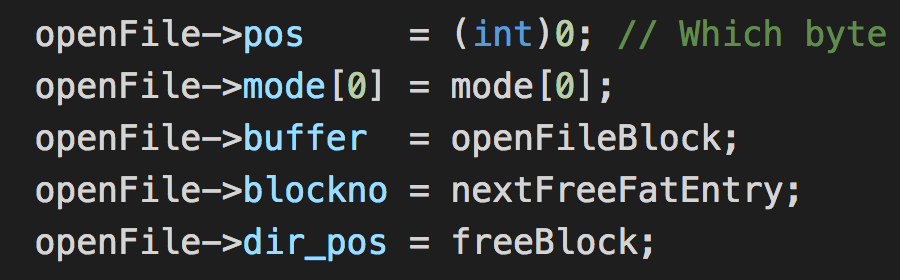
* If found, set the variables of openFile to correspond with the existing file
* If not found, loop through the entrylist of the current directory and find an unused block



* 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

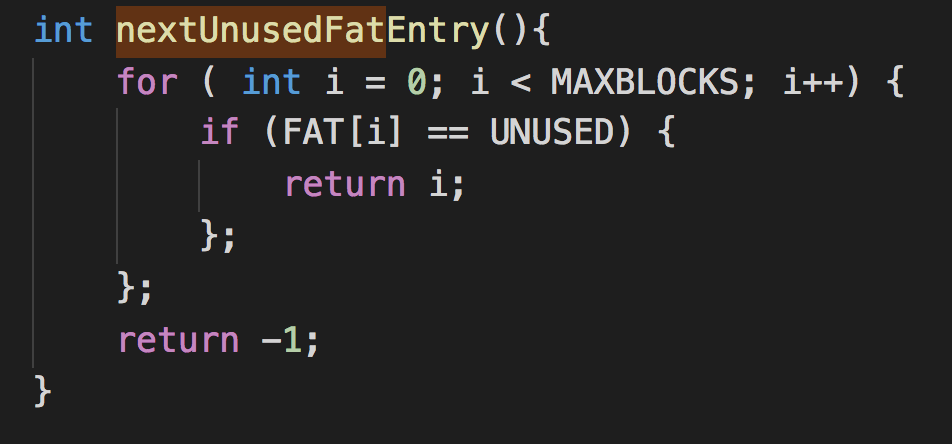


* Save the fat and directory to the virtual disk
* Add all of the gathered information to the openFile
* Return openFile



**nextUnusedFatEntry()**

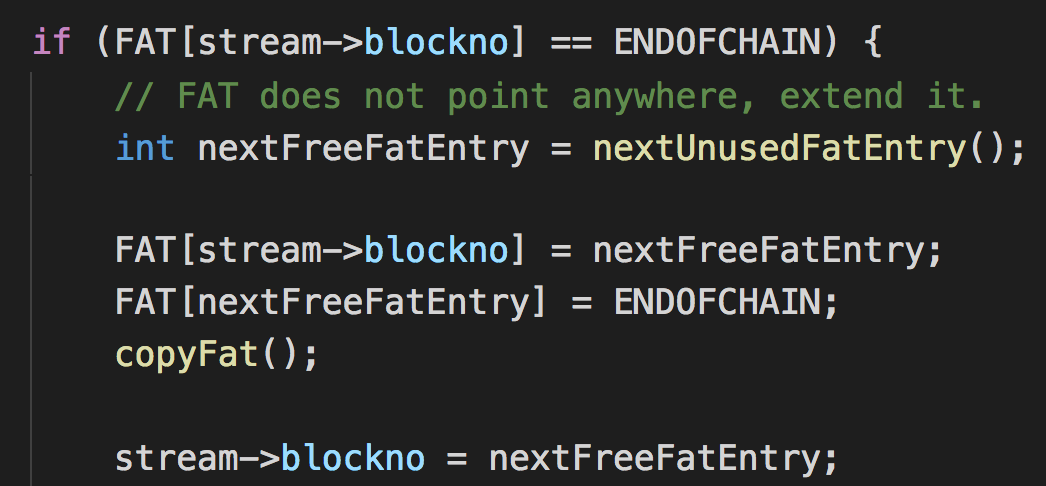
* Loop through all index of the FAT and return an index that is unused



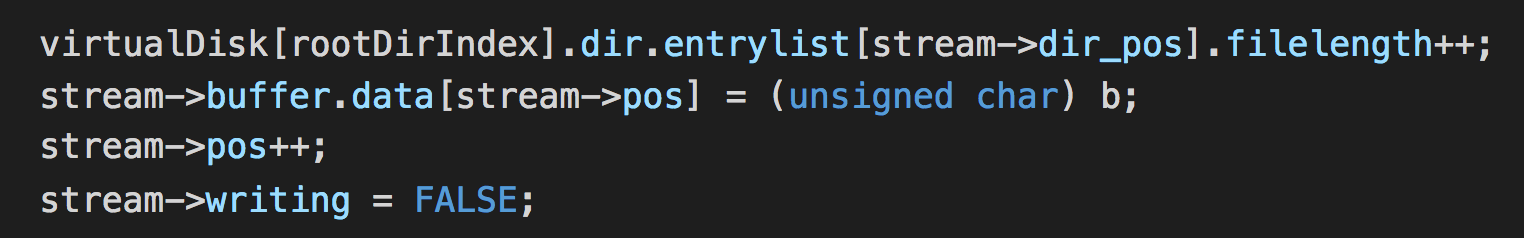
Next, I created a char array of a repeating sequence of the alphabet. Each index was individually sent to myfputc(). Then I closed the file with myfclose, and then opened it again. Then I read the file letter by letter, output the results to the console and wrote them to the file ‘testfileC3\_C1\_copy.txt’.

**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

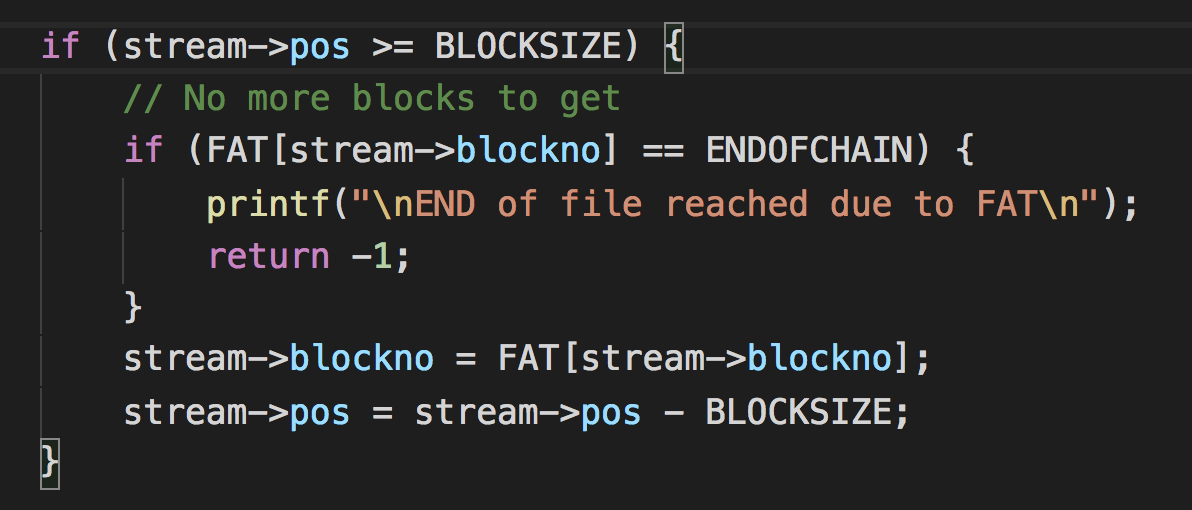


* Flush out buffer
* Increase the filelength of the file with the added ‘dir\_pos’ value to find which the position of the file in the current directory
* Write the value to the buffer at the current position
* Increment the pos counter and stop writing to the file



**myfgetc()**

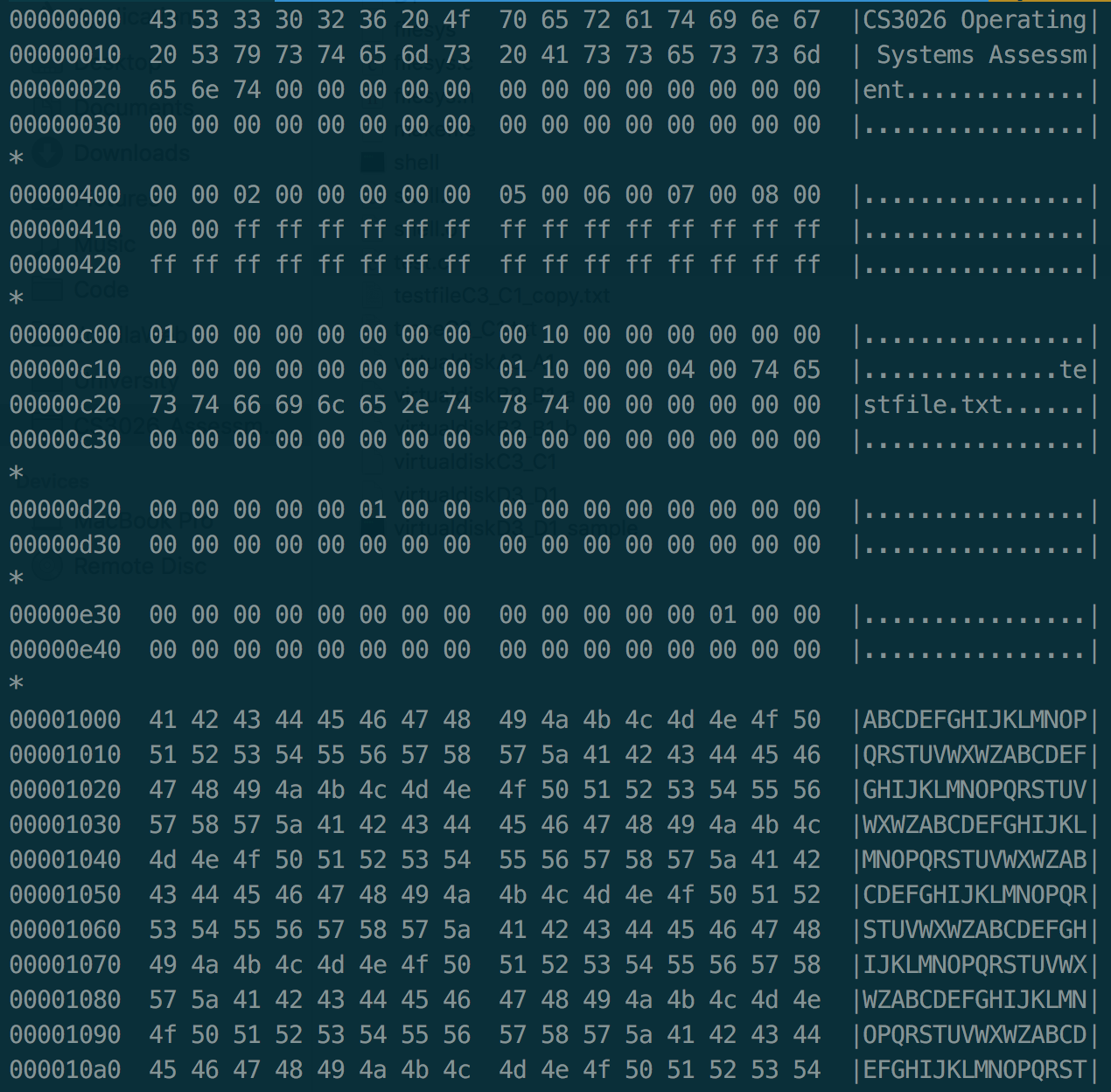
* Initialize the length of the file from its directory entry
* If the position is above the current length, move to the next block with the FAT table if it exists. If yes, adjust the block and position.



* 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



*hexdump -C virtualdiskC3\_C1 output*

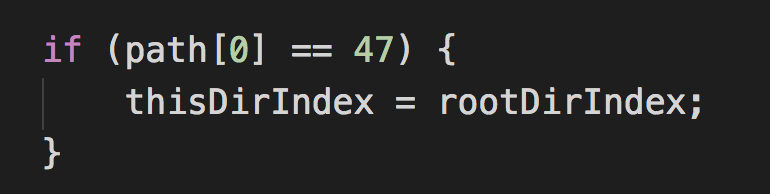
**CGS B3-B1**

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()**

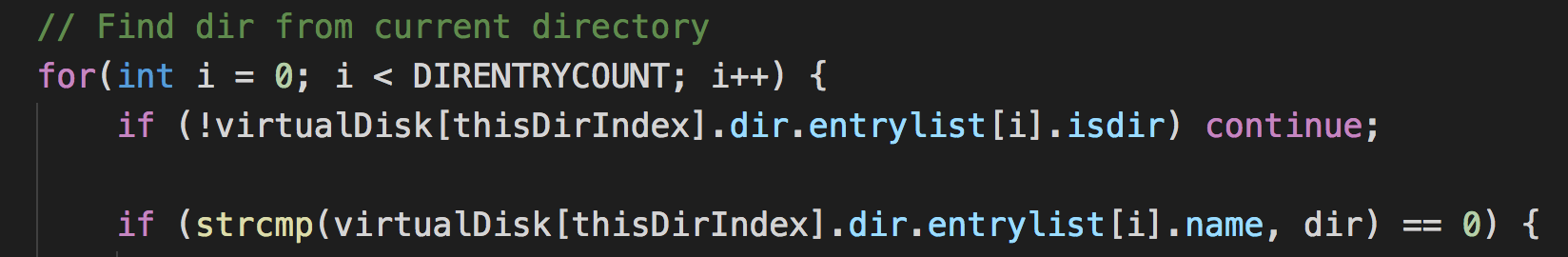
* If path starts with ‘/’, start at root dir. Otherwise start at relative current directory with currentDirIndex

****

* Loop through directories in path with strtok\_r

**../../../../Desktop/Screen%20Shot%202017-11-24%20at%2018.15.45.png**

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

****

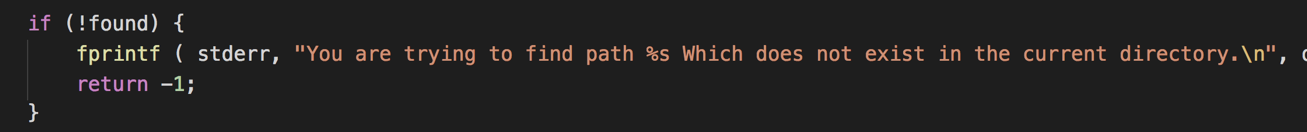
* If yes, change current directory
* If no, find a free block in the virtual disk, set its directory to unused
* Set other variables to it such as block number, name, length

****

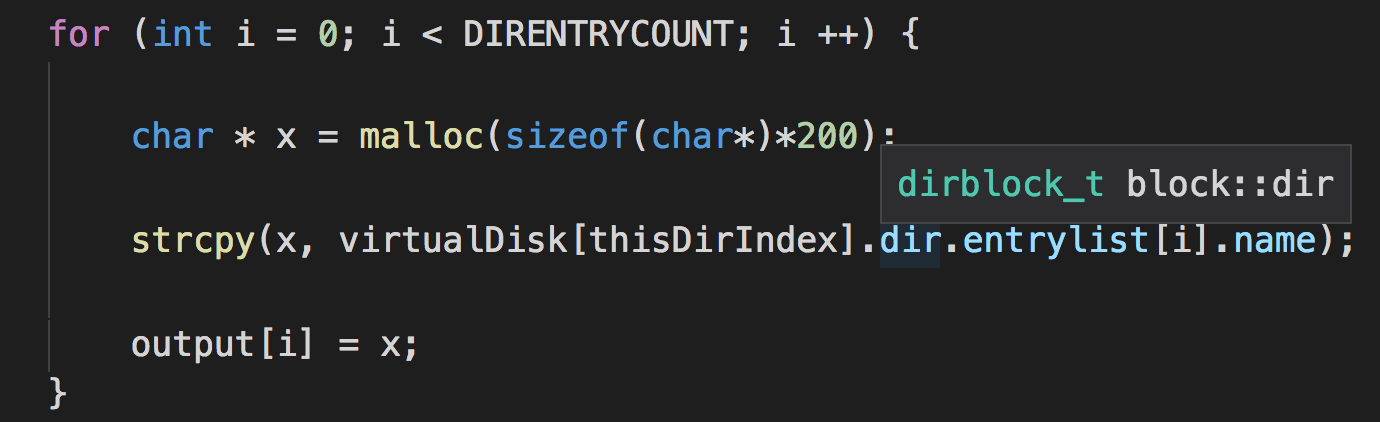
* Set FAT to ENDOFCHAIN
* Write FAT and directory to virtual disk
* Create a new blank disk by cycling ‘\0’
* Write new block to disk

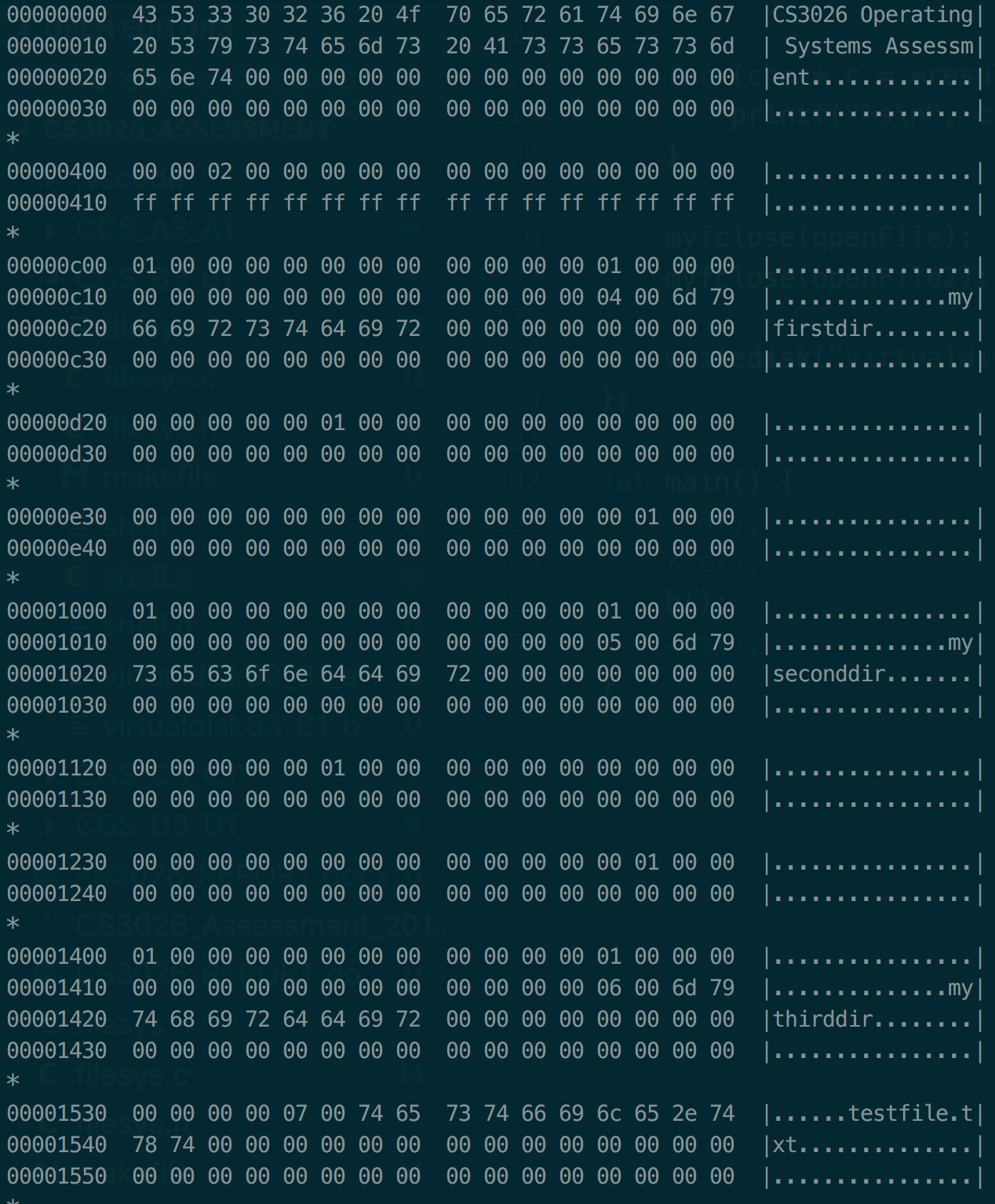
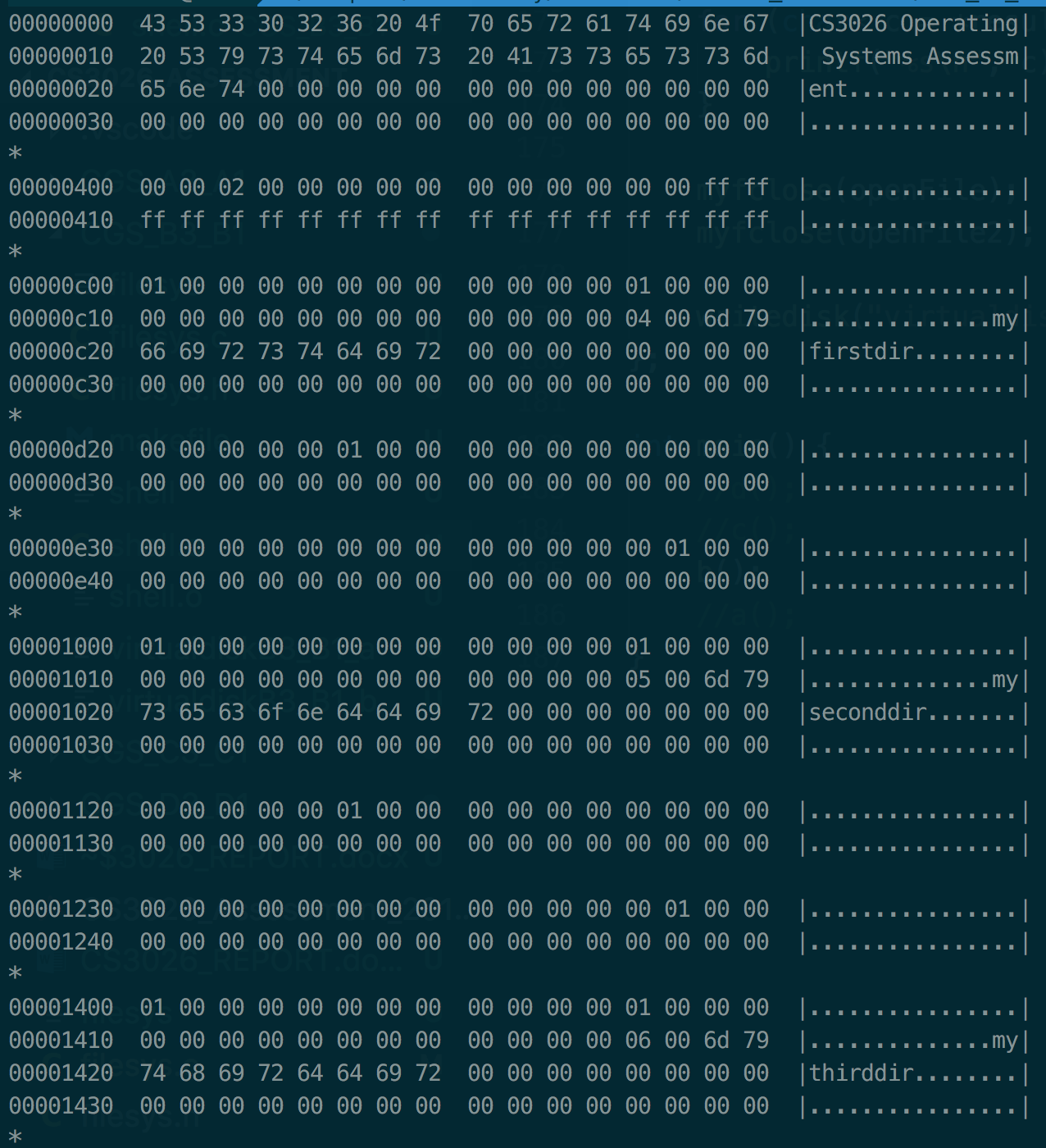
**mylistdir()**

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

****

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

****

****

*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.

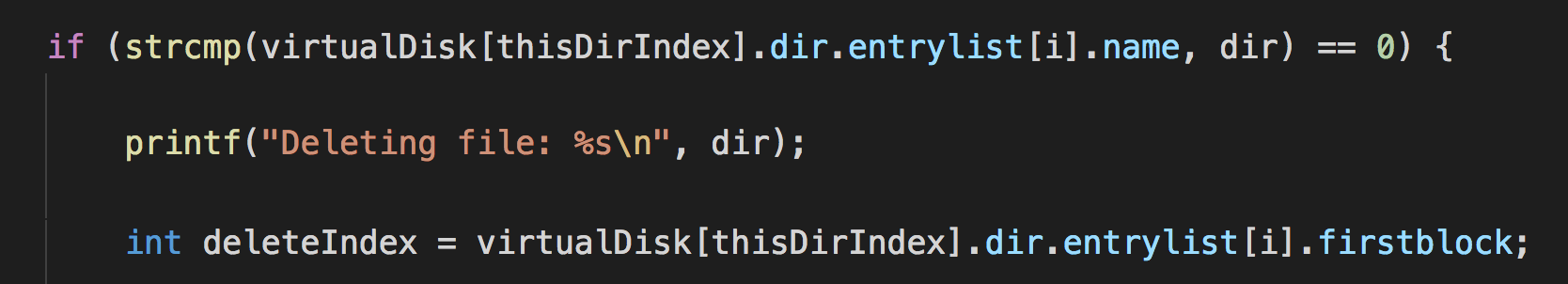
The function a() in shell.c goes through a process outlining each method and what has been implemented. The resulting hexdump is also provided in the end of this section after the process of creation and deletion has completed.

**mychdir()**

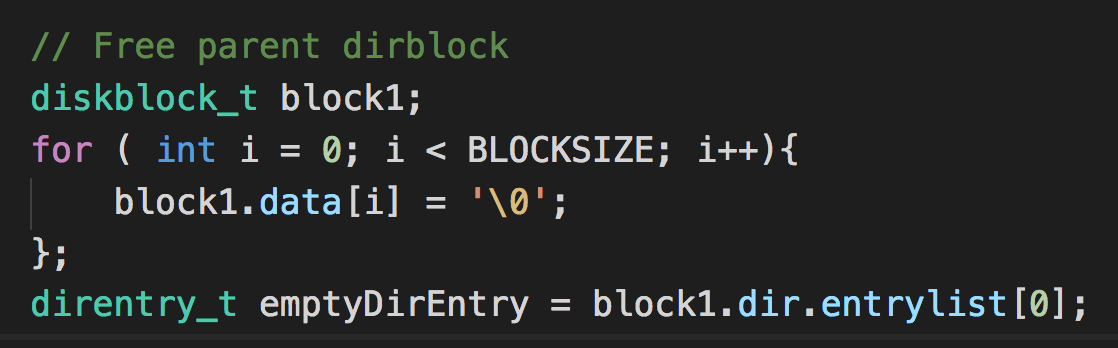
* Go through same process as in mylistdir, except that once all directories have been processed, set the current directory

**myremove()**

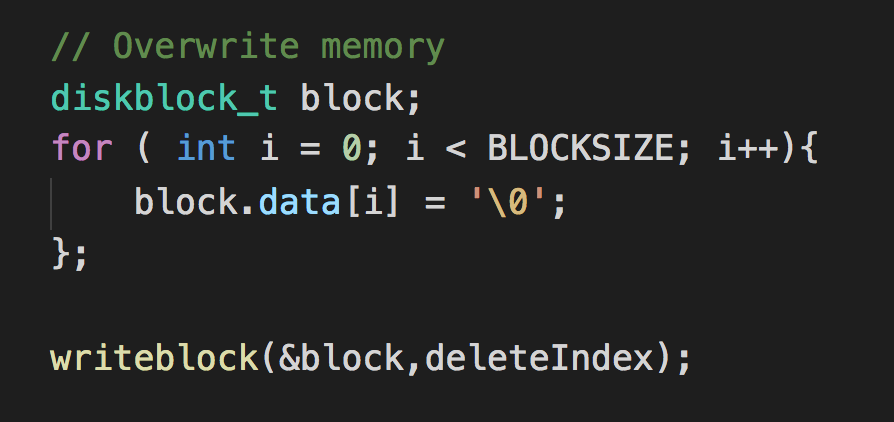
* Go through same process of going through the path as in previous method
* If the entry is not a directory and its filename matches the name, the file to be deleted has been found

****

* 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

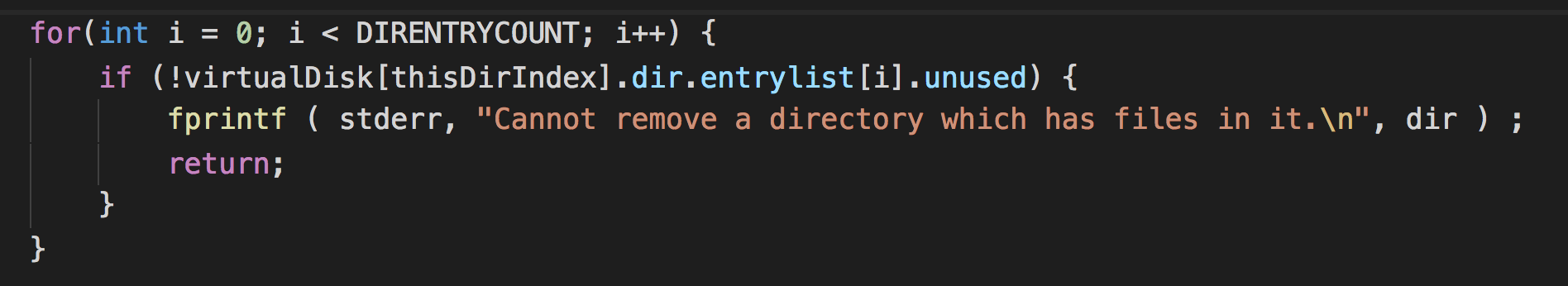
****

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

****

**myrmdir()**

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

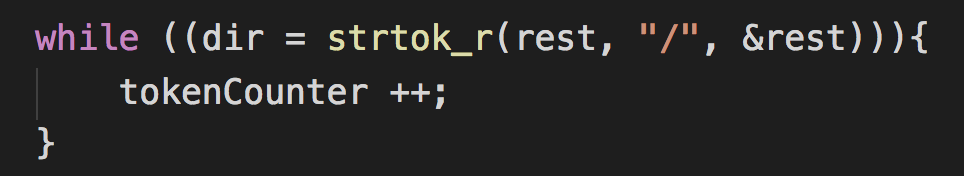
****

* 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

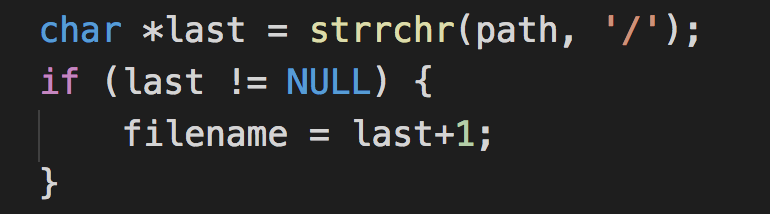
**myfopen()**

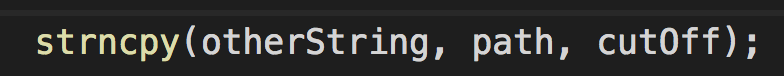
Section before actually writing file:

* Loop through path and find number of tokens

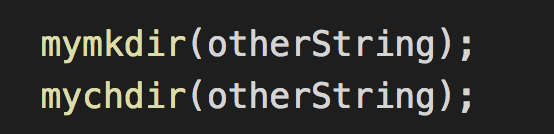
****

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

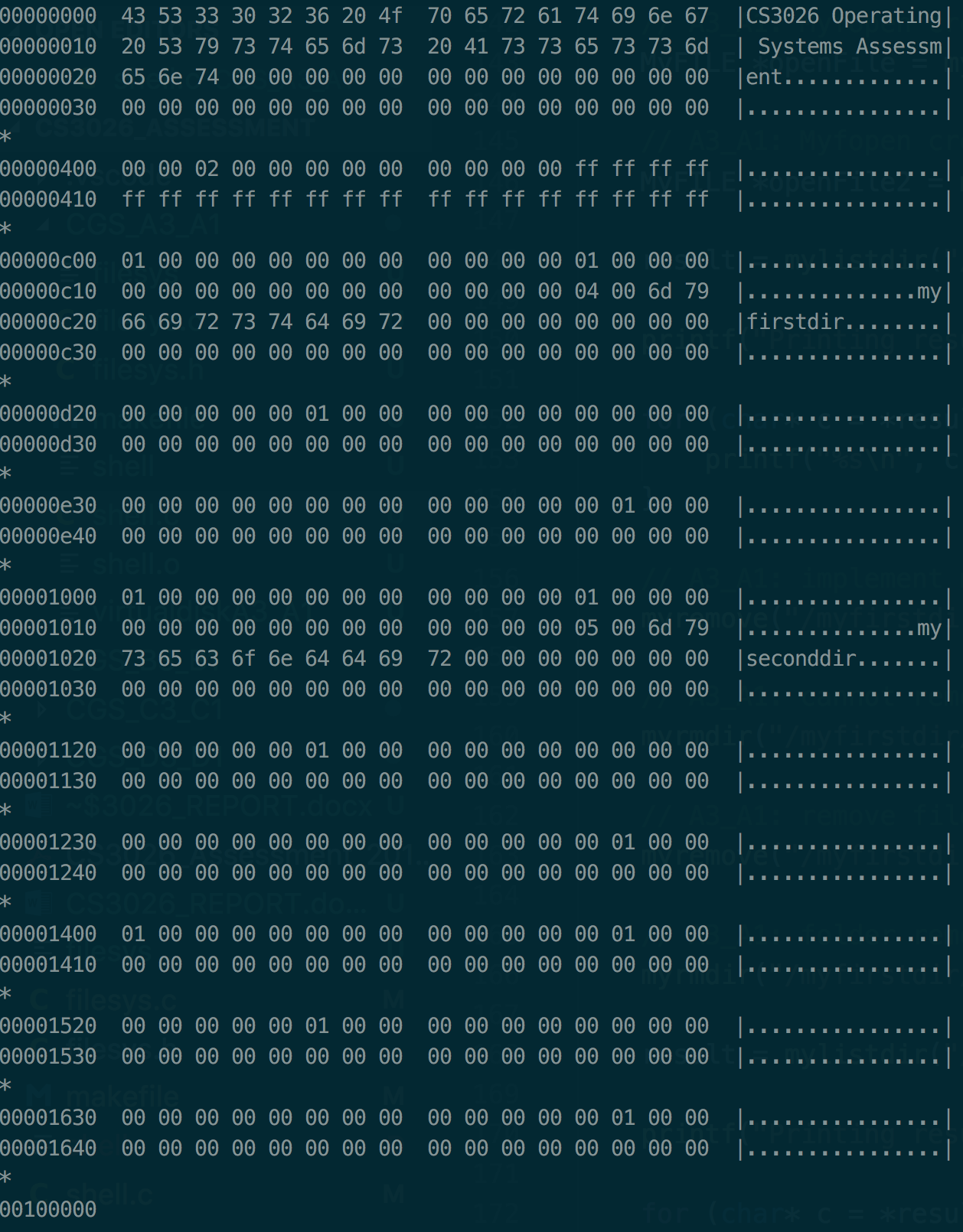


****

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

****

* Continue with the method as shown in section CGS\_C3\_C1

****

*hexdump -C virtualdiskA3\_A1 output after series of operations in shell.c*

**// EOF**