Abstract:

This program is a Winsock echo server program that uses I/O Completion Ports (IOCP) to receive data from and echo data back to a sending client. The server program supports multiple clients connecting via TCP/IP and sending arbitrary sized data buffers which the server then echoes back to the client. For convenience a simple client program, iocpclient was developed to connect and continually send data to the server to stress it. Direct IOCP support was added to Winsock 2 and is fully implemented on the NT platform. IOCPs provide a model for developing very high performance and very scalable server programs.

The basic idea is that this server continuously accepts connection requests from a client program. When this happens, the accepted socket descriptor is added to the existing IOCP and an initial receive (WSARecv) is posted on that socket. When the client then sends data on that socket, a completion packet will be delivered and handled by one of the server's worker threads. The worker thread echoes the data back to the sender by posting a send (WSASend) containing all the data just received. When sending the data back to the client completes, another completion packet will be delivered and again handled by one of the server's worker threads.

Assuming all the data that needed to be sent was actually sent, another receive (WSARecv) is once again posted and the scenario repeats itself until the client stops sending data. When using IOCPs it is important to remember that the worker threads must be able to distinguish between I/O that occurs on multiple handles in the IOCP as well as multiple I/O requests initiated on a single handle. The per handle data (PER\_SOCKET\_CONTEXT) is associated with the handle as the CompletionKey when the handle is added to the IOCP using CreateIoCompletionPort. The per IO operation data (PER\_IO\_CONTEXT) is associated with a specific handle during an I/O operation as part of the overlapped structure passed to either WSARecv or WSASend. Please notice that the first member of the PER\_IO\_CONTEXT structure is a WSAOVERLAPPED structure (compatible with the Win32 OVERLAPPED structure).

When the worker thread unblocks from GetQueuedCompletionStatus, the key associated with the handle when the handle was added to the IOCP is returned as well as the overlapped structure associated when this particular I/O operation was initiated.

This program cleans up all resources and shuts down when CTRL-C is pressed. This will cause the main thread to break out of the accept loop and close all open sockets and free all context data. The worker threads get unblocked by posting special I/O packets with a NULL CompletionKey to the IOCP. The worker threads check for a NULL CompletionKey and exits if it encounters one. If CTRL-BRK is pressed instead, cleanup process is same as above but instead of exit the process, the program loops back to restart the server.

Another point worth noting is that the Win32 API CreateThread() does not initialize the C Runtime and therefore, C runtime functions such as printf() have been avoid or rewritten (see myprintf()) to use just Win32 APIs.

Usage:

Start the server and wait for connections on port 6001

iocpserver -e:6001

Build:

Use the headers and libs from the April98 Platform SDK or later.

Link with ws2\_32.lib

===================================================================================

Аннотация: Эта программа представляет собой серверную программу Winsock, которая использует порты завершения ввода-вывода (IOCP) для получения данных и их передачи обратно клиенту-отправителю. Серверная программа поддерживает подключение нескольких клиентов по протоколу TCP/IP и отправку буферов данных произвольного размера, которые сервер затем передаёт обратно клиенту. Для удобства была разработана простая клиентская программа iocpclient, которая подключается к серверу и постоянно отправляет ему данные для его нагрузочного тестирования. Поддержка прямого IOCP была добавлена в Winsock 2 и полностью реализована на платформе NT. IOCP предоставляют модель для разработки очень производительных и масштабируемых серверных программ. Основная идея заключается в том, что этот сервер непрерывно принимает запросы на подключение от клиентской программы. Когда это происходит, дескриптор принятого сокета добавляется в существующий IOCP, и на этот сокет отправляется запрос на получение (WSARecv). Когда клиент отправляет данные на этот сокет, один из рабочих потоков сервера получает и обрабатывает пакет завершения. Рабочий поток отправляет данные обратно отправителю, отправляя пакет (WSASend), содержащий все только что полученные данные. Когда отправка данных обратно клиенту завершается, доставляется ещё один пакет завершения, который снова обрабатывается одним из рабочих потоков сервера. Если все данные, которые нужно было отправить, были отправлены, отправляется ещё один пакет получения (WSARecv), и сценарий повторяется до тех пор, пока клиент не перестанет отправлять данные. При использовании IOCP важно помнить, что рабочие потоки должны уметь различать ввод-вывод, выполняемый с несколькими дескрипторами в IOCP, а также несколько запросов ввода-вывода, инициированных с одним дескриптором. Данные для каждого дескриптора (PER\_SOCKET\_CONTEXT) связываются с дескриптором в качестве ключа завершения при добавлении дескриптора в IOCP с помощью CreateIoCompletionPort. Данные для каждой операции ввода-вывода (PER\_IO\_CONTEXT) связаны с конкретным дескриптором во время операции ввода-вывода как часть перекрывающейся структуры, передаваемой в WSARecv или WSASend. Обратите внимание, что первым элементом структуры PER\_IO\_CONTEXT является структура WSAOVERLAPPED (совместимая со структурой OVERLAPPED в Win32). Когда рабочий поток выходит из GetQueuedCompletionStatus, возвращается ключ, связанный с дескриптором, когда дескриптор был добавлен в IOCP, а также перекрытая структура, связанная с этой конкретной операцией ввода-вывода. Эта программа очищает все ресурсы и завершает работу при нажатии CTRL-C. Это приведет к выходу основного потока из цикла приема и закрытию всех открытых сокетов, а также освобождению всех контекстных данных. Рабочие потоки разблокируются путем отправки в IOCP специальных пакетов ввода-вывода с нулевым ключом завершения. Рабочие потоки проверяют наличие нулевого ключа завершения и завершают работу, если обнаруживают его. Если вместо этого нажать CTRL-BRK, процесс очистки будет таким же, как описано выше, но вместо завершения процесса программа вернется к перезапуску сервера. Следует также отметить, что Win32 API CreateThread() не инициализирует среду выполнения C, поэтому функции среды выполнения C, такие как printf(), были исключены или переписаны (см. myprintf()) для использования только Win32 API. Использование: Запустите сервер и дождитесь подключений на порту 6001 iocpserver -e:6001 Сборка: Используйте заголовки и библиотеки из April98 Platform SDK или более поздней версии. Ссылка на файл ws2\_32.lib