Отчет по лабораторной работе

Технологии компьютерных сетей

Работу выполнил:студент группа 43501/3 Киселёв Антон

Описание протокола взаимодействия клиентской и серверной части

Задание: "Система терминального доступа"

Система терминального доступа представляет сервис клиенту для осуществления команд в терминале на удаленном компьютере. Система состоит из двух частей:

1) **Серверная часть** - находится на галвном компьютере, предоставляющем свой терминал для пользования другим компьютерам и осуществления операций согласно правам доступа клиента к данному компьютеру.

2) **Клиентская часть** - является внешним компьютером, получающим возможность обращения к компьютеру на сервере для осуществления необходимых операций.

Сервер предоставляет право доступа к терминалу для нескольких подключенных пользователей. В его основные задачи входит проверка пользователя и пароля, предоставление списка доступных команд и проверка их ввода клиентом, обнаружение и исправление ошибок.

Клиент имеет право запускать только те команды, которые указаны в списке допустимых, а также осуществляет мониторинг ошибок при соединении с сервером.

Протокол взаимодействия опишем отдельно для клиента и для сервера и для разных реализаций TCP и UDP.

Описание взаимодействий по протоколу TCP

Описание протокола взаимодействия серверной части

Серверная часть инициализируется запуском с указанием используемого TCP-порта. После чего сервер переводится в режим прослушивания порта. Опишем как работает протокол для взаимодействия с один клиентом. Сервер осуществляет многопоточное соединение с различными подключающимися клиентами. Сервер ожидает подключения клиента и получения первого сообщения с указанием имени пользователя и пароля. Имя пользователя и пароль присылаются в следующем формате:

<username>_<password>

Синтаксический обработчик прочтет имя пользователя и пароль и запишет их в переменные. Далее сервер начнет проводить аутентификацию имени пользователя и пароля, аутентификация будет происходить как на серверной, так и на клиентской части. Сначала сервер обратиться к файлу, содержащему имена и пароли всех компьютеров, у которых есть доступ к удаленному терминалу. Данный файл имеет формат .txt и содержит данные о пользователе, пароле и используемых командах в следующем формате:

Идентификатор	Значение
Имя пользователя	Антон
Пароль	1234
Доступные команды терминала	cd ls mkdir
Текущий каталог	/home/anton
Имя пользователя	User
Пароль	5678
Доступные команды терминала	cd ls

Доступные команды терминала	cd ls
Текущий каталог	/home/anton
Имя пользователя	Root
Пароль	0912
Доступные команды терминала	cd ls rm
Текущий каталог	/home/anton

Данная таблица представлет структуру файла для трёх пользователей.

Сервер произведет сравнение данных, полученных от пользователя с данными в файле: при положительном результате клиент будет переведен на второй этап аутентификации пользователя и пароля. Клиенту будет отправлено число для пересчета хэш-функции имени пользователя и пароля. Сервер также будет осуществлять подсчет хэш-фугкций имени пользователя и пароля. Клиент, имеющий доступ к терминалу владеет специальной функцией подсчета хэш-кода, как и сервер. Для подсчета кода взята функция Ly.

Ly- функция

```
unsigned int HashLy(const char * str)
{
    unsigned int hash = 0;

    for(; *str; str++)
        hash = (hash * 1664525) + (unsigned char)(*str) + 1013904223;

    return hash;
}
```

После подсчет хэш-функций клиентом и сервером, сервер осуществит сравнение полученных данных и в случае полодительной проверки подлинности предоставит терминал клиенту на право пользования, в противном случае "обрубит соединение". В случае положительной проверки пользовательских данных клиент получит сообщение: **"Добро пожаловать!"** Далее клиенту будет отпрален список всех возможных команд для использования.

Сервер будет осуществлять запуск команд от клиента, извлекая их из сообщений. Каждое сообщение для удобства будет помещаться в буфер типа char размером до 256 байт.

Каждую полученную команду сервер будет проверять по файлу всех команд для данного пользователя.

Команды будут вызываться в терминале с помощью функции **popen** из библиотеки **stdio.h**. Результат выполнения будет записываться в бубчер размером 1024 байта и отправляться клиенту.

Обработка сервером ошибок

В случае если на сервере появилась ошибка при запуске команды из терминала, то клиенту будет отправлено соответствующее сообщение об ошибке, а сервер приостановит процесс выполнения команд и отключит клиента.

Ошибки соединения контролируются пользователем на компьютере клиента, в случае неудачного присоединения пользователь сам совершит переприсоединение к серверу.

Описание протокола взаимодействия клиентской части

Клиентская часть осуществляет запрос на главный компьютер с целью получить терминал компьютера в пользование. Все сообщения посылаемые клиентом помещаются в символьный буфер размером 256 байт, за исключением сообщения от команды who (1024 байта). Запрос на подключение к терминалу сервера осуществляется автоматически: клиент инициирует соединение, сервер запрашивает у него имя пользователя и пароль, которые далее отправляются. После сервер отправляет клиенту случаную последовательность символов. Полученное сообщение конкатенируется с

последовательность символов. Полученное сообщение конкатенируется с паролем и для них просчитывается хэш-код. Клиент отправляет свой вычисленный хэш и пользователю останется только дождаться сообщения о готовности использования терминала: "Добро пожаловать!".

Пользователю достаточно вводить используемые команды в командной строке при наличии подключения. Все команды отправляются в виде 256-байтного сообщения, введенного с консоли. Клиент принимает сообщения от сервера в буфер размером 256 либо 1024 байта.

На различные исключительные ситуации расставлены обработчики, осуществляющие обработку и исправление ошибок.

Список вводимых клиентом команд:

- ls: получение списка файлов и директорий в текущей директории;
- cd: произвести переход к другой папке;
- who: запрос на получение списка всех пользователей и их текущих директорий;
- logout: завершение сеанса;

Многопоточное взаимодействие для связи с несколькими клиентами.

Для осуществления многоклиентского взаимодействия используются потоки, это удобно посколько с каждым клиентом устанавливается логическое соединение, что удобно, потому что каждый поток будет защищен от другого клиента. Сервер при постоянном режиме прослушивает необходимый порт и при подключении клиента, выделяет ему отдельный поток и переводит себя в стандартный режим общения с клиентом.

Описание взаимодействий по протоколу UDP

Описание клиентской части

Клиент UDP является управляющим и инициирующем всю систему, порядок работы настраивает именно он, поскольку сервер выполняет функцию обработчика команд, то есть в зависимости от присланного клиентом сообщения сервер будет передавать одно или иное сообщение.

Клиент в данном варианте инициализирует соединение. Пользователь вводит с консоли любую команду и отправляет её серверу и подсоединяется к нему. После чего в обычном порядке происходит стандартное взаимодействие. Единственным отличием от TCP-взаимодействия является то, что каждая команда, которую собирается набрать пользователь предварительно передается с ключевым значением. Всего имеются следующие ключевые слова:

1. My data - говорит серверу, что далее будет осуществляться аутентификация клиента.

2. commands - говорит серверу, что далее будет осуществляться переслыка команд терминала.

3. иное - говорит серверу инициировать общение с клиентом.

После передачи каждой такой команды осуществляется соответствующее действие по аутентификации, или передачи команд аналогично протоколу UDP.

Описание серверной части

Сервер UDP содержит необходимую логику для взаимодействия с несколькими клиентами в отличии от протокола TCP, так как данный протокол не устанавливает логического соединения. Сервер находится в постоянной работе по принятию дейтаграммных сообщений и в зависимости от присланного сообщения осуществляет те или иные действия по взаимодействию.

Сервер дожидается специальных команд, после чего каждую принятую он обрабатывает на соответствие и переводит себя в необходимое состояние для

работы. Каждый блок для специальных команд полностью аналогичен взаимодействию по протоколу TCP.

Многоклиентское взаимодействие

В данной задаче, так как не предусматрвиается установления логического соединения, то удобна организация асинхронного сервера, реагирующего на сообщения. Поэтому реализация такой задачи была осуществлена с помощью логики, а не потоков.

Тестирование всего приложения

Для тестирования была проведена необходимая отладка отдельных частей и компонентов. Были проверены вводимые данные и производился поиск основных ошибок, связанных с число передаваемых байт, аутентификация чтение из файла и выполнение команд в терминале.

Протокол TCP тестировался на создание логического соединения, на осуществления правильной аутентификации и на передачу команд, предусмотрены некоторые исключения, касающиеся неправильно введенных данных и неправильно подсчитанных хэш-функций.

Протокол UDP тестировался на ввод специальных команд и на выполнение основных функции по выполнению команд в терминале и чтению файлов.

Оба протокола были протестированы подключением 2-3 клиентов и проверкой правильного взаимодействия сервера с обоими клиентами.