Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №7

по дисциплине «Основы операционных систем»

на тему «Работа с файлами в асинхронном режиме»

Выполнили студенты группы 22ВВП1

Демин М. С.

Беляев Д. И.

Приняли:

Егоров В. Ю.

Федюнин Р. Н.

Пенза 2024

**Название**

Работа с файлами в асинхронном режиме

**Цель работы**

Изучение способов работы с файлами в Win32 API. Изучение асинхронного режима работы с файлами.

**Лабораторное задание**

Разработать программу, копирующую файл из одного каталога в другой. Операции копирования файла должны выполняться одной нитью в асинхронном режиме.

**Описание данных**

1. BUFFER\_SIZE: Константа, определяющая размер буфера для чтения и записи файлов. В данном случае размер равен 4096 байт.

2. CopyFileParams: Структура, содержащая два поля:

• srcPath: Указатель на строку типа WCHAR, представляющую путь к исходному файлу, который нужно скопировать.

• dstPath: Указатель на строку типа WCHAR, представляющую путь к целевому файлу, куда будет осуществляться копирование.

3. OVERLAPPED: Структуры OVERLAPPED (olRead и olWrite) используются для асинхронного ввода-вывода. Каждая из них содержит информацию о состоянии операции ввода-вывода, включая указатель на событие (hEvent), который используется для синхронизации.

4. BYTE buffer[BUFFER\_SIZE]: Массив байтов, используемый для временного хранения данных, считываемых из исходного файла перед записью их в целевой файл.

5. DWORD bytesRead, bytesWritten: Переменные, хранящие количество байт, считанных из исходного файла и записанных в целевой файл соответственно.

6. bool isReading: Логическая переменная, указывающая, продолжается ли процесс чтения данных из исходного файла.

**Описание структуры программы**

1. Функция CopyFileThread:

• Эта функция запускается в отдельном потоке и отвечает за копирование файла из srcPath в dstPath.

• Сначала она открывает исходный файл для чтения и целевой файл для записи. Если не удается открыть один из файлов, выводится сообщение об ошибке, и функция завершает свою работу.

• Создаются события для асинхронных операций чтения и записи.

• В цикле происходит чтение данных из исходного файла и запись их в целевой файл. Если операции чтения или записи выполняются асинхронно, поток ожидает завершения этих операций с помощью WaitForSingleObject.

• После успешного чтения данных из исходного файла и их записи в целевой файл обновляются смещения (Offset) для последующих операций.

• После завершения всех операций закрываются дескрипторы файлов и событий, а также освобождается выделенная память для параметров.

2. Функция main:

• В функции main задаются пути к исходному и целевому файлам.

• Выделяется память для структуры CopyFileParams, в которую записываются пути к файлам.

• Создается новый поток с помощью CreateThread, который запускает функцию CopyFileThread.

• Если создание потока прошло успешно, основной поток ожидает его завершения с помощью WaitForSingleObject. В противном случае выводится сообщение об ошибке.

• После завершения работы потока дескриптор потока закрывается.

**Текст программы**

**main.c [code]**

#include <windows.h>

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define BUFFER\_SIZE 4096

typedef struct {

const WCHAR\* srcPath;

const WCHAR\* dstPath;

} CopyFileParams;

DWORD WINAPI CopyFileThread(LPVOID params) {

CopyFileParams\* paths = (CopyFileParams\*)params;

HANDLE hSrcFile = CreateFileW(paths->srcPath, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_FLAG\_OVERLAPPED, NULL);

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

wprintf(L"Failed to open source file.\n");

return 1;

}

HANDLE hDstFile = CreateFileW(paths->dstPath, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_FLAG\_OVERLAPPED, NULL);

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

wprintf(L"Failed to create destination file.\n");

CloseHandle(hSrcFile);

return 1;

}

BYTE buffer[BUFFER\_SIZE];

OVERLAPPED olRead = { 0 };

OVERLAPPED olWrite = { 0 };

olRead.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

olWrite.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

if (olRead.hEvent == NULL || olWrite.hEvent == NULL) {

wprintf(L"Failed to create events.\n");

CloseHandle(hSrcFile);

CloseHandle(hDstFile);

return 1;

}

DWORD bytesRead, bytesWritten;

bool isReading = true;

while (isReading) {

if (!ReadFile(hSrcFile, buffer, BUFFER\_SIZE, NULL, &olRead)) {

if (GetLastError() == ERROR\_IO\_PENDING) {

WaitForSingleObject(olRead.hEvent, INFINITE);

}

else {

wprintf(L"ReadFile failed.\n");

break;

}

}

if (GetOverlappedResult(hSrcFile, &olRead, &bytesRead, FALSE) && bytesRead > 0) {

if (!WriteFile(hDstFile, buffer, bytesRead, NULL, &olWrite)) {

if (GetLastError() == ERROR\_IO\_PENDING) {

WaitForSingleObject(olWrite.hEvent, INFINITE);

}

else {

wprintf(L"WriteFile failed.\n");

break;

}

}

if (!GetOverlappedResult(hDstFile, &olWrite, &bytesWritten, FALSE) || bytesWritten != bytesRead) {

wprintf(L"WriteFile incomplete.\n");

break;

}

olRead.Offset += bytesRead;

olWrite.Offset += bytesWritten;

}

else {

isReading = false;

}

}

CloseHandle(olRead.hEvent);

CloseHandle(olWrite.hEvent);

CloseHandle(hSrcFile);

CloseHandle(hDstFile);

free(paths);

return 0;

}

int main() {

const WCHAR\* srcPath = L"E:\\Study\\InstituteRep\\semester\_5\\OOS\\lab7\\from\\temp.txt";

const WCHAR\* dstPath = L"E:\\Study\\InstituteRep\\semester\_5\\OOS\\lab7\\to\\temp.txt";

CopyFileParams\* params = (CopyFileParams\*)malloc(sizeof(CopyFileParams));

params->srcPath = srcPath;

params->dstPath = dstPath;

HANDLE hThread = CreateThread(NULL, 0, CopyFileThread, params, 0, NULL);

if (hThread == NULL) {

wprintf(L"Failed to create thread.\n");

free(params);

}

else {

WaitForSingleObject(hThread, INFINITE);

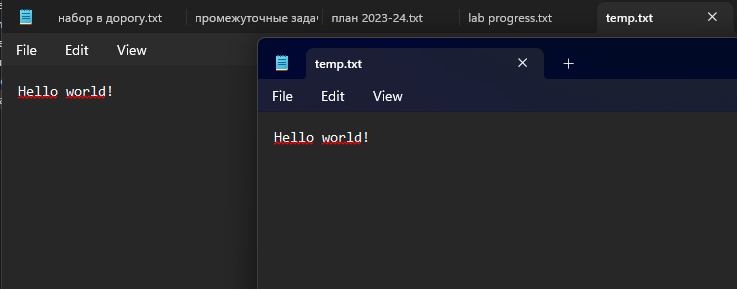
CloseHandle(hThread);

}

return 0;

}

**Результат работы программы**



**Вывод**

Изучили способы работы с файлами в Win32 API. Изучили асинхронный режим работы с файлами