**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 11

по дисциплине «Операционные системы»

на тему: «Реализация файловой системы»

Выполнил: студент гр. ИТП 11

Трацевский И.С.

Принял: преподаватель-стажер

Карась О.В.

Гомель 2022

**Цель работы:** разработать модель файловой системы.

**Задание:** разработать приложение, создающее виртуальный файл и позволяющее

* форматировать виртуальный файл с возможностью задания размера кластера;
* создавать каталоги в виртуальном файле;
* производить учёт свободного пространства;
* реализовывать поиск файлов и директорий;
* удалять файлы из виртуального файла;

**Ход работы:** в поля ввода вводится информация о файле (название, папка, размер). По нажатию на кнопку «добавить файл» файл записывается на диск (если его размер не превышает свободное место на диске) пример добавления приведен на рисунке 1. Удалить файл можно на кнопку «удалить файл», предварительно введя его название(или полный путь, если он находится в папке), удаление файла изображено на рисунке 2. На кнопку «дефрагментация» производится дефрагментация диска. Дефрагментация показана на рисунке 3. Поиск файлов и каталог осуществляется с помощью хэш-таблицы.



Рисунок 1 – Добавление файла



Рисунок 2 – *File2* удален



Рисунок 3 – произведена дефрагментация

Листинг *JavaScript* файла представлен в приложении А. Листинг *HTML* файла представлен в приложении Б. Листинг *CSS* файла представлен в приложении В.

**Вывод:** в результате выполнения лабораторной работы быларазработана модель файловой системы.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг *JavaScript*-файла программы**

const root = document.getElementById("root");

const fileNameInput = document.getElementById("fileNameInput");

const fileFolderInput = document.getElementById("fileFolderInput");

const fileSizeInput = document.getElementById("fileSizeInput");

const addButton = document.getElementById("sendFile");

const deleteButton = document.getElementById("deleteButton");

const deleteId = document.getElementById("deleteId");

const diskDiv = document.getElementById("diskDiv");

const defragmentationButton = document.getElementById("defragmentationButton");

const counterFreeSpace = document.getElementById("counterFreeSpace");

const blockSize = 4;

const disk = [];

disk.length = 44;

disk.fill(0);

class Block {

constructor(name, type, size) {

this.name = name;

this.type = type;

this.size = size;

}

name;

folder;

address;

type;

size;

nextAddress;

position="";

}

addButton.addEventListener("click", () => addFile({fileName: fileNameInput.value, folderName: fileFolderInput.value}, fileSizeInput.value));

deleteButton.addEventListener("click", () => removeFile(deleteId.value));

const getName = ({fileName, folderName }) => {

if (folderName) {

return `${folderName}/${fileName}`

}

return fileName

}

let hashTable = new Map();

function addFile(fileMetaData, size) {

let previousAddress;

const fileName = getName(fileMetaData);

for(let i = 0; i<size; i+=blockSize) {

const block = new Block(fileName, "data",size-i>blockSize ? blockSize : size-i);

const address = findAddress();

if (address === -1) {

console.error('Disk has no space to record this file');

return;

}

disk[address]=block;

block.address = address;

if(typeof (previousAddress) === "number")

disk[previousAddress].nextAddress=address;

if (i+1===size) {

block.position = "EOF";

}

if(i===0) {

block.position = 0;

block.folder = fileMetaData.folderName;

hashTable.set(fileName, address);

} else {

block.position = i;

}

previousAddress=address;

}

draw();

}

function findAddress() {

let i=0;

while (i<disk.length) {

if(disk[i] === 0) {

return i;

}

i++;

}

return -1;

}

function removeFile(fileName) {

let deleteAddress = hashTable.get(fileName);

if (deleteAddress === undefined) {

console.error('Such file doesn\'t exist');

draw();

return

}

hashTable.delete(fileName);

while (true) {

const nextAddress = disk[deleteAddress].nextAddress;

disk[deleteAddress] = 0;

deleteAddress = nextAddress;

if (deleteAddress === undefined) {

draw();

return;

}

}

}

//Отрисовка интерфейса

const getFiles = () => {

const files = [];

[...hashTable.keys()].forEach(fullPath => {

if (fullPath.indexOf('/') > -1) {

const [folderName, fileName] = fullPath.split('/')

files.push({ fileName, folderName })

} else {

files.push({ fileName: fullPath, folderName: null })

}

})

return files

}

const draw = () => {

const files = getFiles();

const folders = {};

root.innerHTML = '';

const list = document.createElement('ul');

list.id = "list";

root.appendChild(list);

files.forEach(({ fileName, folderName }) => {

const file = document.createElement('li');

file.innerText = fileName

if (folderName) {

if (!folders[folderName]) {

const folderWrapper = document.createElement('li');

folderWrapper.className = "folderLi"

const folderElement = document.createElement('ul');

folderWrapper.innerText = folderName;

folderWrapper.appendChild(folderElement);

list.appendChild(folderWrapper);

folders[folderName] = folderElement;

}

folders[folderName].appendChild(file)

} else {

list.appendChild(file)

}

})

drawDisk();

freeSpace();

}

function drawDisk() {

diskDiv.innerHTML = '';

for(let i = 0; i<disk.length; i++) {

const div = document.createElement("div");

div.className = "disk";

if(disk[i]) {

div.style.backgroundColor = "yellow";

div.textContent = disk[i].name;

}

diskDiv.appendChild(div);

}

}

defragmentationButton.addEventListener("click", defragmentation)

function defragmentation() {

const nextAddresses = {};

let left = 0;

let right = 1;

while (right < disk.length) {

if (right == left) {

right++;

}

if (disk[left] != 0) {

left++;

continue;

}

if (disk[right] == 0) {

right++;

continue;

}

const curr = disk[right];

if (disk[right]) {

nextAddresses[curr.nextAddress] = curr;

disk[left] = curr;

disk[right] = 0;

if (nextAddresses[right]) {

nextAddresses[right].nextAddress = left;

}

if (curr.position === 0) {

hashTable.set(curr.name, left);

}

}

}

draw();

}

function getDestructionName(i) {

if(disk[i].name.includes("/")) {

const [folderName, fileName] = disk[i].name.split('/');

return {folderName: folderName, fileName: fileName};

}

else return {fileName: disk[i].name}

}

function freeSpace( ) {

let freeSpace=disk.length\*4;

for(let i = 0; i<disk.length; i++) {

if(disk[i]!==0) {

freeSpace-=disk[i].size;

}

}

counterFreeSpace.innerText = `Свободно ${freeSpace} Кб`;

}

freeSpace()

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Листинг *HTML*-файла программы**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Title</title>

<link rel="stylesheet" href="style.css" />

</head>

<body>

<div id="form">

<label>

Введите название файла

<input type="text" id="fileNameInput" />

</label>

<label>

Введите папку (необязательно)

<input type="text" id="fileFolderInput" />

</label>

<label>

Введите размер файла

<input type="number" id="fileSizeInput" />

</label>

<button id="sendFile">Добавить файл</button>

<label>

<input type="text" id="deleteId" />

<button id="deleteButton">Удалить файл</button>

</label>

<button id="defragmentationButton">Дефрагментация</button>

</div>

<div id="root"></div>

<p id="counterFreeSpace"></p>

<div id="diskDiv"></div>

<script rel="script" src="main.js"></script>

</body>

</html>

**ПРИЛОЖЕНИЕ В**

(обязательное)

**Листинг *CSS*-файла программы**

#form {

display: grid;

justify-content: space-around;

justify-items: end;

}

#diskDiv {

display: flex;

flex-wrap: wrap;

align-items: center;

}

.disk {

display: grid;

height: 2vh;

width: 60px;

background-color: wheat;

border: 1px black solid;

justify-content: center;

}

.folderLi,

#list {

display: list-item;

}

ul li::marker {

color: red;

}