КАФЕДРА №

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

Межсетевое взаимодействие между процессами.

по курсу: Операционные системы

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №			
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

Цель работы

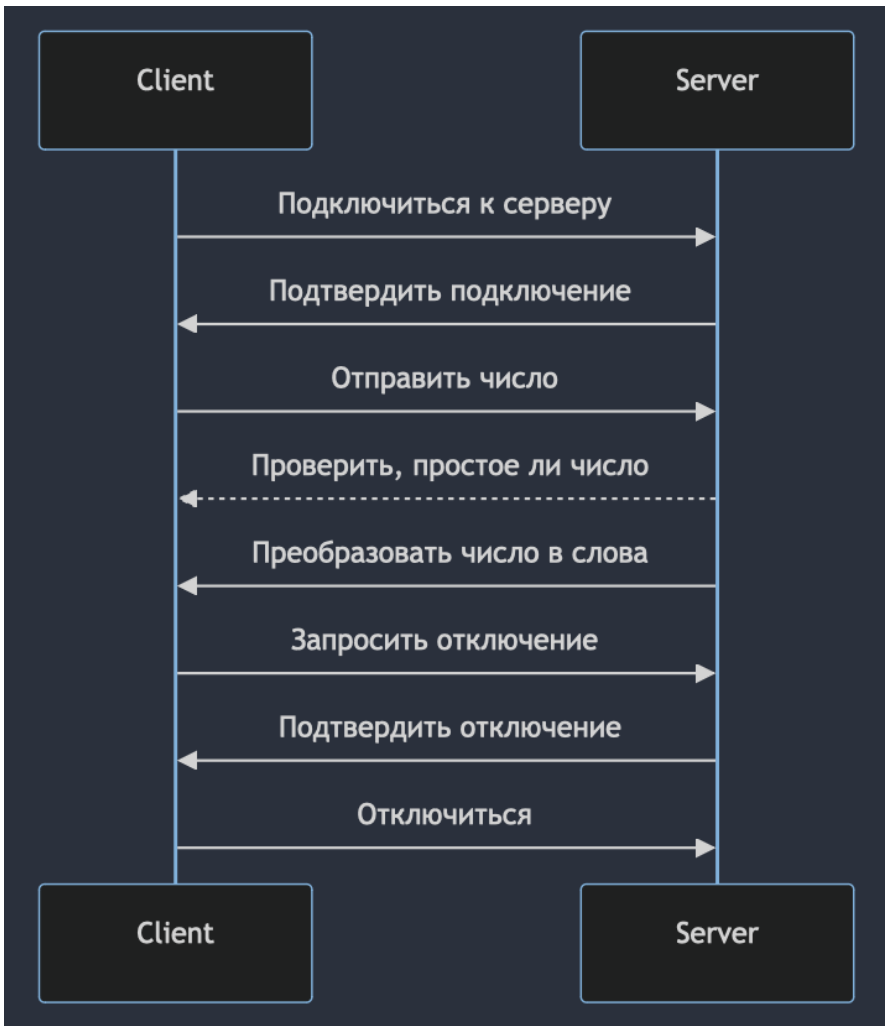
Организация межсетевого взаимодействия средствами WinAPI и POSIX.

Задание на лабораторную работу

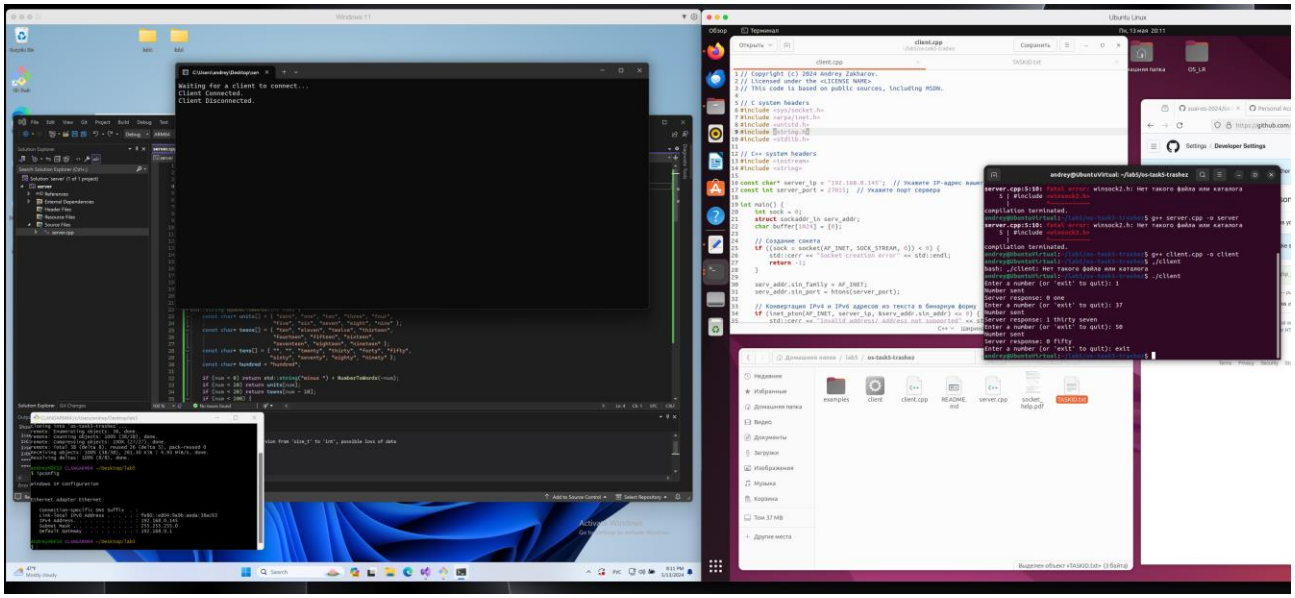
№ варианта	Индивидуальное задание	Протокол	Сервер	Клиент
13	13	TCP	Windows	Linux

Простые числа. Приложение-клиент запрашивает у пользователя ввод числа и передает его на сервер. Сервер проверяет является ли полученное число простым и сообщает результат клиенту. Также сервер присылает клиенту строку, содержащую полученное ранее число, записанное прописью. Например: Пользователь вводит: 37. Ответ сервера: 1 тридцать семь (или 1 thirty seven). Пользователь вводит: 10. Ответ сервера: 0 десять (или 0 ten).

Схема взаимодействия между клиентом и сервером



Результат выполнения работы



Исходный код программы с комментариями

Server.cpp – windows

```
// Copyright (c) 2024 Andrey Zakharov.
// Licensed under the <LICENSE NAME>
// This code is based on public sources, including MSDN.

#include <winsock2.h>
#include <ws2tcpip.h>
#include <iostream>
#include <vector>
#include <string>

// Other headers
#pragma comment(lib, "Ws2_32.lib")

bool IsPrime(int num) {
  if (num <= 1) return false;
  for (int i = 2; i * i <= num; ++i) {
    if (num % i == 0) return false;
  }
  return true;
}

std::string NumberToWords(int num) {
  const char* units[] = {"zero", "one", "two", "three", "four",
                         "five", "six", "seven", "eight", "nine"};
  const char* teens[] = {"ten", "eleven", "twelve", "thirteen",
                         "fourteen", "fifteen", "sixteen",
```

```
                    "seventeen", "eighteen", "nineteen"};
  const char* tens[] = {"", "", "twenty", "thirty", "forty", "fifty",
                        "sixty", "seventy", "eighty", "ninety"};
  const char* hundred = "hundred";

  if (num < 0) return "minus " + NumberToWords(-num);
  if (num < 10) return units[num];
  if (num < 20) return teens[num - 10];
  if (num < 100) {
    return tens[num / 10] + (num % 10 ? " " + units[num % 10] : "");
  }
  if (num < 1000) {
    return units[num / 100] + " " + hundred +
          (num % 100 ? " and " + NumberToWords(num % 100) : "");
  }
  return "number too large";
}

int main() {
  WSADATA wsaData;
  int result = WSAStartup(MAKEWORD(2, 2), &wsaData);
  if (result != 0) {
    std::cerr << "WSAStartup failed: " << result << std::endl;
    return 1;
  }

  SOCKET m_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (m_socket == INVALID_SOCKET) {
    std::cerr << "Error at socket(): " << WSAGetLastError() << std::endl;
    WSACleanup();
    return 1;
  }

  sockaddr_in service = {0};
  service.sin_family = AF_INET;
  service.sin_addr.s_addr = INADDR_ANY;
  service.sin_port = htons(27015);

  if (bind(m_socket,
  reinterpret_cast<SOCKADDR*>(&service),
  sizeof(service)) == SOCKET_ERROR) {
    std::cerr << "bind() failed." << std::endl;
    closesocket(m_socket);
    WSACleanup();
    return 1;
  }

  if (listen(m_socket, 1) == SOCKET_ERROR) {
    std::cerr << "Error listening on socket." << std::endl;
    closesocket(m_socket);
    WSACleanup();
    return 1;
```

```
  }

  std::cout << "Waiting for a client to connect..." << std::endl;
  while (true) {
    SOCKET AcceptSocket = accept(m_socket, NULL, NULL);
    if (AcceptSocket == INVALID_SOCKET) {
      std::cerr << "accept failed: " << WSAGetLastError() << std::endl;
      continue;
    }
    std::cout << "Client Connected." << std::endl;

    while (true) {
      char recvbuf[32] = {0};
      int bytesRecv = recv(AcceptSocket, recvbuf, sizeof(recvbuf) - 1, 0);
      if (bytesRecv <= 0) break;

      recvbuf[bytesRecv] = '\0';
      int num = atoi(recvbuf);
      bool prime = IsPrime(num);
      std::string words = NumberToWords(num);

      char sendbuf[256];
      snprintf(sendbuf, sizeof(sendbuf), "%d %s", prime ? 1 : 0, words.c_str());
      send(AcceptSocket, sendbuf, strlen(sendbuf), 0);
    }
    closesocket(AcceptSocket);
    std::cout << "Client Disconnected." << std::endl;
  }

  closesocket(m_socket);
  WSACleanup();
  return 0;
}
```

Client.cpp – linux

```
// Copyright (c) 2024 Andrey Zakharov.
// Licensed under the <LICENSE NAME>
// This code is based on public sources, including MSDN.

// C system headers
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

// C++ system headers
#include <iostream>
#include <string>

const char* server_ip = "192.168.31.70";  // Укажите IP-адрес вашего сервера
const int server_port = 27015;  // Укажите порт сервера
```

```
int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char buffer[1024] = {0};

    // Создание сокета
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        std::cerr << "Socket creation error" << std::endl;
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(server_port);

    // Конвертация IPv4 и IPv6 адресов из текста в бинарную форму
    if (inet_pton(AF_INET, server_ip, &serv_addr.sin_addr) <= 0) {
        std::cerr << "Invalid address/ Address not supported" << std::endl;
        return -1;
    }

    // Подключение к серверу
    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        std::cerr << "Connection Failed" << std::endl;
        return -1;
    }

    while (true) {
        std::cout << "Enter a number (or 'exit' to quit): ";
        std::string input;
        std::cin >> input;
        if (input == "exit") break;  // Условие выхода из цикла

        // Отправка числа
        send(sock, input.c_str(), input.length(), 0);
        std::cout << "Number sent" << std::endl;

        // Получение ответа от сервера
        int valread = read(sock, buffer, 1024);
        if (valread > 0) {
            buffer[valread] = '\0';
            std::cout << "Server response: " << buffer << std::endl;
        }
        memset(buffer, 0, sizeof(buffer));  // Очистка буфера
    }

    close(sock);
    return 0;
}
```

Вывод

Эта работа демонстрирует реализацию клиент-серверного приложения для проверки чисел на простоту и их представления словами с использованием WinAPI и POSIX для сетевого взаимодействия. Задача приложения заключается в обеспечении эффективной обработки пользовательского ввода и коммуникации между клиентом и сервером, что позволяет динамично обрабатывать запросы и предоставлять обратную связь в удобной для пользователя форме.