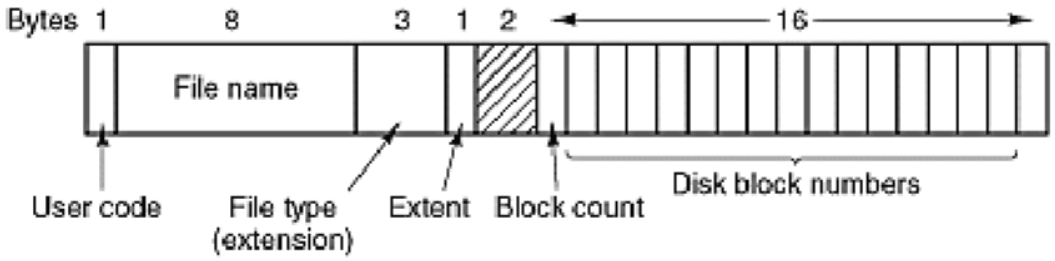
COMP 7500 Advanced Operating Systems
Project 4: cpmFS – A Simple File System

The goal of the project is to implement a simple file system. Objectives are to list directories, rename files, copy files, and remove files. Dr Qin has provided majority of the implementation; our job was to implement cpmfsys.c with 9 functions defined by Dr Qin. Functions are explained in function prototype section.

Design of cpmFS

The CP/M operating system provides 38 system calls, mostly file services for user programs. Important services of these include reading and writing files. To read a file, it must be opened so when CP/M gets an open system call, it must read and search the one and only directory. To save resources, the directory is not kept in memory all the time. When CP/M finds the entry, it immediately has the disk block numbers, since they are stored in directory entry, as well as it contains all the attributes. Format is given below: -



Function Prototypes: -

As per project specification, I have implemented these functions.

1. DirStructType *mkDirStruct(int index,uint8_t *e);
   This function allocates memory for a DirStructType (see above), and populates it, given a pointer to a buffer of memory holding the contents of disk block 0 (e), and an integer index, which tells which extent from block zero (extent numbers start with 0) to use to make the DirStructType value to return.

2. void writeDirStruct(DirStructType *d, uint8_t index,uint8_t *e);
   This function writes contents of a DirStructType struct back to the specified index of the extent in block of memory (disk block 0) pointed to by e

3. void makeFreeList();
   This function populates the FreeList global data structure. freeList[i] == true means that

block i of the disk is free. block zero is never free since it holds the directory. freeList[i] == false means the block is in use.

4. Void printFreeList();
   This is a debugging function, which prints out the contents of the free list in 16 rows of 16, with each row prefixed by the 2-digit hex address of the first block in that row. Denote a used block with a *, a free block with a

5. Void cpmDir();
   This function prints the file directory to stdout. Each filename should be printed on its own line, with the file size, in base 10, following the name and extension, with one space between the extension and the size. If a file does not have an extension, it is acceptable to print the dot anyway, e.g.,"myfile. 234" would indicate a file whose name was myfile, with no extension and a size of 234 bytes. This function returns no error codes since it should never fail unless something is seriously wrong with the disk.

6. bool checkLegalName(char *name);
   It is an internal function, returns true for legal name (8.3 format), false for illegal (name or extension too long, name blank, or illegal characters in name or extension)

7. int findExtentWithName(char *name, uint8_t *block0);
   This is an internal function, which returns -1 for illegal name or name not found; otherwise returns extent nunber 0-31.

8. int cpmDelete (char *name);
   The function deletes the file named name, and frees its disk blocks in the free list

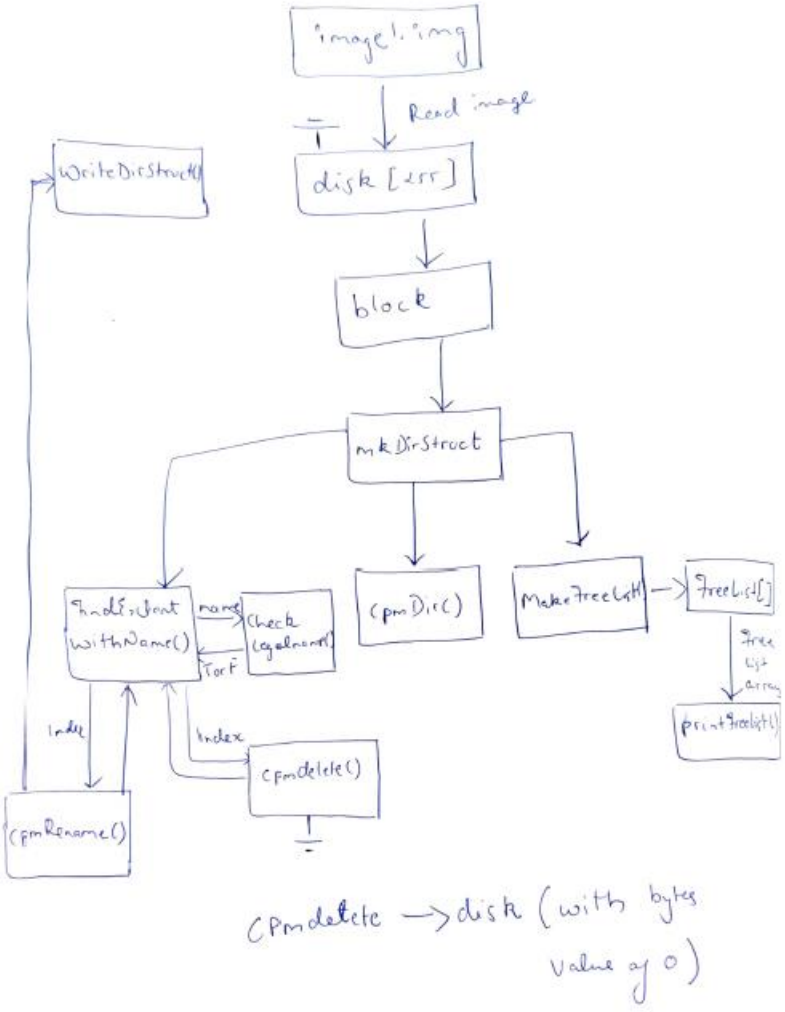
9. int cpmRename (char *oldName, char *newName);
   This function reads directory block, modifies the extent for file named oldName with newName, and write to the disk.

Design Issues: -

- I had issues in writing to the block, for some unknown issue data would be corrupted.
- I had issues with deleting files and couldn’t produce the same output as given by Dr Qin. Each execution was displaying garbage values instead of deleting files.

- While executing, randomly it would display segmentation fault, but next execution would display the correct output. Upon analyzing, noticed that it is because of my VM configuration that caused the problem.

Data Flow Diagram: -



Program Output: -

```
[centos@localhost ~]$ cd project\ 4
[centos@localhost project 4]$ \ls
cpmfsys.c  diskSimulator.c  fsysdriver.c  Makefile
cpmfsys.h  diskSimulator.h  image1.img
[centos@localhost project 4]$ make
gcc -c diskSimulator.c
gcc -c cpmfsys.c
gcc -c fsysdriver.c
gcc -o cpmRun diskSimulator.o cpmfsys.o fsysdriver.o
[centos@localhost project 4]$ ./cpmRun

DIRECTORY LISTING
mytestf1.txt 15874
holefile.txt 1152
shortf.ps  1032
mytestf.    1026

FREE BLOCK LIST: (* Means in-use)
   0:  * . * . * . * . * . * . * . * .
  10:  * . . . . . . . . . . . . . . .
  20:  . . . . . . . . . . . . . . . .
  30:  . . . . . . . . . . . . . . . .
  40:  . . . . . . . . . . . . . . . .
  50:  . . . . . . . . . . . . . . . .
```

```
  60:  . . . . . . . . . . . . . . . .
  70:  . . . . . . . . . . . . . . . .
  80:  . . . . . . . . . . . . . . . .
  90:  . . . . . . . . . . . . . . . .
  a0:  . . . . . . . . . . . . . . . .
  b0:  . . . . . . . . . . . . . . . .
  c0:  . . . . . . . . . . . . . . . .
  d0:  . . . . . . . . . . . . . . . .
  e0:  . . . . . . . . . . . . . . . .
  f0:  . * . * . * . * . * . * . * . *


DIRECTORY LISTING
mytestf1.txt 15874
holefile.txt 1152
mytestf.    1026
cpmRename return code = 0,

DIRECTORY LISTING
mytest2.tx 13826
holefile.txt 1152
mytestv2.x 7170

FREE BLOCK LIST: (* Means in-use)
```

Lessons learned: -

- To handle better memory allocations in functions
- How the basic structure of a file system works including the functioning of directories.
- Working of pointers and filling memory, then how to return a pointer to the memory.
- Most important lesson I learnt was given most of the code, I was tasked to write certain functions. So, I had to study the other files and understand so that I could make compatible functions and work with it.