Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. Тихонова

Департамент электронной инженерии

ОТЧЕТ

О ПРАКТИЧЕСКОЙ РАБОТЕ №6

по дисциплине «Системное программирование»

«Файловый ввод-вывод»

Вариант 24

Студент гр. БИБ201

Шадрунов Алексей

Дата выполнения: 25 февраля 2023 г.

Преподаватель:

Морозов В. И.

«___» _____ 2023 г.

Москва, 2023

Содержание

1 Цель работы

Программа должна выводить в консоль первые N байт содержимого файла, название которого передано в качестве аргумента командной строки. Число N передаётся в качестве второго аргумента командной строки. Если число N больше количества имеющихся данных, необходимо вывести те данные, которые доступны. Если файл пуст или не существует, необходимо вывести соответствующее сообщение об ошибке и завершить работу программы.

2 Ход работы

2.1 Описание алгоритма

Работа программы состоит из нескольких этапов:

1. Проверить количество входных аргументов. Если их не два, вывести подсказку.
2. Сохранить аргументы в переменные path (путь к файлу) и N (число байт).
3. Вывести полученные параметры в консоль (функция print).
4. Открыть файл на чтение. Если открыть не удалось (не существует файла), вывести сообщение об ошибке.
5. Выделить N байт в динамической памяти. В случае ошибки вывести сообщение.
6. Прочитать N байт из файла. В случае ошибки вывести сообщение. Если прочитано 0 байт, вывести сообщение о том, что файл пуст.
7. Вывести буфер в консоль. В случае ошибки вывести сообщение.
8. Освободить память и ресурсы.

Так как программа использует API операционных систем, исходные коды программы для компиляторов gcc и MSVC различаются и приведены в приложении.

2.2 Компилятор gcc

При использовании компилятора gcc на Linux мы пользуемся системными вызовами из файла unistd.h: write, open, read, close. Они используются для работы с файлами и консолью.

Для сборки используется команда: `gcc main.c`

Далее продемонстрируем работу программы (рисунки 1-3).

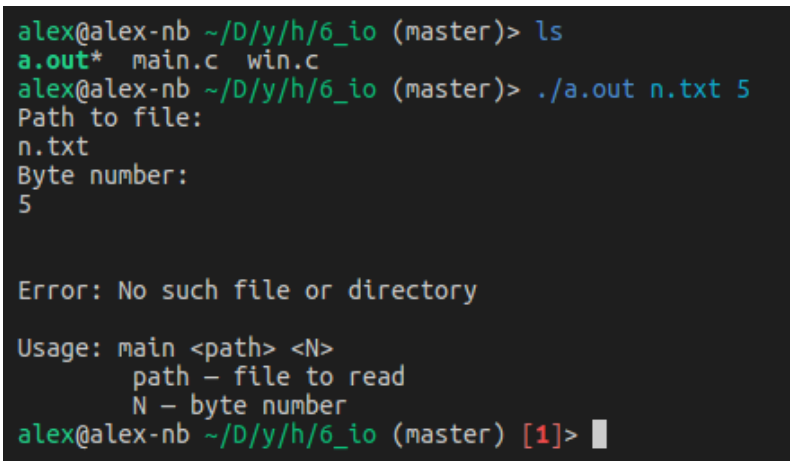

Рисунок 1 – Нет файла

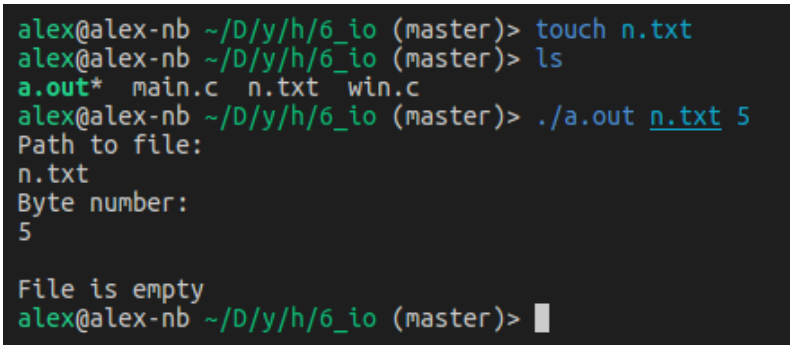

Рисунок 2 – Файл пустой

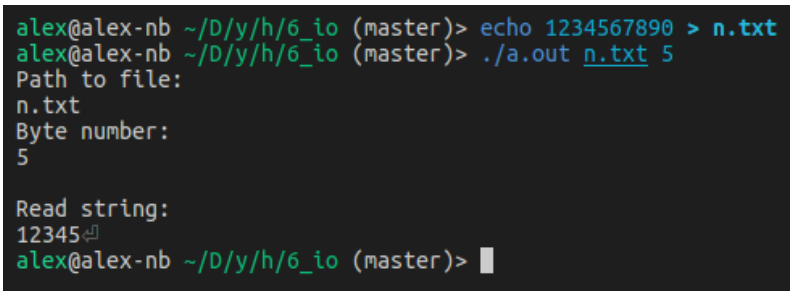

Рисунок 3 – Работа программы

2.3 Компилятор MSVC

Чтобы запустить эту программу на Windows, нужно заменить системные вызовы на WinAPI. Для этого подключаем файл Windows.h и используем функции WriteConsole, CreateFile, HeapCreate, HeapAlloc, HeapDestroy, ReadFile, CloseHandle.

Для компиляции и сборки программы используем Developer Command Prompt и команду: `cl win.c`

Далее продемонстрируем работу программы (рисунки 1-3).

```
C:\Users\alex\Documents\io>dir
 Volume in drive C has no label.
 Volume Serial Number is 3695-66F5

 Directory of C:\Users\alex\Documents\io

02/26/2023  11:13 AM    <DIR>          .
02/26/2023  11:13 AM    <DIR>          ..
02/26/2023  10:39 AM             3,599 win.c
02/26/2023  10:39 AM           112,640 win.exe
02/26/2023  10:39 AM             3,839 win.obj
               3 File(s)        120,078 bytes
               2 Dir(s)  33,572,274,176 bytes free

C:\Users\alex\Documents\io>win.exe n.txt 10
Path to file:
n.txt
Byte number:
10

The system cannot find the file specified.


Usage: main <path> <N>
        path - file to read
        N - byte number
```

Рисунок 4 – Нет файла

```
C:\Users\alex\Documents\io>type nul > n.txt

C:\Users\alex\Documents\io>win.exe n.txt 10
Path to file:
n.txt
Byte number:
10

File is empty
```

Рисунок 5 – Файл пустой

```
C:\Users\alex\Documents\io>echo 123456789abcdef > n.txt

C:\Users\alex\Documents\io>win.exe n.txt 10
Path to file:
n.txt
Byte number:
10

Read string:
123456789a
```

Рисунок 6 – Работа программы

3 Выводы о проделанной работе

В рамках данной работы я написал программу, в которой выводятся в консоль первые N байт содержимого файла, название которого передано в качестве аргумента командной строки. Число N передаётся в качестве второго аргумента командной строки. Если число N больше количества имеющихся данных, необходимо вывести те данные, которые доступны. Если файл пуст или не существует, необходимо вывести соответствующее сообщение об ошибке и завершить работу программы. Собрал программу с помощью компилятора gcc и MSVC.

Приложение А. Код main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

/*Программа
 должна выводить в консоль первые N байт содержимого файла, название
    которого передано в качествеаргумента
 командной строки. Число N передаётся в качестве второго аргумента
    командной строки. Если число Nбольше
 количества имеющихся данных, необходимо вывести те данные, которые
    доступны. Если файл пуст или несуществует
, необходимо вывести соответствующее сообщение об ошибке и завершить
    работу программы.
*/

void print(const void *buf, size_t n)
{
    write(STDOUT_FILENO, buf, n);
    write(STDOUT_FILENO, "\n", 1);
}

/**
 * Prints help message to console
 */
int print_help()
{
    print("\nUsage: main <path> <N>", 23);
    print("\tpath — file to read", 22);
    print("\tN — byte number", 18);
    return 1;
}

int catch ()
{
    perror("\nError");
    print_help();
}

int main(int argc, char **argv)
{
    // check number of arguments
    if (argc != 3)
        return print_help();

    // get byte numbers and file path
    char *path = argv[1];
    int N = atoi(argv[2]);

    // print input values
    print("Path to file:", 13);
    print(path, strlen(path));
    print("Byte number:", 12);
    print(argv[2], strlen(argv[2]));
    print("", 0);

    // open file to read
    int file = open(path, O_RDONLY);
    if (file < 0)
        return catch ();

    // allocate N bytes
    char *readbuf = malloc(N);
    if (!readbuf)
        return catch ();

```

```
64      // read N bytes from file
65      int bytesRead = read(file, readbuf, N);
66      if (bytesRead < 0)
67          return catch ();
68      if (bytesRead == 0)
69      {
70          print("File is empty", 13);
71          return 0;
72      }
73
74      // print buffer to stdout
75      print("Read string:", 12);
76      int bytesWrite = write(STDOUT_FILENO, readbuf, bytesRead);
77      if (bytesWrite < 0)
78          return catch ();
79
80      // release memory and resources
81      int file_closed = close(file);
82      if (file_closed < 0)
83          return catch ();
84      free(readbuf);
85
86      return 0;
87  }
```

Приложение Б. Код win.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <Windows.h>

/*Программа
 должна выводить в консоль первые N байт содержимого файла, название которого передано в качествеаргумента
 командной строки. Число N передаётся в качестве второго аргумента командной строки. Если число Nбольше
 количества имеющихся данных, необходимо вывести те данные, которые доступны. Если файл пуст или несуществует
, необходимо вывести соответствующее сообщение об ошибке и завершить работу программы.
*/


/**
 * Prints string buffer and CR to console
 */
void print(const void *buf, DWORD n)
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    WriteConsole(hConsole, buf, n, NULL, NULL);
    WriteConsole(hConsole, "\n", 1, NULL, NULL);
}

/**
 * Prints help message to console
 */
int print_help()
{
    print("\nUsage: main <path> <N>", 23);
    print("\tpath - file to read", 22);
    print("\tN - byte number", 18);
    return 1;
}

/**
 * Prints error message and help message to console
 * and closes the program
 */
int catch ()
{
    LPSTR message;
    DWORD dwMessageLen = FormatMessage(
        FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_ALLOCATE_BUFFER,
        NULL, GetLastError(), MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPSTR)&message, 0, NULL);
    print(message, dwMessageLen);
    print_help();
    return 1;
}

int main(int argc, char **argv)
{
    // check number of arguments
    if (argc != 3)
        return print_help();

    // get byte numbers and file path
    char *path = argv[1];
    int N = atoi(argv[2]);

    // print input values
    print("Path to file:", 13);
    print(path, strlen(path));
```

```
64      print("Byte number:", 12);
65      print(argv[2], strlen(argv[2]));
66      print("", 0);
67
68      // open file to read
69      HANDLE hFile = CreateFile(path, GENERIC_READ, 0, NULL, OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL, NULL);
70      if (hFile == INVALID_HANDLE_VALUE)
71          return catch ();
72
73      // allocate N bytes
74      HANDLE hHeap = HeapCreate(0, 0x01000, 0);   // create heap
75      if (hHeap == NULL)
76          return catch();
77
78      // LPSTR readbuf = (char*)HeapAlloc(hHeap, 0, N);
79      LPSTR readbuf = HeapAlloc(hHeap, 0, N);
80      if (readbuf==NULL)  // if error allocating
81          if (HeapDestroy(hHeap) == 0)  // if error destroying heap
82              return catch();
83
84      DWORD bytesRead = 0;
85
86      // read N bytes from file
87      BOOL result = ReadFile(hFile, readbuf, N, &bytesRead, NULL);
88      if (!result)
89          return catch ();
90      if (bytesRead == 0)
91      {
92          print("File is empty", 13);
93          return 0;
94      }
95
96      // print buffer to stdout
97      LPDWORD dwBytesWritten;
98      print("Read string:", 12);
99      HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
100     BOOL bytesWrite = WriteConsoleA(hConsole, readbuf, bytesRead,
        dwBytesWritten, NULL);
101     if (!bytesWrite)
102         return catch ();
103
104     // release memory and resources
105     BOOL file_closed = CloseHandle(hFile);
106
107     if (!file_closed)
108         return catch ();
109     if (HeapFree(hHeap, 0, readbuf) == 0)
110         return catch();
111     if (HeapDestroy(hHeap) == 0)  // if error destroying heap
112         return catch();
113
114     return 0;
115 }
```