

Operating System Principles CSC 415

Project Name

File System

Team Details / Section 01

Team Name: MJ's

Madina Ahmadzai (ID# : 921835158)

Muhammed Nafees (ID# : 921941329)

Janelle Lara (ID# : 920156598)

John Santiago (ID# : 909606963)

GitHub Name: aktaails

Table of Contents

Github Repository.....	3
Project Introduction.....	3
File System Implementation	
fslnit.h.....	3
mfs.h.....	9
b_io.h.....	14
fsshell.c.....	16
Issues and Resolutions.....	16
Screenshots of Working Shell Commands.....	18

File System Documentation

GitHub Repository

<https://github.com/CSC415-2022-Spring/csc415-filessystem-aktails>

Project Introduction

For this project we were tasked with making our own file system. Although it does not include all components of the FAT32 specifications, the file system we created is FAT32 compliant. We created a volume that will represent our file system which can also run alongside the file system built into the Linux OS. Some features that we implemented in our file system are the ability to make directories, delete files/directories, change directories, move files between directories, copy files to different directories, and finally copy files to our file system from the Linux file system and vice versa. The size of our file system volume is 5.12MB, 10000 blocks of 512 byte sectors.

File System Implementation

In this section we'll be going over each header file's functions and data structures.

fsinit.h

Data Structures

VCB - Holds important information about our file system.

Field Name	Offset	Size (Bytes)	Description
jmpBoot	0	3	Jump instruction to boot code. Not needed for our file system but included for FAT32 compliance.
OEMName	3	8	String indicator of what system formatted the volume.
BytesPerSector	11	2	Count of bytes per sector.
SectorPerCluster	13	1	Number of sectors per allocation unit (cluster).

RsvdSectorCount	14	2	Count of reserved sectors of the volume.
NumOfFATs	16	1	Count of FAT data structures on the volume.
RootEntryCount	17	2	For FAT12 and FAT16 volumes. FAT32 volumes have this value set to 0.
TotalSectors16	19	2	For FAT12 and FAT16 volumes. FAT32 volumes have this value set to 0.
Media	21	1	0xF8 is used for “fixed” media. 0xF0 for removable media.
FATSz16	22	2	For FAT12 and FAT16 volumes. FAT32 volumes have this value set to 0.
SectorsPerTrack	24	2	Not sure if this is used for our file system so we just set it to 0.
NumberOfHeads	26	2	Not sure if this is used for our file system so we just set it to 0.
HiddenSectors	28	4	Not sure if this is used for our file system so we just set it to 0.
TotalSectors32	32	4	Count of all Sectors on the volume.
FATSz32	36	4	Count of sectors occupied by one FAT data structure.
ExtFlags	40	2	Bits 0-3: Active FAT starting from 0. Only valid when bit7 is 1. Bits 4-6: Reserved. Bit 7: 0 means that the FAT is mirrored into all FATs at runtime. 1 means that only one FAT indicated by bits 0-3 is active.

			Bits 8-15: Reserved.
FSVer	42	2	Version number of FAT32 volume.
RootCluster	44	4	Set to the cluster number of the first cluster of the root directory or the first usable cluster available.
FSInfo	48	2	Sector number of FSInfo data structure in the volume.
BkBootSector	50	2	If non-zero, indicates the sector number of the volume that holds a copy of the boot record.
Reserved0	52	12	Reserved.
DriverNumber	64	1	Usually set to 0x00 for floppy disks and 0x80 for hard disks. Operating system specific.
Reserved1	65	1	Reserved.
BootSig	66	1	Extended boot signature.
VolumID	67	4	Volume serial number.
VolumeLabel	71	11	Volume label/name.
FileSystemType	82	8	Informational string that does not determine FAT type.
Reserved2	90	420	Reserved.
Signature	510	2	Signature for FAT32 Volume. Byte 510 = 0x55, Byte 511 = 0xAA

FSInfo - Keeps track of how much free space our file system volume currently has. Also keeps track of the first free block of the volume.

Field Name	Offset	Size (Bytes)	Description
FSI_LeadSig	0	4	Lead signature used to validate start of FSInfo structure in sector. Value set to 0x41615252
FSI_Reserved1	4	480	Reserved Space, all set to 0
FSI_StrucSig	484	4	Additional signature used to validate the integrity of FSInfo structure. Value set to 0x61417272.
FSI_Free_Count	488	4	Last known free cluster block on the volume. Free count isn't known if its value is 0xFFFFFFFF. Contents must be validated at volume mount.
FSI_Nxt_Free	492	4	Cluster number of next available cluster on the volume.
FSI_Reserved2	496	12	Reserved Space, all set to 0
FSI_TrailSig	508	4	Trail signature used to validate the integrity of data in the sector with FSInfo structure. Value is set to 0xAA550000

fsFat - Holds the File Allocation Table (FAT). The FAT is represented as an array where each element of the array represents a block of the file system volume.

Field Name	Description
fat[10000]	FAT array of 10000 blocks. -Free blocks are marked as 0x00000000 -Reserved blocks are marked as 0x00000001 -Allocated blocks are marked as

	0x00000002 - 0x0000270F (9999). The value that the element holds references the next block of the file. -Allocated blocks that are also the last block of a file are marked as 0xFFFFFFFF.
startingBlock	The starting block of the FAT in the file system volume.

DirectoryEntry - This is a FAT32 compliant directory entry that stores all the information of a file for a Directory.

Field Name	Offset	Size (Bytes)	Description
DIR_Name	0	11	File name which is only limited to 11 characters.
DIR_Attr	11	1	The 5th bit is set when the file is a directory.
DIR_NTRes	12	1	This is reserved and needs to be set to 0.
DIR_CrtTimeTenth	13	1	Optional. Set to 0.
DIR_CrtTime	14	2	Optional. Set to 0.
DIR_CrtDate	16	2	Optional. Set to 0.
DIR_LstAccDate	18	2	Optional. Set to 0.
DIR_FstClusHI	20	2	This is the high word of the first data cluster that is described by this entry.

DIR_WrtTime	22	2	This is the last modification (write) time.
DIR_WrtDate	24	2	This is the last modification (write) date.
DIR_FstClusLO	26	2	The low word of the first data cluster number for the file/directory with this entry.
DIR_FileSize	28	4	32-bit quantity which contains the size of the file/directory described in bytes.

Directory - This structure represents a directory, which is just an array of directory entry structures.

Field Name	Description
DirectoryEntry Directory[16]	Each directory in our file system can hold a maximum of 16 directory entries.

Global Variables

All of these variables can be readily accessed by any of the source files included in this project. It saved me the headache of trying to pass and keep track of multiple pointers between source files.

extern struct VCB * VCBptr - Pointer to keep the VCB in memory.

extern struct FSInfo * FSInfo - Pointer to keep FSInfo struct in memory.

extern struct fsFat *FATptr1, *FATptr2 - Pointers to both FAT tables.

extern struct Directory * ROOTptr - Pointer to the Root Directory of our file system.

extern struct Directory * cwdptr - Pointer to the current working directory.

extern struct Directory * searchDir - Pointer to the last searched directory.

Functions

initVCB - Creates and initializes the Volume Control Block of our file system. It also writes the VCB to the first block of our volume.

Parameters: uint64_t numberOfBlocks, uint64_t blockSize

Return Value: 0

initFAT - Creates and initializes our free space management system. The FSInfo struct and both FAT tables are initialized and written to our file system volume.

Parameters: uint64_t numberOfBlocks, uint64_t blockSize

Return Value: 0

initRootDirectory - Creates and initializes the root directory of our file system. Pointers ROOTptr, cwdptr, and searchDir all point to the root directory upon initialization. The root directory is written to disk and the free space management system is updated.

Parameters: N/A

Return Value: 0

getCurrentTime - This is a simple function getting the current local time when files are created or modified. Return value is given in a way that complies with the FAT32 directory entry specification.

Parameters: N/A

Return Value: uint16_t

getCurrentDate - This is a simple function getting the current local time when files are created or modified. Return value is given in a way that complies with the FAT32 directory entry specification.

Parameters: N/A

Return Value: uint16_t

mfs.h

Data Structures

fs_diriteminfo - This was mainly used for when the user wanted a list of all the files in a directory. Holds any important information needed for the opened directory.

Field Name	Description
unsigned short d_reclen	This variable was unused but was kept just in case it would be of use somewhere in the future.
unsigned char fileType	Didn't find much use for this since we could get this information by checking the DIR_Attr field in the directory entry.
char d_name[256]	This was used to save the name of the current directory which was taken from its parent directory.

fdDir - This structure stores information that opendir and readdir need to use in order to display the information of the open directory's directory entries to the user.

Field Name	Description
unsigned short d_reclen	Same as above.
unsigned short dirEntryPosition	This field allows the function fs_readdir to iterate and read through all the directory entries in the opened directory.
UInt64_t directoryStartLocation	This was used to save the starting block of the directory that was just opened.

fs_stat - This struct was used to hold information about the directory entries.

Field Name	Description
off_t st_size	Size of file in bytes.
blksize_t st_blksize	Didn't really see the use for this but

	kept it just in case it might be useful later.
blkcnt_t st_blocks	Same as above.
time_t st_accesstime	We initially decided not to use this variable but kept it on the off chance that we changed our minds.
time_t st_modtime	Same as above.
time_t st_createtime	Same as above.

Global Variables

extern char lastFileName[11] - This was added in to help keep track of the name of the last file search. This was especially useful when creating or deleting new files.

Functions

fs_mkdir - This is the primary function of creating a new directory. It first validates the filename's path, making sure the parent directory exists and the directory itself doesn't already exist. It checks if the parent directory has an available directory entry slot open, then it requests blocks to be allocated. If there is space available, the necessary information is added as a new directory entry for the parent directory and the new directory is created and initialized. Both directories are then written to disk.

Parameters: const char *pathname, mode_t mode

Return Value: 0 if successful, 1 if unsuccessful

fs_rmdir - This function deletes a directory that is empty, i.e. the only directory entries are "." and "..". The function first checks that the directory to be deleted is NOT the root directory, then it checks that the directory is empty. The parent directory is then read from disk so that the directory entry for the directory to be deleted is marked as free, then written back to disk. The starting block for the directory to be deleted is sent to the free space management system which will then free up the space on disk.

Parameters: const char *pathname

Return Value: 0

fs_opendir - This function opens a directory and creates a pointer to a structure that is initialized and populated with the necessary information. It sets a flag that slightly modifies the function of fs_isDir, making sure it doesn't free up memory that will be needed for future function calls. The return value is the pointer to the structure.

Parameters: const char *name

Return Value: fdDir*

fs_readdir - This function iterates and reads through each directory entry in the open directory.

Parameters: fdDir *dirp

Return Value: struct fs_direntinfo*

fs_closedir - This function resets the flag that opendir set and frees any memory that was used to store the directory.

Parameters: fdDir *dirp

Return Value: 0

fs_getcwd - Returns the current working directory as a string.

Parameters: char *buf, size_t size

Return Value: char*

fs_setcwd - Sets the current working directory to the parameter "buf".

Parameters: char *buf

Return Value: 0

fs_isFile - First checks if the pathname is valid, then checks the directory entry's DIR_Attr field to see if it is a file.

Parameters: char * path

Return Value: 1 if file, 0 otherwise

fs_isDir - First checks if the pathname is valid, then checks the directory entry's DIR_Attr field to see if it is a directory.

Parameters: char * path

Return Value: 1 if directory, 0 otherwise

fs_delete - Reads the parent directory from disk, gets the file information that is necessary for deletion, updates the directory entry corresponding to the file to be deleted as free, then gets written back to disk. File starting

block is sent to the free space management system to be freed up on the disk.

Parameters: char* filename

Return Value: 0

fs_stat - Sets the filesize of the given directory entry that is about to be displayed to the user.

Parameters: const char *path, struct fs_stat *buf

Return Value: 0

isValidPath - This function checks whether or not the given pathname to a file is valid. Works for both relative and absolute pathnames. Unfortunately, this does not work properly for chained instances of "."s and ".."s.

Parameters: const char * path, int full

Return Value: 0 if successful, 1 otherwise

resetSearch - This function frees up memory being used to hold a pointer to the directory that was most recently searched.

Parameters: N/A

Return Value: N/A

getStartingBlock - This is a helper function to get the starting block of a file given the directory entry of the last searched directory.

Parameters: int DE

Return Value: startingBlock

allocateBlocks - This function is the part of our free space management that checks for available space on the disk, and allocates memory for the caller. The return value is the first available free block on disk.

Parameters: uint64_t fileSize

Return Value: nextFreeCluster

releaseBlocks - This function is the part of our free space management system that marks blocks as free. Does not necessarily delete the data off disk, but instead allows it to be overwritten.

Parameters: uint64_t startingBlock, uint64_t fileSize

Return Value: N/A

b_io.h

Data Structures

b_fcb - Holds information about a specific file opened from our file system.

Field Name	Description
char fileName[11]	Name of file - maximum of 11 characters
int flag	Used to determine how the file will be used, i.e. O_RDONLY, WR_ONLY, etc.
struct Directory * parentDirectory	A pointer to the directory which the file is currently in. Allows access to file's attributes.
int fileStartingBlock	The starting block of the file's location in our file system's volume.
int fileSize	Size of the file in bytes.
int blockCount	Amount of blocks that the file occupies on the volume.
char * buf	Buffer that holds the character bytes of the file to be read from or to be written to disk.
int index	Keeps track of the current byte position of the character buffer.
int buflen	Keeps track of how many valid bytes are in the character buffer

Global Variables

typedef int b_io_fd - This represents a file descriptor which references a file control block in an array.

char copyFileName[11] - This variable was used to store the file name of the Linux file being copied into our file system.

b_fcb fcbArray[MaxFCBS] - This is an array of all the file control blocks which contain information about files that are currently opened in our file system. One file control block per file.

int startup - This indicates whether the array of file control blocks has already been initialized.

Functions

b_init - Initializes the array of file control blocks for the file system.

Parameters: N/A

Return Value: N/A

b_getFCB - Searches for a free element in the array of file control blocks.

Parameters: N/A

Return Value: b_io_fd

b_open - Opens a file from our file system. Checks the path of the filename parameter to see if it exists. Reads the file from the volume/disk and populates the b_fcb structure with the necessary information in preparation for future read or write commands.

Parameters: char * filename, int flags

Return Value: b_io_fd

b_read - Reads amount of bytes specified by the parameter "count" from the buffer in the file control block referenced by the parameter "fd" into the parameter "buffer". The amount of bytes read is set as the return value.

Parameters: b_io_fd fd, char * buffer, int count

Return Value: bytesRead

b_write - Writes amount of bytes specified by the parameter "count" from the parameter "buffer" into the file control block's buffer in preparation for writing the file to disk. The amount of bytes written is set as the return value

Parameters: b_io_fd fd, char * buffer, int count

Return Value: count

b_seek - Sets the index (offset) of the byte location for the file control block's buffer.

Parameters: b_io_fd fd, off_t offset, int whence

Return Value: newOffset

b_close - Allocates and writes the file to disk. Also updates the parent directory of the associated file. Clears the file control block in the array referenced by parameter “fd”.

Parameters: b_io_fd fd

Return Value: N/A

fshell.c

Most of the functions in this source file were done for us, but I implemented cmd_mv and a helper function to grab file names being copied into our file system volume.

cmd_mv - This function moves a file in a directory to another directory within our file system. It removes the directory entry for the file in the original parent directory, then adds a new directory entry to the destination directory. Both directories are rewritten to disk.

Parameters: int argcnt, char *argvec[]

Return Value: 0

setFileName - This is a helper function to get the file name of a Linux file system file while copying it into our file system.

Parameters: char* path

Return Value: N/A

Issues and Resolutions

General Issues

There are memory leaks. Not something I’m proud of admitting but there definitely are a few of them existing throughout the project. I know that it’s good practice to always use a free() whenever you’re using malloc(), but I would sometimes get a “double free” error once in a while. Whenever a function tries to check if a pathname is valid, like making a new directory, sometimes, but not always, I would get the “double free” error and since it wasn’t happening consistently, it was hard to pinpoint the problem. I did however notice that it occurred fairly often whenever the root directory was involved, so my solution was to check whether the pointer I was freeing was pointing to the root directory global pointer, at which point I would not free() it and just set it to NULL instead. This solution did patch up the holes for a while, but as the project went on, it proved to not be **the** solution.

b_io.h

The functions associated with this header file weren't too difficult to deal with. The three part diagram of the read function was a bit confusing to understand but the way I coded it seemed to work out fine for all the shell commands that involved movement of a file. I'm unsure whether this is related to the way I implemented the read function, but I was unable to get the cp2l shell command to work and I'm at a loss as to where to start fixing that problem.

I'm not sure if this classifies as an issue, but I thought it would be a bit too complicated to write to disk after each b_write call considering how often the free space system would need to be called to allocate space and the process of updating and writing both the parent directory and file to disk. I'm not entirely sure this is correct but to remedy this, I thought it would be better if we did a single allocation and write for new files in the close function, cutting down the total amount of writes needed for a single shell command.

fshell.c

I was having trouble finding a way to get the file name of the linux system file when using cp2fs. I would either show up on the hexdump without a name or have a name full of garbage. I created a helper function to get the last part of the string of the second command line argument, and this seemed to do the job well enough. For example, if the file I'm copying from the linux file system to our file system is /home/student/Desktop/testfile, the helper function would be able to pull "testfile" from that path and properly name the file in our file system.

mfs.h

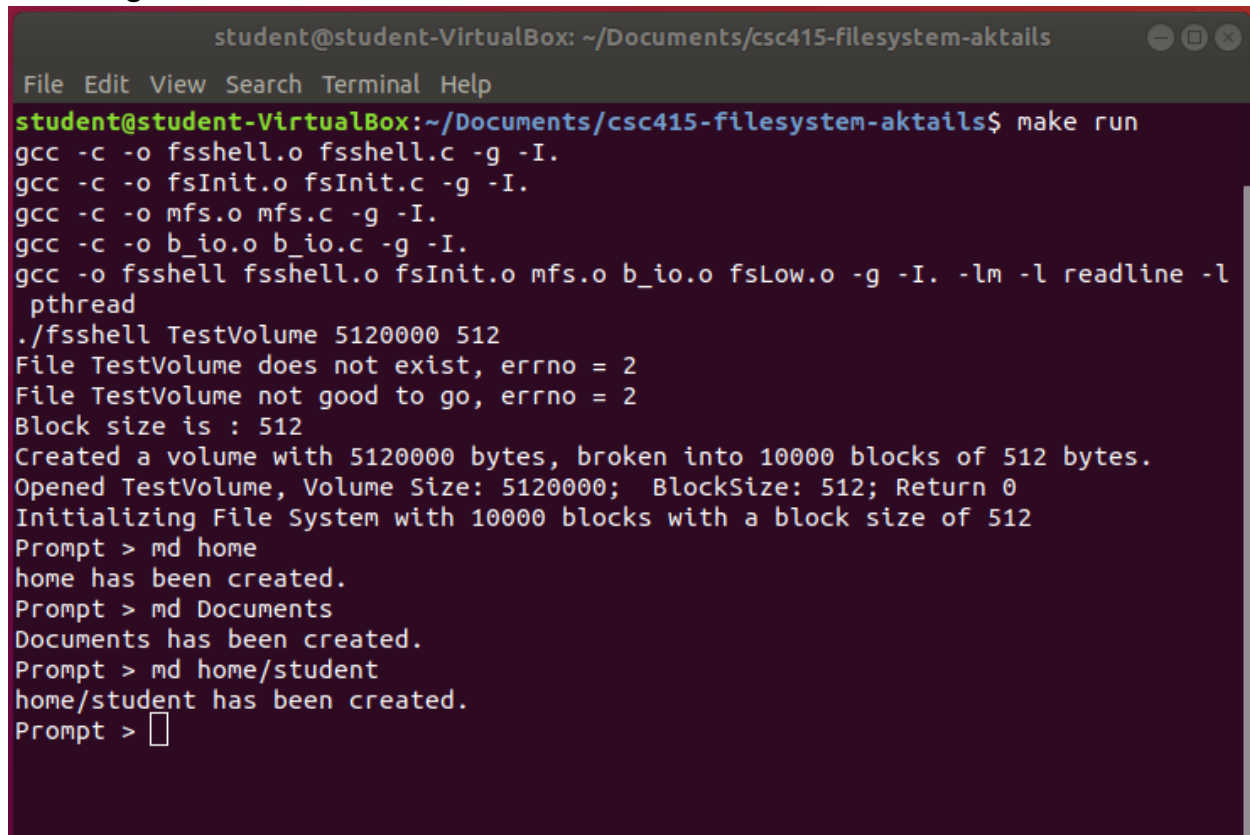
The main issue here was our method of parsing pathnames. I kind of threw one together quickly in order to test other functions and just made adjustments along the way when I encountered any issue. Unfortunately, I realized after I finished implementing all the functions that I wasn't able to correctly parse chained instances of "." and "..". I tried drawing up a new plan for it but realized that it would also require me to change all the other functions that relied on the way it was originally coded. In the end, I made a note of it and decided to come back to it later but I just wasn't able to find the time to do *everything*.

fsInit.h

There weren't too many issues here. Once we realized that we needed to make a file system that was compliant with the FAT32 system rather than just creating a file system similar to the FAT32 system from scratch, using the FAT32 specifications gave us a lot more guidance on where to start and how most things should look. The main challenge was getting everyone's code to work with each other. This probably goes for the others as well, but this was my first time working on the project with other people. We all had different coding styles, whether it be naming convention, indentation preference, or overall design, it was definitely a new and difficult situation to deal with at first.

Screenshots of Working Shell Commands**cmd_ls**

Creating a few directories:



```

student@student-VirtualBox: ~/Documents/csc415-filesystem-aktails
File Edit View Search Terminal Help
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ make run
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -c -o fsInit.o fsInit.c -g -I.
gcc -c -o mfs.o mfs.c -g -I.
gcc -c -o b_io.o b_io.c -g -I.
gcc -o fsshell fsshell.o fsInit.o mfs.o b_io.o fsLow.o -g -I. -lm -l readline -l
pthread
./fsshell TestVolume 5120000 512
File TestVolume does not exist, errno = 2
File TestVolume not good to go, errno = 2
Block size is : 512
Created a volume with 5120000 bytes, broken into 10000 blocks of 512 bytes.
Opened TestVolume, Volume Size: 5120000; BlockSize: 512; Return 0
Initializing File System with 10000 blocks with a block size of 512
Prompt > md home
home has been created.
Prompt > md Documents
Documents has been created.
Prompt > md home/student
home/student has been created.
Prompt > 

```

Hexdump of Root Directory, home, Documents, and home/student

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 161
Dumping file TestVolume, starting at block 161 for 1 block:
```

```
014200: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014210: 00 00 00 00 00 00 B8 93 9B 54 A0 00 00 02 00 00 | .....T.....
014220: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014230: 00 00 00 00 00 00 71 93 9B 54 A0 00 00 02 00 00 | .....qT.....
014240: 68 6F 6D 65 00 00 00 00 00 00 00 10 00 00 00 00 | home.....
014250: 00 00 00 00 00 00 AB 93 9B 54 A1 00 00 02 00 00 | .....T.....
014260: 44 6F 63 75 6D 65 6E 74 73 00 00 10 00 00 00 00 | Documents.....
014270: 00 00 00 00 00 00 B8 93 9B 54 A2 00 00 02 00 00 | .....T.....
014280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 162
Dumping file TestVolume, starting at block 162 for 1 block:
```

```
014400: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014410: 00 00 00 00 00 00 C2 93 9B 54 A1 00 00 02 00 00 | .....T.....
014420: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014430: 00 00 00 00 00 00 AB 93 9B 54 A0 00 00 02 00 00 | .....T.....
014440: 73 74 75 64 65 6E 74 00 00 00 00 10 00 00 00 00 | student.....
014450: 00 00 00 00 00 00 C2 93 9B 54 A3 00 00 02 00 00 | .....T.....
014460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014470: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014480: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014490: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 163
Dumping file TestVolume, starting at block 163 for 1 block:
```

```
014600: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014610: 00 00 00 00 00 00 B8 93 9B 54 A2 00 00 02 00 00 | .....T.....
014620: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014630: 00 00 00 00 00 00 B8 93 9B 54 A0 00 00 02 00 00 | .....T.....
014640: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014650: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014680: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014690: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 164
Dumping file TestVolume, starting at block 164 for 1 block:
```

```
014800: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014810: 00 00 00 00 00 00 C2 93 9B 54 A3 00 00 02 00 00 | .....T.....
014820: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014830: 00 00 00 00 00 00 C2 93 9B 54 A1 00 00 02 00 00 | .....T.....
014840: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014850: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

cmd_cp2fs

Copying a small section of the file used in Assignment 4 to our file system

```
Prompt > md home
home has been created.
Prompt > md Documents
Documents has been created.
Prompt > md home/student
home/student has been created.
Prompt > cp2fs /home/student/Desktop/file /home/student
Prompt > 
```

```
student@student-VirtualBox: ~/Documents/csc415-filesystem-aktails
File Edit View Search Terminal Help
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 165
Dumping file TestVolume, starting at block 165 for 1 block:

014A00: EF BB BF 0A 54 68 65 20 50 72 6F 6A 65 63 74 20 | .The Project
014A10: 47 75 74 65 6E 62 65 72 67 20 45 42 6F 6F 6B 20 | Gutenberg EBook
014A20: 6F 66 20 57 61 72 20 61 6E 64 20 50 65 61 63 65 | of War and Peace
014A30: 2C 20 62 79 20 4C 65 6F 20 54 6F 6C 73 74 6F 79 | , by Leo Tolstoy
014A40: 0A 0A 54 68 69 73 20 65 42 6F 6F 6B 20 69 73 20 | ..This eBook is
014A50: 66 6F 72 20 74 68 65 20 75 73 65 20 6F 66 20 61 | for the use of a
014A60: 6E 79 6F 6E 65 20 61 6E 79 77 68 65 72 65 20 61 | nyone anywhere a
014A70: 74 20 6E 6F 20 63 6F 73 74 20 61 6E 64 20 77 69 | t no cost and wi
014A80: 74 68 20 61 6C 6D 6F 73 74 0A 6E 6F 20 72 65 73 | th almost.no res
014A90: 74 72 69 63 74 69 6F 6E 73 20 77 68 61 74 73 6F | trictions whatso
014AA0: 65 76 65 72 2E 20 59 6F 75 20 6D 61 79 20 63 6F | ever. You may co
014AB0: 70 79 20 69 74 2C 20 67 69 76 65 20 69 74 20 61 | py it, give it a
014AC0: 77 61 79 20 6F 72 20 72 65 2D 75 73 65 0A 69 74 | way or re-use.it
014AD0: 20 75 6E 64 65 72 20 74 68 65 20 74 65 72 6D 73 | under the terms
014AE0: 20 6F 66 20 74 68 65 20 50 72 6F 6A 65 63 74 20 | of the Project
014AF0: 47 75 74 65 6E 62 65 72 67 20 4C 69 63 65 6E 73 | Gutenberg Licens
```

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 164
Dumping file TestVolume, starting at block 164 for 1 block:

014800: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014810: 00 00 00 00 00 00 0C 97 9B 54 A3 00 00 02 00 00 | .....T.....
014820: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014830: 00 00 00 00 00 00 00 97 9B 54 A1 00 00 02 00 00 | .....T.....
014840: 66 69 6C 65 00 00 00 00 00 00 00 00 00 00 00 00 | file.....
014850: 00 00 00 00 00 00 0C 97 9B 54 A4 00 A3 02 00 00 | .....T.....
014860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```


cmd_cd and cmd_pwd

Moving between directories

```
Prompt > md home/student
home/student has been created.
Prompt > cp2fs /home/student/Desktop/file /home/student
Prompt > pwd
/
Prompt > cd home
Prompt > pwd
/home
Prompt > cd student
Prompt > pwd
/home/student
Prompt > 
```

cmd_ls

Listing directory entries, ls including -a, -l, and -la

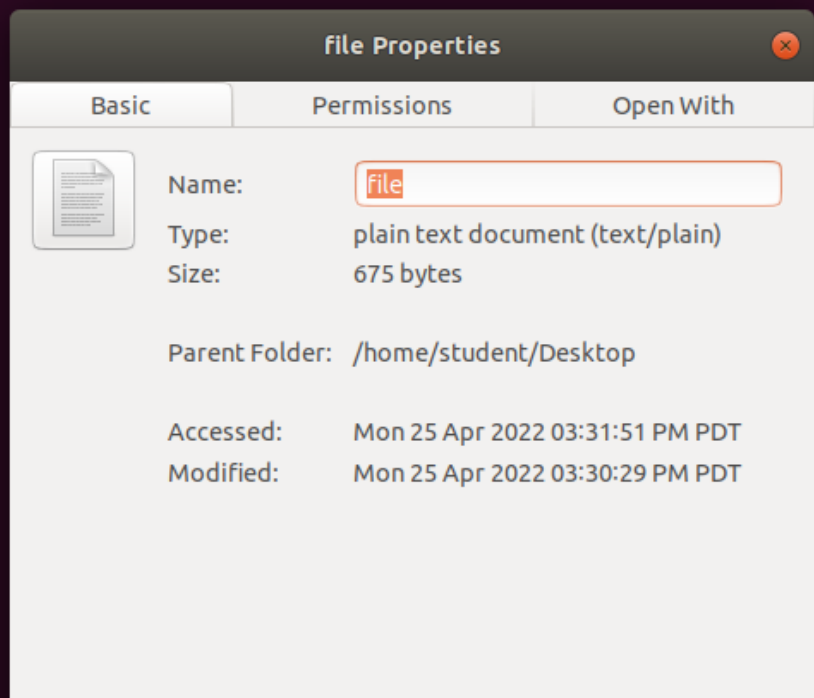
```
Prompt > cd home
Prompt > pwd
/home
Prompt > cd student
Prompt > pwd
/home/student
Prompt > ls

file
Prompt > ls -a

.
..
file
Prompt > ls -l

-          675  file
Prompt > ls -la

D          512  .
D          512  ..
-          675  file
Prompt > 
```



cmd_mv

Moving file from /home/student to /Documents

```
Prompt > ls -la
D          512  .
D          512  ..
-          675  file
Prompt > mv file /Documents
Prompt > ls

Prompt > pwd
/home/student
Prompt > ls

Prompt > cd /Documents
Prompt > ls

file
Prompt > █
```

In our file system, setting the first byte of the directory entry is equivalent to being “free”.

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 164
Dumping file TestVolume, starting at block 164 for 1 block:

014800: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014810: 00 00 00 00 00 00 0B A5 9B 54 A3 00 00 02 00 00 | .....T.....
014820: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014830: 00 00 00 00 00 00 4A 99 9B 54 A1 00 00 02 00 00 | .....J.....
014840: 00 69 6C 65 00 00 00 00 00 00 00 00 00 00 00 00 | .ile.....
014850: 00 00 00 00 00 00 65 99 9B 54 A4 00 A3 02 00 00 | .....e.....
014860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

014900: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014910: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014920: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

You can see that file moved to the /Documents directory

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 163
Dumping file TestVolume, starting at block 163 for 1 block:

014600: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014610: 00 00 00 00 00 00 0B A5 9B 54 A2 00 00 02 00 00 | .....T.....
014620: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014630: 00 00 00 00 00 00 48 99 9B 54 A0 00 00 02 00 00 | .....HoT.....
014640: 66 69 6C 65 00 00 00 00 00 00 00 00 00 00 00 00 | file.....
014650: 00 00 00 00 00 00 65 99 9B 54 A4 00 A3 02 00 00 | .....eT.....
014660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014680: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014690: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```


Copying another file into the directory will overwrite all the bytes left over from the previous removed file.

```
Prompt > cd /Documents
Prompt > ls

file
Prompt > cp2fs /home/student/Desktop/newfile /home/student
Prompt > pwd
/Documents
Prompt > cd ..
Prompt > pwd
/
Prompt > cd home/student
Prompt > ls

newfile
Prompt > ls -la

D          512  .
D          512  ..
-           38  newfile
Prompt > 
```

Open ▾  nothing of importance is in this file


```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 164
Dumping file TestVolume, starting at block 164 for 1 block:
```

```
014800: 2E 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014810: 00 00 00 00 00 00 00 17 B1 9B 54 A3 00 00 02 00 00 | .....T.....
014820: 2E 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014830: 00 00 00 00 00 00 00 4A 99 9B 54 A1 00 00 02 00 00 | .....J.....
014840: 6E 65 77 66 69 6C 65 00 00 00 00 00 00 00 00 00 00 | newfile.....
014850: 00 00 00 00 00 00 00 17 B1 9B 54 A6 00 26 00 00 00 00 | .....T.&...
014860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 167
Dumping file TestVolume, starting at block 167 for 1 block:
```

```
014E00: 6E 6F 74 68 69 6E 67 20 6F 66 20 69 6D 70 6F 72 | nothing of impor
014E10: 74 61 6E 63 65 20 69 73 20 69 6E 20 74 68 69 73 | tance is in this
014E20: 20 66 69 6C 65 0A 00 00 00 00 00 00 00 00 00 00 | file.....
014E30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014E40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014E50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014E60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014E70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014E80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014E90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014EA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014EB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014EC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014ED0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014EE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014EF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

cmd_rm

Removing a file

```
Prompt > cd home/student
Prompt > ls

newfile
Prompt > ls -la

D          512  .
D          512  ..
-           38  newfile
Prompt > rm newfile
Prompt > ls

Prompt > 
```

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 164
Dumping file TestVolume, starting at block 164 for 1 block:

014800: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014810: 00 00 00 00 00 00 FA B2 9B 54 A3 00 00 02 00 00 | .....T.....
014820: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014830: 00 00 00 00 00 00 00 4A 99 9B 54 A1 00 00 02 00 00 | .....J.....
014840: 00 65 77 66 69 6C 65 00 00 00 00 00 00 00 00 00 | .ewfile.....
014850: 00 00 00 00 00 00 EA B2 9B 54 A7 00 26 00 00 00 | .....T.&...
014860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

Removing a directory

```
Prompt > ls

Prompt > cd ..
Prompt > pwd
/home
Prompt > cd ..
Prompt > pwd
/
Prompt > ls

home
Documents
Prompt > rm home
Unable to delete directory home - Directory must be empty.
Prompt > ls home

student
Prompt > rm home/student
home/student deleted.
Prompt > ls home

Prompt > rm home
home deleted.
Prompt > ls

Documents
Prompt > █
```

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 162
Dumping file TestVolume, starting at block 162 for 1 block:

014400: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014410: 00 00 00 00 00 00 CC BA 9B 54 A1 00 00 02 00 00 | .....T.....
014420: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014430: 00 00 00 00 00 00 76 B8 9B 54 A0 00 00 02 00 00 | .....vT.....
014440: 00 74 75 64 65 6E 74 00 00 00 00 10 00 00 00 00 | .tudent.....
014450: 00 00 00 00 00 00 7C B8 9B 54 A3 00 00 02 00 00 | .....|T.....
014460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014470: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014480: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014490: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0144F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```

student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 161
Dumping file TestVolume, starting at block 161 for 1 block:

014200: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014210: 00 00 00 00 00 00 00 D0 BA 9B 54 A0 00 00 02 00 00 | .....KTT.....
014220: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014230: 00 00 00 00 00 00 00 74 B8 9B 54 A0 00 00 02 00 00 | .....tTT.....
014240: 00 6F 6D 65 00 00 00 00 00 00 00 10 00 00 00 00 | .ome.....
014250: 00 00 00 00 00 00 00 76 B8 9B 54 A1 00 00 02 00 00 | .....vTT.....
014260: 44 6F 63 75 6D 65 6E 74 73 00 00 10 00 00 00 00 | Documents.....
014270: 00 00 00 00 00 00 00 78 B8 9B 54 A2 00 00 02 00 00 | .....xTT.....
014280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0142F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

cmd_cp

Copying a file to another directory in the file system

```

Prompt > cp2fs /home/student/Desktop/file /home/student
Prompt > ls

home
Documents
Prompt > ls home/student

file
Prompt > cp home/student/file /Documents
Prompt > ls home/student

file
Prompt > ls Documents

file
Prompt > 

```

The file in Documents.

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 163
Dumping file TestVolume, starting at block 163 for 1 block:

014600: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014610: 00 00 00 00 00 00 87 BB 9B 54 A2 00 00 02 00 00 | .....T.....
014620: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014630: 00 00 00 00 00 00 6C BB 9B 54 A0 00 00 02 00 00 | .....T.....
014640: 66 69 6C 65 00 00 00 00 00 00 00 00 00 00 00 00 | file.....
014650: 00 00 00 00 00 00 87 BB 9B 54 A6 00 A3 02 00 00 | .....T.....
014660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014680: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014690: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0146F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

014700: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014710: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

Same file in home/student

```
student@student-VirtualBox:~/Documents/csc415-filesystem-aktails$ Hexdump/hexdump
p.linux --file TestVolume --count 1 --start 164
Dumping file TestVolume, starting at block 164 for 1 block:

014800: 2E 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014810: 00 00 00 00 00 00 7B BB 9B 54 A3 00 00 02 00 00 | .....{T.....
014820: 2E 2E 00 00 00 00 00 00 00 00 00 10 00 00 00 00 | .....
014830: 00 00 00 00 00 00 6D BB 9B 54 A1 00 00 02 00 00 | .....mT.....
014840: 66 69 6C 65 00 00 00 00 00 00 00 00 00 00 00 00 | file.....
014850: 00 00 00 00 00 00 7B BB 9B 54 A4 00 A3 02 00 00 | .....{T.....
014860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
014890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0148F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```