Лабораторная работа: Отладка параллельных MPI программ в среде Microsoft Visual Studio 2005

Лабораторная работа: Отладка параллельных MPI программ в среде Microsoft Visual Studio 2005 1
Цель лабораторной работы 2
Обзор методов отладки параллельных программ в среде Microsoft Visual Studio 2005 2
Упражнение 1 – Тестовый локальный запуск параллельных программ 3
Задание 1 – Локальный запуск параллельной программы вычисления числа Пи на рабочей станции 3
Задание 2 – Настройка Microsoft Visual Studio 2005 для локального запуска параллельной MPI программы в режиме отладки 5
Задание 3 – Запуск параллельной MPI программы в режиме отладки из Microsoft Visual Studio 2005 7
Упражнение 2 – Отладка параллельной программы в Microsoft Visual Studio 2005 8
Задание 1 – Знакомство с основными методами отладки 8
Задание 2 – Использование точек остановки 15
Задание 3 – Особенности отладки параллельных MPI программ 19
Задание 4 – Обзор типичных ошибок при написании параллельных MPI программ 20
Контрольные вопросы 21

Одним из важнейших этапов разработки программ является процесс поиска и устранения ошибок, неизбежно возникающих при написании сложных программных систем. Для того, чтобы упростить и во многом автоматизировать этот процесс, необходимо использовать специальные программы, называемые *отладчиками*. Несмотря на то, что сложность написания параллельных программ часто существенно превосходит сложность написания последовательных аналогов, длительное время наблюдалась нехватка качественных отладчиков для параллельных программ, разрабатываемых с использованием технологии MPI. Новейшая интегрированная среда разработки Microsoft Visual Studio 2005 имеет в своем составе отладчик, предоставляющий широкие возможности для поиска и устранения ошибок, в том числе и для отладки MPI приложений.

Цель лабораторной работы

Цель данной лабораторной работы – получение практических навыков отладки параллельных MPI программ в среде **Microsoft Visual Studio 2005**. При этом будет предполагаться, что на рабочей станций станции установлен Compute Cluster Server SDK, и, следовательно, используется реализация MPI от компании Microsoft (библиотека MS MPI):

- Упражнение 1 – Тестовый локальный запуск параллельных программ,
- Упражнение 2 – Отладка параллельной программы в Microsoft Visual Studio 2005.

Примерное время выполнения лабораторной работы: **90 минут**.

Обзор методов отладки параллельных программ в среде Microsoft Visual Studio 2005

Отладка (поиск и устранение ошибок) – один из важнейших этапов в написании программных систем, часто занимающий у разработчика больше времени, чем написание отлаживаемого кода. Для того, чтобы максимально снизить время, затрачиваемое на отладку, необходимо придерживаться рекомендаций крупнейших производителей программного обеспечения и ведущих исследователей в области компьютерных наук еще на этапах проектирования и программирования. Например, подобные рекомендации обычно требуют подготавливать автоматические тесты для всех участков разрабатываемой системы и контролировать корректность значений внутренних переменных с использованием макроса **ASSERT**. Но и сама процедура отладки должна проводиться эффективно, в соответствии с рекомендациями ведущих экспертов в этой области. Огромное значение здесь имеет правильный выбор **отладчика** – специальной программы, существенно упрощающий процесс поиска ошибок, позволяющий выполнять программу в пошаговом режиме, отслеживать значения переменных, устанавливать точки остановки и т.д. Важнейшими критериями выбора отладчика являются богатство предоставляемого программисту инструментария и удобство использования. В ходе выполнения данной лабораторной работы мы будем использовать стандартный отладчик среды Microsoft Visual Studio 2005, удачно сочетающий в себе интуитивно понятный пользовательский интерфейс и широкий спектр предоставляемых возможностей.

Наиболее частым приемом отладки является приостановка работы программы в заданный момент времени и анализ значений ее переменных. Момент, в который программа будет приостановлена, определяется выбором, так называемых, точек остановки, то есть указанием строк кода исходной программы, по достижении которых выполнение останавливается до получения соответствующей команды пользователя. Помимо простого указания строк кода, где произойдет приостановка, возможно также указание условий, которые должны при этом выполняться. Например, можно дать указание приостановить выполнение программы на заданной строке только в том случае, если значение некоторой переменной программы превысило заданную константу. Правильный выбор момента остановки имеет решающее значение для успеха отладки. Так, зачастую анализ значений переменных непосредственно перед моментом падения программы дает достаточно информации для определения причин некорректной работы.

Другим исключительно полезным инструментом является возможность выполнять программу в пошаговом режиме – по одной строке исходного кода отлаживаемой программы при каждом нажатии пользователем кнопки **F10**. При этом пользователь также имеет возможность заходить внутрь функции, вызов которой происходит на данной строке, или просто переходить к следующей строке. Таким образом, пользователь всегда находится на том уровне детализации, который ему необходим.

Для того, чтобы в полной мере оценить преимущества отладчика, необходимо предварительно скомпилировать программу в специальной отладочной конфигурации (чаще всего такая конфигурация называется “**Debug**”), особенностью которой является добавление в генерируемые бинарные файлы специальной отладочной информации, которая позволяет видеть при отладке исходный код выполняемый программы на языке высокого уровня.

К числу других часто используемых инструментов отладки Microsoft Visual Studio 2005 относятся:

- Окно "**Call Stack**" - окно показывает текущий стек вызова функций и позволяет переключать контекст на каждую из функций стека (при переключении контекста программист получает доступ к значениям локальных переменных функции),
- Окно "**Autos**" - окно показывает значения переменных, используемых на текущей и на предыдущих строках кода. Кроме того, это окно может показывать значения, возвращаемые вызываемыми функциями. Список отображаемых значений определяется средой автоматически,
- Окно "**Watch**" - окно позволяет отслеживать значения тех переменных, которых нет в окне "**Autos**". Список отслеживаемых переменных определяется пользователем,
- Окно "**Threads**" - окно позволяет переключаться между различными потоками команд процесса,
- Окно "**Processes**" - окно позволяет переключаться между различными отлаживаемыми процессами (например, в случае отладки MPI – программы).

Отладка параллельных MPI программ имеет ряд особенностей, обусловленных природой программирования для кластерных систем. Напомним, что в параллельной программе над решением задачи работают одновременно несколько процессов, каждый из которых, в свою очередь, может иметь несколько потоков команд. Это обстоятельство существенно усложняет отладку, так как помимо ошибок, типичных для последовательного программирования, появляются ошибки, совершаемые только при разработке параллельных программ. К числу таких ошибок можно отнести, например, сложно контролируемую ситуацию *гонки процессов*, когда процессы параллельной программы взаимодействуют между собой без выполнения каких-либо синхронизирующих действий. В этом случае, в зависимости от состояния вычислительной системы, последовательность выполняемых действий может различаться от запуска к запуску параллельной программы. Как результат, при наличии гонки процессов сложным становится применение одного из основных принципов отладки – проверка работоспособности при помощи тестов (однократное выполнение теста для параллельной программы может не выявить ситуации гонки процессов).

Однако инструменты и приемы, используемые в Microsoft Visual Studio 2005 для отладки как последовательных, так и параллельных программ схожи, поэтому, если Вы имеете хороший опыт отладки последовательных программ, то и отладка параллельных программ не покажется Вам слишком сложной.

Упражнение 1 – Тестовый локальный запуск параллельных программ

Перед тем, как запускать скомпилированную программу на многопроцессорной вычислительной системе (кластере), желательно провести несколько **локальных экспериментов** на обычном персональном компьютере(в случае использования однопроцессорного компьютера все процессы параллельной программы запустятся в режиме разделения времени одного имеющегося процессора). Такой способ выполнения параллельных программ является, как правило, более оперативным (не требуется удаленный доступ к кластеру, и отсутствует время ожидания запуска программы в очереди заданий). Кроме того, такие эксперименты выполняются обычно для более простых постановок решаемых задач при небольших размерах исходных данных. Все это позволяет достаточно быстро устранить большое количество имеющихся в параллельной программе ошибок.

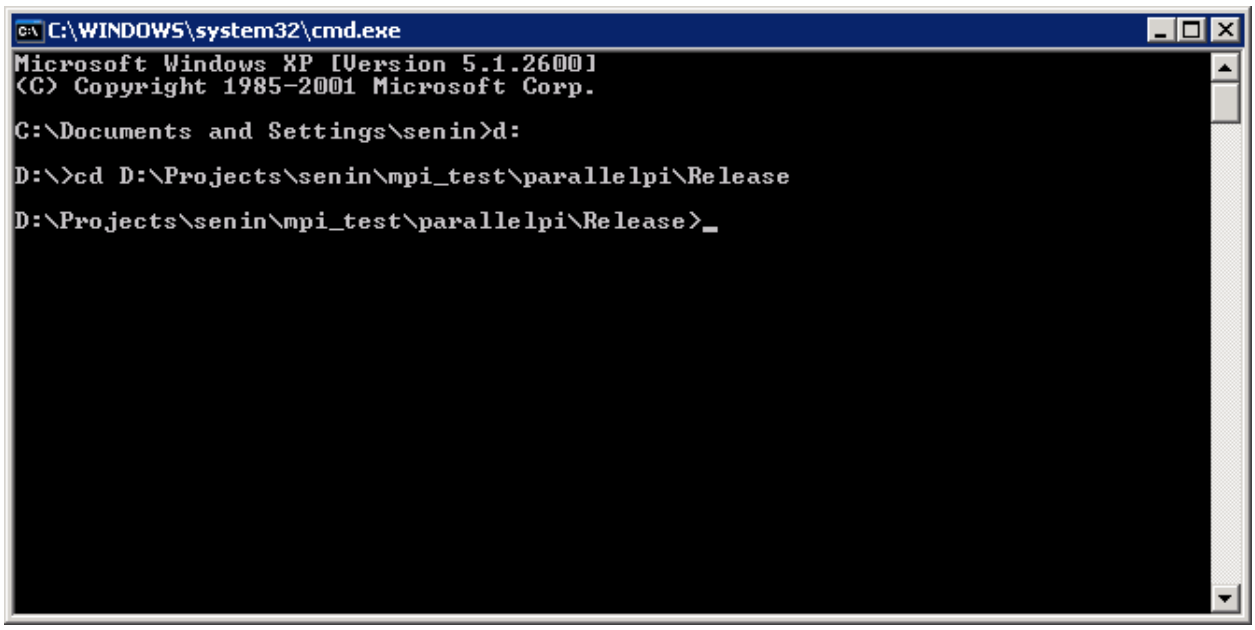
Для выполнения локальных экспериментов при установке клиентской части Microsoft Compute Cluster Pack на рабочей станции необходимо установить также Microsoft Compute Cluster Server SDK.

В данном упражнении мы познакомимся с заданием необходимых параметров среды Microsoft Visual Studio 2005 для отладки параллельных MPI программ и запустим программу в режиме отладки.

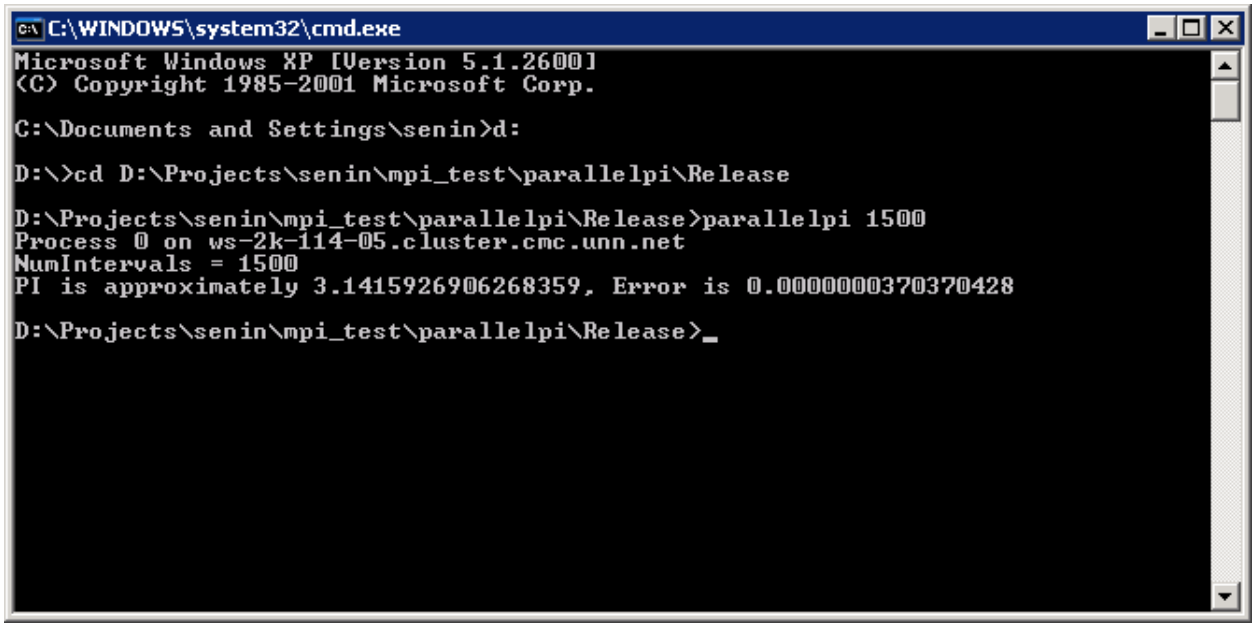
Задание 1 – Локальный запуск параллельной программы вычисления числа Пи на рабочей станции

- Скомпилируйте проект параллельного вычисления числа Пи (**parallelpi**) в соответствии с инструкциями лабораторной работы "Выполнение заданий под управлением Microsoft Compute Cluster Server 2003" в конфигурации **Release**,
- Откройте командный интерпретатор ("**Start->Run**", введите команду "**cmd**" и нажмите клавишу "**Ввод**"),

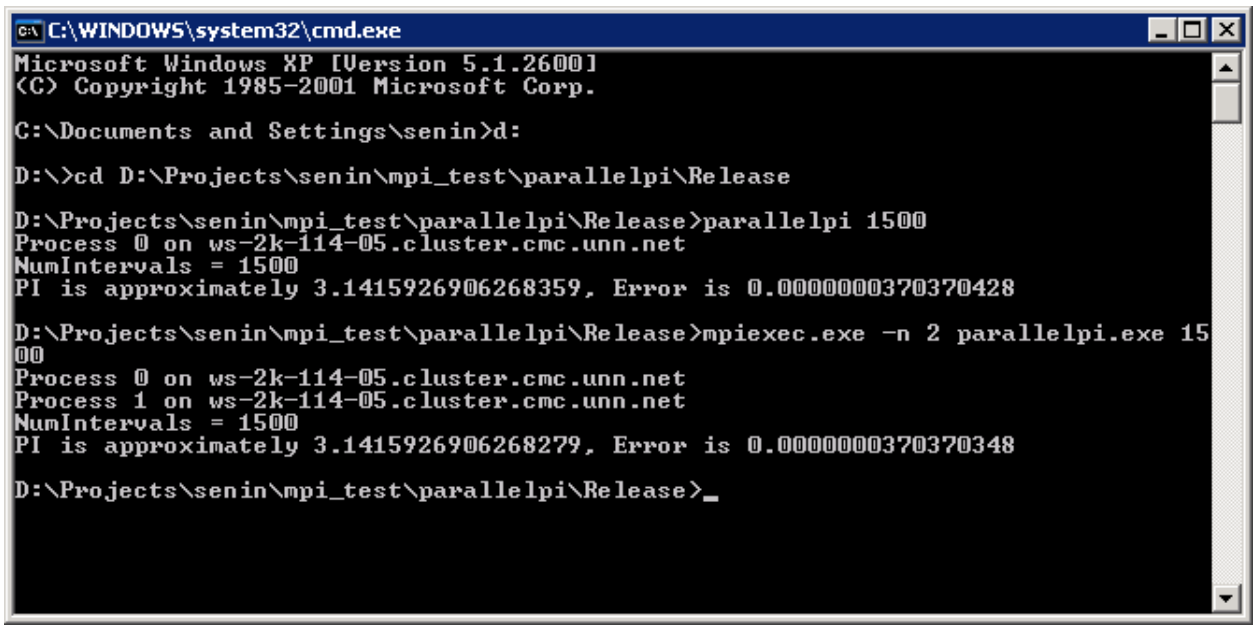
- Перейдите в папку, содержащую скомпилированную программу (например, для перехода в папку "**D:\Projects\senin\mpi_test\parallelpi\Release**" введите команду смены диска "**d:**", а затем перейдите в нужную папку, выполнив команду "**cd D:\Projects\senin\mpi_test\parallelpi\Release**"),



- Для запуска программы в последовательном режиме (1 процесс) достаточно просто ввести имя программы и аргументы командной строки. Например, “**parallelpi.exe 1500**”,



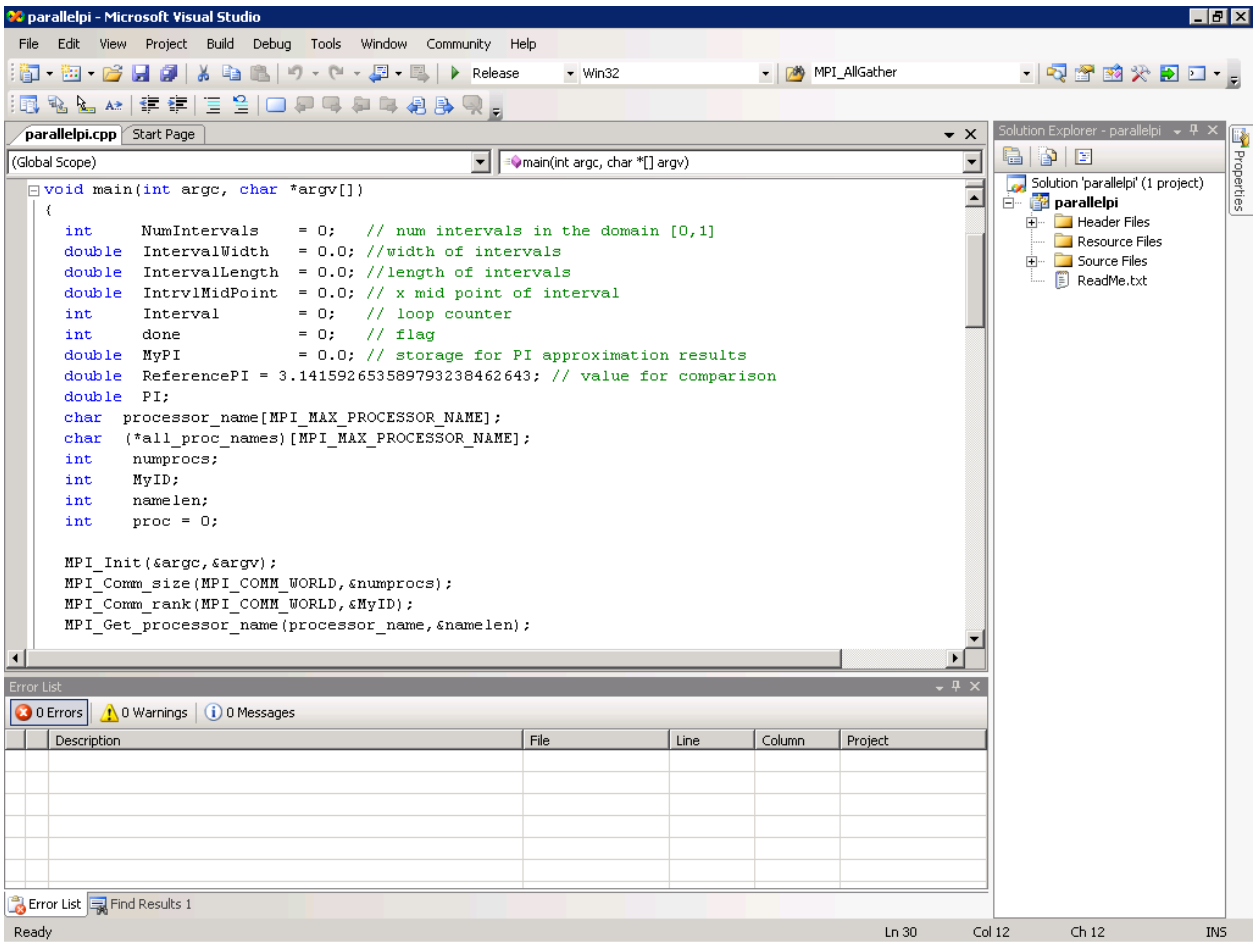
- Для тестового запуска программы в параллельном режиме на локальном компьютере введите “**mpiexec.exe –n <число процессов> parallelpi.exe <параметры командной строки>**”. В нашем случае параметром командной строки программы вычисления числа Пи является число интервалов разбиения при вычислении определенного интеграла.



Задание 2 – Настройка Microsoft Visual Studio 2005 для локального запуска параллельной MPI программы в режиме отладки

Для того чтобы получить возможность отлаживать параллельные MPI программы, необходимо соответствующим образом настроить **Microsoft Visual Studio 2005**:

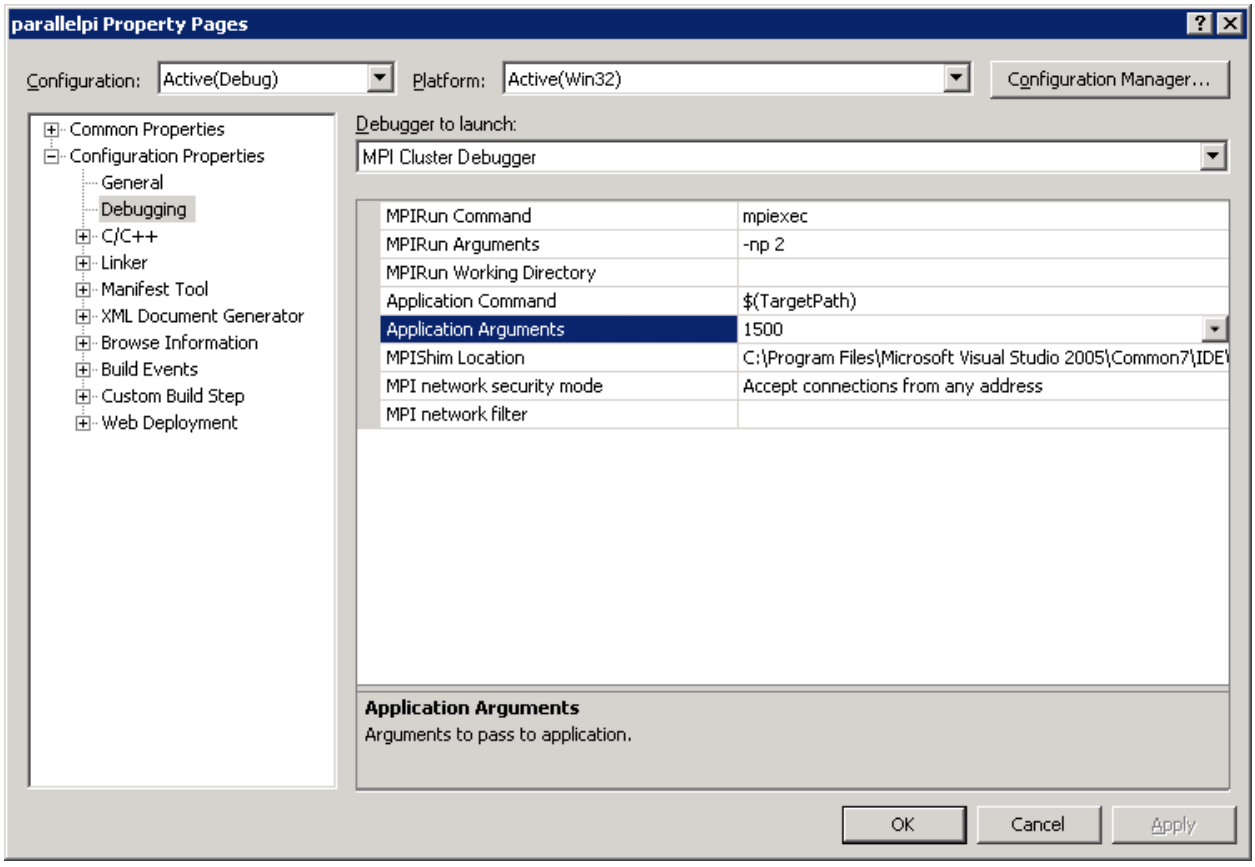
- Откройте проект параллельного вычисления числа Пи (**parallelpi**) в Microsoft Visual Studio 2005,



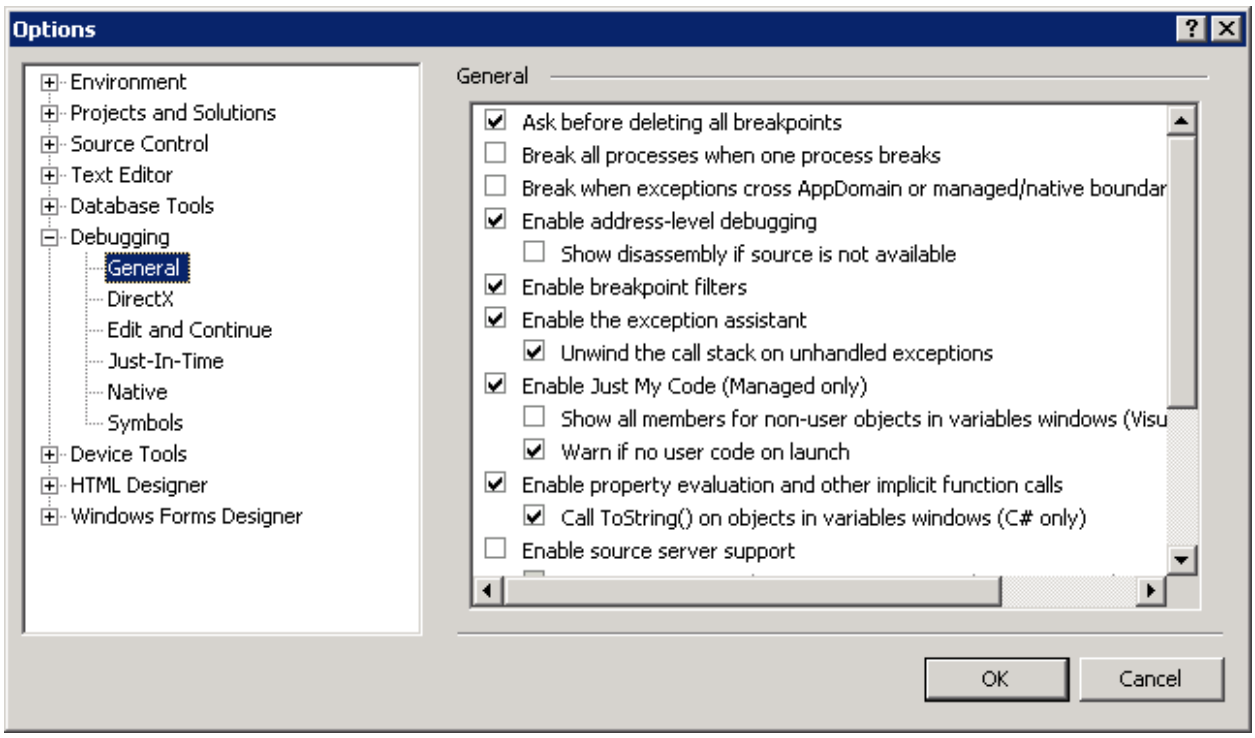
- Выберите пункт меню “**Project->paralellpi Properties…**”. В открывшемся окне настроек проекта выберите пункт “**Configuration Properties->Debugging**”. Введите следующие настройки:
  - В поле “**Debugger to launch**” выберите “**MPI Cluster Debugger**”,

- В поле “**MPIRun Command**” введите “**mpiexec.exe**” – имя программы, используемой для запуска параллельной MPI программы,
- В поле “**MPIRun Argument**” введите аргументы программы “**mpiexec.exe**”. Например, введите “**-np <число процессов>**” для указания числа процессов, которые будут открыты,
- В поле “**Application Command**” введите путь до исполняемого файла программы,
- В поле “**Application Argument**” введите аргументы командной строки запускаемой программы,
- В поле “**MPIShim Location**” введите путь до “**mpishim.exe**” – специальной программы, поставляемой вместе с Microsoft Visual Studio 2005, используемой для отладки удаленных программ,

- Нажмите “**OK**” для сохранения внесенных изменений,

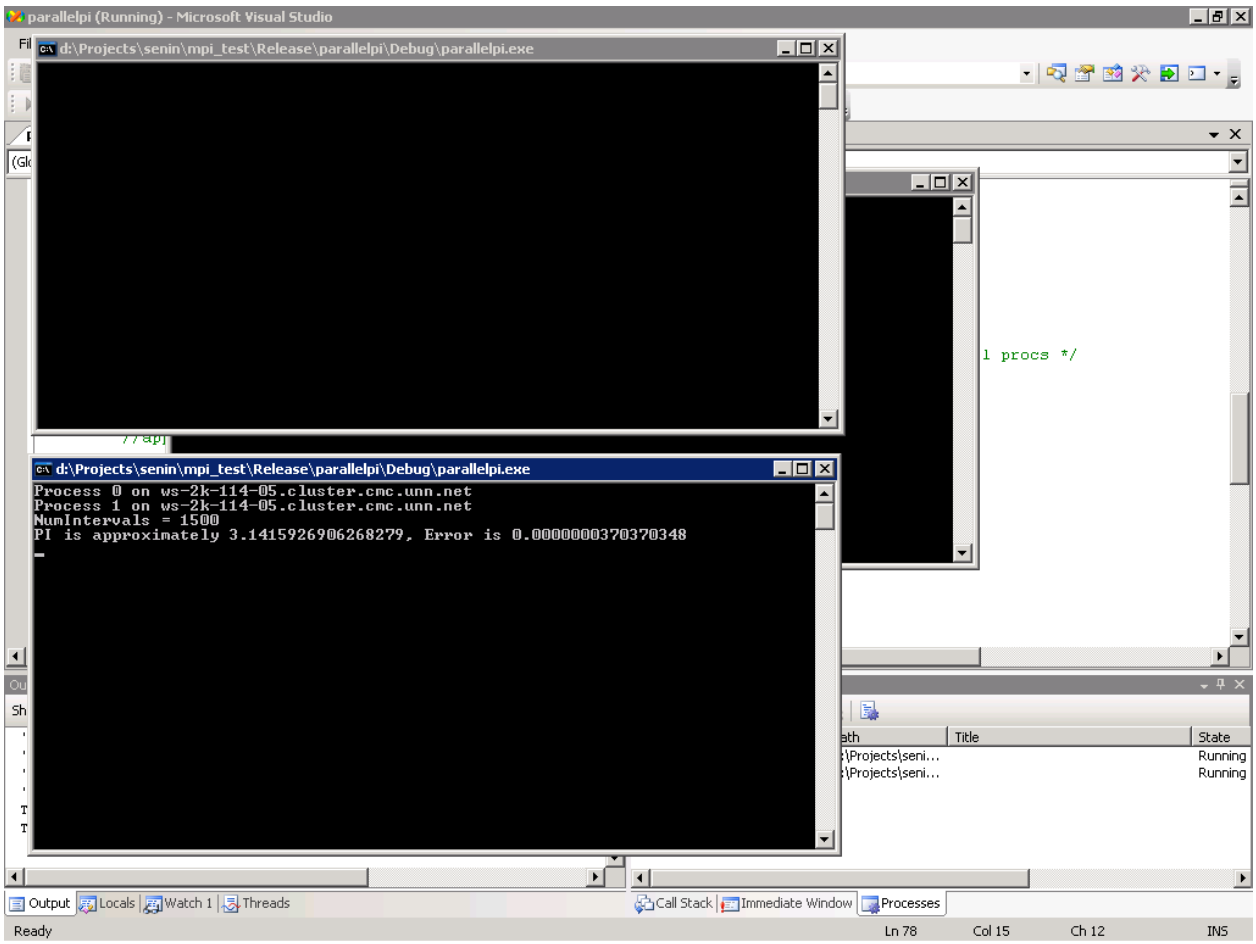


- Выберите пункт меню “**Tools->Options**”. В открывшемся окне настроек проекта выберите пункт “**Debugging->General**”. Поставьте флаг около пункта “**Break all processes when one process breaks**” для остановки всех процессов параллельной программы в случае, если один из процессов будет остановлен. Если Вы хотите, чтобы при остановке одного процесса остальные продолжали свою работу, снимите флаг (рекомендуется).



Задание 3 – Запуск параллельной MPI программы в режиме отладки из Microsoft Visual Studio 2005

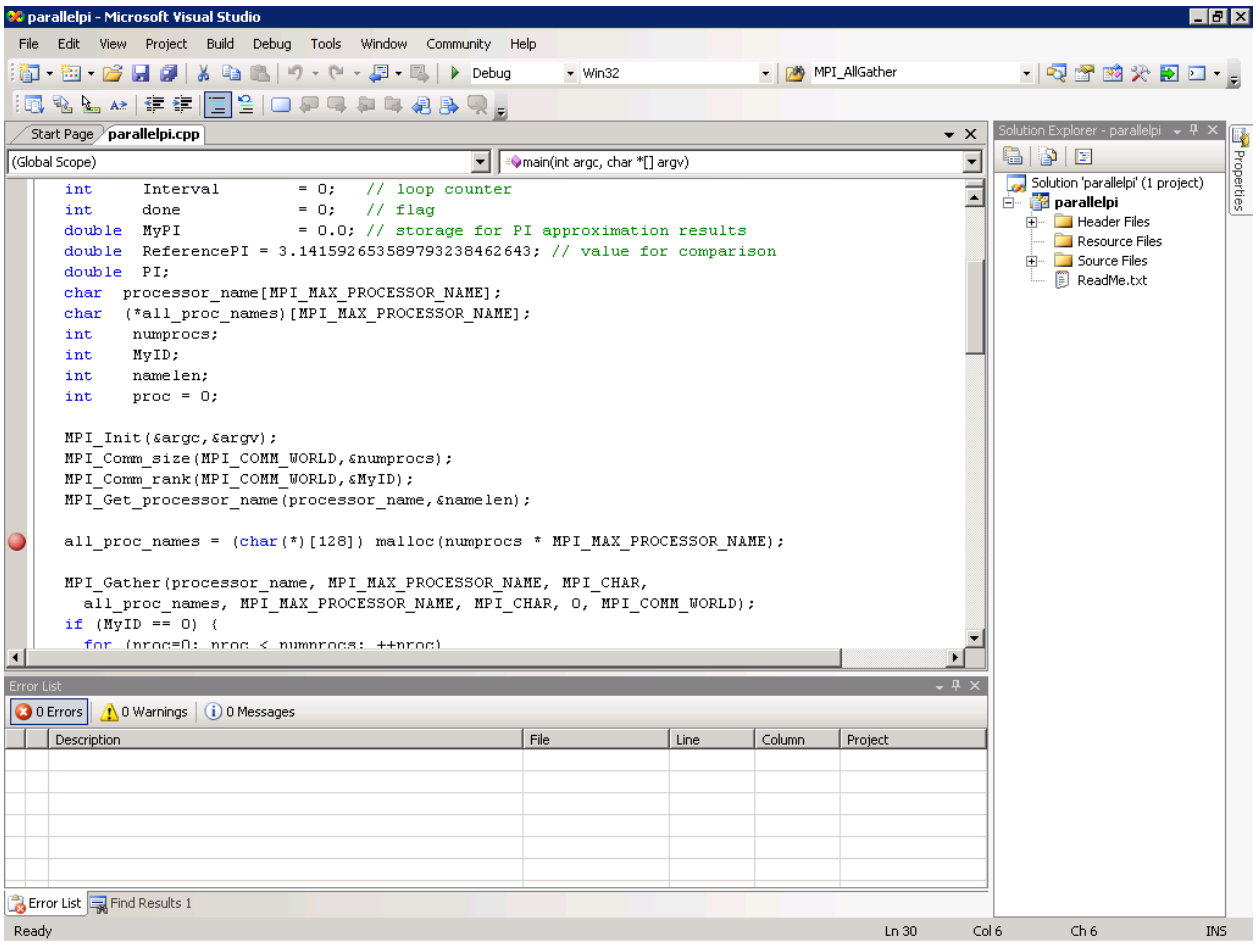
- После настроек, указанных в предыдущем задании, Вы получите возможность отлаживать параллельные MPI программы. Так, запустив программу из меню Microsoft Visual Studio 2005 (команда "**Debug->Start Debugging**"), будут запущены сразу несколько процессов (их число соответствует настройкам предыдущего задания). Каждый из запущеных процессов имеет свое консольное окно. Вы можете приостанавливать любой из процессов средствами Microsoft Visual Studio 2005 и проводить отладку. Подробнее об отладке будет рассказано в следующем упражнении.



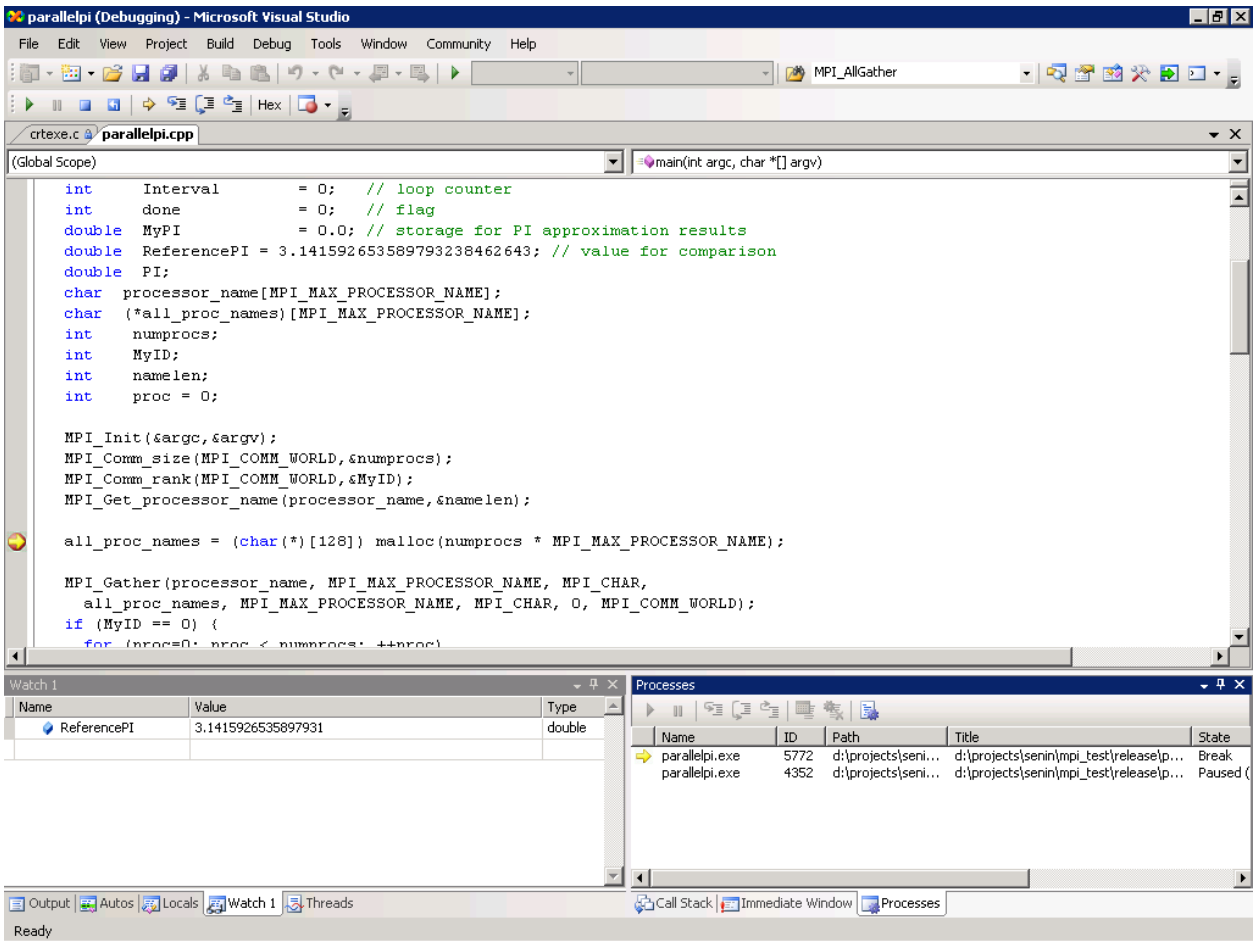
Упражнение 2 – Отладка параллельной программы в Microsoft Visual Studio 2005

Задание 1 – Знакомство с основными методами отладки

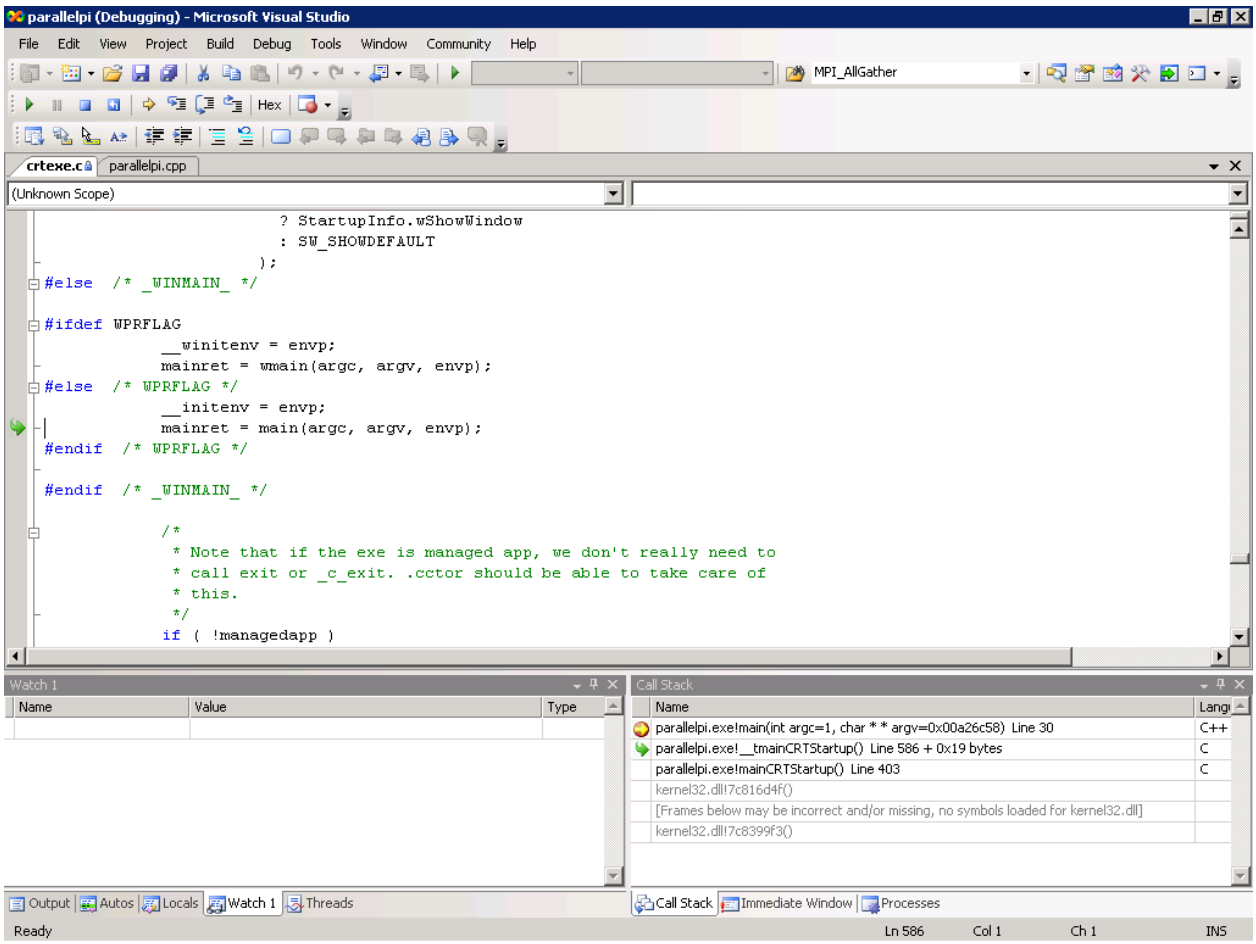
- Откройте проект параллельного вычисления числа Пи (**parallelpi**) и выполните настройки, необходимые для проведения отладки MPI программ (если это еще не было сделано), указав число запускаемых процессов равное 2. Откройте файл “**parallelpi.cpp**” (дважды щелкните на файле в окне “**Solution Explorer**”),
- Поставьте точку остановки на строке выделения памяти с использованием функции **malloc**: установите мигающий указатель вводимого текста на ту строку, на которой Вы хотите установить точку остановки и нажмите кнопку “**F9**”,



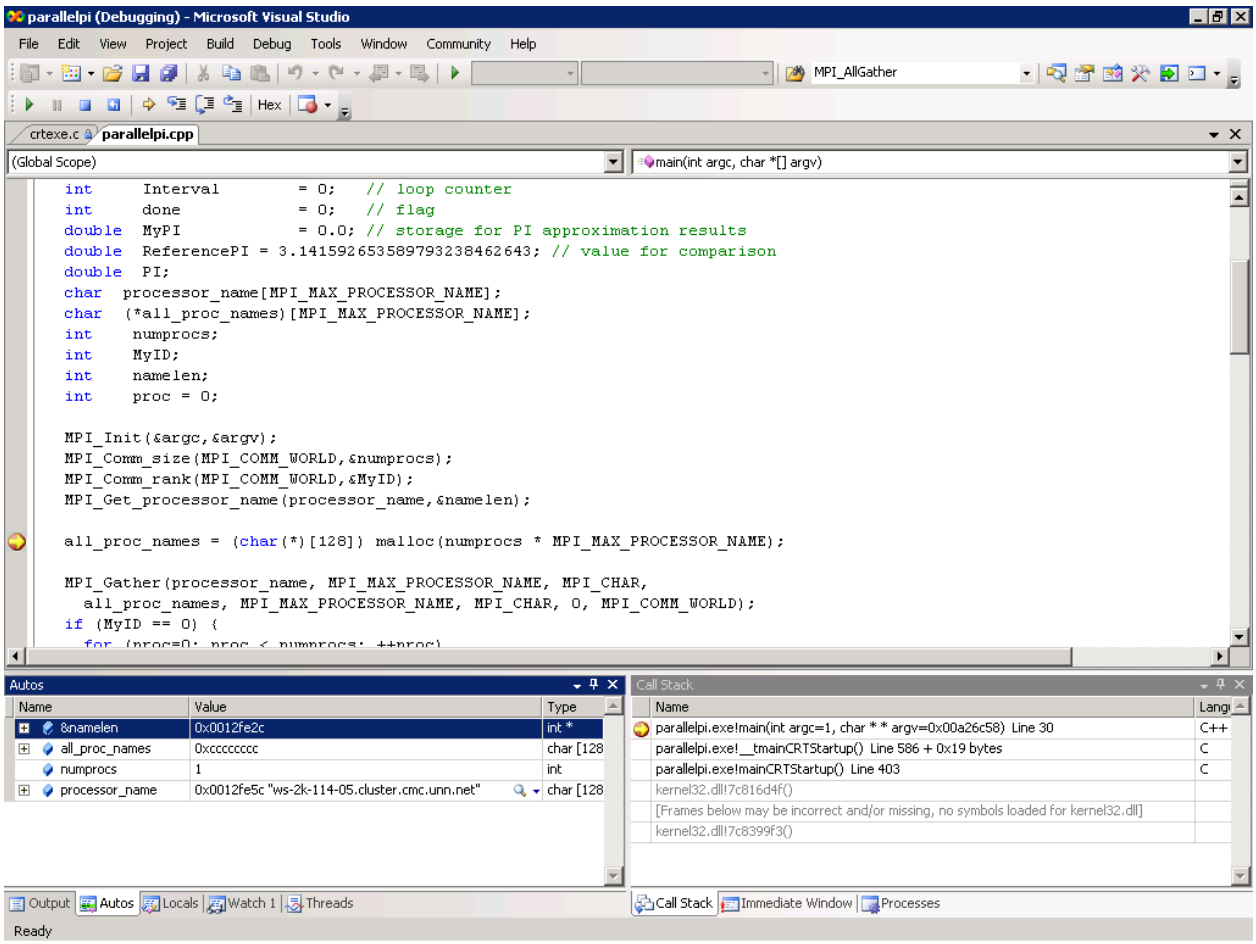
- Выберите отладочную конфигурацию (“**Debug**”) в выпадающем списке конфигураций проекта на панели инструментов. Запустите программу в режиме отладки: выполните команду меню “**Debug->Start Debugging**” или нажмите на кнопке с зеленым треугольником ( ) в панели инструментов Microsoft Visual Studio 2005. Программа запустится, откроется 3 консольных окна: окно процесса “**mpiexec.exe**” и 2 консольных окна параллельной программы вычисления числа Пи. По достижении точки остановки выполнение процессов приостановится, так как оба процесса достигнут указанной точки,



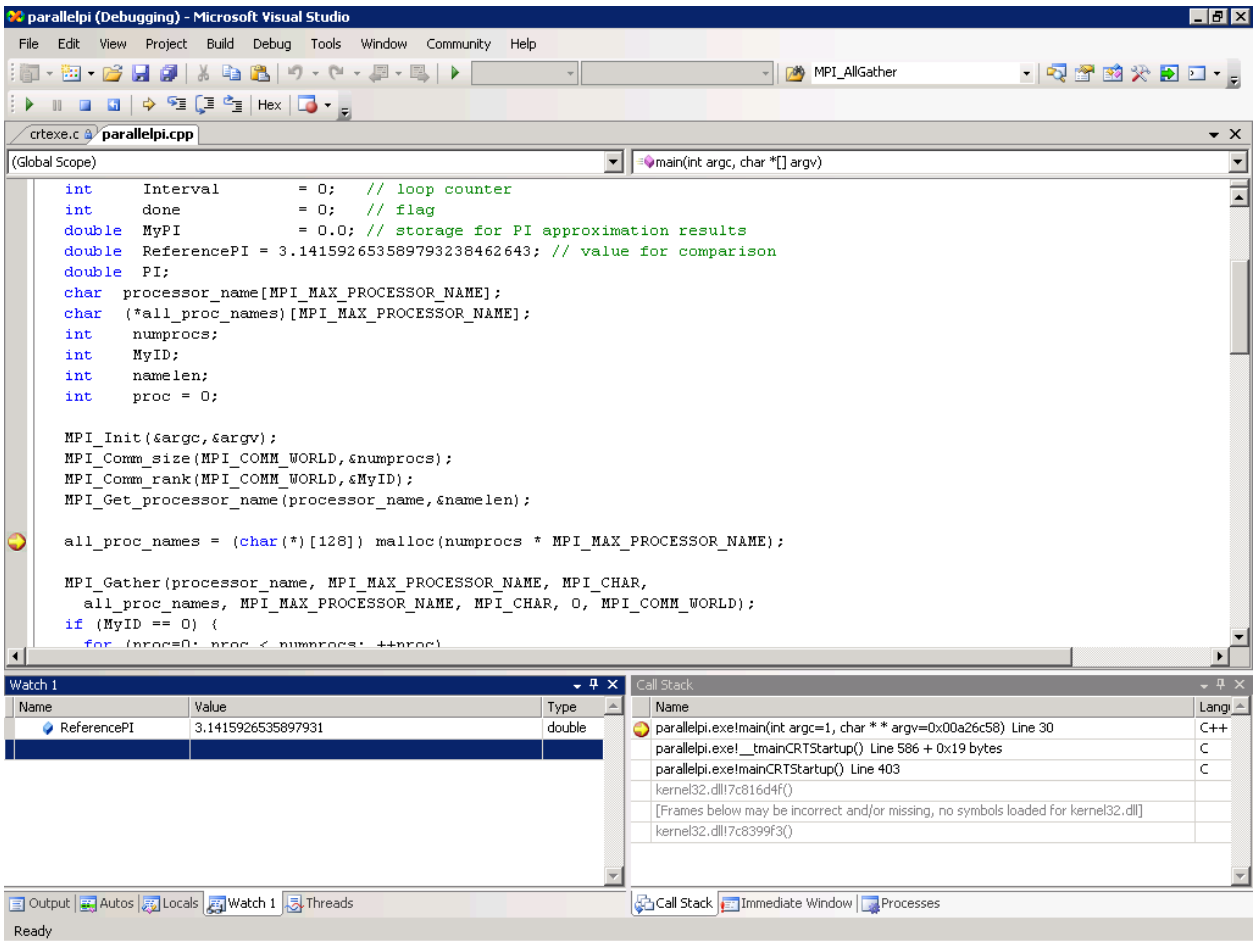
- Откройте окно “**Call Stack**”: выполните команду меню “**Debug->Windows->Call Stack**”. Окно показывает текущий стек вызова функций и позволяет переключать контекст на каждую из функций в стеке. При отладке это помогает понять, как именно текущая функция была вызвана, а так же просмотреть значения локальных переменных функций в стеке. Щелкните 2 раза по функции, находящейся ниже текущей функции в стеке. Текущий контекст изменится, и Вы увидите библиотечную функцию, вызвавшую функцию “**main**”,



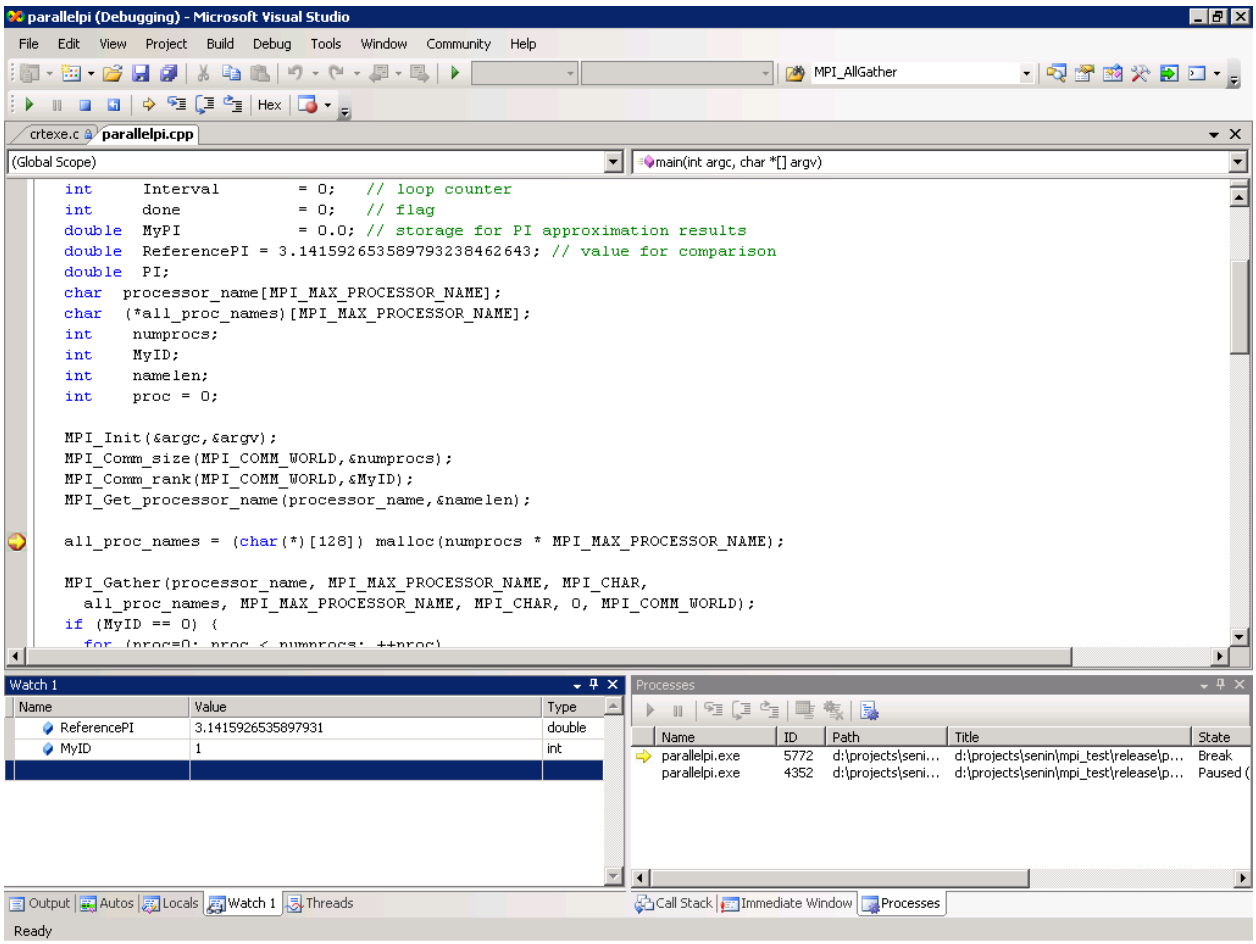
- Щелкните 2 раза на функции “**main**” в окне “**Call Stack**”. Откройте окно “**Autos**” (выполните команду меню “**Debug->Windows->Autos**”). Окно “**Autos**” показывает значения переменных, используемых на текущей и на предыдущих строках кода. Вы не можете изменить состав переменных, так как он определяется средой Microsoft Visual Studio 2005 автоматически,



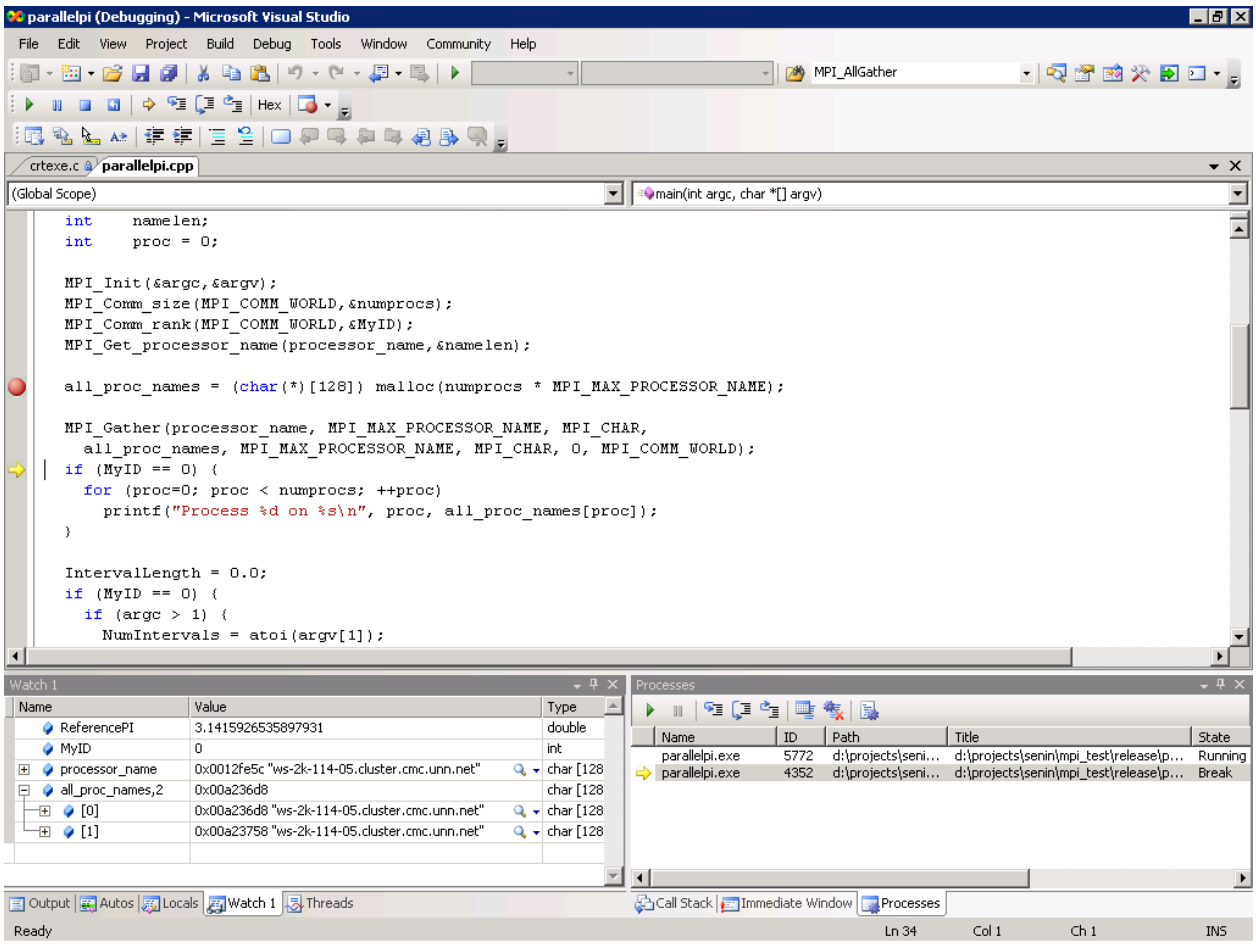
- Откройте окно “**Watch**” (выполните команду меню “**Debug->Windows->Watch->Watch 1**”). В этом окне Вы можете просматривать значения переменных, которых нет в окне “**Autos**”. Например, введите в колонке “**Name**” окна “**Watch**” имя переменной “**ReferencePI**”, нажмите клавишу “**Ввод**”. В колонке “**Value**” появится текущее значение переменной, в колонке “**Type**” – ее тип,



- Откройте окно “**Processes**” (выполните команду меню “**Debug->Windows->Processes**”). Появится окно со списком процессов параллельной MPI программы. В нашем случае их 2. Введите в окне “**Watch**” переменную “**MyID**” – идентификатор текущего процесса. Запомните значение переменной. Затем измените текущий процесс, дважды щелкнув по другому процессу в окне “**Processes**”. Посмотрите значение переменной “**MyID**” **–** оно должно отличаться от предыдущего, так как у каждого процесса в одной MPI задаче идентификаторы различны. При использовании окна “**Processes**” необходимо учитывать следующие 2 момента:
  - Вы можете сделать активным только приостановленный процесс (для приостановки процесса можно поставить точку остановки или выделить работающий процесс в окне “**Processes**” и выполнить команду “**Break Process**” – кнопка в виде двух вертикальных линий в окне “**Processes**”),
  - В окне “**Processes**” есть колонка “**ID**” – идентификатор процесса в операционной системе Windows. Важно помнить, что указанный идентификатор не имеет отношения к рангу (идентификатору) процесса в MPI задаче,



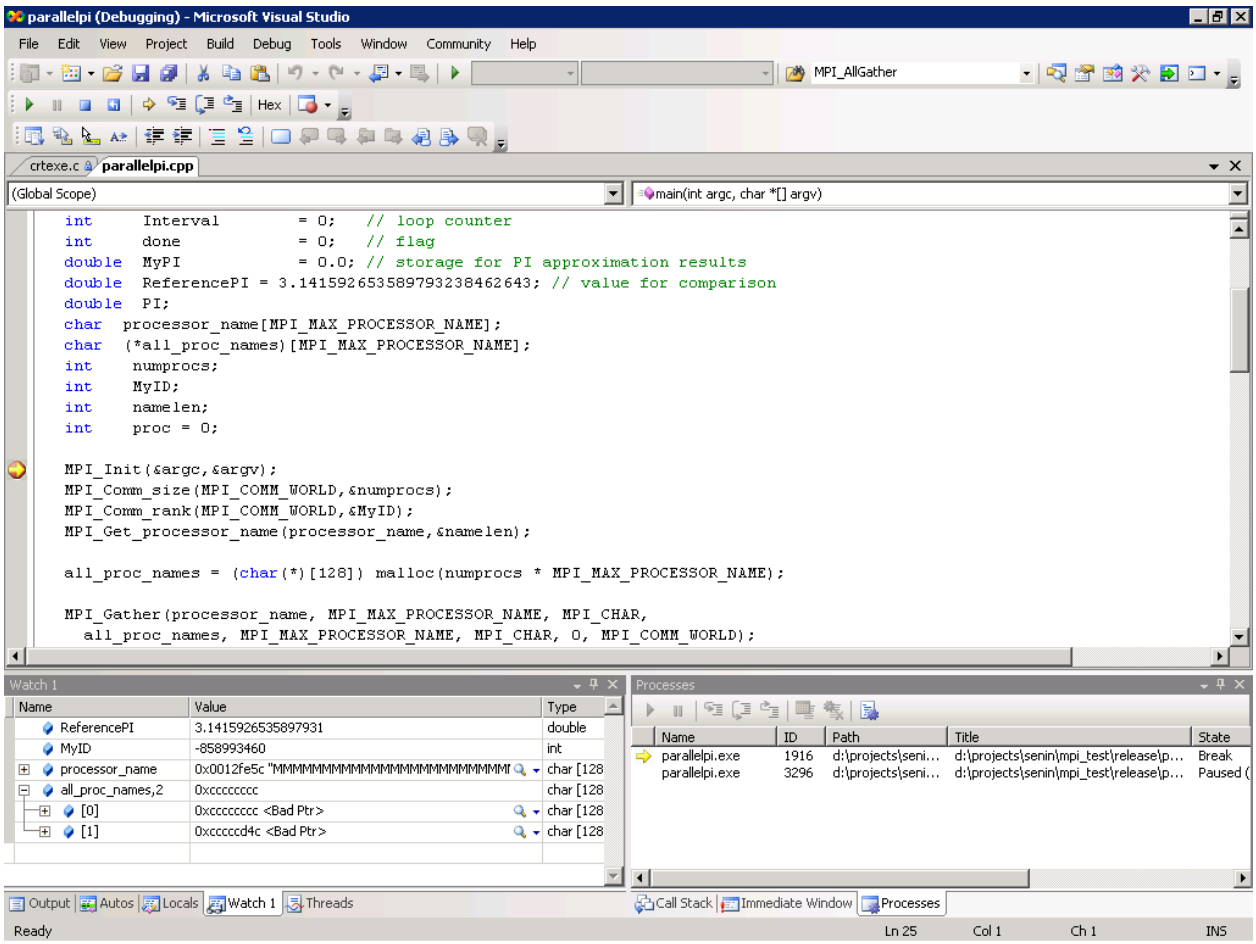
- Перейдите на процесс с нулевым идентификатором “**MyID**”. Введите в окне “**Watch**” переменные “**processor_name**” и “**all_proc_names**”. “**all_proc_names**” – это массив названий узлов, на которых запущены параллельные процессы. Массив используется только на процессе с индексом 0. Для отображения значений, например, двух элементов массива введите “**all_proc_names,2**”. После этого Вы сможете просмотреть значения двух первых элементов массива, нажав на “+” слева от имени переменной. Нажимайте кнопку “**F10**” для пошагового выполнения программы. После каждого шага Вы можете видеть, как изменяются значения внутренних переменных. Так, после вызова функции **MPI_Gather** процесс с индексом 0 будет содержать названия всех узлов, на которых запущены параллельные процессы. Для входа “внутрь” функции Вы можете нажимать кнопку “**F11**” (в рассматриваемом примере все вызываемые функции являются системными, поэтому Вы не сможете увидеть их исходный код).



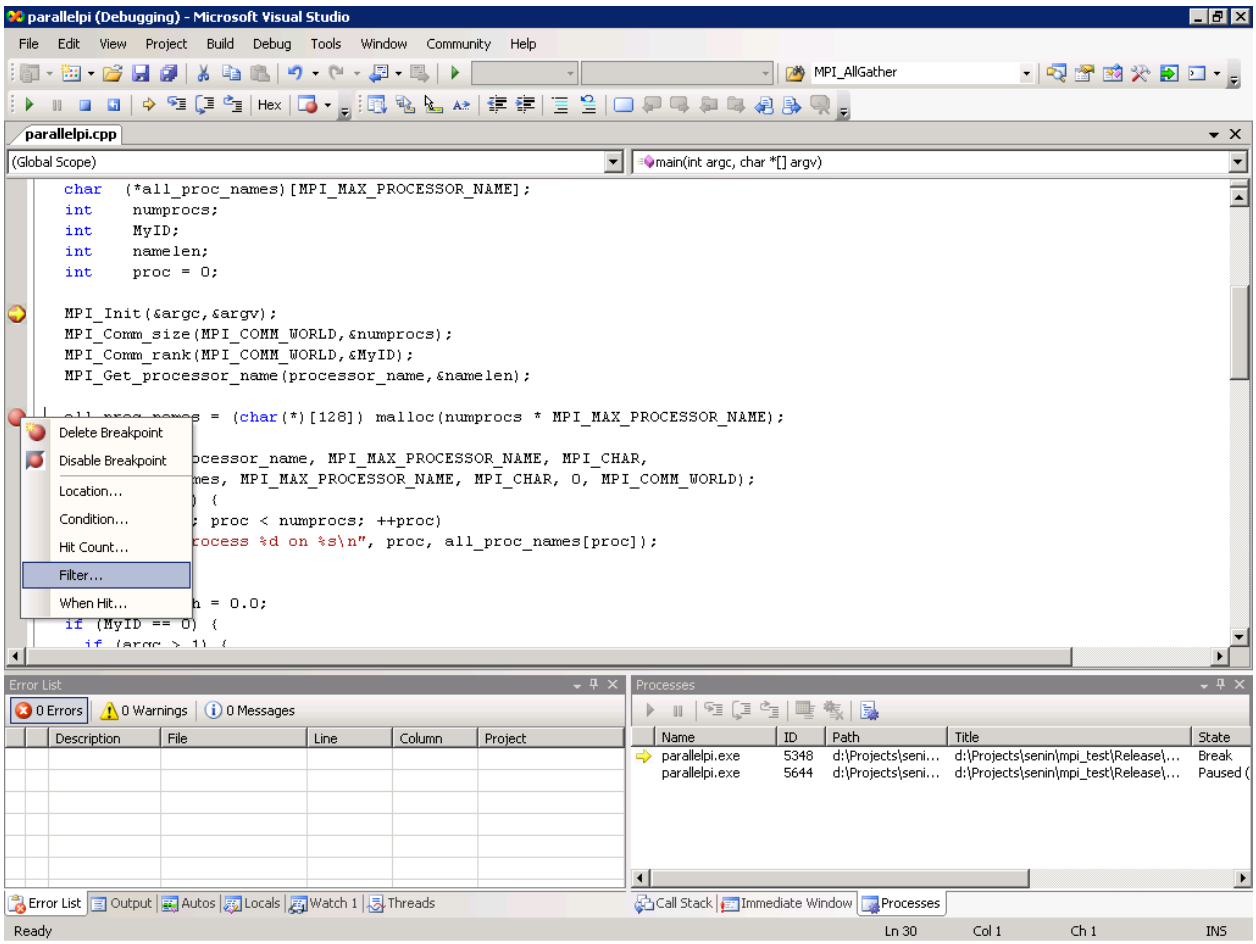
Задание 2 – Использование точек остановки

Мы уже использовали точки остановки в предыдущих пунктах. В данном задании мы рассмотрим вопросы использования **условных точек остановки**, то есть точек остановки, позволяющих указать условия, которые должны выполняться для приостановки работы программы. Использование точек остановки с условиями в некоторых случаях существенно более эффективно, так как позволяет программисту более точно указать необходимые моменты остановки процесса выполнения параллельной программы.

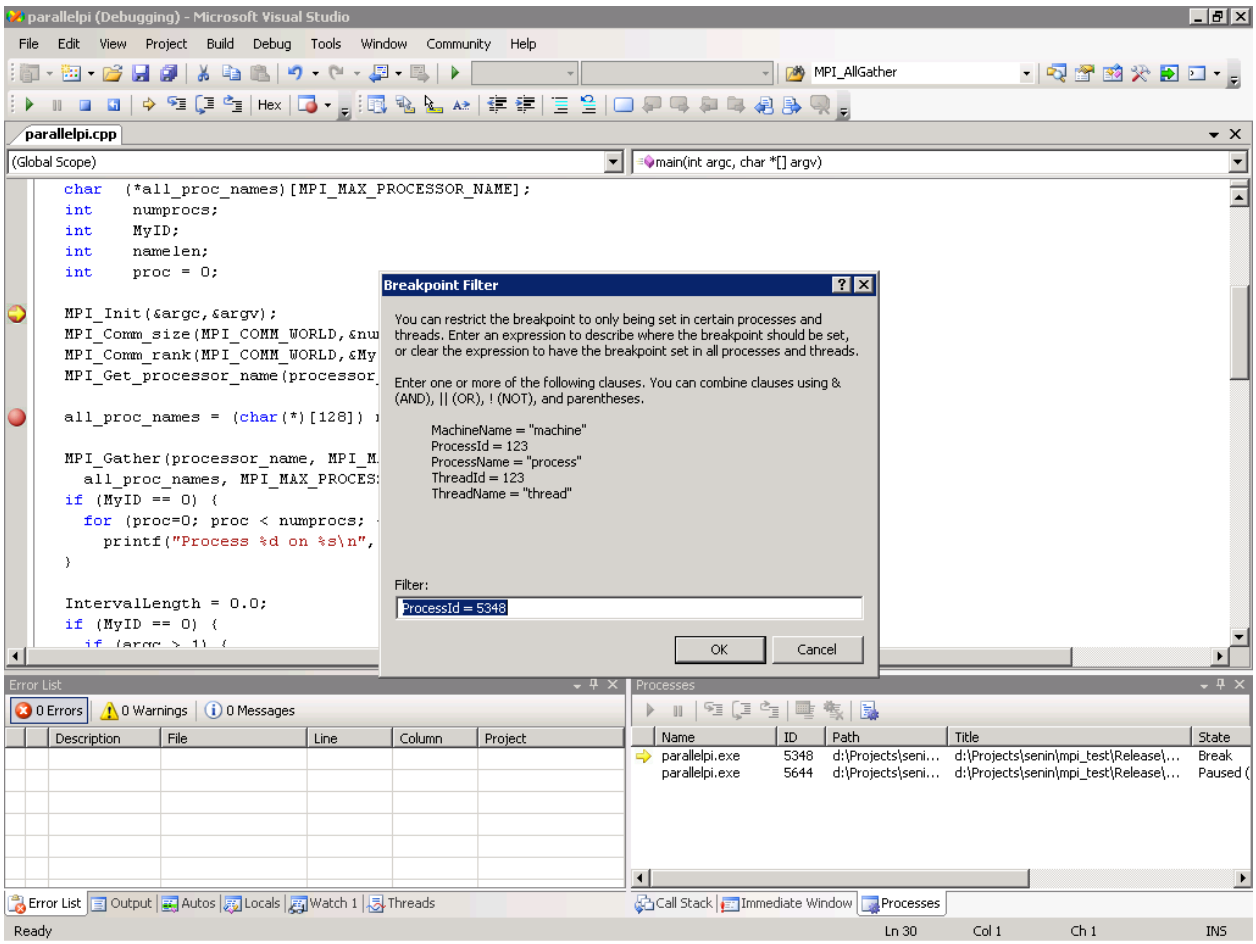
- Откройте проект параллельного вычисления числа Пи (**parallelpi**) и выполните настройки, необходимые для проведения отладки MPI программ (если это еще не было сделано), указав число запускаемых процессов равное 2. Откройте файл “**parallelpi.cpp**” (дважды щелкните на файле в окне “**Solution Explorer**”),
- Поставьте точку остановки на любой строке текста программы (например, на строке с “**MPI_Init**”) и запустите отладку. По достижении точки остановки выполнение процессов приостановится,



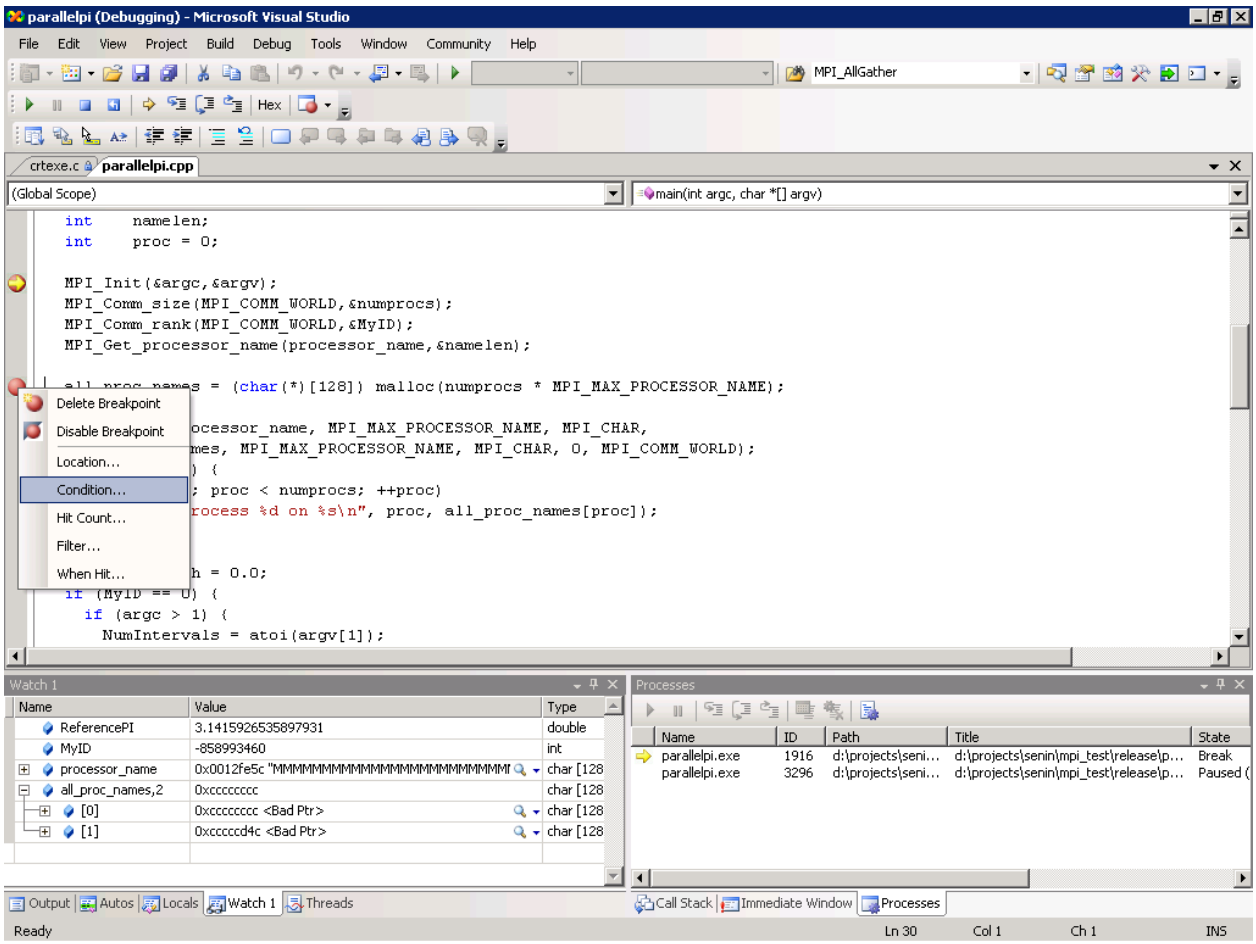
- Вы можете устанавливать точки остановки, которые будут приостанавливать работу только тех процессов (или потоков), которые Вам нужны. Для этого поставьте другую точку остановки ниже по тексту программы, щелкните по красному значку новой точки остановки правой кнопкой мыши и выберите в открывшемся меню пункт “**Filter**”,



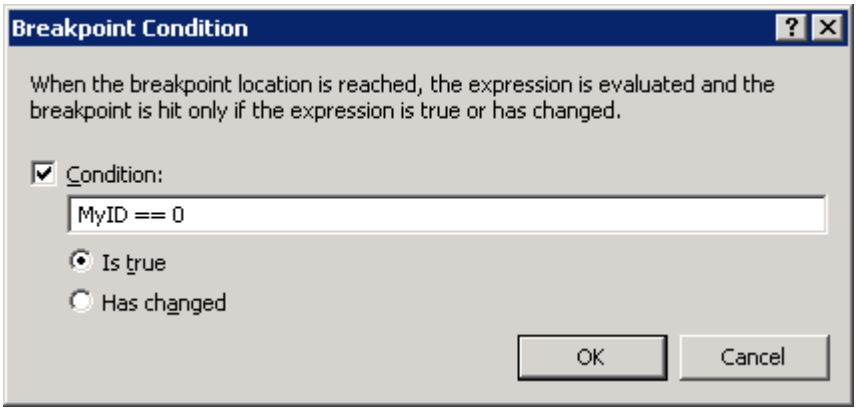
- В открывшемся окне можно указать те процессы и потоки, выполнение которых должно быть прервано по достижении указанной точки остановки. Кроме того, можно указать имя вычислительного узла, на котором точка остановки будет приостанавливать процессы (параметр используется в случае, если часть процессов отлаживаемой задачи запущена на удаленных узлах). Список названий параметров указан в подсказке на окне. При написании логических выражений можно использовать логические функции “**и**” (“**AND**”, “**&**”) , “**или**” (“**OR**”, “||”), “**не**” (“**NOT**”, “**!**”). Например, для указания того, что приостановить работу должен только один процесс, используется выражение “**ProcessId = <идентификатор процесса>**”, где идентификатор процесса можно узнать в окне “**Processes**”. Нажмите “**OK**” для сохранения условия,



- Кроме того, Вы можете указать условия на значения переменных, которые должны выполняться для приостановки работы программы. Щелкните правой кнопкой мыши на точке остановки и выберите пункт “**Conditions**”,



- В открывшемся окне Вы можете указать условия 2 типов:
  - Условие первого типа – приостановить работу программы, если выполнены условия. Для использования этого типа условий установите флаг “**Is true**” и введите условное выражение в поле “**Conditions**”. При вводе выражения используется синтаксис, принятый в C++. Так, для остановки только процесса с индексом 0 следует ввести условие “**MyID == 0**”,
  - Условие второго типа – приостановить работу программы, если значение введенного выражения (необязательно логического) изменилось с момента предыдущего выполнения строки, на которой находится точка остановки. Для использования этого типа условий установите флаг “**Has changed**”,
- Поставьте флаг “**Is true**” и введите условие остановки “**MyID == 0**”. Нажмите “**OK**” для сохранения введенного условия.



Задание 3 – Особенности отладки параллельных MPI программ

Отладчик параллельных MPI программ в Microsoft Visual Studio появился впервые в версии Visual Studio 2005. Однако компиляцию и отладку MPI программ можно производить и в более ранних версиях среды разработки. Обычными приемами в случае отсутствия истинной поддержки MPI со стороны отладчика являются следующие:

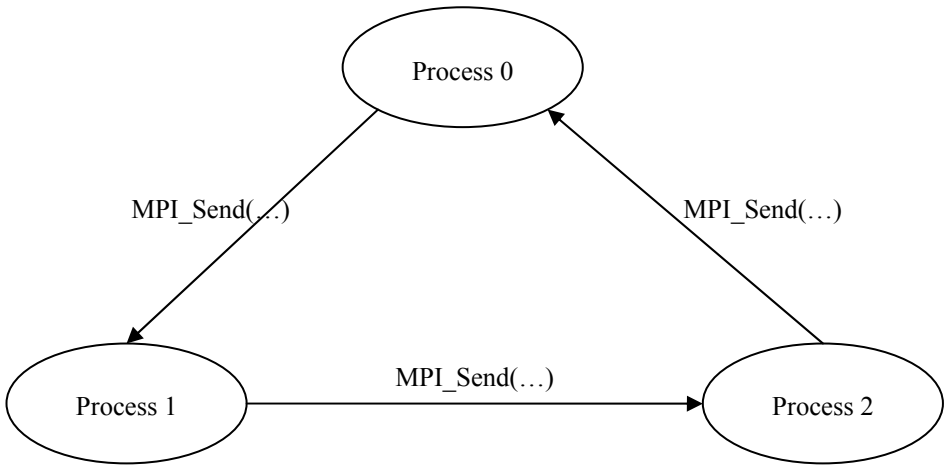
- **Использование текстовых сообщений**, выводимых в файл или на экран, со значениями интересуемых переменных и/или информацией о том, какой именно участок программы выполняется. Такой подход часто называют "**printf отладкой**" ("**printf debugging**"), так как чаще всего для вывода сообщений на консольный экран используется функция "**printf**". Данный подход хорош тем, что он не требует от программиста специальных навыков работы с каким-либо отладчиком, и при последовательном применении позволяет найти ошибку. Однако для того, чтобы получить значение очередной переменной, приходится каждый раз писать новый код для вывода сообщений, перекомпилировать программу и производить новый запуск. Это занимает много времени. Кроме того, добавление в текст программы нового кода может приводить к временному исчезновению проявлений некоторых ошибок,
- **Использование последовательного отладчика**. Так как MPI задача состоит из нескольких взаимодействующих процессов, то для отладки можно запустить несколько копий последовательного отладчика, присоединив каждую из них к определенному процессу MPI задания. Данный подход позволяет в некоторых случаях более эффективно производить отладку, чем при использовании текстовых сообщений. Главным недостатком подхода является необходимость ручного выполнения многих однотипных действий. Представьте, например, необходимость приостановки 32 процессов MPI задания: в случае использования последовательного отладчика придется переключиться между 32 процессами отладчика и вручную дать команду приостановки.

Параллельный отладчик Microsoft Visual Studio 2005 лишен указанных недостатков и позволяет существенно экономить время на отладку. Это достигается за счет того, что среда рассматривает все процессы одной MPI задачи как единую параллельно выполняемую программу, максимально приближая отладку к отладке последовательных программ. К числу особенностей параллельной отладки относятся уже рассмотренное окно "**Processes**", позволяющего переключаться между параллельными процессами, а также дополнительные настройки среды (см. задание 2 упражнения 1).

Задание 4 – Обзор типичных ошибок при написании параллельных MPI программ

Все ошибки, которые встречаются при последовательном программировании, характерны также и для параллельного программирования. Однако кроме них есть ряд специфических типов ошибок, обусловленных наличием в MPI задаче нескольких взаимодействующих процессов. Дополнительные сложности в разработке параллельных программ обуславливают повышенные требования к квалификации программистов, реализующих параллельные версии алгоритмов. И хотя перечислить все типы ошибок, которые могут возникнуть при программировании с использованием технологии MPI, крайне затруднительно, дадим краткую характеристику ряду наиболее характерных ошибок, преследующих начинающих разработчиков. К числу наиболее типичных ошибок при параллельном программировании с использованием технологии MPI следует отнести:

- **Взаимная блокировка при пересылке сообщений**. Предположим, что n процессов передают информацию друг другу по цепочке так, что процесс с индексом **i** передает информацию процессу с индексом **i+1** (для индексов **i=0,…,n-2**), а процесс с индексом **n-1** передает информацию процессу с индексом **0**. Передача осуществляется с использованием функции отправки сообщений **MPI_Send** и функции приема сообщений **MPI_Recv**, при этом каждый процесс сначала вызывает **MPI_Send**, а затем **MPI_Recv**. Функции являются блокирующими, то есть **MPI_Send** возвратит управление только тогда, когда передача будет завершена, или когда сообщение будет скопировано во внутренний буфер. Таким образом, стандарт допускает возможность, что функции отправки сообщения вернут управление только после того, как передача будет завершена. Но передача не может завершиться, пока принимающий процесс не вызовет функцию **MPI_Recv**. А функция **MPI_Recv** будет вызвана, в свою очередь, только после того, как процесс завершит отправку своего сообщения. Так как цепочка передачи сообщений замкнута, отправка сообщений может не завершиться никогда, потому что каждый процесс будет ожидать завершения отправки своего сообщения, и никто не вызовет функцию приема сообщения. Избежать подобной блокировки можно, например, вызывая на процессах с четными индексами сначала **MPI_Send**, а затем **MPI_Recv**, а на процессах с нечетными индексами – в обратной порядке. Несмотря на очевидность ошибки, ее часто допускают, так как при небольших сообщениях высока вероятность, что сообщения уберутся во внутренние структуры библиотеки, функция отправки сообщения вернет управление, и ошибка не даст о себе знать,



- **Освобождение или изменение буферов, используемых при неблокирующей пересылке**. При вызове неблокирующих функций возврат из них происходит немедленно, а адреса массивов с сообщениями запоминаются во внутренних структурах библиотеки MPI. При непосредственной передаче сообщений по сети, которая будет выполнена в отдельном потоке, происходит обращение к исходному массиву с сообщением, поэтому важно, чтобы этот участок памяти не был удален или изменен до окончания передачи (установить, что передача завершена можно вызовом специальных функций). Однако часто об этом правиле забывают, что приводит к непредсказуемому поведению программы. Такого рода ошибки являются одними из самых сложных для отладки,
- **Несоответствие вызовов функций передачи и приема сообщений**. Важно следить, чтобы каждому вызову функции отправки сообщения соответствовал вызов функции приема и наоборот. Однако в том случае, когда число передач заранее не известно, а определяется в ходе вычислений, бывает легко ошибиться и не обеспечить нужных гарантий выполнения данного условия. К этому же типу ошибок можно отнести вызов функций передачи и приема сообщений с несоответствующими тэгами идентификации сообщений.

Контрольные вопросы

- Как локально протестировать работоспособность параллельной программы, разработанной для использования с библиотекой MS MPI до запуска ее на кластере? Какое программное обеспечение должно быть установлено для этого на Вашей рабочей станции?
- Перечислите и дайте краткое описание основным окнам среды Microsoft Visual Studio 2005, используемым при отладке?
- Что такое и для чего используются точки остановки? Что такое условные точки остановки?
- В чем принципиальная особенность отладки параллельных MPI программ в среде Microsoft Visual Studio 2005?
- Какие типичные ошибки при программировании с использованием технологии MPI Вы знаете?