|  |
| --- |
| **Лабораторная работа. Процессы, нити, волокна в операционной системе Microsoft Windows.** |
|  |  |
| **Цель работы**: изучить возможности по созданию, управлению выполнением и завершению процессов, нитей и волокон в операционной системе Windows.  **Теоретические сведения**  *1. Краткая инструкция по созданию консольных приложений в среде программирования Visual C*  Для начала создания консольного приложения в среде программирования Visual C необходимо создать новый проект (последовательность действий через пункты меню и закладки должна быть следующей: **File 🡪 New 🡪 Projects 🡪 Win32 Console Application**, в строке ввода **Project Name** необходимо указать имя проекта, а в строке ввода **Location** – путь к файлам проекта). Для упрощения создания файлов проекта можно выбрать в мастере создания проектов один из вариантов: пустой проект, простое консольное приложение (содержит главный файл приложения с кодом функции main), простое консольное приложение «Hello, World!» (приложение, содержащее код простейшей программы, выводящей на экран указанное сообщение), а также приложение с использование MFC. В данной лабораторной работе необходимо выбрать простое консольное приложение.  Главный файл проекта с исходным текстом программы будет называться также как и сам проект и иметь расширение cpp. Открыть этот файл для редактирования можно через закладку **FileView 🡪 WorkSpace 🡪 Files 🡪 Source Files**.  Выполнение компиляции производится путем нажатия комбинации клавиш Ctrl+F7, создание исполняемого файла – F7, запуск программы на выполнение – Ctrl+F5. Настройки создаваемого проекта могут быть изменены с помощью пункта меню **Projects 🡪 Settings** или по нажатию комбинации клавиш Alt+F7.    *2. Процессы, нити и волокна в ОС Windows*  Процессом обычно называют экземпляр выполняемой программы. Хотя на первый взгляд кажется, что программа и процесс понятия практически одинаковые, они фундаментально отличаются друг от друга. Программа представляет собой статический набор команд, а процесс это набор ресурсов и данных, использующихся при выполнении программы. Процесс в Windows состоит из следующих компонентов:  - Структура данных, содержащая всю информацию о процессе, в том числе список открытых дескрипторов различных системных ресурсов, уникальный идентификатор процесса, различную статистическую информацию и т.д.;  - Адресное пространство - диапазон адресов виртуальной памяти, которым может пользоваться процесс;  - Исполняемая программа и данные, проецируемые на виртуальное адресное пространство процесса.  Процессы инертны. Отвечают же за исполнение кода, содержащегося в адресном пространстве процесса, нити. Нить (thread) - некая сущность внутри процесса, получающая процессорное время для выполнения. В каждом процессе есть минимум одна нить. Эта первичная нить создается системой автоматически при создании процесса. Далее эта нить может породить другие нити, те в свою очередь новые и т.д. Таким образом, один процесс может владеть несколькими нитями, и тогда они одновременно исполняют код в адресном пространстве процесса. Каждая нить имеет:  - Уникальный идентификатор нити;  - Содержимое набора регистров процессора, отражающих состояние процессора;  - Два стека, один из которых используется нитью при выполнении в режиме ядра, а другой - в пользовательском режиме;  - Закрытую область памяти, называемую локальной памятью нити (thread local storage, TLS) и используемую подсистемами, run-time библиотеками и DLL.  Чтобы все нити работали, операционная система отводит каждой из них определенное процессорное время. Тем самым создается иллюзия одновременного выполнения нитей (разумеется, для многопроцессорных компьютеров возможен истинный параллелизм). В Windows реализована система вытесняющего планирования на основе приоритетов, в которой всегда выполняется нить с наибольшим приоритетом, готовая к выполнению. Выбранная для выполнения нить работает в течение некоторого периода, называемого квантом. Квант определяет, сколько времени будет выполняться нить, пока операционная система не прервет ее. По окончании кванта операционная система проверяет, готова ли к выполнению другая нить с таким же (или большим) уровнем приоритета. Если таких нитей не оказалось, текущей нити выделяется еще один квант. Однако нить может не полностью использовать свой квант. Как только другая нить с более высоким приоритетом готова к выполнению, текущая нить вытесняется, даже если ее квант еще не истек.  Квант не измеряется в каких бы то ни было единицах времени, а выражается целым числом. Для каждой нити хранится текущее значение ее кванта. Когда нити выделяется квант процессорного времени, это значит, что ее квант устанавливается в начальное значение. Оно зависит от операционной системы. Например, для Win2000 Professional начальное значение кванта равно 6, а для Win2000 Server - 36.  Всякий раз, когда возникает прерывание от таймера, из кванта нити вычитается 3, и так до тех пор, пока он не достигнет нуля. Частота срабатывания таймера зависит от аппаратной платформы. Например, для большинства однопроцессорных x86 систем он составляет 10мс, а на большинстве многопроцессорных x86 систем - 15мс.  В любом случае операционная система должна определить, какую нить выполнять следующей. Выбрав новую нить, операционная система переключает контекст. Эта операция заключается в сохранении параметров выполняемой нити (регистры процессора, указатели на стек ядра и пользовательский стек, указатель на адресное пространство, в котором выполняется нить и др.), и загрузке аналогичных параметров для другой нити, после чего начинается выполнение новой нити.  Планирование в Windows осуществляется на уровне нитей, а не процессов. Это кажется понятным, так как сами процессы не выполняются, а лишь предоставляют ресурсы и контекст для выполнения нитей. Поэтому при планировании нитей, система не обращает внимания на то, какому процессу они принадлежат. Например, если процесс А имеет 10 готовых к выполнению нитей, а процесс Б - два, и все 12 нитей имеют одинаковый приоритет, каждая из них получит 1/12 процессорного времени.  В Windows существует 32 уровня приоритета, от 0 до 31. Они группируются так: 31 - 16 уровни реального времени; 15 - 1 динамические уровни; 0 - системный уровень, зарезервированный для процесса обнуления страниц (zero-page thread).  Приоритет каждой нити (базовый приоритет нити) складывается из приоритета ее процесса и относительного приоритета самой нити. Есть семь относительных приоритетов нитей:  Normal: такой же как и у процесса;  Above normal: +1 к приоритету процесса;  Below normal: -1;  Highest: +2;  Lowest: -2;  Time critical: устанавливает базовый приоритет потока для Real time класса в 31, для остальных классов в 15.  Idle: устанавливает базовый приоритет потока для Real time класса в 16, для остальных классов в 1.  Если операционная система выполняется на машине, где установлено более одного процессора, то по умолчанию, нить выполняется на любом доступном процессоре. Однако в некоторых случаях, набор процессоров, на которых нить может работать, может быть ограничен. Это явление называется привязкой к процессорам (processor affinity). Можно изменить привязку к процессорам программно, через Win32-функции планирования.  Волокна поддерживаются в WIN32 API, начиная с Windows2000. Под волокном понимается упрощенная нить, выполнение которой планируется самим приложением, а не планировщиком процессорного времени ОС. Планирование волокон может осуществляться только путем переключения на них только из других волокон. Волокна выполняются в контексте нитей, в которых планируется их применение, и допускают полную их идентификацию с нитями. В каждой нити может быть запланировано несколько волокон. Для каждого волокна создается собственный стек, в котором хранится информация о состоянии волокна.    *3. Создание процессов*  Создание Win32 процесса осуществляется вызовом одной из таких функций, как **CreateProcess**, **CreateProcessAsUser** (для WinNT/2000) и **CreateProcessWithLogonW** (начиная с Win2000) и происходит в несколько этапов:  - Открывается файл образа (EXE), который будет выполняться в процессе. Если исполняемый файл не является Win32 приложением, то ищется образ поддержки (support image) для запуска этой программы. Например, если исполняется файл с расширением .bat, запускается cmd.exe и т.п.  В WinNT/2000 для отладки программ реализовано следующее. **CreateProcess**, найдя исполняемый Win32 файл, ищет в SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Option раздел с именем и расширением запускаемого файла, затем ищет в нем параметр Debugger, и если строка не пуста, запускает то, что в ней написано вместо данной программы.  - Создается объект Win32 "процесс".  - Создается первичная нить (стек, контекст и объект "нить").  - Подсистема Win32 уведомляется о создании нового процесса и нити.  - Начинается выполнение первичной нити.  - В контексте нового процесса и потока инициализируется адресное пространство (например, загружаются требуемые DLL) и начинается выполнение программы.    *4. Завершение процессов*  Процесс завершается если:  - Входная функция первичной нити возвратила управление.  - Одна из нитей процесса вызвала функцию **ExitProcess**.  - Нить другого процесса вызвала функцию **TerminateProcess**.  Когда процесс завершается, все User- и GDI-объекты, созданные процессом, уничтожаются, объекты ядра закрываются (если их не использует другой процесс), адресное пространство процесса уничтожается.  Пример: Программа создает процесс "Калькулятор".    #include <windows.h>  int main(int argc, char\* argv[])  {  STARTUPINFO si;  PROCESS\_INFORMATION pi;  ZeroMemory( &si, sizeof(si) );  si.cb = sizeof(si);  ZeroMemory( &pi, sizeof(pi) );  if( !CreateProcess( NULL, "c:\\windows\\calc.exe", NULL, NULL, FALSE,  0, NULL, NULL, &si, &pi))  return 0;  // Close process and thread handles.  CloseHandle( pi.hProcess );  CloseHandle( pi.hThread );  return 0;  }    *5. Создание нитей*  Первичная нить создается автоматически при создании процесса. Остальные нити создаются функциями **CreateThread** и **CreateRemoteThread** (только в WinNT/2000/XP).    *6. Завершение нитей*  Нить завершается если  - Функция нити возвращает управление.  - Нить самоуничтожается, вызвав **ExitThread**.  - Другая нить данного или стороннего процесса вызывает **TerminateThread**.  - Завершается процесс, содержащий данную нить.    *7. Создание волокон.*  Волокно может быть создано с помощью системного вызова **CreateFiber** из основной нити процесса или получено путем преобразования текущей нити с помощью функции **ConvertThreadToFiber**. Переключение между волокнами может быть организовано с помощью функции **SwitchToFiber**, но ее вызов можно осуществлять только из волокна.  Пример: Программа создает 4 волокна, каждое из которых выполняет переключение на следующее волокно, если число переключений больше 10, работа завершается.    // для работы с волокнами следующие 2 строки обязательны!  #define WIN32\_LEAN\_AND\_MEAN  #define \_WIN32\_WINNT 0x0400  #include "stdio.h"  #include "windows.h"  int Counter;  void \*fiber[5];  void WINAPI Func(void \*)  {  for(;;) {  printf("Fiber Number %d\n",Counter%4);  if ((Counter++)<10) {  SwitchToFiber(fiber[Counter%4]);  } else break;  }  }  int main(int argc, char\* argv[])  {  Counter=0;  fiber[0]=CreateFiber(0,Func,NULL);  fiber[1]=CreateFiber(0,Func,NULL);  fiber[2]=CreateFiber(0,Func,NULL);  fiber[3]=CreateFiber(0,Func,NULL);  // для переключения на первое волокно необходимо преобразовать текущую нить в волокно  fiber[4]=ConvertThreadToFiber(NULL);  SwitchToFiber(fiber[0]);  return 0;  }    *8. Уничтожение волокон.*  Для принудительного уничтожения волокна применяется функция **DeleteFiber**. Также для уничтожения волокна могут быть использованы все функции, предназначенные для уничтожения нитей.    *9. Wait функции*  Для приостановки выполнения работы нити можно использовать различные способы.  Функция **Sleep()** приостанавливает работу нити на заданное число миллисекунд. Если в качестве аргумента вы укажите 0 ms, то произойдет следующее. Нить откажется от своего кванта процессорного времени, однако тут же появится в списке нитей, готовых к выполнению. Иными словами произойдет намеренное переключение нитей (вернее сказать, попытка переключения - ведь следующей для выполнения нитью вполне может стать та же самая).  Пример. Программа создает процесс "Калькулятор" и через 15 секунд его уничтожает.    #include <windows.h>  int main(int argc, char\* argv[])  {  STARTUPINFO StartUpInfo;  PROCESS\_INFORMATION ProcessInfo;  memset(&StartUpInfo,0,sizeof(STARTUPINFO));  StartUpInfo.cb=sizeof(STARTUPINFO);  If (CreateProcess(NULL,"c:\\windows\\calc.exe",NULL,NULL,FALSE,  NORMAL\_PRIORITY\_CLASS,NULL,NULL,&StartUpInfo,&ProcessInfo))  {  Sleep(15000);  TerminateProcess(ProcessInfo.hProcess, 0);  }  ExitProcess(0);  }    Для получения полной информации о назначении, использовании и синтаксисе всех функций WIN32 API необходимо воспользоваться системой помощи MS SDK, входящей в состав сред программирования Borland Delphi или CBuilder, а также MSDN, поставляемым в составе системы программирования Visual C.    **Постановка задачи**: разработать консольное приложение в среде программирования Visual C, которое выполняет следующие действия: запускает и уничтожает процессы по определенным, самостоятельно выбранным, условиям; обязательно использует нити и волокна. | |