Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Операционные системы и системное программирование

# ОТЧЕТ
по лабораторной работе №8
на тему
“Сокеты. Взаимодействие процессов”

Выполнил: Студент группы 350501
М.А. Василюк

Проверил: старший преподаватель каф. ЭВМ
Л.П. Поденок

Минск 2025

**СОДЕРЖАНИЕ**

# 1 ЗАДАНИЕ ЛАБОРАТОРНОЙ РАБОТЫ

## 1.1 Цель работы

Разработка многопоточного сервера и клиента, работающих по простому протоколу.

## 1.2 Исходные данные к работе

Протокол должен содержать следующие запросы:

`ECHO` – эхо-запрос, возвращающий без изменений полученное от клиента;

`QUIT` – запрос на завершение сеанса;

`INFO` – запрос на получения общей информации о сервере;

`CD` – изменить текущий каталог на сервере;

`LIST` – вернуть список файловых объектов из текущего каталога.

Протокол может содержать дополнительные запросы по выбору студента, не выходящие за пределы корневого каталога сервера и не изменяющих файловую систему в его дереве. Запросы клиенту отправляются на `stdin`.

Ответы сервера и ошибки протокола выводятся на `stdout`.

Ошибки системы выводятся на `stderr`.

Подсказка клиента для ввода запросов – символ '>'.

Сервер принимает корневой каталог сервера, с которым он будет работать, и номер порта, на котором будет слушать, и выводит протокол работы в `stdout`.

Формат протокола произвольный, каждое событие занимает одну строку, первое поле – дата и время в формате `YYYY.MM.DD-hh:mm:ss.sss`).

Клиент помимо интерактивных запросов принимает запросы из файла. Файл с запросами указывается с использованием префикса '@'

# 2 ВЫПОЛНЕНИЕ РАБОТЫ

## 2.1 Описание алгоритма выполнения работы

Программа представляет собой многопоточный файловый сервер на `C`, работающий с отдельным клиентом, и реализует простой протокол для взаимодействия через сокеты домена `AF_INET`. Сервер, использующий `POSIX Threads`, позволяет каждому клиенту работать в собственном потоке, поддерживая независимые текущие каталоги для каждого соединения. Он обрабатывает команды вроде `ECHO` (эхо-запрос), `QUIT` (завершение сеанса), `INFO` (информация о сервере), `CD` (изменение каталога в пределах заданного корня) и `LIST` (получение содержимого текущего каталога, включая разрешения симлинков), при этом строго контролируя доступ к файловой системе, чтобы клиенты не могли выйти за пределы назначенного корневого каталога.

## 2.2 Описание основных функций

### Функция read_line()

Функция `read_line()` в клиенте отвечает за чтение одной строки данных, отправленной сервером. Она буферизует входящие данные из сокета, чтобы эффективно обрабатывать неполные или слишком длинные сообщения. При каждом вызове она последовательно извлекает символы из внутреннего буфера до тех пор, пока не встретит символ новой строки (`\n`) или пока буфер не опустеет. Если буфер пуст, она пытается принять больше данных из сокета. Функция динамически выделяет память для возвращаемой строки, что позволяет ей обрабатывать строки различной длины. В случае ошибки чтения или закрытия соединения она возвращает `NULL`.

### Функция handle_server()

Функция `handle_server()` в клиенте предназначена для обработки ответов, поступающих от сервера. Она в цикле вызывает `read_line()` для получения строк от сервера. Каждая полученная строка выводится на стандартный вывод клиента. Особое внимание уделяется строкам, которые заканчиваются символом '>', поскольку это является подсказкой (`prompt`) от сервера. При обнаружении подсказки, функция обновляет переменную `prompt` клиента (которая используется для отображения текущего каталога на сервере) и прекращает чтение, ожидая следующего ввода пользователя. Если клиент получает строку `"BYE"`, это сигнализирует о завершении сеанса, и программа клиента завершает свою работу.

### Функция client_thread()

Функция `client_thread()` является основной функцией для каждого потока сервера, обслуживающего отдельного клиента. Она получает дескриптор сокета клиента и путь к корневому каталогу сервера. При подключении клиента она отправляет ему приветственное сообщение и начальную подсказку. Затем функция входит в бесконечный цикл, где принимает команды от клиента, обрабатывает их и отправляет соответствующие ответы. Она распознает и выполняет команды `ECHO`, `QUIT`, `INFO`, `CD` и `LIST`. Особое внимание уделяется обработке команд `CD` и `LIST` для обеспечения того, чтобы клиент не мог выйти за пределы заданного корневого каталога сервера. После обработки каждой команды, она отправляет клиенту новую подсказку, отражающую текущий рабочий каталог. При получении команды `QUIT` поток отправляет `"BYE"` клиенту и завершает свою работу, освобождая ресурсы.

## 3 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

```
parallels@fedora-linux-40:~/Documents/tar_work-
ing_dir/ВасилюкМ.А./lab08/build/debug$ ./myclient
127.0.0.1 12345
Hello from 'myserver'
> LIST
file1.txt
dir1/
dir2/
> ECHO hello
hello
> CD dir2
dir2> LIST
symlink_to_file2.txt --> ../dir1/file2.txt
dir2> INFO
Hello from 'myserver'
dir2> QUIT
BYE
```

Рисунок 1 – действия на стороне клиента

```
parallels@fedora-linux-40:~/Documents/tar_work-
ing_dir/ВасилюкМ.А./lab08/build/debug$ ./myserver
server_root 12345
2025.05.21-11:03:44.636 Server started port=12345,
root=/home/parallels/Documents/tar_work-
ing_dir/ВасилюкМ.А./lab08/build/debug/server_root
2025.05.21-11:04:00.084 Client 4 connected, greeting sent
2025.05.21-11:04:07.735 Client 4 sent: LIST
2025.05.21-11:04:17.981 Client 4 sent: ECHO hello
2025.05.21-11:04:32.855 Client 4 sent: CD dir2
2025.05.21-11:04:37.105 Client 4 sent: LIST
2025.05.21-11:04:56.440 Client 4 sent: INFO
2025.05.21-11:05:00.750 Client 4 sent: QUIT
2025.05.21-11:05:00.750 Client 4 disconnected
```

Рисунок 2 – действия на стороне сервера

## 4 ВЫВОД

В ходе выполнения лабораторной работы была разработана многопоточная клиент-серверная программа на C, демонстрирующая принципы сетевого взаимодействия и одновременного обслуживания нескольких клиентов.

Сервер, реализованный с использованием `POSIX Threads`, создает отдельный поток для каждого входящего соединения, обеспечивая независимые сеансы для клиентов, включая уникальный текущий каталог для каждого. Программа реализует простой текстовый протокол, позволяющий клиентам выполнять эхо-запросы (`ECHO`), запрашивать общую информацию о сервере (`INFO`), изменять текущий каталог (`CD`) и получать список файлов и подкаталогов (`LIST`) в пределах заданного корневого каталога сервера. Особенностью реализации является строгое ограничение файловых операций в рамках корневого каталога, что предотвращает несанкционированный доступ, а также поддержка вывода информации о символических ссылках. Весь процесс работы сервера протоколируется в стандартный вывод с подробными метками времени, что позволяет отслеживать активность клиентов и сервера в реальном времени.