CSE 312 HOMEWORK 3 REPORT

EMRE SEZER

1901042640

Directory Structure:

Directory entries take %5 of the total file system's size. At the beginning, "Directories" is written to the data file. After that, each directory entry is written to the data file. Basically "Directories" is the title of this part at the data file. File size is 16 MB, %5 of this file is like 819 KB. Each directory entry size is 270 bytes. Each directory entry can hold 8 files and 8 directories.

Directory entry is represented as below:

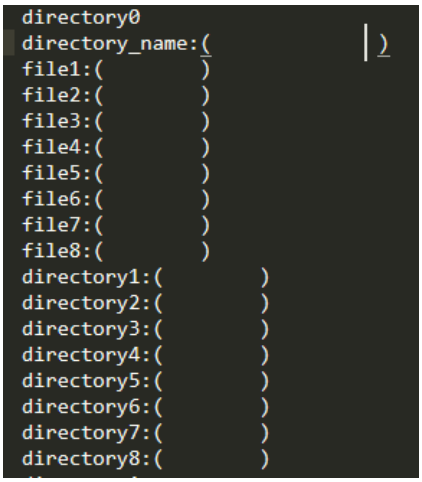
Directory Number: Shows which directory is this with an integer.

Directory Name: Shows name of the directory with a string. Max directory name size is 14 bytes.

File: It holds the i-node number of the file it holds in this directory. It is represented with an integer number. Maximum number size is 8 bytes.

Each directory entry can hold maximum 8 files.

Directory: It holds the directory number of the directory it holds in this directory. It is represented with an integer number. Maximum number size is 8 bytes. Each directory entry can hold maximum 8 directories.



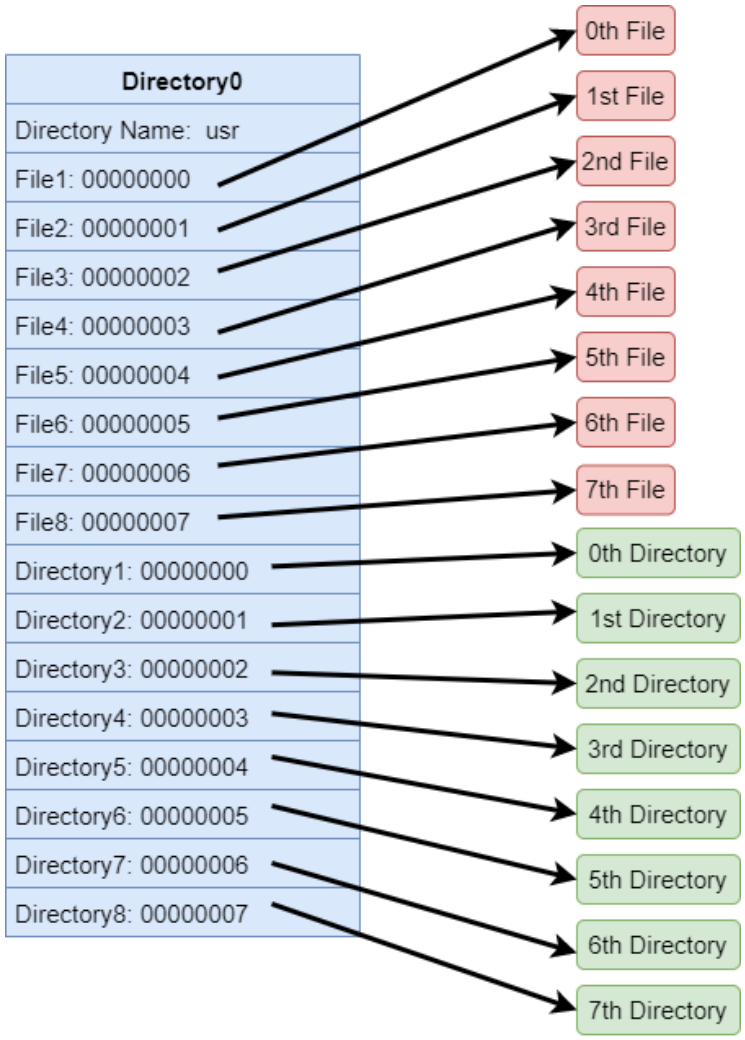
Whenever a directory is created even on the root directory, it will be added to the directories like the picture above. Each directory entry is like picture above. While following a given path, first will check root directory, then will continue with the position that it points.

In order to link a file to the directory, the inode number of the file is saved between the parentheses in the file lines.

In order to link a directory to the a directory, the directory number of the directory is saved between the paranthesis in the directory lines.

Directory number can be represented as x at the “directoryx”.

At the picture above directory number is 0 for example.



Root Directory:

Root directory takes 1 block from the total file system. At the beginning “Root Directory” is written to the data file. Basically “Root Directory” is the title of this part at the data file. Root directory entry size is 75 bytes.

Root directory entry is represented as below:

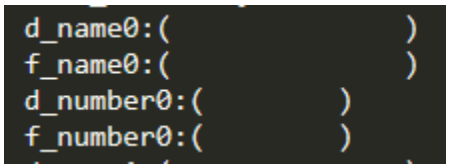
Root Directory Number: It holds the number of the directory as an integer.

Directory Name: Shows the name of the directory with a string. Max directory name size is 14 bytes.

File Name: Shows the name of the file with a string. Max file name size is 14 bytes.

Directory Number: Shows the directory number of the directory. Basically it points to a directory position.

File Number: Shows the directory i-node of the file. Basically it points to a i-node position.



A root directory entry is like above.

In order to link a file to the root directory, the inode number of the file is saved between the parentheses in the f_number lines.

In order to link a directory to the the root directory, the directory number of the directory is saved between the paranthesis in the d_number lines.

I-Node Structure:

I-node entries take %10 of the total file system's size. At the beginning, "I-Nodes" is written to the data file. After that, each i-node entry is written to the data file. Basically "I-Nodes" is the title of this part at the data file. File size is 16 MB, %10 of this file is like 1638 KB. Each directory entry size is 375 bytes. Each i-node entry can hold 8 data blocks.

I-node entry is represented as below:

I-Node Number: It holds the number of the i-node as an integer.

File Name: It represents the file name of the file as a string. Size is 14 bytes max.

Data: This number size is 8 bytes max.number. This number size is 8 bytes max.Represents the data blocks hold the content of the file.

Single Indirect: It represents the single indirect block number as an integer. This number size is 8 bytes max.

Double Indirect: It represents the double indirect block number as an integer. This number size is 8 bytes max.

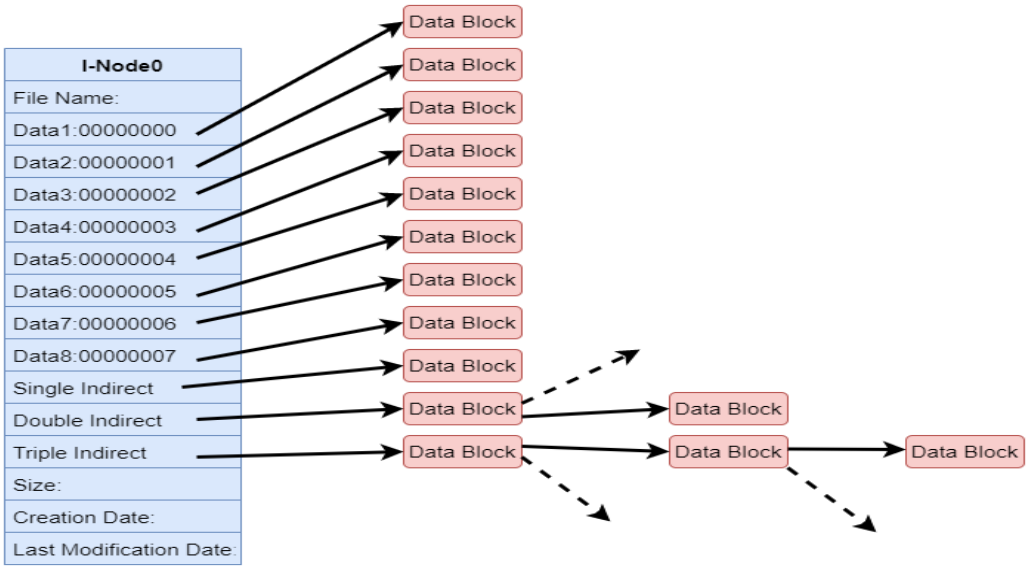
Triple Indirect: It represents the triple indirect block number as an integer. This number size is 8 bytes max.

Size: It represents the size of the file.

Creation Date: It represents the creation date of the file.

Last Modification Date: It represents the last modification date of the file.

```
inode0
file_name:(                )
data1:(         )
data2:(         )
data3:(         )
data4:(         )
data5:(         )
data6:(         )
data7:(         )
data8:(         )
single_indirect:(               )
double_indirect:(               )
triple_indirect:(               )
size:(         )
creation_date:(                      )
last_modification_date:(                      )
```



Whenever a file is created even on the root directory, it will be added to the i-nodes block like the picture above. Each i-node entry is like picture above. Root directory can point i-nodes and each directory can hold i-nodes.

i-node number can be represented as x at the “inodex”.

At the picture above i-node number is 0 for example.

Superblock:

First block of the file system is given to the superblock. At the beginning, “Superblock” is written to the data file. Basically “Superblock” is the title of this part at the data file.

It holds all of the information about the file system. You can see the data it holds at the picture below.

Superblock is represented as below:

Number of Blocks: Represents the total number of blocks in the file system.

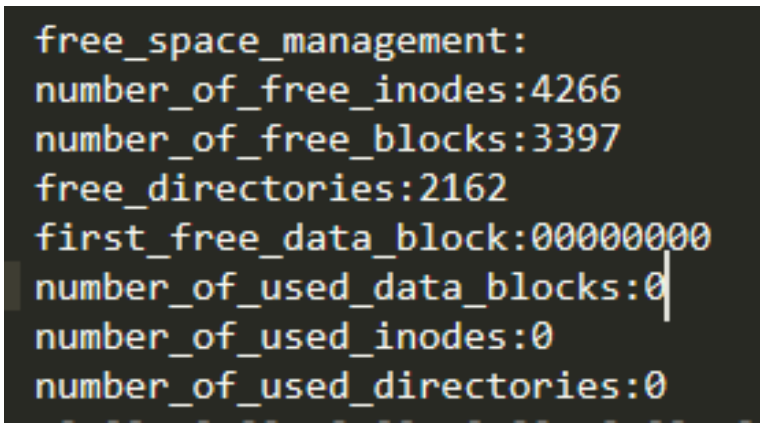
Number of Directory Blocks : Represents the total number of directory blocks in the file system.
Number of directory blocks : Represents the total number of directory blocks in the file system.
Number of i-nodes : Represents the total number of i-nodes in the file system.
Number of Sirectories : Represents the total number of directories
Block size : Represents the size of a block
i-node block starts At: Represents the location where the i-nodes start.
Free Space Management Block Starts At: Holds the byte where the free space management block starts.
Root Directory Block Starts At: Holds the byte where the root directory block starts.
Directory Block Starts At: Holds the location where the directories start
Data Block Starts At: Represents the location where the data blocks start.

```
superblock:
number_of_blocks:4000
block_size:4000
number_of_directory_blocks:200
number_of_directories:2162
number_of_inodes:4266
inode_block_starts_at:4000
free_space_management_block_starts_at:1604000
root_directory_blocks_starts_at:1608000
directory_block_starts_at:1612000
data_block_starts_at:2412000
number_of_root_directory:40
```

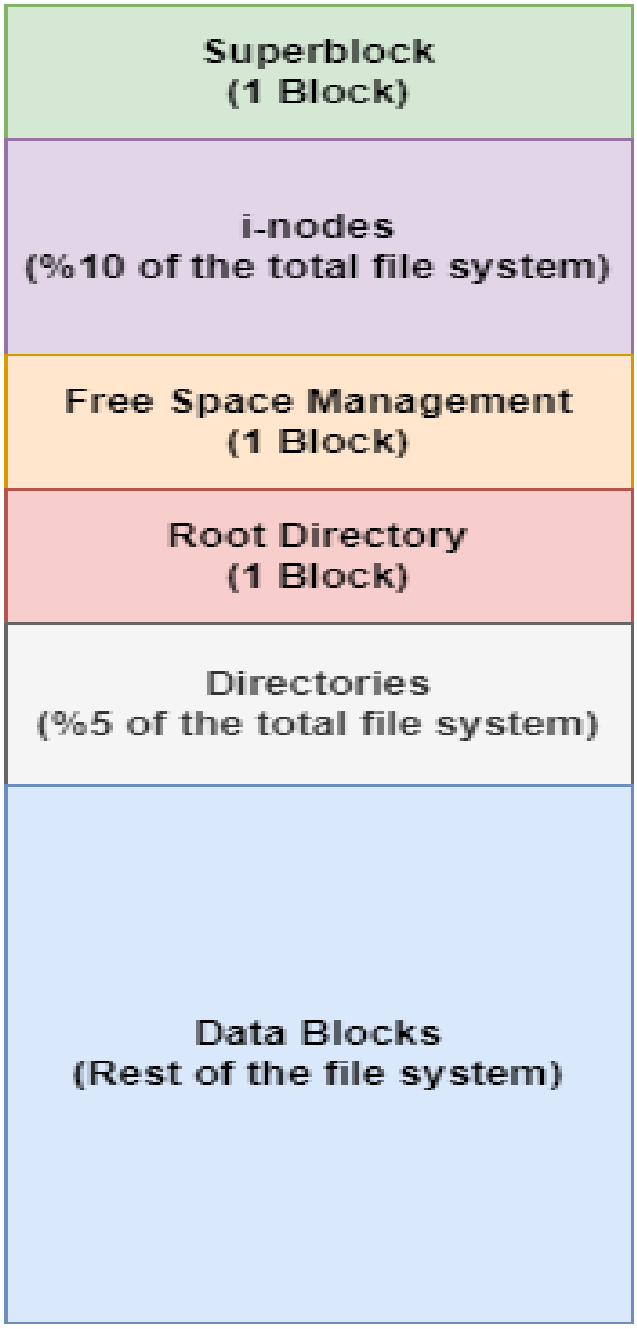
In my file system size 16mb fixed. The numbers are above are calculated at part2 of the homework.

Free Blocks:

Number of Free i-nodes : Represents the number of free i-nodes in the file system.
Number of Free Blocks : Represents the number of free blocks in the file system.
Number of Free Directories : Represents the number of free directories in the filesystem.
First Free Aata Block Starts At: Represents the first free data block's number.
Number of Used i-nodes : Represents the number of reserved i-nodes in the file system.
Number of Used Data Blocks : Represents the number of reserved blocks in the file system.
Number of Used Directories : Represents the number of reserved directories in the filesystem.



File System's Total Space Divisions:



Functions of Part3:

MAINS:

void mkdir(char * path):

Creates a directory by checking the first available position at the directories. When finds one writes the information there. If path is on root directory, searches for the first empty directory position. Prints error message and exits if an error occurs. Writes its directory number to its parent folder if exists.

void rmdir(char * path):

Removes the directory from the directories and removes its directory number from its parent directory if it is not on the root directory. If it is on root just removes it from the root directory.

void write(char * path, char * fileName2):

Creates a file by checking the first available position at thei-nodes. When finds one writes the information there. If path is on root directory, searches for the first empty i-node position. Prints error message and exits if an error occurs. Writes its i-node number to its parent folder.

void read(char * path, char * fileName2):

Follows the path and prints the data block content which i-node points.

void del(char * path):

Removes the file from thei-nodes and removes itsi-node number from its parent directory if it is not on the root directory. If it is on root just removes it from the root directory.

void dumpe2fs():

Prints the free space management block content.

void dir(char * path):

Follows the path and prints the file and directories the current directory includes.

HELPERS:

char * makeNumberString(int num)

char * makeNullString(char * input)

void makeNull()

int countCharOccurences(char * str, char c)

void pickFromSuperblock()

int findEmptyRootDirectory()

int findEmptyRootINode()

void addRootDirectory(char * name, int num, int location)

void addRootINode(char * name, int num, int location)

int searchRootDirectory(char * name)

int searchRootINode(char * name)

int searchRootFile(char * name)

void addDirectory(char * name, int num)

void addINodeToDirectory(int num, int nodeNum)

void addINode(char * name, int num)

void addSizeToINode(int nodeNum)

void addCreationTimeToINode(int nodeNum)

void addLastModificationTimeToINode(int nodeNum)

void addDirectoryOfDirectory(int num, int index)

int checkDirectoryNumber(char * name, int num)

int checkINodeNumber(char * name, int num)

```
int findEmptyDirectory()
int findEmptyINode()
int checkDirectoriesOfDirectory(char * name, int num)
int checkINodesOfDirectory(char * name, int num)
void removeDirectoryFromDirectory(int source, int goal)
void removeINodeFromDirectory(int source, int goal)
int checkIfEmpty(int num)
void resetDirectory(int num)
void resetINode(int num)
int getRootDirectoryNum(char * path)
int getRootINodeNum(char * path)
void resetRootDirectory(int num)
void resetRootINode(int num)
void readFreeSpaceManagement()
void writeFreeSpaceManagament()
int getReadLocation(int nodeNum)
void readDir(int location, char * fileName2)
void writeNewDataBlock(char * fileName2)
int checkMaxSizeOfFile(char * fileName2)
void writeDataBlocksOfINode(int num)
void resetDataBlocksOfINode(int nodeNum)
void printRootDirectory()
void printNameOfDirectory(int num)
void printNameOfINode(int num)
```

void checkContentOfDirectory(int num)