Group Name: META

Members: Alec Nagal, Edward Chen, Ivan Ramos, Karl Layco

Student ID#: 918485625, 920853606, 920520754, 922536937

Primary Github: acraniaaa

Github Link:
https://github.com/CSC415-2023-Fall/csc415-filesystem-acraniaaa/tree/main

File System Description

We implemented a file system that utilizes commands that are similar to the linux commands like *ls, cp, mv, md, rm, touch, cat, cd, and pwd*. It also utilizes commands like *cp2fs,* and *cp2l* to copy to or from the linux file system. Our file system uses bitmap to handle the free space system. Each bit represents the availability of the block: 1 means the block is free to use and 0 means the block is allocated. Our file system also uses contiguous allocation for storing.

Milestone 1

VCB Structure Description

- *Signature*: This is used to identify if the volume control block is valid or if the file system is correctly formatted. Our signature for our VCB is "METAGRP" or *0x005052474154454D* in Hexadecimal.
- *Total Blocks:* This represents the total number of blocks in the file system. It essentially tells you the total capacity of the file system in terms of how many blocks it can hold.
- *Block Size:* This is the size of each block in bytes. This determines how much data can be written to or read from each block.
- *Root Location:* This represents the block location of the root directory in the file system.
- *Free Space Bitmap*: This represents the starting block number where free space bitmap starts.
- *VCB Name:* This holds the name we gave to the VCB we initialized.

Free Space System Description

As mentioned earlier, our file system uses bitmap for free space management. We have decided to use a bitmap array of 64-bit unsigned integers. When initializing, we allocated memory the size of block count, which is 512 in default, and multiplied by block count needed, which is five in default. Afterwards, we set each bit in the bitmap to 1 meaning every bit is free. Then, write the bitmap to disk where the bitmap is located from block 1 to block 5. We made two functions to switch each bit to either used or free: *setBitUsed()* and *setBitFree()*. Another function we included is to check if the bit is used or not: *isBitUsed()*. The function *allocBlocks(n)* allocates blocks contiguously. It looks for a block that is free, when it does find one, it stores the current index and checks if *n* blocks after are free. If the function finds a used block before it reaches the number of blocks requested, it will stop and continue looking from the index. When the function finds *n* free blocks, it will set the bits in the bitmap to *used* then return the starting block. The function *releaseBlocks()* does the opposite, which releases the blocks to the free space system.

Directory System Description

- *Name*: This indicates the name of the directory entry. This is used as an identifier to differentiate between other entries.
- *Location:* An unsigned 64-bit integer struct that contains the starting block of the directory entry and the offset.

- *Size:* An unsigned 64-bit integer that represents the size of the directory entry. This is calculated by multiplying the actual number of directory entries and the size of the directory entry structure.
- *Date Created*: This stores the date and time of directory entry creation.
- *Date Modified:* This stores the date and time when a directory entry is modified.
- *Last Accessed:* This stores the date and time when a user last accessed the directory entry.
- *Is Dir*: This indicates if a directory entry is a directory or file. The values stored: 0 (not a directory) and 1 (a directory).
- *Is Used:* This indicates if a directory is used or free. The values stored: 0 (Used) and 1 (Free).

For initializing the root Directory, we used the idea Professor Bierman showed in class. We made a function that takes in the initial amount of directory entry (initDE) and the parent of the directory entry (parent). First, we calculate the bytesNeeded, which is *initDe * sizeof(DirectoryEntry)*. Then, we calculate the blocksNeeded using the equation: *(m+(n-1))/n*. Then, we update the bytesNeeded by calculating *blocksNeeded * block size*. Then, we calculate the actual amount of the directory entry we can fit by division, *bytesNeeded ÷ sizeof(DirectoryEntry)*. After calculating, we allocate memory for a pointer to a directory entry (dir). We then ask the free space system to allocate blocks using *allocBlocks()*, and store the block number where the allocated blocks start. Using a for-loop, we initialize the directory entries and flag each entry as free, by setting the name to null and *isUsed* to free(1). Then we set up the '.' and '..', in other words, set up the root directory. After setting both up, we write to disk and return the start block.

Milestone 2

- ***fs_mkdir*** - This is a function that handles making a new directory in the directory entry. Our approach to this is similar to the idea Professor Bierman showed in class. This function parses the path name that was passed into it. If the last element does not exist, we make the directory. If it does exist, the function would not create the directory. We also check if the parent directory of the last element from the path name is full or not. If full, the function returns -1, otherwise it will create the directory.
- ***fs_rmdir*** - This function handles deleting the directory. Similar to fs_mkdir, this function utilizes *parsePath()* to parse the pathname. Our approach to this consists of different if-statements to check cases. This function first checks if the directory exists, if not, the function will return -1, otherwise continue. This function checks if the last element is '.' or '..', if so, return -1, otherwise continue. Next, we check if the last element is a directory, if not, return -1, otherwise continue. Lastly, we check if the directory is empty, if not, we return -1, otherwise continue. After all the if-statements, this function will delete the directory. First, it releases the blocks used by the directory back to the free space. Second, it marks the directory entry in the parent as unused. Lastly, we write the change on disk.
- ***fs_opendir*** - This function opens the requested directory. The first thing this function does is parse the pathname passed into the function. Then, this function allocates memory for the *fdDir* structure. Our approach to this is to check if this function is called from the root directory or from a different directory then we populate the *fdDir* structure.

- ***fs_readdir*** - This function reads the requested directory. Our approach to this is similar to the idea Professor Bierman showed in class. This function iterates through the directory and reads the information of the directory entry. If the entry is used, copy the name, and filetype of the entry then set the pointer to the next entry. Afterwards we return the *fs_diriteminfo* structure to the caller.
- ***fs_closedir*** - This function closes the opened directory called by *fs_opendir*. It updates the entries' *lastAccessed* then free the memory allocated by *fdDir* structure.
- ***fs_getcwd*** - This function handles getting the path name of the current working directory. Our approach to this is working backwards until the function gets to the root directory. Using *strcpy* and *strcat*, we append the entry name in front of the buffer. An example is the current working directory is *dir2*. Its parent is *dir1* and *dir1*'s parent is *root*. This function will load up *cwd* and *parent.* The first iteration of the while loop will give "/dir2", then we set *dir2* as the *cwd* and *root* as the *parent*. The second iteration will give "/dir1/dir2". Since we reached *root*, we return "/dir1/dir2" to the caller.
- ***fs_setcwd*** - This function takes a *pathname* as input and attempts to set the current working directory to the specified directory in *pathname*. We first allocate memory to a *PPinfo* structure and, using *parsePath()*, parse the *pathname* and check if the path is valid. If *parthPath()* is successful, then it verifies that the *lastElement* is a directory, and then it sets the current working directory accordingly, using the *loadDir()* function, returning 0. If, for whatever reason, the parsing fails or if the *lastElement* is not a directory, then the function will print an error message and return -1. In both cases, the memory allocated within the function is freed.
- ***fs_isFile*** - This function checks if an entry is a file. Our approach to this is to check the entries' *isDir* variable. Using a helper function, if the function returns 1, this means the entry is a file.
- ***fs_isDir*** - This function checks if an entry is a directory. Our approach to this is similar to *fs_isFile()*, which is to check the entries' *isDir* variable. Using a helper function, if the function returns 1, this means the entry is a directory.
- ***fs_delete*** - This function handles deleting files. Similar to *fs_rmdir*, we used the idea Professor Bierman showed in class. Our approach to this is to do multiple case checkers using if-statements. First, it checks if the file exists, if not, return -1, otherwise continue. Second, it checks if the last element from parsing is a file, if not, return -1, otherwise continue. After checking, we load up the directory entry. Then, we free the blocks used by the file back to the free space. Then, we mark the file unused in the parent. Since we loaded the entry information and not the address, we needed to update the name and *isUsed* status then write the information into the disk.
- ***fs_stat -*** This function handles getting the entry information and stores it in the *fs_stat* structure. Our approach to this is to first check if the path given is valid. If path is valid, copy the directory entry information to caller's *fs_stat*.

Milestone 3

- ***b_open*** - This function's purpose is to initialize and return a file descriptor (b_io_fd). It first checks if the system has been initialized, and will initialize it if not already. Then, it gets a free file descriptor from *b_getFCB()*, and then parses the *filename* to make sure it is valid. Depending on the specified flags, the function performs various tasks, such as

creating a new file, truncating an existing file, loading directory entries, setting the file pointer, and allocating memory for the file control block (FCB). Finally, it will initialize the FCB with all the necessary data and then return a valid file descriptor.

- ***b_read*** - This function reads in data from a file associated with the given file descriptor *fd* with a specified buffer, up to a specified (*count*) amount of bytes. It first checks if the system has been initialized and makes sure the file descriptor is valid. Then, if the file descriptor is, in fact, in use, and it has the appropriate flags, then the function reads the data from the file into the buffer using *LBAread()*. The function handles reading the data in chunks from the FCB and updates the position accordingly, returning the amount of bytes read. If the file does not exist, has invalid flags, or encounters an error, then the function returns -1.
- ***b_write*** - This function writes data from the specified buffer to a file associated with the given file descriptor *fd*. It first checks if the system has been initialized and validates the file descriptor, then it checks if the appropriate flags are present. If they are, then the function utilizes the *LBAwrite()* function to manage writing data in chunks to the file control block. Then, the file position and size are updated, and if necessary (to avoid trampling over other data), relocates the data on disk using *fileRelocate()*. The function returns the total number of bytes written, or -1 if an error of some kind occurs.
- ***b_seek*** - This function changes the current file pointer of the FCB, which is used to track where the current position is within the file itself, it has 3 forms of operation based on its provided *whence* flag: sets the pointer to the beginning of the file, the end of the file, or the current pointer position, with an added offset value, Additionally, should the new file pointer position exits the current block allocation, the new block will be pulled into the buffer and the current block and buffer position will be adjusted accordingly.
- ***b_close*** - This function handles closing a file, given a file descriptor. If the system is not initialized, then the function won't do anything and return 0. In cases where the file buffer is not empty and write operations are pending, it relocates the file entry to prevent data corruption, writes the remaining data from the file buffer to disk using *LBAwrite()*, and updates relevant file information. If the append flag is set, then a similar process is executed. The function then updates the file information block and the file information in the directory using *LBAwrite()*, frees allocated memory for the FCB and PPinfo, and returns 0, successfully closing the file.

Issues:

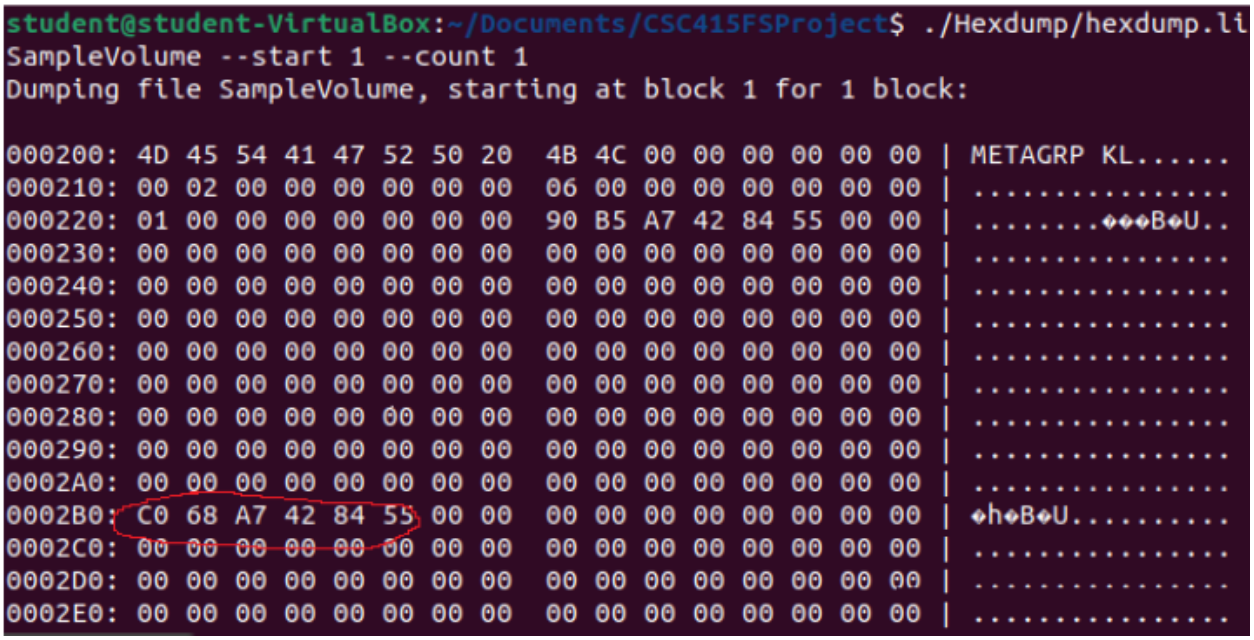
- The first issue we faced was figuring out how the bitmap array would work. Setting each entry in the array to 1 (ex. bitmap[0] = 1) would not work, since that would take up way more space than what would be needed, so Ivan did some research on how to change the value of each bit in an entry. He found that using memset() to set each byte in the bitmap to 0xFF (11111111 in binary) did the job. We double-checked to make sure this worked by utilizing the hexdump, and sure enough, it worked since every byte was set to FF.
- The second issue we got was allocating blocks from free space. When we were analyzing the dump for freespace, it was not showing that it was contiguously allocated. The dump showed that the function jumps from bit/block 31 to bit/block 64, then from 95 to 128. It was skipping 32 bits. To resolve this, in the setBitUsed() and setBitFree() functions, we added ULL to these:

```
bitmap[bitNumber / 64] &= ~(1ULL << (bitNumber % 64));
```

```
bitmap[bitNumber / 64] |= (1ULL << (bitNumber % 64));
```

This suffix indicates that the literal should be considered as an unsigned long long integer, which has a size of 64 bits.

- The third issue we got was an unknown pointer was showing up in the dump for VCB.



We could not figure out what this pointer is for, and it shows when we only use LBAWrite() in the initFileSystem. To fix this issue, we stored the VCB information to a VCB pointer instead of storing it in a VCB variable. This removed unnecessary pointers in the VCB dump.

- One of the issues we had was with *fs_delete*. When we use the command *mv file1 /dir1*, where *file1* is a file and *dir1* is a directory, *fs_delete* does not delete *file1* properly, which breaks *ls* since the function will add more information on the file information block instead of clearing the name. It will add the information of the deleted file back. To fix this, I had to make sure I load the directory entry from the disk instead of loading the address of the directory entry. This way, we make sure that the function will clear the name and it will write unnecessary data.
- Another issue we got was from the command *md.* When we were starting with *fs_mkdir*, when we called *md /dir1/foo* or *md dir1/foo*, where *dir1* exists, we got a *malloc error*. Calling *md /foo* or *md foo* makes the directory fine. To solve this problem, we had to change the size of the allocated memory from one of our helper functions. Instead of allocating size of the entry, we allocated 512 * blocks used, which is a higher product to the first value. An example of this is the entry size is 51 * 120 (Directory entries *

sizeof(DirectoryEntry)) = 6120. Initially, we were allocating 6120 instead of 512 * 12 = 6144. So we changed the malloc size to 6144 to include the rest of the block.

- Another issue we faced was when we use either *cp, cp2fs,* or *cp2l,* since we are doing contiguous allocation, we trample over the other data. To fix this, we decided everytime we update a file, we use *fileRelocate()*, a helper function we made, to relocate the file to accommodate contiguous allocation and prevent trampling over other data.
- We also had a couple of issues with the *mv* command. The first issue was when we tried to move a file in a directory to either another directory, or the same directory (ex: **mv /dir1/file1 /dir2/file1**). When we ran this command, we would get the error stating that the source is a directory. When we viewed **dir2** to see if the file was moved, it was actually in the directory. But, the file was still in **dir1** (our implementation deletes the source after it is copied over). After some testing, we found that the filename was being changed (it went from **/dir1/file1 to /dir**), and we found that this happened in b_open. After a while, we realized that b_open was not copying the filename, so when we parsePath'ed the filename, it changed the string itself, since it was a pointer. We fixed this by making a copy of the filename string first before passing it.
- Another issue we had is with *fs_mkdir*. Every time a user calls *md <relative path>,* the system will run normally and make the directory, but will stop when the *ls* command is called. When a user calls *md <absolute path>,* it works for the first and/or second call but will break after.

```
Prompt > ls

Prompt > md dir1
Directory Successfully Created
Prompt > md /dir1/test1
Directory Successfully Created
Prompt > md /dir1/test2
md Failed: Invalid Path
Prompt > md /dir1/test3
md Failed: Invalid Path
Prompt > ls
make: *** [Makefile:67: run] Segmentation fault (core dumped)
student@student-VirtualBox:~/Documents/FileSystem$
```

The reason we were getting this error is when we use *parsePath()* and create the directory, after the directory is created, we have a line where it assigns the parent directory from *parsePath()* to our global variable for root directory.

```
        ppinfo->parent->location.dirLocation);
rootDir = ppinfo->parent;
printf("Directory Successfully Created\n");
```

So basically, everytime a directory is created, the root directory will also change. Our simple fix to this is we just removed that line.

- Another issue we got is with the *mv* command. When we do *mv /dir1/file /dir1/file* or move the file into the same file, instead of not doing anything, our command deletes the

file instead. We recognized that since the command makes a new copy of the file with the same name, when we delete the source, we also delete the destination. To fix this, before moving the file, we check if the source and destination are the same file from the same directory. If so, do nothing and return, otherwise, do the moving procedure.

Commands:

- ***md *pathname**** **-** Creates a directory if the requested name does not exist.

```
student@student-VirtualBox:~/Documents/FileSystem$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does not exist, errno = 2
File SampleVolume not good to go, errno = 2
Block size is : 512
Created a volume with 9999872 bytes, broken into 19531 blocks of 512 bytes.
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512

File system is not initialized. Formatting...
|---------------------------------|
|------- Command ------|- Status -|
| ls                   |    ON    |
| cd                   |    ON    |
| md                   |    ON    |
| pwd                  |    ON    |
| touch                |    ON    |
| cat                  |    ON    |
| rm                   |    ON    |
| cp                   |    ON    |
| mv                   |    ON    |
| cp2fs                |    ON    |
| cp2l                 |    ON    |
|---------------------------------|
Prompt > md dir1
Directory Successfully Created
Prompt > md dir2
Directory Successfully Created
Prompt > md dir3
Directory Successfully Created
Prompt > ls

dir1
dir2
dir3
Prompt >
```

- ***touch *pathname**** **-** Creates a file if the requested file does not exist. Access the requested file if the requested file exists.

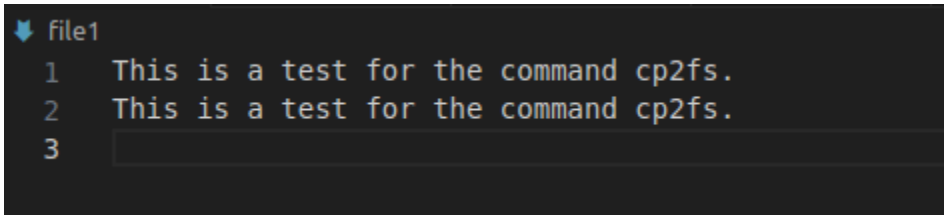
```
Prompt > ls

dir1
dir2
dir3
dir4
Prompt > touch file1
Prompt > ls

dir1
dir2
dir3
dir4
file1
Prompt > touch file2
Prompt > ls

dir1
dir2
dir3
dir4
file1
file2
Prompt > touch file1
Prompt >
```

- ***cp2fs *src* *dest** -** Copy a file from the linux file system to our file system. To test this command, we made three files for testing purposes:
  File1



File2

file2

```
1   This is a test for the command cp2l.
2
```

```
Prompt > ls -l -a

D          6120   .
D          6120   ..
D          6120   dir1
D          6120   dir2
D          6120   dir3
D          6120   dir4
-             0   file1
-             0   file2
Prompt > cp2fs file1 file1
Prompt > ls -l -a

D          6120   .
D          6120   ..
D          6120   dir1
D          6120   dir2
D          6120   dir3
D          6120   dir4
-            76   file1
-             0   file2
Prompt > cp2fs file2 file2
Prompt > ls -l -a

D          6120   .
D          6120   ..
D          6120   dir1
D          6120   dir2
D          6120   dir3
D          6120   dir4
-            76   file1
-            37   file2
Prompt >
```

- ***cat *pathname**** **-** (Limited functionality) Prints out the contents of the file.

```
Prompt > cat file1
This is a test for the command cp2fs.
This is a test for the command cp2fs.
Prompt > cat file2
This is a test for the command cp2l.
Prompt > touch file3
Prompt > cat file3
Prompt >
```

- ***cp *src* *dest* -*** This command copies the content of a file to a different file.

```
Prompt > cat file1
This is a test for the command cp2fs.
This is a test for the command cp2fs.
cPrompt > cat file3
Prompt > cp file1 file3
Prompt > cat file1
This is a test for the command cp2fs.
This is a test for the command cp2fs.
Prompt > cat file3
This is a test for the command cp2fs.
This is a test for the command cp2fs.
Prompt >
```

- ***mv *src* *dest* -*** moves a file to a specified destination. If the destination is a file, the command moves the data in the file to the requested file then the old file is deleted. If the requested file does not exist, the command will make a new file. If the destination is a directory, the command moves the file inside the requested directory. Here are some cases:

- **File to existing file**

```
Prompt > touch file4
Prompt > ls -l

D          6120   dir1
D          6120   dir2
D          6120   dir3
D          6120   dir4
-            37   file2
-            76   file1
-            76   file3
-             0   file4
Prompt > mv file2 file4
Prompt > ls -l

D          6120   dir1
D          6120   dir2
D          6120   dir3
D          6120   dir4
-            76   file1
-            76   file3
-            37   file4
Prompt >
```

- **File to nonexistent file**

```
Prompt > mv file4 file5
Prompt > ls -l

D          6120   dir1
D          6120   dir2
D          6120   dir3
D          6120   dir4
-            37   file5
-            76   file1
-            76   file3
Prompt >
```

- **File to a directory without the file**

```
Prompt > ls dir2

Prompt > mv file1 /dir2
Prompt > ls -l

D          6120   dir1
D          6120   dir2
D          6120   dir3
D          6120   dir4
-            37   file5
-            76   file3
Prompt > cd dir2
Prompt > ls -l

-            76   file1
Prompt > pwd
/dir2
Prompt >
```

- **File inside a directory to root directory**

```
Prompt > mv file1 /
Prompt > pwd
/dir2
Prompt > ls -l

Prompt > cd ..
Prompt > ls -l

D          6120   dir1
D          6120   dir2
D          6120   dir3
D          6120   dir4
-            37   file5
-            76   file1
-            76   file3
Prompt >
```

- **Move a file from one directory to another directory.**

```
Prompt > mv file1 /dir3
Prompt > mv file3 /dir4
Prompt > ls -l

D         6120   dir1
D         6120   dir2
D         6120   dir3
D         6120   dir4
-           37   file5
Prompt > ls dir3

file1
Prompt > ls dir4

file3
Prompt > mv /dir3/file1 /dir4
Prompt > ls dir3

Prompt > ls dir4

file3
file1
Prompt >
```

- ***rm *pathname**** - Removes either a file or a directory (directory must be empty).

```
Prompt > touch file6
Prompt > ls -l

D         6120   dir1
D         6120   dir2
D         6120   dir3
D         6120   dir4
-           37   file5
-            0   file6
Prompt > rm dir2
Prompt > rm dir1
rm Failed: Directory is not empty
Prompt > rm file6
Prompt > ls -l

D         6120   dir1
D         6120   dir3
D         6120   dir4
-           37   file5
Prompt >
```

```
Prompt > ls

dir1
dir3
dir4
file5
Prompt > ls dir1

test1
test2
Prompt > rm /dir1/test1
Prompt > ls dir1

test2
Prompt >
```

- ***ls *pathname**** **-** Lists the files in a given directory. If the pathname is a file, this command will print the file name.

```
Prompt > ls

dir1
dir3
dir4
file5
Prompt > ls -a

.
..
dir1
dir3
dir4
file5
Prompt > ls -l

D         6120   dir1
D         6120   dir3
D         6120   dir4
-           37   file5
Prompt > ls -l -a

D         6120   .
D         6120   ..
D         6120   dir1
D         6120   dir3
D         6120   dir4
-           37   file5
Prompt > ls dir1

test2
Prompt > ls file5
file5
Prompt >
```

- ***cd *pathname**** **-** Changes the current working directory
- ***pwd*** **-** Prints the path name of the current working directory.

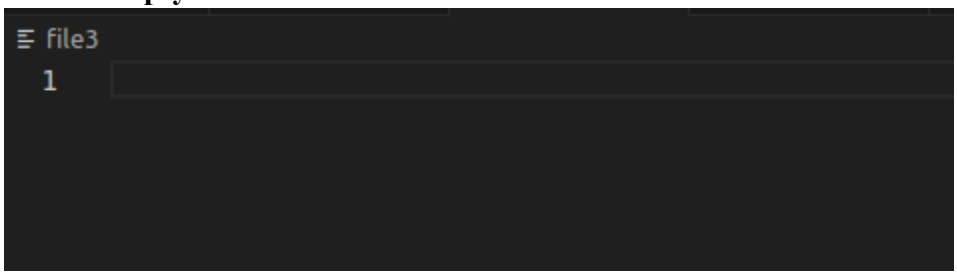
```
Prompt > pwd
/
Prompt > ls

dir1
dir3
dir4
file5
Prompt > cd dir1
Prompt > pwd
/dir1
Prompt > ls

test2
Prompt > cd test2
Prompt > pwd
/dir1/test2
Prompt >
```

- ***cp2l *src* *dest**** **-** Copies a file from our file system to the linux file system.

- Empty File3




```
Prompt > ls -l

D          6120   dir1
D          6120   dir3
D          6120   dir4
-            37   file5
Prompt > cat file5
This is a test for the command cp2l.
Prompt > cp2l file5 file3
Prompt >
```

- Populated File3

file3

```
1   This is a test for the command cp2l.
2
```

Hex Dump from the command outputs

```
student@student-VirtualBox:~/Documents/CSC415FSProject$ ./Hexdump/hexdump.linux SampleVolume --start 1 --count 1
Dumping file SampleVolume, starting at block 1 for 1 block:

000200: 4D 45 54 41 47 52 50 00  4B 4C 00 00 00 00 00 00 | METAGRP.KL......
000210: 00 02 00 00 00 00 00 00  06 00 00 00 00 00 00 00 | ................
000220: 01 00 00 00 00 00 00 00  49 6E 69 74 69 61 6C 20 | ........Initial
000230: 56 43 42 00 00 00 00 00  00 00 00 00 00 00 00 00 | VCB.............
000240: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000250: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000260: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000270: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000280: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000290: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0002A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0002B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0002C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0002D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0002E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0002F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

000300: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000310: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000320: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000330: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000340: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000350: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000360: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000370: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000380: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000390: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0003A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0003B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0003C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0003D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0003E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0003F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

student@student-VirtualBox:~/Documents/CSC415FSProject$
```

Data from the VCB Dump is stored in little Endian Format:

- From 000200-000207: 8 bytes is the Signature we assigned. The value 0x205052474154454D translates to “METAGRP.”
- From 000208-000209: 2 bytes is the unsigned 64-bit integer blocksNeeded. Value: 0x0000000000004C4B converted to decimal is 19531.
- From 000210-000211: 2 byte is the unsigned 64-bit integer blockSize. Value: 0x0000000000000200 converted to decimal is 512.
- From 000218: 1 byte is the unsigned 64-bit integer starting block of the root directory.
- From 000220: 1 byte is the unsigned 64-bit integer starting block of the VCB.
- From 000228-000232: 11 bytes is the character part of the VCB name. This translates to “Initial VCB.“
-

- Free space block

```
student@student-VirtualBox:~/Documents/FileSystem$ ./Hexdump/hexdump.linux SampleVolume --start 2 --count 5
Dumping file SampleVolume, starting at block 2 for 5 blocks:

000400: 00 00 00 C0 FF 03 00 00  FC 3F 00 00 FF FF FF FF | ...◆◆...◆?..◆◆◆◆
000410: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000420: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000430: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000440: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000450: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000460: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000470: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000480: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000490: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

000500: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000510: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000520: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000530: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000540: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000550: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000560: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000570: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000580: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000590: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```

```
000600: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000610: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000620: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000630: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000640: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000650: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000660: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000670: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000680: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000690: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0006A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0006B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0006C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0006D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0006E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0006F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

000700: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000710: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000720: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000730: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000740: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000750: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000760: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000770: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000780: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000790: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0007A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0007B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0007C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0007D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0007E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0007F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```

```
000800: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000810: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000820: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000830: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000840: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000850: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000860: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000870: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000880: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000890: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0008A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0008B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0008C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0008D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0008E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0008F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

000900: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000910: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000920: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000930: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000940: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000950: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000960: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000970: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000980: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000990: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0009A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0009B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0009C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0009D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0009E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0009F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```

```
000A00: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A10: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A20: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A30: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A40: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A50: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A60: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A70: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A80: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A90: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AA0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AB0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AC0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AD0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AE0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AF0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

000B00: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B10: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B20: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B30: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B40: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B50: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B60: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B70: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B80: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B90: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BA0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BB0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BC0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BD0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BE0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BF0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```

```
000C00: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000C10: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000C20: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000C30: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000C40: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000C50: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000C60: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000C70: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000C80: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000C90: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000CA0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000CB0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000CC0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000CD0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000CE0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000CF0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

000D00: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000D10: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000D20: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000D30: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000D40: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000D50: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000D60: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000D70: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000D80: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000D90: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000DA0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000DB0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000DC0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000DD0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000DE0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000DF0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

student@student-VirtualBox:~/Documents/CSC415FSProject$
```

- Populated blocks of root directory

```
student@student-VirtualBox:~/Documents/FileSystem$ ./Hexdump/hexdump.linux SampleVolume --start 7 --count 12
Dumping file SampleVolume, starting at block 7 for 12 blocks:

000E00: 2E 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E40: 06 00 00 00 00 00 00 00  0C 00 00 00 00 00 00 00 | ................
000E50: E8 17 00 00 00 00 00 00  7D 5E 67 65 00 00 00 00 | ........}^ge....
000E60: 7D 5E 67 65 00 00 00 00  7D 5E 67 65 00 00 00 00 | }^ge....}^ge....
000E70: 01 00 00 00 00 00 00 00  2E 2E 00 00 00 00 00 00 | ................
000E80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EB0: 00 00 00 00 00 00 00 00  06 00 00 00 00 00 00 00 | ................
000EC0: 0C 00 00 00 00 00 00 00  E8 17 00 00 00 00 00 00 | ................
000ED0: 7D 5E 67 65 00 00 00 00  7D 5E 67 65 00 00 00 00 | }^ge....}^ge....
000EE0: 7D 5E 67 65 00 00 00 00  01 00 00 00 00 00 00 00 | }^ge............
000EF0: 64 69 72 31 00 00 00 00  00 00 00 00 00 00 00 00 | dir1............

000F00: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F30: 12 00 00 00 00 00 00 00  0C 00 00 00 00 00 00 00 | ................
000F40: E8 17 00 00 00 00 00 00  89 5E 67 65 00 00 00 00 | .........^ge....
000F50: 89 5E 67 65 00 00 00 00  89 5E 67 65 00 00 00 00 | .^ge....^ge....
000F60: 01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FA0: 00 00 00 00 00 00 00 00  1E 00 00 00 00 00 00 00 | ................
000FB0: 0C 00 00 00 00 00 00 00  E8 17 00 00 00 00 00 00 | ................
000FC0: 8B 5E 67 65 00 00 00 00  8B 5E 67 65 00 00 00 00 | .^ge....^ge....
000FD0: 8B 5E 67 65 00 00 00 00  01 00 00 00 01 00 00 00 | .^ge............
000FE0: 64 69 72 33 00 00 00 00  00 00 00 00 00 00 00 00 | dir3............
000FF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

```
001000: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001020: 2A 00 00 00 00 00 00 00  0C 00 00 00 00 00 00 00 | *...............
001030: E8 17 00 00 00 00 00 00  8E 5E 67 65 00 00 00 00 | .........^ge....
001040: 8E 5E 67 65 00 00 00 00  8E 5E 67 65 00 00 00 00 | .^ge....^ge....
001050: 01 00 00 00 00 00 00 00  64 69 72 34 00 00 00 00 | ........dir4....
001060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001090: 00 00 00 00 00 00 00 00  36 00 00 00 00 00 00 00 | ........6.......
0010A0: 0C 00 00 00 00 00 00 00  E8 17 00 00 00 00 00 00 | ................
0010B0: 90 5E 67 65 00 00 00 00  90 5E 67 65 00 00 00 00 | .^ge....^ge....
0010C0: 90 5E 67 65 00 00 00 00  01 00 00 00 00 00 00 00 | .^ge............
0010D0: 66 69 6C 65 35 00 00 00  00 00 00 00 00 00 00 00 | file5...........
0010E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0010F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

001100: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001110: 5C 00 00 00 00 00 00 00  02 00 00 00 00 00 00 00 | \...............
001120: 25 00 00 00 00 00 00 00  AB 61 67 65 00 00 00 00 | %.......age....
001130: AB 61 67 65 00 00 00 00  7F 65 67 65 00 00 00 00 | .age....ege....
001140: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001150: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001160: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001170: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001180: 00 00 00 00 00 00 00 00  60 00 00 00 00 00 00 00 | ........`.......
001190: 01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0011A0: A5 63 67 65 00 00 00 00  A5 63 67 65 00 00 00 00 | .cge....cge....
0011B0: A5 63 67 65 00 00 00 00  00 00 00 00 01 00 00 00 | .cge............
0011C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0011D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0011E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0011F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

```
001200: 5E 00 00 00 00 00 00 00  02 00 00 00 00 00 00 00 | ^...............
001210: 4C 00 00 00 00 00 00 00  EC 62 67 65 00 00 00 00 | L........bge....
001220: F1 62 67 65 00 00 00 00  0F 63 67 65 00 00 00 00 | bge.....cge....
001230: 00 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00 | ................
001240: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001250: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001260: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001270: 00 00 00 00 00 00 00 00  60 00 00 00 00 00 00 00 | ........`.......
001280: 02 00 00 00 00 00 00 00  25 00 00 00 00 00 00 00 | ........%.......
001290: 5E 61 67 65 00 00 00 00  68 61 67 65 00 00 00 00 | ^age....hage....
0012A0: AB 61 67 65 00 00 00 00  00 00 00 00 01 00 00 00 | age.............
0012B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

001300: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001310: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001320: 00 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00 | ................
001330: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001340: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001350: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001360: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001370: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001380: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001390: 00 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00 | ................
0013A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

- File1 blocks

```
student@student-VirtualBox:~/Documents/FileSystem$ ./Hexdump/hexdump.linux SampleVolume --start 91 --count 10
Dumping file SampleVolume, starting at block 91 for 10 blocks:

00B600: 66 69 6C 65 31 00 00 00  00 00 00 00 00 00 00 00 | file1...........
00B610: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B620: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B630: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B640: 5A 00 00 00 00 00 00 00  02 00 00 00 00 00 00 00 | Z...............
00B650: 4C 00 00 00 00 00 00 00  3D 63 67 65 00 00 00 00 | L.......=cge....
00B660: 3D 63 67 65 00 00 00 00  3D 63 67 65 00 00 00 00 | =cge....=cge....
00B670: 00 00 00 00 00 00 00 00  E0 C7 15 03 02 56 00 00 | ........◆◆...V..
00B680: 00 00 00 00 41 02 00 00  00 00 00 00 00 00 00 00 | ....A...........
00B690: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B6A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B6B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B6C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B6D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B6E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B6F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

00B700: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B710: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B720: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B730: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B740: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B750: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B760: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B770: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B780: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B790: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B7A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B7B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B7C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B7D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B7E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00B7F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

```
00B800: 54 68 69 73 20 69 73 20  61 20 74 65 73 74 20 66 | This is a test f
00B810: 6F 72 20 74 68 65 20 63  6F 6D 6D 61 6E 64 20 63 | or the command c
00B820: 70 32 66 73 2E 0A 54 68  69 73 20 69 73 20 61 20 | p2fs..This is a
00B830: 74 65 73 74 20 66 6F 72  20 74 68 65 20 63 6F 6D | test for the com
00B840: 6D 61 6E 64 20 63 70 32  66 73 2E 0A FF FF FF FF | mand cp2fs..····
00B850: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B860: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B870: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B880: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B890: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B8A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B8B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B8C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B8D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B8E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B8F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················

00B900: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B910: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B920: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B930: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B940: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B950: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B960: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B970: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B980: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B990: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B9A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B9B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B9C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B9D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B9E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
00B9F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ················
```

- File3 blocks

```
00BE00: 66 69 6C 65 33 00 00 00  00 00 00 00 00 00 00 00 | file3...........
00BE10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BE20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BE30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BE40: 5E 00 00 00 00 00 00 00  02 00 00 00 00 00 00 00 | ^...............
00BE50: 4C 00 00 00 00 00 00 00  17 63 67 65 00 00 00 00 | L........cge....
00BE60: 17 63 67 65 00 00 00 00  17 63 67 65 00 00 00 00 | .cge.....cge....
00BE70: 00 00 00 00 00 00 00 00  B0 46 12 03 02 56 00 00 | ........°F...V..
00BE80: 00 00 00 00 41 02 00 00  00 00 00 00 00 00 00 00 | ....A...........
00BE90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BEA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BEB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BEC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BED0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BEE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BEF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

00BF00: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BF10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BF20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BF30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BF40: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BF50: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BF60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BF70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BF80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BF90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BFA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BFB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BFC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BFD0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BFE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BFF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

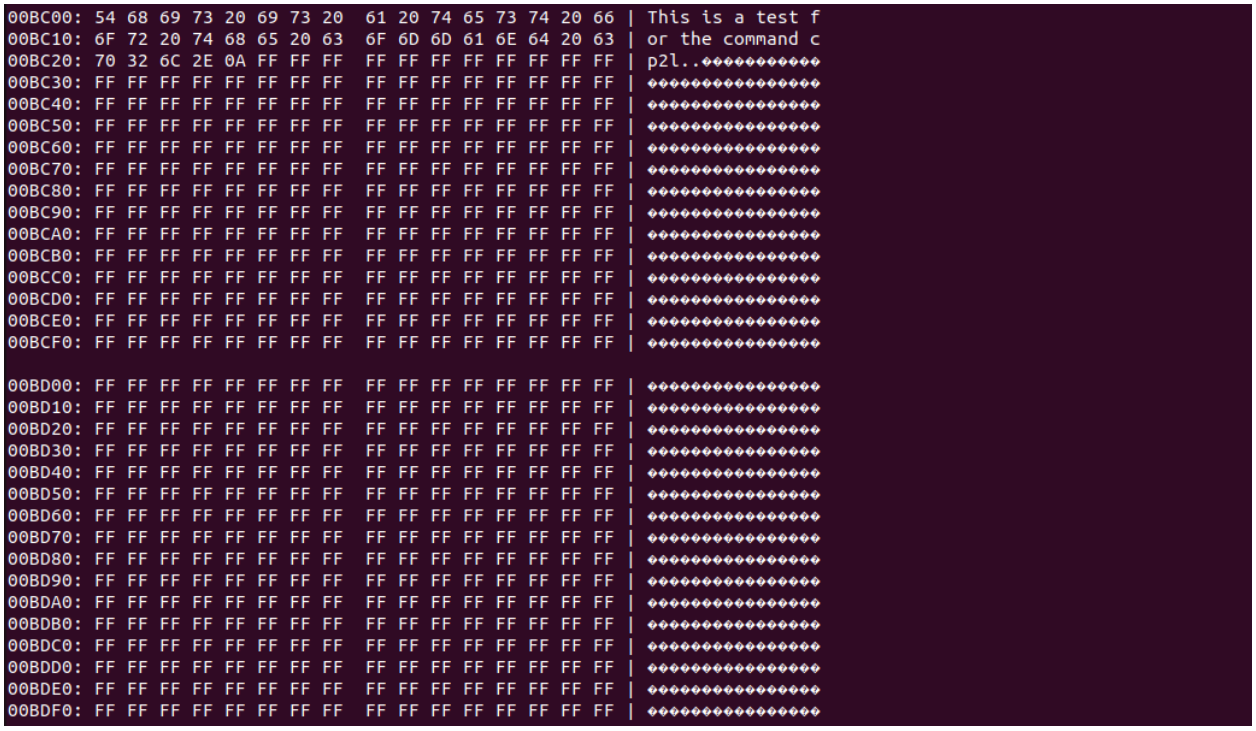
```
00C000: 54 68 69 73 20 69 73 20  61 20 74 65 73 74 20 66 | This is a test f
00C010: 6F 72 20 74 68 65 20 63  6F 6D 6D 61 6E 64 20 63 | or the command c
00C020: 70 32 66 73 2E 0A 54 68  69 73 20 69 73 20 61 20 | p2fs..This is a
00C030: 74 65 73 74 20 66 6F 72  20 74 68 65 20 63 6F 6D | test for the com
00C040: 6D 61 6E 64 20 63 70 32  66 73 2E 0A 00 00 00 00 | mand cp2fs......
00C050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C0A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C0B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C0C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C0D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C0E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C0F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

00C100: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C110: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C120: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C130: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C140: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C150: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C160: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C170: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C180: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C190: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C1A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C1B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C1C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C1D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C1E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00C1F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

- File5 blocks

```
00BA00: 66 69 6C 65 35 00 00 00  00 00 00 00 00 00 00 00 | file5...........
00BA10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BA20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BA30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BA40: 5C 00 00 00 00 00 00 00  02 00 00 00 00 00 00 00 | \...............
00BA50: 25 00 00 00 00 00 00 00  AB 61 67 65 00 00 00 00 | %........age....
00BA60: AB 61 67 65 00 00 00 00  7F 65 67 65 00 00 00 00 | age....ege....
00BA70: 00 00 00 00 00 00 00 00  B0 16 DC 3E D7 55 00 00 | ........>U..
00BA80: 25 00 00 00 00 00 00 00  00 00 00 00 25 00 00 00 | %...........%...
00BA90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BAA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BAB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BAC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BAD0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BAE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BAF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

00BB00: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BB10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BB20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BB30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BB40: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BB50: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BB60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BB70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BB80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BB90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BBA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BBB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BBC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BBD0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BBE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
00BBF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```



- An example of what a deleted file looks like. Blocks with deleted entry will be overwritten by the free space system.

```
00C400: 54 68 69 73 20 69 73 20  61 20 74 65 73 74 20 66 | This is a test f
00C410: 6F 72 20 74 68 65 20 63  6F 6D 6D 61 6E 64 20 63 | or the command c
00C420: 70 32 66 73 2E 0A 54 68  69 73 20 69 73 20 61 20 | p2fs..This is a
00C430: 74 65 73 74 20 66 6F 72  20 74 68 65 20 63 6F 6D | test for the com
00C440: 6D 61 6E 64 20 63 70 32  66 73 2E 0A FF FF FF FF | mand cp2fs..◆◆◆◆
00C450: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C460: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C470: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C480: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C490: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C4A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C4B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C4C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C4D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C4E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C4F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

00C500: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C510: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C520: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C530: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C540: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C550: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C560: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C570: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C580: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C590: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C5A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C5B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C5C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C5D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C5E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00C5F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```

Work Table

Name	**Assigned Work**
Karl Layco	Initializing Root Directory, Allocating Blocks from Free Space (implement allockBlocks()), implement fs_delete and fs_rmdir(), b_read(), b_write(), and b_open(), manage helper functions, debugging.
Ivan Ramos	Initializing Free Space management and functions to set bits to free/allocated, implement fs_setcwd(), fs_isFile(), fs_mkdir(), mv command, debugging
Alec Nagal	Initialize the volume control block, analyzing the dump of VCB, Free Space, and root directory, implement fs_opendir(), fs_readdir(), and fs_closedir().
Edward Chen	Allocating Blocks from Free Space (implement allockBlocks()), implement fs_stat(), fs_isDir(), and fs_getcwd, implement b_seek() and b_open().