Gebze Technical University
Computer Engineering

CSE 312 - 2020 Spring

MIDTERM REPORT

TAYLAN ÖNDER
151044015

1 INTRODUCTION

1.1 Problem Definition

In this project, I coded the representation unix file system using the c and c ++ languages. Problem is designing a file system that uses i-nodes blocks and data blocks to keep my files. Posix file system operations and information are realized by simulating the ways of access and use.

2 METHOD

2.1 Problem Solution Approach

- Define your directory structure and directory entries;

This file system, which is designed according to The UNIX V7 File System, allows the inodes to keep the directory and file information on the file system and keep their contents in a block structure. With direct access, it allows to store information for directory creation and file copying operations for 10 different datablocks. It is aimed to inform operations that they will use with errors

- Define how you keep the free blocks and free i-nodes;

In my file system, I used a bitmap array to keep the full empty information in two different arrays. I returned the next empty inode and block information in the process to be used for the unused inodes and blocks by assigning 0 and -1 values that contain empty inode and block information.

For the size of the inode bitmap directory, I used the inode number in the given input . For the size of the bitmap directory, I calculated a block numbers for in my file system based on the size of the block given by subtracting the inodes, super block, and the entire size of the inode bitmap directory, and this number was the size of the block bitmap.

- Define your i-node structure

```
struct Inode {
    int size;
    int number;
    char lastDate[timeSize];
    char time[timeSize];
    char fileName[fileSize];
    int blockAddr;
    int directBlocks[directSize];
    int type;
};
 int const timeSize=25;
 int const directSize=10;
 int const fileSize=14;
```

PICTURE 1

Inode structure has attributes and address like this. Size, last modification date and time, and name of the file was also the information requested in the homework pdf. For the time information, I gave you the maximum that it would normally use. For Filename datablockta Figure 4-32. A I have given 14 value according to the information in UNIX V7 directory entry.Type is keep the file or directory information to use in some operators to use for not doing anything wrong . .Single, double and triple link coding is missing however, I kept the contents of the files and the file information contained in the directories in the array, which has a constant area of 10 that it can access directly, as data packages in the blocks with the block address access held in the inode.

- Define your superblock that contains crucial information about the file system such as the block size, i-node positions, block positions, etc.

```
struct SuperBlock {
    int numberInode;
    int inodeAddr;
    int blockAddr;
    int bitMapAddr;
    int blockNumber;
    int blockBitmapAddr;
    int blockSize;
};
```

PICTURE 2

Superblock in Picture2 has Inode , block number,block size and adress of starting Inodes,blocks and bitmaps. This information is taken at the very beginning of everything in the file system and kept ready for use in operators.

To summarize, the information about the files and directories that I created in the file system is kept with the inode struct. The files and directories in the subdirectory for the directories are accessed through the variables I hold as directblock arrays and block addresses. In cases where this block address is not sufficient, it can be accessed directly from the block array directly from the block array.

In the block address, access to file and directory information and inode information in its subdirectory related to the contents of the file copied for the file in 4kb area. Bitmap

sequences defined for inode and bitmap are used in new file and directory creation operations.All delete and create operations, arrays have updated.Also I used to operationType global variable to use same method to prevent code duplication.

3 RESULT

3.1 Running Results

```
cse312@ubuntu:~/Desktop/vize$ g++ part3.cpp -o part3 -Wall
cse312@ubuntu:~/Desktop/vize$ ./part3 mySystem.data dumpe2fs
Block count 243
Inode count 400
Free Block 242
Free Inode 399
Number of Files 0
Number of Directories 1
Block Size 4
Filename        Inode no        Block numbers
/                   0               0
cse312@ubuntu:~/Desktop/vize$ ./part3 mySystem.data mkdir "/usr"
cse312@ubuntu:~/Desktop/vize$ ./part3 mySystem.data mkdir "/usr/q"
cse312@ubuntu:~/Desktop/vize$ ./part3 mySystem.data dumpe2fs
Block count 243
Inode count 400
Free Block 240
Free Inode 397
Number of Files 0
Number of Directories 3
Block Size 4
Filename        Inode no        Block numbers
/                   0               0
usr                 1               1
q                   2               2
cse312@ubuntu:~/Desktop/vize$ ./part3 mySystem.data write "/usr/file5" linuxfile.txt
cse312@ubuntu:~/Desktop/vize$ ./part3 mySystem.data dumpe2fs
Block count 243
Inode count 400
Free Block 230
Free Inode 396
Number of Files 1
Number of Directories 3
Block Size 4
Filename        Inode no        Block numbers
/                   0               0
usr                 1               1
q                   2               2
file5               3               3 - 4- 5- 6- 7- 8- 9- 10- 11- 12
cse312@ubuntu:~/Desktop/vize$ ./part3 mySystem.data read "/usr/file5" linuxfile1.txt
cse312@ubuntu:~/Desktop/vize$ cmp linuxfile1.txt linuxfile2.txt
cmp: linuxfile2.txt: No such file or directory
cse312@ubuntu:~/Desktop/vize$ cmp linuxfile1.txt linuxfile.txt
cse312@ubuntu:~/Desktop/vize$
```

Example of giving start and some operation outputs.

```
void mkdir();
void rmdir();
void list(FILE *fp,char* fileName);
void dumpe2fs(FILE *fp);
void writeFile(FILE *fp,char* path,char* fileName);
void readFile(FILE *fp,char* path,char* fileName);
void deleteCommand(FILE *fp,char* path);
```

mkdir() : makes a new directory in file system if possible.
rmdir(): delete a directory in file system if possible.
list(); lists the contents of the given path directory.
dumpe2fs(): give information about in file system. It will list block count, i-node count, free block and i-nodes, number of files and directories, and block size and i-nodes, blocks and the file names information.
writeFile(): create and copies the contents of giving file in parameter.
readFile(): reads the file system and write contents given filename in Linux.
deleteCommand(): Deletes the file name in file system.