МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ СИСТЕМ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчет по лабораторной №3

Функции работы с файловой системой

по дисциплине «Системное программирование»

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил | студент гр. ФИб-3301-51-00 |  | Крупин М. С. |
| Проверил | преподаватель каф. ПМиИ |  | Белиц А. Б. |

Киров 2021

**Цель лабораторной работы**

Познакомиться с функциями работы с файловой системой на языках программирования C++ и Python в операционных системах Windows и Linux.

**Задания на лабораторную работу**

**Задание 1.**

В ходе лабораторной работы необходимо разработать консольное приложение на языке C++ c использованием функций WinAPI, которое сможет выполнять следующие операции:

* Отображение списка файлов и папок заданной директории с указанием их размера.
* Создание нового файла
* Копирование нового файла
* Удаление файла/директории
* Переименование/перемещение директории
* Установка атрибута «Только для чтения»
* Снятие атрибута «Только для чтения»
* Проверка атрибута «Скрытый»

**Задание 2.**

В ходе лабораторной работы необходимо разработать программу, на языке python, которая будет позволять операции аналогичные заданию 1.

**Задание 3.**

Протестировать работоспособность программы №2 в среде Linux.

**Полученные результаты**

**Задание 1.**

Тесты будут проводиться на следующей папке:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

1)

Изображение выглядит как текст

Автоматически созданное описание

2)



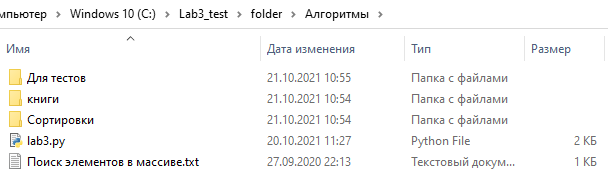
Изображение выглядит как текст

Автоматически созданное описание

3)

Изображение выглядит как текст

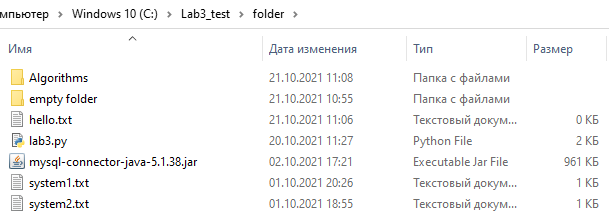
Автоматически созданное описание



5)

Изображение выглядит как текст

Автоматически созданное описание



6)

Изображение выглядит как текст, оранжевый, закрыть

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

7)

Изображение выглядит как текст, устройство, оранжевый, закрыть

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

8)

Изображение выглядит как текст, оранжевый

Автоматически созданное описание

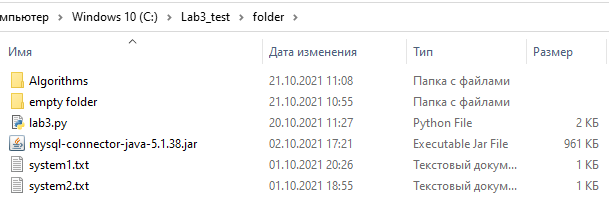
Изображение выглядит как текст

Автоматически созданное описание

4)

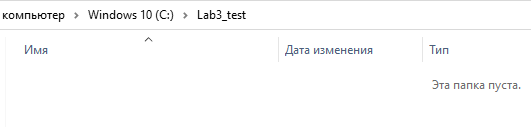
Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание



Листинг программы приведён в приложении А.

**Задание 2.**

Тесты проводятся на той же папке.

Изображение выглядит как текст

Автоматически созданное описание

1)

Изображение выглядит как текст

Автоматически созданное описание

2)

Изображение выглядит как текст

Автоматически созданное описание

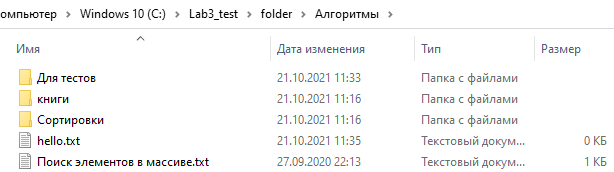
Изображение выглядит как стол

Автоматически созданное описание

3)

Изображение выглядит как текст

Автоматически созданное описание



5)

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

6)

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

7)

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

8)

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

4)

Изображение выглядит как текст

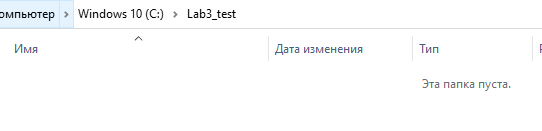
Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



Листинг программы приведён в приложении Б.

**Задание 3.**

Тесты проводятся на той же папке.

Изображение выглядит как текст

Автоматически созданное описание

1)

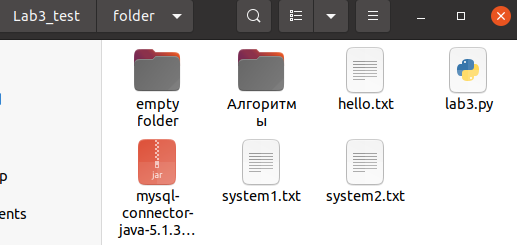
Изображение выглядит как текст

Автоматически созданное описание

2)

Изображение выглядит как текст

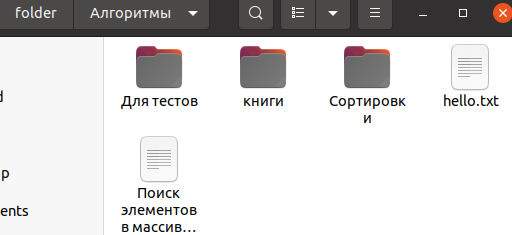
Автоматически созданное описание



3)

Изображение выглядит как текст

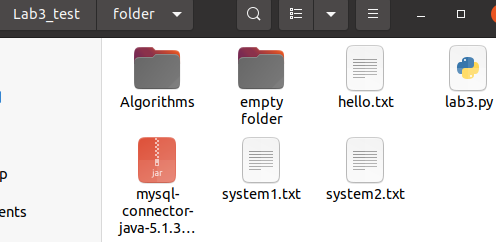
Автоматически созданное описание



5)

Изображение выглядит как текст

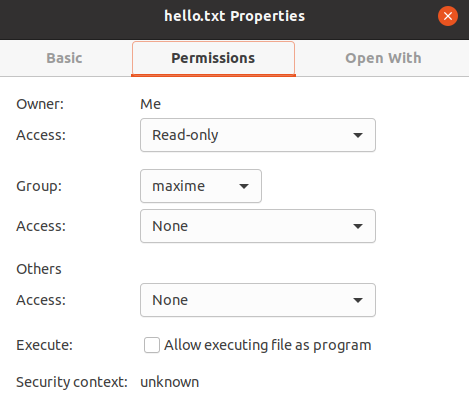
Автоматически созданное описание



6)

Изображение выглядит как текст

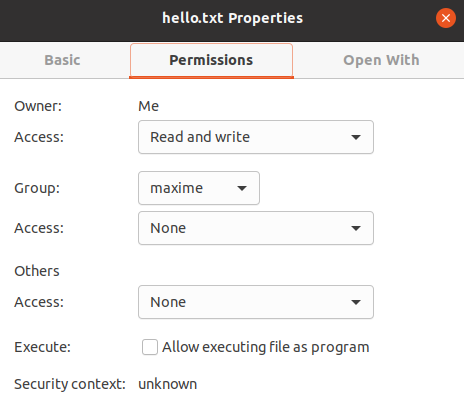
Автоматически созданное описание



7)

Изображение выглядит как текст

Автоматически созданное описание



8)

Изображение выглядит как текст

Автоматически созданное описание

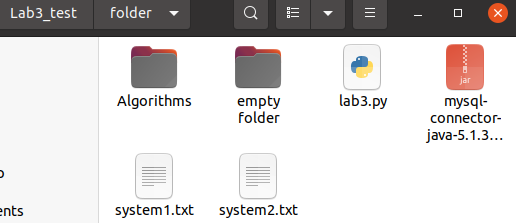
Изображение выглядит как текст

Автоматически созданное описание

4)

Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Вывод**

Мы ознакомились с функциями работы с файловой системой на языках программирования C++ и Python в операционных системах Windows и Linux, написав программы для вывода информации о файлах и управления ими.

**Листинги программ**

**Приложение А. Листинг задания 1**

#include <iostream>

#include <windows.h>

#include <string>

#include <vector>

using namespace std;

bool isDirectory(wstring path);

vector<wstring> getItems(wstring dirName);

void printItems(vector<wstring> items);

wstring toRussian(wstring str);

double getItemSize(wstring path);

double getDirectorySize(wstring path);

bool createFile(wstring path);

bool copyFile(wstring pathFrom, wstring pathTo);

void deleteItem(wstring path);

bool moveDirectory(wstring pathFrom, wstring pathTo);

bool isHidden(wstring path);

wstring getFileName(wstring absPath);

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_CTYPE, "rus");

bool active = true;

while (active)

{

cout << "1 - Отображение списка файлов заданной директории\n";

cout << "2 - Создание файла\n";

cout << "3 - Копирование файла\n";

cout << "4 - Удалить файл/директорию\n";

cout << "5 - Переместить/переименовать директорию\n";

cout << "6 - Установить атрибут \"Только для чтения\"\n";

cout << "7 - Снять атрибут \"Только для чтения\"\n";

cout << "8 - Проверить, является ли файл/директория скрытым\n";

cout << "9 - Выйти из программы\n";

int choice;

cout << "Ваш выбор: ";

cin >> choice;

cin.get();

string name, copyName;

switch (choice)

{

case 1:

cout << "Введите путь к директории: ";

getline(cin, name, '\n');

printItems(getItems(toRussian(wstring(name.begin(), name.end()))));

break;

case 2:

cout << "Введите путь к новому файлу: ";

getline(cin, name, '\n');

if (createFile(toRussian(wstring(name.begin(), name.end()))))

cout << "Файл успешно создан\n";

else

cout << "Не удалось создать файл\n";

break;

case 3:

cout << "Введите путь копируемого файла: ";

getline(cin, name, '\n');

cout << "Введите путь для файла-копии: ";

getline(cin, copyName, '\n');

if (copyFile(toRussian(wstring(name.begin(), name.end())),

toRussian(wstring(copyName.begin(), copyName.end()))))

cout << "Копирование прошло успешно\n";

else

cout << "Копирование не удалось\n";

break;

case 4:

cout << "Введите путь к файлу/директории: ";

getline(cin, name, '\n');

deleteItem(toRussian(wstring(name.begin(), name.end())));

cout << "Удаление прошло успешно\n";

break;

case 5:

cout << "Введите путь перемещаемой директории: ";

getline(cin, name, '\n');

cout << "Введите новый путь для директории: ";

getline(cin, copyName, '\n');

if (moveDirectory(toRussian(wstring(name.begin(), name.end())),

toRussian(wstring(copyName.begin(), copyName.end()))))

cout << "Перемещение директории прошло успешно\n";

else

cout << "Не удалось переместить директорию\n";

break;

case 6:

cout << "Введите путь к файлу: ";

getline(cin, name, '\n');

if (SetFileAttributesW(toRussian(wstring(name.begin(), name.end())).c\_str(),

FILE\_ATTRIBUTE\_READONLY))

cout << "Атрибут успешно установлен\n";

else

cout << "Не удалось установить атрибут\n";

break;

case 7:

cout << "Введите путь к файлу: ";

getline(cin, name, '\n');

if (SetFileAttributesW(toRussian(wstring(name.begin(), name.end())).c\_str(),

FILE\_ATTRIBUTE\_NORMAL))

cout << "Атрибут успешно снят\n";

else

cout << "Не удалось снять атрибут\n";

break;

case 8:

cout << "Введите путь к файлу/директории: ";

getline(cin, name, '\n');

cout << "Файл/директория ";

if (isHidden(toRussian(wstring(name.begin(), name.end()))))

cout << "скрытый\n";

else

cout << "не скрытый\n";

break;

case 9:

active = false;

break;

default:

cout << "Неверное введено число\n";

break;

}

cout << '\n';

}

return 0;

}

vector<wstring> getItems(wstring dirName)

{

if (!isDirectory(dirName))

{

cout << "Нужен путь к папке, а не файлу\n";

return {};

}

wstring pathEnd = L"\*";

if (dirName[dirName.length() - 1] != L'\\' && dirName[dirName.length() - 1] != L'/')

pathEnd = L"\\\*";

dirName += pathEnd;

vector<wstring> items;

WIN32\_FIND\_DATAW file;

HANDLE hFind = FindFirstFileW(dirName.c\_str(), &file);

FindNextFileW(hFind, &file);

dirName.erase(dirName.length() - 1, 1);

if (hFind != INVALID\_HANDLE\_VALUE)

{

while (FindNextFileW(hFind, &file) != NULL)

items.push\_back(dirName + &file.cFileName[0]);

FindClose(hFind);

}

else

cout << "Ошибка чтения директории\n";

return items;

}

void printItems(vector<wstring> items)

{

for (vector<wstring>::iterator it = items.begin(); it < items.end(); ++it)

{

wcout << getFileName(\*it);

cout << " (" << getItemSize(\*it) << " КБ)\n";

}

}

double getItemSize(wstring path)

{

if (isDirectory(path))

return getDirectorySize(path);

HANDLE file = CreateFileW(path.c\_str(),

GENERIC\_READ, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, nullptr,

OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, nullptr);

LARGE\_INTEGER fileSize = { 0 };

GetFileSizeEx(file, &fileSize);

CloseHandle(file);

return fileSize.QuadPart / (double)1024;

}

bool isDirectory(wstring path)

{

return GetFileAttributesW(path.c\_str()) != INVALID\_FILE\_ATTRIBUTES &&

GetFileAttributesW(path.c\_str()) & FILE\_ATTRIBUTE\_DIRECTORY;

}

double getDirectorySize(wstring dirName)

{

double size = 0;

WIN32\_FIND\_DATAW file;

dirName += L"\\\*";

HANDLE hFind = FindFirstFileW(dirName.c\_str(), &file);

FindNextFileW(hFind, &file);

dirName.erase(dirName.length() - 1, 1);

if (hFind != INVALID\_HANDLE\_VALUE)

{

while (FindNextFileW(hFind, &file) != NULL)

size += getItemSize(dirName + &file.cFileName[0]);

FindClose(hFind);

}

return size;

}

wstring toRussian(wstring str)

{

for (int i = 0; i < str.length(); ++i)

{

if (str[i] > 255)

str[i] -= 64432;

}

return str;

}

bool createFile(wstring path)

{

HANDLE file = CreateFileW(path.c\_str(), GENERIC\_READ |

GENERIC\_WRITE, FILE\_SHARE\_WRITE | FILE\_SHARE\_READ, NULL, CREATE\_NEW,

FILE\_ATTRIBUTE\_NORMAL | FILE\_FLAG\_OVERLAPPED, 0);

CloseHandle(file);

return file != INVALID\_HANDLE\_VALUE;

}

bool copyFile(wstring pathFrom, wstring pathTo)

{

if (isDirectory(pathFrom))

return false;

HANDLE file = CreateFileW(pathTo.c\_str(),

GENERIC\_READ, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, nullptr,

OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, nullptr);

if (file != INVALID\_HANDLE\_VALUE)

{

char rewrite;

cout << "Файл-копия уже существует. Перезаписать его? (y/n): ";

cin >> rewrite;

if (rewrite == 'n')

return false;

}

CloseHandle(file);

return CopyFileW(pathFrom.c\_str(), pathTo.c\_str(), false);

}

void deleteItem(wstring path)

{

if (!isDirectory(path))

{

DeleteFileW(path.c\_str());

return;

}

vector<wstring> items = getItems(path);

for (vector<wstring>::iterator it = items.begin(); it < items.end(); ++it)

{

if (isDirectory(\*it))

deleteItem(\*it);

else

{

if (!DeleteFileW((\*it).c\_str()))

{

cout << "Не удаётся удалить файл ";

wcout << \*it;

cout << '\n';

}

}

}

if (!RemoveDirectoryW((path).c\_str()))

{

cout << "Не удаётся удалить директорию ";

wcout << path;

cout << '\n';

}

}

bool moveDirectory(wstring pathFrom, wstring pathTo)

{

if (!isDirectory(pathFrom))

return false;

return MoveFileExW(pathFrom.c\_str(), pathTo.c\_str(),

MOVEFILE\_COPY\_ALLOWED | MOVEFILE\_WRITE\_THROUGH);

}

bool isHidden(wstring path)

{

return GetFileAttributesW(path.c\_str()) & FILE\_ATTRIBUTE\_HIDDEN;

}

wstring getFileName(wstring absPath)

{

int lastSeparatorIndex = max((int)absPath.rfind('\\'), (int)absPath.rfind('/'));

if (lastSeparatorIndex >= 0)

{

absPath.erase(0, lastSeparatorIndex + 1);

return absPath;

}

else

{

cout << "Неверный ввод пути файла\n";

return L"";

}

}

**Приложение Б. Листинг задания 2**

import os  
import shutil  
import stat  
import sys  
  
  
def is\_hidden(path):  
 if not os.path.exists(path):  
 raise Exception("File not found")  
  
 if sys.platform == "win32":  
 return os.stat(path).st\_file\_attributes & stat.FILE\_ATTRIBUTE\_HIDDEN != 0  
 elif sys.platform == "linux2" or sys.platform == "linux":  
 return os.path.split(path)[1][0] == '.'  
 else:  
 raise Exception("Incorrect platform")  
  
  
def get\_size(path):  
 if os.path.isfile(path):  
 return os.path.getsize(path)  
  
 size = 0  
 for item in os.listdir(path):  
 item\_name = path + '/' + item  
 if os.path.isfile(item\_name):  
 size += os.path.getsize(item\_name)  
 else:  
 size += get\_size(item\_name)  
  
 return size  
  
  
def print\_files(path):  
 for file in os.listdir(path):  
 print(file, " (", get\_size(path + "/" + file) / 1024.0, " KB)")  
  
  
def delete\_item(path):  
 if os.path.isfile(path):  
 os.remove(path)  
 else:  
 shutil.rmtree(path)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 active = True  
 while (active):  
 print("1 - Print directory items list")  
 print("2 - Create new file")  
 print("3 - Copy file")  
 print("4 - Delete file/directory")  
 print("5 - Move/rename directory")  
 print("6 - Set readonly attribute")  
 print("7 - Remove readonly attribute")  
 print("8 - Check if file is hidden")  
 print("9 - Quit program")  
 choice = input("Your choice: ")  
 try:  
 if choice == '1':  
 print\_files(input("Input directory path: "))  
 elif choice == '2':  
 open(input("Input file path: "), 'a').close()  
 elif choice == '3':  
 shutil.copyfile(input("Input source path: "), input("Input destination path: "))  
 elif choice == '4':  
 delete\_item(input("Input path: "))  
 elif choice == '5':  
 shutil.move(input("Input source path: "), input("Input destination path: "))  
 elif choice == '6':  
 os.chmod(input("Input path: "), stat.S\_IREAD)  
 elif choice == '7':  
 os.chmod(input("Input path: "), stat.S\_IWRITE | stat.S\_IREAD)  
 elif choice == '8':  
 if is\_hidden(input("Input path: ")):  
 print("File is hidden")  
 else:  
 print("File in not hidden")  
 elif choice == '9':  
 active = False  
 else:  
 raise Exception("Incorrect input")  
  
 print("Successfully")  
 except:  
 print("Exception occurred while executing")  
  
 print('\n')