Вопрос №50 Модель компонентных объектов COM.

После компиляции приложение состоит из одного монолитного двоичного файла, который в соответствии с традиционными технологиями остается неизменным пока не будет скомпилирована новая версия. “Модель компонентных объектов Microsoft СОМ (Component Object Model) позволяет разбить монолитное приложение на отдельные части, называемые компонентами . В процессе работы приложения одни версии компонентов могут заменяться другими версиями. СОМ является стандартной спецификацией общего метода создания компонентов и построения из них приложений. Преимущества компонентной модели:

1. Способность приложения эволюционировать с течением времени путем замены устаревших версий компонентов более современными версиями;

2. Адаптация приложения к различным пользователям путем использования компонентов, наиболее адекватных потребностям пользователя;

3. Возможность быстрой сборки приложения из компонентов библиотеки;

4. Повышение эффективности разработки распределенных клиент-серверных приложений.

Общие требования к компонентам:

1. Подключение компонентов во время выполнения приложения требует применения динамической компоновки;

2. Применение принципа инкапсуляции к компонентам; компоненты должны разбиваться на две основные части интерфейс с внешним миром и внутреннюю реализацию;

3. Обладание способностью реализации внутри одного процесса, в разных процессах и на разных машинах, должно обеспечиваться перемещение компонентов в компьютерной сети;

4. Поддерживание клиент-серверной архитектуры приложений,в которой сервер реализуется компонентом, а клиент общается с сервером посредством соответствующих интерфейсов;

5. Разработка отдельных компонентов и целых многокомпонентных приложений должна обебпечиваться на разных языках программирования;

6. Поддерживание библиотечного сервиса управления компонентами.

43. Интерфейс OpenMP.

ОреnМР реализует параллельные вычисления на машинах с несколькими процессорами с помощью многопоточности, которая строится автоматически из последовательного кода по директивам препроцессора, называемым pragma. Количество создаваемых потоков, регулируемое посредством директив, может превышать количестве доступных процессоров. ОреnМР ориентирован на системы с общей памятью, к которым относятся многоядерные системы c общим кэшем. В стандарт ОреnМР входят спецификации набора директив компилятора, процедур и переменных среды. Ключевыми элементами ОреnМР являются: конструкции для создания потоков (директива parallel); конструкции распределения работы между потоками (директивы do/for и section); конструкции для управления работой с данными (выражения shared и private для определения класса памяти переменных); конструкции для синхронизации потоков (директивы critical, atomic и barrier) и др.

ОреnМР можно рассматривать как высокоуровневую надстройку над многопоточностью. Программная модель ОреnМР представляет собой fork-join параллелизм, в котором главный поток по необходимости порождает вспомогательные потоки при вхождении программы в параллельные области. ОреnМР позволяет быстро распараллелить программы с циклами, выполняющими большой объем вычислений. Одна ОреnМР -программа выполняется параллельно на многопроцессорной систем и выполняется последовательно на однопроцессорной системе. Для языка программирования С тины данных и функции ОреnМР определены в подключаемом файле omp.h.

Пример программы на языке С, использующей ‚директивы ОреnМР ,приведен на рис. 5.3. Внешняя функция average с тремя входными аргументами возвращает их среднее значение. Функция master\_example имеет три аргумента: массив х; массив xold; число n элементов в массивах x и xold; пороговое значение tol.

За объявлениями локальных переменных с, i, toobig, error, y следует директива #pragma omp parallel. Эта важнейшая директива paraller указывает на необходимоств автоматического распараллеливания нижеследующего блока, заключенного в {}. Блок состоит из одного оператора цикла do, завершающего вычисления в случае, когда ни одно среднее значение трех соседних элементов массива х не превышает порогового значения tol.

Следующая директива #pragma omp for private(i) относится к последующему циклу for. Благодаря присутствию private(i), для каждого вспомогательного потока, создаваемого с целью параллельной реализации различных итераций цикла, вводится своя копия переменной i. Директива #pragma omp single указывает на то, что только один поток (не обязательно ведущий) выполняет нижеследующий блок, а именно, присваивание переменной toobig значения 0.

Директива #pragma omp for private(i, y, error) reduction(+:toobig) относится к последующему циклу for. Вспомогательные потоки, реализующие различные итерации цикла, имеют свои копии переменных i, y, error. Слово reduction указывает в момент завершения цикла на суммирование (+) значений всех копий переменной toobig, вычисленных различными вспомогательными потоками. Директива #pragma omp master специфицирует нижеследующий блок, исполняемый только ведущим потоком.