МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 32**

по дисциплине: “Системное программирование”

на тему:**“Сокеты”**

Вариант 12,5

Выполнил**:** студент группы 10701222 Походня Д. А.

Ерченко К. Д.

Принял**:** ст. пр. Давыденко Н. В.

Минск 2024

# Лабораторная работа № 3.1.

**Цель: изучить механизм сокетного сетевого взаимодействия в LINUX. Научиться выстраивать клиент-серверную архитектуру приложений с помощью сокетов.**

## Задание 1.

Напишите программы с клиент-серверной архитектурой по следующим тре

бованиям:

1. Обменяться числами. Сервер возводит получаемое от клиента число в

квадрат и возвращает ответ клиенту.

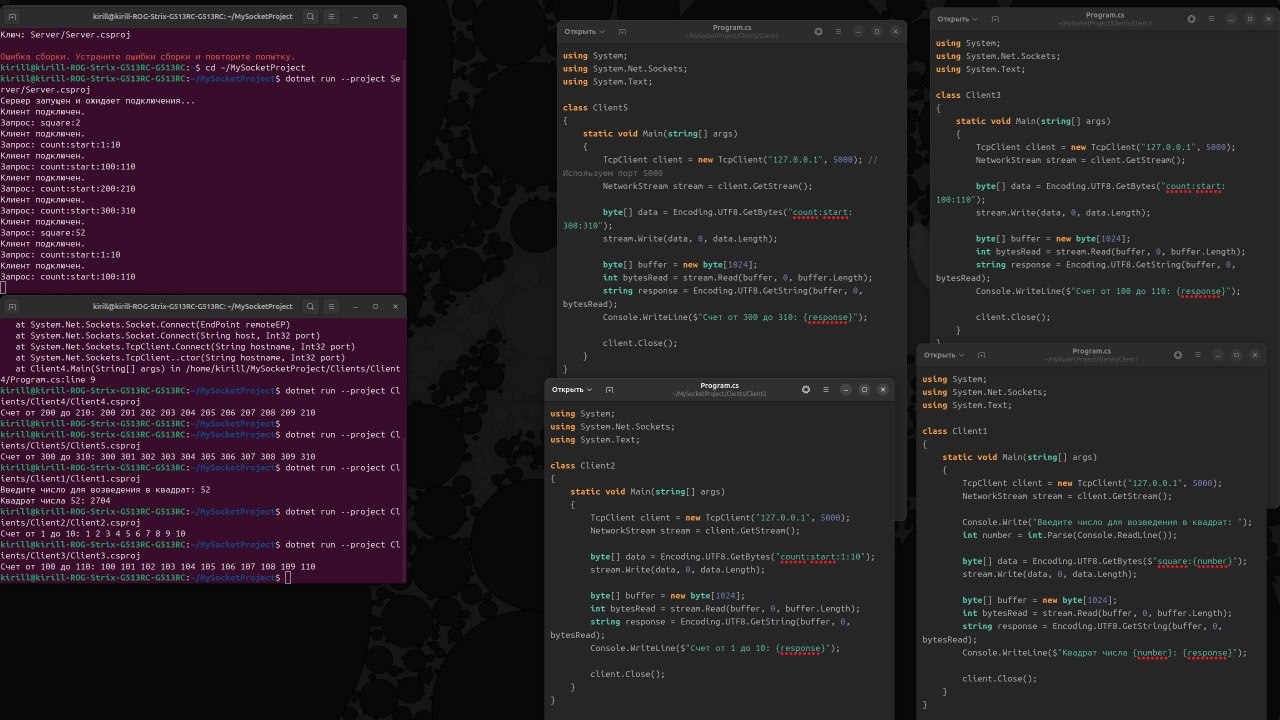
2. Создать два клиента на один и тот же сокет. Используют один и тот же

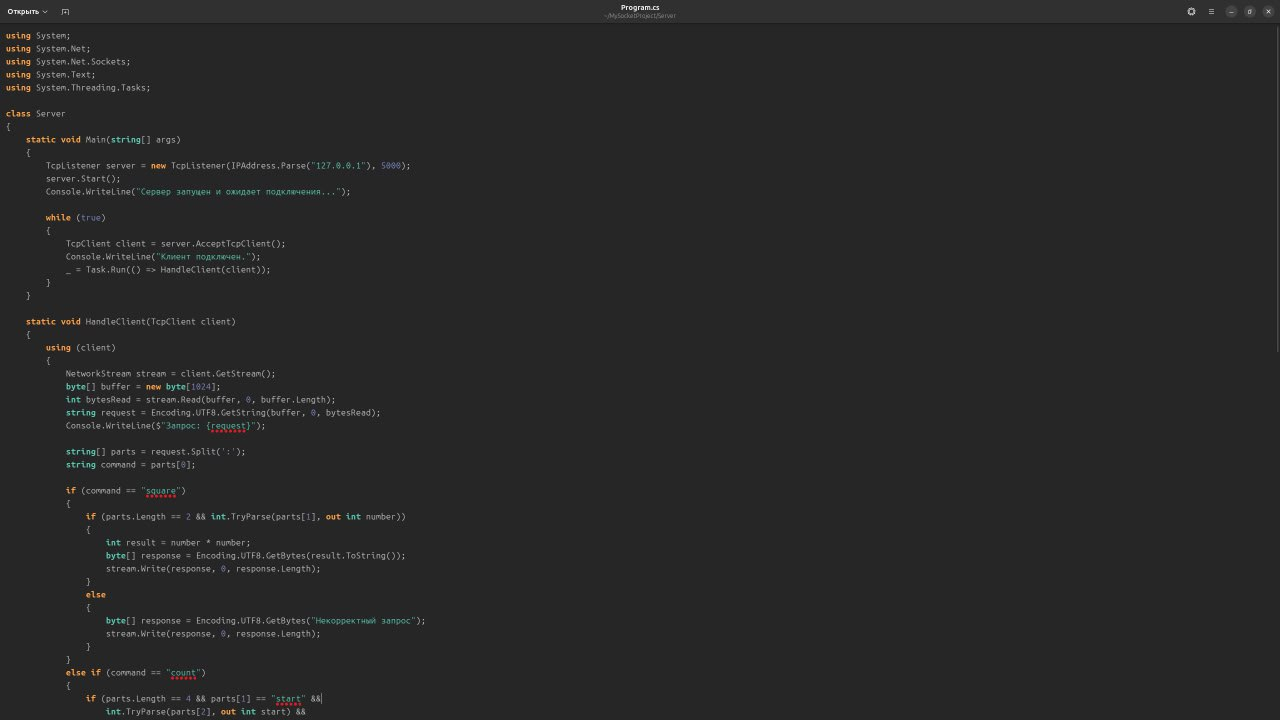
сокет. Первый клиент получает результат счета от 1 до 10, второй – ре

зультат счета от 100 до 110 (подумайте, как распознать, какой клиент

обратился к серверу).

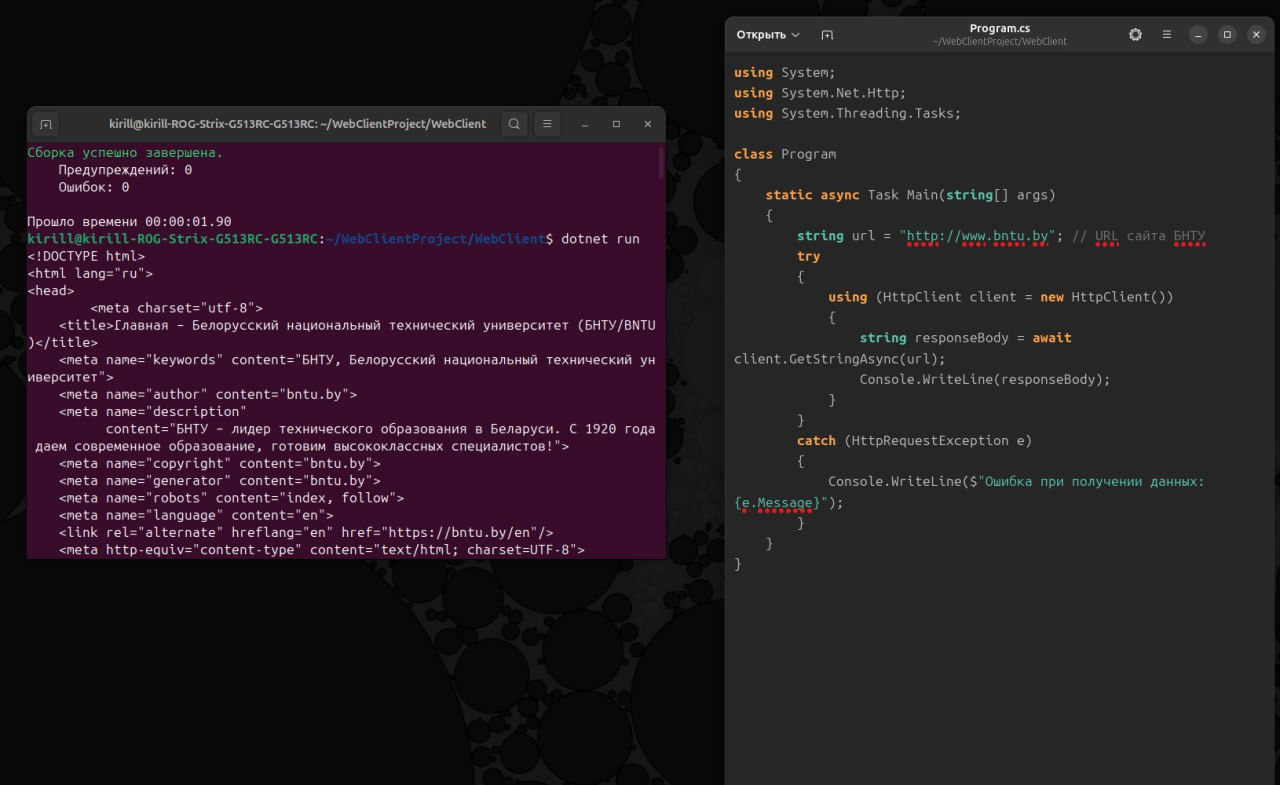
3. Создать два клиента на один и тот же сервер. Сокеты должны быть раз ными (именованное соединение в каждом случае должно быть различ ным). Первый клиент получает результат счета от 1 до 10, второй – от 100 до 110.





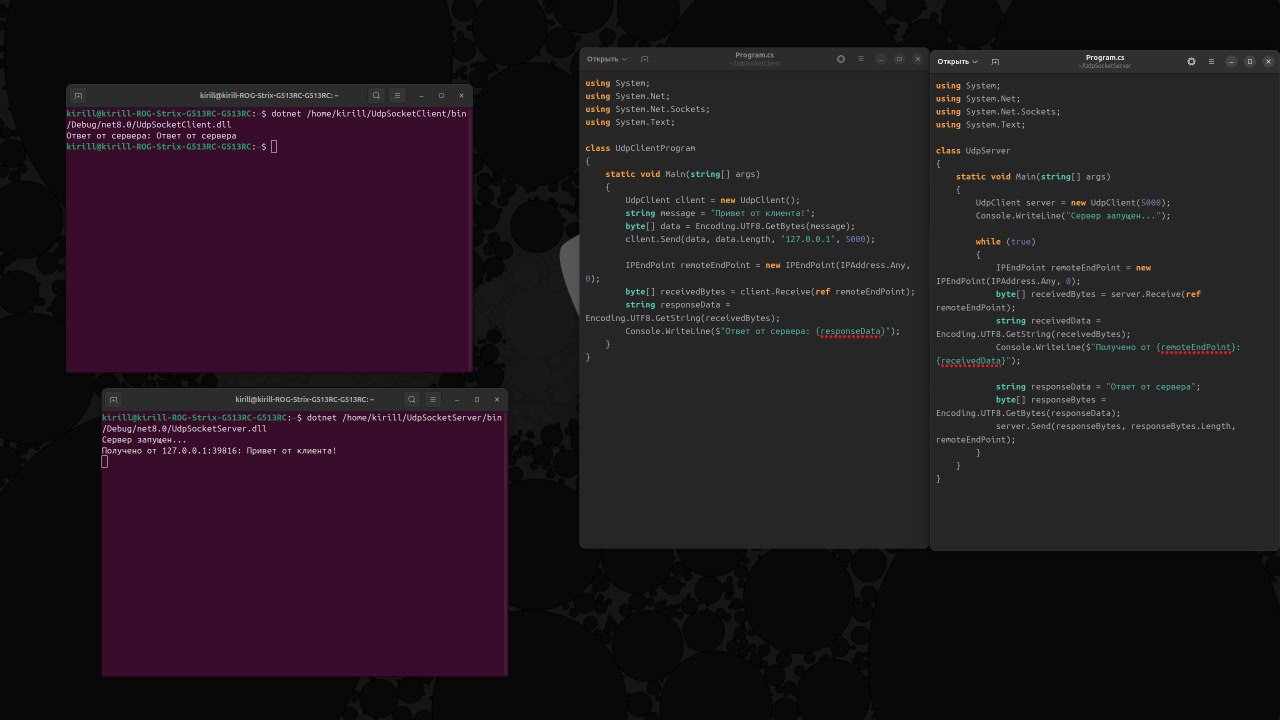
## Задание 2.

Напишите программу, которая читает главную страницу из какого-нибудь Web сервера. Запуск программы должен производиться с консоли.



## Задание 3.

Напишите программу с клиент-серверной архитектурой для передачи данных через дейтаграммные сокеты. Количество клиентов должно соответствовать количеству членов вашей команды, но не менее двух. Семейства протоколов и способы взаимодействия сокетов-клиентов должны отличаться. Запустите программы в разных терминалах и определите какие программы клиенты получили данные. Объясните полученный результат.



Контрольные вопросы

Вариант 12

1. Перечислите основные типы сокетов.

Существует несколько основных типов сокетов, каждый из которых используется для различных целей в сетевом программировании:

1. **Сокеты потоков (Stream Sockets)**:
   * Используют протокол TCP.
   * Обеспечивают надежное, двустороннее, ориентированное на соединение взаимодействие.
   * Подходят для приложений, где важна целостность данных (например, веб-браузеры, электронная почта).
2. **Сокеты датаграмм (Datagram Sockets)**:
   * Используют протокол UDP.
   * Обеспечивают ненадежное, без установления соединения взаимодействие.
   * Подходят для приложений, где важна скорость, а не целостность данных (например, потоковое видео, VoIP).
3. **Сокеты управления (Raw Sockets)**:
   * Позволяют отправлять и получать IP-пакеты без какого-либо протокола транспортного уровня.
   * Используются для сетевого мониторинга и анализа (например, для снифферов и инструментов диагностики).
4. **Сокеты по умолчанию (Default Sockets)**:
   * Сокеты, которые не явно указаны как потоковые или датаграммные, могут использоваться в зависимости от реализации.

Каждый тип сокета имеет свои особенности и подходит для различных сценариев использования в сетевых приложениях.

Ответ:

Вариант 5

Какой способ взаимодействия процессов обеспечивает сокет?

Ответ:

Сокеты обеспечивают взаимодействие процессов через механизм межпроцессного взаимодействия (IPC). Они позволяют процессам обмениваться данными, как на одной машине, так и между разными машинами по сети. Основные способы взаимодействия через сокеты включают:

1. **Сетевые сокеты**:
   * Используются для обмена данными между процессами, находящимися на разных компьютерах в сети.
   * Поддерживают различные протоколы, такие как TCP (для надежного соединения) и UDP (для быстрого, но ненадежного обмена).
2. **Локальные сокеты (Unix-сокеты)**:
   * Используются для взаимодействия процессов на одной машине.
   * Обеспечивают более быстрый обмен данными по сравнению с сетевыми сокетами, так как не требуют сетевой обработки.

Сокеты обеспечивают двустороннюю связь, что позволяет процессам как отправлять, так и принимать данные.