# Лекция 07.

Сетевое программирование

# Сокет

- «Портал» для пользовательского приложения в сеть
- Приложение пишет или читает данные в/из сокета
- Установка соединения, буферизация, отправка, получение данных находится «под капотом» сокета
- В приложении идентифицируется дескриптором – число; все системные вызовы принимают на вход данный дескриптор
- Кроссплатформенность

# Socket api

**Таблица 6.1**. Примитивы простой транспортной службы

| Примитив | Посланный модуль данных транспортного протокола | Значение |
|---|---|---|
| LISTEN (ОЖИДАТЬ) | (нет) | Блокировать сервер, пока какой-либо процесс не попытается соединиться |
| CONNECT (СОЕДИНИТЬ) | ЗАПРОС СОЕДИНЕНИЯ | Активно пытаться установить соединение |
| SEND(ПОСЛАТЬ) | ДАННЫЕ | Послать информацию |
| RECEIVE (ПОЛУЧИТЬ) | (нет) | Блокировать сервер,пока не прибудут данные |
| DISCONNECT (РАЗЪЕДИНИТЬ) | ЗАПРОС РАЗЪЕДИНЕНИЯ | Прервать соединение |

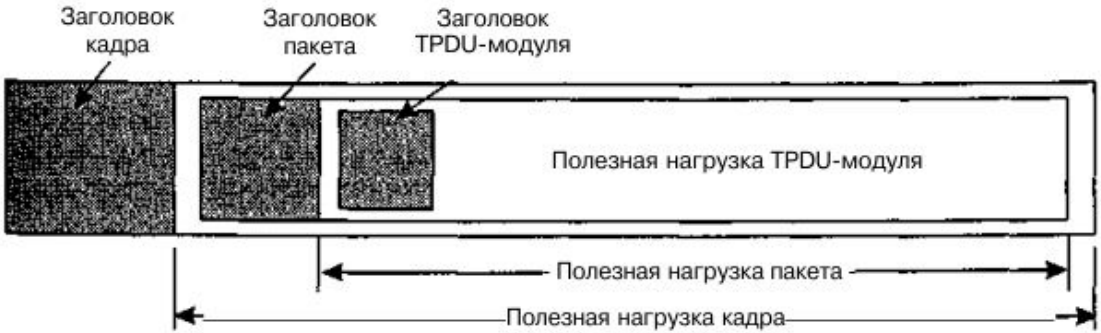



**Рис.** 6.2. Вложенность модулей данных транспортного протокола, пакетов и кадров

# Socket API

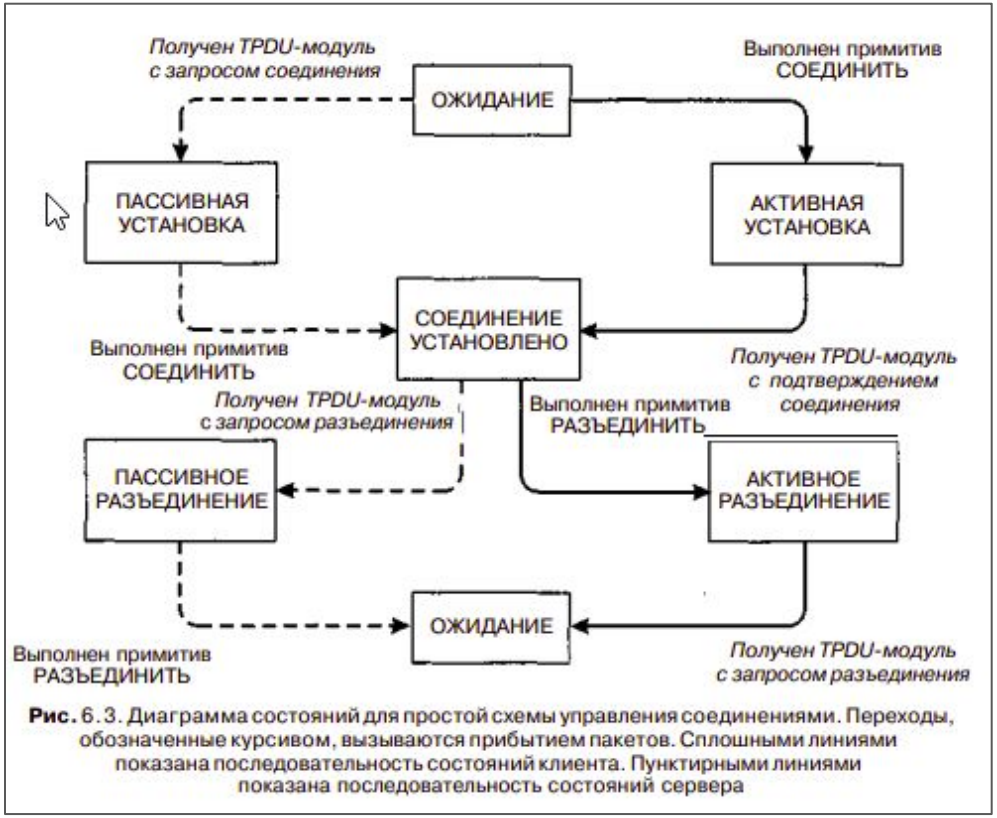
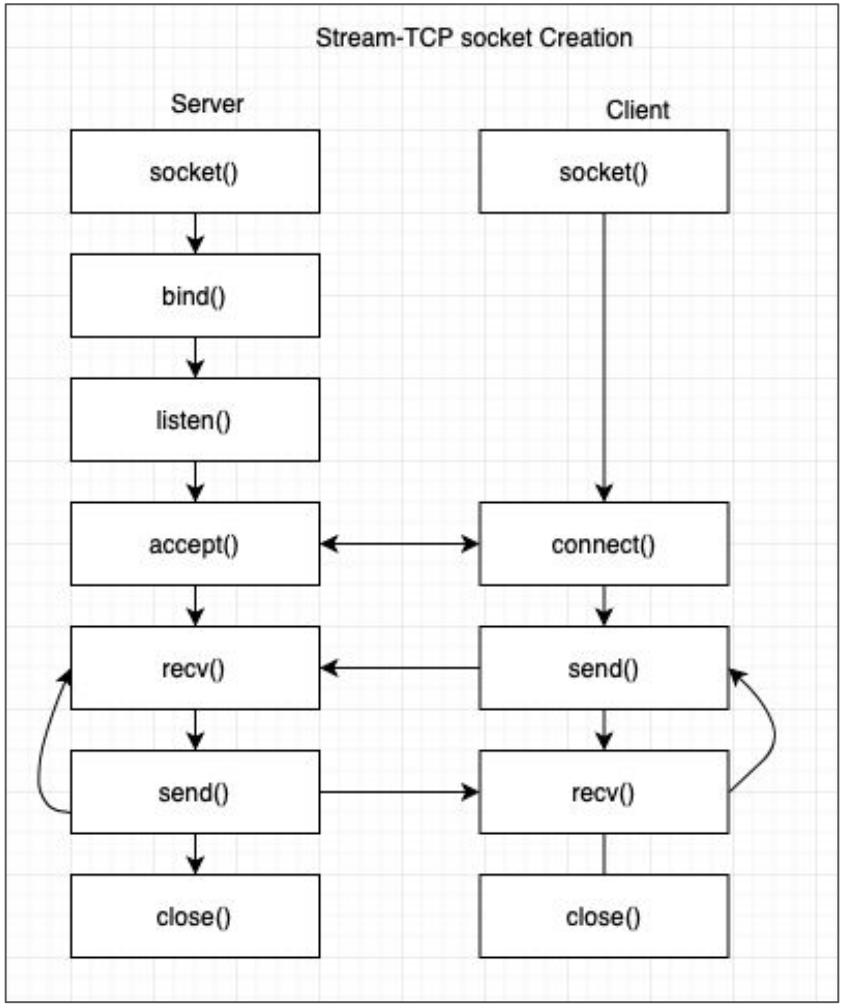



**Рис.** 6.3. Диаграмма состояний для простой схемы управления соединениями. Переходы, обозначенные курсивом, вызываются прибытием пакетов. Сплошными линиями показана последовательность состояний клиента. Пунктирными линиями показана последовательность состояний сервера



# Создание сокета

- int socket(int domain, int type, int protocol);
- Domain – пространство адресов; AF_INET и AF_INET6 – для сетевого взаимодействия
- Type – способ передачи данных по сети
  - SOCK_DGRAM – udp
  - SOCK_STREAM – tcp
  - SOCK_RAW – возможность управления сообщениями протоколов более низких уровней (IP, ICMP, ARP и т.д.)
- Protocol – используемый протокол (0 для sock_dgram, sock_stream)

# Установка соединения (сервер)

- int bind(int sockfd, struct sockaddr *addr, int addrlen)
  - именование сокета, привязка сокета к адресу (адрес интерфейса на локальном компьютере либо групповой адрес)
- int listen(int sockfd, int backlog)
  - перевод в режим ожидания (backlog – длина очереди)
  - для udp не требуется
- int accept(int sockfd, void *addr, int *addrlen)
  - ожидает подключения клиента и возвращает дескриптор сокета для общения с ним
  - для udp не требуется

# Установка соединения (клиент)

- Для udp никаких действий не требуется (кроме создания сокета)
- Можно вызвать bind для привязки к локальному адресу (отправитель)
- int connect(int sockfd, struct sockaddr *serv_addr, int addrlen)
  - установка tcp соединения

# sys/socket.h

```
27 struct sockaddr {
28   sa_family_t        sa_family;  /* address family, AF_xxx   */
29   char               sa_data[14];    /* 14 bytes of protocol address */
30 };
```

```
57 struct msghdr
58 {
59   void *        msg_name;   /* Socket name          */
60   socklen_t     msg_namelen;    /* Length of name       */
61   struct iovec *    msg_iov;    /* Data blocks          */
62   int           msg_iovlen; /* Number of blocks     */
63   void *        msg_control;    /* Ancillary data       */
64   socklen_t     msg_controllen; /* Ancillary data buffer length */
65   int           msg_flags;  /* Received flags on recvmsg    */
66 };
```

```
24 typedef __sa_family_t sa_family_t;
   typedef __socklen_t socklen_t;
```

```
#define __machine_sa_family_t_defined
typedef __uint16_t __sa_family_t;
```

# man sys_socket.h

```
The following shall be declared as functions and may also be defined as macros.

    int     accept(int, struct sockaddr *restrict, socklen_t *restrict);
    int     bind(int, const struct sockaddr *, socklen_t);
    int     connect(int, const struct sockaddr *, socklen_t);
    int     getpeername(int, struct sockaddr *restrict, socklen_t *restrict);
    int     getsockname(int, struct sockaddr *restrict, socklen_t *restrict);
    int     getsockopt(int, int, int, void *restrict, socklen_t *restrict);
    int     listen(int, int);
    ssize_t recv(int, void *, size_t, int);
    ssize_t recvfrom(int, void *restrict, size_t, int,
            struct sockaddr *restrict, socklen_t *restrict);
    ssize_t recvmsg(int, struct msghdr *, int);
    ssize_t send(int, const void *, size_t, int);
    ssize_t sendmsg(int, const struct msghdr *, int);
    ssize_t sendto(int, const void *, size_t, int, const struct sockaddr *,
            socklen_t);
    int     setsockopt(int, int, int, const void *, socklen_t);
    int     shutdown(int, int);
    int     sockatmark(int);
    int     socket(int, int, int);
    int     socketpair(int, int, int, int [2]);
```

# Socket options (socket.h)

The <sys/socket.h> header shall define the following symbolic constants with distinct values for use as the option_name argument in

getsockopt() or setsockopt() calls (see the System Interfaces volume of POSIX.1-2008, Section 2.10.16, Use of Options):

```
SO_ACCEPTCONN Socket is accepting connections.
SO_BROADCAST  Transmission of broadcast messages is supported.
SO_DEBUG      Debugging information is being recorded.
SO_DONTROUTE  Bypass normal routing.
SO_ERROR      Socket error status.
SO_KEEPALIVE  Connections are kept alive with periodic messages.
SO_LINGER     Socket lingers on close.
SO_OOBINLINE  Out-of-band data is transmitted in line.
SO_RCVBUF     Receive buffer size.
SO_RCVLOWAT   Receive ``low water mark''.
SO_RCVTIMEO   Receive timeout.
SO_REUSEADDR  Reuse of local addresses is supported.
SO_SNDBUF     Send buffer size.
SO_SNDLOWAT   Send ``low water mark''.
SO_SNDTIMEO   Send timeout.
SO_TYPE       Socket type.
```

# Прием/передача данных

- С установкой соединения
  - int send(int sockfd, const void *msg, int len, int flags);
  - int recv(int sockfd, void *buf, int len, int flags);
- Без установки соединения
  - int sendto(int sockfd, const void *msg, int len, unsigned int flags, const struct sockaddr *to, int tolen);
  - int recvfrom(int sockfd, void *buf, int len, unsigned int flags, struct sockaddr *from, int *fromlen);

# Закрытие сокета

- int close(int fd)
  - Разрывает соединение
  - Освобождает ресурсы (дескриптора сокета)
- int shutdown(int sockfd, int how);
  - How = 0 - запретить чтение из сокета
  - How = 1 - запретить запись в сокет
  - How = 2 - запретить и то и другое
  - Close все равно надо вызвать
- После закрытия сокета с одной стороны, вторая сторона получает 0 в качестве результата recv

# client.c

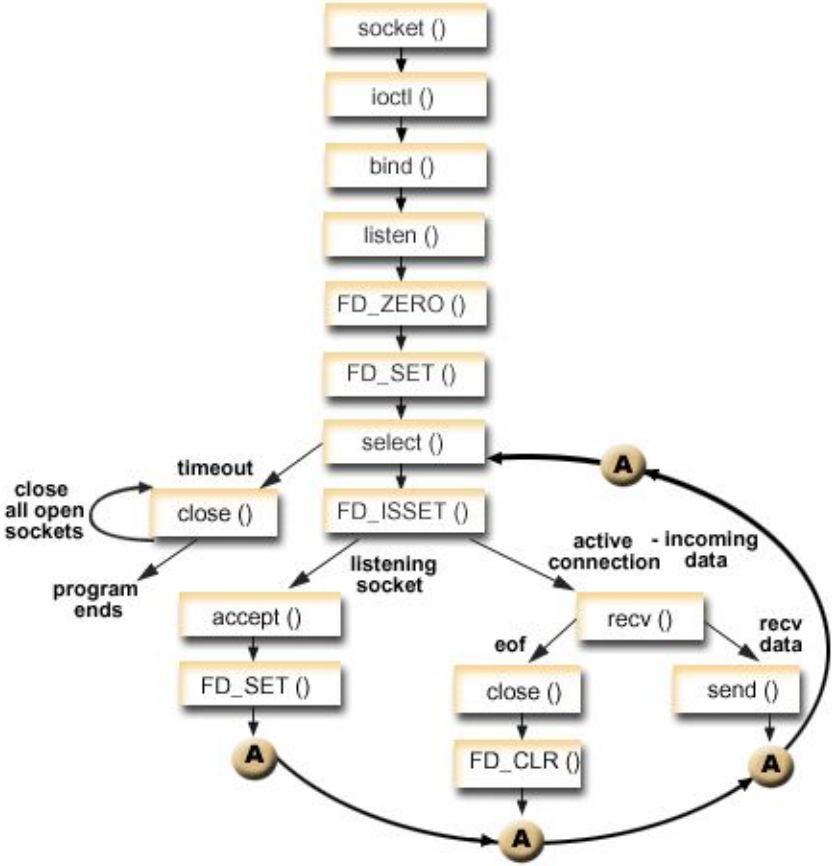
```c
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define UDP_PORT 4444
#define UDP_ADDRESS "127.0.0.1"

int main()
{
    struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(UDP_PORT);
    addr.sin_addr.s_addr = inet_addr(UDP_ADDRESS);

    int udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
    char msg[] = "Hello world\n";

    sendto(udp_socket, msg, strlen(msg) + 1, 0, (struct sockaddr*)&addr, sizeof(addr));
    close(udp_socket);

    return 0;
}
```

# server.c

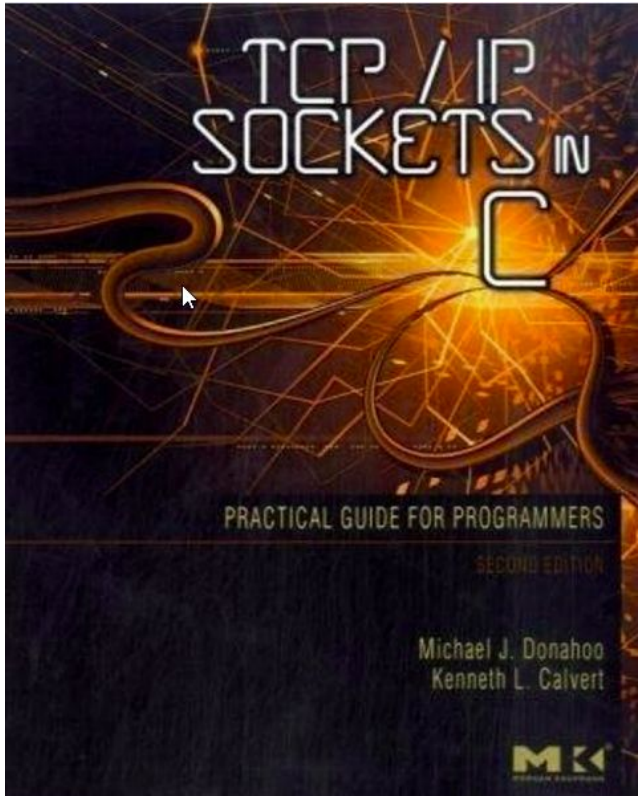
```c
#define SERVER_PORT 4444
#define SERVER_ADDRESS "127.0.0.1"
#define MESSAGE_SIZE 255

int main()
{
    struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(SERVER_PORT);
    addr.sin_addr.s_addr = inet_addr(SERVER_ADDRESS);

    int udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
    bind(udp_socket, (struct sockaddr*)&addr, sizeof(addr));

    char msg[MESSAGE_SIZE];

    recvfrom(udp_socket, msg, MESSAGE_SIZE, 0, NULL, NULL);

    printf("Message: %s\n", msg);

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

# Fork, select, poll

- … -> accept -> **fork()** -> … -> close()



# Books


TCP/IP
SOCKETS IN
C
PRACTICAL GUIDE FOR PROGRAMMERS
SECOND EDITION
Michael J. Donahoo
Kenneth L. Calvert
MK