Университет ИТМО

кафедра ИПМ

Системное программное обеспечение

Лабораторная работа № 1

Копирование файлов

Работу выполнил:

Студент 2 курса

Группы № 2125

Назарьев Сергей

Санкт-Петербург

2014 г.

**Цель работы:**

Написать на любом языке (но лучше С, С++) программу, которая создаст файл запишет в него массив структур. Структура должна содержать следующие поля:

1) номер семестра

2) ФИО

3) Дата рождения

4) название предмета

5) оценка за предмет

Вывести этот массив на экран (консоль), а также средний балл по этому предмету. Затем необходимо скопировать файл в другую директорию тремя различными способами:

1) с использованием "сишных" библиотек;

2) с использованием windows;

3) c использованием вспомогательной функции Windows copyFile.

**Код программы:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#define STRUCTURES\_COUNT 5000

#define MAX\_BUFFER 1024

#define IN\_FILE "journal.bin"

#define IN\_FILE\_L L"journal.bin"

#define OUT\_FILE "journal2.bin"

#define OUT\_FILE\_L L"journal2.bin"

#define uint32\_t unsigned int

#define uint64\_t unsigned long

**typedef** **struct** {

**uint32\_t** semester;

**char**\* name;

**char**\* birthday;

**char**\* discipline;

**uint32\_t** grade;

} STRUCTURE;

**int** **copy**(**char**\* in\_file\_str, **char**\* out\_file\_str)

{

**FILE**\* a;

**FILE**\* b;

**char** buffer[MAX\_BUFFER];

**size\_t** readed;

a = fopen(in\_file\_str, "rb");

b = fopen(out\_file\_str, "wb");

**if** (a < **0** || b < **0**)

**return** **2**;

**while** ((readed = fread(buffer, **sizeof**(**char**), MAX\_BUFFER, a)) > **0**)

fwrite(buffer, **sizeof**(**char**), readed, b);

fclose(a);

fclose(b);

**return** **1**;

}

**int** **windows\_copy**(LPCTSTR in\_file\_str, LPCTSTR out\_file\_str) {

HANDLE hIn, hOut;

DWORD nIn, nOut;

CHAR Buffer[MAX\_BUFFER];

hIn = CreateFile(in\_file\_str, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

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

printf("Error: invalid hadle value! Code: %x**\n**", GetLastError());

**return** **2**;

}

hOut = CreateFile(out\_file\_str, GENERIC\_WRITE, **0**, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

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

printf("Error: error read file! Code: %x**\n**", GetLastError());

**return** **3**;

}

**while** (ReadFile(hIn, Buffer, MAX\_BUFFER, &nIn, NULL) && nIn > **0**) {

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

**if** (nIn != nOut) {

printf("Error: error write file! Code: %x**\n**", GetLastError());

**return** **4**;

}

}

CloseHandle(hIn);

CloseHandle(hOut);

**return** **1**;

}

**int** **windows\_copy\_file**(LPCTSTR in\_file\_str, LPCTSTR out\_file\_str) {

CopyFile(in\_file\_str, out\_file\_str, FALSE);

**return** **0**;

}

**int** **main**() {

SYSTEMTIME st, ft;

**uint32\_t** i, size;

**char**\* names[] = {"Ivanov Ivan Ivanovich", "Dmitrov Dmitriy Dmitrievich", "Denisov Denis Denisovich", "Sergeev Sergey Sergeevich", "Andreev Andrey Andreevich"};

**char**\* disciplines[] = {"IVT", "OVT", "AVT", "BVT", "ZVT", "LOL"};

STRUCTURE\* structures[STRUCTURES\_COUNT];

**FILE**\* file;

**char** buf[MAX\_BUFFER];

**int** a = **0**, divisor = **0**;

srand(time(NULL));

**for**(i = **0**; i < STRUCTURES\_COUNT; i++) {

structures[i] = (STRUCTURE\*)malloc(**sizeof**(STRUCTURE));

structures[i]->name = names[rand() % (**sizeof**(names) / **sizeof**(names[**0**]))];

structures[i]->birthday = "01.02.03";

structures[i]->discipline = disciplines[rand() % (**sizeof**(disciplines) / **sizeof**(disciplines[**0**]))];

structures[i]->grade = (rand()%**5**)+**1**;

structures[i]->semester = **1**;

}

file = fopen(IN\_FILE, "wb");

**for**(i = **0**; i < STRUCTURES\_COUNT; i++) {

fwrite(&(structures[i]->semester), **sizeof**(structures[i]->semester), **1**, file);

fwrite(&(structures[i]->name), strlen(structures[i]->name)+**1**, **1**, file);

fwrite(&(structures[i]->birthday), strlen(structures[i]->birthday)+**1**, **1**, file);

fwrite(&(structures[i]->discipline), strlen(structures[i]->discipline)+**1**, **1**, file);

fwrite(&(structures[i]->grade), **sizeof**(structures[i]->grade), **1**, file);

printf("Structure %d**\r\n**Semester %d**\r\n**Name: %s**\r\n**Birthday: %s**\r\n**Discipline: %s**\r\n**Grade: %d**\r\n\r\n**",

i,

structures[i]->semester,

structures[i]->name,

structures[i]->birthday,

structures[i]->discipline,

structures[i]->grade

);

}

fclose(file);

printf("Discipline name: ");

fgets(buf, MAX\_BUFFER, stdin);

size = strlen(buf);

buf[size-**1**] = '\0';

**for**(i = **0**; i < STRUCTURES\_COUNT; i++) {

**if**(!stricmp(buf, structures[i]->discipline)) {

a += structures[i]->grade;

divisor++;

}

}

**if**(divisor == **0**)

puts("No one entries");

**else**

printf("Average grade: %.1f**\n**", ((**float**)a / (**float**)divisor));

GetSystemTime(&st);

copy(IN\_FILE, OUT\_FILE);

GetSystemTime(&ft);

printf("Copy with C libraries time: %d ms**\n**", (ft.wMilliseconds - st.wMilliseconds));

GetSystemTime(&st);

windows\_copy(IN\_FILE, OUT\_FILE);

GetSystemTime(&ft);

printf("Copy with Windows time: %d ms**\n**", (ft.wMilliseconds - st.wMilliseconds));

GetSystemTime(&st);

windows\_copy\_file(IN\_FILE, OUT\_FILE);

GetSystemTime(&ft);

printf("Copy with CopyFile time: %d ms**\n**", (ft.wMilliseconds - st.wMilliseconds));

**return** **0**;

}

**Результат работы програмы:**

Copy with C libraries time: 4 ms

Copy with Windows time: 4 ms

Copy with CopyFile time: 1 ms

**Вывод:**

В ходе выполнения лабораторной работы были изучены основные типы копирования файлов: с использованием встроенной библиотеки С, с использованием Windows.и с использованием вспомогательной функции Windows – CopyFile.

В бинарный файл journal.bin записывался массив из структур с помощью стандартной библиотеки ввода/вывода С.

Основным критерием для выяснения эффективности работы метода служило время выполнения метода, т.к. все скопированные файлы были точными копиями оригинала и данные не пропадали. В ряде проделанных экспериментов выяснилось, что метод CopyFile работает всегда быстрее остальных. Примерно одинаковое время выдают варианты с использованием stdio и с помощью обёрток над stdio в Windows.

Что касается простоты реализации, то CopyFile является лидером, т.к. там нужно просто вызвать функцию, передав ей в качестве аргументов имена файлов, и логическую переменную проверки. Зато метод из библиотеки C является более гибким и понятным, легче поймать ошибку (узнать, по какой причине не открылся файл можно просто посмотрев на номер дескриптора, который возвращает fopen) и можно реализовать дополнительный функционал. Так как обёртки об stdio в Windows являются именно обёртками, то по работе и виду они практически идентичны.