**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ**

**(МТУСИ)**



Кафедра системного программирования

Курсовая работа.

Вариант №12

**Выполнили:**

Студенты группы БВТ1903

Князев И.В.

**Проверила:**

Алексанян Д.А.

Москва 2021

**Цель работы:**

Закрепление и углубление теоретических знаний в области современных операционных систем, приобретение практических навыков разработки клиент-серверных приложений, использующих стандартные механизмы межпроцессного взаимодействия.

**Задание:**

Разработать многозадачное приложение, выполняющее получение, сбор и вывод системной информации в соответствии с вариантом задания. Номер варианта задания на курсовую работу совпадает с номером, под которым записана фамилия студента в журнале группы.

**Функционал, который необходимо реализовать в сервере №1:**

* Возможность получения названия используемого видеоадаптера сервера;
* возможность скрывать окно серверного процесса на переданное время (от 1000 до 10000 мс), а также возвращать результат выполнения (успех/ошибка).

**Функционал, который необходимо реализовать в сервере №2:**

* Возможность получения процента используемой физической памяти;
* возможность получения процента используемой виртуальной памяти.

**Постановка задачи:**

Необходимо реализовать многозадачное приложение на основе технологии «клиент-сервер» средствами интегрированной среды разработки Microsoft Visual Studio. Процессы 1 и 2 выступают как серверы приложений, третий процесс является их клиентом. Обмен данными между клиентами и серверами осуществляется при помощи сокетов.

И клиентский, и серверные процессы оформляются как отдельные приложения, запускаются и завершают работу независимо друг от друга. Серверы должны быть многопоточными, то есть допускать одновременное подключение нескольких клиентов. Серверные процессы по запросу получают системную информацию в соответствии с вариантом задания и передают ее клиенту с указанием текущего времени. На стороне клиента необходимо реализовать интерфейс, позволяющий пользователю подключиться по выбору к первому или второму серверу, а также возможность раздельного получения и отображения данных по каждому из пунктов индивидуального задания.

Необходимо предусмотреть проверку на повторный запуск серверов, т.е. исключить ситуацию, когда запускается более чем один экземпляр каждого сервера. Клиентов может быть несколько. В течение всего времени подключения клиента к серверу информация в клиентской части должна обновляться либо автоматически, либо по команде от пользователя.

**Ход работы:**

Для выполнения поставленной задачи было реализовано два сервера в формате консоли (рис.1) и один клиент. После запуска сервера он переходит в режим ожидания подключения к нему клиентов. При подключении нового клиента (рис.2) сервер создает новый поток для обмена сообщениями с клиентом.

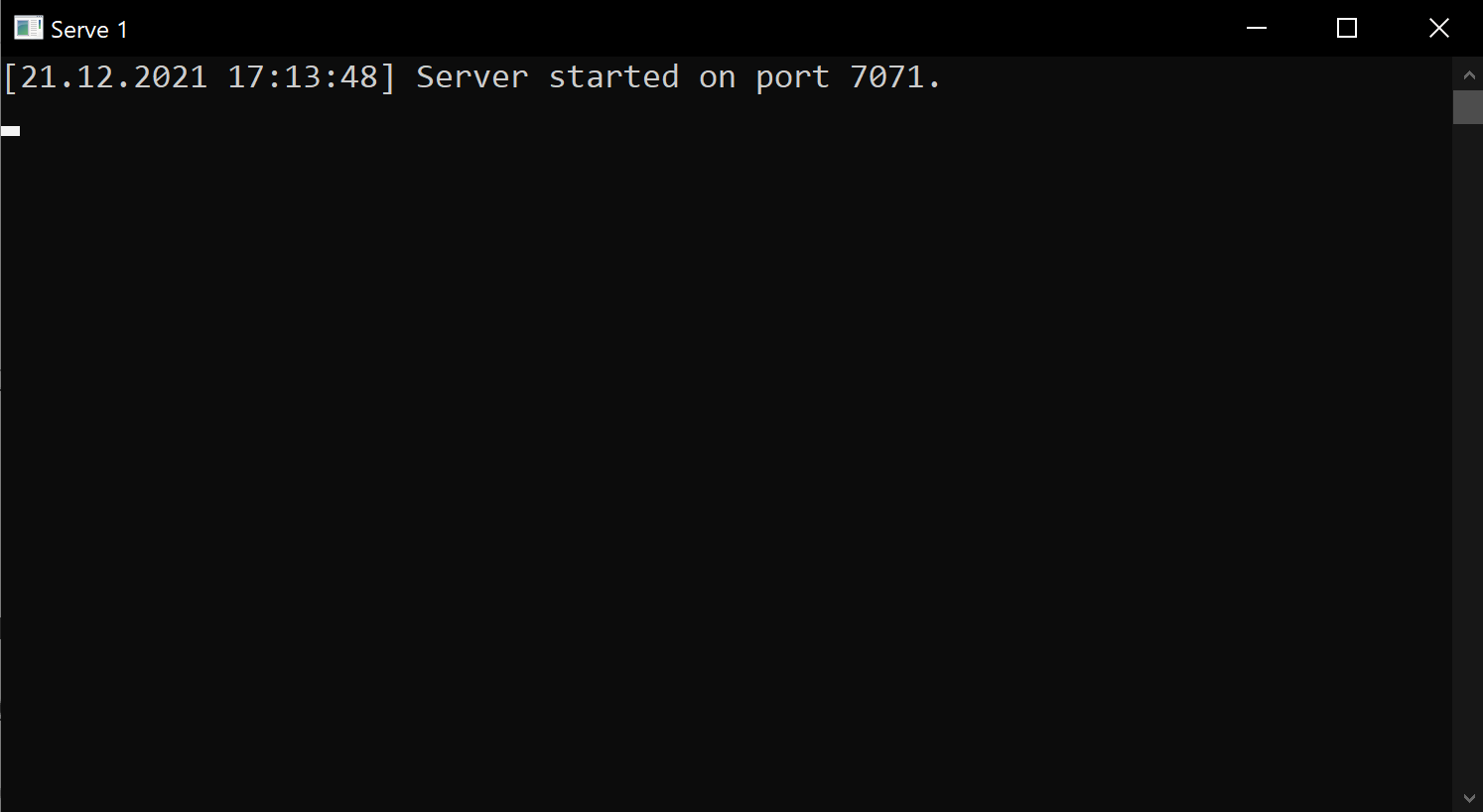


Рисунок 1. Скриншот работающего сервера №1.

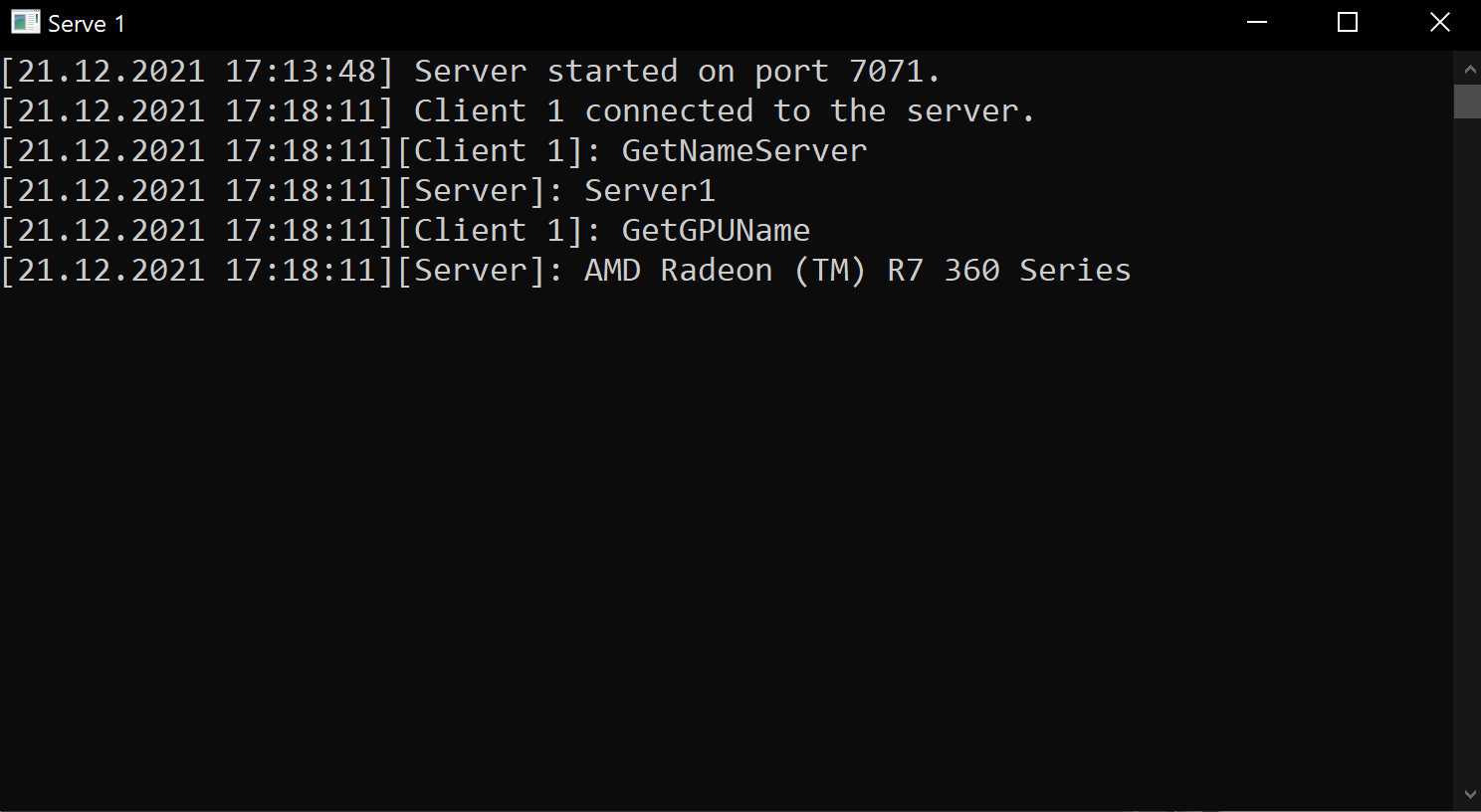


Рисунок 2. Реакция сервера №1 на подключение нового клиента.

Клиент может слать запросы серверу в виде текстового сообщение, которое переводится в массив байтов. Для получения сообщений от сервера и вывод их пользователю у приложения клиента есть 3 пользовательских интерфейса (рис.3,4,5).

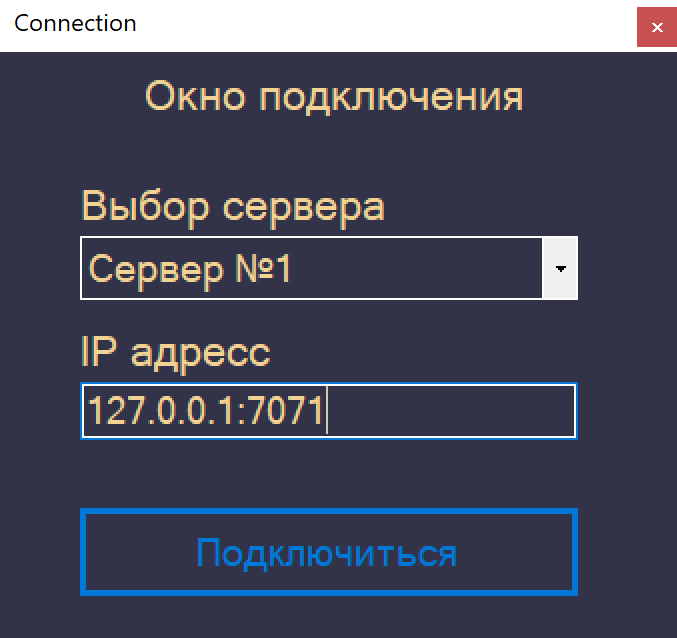


Рисунок 3. UI окна подключения.

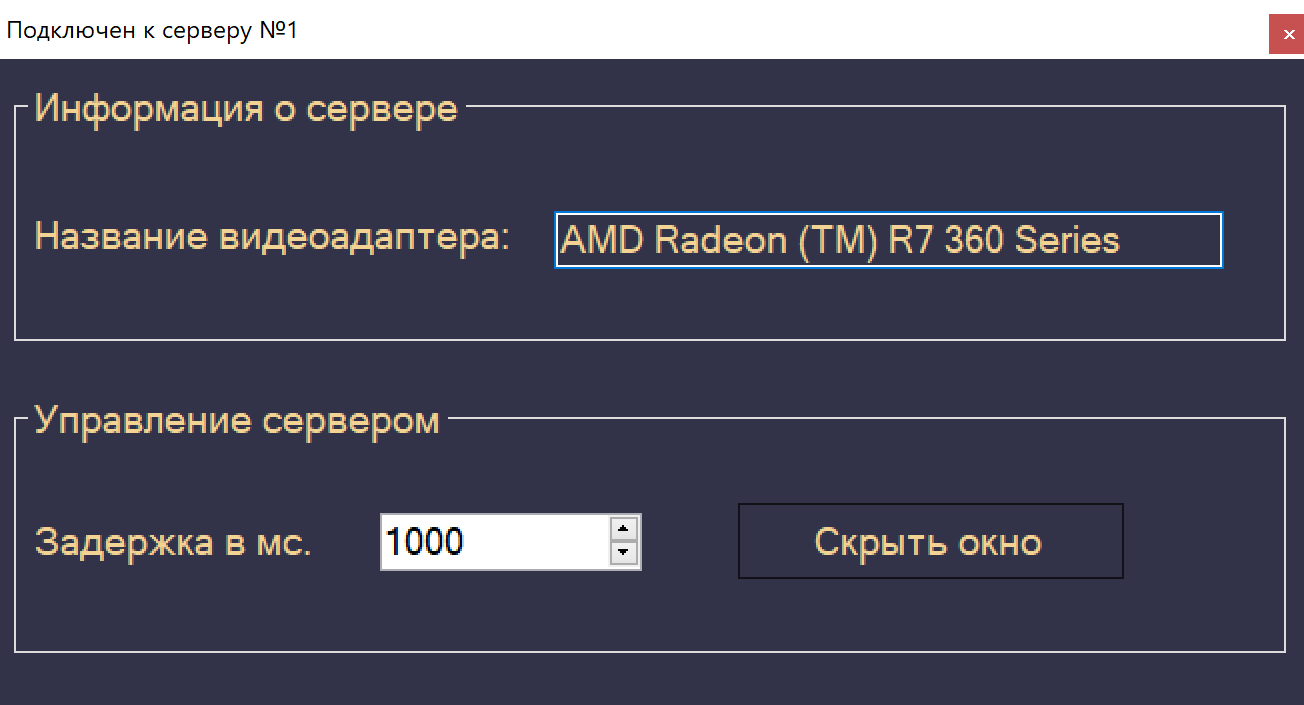


Рисунок 4. UI окна работы с сервером №1.

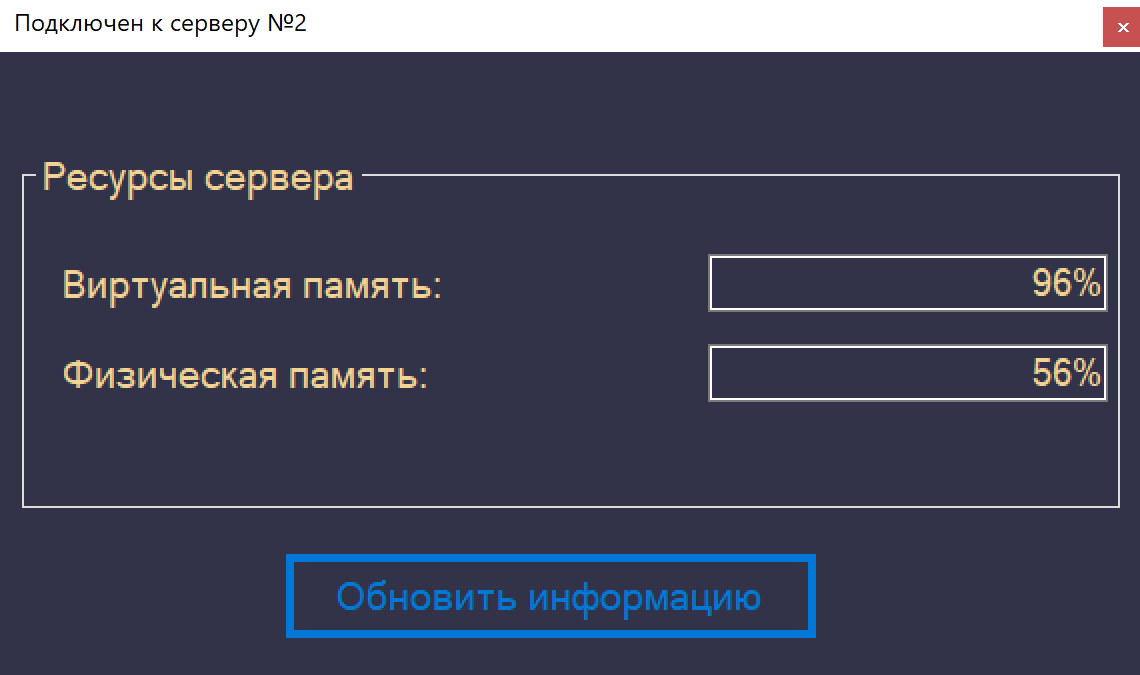


Рисунок 5. UI окна работы с сервером №2.

Так же клиент может автоматически делать запросы серверу на обновление информации о ресурсах сервера. В случае потери соединения с сервером, клиент сообщит об этом пользователю (рис 6.).

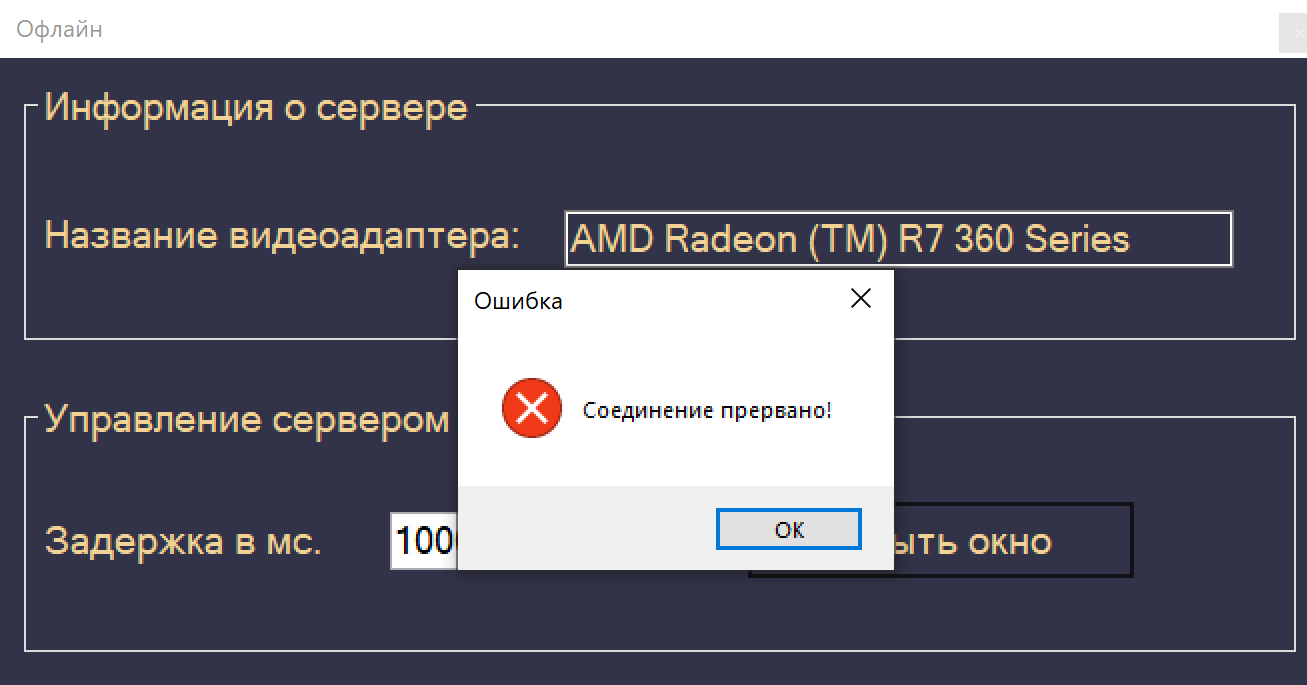


Рисунок 6. Вывод ошибки при разрыве соединения.

**Вывод:**

В ходе выполнения данной курсовой работы я познакомился с реализацией «клиент-сервер» приложения, а также с принципом работы сокета и то как происходит обмен сообщениями между клиентом и сервером.