Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологии»

ДОМАШНЯЯ РАБОТА №2

«Разработка клиент-серверного приложения»

по дисциплине: «*Технологии системного программного обеспечения*»

Выполнил: студент группы ИУК4-62Б _____ Губин Е.В.
(Подпись) (И.О. Фамилия)

Проверил: _____ Красавин Е.В.
(Подпись) (И.О. Фамилия)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2025

Цель: получение практических навыков по написанию и отладке программ.

Задачи:

1. Научиться разрабатывать, компилировать и отлаживать
клиент-серверные приложения под ОС FreeBSD.

Результатами работы являются:

2. Исполняемый файл, содержащий программу,
разработанную согласно варианту;
3. Подготовленный отчет.

Вариант 8

Формулировка задания:

Вариант 8

Написать комплекс программ для преобразования десятичных чисел в римскую систему счисления. Результат выдать на экран и сохранить в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Листинг:

roman.cpp:

```
#include "roman.h"

std::string toRoman(int num) {
    struct RomanMap { int value; const char* numeral; };
    RomanMap roman[] = {
        {1000, "M"}, {900, "CM"}, {500, "D"}, {400, "CD"},
        {100, "C"},  {90, "XC"},  {50, "L"},  {40, "XL"},
        {10, "X"},   {9, "IX"},   {5, "V"},   {4, "IV"},
        {1, "I"}
    };

    std::string result;
    for (auto &r : roman) {
        while (num >= r.value) {
            result += r.numeral;
            num -= r.value;
        }
    }
    return result;
}
```

roman.h:

```
#pragma once
#include <string>
std::string toRoman(int num);
```

client.cpp:

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include "roman.h"

int main() {
    int number;
    std::cout << "Enter the number: (1-3999): ";
    std::cin >> number;

    if (number < 1 || number > 3999) {
        std::cerr << "Number out of range.\n";
        return 1;
    }

    std::string roman = toRoman(number);
    std::cout << "Roman number: " << roman << std::endl;

    std::ofstream out("result.txt");
    out << "Decimal: " << number << "\nRoman: " << roman << std::endl;
    out.close();

    int sock = socket(AF_INET, SOCK_STREAM, 0);
    sockaddr_in serv_addr{};
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(9000);
    inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr);

    if (connect(sock, (sockaddr*)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("Connect failed");
        return 1;
    }

    std::ifstream file("result.txt", std::ios::binary);
    std::string content((std::istreambuf_iterator<char>(file)),
                        std::istreambuf_iterator<char>());
```

```
    send(sock, content.c_str(), content.size(), 0);
    std::cout << "File send on server" << std::endl;

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

server.cpp:

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <unistd.h>
#include <netinet/in.h>

int main() {
    int server_fd = socket(AF_INET, SOCK_STREAM, 0);
    sockaddr_in address{};
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(9000);

    bind(server_fd, (sockaddr*)&address, sizeof(address));
    listen(server_fd, 1);
    std::cout << "Wait a client..." << std::endl;

    int addrlen = sizeof(address);
    int new_socket = accept(server_fd, (sockaddr*)&address, (socklen_t*)&addrlen);

    char buffer[1024];
    std::ofstream outfile("received.txt");

    int bytes;
    while ((bytes = recv(new_socket, buffer, sizeof(buffer), 0)) > 0) {
        outfile.write(buffer, bytes);
    }

    std::cout << "File received and saved 'received.txt'\n";
    outfile.close();
    close(new_socket);
    close(server_fd);
    return 0;
}
```

Makefile:

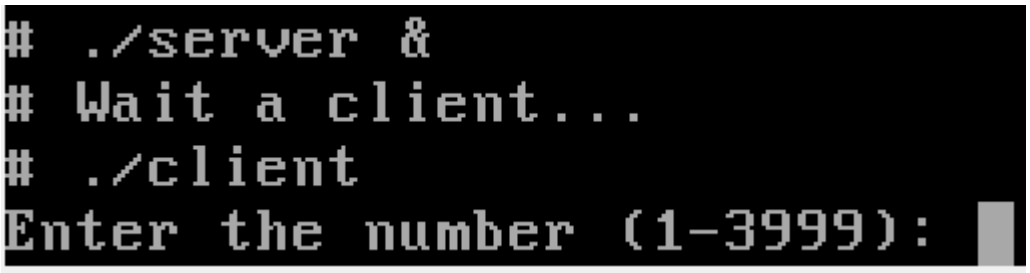
```
all: client server
```

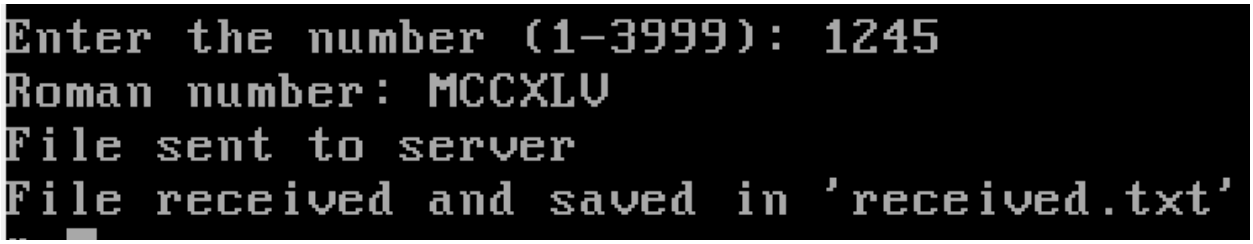
```
client: client.cpp roman.cpp
	clang++ client.cpp roman.cpp -o client

server: server.cpp
	clang++ server.cpp -o server

clean:
	rm -f client server result.txt received.txt
```

Результаты работы:





Вывод: в ходе лабораторной работы были получены практические навыки по разработке клиент-серверного приложения на языке C++ в ОС FreeBSD.