Лекция 5. ВЗАИМОДЕЙСТВИЕ РАСПРЕДЕЛЕН-НЫХ ПРОЦЕССОВ ПО СХЕМЕ "КЛИЕНТ-СЕРВЕР".

Одной из типичных схем взаимодействия между процессами в параллельных (распределенных) программах является взаимосвязь «клиент-сервер». Процесс-клиент при этом запрашивает некоторый сервис, затем ожидает обработку запроса. Процесс-сервер многократно ожидает запрос (на использование ресурса), обрабатывает его и посылает ответ. При этом существует двунаправленный поток информации от клиента к серверу и обратно. Таким образом, сервер – это процесс, постоянно обрабатывающий запросы от клиентских процессов, используя при этом асинхронную передачу сообщений (рис 1.1).

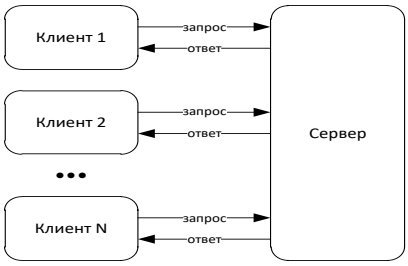


Рисунок 1.1 – Схема взаимодействия типа «клиент-сервер»

В распределенных системах (системах с распределенной памятью) сервер реализуется набором подпрограмм, обеспечивающих доступ к ресурсу или структурам данных, представляющих собой ресурс.

Взаимодействие между клиентом и сервером реализуется вызовом соответствующих процедур доступа к ресурсу.

Серверы могут быть реализованы в соответствии со следующими схемами управления:

1) с помощью рассылки сообщений;
2) с использованием рандеву.

1.1. Процедура обмена сообщениями при взаимодействии клиента и сервера и алгоритмы функционирования сервера

Взаимодействие клиента и сервера с помощью рассылки сообщений при выделении ресурсов инициируется следующим образом: клиентский процесс отправляет сообщение в канал запроса, а затем получает ответ (результат) из канала ответа. При этом каждому клиенту выделяется собственный канал ответа.

В структуре сообщения должно быть определено, какую опцию вызывает клиент (запрос или освобождение ресурса). Необходимо также передавать аргументы сообщения (номер выделяемого ресурса и его запрашиваемое количество, если ресурс выделяется не целиком, а по частям). Когда нет доступных элементов ресурса, сервер не может ожидать, обслуживая запрос с выделением ресурса. Он запоминает запрос и откладывает посылку ответа (тем самым блокируя клиента). Когда ресурс освобождается, сервер возвращается к сохраненному запросу и передает освободившийся элемент запрашивающему процессу. После отправки сообщения с запросом на выделение ресурса клиент ждет получения элемента ресурса, освобождая ресурс, клиент не ждет подтверждения его освобождения. Серверу приходится сохранять ожидающий запрос, а также впоследствии проверять очередь ожидающих процессов. Обобщенные алгоритмы функционирования клиента и сервера представлены на рис 1.2 и 1.3.

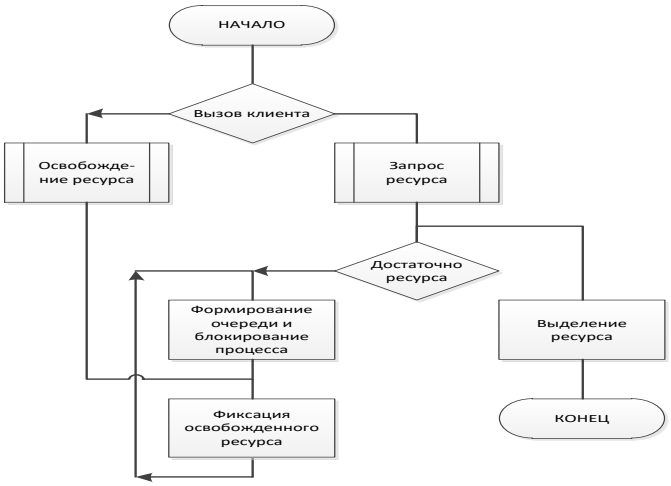


Рисунок 2.2 – Алгоритм функционирования сервера

При взаимодействии клиента и сервера при разделении ресурсов осуществляется обмен между ними тремя сообщениями (запрос, подтверждение, освобождение) и выделение ресурса клиенту, если он свободен.

Рисунок 1.3 – Алгоритм функционирования клиента

1.2. Реализация концепции рандеву при функционировании сервера

Реализация рандеву предполагает инициализацию выполнения некоторой процедуры, доступ к которой регулируется сервером, в ответ на запрос клиента. Действия этой процедуры и клиентов необходимо синхронизировать (клиенты-поставщики информации для обработки этой процедурой).

Процедуре и клиенту необходимо рандеву (взаимодействие), которое синхронизируется сервером. Возможно отличие от первой схемы, когда клиент непосредственно не взаимодействует с процедурой, запрашивает ресурс (обращаясь к серверу), ждет от сервера результатов обработки, сообщение о завершении этой обработки отсылает серверу сама вычислительная процедура (совместно с результатами). Как и в описанном выше случае, сервер обслуживает любой процесс, запросивший ресурс, клиент должен ждать освобождения ресурса. Таким образом, существует два способа решения задачи диспетчеризации при организации серверов:

1) отделение сервера от ресурса; в этом случае клиент осуществляет вызов сервера для получения доступа к ресурсу (запрос ресурса) и в случае получения доступа непосредственно обращается к ресурсу; В этом случае используется пять сообщений (рис 1.4).

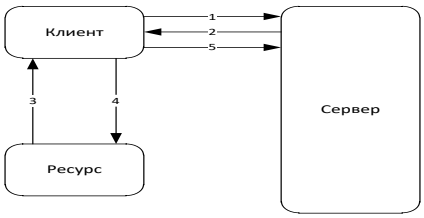


Рисунок 1.4 – Схема взаимодействия сервера, клиента и ресурса

2) сервер является посредником между клиентом и ресурсом (процессом, являющимся ресурсом); клиенты вызывают сервер, указывая в нем требуемый вид обработки ресурса, далее сервер сам обращается к ресурсу, разграничивая доступ клиентов; в данном случае инициируется передача 4 сообщений (рис 1.5).

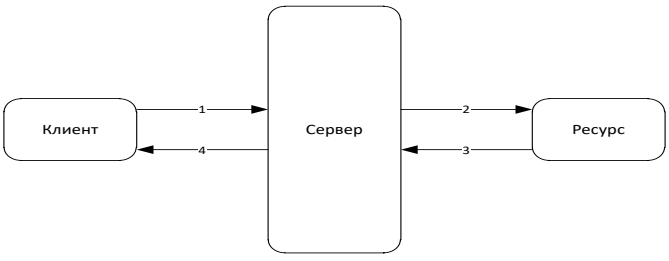


Рисунок 1.5 – Схема сервера-посредника между клиентом и ресурсом

1.3. Информационные структуры сервера

В случае, если ресурс представляет собой совокупность дискретных элементов и выделяется клиентом не целиком, а по частям, сервером должны быть использованы следующие информационные структуры:

1) таблица ресурсов, каждый элемент которой должен содержать поля (рис 1.6), где $V_{св}$– количество свободного ресурса, которое сравнивается с количеством требуемого клиенту ресурса ($V_{тр}$), передаваемом в

запросе; в случае превышения свободного количества над требуемым ресурс выделяется в использовании клиентом; в противном случае клиент блокируется; при выделении ресурса клиенту вносятся изменения в количество свободного ресурса.

№ ресурса	Информация	
	Количество свободного ресурса $V_{св}$	Ссылка на очередь клиентов

Рисунок 1.6. – Таблица ресурсов сервера

2) очередь ожидающих данный ресурс клиентов; она может представлять собой массив структур, каждая строка которого соответствует номеру ресурса, а каждый элемент, которого имеет вид, представленный на рис 1.7.

Идентификатор процесса	Требуемое количество ресурса

Рисунок 1.7 – Элемент очереди доступа к ресурсу

В этом случае действия сервера при выполнении запроса и освобождения ресурсов могут быть определены в виде фрагмента алгоритма, представленного на рис 1.8.

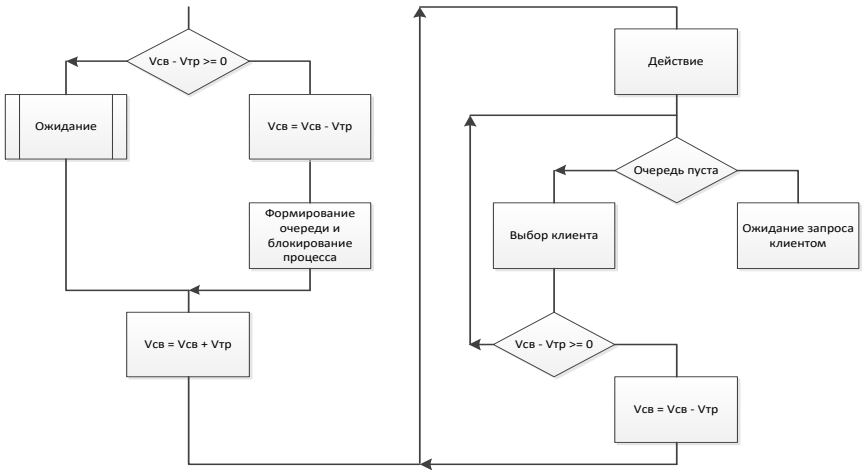


Рисунок 1.8 –Фрагмент алгоритма контроля информационных структур сервера при запросе и освобождении ресурса

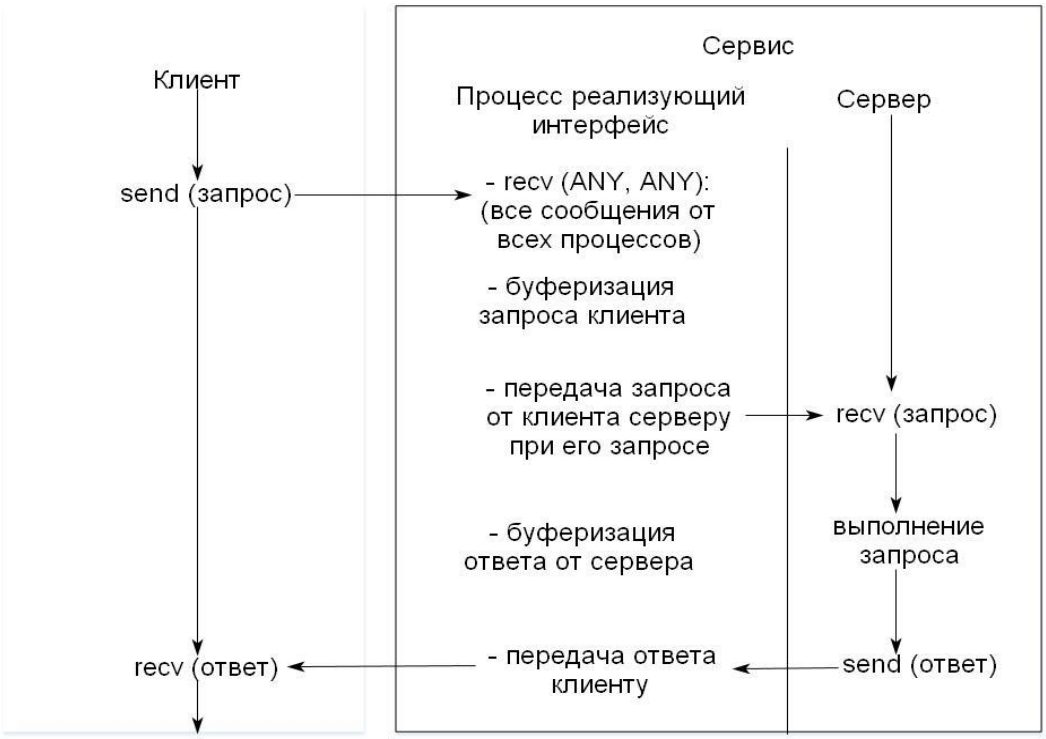
Случай, когда отправляющий процесс не может быть блокирован

Процесс является некоторым сервисом (в частности, системным сервисом), который отправил ответ клиенту на его запрос (ожидание ответа клиентам не гарантируется, в случае синхронной передачи сервис д/б заблокирован, однако заблокированным он быть не может).

Пример:

Процесс, реализующий ввод данных и передачу их клиентам (процесс, вводящий данные и передающий их клиентам, заблокирован быть не может).

Схема промежуточной буферизации запросов/ответов, позволяющая исключить блокирование при синхронной передаче


Клиент
send (запрос)
recv (ответ)
Сервис
Процесс реализующий интерфейс
Сервер
- recv (ANY, ANY): (все сообщения от всех процессов)
- буферизация запроса клиента
- передача запроса от клиента серверу при его запросе
recv (запрос)
выполнение запроса
- буферизация ответа от сервера
- передача ответа клиенту
send (ответ)