Gebze Technical University

Department of Computer Engineering

CSE 312 / CSE 504

Spring 2020

Midterm Exam Project – File Systems

Report

In this project an i-node based file system was created. I-nodes are structures that hold file attributes and data blocks.

An entry is 16 bytes hold 14 bytes of file or directory name and 2 bytes hold the i-node id. When a file is added to the file system, an entry is created and added to a block. This block is filled with entries indicating many files of the same level. The first two blocks are reserved for current and parent entries. The starting point is the root directory. Current and parent entries in the root directory show i-node number 0.(In my design, the i-node ids starts at 0).

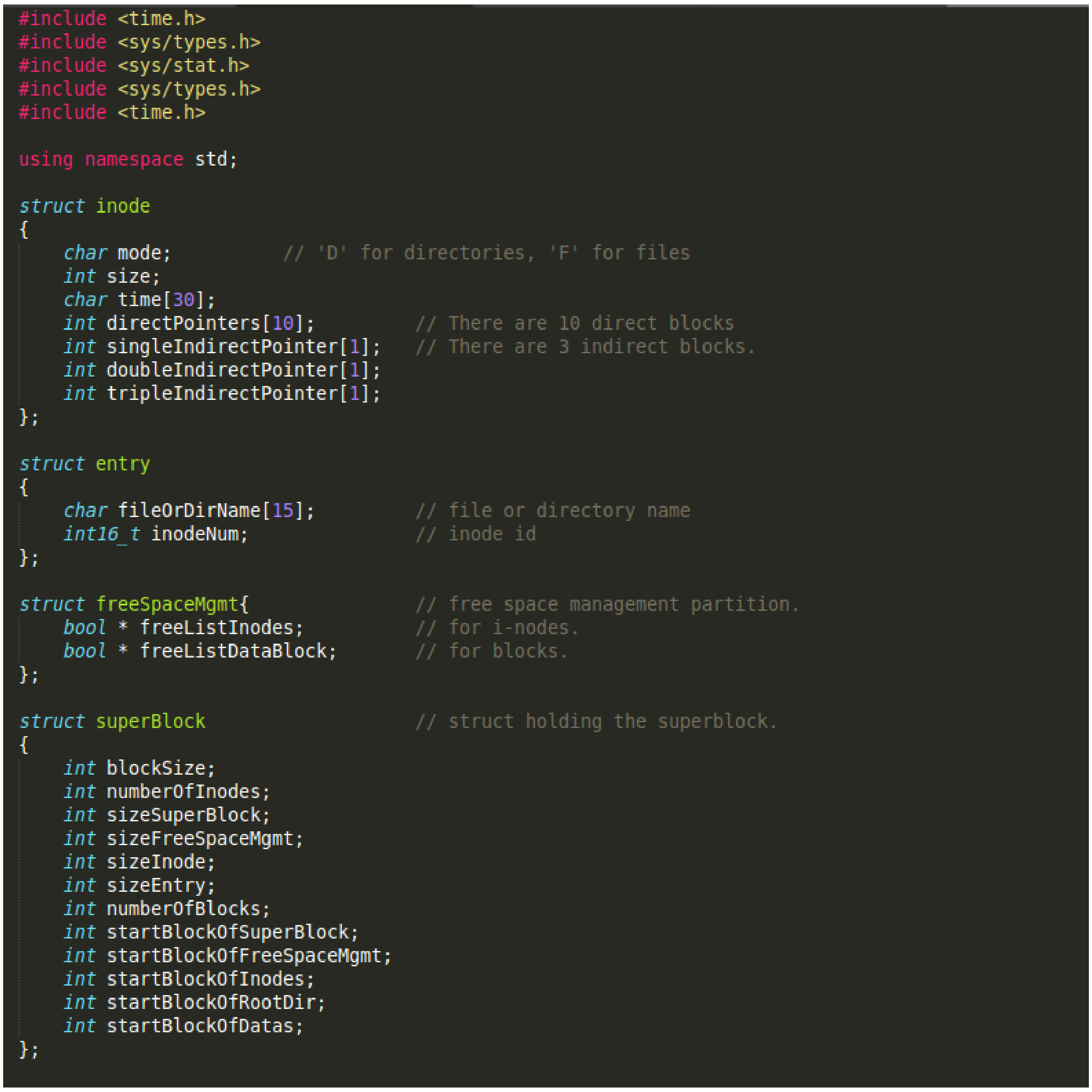
The location where the i-node blocks are held is known. The inode id kept by the entry in the directory contains the required i-node.

For example, let's look for a folder named “/usr” in the root directory. The location of the root directory is known. It is examined by reading the entries from the root directory. The entry with the name “/usr” will show the desired inode. Let the inode id be 6. Looking at the 6th inode in the inode list, the correct inode is found. This inode holds mode, size, date and time, direct and indirect data block addresses. These block addresses are blocks that holds child directory or files.

A free space management partition showing empty and full spaces is required. Two lists constitute this partition in the structure I designed. These two lists hold all empty inodes and blocks. If a place is used, that element is boolean false in the list. Empty and full places can be determined by examining these lists.

The disk also has a super block containing the necessary information. Super block is the first block on disk. It holds block size, number of i-nodes, size of super block, size of free-space management, size of an inode, size of an entry, number of blocks and start blocks of super block, free-space management partition, i-nodes list, root directory and data blocks.

The structures that hold these partitions on the disk are shown below.



```
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <time.h>

using namespace std;

struct inode
{
    char mode;          // 'D' for directories, 'F' for files
    int size;
    char time[30];
    int directPointers[10];         // There are 10 direct blocks
    int singleIndirectPointer[1];   // There are 3 indirect blocks.
    int doubleIndirectPointer[1];
    int tripleIndirectPointer[1];
};

struct entry
{
    char fileOrDirName[15];         // file or directory name
    int16_t inodeNum;               // inode id
};

struct freeSpaceMgmt{               // free space management partition.
    bool * freeListInodes;          // for i-nodes.
    bool * freeListDataBlock;       // for blocks.
};

struct superBlock                   // struct holding the superblock.
{
    int blockSize;
    int numberOfInodes;
    int sizeSuperBlock;
    int sizeFreeSpaceMgmt;
    int sizeInode;
    int sizeEntry;
    int numberOfBlocks;
    int startBlockOfSuperBlock;
    int startBlockOfFreeSpaceMgmt;
    int startBlockOfInodes;
    int startBlockOfRootDir;
    int startBlockOfDatas;
};
```

Part3 has functions that perform some disk operations. Let's examine them now. The function of the command entered is executed.

mkdirCommandRun;

This function has the slash number with the help of the findSlashNum() function. Slash separated sections are examined and the last piece is reached with getFirstPathPartition(). At first the length of the directory is examined, because this is constant. Inode is created and data block and inode id are found with the help of select functions. If there is no error, the inode is load into the disk. Also, the entry showing this inode is uploaded to the parent directory.

rmdirCommandRun;

This function like mkdirCommanRun examines up to the last path. If there is no problem, remove the directory with the help of the deleteTheBlock(), deleteTheInode() and deleteTheEntry() functions. And updates the free space partition.

writeCommandRun;

When this function is on the right path, it reads the desired file and writes it to the blocks. If there is no place in the direct blocks, it saves to the indirect. And updates the free space partition.

readCommandRun;

This function reads a file in the system and writes it to the desired physical file. All blocks are read except for the last block. The full part of the last block is read. It is found with size of file – written byte.

dumpe2fsCommandRun;

It will list block count, i-node count, free block and i-nodes, number of files and directories, and block size. All inodes are checked and their information is printed. Which entry is showing that inode is found by browsing the lists in free space management.

listCommandRun;

This function lists the content of given directory. The correct entry is found and the inode indicated by that entry is found.

delCommandRun;

This function deletes file from the given path. It works like rmdir, but unlike rmdir, there are many block addresses, and these addresses should be deleted. It is done with help of delete functions. Free space is updated.

Fatih Dural

151044041