

Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

ОТЧЕТ
по лабораторной работе

«Язык SQL-DDL»

Базы данных

Работу выполнил студент

группа 43501/3 Крылов И.С.

Работу принял преподаватель

_____ Мясов А.В.

Санкт-Петербург

2018

Содержание

1	Техническое задание	3
2	Прикладной протокол	4
2.1	Запрос	4
2.2	Ответ	6
3	Описание архитектур	6
3.1	Модуль UDP	7
3.2	Модуль TCP	8
3.3	Модуль remote_terminal_protocol	8
4	Особенности реализации	8
5	Результаты тестирования	8
6	Выводы	8

Техническое задание

Система терминального доступа

• Задание

Разработать клиент-серверную систему терминального доступа, позволяющую клиентам подсоединяться к серверу и выполнять элементарные команды операционной системы.

• Основные возможности серверного приложения

1. Прослушивание определенного порта
2. Обработка запросов на подключение по этому порту от клиентов
3. Поддержка одновременной работы нескольких терминальных клиентов через механизм нитей
4. Проведение аутентификации клиента на основе полученных имени пользователя и пароля
5. Выполнение команд пользователя:
 - > ls – выдача содержимого каталога
 - > cd – смена текущего каталога
 - > who – выдача списка зарегистрированных пользователей с указанием их текущего каталога
 - > kill – Привилегированная команда. Завершение сеанса другого пользователя
 - > logout – выход из системы
6. Принудительное отключение клиента

• Клиентское приложение должно реализовывать следующие функции:

1. Установление соединения с сервером
2. Посылка аутентификационных данных клиента (имя и пароль)
3. Посылка одной из команд (ls, cd, who, kill, logout) серверу
4. Получение ответа от сервера
5. Разрыв соединения
6. Обработка ситуации отключения клиента сервером или другим клиентом

• Настройки приложений

Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени удалённого терминального сервера и номера порта, используемого сервером. Разработанное серверное приложение должно хранить аутентификационные

данные для всех пользователей, а также списки разрешенных каждому пользователю команд.

- **Методика тестирования**

Для тестирования приложений запускается терминальный сервер и несколько клиентов. В процессе тестирования проверяются основные возможности сервера по параллельной работе нескольких клиентов, имеющих различные привилегии (списки разрешенных команд). Проверяется корректность выполнения всех команд в различных ситуациях.

Прикладной протокол

Для реализации технического задания был разработан прикладной протокол передачи данных.

Запрос

Протоколом задаётся два формата запроса для взаимодействия клиента с сервером:

- запрос аутентификации с помощью пары логин:пароль 2.1
- запрос выполнения определённой команды 2.2

Login:Password:	Package Index
[0 - 507]	[508 - 511]

Таблица 2.1: Формат запроса аутентификации

Message Length	Command Descriptor	Command Parameters	Package Index
[0 - 3]	[4]	[5 - 507]	[508 - 511]

Таблица 2.2: Формат запроса выполнения команды

Оба запроса имеют одинаковый размер - 512 байт.

В таблице формата аутентификации 2.1:

- Login:Password - поле, содержащее передаваемые клиентом логин и пароль, необходимые для подключения к серверу. Протоколом задается формат ввода пары следующим образом:

Поле занимает 508 байт, задавая тем самым максимально возможную длину пары логин:пароль равной 508 символам.

[login] : [password] :

- Package Index - поле, хранящее индекс пересылаемого пакета. Благодаря наличию этого поля протокол гарантирует последовательный приём пакетов (защита от перемешивания). Так же, контроль номера пакета усложняет возможность атаки с имитацией адреса клиента посторонними. Длина поля - 4 байта, что позволяет обеспечить до 9999 последовательных запросов клиента серверу. В условиях технического задания данная продолжительность взаимодействия клиента с сервером более чем достаточна. При превышении этого значения клиент будет отключён от сервера. Тем самым гарантируется пресечение чрезмерно активного трафика, исходящего от клиента, который может свидетельствовать о зловредном характере запросов клиента

В таблице запроса выполнения команды 2.2:

- Message Length - длина параметров команды. В случае если команда не имеет параметров, данное поле заполняется нулями. Под данное поле выделено 4 байта.
- Command Descriptor - целое число - дескриптор команды, однозначно определяющий требуемую команду:
 - 1 – выдача содержимого каталога ls
 - 2 – смена текущего каталога cd
 - 3 – выдача списка зарегистрированных пользователей с указанием их текущего каталога who
 - 4 – Привилегированная команда. Завершение сеанса другого пользователя kill
 - 5 – выход из системы logout

Список поддерживаемых протоколом команд ограничиваются пятью, вследствие чего под дескриптор задачи выделено поле в 1 байт. Использование схемы взаимодействия клиента с сервером посредством передачи дескриптора команды снимает с сервера задачу проверки правильности введённой с консоли пользователем команды, тем самым минимизируя количество пересылаемых пакетов и ускоряя работу сервера.

- Command Parameters - поле длиной 499 байт, содержит параметры команд: cd и kill
- Package Index - поле, хранящее индекс пересылаемого пакета. Идентично одноимённому полю запроса аутентификации

Ответ

В соответствии с имеющимися форматами запросов, протокол определено два формата ответа:

- ответ аутентификации
- ответ выполнения команды

Ответ аутентификации - пакет длиной 1 байт, содержащий код результата аутентификации:

- 0 - неверная пара логин:пароль, отказ в доступе
- 1 - аутентификация прошла успешно, получен доступ к удалённому терминалу
- 2 - пользователь с данным логином уже вошёл в систему с другого устройства, отказ в доступе
- * - неверный формат ввода, отказ в доступе. Имеет специальное назначение для ответа выполнения команды

Ответ выполнения команды - пакет длиной 8192 байт, содержащий результат выполнения команды на удалённом терминале.

Исключительной ситуацией является попытка запроса на выполнение команды клиентом, сеанс которого был завершён другим пользователем, обладающим правами администратора. Такая ситуация возможна исключительно при использовании протокола UDP, вследствие того, что отключение клиента происходит по таймауту запроса ответа от сервера. В таком случае в качестве ответа на запрос посылается одnobайтовый ответ аутентификации "*" обёрнутый в формат ответа выполнения команды с помещением на первый (с индексом [0]) байт. Особенности устройства файловых систем ОС Windows и UNIX-подобных исключает возможность использования символа "*" в названиях файлов и директорий. Протоколом же запрещено создание логинов, начинающихся с символа "*". Тем самым исключается наличие и ошибочное обнаружение символа "*" в стандартном ответе выполнения команды.

Описание архитектур

Приложение было разбито на несколько модулей, реализующих имплементацию протокола с TCP и UDP, а также саму реализацию протокола.

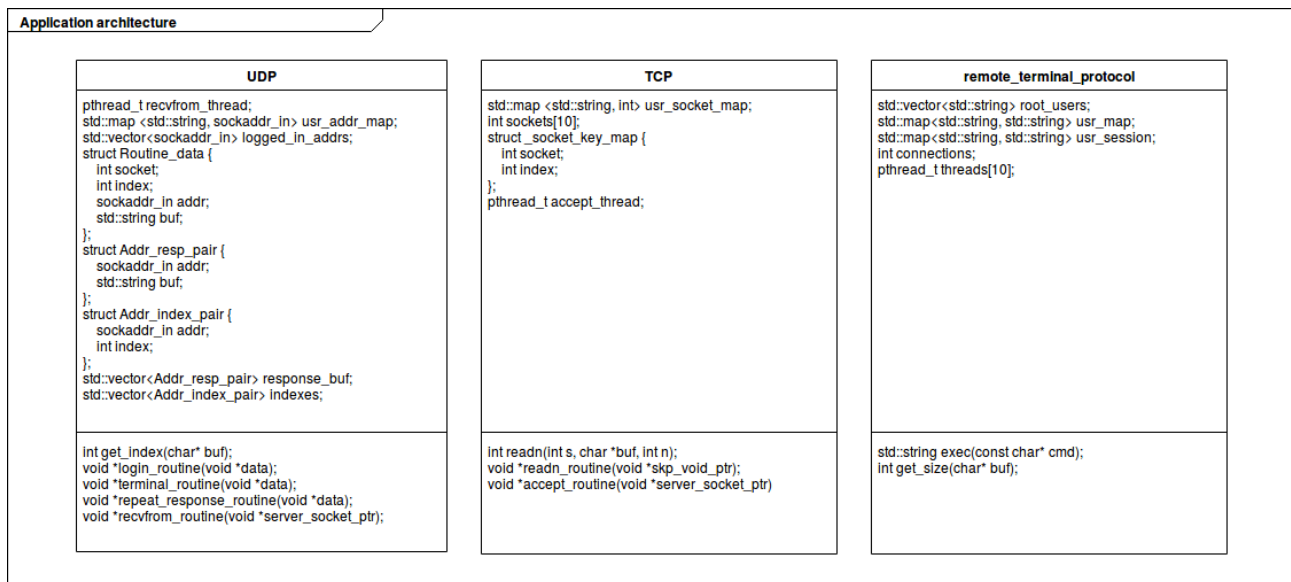


Рис. 3.1: Диаграмма модулей

Модуль UDP

- `recvfrom_thread` - дескриптор потока приёма всех входящих запросов
 - `usr_addr_map` - структура данных, хранящая имя пользователя и соответствующий данному имени адрес и порт подключения. Необходима для определения адреса отправки ответа по имени пользователя
 - `logged_in_addrs` - список адресов подключенных пользователей. С помощью него можно определить - входящий пакет поступает от нового подключения или является пакетом сессии одного из уже подключенных пользователей
 - `Routine_data` - структура данных, содержащая основную информацию для отправки ответа на входящий запрос. Используется для удобной транспортировки данных между потоками
 - `Addr_resp_pair` - структура данных, объединяющая буфер ответа с адресом пользователя
 - `Addr_index_pair` - структура данных для хранения данных нумерации пакетов в соответствии с адресом пользователя
 - `response_buf` - буфер, хранящий предыдущий ответ для заданного адреса. Данные хранятся парами - каждому адресу ставится в соответствие определенный буфер с ответом. Таким образом можно однозначно получить предыдущий ответ для конкретного адреса, в случае повторного поступления пакета запроса
 - `indexes` - массив пар адрес:номер пакета. Обеспечивает хранение индексов пакетов индивидуально для каждой сессии
- > `get_index` - функция распаковки значения индекса принятого запроса

Модуль TSP

- `usr_socket_map` - структура данных для хранения пар `имя_пользователя:дескриптор_сокета`. Позволяет соотнести заданного пользователя к конкретному сокету
- `sockets` - массив дескрипторов активных сокетов
- `socket_key_map` - структура данных для удобного передачи информации между потоками
- `accept_thread` - дескриптор потока приёма всех входящих запросов

Модуль `remote_terminal_protocol`

- `root_users` - список имён пользователей, обладающих привилегированными правами
- `usr_map` - пары `логин:пароль` зарегистрированные для доступа к удалённому терминалу
- `usr_session` - пара `имя_пользователя:текущая_директория`
- `connections` - счётчик активных подключений к серверу. Контролирует входящую нагрузку
- > `exec` - функция вызова входящей команды в терминале сервера
- > `get_size` - функция распаковки значения фактической длины параметров принятого запроса

Особенности реализации

Результаты тестирования

Выводы