BМИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем

Кафедра «Прикладная математика и фундаментальная информатика»

**Домашнее задание**

|  |  |  |
| --- | --- | --- |
| по дисциплине | Операционные системы |  |

|  |  |
| --- | --- |
| Студента | Иванютин Сергей Александрович |
|  | фамилия, имя, отчество полностью |
| Курс | 2 Группа ФИТ-221 |
| Направление | 02.03.02 Фундаментальная информатика |
|  | и информационные технологии |
|  | код, наименование |
| Руководитель | ассистент |
|  | должность, ученая степень, звание |
|  | Карабцов Р.Д. |
|  | фамилия, инициалы |
| Выполнил | 04.06.2024 |
|  | дата, подпись студента |
| баллы |  |
|  | дата, подпись руководителя |

Омск-2024

**Задание**

Разработать программу для Windows, которая должна запускаться в двух экземплярах - каждый в своем окне командной оболочки FAR или из ПРОВОДНИКа операционной системы. Программа использует заранее подготовленный текстовый файл. Она пытается открыть этот файл для чтения с указываемым при этом запрете для других использовать этот файл. По результатам выполнения системной функции открытия на экран выдается сообщение - удалось ли открыть файл и, если не удалось по причине отсутствия доступа к одновременно выполняемым программам, то сообщение именно об этой причине. При отсутствии указанного в программе файла после сообщения об этом отсутствии программа прекращает работу. При его наличии, но невозможности продолжения действий из-за блокировки, установленной другим экземпляром запущенной программы, выполняется ожидание освобождения файла от блокировки. При отсутствии указанной причины доступа программа должна ждать освобождения файла. В обоих случаях - ожидания освобождения или исходной доступности - программа читает из этого файла все находящиеся в нем данные и выводит их на экран. Сообщения должны выводиться цветные и в середине консольного окна

**Контрольные вопросы:**

1. Чем стандартный ввод-вывод отличается от нестандартного, привести примеры использования того и другого.

2. Зачем в общем случае закрывать файл, к каким возможным последствиям и когда приводит не закрытие файла.

3. Какой код ошибки возвращает функция открытия файла, если в ней задано ограничение по одновременной работе с файлом и этот файл в момент выполнения попытки открытия уже захвачен для работы другим вычислительным процессом.

**Решение**

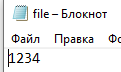
****

Рисунок 1. Содержание текстового файла

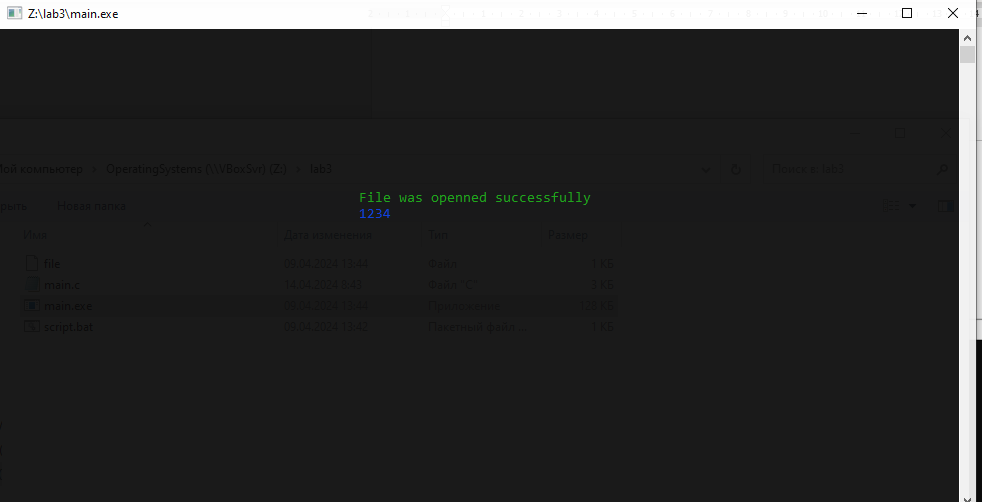
****

Рисунок 2. Вывод программы

1. Стандартный ввод-вывод отличается от нестандартного тем, что стандартный ввод-вывод использует предопределенные потоки данных (например, stdin для ввода с клавиатуры, stdout для вывода на экран), в то время как нестандартный ввод-вывод позволяет работать с файлами, сетевыми соединениями и другими источниками данных. Примеры использования стандартного ввода-вывода: чтение данных с клавиатуры, вывод на экран. Примеры нестандартного ввода-вывода: чтение данных из файла, отправка данных по сети.

2. В общем случае закрытие файла важно для освобождения ресурсов, связанных с файлом, и предотвращения утечки памяти. Не закрытие файла может привести к блокировке доступа к файлу другим процессам, возможным повреждениям данных при записи, а также к неожиданным ошибкам или сбоям программы из-за некорректной работы с файлом.

3. Если файл уже захвачен для работы другим вычислительным процессом и попытаться открыть его с ограничением на одновременную работу, функция открытия файла вернет код ошибки, который обычно является кодом ошибки доступа или блокировки. Например, в операционной системе Windows это может быть код ошибки ERROR\_SHARING\_VIOLATION (32), который указывает на то, что файл уже используется другим процессом и недоступен для чтения или записи.

**Листинг программы**

#include <stdio.h>

#include "/usr/x86\_64-w64-mingw32/include/windows.h"

#include <windows.h>

void display(HANDLE h, char \*text, int len, WORD attr);

int main()

{

HANDLE hout, hin, h;

COORD pos;

DWORD actlen, readed, err\_code, len;

char buffer[300], err\_msg[300], err\_msg\_to\_print[300];

BOOL rc;

actlen = strlen(buffer);

hin = GetStdHandle(STD\_INPUT\_HANDLE);

if (hin == INVALID\_HANDLE\_VALUE)

return 1;

hout = GetStdHandle(STD\_OUTPUT\_HANDLE);

if (hout == INVALID\_HANDLE\_VALUE)

return 1;

h = CreateFile("file", GENERIC\_READ, 0, 0, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

if (h == INVALID\_HANDLE\_VALUE)

{

display(hout, "Cannot open file", 16, FOREGROUND\_RED);

err\_code = GetLastError();

len = FormatMessage(

FORMAT\_MESSAGE\_FROM\_SYSTEM,

NULL,

err\_code,

MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT),

err\_msg,

sizeof(err\_msg),

NULL

);

CharToOem(err\_msg, err\_msg\_to\_print);

switch (err\_code)

{

case 2:

display(hout, "error: file doesn\'t exist", 24, FOREGROUND\_RED);

getchar();

return 1;

case 32:

display(hout, "error: file already open", 24, FOREGROUND\_RED);

while (h == INVALID\_HANDLE\_VALUE)

{

h = CreateFile("file", GENERIC\_READ, 0, 0, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

}

break;

default:

display(hout, err\_msg\_to\_print, len, FOREGROUND\_RED);

break;

}

}

display(hout, "File was openned successfully", 29, FOREGROUND\_GREEN);

ReadFile(h, buffer, 300, &actlen, NULL);

display(hout, buffer, actlen-1, FOREGROUND\_BLUE);

CloseHandle(h);

getchar();

}

static COORD pos;

void display(HANDLE h, char \*text, int len, WORD attr)

{

CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

SMALL\_RECT rect;

int rows, cols, roffset, coffset;

roffset = 5;

coffset = 15;

GetConsoleScreenBufferInfo(h, &csbi);

cols = csbi.srWindow.Right - csbi.srWindow.Left + 1;

rows = csbi.srWindow.Bottom - csbi.srWindow.Top + 1;

int row = rows/2 - roffset;

int col = cols/2 - coffset;

int row\_length = coffset\*2;

if (pos.X == 0)

pos.X = col;

if (pos.Y == 0)

pos.Y = row;

while (len > 0)

{

int cur\_len;

DWORD actlen;

if (len < row\_length)

{

cur\_len = len;

}

else

{

cur\_len = row\_length;

}

len -= row\_length;

WriteConsoleOutputCharacter(h, text, cur\_len, pos, &actlen);

FillConsoleOutputAttribute(h, attr, cur\_len, pos, &actlen);

text += cur\_len;

pos.Y += 1;

}

}