DESIGN RATIONALE

SUMMARY

In this assignment, I have implemented a file-system analogue supporting following operations.

Function Names	Commands
Mkdir	createDirectory
Rmdir	removeDirectory
read	copyFromOtherFileSystem
write	copyIntoOtherFileSystem
dumpe2fs	printFileSystemMeta
Del	removeFile
Ln***	createLink
fsck	fsck_disk
Ln -s	NOT_IMPLEMENTED

****Note about Ln:** **My design does not create hard link between files in the same folder. Example run as below.**
./fileSystemOper mysystem.dat ln /file2.js /usr/file4.js (different folder)

1. Super Blocks and Filesystem meta.

In my file system, I designed a superblock that holds the information about physical disk and the logical parameters like, block size inode size etc. My file system supports multiple **inode** and data-block sizes. Logical parameters and their addresses on the disks are **calculated dynamically** for given inode size and datablock parameters.

```
struct SuperBlock { //48 bytes

      int blocksize; //4
      int diskSize; //4
      int iNodeStartAddress; //4
      int dataBlockStartAddress; //4
      int emptyInodeCount; //4
      int emptyDataBlockCount; //4
      int inodeCount; //4
      int rootAddress; //4
      int emptyInodeAdress; //4
      int emptyDataBlockAddress; //4
      int dentryAdress; //4;
      int blockCount; //4;

};
```

Superblock info is directly written or read from the disk adress 0x0000. There program stores all of the necessary information to access the file system correctly. Similar to Linux VFS **inodes** custom design inodes support 3 level indirection, instead of 12. I use only 1 direct block. The reason behind this choice is to ensure that filesystem consume and release datablocks as fast can it can so that we can observe the fragmantation and resource exhaustion very rapidly. Another intersesting design

choice was the depth of indirection in the inodes. Theoretically, My inodes can support quadruple or even more level of indirection with only changing single line of code.

2. Inode Structure

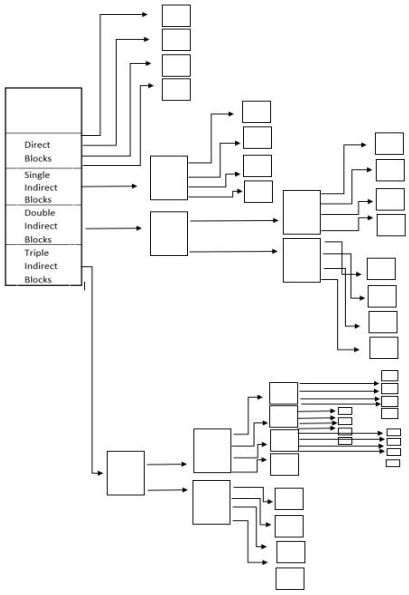


Figure 1: INode data pointer Structure

```
typedef struct inode { //44 byte
      int fileSize = 0;//4
      int id = 0; //4 byte
      int nodeType; //4 byte
      int size = 0; //4 byte
      long lastModificationdate = 0; //8 byte
      int referenceCount = 0; //4 byte
      int references[ADDRESSABLE_DATA_BLOCKS]; //16 byte

 } INODE;
```

A 44 byte iNode structre holds the information about fileSize, inode id, size and data-blocks. As mentioned before inodes have 4 address pointers supoorting where each indirect block can address 4 more address block. **referenceCount** field is used to hold symbolic and hard-links with file system.
I have used;

- 1 direct block pointer: The block# contained in every direct block pointer is for the actual data block that contains the file data.
- 1 single indirect block pointer: One-level indirection implying the indirect block will contain a set of (limited by the size of block) of direct data block numbers
- 1 double indirect block pointer: Two-level indirection implying the indirect block will contain a set of single indirect block numbers. Each of these single indirect blocks will have a set of direct data block numbers.
- 1 triple indirect block pointer: three-level indirection implying the indirect block will contain a set of double indirect block numbers. Each double indirect block will have a set of single indirect block numbers and each indirect block will have a set of direct data block numbers.

This choice allows usage of at most 85 data-blocks.

$$Number\ of\ Blocks = \sum_{n=1}^{DEPTH} IndirectBlock^{(n)} = \frac{IndirectBlock^{(DEPTH+1)}}{IndirectBlock - 1}$$

3.Free blocks, iNodes and Directories and Directory Structure.

When the file-system is formatted with the given block size and inode size, our program calculates the position of each reference in the SuperBlock. And immediately creates a root directory. Using one free inode and datablock. Free data-Blocks and free inodes are pointed out by the superblock. This list is updated both in memory and in the file system whenever an inode or a data-block is **consumed** or **freed**. Operations on these lists are like stack operations and. It behaves as **LIFO**.

Similar to VFS each directory and file consumes an inode and at least one data-block. Furthermore, I also use a data structure called **FileDescriptor** for logical description of a file and physical pointer to the **inode**. In summary **FileDescriptor** holds the filename information type of the descriptor, size(which we use in directory operations.) and the reference inode. In my design, a file can have a name maximum length of **50 character.** For each directory, file system creates two FileDescripor. First one is appended to the file descriptorlist of its parent and the other is kept in the data-block of the recently created folder. Size property of the descriptor shows the number of entries in that directory and I only keep up-to-date for the first entry(Descriptor of the directory). For files or links I only keep one descriptor in the parents folder.

You can follow the rest of this document to understand the file-system in depth.

PART 2

In Part 2 I have implemented formatDisk function in FileSystem.h. And RawDisks class in RawAccess.h has been created to perform reading and writing to disk. FileSystem.h provides both low and hi level file system operations. RawDisks.h emulates low-level file read write operations on raw addresses.

fileSystem->**formatDisk**(blockSize * KILO, freeInodeNumber);

This function calculates block count with given parameters from the console. And then create superBlock and initializes the super block information. (Blocksize. BlockCount, diskSize, inodeCount, emptyInodeCount, emptyDataBlockCount, iNodeStartAddress, emptyInodeAdress, emptyDataBlockAddress, dentryAdress, dataBlockStartAddress, rootAddress, blockCount, emptyDataBlockCount)

struct **SuperBlock** { //48 bytes

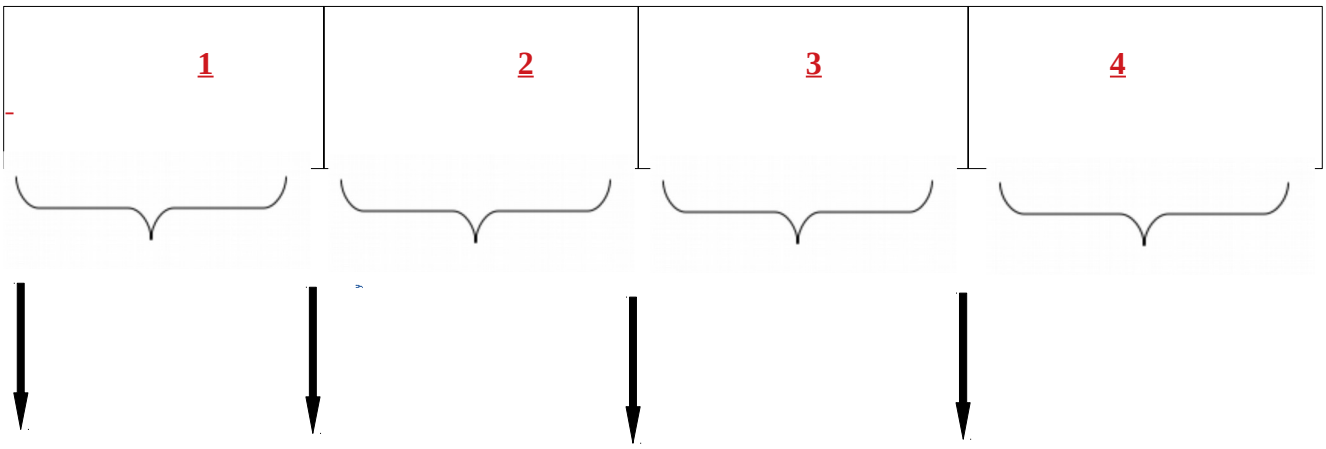
```
int blocksize;
int diskSize;
int iNodeStartAddress;
int dataBlockStartAddress;
int emptyInodeCount;
int emptyDataBlockCount;
int inodeCount;
int rootAddress;
int emptyInodeAdress;
int emptyDataBlockAddress;
int dentryAdress;
int blockCount;
```

};

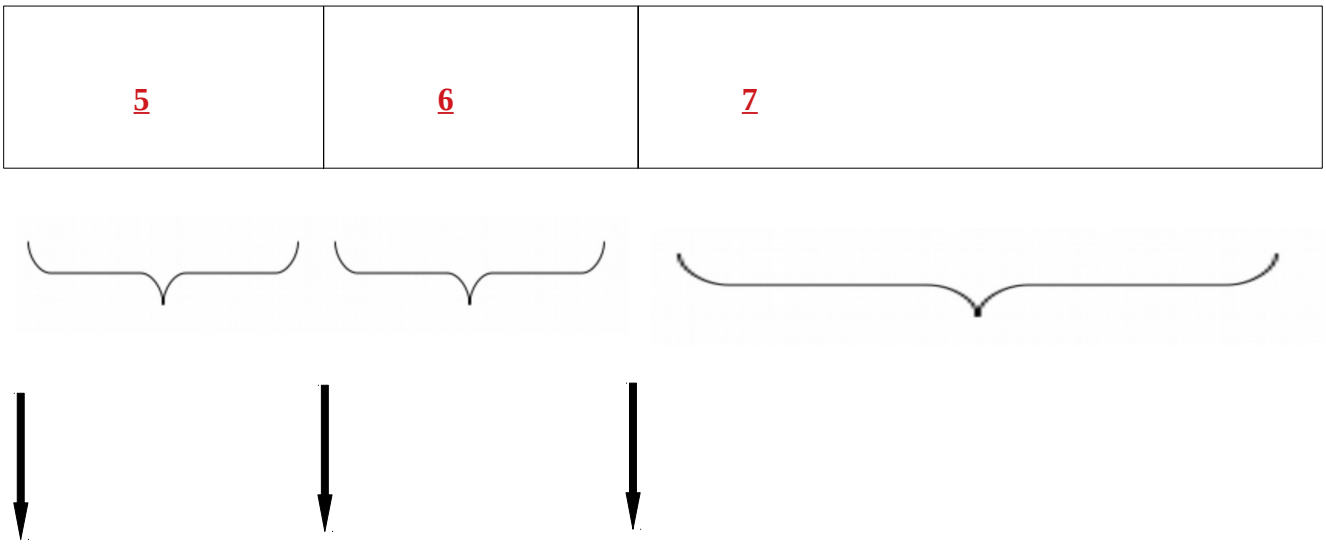
formatDisk function assigns the counts and sets the start addresses. And creates inodes and initializes free iNodes.
int blockCount = diskSize / blockSize;
superBlock.**blocksize** = blockSize;
superBlock.**blockCount** = blockCount;

```
superBlock.diskSize = diskSize;
superBlock.inodeCount = inodeCount;
superBlock.emptyInodeCount = inodeCount;
superBlock.emptyDataBlockCount = superBlock.blockCount;
superBlock.iNodeStartAddress = sizeof(superBlock);
superBlock.emptyInodeAdress = superBlock.iNodeStartAddress
                    + sizeof(INODE) * inodeCount;
superBlock.emptyDataBlockAddress = superBlock.emptyInodeAdress
                    + sizeof(int) * inodeCount;
superBlock.dentryAdress = superBlock.emptyDataBlockAddress
                    + sizeof(int) * blockCount;
int dataBlockStartAddress = (superBlock.dentryAdress
                    + sizeof(int) * blockCount) / blockSize + 1;

superBlock.dataBlockStartAddress = dataBlockStartAddress;
superBlock.rootAddress = dataBlockStartAddress;
superBlock.blockCount = superBlock.blockCount
                    - superBlock.dataBlockStartAddress;
superBlock.emptyDataBlockCount = superBlock.blockCount;
```

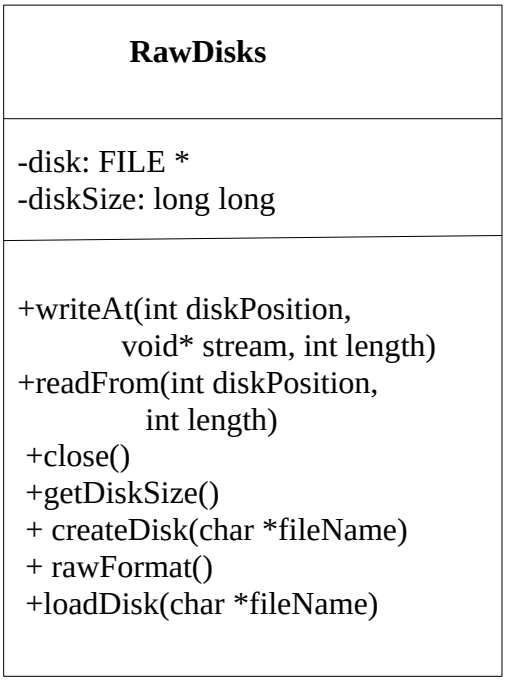


iNodeStartAddress emptyInodeAdress emptyDataBlock Address dentryAdress



dataBlockStart rootAddress emptyDataBlockCount
Address

RawAccess.h



RawDisks class has been created to perform low-level read and write operations to the virtual disk.

- **WriteAt**(): The diskPosition parameter ,that comes with the writeAt function, finds the location to be written with fseek and then writes to the disk as long as the incoming length in bytes.
- **ReadFrom**(): Firstly find the position of the disk wanted to read with fseek and assigns the value of length read from the file to the memory indicated by buffer-memory and return the buffer.
- **Close**(): close the disk.
- **GetDiskSİze**(): returns the size of the disk.
- **createDisk**(char *fileName) : Opens the file with the file name that comes in the parameter and initializes 0 as much as disk space with the rawFormat () function.
- **RawFormat**(): fills the file with 0 as much as disk space.
- **loadDisk**(char *fileName): If the disc was previously created with that name, it loads the created file.

SystemHeaders.h

SystemHeaders.h defines necessary structures for file operations and logical groups of related concepts and definitions.

Structs are:

1) FILE_DESCRIPTOR
2) INODE
3) SuperBlock

```
typedef struct FileDescriptor { //36bytes                    int
parent = 0;
    int type = 0;
    int size = 0;
    int iNode = 0;
    char fileName[MAX_FILE_NAME_LENGTH];
} FILE_DESCRIPTOR;
```

```
#define F_FILE 0
#define F_FOLDER 1
#define F_ROOT 2
#define F_S_LINK 3
#define F_H_LINK 4
```

FileDescriptor struct consist of file parent of file, id of node, inode type, size, and file name.
I define maximum name length of a file 50.(MAX_FILE_NAME_LENGTH])
In the FileDescriptor struct, the type value can take the values F_FOLDER, F_FILE, F_ROOT, F_H_LINK, F_S_LINK.

- It can be distinguished whether it is a file or a folder with these F_FOLDER and F_FILE values.
- When finding the parent of a file, it is checked if it is F_ROOT. F_ROOT is the root folder. If not, the parent call continues.
- F_H_LINK and F_S_LINK are used when creating a link.

```
typedef struct inode { //44 byte
    int fileSize = 0;
    int id = 0;
    int nodeType;
    int size = 0;
    long lastModificationdate = 0;
    int referenceCount = 0;
    int references[ADDRESSABLE_DATA_BLOCKS];

} INODE;
```

```
#define EMPTY -1
```

Inode struct consists of file size, id of node, nodeType, size, last modification date, reference count and references of data blocks.

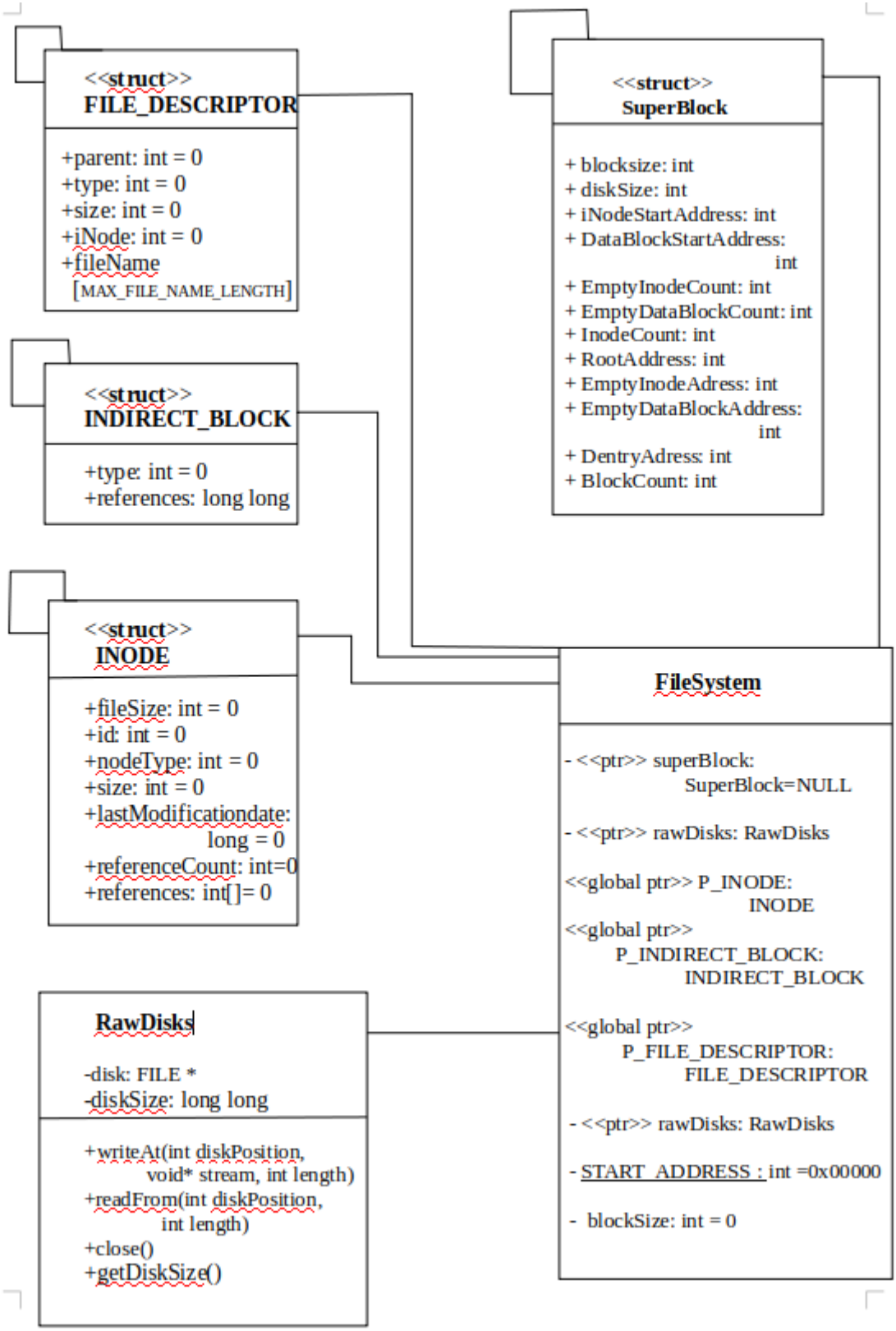
The nodeType value can take the value EMPTY.

- It can be distinguished whether it is empty or not.

The referenceCount value increase with the creating link and creating folder.

SystemUtils.h

In systemUtils.h there is SemiDynamicArray class and parsePath function.
SemiDynamicArray is like ArrayList. I use it for utilization function like calculating dynamic addresses or short-term memory operations



SCREENSHOTS OF THE PROGRAM

1) Creates an file system for PART2

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_2_Program$ ./makeFileSystem 4 400
 mysystem.dat

## WELCOME TO THE FILE SYSTEM ##
## PLEASE ENTER THE FOLLOWING COMMAND ##

Disk Size => 1048576
Block Size is => 4096
 Block Count is => 250
iNode Count is => 400
Empty iNode Count is => 399
Empty Data Block Count is => 249
iNode Start Address is => 48
Empty iNode Start Address is => 19248
Data Block Start Address is => 6
Empty Data Block Start Address is => 20848
Dentry Address is => 21872
Root Address is => 6
File System Info
_____________________________________________________________
*** Shows Hard link
Total File count is 0
Total Folder count is 0
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_2_Program$
```

2) Trying the create an file system for PART2 with wrong parameter. Program gives a warning.

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_2_Program$ ./makeFileSystem 4 400

## WELCOME TO THE FILE SYSTEM ##
## PLEASE ENTER THE FOLLOWING COMMAND ##

 Invalid number of argument !

~makeFileSystem - To initialize the file system
~Example :: ./makeFileSystem [blockSize] [numberOfFreeInodes] [fileName]
~Example :: ./makeFileSystem 4 400 mySystem.dat
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_2_Program$
```

PART3 SCREENSHOTS

3) For PART3, Firstly I want to get information about the file system. as we see in the figure below, total file count is 0 and total folder count is 0. Inode count is 400 which is given with parameter in part2.

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat dumpe2fs

Disk Size => 1048576
Block Size is => 4096
 Block Count is => 250
iNode Count is => 400
Empty iNode Count is => 399
Empty Data Block Count is => 249
iNode Start Address is => 48
Empty iNode Start Address is => 19248
Data Block Start Address is => 6
Empty Data Block Start Address is => 20848
Dentry Address is => 21872
Root Address is => 6
File System Info
_____________________________________________________________
*** Shows Hard link
Total File count is 0
Total Folder count is 0
```

4) In the figure below, First I create a folder usr with mkdir. Then I display the information about system. Here we can see usr is created and total folder count is increased. And also empty inode count is decreased.

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat mkdir /usr
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat dumpe2fs

Disk Size => 1048576
Block Size is => 4096
 Block Count is => 250
iNode Count is => 400
Empty iNode Count is => 398
Empty Data Block Count is => 248
iNode Start Address is => 48
Empty iNode Start Address is => 19248
Data Block Start Address is => 6
Empty Data Block Start Address is => 20848
Dentry Address is => 21872
Root Address is => 6
File System Info
/usr                                            1         1
_____________________________________________________________
*** Shows Hard link
Total File count is 0
Total Folder count is 1
```

5) In the figure below, I create another folder under usr with mkdir. Then I display the information about system. Here we can see deneme1 is created under usr and total folder count is increased. And also empty inode count, empty data block count is decreased.

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat mkdir /usr/deneme1
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat dumpe2fs

Disk Size => 1048576
Block Size is => 4096
 Block Count is => 250
iNode Count is => 400
Empty iNode Count is => 397
Empty Data Block Count is => 247
iNode Start Address is => 48
Empty iNode Start Address is => 19248
Data Block Start Address is => 6
Empty Data Block Start Address is => 20848
Dentry Address is => 21872
Root Address is => 6
File System Info
/usr                                        1         1
/usr/deneme1                                2         2

_____________________________________________________
*** Shows Hard link
Total File count is 0
Total Folder count is 2
```

6) In the figure below, I list contents of the directory /. Here we can see there is . , .. and usr under /

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat mkdir /usr/deneme1/deneme2
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat list /

File Name                       Parent Inode    Type    Inode   Size   Last Modification Date
.                                    0           2       0       0    30 05 2020 21:24:36
..                                   0           2       0       0    30 05 2020 21:24:36
usr                                  0           1       1       0    30 05 2020 21:24:36
```

7) In the figure below, I also list contents of the directory /usr and usr/deneme1. Here we can see the correct result as I created.

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat mkdir /usr/deneme1/deneme2
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat list /

File Name                       Parent Inode    Type    Inode   Size   Last Modification Date
.                                    0           2       0       0    30 05 2020 21:24:36
..                                   0           2       0       0    30 05 2020 21:24:36
usr                                  0           1       1       0    30 05 2020 21:24:36
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat list /usr

File Name                       Parent Inode    Type    Inode   Size   Last Modification Date
.                                    1           1       1       0    30 05 2020 21:24:36
..                                   0           1       1       0    30 05 2020 21:24:36
deneme1                              1           1       2       0    30 05 2020 21:24:36
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat list /usr/deneme1

File Name                       Parent Inode    Type    Inode   Size   Last Modification Date
.                                    2           1       2       0    30 05 2020 21:24:36
..                                   0           1       2       0    30 05 2020 21:24:36
deneme2                              2           1       3       0    30 05 2020 21:24:36
```

8) In the figure below, after creating deneme1 and deneme2 under usr with mkdir command, I display the information about system with dumpe2fs. And program display the correct result with correct folder count and empty Inode count and file system info.

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat dumpe2fs

Disk Size => 1048576
Block Size is => 4096
 Block Count is => 250
iNode Count is => 400
Empty iNode Count is => 396
Empty Data Block Count is => 246
iNode Start Address is => 48
Empty iNode Start Address is => 19248
Data Block Start Address is => 6
Empty Data Block Start Address is => 20848
Dentry Address is => 21872
Root Address is => 6
File System Info
/usr                                        1         1
/usr/deneme1                                2         2
/usr/deneme1/deneme2                        3         3
------------------------------------------------------------
*** Shows Hard link
Total File count is 0
Total Folder count is 3
```

9) In the figure below, I wanted to check when I want to create a folder with the same name. Here we can see the warning message. (Object with the same name exist.)

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat mkdir /usr/deneme1/deneme2

Object with the same name exits!.
```

10) I delete the folder deneme2 under *usr*deneme1 with rmdir command. And then display the filesystem information to check if it is delete or not.
Here we can see decrease from 3 to 2. And also empty data block number and empty inode count is increased.

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat rmdir /usr/deneme1/deneme2
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat dumpe2fs

Disk Size => 1048576
Block Size is => 4096
 Block Count is => 250
iNode Count is => 400
Empty iNode Count is => 397
Empty Data Block Count is => 247
iNode Start Address is => 48
Empty iNode Start Address is => 19248
Data Block Start Address is => 6
Empty Data Block Start Address is => 20848
Dentry Address is => 21872
Root Address is => 6
File System Info
/usr                                    1         1
/usr/deneme1                            2         2
------------------------------------------------------------
*** Shows Hard link
Total File count is 0
Total Folder count is 2
```

11) And also we can see the result with list. Now there isn't deneme2 folder under *usr*deneme1

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat list /usr/deneme1

File Name                       Parent Inode    Type    Inode   Size    Last Modification Date
.                                   2           1       2       0    30 05 2020 21:44:47
..                                  0           1       2       0    30 05 2020 21:44:47
```

12)

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat dumpe2fs

Disk Size => 1048576
Block Size is => 4096
 Block Count is => 250
iNode Count is => 400
Empty iNode Count is => 397
Empty Data Block Count is => 247
iNode Start Address is => 48
Empty iNode Start Address is => 19248
Data Block Start Address is => 6
Empty Data Block Start Address is => 20848
Dentry Address is => 21872
Root Address is => 6
File System Info
/usr                                            1         1
/usr/deneme1                                    2         2
_____________________________________________________________
*** Shows Hard link
Total File count is 0
Total Folder count is 2
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$
```

13) We can see the result of fsck command in the figure below.

```
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/Part_3_Program$ ./fileSystemOper mysystem.dat fsck
command not found: %s
fsck
Block Number:  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47
  48  49
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------
INode Blocks:  1   1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0
Empty Blocks:  0   0   0   0   0   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
   1   1

Block Number: 50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97
  98  99
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------
INode Blocks:  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0
Empty Blocks:  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
   1   1

Block Number:100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
 148 149
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------
INode Blocks:  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0
Empty Blocks:  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
   1   1

Block Number:150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------
INode Blocks:  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0
Empty Blocks:  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
   1   1

Block Number:200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247
 248 249
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------
INode Blocks:  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0
Empty Blocks:  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
   1   0

Block Number:
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------
INode Blocks:
Empty Blocks:
```