San Francisco State University

CSC 415-01 Operating System Principles - Professor Robert Bierman

# File System WriteUp

Shahriz Malek, 920378989

Duccio Rocca, 922254031

Abhiram Rishi Prattipati, 921346982

Christopher Solorzano, 920528216

GitHub: RINO-GAELICO

https://github.com/CSC415-2022-Spring/csc415-filesystem-RINO-GAELICO

April 27, 2022

## Assignment 1 - File System Design

**What does your Volume Control Block look like (what fields and information it might contain)?**

The Volume Control Block on FAT32 file system is called Boot record. It is located in the reserved region and contains the data structure BPB (BIOS Parameter Block). The first sector of the volume is sometimes called the “boot sector”, or the “0th sector”. The BPB describes the physical layout of the data storage volume - it contains volume details such as the total number of blocks, number of free blocks, block size, free block pointers, pointer to the root directory, and type of volume. The Volume Control Block is essential in the File-System implementation.
**VCB** in C structure definition with details of each field:

```
struct BS_BPB {
      char BS_jmpBoot[3]; // see docs for possible values
      char BS_OEMName[8];//system formatting the volume (LINUX?)
      short BPB_BytsPerSec;// It is 512 as per instructions
      char BPB_SecPerClus;//BPB_BytsPerSec*BPB_SecPerClus not>32K
      char BPB_RsvdSecCnt; //in FAT32 is typically 32.
      char BPB_NumFATs; // # of FATs, probably = 2
      short BPB_RootEntCnt; // for FAT32 this must be set to 0
      short BPB_TotSec16; // for FAT32 this must be set to 0
      char BPB_Media; // 0xF8 for fixed → low byte of FAT[0]
      short BPB_FATSz16; // for FAT32 this must be set to 0
      short BPB_SecPerTrk;// Sectors per track for interrupt 0x13
      short BPB_NumHeads; // Number of heads for interrupt 0x13
      int BPB_HiddSec; // = 0 on media that are not partitioned
      int BPB_TotSec32; // total count of sectors on the volume
      int BPB_FATSz32 ; //32bit count of blocks taken by ONE FAT
      short BPB_ExtFlags ; // see tech sheet
      short BPB_FSVer ; // version number of the FAT32 volume
      int BPB_RootClus ;// 1st cluster of the root dr, usually 2
      short BPB_FSInfo ; // Block number of FSINFO, usually 1
      short BPB_BkBootSec ;//block # in reserved area of copy of  //the boot
      record, usually 6
      char BPB_Reserved[12];// Reserved for future expansion.
```

```
// When formatting FAT32 volume, we should set all of the bytes // of this
field to 0.
      char BS_DrvNum ;//This field is operating system specific
      char BS_Reserved1 ;//at formatting set this byte to 0
      char BS_BootSig ;//signature byte
      int BS_VolID ;//Volume serial number
      char BS_VolLab[11]; //Volume label. Match the Vlab in root
      char BS_FilSysType[8];// based on the FAT type: "FAT32   "
}
```

**How will you track the free space?**

In a FAT system, the free space is tracked through the **file-allocation table** (**FAT**). This table uses a variation of a linked allocation to keep track of the position of the files and also of the free space within the disk. To implement the FAT within the disk a region of storage has to be reserved to contain the table. The table has an entry of 32-bit (word) for each block and it is indexed by block number. Each block is pointing to the next to form a sort of linked list. The chain terminates with a special value that indicates the EOF.

Allocating a new block to a file is done by finding the first free block in the table and replacing the previous EOF value with the address of the new block.

**What your directory entry (also called FCB [File Control Block]) will look like (include a structure definition)?**

In a FAT system, a directory is just a "file" composed of an array of 32-byte structures. These structures are called "directory entries". There is only one special directory and it is the root directory. In a FAT32 system, the root directory can be of variable size and is represented by a cluster chain, just like any other directory. There is a specific field in the BPB called BPB_RootClus, that stores the location of the first cluster of the root directory. The root directory does not have a filename and there is no info on file date and file times. Also, the root directory does not contain the classical dot "." and dot dot "..", which are the first and second entries in every other directory.

In a FAT32 system, there are short directory and long directory entries. The latter are just regular (short) directory entries in which the attribute field has a value of ATTR_LONG_NAME,

this indicates that the “file” is part of the long name entry for some other file. The long directory entries contain the long name of a file. The name contained in a short entry is called the alias name.

A long directory entry works as an extension to the short directory entries. A long entry cannot exist by itself. Long entries that are found without being coupled with a valid short entry, are called orphans.

Also, the long directory entries are located in close physical proximity to the short directory entries they are associated with. They are immediately contiguous to the short directory entries that are associated with. The reason why there are these two types is that long directory entries are invisible on old versions of MS-DOS/Windows.

We decided to describe, here below, both short and long directory entry structures. The short/regular directory entry is simply denominated as DIR_ENTRY, while the long one is denominated as LN_DIR_ENTRY.

```
struct DIR_ENTRY {
    char DIR_Name[11]; // filename
    char DIR_Attr; // see tech sheet for values
    char DIR_NTRes; //reserved: set to 0 when a file is created
    short DIR_CrtTimeMs; // Millisecond stamp at file creation
    short DIR_CrtTime; // Time file was created
     short DIR_CrtDate; // Date file was created.
    short DIR_LstAccDate; // Last access date
    char DIR_FstClusHI[2];//High word this entry’s 1st cluster #
    short DIR_WrtTime; // last time modified
    short DIR_WrtDate; // date of last time modified
    char DIR_FstClusLO[2]; // Low word of 1st cluster #
    int DIR_FileSize; //holding this file’s size in bytes
}DIR_ENTRY;

struct LN_DIR_ENTRY {
      char LDIR_Ord; // sequence # in the set of LN_DIR_ENTRYs
      short LDIR_Name1[5]; // utf 16-characters 1-5 of long-name
      char LDIR_Attr ;     // must be ATTR_LONG_NAME
      char LDIR_Type; //0x00 indicate sub-component of long name
```

```
      char LDIR_Chksum;// 11 characters of the alias in DIR_ENTRY // are used
      in the checksum calculation,
      // see docs for algorithm

      short LDIR_Name2[6]; // 6 more utf 16-characters of name
      short LDIR_FstClusLO;  // Must be 0x0000
      short LDIR_Name3[2]; // Characters 12-13 in utf-16
} LN_DIR_ENTRY;
```

**What metadata do you want to have in the file system?**

This metadata is described in detail in the Directory Entry structure. It includes filename, file size, file dates and times (create, last access, modified), ACL, and location.

## Milestone 1 - File System Initialization

**A dump (use the provided HexDump utility) of the volume file that shows the VCB, FreeSpace, and root directory.**

**FATMAP** (Too big for screenshots over 116 pages) ⇐ HyperLinked the Volume File Document

**VCB**

Dumping file SampleVolume, starting at block 1 for 1 block:

```
000200: EB 58 90 4D 53 57 49 4E  34 2E 31 00 02 01 20 00 | ëXÉMSWIN4.1... .
000210: 02 00 00 00 00 F8 00 00  20 00 40 00 00 00 00 00 | .....°.. .@.....
000220: 4B 4C 00 00 96 00 00 00  00 00 00 00 02 00 00 00 | KL..û...........
000230: 01 00 06 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000240: 80 00 29 3B 15 6A 62 20  20 20 20 20 20 20 20 20 | Ç.);.jb
000250: 20 20 46 41 54 33 32 20  20 20 00 00 00 00 00 00 |   FAT32   ......
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 55 AA | ..............U
```

```
struct BS_BPB {
	char BS_jmpBoot[3]; // 3 bytes
	char BS_OEMName[8];// 8 bytes
	short BPB_BytsPerSec;// 2 bytes
	char BPB_SecPerClus;// 1 byte
	char BPB_RsvdSecCnt; // 2 bytes.
	char BPB_NumFATs; // 1 byte
	short BPB_RootEntCnt; // 2 bytes
	short BPB_TotSec16; // 2 bytes
	char BPB_Media; // 1 byte
	short BPB_FATSz16; // 2 bytes
	short BPB_SecPerTrk;// 2 bytes
	short BPB_NumHeads; // 2 bytes
	int BPB_HiddSec; // 4 bytes
	int BPB_TotSec32; // 4 bytes
	int BPB_FATSz32 ; // 4 bytes
	short BPB_ExtFlags ; // 2 bytes
	short BPB_FSVer ; // 2 bytes
	int BPB_RootClus ;// 4 bytes
	short BPB_FSInfo ; // 2 bytes
	short BPB_BkBootSec ;// 2 bytes
	char BPB_Reserved[12];// 12 bytes
	char BS_DrvNum ;// 1 byte
	char BS_Reserved1 ;//1 byte
	char BS_BootSig ;// 1 byte
	int BS_VolID ;// 4 bytes
	char BS_VolLab[11]; //11 bytes
	char BS_FilSysType[8];// 8 bytes
}
```

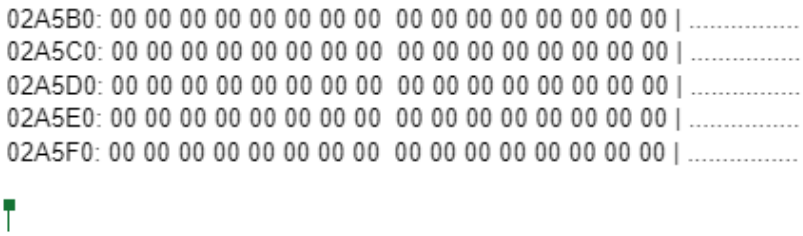
## ROOT DIRECTORY

Dumping file SampleVolume, starting at block 335 for 4 blocks:

```
029E00: 2E 20 20 20 20 20 20 20  20 20 20 00 00 00 00 00 | .          .....
029E10: 00 00 00 00 FF FF 00 00  00 00 00 02 00 00 00 00 | ....  ..........
029E20: 2E 2E 20 20 20 20 20 20  20 20 20 00 00 00 00 00 | ..         .....
029E30: 00 00 00 00 FF FF 00 00  00 00 00 02 00 00 00 00 | ....  ..........
029E40: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029E50: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029E60: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029E70: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029E80: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029E90: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029EA0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029EB0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029EC0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029ED0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029EE0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029EF0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....

029F00: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029F10: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029F20: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029F30: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029F40: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029F50: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029F60: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029F70: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029F80: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029F90: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029FA0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029FB0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029FC0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029FD0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
029FE0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
029FF0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....

02A000: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A010: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A020: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A030: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A040: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A050: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A060: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A070: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
```

```
02A080: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A090: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A0A0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A0B0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A0C0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A0D0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A0E0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A0F0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....

02A100: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A110: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A120: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A130: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A140: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A150: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A160: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A170: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A180: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A190: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A1A0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A1B0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A1C0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A1D0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A1E0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A1F0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....

02A200: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A210: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A220: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A230: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A240: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A250: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A260: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A270: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A280: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A290: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A2A0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A2B0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A2C0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A2D0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A2E0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
02A2F0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....

02A300: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | ..... .....
```

```
02A310: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A320: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A330: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A340: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A350: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A360: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A370: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A380: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A390: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A3A0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A3B0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A3C0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A3D0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....
02A3E0: 00 00 00 00 00 00 00 00  00 00 E5 00 00 00 00 00 | .......... .....
02A3F0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....  ....  ....

02A400: 00 00 00 00 00 00 00 00  B1 D2 00 00 00 00 00 00 | ........  ......
02A410: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A420: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A430: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A440: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A450: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A460: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A470: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A480: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A490: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A4A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A4B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A4C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A4D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A4E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A4F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

02A500: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A510: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A520: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A530: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A540: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A550: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A560: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A570: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A580: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A590: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A5A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```




**A table of who worked on which components**

| | |
|---|---|
| Volume Control Block | Duccio |
| FAT Map / Free Space | Christopher, Duccio, Abhiram |
| Root Directory | Shahriz, Duccio |
| PDF | Everyone |

**How did your team work together, how often you met, how did you meet, how did you divide up the tasks**.

Our team worked very well together. If we had any questions or concerns about something we would first ask each other and if someone has the answer they would try to explain to the best they can. The questions we couldn't answer we would ask the professor or search on Google, then come together to make sure everyone understood the question the best they could. We would note down questions and ask as many as we can in class or office hours. We met many times through different sources. Sometimes we would meet after class for quick questions and concerns. Other times we would meet on Zoom or text through Discord and answer questions. We divided the task in four different ways, two people will work on the FAT map or the free space initialization. One would be working on the Volume control block and the other would be

working on the root directory. Whoever finishes first can help the rest finish their part. Although, say if one person is having trouble with the root directory and no one is finished with their parts, someone else would set aside their work and help the person working on the root directory. Duccio helped us with our tasks and oversaw the project. He kept everyone on track and if we were confused about something he would visualize the best he can for everyone.

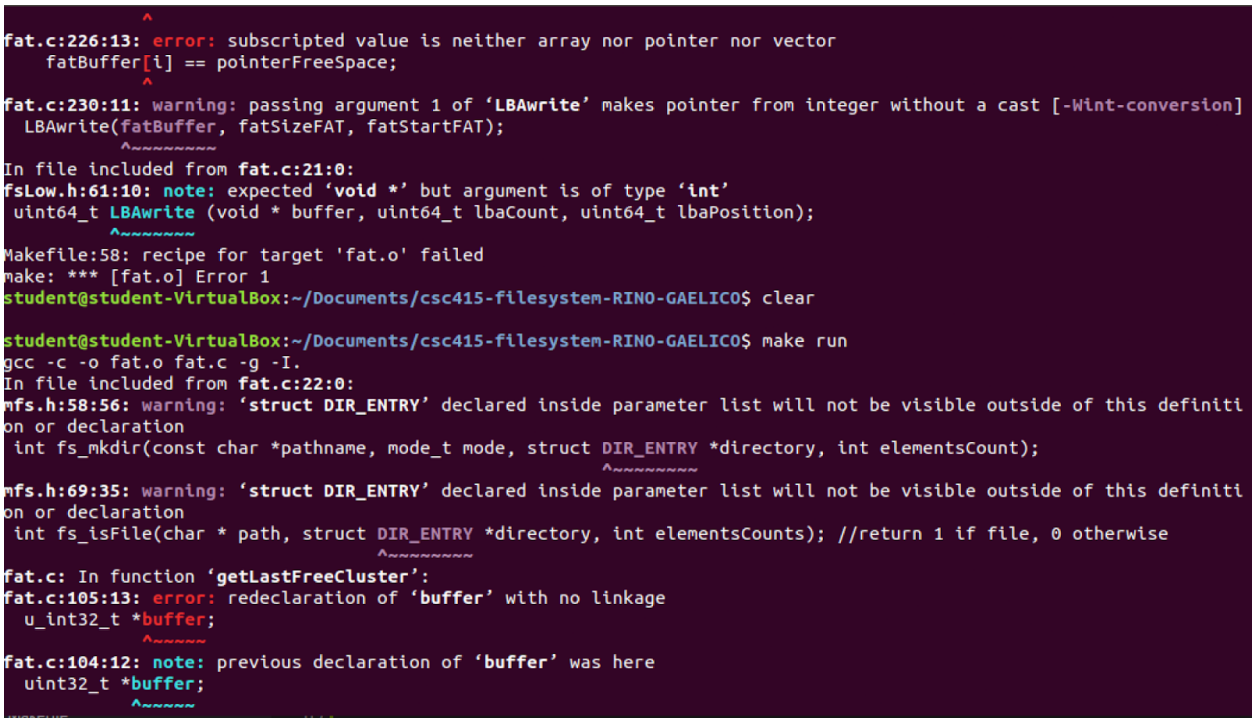
**A discussion of what issues you faced and how your team resolved them.**

The issues that we faced were at first, having trouble with understanding the concepts of a FAT file system and breaking it down into chunks. We spent a few days researching and trying to understand the full picture of the FAT. We went over the FAT32 specs and watched YouYube videos to get a better understanding of how everything works. We also looked at visual images of the FAT32 system to get a better idea of how everything is connected which gave us a better way on how to attack the problem at hand.

Another issue we faced was the last free cluster being initialized to -1. We resolved the issue by deleting the code and leaving it out.

```
student@student-VirtualBox:~/Documents/csc415-filesystem-RINO-GAELICO$ make run
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -c -o fsInit.o fsInit.c -g -I.
gcc -c -o fat.o fat.c -g -I.
gcc -c -o directory.o directory.c -g -I.
gcc -o fsshell fsshell.o fsInit.o fat.o directory.o fsLow.o -g -I. -lm -l readline -
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
FATSIZE IN BLOCKS: 153
CLUSTER IN BLOCKS: 1
DATA START: 338
Number of Blocks in Volume: 19531
BLOCK SIZE, 512
Initialization of FAT successful
Last free cluster: -1
Root start: 2
Prompt > exit
System exiting
student@student-VirtualBox:~/Documents/csc415-filesystem-RINO-GAELICO$
```

Another issue we had was a redeclaration error with no linkage. We spent a good amount of time trying to debug this without realizing how easy of a fix it was.




We solved this by changing the variable type of buffer to u_32_t in order to fix this issue. With fixing this issue, we were able to run the program correctly.

**Milestone 2 - File System Directory**

After parse path was implemented, the other command functions were created such as loadDirectory, fs_mkdir, fs_rmvdir, fs_getcwd, fs_setcwd, fs_isFile, fs_isDir, fs_delete, fs_openDir, fs_diriteminfo, fs_stat. Parse path was an integral method that needed to be created in order for all the other functions to be created. loadDirectory was also a method that was needed for the other methods. Fs_mkdir needed the loadDirectory to be able to load the parent directory and to iterate through it to be able to find an unused Directory Entry to create a Directory Entry. Once the unused Directory Entry was found in the parent, space was allocated for the new directory and the parent array is filled in with the meta data in place for the new directory entry such as the last time modified, fate of last time modified, date file was created, etc. Then the new directory entry is written to disk using LBAwrite and a while loop. Then the parent directory is written to disk. Once the parent directory and the new directory entry is written to disk, then all the memory that was malloced then gets free for proper memory management. All the other methods use parse path heavily to be able to traverse through the Directories.

**Key directory functions:**

**int fs_mkdir(const char *pathname, mode_t mode, struct DIR_ENTRY *directory)**{
(returns 1 = failed, 0 = success)

- relative path
- absolute path
- STEP 1: validation to check if DIR_ENTRY is in the parent
    - it should be a valid path but the last element should not be in the parent dir
- STEP 2: return DE or a pointer to the last element
- STEP 3: find unused DE in parent
- STEP 4: Allocate space for new dir
- STEP 5: init the DE for new DIR
- STEP 6: Fill in parent array at the positionIndex
- STEP 7: write new dir to disk

- STEP 8: parent dir to disk

}

**int fs_rmdir(const char *pathname){**

(returns 1 = failed, 0 = success)

- STEP 1: Validate: make sure the path to the file exists
    - The only used directory entries inside of this directory can be "." and ".." all other directories have to be unused.
- STEP 2:Release the blocks with the directory
- STEP 3: mark the DE as unused in parent
- STEP 4: write to disk
- STEP 5: return results

**}**

**int fs_isFile(char * path, struct DIR_ENTRY *directory, int elementsCount)**{

(return 1 if file, 0 otherwise)

- Relative path
- Absolute path
- STEP 1: Check if the path exists
- STEP 2: Check if it is a file
    - If it is then return 1
    - If not then return 0

}

**int fs_isDir(char * path)**{

(return 1 if directory, 0 otherwise)

- Relative path
- Absolute path
- STEP 1: Check is it is a directory
    - If it is then return

    - If not then return 0

}

**int fs_delete(char* filename){ //removes a file**

- Relative path
- Absolute path
- STEP 1: Validate: make sure the path to the file exists
- STEP 2: Release Blocks associated w/file
- STEP 3: Mark the associated DE as unused
- STEP 4: write dir to disk
- STEP 5: return results

**}**

**Directory iteration functions**

**fdDir * fs_opendir(const char *name)**{

- relative path
- absolute path
- STEP 1: openDir should open the directory
- STEP 2: Check if its a directory
- STEP 3: Open the file

}

**int fs_closedir(fdDir *dirp)**{

- FREE the directory
- Return 0

}

**Misc directory functions**

**char * fs_getcwd(char *buf, size_t size)**{

- Get the current directory and return the buffer

}

**int fs_setcwd(char *buf)**{

- relative path
- absolute path
- Set the current directory
    - If path is not equal to “/” add “/” to it
    - At the end of it free the data

}

**int fs_isFile(char * path)**{

- Creating a directory entry holder
- Parses the path that is being passed through the parameter
- Goes through if statements to check if what is parsed is a directory, if it is not a directory, returns 1 and if it is a directory that it will return 0
- Then within each if statements, frees the direct holder’s components

}

**int fs_isDir(char * path)**{

- Uses parse path function and parses the path that gets passed into the parameters
- Created a directory entry holder to place the path into
- If statement to check if the directory entry holder is a directory if it is then it will free the directory entry holder and return 1.
- If isDir is set to 1 then that means its a directory, otherwise, it is not a directory

}

**struct fs_diriteminfo *fs_readdir(fdDir  *dirp)** {

- Takes in a file descriptor for the directory entry

- Check to make sure that it is a directory or file
- Once it checks, sets the file type to either a FT_Directory or FT_REGFILE
- Increased the dirEntryPosition by 1
- Returns the pointer to the directory item

}

**int fs_stat(const char *path, struct fs_stat *buf)** {

- Takes in a path through the parameters and then parses it based off of if it is a relative or absolute path
- Sets the buffers attributes from the directory entry that is passed in through the method such as creationtime, writetime, filesize, lastAccessedDate, ect
- The buffer is what is being passed in which is the struct fs_stat

}

**Milestone 3 Final - File Control Block**

**b_io_fd b_open (char * filename, int flags)**{

- Parse the path
    - if path is invalid -> error out
    - last component exists or does not exist
        - if last component exits AND if component is a directory - error out
- Only handle files in the open
- If truncate flag is specified here AND you are opening up for write NOT read only,
- set file size to 0 -> delete blob that was associated with the file and set it to 0
- purpose of setting the flags to write, create, and truncate
- If does not exist
    - then create flag is set AND not read only
    - continue or else - error
- have to be opening the file for write or read write
- need to allocate directory entry and file size == 0
- truncate flag specified if it does not exist the create file is specified
- If nothing ignore it

In b_open function, the file system writes to a file if it's in the disk. If it doesn't exist, then the method will create flags related to files such as truncate, create, append, etc. These set the file position to the end of the file before starting to write on the end of the file.

- need to initialize the FCB (file control block) to maintain this file
- malloc the buffer
- set the index to 0
- set valid bytes in buffer to 0 bc haven't read anything
- save ptr to this files directory entry
- set size variable in the structure, how big is this file
- size -> 0

- set file file position , unless append set it to size
- current block #
- isBufferDirty
- ptr to parent directory

}

**int b_seek (b_io_fd fd, off_t offset, int whence)** {

- Initialized our system
- Checks to see if the file descriptor is valid or invalid
- Calculates the end of the buffer
- Checks to see if the offset is reater than or equal to the distance to the end of the buffer
- Then checks to see if the buffer is dirty, if it is then it will flush out the buffer
- If it is not dirty then we stay in the same buffer and add the offset to the index and set the file descriptor file position to the whence plus the offset

}

**int b_write (b_io_fd fd, char * buffer, int count)**{

- Initialized our system
- Checks to see if the file descriptor is valid or invalid
- Checks to see if we need a new allocation for a brand new file
- If the count is greater than the minimum block size then it will write to more than one block
- A block is written one at a time, the the relative fields get updated such as the relative file block position, file position, and the size
- the buffer then gets written on disk and the count gets decreased
- Then the remaining bytes get loaded and written on disk
- If we just need one block then one block gets allocated using are fille allocation free space routine
- If the file is already allocated but they need more space then we give it to them
- We continue to write on top of the buffer until finished

- Whatever is left gets written onto disk
- If there is no more space in the buffer than the buffer gets flushed and reloads once

}

**int b_read (b_io_fd fd, char * buffer, int count)**{

- The bytes to read and the bytes to give is set to 0
- Initialized our system
- Checks to see if the file descriptor is valid or invalid
- If statement to help filling from the existing buffer
- If the count is less than the data that is available then we transferred only the count
- If the count is greater than 0 we fill in multiples of block size
- If we need more blocks if still over 512 then we give it to them until the remaining is less than 512 bytes

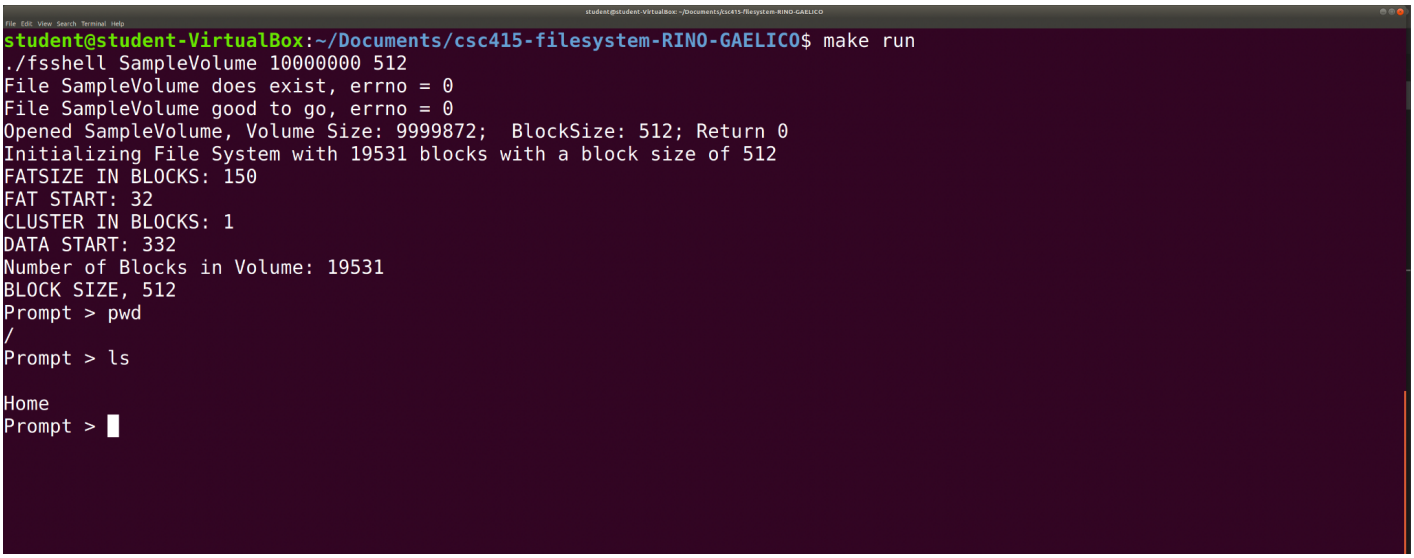
}

**void b_close (b_io_fd fd)**{

- Commit the file to disk if it is open for write, for read you dont
    - the access date, the file size, the mod date changes so need to update the directory
- entry to the file
- make sure the changes get committed to the directory entry
- commitDE(ptrtoDir, indexDE);
- free the file descriptor
- free up the buffer

}

## LS Command Working




Here above we listed the content of / directory using the ls command.

And below you can see the hexdump of the first block of the '/' directory. The first three DE are occupied as expected by dot, dot dot, and Home. The remaining 61 DEs are unused. We decided to use 0x00 as first byte of the name to flag the DE as unused.:

```
029E00: 2E 00 00 00 00 00 00 00  00 00 00 10 00 28 A6 A2 | .............(??
029E10: 8D 54 8D 54 00 00 A6 A2  8D 54 02 02 00 08 00 00 | ?T?T..???T......
029E20: 2E 2E 00 00 00 00 00 00  00 00 00 10 00 28 A6 A2 | .............(??
029E30: 8D 54 8D 54 00 00 A6 A2  8D 54 02 02 00 08 00 00 | ?T?T..???T......
029E40: 48 6F 6D 65 00 00 00 00  00 00 00 10 00 32 AB A2 | Home.........2??
029E50: 8D 54 8D 54 00 00 AB A2  8D 54 06 06 00 08 00 00 | ?T?T..???T......
029E60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029E70: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
029E80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029E90: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
029EA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029EB0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
029EC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029ED0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
029EE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029EF0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
```

```
029F00: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029F10: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....ﾀﾀ....ﾀﾀ....
029F20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029F30: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....ﾀﾀ....ﾀﾀ....
029F40: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029F50: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....ﾀﾀ....ﾀﾀ....
029F60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029F70: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....ﾀﾀ....ﾀﾀ....
029F80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029F90: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....ﾀﾀ....ﾀﾀ....
029FA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029FB0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....ﾀﾀ....ﾀﾀ....
029FC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029FD0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....ﾀﾀ....ﾀﾀ....
029FE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029FF0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....ﾀﾀ....ﾀﾀ....
```

## Md Command Working

```
student@student-VirtualBox:~/Documents/csc415-filesystem-RINO-GAELICO$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
FATSIZE IN BLOCKS: 150
FAT START: 32
CLUSTER IN BLOCKS: 1
DATA START: 332
Number of Blocks in Volume: 19531
BLOCK SIZE, 512
Prompt > pwd
/
Prompt > ls

Home
Prompt > md Student
Prompt > ls

Home
Student
Prompt >
```

Here above we created a new directory inside '/' called Student.

And below you can see the hexdump of the first half block of the '/' directory.  Student's DE is the fourth DE as expected.

Dumping file SampleVolume, starting at block 335 for 4 blocks:

```
029E00: 2E 00 00 00 00 00 00 00  00 00 00 10 00 28 A6 A2 | .............(??
029E10: 8D 54 8D 54 00 00 A6 A2  8D 54 02 02 00 08 00 00 | ?T?T..???T......
029E20: 2E 2E 00 00 00 00 00 00  00 00 00 10 00 28 A6 A2 | ..............(??
029E30: 8D 54 8D 54 00 00 A6 A2  8D 54 02 02 00 08 00 00 | ?T?T..???T......
029E40: 48 6F 6D 65 00 00 00 00  00 00 00 10 00 32 AB A2 | Home.........2??
029E50: 8D 54 8D 54 00 00 AB A2  8D 54 06 06 00 08 00 00 | ?T?T..???T......
029E60: 53 74 75 64 65 6E 74 00  00 00 00 10 00 F6 BD A5 | Student......???
029E70: 8D 54 8D 54 00 00 BD A5  8D 54 0E 0E 00 08 00 00 | ?T?T..???T......
029E80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029E90: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
029EA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029EB0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
029EC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029ED0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
029EE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
029EF0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
```

Here below we can see where the directory has been allocated in the FAT map:

```
Dumping file SampleVolume, starting at block 33 for 1 block:

004200: F8 FF FF 0F F8 FF FF 0F  03 00 00 00 04 00 00 00 |....????........
004210: 05 00 00 00 FF FF FF FF  07 00 00 00 08 00 00 00 |....????........
004220: 09 00 00 00 FF FF FF FF  0B 00 00 00 0C 00 00 00 |....????........
004230: 0D 00 00 00 FF FF FF FF  0F 00 00 00 10 00 00 00 |....????........
004240: 11 00 00 00 FF FF FF FF  13 00 00 00 14 00 00 00 |....????........
004250: 15 00 00 00 16 00 00 00  17 00 00 00 18 00 00 00 |................
004260: 19 00 00 00 1A 00 00 00  1B 00 00 00 1C 00 00 00 |................
004270: 1D 00 00 00 1E 00 00 00  1F 00 00 00 20 00 00 00 | ............... 
004280: 21 00 00 00 22 00 00 00  23 00 00 00 24 00 00 00 |!...".../#...$...
004290: 25 00 00 00 26 00 00 00  27 00 00 00 28 00 00 00 |%...&.../'...(...
0042A0: 29 00 00 00 2A 00 00 00  2B 00 00 00 2C 00 00 00 |)...*.../+.../,...
0042B0: 2D 00 00 00 2E 00 00 00  2F 00 00 00 30 00 00 00 |-......./.../0...
0042C0: 31 00 00 00 32 00 00 00  33 00 00 00 34 00 00 00 |1.../2.../3.../4...
0042D0: 35 00 00 00 36 00 00 00  37 00 00 00 38 00 00 00 |5.../6.../7.../8...
0042E0: 39 00 00 00 3A 00 00 00  3B 00 00 00 3C 00 00 00 |9.../:.../;.../<...
0042F0: 3D 00 00 00 3E 00 00 00  3F 00 00 00 40 00 00 00 |=.../>.../?.../@...
```

After Root (in red, 4 blocks) and Home (in blue, 4 blocks), we can see Student (in yellow, 4 blocks). Student starts at position 10 (which is pointing to 11, in hexa 0x0B) and ends at position 0x0D (occupied by the flag 0xFF FF FF FF for EOF).

Here below we can see the same directory in the Data section of the disk. Since it starts at block 10th, performing a hexdump of the 10th block after the data section starting point we can see the first block of the Student directory.

Dumping file SampleVolume, starting at block 343 for 4 blocks:

```
02AE00: 2E 00 00 00 00 00 00 00  00 00 00 10 00 52 2B AA | .............R+?
02AE10: 8D 54 8D 54 00 00 2B AA  8D 54 0A 0A 00 08 00 00 | ?T?T..+??T......
02AE20: 2E 2E 00 00 00 00 00 00  00 00 00 10 00 3B 1D AA | ..............;.?
02AE30: 8D 54 8D 54 00 00 8D 54  8D 54 02 02 00 08 00 00 | ?T?T..?T?T......
02AE40: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02AE50: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
02AE60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02AE70: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....??....??....
```

```
02AE80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02AE90: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....××....××....
02AEA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02AEB0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....××....××....
02AEC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02AED0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....××....××....
02AEE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02AEF0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....××....××....
```

The first two DE are “dot” and “dot dot” as expected.

## RM Command Working

Here we used rm to remove the Student directory from '/' directory.

```
student@student-VirtualBox:~/Documents/csc415-filesystem-RINO-GAELICO$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
FATSIZE IN BLOCKS: 150
FAT START: 32
CLUSTER IN BLOCKS: 1
DATA START: 332
Number of Blocks in Volume: 19531
BLOCK SIZE, 512
Prompt > ls -l

D         2048   Home
D         2048   Student
Prompt > rm Student
Prompt > ls -la

D         2048   .
D         2048   ..
D         2048   Home
Prompt >
```

As we can see here below, Student is no more present in the FAT map either.

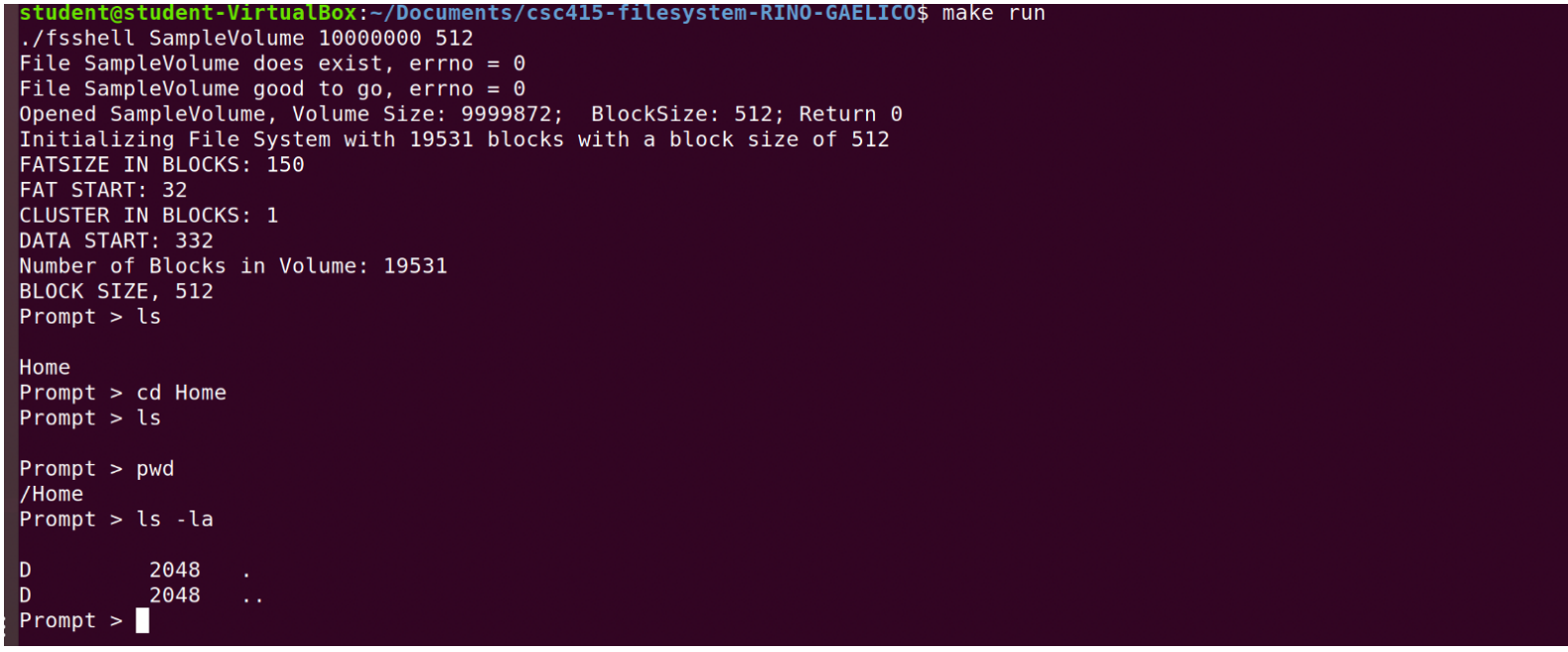
Dumping file SampleVolume, starting at block 33 for 4 blocks:

```
004200: F8 FF FF 0F F8 FF FF 0F  03 00 00 00 04 00 00 00 | ...................
004210: 05 00 00 00 FF FF FF FF  07 00 00 00 08 00 00 00 | ....????........
004220: 09 00 00 00 FF FF FF FF  0B 00 00 00 0C 00 00 00 | ....????........
004230: 0D 00 00 00 0E 00 00 00  0F 00 00 00 10 00 00 00 | ................
004240: 11 00 00 00 12 00 00 00  13 00 00 00 14 00 00 00 | ................
004250: 15 00 00 00 16 00 00 00  17 00 00 00 18 00 00 00 | ................
004260: 19 00 00 00 1A 00 00 00  1B 00 00 00 1C 00 00 00 | ................
004270: 1D 00 00 00 1E 00 00 00  1F 00 00 00 20 00 00 00 | ............ ...
004280: 21 00 00 00 22 00 00 00  23 00 00 00 24 00 00 00 | !..."...#...$...
004290: 25 00 00 00 26 00 00 00  27 00 00 00 28 00 00 00 | %...&...'...(...
0042A0: 29 00 00 00 2A 00 00 00  2B 00 00 00 2C 00 00 00 | )...*...+...,...
0042B0: 2D 00 00 00 2E 00 00 00  2F 00 00 00 30 00 00 00 | -......./...0...
0042C0: 31 00 00 00 32 00 00 00  33 00 00 00 34 00 00 00 | 1...2...3...4...
0042D0: 35 00 00 00 36 00 00 00  37 00 00 00 38 00 00 00 | 5...6...7...8...
```

```
0042E0: 39 00 00 00 3A 00 00 00  3B 00 00 00 3C 00 00 00 | 9...:...;...<...
0042F0: 3D 00 00 00 3E 00 00 00  3F 00 00 00 40 00 00 00 | =...>...?...@...
```

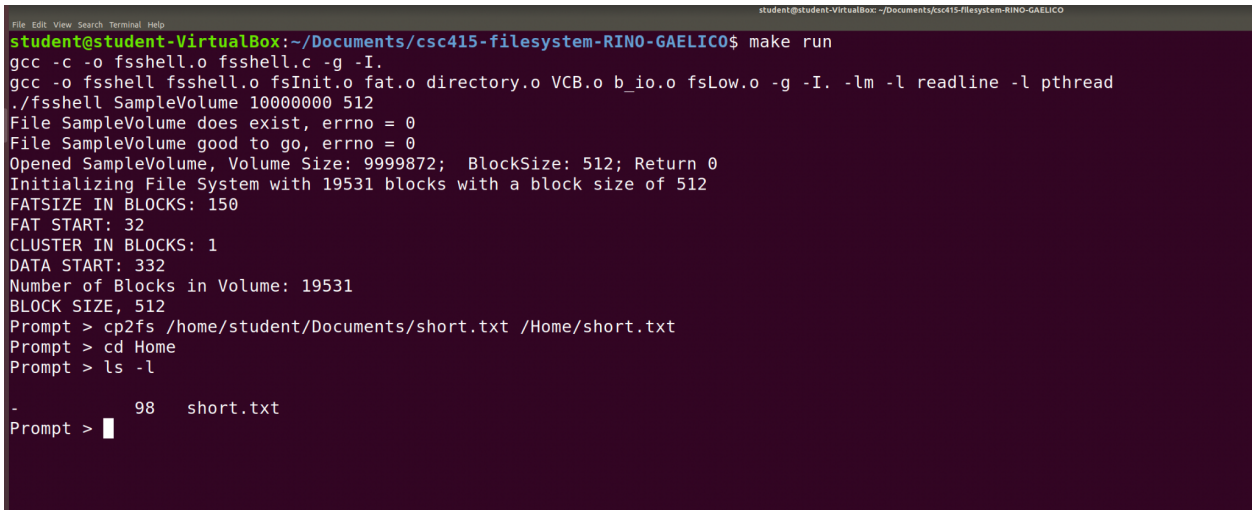
**PWD and CD Command Working**

Here below we used pwd and cd to change to the current working directory to /Home and then printed the current working directory.

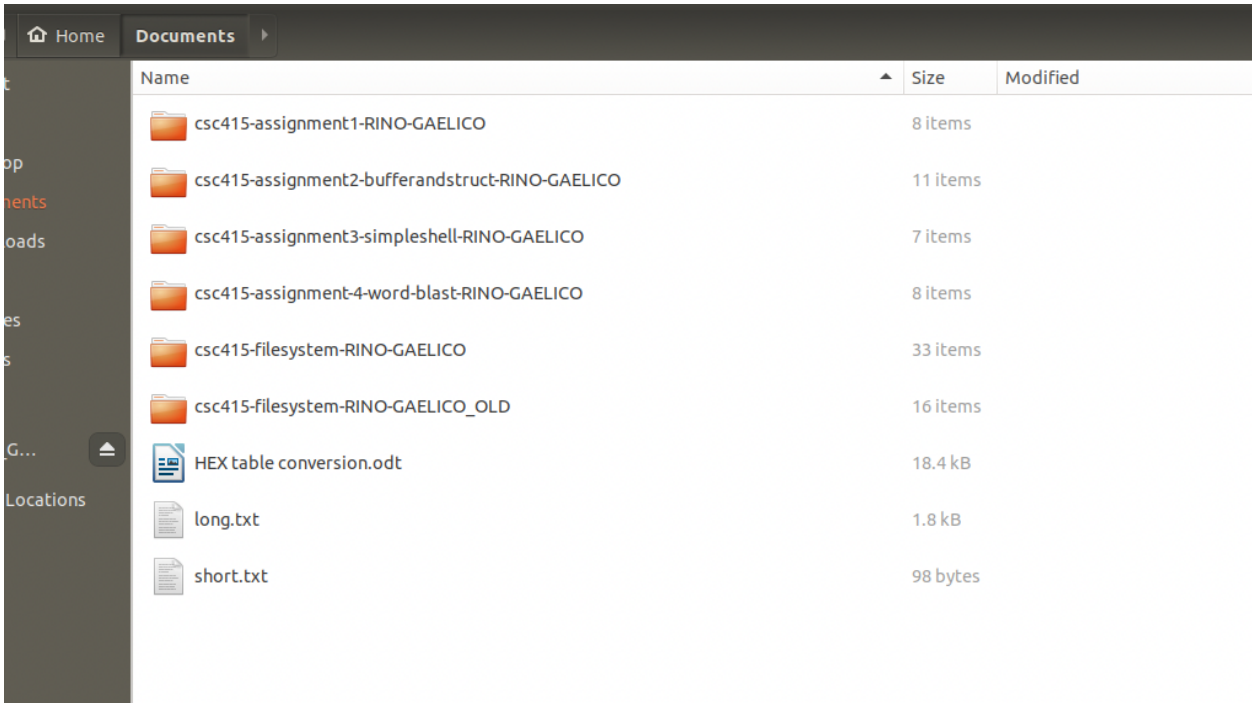


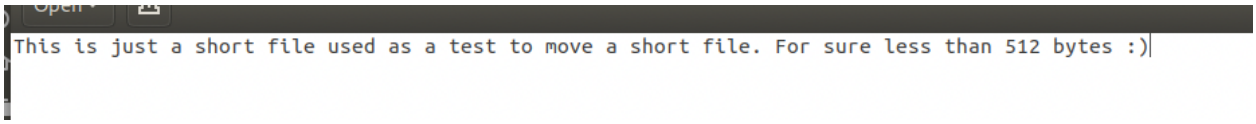
## cp2fs Command Working

Here we used cp2fs to copy a long file and a short file from Linux into our fs:




As we can see here the file src is in linux in /home/student/short.txt and it is 98 bytes long:






Here is the hexdump in our file system:

Dumping file SampleVolume, starting at block 343 for 1 block:

```
02AE00: EF BB BF 54 68 69 73 20  69 73 20 6A 75 73 74 20 | This is just
02AE10: 61 20 73 68 6F 72 74 20  66 69 6C 65 20 75 73 65 | a short file use
02AE20: 64 20 61 73 20 61 20 74  65 73 74 20 74 6F 20 6D | d as a test to m
02AE30: 6F 76 65 20 61 20 73 68  6F 72 74 20 66 69 6C 65 | ove a short file
02AE40: 2E 20 46 6F 72 20 73 75  72 65 20 6C 65 73 73 20 | . For sure less
02AE50: 74 68 61 6E 20 35 31 32  20 62 79 74 65 73 20 3A | than 512 bytes :
02AE60: 29 0A 0A CB FC 55 00 00  20 27 0A CB FC 55 00 00 | ).....U.. '.....U..
02AE70: 40 27 0A CB FC 55 00 00  60 27 0A CB FC 55 00 00 | @'.....U..`'.....U..
02AE80: 80 27 0A CB FC 55 00 00  A0 27 0A CB FC 55 00 00 | .'.....U...'.....U..
02AE90: C0 27 0A CB FC 55 00 00  E0 27 0A CB FC 55 00 00 | .'.....U...'.....U..
02AEA0: 00 28 0A CB FC 55 00 00  20 28 0A CB FC 55 00 00 | .(.....U.. (.....U..
02AEB0: 40 28 0A CB FC 55 00 00  60 28 0A CB FC 55 00 00 | @(.....U..`(.....U..
02AEC0: 80 28 0A CB FC 55 00 00  A0 28 0A CB FC 55 00 00 | .(.....U...(.....U..
02AED0: C0 28 0A CB FC 55 00 00  E0 28 0A CB FC 55 00 00 | .(.....U...(.....U..
02AEE0: 00 29 0A CB FC 55 00 00  20 29 0A CB FC 55 00 00 | .).....U.. ).....U..
02AEF0: 40 29 0A CB FC 55 00 00  60 29 0A CB FC 55 00 00 | @).....U..`).....U..
```

Here, below, is the hexdump of the FAT map with the file occupying only one block, therefore marked with EOF flag.

Dumping file SampleVolume, starting at block 33 for 1 block:

```
004200: F8 FF FF 0F F8 FF FF 0F  03 00 00 00 04 00 00 00 | ???.???.........
004210: 05 00 00 00 FF FF FF FF  07 00 00 00 08 00 00 00 | ....????........
004220: 09 00 00 00 FF FF FF FF  FF FF FF FF 0C 00 00 00 | ....????????....
004230: 0D 00 00 00 0E 00 00 00  0F 00 00 00 10 00 00 00 | ................
004240: 11 00 00 00 12 00 00 00  13 00 00 00 14 00 00 00 | ................
004250: 15 00 00 00 16 00 00 00  17 00 00 00 18 00 00 00 | ................
004260: 19 00 00 00 1A 00 00 00  1B 00 00 00 1C 00 00 00 | ................
004270: 1D 00 00 00 1E 00 00 00  1F 00 00 00 20 00 00 00 | ............ ...
004280: 21 00 00 00 22 00 00 00  23 00 00 00 24 00 00 00 | !..."...#...$...
004290: 25 00 00 00 26 00 00 00  27 00 00 00 28 00 00 00 | %...&...'...(...
0042A0: 29 00 00 00 2A 00 00 00  2B 00 00 00 2C 00 00 00 | )...*...+...,...
0042B0: 2D 00 00 00 2E 00 00 00  2F 00 00 00 30 00 00 00 | -......./...0...
0042C0: 31 00 00 00 32 00 00 00  33 00 00 00 34 00 00 00 | 1...2...3...4...
0042D0: 35 00 00 00 36 00 00 00  37 00 00 00 38 00 00 00 | 5...6...7...8...
0042E0: 39 00 00 00 3A 00 00 00  3B 00 00 00 3C 00 00 00 | 9...:...;...<...
0042F0: 3D 00 00 00 3E 00 00 00  3F 00 00 00 40 00 00 00 | =...>...?...@...
```

And finally here is the hexdump of the /Home directory with the DE of the short.txt file. As expected it is the third DE after ‘dot’ and ‘dot dot’.

Dumping file SampleVolume, starting at block 339 for 1 block:

```
02A600: 2E 00 00 00 00 00 00 00  00 00 00 10 00 42 23 AA | .............B#?
02A610: 8D 54 8D 54 00 00 23 AA  8D 54 06 06 00 08 00 00 | ?T?T..#??T......
02A620: 2E 2E 00 00 00 00 00 00  00 00 00 10 00 3B 1D AA | .............;.?
```

```
02A630: 8D 54 8D 54 00 00 8D 54  8D 54 02 02 00 08 00 00 | ▯T▯T..▯T▯T......
02A640: 73 68 6F 72 74 2E 74 78  74 00 00 00 00 2F 5B AE | short.txt..../[▯
02A650: 8D 54 8D 54 00 00 5B AE  8D 54 0A 0A 62 00 00 00 | ▯T▯T..[▯▯T..b...
02A660: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A670: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....▯▯....▯▯....
02A680: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A690: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....▯▯....▯▯....
02A6A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A6B0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....▯▯....▯▯....
02A6C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A6D0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....▯▯....▯▯....
02A6E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A6F0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....▯▯....▯▯....
```

One more example we did with a long file. It is 1.8kb length.

Here is the content of the file:

```
1)First line: this is a test for a long file:
2)second line
3)third line: it should be longer than 512 for sure
4)fourth
5)fifth: possibly more than 1000 bytes, do yu think is possible?
6)This is a test... lalala... so wonderful
7)seventh
8) eighth
9)nineth
10) wow we got this far, fantastic
11) more?
12) twelth
13) This is the longest line and it will help to make this file very long I am sure about it
14) it takes a lot of words to get to a thousand bytes file
15) I didn't image it would take this long;
16) POEM:  Curtains forcing their willagainst the wind,children sleep,exchanging dreams withseraphin. The citydrags itself awake on\subway straps; and, an alarm, awake as arumor of warlie stretching into dawn unasked and unheeded.
17) POEM2: when the risk to remain tight in a bud was more painful than the risk it took to blossom.
18) POEM3: this gin-heavy heaven, blessed ground to think gay & mean we. /bless the fake id & the bouncer who knew /this need to be needed, to belong, to know how /a man taste full on vodka & free of sin. i know not which god to pray to. /i look to christ, i look to every
mouth on the dance floor, i order /a whiskey coke, name it the blood of my new savior. he is just. /he begs me to dance, to marvel men with the / of hips i brought, he deems my mouth in some stranger's mouth necessary. / sssssbless that man's mouth, the song we sway sloppy
to, the beat, the bridge, the length / of his hand on my thigh & back & i know not which country i am of. / i want to live on his tongue, build a home of gospel & gayety / i want to raise a city behind his teeth for all boys of choirs & closets to refuge in. / i want my new
god to look at the mecca i built him & call it damn good / or maybe i'm just tipsy & free for the first time, willing to worship anything i can taste. /
```

```
student@student-VirtualBox:~/Documents/csc415-filesystem-RINO-GAELICO$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
FATSIZE IN BLOCKS: 150
FAT START: 32
CLUSTER IN BLOCKS: 1
DATA START: 332
Number of Blocks in Volume: 19531
BLOCK SIZE, 512
Prompt > ls

Home
Prompt > cd Home
Prompt > rm short.txt
Prompt > ls

Prompt > cp2fs /home/student/Documents/long.txt /Home/long.txt
Prompt > ls

long.txt
Prompt > ls -l

-         1829   long.txt
Prompt >
```

Here is the DE for  LONG.TXT:

Dumping file SampleVolume, starting at block 339 for 1 block:

```
02A600: 2E 00 00 00 00 00 00 00  00 00 00 10 00 42 23 AA | .............B#
02A610: 8D 54 8D 54 00 00 23 AA  8D 54 06 06 00 08 00 00 | T T..#T......
02A620: 2E 2E 00 00 00 00 00 00  00 00 00 10 00 3B 1D AA | .............;.
02A630: 8D 54 8D 54 00 00 8D 54  8D 54 02 02 00 08 00 00 | T T..T T......
02A640: 6C 6F 6E 67 2E 74 78 74  00 00 00 00 00 F9 6A B1 | long.txt.....j
```

```
02A650: 8D 54 8D 54 00 00 6A B1  8D 54 0A 0A 25 07 00 00 | ▯T▯T..j▯▯T..%...
02A660: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A670: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....▯▯....▯▯....
02A680: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A690: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....▯▯....▯▯....
02A6A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A6B0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....▯▯....▯▯....
02A6C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A6D0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....▯▯....▯▯....
02A6E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02A6F0: 00 00 00 00 FF FF 00 00  00 00 FF FF 00 00 00 00 | ....▯▯....▯▯....
```

And finally the file itself:

Dumping file SampleVolume, starting at block 343 for 5 blocks:

```
02AE00: 31 29 46 69 72 73 74 20  6C 69 6E 65 3A 20 74 68 | 1)First line: th
02AE10: 69 73 20 69 73 20 61 20  74 65 73 74 20 66 6F 72 | is is a test for
02AE20: 20 61 20 6C 6F 6E 67 20  66 69 6C 65 3A 20 0A 32 |  a long file: .2
02AE30: 29 73 65 63 6F 6E 64 20  6C 69 6E 65 0A 33 29 74 | )second line.3)t
02AE40: 68 69 72 64 20 6C 69 6E  65 3A 20 69 74 20 73 68 | hird line: it sh
02AE50: 6F 75 6C 64 20 62 65 20  6C 6F 6E 67 65 72 20 74 | ould be longer t
02AE60: 68 61 6E 20 35 31 32 20  66 6F 72 20 73 75 72 65 | han 512 for sure
02AE70: 20 0A 34 29 66 6F 75 72  74 68 0A 35 29 66 69 66 |  .4)fourth.5)fif
02AE80: 74 68 3A 20 70 6F 73 73  69 62 6C 79 20 6D 6F 72 | th: possibly mor
```

```
02AE90: 65 20 74 68 61 6E 20 31  30 30 30 20 62 79 74 65 | e than 1000 byte
02AEA0: 73 2C 20 64 6F 20 79 75  20 74 68 69 6E 6B 20 69 | s, do yu think i
02AEB0: 73 20 70 6F 73 73 69 62  6C 65 3F 0A 36 29 54 68 | s possible?.6)Th
02AEC0: 69 73 20 69 73 20 61 20  74 65 73 74 2E 2E 2E 20 | is is a test...
02AED0: 6C 61 6C 61 6C 61 2E 2E  2E 20 73 6F 20 77 6F 6E | lalala... so won
02AEE0: 64 65 72 66 75 6C 0A 37  29 73 65 76 65 6E 74 68 | derful.7)seventh
02AEF0: 0A 38 29 20 65 69 67 68  74 68 0A 39 29 6E 69 6E | .8) eighth.9)nin
02AF00: 65 74 68 0A 31 30 29 20  77 6F 77 20 77 65 20 67 | eth.10) wow we g
02AF10: 6F 74 20 74 68 69 73 20  66 61 72 2C 20 66 61 6E | ot this far, fan
02AF20: 74 61 73 74 69 63 0A 31  31 29 20 6D 6F 72 65 3F | tastic.11) more?
02AF30: 0A 31 32 29 20 74 77 65  6C 74 68 0A 31 33 29 20 | .12) twelth.13)
02AF40: 54 68 69 73 20 69 73 20  74 68 65 20 6C 6F 6E 67 | This is the long
02AF50: 65 73 74 20 6C 69 6E 65  20 61 6E 64 20 69 74 20 | est line and it
02AF60: 77 69 6C 6C 20 68 65 6C  70 20 74 6F 20 6D 61 6B | will help to mak
02AF70: 65 20 74 68 69 73 20 66  69 6C 65 20 76 65 72 79 | e this file very
02AF80: 20 6C 6F 6E 67 20 49 20  61 6D 20 73 75 72 65 20 |  long I am sure
02AF90: 61 62 6F 75 74 20 69 74  0A 31 34 29 20 69 74 20 | about it.14) it
02AFA0: 74 61 6B 65 73 20 61 20  6C 6F 74 20 6F 66 20 77 | takes a lot of w
02AFB0: 6F 72 64 73 20 74 6F 20  67 65 74 20 74 6F 20 61 | ords to get to a
02AFC0: 20 74 68 6F 75 73 61 6E  64 20 62 79 74 65 73 20 |  thousand bytes
02AFD0: 66 69 6C 65 0A 31 35 29  20 49 20 64 69 64 6E 27 | file.15) I didn'
02AFE0: 74 20 69 6D 61 67 65 20  69 74 20 77 6F 75 6C 64 | t image it would
02AFF0: 20 74 61 6B 65 20 74 68  69 73 20 6C 6F 6E 67 3B |  take this long;
02B000: 0A 31 36 29 20 50 4F 45  4D 3A 20 20 43 75 72 74 | .16) POEM:  Curt
02B010: 61 69 6E 73 20 66 6F 72  63 69 6E 67 20 74 68 65 | ains forcing the
```

```
02B020: 69 72 20 77 69 6C 6C 61  67 61 69 6E 73 74 20 74 | ir willagainst t
02B030: 68 65 20 77 69 6E 64 2C  63 68 69 6C 64 72 65 6E | he wind,children
02B040: 20 73 6C 65 65 70 2C 65  78 63 68 61 6E 67 69 6E |  sleep,exchangin
02B050: 67 20 64 72 65 61 6D 73  20 77 69 74 68 73 65 72 | g dreams withser
02B060: 61 70 68 69 6D 2E 20 54  68 65 20 63 69 74 79 64 | aphim. The cityd
02B070: 72 61 67 73 20 69 74 73  65 6C 66 20 61 77 61 6B | rags itself awak
02B080: 65 20 6F 6E 5C 73 75 62  77 61 79 20 73 74 72 61 | e on\subway stra
02B090: 70 73 3B 20 61 6E 64 2C  20 61 6E 20 61 6C 61 72 | ps; and, an alar
02B0A0: 6D 2C 20 61 77 61 6B 65  20 61 73 20 61 72 75 6D | m, awake as arum
02B0B0: 6F 72 20 6F 66 20 77 61  72 6C 69 65 20 73 74 72 | or of warlie str
02B0C0: 65 74 63 68 69 6E 67 20  69 6E 74 6F 20 64 61 77 | etching into daw
02B0D0: 6E 20 75 6E 61 73 6B 65  64 20 61 6E 64 20 75 6E | n unasked and un
02B0E0: 68 65 65 64 65 64 2E 0A  31 37 29 20 50 4F 45 4D | heeded..17) POEM
02B0F0: 32 3A 20 77 68 65 6E 20  74 68 65 20 72 69 73 6B | 2: when the risk

02B100: 20 74 6F 20 72 65 6D 61  69 6E 20 74 69 67 68 74 |  to remain tight
02B110: 20 69 6E 20 61 20 62 75  64 20 77 61 73 20 6D 6F |  in a bud was mo
02B120: 72 65 20 70 61 69 6E 66  75 6C 20 74 68 61 6E 20 | re painful than
02B130: 74 68 65 20 72 69 73 6B  20 69 74 20 74 6F 6F 6B | the risk it took
02B140: 20 74 6F 20 62 6C 6F 73  73 6F 6D 2E 0A 31 38 29 |  to blossom..18)
02B150: 20 50 4F 45 4D 33 3A 20  74 68 69 73 20 67 69 6E |  POEM3: this gin
02B160: 2D 68 65 61 76 79 20 68  65 61 76 65 6E 2C 20 62 | -heavy heaven, b
02B170: 6C 65 73 73 65 64 20 67  72 6F 75 6E 64 20 74 6F | lessed ground to
02B180: 20 74 68 69 6E 6B 20 67  61 79 20 26 20 6D 65 61 |  think gay & mea
02B190: 6E 20 77 65 2E 20 2F 62  6C 65 73 73 20 74 68 65 | n we. /bless the
```

```
02B1A0: 20 66 61 6B 65 20 69 64  20 26 20 74 68 65 20 62 |  fake id & the b
02B1B0: 6F 75 6E 63 65 72 20 77  68 6F 20 6B 6E 65 77 20 | ouncer who knew
02B1C0: 2F 74 68 69 73 20 6E 65  65 64 20 74 6F 20 62 65 | /this need to be
02B1D0: 20 6E 65 65 64 65 64 2C  20 74 6F 20 62 65 6C 6F |  needed, to belo
02B1E0: 6E 67 2C 20 74 6F 20 6B  6E 6F 77 20 68 6F 77 20 | ng, to know how
02B1F0: 2F 61 20 6D 61 6E 20 74  61 73 74 65 20 66 75 6C | /a man taste ful

02B200: 6C 20 6F 6E 20 76 6F 64  6B 61 20 26 20 66 72 65 | l on vodka & fre
02B210: 65 20 6F 66 20 73 69 6E  2E 20 69 20 6B 6E 6F 77 | e of sin. i know
02B220: 20 6E 6F 74 20 77 68 69  63 68 20 67 6F 64 20 74 |  not which god t
02B230: 6F 20 70 72 61 79 20 74  6F 2E 20 2F 69 20 6C 6F | o pray to. /i lo
02B240: 6F 6B 20 74 6F 20 63 68  72 69 73 74 2C 20 69 20 | ok to christ, i
02B250: 6C 6F 6F 6B 20 74 6F 20  65 76 65 72 79 20 6D 6F | look to every mo
02B260: 75 74 68 20 6F 6E 20 74  68 65 20 64 61 6E 63 65 | uth on the dance
02B270: 20 66 6C 6F 6F 72 2C 20  69 20 6F 72 64 65 72 20 |  floor, i order
02B280: 2F 61 20 77 68 69 73 6B  65 79 20 63 6F 6B 65 2C | /a whiskey coke,
02B290: 20 6E 61 6D 65 20 69 74  20 74 68 65 20 62 6C 6F |  name it the blo
02B2A0: 6F 64 20 6F 66 20 6D 79  20 6E 65 77 20 73 61 76 | od of my new sav
02B2B0: 69 6F 72 2E 20 68 65 20  69 73 20 6A 75 73 74 2E | ior. he is just.
02B2C0: 20 2F 68 65 20 62 65 67  73 20 6D 65 20 74 6F 20 |  /he begs me to
02B2D0: 64 61 6E 63 65 2C 20 74  6F 20 6D 61 72 76 65 6C | dance, to marvel
02B2E0: 20 6D 65 6E 20 77 69 74  68 20 74 68 65 20 2F 20 |  men with the /
02B2F0: 6F 66 20 68 69 70 73 20  69 20 62 72 6F 75 67 68 | of hips i brough

02B300: 74 2C 20 68 65 20 64 65  65 6D 73 20 6D 79 20 6D | t, he deems my m
```

```
02B310: 6F 75 74 68 20 69 6E 20  73 6F 6D 65 20 73 74 72 | outh in some str
02B320: 61 6E 67 65 72 E2 80 99  73 20 6D 6F 75 74 68 20 | anger’s mouth
02B330: 6E 65 63 65 73 73 61 72  79 2E 20 2F 20 73 73 73 | necessary. / sss
02B340: 73 73 73 62 6C 65 73 73  20 74 68 61 74 20 6D 61 | sssbless that ma
02B350: 6E E2 80 99 73 20 6D 6F  75 74 68 2C 20 74 68 65 | n’s mouth, the
02B360: 20 73 6F 6E 67 20 77 65  20 73 77 61 79 20 73 6C |  song we sway sl
02B370: 6F 70 70 79 20 74 6F 2C  20 74 68 65 20 62 65 61 | oppy to, the bea
02B380: 74 2C 20 74 68 65 20 62  72 69 64 67 65 2C 20 74 | t, the bridge, t
02B390: 68 65 20 6C 65 6E 67 74  68 20 2F 20 6F 66 20 68 | he length / of h
02B3A0: 69 73 20 68 61 6E 64 20  6F 6E 20 6D 79 20 74 68 | is hand on my th
02B3B0: 69 67 68 20 26 20 62 61  63 6B 20 26 20 69 20 6B | igh & back & i k
02B3C0: 6E 6F 77 20 6E 6F 74 20  77 68 69 63 68 20 63 6F | now not which co
02B3D0: 75 6E 74 72 79 20 69 20  61 6D 20 6F 66 2E 20 2F | untry i am of. /
02B3E0: 20 69 20 77 61 6E 74 20  74 6F 20 6C 69 76 65 20 |  i want to live
02B3F0: 6F 6E 20 68 69 73 20 74  6F 6E 67 75 65 2C 20 62 | on his tongue, b

02B400: 75 69 6C 64 20 61 20 68  6F 6D 65 20 6F 66 20 67 | uild a home of g
02B410: 6F 73 70 65 6C 20 26 20  67 61 79 65 74 79 20 2F | ospel & gayety /
02B420: 20 69 20 77 61 6E 74 20  74 6F 20 72 61 69 73 65 |  i want to raise
02B430: 20 61 20 63 69 74 79 20  62 65 68 69 6E 64 20 68 |  a city behind h
02B440: 69 73 20 74 65 65 74 68  20 66 6F 72 20 61 6C 6C | is teeth for all
02B450: 20 62 6F 79 73 20 6F 66  20 63 68 6F 69 72 73 20 |  boys of choirs
02B460: 26 20 63 6C 6F 73 65 74  73 20 74 6F 20 72 65 66 | & closets to ref
02B470: 75 67 65 20 69 6E 2E 20  2F 20 69 20 77 61 6E 74 | uge in. / i want
02B480: 20 6D 79 20 6E 65 77 20  67 6F 64 20 74 6F 20 6C |  my new god to l
```

```
02B490: 6F 6F 6B 20 61 74 20 74  68 65 20 6D 65 63 63 61 | ook at the mecca
02B4A0: 20 69 20 62 75 69 6C 74  20 68 69 6D 20 26 20 63 |  i built him & c
02B4B0: 61 6C 6C 20 69 74 20 64  61 6D 6E 20 67 6F 6F 64 | all it damn good
02B4C0: 20 2F 20 6F 72 20 6D 61  79 62 65 20 69 E2 80 99 |  / or maybe i’
02B4D0: 6D 20 6A 75 73 74 20 74  69 70 73 79 20 26 20 66 | m just tipsy & f
02B4E0: 72 65 65 20 66 6F 72 20  74 68 65 20 66 69 72 73 | ree for the firs
02B4F0: 74 20 74 69 6D 65 2C 20  77 69 6C 6C 69 6E 67 20 | t time, willing

02B500: 74 6F 20 77 6F 72 73 68  69 70 20 61 6E 79 74 68 | to worship anyth
02B510: 69 6E 67 20 69 20 63 61  6E 20 74 61 73 74 65 2E | ing i can taste.
02B520: 20 2F 0A 0A 0A E2 80 99  73 20 6D 6F 75 74 68 20 |  /...’s mouth
02B530: 6E 65 63 65 73 73 61 72  79 2E 20 2F 20 73 73 73 | necessary. / sss
02B540: 73 73 73 62 6C 65 73 73  20 74 68 61 74 20 6D 61 | sssbless that ma
02B550: 6E E2 80 99 73 20 6D 6F  75 74 68 2C 20 74 68 65 | n’s mouth, the
02B560: 20 73 6F 6E 67 20 77 65  20 73 77 61 79 20 73 6C |  song we sway sl
02B570: 6F 70 70 79 20 74 6F 2C  20 74 68 65 20 62 65 61 | oppy to, the bea
02B580: 74 2C 20 74 68 65 20 62  72 69 64 67 65 2C 20 74 | t, the bridge, t
02B590: 68 65 20 6C 65 6E 67 74  68 20 2F 20 6F 66 20 68 | he length / of h
02B5A0: 69 73 20 68 61 6E 64 20  6F 6E 20 6D 79 20 74 68 | is hand on my th
02B5B0: 69 67 68 20 26 20 62 61  63 6B 20 26 20 69 20 6B | igh & back & i k
02B5C0: 6E 6F 77 20 6E 6F 74 20  77 68 69 63 68 20 63 6F | now not which co
02B5D0: 75 6E 74 72 79 20 69 20  61 6D 20 6F 66 2E 20 2F | untry i am of. /
02B5E0: 20 69 20 77 61 6E 74 20  74 6F 20 6C 69 76 65 20 |  i want to live
02B5F0: 6F 6E 20 68 69 73 20 74  6F 6E 67 75 65 2C 20 62 | on his tongue, b
```

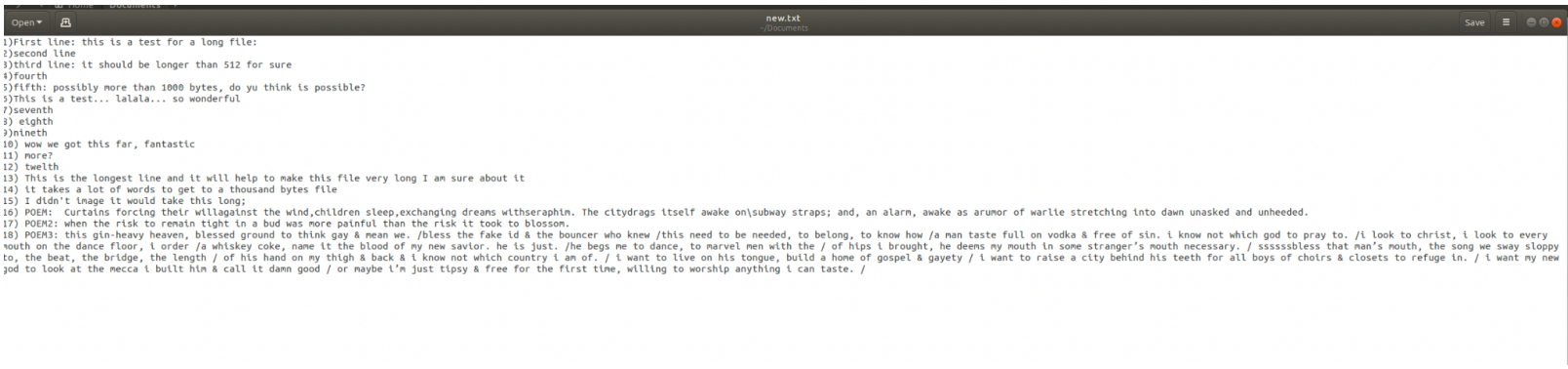
```
02B600: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B610: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B620: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B630: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B640: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B650: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B660: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B670: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B680: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B690: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B6A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B6B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B6C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B6D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B6E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B6F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

02B700: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B710: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B720: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B730: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B740: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B750: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B760: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B770: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

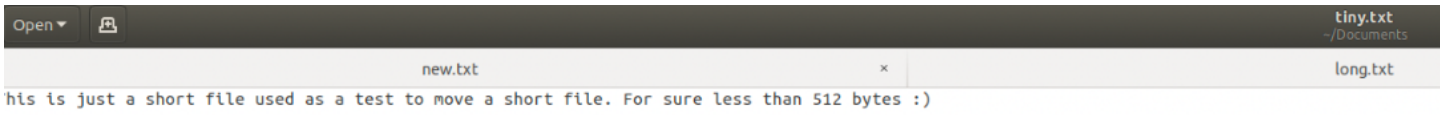
```
02B780: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B790: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B7A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B7B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B7C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B7D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B7E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
02B7F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

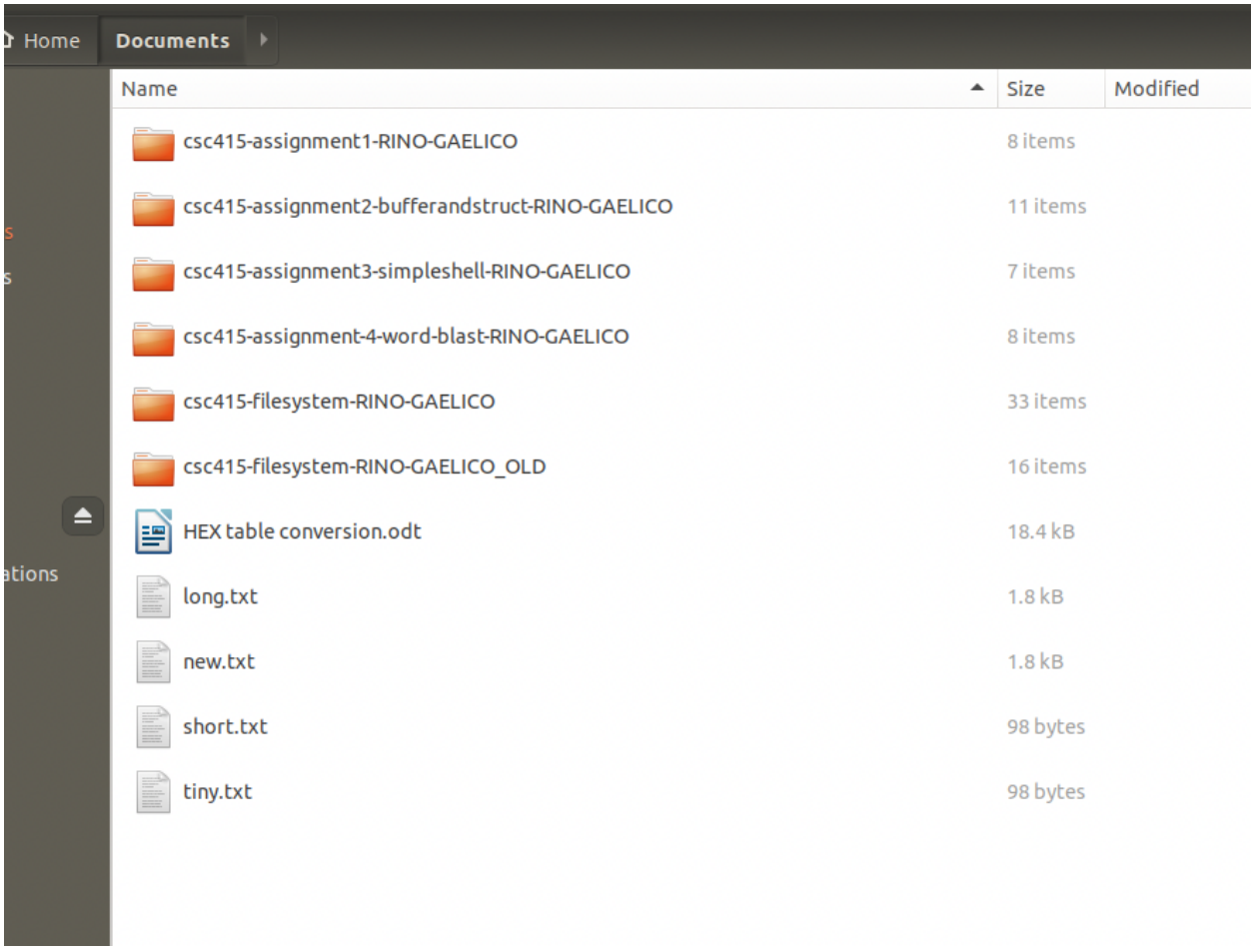
## cp2l Command Working

Here we used cp2l to copy back the short file and the long file into linux with different names.

```
Prompt > cd Home
Prompt > ls

long.txt
short.txt
Prompt > cp2l /Home/long.txt /home/student/Documents/new.txt
Prompt > cp2l /Home/short.txt /home/student/Documents/tiny.txt
Prompt > ls

long.txt
short.txt
Prompt >
```




Open
tiny.txt
~/Documents
new.txt
long.txt
his is just a short file used as a test to move a short file. For sure less than 512 bytes :)


Home
Documents
Name
Size
Modified
csc415-assignment1-RINO-GAELICO
8 items
csc415-assignment2-bufferandstruct-RINO-GAELICO
11 items
csc415-assignment3-simpleshell-RINO-GAELICO
7 items
csc415-assignment-4-word-blast-RINO-GAELICO
8 items
csc415-filesystem-RINO-GAELICO
33 items
csc415-filesystem-RINO-GAELICO_OLD
16 items
HEX table conversion.odt
18.4 kB
long.txt
1.8 kB
new.txt
1.8 kB
short.txt
98 bytes
tiny.txt
98 bytes

**mv Command Working**

Here we used the mv command to move the file  new.txt from root to Home directory .

```
short.txt
Prompt > cd ..
Prompt > ls

Home
new.txt
Prompt > mv new.txt /Home/new.txt
Prompt > ls

Home
Prompt > cd Home
Prompt > ls

long.txt
short.txt
new.txt
Prompt > exit
System exiting
student@student-VirtualBox:~/Documents/csc415-filesystem-RINO-GAELICO$
```

**cp Command Working**

Here we used cp command to copy a file called long.txt to a file called new.txt inside the root directory.

```
student@student-VirtualBox:~/Documents/csc415-filesystem-RINO-GAELICO$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
FATSIZE IN BLOCKS: 150
FAT START: 32
CLUSTER IN BLOCKS: 1
DATA START: 332
Number of Blocks in Volume: 19531
BLOCK SIZE, 512
Prompt > ls

Home
Prompt > cd Home
Prompt > ls

long.txt
short.txt
Prompt > cp long.txt /new.txt
Prompt > ls

long.txt
short.txt
Prompt > cd ..
Prompt > ls

Home
new.txt
Prompt >
```

Work Cited

"Microsoft Extensible Firmware Initiative FAT32 File System Specification." *Microsoft Hardware White Paper*, Version 1.03, Microsoft Corporation, 6 Dec. 2000, www.cs.fsu.edu/~cop4610t/assignments/project3/spec/fatspec.pdf