Федеральное агентство РФ по образованию
Государственное образовательное учреждение
Высшего профессионального образования
Донской государственный технический университет
Кафедра "ПОВТ и АС"

Синхронизация и взаимодействие нескольких процессов в среде Windows

Методические указания к лабораторной работе
по дисциплине «Операционные системы и оболочки»

Ростов-на-Дону
2006 г.

Составители: к.т.н., доц. Долгов В.В.

УДК 512.3

Синхронизация и взаимодействие нескольких процессов в среде Windows: методические указания – Ростов н/Д: Издательский центр ДГТУ, 2006. – 8 с.

В методической разработке рассматриваются способы синхронизации работы нескольких процессов и способы передачи данных между процессами в операционных системах семейства Microsoft Windows. Даны задания по выполнению лабораторной работы. Методические указания предназначены для студентов специальностей 010503 "Математическое обеспечение и администрирование информационных систем".

Ответственный редактор: д.т.н., проф. Нейдорф Р.А.

1. Способы синхронизации работы процессов в среде Windows

К стандартным способам синхронизации работы нескольких процессов можно отнести такие объекты как *критическая секция*, *событие*, *семафор*. Каждый из этих способов обладает определенным функциональным наполнением и различным поведением. Так ***критическая секция*** предназначена для разграничения доступа к некоторому уникальному[1] ресурсу нескольких процессов конкурирующих за этот ресурс. На практике, реализация механизма *критической секции* сводится к разделению во времени моментов выполнения участков различных процессов с помощью системных вызовов «*Вход в критическую секцию*»(Enter) и «Выход из критической секции»(Leave) (рис. 1).

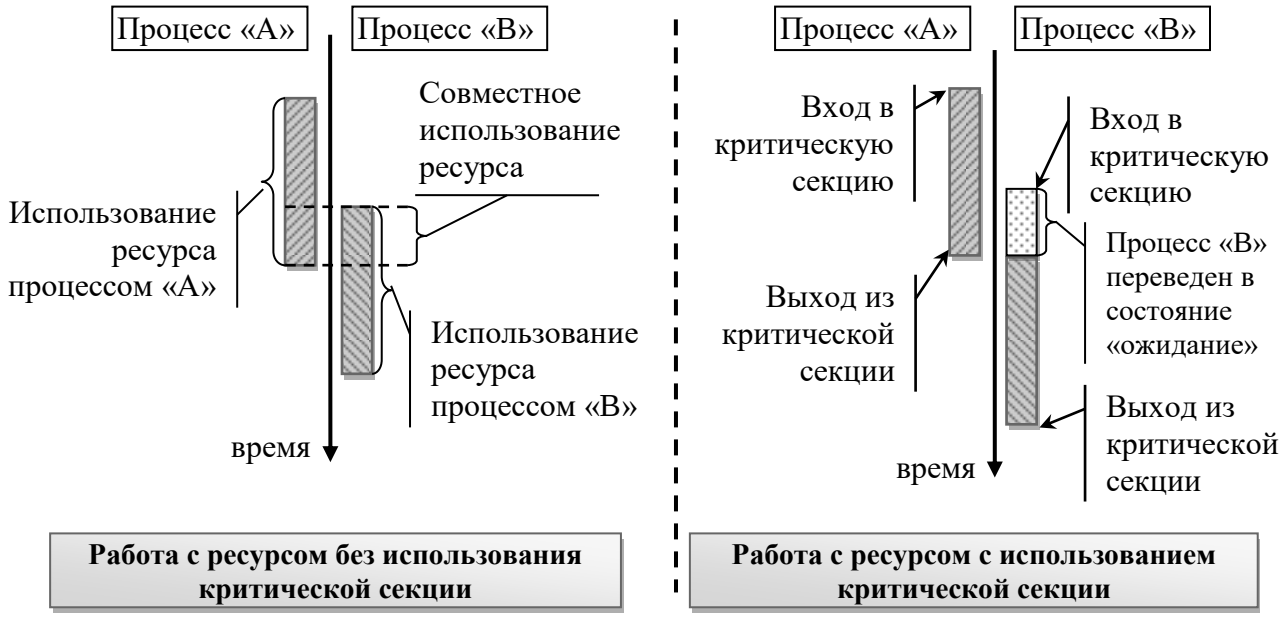

рис. 1. Схема реализации механизма «критической секции»

Событие используется в случаях, когда одному из процессов необходимо сообщить другому(им) процессу(ам) о выполнении некоторого условия (завершение расчета, окончание операции, получение данных и т.д.). При этом ожидающие события процессы будет переведены операционной

[1] Уникальным будем называть такой ресурс, одновременная работа с которым по некоторым причинам может осуществляться только одним процессом (потоком).

системой в состояние «ожидание» и не будут занимать процессорного времени до момента наступления требуемого события.

Семафоры, представляемые неотрицательными целочисленными переменными, используются для контроля доступа процессов к нескольким однотипным уникальным ресурсам, помогая быстро определить существование свободного экземпляра требуемого ресурса (но не сам экземпляр). Семафор является более общим способом синхронизации по отношению к критической секции, поскольку последняя может быть реализована с помощью семафора, максимальное значения счетчика которого установлено в единицу.

В ОС Windows критическая секция реализована в виде объекта «***Mutex***», а событие и семафор в виде соответствующих им по названию объектов «***Event***» и «***Semaphore***». Эти объекты создаются с помощью функций *CreateMutex(…), CreateEvent(…)* и *CreateSemaphore(…)* соответственно, а уничтожаются с помощью функции *CloseHandle(…).* Среди функций непосредственной работы с указанными объектами стоит отдельно выделить функцию *WaitForSingleObject(…)* «ожидающую» каждый из объектов (в общем случае любой объект в ядре ОС, для которого применимо понятие «ждать»). При этом «ожидание» для объекта *Mutex* соответствует операции «Вход в критическую секцию», для объекта *Event* – ожидание наступления события, а для объекта *Semaphore* – ожидание и захват свободой единицы ресурса. Выход из критической секции осуществляется функцией *ReleaseMutex(…),* а освобождение семафора – *ReleaseSemaphore(…).* Поведение события после завершения ожидания зависит от типа события: события с ручным сбросом остаются установленными; с автоматическим – сбрасываются сразу после того, как любой процесс дождется момента его установки.

ЗАДАНИЕ 1.

Реализовать процесс, осуществляющий вывод текстового файла на консоль. Считая консоль уникальным ресурсом, блокировать процесс вывода на неё всего файла с помощью критической секции. Запустить несколько экземпляров процесса и проверить, что единовременно на экран будет выводиться информация только из одного файла.

ЗАДАНИЕ 2.

Реализовать два процесса, один из которых дожидается нажатия клавиши (или кнопки на окне) и сообщает об этом другому процессу с помощью события. После того, как второй процесс дождется события, он должен вывести на экран сто разноцветных геометрических фигур.

ЗАДАНИЕ 3.

На бензозаправке было построено 4 заправочных места. Считая их одинаковыми, подсчитать, какое количество машин вынуждено было проехать мимо, так как все места были заняты. Новая машина подъезжает к заправке через 1-2 мин. Время заправки одной машины – 1-4 мин. Для контроля количества свободных мест на заправке использовать семафор.

2. Средства взаимодействия процессов (IPC)

В ОС Windows/Win32 существует достаточно обширный набор средств для передачи данных между процессами. Выбор конкретного способа взаимодействия должен производиться исходя из потребностей решаемых задач и условий их работы. Самым быстрым способом является ***отображение файлов (File Mapping)*** и его частный случай – ***общая память (Shared Memory)*** реализуемая через механизм виртуальных страниц и функции отображения файлов на память, таких как *CreateFileMapping(...)*,

MapViewOfView(…), UnmapViewOfFile(…). Особенностью данного метода является то, что работа с данными в процессах осуществляется через указатели в адресном пространстве самого процесса, что очень удобно, так как могут использоваться наработанные алгоритмы обработки данных в памяти. Синхронизацию данных с файлами на физическом носителе берет на себя ОС. Однако общую память нельзя использовать, если процессы должны работать на разных машинах в сети, что существенно ограничивает область использования этой технологии.

Именованные каналы (Named Pipes) используются в случае, когда процессам должны взаимодействовать между собой в независимости от того выполняются они на одном компьютере или на разных. Обычно, этот способ взаимодействия используется при реализации взаимодействия вида клиент-сервер. При этом сервер должен создать один или несколько экземпляров канала с именем вида \\.*pipe*\<*имя_канала*> (имя канала должно быть известно клиенту), а клиент – открыть этот канал с помощью функции *CreateFile(…)* используя в качестве имени файла строку \\<*имя_машины_сервера_в_сети*>*pipe*\<*имя_канала*>. В дальнейшем работа с таким каналом происходит с использованием тех же функция, что и работа с файлами (каналы выступают как файлы последовательного доступа).

Похожим на каналы по способу использования средством взаимодействия являются **почтовые слоты** (**Mail Slots**). Отличительной особенностью почтовых слотов служит возможность передачи некоторого (короткого) сообщения сразу на большое количество компьютеров в сети, что при некоторых задачах очень удобно. Однако в отличие от каналов, почтовые слоты являются средством с негарантированной доставкой, то есть программа отправившая пакет с данными не может гарантировать, что этот пакет был принят получателем.

Сокеты используются для создания сетевых приложений, взаимодействующих между собой через различные протоколы передачи данных. Их существенным отличием от именованных каналов является возможность взаимодействия между процессами, выполняемыми не только на разных компьютерах в сети, но и на разных операционных системах или средах.

ЗАДАНИЕ 1.

Написать процесс, осуществляющий копирование файлов (сервер копирования). Имена исходного файла и файла-назначения передаются этому процессу через именованный канал. Реализовать клиент, позволяющий передавать задания на сервер копирования.

ЗАДАНИЕ 2.

Разработать два взаимодействующих приложения, осуществляющих шифрацию/дешифрацию текста методом простой подстановки. Одно приложение (клиент) должно передавать введенные текст (или данные файла) в другое приложение (сервер), а после обработки получать их обратно и при необходимости записывать в файл.

ЗАДАНИЕ 3.

Разработать программу, которая бы играла сама с собой в «Морской бой». Для взаимодействия использовать общую память.

ЗАДАНИЕ 4.

Реализовать ту же программу что и в Задании 3, но для взаимодействия использовать именованные каналы.

ЗАДАНИЕ 5.

Реализовать программу, рисующую в окне прямоугольники с заданными координатами углов и заданным цветом (сервер), получающую заданию на прорисовку от программы-клиента.

Одновременно сервер может обслуживать только одного клиента. Запустить 2 сервера и не менее 5 клиентов. Обеспечить правильное функционирование системы, реализовав синхронизацию посредством семафоров, а передачу информации посредством общей памяти.

ЗАДАНИЕ 6.

Реализовать две программы, одна из которых ведет в общей памяти связный двунаправленный список целых чисел, добавляя и удаляя данные из него случайным образом, а другая, сортирует этот список через каждый 10 секунд и выводит результат сортировки на экран. Целостность данных обеспечить с помощью критической секции.

Литература

1. А. Вильямс «Системное программирование в Windows 2000» - СПб.: Питер, 2001. – 624 с.

Редактор А.А. Литвинова

ЛР № 04779 от 18.05.01.	В набор	В печать
Объем 0,5 усл.п.л., уч.-изд.л.	Офсет.	Формат 60x84/16.
Бумага тип №3.	Заказ №	Тираж 140. Цена

Издательский центр ДГТУ

Адрес университета и полиграфического предприятия:

344010, г. Ростов-на-Дону, пл. Гагарина, 1.