Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра «МОП ЭВМ»

ЛАБОРАТОРНАЯ РАБОТА 5

по дисциплине «Операционные системы и оболочки»

## Синхронизация потоков в операционной системе Windows

Студент группы 6ИСб-1 И. Нозимзода

Преподаватель В.Я Столяров

2019

**Цель работы**: Изучить теоретические вопросы управления синхронизацией программных потоков в ОС Windows и освоить приемы практической реализации этого управления с использованием системных функций.

**Задание:** Один поток читает файл, указанный в первом параметре программы, второй поток читает файл, указанный во втором параметре программы. Третий поток сравнивает файлы и выводит на экран номера и значения несовпадающих байтов.

**Выполнение:**

Функция **ShellExecute** может не только выполнять заданное приложение, но и открывать документ и печатать его. Под термином «открыть файл документа» понимается выполнение связанного с ним приложения и загрузка в него этого документа. Например, обычно с документами, имеющими расширение .doc, связан Word. В этом случае открыть файл, например, с именем file.doc означает запус­тить Word и передать ему в качестве параметра имя файла file.doc. Кроме описан­ных возможностей функция ShellExecute позволяет открыть указанную папку. Это означает, что будет запущена программа «Проводник» с открытой указанной папкой.

Для использования функции ShellExecute в операторе uses надо добавить мо­дуль ShellAPI. Автоматически Delphi не включает этот модуль в программу.

Функция ShellExecute определена следующим образом:

function ShellExecute(Wnd: HWnd; Operation, FileName, Parameters,

Directory: PChar; ShowCmd: Integer): THandle;

Параметр Wnd является дескриптором родительского окна, в котором отобра­жаются сообщения запускаемого приложения. Обычно в качестве него можно про­сто указать Handle.

Параметр Operation указывает на строку с нулевым символом в конце, кото­рая определяет выполняемую операцию. Эта строка может содержать текст «open» (открыть) или «print» (напечатать). Для Windows 95/98 и NT определено еще одно значение: «explore» (исследовать) — открыть папку. Если параметр Operation ра­вен nil, то по умолчанию выполняется операция «open».

Параметр FileName указывает на строку с нулевым символом в конце, кото­рая определяет имя открываемого файла или имя открываемой папки.

Параметр Parameters указывает на строку *с*нулевым символом в конце, кото­рая определяет передаваемые в приложение параметры, если FileName определяет выполняемый файл. Если FileName указывает на строку, определяющую откры­ваемый документ или папку, то этот параметр задается равным null.

Параметр Directory указывает на строку с нулевым символом в конце, которая определяет каталог по умолчанию.

Параметр ShowCmd определяет режим открытия указанного файла. Этот па­раметр может иметь множество различных значений. Обычно, как и для функции WinExec, используется значение SW\_RESTORE, при котором окно запускаемого приложения активизируется и отображается на экране.

Текст программы по индивидуальному заданию приведен в листингах 1,2.

Листинг 1 – Текст файла SyncThread.cpp

|  |
| --- |
| #include <iostream>  #include <windows.h>  #include <process.h>  #include <fstream>  #include <cstdlib>  using namespace std;  ////// GLOBAL VARIABLE //////  // События завершения считывания и анализа байтов  HANDLE hEventRead[2], hEventСomparison[2];  // Потоrи для чтения байтов из файлов  ifstream filestream1;  ifstream filestream2;  // Хранение извлеченного байта из первого и второго потока  char byte1;  char byte2;  // Состояние окончания файлов  bool filestream1Compl;  bool filestream2Compl;  // Состояние окончаник сравнения  bool comparisonCompl;  ///////////////////////////  // Побайтовое чтение первого файла  unsigned \_\_stdcall read1(void\* stream)  {  filestream1Compl = false;  while (!comparisonCompl)  {  // Поток comparison закончил сравнение байтов, извлекаем дальше  WaitForSingleObject(hEventСomparison[0], INFINITE);  if (!filestream1.get(byte1)) filestream1Compl = true;  // Сигнализируем о завершении извлечения байта  SetEvent(hEventRead[0]);  }  return 0;  }  // Побайтовое чтение второго файла  unsigned \_\_stdcall read2(void\* stream)  {  filestream2Compl = false;  while (!comparisonCompl)  {  WaitForSingleObject(hEventСomparison[1], INFINITE);  if (!filestream2.get(byte2)) filestream2Compl = true;  SetEvent(hEventRead[1]);  }  return 0;  }  // Сравнение байтов  unsigned \_\_stdcall comparison(void\*)  {  // Хранение номера сравниваемого байта  size\_t indexByte = 0;  comparisonCompl = false;  while (!comparisonCompl)  {  // Третий параметр TRUE, если нужно ждать сигнализации всех событий, FALSE если любого события из массива  WaitForMultipleObjects(2, hEventRead, TRUE, INFINITE);  // Если первый поток и второй поток завершили работу выходим  if (filestream1Compl == true && filestream2Compl == true)  {  comparisonCompl = true;  // Для каждого потока сообщаем что их байты проанализированы  SetEvent(hEventСomparison[0]);  SetEvent(hEventСomparison[1]);  break;    }  ++indexByte;  if (byte1 != byte2)  {  cout << "\*НАЙДЕНО НЕСОВПАДЕНИЕ\*" << endl;  cout << "Номер не совпадающего байта: " << indexByte << endl;  cout << "Байт из первого файла (в виде HEX): ";  if (filestream1Compl) cout << "Отсутствует";  else cout << hex << (int)byte1 << " (" << byte1 << ")";  cout << endl;  cout << "Байт из второго файла (в виде HEX): ";  if (filestream2Compl) cout << "Отсутствует";  else cout << hex << (int)byte2 << " (" << byte2 << ")";  cout << endl << endl;  }    // Для каждого потока сообщаем что их байты проанализированы  SetEvent(hEventСomparison[0]);  SetEvent(hEventСomparison[1]);  }  return 0;  }  int main(int argc, char\* argv[])  {  setlocale(LC\_ALL, "rus");  system("color 0A");  // если передаем аргументы, то argc будет больше 1 (в зависимости от кол-ва аргументов)  if (argc != 3)  {  cout << "При запуске программы передайте два аргумента: путь до первого и второго файла." << endl;  system("pause");  return -1;  }  cout << "Примечание: для корректной работы программы сравниваемые файлы должны быть в кодировке Windows-1251 (или другие однобайтовые кодировки)." << endl << endl;  // Открытие поток по путям из параметров  filestream1.open(argv[1], ios::binary);  filestream2.open(argv[2], ios::binary);  // Проверка открытия потоков  if (filestream1 && filestream2)  {  // Инициализация событий  // 2 пар: Вид сброса событий false - РУЧНОЙ, true - авто  // 3 пар: Начальное состояние события  hEventRead[0] = CreateEvent(NULL, FALSE, FALSE, NULL);  hEventRead[1] = CreateEvent(NULL, FALSE, FALSE, NULL);  hEventСomparison[0] = CreateEvent(NULL, FALSE, TRUE, NULL);  hEventСomparison[1] = CreateEvent(NULL, FALSE, TRUE, NULL);  HANDLE hThread[3];  // Потоки извлечения  // Передаём потокам поток на первый и второй файл  hThread[0] = (HANDLE)\_beginthreadex(NULL, 0, &read1, &filestream1, 0, NULL);  hThread[1] = (HANDLE)\_beginthreadex(NULL, 0, &read2, &filestream2, 0, NULL);  // Поток сравнения  hThread[2] = (HANDLE)\_beginthreadex(NULL, 0, &comparison, NULL, 0, NULL);  // Ожидание завершения всех потоков  WaitForMultipleObjects(3, hThread, TRUE, INFINITE);  cout << "Все потоки завершили работу." << endl;  filestream1.close();  filestream2.close();  }  else cout << "Ошибка открытия файлов!" << endl;  system("pause");  return 0;  } |

Программа должна запускаться с двумя параметрами, содержащими путь к файлам, которые требуется сравнить. После этого она открывает два потока в бинарном режиме которые синхронно извлекают из файлов байты и передают их в поток сравнения. После завершения сравнения байтов, потоки извлечений продолжат свою работу. Синхронность выполняется с помощью событий.

Запуск программы с параметрами через cmd (рисунок 1).



Рисунок 1 – Запуск программы

Файлы из которого будут извлекаться байты показаны на рисунке 1,2.

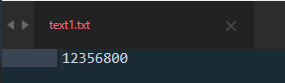


Рисунок 2 – Первый текстовый файл

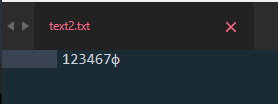


Рисунок 3 – Второй текстовый файл

Для правильной работы программы файлы должны быть сохранены в однобайтовой кодировке, например Windows 1251. Результат работы программы по сравнению файлов показан на рисунке 4.

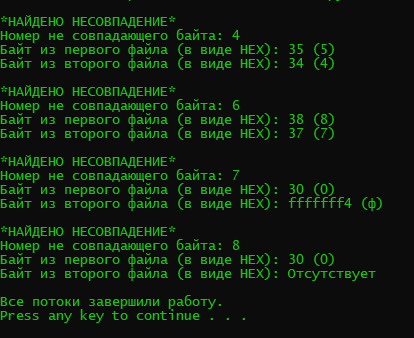


Рисунок 3 – Второй текстовый файл

**Список использованных источников**

1. Саймон, Р. Windows 2000 API Энциклопедия программиста / Р. Саймон. - М: DiaSoft, 2002