МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Безопасность информационных и автоматизированных систем»

СОЗДАНИЕ ПРОЦЕССОВ И НИТЕЙ. СИНХРОНИЗАЦИЯ НИТЕЙ

Методические указания
к выполнению лабораторной работы
по дисциплине «Безопасность операционных систем»
для студентов специальностей 090105, 090303, 090900

Курган 2012

Кафедра: «Безопасность информационных и автоматизированных систем»

Дисциплина: «Безопасность операционных систем» (специальности 090105, 090303, 090900).

Составил: ст. преподаватель А.Г. Рабушко

Утверждены на заседании кафедры «6» марта 2012 г.

Рекомендованы методическим советом университета «30» марта 2012 г.

ПРИБОРЫ, ОБОРУДОВАНИЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

1 Операционная система (Microsoft Windows Xp).
2 Используемая среда программирования (Visual Studio C++).

ЦЕЛЬ РАБОТЫ: Изучение основных функций операционной системы (ОС) для создания процессов и нитей и средств их синхронизации

ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

Процесс на уровне Win API создается функцией семейства CreateProcess[*]

```
BOOL WINAPI CreateProcess(
  __in        LPCTSTR lpApplicationName,
  __in_out     LPTSTR lpCommandLine,
  __in        LPSECURITY_ATTRIBUTES lpProcessAttributes,
  __in        LPSECURITY_ATTRIBUTES lpThreadAttributes,
  __in        BOOL bInheritHandles,
  __in        DWORD dwCreationFlags,
  __in        LPVOID lpEnvironment,
  __in        LPCTSTR lpCurrentDirectory,
  __in        LPSTARTUPINFO lpStartupInfo,
  __out       LPPROCESS_INFORMATION lpProcessInformation
);
```

При создании процесса можно указать его атрибуты безопасности и атрибуты безопасности главной нити:

```
typedef struct _SECURITY_ATTRIBUTES {
   DWORD nLength;
   LPVOID lpSecurityDescriptor;
   BOOL bInheritHandle;
} SECURITY_ATTRIBUTES, *PSECURITY_ATTRIBUTES,
*LPSECURITY_ATTRIBUTES;
```

В поле dwCreationFlags можно указать в частности класс приоритета процесса:

```
#define NORMAL_PRIORITY_CLASS          0x00000020
#define IDLE_PRIORITY_CLASS            0x00000040
#define HIGH_PRIORITY_CLASS            0x00000080
#define REALTIME_PRIORITY_CLASS        0x00000100
#define BELOW_NORMAL_PRIORITY_CLASS    0x00004000
#define ABOVE_NORMAL_PRIORITY_CLASS    0x00008000
```

Важную роль при создании процесса играет структура

```
typedef struct _STARTUPINFO {
   DWORD   cb;
   LPSTR   lpReserved;
   LPSTR   lpDesktop;
   LPSTR   lpTitle;
   DWORD   dwX;
   DWORD   dwY;
   DWORD   dwXSize;
   DWORD   dwYSize;
   DWORD   dwXCountChars;
   DWORD   dwYCountChars;
   DWORD   dwFillAttribute;
   DWORD   dwFlags;
   WORD    wShowWindow;
   WORD    cbReserved2;
   LPBYTE  lpReserved2;
   HANDLE  hStdInput;
   HANDLE  hStdOutput;
   HANDLE  hStdError;
} STARTUPINFO, *LPSTARTUPINFO;
```

В ней, в частности, можно указать, на каком десктопе создается процесс, и переназначить стандартный ввод-вывод.

dwFlags - содержит набор флагов, позволяющих управлять созданием дочернего процесса. Большая часть флагов просто сообщает функции CreateProcess, содержат ли прочие элементы структуры

STARTUPINFO полезную информацию или некоторые из них можно игнорировать. Ниже приведен список допустимых флагов.

STARTF_USESIZE

Заставляет использовать элементы divSize и dwYSize

STARTF_USESHOWWINDOW

Заставляет использовать элемент wShowWindow

STARTF_USEPOSITION

Заставляет использовать элементы dwX и dwY

STARTF_USECOTUNTCHARS

Заставляет использовать элементы dwXCountChars и dwYCountCbars

STARTF_USEFILLATTRIBUTE

Заставляет использовать элемент dwFillAttnbute

STARTF_USESTDHANDLES

Заставляет использовать элементы hStdlnput, hStdOutput и bStdError

STARTF_RUNFULLSCREEN

Приводит к тому, что консольное приложение запускается в полноэкранном режиме.

Два дополнительных флага — STARTF_FORCEONFEEDBACK и STARTF_FORCEOFFFEEDBACK — позволяют контролировать форму курсора мыши в момент запуска нового процесса.

При успешном создании информация о процессе и нити оформляется в виде структуры

```
typedef struct _PROCESS_INFORMATION {
   HANDLE hProcess;
   HANDLE hThread;
   DWORD dwProcessId;
   DWORD dwThreadId;
} PROCESS_INFORMATION, *PPROCESS_INFORMATION, *LPPROCESS_INFORMATION;
```

Созданному объекту ядра «процесс» присваивается уникальный идентификатор; ни у каких других объектов этого типа в системе не может быть одинаковых идентификаторов. Это же касается и объектов ядра «нить». Причем идентификаторы процесса и нити тоже разные, и их значения никогда не бывают нулевыми. Завершая свою работу, CreateProcess заносит значения идентификаторов в элементы dwProcessId и dwThreadId структуры PROCESS_INFORMATION. Эти идентификаторы просто облегчают определение процессов и нитей в системе; их используют, как правило, лишь специализированные утилиты вроде Task Manager.

Наиболее важные функции для использования с процессами:

CreateProcess CreateProcessAsUser CreateProcessWithLogonW CreateProcessWithTokenW ExitProcess FlushProcessWriteBuffers FreeEnvironmentStrings GetCommandLine GetCurrentProcess GetCurrentProcessId GetCurrentProcessorNumber GetEnvironmentStrings

GetEnvironmentVariable GetExitCodeProcess GetGuiResources GetLogicalProcessorInformation GetPriorityClass GetProcessAffinityMask

GetProcessHandleCount GetProcessId GetProcessIdOfThread GetProcessIoCounters GetProcessPriorityBoost GetProcessShutdownParameters

GetProcessTimes GetProcessVersion GetProcessWorkingSetSize

GetProcessWorkingSetSizeEx GetStartupInfo NeedCurrentDirectoryForExePath

OpenProcess QueryFullProcessImageName QueryProcessAffinityUpdateMode

QueryProcessCycleTime SetEnvironmentVariable SetPriorityClass

SetProcessAffinityMask SetProcessAffinityUpdateMode SetProcessPriorityBoost

SetProcessShutdownParameters SetProcessWorkingSetSize

SetProcessWorkingSetSizeEx TerminateProcess

Нить на уровне Win API создается функцией

CreateThread(

```
    __in_opt  LPSECURITY_ATTRIBUTES lpThreadAttributes,
    __in      SIZE_T dwStackSize,
    __in      LPTHREAD_START_ROUTINE lpStartAddress,
    __in_opt  LPVOID lpParameter,
    __in      DWORD dwCreationFlags,
    __out_opt LPDWORD lpThreadId
    );
```

Нить создается с приоритетом THREAD_PRIORITY_NORMAL. Для его изменения модно использовать функции GetThreadPriority and SetThreadPriority.

Для создания нити в адресном пространстве другого процесса при наличии соответствующих прав используется функция

```
CreateRemoteThread(
    __in      HANDLE hProcess,
    __in_opt  LPSECURITY_ATTRIBUTES lpThreadAttributes,
    __in      SIZE_T dwStackSize,
    __in      LPTHREAD_START_ROUTINE lpStartAddress,
    __in_opt  LPVOID lpParameter,
    __in      DWORD dwCreationFlags,
    __out_opt LPDWORD lpThreadId
    );
```

Для работы с критическими секциями используются функции:

```
WINBASEAPI
VOID
WINAPI
InitializeCriticalSection(
    __out LPCRITICAL_SECTION lpCriticalSection
    );

WINBASEAPI
VOID
WINAPI
EnterCriticalSection(
    __inout LPCRITICAL_SECTION lpCriticalSection
    );

WINBASEAPI
VOID
WINAPI
LeaveCriticalSection(
    __inout LPCRITICAL_SECTION lpCriticalSection
    );
```

Первое, что делает *EnterCriticalSection*, — исследует значения элементов структуры CRITICAL_SECTION. Если ресурс занят, в них содержатся сведения о том, какая нить пользуется ресурсом. *EnterCriticalSection* выполняет следующие действия.

- если ресурс свободен, EnterCriticalSection модифицирует элементы структуры, указывая, что вызывающая нить занимает ресурс, после чего немедленно возвращает управление, и нить продолжает свою работу (получив доступ к ресурсу);
- если значения элементов структуры свидетельствуют, что ресурс уже захвачен вызывающей нитью, EnterCriticalSection обновляет их, отмечая тем самым, сколько раз подряд эта нить захватила ресурс, и немедленно возвращает управление. Такая ситуация бывает нечасто — лишь тогда, когда нить два раза подряд вызывает EnterCriticalSection без промежуточного вызова LeaveCriticalSection;
- если значения элементов структуры указывают на то, что ресурс занят другой нитью, EnterCriticalSection переводит вызывающую нить в режим ожидания. Нить, пребывая в ожидании, не тратит ни кванта процессорного времени. Система запоминает, что данная нить желает получить доступ к ресурсу, и как только нить, занимавшая этот ресурс, вызывает LeaveCriticalSection — вновь начинает выделять вызывающей нити процессорное время. При этом она передает ему ресурс, автоматически обновляя элементы структуры CRITICAL_SECTION.

Мьютексы создаются функцией

```
CreateMutex(
  __in_opt LPSECURITY_ATTRIBUTES lpMutexAttributes,
  __in     BOOL bInitialOwner,
  __in_opt LPCSTR lpName
  );
```

Параметр bInitialOwner определяет начальное состояние мьютекса. Если в нем передается FALSE (что обычно и бывает), объект-мьютекс не принадлежит ни одной из нитей и поэтому находится в свободном состоянии.

С помощью параметра lpName мьютексу можно задать имя для синхронизации нитей разных процессов.

Когда ожидание мьютекса нитью успешно завершается, последний получает монопольный доступ к защищенному ресурсу. Все остальные нити, пытающиеся обратиться к этому ресурсу, переходят в состояние ожидания. Когда нить, занимающая ресурс, заканчивает с ним работать, она должна освободить мьютекс вызовом функции

BOOL ReleaseMutex(HANDLE hMutex);

Wait-функции.

Wait-функции позволяют нити в любой момент приостановиться и ждать освобождения какого-либо объекта ядра. Из всего семейства этих функций чаще всего используется WaitForSingleObject:

DWORD WaitForSingleObject(HANDLE hObject, DWORD dwMilliseconds);

Когда нить вызывает эту функцию, первый параметр, hObject, идентифицирует объект ядра, поддерживающий состояния «свободен-занят». Второй параметр, dwMilliseconds, указывает, сколько времени (в миллисекундах) нить готова ждать освобождения объекта.

Следующий вызов сообщает системе, что нить будет ждать до тех пор, пока не завершится процесс, идентифицируемый описателем hProcess.

WaitForSingleObject(hProcess, INFINITE);

Мьютексы и критические секции одинаковы в том, как они влияют на планирование ждущих нитей, но различны по некоторым другим характеристикам. Эти объекты сравниваются в таблице 1.

Таблица 1

Характеристики	Объект-мьютекс	Обьект — критическая секция
Быстродействие	Малое	Высокое
Возможность использования за границами процесса	Да	Нет
Объявление	*HANDLE hmfx;*	*CRITICAL_SECTION cs;*
Инициализация	*hmtx =CreateMutex (NULL, FALSE, NULL);*	*InitializeCriticalSection(&cs);*
Очистка	*CloseHandle(hmtx);*	*DeleteCriticalSection(&cs);*
Бесконечное ожидание	*WaitForSingleObject (hmtx, INFINITE);*	*EnterCrittcalSection(&cs);*
Ожидание в течение 0 мс	*WaitForSingleObject (hmtx, 0);*	*TryEnterCriticalSection (&cs);*
Ожидание в течение произвольного периода времени	*WaitForSingleObject (hmtx, dwMilliseconds);*	Невозможно
Освобождение	*ReleaseMutex(hmtx);*	*LeaveCriticalSecliun(&cs);*
Возможность параллельного ожидания других объектов ядра	Да (с помощью *WaitForMultipleObjects* или аналогичной функции)	Нет

СПИСОК ЗАДАНИЙ

1 Создать процесс notepad.exe, открывающий текстовый файл.
2 Создать процесс calc.exe в suspended-режиме.
3 Создать процесс calc.exe на новом десктопе.
4 Создать полноэкранный процесс.
5 Вывести pid созданного процесса.
6 Вывести класс приоритета созданного процесса.
7 Вывести tid главной нити созданного процесса.
8 Создать нить, рекурсивно создающую себя с завершением.
9 Создать нить, рекурсивно создающую себя без завершения.
10 Создать нить, выводящую через каждую секунду свой текуший приоритет.
11 Создать нить в процессе explorer.exe.
12 Создать нить, ожидающую своего завершения.
13 Создать ситуацию взаимоблокировки двух нитей.
14 Создать нить, ожидающую завершения процесса explorer.exe.
15 Синхронизовать с помощью критической секции доступ к глобальному массиву.
16 Синхронизовать с помощью мьютекса доступ к глобальному массиву.
17 Организовать вычисление факториала с помощью 2 нитей.

СПИСОК ЛИТЕРАТУРЫ

1 Руссинович, М. Внутреннее устройство Microsoft Windows Мастер-класс / М. Руссинович, Д. Соломон.– СПб.: Питер, 2005.
2 Дейтел, Х.М. Операционные системы. Ч. 2: Распределенные системы, сети, безопасность / Х.М. Дейтел, П.Дж. Дейтел, Д.Р. Чофнес. – М.: Бином, 2006.
3 Дейтел, Х.М. Операционные системы. Ч. 1: Основы и принципы / Х.М. Дейтел, П.Дж. Дейтел, Д.Р. Чофнес. – М. : Бином, 2006.
4 Гордеев, А.В. Операционные системы : учебник для вузов / А.В. Гордеев. – СПб. : Питер, 2004. – 416 с.
5 Олифер, В.Г. Сетевые операционные системы / В.Г. Олифер, Н.А. Олифер. – СПб. : Питер, 2001. – 544 с.
6 Кастер, Х. Основы Windows NT и NTFS. Русская редакция / Х. Кастер. – М., 1996.
7 Проскурин, В.Г. Защита в операционных системах / В.Г. Проскурин, С.В. Крутов, И.В. Мацкевич. – М. : Радио и связь, 2000.

Рабушко Артур Германович

СОЗДАНИЕ ПРОЦЕССОВ И НИТЕЙ. СИНХРОНИЗАЦИЯ НИТЕЙ

Методические указания
к выполнению лабораторной работы
по дисциплине «Безопасность операционных систем»
для студентов специальностей 090105, 090303, 090900

Редактор О.Д. Постовалова

Подписано к печати | Формат 60×84 1/16 | Бумага тип. №1
Печать трафаретная | Усл. печ.л. 0,75 | Уч.-изд.л.0,5
Заказ | Тираж э/в | Цена свободная

РИЦ Курганского государственного университета.
640669, г. Курган, ул. Гоголя, 25.
Курганский государственный университет.