1. Санкт-Петербургский Политехнический Университет Петра Великого
2. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 1**

**«Удаленный файловый менеджер с имперсонацией клиента»**

1. по дисциплине «Безопасности современных информационных технологий»
2. Выполнил
3. студент группы
4. 4831001/00302 Ташкент В.В.

<*подпись*>

1. Проверил

ассистент Соловей Р. С

<*подпись*>

1. Санкт-Петербург
2. 2022
3. **СОДЕРЖАНИЕ**

[1. Цель работы: 3](#_Toc121439972)

[2. Задача: 3](#_Toc121439973)

[3. Теоретические сведения: 4](#_Toc121439974)

[4. Ход Работы 6](#_Toc121439975)

[5. Ответы на контрольные вопросы 12](#_Toc121439976)

[6. Вывод 13](#_Toc121439977)

2. **Цель работы:**

Получить навыки работы с механизмом удаленного вызова процедур (RPC)

1. **Задача:**

Написать программу-сервер и программу-клиент, работающие под Windows 7-10. Сервер должен предоставлять доступ локальным и удаленным клиентам к файлам в своей файловой системе.

Требования:

- Statefull сервер;

- Сервер не должен быть интерактивным (интерфейс командной строки);

- Взаимодействие с клиентами должно осуществляться с помощью механизма RPC;

- При обслуживании клиента должна осуществляться его имперсонация;

- Пользователю должны предоставляться следующие операции: копирование указанного файла с клиента на сервер, загрузка указанного файла с сервера на клиента, удаление указанного файла на сервере;

- Имя файла передается в формате UNC.

В ходе работы должны быть проведены эксперименты с запуском сервера и клиентов под различными учетными записями с демонстрацией работы механизмов контроля доступа. Эксперименты должны показывать, что механизм имперсонации работает как при повышении, так и при понижении прав потока по отношению к процессу сервера.

1. **Теоретические сведения:**

Удаленный вызов процедур (англ. Remote Procedure Call (RPC)) – класс технологий, позволяющий компьютерным программам вызывать функции и процедуры в другом адресном пространстве (как правило, на удаленных компьютерах).

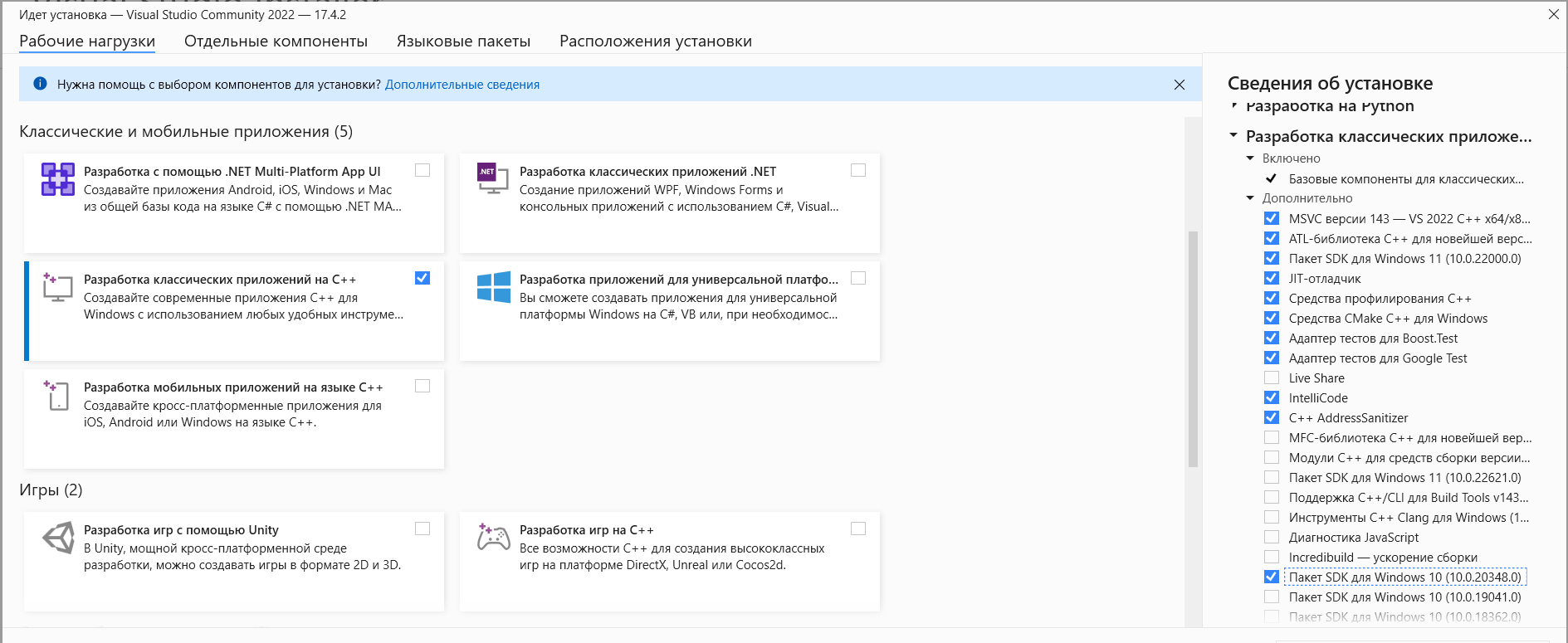
Реализуемый сервер RPC может быть statefull сервером– это значит, что клиент сначала должен идентифицироваться, после чего сервер его «запоминает» и клиент способен выполнять доступные ему действия на сервере до тех пор, пока сеанс связи между клиентом и сервером не будет разорван и инсцинирующей разрыв стороной не будет являться клиент. Таким образом, сеанс связи между клиентом и сервером не прерывается. Если же описанного выше функционала сервера не наблюдается, сервер называется stateless сервером.

Для разработки RPC клиент-серверного приложения с использованием языков C/C++ нужно сначала разработать интерфейс, а после этого реализовать этот интерфейс с помощью сначала сервера, а потом клиента. Это необходимо для того, чтобы клиент и сервер взаимодействовали по одним и тем же наборам правил. Распространенной практикой является разработка серверного и клиентского проектов параллельно, но интерфейс всегда разрабатывается первым.

Упомянутый интерфейс RPC должен описывать функции, которые будет реализовывать клиент-серверное приложение. Интерфейс состоит из названия интерфейса, некоторых атрибутов, необязательных определений типов и констант и набора объявлений процедур-прототипов. Все параметры объявляемых процедур должны быть снабжены указателями направления передачи данных [in], [out] или [in,out] (данные передаются для удаленной процедуры, возвращаются удаленной процедурой, либо данные как передаются, так и возвращаются).

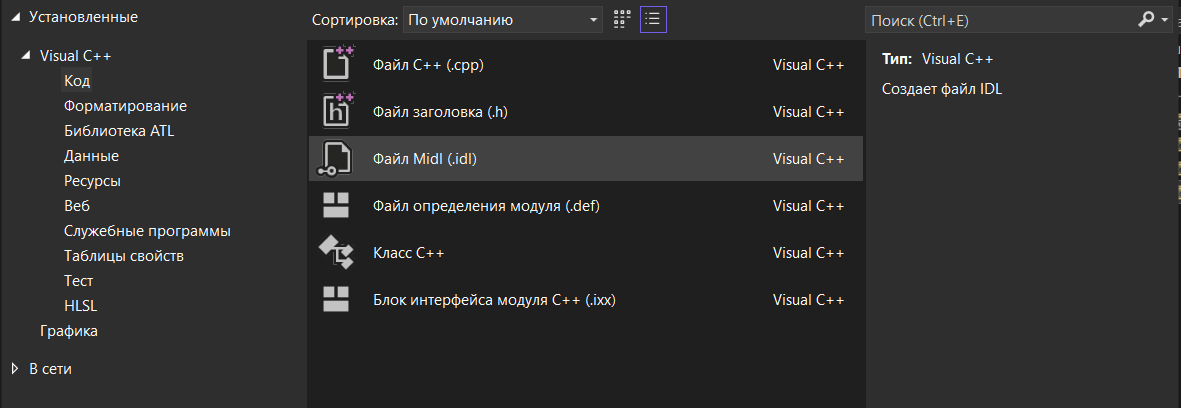
1. **Ход Работы**

В начале разработки RPC приложения необходимо разработать интерфейс взаимодействия сервера и клиента, а также присвоить ему UUID.

Разработка приложения велась в IDE Microsoft Visual Studio Community 2022 c установленным пакетом SDK.

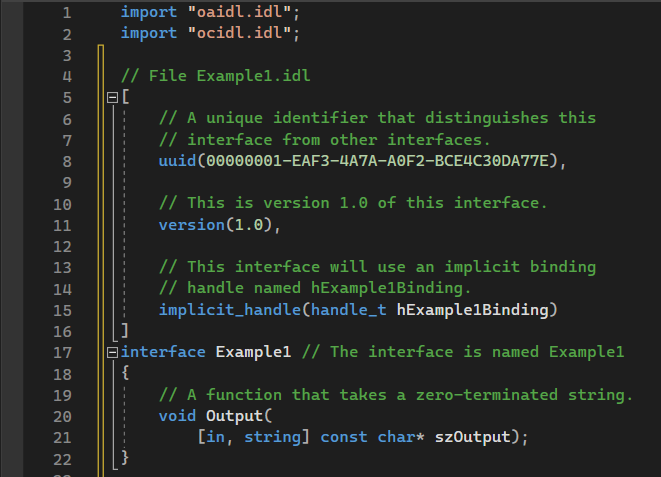
*Рис.1 Скриншот необходимых компонентов для разработки.*

Далее в среде разработки был создан необходимый файл MIDL(.idl), в котором и будет описан интерфейс.



*Рис.2 Скриншот создания .idl файла.*

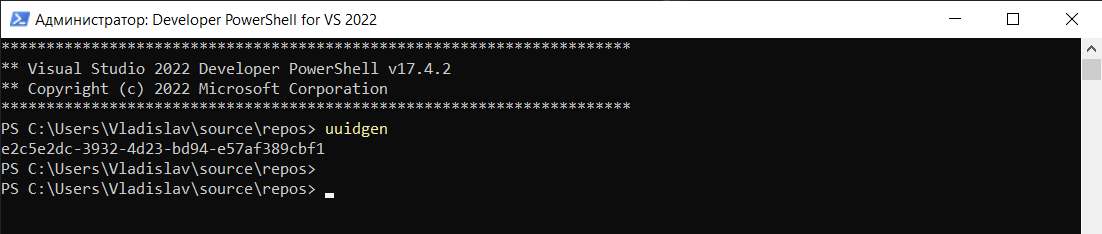
Затем был взят шаблон интерфейса: (источник <https://www.codeproject.com/Articles/4837/Introduction-to-RPC-Part-1>)



*Рис.3 Скриншот шаблона интерфейса*

После этого при помощи Developer PowerShell for VS22 командой uuidgen был получен уникальный uuid.

UUID (Universal Unique Identifier = Универсальный Уникальный Идентификатор) нужен для того, чтобы каждый созданный интерфейс мог быть определен уникальным образом.



*Рис.4 Получение uuid.*

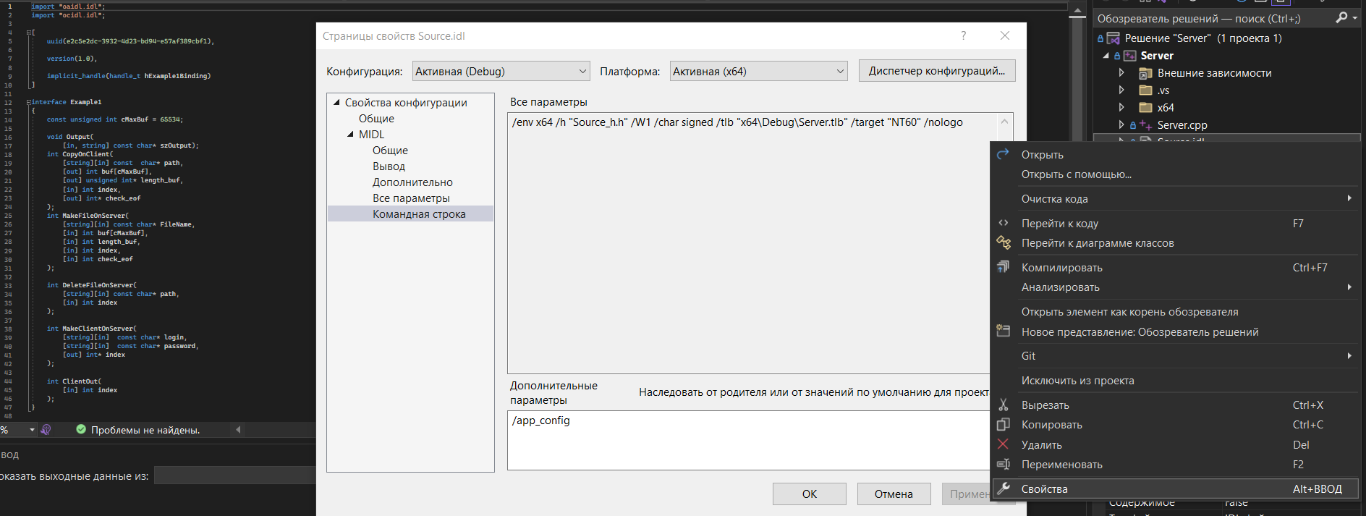
UUID был присвоен интерфейсу, а после этого были разработаны требуемые функции.

В таблице 1 представлены данные функции. Все параметры объявляемых процедур снабжены указателями направления передачи данных [in], [out] или [in,out] (данные передаются для удаленной процедуры, возвращаются удаленной процедурой, либо данные как передаются, так и возвращаются). Определение интерфейса содержится в текстовом файле с расширением .idl. Данный файл в результате компиляции создаёт один заголовочный файл и два файла на языке С – заглушки. Одна заглушка будет использоваться для сервера, другая – для клиента. Заголовочный файл будет использоваться в исходных кодах клиентского и серверного приложения.

Важная особенность в компиляции файла интерфейса – прописать в командной строке MIDL параметр “/app\_config”, потому как в противном случае компилятор не сможет создать необходимые файлы и выдаст об этом соответствующие ошибки:



*Рис.4 Важная особенность компиляции MIDL-файла в VS.*

**

*Рис.5 Прописывание параметра в командной строке MIDL.*

|  |  |  |
| --- | --- | --- |
| Название функции | Аргументы функции | Описание функции |
| Int makeClient() | [in][string] const unsigned char \*login – логин пользователя,  [in][string] const unsigned char\* password – пароль пользователя,  [in] int \*index – индекс пользователя | Функция создаёт клиента на сервере и записывает его в структуру Client. Также проверяет существует ли такой пользователь в системе и позволяет вызывающему потоку использовать контекст безопасности вошедшего в систему пользователя. Если клиента невозможно записать в базу данных на сервере, функция возвращает: -2, если ошибка связана с идентификацией, то функция возвращает: -1. В случае успеха функция возвращает: 1. |
| int sendFile( ) | [in][string] const unsigned char \*path – путь до файла,  [out][string] unsigned char buf[65534] – данные файла,  [in] int index – индекс клиента в базе сервера | Функция позволяет клиенту скачать необходимый файл, до которого указан путь path, с сервера к себе в систему, если у пользователя, через которого работает клиент, есть права на чтения файла. |
| int recvFile() | [in][string] const unsigned char \*FileName – Имя файла, который будет создаваться, и путь к нему, [in][string] const unsigned char \*buf – данные файла, [in] int index - индекс клиента в базе сервера | Функция позволяет загрузить файл с клиента на сервер в папку, указанную в FileName, если у пользователя, через которого выполняет, действия есть права на чтения на запись в данной папке то функция вернёт: 1, иначе -1. |
| int deleteFile() | [in][string] const unsigned char \*path - путь до файла, [in] int index - индекс клиента в базе сервера | Функция выполняет удаление файла в директории на сервере. Если он там имеется и, если у пользователя, через которого выполняет данное действие, есть права на удаление данного файла, то функция возвращает 1, иначе -1. |

1. Разработанный сервер является statefull, то есть работает по следующей схеме: клиент сначала идентифицируется, после чего сервер его «запоминает» и клиент способен выполнять доступные ему действия на сервере до тех пор, пока сеанс связи между клиентом и сервером не будет разорван и инсценирующей разрыв стороной не будет являться клиент.

2.1) На рисунке 6 представлен скриншот участка кода серверного приложения.

На данном рисунке можно увидеть, что максимально возможное количество клиентов, одновременно подключенных к серверу 10.

Переменная cMaxClient определяет это значение.

Данное значение можно изменить без потери работоспособности сервера.

2.2) Также тут представлена структура Client.

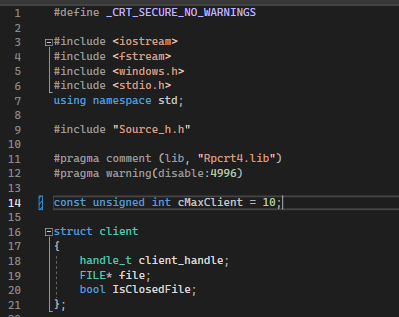
Данная структура имеет 3 поля:

1. Поле *handle\_t client\_handle* требуется для хранения дескриптора токена, который служит для олицетворением указанного пользователя.

2. Поле *file* – это указатель на файл, в который сервер записывает данные.

3. Поле *bool IsClosedFile* – это переменная тип bool, которая служит для того, чтобы понять был ли открыт файл или нет.

Также на рисунке 6 можно увидеть массив структур, в котором хранятся все клиенты, а также переменную-счётчик, которая показывает текущее количество подключённых клиентов.



*Рис. 6 Участок кода серверного приложения*

1. Следующим шагом был разработан клиент.

3.1) Для подключения клиенту необходимо ввести IPv4, порт, который прослушивает сервер, в данном случае это порт 9000, логин и пароль пользователя, от имени которого серверное приложение будет работать.

3.2) После ввода данных клиент дожидается ответа от сервера, а именно проверяет возвращаемое значении функции MakeClientOnServer().

Каждому клиенту присваивается уникальный индекс, который нужен работы с сервером в том случае, если к серверу подключено более одного клиента. После успешного завершения функции клиенту предлагается на выбор 4 действия – загрузка файла на сервер, загрузка файла с сервера, удаление файла и отключение.

3.3) В реализации первого и второго действия возникла такая сложность, как передача больших файлов с клиента на сервер и с сервера на клиент. Было принято решение передавать по 65534 символа (216 – 2).

1. **Ответы на контрольные вопросы**

**1. В чем различия между statefull и stateless серверами?**

В отличии от stateless сервера, statefull сервер работает по следующей схеме: клиент сначала идентифицируется, после чего сервер его «запоминает» и клиент способен выполнять доступные ему действия на сервере до тех пор, пока сеанс связи между клиентом и сервером не будет разорван и инсценирующей разрыв стороной не будет являться клиент. Таким образом, сеанс связи между клиентом и сервером не прерывается.

**2. Что такое имперсонация?**

Имперсонация — это концепт безопасности присущий Windows, что позволяет серверному приложению временно «быть» клиентом для доступа к охраняемому объекту. Имперсонализация состоит из трёх уровней: идентификация, позволяющая серверу проверять подлинность клиента, имперсонализация, позволяющая серверу работать от имени клиента, и делегация: то же, что и имперсонализация, только расширена на работу с удалёнными системами, с которыми связывается сервер.

**3. Что такое LPC? чем этот механизм отличается от RPC?**

Аббревиатура LPC расшифровывается как «Local Procedure Call» - локальный вызов процедур. LPC используется операционной системой для передачи сообщений между подсистемами через специальные объекты – порты. Механизм LPC использует клиент-серверную архитектуру и передает сообщения между клиентским и серверным процессами на одном компьютере, в то время как RPC позволяет работать и с удалёнными клиентами.

1. **Вывод**

В ходе выполнения данной работы были разработаны программа-клиент программа – сервер, работающие под Windows 7–10.

Были проведены эксперименты с запуском сервера и клиентов под различными учетными записями с демонстрацией работы механизмов контроля доступа.

Также был передан файл “Wireshark.exe” объёмом в 73.6 Мб. Переданный и полученный файлы были одинаковы.