1 Memory Management

Here are some snippets we will be discussing from `file_system.c`:

Listing 1: Fundamental memory variables

```
char myfs[NUM_BLOCKS][BLOCK_SIZE];  // my file system root
int myfs_f;
int current_block = 3;    // starts allocating from block 3
```

The myfs is a two-dimensional array that has 128 blocks, each of 1024 bytes, or 1 KB. The $myfs_f$ is a file descriptor which will be used for the file system. The $current_block$ is an integer that is used to keep track of the current block that is being allocated. When the data will be updated, the following function will be used to write on the hard disk

Listing 2: Function to write on Hard Disk

```
// Function to write the hard disk state to the "myfs_f"
    file
void writeHardDiskState() {
    int myfs_f = open("myfs", O_CREAT | O_RDWR, 0666);
    if (myfs_f == -1) {
        printf("Error: Cannot create file system myfs");
        exit(1);
    }

    for (int i = 0; i < NUM_BLOCKS; i++) {
        lseek(myfs_f, i * BLOCK_SIZE, SEEK_SET);
        write(myfs_f, myfs[i], BLOCK_SIZE);
    }

    close(myfs_f);
}
```

This function writes the updated array contents into the suitable memory blocks of the file system.

2 Memory Initialization

Below is a small part of the $mem_init()$ function's implementation of how we initialized the data memory

Listing 3: Initializing Data blocks example

```
char Data[BLOCK_SIZE];
memset(Data, 0, BLOCK_SIZE);  // filling the single data
    block with NULL characters

// writing the zeroes to the data blocks from 3 to 127.
for (int i = 3; i < NUM_BLOCKS; i++) {
```

```
6       memcpy(myfs[i], Data, BLOCK_SIZE);
7   }
```

This part deals with creating an array of 1024 bytes and filling it with NULL characters. Then, we copy the contents of this array into the data blocks from 3 to 127. This is done by using the *memcpy*() function.

With regards to the data structure used in the assignment, ideally, a tree would have sufficed; however, the lack of completeness of the functions make a hindrance towards the claim. One of the possible solutions could have been through a linked list as well, as the memory has been managed through pointers, and logically, things pointing to one another would be linked together, thus forming a linked list data structure.