УДК 004.41 DOI: 10.31799/2077-5687-2024-2-25-30

ВАРИАНТЫ РЕАЛИЗАЦИИ КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ TCP СОКЕТОВ В ОС LINUX

Н. Н. Петров, Я. М. Родичев, Т. Л. Прокофьева
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Статья представляет обзор вариантов проектирования и реализации клиент-серверных приложений на основе сокетов TCP/IP в системах Linux. Рассматриваются аспекты реализации систем чата и логистического центра, затронуты темы процессов аутентификации, авторизации и регистрации, а также использование многопоточности. Обсуждаются средства языка C в ОС Linux и подчеркиваются ключевые особенности созданных клиент-серверных систем.

Ключевые слова: клиент, сервер, сокеты, TCP/IP, многопоточность.

Для цитирования:
Петров, Н. Н. Варианты реализации клиент-серверных приложений с использованием TCP сокетов в ОС LINUX / Н. Н. Петров, Я. М. Родичев, Т. Л. Прокофьева // Системный анализ и логистика. – 2024. – № 2(40). – с. 25 – 30. DOI: 10.31799/2077-5687-2024-2-25-30.

OPTIONS FOR IMPLEMENTING CLIENT-SERVER APPLICATIONS USING TCP SOCKETS IN LINUX OS

N. N. Petrov, I. M. Rodichev, T. L. Prokofieva
St. Petersburg State University of Aerospace Instrumentation

The article provides an overview of options for designing and implementing client-server applications based on TCP/IP sockets in Linux systems. Aspects of the implementation of chat and logistics center systems are considered, the topics of authentication, authorization and registration processes, as well as the use of multithreading are touched upon. The C language tools in Linux OS are discussed and the key features of the created client-server systems are emphasized.

Keywords: client, server, sockets, TCP/IP, multithreading.

For citation:
Petrov, N. N. Options for implementing client-server applications using TCP sockets in LINUX OS / N. N. Petrov, I. M. Rodichev, T. L. Prokofieva // System analysis and logistics. – 2024. – № 2(40). – p. 25 – 30. DOI: 10.31799/2077-5687-2024-2-25-30.

Введение

В современном мире продолжает возрастать влияние клиент-серверных технологий в различных сферах жизни общества. В основе этих технологий зачастую используются TCP или UDP сокеты. Целью данного исследования является выделение некоторых особенностей проектирования и реализации приложений, использующих клиент-серверное взаимодействие, на основе сокетов TCP/IP в семействе операционных систем Linux [1].

1. Взаимодействие клиента и сервера

В сфере информационных технологий, где клиент-серверное взаимодействие становится неотъемлемой частью повседневной деятельности, обеспечение безопасности и эффективности передачи данных играет ключевую роль. Процедуры аутентификации и авторизации выступают важным звеном этого процесса, обеспечивая не только защиту от несанкционированного доступа, но и управление различными уровнями доступа к ресурсам. Как правило, эти процедуры выполняются сразу после установления соединения клиента с сервером и являются неотъемлемой составляющей реализации сложных клиент-серверных приложений. В процессе аутентификации проверяется корректность введенных пользователем данных, а процесс авторизации используется для контроля доступа к различным ресурсам серверной части приложения после успешного прохождения аутентификации. Для случая, когда пользователь не был ранее зарегистрирован, существует

процедура регистрации, позволяющая пользователю сохранить введенные им данные в базе данных на сервере.

После успешного прохождения описанных выше процедур пользователь получает права доступа для использования основных функций приложения. Соответственно, для удобного взаимодействия между клиентом и сервером можно определить различные типы сообщений, на основании которых участники взаимодействия определяют, какая функция требует выполнения. Таким образом, типы сообщений устанавливают определенные правила взаимодействия клиента и сервера.

В сетевой модели TCP/IP наиболее часто используются два протокола транспортного уровня: TCP и UDP [2]. Протокол выбирается в зависимости от требований, которые предъявляются к реализуемой системе. В нашем исследовании мы использовали TCP сокеты для надежного соединения и гарантированной доставки сообщений. Следует отметить, что для надежной передачи данных также можно использовать протокол UDP с использованием подтверждения, что позволит гарантировать доставку сообщений от клиента серверу и обратно.

Важной особенностью при проектировании крупных клиент-серверных приложений является синхронизация отправки и получения сообщений на основании их типа. Это подразумевает согласование передачи сообщений, то есть ситуации, когда одна сторона ожидает получения сообщения, а другая гарантированно его отправляет. Как правило, в современных ОС семейства Linux функции отправки и получения уже реализованы, а взаимодействие с ними может осуществляться при помощи встроенных библиотек [1]. При использовании такого метода синхронизации выстраивается корректное взаимодействие клиента и сервера.

2. Описание собственной реализации систем чата и логистического центра

В зависимости от назначения проектируемого приложения, очевидно, что его внутреннее устройство изменяется в соответствии с требованиями и целями.

Чат может быть одноранговым или многоранговым в зависимости от поставленной цели. Под одноранговым подразумевается такое устройство взаимодействие пользователей, при котором все имеют одинаковые права и возможности. Соответственно, чат может быть многоранговым, когда у одних пользователей возможностей больше, чем у других. Например, в чате присутствуют администраторы, которые могут регулировать общение других пользователей, блокировать их, удалять сообщения и др.

В процессе нашего исследования мы занимались созданием чата с одноранговым взаимодействием пользователей. При этом также стояла цель создать возможность взаимодействия между пользователями не только при помощи сервера, но и напрямую. В нашей системе сервер при получении сообщения от пользователя передает его всем авторизованным участникам общего чата. Для достижения прямой коммуникации каждый из пользователей может запросить у сервера таблицу активных участников чата, содержащую IP-адрес и порт, для отправки сообщения напрямую одному из пользователей. Упрощенная структура данного взаимодействия представлена на рисунке 1.

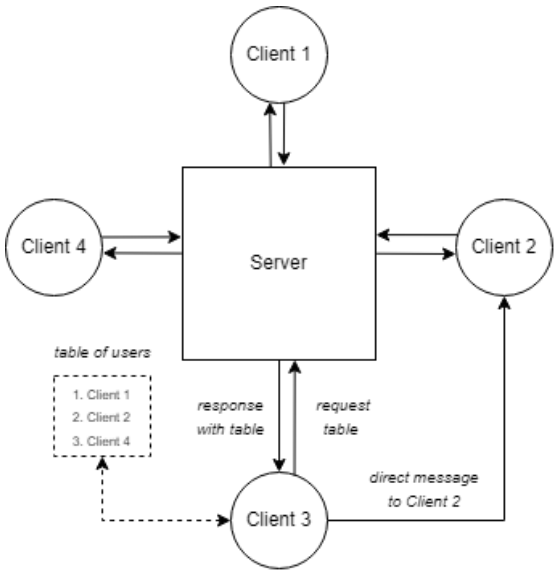

Рис. 1. Схема получения таблицы участников чата от сервера

Выделение пользовательских ролей характерно, в целом, для всех систем, требующих обеспечение разного уровня доступа к ресурсам. Так, при проектировании небольшого логистического центра нами были выделены две роли: клиента и работника склада. Пользователь с ролью первого типа может создавать заказы для доставки на другой склад, а также отслеживать отправленные и ожидаемые посылки. Пользователи со второй ролью могут отслеживать посылки как на своем складе, так и транспортируемые на их склад, а также изменять статусы заказов.

В автоматизированных системах, где возникает необходимость в сохранение структурированных и тесно связанных между собой данных, зачастую используются базы данных. В нашем исследовании для логистического центра была реализована упрощенная реляционная база данных с помощью обычных текстовых файлов, содержащих информацию о складах, клиентах, работниках склада и заказах.

3. Реализация серверной части системы

Главной задачей сервера является ожидание запросов на соединение от множества пользователей, получение сообщений и отправка ответа в зависимости от типа запрашиваемой клиентом услуги. При этом пользователи должны иметь возможность запрашивать услуги одновременно, что напрямую влияет на реализацию сервера. Для осуществления такого взаимодействия использовалась многопоточность [1]. В наших приложениях у сервера есть главный поток, ожидающий запросы на подключение от пользователей, а также после установления соединения для каждого клиента создается собственный поток, используемый для взаимодействия с пользователем.

Учитывая, что сервер осуществляет процедуры авторизации и регистрации, в рамках этого он контролирует процесса корректность и, для отдельных полей сообщения, уникальность переданных данных.

Из-за того, что в один момент времени к некоторым данным, хранящимся на сервере, могут одновременно обращаться несколько клиентов, важно осуществлять контроль за разделяемыми ресурсами системы. Так, сервер должен контролировать, например, правильное взаимодействие с базой данных. В нашей реализации обеих систем для ограничения доступа к критическим секциям мы использовали мьютексы [3, 4]. Это простой способ обеспечения атомарности, а именно: взаимного исключения доступа к одному ресурсу.

4. Реализация клиентской стороны системы

Клиентская часть, как и серверная, может достаточно сильно различаться в зависимости от проектируемой системы. В нашем случае отличаются реализации в чате и логистическом центре. В первой системе должна реализовываться возможность получения сообщения от других клиентов при ожидании ввода пользователем своего сообщения. Это вынуждает использовать более одного потока. Так, главный поток может взаимодействовать с вводимыми пользователем командами и сообщениями, а второй поток может получать сообщения от других пользователей, присылаемые ему сервером. В логистическом центре во втором потоке необходимости нет, так как клиент может взаимодействовать только с сервером, то есть общения между пользователями через сервер не происходит.

В нашей реализации чата стояла задача создать прямое взаимодействие между клиентами, минуя сервер. Для этого у каждого клиента есть поток, выполняющий упрощенные функции сервера - ожидание подключения и прием личного сообщения. При регистрации или авторизации клиент передает серверу данные для осуществления подключения к нему других клиентов. Каждый из клиентов может запросить у сервера список активных пользователей, чтобы отправить ему сообщение напрямую.

5. Реализация взаимодействия с помощью языка С и сокетов в ОС Linux

В данном разделе представлены некоторые особенности используемых для реализации систем средств языка С в ОС Linux и рекомендации по их использованию.

Для удобства реализации взаимодействия клиента и сервера нами был создан тип данных “Сообщение”, реализованный при помощи структуры языка С [5]. Обязательным атрибутом данной структуры является тип сообщения, остальные поля могут быть определены произвольно в зависимости от проектируемой системы. Множество типов сообщений также может отличаться от системы к системе. В нашем случае оно определяется при помощи типа данных “Тип сообщения”, реализованный при помощи перечисления в языке С [5].

В нашей работе мы использовали библиотеки языка С, которые предоставляют функции для использования системных вызовов для работы с сетью в ОС Linux [6]. Основная библиотека, предоставляющая функции для создания TCP и UDP сокетов, называется socket.h. В ней находятся основные функции для создания, настройки сокетов и взаимодействия между ними. Также для удобства назначения адреса и порта для сокета в стеке протоколов TCP/IP была использована библиотека netinet/in.h.

Для реализации надежной и безотказной системы важно правильно обрабатывать поступающие от операционной системы сигналы [1]. Это позволит корректно завершить работу клиента или сервера (или обоих) в случае возникновения экстренной ситуации.

Заключение

Таким образом, в рамках данной статьи мы осветили основные черты реализации систем чата и логистического центра. Можно выделить следующие основные черты и особенности:

- Процессы аутентификации, авторизации и регистрации. Они характерны для всех систем вне зависимости от их назначения;
- Многопоточная обработка клиентов сервером. Характерна для всех систем, требующих одновременного подключения нескольких клиентов к серверу;
- Реализация многопоточного приложения на стороне клиента. Зависит от реализуемой системы и требований к клиентской стороне;
- Доступ к разделяемым ресурсам несколькими клиентами одновременно. Характерно для систем, когда несколько клиентов могут одновременно иметь доступ к одним и тем же ресурсам сервера;
- Хранение данных от клиентов на сервере. Характерно для любой сложной системы, требующей сохранения пользовательских данных;

- Структура передаваемого сообщения и его тип. Для удобства взаимодействия и различия услуг, запрашиваемых от сервера, при проектировании определяется структура сообщения и его тип.

Безусловно, в данной работе освещена лишь часть черт и концепций, которые могут использоваться при проектировании и реализации клиент-серверных приложений. Так, например, для защиты передаваемых и хранимых данных можно использовать шифрование, при регистрации можно запрашивать подтверждение пароля. Наличие графического интерфейса сделало бы созданные приложения более доступными для большего числа пользователей.

Несмотря на то, что в современном мире растет уровень абстракций средств реализации клиент-серверных приложений, знание базового устройства взаимодействия позволяет создавать собственные комплексные реализации как приложений, так и программных продуктов, при помощи которых они создаются.

СПИСОК ЛИТЕРАТУРЫ

1. Таненбаум Э. Современные операционные системы. / Таненбаум Э., Бос Х. – Спб.: Питер, 2017. – 1120 с.
2. Олифер В. Г. Компьютерные сети. Принципы, технологии, протоколы: Юбилейное издание / Олифер В. Г., Олифер Н. В. – Спб.: Питер, 2024. – 1008 с.
3. Прокофьева Т. Л. Методы организации взаимодействия процессов: метод. пособие / Прокофьева Т. Л. – Спб.: ГУАП, 2019. – 57 с.
4. Прокофьева Т. Л. Механизмы межпроцессного взаимодействия System V. Реализация: учеб.-метод. пособие / Прокофьева Т. Л. – СПб.: ГУАП, 2022. – 39 с.
5. Брайан К. Язык программирования С. / Брайан К., Деннис Р. – М.: Вильямс, 2015. – 304 с.
6. Hall B. Beej's Guide to Network Programming: Using Internet Sockets / Hall B. – Independently Published, 2019. — 180 p.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Петров Николай Николаевич
Студент
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Россия, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: kolya.petrow38@gmail.com

Родичев Ярослав Михайлович
Студент
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
e-mail: jaro_r@mail.ru

Прокофьева Татьяна Львовна
Старший преподаватель
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
e-mail: tlfx@mail.ru

INFORMATION ABOUT THE AUTHORS

Petrov Nikolay Nikolaevich
Student
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: kolya.petrow38@gmail.com

Rodichev Iaroslav Mikhailovich
Student
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: jaro_r@mail.ru

Prokofieva Tatyana Lvovna
Senior Lecturer
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: tlfx@mail.ru