|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ |
| **«Национальный исследовательский ядерный университет «МИФИ»** |
| ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ |
| КАФЕДРА «КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ» (№12) |

**Курсовой проект**

**по дисциплине**

**Архитектуры современных операционных систем**

|  |  |  |  |
| --- | --- | --- | --- |
| **Тема: Разработка конвейера обработки сообщений, состоящего из подключаемых обработчиков.** | | | |
|  | | | |
| Студент | Ягненков Алексей Дмитриевич | Группа | С19-501 |
|  | ФИО |  |  |
| Руководитель | Кургинян Элла Аршаковна, ассистент кафедры №12 «Компьютерные системы и технологии» | | |
|  | ФИО, степень, звание, должность | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Ягненков А. Д. |
|  | подпись |  | ФИО |
| Руководитель |  |  | Кургинян Э. А. |
|  | подпись |  | ФИО |

Москва, 2022

Оглавление

[ВВЕДЕНИЕ 3](#_Toc97131782)

[1. ПОСТАНОВКА ЗАДАЧИ 4](#_Toc97131783)

[2. АРХИТЕКТУРА СИСТЕМЫ 4](#_Toc97131784)

[2.1. Принципиальная схема 4](#_Toc97131785)

[2.2. Принцип работы сервера и клиента 5](#_Toc97131786)

[2.3. Формат ввода команд 6](#_Toc97131787)

[2.4. Формат обмена сообщениями 6](#_Toc97131788)

[3. РАЗРАБОТКА ПРОГРАММЫ 6](#_Toc97131789)

[3.1. Выбор языка программирования 6](#_Toc97131790)

[3.2. Необходимые классы языка JAVA 7](#_Toc97131791)

[3.3. Классы программы 7](#_Toc97131792)

[4. ТЕСТИРОВАНИЕ 8](#_Toc97131793)

[ЗАКЛЮЧЕНИЕ 11](#_Toc97131794)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 12](#_Toc97131795)

# ВВЕДЕНИЕ

В современном мире люди практически каждый день используют концепцию, называемую клиент-сервером, даже не задумываясь об этом. Происходит это, когда пользователи выходят в Интернет, чтобы пообщаться с друзьями, оплатить счета или просто почитать новости.

Данная концепция включает в себя 2 стороны: клиентскую и серверную. Клиент – это своеобразный заказчик услуги, а сервер – её поставщик. Типичным клиентом является браузер, в качестве серверной стороны могут выступать HTTP сервера, Denwer, MySQL сервер и другие. Клиент и сервер взаимодействуют посредством разнообразных сетевых протоколов. Программное обеспечение обеих сторон обычно находится на разных машинах, хотя их работа возможна и на одном компьютере (что будет продемонстрировано в данном курсовом проекте).

В то время как обычным пользователям не важно, как устроено клиент-серверное взаимодействие, главное, чтобы необходимая услуга была выполнена качественно и вовремя, веб-разработчикам, сетевым администраторам, инженерам связи не обойтись без понимания этих процессов. Получение начальных сведений о клиент-серверной архитектуре и навыков её создания является целью данного курсового проекта.

# ПОСТАНОВКА ЗАДАЧИ

Разработать систему, позволяющую обрабатывать команду пользователя (в качестве команды должна выступать строка символов), передавая управление ряду зарегистрированных обработчиков (обработчики поднимают символ на определенной позиции строки в верхний регистр). Процессы-обработчики при запуске регистрируются в некоем глобальном регистре, и получают доступ к разделяемой области памяти, которая содержит очередную заявку, требующую обработки, при наступлении их очереди они устанавливают блокировку на заявку, совершают свою часть работы над ней, снимают блокировку и вызывают очередной обработчик. Команды должны генерироваться автоматически процессом-генератором, который так же имеет доступ в разделяемую память.

# АРХИТЕКТУРА СИСТЕМЫ

## **Принципиальная схема**

Принципиальная схема концепции клиент-сервер для поставленной задачи представлена на рисунке 1. Использованы следующие обозначения:

* User – клиент (пользователь);
* Server – сервер;
* Socket – сокет (программный интерфейс для обеспечения обмена данными между процессами);
* Processing – обработка;
* Shared memory area – разделяемая область памяти;
* Internal queue in shared memory area – внутренняя очередь в разделяемой области памяти;
* Global Register – глобальный регистр;
* Handler – обработчик.

Пунктирным прямоугольником выделена серверная сторона системы.

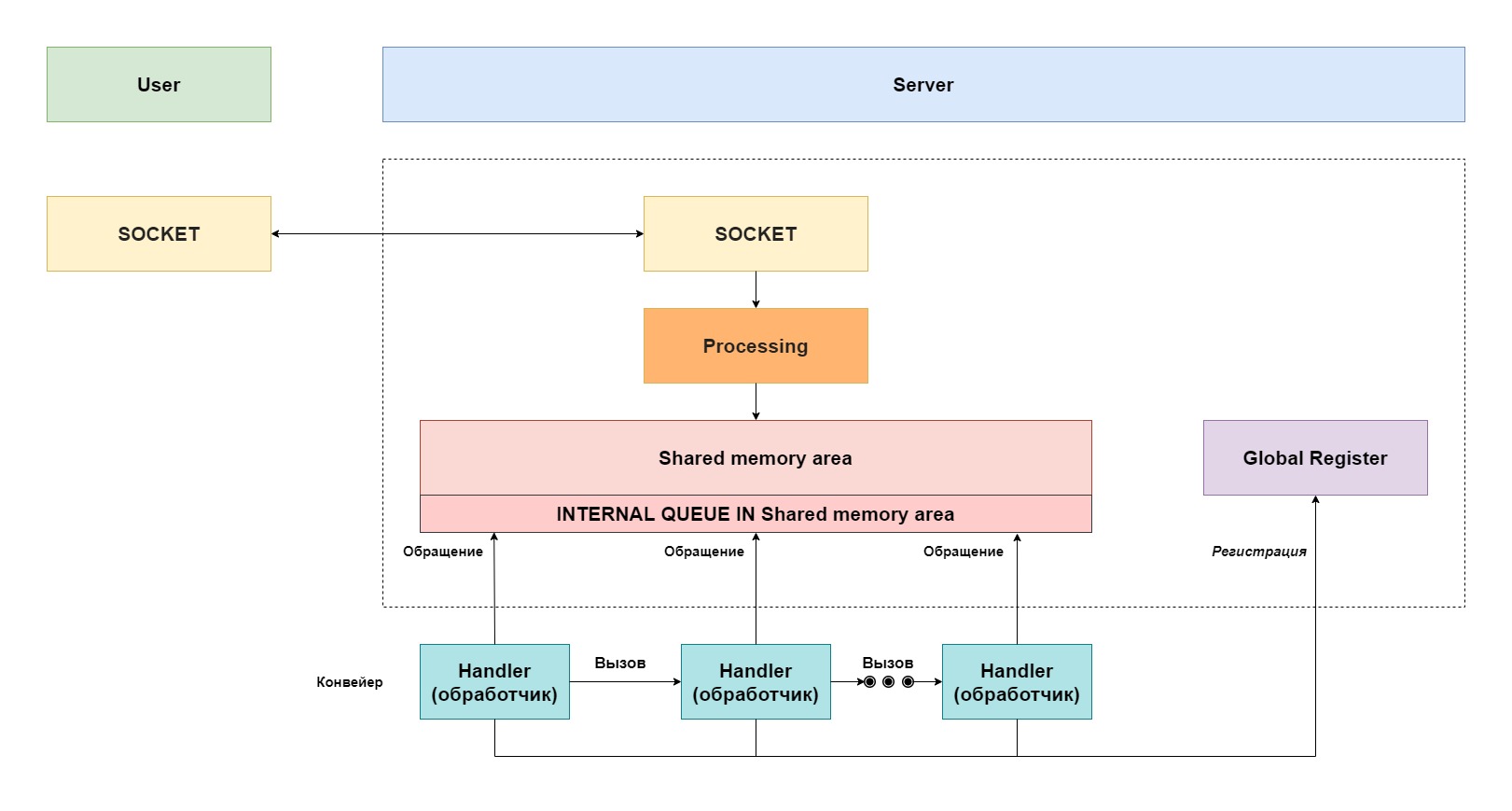


Рисунок 1. Принципиальная схема

## **Принцип работы сервера и клиента**

Работа клиент-серверной архитектуры начинается с запуска сервера, потому что сервер без клиентов работать может, а клиенты без сервера - нет. При включении клиент (при помощи генератора) вырабатывает команду, представленную строкой символов. Взаимодействие клиента и сервера осуществляется при помощи сокетов, играющих роль точки соединения. Сокет создаётся на клиентской стороне, а сервер воссоздаёт его при получении сигнала на подключение. В сокете указываются имя хоста (определённый узел сети) или ip-адрес машины и порт. На серверной стороне создаётся сервер-сокет. Для него нужно указать порт, указывать адрес соединения не нужно, так как общение происходит на машине сервера. Команда, полученная сервером от клиента, поступает в очередь, находящуюся в разделяемой области памяти. Клиент получает сообщение о принятии команды сервером.

Для обработки команд включается конвейер обработчиков. Суть конвейера состоит в том, что определённая его стадия (в данном случае определённый обработчик) проводит операцию не над всей командой (строкой символов), а над её частью и передаёт управление следующей стадии (обработчику). При этом все обработчики работают параллельно. Таким образом, достигается повышение производительности системы за счёт выполнения не одной операции, а сразу нескольких операций в единицу времени. После регистрации в глобальном регистре обработчики получают доступ к разделяемой области памяти и выполняют свою часть работы над командами, передавая управление друг другу.

После обработки всех полученных команд сервер отправляет клиенту результат. Соединение сервера и клиента так же осуществляется посредством сокетов.

## **Формат ввода команд**

По условию задачи пользователь посылает на сервер команду в виде строки, обработка заключается в поднятии символов в верхний регистр, поэтому пользователь (при помощи генератора) создаёт строки символов (букв) в нижнем регистре без пробелов и других знаков. Так как в данном курсовом проекте разработка программы осуществляется на языке программирования JAVA, максимальный размер строки – 2147483647.

## **Формат обмена сообщениями**

Клиент и сервер обмениваются сообщениями в виде строк (String) из класса java.lang.String. При этом используются следующие классы: InputStreamReader и OutputStreamWriter - классы, которые представляют собой мост, позволяющий преобразовать byte stream в character stream и character stream в byte stream соответственно (байтовые потоки в символьные и наоборот), и [BufferedReader](https://javarush.ru/groups/posts/593-bufferedreader-i-bufferedwritter) и BufferedWriter, использующие при считывании данных и записи данных соответственно специальную область — буфер, куда они «складывают» и откуда «берут» прочитанные символы, что позволяют экономить ресурсы.

|  |
| --- |
| Строка символов |

Таблица 1. Формат обмена сообщениями

# РАЗРАБОТКА ПРОГРАММЫ

## **Выбор языка программирования**

Для написания клиент-серверной архитектуры в данном курсовом проекте выбран язык программирования JAVA. На сегодняшний день JAVA является одной из мощных платформ для разработки, в том числе клиент-серверных приложений. Этот язык программирования обладает достаточной гибкостью и большим набором встроенных классов. В пакете java.net есть классы Socket и ServerSocket, которые удобно использовать при написании программы для решения поставленной задачи.

## **Необходимые классы языка JAVA**

При написании программы были использованы следующие встроенные классы:

*для сервера:*

* java.net.ServerSocket;
* java.net.Socket;
* java.util.concurrent.ConcurrentLinkedQueue;
* java.util.concurrent.TimeUnit;

а также некоторые классы из пакета java.io;

*для клиента:*

* java.net.Socket;

а также некоторые классы из пакета java.io.

## **Классы программы**

При написании программы были созданы следующие классы:

* Класс Server – серверная часть архитектуры. В нём создаются сервер-сокет, прослушивающий порт 4004, разделяемая область памяти с очередью, глобальный регистр, регистрирующий обработчики. Сервер принимает заявки (команды) клиента, передаёт управление обработчикам и отправляет ответ клиенту.
* Класс Client – клиентская часть архитектуры. В нём создаются сокет с адресом localhost и портом 4004 (как и у сервера) и генератор команд. Клиент осуществляет общение с сервером.
* Класс TaskGenerator – вспомогательный класс для генерации команд. Его экземпляр создаётся в клиентской части.
* Класс Task – команда, отправляемая клиентом серверу. В качестве полей имеет массив Letter (массив букв в строке), признак блокировки в памяти и счётчик количества обработанных символов.
* Класс Letter – символы, из которых состоит команда. В качестве полей имеет определённый символ и признак обработки.
* Класс Handler – обработчик, реализующий класс Runnable. Имеет идентификатор, внутреннюю очередь необработанных команд, ссылку на следующий обработчик и общую для всех обработчиков очередь обработанных команд.
* Класс GlobalRegister – глобальный регистр сервера. В качестве полей имеет массив зарегистрированных обработчиков и их количество.

Исходный код программы, а также результаты её работы представлены в Приложениях.

# ТЕСТИРОВАНИЕ

Для тестирования работы программы созданы условия, при которых клиент каждый раз отправляет одинаковое количество слов (5), количество букв в словах изменяется от 1 до 5 в соответствии с порядковым номером. Время обработки выводится после отчёта о работе обработчиков.

1. В данном тесте измеряется время обработки команд при наличии одного работающего обработчика (режим без конвейера).

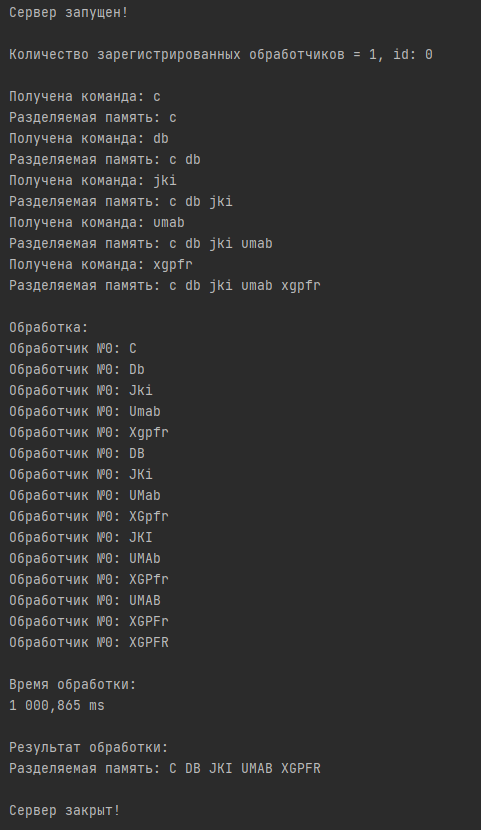


Рисунок 2. Результат работы программы с одним обработчиком

1. В данном тесте измеряется время обработки команд при наличии двух работающих обработчиков.

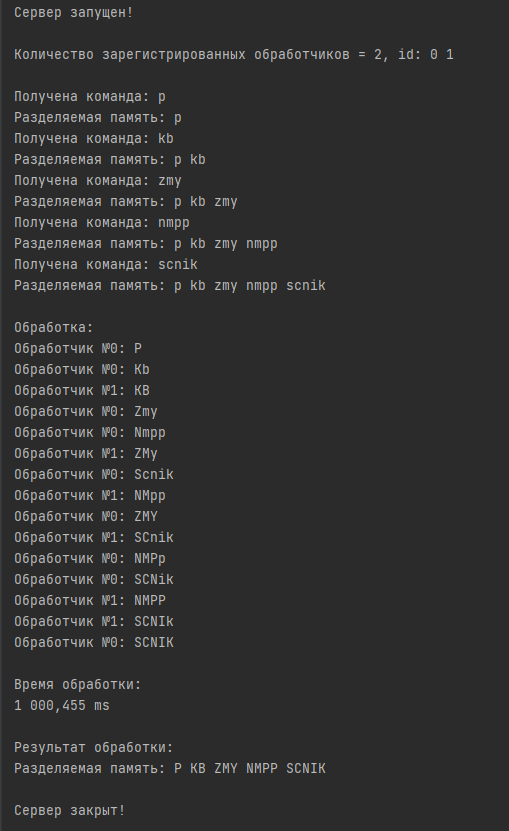


Рисунок 3. Результат работы программы с двумя обработчиками

Как видно из результатов, система с двумя обработчиками выполняет обработку команд быстрее системы с одним обработчиком. Это обусловлено тем, что во втором случае команды обрабатывались параллельно, а в первом – последовательно.

# ЗАКЛЮЧЕНИЕ

При выполнении данного курсового проекта были получены начальные сведения об архитектуре клиент-серверных приложений, построена принципиальная схема системы, изучены классы языка программирования JAVA для реализации взаимодействия двух сторон этой концепции, написана программа, реализующая заданную систему.

Задание выполнено в полном объёме и в срок.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. [Электронный ресурс]. Электрон. дан. — Режим доступа: <https://javarush.ru/groups/posts/654-klassih-socket-i-serversocket-ili-allo-server-tih-menja-slihshishjh>
2. [Электронный ресурс]. Электрон. дан. — Режим доступа:

<https://zametkinapolyah.ru/>