Учреждение образования Республики Беларусь

«Гомельский государственный технический университет им. П. О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Отчет по лабораторной работе №11

по дисциплине «Операционные системы»

Выполнил: студент группы ИТП-11

Половцев М. С.

Принял: преподаватель-стажер

Карась О. В.

Гомель, 2022

**Цель:** 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 учет свободного пространства;

- реализовывать поиск файлов и директорий;

- сохранять в виртуальный файл файлы с жесткого диска;

- удалять файлы из виртуального файла;

- записывать на жесткий диск файлы из виртуального файла;

- создавать в виртуальном файле текстовые файлы;

Дополнить программу вспомогательной утилитой (кэширование).

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Условие задачи | Учет свободных блоков | Поиск файлов и папок |
| 6 | Индексно-последовательная файловая система. Связанный список. | Связанный список | Хэш-таблицы |

Внешняя фрагментация может быть устранена за счет представления файла в виде связного списка блоков диска. Запись в директории содержит указатель на первый и последний блоки файла.

Хеширование - другой способ, который может использоваться для размещения и последующего поиска имени файла в директории. В данном методе имена файлов также хранятся в каталоге в виде линейного списка, но дополнительно используется хеш-таблица. Хеш-таблица, точнее построенная на ее основе хеш-функция, позволяет по имени файла получить указатель на имя файла в списке. Таким образом, можно существенно уменьшить время поиска.

Кэш диска представляет собой буфер в оперативной памяти, содержащий ряд блоков диска. Если имеется запрос на чтение/запись блока диска, то сначала производится проверка на предмет наличия этого блока в кэше. Если блок в кэше имеется, то запрос удовлетворяется из кэша, в противном случае запрошенный блок считывается в кэш с диска. Сокращение количества дисковых операций оказывается возможным вследствие присущего ОС свойства локальности

Результат создания модели файловой системы представлен на рисунке 1:



Рисунок 1 – Результат создания модели файловой системы

**Вывод:** разработал модель файловой системы.

Приложение А

Листинг программы

let indexTable = new Map();

let cach = new Map();

class block {

name;

start;

}

let memory = [];

memory.length = 40;

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

memory[i] = 0;

}

let freeSize = memory.length;

const blockSize = 4;

let blocks = [];

function addFile() {

let fileName = document.getElementById('addFileName').value;

let dirName = document.getElementById('addDirName').value;

if (dirName !== '') {

fileName = `${dirName}/${fileName}`;

}

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

if (blocks[i].name === fileName) {

alert('Файл с таким именем уже существует');

return;

}

}

let sizeInKbs = parseInt(document.getElementById('addFileSize').value);

let sizeInBlocks = Math.floor(sizeInKbs / blockSize);

if (sizeInKbs % blockSize !== 0) {

sizeInBlocks++;

}

if (sizeInBlocks <= freeSize) {

let f = 0;

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

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

let freeSeg = 0;

let start = i;

let j = i;

while (memory[j] === 0) {

freeSeg++;

j++;

}

if (sizeInBlocks <= freeSeg) {

indexTable.set(fileName, start);

cach.set(fileName, start);

for (let k = i; k < i + sizeInBlocks; k++) {

memory[k] = `${fileName}`;

blocks.push(new block);

blocks[blocks.length - 1].name = fileName;

blocks[blocks.length - 1].start = start;

start++;

}

f = 1;

break;

}

}

}

if (f === 0) {

alert('Недостаточно памяти для добавления файла');

} else {

freeSize -= sizeInBlocks;

document.getElementById('freeSize').innerText = `${freeSize \* 4} Кб`;

createTable();

}

} else {

alert('Недостаточно памяти для добавления файла');

}

}

let addFileButton = document.getElementById('addFile');

addFileButton.addEventListener('click', addFile);

function deleteFile() {

let fileName = document.getElementById('delFileName').value;

let start = cach.get(fileName);

cach.delete(fileName);

if (start === undefined) {

start = indexTable.get(fileName);

indexTable.delete(fileName);

if (start === undefined) {

alert('Файл не найден');

return;

}

} else {

let size = 0;

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

if (blocks[i].name === fileName) {

size++;

blocks.splice(i, 1);

i--;

}

}

for (let i = start; i < start + size; i++) {

memory[i] = 0;

}

freeSize += size;

document.getElementById('freeSize').innerText = `${freeSize \* 4} Кб`;

createTable();

}

}

let delFileButton = document.getElementById('deleteFile');

delFileButton.addEventListener('click', deleteFile);

function createTable() {

document.getElementById('table').innerHTML = '';

let table = document.createElement('table');

table.setAttribute('border', '1');

table.innerHTML = '<tr>';

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

let td = document.createElement('td');

td.setAttribute('width', `50px`);

td.setAttribute('height', `25px`);

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

td.setAttribute('bgcolor', `#00FF00`);

td.innerText = memory[i];

}

table.append(td);

if (i === memory.length / 2 - 1) {

table.innerHTML += '</tr><tr>';

}

}

table.innerHTML += '</tr>';

document.getElementById('table').append(table);

}