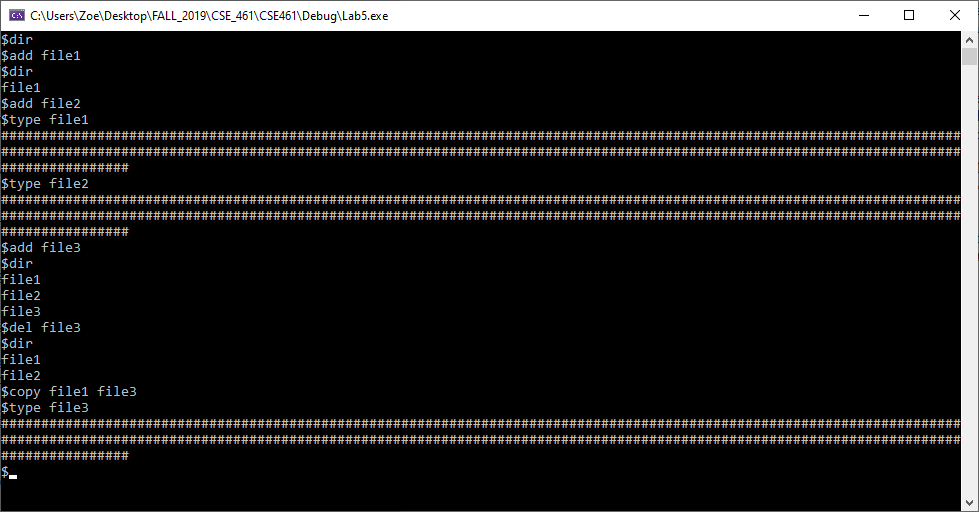
Zoe Veale

Lab 6



#include <iostream>

#include "shell.h"

int main(int argc, char\* argv) {

//

//This main program inputs commands to the shell.

//It inputs commands as : command op1 op2

//You should modify it to work for your implementation.

//

Sdisk diskA("diskA", 256, 128);

FileSystem fsys("diskA", 256, 128);

Shell shell("diskA", 256, 128);

std::string s;

std::string command = "go";

std::string op1, op2;

while (command != "quit") {

command.clear();

op1.clear();

op2.clear();

std::cout << "$";

std::getline(std::cin, s);

int firstblank = s.find(' ');

if (firstblank < s.length()) s[firstblank] = '#';

int secondblank = s.find(' ');

command = s.substr(0, firstblank);

if (firstblank < s.length())

op1 = s.substr(firstblank + 1, secondblank - firstblank - 1);

if (secondblank < s.length())

op2 = s.substr(secondblank + 1);

if (command == "dir") {

// use the ls function

shell.Directory();

}

if (command == "add") {

// The variable op1 is the new file

shell.Add(op1);

}

if (command == "del") {

// The variable op1 is the file

shell.Delete(op1);

}

if (command == "type") {

// The variable op1 is the file

shell.Type(op1);

}

if (command == "copy") {

// The variable op1 is the source file and the variable op2 is the destination file.

shell.Copy(op1, op2);

}

}

return 0;

}

#include "fileSystem.h"

class Shell : public FileSystem {

public:

Shell(std::string filename, int numberofblocks, int blocksize);

int Directory();// lists all files

int Add(std::string file);// add a new file using input from the keyboard

int Delete(std::string file);// deletes the file

int Type(std::string file);//lists the contents of file

int Copy(std::string file1, std::string file2);//copies file1 to file2

private:

const int FILENAME\_SIZE = 5;

};

#include "shell.h"

Shell::Shell(std::string filename, int numberofblocks, int blocksize) :

FileSystem(filename, numberofblocks, blocksize){

}

int Shell::Directory() {

std::vector<std::string> fileList = List();

for (unsigned int i = 0; i < fileList.size(); i++) {

printf("%s\n", fileList[i].c\_str());

}

return 0;

}

int Shell::Add(std::string file) {

if (file.size() > FILENAME\_SIZE) {

printf("file name to large\n");

return 0;

}

if (NewFile(file))

return 1;

printf("file already exsits\n");

return 0;

}

int Shell::Delete(std::string file) {

RemoveFile(file);

return 0;

}

int Shell::Type(std::string file) {

int firstBlock = GetFirstBlock(file).first;

if (firstBlock == 0) {

printf("file does not exist\n");

return 0;

}

std::string buffer;

for (int i = firstBlock; i != 0; i = NextBlock(file, i)) {

ReadBlock(file, i, buffer);

printf("%s", buffer.c\_str());

}

printf("\n");

return 1;

}

int Shell::Copy(std::string file1, std::string file2) {

int firstBlock = GetFirstBlock(file1).first;

if (firstBlock == 0) {

printf("file does not exist\n");

return 0;

}

std::string buffer;

std::string bufferCopy{ "" };

int k{ 0 };

for (int i = firstBlock; i != 0; i = NextBlock(file1, i)) {

ReadBlock(file1, i, buffer);

bufferCopy += buffer;

++k; //number of blocks needed

}

unsigned int bufferSize{ buffer.size() };

int j{ 0 };

firstBlock = GetFirstBlock(file2).first;

if (firstBlock == 0) {

Add(file2);

firstBlock = GetFirstBlock(file2).first;

}

for (int i = firstBlock; j != k; i = NextBlock(file2, i)) {

if (i == 0) {

buffer = bufferCopy.substr(bufferSize \* j, bufferSize);

AddBlock(file2, buffer);

}

else {

buffer = bufferCopy.substr(bufferSize \* j, bufferSize);

WriteBlock(file2, i, buffer);

}

++j;

}

return 0;

}