**ИТМО Кафедра Информатики и прикладной математики**

Отчет по лабораторной работе №1

«Копирование файлов»

**Выполнил: студент группы P3217**

**Плюхин Дмитрий**

**Преподаватель: Зыков А. Г.**

**2017 год**

1. **Задание**

Создайте файл, содержащий массив структур, каждая из которых содержит информацию о Ваших результатах за три семестра (фамилия, семестр, дисциплина, оценка).

Реализуйте простое последовательное копирование содержимого файла тремя различными способами:

1. С использованием библиотеки С.

2. С использованием Windows.

3. С использованием вспомогательной функции Windows — CopyFile.

Результат вывести на печать рез-ты копирования и средний балл.

Для анализа способов копирования создайте произвольный файл большого размера (экспериментально увеличивая для получения наглядных результатов) и определите временные характеристики каждого из способов. Результаты выведите в виде таблицы.

Проведите сравнительный анализ способов. Достоинства и недостатки.

1. **Листинг основной части программы**

Файл main.cpp реализует первую часть задания, а именно, создает массив структур данных, записывает его в файл, производит копирование тремя разными способами, а также демонстрирует использование Windows API для вывода на экран текущей директории. Код, не относящийся непосредственно к копированию файлов, опущен.

#include <iostream>

#include <cstdlib>

#include <stdio.h>

#include <string.h>

#include <fstream>

#include <windows.h>

#define BUF\_SIZE 256

using namespace std;

struct mark{

char surname[10], discipline[40];

int semester, mark;

};

int copyByCLib(const char\* src, const char\* dest){

FILE \*in\_file, \*out\_file;

char rec[BUF\_SIZE];

size\_t bytes\_in, bytes\_out;

in\_file = fopen(src,"rb");

if (in\_file == NULL){

return 1;

}

out\_file = fopen(dest,"wb");

if (out\_file == NULL){

return 1;

}

while((bytes\_in = fread(rec, 1, BUF\_SIZE, in\_file))>0){

bytes\_out = fwrite(rec, 1, bytes\_in, out\_file);

if (bytes\_out != bytes\_in){

return 2;

}

}

fclose(in\_file);

fclose(out\_file);

return 0;

}

int copyByWin(const char\* src, const char\* dest){

HANDLE hIn, hOut;

DWORD nIn, nOut, nBytes;

CHAR Buffer[BUF\_SIZE];

CHAR DirName[BUF\_SIZE];

nBytes = GetCurrentDirectory(BUF\_SIZE, DirName);

cout << "Current directory : " << DirName << endl;

hIn = CreateFile(src, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, 0, NULL);

if (hIn == INVALID\_HANDLE\_VALUE){

return 1;

}

hOut = CreateFile(dest, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hOut == INVALID\_HANDLE\_VALUE){

return 1;

}

while(ReadFile(hIn,Buffer,BUF\_SIZE,&nIn,NULL)&&nIn>0){

WriteFile(hOut, Buffer, nIn, &nOut, NULL);

if (nIn != nOut){

return 2;

}

}

CloseHandle(hIn);

CloseHandle(hOut);

return 0;

}

int copyByCopyFile(const char\* src, const char\* dest){

if (!CopyFile(src, dest, FALSE)){

return 1;

}

return 0;

}

float getAverage(const char\* fileName){

mark\* marks\_loaded = loadMarks(fileName);

int sum = 0;

for (int i = 0; i < 14; i++){

sum += marks\_loaded[i].mark;

}

return (float)sum/14;

}

int main(){

mark\* marks = getMarks();

saveMarks("data/data.dat", marks);

if (copyByWin("data/data.dat","data/data\_win\_copied.dat")==0){

cout << "Succesfully copied by Win API" << endl;

} else {

cout << "Error when copying by Win API" << endl;

}

if (copyByCLib("data/data.dat","data/data\_c\_copied.dat")==0){

cout << "Succesfully copied by C lib" << endl;

} else {

cout << "Error when copying by C lib" << endl;

}

if (copyByCopyFile("data/data.dat", "data/data\_copyfile\_copied.dat")==0){

cout << "Succesfully copied by Copy File" << endl;

} else {

cout << "Error when copying by CopyFile" << endl;

}

cout << "Average mark is "<<getAverage("data/data\_c\_copied.dat") << endl;

return 0;

}

Файл tester.cpp реализует вторую часть задания, а именно, производит замеры времени копирования больших файлов (в несколько тысяч килобайт) разными способами, и выводит их на экран в виде таблицы.

int createFile(int size){

ofstream fout("data/file.dat");

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

fout.write("DATA",1024);

}

fout.close();

}

float getTimeExcecuting(int (\*func)(const char\*, const char\*),const char\* src, const char\* dest){

float fTimeStart = clock()/(float)CLOCKS\_PER\_SEC;

func(src, dest);

float fTimeStop = clock()/(float)CLOCKS\_PER\_SEC;

return fTimeStop - fTimeStart;

}

int initializeCell(){

cout << "| ";

cout.setf(ios::left);

cout.width(10);

return 0;

}

int showCell(const char\* content){

initializeCell();

cout << content;

}

int showCell(const char\* content, const char\* finalizer){

initializeCell();

cout << content << finalizer << endl;

}

int showCell(int content){

initializeCell();

cout << content;

}

int showCell(float content){

initializeCell();

cout << content;

}

int showCell(float content, const char\* finalizer){

initializeCell();

cout << content << finalizer << endl;

}

int drawHorizontalLine(int length){

cout.width(length);

cout.fill('-');

cout << '-' << endl;

cout.fill(' ');

}

int showTableHead(){

drawHorizontalLine(50);

showCell("Size(Kb)");

showCell("C lib");

showCell("Win API");

showCell("CopyFile", " |");

drawHorizontalLine(50);

return 0;

}

int main(){

int size = 0;

showTableHead();

for (int i = 1; i < 10; i++){

size = i\*10000;

createFile(size);

showCell(size);

showCell(getTimeExcecuting(copyByCLib,"data/file.dat","data/file1.dat"));

showCell(getTimeExcecuting(copyByWin,"data/file.dat","data/file2.dat"));

showCell(getTimeExcecuting(copyByCopyFile,"data/file.dat","data/file3.dat"), " |");

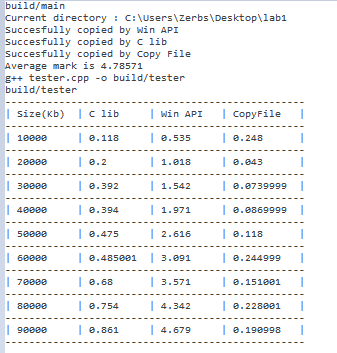
drawHorizontalLine(50);

}

}

**3. Результаты работы программы**

В результате работы программы получили следующий вывод:



1. **Вывод**

Таким образом, существует три способа копирования файлов в системе Windows: первый, использующий библиотеку языка С, может быть применен в том случае, если требуется возможность как можно более быстрой адаптации программы под альтернативные операционные системы, однако не предъявляется особых требований к производительности; второй способ, использующий функцию CopyFile, является наиболее эффективным в плане быстродействия. Наименее «приоритетным» в рассмотренных отношениях представляется способ, использующий Windows API, поскольку он как сильно проигрывает по времени работы, так и не предоставляет достаточной переносимости программ, хотя и обеспечивает высокую гибкость, если программа запускается только в рамках Windows.