

# File System Writeup

## Bierman's Baddies

Avinh Huynh - 923666650

Hilary Lui - 922142725

Jacob Vazquez- 923077698

Lita Hernandez-Gonzalez - 921182337

Casey Steven - 921956643



Github: [Jacob9610](#)

# Table Of Contents:

<b>Description:</b>	<b>4</b>
<b>Approach:</b>	<b>5</b>
Freespace Approach:	5
Freespace Approach 1:	5
Freespace Approach 2:	8
Directory Approach:	11
Function Name: loadDir	11
Function Name: writeDir	12
Function Name: parsePath	13
Function Name: findInDir	14
Approach mfs	15
Function Name: fs_mkdir	15
Function Name: fs_rmdir	16
Function Name: fs_delete	17
Function Name: fs_mov	18
Function Name: fs_getcwd	19
Function Name: fs_setcwd	20
Purpose:	20
Input Parameters:	20
Processing Details:	21
Return Values:	21
Output:	22
Edge Cases:	22
Key Assumptions:	22
Function Name: fs_isFile	22
Function Name: fs_isDir	23
<b>Milestones:</b>	<b>29</b>
Milestone 1:	29
1. A dump (use the provided HexDump utility) of the volume file that shows the VCB, FreeSpace, and complete root directory	29
Milestone 2: File System Operations	37
Key File System Operations to Implement	37
Functions to Implement	38
Interrelationships Between Functions	40

The core interactions between these functions can be visualized as follows:.....	40
• Directory Management: Functions like fs_opendir, fs_readdir, and fs_closedir form a directory handling subsystem. fs_opendir opens a directory, fs_readdir reads its contents, and fs_closedir frees resources when done.....	40
• Path Management: Functions like fs_setcwd, fs_getcwd, fs_isFile, and fs_isDir rely on parsing the path and identifying whether it's a file or directory. These functions interact with the directory structure to provide information about file system state.....	40
• File Manipulation: fs_mkdir, fs_delete, and fs_rmdir manipulate files and directories by adding or removing entries in the file system.....	40
Conclusion.....	40
By implementing these functions, we will provide essential file system capabilities that are necessary for the shell to interact with directories and files. Functions for manipulating directories, obtaining file information, and managing the current working directory will be crucial for the next steps in developing the file system's command-line interface.....	40
Milestone 3:.....	41
Functions to Implement.....	41
<b>Issues and Resolutions:.....</b>	<b>43</b>
General Issues:.....	43
Time_t:.....	43
Could not initialize freespace:.....	44
Global Variables showing undefined:.....	45
Freespace Issues:.....	47
Directory Issues:.....	48
Make Dr not updating the parent dir with child info.....	48
fs_readdir not implemented correctly.....	49
B_io issues:.....	54
<b>Analysis:.....</b>	<b>56</b>
VolumeControlBlock: 1st block.....	57
DirectoryEntries: 2nd-3rd block.....	60
<b>Screen shot of compilation:.....</b>	<b>65</b>
make:.....	65
<b>Screen shot(s) of the execution of the program:.....</b>	<b>66</b>
make run.....	66
ls.....	67
CP.....	68
MV.....	68
MD.....	70

RM.....	71
TOUCH.....	72
CAT.....	73
CP2L.....	73
Screen shot(s) of the Analysis hexdump:.....	74

## File System Writeup

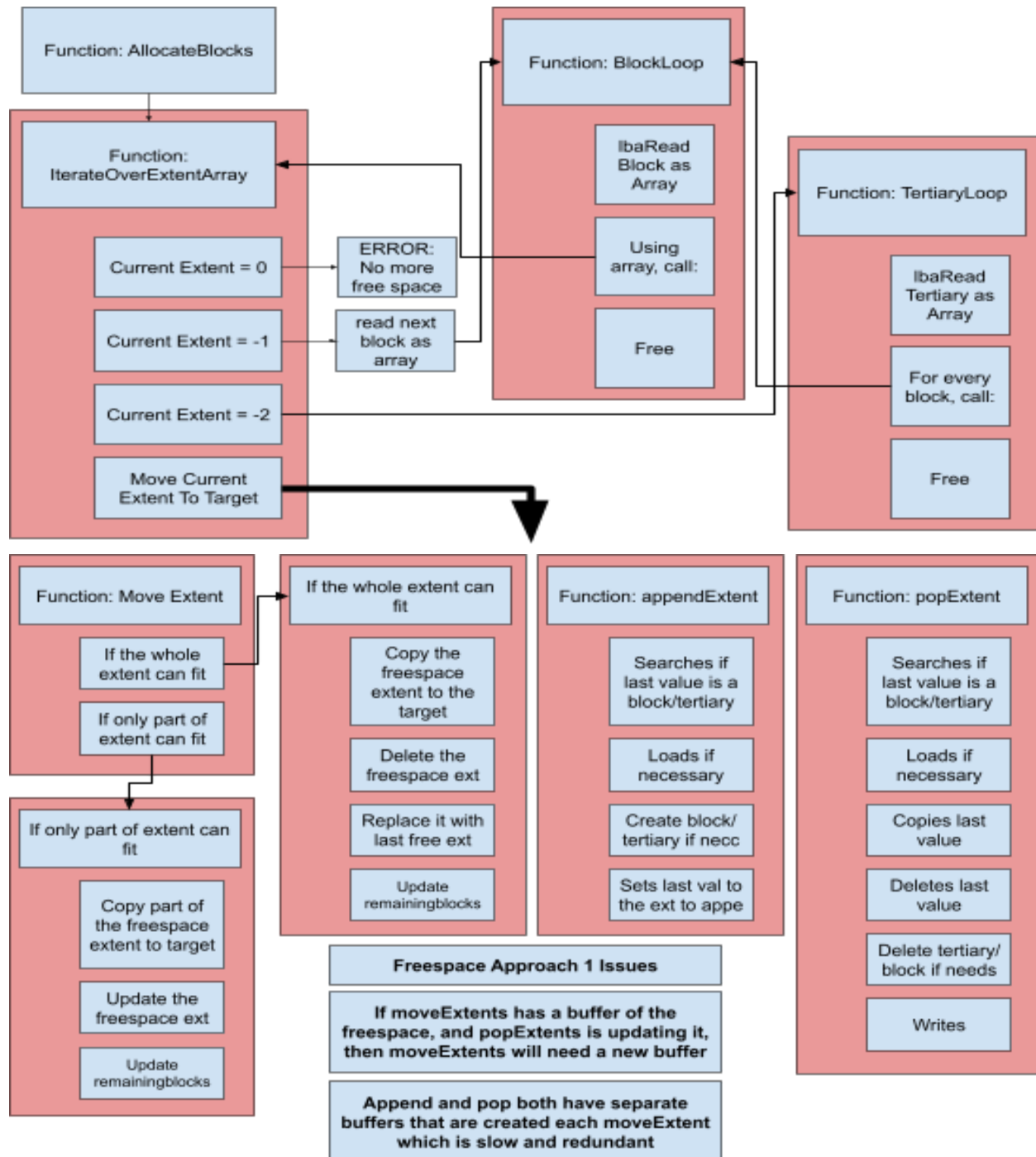
### **Description:**

We are building a file system where we will need to format a volume, create and maintain a free space management system, initialize a root directory and maintain directory information, create, read, write, and delete files, and display info.

## Approach:

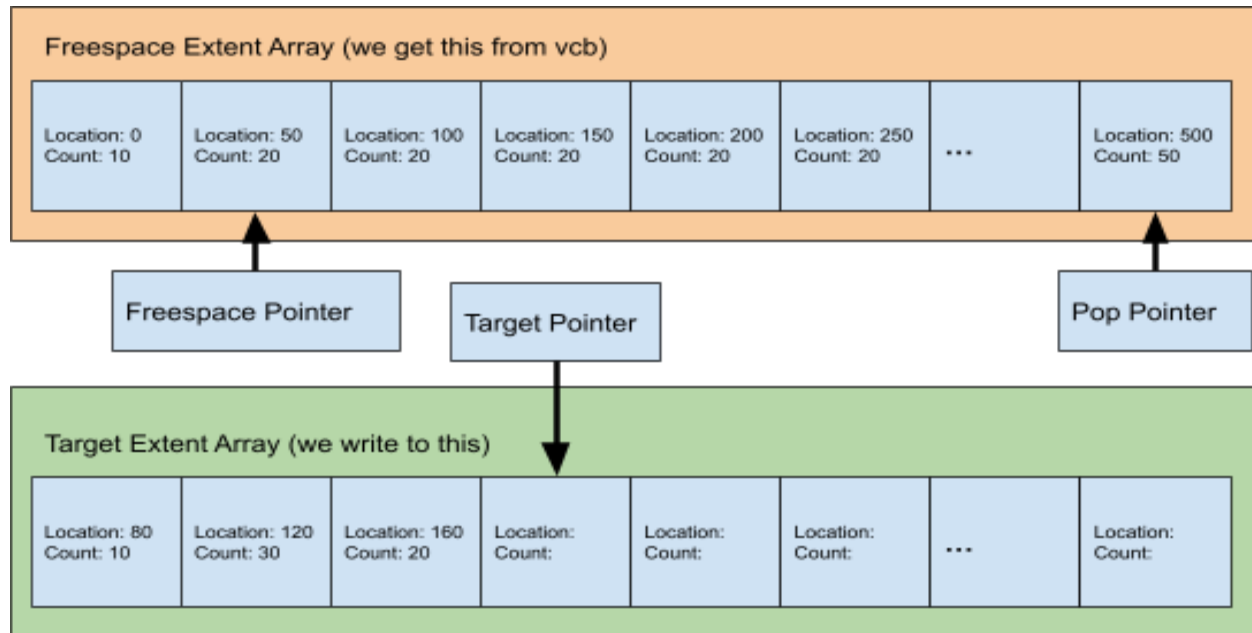
### Freespace Approach:

#### Freespace Approach 1:

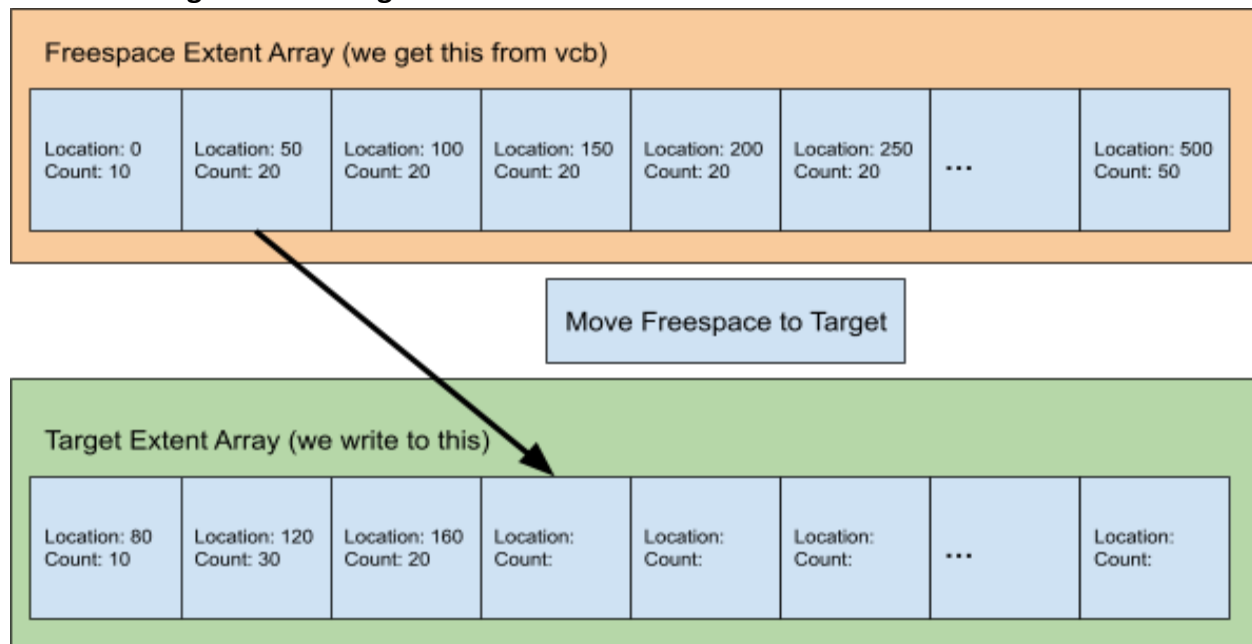


The last attempt iterated through every value of freespace to a valid extent. When a valid extent was found, it would iterate through the target extent array to find where to place it. Then it would delete the original value in freespace we just moved. In order to delete it, it iterated backwards through freespace to pop the last value and replace our original with that.

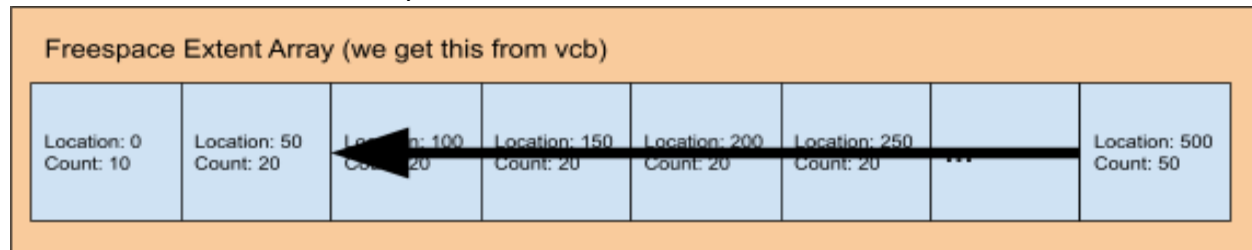
### Each Iteration:



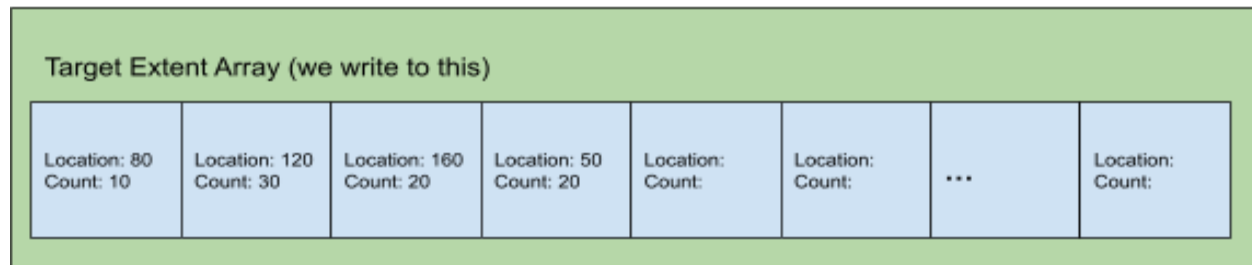
### Moving Extent to Target:



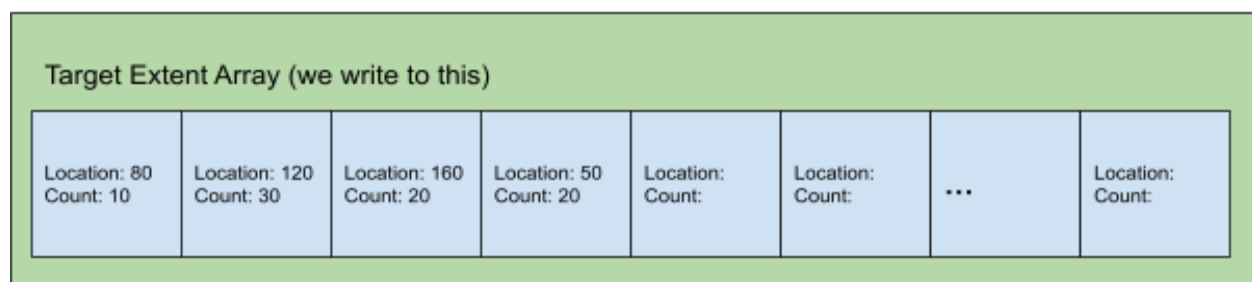
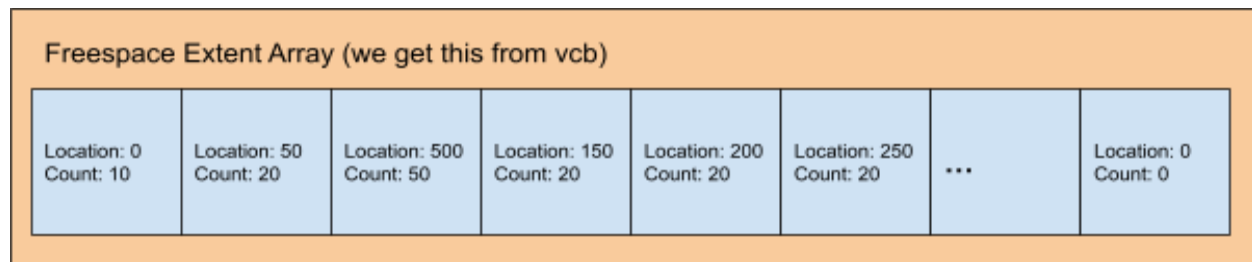
### Pop last freespace:



Move Last Freespace to Index



**Result:**



Now we have one extent moved over, and we can continue iteration. Since we just popped, we can keep the freespace index the same and then test it for contiguousBlock requirement.

However, this approach doesn't work because of buffers. When implementing this, there were issues with buffers for the pop pointer and the freespace pointer. The way it was implemented before would force me to LBAread everytime I popped and to check if the buffers were the same to know to LBAread or not. This could mean upwards of 3 LBAreads per iteration. Seeing this, I scrapped the idea and started from the beginning.



## Freespace Approach 2:

In the second attempt, I wanted to encapsulate all of the Extent functions to be handled by an ExtentController, and then my allocate blocks function would only have to work with that.

These simple functions would have their own ExtentController data structure that would handle buffers, blocks, tertiary, if they were ever empty or full. Then I could treat the extents as an arraylist and use the same pseudocode as above.

With only these functions, I was able to type out the pseudocode for allocateBlocks and then write it out in vscode.

### Extent Controller Functions:

- getNext()
- getPrevious()
- append()
- pop()

### ExtentController Original:

```
ExtentArrayController {  
    Extent *start    // the first value in the extent array  
    Extent *end      // the last value in the extent array  
    Extent *array    // buffer of what is currently being read  
    int arrayIndex   // where in the array it is  
    int arraySize    // array size  
    int blockNumber  // relative block number  
}
```

There would be an ExtentController (EC) to replace the target pointer, freespace pointer, and pop pointer. Then the data structure would manage their own buffers. The ExtentController have changed since then, and to see the rest of the pseudocode see Freespace Issues.

Continue in Freespace Issues

**AllocateBlocks Original:**

AllocateBlocks(int remainingBlocks, int contiguousBlocks, Extent \*target):

```
/* Error checking for inputs here */

VolumeControlBlock vcb = malloc()
lbaRead(vcb, 0, 1)

/* Error checking for malloc and lbaRead here */

target = &target.getLast().getNext()      // target (where to write to)
Extent *fs = vcb.freespace[0]             // freespace (where to write from)
Extent *end = vcb.freespace[0].getLast()   // last value of fs (used to delete fs)

while(0):
    if (fs.blockCount >= remainingBlocks): // add this extent
        if (remainingBlocks >= fs.blockCount): // add the whole extent

            // set target := fs (the values and not the pointer)
            target.blockCount = fs.blockCount
            target.location = fs.location

            // update target
            target = target.getNext()

            // update remainingBlocks
            remainingBlocks -= fs.blockCount

            // delete fs and replace it with a new value
            fs.blockCount = end.blockCount
            fs.location = end.location

            // replace end
            if ( // check if replacing this will make it the same block as fs
                end.arrayIndex == 0 AND
                end.blockNumber == fs.blockNumber + 1
            ):
                free(end.array)
                // share the same buffer so fs can handle lbaWrite
                end.array = fs.array
                end.blockNumber = end.blockNumber - 1
                end.arraySize = g_blocksize / sizeof(Extent)
                end.arrayIndex = arraySize - 1
                end.end = array[arrayIndex]
            else:
```

```
        end = end.getPrevious()

        // update index to retry this index
        fs.index -= 1

    else:    // allocate only part of the block

        // set target to only part of fs
        target.blockCount = remainingBlocks
        target.location = fs.location

        // update fs to reflect part has been allocated
        fs.blockLocation += remainingBlocks
        fs.blockCount -= remainingBlocks

        // update remainingBlocks
        remainingBlocks = 0

        // if we are done allocating (can occur from partial allocation or
        // allocating exactly from a whole extent)
        if remainingBlocks = 0:

            free(target)
            free(fs)
            free(end)

            return 0

    // Either we allocated one extent and we still have remaining blocks or the extent
    // wasn't big enough for contiguous blocks. Iterate fs to next.
    fs = fs.getNext()

    if (fs.getNext() = -1):

    If end.getPrevious runs into fs.getNext()

    if end.getPrevious is one block and fs.getNext() is another

    check if they are equal, and one more after that throw an error
```

## Directory Approach:

Function Name: **loadDir**

### Purpose:

The **loadDir** function dynamically loads the content of a directory into memory. It performs several validation checks to ensure that the provided input is a valid directory and then allocates sufficient memory to hold the directory's data. The data is read from disk using block-based I/O.

### Input Parameters:

- **entry**: A pointer to a **DirectoryEntry** structure that identifies the directory to load. If this parameter is **NULL** or points to a file rather than a directory, the function will return **NULL**.

### Processing Details:

1. The function checks whether the provided **DirectoryEntry** is **NULL** or represents a file (**entry->isDir == 'F'**). If either condition is true, it returns **NULL**.
2. It calculates the number of blocks needed to hold the directory based on its size and the global block size (**g\_blockSize**).
3. The total memory required for the directory is computed and dynamically allocated using **malloc**.
4. The **LBRead** function is used to read the directory data from the disk into the allocated memory.

### Output:

The function returns a pointer to a newly allocated **DirectoryEntry** structure containing the loaded directory data. If the input is invalid, it returns **NULL**.

### Edge Cases:

- If the input is **NULL** or points to a file, the function does not proceed with loading and safely returns **NULL**.
- The function assumes that the directory data does not exceed the bounds of the disk storage.

**Purpose:**

The **writeDir** function writes the data for a directory to the filesystem, ensuring that it is stored persistently on the disk.

**Input Parameters:**

- **dir**: A pointer to a **DirectoryEntry** structure containing the directory's metadata, including its size and the starting block location.

**Processing Details:**

1. **Block Calculation:**

The function calculates the number of blocks required to store the directory using the formula:

$$\text{blocks} = \frac{\text{dir}[0].\text{size} + (\text{g\_blockSize} - 1)}{\text{g\_blockSize}}$$

This accounts for potential alignment with the block size.

2. **Data Writing:**

The **LBAwrite** function is used to write the directory data to the specified starting block. The return value from **LBAwrite** indicates the number of blocks successfully written.

3. **Diagnostic Output:**

The function prints debugging information, including the size of the directory, the global block size, and the return value from **LBAwrite**.

**Output:**

The function does not return a value but provides diagnostic output to verify that the directory was written successfully.

**Edge Cases:**

- The directory's size must not exceed the maximum capacity of the allocated blocks. If this is a possibility, additional validation should be added.

**Key Assumptions:**

- The **LBAwrite** function correctly handles writing the data to the specified blocks.
- The directory's starting block location and size are accurately set in the **DirectoryEntry** structure.

**Purpose:**

To parse and resolve a file path in the custom filesystem, identifying its parent directory and the location of the target element.

**Input Parameters:**

- **`path`**: The file path as a string. Must begin with `/` for root-based paths or relative otherwise.
- **`retParent`**: A pointer where the function stores the resolved parent directory of the target element.
- **`retIndex`**: A pointer to an integer where the index of the target element within its parent directory is stored.
- **`lastElementName`**: A pointer to a string to store the name of the last element in the path.

**Processing Details:**

1. **Path Validation:**  
The function ensures the `path` is neither NULL nor empty. If invalid, it returns `-1`.
2. **Starting Directory:**  
Determines whether to begin traversal from the root directory (`g_root`) or the current working directory (`g_cwd`), based on whether the path starts with `/`.
3. **Tokenization:**  
Uses `strtok_r` to split the path into tokens, processing each component iteratively.
4. **Directory Traversal:**
  - Looks up each token in the current directory using `findInDir`.
  - Validates if the current token refers to a directory. If it's a file, traversal stops with an error.
5. **Parent Directory Resolution:**  
If the last token is reached, the function stores the parent directory, index of the last element, and the element's name in the provided pointers.
6. **Dynamic Memory Management:**  
Frees the old parent directory and loads the new one during traversal.

**Output:**

- On success, the function updates the `retParent`, `retIndex`, and `lastElementName` with the resolved values and returns `0`.
- Returns `-1` if the path cannot be resolved (e.g., file not found, invalid path).

**Edge Cases:**

- Input path is `/`, resulting in the root directory being returned as the parent.

- Paths containing non-existent elements or invalid tokens (e.g., referencing files as directories).

#### Key Assumptions:

- The **findInDir** and **loadDir** helper functions correctly resolve directory entries and handle dynamic memory operations.

Function Name: **findInDir**

#### Purpose:

Searches for a specified entry by name within a directory's entries and returns its index.

#### Input Parameters:

- **dir**: A pointer to the array of **DirectoryEntry** structs representing the directory to search within.
- **name**: The name of the entry to locate within the directory.

#### Processing Details:

1. **Validation:**  
Ensures the **dir** pointer and **name** string are not NULL. If either is invalid, the function returns **-2**.
2. **Number of Entries:**  
Computes the number of directory entries using the size field in the first entry of the **dir** array.
3. **Iteration:**
  - Iterates through each directory entry.
  - Checks if the directory entry is marked as used using the helper function **dirEntryUsed**.
  - Compares the entry's name with the provided **name** using **strcmp**.
4. **Return Matching Index:**  
If a match is found, the index of the matching entry is returned.
5. **Entry Not Found:**  
If no match is found after iterating through all entries, the function returns **-1**.

#### Output:

- Index of the matching directory entry if found.
- **-1** if the entry is not found in the directory.
- **-2** if the input directory or name is invalid.

#### Edge Cases:

- If the directory is empty, the function will return **-1**.

- If the directory or name is NULL, the function will return **-2**.

#### Key Assumptions:

- The **dirEntryUsed** function correctly identifies whether a directory entry is valid and in use.
- The **dir** array is properly formatted, with the first entry containing size information.

#### Approach mfs

Function Name: **fs\_mkdir**

#### Purpose:

Creates a new directory at the specified path by validating the path, locating the parent directory, and populating a new directory entry in the parent directory.

#### Input Parameters:

- **pathname**: The full path where the directory is to be created. This path includes the name of the new directory as its last element.
- **mode**: (Currently unused) Specifies the permissions for the directory.

#### Processing Details:

1. Path Parsing:
  - Uses **parsePath** to validate the given path and identify the parent directory and the name of the new directory.
  - If **parsePath** fails (returns **-1**), the function exits early, as the path is invalid or does not exist.
2. Directory Existence Check:
  - If the directory already exists in the parent directory (i.e., **index** is not **-1**), the function exits with an error.
3. Directory Creation:
  - Calls **createDir** to allocate and initialize the new directory structure.
  - Finds an unused directory entry slot in the parent directory using the helper function **dirEntryUsed**.
  - Populates the unused slot with the newly created directory's metadata and assigns it the provided name.
4. Parent Directory Update:
  - Updates the parent directory to reflect the new directory entry.
  - Writes the updated parent directory to storage using **writeDir**.
5. Return Values:
  - Returns **0** on successful directory creation.
  - Returns **-1** if the path is invalid or the directory already exists.



- A newly created directory at the specified location within the file system.

#### Edge Cases:

- If the path ends in a separator (e.g., /), the function creates a directory at the root or the current working directory.
- Handles the case where all directory entries in the parent are already in use.

#### Key Assumptions:

- `parsePath`, `createDir`, and `dirEntryUsed` are implemented correctly and handle edge cases.
- Parent directories are large enough to accommodate additional entries.

Function Name: `fs_rmdir`

#### Purpose:

Deletes an existing directory from the file system by removing its directory entry and freeing the blocks allocated to it.

#### Input Parameters:

- `pathname`: The full path of the directory to be removed.

#### Processing Details:

1. Path Parsing:
  - Uses `parsePath` to locate the directory and its parent.
  - Validates the path and retrieves the `index` of the directory entry in its parent.
2. Validation:
  - Checks if the specified path points to a directory using `fs_isDir`.
  - If the path is invalid, or the target is not a directory, the function returns an error.
3. Directory Removal:
  - Calls `freeBlocks` to release the storage blocks allocated to the directory.
  - Marks the directory entry as unused by setting its `isDir` attribute to 'N'.
4. Parent Directory Update:
  - Updates the parent directory to reflect the removed entry.
  - Writes the updated parent directory back to storage using `writeDir`.
5. Return Values:
  - Returns `0` on successful directory removal.
  - Returns `-1` if:
    - The path is invalid.
    - The specified target is not a directory.

- Other errors occur during the process.

#### Output:

- The directory at the specified path is deleted from the file system.

#### Edge Cases:

- If the directory is not empty, additional validation may be needed to prevent its removal (not implemented here).
- Handles invalid paths gracefully without crashing.

#### Key Assumptions:

- `parsePath`, `fs_isDir`, and `freeBlocks` are implemented correctly.
- The parent directory has sufficient capacity to be updated without causing additional issues.

#### Function Name: `fs_delete`

#### Purpose:

Removes a file from the file system by releasing its storage blocks and marking its entry in the parent directory as unused.

#### Input Parameters:

- `filename`: The full path to the file that needs to be deleted.

#### Processing Details:

1. Path Parsing:
  - Uses `parsePath` to find the file and its parent directory.
  - Retrieves the `index` of the file entry in the parent directory and its name (`lastElementName`).
2. Validation:
  - Confirms the path is valid. If `parsePath` fails, the function returns an error.
  - Verifies that the target is a file, not a directory, using `fs_isDir`. If the target is a directory, the function returns an error.
3. File Deletion:
  - Calls `freeBlocks` to release the allocated storage blocks for the file.
  - Updates the directory entry to indicate it is unused by setting its `isDir` attribute to `'N'`.
4. Parent Directory Update:
  - Updates the parent directory after marking the entry unused.
  - Writes the updated parent directory back to storage using `writeDir`.
5. Return Values:

- Returns **0** on successful deletion.
- Returns **-1** if:
  - The path is invalid.
  - The target is not a file.
  - Errors occur during the process.

#### Output:

- The specified file is removed from the file system.

#### Edge Cases:

- Ensures that an attempt to delete a directory instead of a file is correctly flagged as an error.
- Handles invalid paths gracefully without crashing.

#### Key Assumptions:

- **parsePath**, **fs\_isDir**, and **freeBlocks** functions are implemented correctly.
- Parent directories can be safely updated without additional complications.

#### Function Name: **fs\_mov**

#### Purpose:

Relocates a file from a source path to a destination directory within the file system.

#### Input Parameters:

- **filename**: The path to the file that needs to be moved.
- **pathname**: The path to the target directory where the file will be moved.

#### Processing Details:

1. Path Parsing:
  - Uses **parsePath** to locate the source file (**filename**) and the destination directory (**pathname**).
  - Retrieves parent directory pointers (**parent1**, **parent2**), indices (**index1**, **index2**), and the last element names (**lastElementName1**, **lastElementName2**) for both paths.
2. Validation:
  - Confirms both paths are valid.
  - Ensures the source path (**filename**) is a file and not a directory using **fs\_isDir**.
  - Verifies the destination path (**pathname**) is a directory.
3. Destination Directory Handling:
  - Loads the destination directory's entries into memory.

- Identifies the first available (unused) entry in the destination directory for placing the file.
- 4. File Move:
  - Copies the source file's metadata from the parent directory's entry (`parent1[index1]`) into the destination directory's unused entry.
  - Updates the destination directory to reflect the new entry.
- 5. Post-Move Update:
  - The parent directory of the source file is updated to mark its entry as unused.
- 6. Return Values:
  - Returns `0` on successful completion.
  - Returns `-1` if:
    - Either path is invalid.
    - The source is a directory instead of a file.
    - The destination is not a directory.
    - Other errors occur during the process.

**Output:**

- The file is removed from its original location and added to the destination directory.

**Edge Cases:**

- Handles scenarios where the source path or destination path is invalid.
- Ensures the target file isn't overwritten by mistake by checking for unused directory entries.

**Key Assumptions:**

- Functions like `parsePath`, `fs_isDir`, and `dirEntryUsed` work as expected.
- The destination directory can accommodate the new entry without requiring expansion.

**Function Name:** `fs_getcwd`

**Purpose:**

Provides the current working directory's path to the calling process, storing it in a user-provided buffer.

**Input Parameters:**

- `pathname`: A pointer to a buffer where the current working directory path will be copied.
- `size`: The size of the buffer, ensuring it can safely hold the directory path without causing overflow.

**1. Path Copying:**

- Copies the path stored in the global variable `g_cwd_path` into the provided `pathname` buffer using `strncpy`.
- Ensures that the number of bytes copied does not exceed the buffer size (`size`).

**2. Return Value:**

- On success, returns the `pathname` pointer containing the current working directory's path.
- If the operation fails (though none is explicitly handled here), typically `NULL` would be returned.

**3. Buffer Safety:**

- The use of `strncpy` ensures that the function respects the buffer size (`size`) and prevents buffer overflows.

**Output:**

- The function writes the current working directory path to the buffer pointed to by `pathname`.

**Edge Cases:**

- The function does not validate whether the `pathname` buffer is large enough to hold the full path. If `size` is smaller than the length of `g_cwd_path`, the resulting string will not be null-terminated, leading to potential undefined behavior.

**Key Assumptions:**

- `g_cwd_path` is a valid, null-terminated string representing the current working directory path.
- The calling function provides a sufficiently large buffer for the directory path.

**Function Name: `fs_setcwd`**

**Purpose:**

Sets the current working directory to the specified path by validating and processing the path, then updating global variables representing the current working directory.

**Input Parameters:**

- `pathname`: The path of the directory that should be set as the current working directory.
  - *Type:* `char*`

- **Constraints:** This path must be valid, pointing to an existing directory. The path can be either absolute (starting with `/`) or relative.

#### Processing Details:

- **Path Parsing:**
  - The function calls `parsePath` to process the given `pathname`. This function attempts to locate the directory and gather details like the parent directory and index of the directory in the file system structure.
  - If parsing fails or the path is invalid, the function returns `-1`.
- **Directory Validation:**
  - The function checks whether the path points to a directory by verifying the `isDir` field in the directory entry.
  - If the path is not a directory, an error message is printed, and the function returns `-1`.
- **Directory Load:**
  - If the path is valid and points to a directory, the `loadDir` function is called to load the directory's contents into memory.
  - If the directory fails to load, the function prints an error and returns `-1`.
- **Global CWD Update:**
  - The global variable `g_cwd` is updated to the newly loaded directory, making it the current working directory.
- **Path Update:**
  - The global variable `g_cwd_path` is updated based on the provided `pathname`.
  - If the path is absolute, `g_cwd_path` is directly set to `pathname`.
  - If the path is relative, the function appends the last element of the parsed path to `g_cwd_path`.
- **Path Simplification:**
  - The function performs path normalization by handling special cases like `.` (current directory) and `..` (parent directory). These are processed using `strtok_r` to break the path into tokens and resolve any redundant or relative components.
  - After resolving these elements, the function builds the new, simplified `g_cwd_path`.

#### Return Values:

- **0:** The current working directory is successfully set.
- **-1:** If any of the following conditions occur:
  - The path is invalid.
  - The path does not point to a directory.
  - An error occurs while loading the directory.

- **g\_cwd**: The global current working directory is updated to the new directory.
- **g\_cwd\_path**: The global path is updated to reflect the new working directory, with normalization applied to handle redundant elements in the path.

#### Edge Cases:

- **Invalid Path**: If the path does not exist or cannot be parsed, the function will return **-1**.
- **Non-directory Path**: If the path does not point to a directory, the function will report an error.
- **Relative Path Handling**: The function correctly handles relative paths and resolves elements like **.** and **..** appropriately.

#### Key Assumptions:

- The helper functions **parsePath**, **loadDir**, and **strtok\_r** work as expected.
- The global variables **g\_cwd** and **g\_cwd\_path** are properly initialized and managed elsewhere in the system.

#### Function Name: **fs\_isFile**

##### Purpose:

Determines whether a given pathname points to a file (not a directory).

##### Input Parameters:

- **filename**: The path of the file to check. It can be either absolute or relative.

##### Processing Details:

- **Path Parsing**:  
The function uses **parsePath** to resolve the provided filename into a directory entry, obtaining the parent directory, index, and last element name.
- **Path Validation**:  
If **parsePath** fails (returns -1), the function prints an error message and returns **0**.
- **Index Lookup**:  
The **findInDir** function is used to search for the entry corresponding to the last element of the filename within the directory. If found, the index of the file is obtained.
- **File Type Check**:  
The function checks the **isDir** attribute of the directory entry at the specified index. If **isDir == 'F'**, it indicates a file and returns **1**. Otherwise, it returns **0**.

##### Return Values:

- **1** if the specified path points to a file.

- **0** if the specified path does not point to a file or if any errors occur (e.g., invalid path).

#### Edge Cases:

- The function handles invalid paths by returning **0** and printing an error message if **parsePath** cannot find the pathname.
- If the path points to a directory or an invalid file type, it returns **0**.

#### Key Assumptions:

- The **parsePath** and **findInDir** functions work as expected and correctly resolve the file's path and directory entries.
- The **isDir** field of **DirectoryEntry** accurately represents the type of the file system entry.

Function Name: **fs\_isDir**

#### Purpose:

Determines whether a given pathname points to a directory (not a file).

#### Input Parameters:

- **pathname**: The path of the directory to check. It can be either absolute or relative.

#### Processing Details:

- **Path Parsing:**  
The function uses **parsePath** to resolve the provided pathname into a directory entry, obtaining the parent directory, index, and last element name.
- **Path Validation:**  
If **parsePath** fails (returns -1), the function prints an error message and returns **0**.
- **Index Lookup:**  
The **findInDir** function is used to search for the entry corresponding to the last element of the pathname within the directory. If found, the index of the directory is obtained.
- **Directory Type Check:**  
The function checks the **isDir** attribute of the directory entry at the specified index. If **isDir == 'T'**, it indicates a directory and returns **1**. Otherwise, it returns **0**.

#### Return Values:

- **1** if the specified path points to a directory.
- **0** if the specified path does not point to a directory or if any errors occur (e.g., invalid path).

#### Edge Cases:



- The function handles invalid paths by returning **0** and printing an error message if **parsePath** cannot find the pathname.
- If the path points to a file or an invalid directory type, it returns **0**.

#### Key Assumptions:

- The **parsePath** and **findInDir** functions work as expected and correctly resolve the directory's path and entries.
- The **isDir** field of **DirectoryEntry** accurately represents the type of the file system entry.

#### Function Name: **fs\_opendir**

##### Purpose:

Opens a directory specified by the given path and prepares it for iteration. The function loads the directory's entries and provides a file descriptor structure that can be used to iterate over the directory's content.

##### Input Parameters:

- **pathname**: The path to the directory that needs to be opened. The path should point to an existing directory.

##### Processing Details:

- **Path Parsing:**  
The function calls **parsePath** to parse the given path (**pathname**). It retrieves pointers to the parent directory, the index of the last element, and the last element's name in the directory structure.
- **Directory Validation:**  
The directory is checked for validity by verifying the parsed path. If the path is invalid, the function logs an error message and returns **NULL**.
- **Directory Entry Loading:**  
After validating the path, the function calls **loadDir** to load the directory's entries into memory. It then calculates the number of entries in the directory by dividing the directory size by the size of a **DirectoryEntry**.
- **File Descriptor Creation:**  
The function allocates memory for an **fdDir** structure. This structure contains:
  - A pointer to the loaded directory entries (**directory**).
  - The number of entries in the directory (**d\_reclen**).
  - A pointer to a **fs\_diriteminfo** structure that holds directory item information.
  - The current position within the directory (**dirEntryPosition**).

##### Return Values:

- Returns a pointer to an **fdDir** structure on success. The structure contains the directory's entries and metadata for iteration.
- Returns **NULL** if there is an error in parsing the path or loading the directory.

#### Edge Cases:

- The function handles cases where the provided pathname is empty or invalid. If the pathname is invalid, it logs an error and returns **NULL**.
- It assumes the directory entries can fit in memory and does not handle cases where the directory is too large to load.

#### Key Assumptions:

- The **parsePath** function correctly resolves the path and provides valid pointers and indices.
- The **loadDir** function is capable of loading the directory entries into memory correctly.
- The directory size is appropriate for the number of entries it contains, as the function calculates the number of entries by dividing the directory size by the size of each directory entry.

#### Function Name: **fs\_readdir**

**Purpose:** Reads the next directory entry from the directory stream and returns the associated file or directory information.

#### Input Parameters:

- **dirp**: A pointer to a **fdDir** structure, which contains the current position and directory entries for iteration.

#### Processing Details:

##### Directory Entry Processing:

- The function checks the type of the directory entry at the current position (**dirp->dirEntryPosition**).
  - If the entry is a directory (**isDir == 'T'**), the function sets the **fileType** in the **fs\_diriteminfo** structure to **FT\_DIRECTORY**, copies the entry's name, and sets the record length based on the entry's size.
  - If the entry is a regular file (**isDir == 'F'**), it similarly sets the **fileType** to **FT\_REGFILE**, copies the name, and calculates the record length.
  - If the entry type is unrecognized, it sets the entry's name to an empty string.

#### End of Directory:

- If the current directory entry position matches the total number of entries (`dirp->dirEntryPosition == dirp->d_reclen`), the function sets the `fs_diriteminfo` pointer to `NULL`, signaling the end of the directory stream.

#### Return Values:

- Returns a pointer to a populated `fs_diriteminfo` structure containing the current directory entry's name, type, and record length.
- Returns `NULL` when the end of the directory is reached.

#### Output:

- The directory entry data is copied into the `fs_diriteminfo` structure for further processing or use.
- The `dirp->dirEntryPosition` is incremented to point to the next directory entry for the next call.

#### Edge Cases:

- Handles directories and regular files separately.
- If the directory stream reaches the end, it returns `NULL`.

#### Key Assumptions:

- Assumes that the `fdDir` structure and `fs_diriteminfo` are properly initialized.
- Assumes the `isDir` field in the `DirectoryEntry` structure correctly identifies the type of entry (`T` for directory, `F` for regular file).

#### Function Name: `fs_closedir`

**Purpose:** Closes the directory stream, releases memory, and resets the directory pointer to prevent memory leaks.

#### Input Parameters:

- `dirp`: A pointer to a `fdDir` structure, representing the directory stream that needs to be closed.

#### Processing Details:

- The function first checks if the `dirp` pointer is `NULL`. If it is, the function returns `-1`, indicating an error, as it is not possible to close a `NULL` directory stream.
- If `dirp` is valid, the function:
  - Sets the `directory` field in the `fdDir` structure to `NULL`.
  - Sets the `dirp` pointer itself to `NULL` to ensure it no longer points to the now-freed memory.

- Frees the memory allocated for **dirp**, releasing the resources associated with the directory stream.

#### Return Values:

- Returns **0** on success, indicating that the directory stream was successfully closed.
- Returns **-1** if the input **dirp** is **NULL**, indicating an invalid directory stream.

#### Output:

- The memory associated with the directory stream is deallocated, ensuring that resources are properly released and no memory leak occurs.

#### Edge Cases:

- Handles the case where the **dirp** pointer is **NULL** and prevents further operations on an invalid directory stream.

#### Key Assumptions:

- Assumes that the **fdDir** structure was previously allocated and initialized correctly.
- Assumes that memory management (e.g., using **malloc** or **free**) is correctly handled outside of this function.

#### Function Name: **fs\_stat**

**Purpose:** Retrieves and stores file statistics for a specified file or directory in the **fs\_stat** structure.

#### Input Parameters:

- **path**: A string representing the path of the file or directory whose statistics need to be retrieved.
- **buf**: A pointer to an **fs\_stat** structure where the file statistics (e.g., access time, modification time, size, etc.) will be stored.

#### Processing Details:

- The function first uses **parsePath** to resolve the path, returning a pointer to the parent directory, the index of the entry, and the last element in the path.
  - If the path is invalid or parsing fails (**check == -1**), the function returns **-1** to indicate an error.
- The function then loads the directory entry corresponding to the path using **loadDir**.
- Once the directory entry is loaded, the following statistics are populated into the **fs\_stat** structure (**buf**):
  - **st\_accesstime**: Set to the last accessed time of the file or directory.

- **st\_blocks**: The number of blocks the file or directory occupies, calculated based on its size and the block size (**g\_blockSize**).
- **st\_blksize**: The block size used for file storage (**g\_blockSize**).
- **st\_createtime**: Set to the creation time of the file or directory.
- **st\_modtime**: Set to the last modification time of the file or directory.
- **st\_size**: The total size of the file or directory.

#### Return Values:

- Returns **0** on success, indicating that the file statistics were successfully retrieved and stored in the **buf** structure.
- Returns **-1** if the path is invalid or an error occurs while retrieving file information.

#### Output:

- The function fills the **fs\_stat** structure with the file's or directory's statistics.

#### Edge Cases:

- Handles cases where the path is invalid or the file/directory cannot be found, returning **-1** in those cases.

#### Key Assumptions:

- Assumes that the **parsePath** and **loadDir** functions correctly resolve and load the file or directory.
- Assumes that the global variable **g\_blockSize** is properly initialized and holds the correct block size for file storage.

## Milestones:

### Milestone 1:

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

a. [Full Hexdump](#)

b. VCB

```

000200: 4C 42 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 | .L.....
000210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000220: 00 00 00 00 00 00 00 00 09 00 01 00 00 00 00 00 | .....
000230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000250: F5 6B C1 FA 00 00 00 00 00 96 98 00 00 02 00 00 | k.....
000260: 4B 4C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | KL.....
000270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

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

```

Extent freeSpaceMapLocation[EXTENT_ARRAY_SIZE];	Size = 10 x 4 Bytes 1st value is 0x 42 4C & 00 0A blockCount = 0x 42 4C = 16972 blockLocation = 0x 00 0A = 10
Extent rootLocation[EXTENT_ARRAY_SIZE];	Size = 10 x 4 Bytes 1st value is 0x 00 09 & 00 01 blockCount = 0x 00 09 = 9 blockLocation = 0x 00 01 = 1

<code>long long signature;</code>	Signature = 0x 00 00 00 00 FA C1 6B F5 = 4206980085
<code>unsigned int totalBytes;</code>	totalBytes = 0x 00 98 96 00 = 9999872
<code>unsigned int blockSize;</code>	blockSize = 0x 00 00 02 00 = 512
<code>short totalBlocks;</code>	totalBlocks = 0x 4C 4B = 19531

c. Directory Entry

```

000400: 09 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000410: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000420: 00 00 00 00 00 00 00 00 00 12 00 00 00 00 00 00 | .....9.....
000430: D7 C1 29 67 00 00 00 00 D7 C1 29 67 00 00 00 00 | (??)g...(??)g...
000440: D7 C1 29 67 00 00 00 00 2E 00 00 00 00 00 00 00 | (??)g.....
000450: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000470: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000480: 00 00 00 00 00 00 00 00 00 74 00 00 00 00 00 00 | .....t.....
000490: 1D 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004B0: 00 00 00 00 00 00 00 00 00 39 00 00 00 00 00 00 | .....9.....
0004C0: FB BC 29 67 00 00 00 00 FB BC 29 67 00 00 00 00 | (??)g...(??)g...
0004D0: FB BC 29 67 00 00 00 00 2E 2E 00 00 00 00 00 00 | (??)g.....
0004E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....


000500: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000510: 00 00 00 00 00 00 00 00 00 74 00 00 00 00 00 00 | .....t.....
000520: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000530: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000540: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000550: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000560: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000570: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000580: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000590: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

0005B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

<code>Extent location[EXTENT_ ARRAY_SIZE];</code>	Size = 10 x 4 Bytes 1st value is 0x 00 09 & 00 01 blockCount = 0x 00 09 = 9 blockLocation = 0x 00 01 = 1
<code>unsignedlong size;</code>	Size = 0x 00 00 00 00 00 00 12 00 = 4608
<code>time_t accessed;</code>	0x 00 00 00 00 67 29 C1 D7 = 1730789847 = Mon Nov 04 2024 22:57:27
<code>time_t modified;</code>	0x 00 00 00 00 67 29 C1 D7 = 1730789847 = Mon Nov 04 2024 22:57:27
<code>time_t created;</code>	0x 00 00 00 00 67 29 C1 D7 = 1730789847 = Mon Nov 04 2024 22:57:27
<code>Char name[65];</code> 	0x 00 2E = 46 = 'E'
<code>Char isDr;</code>	0x74 = 116 = 't'

## 2. A description of the VCB structure



```
typedef struct VolumeControlBlock
{
    Extent freeSpaceMapLocation[EXTENT_ARRAY_SIZE];
    Extent rootLocation[EXTENT_ARRAY_SIZE];
    long long signature;
    unsigned int totalBytes;
    unsigned int blockSize;
    unsigned short totalBlocks;
} VolumeControlBlock;
```

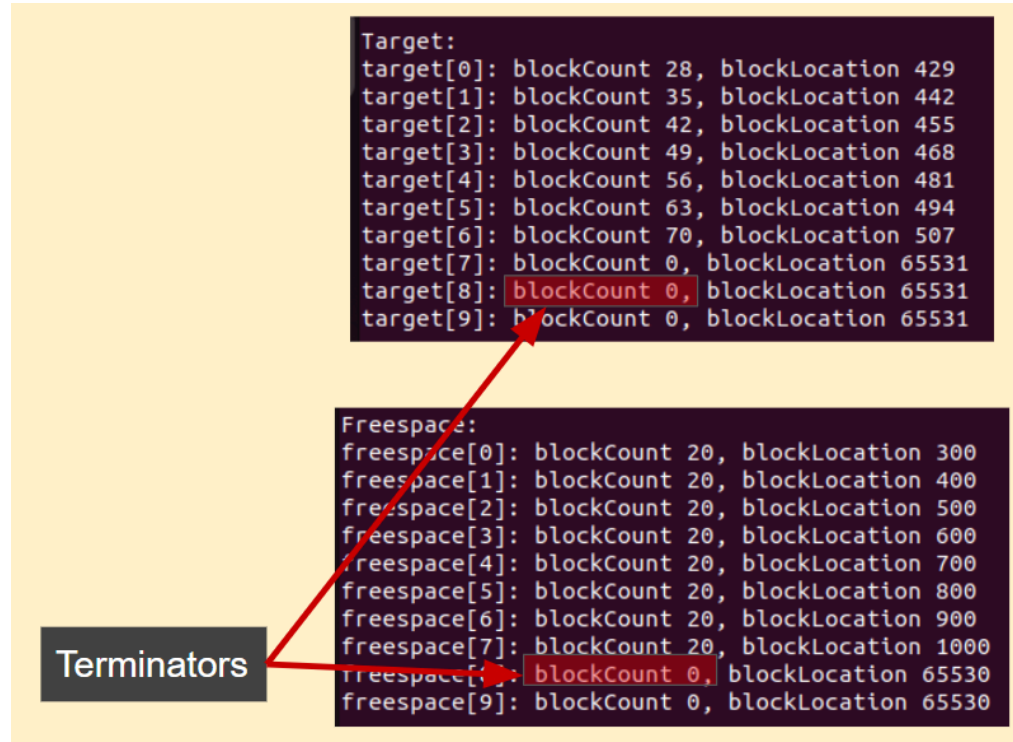
- a.
- b. We are using two Extents for our free space allocation. The first Extent array is a location of freespace. The second is the location of the Extents. The Signature of 4206980085 to check that the VCB is ours. totalBlocks is the amount of blocks in our volume. blockSize is the size of our blocks, typically 512. totalBytes is to keep track fo the total bytes in our volume.

### 3. A description of the Free Space structure

- a. We are using Extents for our free space allocation. To do this, we have a [struct Extent](#) that defines a row of contiguous blocks.
- b. Extent contains a blockLocation for where on disk this extent starts, and blockCount that describes how many contiguous blocks there are.

```
typedef struct Extent
{
    short blockCount;
    unsigned short blockLocation;
} Extent;
```

- c.
- d. We can use an array of Extents to define which blocks are allocated to a directory, or which blocks are allocated to free space.
- e. Our arrays are terminated by blockCount being 0. We would never allocate an extent of blockCount = 0, so we are safe to use this as our terminator 🤖



f.

```
#define EXTENT_ARRAY_SIZE 10
```

g.

Our array size is not dynamic, and is a macro called EXTENT\_ARRAY\_SIZE. This array size is the default for *most* of our extent arrays, and we use this size for allocating blocks for DirectoryEntries (files, directories, and blobs) and the freespace map in VCB.

h.

```
typedef struct VolumeControlBlock
{
    Extent freeSpaceMapLocation[EXTENT_ARRAY_SIZE];
    Extent rootLocation[EXTENT_ARRAY_SIZE];
    long long signature;
    unsigned int totalBytes;
    unsigned int blockSize;
    unsigned short totalBlocks;
} VolumeControlBlock;
```

i.

```
typedef struct DirectoryEntry
{
    Extent location[EXTENT_ARRAY_SIZE];
    unsigned long size;
    time_t accessed;
    time_t modified;
    time_t created;
    char name[65];
    char isDir;
} DirectoryEntry;
```

- j. When we exceed the array size of 10, and we need more blocks, we set the last array value's blockCount to -1, and blockLocation to the location of a new block that we will write more extents to.
- k. If we fill that entire block, we create another block called a tertiary block, that contains an array of block address that each contain an array of extents. We then go back and change the -1 to be -2, and change the location to the tertiary's location.

```
Freespace:
freespace[0]: blockCount 20, blockLocation 300
freespace[1]: blockCount 20, blockLocation 400
freespace[2]: blockCount 20, blockLocation 500
freespace[3]: blockCount 20, blockLocation 600
freespace[4]: blockCount 20, blockLocation 700
freespace[5]: blockCount 20, blockLocation 800
freespace[6]: blockCount 20, blockLocation 900
freespace[7]: blockCount 20, blockLocation 1000
freespace[8]: blockCount 20, blockLocation 1100
freespace[9]: blockCount -1, blockLocation 100

Go to block 100

block[0]: blockCount 10, blockLocation 500
block[1]: blockCount 10, blockLocation 700
block[2]: blockCount 10, blockLocation 900
block[3]: blockCount 10, blockLocation 1100
block[4]: blockCount 10, blockLocation 1300
block[5]: blockCount 10, blockLocation 1500
block[6]: blockCount 10, blockLocation 1700
block[7]: blockCount 10, blockLocation 1900
block[8]: blockCount 10, blockLocation 2100
block[9]: blockCount 10, blockLocation 2300
block[10]: blockCount 10, blockLocation 2500
block[11]: blockCount 10, blockLocation 2700
block[12]: blockCount 10, blockLocation 2900
block[13]: blockCount 10, blockLocation 3100
block[14]: blockCount 10, blockLocation 3300
block[15]: blockCount 10, blockLocation 3500
block[16]: blockCount 10, blockLocation 3700
block[17]: blockCount 10, blockLocation 3900
block[18]: blockCount 10, blockLocation 4100
block[19]: blockCount 10, blockLocation 4300
block[20]: blockCount 10, blockLocation 4500

Block 100 has more extents
```

- l.
- m.

#### 4. A description of the Directory system

- a. Our [directory entry struct](#) allows us to track the metadata for a file or a directory.

We want to track the location which is done using Extents for our space allocation. We have a variable for name so that the user can interact with the file system. We also like to track if it is a file or a directory for proper handling of the entry.

```
typedef struct DirectoryEntry
{
    Extent location[EXTENT_ARRAY_SIZE];
    unsigned long size;
    time_t accessed;
    time_t modified;
    time_t created;
    char name[65];
    char isDir;
} DirectoryEntry;
```

createDir()

Return type is a directory or more simply an array of directory entries

Parameters

1. entriesWanted - defines how many entries the user wants to create.

2. parent - defines the parent of the directory the function will create.

Description- createDir() will take in the amount of entries wanted for the new directory and ensure that we allocate enough blocks for the amount of entries we want, once we calculated the blocks, we then can ensure we are not wasting any space in the blocks by filling the block to the brim with directory entries. At the same time we use the parent parameter to define the “.” and “..” parent and self directories and then we call the function passing in the new directory to be written to disk writeDir(newDir) before returning the newdir back to the caller.

writeDir()

void return type

Parameters

1. Directory\_Entry\* - A directory to be written to disk

Description- will take in a directory and ensure we calculate the correct amount of blocks to write and will use the first, aka [0], directory entry of the to write to the correct location then will write to disk return to caller.

## 5. A table of who worked on which components

Jacob	Avinh	Hilary	Lita	Casey
<ul style="list-style-type: none"> <li>- Directory Structure</li> <li>- Volume Control block initialization</li> <li>- Writeup</li> </ul>	<ul style="list-style-type: none"> <li>- Free Space</li> <li>- Freespace tests</li> <li>- All the .h files</li> <li>- Globals</li> <li>- Directory Structure</li> <li>- Writeup</li> </ul>	<ul style="list-style-type: none"> <li>- Directory Structure</li> <li>- Hexdump Analysis</li> <li>- Minimizing internal padding in structs</li> <li>- Writeup</li> </ul>	-	Debugging

**6. How did our team work together, how often we met, how did we meet, how did we divide up the tasks.**

- a. How did your team work together, how often you met, how did you meet, how did you divide up the tasks.
  - i. We initially had one meeting in person with the whole group to decide between FAT and Extent file systems, and we chose extent. We didn't have scheduled weekly meetings or anything like that. The meetings were just suggested as the due date grew closer. We mostly met online with Lita and Casey, while Jacob, Avinh, and Hilary met in person. We didn't divide up the tasks well; people just chose what they would work on.

**7. A discussion of what issues we faced and how our team resolved them.**

- a. A discussion of what issues you faced and how your team resolved them.
  - i. We just kept pulling all nighters
  - ii. A lot of jacob juice

## Milestone 2: File System Operations

This milestone focuses on implementing core file system operations, with a focus on the ability to manipulate and interact with directories and files, specifically integrating `fsshell` to support basic commands like `mkdir`, `ls`, `pwd`, and `cd`. Below is a detailed breakdown of how these operations should be implemented, including key tasks and the functions that need to be developed.

---

### Key File System Operations to Implement

**1. `fs_setcwd`**

- **Purpose:** Sets the current working directory to the specified path.
- **Status:** Almost complete; needs checks and minor tweaks (e.g., handling of errors and path validation).
- **Steps:**
  - Parse the given path using `parsePath`.
  - Ensure that the directory exists and is valid.
  - Load the directory entry and set the new current working directory (`g_cwd`).

- Update the global current working directory path (`g_cwd_path`), ensuring both absolute and relative paths are handled.
  - Perform checks for validity (e.g., path not pointing to a file).
  - 2. **fs\_getcwd**
    - **Purpose:** Returns the current working directory.
    - **Status:** Completed.
    - **Steps:**
      - Simply return the `g_cwd_path` which holds the current directory.
  - 3. **fs\_isFile**
    - **Purpose:** Checks if a path points to a file.
    - **Status:** Completed.
    - **Steps:**
      - Parse the given path using `parsePath`.
      - Find the matching entry in the directory and check the `isDir` flag.
      - Return `1` if the file is found and is a regular file (`isDir == 'F'`), otherwise return `0`.
  - 4. **fs\_isDir**
    - **Purpose:** Checks if a path points to a directory.
    - **Status:** Completed.
    - **Steps:**
      - Parse the path and check if the entry is a directory (`isDir == 'T'`).
      - Return `1` if the path is a directory, otherwise return `0`.
  - 5. **fs\_mkdir**
    - **Purpose:** Creates a new directory at the specified path.
    - **Status:** Completed.
    - **Steps:**
      - Parse the path to ensure that the parent directory exists.
      - Allocate space for the new directory entry and create the directory structure.
      - Add the new directory to the parent directory and update the global file system structure.
- 

## Functions to Implement

These functions are essential to handle directory operations and to complete the command-line interface for managing files and directories.

1. **fs\_opendir**
  - **Purpose:** Opens a directory for reading.
  - **Steps:**
    - Validate the directory path using `parsePath`.
    - Load the directory entry and prepare to iterate through its contents.

- Return a pointer to an `fdDir` struct, which will hold information about the directory, including the position in the directory and a pointer to the directory entries.

## 2. `fs_readdir`

- **Purpose:** Reads the next entry from an open directory.
- **Steps:**
  - Retrieve the current entry from the open directory and fill the `fs_diriteminfo` struct.
  - Increment the directory entry position to point to the next entry.
  - Return the filled `fs_diriteminfo` struct. If there are no more entries, return `NULL`.

## 3. `fs_closedir`

- **Purpose:** Closes an open directory.
- **Steps:**
  - Free any resources associated with the `fdDir` struct.
  - Set the `directory` pointer to `NULL` and deallocate memory.

## 4. `fs_stat`

- **Purpose:** Returns the status of a file or directory (similar to `stat` in Unix).
- **Steps:**
  - Parse the path to retrieve the directory entry.
  - Populate the `fs_stat` struct with information such as access time, modification time, size, and block size.
  - Return the populated `fs_stat` struct.



## Interrelationships Between Functions

The core interactions between these functions can be visualized as follows:

- Directory Management: Functions like `fs_opendir`, `fs_readdir`, and `fs_closedir` form a directory handling subsystem. `fs_opendir` opens a directory, `fs_readdir` reads its contents, and `fs_closedir` frees resources when done.
- Path Management: Functions like `fs_setcwd`, `fs_getcwd`, `fs_isFile`, and `fs_isDir` rely on parsing the path and identifying whether it's a file or directory. These functions interact with the directory structure to provide information about file system state.
- File Manipulation: `fs_mkdir`, `fs_delete`, and `fs_rmdir` manipulate files and directories by adding or removing entries in the file system.

## Conclusion

By implementing these functions, we will provide essential file system capabilities that are necessary for the shell to interact with directories and files. Functions for manipulating directories, obtaining file information, and managing the current working directory will be crucial for the next steps in developing the file system's command-line interface.

### Milestone 3:

---

#### Key File System Operations to Implement

1. **b\_open**

Purpose: Open existing file, create it if not

- First checks to see if there is a file or not
- If not, check for the CREAT flag and write a new file to the parent directory in an open location
- If true, check for APPEND or TRUNC flags to either append to the end of a file or truncate it
- Allocate the internal buffer
- Set up variables for the FCB
- Returns the current FD in the FCB Array

2. **b\_read**

Purpose: Write from external buffer to internal file

- First checks the file to see if it can even read from it
- First, it fills into an internal buffer from the provided buffer by the end user
- This internal buffer calls LBAread in order to read it from the file location
- Copies all data from our internal buffer to the external buffer
- Returns total amount of bytes read

3. **b\_write**

Purpose: Write from internal file to external buffer

- First, checks the file to see if it can write from it
- Set buffer to chunk size if data does not fit
- Reset the buffer to remove old characters
- Copy to internal buffer from external buffer
- This internal buffer calls LBAwrite in order to write it to the file location
- Updates any internal variables that track the buffer and file
- Update modified time, since the file has been modified
- Return total number of bytes written

4. **b\_seek**

Purpose: Change file index based on whence value

- Takes a file descriptor, offset, and whence
- Uses the value of whence to determine the file index
- Return file index

5. **b\_close**

Purpose: Frees buffer

- Frees the buffer inside the FCB
- 

#### Functions to Implement

1. touch  
Purpose: create new file
  - touch implements b\_open to create this new file on the SampleVolume
2. cat  
Purpose: print file
  - cat implements b\_open and b\_read in order to read a file from SampleVolume
  - cat only has a limited functionality
3. cp2fs  
Purpose: copy file from Linux System to SampleVolume
  - cp2fs implements b\_open and b\_write to write to a file from the Linux system to the SampleVolume
  - cp2fs implements open() read() and close() as well
4. cp2l  
Purpose: copy file from SampleVolume to Linux System
  - cp2l implements b\_open and b\_read to write to a file from the SampleVolume to the Linux system
  - cp2l implements open() write() and close() as well
5. cp  
Purpose: copy one file to another
  - cp implements b\_open to open both the source and destination files
  - cp implements b\_read from the source file, which b\_writes into the destination file

## Interrelationships Between Functions

Both CP, CP2L and CP2FS use b\_read() and/or b\_write() to copy files between each other. Every single function uses b\_open() to open a file, before calling any other b\_ commands.

## Conclusion

Many functions here use b\_read and b\_write to both copy from an initial buffer and to an external buffer, and b\_read and b\_write are often used to copy in this way, and when files are transferred on the linux system, they can also be written to the SampleVolume. All functions here use b\_io in order to accomplish what they are required to do.

## Issues and Resolutions:

### General Issues:

#### Time\_t:

When we were creating our Directory Entry struct, we got a comment suggesting to us to do timestamps and in class we watched bierman add time\_t as a part of the DE struct. However, I had never used this before so I looked it up and it seems that time\_t is a struct that we can include in our DE, so we did that 😊



Standard C99:

```
9  #include <time.h>

time_t t0 = time(0);
// ...
time_t t1 = time(0);
double datetime_diff_ms = difftime(t1, t0) * 1000.;

clock_t c0 = clock();
// ...
clock_t c1 = clock();
double runtime_diff_ms = (c1 - c0) * 1000. / CLOCKS_PER_SEC;
```

The precision of the types is implementation-defined, ie the datetime difference might only return full seconds.

Share Improve this answer Follow 16 10 2000 14734 16 10 2000 14746

<https://stackoverflow.com/questions/1444428/time-stamp-in-the-c-programming-language>

After this, we needed to split our code into multiple different .h files so that it is readable and everyone knows all of the function prototypes to use for later. We decided on 3 .h files for milestone 1: fsInit.h, dir.h, and freespace.h. We concluded this because we want our files to be loosely coupled, and if we develop our files correctly, the files should be interchangeable with other file systems. This is following the principle that despite the fact we are doing Ext, the other systems shouldn't care about the backend of our freespace.c, and should be able to make directories, read files, etc. regardless of what file format we have.

After creating our .h files, we ran into the following bug that VolumeControlBlock was not found despite being imported. This was because we had a circular dependency through all of our .h files, where fsInit.h dependent on freespace.h, and freespace depended on fsInit. To get around this, we created a forward declaration of VolumeControlBlock in freespace.h

```
student@student:~/code/csc415-filessystem-Jacob9610$ make
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -c -o fsInit.o fsInit.c -g -I.
In file included from fsInit.h:18,
                 from fsInit.c:27:
freespace.h:40:19: error: unknown type name 'VolumeControlBlock'
   40 | int initFreeSpace(VolumeControlBlock *vcv, int numFreeBlocks, int initBlockLocation);
      |                   ^
make: *** [Makefile:58: fsInit.o] Error 1
student@student:~/code/csc415-filessystem-Jacob9610$
```

```
24
25 // #ifndef VCB
26 // #define VCB
27 typedef struct VolumeControlBlock VolumeControlBlock;
28 // #endif
29
```

### Could not initialize freespace:

After pushing a supposedly working dir.c and dir.h file I pulled the code and ran it it worked fine. Then I deleted the SampleVolume to remake it and it could not initialize the file system

```
student@student:~/CSC 413/csc415-filessystem-Jacob9610$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does not exist, errno = 2
File SampleVolume not good to go, errno = 2
Block size is : 512
Created a volume with 9999872 bytes, broken into 19531 blocks of 512 bytes.
Opened SampleVolume, Volume Size: 9999872; BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
ERROR: Could not initialize file systemSystem exiting
|-----|
|----- Command -----| Status |
| ls                  | OFF  |
| cd                  | OFF  |
| md                  | OFF  |
| pwd                 | OFF  |
| touch               | OFF  |
| cat                 | OFF  |
| rm                  | OFF  |
| cp                  | OFF  |
| mv                  | OFF  |
| cp2fs               | OFF  |
| cp2l                | OFF  |
|-----|
Prompt >
```

The fix was that we were returning false when initFreeSpace worked which is not what we want.

```
if (vcv->signature != MY_SIGNATURE)
{
    if (!initFreeSpace(vcb, numBlocks, 1)) return -1;
}
60 if (vcv->signature != MY_SIGNATURE)
61 {
62     printf("VCB Signature doesn't match\n");
63     if (!initFreeSpace(vcb, numBlocks, 1)) return -1;
64     printf("Initialized Free Space\n");
}
```

We kept running into issues where we couldn't initialize the global block size (g\_blocksize). This was because it was being called before it was declared in a different file.

```
fsshell: malloc.c:2617: sysmalloc: Assertion `(old_top == initial_top (av) && old_size == 0) || ((unsigned long) (old_size) >= MINSIZE && prev_inuse (old_top) && ((unsigned long) old_end & (pagesize - 1)) == 0)' failed.  
make: *** [Makefile:67: run] Aborted (core dumped)
```

```
if (initVCB(numberOfBlocks, blockSize)) {  
    printf("ERROR: Could not initialize file system");  
    exitFileSystem();  
};  
  
g_blockSize = blockSize;
```

I currently have the issue that the whole way allocate blocks is coded is very bad. I have it working for block iterations of extents, but I have small edge cases that my code is not prepared for. Allocate blocks is iterating over the free space array, checking if that could fit in the target, appending it to the end of the extent array if it does, and then popping the last value of the freespace to be replaced into the free space array. All of this iterates over the entirety of 3 different extent arrays for every extent being added; this is very slow.

#### Global Variables showing undefined:

We created a global variable for both the root and the current working directory and as such we were not expecting any problems from them as they would always be in memory, but as we were programming other functions that use these functions is that when we load our volume control block from memory we actually never load into memory our `g_root` and `g_cwd`. We believed that it would take two lines of code to set them into memory but found that in order to load the Directory into memory that we would have to allocate the size of the directory which led to our next problem of having to `LBaread` our Directory entry then from there would be able to calculate the correct number of blocks we are going to have to read from disk for the full entry. The fix looks like this. Note this error only occurred after our first run meaning when we first initialized the volume control block as we set both `g_cwd` and `g_root` when initializing the VCB.

```
DirectoryEntry* rootDirectoryEntry = malloc(g_blockSize);
if (rootDirectoryEntry == NULL) {
    perror("Failed to allocate memory for rootDirectoryEntry");
    exit(EXIT_FAILURE);
}

// Read the first block of the root directory
LBRead(rootDirectoryEntry, 1, vcb->rootLocation[0].blockLocation);

// Calculate the total number of blocks required
int blocks = (rootDirectoryEntry[0].size + (g_blockSize - 1)) / g_blockSize;

// Allocate the correct amount of memory for the entire root directory
DirectoryEntry* rootDirectoryEntries = malloc(blocks * g_blockSize);
if (rootDirectoryEntries == NULL) {
    perror("Failed to allocate memory for rootDirectoryEntries");
    free(rootDirectoryEntry);
    exit(EXIT_FAILURE);
}

// Read the entire root directory into memory
LBRead(rootDirectoryEntries, blocks, vcb->rootLocation[0].blockLocation);

// Assign the root and current working directory pointers
g_cwd = rootDirectoryEntries;
g_root = rootDirectoryEntries;

// Print information for debugging
printf("blocks: %d \n", blocks * g_blockSize);
printf("the cwd at init: %s\n", g_cwd->name);
printf("the root at init: %s\n", g_root->name);

// Free the temporary allocation for the first block
free(rootDirectoryEntry);
```

Bierman's Baddies  
Avinh Huynh, Hilary Lui, Jacob Vazquez,  
Lita Hernandez-Gonzalez, Casey Steven

CSC415 Operating Systems  
Github: [Jacob9610](#)

## **Freespace Issues:**



## Directory Issues:

### Make Dr not updating the parent dir with child info

When I was coding up the mldr function I found that the name of the directory was not being updated by the function it was only changing the name of it in the new dir even then it was not being updated at the disk. This was due to only having line 38 shown below.

```
35
36     DirectoryEntry * newDir = createDir(30, parent);
37
38     //strcpy(newDir->name, lastElementName);
39
40     //find an empty element
41     //i need to update the value of the new dir into an unused index in parse path
42     int numEntries = parent[0].size/sizeof(DirectoryEntry);
43
44     for (int i = 0; i < numEntries; i++)
45     {
46
47         int isUsed = dirEntryUsed(&(parent[i]));
48         printf("is used: %d \n", isUsed);
49         if( isUsed== 1 )
50         {
51             parent[i] = *newDir;
52             strcpy(parent[i].name, lastElementName);
53             printf("new entry neme in parent : %s \n ", parent[i].name);
54             break;
55         }
56     }
57
58
59     writeDir(parent);
60     // char * setwdpath = fs_getcwd("/dope", 4096);
61
62
63     return 0;
```

once I figured that the parent needed to be updated i went and found an open space for my directory entry and updated the values in that space with the new directory items such as the name of the directory and the location and the rest of the meta data for the directory lines shown above from 42 to 59.

### fs\_readdir not implemented correctly

```
struct fs_diritemInfo* fs_readdir(fdDir *dirp)
{
    // idea here is to fill a dirItemInfo struct with the data from disk that is
    // being referred to in the args to this fn. we want to fill this de
    // struct so we can access the data from the dir. the arg. asks for.
    // i think could be wrong but its my best guess.

    printf("\n in fs_readdir\n");
    fs_diritemInfo *info = malloc(sizeof(fs_diritemInfo));
    DirectoryEntry* dirBuf = loadDir(dirp->directory); // this feels unnecessary as its already loaded into memory double

    // increment dir position
    dirp->dirEntryPosition++;

    info->d_reclen = dirp->d_reclen;
    if (dirBuf->isDir == 'T')
    {
        info->fileType = FT_DIRECTORY;
    }
    else if (dirBuf->isDir == 'F')
    {
        info->fileType = FT_REGFILE;
    }
    else
    {
        printf("dirbuf.isdir is neither T or F\n");
    }

    strcpy(info->d_name, dirBuf[dirp->dirEntryPosition].name);

    return info;
}
```

Here I reinitialized the directory into dirBuf but as I came to my senses and was less sleep deprived I learned that I already had the directory loaded in ram within the dirp that was passed I just needed to access it. I also ended up doing an extra malloc for the diritemInfo named info as I was unsure how to use it so I created an instance of it within my scope which may have been just a waste of memory but I ended up not updating the other values for the infostruct. i also was not setting the info->di if it was neither a file nor a dir to null which led to an infinite loop set in the fs\_shell file shown in the photo below.

```
dirp->directory[dirp->dirEntryPosition]34601
dirp->directory[dirp->dirEntryPosition].isDiris neither T or F
yea here
  in fs_readdir
dirp->directory[dirp->dirEntryPosition]34602
dirp->directory[dirp->dirEntryPosition].isDiris neither T or F
yea here
  in fs_readdir
dirp->directory[dirp->dirEntryPosition]34603
dirp->directory[dirp->dirEntryPosition].isDiris neither T or F
yea here
  in fs_readdir
dirp->directory[dirp->dirEntryPosition]34604
dirp->directory[dirp->dirEntryPosition].isDiris neither T or F
yea here
  in fs_readdir
dirp->directory[dirp->dirEntryPosition]34605
dirp->directory[dirp->dirEntryPosition].isDiris neither T or F
yea here
  in fs_readdir
dirp->directory[dirp->dirEntryPosition]34606
dirp->directory[dirp->dirEntryPosition].isDiris neither T or F
yea here
  in fs_readdir
dirp->directory[dirp->dirEntryPosition]34607
dirp->directory[dirp->dirEntryPosition].isDiris neither T or F
yea here
  in fs_readdir
dirp->directory[dirp->dirEntryPosition]34608
dirp->directory[dirp->dirEntryPosition].isDiris neither T or F
yea here
  in fs_readdir
dirp->directory[dirp->dirEntryPosition]34609
dirp->directory^Cmake: *** [Makefile:67: run] Interrupt
```

```
struct fs_diriteminfo* fs_readdir(fdDir *dirp)
{
    // idea here is to fill a dirItemInfo struct with the data from disk that is
    // being referred to in the args to this fn. we want to fill this de
    // struct so we can access the data from the dir. the arg. asks for.
    //i think could be wrong but its my best guess.

    printf("\n in fs_readdir\n");

    // increment dir position

    printf("dirp->directory[dirp->dirEntryPosition]d\n", dirp->dirEntryPosition);

    if (dirp->directory[dirp->dirEntryPosition].isDir == 'T')
    {
        dirp->di->fileType = FT_DIRECTORY;
        strcpy( dirp->di->d_name, dirp->directory[dirp->dirEntryPosition].name);
        dirp->di->d_reclen = 5;//dirp->directory[dirp->dirEntryPosition].size/sizeof(fdDir);//i dont like this at all
    }
    else if (dirp->directory[dirp->dirEntryPosition].isDir == 'F')
    {
        dirp->di->fileType = FT_REGFILE;
        strcpy( dirp->di->d_name, dirp->directory[dirp->dirEntryPosition].name);
        dirp->di->d_reclen = 5;// dirp->directory[dirp->dirEntryPosition].size/sizeof(fdDir);//i dont like this at all
    }
    else
    {
        printf("dirp->directory[dirp->dirEntryPosition].isDir is neither T or F\n");
        dirp->di = NULL;
    }

    dirp->dirEntryPosition++;
    return dirp->di;
}
```

While I am still working I do think that the fix for the infinite loop by setting `dirp->di` to null and the better use of the `dirp` deserve to be noted. Here I have simplified the function to reuse already established structs in my favor, reducing the amount of memory used.

future fixes to this are to figure the correct use of the `reclen`th.

I would also like to make sure that the loop does not stop early because I have `dirp->di` stopping when it finds an unused directory entry. My current path of thoughts is telling me that the `reclen`th is the num of entries and if I get to the end of the directory, set the next place to null and return `dirp->di` to null.

mv issues-----

SO many issues

- same parents = misunderstanding of parsepath, used loader to make new entry to reference
- it only deleted the file and didnt move it and i was focusing on the for loop but then as i was explaining my issue i release i never wrotedir the new entry i made :D

nested dir issues-----

making a dir in a dir then trying to cd into is broken it only recognizes the first character of the nested dir and when we cd into it there are 400+ entries inside.

root has two dirs dir1 and dir2:

cd into dir1 and md dir3:

```
Prompt > ls
in fs_opendir
index: 0 (T)
index: 1 (T) index: 2 (T) dir1
index: 3 (T) dir2
index: 4 is free
index: 5 is free
index: 6 is free
index: 7 is free
index: 8 is free
index: 9 is free
index: 10 is free
index: 11 is free
index: 12 is free
index: 13 is free
index: 14 is free
index: 15 is free
index: 16 is free
index: 17 is free
index: 18 is free
index: 19 is free
index: 20 is free
index: 21 is free
index: 22 is free
index: 23 is free
index: 24 is free
index: 25 is free
index: 26 is free
index: 27 is free
index: 28 is free
index: 29 is free
index: 30 is free
index: 31 is free
index: 32 is free
in fs_closedir
Prompt >
```

```
Prompt > cd dir1
lastElementName: dir1
Current working directory updated to: /dir1/
Prompt > ls
in fs_opendir
index: 0 (T)
index: 1 (T) index: 2 is free
index: 3 is free
index: 4 is free
index: 5 is free
index: 6 is free
index: 7 is free
index: 8 is free
index: 9 is free
index: 10 is free
index: 11 is free
index: 12 is free
index: 13 is free
index: 14 is free
index: 15 is free
index: 16 is free
index: 17 is free
index: 18 is free
index: 19 is free
index: 20 is free
index: 21 is free
index: 22 is free
index: 23 is free
index: 24 is free
index: 25 is free
index: 26 is free
index: 27 is free
index: 28 is free
index: 29 is free
index: 30 is free
index: 31 is free
index: 32 is free
in fs_closedir
Prompt >
```

cd into dir3 inside dir1:

```
index: 414 is free
index: 415 is free
index: 416 is free
index: 417 is free
index: 418 is free
index: 419 is free
index: 420 is free
index: 421 is free
index: 422 is free
index: 423 is free
index: 424 is free
index: 425 is free
index: 426 is free
index: 427 is free
index: 428 is free
index: 429 is free
index: 430 is free
index: 431 is free
index: 432 is free
index: 433 is free
index: 434 is free
index: 435 is free
index: 436 is free
index: 437 is free
index: 438 is free
index: 439 is free
index: 440 is free
index: 441 is free
index: 442 is free
index: 443 is free
index: 444 is free
index: 445 is free
index: 446 is free
index: 447 is free
index: 448 is free
index: 449 is free
index: 450 is free
index: 451 is free
index: 452 is free
index: 453 is free
index: 454 is free
index: 455 is free
in fs_closedir
Prompt > pwd
/dir1/d/
Prompt >
```

returns that I've cd into a directory called d in

dir1, which has 455 entries when I ls. No current fix

I suspect that it is this line 272

```
263 // print size of g_cwd_path
264 printf("Size of g_cwd_path: %ld\n", sizeof(g_cwd_path));
265 if (pathname[0] == '/') {
266     // Absolute path
267     printf("Absolute path with g_cwd_path: %s\n", g_cwd_path);
268     strncpy(g_cwd_path, pathname, sizeof(g_cwd_path) - 1);
269 } else {
270     // Relative path
271     printf("Relative path with g_cwd_path: %s\n", g_cwd_path);
272     strcat(g_cwd_path, pathname, sizeof(g_cwd_path) - strlen(g_cwd_path) - 1); // I think it's this line
273 }
274 g_cwd_path[sizeof(g_cwd_path) - 1] = '\0';
```

Fixed it was line 272 and also 274 I'm not sure the purpose of those lines because I didn't write this function, but I took out 274 and for 272 i changed it from `strncat(g_cwd_path, pathname,1)` to just `strcat(g_cwd_path, lastElement)` because parth path should have already isolated the directory we wanted.

## B\_io issues:

Issue 1 was that the code was originally making files on the disk using `open()`, `read()` and `write()` to get the File Descriptor instead of using the `DirectoryEntry` struct. The code should have used `LBRead` and `LBWrite` in place of `read()` and `write()` to write to the `SampleVolume` and not the `LFS`. Both of them write 1 block at a time to `SampleVolume`

Issue 2 was with `CP2L` and `CP2FS`, originally it did not recognize the `S_IRUSR` (Read by owner) in the permissions. `Fsshell.c` imports `mfs.h`, which imports `b_io.h`. So in `b_io.h` I made sure to include the header file for `<sys/stat.h>`, which includes the permission, fixing the issue.

Issue 3 stemmed from the `Cat` command, which was not correctly reading the file. I reworked `b_write` to clear the buffer before running, which fixed the issue, not leaving any leftover data anymore.

Issue 4 stemmed from `CP2FS` and `CP2L` only copying the last buffer to the file. I reworked `b_write` to use `b_seek` to find the current byte end of a file using `lseek()` and will then write to the end of the previous file. This led to a second issue where it was then reading the first buffer, and a few characters at the end of the end buffer. This was caused by using `B_CHUNK_SIZE` in many places where `count` should have been used instead, as `count` is called as 200 in the function by `CP2FS`, `CP2L` and `cat`. 200 is lower than the chunk size of 512, so it would overwrite because of that reason, so fixing the math to use `count` instead of `B_CHUNK_SIZE` fixed the issue.

```
fsshell: malloc.c:2617: sysmalloc: Assertion `(old_top == initial_top (av) && old_size == 0) || ((unsigned long) (old_size) >= MINSIZE && prev_inuse (old_top) && ((unsigned long) old_end & (pagesize - 1)) == 0)' failed.
make: *** [Makefile:67: run] Aborted (core dumped)
student@student:~/csc415-filesystem-Jacob9610$
```

Issue 5 - `Malloc` did not properly work when trying to create multiple `Extents` for the same file, and also the previous “fix” for Issue 4 was not a fix and ended up overwriting our file system. I ended up having to revert to before the previous “fix”.

`md` makes it's own directory appear in `ls`-----  
ls after freshly calling `md dir1` and `cd` into `dir1`:                      ls after calling `md dir2` in `dir1`:

```
Current working directory updated to: dir1  
Prompt > ls
```

```
in fs_opendir
```

```
in fs_closedir  
Prompt > █
```

**ISSUE IN PROGRESS**

```
in fs_opendir
```

```
dir1  
dir2
```

```
in fs_closedir  
Prompt > █
```



## Analysis:

What my file system looks like:

Root:

```
Prompt > ls
```

```
dir1  
dir2  
file3
```

```
Prompt > pwd
```

```
/
```

```
Prompt >
```

dir1:

```
Prompt > ls
```

```
file1  
file2
```

```
Prompt > pwd
```

```
/dir1/
```

```
Prompt >
```

dir2:

```
Prompt > ls
```

```
Prompt > pwd
```

```
/dir2/
```

```
Prompt >
```

## VolumeControlBlock: 1st block

```
student@student:~/CSC 413/csc415-filesystem-Jacob9610$ Hexdump/hexdump.linux SampleVolume --start 1 --count 15
Dumping file SampleVolume, starting at block 1 for 15 blocks:

000200: 30 4C 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 | 0L.....
000210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000220: 00 00 00 00 00 00 00 00 09 00 01 00 00 00 00 00 | .....
000230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000250: F5 6B C1 FA 00 00 00 00 00 96 98 00 00 02 00 00 | k.....
000260: 4B 4C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | KL.....
000270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

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

```
typedef struct VolumeControlBlock
{
    Extent freespace[EXTENT_ARRAY_SIZE];
    Extent rootLocation[EXTENT_ARRAY_SIZE];
    long long signature;
    unsigned int totalBytes;
    unsigned int blockSize;
    unsigned short totalBlocks;
} VolumeControlBlock;
```

```
typedef struct Extent
{
    short blockCount;
    unsigned short blockLocation;
} Extent;
```

000200:	30 4C 1C 00 00 00 00 00	00 00 00 00 00 00 00 00	0L.....
000210:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000220:	00 00 00 00 00 00 00 00	09 00 01 00 00 00 00 00	.....
000230:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000240:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000250:	F5 6B C1 FA 00 00 00 00	00 96 98 00 00 02 00 00	♦k♦♦.....♦♦.....
000260:	4B 4C 00 00 00 00 00 00	00 00 00 00 00 00 00 00	KL.....
000270:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000280:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000290:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0002A0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0002B0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0002C0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0002D0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0002E0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0002F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000300:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000310:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000320:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000330:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000340:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000350:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000360:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000370:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000380:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000390:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0003A0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0003B0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0003C0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0003D0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0003E0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0003F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....

	Address	Type	Offset	Size	Hex	Native
Extent freespace [EXTENT_ARRAY_SIZE]	000200-000227	Extent *	0	Extent = 4 Bytes Extent_array_size = 10 Total = 4x10 = 40 Bytes	0x 0000 0000 0000 0000 0000 0000 0000 0000 001C 4C30	1st value is 0x001C 4C30 blockCount = 0x 4C 30 = 19504 blockLocation = 0x 00 1C = 28
Extent rootLocation[EXTENT_ARRAY_SIZE];	000228-00024A	Extent *	8	Extent = 4 Bytes Extent_array_size = 10 Total = 4x10 = 40 Bytes	0x 0000 0000 0000 0000 0000 0000 0000 0000 0001 0009	1st value is 0x 00 09 & 00 01 blockCount = 0x 00 09 = 9 blockLocation = 0x 00 01 = 1
long long signature;	000250-000257	long long	0	8 Bytes	0x 00 00 00 00 FA C1 6B F5	= 4206980085
unsigned int totalBytes;	000258-00025B	unsigned int	4	4 Bytes	0x00 98 96 00	= 9999872
unsigned int blockSize;	00025C-00025F	unsigned int	8	4 Bytes	0x00 00 02 00	= 512
unsigned short totalBlocks;	000260-000261	unsigned short	12	2 Bytes	0x4C 4B	= 19531

## DirectoryEntries: 2nd-3rd block

### 2nd Block:

000400:	09 00 01 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000410:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000420:	00 00 00 00 00 00 00 00	00 12 00 00 00 00 00 00	.....
000430:	21 4D 51 67 00 00 00 00	21 4D 51 67 00 00 00 00	!MQg....!MQg....
000440:	21 4D 51 67 00 00 00 00	2E 00 00 00 00 00 00 00	!MQg.....
000450:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000460:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000470:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000480:	00 00 00 00 00 00 00 00	00 54 00 00 00 00 00 00	.....T.....
000490:	09 00 01 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0004A0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0004B0:	00 00 00 00 00 00 00 00	00 12 00 00 00 00 00 00	.....
0004C0:	21 4D 51 67 00 00 00 00	21 4D 51 67 00 00 00 00	!MQg....!MQg....
0004D0:	21 4D 51 67 00 00 00 00	2E 2E 00 00 00 00 00 00	!MQg.....
0004E0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0004F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000500:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000510:	00 00 00 00 00 00 00 00	00 54 00 00 00 00 00 00	.....T.....
000520:	09 00 0A 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000530:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000540:	00 00 00 00 00 00 00 00	00 12 00 00 00 00 00 00	.....
000550:	2C 4D 51 67 00 00 00 00	2C 4D 51 67 00 00 00 00	,MQg....,MQg....
000560:	2C 4D 51 67 00 00 00 00	64 69 72 31 00 00 00 00	,MQg....dir1....
000570:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000580:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000590:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0005A0:	00 00 00 00 00 00 00 00	00 54 00 00 00 00 00 00	.....T.....
0005B0:	09 00 13 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0005C0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0005D0:	00 00 00 00 00 00 00 00	00 12 00 00 00 00 00 00	.....
0005E0:	2F 4D 51 67 00 00 00 00	2F 4D 51 67 00 00 00 00	/MQg..../MQg....
0005F0:	2F 4D 51 67 00 00 00 00	64 69 72 32 00 00 00 00	/MQg....dir2....

### 3rd block:

000600:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000610:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000620:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000630:	00 00 00 00 00 00 00 00	00 54 00 00 00 00 00 00	.....T.....
000640:	8B 21 BB 21 06 00 00 00	00 00 00 00 00 00 00 00	◆!◆!.....
000650:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000660:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000670:	52 4D 51 67 00 00 00 00	52 4D 51 67 00 00 00 00	RMQg....RMQg....
000680:	52 4D 51 67 00 00 00 00	66 69 6C 65 33 00 00 00	RMQg....file3...
000690:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0006A0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0006B0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0006C0:	00 00 00 00 00 00 00 00	00 46 00 00 00 00 00 00	.....F.....

```

0006D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

000700: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000710: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000720: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000730: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000740: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000750: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000760: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000770: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000780: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000790: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

1st highlights	Address	Type	Offset	Size	Hex	Native
Extent location[EXTENT_ARRAY_SIZE];	000400-000427	Extent *	0	Extent = 4 Bytes Extent_array_size = 10 Total = 4x10 = 40 Bytes	0x 0000 0000 0000 0000 0000 0000 0000 0000 0001 0009	1st value is 0x001C 4C30 blockCount = 0x 00 09 = 9 blockLocation = 0x 00 01 = 1
unsigned long size;	000428-00042F	unsigned long	8	8 Bytes	0x 0000 0000 0000 1200	= 4608
time_t accessed;	000430-000437	time_t	0	8 Bytes	0x 00 00 00 00 67 51 4D 21	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
time_t modified;	000438-00043F	time_t	8	8 Bytes	0x00 98 96 00	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
time_t created;	000440-000447	time_t	0	8 Bytes	0x00 00 02 00	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
char name[65];	000448-000488	char *	8	65 Bytes	0x000000000000 000000000000 000000000000	= .

					000000000000 000000000000 0000002E	
char isDir;	449	char	9	1 Byte	0x54	= 'T'

3rd highlights	Address	Type	Offset	Size	Hex	Native
Extent location[EXTENT _ARRAY_SIZE];	000400- 000427	Extent *	0	Extent = 4 Bytes Extent_array_size = 10 Total = 4x10 = 40 Bytes	0x 0000 0000 0000 0000 0000 0000 0000 0000 0001 0009	1st value is 0x001C 4C30 blockCount = 0x 00 09 = 9 blockLocation = 0x 00 01 = 1
unsigned long size;	000428- 00042F	unsigned long	8	8 Bytes	0x 0000 0000 0000 1200	= 4608
time_t accessed;	000430- 000437	time_t	0	8 Bytes	0x 00 00 00 00 67 51 4D 21	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
time_t modified;	000438- 00043F	time_t	8	8 Bytes	0x00 98 96 00	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
time_t created;	000440- 000447	time_t	0	8 Bytes	0x00 00 02 00	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
char name[65];	000448- 000488	char *	8	65 Bytes	0x000000000000 000000000000 000000000000 000000000000 000000000000 0000002E	= ..
char isDir;	449	char	9	1 Byte	0x54	= 'T'

4th highlights	Address	Type	Offset	Size	Hex	Native
Extent location[EXTENT _ARRAY_SIZE];	000400- 000427	Extent *	0	Extent = 4 Bytes Extent_array_size = 10 Total = 4x10 = 40 Bytes	0x 0000 0000 0000 0000 0000 0000 0000 0000 0001 0009	1st value is 0x001C 4C30 blockCount = 0x 00 09 = 9 blockLocation = 0x 00 01 = 1
unsigned long size;	000428- 00042F	unsigned long	8	8 Bytes	0x 0000 0000 0000 1200	= 4608
time_t accessed;	000430- 000437	time_t	0	8 Bytes	0x 00 00 00 00 67 51 4D 21	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
time_t modified;	000438- 00043F	time_t	8	8 Bytes	0x00 98 96 00	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
time_t created;	000440- 000447	time_t	0	8 Bytes	0x00 00 02 00	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
char name[65];	000448- 000488	char *	8	65 Bytes	0x000000000000 000000000000 000000000000 000000000000 000000000000 0000002E	= dir1
char isDir;	449	char	9	1 Byte	0x54	= 'T'

1st highlights	Address	Type	Offset	Size	Hex	Native
Extent location[EXTENT _ARRAY_SIZE];	000400- 000427	Extent *	0	Extent = 4 Bytes Extent_array_size = 10 Total = 4x10 = 40 Bytes	0x 0000 0000 0000 0000 0000 0000 0000 0000 0001 0009	1st value is 0x001C 4C30 blockCount = 0x 00 09 = 9 blockLocation = 0x 00 01 = 1
unsigned long size;	000428- 00042F	unsigned long	8	8 Bytes	0x 0000 0000 0000 1200	= 4608



time_t accessed;	000430- 000437	time_t	0	8 Bytes	0x 00 00 00 00 67 51 4D 21	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
time_t modified;	000438- 00043F	time_t	8	8 Bytes	0x00 98 96 00	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
time_t created;	000440- 000447	time_t	0	8 Bytes	0x00 00 02 00	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
char name[65];	000448- 000488	char *	8	65 Bytes	0x0000000000 000000000000 000000000000 000000000000 000000000000 00000002E	= dir2
char isDir;	449	char	9	1 Byte	0x54	= 'T'

5th highlights	Address	Type	Offset	Size	Hex	Native
Extent location[EXTENT _ARRAY_SIZE];	000400- 000427	Extent *	0	Extent = 4 Bytes Extent_array_size = 10 Total = 4x10 = 40 Bytes	0x 0000 0000 0000 0000 0000 0000 0000 0000 0001 0009	1st value is 0x001C 4C30 blockCount = 0x 00 09 = 9 blockLocation = 0x 00 01 = 1
unsigned long size;	000428- 00042F	unsigned long	8	8 Bytes	0x 0000 0000 0000 1200	= 4608
time_t accessed;	000430- 000437	time_t	0	8 Bytes	0x 00 00 00 00 67 51 4D 21	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
time_t modified;	000438- 00043F	time_t	8	8 Bytes	0x00 98 96 00	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM
time_t created;	000440- 000447	time_t	0	8 Bytes	0x00 00 02 00	= 1733381409 = Wednesday, December 4, 2024 10:50:09 PM

char name[65];	000448-000488	char *	8	65 Bytes	0x0000000000 000000000000 000000000000 000000000000 000000000000 0000002E	= file3
char isDir;	449	char	9	1 Byte	0x54	= 'F'

## Screen shot of compilation:

make:

```
student@student:~/CSC 413/csc415-filesystem-Jacob9610$ make
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -c -o fsInit.o fsInit.c -g -I.
gcc -c -o freespace.o freespace.c -g -I.
gcc -c -o freespace_test.o freespace_test.c -g -I.
gcc -c -o dir.o dir.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 freespace.o freespace_test.o dir.o mfs.o b_io.o fsLow.o -g -I. -lm -l readline -l pthread
student@student:~/CSC 413/csc415-filesystem-Jacob9610$ make run
```

## Screen shot(s) of the execution of the program:

make run

```
student@student:~/CSC 413/csc415-filesystem-Jacob9610$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
|-----|
|----- Command -----| Status |
| ls                      | ON   |
| cd                      | ON   |
| md                      | ON   |
| pwd                    | ON   |
| touch                  | ON   |
| cat                    | ON   |
| rm                     | ON   |
| cp                     | ON   |
| mv                     | ON   |
| cp2fs                  | ON   |
| cp2l                   | ON   |
|-----|
Prompt > 
```



## CP

## MV

```
Prompt > touch fire
Prompt > ls

fire

~/Documents/csc415-filesystem-Jacob9610/free

Prompt > md dir
```

Bierman's Baddies

Avinh Huynh, Hilary Lui, Jacob Vazquez,  
Lita Hernandez-Gonzalez, Casey Steven

CSC415 Operating Systems

Github: [Jacob9610](#)

```
Prompt > md dir  
allocating  
Prompt > mv fire dir  
Prompt > cd dir  
Prompt > ls
```

```
fire
```

```
Prompt > █
```

<pre>Prompt &gt; ls hello foo directioataraes wreerre</pre>	<pre>Prompt &gt; md figuirinos Prompt &gt; ls hello figuirinos foo directioataraes wreerre</pre>
<pre>Prompt &gt;</pre>	<pre>Prompt &gt;</pre>

## RM

```
Prompt > ls
```

```
hello  
figuirinos  
foo  
directioatares  
wreerre
```

```
Prompt > rm hello  
Prompt > ls
```

```
figuirinos  
foo  
directioatares  
wreerre
```

```
Prompt > rm hello  
Prompt > ls
```

```
figuirinos  
foo  
directioatares  
wreerre
```

```
Prompt >
```



## TOUCH

```
Prompt > ls
```

```
figuirinos  
foo  
directioataraes  
wreerre
```

```
Prompt > touch fefefe  
Prompt > ls
```

```
fefefe  
figuirinos  
foo  
directioataraes  
wreerre
```

```
Prompt > touch fefefe  
Prompt > ls
```

```
fefefe  
figuirinos  
foo  
directioataraes  
wreerre
```

```
Prompt >
```

## CAT

Doesn't work

## CP2L

```
student@student:~/code/csc415-filesystem-Jacob9610$ ls
Baddiez_Bierman_FileSystem_Writeup.pdf  dir.o          freespace_test.o  fsLow.o          Makefile          SampleVolume
b_io.c                                  freespace.c     fsInit.c          fsshell          mfs.c
b_io.h                                  freespace.h     fsInit.h          fsshell.c        mfs.h
b_io.o                                  freespace.o     fsInit.o          fsshell.o        mfs.o
dir.c                                   freespace_test.c fsLow.h           global.h          milestones
dir.h                                   freespace_test.h fsLowM1.o         Hexdump          README.md
```

```
Prompt > ls
```

```
fefefe
figuirinos
foo
directioataraes
wreerre
testcp2l
demo
```

```
Prompt > cp2l demo
Prompt > S
```

```
student@student:~/code/csc415-filesystem-Jacob9610$ ls
Baddiez_Bierman_FileSystem_Writeup.pdf  dir.h          freespace_test.h  fsLowM1.o        Hexdump          README.md
b_io.c                                  dir.o          freespace_test.o  fsLow.o          Makefile          SampleVolume
b_io.h                                  freespace.c     fsInit.c          fsshell          mfs.c
b_io.o                                  freespace.h     fsInit.h          fsshell.c        mfs.h
demo                                   freespace.o     fsInit.o          fsshell.o        mfs.o
dir.c                                   freespace_test.c fsLow.h           global.h          milestones
student@student:~/code/csc415-filesystem-Jacob9610$
```

Added demo to linux

## Screen shot(s) of the Analysis hexdump:

Dumping file SampleVolume, starting at block 1 for 15 blocks:

```
000200: 30 4C 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 | 0L.....
000210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000220: 00 00 00 00 00 00 00 00 09 00 01 00 00 00 00 00 | .....
000230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000250: F5 6B C1 FA 00 00 00 00 00 96 98 00 00 02 00 00 | k.....
000260: 4B 4C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | KL.....
000270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0002F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

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

000400: 09 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000410: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000420: 00 00 00 00 00 00 00 00 00 12 00 00 00 00 00 00 | .....
000430: 21 4D 51 67 00 00 00 00 21 4D 51 67 00 00 00 00 | !MQg...!MQg...
000440: 21 4D 51 67 00 00 00 00 2E 00 00 00 00 00 00 00 | !MQg.....
```

```

000450: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000470: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000480: 00 00 00 00 00 00 00 00 00 54 00 00 00 00 00 00 | .....T.....
000490: 09 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004B0: 00 00 00 00 00 00 00 00 00 12 00 00 00 00 00 00 | .....
0004C0: 21 4D 51 67 00 00 00 00 21 4D 51 67 00 00 00 00 | !MQg....!MQg....
0004D0: 21 4D 51 67 00 00 00 00 2E 2E 00 00 00 00 00 00 | !MQg.....
0004E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

000500: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000510: 00 00 00 00 00 00 00 00 00 54 00 00 00 00 00 00 | .....T.....
000520: 09 00 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000530: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000540: 00 00 00 00 00 00 00 00 00 12 00 00 00 00 00 00 | .....
000550: 2C 4D 51 67 00 00 00 00 2C 4D 51 67 00 00 00 00 | ,MQg....,MQg....
000560: 2C 4D 51 67 00 00 00 00 64 69 72 31 00 00 00 00 | ,MQg....dir1....
000570: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000580: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000590: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005A0: 00 00 00 00 00 00 00 00 00 54 00 00 00 00 00 00 | .....T.....
0005B0: 09 00 13 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005D0: 00 00 00 00 00 00 00 00 00 12 00 00 00 00 00 00 | .....
0005E0: 2F 4D 51 67 00 00 00 00 2F 4D 51 67 00 00 00 00 | /MQg..../MQg....
0005F0: 2F 4D 51 67 00 00 00 00 64 69 72 32 00 00 00 00 | /MQg....dir2....

000600: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000610: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000620: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000630: 00 00 00 00 00 00 00 00 00 54 00 00 00 00 00 00 | .....T.....
000640: 8B 21 BB 21 06 00 00 00 00 00 00 00 00 00 00 00 | ?!?!.....
000650: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000670: 52 4D 51 67 00 00 00 00 52 4D 51 67 00 00 00 00 | RMQg....RMQg....
000680: 52 4D 51 67 00 00 00 00 66 69 6C 65 33 00 00 00 | RMQg....file3...
000690: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006C0: 00 00 00 00 00 00 00 00 00 46 00 00 00 00 00 00 | .....F.....
0006D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```
000700: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000710: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000720: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000730: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000740: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000750: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000760: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000770: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000780: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000790: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0007F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000800: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000810: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000820: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000830: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000840: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000850: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0008A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0008B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0008C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0008D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0008E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0008F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000900: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000910: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000920: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000930: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000940: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000950: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000960: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000970: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000980: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000990: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```

0009A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0009B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0009C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0009D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0009E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0009F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

000A00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000A10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000A20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000A30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000A40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000A50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000A60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000A70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000A80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000A90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000AA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000AB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000AC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000AD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000AE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000AF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

000B00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000B10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000B20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000B30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000B40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000B50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000B60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000B70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000B80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000B90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000BA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000BB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000BC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000BD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000BE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000BF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

000C00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000C10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000C20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000C30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

000C40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000C50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000C60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000C70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000C80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000C90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000CA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000CB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000CC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000CD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000CE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000CF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

000D00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000D10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000D20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000D30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000D40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000D50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000D60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000D70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000D80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000D90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000DA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000DB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000DC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000DD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000DE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000DF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

000E00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000E10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000E20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000E30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000E40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000E50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000E60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000E70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000E80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000E90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000EA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000EB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000EC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000ED0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000EE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```
000EF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000F00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000F10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000F20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000F30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000F40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000F50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000F60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000F70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000F80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000F90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000FA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000FB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000FC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000FD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000FE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
000FF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
0010A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
0010B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
0010C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
0010D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
0010E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
0010F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```



```

001190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0011A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0011B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0011C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0011D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0011E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0011F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

001200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0012A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0012B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0012C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0012D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0012E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0012F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

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

```

```

001400: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001410: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001420: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

001430: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001440: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001450: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001470: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001480: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001490: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0014A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0014B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0014C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0014D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0014E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0014F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

001500: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001510: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001520: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001530: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001540: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001550: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001560: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001570: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001580: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001590: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0015A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0015B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0015C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0015D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0015E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0015F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

001600: 09 00 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001610: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001620: 00 00 00 00 00 00 00 00 00 12 00 00 00 00 00 00 | .....
001630: 2C 4D 51 67 00 00 00 00 2C 4D 51 67 00 00 00 00 | ,MQg...,MQg....
001640: 2C 4D 51 67 00 00 00 00 2E 00 00 00 00 00 00 00 | ,MQg.....
001650: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001680: 00 00 00 00 00 00 00 00 00 54 00 00 00 00 00 00 | .....T.....
001690: 09 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0016A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0016B0: 00 00 00 00 00 00 00 00 00 12 00 00 00 00 00 00 | .....
0016C0: 21 4D 51 67 00 00 00 00 21 4D 51 67 00 00 00 00 | !MQg...!MQg....
0016D0: 21 4D 51 67 00 00 00 00 2E 2E 00 00 00 00 00 00 | !MQg.....

```

```

0016E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0016F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

001700: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001710: 00 00 00 00 00 00 00 00 00 54 00 00 00 00 00 00 | .....T.....
001720: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001730: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001740: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001750: 52 4D 51 67 00 00 00 00 52 4D 51 67 00 00 00 00 | RMQg....RMQg....
001760: 52 4D 51 67 00 00 00 00 66 69 6C 65 31 00 00 00 | RMQg....file1...
001770: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001780: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001790: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0017A0: 00 00 00 00 00 00 00 00 00 46 00 00 00 00 00 00 | .....F.....
0017B0: 8B 21 BB 21 06 00 00 00 00 00 00 00 00 00 00 00 | ❖!❖!.....
0017C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0017D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0017E0: 52 4D 51 67 00 00 00 00 52 4D 51 67 00 00 00 00 | RMQg....RMQg....
0017F0: 52 4D 51 67 00 00 00 00 66 69 6C 65 32 00 00 00 | RMQg....file2...

001800: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001810: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001820: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001830: 00 00 00 00 00 00 00 00 00 46 00 00 00 00 00 00 | .....F.....
001840: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001850: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0018A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0018B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0018C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0018D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0018E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0018F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

001900: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001910: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001920: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001930: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001940: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001950: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001960: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001970: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

001980: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001990: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0019A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0019B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0019C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0019D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0019E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0019F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

001A00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001A10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001A20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001A30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001A40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001A50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001A60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001A70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001A80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001A90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001AA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001AB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001AC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001AD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001AE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001AF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

001B00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001B10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001B20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001B30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001B40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001B50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001B60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001B70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001B80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001B90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001BA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001BB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001BC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001BD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001BE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001BF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

001C00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001C10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

001C20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001C30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001C40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001C50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001C60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001C70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001C80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001C90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001CA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001CB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001CC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001CD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001CE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001CF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

001D00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001D10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001D20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001D30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001D40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001D50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001D60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001D70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001D80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001D90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001DA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001DB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001DC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001DD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001DE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001DF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```

001E00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001E10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001E20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001E30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001E40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001E50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001E60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001E70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001E80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001E90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001EA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001EB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
001EC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

```

```
001ED0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001EE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001EF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

```
001F00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001F10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001F20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001F30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001F40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001F50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001F60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001F70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001F80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001F90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001FA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001FB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001FC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001FD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001FE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....  
001FF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```