**1.** Серверы по методу облуживания подразделяются на итеративные и параллельные серверы.

Работа итеративного сервера описывается циклом из четырех шагов: 1) ожидание запроса от клиента; 2) обработка запроса; 3) отправка результата запроса; 4) возврат в ждущее состояние 1.

Сервер этого класса может применяться в том случае, если предполагаются короткие запросы от клиентов, не требующие больших затрат на обработку и длинных ответов сервера. Как правило, подобные серверы работают над UDP,

Параллельные серверы имеют другой цикл работы: 1) ожидание запроса от клиента; 2) запуск нового сервера для обработки текущего запроса; 3) возврат в ждущее состояние 1. Преимущество параллельных серверов заключается в том, что он лишь порождает новые серверы, которые и занимаются обработкой запросов клиентов.

Параллельные серверы целесообразно использовать, если предполагается наличие относительно длительного сеанса связи между клиентом и сервером. Как правило, параллельные серверы работают над TCP.

**2.** Структура параллельного сервера, назначением которого является, одновременное обслуживание нескольких клиентских программ.



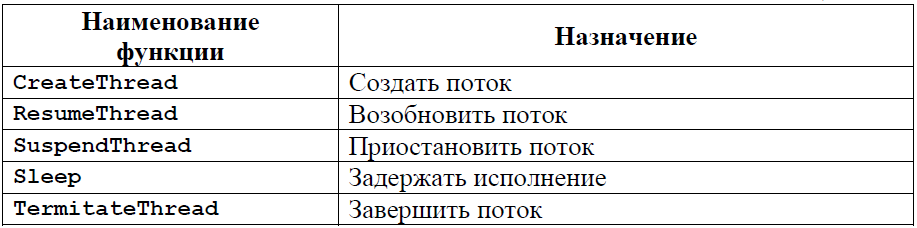
Мы имеем процесс c именем main, является главным процессом сервера, который получает управление от операционной системы. Этот процесс создает и запускает новые процессы AcceptServer, ConsolePipe и GarbageCleaner (основным назначением процесса GarbageCleaner является удаление элемента списка подключений ListContact, после отключения программы клиента.). Процесс AcceptServer, в свою очередь создает

несколько процессов с именем EchoServer (Кроме того, AcceptServer создает список подключений, который далее будем называть ***ListContanct***. При подключении очередного клиента, процесс AcceptServer добавляет в ListContact элемент, предназначенный для хранения информации о состоянии данного подключения).

Сплошными двунаправленными линиями обозначается перемещение данных. Данные перемещаются между EchoServer-процессами сервера и клиентскими программами (с именем Client), обозначенными на рисунке овалами, а также между программой с именем RConsole, реализующей клиентскую часть консоли управления, и процессом ConsolePipe, который реализует серверную часть консоли управления. Штриховой линией, соединяющей изображение клиентских программ и изображение процесса AcceptServer, обозначается процедура создания соединения между клиентом и сервером.

**3.** Понятие потока тесно связано с последовательностью действий процессора во время выполнения программы. Исполняя программу, процессор последовательно выполняет инструкции (машинные коды) программы, иногда осуществляя переходы. Такая последовательность выполнения инструкций называется потоком управления.

**4.** Каждое приложение, работающее в среде Windows, имеет, по крайней мере, один поток, который называется первичным или главным потоком. В консольных приложениях этот поток исполняет функцию main.

****

**5.** Для организации параллельной работы программ в операционной системе Windows предусмотрены два специальных механизма: механизм потоков и механизм процессов.

**6.** Будем говорить, что программа является многопоточной, если в ней существуют одновременно несколько потоков управления. Сами потоки в этом случае называются параллельными. Если в программе может существовать только один поток, то такая программа называется однопоточной.

**7.** Безопасная для потоков функции обладает двумя свойствами: 1) свойством ***реентерабельности***; 2) функция обеспечивает 108 блокировку доступа к критическим ресурсам, которые она использует.

В общем случае функция называется реентерабельной, если она не изменяет собственный код или собственные статические данные.

Блокировка требуется в том случае, если функцией используется ресурс, доступ к которому может быть только упорядоченным.

**8.** Каждому потоку принадлежат следующие ресурсы:

- код исполняемой функции;

- набор регистров процессора;

- область оперативной памяти;

- стек для работы приложения;

- стек для работы операционной системы;

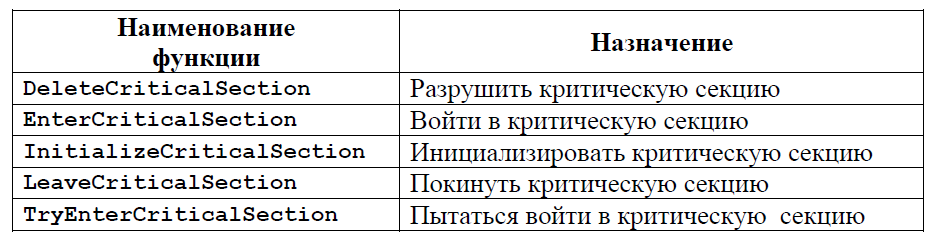
- маркер доступа, содержащий информацию для системы безопасности.

**9.** Под процессом операционной системы Window понимается объект ядра, которому принадлежат системные ресурсы, используемые исполняемым приложением операционной системы. Выполнение процесса начинается с первичного потока main. Во время выполнения, процесс может создавать новые потоки и порождать новые процессы.

**10.** Синхронизация будет осуществляться между двумя потоками: AcceptServer и GarbageCleaner. Синхронизация необходима в связи с тем, что одновременное добавление элемента в список ListContact, выполняемое потоком AcceptServer, и удаление элемента списка, выполняемое потоком GarbageCleaner может привести к непредсказуемым последствиям.

**11.** Критическая секция — это участок кода в многопоточной программе, который может быть выполнен только одним потоком в определенный момент времени. Это необходимо для того, чтобы избежать конфликтов при доступе к общим ресурсам, таким как переменные, файлы или устройства ввода-вывода.

Критические секции обычно реализуются с помощью мьютексов, семафоров или других механизмов синхронизации.

****

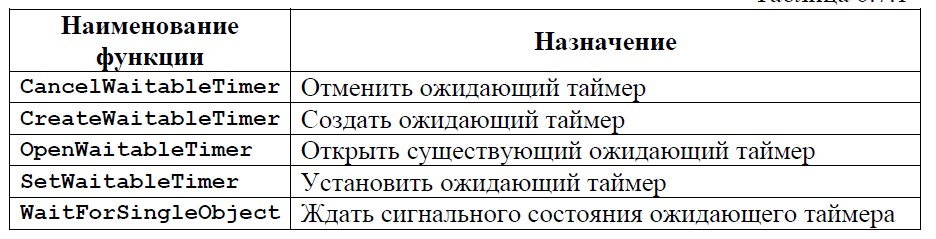
**12.** Асинхронной процедурой называется функция, которая выполняется асинхронно в контексте какого-нибудь потока. Для исполнения асинхронной процедуры необходимо определить асинхронную процедуру, указать поток, в контексте которого она будет выполняться, и дать разрешение на выполнение асинхронной процедуры.

**13, 14, 15.** Ожидающим таймером в Windows, называется объект синхронизации, который переходит в сигнальное состояние при наступлении заданного момента времени. Если ожидающий таймер ждет момента перехода в сигнальное состояние, то говорят, что он находится в активном состоянии. Другое состояние ожидающего таймера пассивное – из этого состояния он не может перейти в сигнальное состояние.

**16.** Таймер с ручным и автоматическим сбросом — это таймер, который может быть сброшен вручную или автоматически.

Ручной сброс означает, что пользователь может в любой момент сбросить таймер, нажав на соответствующую кнопку или выполнив другое действие.

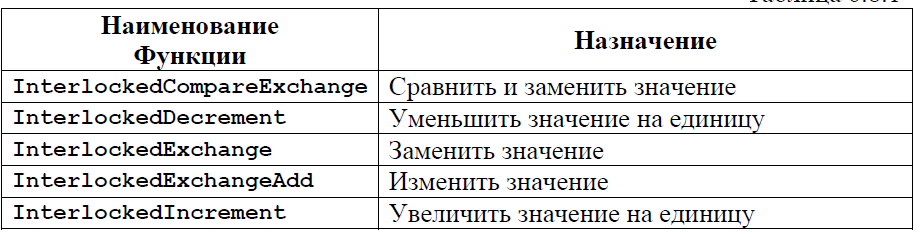
Автоматический сброс означает, что таймер сбрасывается автоматически после достижения определенного времени.



**17.** Периодическое состояние — это состояние системы, которое повторяется через определенные промежутки времени.

Непериодическое состояние - это состояние системы, которое не повторяется через определенные промежутки времени.

**18.** Блокирующие функции выполняют несколько элементарных операций, которые объединяются в одну неделимую операцию, называемую атомарной операцией.

****

**19**. Режим блокировки — это режим работы программы, в котором выполнение одной функции блокирует выполнение других функций до ее завершения. В этом режиме программы могут быть заблокированы на неопределенное время, если функция, которая должна быть выполнена, занимает много времени.

Режим без блокировки - это режим работы программы, в котором выполнение одной функции не блокирует выполнение других функций. В этом режиме программы могут продолжать работать, даже если одна из функций занимает много времени.

Некоторые блокирующие функции могут быть:

1. Функции ввода-вывода, которые блокируют выполнение программы до завершения операции ввода-вывода.
2. Функции сетевого взаимодействия, которые блокируют выполнение программы до завершения операции сетевого взаимодействия.
3. Функции синхронизации, которые блокируют выполнение программы до завершения операции синхронизации.
4. Функции, которые работают с ресурсами, доступ к которым может быть заблокирован другими процессами или потоками.

**20.** Приоритеты потоков и процессов определяют, какой поток или процесс будет выполнен в первую очередь, если в системе есть несколько потоков или процессов, готовых к выполнению. Обычно, чем выше приоритет, тем больше шансов, что поток или процесс будет выполнен в первую очередь.

Приоритеты процессов могут быть следующими:

1. Realtime (Реального времени) - наивысший приоритет, используется для задач, которые должны быть выполнены немедленно и без задержек.
2. High (Высокий) - высокий приоритет, используется для задач, которые должны быть выполнены быстро, но не обязательно немедленно.
3. Normal (Нормальный) - стандартный приоритет, используется для большинства задач.
4. Low (Низкий) - низкий приоритет, используется для задач, которые могут быть выполнены в фоновом режиме.

Функции приоритетов могут быть следующими:

1. SetPriorityClass - устанавливает приоритет процесса.
2. GetPriorityClass - получает текущий приоритет процесса.
3. SetThreadPriority - устанавливает приоритет потока.
4. GetThreadPriority - получает текущий приоритет потока.

Функции потоков могут быть следующими:

1. CreateThread - создает новый поток.
2. ExitThread - завершает текущий поток.
3. SuspendThread - приостанавливает выполнение потока.
4. ResumeThread - возобновляет выполнение приостановленного потока.

**21.** Задача оповещения - это задача, которая заключается в том, чтобы уведомлять программу или пользователя о событиях, происходящих в системе. Например, если программа ожидает завершения длительной операции, задача оповещения может уведомить программу, когда операция завершится, чтобы программа могла продолжить свою работу.

Условная процедура - это процедура, которая выполняется только при выполнении определенного условия. Например, если программа должна выполнить различные действия в зависимости от значения переменной, она может использовать условную процедуру, чтобы выполнить нужные действия в зависимости от значения переменной.

Задача оповещения и условные процедуры могут быть использованы вместе для решения различных задач. Например, если программа ожидает завершения длительной операции, она может использовать задачу оповещения для уведомления о завершении операции и условную процедуру для выполнения нужных действий в зависимости от результата операции.

**22.** Динамические библиотеки - это файлы, содержащие код, который может быть загружен в память программы во время ее выполнения. Они представляют собой набор функций и процедур, которые могут быть использованы программой для выполнения различных задач.

Динамические библиотеки используются для уменьшения размера исполняемого файла программы и повышения ее модульности. Вместо того, чтобы включать весь код в исполняемый файл, программа может ссылаться на динамические библиотеки во время выполнения, загружая только те функции, которые необходимы для выполнения задачи.

Динамические библиотеки могут быть общими для нескольких программ или использоваться только одной программой. Они могут быть написаны на разных языках программирования и могут быть доступны для использования в различных операционных системах.

Примеры динамических библиотек в операционной системе Windows: DLL (Dynamic Link Library), а в операционной системе Linux: .so (Shared Object).

**23.** Функция LoadLibrary применяется для загрузки динамической библиотеки в память компьютера. Для выгрузки библиотеки используется функция FreeLibrary.

**24.**

**25.** Освобождение ожидающих потоков может быть выполнено несколькими способами, в зависимости от того, каким образом потоки были заблокированы:

Использование мьютексов - если потоки были заблокированы на мьютексе, можно освободить их, вызвав функцию

ReleaseMutex()

, которая разблокирует мьютекс и позволит другим потокам получить доступ к защищенному ресурсу.

Использование семафоров - если потоки были заблокированы на семафоре, можно освободить их, вызвав функцию

ReleaseSemaphore()

, которая увеличивает значение семафора и позволяет другим потокам получить доступ к защищенному ресурсу.

Использование условных переменных - если потоки были заблокированы на условной переменной, можно освободить их, вызвав функцию

pthread\_cond\_signal()

(для POSIX-систем) или

WakeConditionVariable()

(для Windows), которая уведомляет ожидающие потоки о том, что условие, на котором они были заблокированы, было выполнено.

Использование прерывания потоков - если потоки были заблокированы на операции ввода-вывода или других блокирующих операциях, можно прервать их, вызвав функцию

pthread\_cancel()

(для POSIX-систем) или

TerminateThread()

(для Windows), которая прерывает выполнение потока и освобождает его.

Важно помнить, что освобождение ожидающих потоков должно быть выполнено осторожно, чтобы избежать возможных проблем с синхронизацией и доступом к защищенным ресурсам.

26. группа администраторов и группа пользователей.