Обеспечение многопользовательской работы на файловой системе

А. М. Акименков

В работе описывается метод, с помощью которого может быть реализовано разделение доступа при многопользовательском интерфейсе на файловой системе.

Введение

При создании программных систем, которые могут использоваться не одним, а одновременно несколькими пользователями, как правило, возникает вопрос — как обеспечить корректное разделение доступа в такой многопользовательской системе. Если речь идет о разделении доступа к некоторой базе данных, то можно использовать для этого внутренний механизм базы данных, который имеют современные базы данных. Однако как обеспечить разделение доступа, например, к файлам или другим ресурсам, не пользуясь средствами базы данных? Ниже мы опишем простой способ на основе файловой системе в операционной системе Windows. Этот способ имеет два преимущества — во-первых, простота, и, во-вторых, отсутствие обращения к базам данных, что может требовать существенных временных затрат.

Данных способ успешно используется уже несколько лет в таких сложных программных продуктах, как Система потокового ввода документов Ежегодного собрания акционеров ОАО «Газпром», и Система потокового ввода документов Пенсионного фонда РФ.

1. Разделение доступа на файловой системе Windows

В основе механизма разделения доступа лежит использование флагового файла. То есть, коротко говоря, если некоторый файл, который назовем флаговым, отсутствует, то программа должна иметь доступ к разделяемым ресурсам, если присутствует, то программа не должна иметь доступа.

Конечно, если механизм разделения доступа реализовывать следующим образом: 1) проверить наличие флагового файла, 2) если его нет, создать его и обратиться к требуемым ресурсам, то этот механизм, очевидно, не будет работать корректно, поскольку шаги 1) и 2) разделены во времени и к моменту выполнения шага 2) условие 1) может измениться.

Таким образом, проверка наличия флагового файла и его создание должны обеспечиваться некоторым одним действием. В качестве такого действия необходимо выбрать исполнение некоторой команды операционной системы. Однако такая команда должна удовлетворять требованию, что при одновременном запросе на исполнение она исполняется только для одного из запросов.

Нужно отметить, что отнюдь не все команды операционной системы Windows, которые на первый взгляд должны удовлетворять такому требованию, ему удовлетворяют.

Например, одновременное исполнение команд MoveFile(orig_file, dest_file1) и MoveFile(orig_file, dest_file2), вопреки ожиданию, при одновременном обращении может привести к следующему результату: возникнут одновременно оба файла dest_file1 и dest_file2.

Однако, как показал опыт длительного использования, команда CreateFile, вызываемая со значением параметра *dwCreationDispostion*= CREATE_NEW, действительно выполняется только для одного из поступающих запросов и возвращает значение INVALID_HANDLE_VALUE для остальных запросов.

Таким образом, является корректным следующий алгоритм разделения доступа к общим ресурсам.

1. Исполнить команду CreateFile(Flag, 0, 0, NULL, CREATE_NEW, FILE_ATTRIBUTE_NORMAL, NULL). Здесь Flag — имя создаваемого флагового файла.
2. Если команда CreateFile создала флаговый файл Flag успешно, предоставляется доступ к разделяемым ресурсам. Если команда CreateFile вернула значение INVALID_HANDLE_VALUE, в доступе к разделяемым ресурсам отказано.
3. Если доступ к разделяемым ресурсам предоставлен, выполнить необходимую работу с этими ресурсами, после чего удалить флаговый файл Flag.

Как показал опыт, описанный механизм разделения доступа работает корректно не только на отдельном компьютере, но и при работе в сети. То есть ресурсы, пользователи и флаговые файлы могут находиться на различных компьютерах, соединенных в сеть. Операционная система обеспечивает требуемую для данного механизма единственность выполнения команды CreateFile при поступлении многих запросов одновременно.