**ОБЩАЯ ТЕОРИЯ**

**LAB 7.1**

*1. UDP - эхо сервер:*

*a. Сделайте UDP-сервер, который принимает данные от клиентов и*

*пересылает их обратно клиенту.*

**UDP** — это протокол транспортного уровня, который обеспечивает передачу данных без установки соединения, в отличие от TCP. Он быстрый, но не гарантирует доставку данных, их порядок или целостность. Передает датаграммы.

**Эхо-сервер** — это сервер, который принимает сообщения от клиента и отправляет их обратно в неизменном виде, демонстрируя двустороннюю связь.

**Сокет** — это программный интерфейс для сетевого взаимодействия. В данном случае он представляет собой точку, через которую программа может отправлять и принимать данные по сети (например, через UDP или TCP).

В языке C **дескриптор сокета** — это целое число, которое операционная система использует для идентификации конкретного сокета. Оно похоже на файловый дескриптор, но предназначено для сетевых операций.

Сокет создается с помощью функции *socket()* из библиотеки <sys/socket.h>. Эта функция возвращает дескриптор (целое число) или -1 в случае ошибки.

*struct sockaddr\_in* — это структура, определенная в <arpa/inet.h>, которая используется для хранения информации об адресе в сетевом программировании (для протокола IPv4).

server\_addr и client\_addr — это две переменные типа struct sockaddr\_in, которые хранят:

- server\_addr: адрес и порт сервера (например, IP-адрес и порт, на котором сервер принимает сообщения).

- client\_addr: адрес и порт клиента, который отправляет сообщения серверу или получает ответы.

Структура sockaddr\_in выглядит так:

struct sockaddr\_in {

sa\_family\_t sin\_family; // Семейство адресов (AF\_INET для IPv4)

in\_port\_t sin\_port; // Номер порта (в сетевом порядке байтов)

struct in\_addr sin\_addr; // IP-адрес (структура для IPv4)

char sin\_zero[8]; // Заполнение для выравнивания (не используется)

};Поля структуры:

- sin\_family: Указывает семейство адресов. Для IPv4 это AF\_INET.

- sin\_port: Порт, на котором работает сервер или клиент (например, 12345). Хранится в сетевом порядке байтов (big-endian), поэтому используется функция htons() для преобразования.

- sin\_addr: Структура struct in\_addr, содержащая 32-битный IPv4-адрес. Поле sin\_addr.s\_addr задает IP-адрес (например, INADDR\_ANY для сервера или inet\_addr("127.0.0.1") для клиента).

- sin\_zero: Массив из 8 байт, который обычно обнуляется. Используется для выравнивания структуры до размера struct sockaddr (универсальной структуры для адресов).

Пример заполнения:

struct sockaddr\_in server\_addr;

memset(&server\_addr, 0, sizeof(server\_addr)); // Обнуляем структуру

server\_addr.sin\_family = AF\_INET; // IPv4

server\_addr.sin\_addr.s\_addr = INADDR\_ANY; // Любой IP-адрес (0.0.0.0)

server\_addr.sin\_port = htons(12345); // Порт 12345

socklen\_t addr\_len = sizeof(client\_addr);*socklen\_t* — это специальный тип данных, определённый в заголовочном файле <sys/socket.h>. Он используется для представления размеров структур адресов в сетевых функциях, таких как bind(), connect(), recvfrom(), sendto() и других.

Сетевые функции, такие как recvfrom(), требуют указания размера структуры адреса (например, struct sockaddr\_in) в виде параметра.

bind(socket\_fd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr))*bind()* — это функция из API сокетов, которая привязывает сокет к конкретному адресу и порту, чтобы он мог принимать входящие соединения или датаграммы. В данном случае она используется для настройки сервера, чтобы он "слушал" входящие UDP-датаграммы на указанном IP-адресе и порте (например, 0.0.0.0:12345).

recvfrom(socket\_fd, buffer, BUFFER\_SIZE, 0, (struct sockaddr \*)&client\_addr, &addr\_len)*recvfrom()* — это функция из API сокетов (определена в <sys/socket.h>), которая используется для получения данных из UDP-сокета.

Она принимает входящую датаграмму, сохраняет её в буфер и заполняет информацию об адресе отправителя (клиента).

recvfrom() — это системный вызов, который:

1. Проверяет, есть ли входящие датаграммы в очереди сокета socket\_fd.

2. Если данных нет, блокирует выполнение (в блокирующем режиме), пока не придёт датаграмма.

3. Когда датаграмма получена, копирует её содержимое (до BUFFER\_SIZE байт) в buffer.

4. Заполняет client\_addr информацией об отправителе (IP и порт).

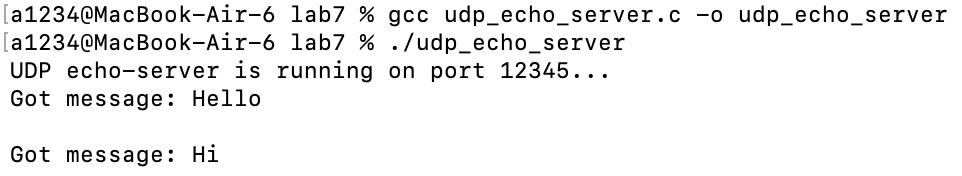
5. Обновляет addr\_len (если нужно) и возвращает количество полученных байт.

sendto(socket\_fd, buffer, n, 0, (struct sockaddr \*)&client\_addr, addr\_len);*sendto()* — это системный вызов из API сокетов (определён в <sys/socket.h>), который отправляет данные по UDP-сокету указанному получателю. В данном коде сервер отправляет содержимое buffer (полученное от клиента) обратно клиенту, используя адрес клиента, сохранённый в client\_addr. Это реализует эхо-функциональность: сервер возвращает клиенту те же данные, которые получил.  
*b. Напишите UDP-клиента, для теста UDP-сервера.*

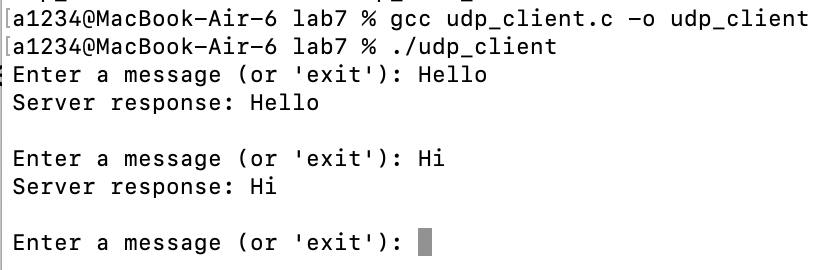
*inet\_addr()* — функция из <arpa/inet.h>, которая преобразует строку с IPv4-адресом в формате "a.b.c.d" (например, "127.0.0.1") в 32-битное число в сетевом порядке байтов.

127.0.0.1 — это localhost, адрес loopback-интерфейса, означающий "этот компьютер".

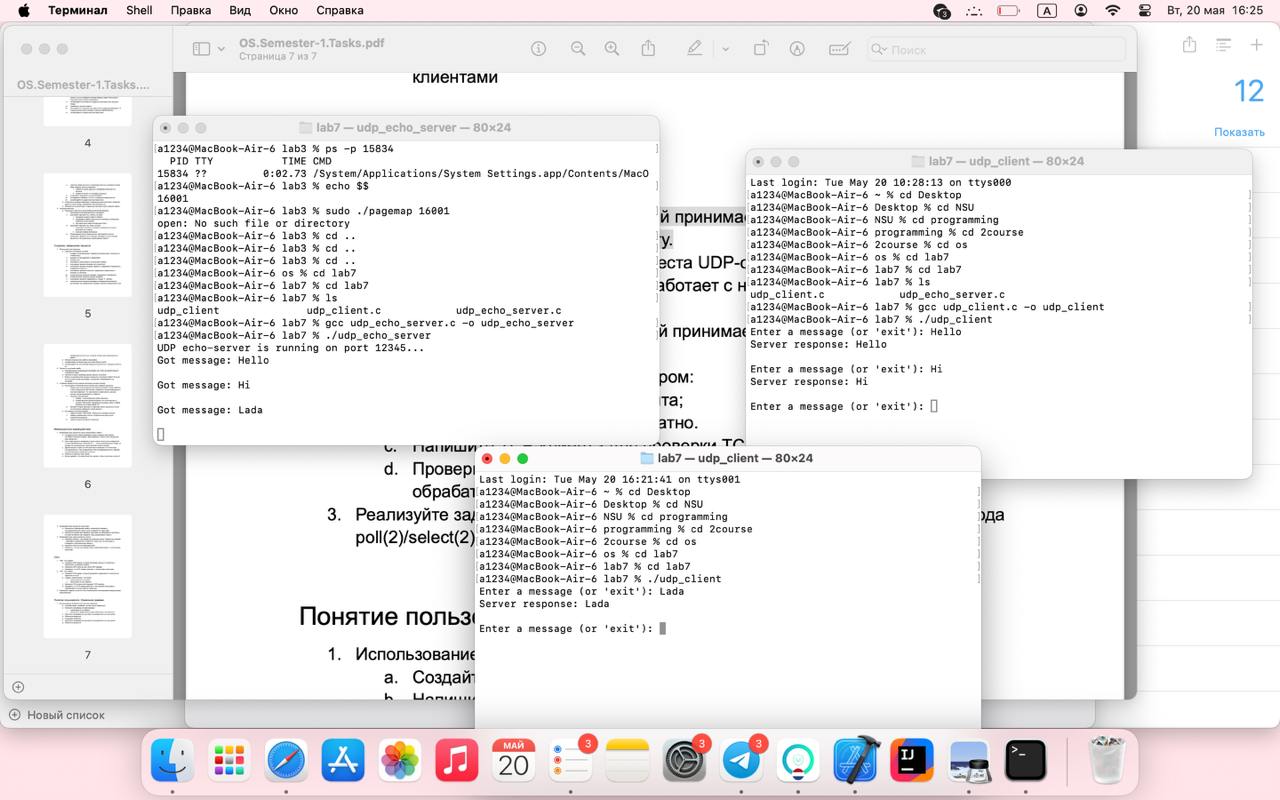
fgets(buffer, BUFFER\_SIZE, stdin);fgets() — библиотечная функция из <stdio.h>, которая считывает строку из входного потока (stdin, обычно клавиатура) и сохраняет её в массив buffer. Считывает до BUFFER\_SIZE символов или до символа новой строки (\n), включая \n в буфер, и добавляет нулевой терминатор (\0).

**Server  
**

**Client**

****

*c. Проверьте, что UDP-сервер, работает с несколькими клиентами.*

****

**Посмотреть, что такое UNIX-сокеты!!!**