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:

client.cpp:

```
#include <arpa/inet.h>
#include <unistd.h>
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <sys/socket.h>
#include <netinet/in.h>

std::string toRoman(int);

int main() {
    const char* server_ip = "127.0.0.1";
    const int port = 12345;
    std::string filename = "results.txt";

    std::ofstream outFile(filename, std::ios::app);
    if (!outFile) {
        std::cerr << "Error to open file";
        return 1;
    }
```

```
    while (true) {
        std::cout << "Enter the number (0 to exit): ";
        int num;
        std::cin >> num;
        if (num == 0) break;

        if (num < 1 || num > 3999) {
            std::cout << "Number out of range [1-3999]\n";
            continue;
        }

        std::string roman = toRoman(num);
        std::string line = std::to_string(num) + " = " + roman + "\n";
        std::cout << line;
        outFile << line;
        outFile.flush();

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

        if (connect(sock, (sockaddr*)&serv_addr, sizeof(serv_addr)) < 0) {
            std::cerr << "Ошибка соединения с сервером\n";
            close(sock);
            continue;
        }

        std::ifstream fileToSend(filename, std::ios::binary);
        std::vector<char> buffer((std::istreambuf_iterator<char>(fileToSend)), {});
        send(sock, buffer.data(), buffer.size(), 0);
        close(sock);
    }

    return 0;
}

std::string toRoman(int number) {
    std::pair<int, std::string> 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& [val, sym] : roman) {
        while (number >= val) {
            result += sym;
            number -= val;
        }
    }
    return result;
}
```

server.cpp:

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

#define PORT 12345
#define BUFFER_SIZE 4096

int main() {
    int server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd < 0) return 1;

    sockaddr_in address{};
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if (bind(server_fd, (sockaddr*)&address, sizeof(address)) < 0)
        return 1;

    if (listen(server_fd, 3) < 0)
        return 1;

    while (true) {
        sockaddr_in client_addr{};
        socklen_t client_len = sizeof(client_addr);
        int client_socket = accept(server_fd, (sockaddr*)&client_addr, &client_len);
        if (client_socket < 0) continue;

        std::ofstream outfile("received.txt", std::ios::app | std::ios::binary);
        if (!outfile) {
            close(client_socket);
            continue;
        }
```

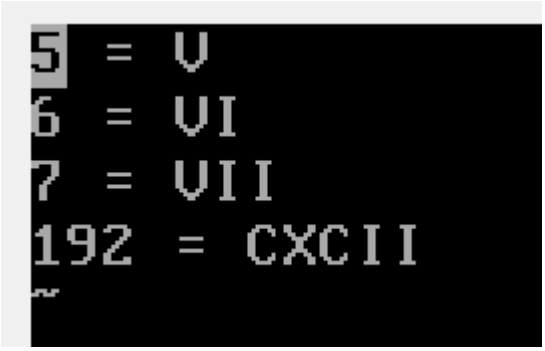
```
        char buffer[BUFFER_SIZE];
        ssize_t bytes_read;
        while ((bytes_read = read(client_socket, buffer, BUFFER_SIZE)) > 0) {
            outfile.write(buffer, bytes_read);
        }

        outfile.close();
        close(client_socket);
    }

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

Makefile:

```
CXX = c++
CXXFLAGS = -Wall -Wextra -std=c++17
TARGETS = client server

all: $(TARGETS)

client: client.cpp
    $(CXX) $(CXXFLAGS) -o client client.cpp

server: server.cpp
    $(CXX) $(CXXFLAGS) -o server server.cpp

clean:
    rm -f $(TARGETS) *.o results.txt

.PHONY: all clean
```

Результаты работы:

```
Enter the number (0 to exit): 5
5 = V
Enter the number (0 to exit): 6
6 = VI
Enter the number (0 to exit): 7
7 = VII
Enter the number (0 to exit): 192
192 = CXCII
Enter the number (0 to exit):
```



Ответы на контрольные вопросы

1. **Особенности ОС FreeBSD:**

- Свободная и открытая Unix-подобная ОС
- Высокая производительность и стабильность
- Полный стек: ядро + утилиты + документация
- Развитая система портов (pkg, ports)
- Поддержка современных файловых систем (например, ZFS)
- Активно используется на серверах

2. **Этапы разработки ПО:**

1. Постановка задачи
2. Анализ требований
3. Проектирование
4. Реализация (кодирование)
5. Тестирование
6. Внедрение
7. Сопровождение и обновление

3. **Этапы, предшествующие написанию кода:**

- Сбор и анализ требований
- Построение технического задания
- Проектирование архитектуры
- Разработка алгоритмов
- Подготовка спецификаций и документации

4. **Интерпретатор — это:**

Программа, выполняющая исходный код построчно без предварительной компиляции.

5. **Интерпретаторы, встроенные в FreeBSD:**

- /bin/sh — интерпретатор командной оболочки
- tcsh, csh — другие оболочки
- python, perl — при установке через порты

6. **Компилятор — это:**

Программа, преобразующая исходный код на языке высокого уровня в исполняемый машинный код.

7. **Компиляторы в FreeBSD:**

- clang (по умолчанию)
- gcc (доступен через pkg install gcc)

8. **Ключи gcc и их назначение:**

- -o — задать имя выходного файла
- -Wall — включить все предупреждения
- -g — включить отладочную информацию
- -O / -O2 / -O3 — оптимизация
- -c — компиляция без линковки
- -std=c99 — указание стандарта языка C

9. **Принцип работы с отладчиком:**

- Запуск программы через отладчик (gdb ./prog)
- Установка точек останова (break)
- Пошаговое выполнение (next, step)
- Просмотр переменных (print var)
- Анализ стеков и переходов (backtrace)

10. **Действия для запуска программы:**

1. Написать исходный код
2. Скомпилировать (gcc prog.c -o prog)

3. Убедиться, что есть права на запуск (chmod +x prog)
4. Запустить: ./prog

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