

# Лекция14

## Обмен данными между процессами (продолжение).

### Сокеты

***Сокет*** – это устройство двунаправленного действия, предназначенное для передачи данных между процессами на одном или разных узлах компьютерной сети.

Для обмена данных между процессами даже на одном компьютере необходимо, чтобы на нем была установлена *сетевая карта* и *сетевые протоколы*. Чаще всего, но не обязательно, используется стек протоколов TCP/IP.

# Самая «сырая» программа

```
#include <stdio.h>
#include <sys/socket.h>
#include <linux/if_ether.h>

int main(){  int sd, bytes_read;
    char data[1024];

    sd = socket(PF_INET, SOCK_PACKET, htons(ETH_P_ALL));

    do{
        bytes_read = recvfrom(sd, data, sizeof(data), 0, 0, 0);
        if ( bytes_read > 0 )
            fwrite(data, 1, bytes_read, stdout);
    }while ( bytes_read > 0 );

    return 0;
}
```

При создании сокета задаются три параметра: *пространство имен, тип взаимодействия и протокол.*

***Пространство имен (каким образом записываются адреса):***

<b>Значение</b>	<b>Описание</b>
PF_INET	Протоколы семейства IPv4; TCP/IP (в настоящее время может использоваться синоним AF_INET)
PF_LOCAL	Локальные именованные каналы в стиле BSD
PF_IPX	Протоколы Novell
PF_INET6	Протоколы семейства IPv6; TCP/IP

## ***Тип взаимодействия:***

<b>Значение</b>	<b>Описание</b>
SOCK_STREAM	Протокол последовательной передачи данных в виде байтового потока с подтверждением доставки (TCP)
SOCK_RDM	Протокол пакетной передачи данных с подтверждением доставки
SOCK_DGRAM	Протокол пакетной передачи данных без подтверждения доставки (UDP)
SOCK_RAW	Протокол передачи низкоуровневых данных без подтверждения доставки

## Ethernet - кадр

MAC-адрес получателя	MAC-адрес отправителя	Тип Eth	Данные (IP-пакет)	CRC
6 байт	6	2	46 - 15000	4

***IP-пакет:***  
 базовый пакет  
 сетевого  
 (межсетевого  
 уровня)

Версия		Длина		Тип службы	
Полная длина					
Идентификатор					
0	DF	MF	Смещение фрагмента		
Число переходов				Протокол	
Контрольная сумма заголовка					
IP-адрес отправителя					
IP-адрес получателя					
Параметры (до 40 байт)					
Данные (до 65535 байт без заголовка)					

```

struct ip_packet {
struct {
uchar dst_eth[ETH_SIZE];
uchar src_eth[ETH_SIZE];
uchar __unkwn[2];
} hw_header;      /* MAC адреса */
uint header_len:4; /* длина заголовка в 32 разрядных словах*/
uint version:4;    /* версия IP протокола */
uint serve_type:8; /* тип обслуживания пакета */
uint packet_len:16; /* размер пакета в байтах */
uint ID:16;        /* идентификатор фрагмента*/
uint frag_offset:13; /* смещение фрагмента при сборке пакета*/
uint more_frags:1; /* флаг указывающий на существование оставшихся фрагментов */
uint dont_frag:1; /* флаг разрешения фрагментации */
uint __reserved:1; /* зарезервированный бит */
uint time_to_live:8; /* количество переходов маршрутизатора*/
uint protocol:8;    /* тип протокола ICMP, UDP, TCP */
uint hdr_chksum:16; /* контрольная сумма */
uchar IPv4_src[IP_SIZE]; /* IP адрес отправителя */
uchar IPv4_dst[IP_SIZE]; /* IP адрес получателя */
uchar options[0];      /* до 40 байт */
uchar data[0];         /* до 64 KB */ /* массив нулевой длины - особенность gcc */
};

```

```

struct ip_packet *ip=(struct ip_packet *)data;

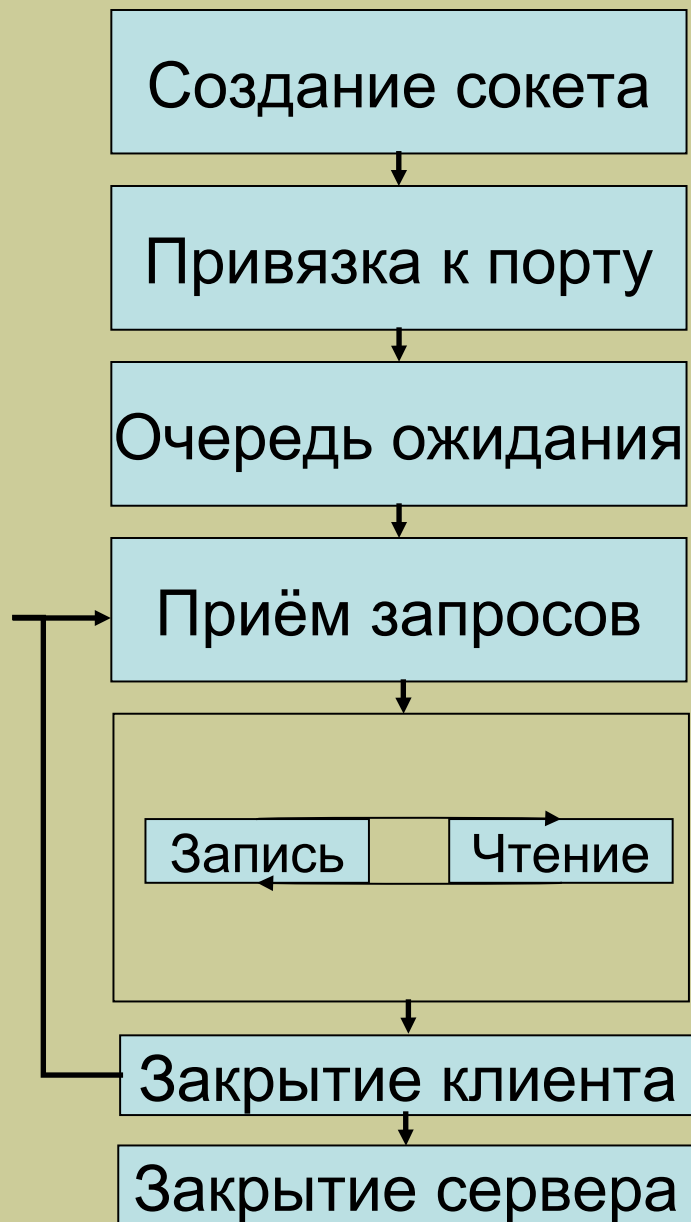
```

```

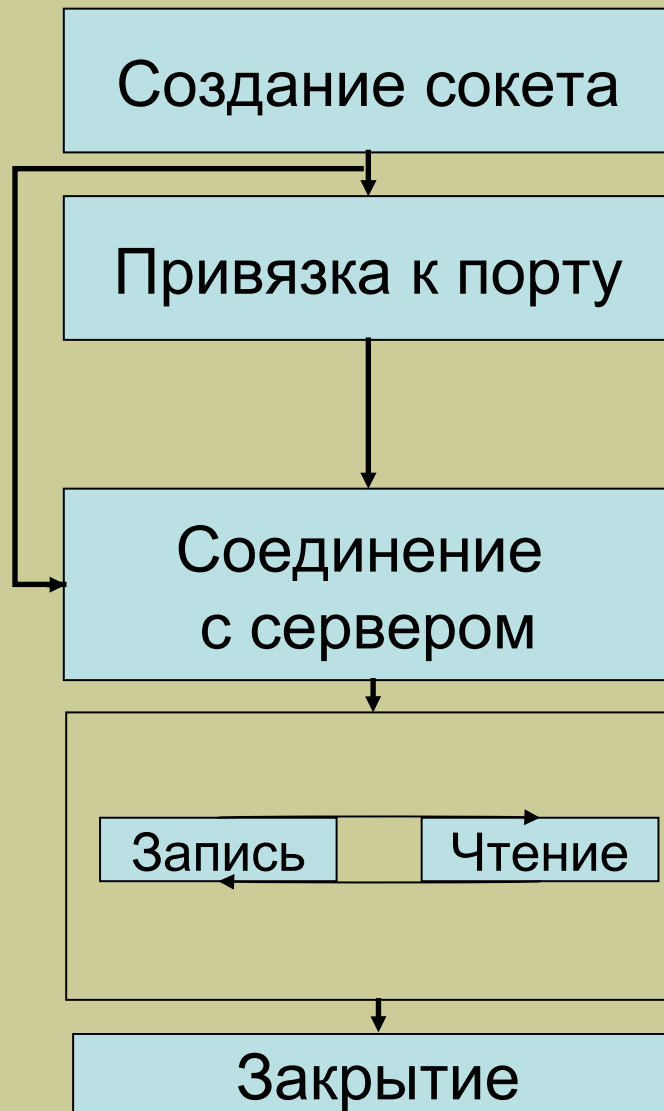
.... ip->IPv4_src;....

```

## сокет-сервер



## сокет-клиент



# Простой *http*-клиент. Отличия сокетов Беркли от *Windows* сокетов.

```
//#include <sys/socket.h>
```

```
//#include <netinet/in.h>
```

```
//#include <arpa/inet.h>
```

```
#include <Winsock2.h>
```

```
#include <string.h>
```

```
#include <stdio.h>
```

```
#define SERVERADD "217.69.139.199"//"94.100.191.203"
```

```
int main(){
```

```
    SOCKET socket_fd; //int socket_fd;
```

```
    struct sockaddr_in name;
```

```
    char buff[1024];
```

```
    char buffer[10000];
```

```
    size_t num_char;
```

```
struct hostent* host;
```

Файл *s1.c*



//Инициализация библиотеки Winsock2

```
if (WSAStartup(0x202,(WSADATA *)&buff[0]))  
{  
    printf("WSAStart error %d\n",WSAGetLastError());  
    return -1;  
}
```

~~host = gethostbyname( "www.mail.ru" );~~

socket\_fd=socket(PF\_INET, SOCK\_STREAM, 0);

name.sin\_family=AF\_INET;

name.sin\_port=htons(82);

```
//inet_aton("217.69.139.199",&name.sin_addr);  
if (inet_addr("217.69.139.199")!=INADDR_NONE)  
    name.sin_addr.s_addr=inet_addr("217.69.139.199");  
else{  
    printf("no such address\n");  
return -2;  
}  
  
if(connect(socket_fd,(struct sockaddr*)&name,sizeof(name))==  
                                SOCKET_ERROR){  
    printf("Couldn't connect to server\n");  
    return -1;  
}  
sprintf(buffer,"GET /\n");  
  
send(socket_fd,buffer,strlen(buffer),0);
```

```
while(1){  
    num_char=recv(socket_fd,buffer,sizeof(buffer) - 1,0);  
    if(num_char==0){  
        closesocket(socket_fd);  
        WSACleanup();  
        return 1;  
    }  
    fwrite(buffer,sizeof(char),num_char,stdout);  
}  
closesocket(socket_fd);  
WSACleanup();  
return 0;  
}
```

***Компиляция:***

```
>cl s1.c Ws2_32.lib
```

```
#include <stdio.h>
```

```
#include <winsock2.h>
```

```
#include <windows.h>
```

```
#define MY_PORT 1952
```

```
int main(int argc, char* argv[])
```

```
{
```

```
    char buff[1024];
```

```
    SOCKET mysocket, client_socket;
```

```
    struct sockaddr_in local_addr, client_addr;
```

```
    int client_addr_size=sizeof(client_addr);
```

```
    if (WSAStartup(0x0202,(WSADATA *) &buff[0]))
```

```
    {
```

```
        printf("Error WSAStartup %d\n",
```

```
            WSAGetLastError());
```

```
        return -1;
```

```
    }
```

Файл ***srv.c***

***Сокет - сервер***

```
if ((mysocket=socket(AF_INET,SOCK_STREAM,0))<0)
{
    printf("Error socket %d\n",WSAGetLastError());
    WSACleanup();
    return -1;
}
local_addr.sin_family=AF_INET;
local_addr.sin_port=htons(MY_PORT);
local_addr.sin_addr.s_addr=0;

if (bind(mysocket,(struct sockaddr *)&local_addr,
        sizeof(local_addr))) {
    printf("Error bind %d\n",WSAGetLastError());
    closesocket(mysocket);
    WSACleanup();
    return -1;
}
```

```
if (listen(mysocket, 0x100))
{
    printf("Error listen %d\n",WSAGetLastError());
    closesocket(mysocket);
    WSACleanup();
    return -1;
}

printf("Waiting for calls\n");

while((client_socket=accept(mysocket, (struct sockaddr *)
    &client_addr, &client_addr_size)))
{
    struct hostent *hst;
    int bytes_recv;
    hst=gethostbyaddr((char *)&client_addr.sin_addr.s_addr,4,
        AF_INET);
```

```
printf("%s [%s] new connect!\n",  
(hst)?hst->h_name:"", inet_ntoa(client_addr.sin_addr));  
  
send(client_socket,"Hello a new client!\n",  
      sizeof("Hello a new client!\n"),0);  
  
while( ( bytes_recv=recv(client_socket, &buff[0], sizeof(buff),0) )  
        && bytes_recv !=SOCKET_ERROR )  
    send(client_socket, &buff[0], bytes_recv, 0);  
  
printf("Client was disconnected\n");  
closesocket(client_socket);  
}  
  
return 0;  
}
```

```
#include <stdio.h>
#include <string.h>
#include <winsock2.h>
#include <windows.h>
```

Файл ***cln.c***

```
#define PORT 1952
#define SERVERADDR "127.0.0.1"
```

***Сокет - клиент***

```
int main(int argc, char* argv[])
{
    char buff[1024];
    struct sockaddr_in dest_addr;
    SOCKET my_sock;
    struct hostent *hst;
    int nsize;
```



**//инициализация библиотеки Winsock**

```
if (WSAStartup(0x202,(WSADATA *)&buff[0]))  
{  
    printf("WSAStart error %d\n",WSAGetLastError());  
    return -1;  
}
```

**// создание сокета**

```
my_sock=socket(AF_INET,SOCK_STREAM,0);  
if (my_sock < 0){  
    printf("Socket() error %d\n",WSAGetLastError());  
    return -1;  
}
```

**//установка соединения**

```
dest_addr.sin_family=AF_INET;  
dest_addr.sin_port=htons(PORT);
```

```
// преобразование IP адреса из символьного в сетевой формат
if (inet_addr(SERVERADDR)!=INADDR_NONE)
    dest_addr.sin_addr.s_addr=inet_addr(SERVERADDR);
else
// получение IP адреса по доменному имени сервера
if (hst=gethostbyname(SERVERADDR))
    ((unsigned long *)&dest_addr.sin_addr)[0]=
    ((unsigned long **)hst->h_addr_list)[0][0];
else
{
    printf("Invalid address %s\n",SERVERADDR);
    closesocket(my_sock);
    WSACleanup();
    return -1;
}
```

**//установка соединения**

```
if (connect(my_sock,(struct sockaddr *)&dest_addr,  
           sizeof(dest_addr))) {  
    printf("Connect error %d\n",WSAGetLastError());  
    return -1;  
}  
printf("Connection with %s was established\n\  
      Type quit for quit\n\n",SERVERADDR);
```

**// чтение и передача сообщений**

```
while((nsize=recv(my_sock, &buff[0], sizeof(buff)-1,0))  
      !=SOCKET_ERROR){  
    buff[nsize]=0;  
    printf("ServerToClient:%s",buff);  
  
    printf("ClientToServer:");  
    fgets(&buff[0], sizeof(buff) - 1, stdin);
```

```
if (!strcmp(&buff[0], "quit\n")) {  
    printf("Exit...");  
    closesocket(my_sock);  
    WSACleanup();  
    return 0;  
}  
send(my_sock, &buff[0], nsize, 0);  
}
```

```
printf("Recv error %d\n", WSAGetLastError());  
closesocket(my_sock);  
WSACleanup();  
  
return -1;  
}
```

**Упражнение:** протестировать разобранные сетевые программы.

**Тема курсовой:** написать многопользовательский «чат» на основе сокетов.

**Тема курсовой:** написать многопользовательский http-сервер (web-сервер) .