CSE 312/504 Operating Systems
Homework 3

Harun ALBAYRAK - 171044014

May 31, 2022

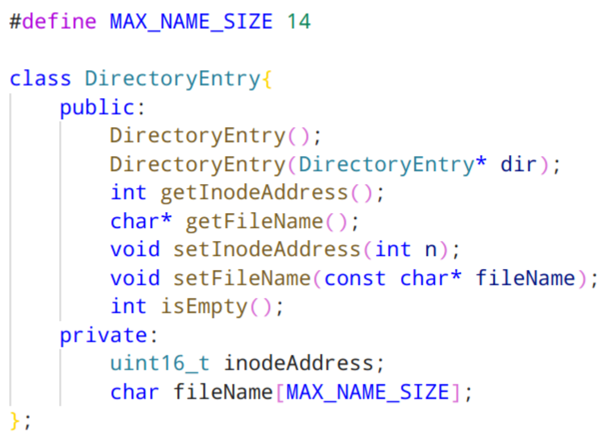
FileSystem Class

```
#define NUMBEROF_INODE 512
#define NUMBEROF_BLOCK 65535

using namespace std;

typedef struct{
    SuperBlock superBlock;
    Inode inodes[NUMBEROF_INODE];
    DataBlock dataBlocks[NUMBEROF_BLOCK];
} SID;

class FileSystem {
    public:
        FileSystem();
        FileSystem(int blockSize);
        FileSystem(int blockSize, string filename);
        FileSystem(string filename);
        int makeFileSystem();
        int createFile();
        int createReader();
        int sendCommand(int command, string path);
        int sendCommand(int command, string path, string filename);
        int dir(string path);
        int mkdir(string path);
        int rmdir(string path);
        int dumpe2fs();
        int write(string path, string filename);
        int read(string path, string filename);
        int del(string path);
    private:
        int getBlockSize();
        int createInode(Inode* inode ,int i);
        int getDataBlock2(const char* db);
        int splitPath(string* path);
        int calculateInodeSize(int inodeNum);
        int writeSIDToFS();
        int readSIDFromFS();
        int incrementI_B();
        int printCommand(int command, string path, string filename);
        int printInodes();

        uint16_t blockSize = 0;
        string filename;
        SID sid;
};
```

FileSytem class

The file system class consists of the filename of the filesystem and a SID struct. The SID struct consists of one **superBlock**, the number of **inodes**, and the number of **data blocks**. Moreover, the file system class has several methods to do some operations on the file system.

DirectoryEntry Class

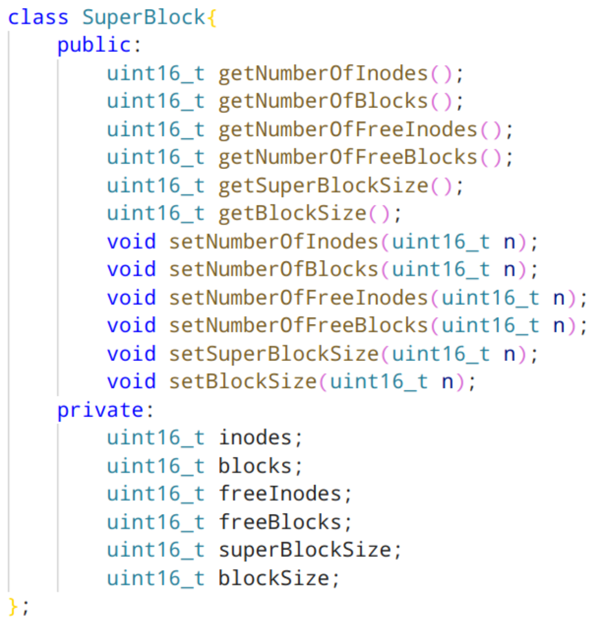


```
#define MAX_NAME_SIZE 14

class DirectoryEntry{
    public:
        DirectoryEntry();
        DirectoryEntry(DirectoryEntry* dir);
        int getInodeAddress();
        char* getFileName();
        void setInodeAddress(int n);
        void setFileName(const char* fileName);
        int isEmpty();
    private:
        uint16_t inodeAddress;
        char fileName[MAX_NAME_SIZE];
};
```

DirectoryEntry Class

The DirectoryEntry class has an address of an inode and filename. The directory entries can be added to the datablocks of the directories.

SuperBlock Class

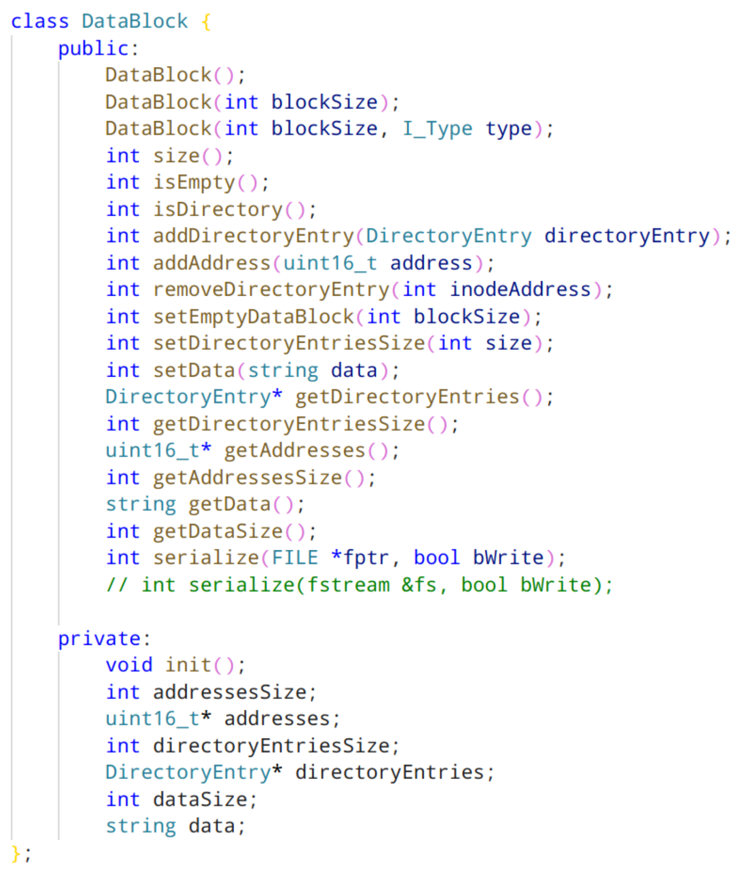


```
class SuperBlock{
    public:
        uint16_t getNumberOfInodes();
        uint16_t getNumberOfBlocks();
        uint16_t getNumberOfFreeInodes();
        uint16_t getNumberOfFreeBlocks();
        uint16_t getSuperBlockSize();
        uint16_t getBlockSize();
        void setNumberOfInodes(uint16_t n);
        void setNumberOfBlocks(uint16_t n);
        void setNumberOfFreeInodes(uint16_t n);
        void setNumberOfFreeBlocks(uint16_t n);
        void setSuperBlockSize(uint16_t n);
        void setBlockSize(uint16_t n);
    private:
        uint16_t inodes;
        uint16_t blocks;
        uint16_t freeInodes;
        uint16_t freeBlocks;
        uint16_t superBlockSize;
        uint16_t blockSize;
};
```

SuperBlock Class

My SuperBlock Class holds the metadata of the file system. For instance, the number of inodes, number of blocks, number of free inodes, and number of free blocks are kept in this class. Moreover, this class has fields that hold the size of the super block and block size of the filesystem.

DataBlock Class

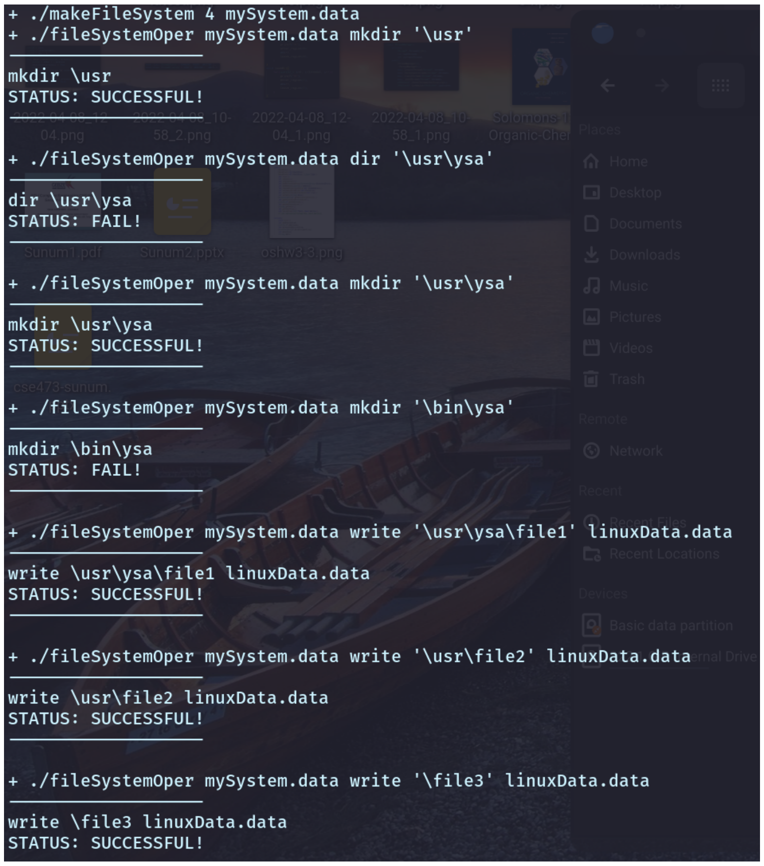


```
class DataBlock {
    public:
        DataBlock();
        DataBlock(int blockSize);
        DataBlock(int blockSize, I_Type type);
        int size();
        int isEmpty();
        int isDirectory();
        int addDirectoryEntry(DirectoryEntry directoryEntry);
        int addAddress(uint16_t address);
        int removeDirectoryEntry(int inodeAddress);
        int setEmptyDataBlock(int blockSize);
        int setDirectoryEntriesSize(int size);
        int setData(string data);
        DirectoryEntry* getDirectoryEntries();
        int getDirectoryEntriesSize();
        uint16_t* getAddresses();
        int getAddressesSize();
        string getData();
        int getDataSize();
        int serialize(FILE *fptr, bool bWrite);
        // int serialize(fstream &fs, bool bWrite);

    private:
        void init();
        int addressesSize;
        uint16_t* addresses;
        int directoryEntriesSize;
        DirectoryEntry* directoryEntries;
        int dataSize;
        string data;
};
```

DataBlock Class

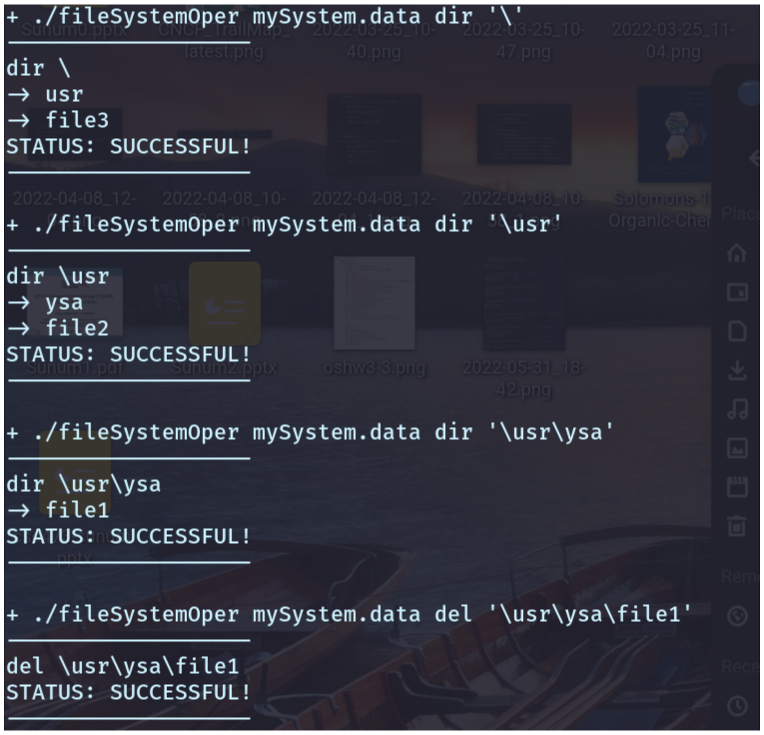
My DataBlock Class may hold addresses of other block addresses or directory entries or a data of a file. The thing that will be kept is decided by the type of a DataBlock. For instance, the DataBlock that is pointed by an inode with a directory type can only hold directory entries or the DataBlock that is pointed by an inode with a regular file type can only hold addresses of other datablocks.

Tests



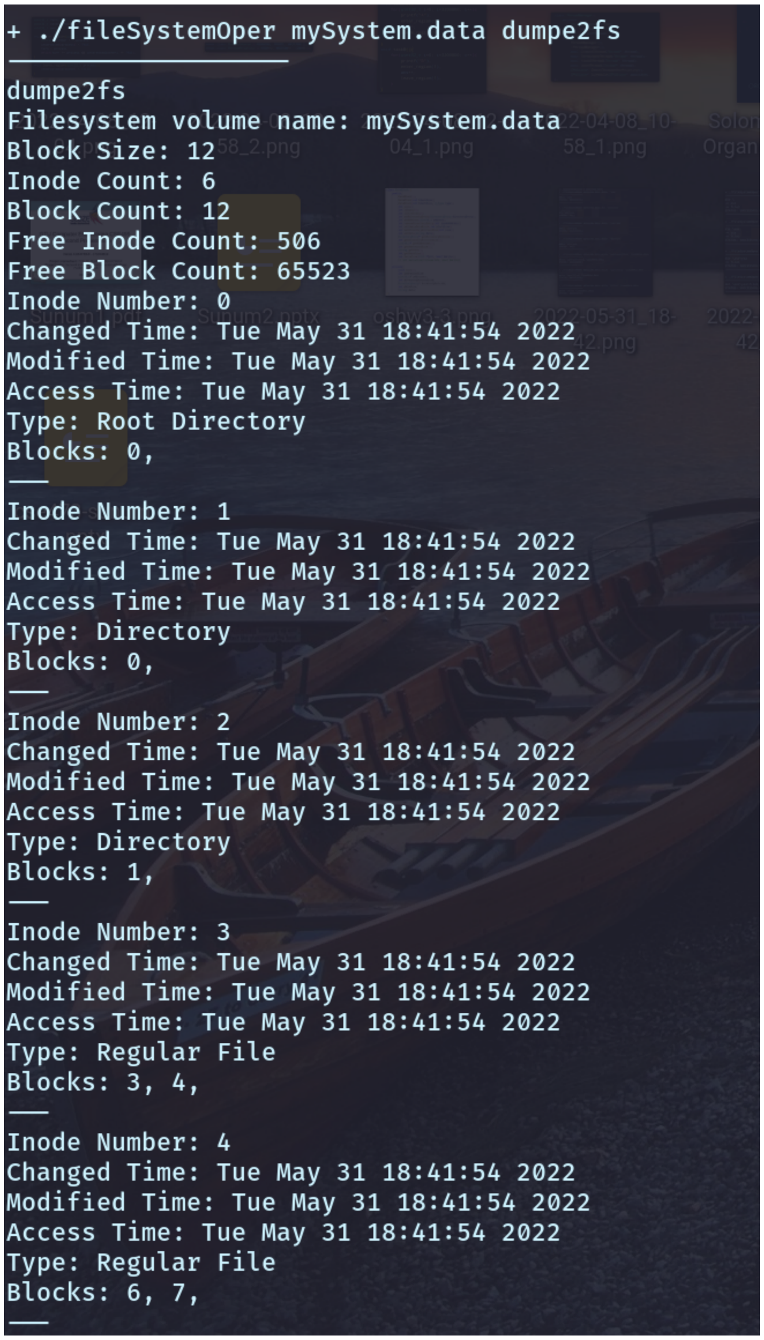
```
+ ./makeFileSystem 4 mySystem.data
+ ./fileSystemOper mySystem.data mkdir '\usr'

mkdir \usr
STATUS: SUCCESSFUL!

+ ./fileSystemOper mySystem.data dir '\usr\ysa'

dir \usr\ysa
STATUS: FAIL!

+ ./fileSystemOper mySystem.data mkdir '\usr\ysa'

mkdir \usr\ysa
STATUS: SUCCESSFUL!

+ ./fileSystemOper mySystem.data mkdir '\bin\ysa'

mkdir \bin\ysa
STATUS: FAIL!

+ ./fileSystemOper mySystem.data write '\usr\ysa\file1' linuxData.data

write \usr\ysa\file1 linuxData.data
STATUS: SUCCESSFUL!

+ ./fileSystemOper mySystem.data write '\usr\file2' linuxData.data

write \usr\file2 linuxData.data
STATUS: SUCCESSFUL!

+ ./fileSystemOper mySystem.data write '\file3' linuxData.data

write \file3 linuxData.data
STATUS: SUCCESSFUL!
```

Test



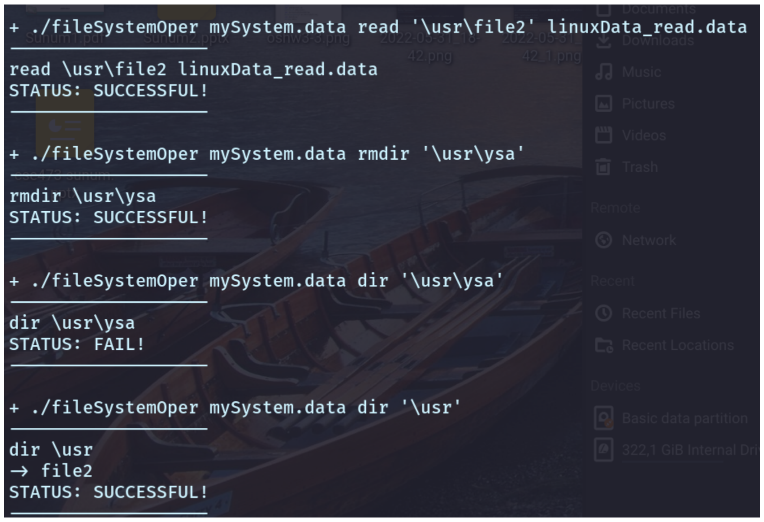
```
+ ./fileSystemOper mySystem.data dir '\'

dir \
→ usr
→ file3
STATUS: SUCCESSFUL!

+ ./fileSystemOper mySystem.data dir '\usr'

dir \usr
→ ysa
→ file2
STATUS: SUCCESSFUL!

+ ./fileSystemOper mySystem.data dir '\usr\ysa'

dir \usr\ysa
→ file1
STATUS: SUCCESSFUL!

+ ./fileSystemOper mySystem.data del '\usr\ysa\file1'

del \usr\ysa\file1
STATUS: SUCCESSFUL!
```

Test



```
+ ./fileSystemOper mySystem.data dumpe2fs

dumpe2fs
Filesystem volume name: mySystem.data
Block Size: 12
Inode Count: 6
Block Count: 12
Free Inode Count: 506
Free Block Count: 65523
Inode Number: 0
Changed Time: Tue May 31 18:41:54 2022
Modified Time: Tue May 31 18:41:54 2022
Access Time: Tue May 31 18:41:54 2022
Type: Root Directory
Blocks: 0,
---
Inode Number: 1
Changed Time: Tue May 31 18:41:54 2022
Modified Time: Tue May 31 18:41:54 2022
Access Time: Tue May 31 18:41:54 2022
Type: Directory
Blocks: 0,
---
Inode Number: 2
Changed Time: Tue May 31 18:41:54 2022
Modified Time: Tue May 31 18:41:54 2022
Access Time: Tue May 31 18:41:54 2022
Type: Directory
Blocks: 1,
---
Inode Number: 3
Changed Time: Tue May 31 18:41:54 2022
Modified Time: Tue May 31 18:41:54 2022
Access Time: Tue May 31 18:41:54 2022
Type: Regular File
Blocks: 3, 4,
---
Inode Number: 4
Changed Time: Tue May 31 18:41:54 2022
Modified Time: Tue May 31 18:41:54 2022
Access Time: Tue May 31 18:41:54 2022
Type: Regular File
Blocks: 6, 7,
---
```

Test



```
+ ./fileSystemOper mySystem.data read '\usr\file2' linuxData_read.data
──────────────────
read \usr\file2 linuxData_read.data
STATUS: SUCCESSFUL!
──────────────────

+ ./fileSystemOper mySystem.data rmdir '\usr\ysa'
──────────────────
rmdir \usr\ysa
STATUS: SUCCESSFUL!
──────────────────

+ ./fileSystemOper mySystem.data dir '\usr\ysa'
──────────────────
dir \usr\ysa
STATUS: FAIL!
──────────────────

+ ./fileSystemOper mySystem.data dir '\usr'
──────────────────
dir \usr
→ file2
STATUS: SUCCESSFUL!
──────────────────
```

Test