Учреждение образования
«Гомельский государственный технический университет имени П. О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10,11
По дисциплине «Операционные системы»

«Реализация файловой системы. Управление файловой системой»

Выполнил: студент гр. ИТ-11

Иванов А.А.

Принял: преподаватель

Петров И.И.

Гомель 2021

Цель работы:

Разработать модель файловой системы. Реализовать дополнительные утилиты для разработанной модели.

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;
- Предоставлять возможность редактировать текстовые файлы внутри виртуального файла.

Вариант 10:

Условие: Индексно – последовательная файловая система. Связанный список.

Учет свободных блоков: Битовый вектор.

Поиск: B-дерево.

Дополнить программу утилитой: Управление доступом.

Листинг FileSystemEmulator.cs:

```
using System;
using System.Windows.Forms;

using lab11.Forms;

namespace lab11
{


    /// <summary>
    /// Файловая система
    /// </summary>
    static class FileSystemEmulator
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
```

```
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new FMain());
        }
    }
}
```

Листинг FileSystem.cs

```
using lab7.Classes.FileSystem.StructElements;

namespace lab11.Classes.FileSystem
{
    /// <summary>
    /// Класс, реализующий файловую систему
    /// </summary>
    public class FileSystem
    {
        public Superblock SB;                   // Суперблок файловой системы
        public FreeMemoryController FMC;        // Контроллер свободного пространства
        public DataBlocks DB;                   // Данные диска
        public UDirectory Root;                 // Корневой каталог
        public const int FSSize = 10240;        // Размер виртуального пространства

        /// <summary>
        /// Конструктор с параметрами
        /// </summary>
        /// <param name="sCluster">Размер кластера</param>
        public FileSystem(int sCluster)
        {
            SB = new Superblock(sCluster);
            FMC = new FreeMemoryController(SB.Size);
            DB = new DataBlocks(SB.Size, SB.Cluster);
            Root = new UDirectory("Virtual Drive");
            Root.Dirs.Add(new UDirectory("C:\\"));
            Root.Dirs.Add(new UDirectory("D:\\"));
        }

        /// <summary>
        /// Резервация блоков под файл
        /// </summary>
        /// <param name="UFile">Файл</param>
        public UFile WriteBlocks(UFile f)
        {
            long numReservationBlocks = f.Size / (SB.Cluster * 1024) + 1;
            f.Blocks = new int[numReservationBlocks];

            for (int i = 0, j = 0; i < SB.Size && numReservationBlocks > 0; i++)
            {
                if (FMC.BitVector[i])
                {
                    FMC.BitVector[i] = false;
                    f.Blocks[j] = i;
                    numReservationBlocks--;
                    j++;
                }
            }

            return f;
        }

        /// <summary>
        /// Удаление файла
```

```
        /// </summary>
        public void DeleteFile(UFile f)
        {
            for (int i = 0; i < f.Blocks.Length; i++)
            {
                DB.Data[f.Blocks[i]] = new Block(SB.Cluster);
                FMC[f.Blocks[i]] = true;
            }

        }
    }
}
```

Листинг Superblock.cs

```
namespace lab11.Classes.FileSystem.StructElements
{
    /// <summary>
    /// Класс, реализующий суперблок
    /// с информацией о файловой системе
    /// </summary>
    public class Superblock
    {
        public int Size;        // Размер файловой системы в блоках
        public int Cluster;     // Размер кластера

        /// <summary>
        /// Конструктор с параметрами
        /// </summary>
        /// <param name="s">Размер файловой системы</param>
        /// <param name="c">Размер кластера</param>
        public Superblock(int c)
        {
            Size = FileSystem.FSSize / c;
            Cluster = c;
        }
    }
}
```

Листинг FreeMemoryController.cs

```
namespace lab11.Classes.FileSystem.StructElements
{
    /// <summary>
    /// Класс, реализующий управление
    /// свободным дисковым пространством
    /// </summary>
    public class FreeMemoryController
    {
        public bool[] BitVector;    // Битовый вектор

        /// <summary>
        /// Конструктор с параметрами
        /// </summary>
        /// <param name="size">Размер файловой системы в блоках</param>
        public FreeMemoryController(int size)
        {
            BitVector = new bool[size];

            for (int i = 0; i < size; i++) BitVector[i] = true;
        }
```

```
        /// <summary>
        /// Индексатор
        /// </summary>
        /// <param name="i">Индекс</param>
        /// <returns>Значение битового вектора</returns>
        public bool this[int i]
        {
            set { BitVector[i] = value; }
            get { return BitVector[i]; }
        }

        /// <summary>
        /// Подсчет количества свободных блоков
        /// </summary>
        /// <returns>Свободные блоки</returns>
        public int ReturnFreeBlocks()
        {
            int numFreeBlocks = 0;
            for (int i = 0; i < BitVector.Length; i++)
            {
                if (BitVector[i]) numFreeBlocks++;
            }
            return numFreeBlocks;
        }
    }
}
```

Листинг DataBlocks.cs:

```
using System.Collections.Generic;

namespace lab11.Classes.FileSystem.StructElements
{
    /// <summary>
    /// Класс, реализующий хранение данных
    /// </summary>
    public class DataBlocks
    {
        public Block[] Data;    // Блоки на диске

        /// <summary>
        /// Конструктор без параметров
        /// </summary>
        public DataBlocks(int s, int blockSize)
        {
            Data = new Block[s];
            for (int i = 0; i < s; i++) Data[i] = new Block(blockSize);
        }

        /// <summary>
        /// Индексатор
        /// </summary>
        /// <param name="i">Индекс</param>
        /// <returns>Блок</returns>
        public Block this[int i]
        {
            set { Data[i] = value; }
            get { return Data[i]; }
        }
    }
}
```

Листинг Block.cs:

```
namespace lab11.Classes.FileSystem.StructElements
{
    /// <summary>
    /// Класс, реализующий блок на диске
    /// </summary>
    public class Block
    {
        public int Size;        // Размер блока
        public char[] Data;     // Данные блока

        /// <summary>
        /// Конструктор с параметрами
        /// </summary>
        /// <param name="s">Размер блока в кб</param>
        public Block(int s)
        {
            Size = s * 1024;

            Data = new char[Size];
            for(int i=0;i<Size;i++) Data[i] = '\0';
        }

        /// <summary>
        /// Получение данных блока
        /// </summary>
        /// <returns>Данные блока</returns>
        public string GetData()
        {
            string res = "";

            for (int i = 0; i < Data.Length; i++)
            {
                if (Data[i] != '\0') res += Data[i];
            }

            return res;
        }
    }
}
```

Листинг UDirectory.cs:

```
using System;
using System.Collections.Generic;

namespace lab7.Classes.FileSystem
{
    /// <summary>
    /// Класс, реализующий директорию
    /// </summary>
    public class UDirectory
    {
        public string Name;             // Имя
        public List<UDirectory> Dirs;   // Директории данной директории
        public List<UFile> Files;       // Файлы директории
        public DateTime LastChanged;    // Дата последних изменений

        public bool isWrite;            // true - доступен для записи
        public bool isDelete;           // true - доступен для удаления

        /// <summary>
```

```
        /// Конструктор с параметрами
        /// </summary>
        /// <param name="n">Имя</param>
        public UDirectory(string n)
        {
            Name = n;
            Dirs = new List<UDirectory>();
            Files = new List<UFile>();
            LastChanged = DateTime.Now;

            isWrite = true;
            isDelete = true;
        }
    }
}
```

Листинг UFile.cs:

```
using System;

namespace lab11.Classes.FileSystem
{
    /// <summary>
    /// Класс, реализующий файл
    /// </summary>
    public class UFile
    {
        public string Name;             // Имя
        public long Size;               // Размер
        public int[] Blocks;            // Блоки файла
        public DateTime LastChanged;    // Дата последних изменений

        public bool isRead;             // true - доступен для чтения
        public bool isWrite;            // true - доступен для записи
        public bool isDelete;           // true - доступен для удаления

        /// <summary>
        /// Конструктор с параметрами
        /// </summary>
        /// <param name="n">Имя</param>
        /// <param name="s">Размер</param>
        /// <param name="cluster">Размер кластера</param>
        public UFile(string n, long s, int cluster, DateTime lc)
        {
            Name = n;
            Size = s * 1024;
            Blocks = new int[Size / cluster];
            LastChanged = lc;

            isRead = true;
            isWrite = true;
            isDelete = true;
        }
    }
}
```

Листинг FInit.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using lab11.Classes.FileSystem;

namespace lab11.Forms
{
    /// <summary>
    /// Форма для запроса о форматировании
    /// </summary>
    public partial class FInit : Form
    {
        public bool State;          // Состояние форматирования
        public int ClusterSize;     // Размер кластера

        /// <summary>
        /// Конструктор
        /// </summary>
        public FInit()
        {
            InitializeComponent();

            State = false;
            ClusterSize = 4;

            textBox1.Text = ClusterSize.ToString();
        }

        /// <summary>
        /// Событие кнопки форматировать
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                ClusterSize = Convert.ToInt32(textBox1.Text);

                if (ClusterSize <= 0 || ClusterSize > FileSystem.FSSize) throw new Exception();
            }
            catch
            {
                ClusterSize = 4;
                textBox1.Text = ClusterSize.ToString();
                return;
            }

            State = true;
            this.Close();
        }
    }
}
```

Листинг FEdit.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using lab11.Classes.FileSystem;

namespace lab11.Forms
{
    /// <summary>
    /// Форма редактора текстовых файлов
    /// </summary>
    public partial class FEdit : Form
    {
        public FMain Main;          // Ссылка на главную форму
        public UFile file;          // Поле - файл
        public string NewFileData;  // Новые данные файла

        /// <summary>
        /// Конструктор
        /// </summary>
        /// <param name="f">Файл для редактирования</param>
        public FEdit(FMain main, UFile f)
        {
            Main = main;
            file = f;
            InitializeComponent();
            this.Text = f.Name;

            string data = "";
            for (int i = 0; i < file.Blocks.Length; i++)
            {
                data += Main.FS.DB[file.Blocks[i]].GetData();
            }
            richTextBox1.Text = data;
        }

        /// <summary>
        /// "Сохранить и выйти"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button1_Click(object sender, EventArgs e)
        {
            if (!file.isWrite)
            {
                MessageBox.Show(
                    "Файл недоступен для записи!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                NewFileData = "NULL";
                return;
            }
            NewFileData = richTextBox1.Text;
            Close();
        }
    }
}
```

Листинг FFeatures.cs:

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using lab11.Classes.FileSystem;

namespace lab11.Forms
{
    /// <summary>
    /// Форма свойства файлов и папок
    /// </summary>
    public partial class FFeatures : Form
    {
        public UFile file;          // Поле - файл
        public UDirectory dir;      // Поле - директория

        /// <summary>
        /// Конструктор
        /// </summary>
        /// <param name="main">Главная форма</param>
        /// <param name="f">Файл</param>
        /// <param name="d">Директория</param>
        public FFeatures(UFile f, UDirectory d)
        {
            InitializeComponent();

            file = f;
            dir = d;
            if (file != null)
            {
                label1.Text = "Свойства: " + file.Name;
                checkBox1.Checked = file.isRead;
                checkBox2.Checked = file.isWrite;
                checkBox3.Checked = file.isDelete;
            }
            else
            {
                label1.Text = "Свойства: " + dir.Name;
                checkBox1.Checked = true;
                checkBox1.Enabled = false;
                checkBox2.Checked = dir.isWrite;
                checkBox3.Checked = dir.isDelete;
            }
        }

        /// <summary>
        /// Обработчик кнопи ОК
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button1_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Text.RegularExpressions;
using System.Xml.Serialization;

using lab7.Classes.FileSystem;
using lab7.Classes.FileSystem.StructElements;

namespace lab7.Forms
{
    /// <summary>
    /// Главная форма
    /// </summary>
    public partial class FMain : Form
    {
        public FileSystem FS;  // Объект класса файловая система

        /// <summary>
        /// Конструктор
        /// </summary>
        public FMain()
        {
            InitializeComponent();

            FInit f = new FInit();
            f.ShowDialog();

            if (f.State)
            {
                FS = new FileSystem(f.ClusterSize);
            }
            else
            {
                MessageBox.Show(
                    "Вы не выбрали размер кластера, поэтому он установлен по умолчанию (4кб)!",
                    "Предупреждение",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Warning);
                FS = new FileSystem(4);
            }

            UpdateInf();
            UpdateTreeView();
        }

        #region Обработчики интерфейса

        /// <summary>
        /// Обновление информации о дисковом пространстве
        /// </summary>
        private void UpdateInf()
        {
            int numFreeBlocks = FS.FMC.ReturnFreeBlocks();
            int freeSize = FS.SB.Cluster * numFreeBlocks;

            label1.Text = "Свободно: " + freeSize + " кб";
            label2.Text = "Размер кластера: " + FS.SB.Cluster + " кб";
        }

        /// <summary>
        /// Обновление дерева каталогов
```

```
/// </summary>
private void UpdateTreeView()
{
    treeView1.Nodes.Clear();

    TreeNode rootNode;

    rootNode = new TreeNode(FS.Root.Name);
    rootNode.Tag = FS.Root;
    GetDirectories(FS.Root.Dirs, rootNode);
    treeView1.Nodes.Add(rootNode);
}

/// <summary>
/// Получение поддиректорий данной директории
/// </summary>
/// <param name="subDirs"></param>
/// <param name="nodeToAddTo"></param>
private void GetDirectories(List<UDirectory> subDirs, TreeNode nodeToAddTo)
{
    TreeNode aNode;
    List<UDirectory> subSubDirs;
    foreach (UDirectory subDir in subDirs)
    {
        aNode = new TreeNode(subDir.Name, 0, 0);
        aNode.Tag = subDir;
        aNode.ImageKey = "folder";
        subSubDirs = subDir.Dirs;
        if (subSubDirs.Count != 0)
        {
            GetDirectories(subSubDirs, aNode);
        }
        nodeToAddTo.Nodes.Add(aNode);
    }
}

/// <summary>
/// События дерева каталогов
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void treeView1_NodeMouseClick(object sender, TreeNodeMouseClickEventArgs e)
{
    TreeNode newSelected = e.Node;
    listView1.Items.Clear();
    UDirectory nodeDirInfo = (UDirectory)newSelected.Tag;
    ListViewItem.ListViewSubItem[] subItems;
    ListViewItem item = null;

    foreach (UDirectory dir in nodeDirInfo.Dirs)
    {
        item = new ListViewItem(dir.Name, 0);
        item.Tag = dir;
        subItems = new ListViewItem.ListViewSubItem[]
            {new ListViewItem.ListViewSubItem(item, "Каталог"),
             new ListViewItem.ListViewSubItem(item,
                                dir.LastChanged.ToShortDateString())};
        item.SubItems.AddRange(subItems);
        listView1.Items.Add(item);
    }
    foreach (UFile file in nodeDirInfo.Files)
    {
        item = new ListViewItem(file.Name, 1);
        item.Tag = file;
        subItems = new ListViewItem.ListViewSubItem[]
```

```
            { new ListViewItem.ListViewSubItem(item, "Файл"),
             new ListViewItem.ListViewSubItem(item,
                                    file.LastChanged.ToShortDateString())};

        item.SubItems.AddRange(subItems);
        listView1.Items.Add(item);
    }

    listView1.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);
}

#endregion

#region Опции

/// <summary>
/// Форматирование
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button1_Click(object sender, EventArgs e)
{
    FInit f = new FInit();
    f.ShowDialog();

    if (f.State)
    {
        FS = new FileSystem(f.ClusterSize);
    }
    else
    {
        MessageBox.Show(
            "Форматирование не было произведено!",
            "Предупреждение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Warning);
        return;
    }

    UpdateInf();
    UpdateTreeView();
}

/// <summary>
/// Сохранить файл с диска
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button2_Click(object sender, EventArgs e)
{
    if (treeView1.SelectedNode == null) { return; }
    UDirectory checker = (UDirectory)treeView1.SelectedNode.Tag;
    if (!checker.isWrite)
    {
        MessageBox.Show(
            "Директория защищена от записи!",
            "Ошибка",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        return;
    }

    OpenFileDialog ofd = new OpenFileDialog();
    ofd.ShowDialog();
```

```
            FileInfo file;
            try { file = new FileInfo(ofd.FileName); }
            catch { return; }
            if (file.Length / 1024 > FS.FMC.ReturnFreeBlocks() * FS.SB.Cluster)
            {
                MessageBox.Show(
                    "Недостаточно свободного места",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                return;
            }
            long fSize = file.Length / (1024 * FileSystem.FSSize);
            UFile ufile = new UFile(file.Name, (file.Length / 1024 == 0) ? 1 : file.Length /
1024, FS.SB.Cluster, file.LastAccessTime);
            ufile = FS.WriteBlocks(ufile);
            if (file.Extension == ".txt")
            {
                StreamReader sr = new StreamReader(file.OpenRead());
                string fileData = sr.ReadToEnd();

                for (int i = 0, j = 0; i < ufile.Blocks.Length && j < fileData.Length; i++)
                {
                    for (int k = 0; k < FS.DB.Data[0].Data.Length && j < fileData.Length; k++)
                    {
                        FS.DB[ufile.Blocks[i]].Data[k] = fileData[j];
                        j++;
                    }
                }
                sr.Close();
            }
            else
            {
                BinaryReader br = new BinaryReader(file.OpenRead());
                string fData = "";
                string x = br.ReadString();
                try
                {
                    while (true)
                    {
                        fData += x;
                        x = br.ReadString();
                    }
                }
                catch { }

                br.Close();

                for (int i = 0, j = 0; i < ufile.Blocks.Length && j < fData.Length; i++)
                {
                    for (int k = 0; k < FS.DB.Data[0].Data.Length && j < fData.Length; k++)
                    {
                        FS.DB[ufile.Blocks[i]].Data[k] = fData[j];
                        j++;
                    }
                }
            }

            //if (treeView1.SelectedNode == null) treeView1.SelectedNode =
treeView1.Nodes[0].Nodes[0];

            UDirectory d = (UDirectory)treeView1.SelectedNode.Tag;
            d.Files.Add(ufile);
            treeView1.SelectedNode.Tag = d;
```

```
    UpdateInf();
    UpdateTreeView();
}

/// <summary>
/// Запись файла на диск
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button3_Click(object sender, EventArgs e)
{
    ListViewItem itm;
    try
    {
        itm = this.listView1.SelectedItems[0];
    }
    catch { return; }

    if (itm.ImageIndex == 0) return;

    UFile file = (UFile)itm.Tag;
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.ShowDialog();

    FileInfo writeFile;

    Regex rExt = new Regex(@"(\w*)\.txt");
    MatchCollection cExt;
    try { cExt = rExt.Matches(file.Name); }
    catch { return; }
    if (cExt.Count > 0)
    {
        writeFile = new FileInfo(sfd.FileName + ".txt");

        StreamWriter sr = new StreamWriter(writeFile.OpenWrite());

        try
        {
            string data = "";
            for (int i = 0; i < file.Blocks.Length; i++)
            {
                data += FS.DB[file.Blocks[i]].GetData();
            }

            sr.Write(data);
        }
        catch
        {
            MessageBox.Show(
                "Данные не были записаны.",
                "Ошибка",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }

        sr.Close();

        return;
    }

    writeFile = new FileInfo(sfd.FileName + ".userfile");
    BinaryWriter bw = new BinaryWriter(writeFile.OpenWrite());

    try
    {
```

```
                string data = "";
                for (int i = 0; i < file.Blocks.Length; i++)
                {
                    data += FS.DB[file.Blocks[i]].GetData();
                }

                bw.Write(data);
            }
            catch
            {
                MessageBox.Show(
                    "Данные не были записаны.",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
            bw.Close();
        }

        #endregion

        #region Действия

        /// <summary>
        /// Создание каталога
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button5_Click(object sender, EventArgs e)
        {
            if (textBox2.Text == "")
            {
                MessageBox.Show(
                    "Введите имя каталога.",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                return;
            }
            if (treeView1.SelectedNode == null)
            {
                MessageBox.Show(
                    "Выберите место для создания каталога!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                return;
            }
            UDirectory checker = (UDirectory)treeView1.SelectedNode.Tag;
            if (!checker.isWrite)
            {
                MessageBox.Show(
                    "Директория защищена от записи!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                return;
            }

            UDirectory addDir = new UDirectory(textBox2.Text);

            //if (treeView1.SelectedNode == null) treeView1.SelectedNode =
treeView1.Nodes[0].Nodes[0];
            UDirectory d = (UDirectory)treeView1.SelectedNode.Tag;
            for (int i = 0; i < d.Dirs.Count; i++)
```

```
            {
                if (d.Dirs[i].Name == addDir.Name)
                {
                    MessageBox.Show(
                        "Каталог с таким именем уже существует!",
                        "Ошибка",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
                    return;
                }
            }
            d.Dirs.Add(addDir);
            treeView1.SelectedNode.Tag = d;

            UpdateInf();
            UpdateTreeView();
        }

        /// <summary>
        /// Создание txt файла
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button6_Click(object sender, EventArgs e)
        {
            if (textBox2.Text == "")
            {
                MessageBox.Show(
                    "Введите имя txt файла",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                return;
            }
            if (treeView1.SelectedNode == null)
            {
                MessageBox.Show(
                    "Выберите место для создания текстового файла!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                return;
            }
            UDirectory checker = (UDirectory)treeView1.SelectedNode.Tag;
            if (!checker.isWrite)
            {
                MessageBox.Show(
                    "Директория защищена от записи!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                return;
            }

            UFile newTxt = new UFile(textBox2.Text + ".txt", 1, FS.SB.Cluster, DateTime.Now);
            FS.WriteBlocks(newTxt);

            //if (treeView1.SelectedNode == null) treeView1.SelectedNode =
treeView1.Nodes[0].Nodes[0];
            UDirectory d = (UDirectory)treeView1.SelectedNode.Tag;
            for (int i = 0; i < d.Files.Count; i++)
            {
                if (d.Files[i].Name == newTxt.Name)
                {
                    MessageBox.Show(
```

```
                "Файл с таким именем уже существует!",
                "Ошибка",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            return;
        }
    }
    d.Files.Add(newTxt);
    treeView1.SelectedNode.Tag = d;

    UpdateInf();
    UpdateTreeView();
}

/// <summary>
/// Удалить
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button7_Click(object sender, EventArgs e)
{
    ListViewItem itm;
    try
    {
        itm = this.listView1.SelectedItems[0];
    }
    catch { return; }

    if (itm.ImageIndex == 0)
    {
        UDirectory dir = (UDirectory)itm.Tag;
        if (!dir.isDelete)
        {
            MessageBox.Show(
                "Каталог защищен!",
                "Ошибка",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            return;
        }
        if (dir.Files.Count != 0 || dir.Dirs.Count != 0)
        {
            if (MessageBox.Show(
                "Каталог не пустой, вы действительно хотите его удалить?",
                "Действие",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Warning) == DialogResult.No)
            {
                return;
            }
            else
            {
                DeleteDirRecursion(dir);
            }
        }
        else
        {
            DeleteDirRecursion(dir);
        }
    }
    else
    {
        UFile f = (UFile)itm.Tag;
        if (!f.isDelete)
        {
```

```
                MessageBox.Show(
                    "Файл защищен!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                return;
            }
            if (MessageBox.Show(
                    "Вы действительно хотите удалить этот файл?",
                    "Действие",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Warning) == DialogResult.No)
            {
                return;
            }
            else
            {
                DeleteFileRecursion(FS.Root, f.Name);
            }
        }

        UpdateInf();
        UpdateTreeView();
    }

    /// <summary>
    /// Удаление директории рекурсивно
    /// </summary>
    /// <param name="root">Директория для удаления</param>
    private void DeleteDirRecursion(UDirectory root)
    {
        for (int i = 0; i < root.Files.Count; i++)
        {
            FS.DeleteFile(root.Files[i]);
        }

        for (int i = 0; i < root.Dirs.Count; i++)
        {
            DeleteDirRecursion(root.Dirs[i]);
        }

        DeleteDirRecursion(FS.Root, root.Name);
    }

    /// <summary>
    /// Удаление директории по ключу рекурсивно
    /// </summary>
    /// <param name="root">Корневая директория</param>
    /// <param name="key">Ключ</param>
    private void DeleteDirRecursion(UDirectory root, string key)
    {
        int f = -1;
        for (int i = 0; i < root.Dirs.Count; i++)
        {
            if (root.Dirs[i].Name == key) { f = i; break; }
        }
        if (f != -1) { root.Dirs.RemoveAt(f); return; }
        else
        {
            for (int i = 0; i < root.Dirs.Count; i++)
            {
                DeleteDirRecursion(root.Dirs[i], key);
            }
        }
    }
```

```
/// <summary>
/// Удаление файла с рекурсивным поиском
/// </summary>
/// <param name="root">Корневая директория</param>
/// <param name="key">Имя удаляемого файла</param>
private void DeleteFileRecursion(UDirectory root, string key)
{
    int f = -1;
    for (int i = 0; i < root.Files.Count; i++)
    {
        if (root.Files[i].Name == key) { f = i; break; }
    }
    if (f != -1) { FS.DeleteFile(root.Files[f]); root.Files.RemoveAt(f); return; }
    else
    {
        for (int i = 0; i < root.Dirs.Count; i++)
        {
            DeleteFileRecursion(root.Dirs[i], key);
        }
    }
}

/// <summary>
/// Вызов свойств файла/каталога
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button9_Click(object sender, EventArgs e)
{
    FFeatures features;

    ListViewItem itm;
    try
    {
        itm = this.listView1.SelectedItems[0];
    }
    catch { return; }

    if (itm.ImageIndex == 1)
    {
        UFile f = (UFile)itm.Tag;
        features = new FFeatures(f, null);
        features.ShowDialog();

        f.isRead = features.checkBox1.Checked;
        f.isWrite = features.checkBox2.Checked;
        f.isDelete = features.checkBox3.Checked;
    }
    else
    {
        UDirectory d = (UDirectory)itm.Tag;
        features = new FFeatures(null, d);
        features.ShowDialog();

        d.isWrite = features.checkBox2.Checked;
        d.isDelete = features.checkBox3.Checked;
    }
}

/// <summary>
/// Редактирование txt файла
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
```

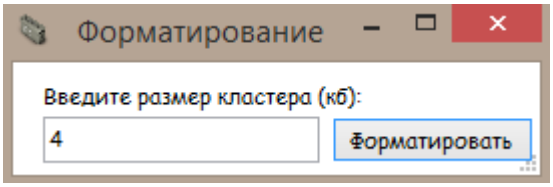
```
private void button8_Click(object sender, EventArgs e)
{
    FEdit fEdit;

    ListViewItem itm;
    try
    {
        itm = this.listView1.SelectedItems[0];
    }
    catch { return; }

    if (itm.ImageIndex == 1)
    {
        UFile file = (UFile)itm.Tag;
        Regex rExt = new Regex(@"(\w+)\.txt");
        MatchCollection cExt;
        try { cExt = rExt.Matches(file.Name); }
        catch { return; }
        if (cExt.Count > 0)
        {
            if (!file.isRead)
            {
                MessageBox.Show(
                    "Файл недоступен для чтения!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                return;
            }
            fEdit = new FEdit(this, file);
            fEdit.ShowDialog();

            string newData;
            int newSize;
            try
            {
                newData = fEdit.NewFileData;
                if (newData == "NULL") throw new Exception();
                newSize = newData.Length;
            }
            catch { return; }
            file.Size = newSize;
            FS.WriteBlocks(file);

            for (int i = 0, j = 0; i < file.Blocks.Length && j < newData.Length; i++)
            {
                for (int k = 0; k < FS.DB.Data[0].Data.Length && j < newData.Length; k++)
                {
                    FS.DB[file.Blocks[i]].Data[k] = newData[j];
                    j++;
                }
            }
        }
    }

    UpdateInf();
}

#endregion

#region Поиск

/// <summary>
/// Поиск
/// </summary>
```

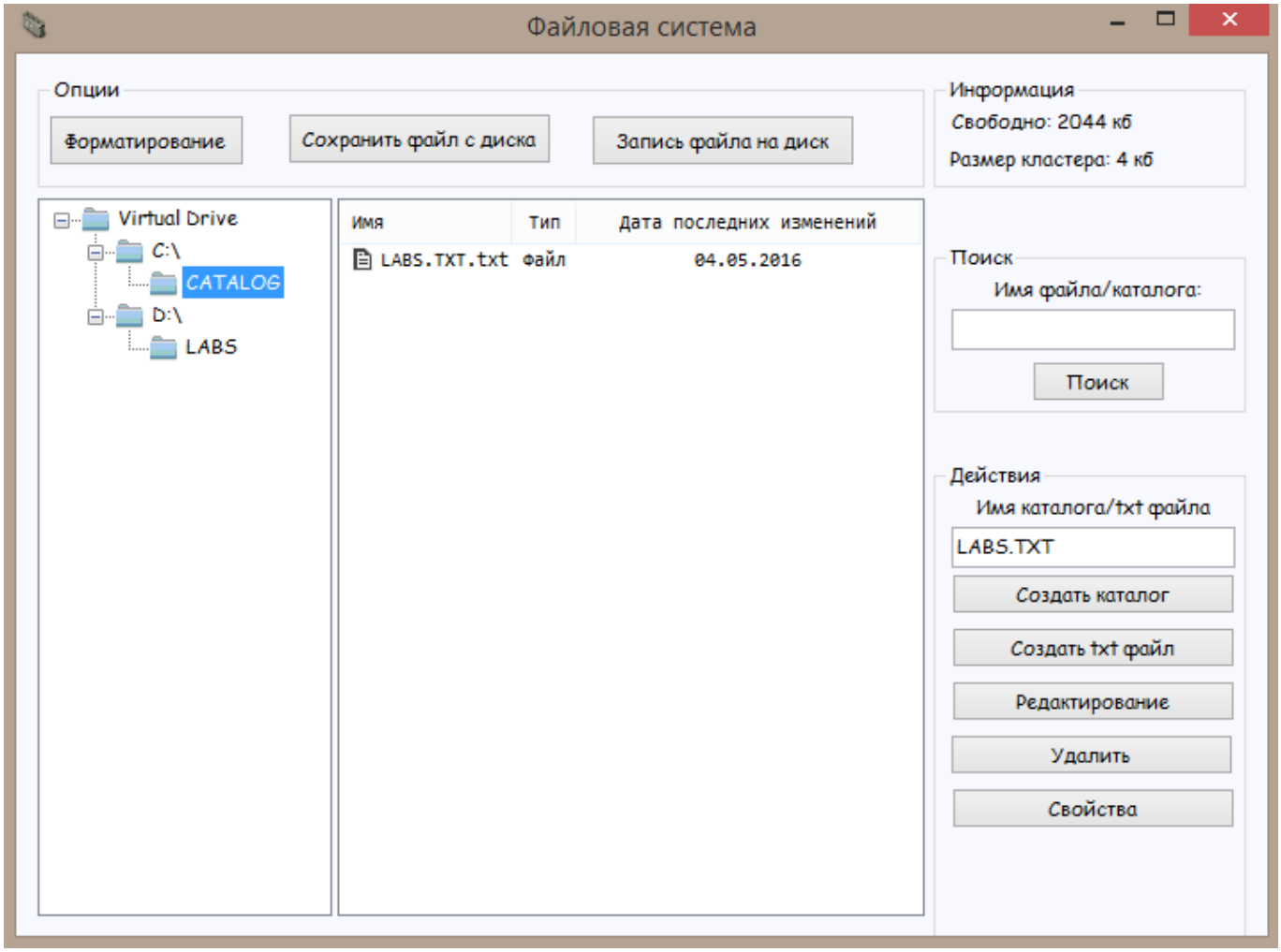
```
/// <param name="sender"></param>
/// <param name="e"></param>
private void button4_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "") return;

    UFile succesFile = null;
    UDirectory succesDir = null;

    Regex rExt = new Regex(@"(\w+)\.(\w+)");
    MatchCollection cExt;
    try { cExt = rExt.Matches(textBox1.Text); }
    catch { return; }
    if (cExt.Count > 0)
    {
        succesFile = SearchFileRecursion(FS.Root, textBox1.Text);
        if (succesFile == null)
        {
            MessageBox.Show(
                "Файла с таким именем не найдено",
                "Информация",
                MessageBoxButtons.OK,
                MessageBoxIcon.Information);
            return;
        }

        listView1.Items.Clear();
        ListViewItem.ListViewSubItem[] subItems;
        ListViewItem item = null;

        item = new ListViewItem(succesFile.Name, 1);
        item.Tag = succesFile;
        subItems = new ListViewItem.ListViewSubItem[]
        { new ListViewItem.ListViewSubItem(item, "Файл"),
            new ListViewItem.ListViewSubItem(item,
                        succesFile.LastChanged.ToShortDateString())};

        item.SubItems.AddRange(subItems);
        listView1.Items.Add(item);

        listView1.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);

        return;
    }
    else
    {
        succesDir = SearchDirRecursion(FS.Root, textBox1.Text);
        if (succesDir == null)
        {
            MessageBox.Show(
                "Директории с таким именем не найдено",
                "Информация",
                MessageBoxButtons.OK,
                MessageBoxIcon.Information);
            return;
        }

        listView1.Items.Clear();
        ListViewItem.ListViewSubItem[] subItems;
        ListViewItem item = null;

        item = new ListViewItem(succesDir.Name, 0);
        item.Tag = succesDir;
        subItems = new ListViewItem.ListViewSubItem[]
        { new ListViewItem.ListViewSubItem(item, "Каталог"),
```

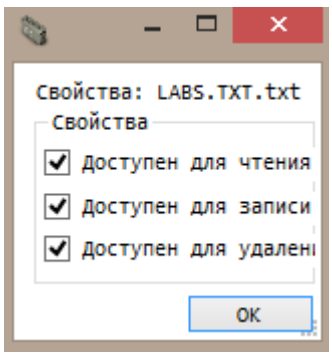
```
            new ListViewItem.ListViewSubItem(item,
                        succesDir.LastChanged.ToShortDateString())};

        item.SubItems.AddRange(subItems);
        listView1.Items.Add(item);

        listView1.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize);

        return;
    }
}

/// <summary>
/// Рекурсивный поиск файла
/// </summary>
/// <param name="root">Корень ФС</param>
/// <param name="key">Ключ поиска</param>
/// <returns>Файл</returns>
private UFile SearchFileRecursion(UDirectory root, string key)
{
    UFile f = null;
    for (int i = 0; i < root.Files.Count; i++)
    {
        if (root.Files[i].Name == key) { f = root.Files[i]; break; }
    }
    if (f != null) { return f; }
    else
    {
        for (int i = 0; i < root.Dirs.Count; i++)
        {
            UFile x = SearchFileRecursion(root.Dirs[i], key);
            if(x != null) return x;
        }
    }

    return null;
}

/// <summary>
/// Рекурсивный поиск директории
/// </summary>
/// <param name="root">Корень ФС</param>
/// <param name="key">Ключ поиска</param>
/// <returns>Директория</returns>
private UDirectory SearchDirRecursion(UDirectory root, string key)
{
    UDirectory f = null;
    for (int i = 0; i < root.Dirs.Count; i++)
    {
        if (root.Dirs[i].Name == key) { f = root.Dirs[i]; break; }
    }
    if (f != null) { return f; }
    else
    {
        for (int i = 0; i < root.Dirs.Count; i++)
        {
            UDirectory x = SearchDirRecursion(root.Dirs[i], key);
            if (x != null) return x;
        }
    }

    return null;
}

#endregion
```

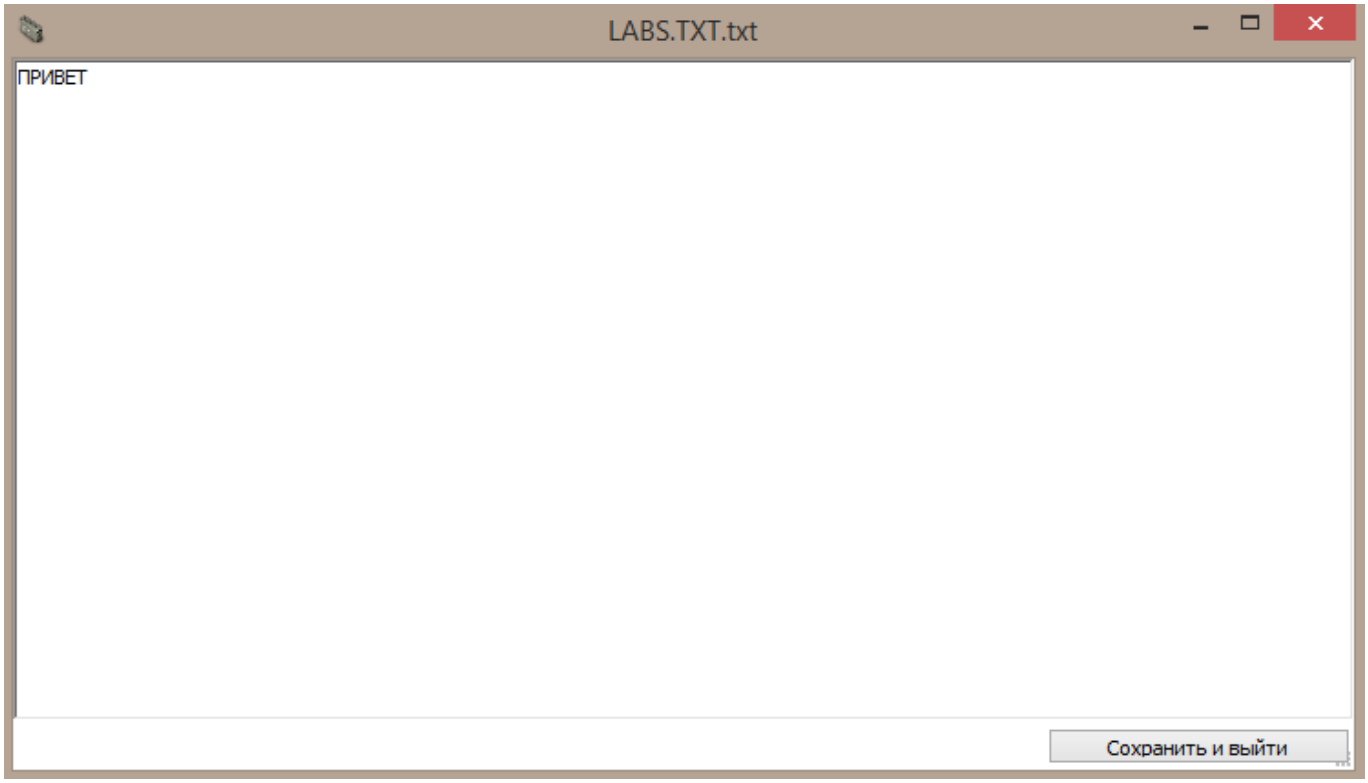
```
}
    }
```

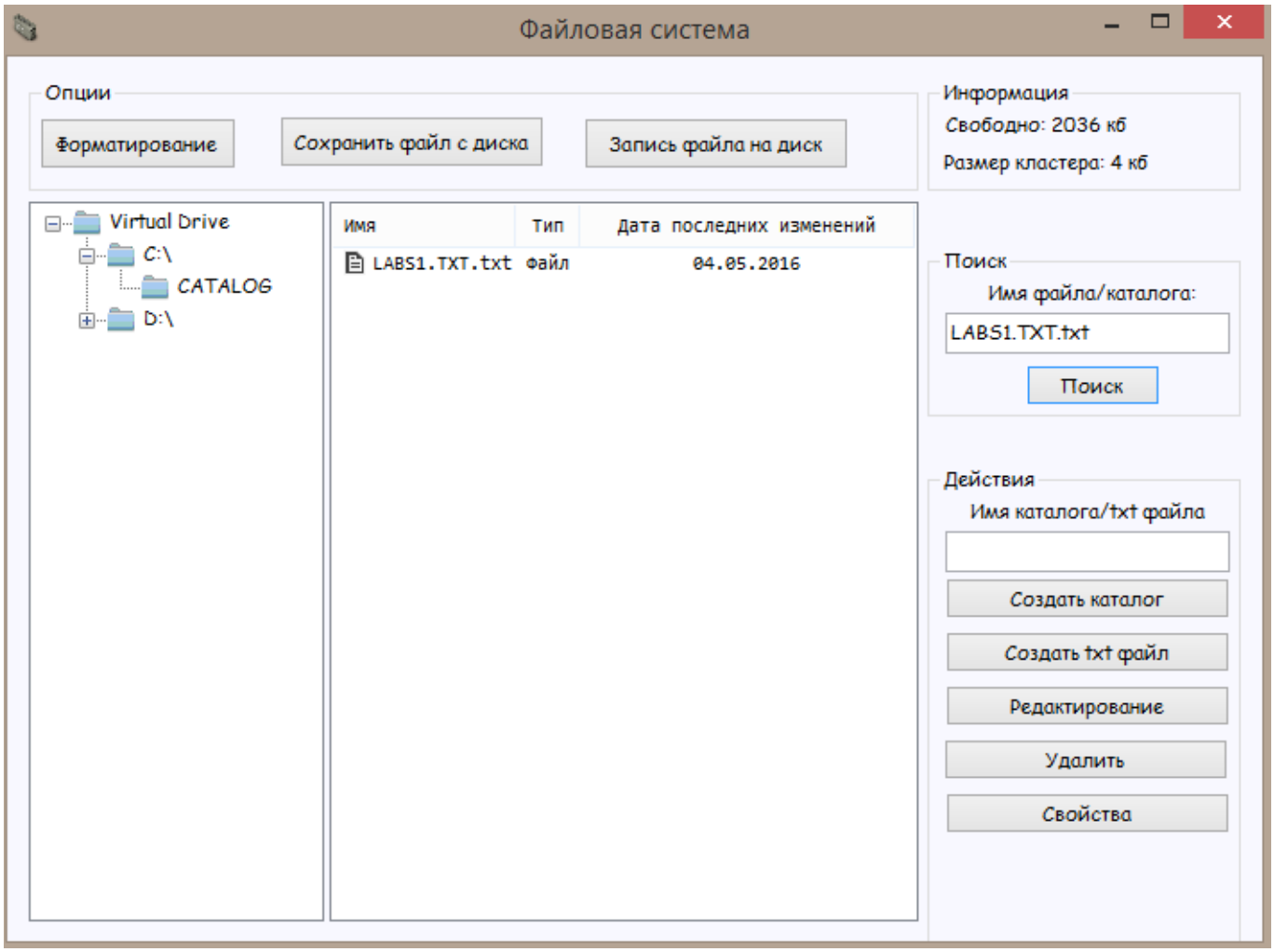
Результат работы программы:








LABS.TXT.txt
ПРИВЕТ
Сохранить и выйти


Файловая система
Опции
Форматирование
Сохранить файл с диска
Запись файла на диск
Информация
Свободно: 2036 кб
Размер кластера: 4 кб
Virtual Drive
C:\
CATALOG
D:\
Имя
Тип
Дата последних изменений
LABS1.TXT.txt
Файл
04.05.2016
Поиск
Имя файла/каталога:
LABS1.TXT.txt
Поиск
Действия
Имя каталога/txt файла
Создать каталог
Создать txt файл
Редактирование
Удалить
Свойства

Выводы:

Изучил структуру файловой системы, способы ее реализации, методы учета свободного пространства и алгоритмы поиска. Разработал модель файловой системы, а также утилиту – управление доступом.