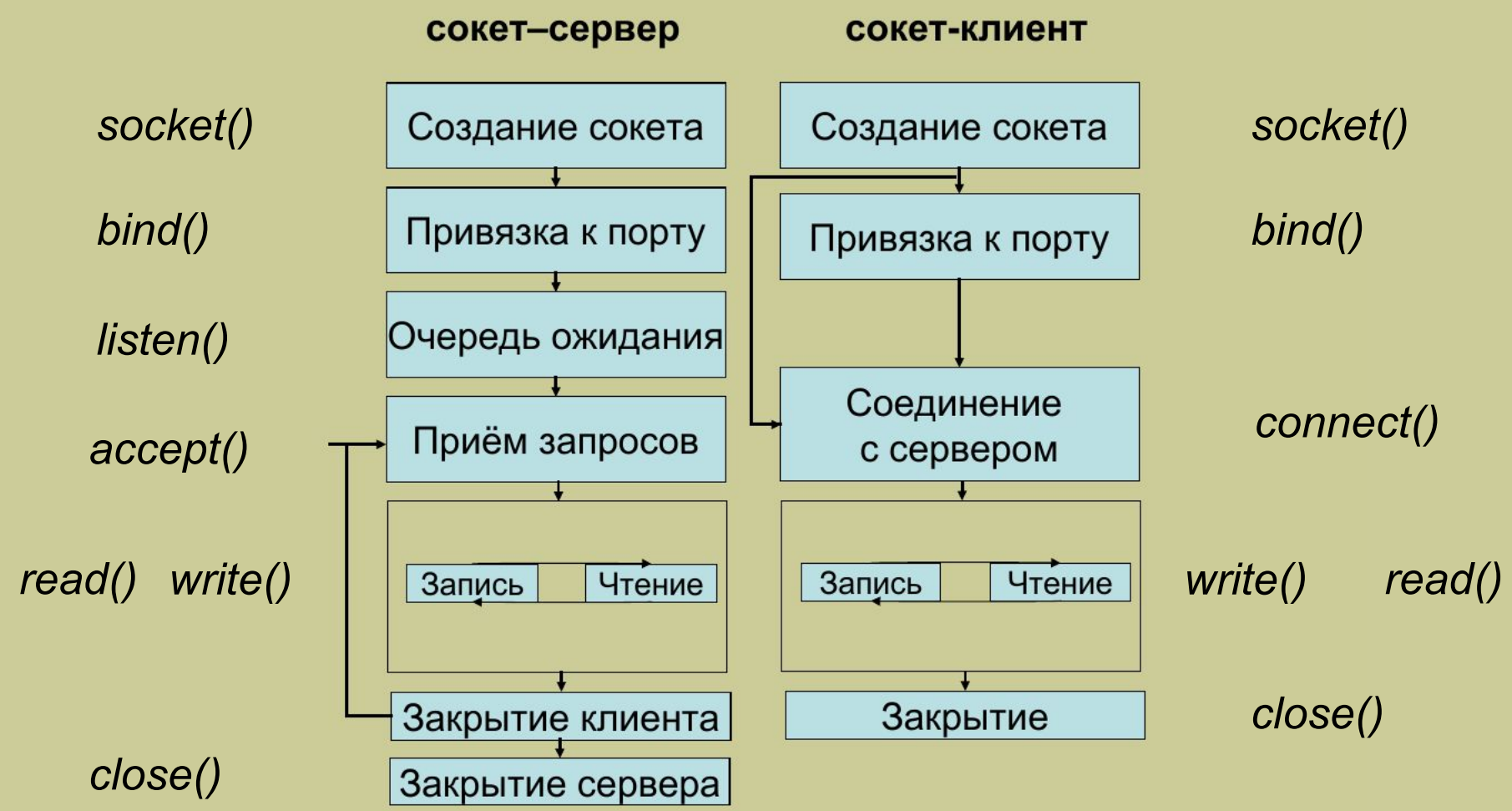
Лекция 14

- UNIX сокеты (локальные сокеты).
- Интернет сокеты.
  - IP пакеты;
  - простой веб-клиент;
  - клиент-сервер.


сокет–сервер
сокет-клиент
socket()
Создание сокета
Создание сокета
socket()
bind()
Привязка к порту
Привязка к порту
bind()
listen()
Очередь ожидания
accept()
Приём запросов
Соединение с сервером
connect()
read()
write()
Запись
Чтение
Запись
Чтение
write()
read()
Закрытие клиента
Закрытие
close()
close()
Закрытие сервера

Локальные сокеты (UNIX сокеты)

lab14S-s.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>

int srv(int cln_sock);

int main(int argc, char** argv){
  if(argc<2){
    fprintf(stderr, "USAGE: prog <socket name> ");
    return -1;
  }
```

```
const char* const sock_name=argv[1];
int sock_fd;
struct sockaddr_un  name;
int cln_sent_quit_message;

sock_fd=socket(PF_LOCAL, SOCK_STREAM, 0);
name.sun_family=AF_LOCAL;
strcpy(name.sun_path, sock_name);
bind(sock_fd, (struct sockaddr *)&name, SUN_LEN(&name));

listen(sock_fd, 5);
```

семейство адресов (семейство протоколов)

тип взаимодействия

протокол

```
struct sockaddr_un {
        sa_family_t    sun_family;     /* Address family */
        char           sun_path[];     /* Socket pathname */
    };
```

```
do{
  struct sockaddr_un    cln_name;
  socklen_t    cln_name_len;
  int   cln_sock_fd;

  cln_sock_fd=accept(sock_fd,
                     (struct sockaddr *)&cln_name,
                     &cln_name_len);
  cln_sent_quit_message=srv(cln_sock_fd);
  close(cln_sock_fd);
} while(!cln_sent_quit_message);
```

```
  close(sock_fd);
  unlink(sock_name);

  return 0;
}
```

```
int srv(int cln_sock){
  while(1){
    int length;
    char* text;
    if(read(cln_sock, &length, sizeof(length))==0)
      return 0;
    text=(char*)malloc(length);
    read(cln_sock, text, length);
```

```
    fprintf(stdout,"%s\n", text);

    if(!strcmp(text, "quit")){
      free(text);
      return 1;
    }

    free(text);
  }
}
```

lab14S-c.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>

void write_text(int , const char* );

int main(int argc, char** argv){
  if(argc<3){
    fprintf(stderr, "USAGE: prog <socket name> <message>");
    return -1;
  }
```

```
  const char* const sock_name=argv[1];
  const char* const message=argv[2];
  int sock_fd;
  struct sockaddr_un  name;

  sock_fd=socket(PF_LOCAL, SOCK_STREAM, 0);
  name.sun_family=AF_LOCAL;
  strcpy(name.sun_path, sock_name);

  connect(sock_fd, (struct sockaddr *)&name, SUN_LEN(&name));
  write_text(sock_fd, message);

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

```
void write_text(int sock_fd, const char* text){
  int txt_length=strlen(text)+1;

  write(sock_fd, &txt_length, sizeof(txt_length));
  write(sock_fd, text, txt_length);
}
```

```
~> ./lab14S-s   tt
```

```
~> ls -ltr
итого 32
-rw-r--r-- 1 malkov users  1292 Dec 12  2022 lab14S-s.c
-rw-r--r-- 1 malkov users   819 Dec 12  2022 lab14S-c.c
-rwxr-xr-x 1 malkov users 11840 Dec 12  2022 lab14S-c
-rwxr-xr-x 1 malkov users 12240 Dec 12  2022 lab14S-s
srwxr-xr-x 1 malkov users     0 Dec  1 17:48 tt
```

```
~> ./lab14S-c tt  "Hi, everybody"
~> ./lab14S-c tt  "Bye, everybody!"
~> ./lab14S-c tt  quit
~>
```

```
~> ./lab14S-s tt
Hi, everybody
Bye, everybody!
quit
~>
```

```
~> ls -ltr
итого 32
-rw-r--r-- 1 malkov users  1292 Dec 12  2022 lab14S-s.c
-rw-r--r-- 1 malkov users   819 Dec 12  2022 lab14S-c.c
-rwxr-xr-x 1 malkov users 11840 Dec 12  2022 lab14S-c
-rwxr-xr-x 1 malkov users 12240 Dec 12  2022 lab14S-s
```

Интернет сокеты

lab14a.c

Самая «сырая» программа

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <linux/if_ether.h>
#include <unistd.h>

int main(){
 int sd, bytes_read;
 char data[1024];
//sd = socket(PF_INET, SOCK_PACKET, htons(ETH_P_ALL));
  sd = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
  do{
   //bytes_read = recvfrom(sd, data, sizeof(data), 0,0,0);
   bytes_read = read(sd, data, sizeof(data));
   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):

Значение	Описание
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:

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

/labs> **sudo ./lab15a**

D3 p E4H@@ M MXZ T F] Q # 5 9 cU192
D3 p E4ō@@Q MXZ Z DU r `# 5 jik"192
D3 p E4 &@@8 W F N j !Q0KL 192
D3 p E4?@@Vv] w 0 l - b a 192
9 U 77U@;
D3 p E>6@@u
*)Y w J} b M k k7 M (/ ƕ 192
" T h #7K ho ق r= &74
^D3F @D- d 192
^ ?E /@ v *_%9E5E7C8F47989526C9BCD95D2408
4F6F0B27C5ED_sub
_googlecast_tcplocal
_233637DE 7
<
192

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)		

lab14a.h

```
#define IP_SIZE          4
#define ETH_SIZE         6
typedef unsigned char uchar;
typedef unsigned int uint;
struct ip_packet {
        struct {
                uchar dst_eth[ETH_SIZE]; uchar src_eth[ETH_SIZE]; uchar __unknwn[2];
        } hw_header;              /* hardware header */
    uint header_len:4;       /* header length in words in 32bit words */
    uint version:4;          /* 4-bit version */
    uint serve_type:8;       /* how to service packet */
    uint packet_len:16;      /* total size of packet in bytes */
    uint ID:16;              /* fragment ID */
    uint frag_offset:13;     /* to help reassembly */
    uint more_frags:1;       /* flag for "more frags to follow" */
    uint dont_frag:1;        /* flag to permit fragmentation */
    uint __reserved:1;       /* always zero */
    uint time_to_live:8;     /* maximum router hop count */
    uint protocol:8;         /* ICMP, UDP, TCP */
    uint hdr_chksum:16;      /* ones-comp. checksum of header */
    uchar IPv4_src[IP_SIZE]; /* IP address of originator */
    uchar IPv4_dst[IP_SIZE]; /* IP address of destination */
    uchar options[0];        /* up to 40 bytes */
    uchar data[0];           /* message data up to 64KB */
};
```

lab14a-d.c

Структурирование пакета

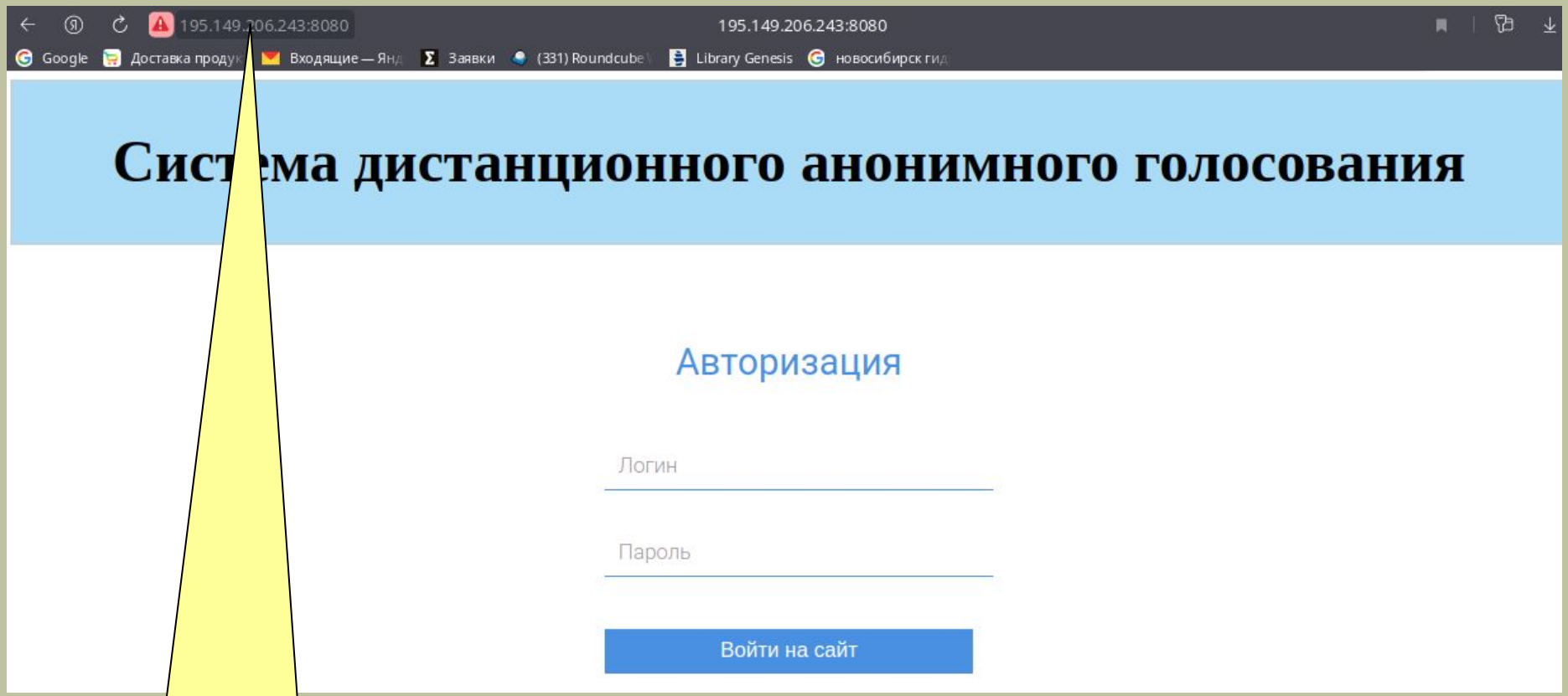
```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include “lab14a.h”
void dump(void* b, int len);

int main(){
  char buff[1024];
  int bytes_read;
  struct ip_packet *ip=(void*)buff;
  do{
   bytes_read = read(fileno(stdin), buff, sizeof(buff));
   if ( bytes_read > 0 )
      dump(buff,bytes_read);
      printf("\nIPv%d: header-len=%d, type=%d, packet-size=%d, ID=%d\n",
                                    ip->version, ip->header_len*4, ip->serve_type,
                                    ntohs(ip->packet_len), ntohs(ip->ID));
 }while ( bytes_read > 0 );
 return 0;
}
```

```
void dump(void* b, int len){
    unsigned char *buf = b;
    int i, cnt=0;
    char str[17];
    memset(str, 0, 17);
    for ( i = 0; i < len; i++ ){
        if ( cnt % 16 == 0 ){
            printf("  %s\n%04X: ", str, cnt);
            memset(str, 0, 17);
        }
        if ( buf[cnt] < ' '  ||  buf[cnt] >= 127 )
            str[cnt%16] = '.';
        else
            str[cnt%16] = buf[cnt];
        printf("%02X ", buf[cnt++]);
    }
    printf("  %*s\n\n", 16+(16-len%16)*2, str);
}
```

~/labs> **sudo ./lab14a | ./lab14a-d > mon.dat**
~/labs> vim mon.dat

```
IPv6: header-len=20, type=99, packet-size=11619, ID=26669
.....................................................................................
0250: 52 65 66 65 72 65 72 3A 20 68 74 74 70 3A 2F 2F   Referer: http://
0260: 31 39 35 2E 31 34 39 2E 32 30 36 2E 32 34 33 3A   195.149.206.243:
0270: 38 30 38 30 2F 0D 0A 41 63 63 65 70 74 2D 45 6E   8080/..Accept-En
0280: 63 6F 64 69 6E 67 3A 20 67 7A 69 70 2C 20 64 65   coding: gzip, de
0290: 66 6C 61 74 65 0D 0A 41 63 63 65 70 74 2D 4C 61   flate..Accept-La
02A0: 6E 67 75 61 67 65 3A 20 72 75 2C 65 6E 3B 71 3D   nguage: ru,en;q=
02B0: 30 2E 39 0D 0A 43 6F 6F 6B 69 65 3A 20 73 65 73   0.9..Cookie: ses
02C0: 73 69 6F 6E 3D 65 79 4A 73 62 32 64 70 62 69 49   sion=eyJsb2dpbiI
02D0: 36 4D 53 77 69 64 58 4E 6C 63 6D 35 68 62 57 55   6MSwidXNlcm5hbWU
02E0: 69 4F 69 4A 74 59 57 78 72 62 33 59 69 66 51 2E   iOiJtYWxrb3YifQ.
02F0: 59 35 37 39 42 67 2E 6C 6A 42 2D 5A 5A 4E 6C 70   Y579Bg.ljB-ZZNlp
0300: 4A 78 66 47 4A 42 66 41 49 2D 53 73 77 31 69 65   JxfGJBfAI-Ssw1ie
0310: 4E 77 0D 0A 0D 0A 6C 6F 67 69 6E 3D 6D 61 6C 6B   Nw….login=malk
0320: 6F 76 26 70 61 73 73 77 6F 72 64 3D 61 6E 79 77   ov&password=anyw
0330: 6F 72 64 26 73 65 6E 64 5F 6C 6F 67 69 6E 3D 25   ord&send_login=%
0340: 44 30 25 39 32 25 44 30 25 42 45 25 44 30 25 42   D0%92%D0%BE%D0%B
0350: 39 25 44 31 25 38 32 25 44 30 25 42 38 2B 25 44   9%D1%82%D0%B8+%D
.....................................................................................
```



http://195.149.206.243:8080/

lab14e.c

Клиент веб-сервера

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>

#define SERVERADD "195.149.206.243" //"127.0.0.1"

int main(){
 int socket_fd;
 struct sockaddr_in name;
 char buff[1024];
 char buffer[10000];
 size_t num_char;
 struct hostent* host;
```

```
struct sockaddr_in {
        sa_family_t     sin_family;     /* AF_INET */
        in_port_t       sin_port;       /* Port number */
        struct in_addr  sin_addr;       /* IPv4 address */
    };
```

```
struct in_addr {
        in_addr_t s_addr;
    };
```

```
    struct hostent {
        char   *h_name;         /* official name of host */
        char  **h_aliases;      /* alias list */
        int      h_addrtype;    /* host address type */
        int      h_length;      /* length of address */
        char  **h_addr_list;    /* list of addresses */
      }
```

```
#define h_addr     h_addr_list[0] /* for backward compatibility */
```

```
socket_fd=socket(PF_INET, SOCK_STREAM, 0);

name.sin_family=AF_INET;
host=gethostbyname(SERVERADD);
if(host==0)
  return -1;
else
  name.sin_addr=*((struct in_addr *)host->h_addr);

name.sin_port=htons(8080);

if(connect(socket_fd,(struct sockaddr*)&name,
                    sizeof(struct sockaddr_in)) ==-1){
   perror("connect");
   return -1;
}
```

```
  //sprintf(buffer,"GET /~malkov/ HTTP/1.0\r\n\r\n");
  sprintf(buffer,"GET /HTTP/1.0\r\n\r\n");
  write(socket_fd,buffer,strlen(buffer));

  while(1){
    num_char=read(socket_fd, buffer, 10000);
     if(num_char==0)
            return 1;
     fwrite(buffer,sizeof(char),num_char,stdout);
 }

 return 0;
}
```

```
/labs> ./lab14e
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 2502
Server: Werkzeug/2.0.3 Python/3.6.2
Date: Mon, 19 Dec 2022 08:47:06 GMT
<!DOCTYPE html>
<html>
  <head>
   <title>Система дистанционного анонимного голосования
   </title>
.....................................................................
  <h3>Авторизация</h3>
  <div class="form-row">
         <input type="text" name="login" required
autocomplete="off"><label for="email">Логин</label>
 </div>
 <div class="form-row">
   <input type="password" name="password" required
autocomplete="off"><label for="password">Пароль</label>
 </div>
```

lab14f-s.c

Сокет-сервер

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

#define BUF_SIZE 500
#define MY_PORT  1952

int main(){
  int   loc_socket_fd, cln_socket_fd;
  struct sockaddr_in   local_addr, client_addr;
  int client_addr_size=sizeof(client_addr);
```

```
loc_socket_fd=socket(AF_INET, SOCK_STREAM, 0);

 local_addr.sin_family=AF_INET;
 local_addr.sin_port=htons(MY_PORT);
 local_addr.sin_addr.s_addr=0;

 bind(loc_socket_fd, (struct sockaddr*)&local_addr,
      sizeof(local_addr) );

 listen(loc_socket_fd, 0x100);
```

```
  while((cln_socket_fd=
    accept(loc_socket_fd, (struct sockaddr *) &client_addr, &client_addr_size))
   )
{
    struct hostent *hst;
    int bytes_read;
    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(cln_socket_fd, "Hello a new client!\n",
      sizeof("Hello a new client!\n"),0);
  }

  return 0;
}
```

```
/labs/srvcln> ./lab14f-s
localhost [127.0.0.1] new connect!
```

```
/labs> telnet
telnet> open 127.0.0.1 1952
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Hello a new client!
Connection closed by foreign host.
```

lab15f-c.c

Сокет-клиент

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>

#define SERVERADD "127.0.0.1"

int main(){
 int socket_fd;
 struct sockaddr_in name;
 char buff[1024];
 char buffer[10000];
 size_t num_char;
 struct hostent* host;
```

```
socket_fd=socket(PF_INET, SOCK_STREAM, 0);

name.sin_family=AF_INET;
host=gethostbyname(SERVERADD);
if(host==0)
  return -1;
else
  name.sin_addr=*((struct in_addr *)host->h_addr);

name.sin_port=htons(1952);

if(connect(socket_fd,(struct sockaddr*)&name,sizeof(struct
                                                    sockaddr_in))==-1){
   perror("connect");
   return -1;
}
```

```
while(1){
    num_char=read(socket_fd, buffer, 10000);
    if(num_char==0){
    return 1;
   }
   fwrite(buffer,sizeof(char),num_char,stdout);
 }
 return 0;
}
```

```
labs/srvcln> ./lab14f-s
localhost [127.0.0.1] new connect!
```

```
labs/srvcln> ./lab14f-c
Hello a new client!
```