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

_____ Мяснов А.В.

Санкт-Петербург

2018

Содержание

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 вех пользователей, а также списки разрешенных каждому пользователю команд.

- **Методика тестирования**

  Для тестирования приложений запускаетсятерминальный 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 команда 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 длинну 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 поторонними. 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, вследствии чего под дескриптор задачи выделено поле в 1 байт. Использование схемы взаимодействия клиента с сервером посредством передачи дескприптора 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 длинной 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 длинной 1 байт, содержащий код результата аутентификации (Таблица 2.3):

- 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 выолнения команды

Ответ выполнения команды - пакет длинной 8192 байт, содержащий результат выполнения команды на удалённом терминале (Таблица 2.4).

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 однобайтовый 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 ислючается наличие и ошибочное обнаружение символа "*"в стандартном ответе выолнения команды.

Response
[0 - 1]

Таблица 2.3: Формат ответа аутентификации

Response
[0 - 8191]

Таблица 2.4: Формат ответа выполнения команды

Описание архитектур

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

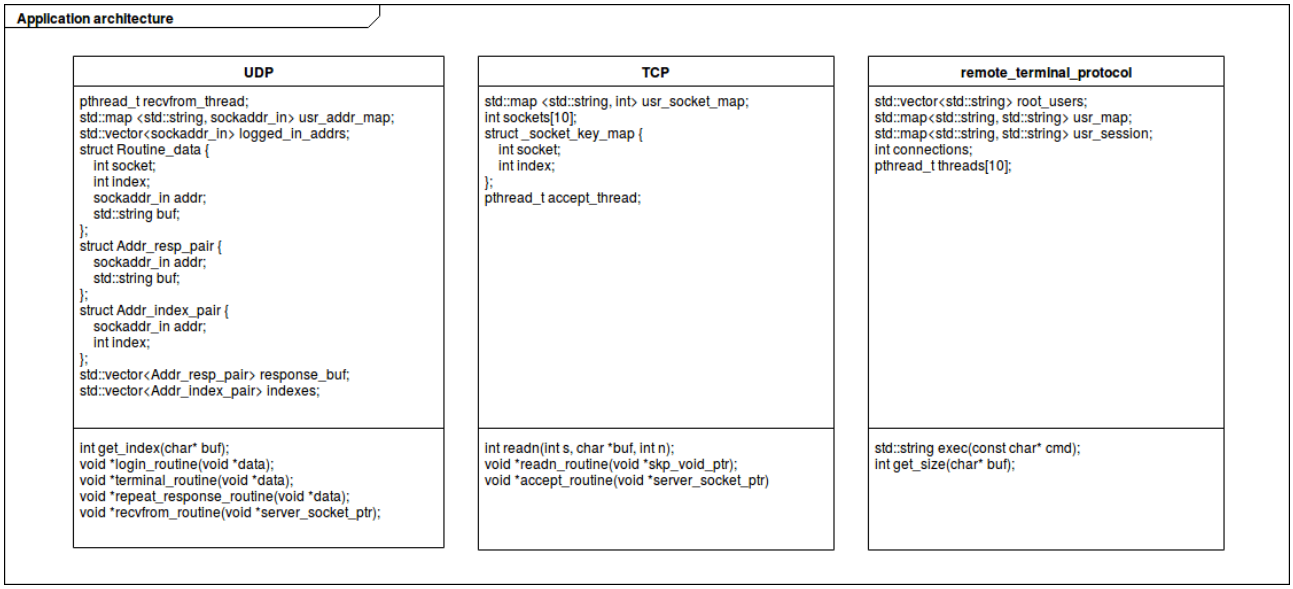

Рис. 3.1: Диаграмма модулей

Модуль UDP

- recvfrom_thread - дескриптор потока приёма всех входящих запросов
- usr_addr_map - структура данных, хранящая имя пользователя и соответсвующий 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 - входящих пакет поступает от нового подключения или является пакетом сессии одного из уже подклченных пользователей
- Routine_data - структура данных, содержащая основную информацию для отправки ответа на входящих запрос. Используется для удобной траспортировки 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 пользовтелся
- 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 адрса, в случае повторного постуления 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 TCP

- usr_socket_map - структура данных для хранения пар имя_ползователя: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 привилигированными 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 реализацции

Приложение на основе TCP

Серверное приложение

При запуске серверного приложения, в главном потоке main thread создаётся сокет и устанавливается на прослушку порта 7500. После чего порождается поток принятия новых соединений accept thread с функцией обработчиком accept_routine(), куда передаётся дескприптор созданного соке-

та. Accept_routine() в бесконечном цикле принимет новые подключеня, порождая новые потоки readn thread с назначенной функцией-обработчиком readn_routine(). Readn_routine() в бесконечном цикле принимает входящие данные с помощью функции readn, которая гарантирую целостность принятых данных. Затем, в том же потоке происходит аутентификация пользователся, отправка дескриптора результата входа в систему и в случае успешного входа в систему - выполнение переданной команды с последующим отправлением результата. По окнчанию соединения, все потоки завершаются. Это гарантируется вызовом функции pthread_join(). Чтобы избежать конфликтов доступа и перезаписи данных, используются мьютексы. Примерное отображение процесса попрождения и завершения потоков отображено на Рис.4.1.

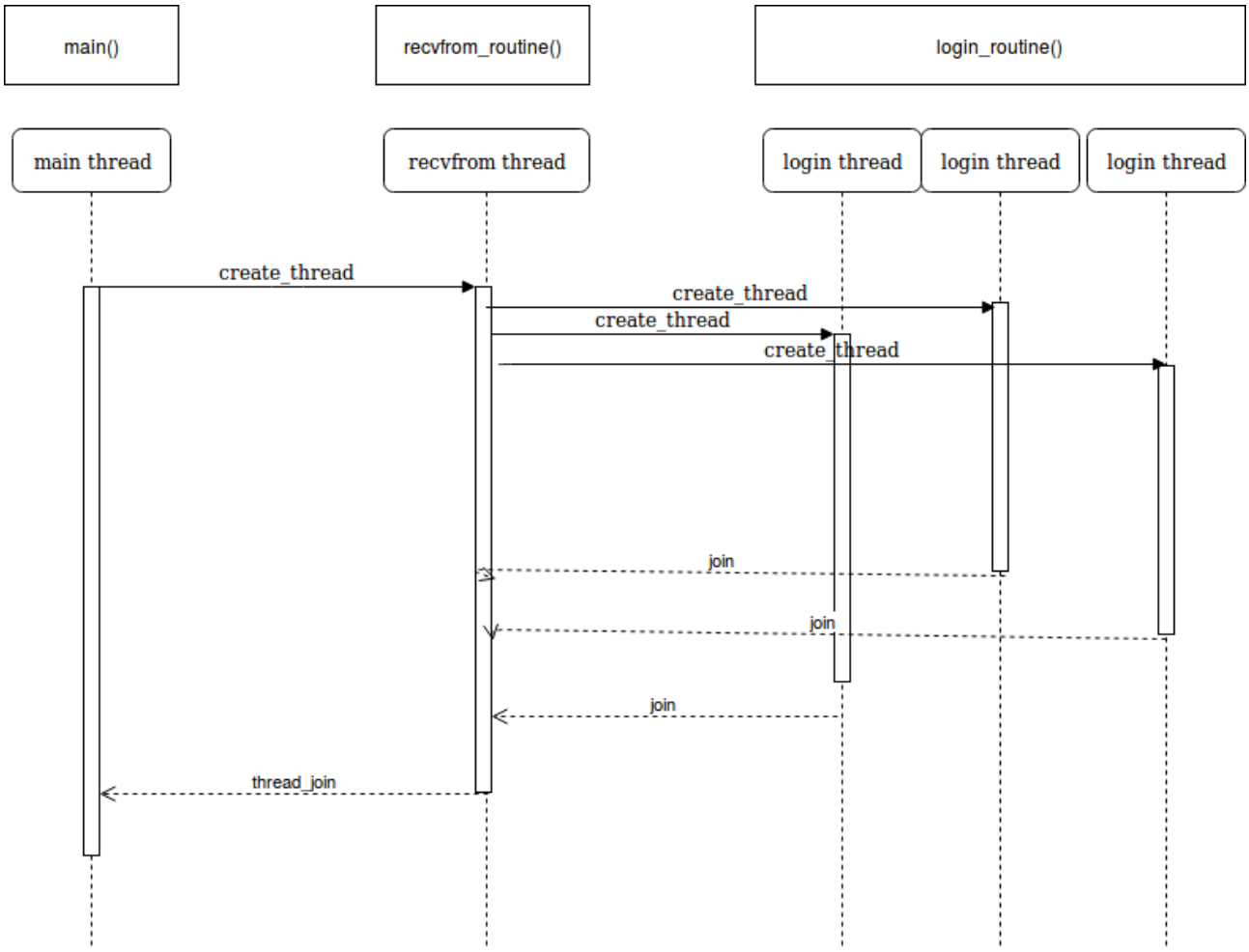

Рис. 4.1: Диаграмма примера потоков в серверном приложении TCP

Клиентское приложение

Клиентское приложение является однопоточным. Создаётся сокет, по которому происходит общение клиента с сервером. Адрес сервера и номер порта для подключения задаются явно. Взаимодействие с сервером происходит последовательно в двух циклах. Первый цикл бесконечный с условием выхода

по положительному ответу сервера на запрос доступа к удалённому терминалу. Во втором бесконечном цикле происходит обработка пользовательского ввода с клавиатуры, упаковка передаваемых данных в соответстви с протоколом 2.2 и отправка посредством команды send с последующим принятием ответа через функцию readn(). Выход из второго и цикла и завершение работы клиента происходит либо при вводе пользователем команды logout, либо по решению администратора, либо по принудительной остановке работы сервера.

Приложение на основе UDP

Серверное приложение

В приложении с использованием UDP аналогично TCP создаётся сокет, привязываеммый к конкретному порту, в нашем случае к порту 7500. В отличие от TCP никакого принятия новых соединений не происходит. В попрождённом потоке recvfrom thread в функции recvfrom_routine напрямую принимаются UDP дэйтаграммы. Определение необходимого сеанса происходит по адрему отправителя. После принятия дэйтаграммы, определения её принадлежности к одной из активных сессий, а также проверки корректности индекса пакета, происходит попрождение нового потока-обработчика с выполнением соответсвующей запросу функцией: login_routine или terminal_routine. Для ускорения работы сервера, в случае повторного приёма одного и того же пакета, порождается поток с фуекцией обработчик repeat_response_routine() - дублирующей предыдущий ответ по данному запросу. Пример порождения и завершения работы потоков представлен на Рис.**??**.

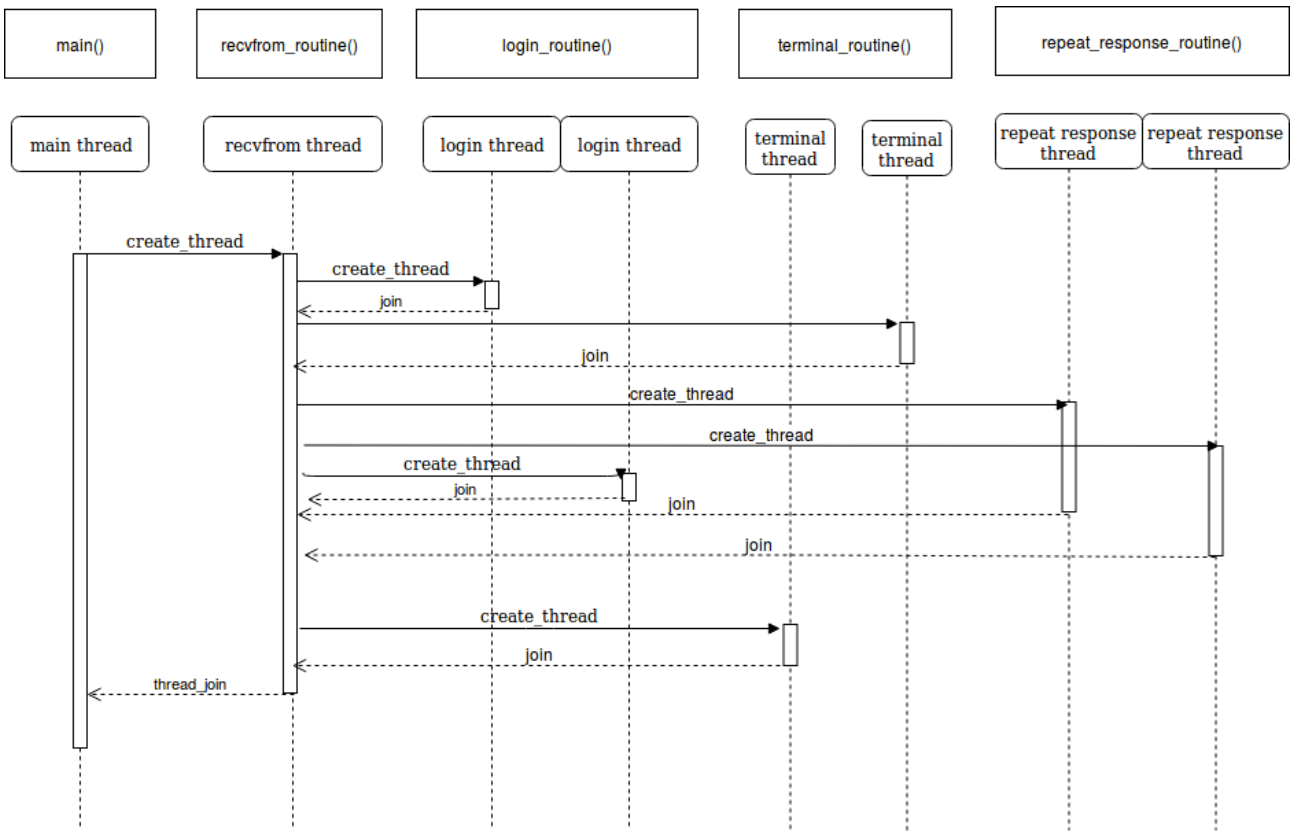

Рис. 4.2: Диаграмма примера потоков в серверном приложении UDP

Клиентское приложение

Результаты тестирования

Выводы