|  |
| --- |
| **Лабораторная работа. Обмен информацией между процессами с помощью отображаемых в память файлов.** |
|  |  |
| **Цель работы**: изучить возможности применения механизма обмена данными между процессами в операционной системе Windows на основе отображаемых в память файлов.    **Темы для предварительной проработки**: отображаемые в память файлы, процессы и нити в операционных системах, жизненный цикл процесса, средства взаимодействия процессов IPC, интерфейс прикладного программирования API WIN32.    **Теоретические сведения**    *Отображаемые в память файлы.*  Способ обмена данными между процессами с помощью файлов, отображенных на память, обладает высоким быстродействием, так как данные передаются между процессами непосредственно через виртуальную память.  Методика работы с файлами, отображаемыми на память, заключается в следующем.  Отображение создается функцией **CreateFileMapping**.  В качестве первого параметра для функции передается идентификатор открытого файла. Если отображение будет использоваться для передачи данных между процессами, удобно указать для него имя. Пользуясь этим именем, другие процессы смогут открыть отображение функцией **OpenFileMapping**.  Если создается отображение только для того чтобы обеспечить передачу данных между процессами, не нужно создавать файл на диске компьютера. Указав в качестве идентификатора файла значение **(HANDLE)0xFFFFFFFF**, создается отображение непосредственно в виртуальной памяти без использования дополнительного файла.  После создания объекта-отображения, следует выполнить отображение файла в память при помощи функции **MapViewOfFile**. В случае успеха эта функция вернет указатель на отображенную область памяти.  Перед завершением своей работы процессы должны отменить отображение файла **UnmapViewOfFile** и освободить идентификатор созданного объекта-отображения **CloseHandle**.    Пример. Один экземпляр программы создает отображаемый в память файл и записывает в него тестовую строку. Второй экземпляр – считывает строку из отображаемого файла. Для синхронизации используются события.    #include "windows.h"  #include "stdio.h"    HANDLE hEvent1, hEvent2;  HANDLE hMFile;  char Ev1[]="MyEvent1";  char Ev2[]="MyEvent2";  char Fn1[]="MyMemoryMappedFile";  char Mask;  LPVOID StartMFile;  char Buf[]="Buffer Is Empty              ";  int main(int argc, char\* argv[])  {              Mask=FALSE;              if(!OpenEvent(EVENT\_ALL\_ACCESS, TRUE, Ev1)) {                          printf("SERVER\n");                          if(hEvent1=CreateEvent( NULL, TRUE, FALSE, Ev1 )) {                                     hEvent2=CreateEvent( NULL, TRUE, FALSE, Ev2 );                                     Mask=TRUE;} else { return 0;}              } else {                          hEvent2=OpenEvent(EVENT\_ALL\_ACCESS, TRUE, Ev2);                          printf("CLIENT\n");              }              if(Mask)              {                          hMFile=CreateFileMapping((HANDLE)0xFFFFFFFF, NULL,                                     PAGE\_READWRITE, 0, 100, Fn1);                          if(!hMFile) { return 0;}                          StartMFile=MapViewOfFile(hMFile, FILE\_MAP\_WRITE, 0, 0, 100);                          if(!StartMFile) {  return 0;}                          printf("BEFORE RECORD:\n%s\n",Buf);                          CopyMemory(StartMFile, "Information From Server", 24);                          printf("AFTER  RECORD:\n%s\n",Buf);                          PulseEvent(hEvent1);                          WaitForSingleObject(hEvent2, INFINITE);              } else              {                          WaitForSingleObject(hEvent1, INFINITE);                          hMFile=OpenFileMapping(FILE\_MAP\_READ, TRUE, Fn1);                          if(!hMFile) { return 0;}                          StartMFile=MapViewOfFile(hMFile, FILE\_MAP\_READ, 0, 0, 100);                          if(!StartMFile) { return 0;}                          printf("BEFORE RECORD: \n%s\n",Buf);                          CopyMemory(Buf, StartMFile, 24);                          printf("AFTER  RECORD: \n%s\n",Buf);                          PulseEvent(hEvent2);              }              UnmapViewOfFile(hMFile);              CloseHandle(hMFile);              return 0;  }    **Порядок выполнения работы**    1. Ознакомиться с постановкой задачи и исходными данными. В соответствии с номером по журналу определить вариант задачи.  2. Согласно рекомендациям, приведенным в исходных данных, сконструировать структуру программы.  3. Составить тексты программ и утвердить их у преподавателя.  4. Пункты 1 - 3 должны быть выполнены предварительно до проведения данной лабораторной работы. Утверждение преподавателем текстов программ является допуском к лабораторной работе.  5. Набрать текст программы.  6. Выполнить компиляцию программы.  7. Провести анализ и исправление обнаруженных синтаксических ошибок в тексте программы и повторить пункты 6. и 7. При устранении всех синтаксических ошибок перейти к выполнению пункта 8.  8. Получить решение и, в случае обнаружения логических ошибок, описать и устранить их. Продемонстрировать преподавателю окончательный вариант программы и ее работу.    **Содержание отчета**    1. Тема лабораторной работы.  2. Цель работы.  3. Индивидуальное задание.  4. Метод и алгоритм решения задачи.  5. Текст программы.  6. Результаты работы программы.  7. Выводы по работе.    **Задачи**    I. Задача «производители-потребители». Решается проблема работы с нескольких процессов (нитей) с одним буфером. Часть процессов являются «производителями»: в случайные моменты времени производят запись информации в буфер. Часть процессов являются «потребителями»: в случайные моменты читают информацию из буфера (после чтения информация в буфере теряется). Необходимо организовать такую работу, чтобы не было коллизий при совместной работе «производителей» и «потребителей».  II. Задача «читатели-писатели». Имеются данные, совместно используемые нескольким процессами (нитями). Имеется несколько процессов (нитей), которые только читают эти данные («читатели») и несколько других, которые только записывают данные (или изменяют их) («писатели»). При этом должны удовлетворяться следующие условия:  – любое число «читателей» могут одновременно читать данные;  – записывать данные в определенный момент может только один «писатель».  – когда «писатель» записывает данные, ни один читатель не может их читать.      **Индивидуальные задания**  15. Задача II. Механизм синхронизации – события. Преимущество имеют «писатели». 1 процесс – «писатель», 3 – «читателя». | |