

Межпроцессное взаимодействие в ОС Linux: сокеты

Цель работы: Разработка простых сетевых приложений с использованием механизма сокетов.

Задание 1.

1.1. Выполните начальную версию программ сервера **server.c** и клиента **client.c** на своем компьютере, запустив их из разных окон, первой запустите сервер. Тексты этих программ следует скопировать из каталога данной лаб. работы. В качестве номера порта сервера задайте **30 000** плюс номер Вашего ПК (1 - 26). Этот номер следует задать в константе **SERV_PORT** в программе сервера перед его компиляцией. При запуске клиента задайте в командной строке два параметра: адрес сервера **127.0.0.1** (это адрес петли обратной связи для локальной отладки сетевых приложений) и номер порта сервера (30 000 плюс номер Вашего ПК). *Что произойдет, если сервер запустить позже клиента; если завершить сервер во время работы клиента?*

```
ant_daddy@Dmitriy lr7 % ./server
connection established
Hello
[
ant_daddy@Dmitriy lr7 %
ant_daddy@Dmitriy lr7 %
ant_daddy@Dmitriy lr7 % ./client 127.0.0.1 30001
Input message to send
Hello
ant_daddy@Dmitriy lr7 %
```

Во время выполнения клиента и сервера получите информацию об активных сокетах:

\$netstat -a

Active Internet connections (including servers)					
Proto	Recv-Q	Send-Q	Local Address	Foreign Address	(state)
tcp4	0	0	localhost.pago-service	localhost.61970	ESTABLISHED
tcp4	0	0	localhost.61970	localhost.pago-service	ESTABLISHED
tcp4	0	0	*.pago-services1	.*	LISTEN
tcp6	0	0	2a03:d000:4224:5.61969	storage.mds.yand.https	ESTABLISHED
tcp4	31	0	192.168.222.87.61968	portal.mail.ru.https	CLOSE_WAIT
tcp4	0	0	192.168.222.87.61967	e.mail.ru.https	ESTABLISHED
tcp6	0	0	2a03:d000:4224:5.61966	s410vla.storage..https	ESTABLISHED
tcp6	0	0	2a03:d000:4224:5.61965	avatars.mds.yand.https	ESTABLISHED
tcp6	0	0	2a03:d000:4224:5.61960	dns.google.https	ESTABLISHED
tcp4	0	0	localhost.ipp	.*	LISTEN
tcp6	0	0	localhost.ipp	.*	LISTEN
tcp4	0	0	192.168.222.87.61955	149.154.167.151.https	ESTABLISHED
tcp4	0	0	192.168.222.87.61953	srv72-132-240-87.https	ESTABLISHED
tcp6	0	0	2a03:d000:4224:5.61896	mc.yandex.ru.https	ESTABLISHED
tcp6	0	0	2a03:d000:4224:5.61869	mc.yandex.ru.https	ESTABLISHED
tcp6	0	0	2a03:d000:4224:5.61859	mc.yandex.ru.https	ESTABLISHED
tcp4	0	0	192.168.222.87.61769	rebus.e.mail.ru.https	ESTABLISHED
tcp4	0	0	192.168.222.87.61757	149.154.167.41.https	ESTABLISHED
tcp6	0	0	2a03:d000:4224:5.61693	yandex.ru.https	ESTABLISHED
tcp6	0	0	2a03:d000:4224:5.61690	rm7jryoszczrt5og.https	ESTABLISHED
tcp6	0	0	2a03:d000:4224:5.61686	music.yandex.ru.https	ESTABLISHED
tcp4	0	0	192.168.222.87.61580	149.154.167.41.https	ESTABLISHED
tcp4	0	0	192.168.222.87.61545	is-radar17-vip-s.https	ESTABLISHED
tcp6	0	0	2a03:d000:4224:5.61400	lm-in-f188.1e100.5228	ESTABLISHED
tcp4	0	0	192.168.222.87.61334	149.154.167.51.https	ESTABLISHED
tcp6	0	0	fe80::aede:48ff:.61220	fe80::aede:48ff:.49166	ESTABLISHED
tcp6	0	0	fe80::aede:48ff:.61147	fe80::aede:48ff:.49169	ESTABLISHED
tcp4	0	0	192.168.222.87.61131	17.57.146.133.5223	ESTABLISHED
tcp4	0	0	192.168.222.87.61128	149.154.167.51.https	ESTABLISHED
tcp6	0	0	*.60837	.*	LISTEN
tcp4	0	0	*.60837	.*	LISTEN
tcp6	0	0	fe80::aede:48ff:.60836	fe80::aede:48ff:.49155	ESTABLISHED
tcp6	0	0	fe80::aede:48ff:.60814	fe80::aede:48ff:.49154	ESTABLISHED
tcp6	0	0	fe80::aede:48ff:.49158	fe80::aede:48ff:.49180	ESTABLISHED
tcp6	0	0	fe80::aede:48ff:.56805	fe80::aede:48ff:.49169	ESTABLISHED
tcp6	0	0	fe80::aede:48ff:.60760	fe80::aede:48ff:.49169	ESTABLISHED
tcp6	0	0	fe80::aede:48ff:.60554	fe80::aede:48ff:.49166	ESTABLISHED
tcp6	0	0	fe80::aede:48ff:.59032	fe80::aede:48ff:.49169	ESTABLISHED

Пояснение. Первая группа сокетов, выводимая по этой команде, - это сокет Интернет, вторая группа - сокет UNIX. Найдите свои сокет в первой группе. Колонка **Local Address** содержит адрес локального конца сокета (на Вашем компьютере), колонка **Foreign Address** содержит адрес удаленного конца сокета (на компьютере, с которым взаимодействует Ваш сокет). Адрес состоит из двух частей: **IP-адрес.порт**. Звездочка в первой части адреса **Local Address** Вашего прослушивающего сокета обозначает **INADDR_ANY**, т.е. универсальный IP-адрес, и означает, что сервер будет принимать соединения, поступающие для любого IP-адреса данного компьютера и для номера порта сервера, указанного после точки (Это номер, заданный в **SERV_PORT**.). Звездочки в обеих частях адреса **Foreign Address** Вашего прослушивающего сокета обозначают универсальный IP-адрес и любой порт, и означают, что

сервер будет принимать соединения от клиента с любым IP-адресом и любым номером порта.

tcp4	0	0	localhost.ipp	.*	LISTEN
tcp6	0	0	localhost.ipp	.*	LISTEN

tcp6	0	0	*.60837	.*	LISTEN
tcp4	0	0	*.60837	.*	LISTEN

2 сокета: один слушающий сокет для прослушивания входящих подключений, один клиентский сокет для установления соединения с сервером.

Сокет для прослушивания lfd создан: `socket(AF_INET, SOCK_STREAM, 0)`

Связывается с адресом сервера с помощью `bind()`.

При каждом соединении создается новый сокет cfd: `accept(lfd, (struct sockaddr *)&cliaddr, &clilen)`.

После завершения общения с клиентом, закрывается с помощью `close(cfd)`.

Если бы клиент и сервер выполнялись на разных компьютерах:

На серверном компьютере был бы один слушающий сокет для прослушивания входящих подключений.

На клиентском компьютере был бы один клиентский сокет для установления соединения с сервером.

Выпишите в отчет информацию о Ваших сокетах. Сколько их? *Объясните*, когда и какими функциями они были созданы, в каких состояниях находятся, какова пара адресов для каждого из сокетов? Если бы клиент и сервер выполнялись на разных компьютерах, сколько сокетов Вы увидели бы на клиентском и серверном компьютерах?

1.2. Выполните программы клиента и сервера на разных компьютерах. Для этого следует запустить сервер на своем компьютере, а программу клиента - с

одного из соседних компьютеров (клиенты на всех компьютерах одинаковы), указав в командной строке IP-адрес Вашего компьютера (ниже объясняется, как определить IP-адрес) и номер порта сервера на Вашем компьютере (30 000 + номер Вашего ПК).

Определение IP-адресов компьютеров в локальной сети.

Получите информацию о сетевых интерфейсах Вашего компьютера при помощи команды

\$netstat -nie

Вы должны увидеть два интерфейса: **lo** (интерфейс обратной связи, или закольцовки) и **eth0** (локальная сеть *Ethernet*) со следующей информацией об интерфейсах: **inet addr** - IP адрес Вашего компьютера, **Linc encap** - используемая технология канального уровня, **Bcast** - широковещательный IP-адрес, **Hwaddr** - физический адрес сетевой карты, **Mask** - маска подсети, **MTU** - максимальный размер кадра. *Запишите и объясните эту информацию в отчете.*

```
ant_daddy — -zsh — 109x59
Q~ en1
Last login: Fri Jun 21 14:00:15 on ttys000
ant_daddy@Dmitriy ~ % ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=1203<RXCSUM, TXCSUM, TXSTATUS, SW_TIMESTAMP>
    inet 127.0.0.1 netmask 0xff000000
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
    nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en5: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether ac:de:48:00:11:22
    inet6 fe80::aede:48ff:fe00:1122%en5 prefixlen 64 scopeid 0x4
    nd6 options=201<PERFORMNUD,DAD>
    media: autoselect (100baseTX <full-duplex>)
    status: active
ap1: flags=8802<BROADCAST,SIMPLEX,MULTICAST> mtu 1500
    options=400<CHANNEL_IO>
    ether 3e:22:fb:bf:40:ff
    media: autoselect
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
    ether 3c:22:fb:bf:40:ff
    inet6 fe80::1882:a9c0:7353:9615%en0 prefixlen 64 secured scopeid 0x6
    inet 192.168.222.87 netmask 0xfffff00 broadcast 192.168.222.255
    inet6 2a03:d000:4224:5a0b:cad:c9db:25fd:d63b prefixlen 64 autoconf secured
    inet6 2a03:d000:4224:5a0b:dcc8:94c:ea0f:d9a3 prefixlen 64 autoconf temporary
    nd6 options=201<PERFORMNUD,DAD>
    media: autoselect
    status: active
awdl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
    ether ba:ee:ad:6c:8b:6a
    inet6 fe80::b8ee:adff:fe6c:8b6a%awdl0 prefixlen 64 scopeid 0x7
    nd6 options=201<PERFORMNUD,DAD>
    media: autoselect
    status: active
llw0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=400<CHANNEL_IO>
    ether ba:ee:ad:6c:8b:6a
    inet6 fe80::b8ee:adff:fe6c:8b6a%llw0 prefixlen 64 scopeid 0x8
    nd6 options=201<PERFORMNUD,DAD>
    media: autoselect
    status: inactive
en4: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    options=460<TSO4,TSO6,CHANNEL_IO>
    ether 5a:27:b3:be:b3:40
    media: autoselect <full-duplex>
    status: inactive
en11: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    options=460<TSO4,TSO6,CHANNEL_IO>
    ether 5a:27:b3:be:b3:45
    media: autoselect <full-duplex>
    status: inactive
en10: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    options=460<TSO4,TSO6,CHANNEL_IO>
    ether 5a:27:b3:be:b3:44
    media: autoselect <full-duplex>
    status: inactive
```

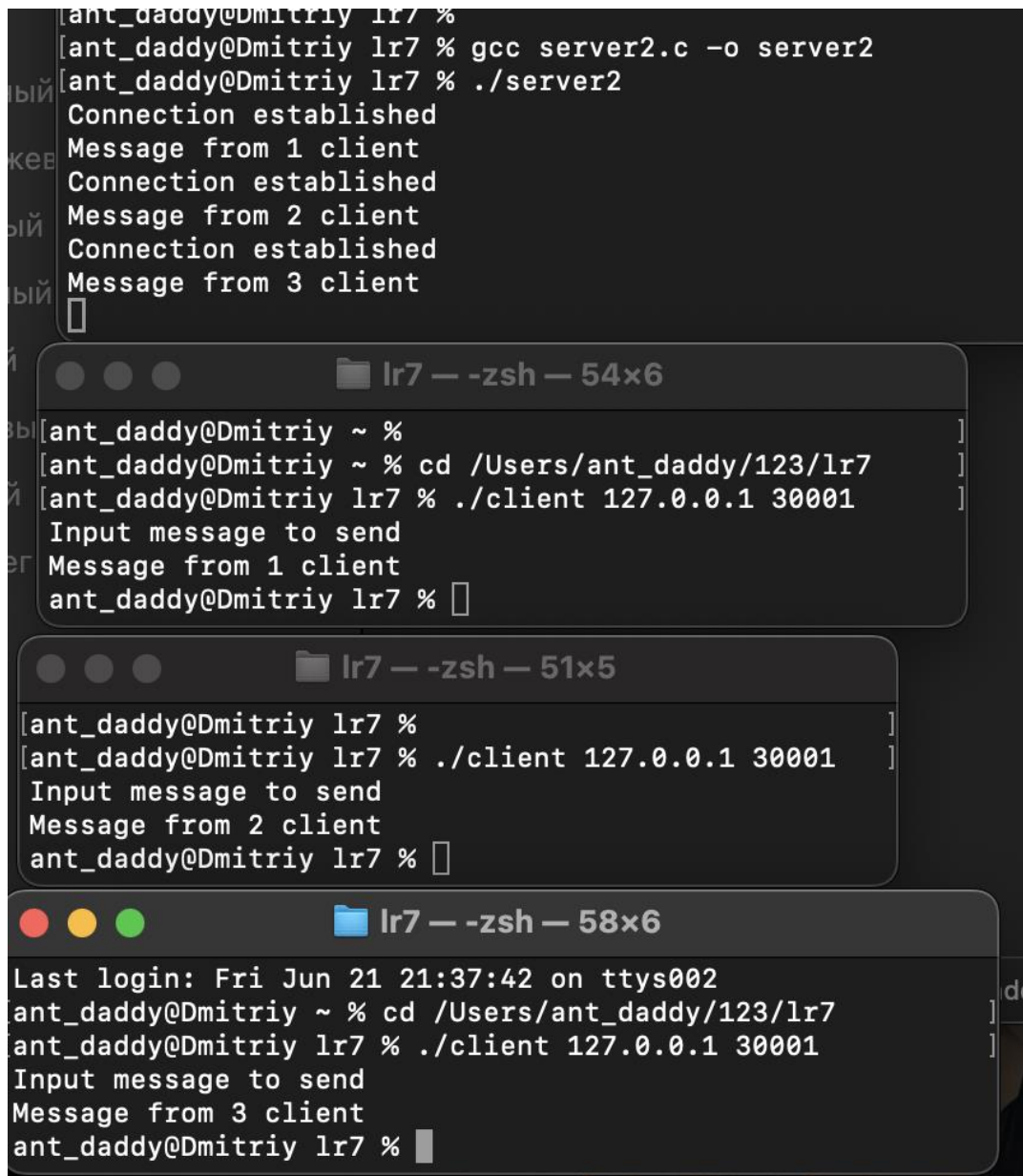
inet addr (IPv4-адрес) - 192.168.222.87

Hwaddr - 5a:27:b3:be:b3:45

Mask - 0xfffff00

MTU – 1500

1.3. Модифицируйте программу сервера (дайте ей имя **server2.c**): перед выводом сообщения на экран добавьте небольшую задержку (10 с), имитирующую процесс обработки запроса сервером. Запустите сервер в одном окне, а из других окон одновременно запустите несколько клиентов.



The screenshot displays three overlapping terminal windows on a dark background. The top window, titled 'lr7 - zsh - 54x6', shows the server's execution: compilation of 'server2.c' into 'server2', followed by running the server which prints 'Connection established' and 'Message from' for three clients. The middle window, titled 'lr7 - zsh - 51x5', shows a client running './client 127.0.0.1 30001' and printing 'Input message to send' and 'Message from 2 client'. The bottom window, titled 'lr7 - zsh - 58x6', shows another client running the same command and printing 'Input message to send' and 'Message from 3 client'. All windows show the user 'ant_daddy@Dmitriy'.

```
[ant_daddy@Dmitriy lr7 %  
[ant_daddy@Dmitriy lr7 % gcc server2.c -o server2  
[ant_daddy@Dmitriy lr7 % ./server2  
Connection established  
Message from 1 client  
Connection established  
Message from 2 client  
Connection established  
Message from 3 client  
[  
[ant_daddy@Dmitriy ~ %  
[ant_daddy@Dmitriy ~ % cd /Users/ant_daddy/123/lr7  
[ant_daddy@Dmitriy lr7 % ./client 127.0.0.1 30001  
Input message to send  
Message from 1 client  
ant_daddy@Dmitriy lr7 %  
[ant_daddy@Dmitriy lr7 %  
[ant_daddy@Dmitriy lr7 % ./client 127.0.0.1 30001  
Input message to send  
Message from 2 client  
ant_daddy@Dmitriy lr7 %  
Last login: Fri Jun 21 21:37:42 on ttys002  
ant_daddy@Dmitriy ~ % cd /Users/ant_daddy/123/lr7  
ant_daddy@Dmitriy lr7 % ./client 127.0.0.1 30001  
Input message to send  
Message from 3 client  
ant_daddy@Dmitriy lr7 %
```

1.4. Модифицируйте сервер (имя **server3.c**): сделайте Ваш последовательный сервер параллельным, обслуживая каждого клиента в отдельном процессе сервера. Для этого после **accept()** создайте дочерний процесс, в который переместите те строки кода, в которых сервер выводит сообщение об установлении соединения, читает из сокета полученные данные,

ожидает 10 с, выводит данные на экран и закрывает присоединенный сокет. Кроме этого, в начале дочернего процесса закройте прослушиваемый сокет, а в конце дочернего процесса добавьте **exit(0)**. Основной цикл родительского сервера должен включать: вычисление **clilen**, **accept()**, **fork()** и закрытие присоединенного сокета. Обязательно удаляйте завершившиеся дочерние процессы, обрабатывая сигнал SIGCHLD.

Выполните параллельный сервер с несколькими одновременно запущенными клиентами. *Поясните в отчете модификации в **server3.c**. Нарисуйте временные диаграммы работы последовательного и параллельного серверов при обслуживании нескольких клиентов*

```
server3.c

int lfd, cfd;
int nread;
unsigned int cliilen;
char buf[BUFSIZE];
struct sockaddr_in servaddr, cliaddr;

signal(SIGCHLD, sigchld_handler);

if ((lfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
    perror("socket");
    exit(1);
}

memset(&servaddr, 0, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(SERV_PORT);

if (bind(lfd, (struct sockaddr *)&servaddr, sizeof(struct sockaddr_in)) < 0)
{
    perror("bind");
    exit(1);
}

if (listen(lfd, 5) < 0)
{
    perror("listen");
    exit(1);
}

for (;;)
{
    cliilen = sizeof(cliaddr);

    if ((cfd = accept(lfd, (struct sockaddr *)&cliaddr, &cliilen)) < 0)
    {
        perror("accept");
        exit(1);
    }

    printf("Connection established\n");

    if (fork() == 0) // docher
    {
        close(lfd);

        if ((nread = read(cfd, buf, BUFSIZE)) < 0)
        {
            perror("read");
            exit(1);
        }

        sleep(10); // delay

        write(1, &buf, nread);
        close(cfd);
        exit(0);
    }

    close(cfd);
}

//close(lfd);
exit(0);
}
```



```
[ant_daddy@Dmitriy lr7 % gcc server3.c -o server3
[ant_daddy@Dmitriy lr7 % ./server3
Connection established
Connection established
Connection established
11111
222222
33333
[]

lr7 — -zsh — 54x6
Input message to send
Message from 1 client
[ant_daddy@Dmitriy lr7 % ./client 127.0.0.1 30001
Input message to send
11111
ant_daddy@Dmitriy lr7 % []

lr7 — -zsh — 51x5
Message from 2 client
[ant_daddy@Dmitriy lr7 % ./client 127.0.0.1 30001
Input message to send
222222
ant_daddy@Dmitriy lr7 % []

lr7 — -zsh — 58x6
Input message to send
Message from 3 client
[ant_daddy@Dmitriy lr7 % ./client 127.0.0.1 30001
Input message to send
33333
ant_daddy@Dmitriy lr7 % []
```

1.5. Модифицируйте параллельный сервер (дайте ему имя **server4.c**): после **connect()** выводите на экран IP-адрес клиента и номер порта клиента. Выполните этот вариант сервера и *выпишите результаты, полученные для нескольких клиентов*.

Пояснение. IP-адрес и номер порта клиента заносятся операционной системой при установлении соединения в структуру, адресованную вторым аргументом функции **accept()**. Для вывода адреса следует преобразовать поле **sin_addr** указанной структуры функцией **inet_ntop** из числового в символьный формат. Для вывода номера порта следует преобразовать поле **sin_port** структуры функцией **ntohs** из сетевого порядка следования байтов в порядок следования байтов данного компьютера. Получить адреса локального и

удаленного концов сокета можно также при помощи функций **getsockname** и **getpeername** СООТВЕТСТВЕННО.

```
server4.c — Изменено

struct sockaddr_in servaddr, cliaddr;

signal(SIGCHLD, sigchld_handler);

if ((lfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
    perror("socket");
    exit(1);
}

memset(&servaddr, 0, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(SERV_PORT);

if (bind(lfd, (struct sockaddr *)&servaddr, sizeof(struct sockaddr_in)) < 0)
{
    perror("bind");
    exit(1);
}

if (listen(lfd, 5) < 0)
{
    perror("listen");
    exit(1);
}

for (;;)
{
    clilen = sizeof(cliaddr);

    if ((cfd = accept(lfd, (struct sockaddr *)&cliaddr, &clilen)) < 0)
    {
        perror("accept");
        exit(1);
    }

    char ip_str[INET_ADDRSTRLEN];
    inet_ntop(AF_INET, &(cliaddr.sin_addr), ip_str, INET_ADDRSTRLEN);

    printf("Connection established with client: %s:%d\n", ip_str,
ntohs(cliaddr.sin_port));

    if (fork() == 0) // Child process
    {
        close(lfd); // Close listening socket in child process

        if ((nread = read(cfd, buf, BUFSIZE)) < 0)
        {
            perror("read");
            exit(1);
        }

        sleep(10); // delay

        write(1, &buf, nread);
        close(cfd);
        exit(0);
    }

    close(cfd);
}

//close(lfd);
exit(0);
}
```

```
ant_daddy@Dmitriy lr7 % gcc server4.c -o server4
ant_daddy@Dmitriy lr7 % ./server4
Connection established with client: 127.0.0.1:60042
Connection established with client: 127.0.0.1:60043
Connection established with client: 127.0.0.1:60044
11111111
22222222
33333333
[]

lr7 - -zsh - 51x5
11111111
ant_daddy@Dmitriy lr7 % ./client 127.0.0.1 30001
Input message to send
11111111
ant_daddy@Dmitriy lr7 % []

lr7 - -zsh - 54x6
Input message to send
22222222
ant_daddy@Dmitriy lr7 % ./client 127.0.0.1 30001
Input message to send
22222222
ant_daddy@Dmitriy lr7 % []

lr7 - -zsh - 58x6
Input message to send
33333333
ant_daddy@Dmitriy lr7 % ./client 127.0.0.1 30001
Input message to send
33333333
ant_daddy@Dmitriy lr7 % []
```

1.6. Запустите для Вашего сервера программу клиента ОС Windows; клиент использует WinSock API. Для этого скопируйте программу клиента (программа **WinClient** из папки **WinClient** данной лабораторной работы) на локальный диск ПК преподавателя (с ОС Windows).

Не знаю, где взять WinClient

1.7.* а) Модифицируйте программы клиента и параллельного сервера (имена **echo_server** и **echo_client**): клиент в цикле передает все строки, считанные из *stdin*, признак окончания ввода - считывание конца файла. Параллельный сервер отправляет принятые строки обратно клиенту, добавляя перед первой строкой информацию о своем адресе и адресе клиента. Клиент получает строки, распечатывает их и завершает работу.

* Необязательное задание. Оценивается дополнительными баллами при условии, что студент полностью выполнил остальные задания.

Пояснение. Когда клиент обнаружит конец файла в *stdin*, он должен выполнить частичное закрытие соединения функцией **shutdown()**. *Поясните в отчете, зачем.* Сервер, обнаружив конец файла при считывании данных из сокета, должен полностью закрыть соединение. Клиент, обнаружив, в свою очередь, конец файла при считывании данных из сокета, также должен полностью закрыть соединение.

```
echo_client.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define MAXLINE 1024

int main(int argc, char *argv[]) {
    int sockfd;
    struct sockaddr_in servaddr;

    if (argc != 3) {
        fprintf(stderr, "Usage: %s <IP address> <Port>\n", argv[0]);
        exit(1);
    }

    sockfd = socket(AF_INET, SOCK_STREAM, 0);

    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);
    servaddr.sin_port = htons(atoi(argv[2]));

    connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

    char buffer[MAXLINE];

    while (fgets(buffer, MAXLINE, stdin) != NULL) {
        write(sockfd, buffer, strlen(buffer));
    }

    // Частичное закрытие соединения
    shutdown(sockfd, SHUT_WR);

    close(sockfd);

    return 0;
}
```

```

echo_server.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define MAXLINE 1024

int main(int argc, char *argv[]) {
    int listenfd, connfd;
    struct sockaddr_in servaddr, cliaddr;
    socklen_t len;
    char buffer[MAXLINE];

    listenfd = socket(AF_INET, SOCK_STREAM, 0);

    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(atoi(argv[1]));

    bind(listenfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

    listen(listenfd, 5);

    len = sizeof(cliaddr);
    connfd = accept(listenfd, (struct sockaddr *)&cliaddr, &len);

    // Получаем информацию о адресах
    char clientAddress[INET_ADDRSTRLEN];
    char serverAddress[INET_ADDRSTRLEN];
    inet_ntop(AF_INET, &cliaddr.sin_addr, clientAddress, INET_ADDRSTRLEN);
    inet_ntop(AF_INET, &servaddr.sin_addr, serverAddress, INET_ADDRSTRLEN);

    sprintf(buffer, "Server Address: %s\nClient Address: %s\n", serverAddress,
clientAddress);
    write(connfd, buffer, strlen(buffer));

    while (read(connfd, buffer, MAXLINE) > 0) {
        write(connfd, buffer, strlen(buffer));
        memset(buffer, 0, MAXLINE);
    }

    close(connfd);
    close(listenfd);

    return 0;
}

```

6) Модифицируйте исходную программу клиента (дайте ей имя **par_client**): клиент одновременно устанавливает пять соединений с одним и тем же сервером. В каждом соединении он посылает одно короткое сообщение, затем одновременно разрывает все соединения. Для этого удалите из текста клиента **close** для сокета, оставив **exit**. Запустите **echo_server** и **par_client** и проверьте наличие серверных процессов-"зомби". Сколько их? Влияет ли на их

количество физическое расстояние между клиентом и сервером? Добейтесь удаления всех "зомби", модифицировав обработчик сигнала SIGCHLD (см. дополнительное задание в лаб. работе 4.)

```
client.c
/*      client.c      */
#include <sys/socket.h>
#include <sys/types.h>
#include <stdio.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define BUFSIZE 100
int main(int argc, char *argv[])
{
    int fd;
    int nread;
    char buf[BUFSIZE];
    struct sockaddr_in servaddr;
    if (argc < 3) {printf("Too few arguments \n"); exit(1);}

    if ((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        {perror("socket creating"); exit(1);}
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;

    if(inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0)
        {perror("bad address"); exit(1);}

    servaddr.sin_port = htons(atoi(argv[2]));

    if (connect(fd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0)
        {perror("connect"); exit(1);}
    write(1, "Input message to send\n", 22);
    nread = read(0, buf, BUFSIZE);
    if (write(fd, buf, nread) < 0)
        {perror("write"); exit(1);}
    close(fd);
    exit(0);
}
```

Задание 2. Напишите и выполните программу клиента (**inet.c**) для web-сервера. Клиент получает в командной строке доменное имя компьютера сервера, например, **www.vc.miet.ru** и посылает запрос на получение от web-сервера любой HTML-страницы. Полученную от web-сервера информацию клиент выводит в *stdout*.

Пояснения.

1) IP-адрес web-сервера следует получить при помощи функции **gethostbyname**:

```
struct hostent* hostinfo;  
  
.....  
  
hostinfo = gethostbyname (argv[1]);  
  
if (hostinfo < 0) { Ошибка };  
  
name.sin_addr = *((struct in_addr *)hostinfo->h_addr);
```

2) Номер порта по умолчанию для web-сервера равен **80**, задайте его в виде константы.

3) В соответствии с протоколом HTTP для запроса страницы по умолчанию достаточно послать web-серверу команду **GET** без параметров, например, "**GET \n**". Для получения какой-либо другой HTML-страницы после команды **GET** следует указывать ее адрес относительно корневого каталога web-сервера. Например, если URL страницы - **www.vc.miet.ru/other/instr.html**, то следует послать сообщение "**GET /other/instr.html \n**".

4) Ответ сервера может быть прислан в нескольких сегментах TCP. Его следует читать в цикле в буфер произвольно выбранной длины; в этом же цикле выводите на экран прочитанную информацию (количество байтов, прочитанное в **read()**). Условие выхода из цикла - **read()** вернул **0**; оно является признаком конца файла и означает, что сервер разорвал соединение, т.е. закрыл присоединенный сокет.

После отладки запустите программу **inet**, перенаправив **stdout** в какой-либо файл, затем просмотрите этот файл в браузере. Почему вид этого файла отличается от того вида, какой Вы получаете, просматривая данную HTML-страницу обычным способом из браузера?

```

inet.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define DEFAULT_PORT 80

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s hostname\n", argv[0]);
        exit(1);
    }

    struct hostent *server;
    struct sockaddr_in serv_addr;
    int sockfd, portno = DEFAULT_PORT;
    char buffer[4096];

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("Error opening socket");
        exit(1);
    }

    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr, "Error, no such host\n");
        exit(1);
    }

    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr, server->h_length);
    serv_addr.sin_port = htons(portno);

    if (connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
        perror("Error connecting");
        exit(1);
    }

    char *request = "GET / HTTP/1.1\r\nHost: %s\r\n\r\n";
    sprintf(buffer, request, argv[1]);
    write(sockfd, buffer, strlen(buffer));

    bzero(buffer, 4096);
    int bytes_read;
    while ((bytes_read = read(sockfd, buffer, 4095)) > 0) {
        printf("%.5s", bytes_read, buffer);
        bzero(buffer, 4096);
    }

    close(sockfd);

    return 0;
}

```

<https://miet.ru/>

```
[ant_daddy@Dmitriy lr7 % ./inet www.miet.ru
[HTTP/1.1 301 Moved Permanently
Server: nginx
[Date: Mon, 24 Jun 2024 08:52:39 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Keep-Alive: timeout=20
Location: https://www.miet.ru/

<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

<https://vk.com>

```
ant_daddy@Dmitriy lr7 % ./inet www.vk.com
[HTTP/1.1 301 Moved Permanently
Server: kittenx
[Date: Mon, 24 Jun 2024 08:52:05 GMT
Content-Type: text/html
Content-Length: 164
Connection: keep-alive
Location: https://www.vk.com/
X-Frontend: front656700
Access-Control-Expose-Headers: X-Frontend
X-Trace-Id: CRnpUF91_6aBrCopCkt0Ggoub5BAgA

<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>kittenx</center>
</body>
</html>
^C
```

<https://www.apple.com/>

```
ant_daddy@Dmitriy lr7 % ./inet www.apple.com ]
HTTP/1.1 301 Moved Permanently
Server: AkamaiGHost
Content-Length: 0
Location: https://www.apple.com/
Cache-Control: max-age=0
Expires: Mon, 24 Jun 2024 08:54:24 GMT
Date: Mon, 24 Jun 2024 08:54:24 GMT
X-Cache: TCP_MISS from a139-45-207-53.deploy.akamaitechnologies.com (AkamaiGHost
/11.5.2-56372494) (-)
Connection: keep-alive
strict-transport-security: max-age=31536000
Set-Cookie: geo=RU; path=/; domain=.apple.com
```

^C

```
ant_daddy@Dmitriy lr7 % ./inet www.stat.mil.ru ]
Error, no such host
ant_daddy@Dmitriy lr7 % ./inet www.government.ru/ ]
Error, no such host
ant_daddy@Dmitriy lr7 % ./inet www.government.ru ]
HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Mon, 24 Jun 2024 09:01:33 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Keep-Alive: timeout=60
Location: http://government.ru/
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
```

```
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
ant_daddy@Dmitriy lr7 % █
```