Введение

Термины ***распределенная архитектура, распределенная обработка данных, распределенные вычисления, распределенная система*** и т.п. широко используются в технической литературе, посвященной современным компьютерным технологиям. Говорят о распределенной архитектуре компьютера (многопроцессорные или многомашинные вычислительные комплексы), о распределенных операционных системах, о распределенных базах данных. Все эти понятия объединяет наличие вычислительных и информационных ресурсов, распределенных в пространстве, а также процессов, использующих эти ресурсы.

Под вычислительными ресурсами, как правило, подразумевают компьютеры (процессоры, оперативная и вторичная память), а под информационными – файлы данных и (или) каналы передачи данных. Процессы – это программы, работающие на компьютерах и использующие вычислительные ресурсы для решения задач.

Простейшим примером распределенной системы может служить два отдельно стоящих компьютера (даже не объединенных в сеть), решающие общую задачу. Такая архитектуры системы может быть применена, например: для повышения надежности системы; для решения трудоемкой, но допускающей распараллеливание задачи (задача разбивается на подзадачи, каждая из которых решается на отдельном компьютере); для решения задачи, требующей постоянного диалога с двумя различными специалистами (примером может служить задача составления бухгалтерского баланса, когда два бухгалтера обрабатывают различные группы счетов).

В некоторых случаях может потребоваться обмен информацией между компьютерами. В простейшем случае это может быть сделано с помощью магнитного носителя (дискеты, CD-диска и т.д.), а может быть использована локальная сеть или другие специальные устройства. Заметим, что обмен информацией не может быть произведен в произвольное время: процесс–источник должен успеть подготовить информацию, а процесс-приемник должен быть готов принять и использовать информацию. В связи с тем, что процессы на разных компьютерах работают асинхронно (не зависимо друг от друга), для обмена данными требуется иногда ***синхронизировать*** их работу.

Рассмотрим более сложный пример распределенной системы. Пусть система содержит триста компьютеров (например, банк), объединенных в единую сеть и решающий единую задачу (продолжая банковскую тему: учет денежных потоков в банке). На каждом компьютере работает несколько программ, обеспечивающих работу нескольких подсистем единой банковской системы (например, кроме собственно банковской системы, в банке может работать внутренняя почтовая система, может существовать выход в глобальные сети и т.д.). Кроме того, как правило, в таких больших компьютерных системах работают компьютеры разного типа, управляемые различными операционными системами и объединенные различными

каналами. Ясно, что управление в такой системе, само по себе является сложной задачей.

Далее будем применять термин ***распределенное приложение***. Под распределенным приложением будем понимать несколько процессов (как правило, работающих на разных компьютерах или разных процессорах), предназначенных для решения общей задачи. Распределенное приложение может принимать различные формы: в простейшем случае, распределенное приложение – это две программы, работающие на разных компьютерах и обменивающиеся данными; в другом – это может быть одна или несколько программ, работающих на разных компьютерах и работающих с сервером базы данных (например, Microsoft SQL Server или Oracle Server); в третьем – это Active Server Page-приложение, для доступа к которому необходим браузер. Заметим, что в рамках распределенного приложения могут использоваться уже готовые компоненты (серверы баз данных, WWW-серверы, браузеры), позволяющие существенно упростить его разработку.

Проанализировав приведенные выше примеры, можно выделить следующие проблемы, с которыми сталкивается разработчик распределенного приложения.

1. Проблема обмена данными между процессами в распределенном приложении.
2. Проблема синхронизации процессов в распределенном приложении.
3. Проблема совместного использования ресурсов процессами распределенного приложения.
4. Проблема взаимодействия процессов, работающих в разных операционных средах.
5. Проблемы масштабирования распределенного приложения.

Проблема совместного использования ресурсов возникает, например, при использовании файла данных двумя или более процессами одного или нескольких приложений. Если при этом один процесс записывает (или обновляет) данные, а другой читает, то требуются специальные механизмы для обеспечения согласованности этих операций. Такая же проблема возникает в многозадачных операционных системах [2].

Под масштабированием понимается возможность расширения распределенного приложения (увеличения числа взаимодействующих процессов, увеличение объемов данных и т.д. и т.п.).

Сетевые и распределенные операционные системы предоставляют собственные методы решения данных проблем. Например, для передачи данных в TCP/IP-сети можно использовать интерфейсы сокетов или именованных каналов, если компьютеры управляются операционными системами Windows или Unix. Для синхронизации процессов, работающих на различных компьютерах, операционная система NetWare предоставляет механизм семафоров. Проблемы совместного использования общих данных обычно решается с помощью серверов систем управления базами данных. Независимость распределенного приложения от операционной среды обеспечивается правильным выбором средств разработки. Например,

распределенное приложение, разработанное на языке Java, которое использует для управления данными СУБД Oracle, может работать практически в любой операционной среде.

Данное учебно-методическое пособие посвящено разработке простейших распределенных приложений в среде Windows. При этом предполагается, что читатель имеет навыки программирования на языке C++, обладает знаниями основ теории операционных систем, локальных компьютерных сетей.

Первая глава пособия посвящена описанию стандартных моделей взаимодействия процессов в распределенном приложении.

Во второй главе рассматривается структура стека протокола TCP/IP, назначение и основные принципы взаимодействия его компонентов.

В третьей главе пособия описываются основные возможности интерфейса Windows Socket API, являющегося основой для построения распределенных приложений.

Четвертая и пятая главы посвящены средствам межпроцессного взаимодействия, которые тоже используются для построения распределенных приложений в локальной сети.

В шестой главе рассматриваются основные принципы построения и программирования параллельного сервера.

Заключительная (седьмая) глава состоит из десяти практических работ, предназначенных для закрепления изученного материала. Для выполнения работ необходимо наличие нескольких компьютеров с операционной системой Windows XP, связанных локальной сетью. Для разработки приложений на языке C++ требуется среда разработки Microsoft Visual Studio не ниже седьмой версии и доступ к соответствующей версии MSDN. Практические работы состоят из нескольких связанных заданий.

Следует предупредить читателя о том, что описания интерфейсов и функций операционной системы Windows, приведенные в пособии, не является полными и не могут быть использованы в качестве серьезного справочника, т.к. эти описания предназначены только для решения той узкой задачи, которую ставит перед собой автор. Для более глубокого и полного изучения следует обратиться к литературе приведенной в конце пособия и, конечно же, к соответствующей документации.