Project 4 Report

Wyatt LeMaster

4/24/2022

COMP 7500

Design:

As can be seen below in the DFD the program is run using data from a disk image named image1.img. This is image is then read into an array and there it is translated into structures when needed.

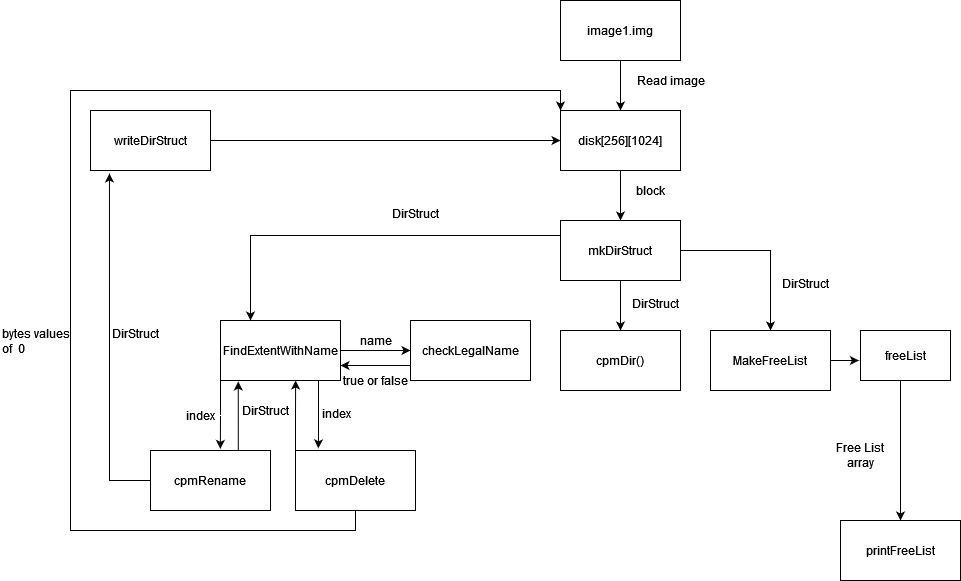
In my implementation I opted to load the block in each function requiring it rather than having it loaded globally. I did this to allow the functions to be run independently of each other or in different orders then presented in the fsysdriver.c. For example if I implemented makeFreeList() to load the block into main memory but then later decided I wanted to call cpmDir() before makeFreeList() this would result in an error due to the block not being loaded. This is why each function is in charge of loading and reading the block themselves.

Besides this change the implementation is fairly simple. Every function relies on mkDirStruct to translate and store the data from block 0 so they can interpret it. If a function needs to write back to the block they can utilize writeDirStruct().

Design Issues:

The greatest design issue I faced was how to handle writing to the block. I had many problems with data being corrupted and overwritten for unknown reason. Many of my functions that require passing a pointer to a structure would result in missing information within the structure such as blocks being flagged as empty or null terminators being removed from strings. I believe I was able to Iron out most of these issues allowing me to get the correct output in my testing.

DFD:



Function Prototypes:

1. DirStructType \*mkDirStruct(int index,uint8\_t \*e);
   1. Function allocates memory for the structure DirStructType and fills it with the contents of block zero pointed to by e at the location denoted by index
2. Void writeDirStruct( DirStructType \*d, uint8\_t index, uint8\_t \*e);
   1. Writes the contents of the structure to the block pointed to by e.
3. void makeFreeList();
   1. makes a list of free and used data blocks
4. void printFreeList();
   1. prints free list for debugging purposes
5. findExtentWithName(char \*name, uint8\_t \*block0);
   1. looks for and returns the extent containing the file name. Function returns index of file.
6. checkLegalName(char \*name);
   1. checks to see if the new name fits the predefined size and allowed contents and returns true or false.
7. cpmDir();
   1. prints the contents of the file directory including file name and sizes
8. cpmRename(char\* oldName, char\* newname)
   1. uses functions findExtentwithName and checkLegalName to find and replace a filename within the directory block
9. cpmDelete(char \* name)
   1. uses functions findExtentWithName to find and set all values to 0. Also clears block in free list.

Lessons Learned:

1. How to better handle memory allocation in functions
2. How to allocate memory, fill that memory then return a pointer to the memory
3. How the basic structure of a file system works. Including how directories are stored
4. How to translate and store data from byte format to structures for use
5. Many different ways of manipulating strings
6. How to be more careful with memory locations, and the issues that arise if one is over written
7. How to read and understand someone else’s code so as to make compatible functions that work alongside the code without modifying it.
8. How to handle and debug memory addresses you cannot see through printing

Program Output:

Text

Description automatically generated

Function error handling:

Checking legal Name: Graphical user interface, text

Description automatically generated

cpmRename:Graphical user interface, text

Description automatically generated  
Graphical user interface, text

Description automatically generated