Операционные системы

Взаимодействие по сети. Сокеты

Олег Французов

2017

Протокол обмена

- Набор соглашений, которым должны следовать участники обмена информацией, чтобы понять друг друга
- Не обязательно речь о компьютерах

Модель ISO/OSI

- Прикладной — Application
- Представительный — Presentation
- Сеансовый — Session
- Транспортный — Transport
- Сетевой — Network
- Канальный — Data Link
- Физический — Physical

Мнемоника

- All
- People
- Seem
- To
- Need
- Data
- Processing

Application

Presentation

Session

Transport

Network

Data Link

Physical

Сокет

- Объект ядра, через который происходит сетевое взаимодействие

- Unix: Сокет ~ fd

Адреса

- TCP/IP: IP address + port
  - IPv4: 192.168.3.12
  - IPv6: 12ff:2001:0055:2eab:0767:1212:f1b1:a00a
- AF_UNIX: имя файла

Параметры сокета

- Семейства адресации
  AF_INET, AF_UNIX

- Типы взаимодействия
  дейтаграммный, канальный

Создание сокета

```
int socket(int addr_family,
           int type, int protocol);
```

-> fd или -1

addr_family: AF_INET, AF_UNIX
type: SOCK_STREAM, SOCK_DGRAM

Связывание сокета с адресом

```
int bind(int sockfd,
         struct sockaddr *addr,
         int addrlen);

addr: sockaddr_in
      - sin_family = AF_INET
      - sin_port = htons(<port>)
      - sin_addr.s_addr = <ip>
```

Связывание сокета с адресом

```
int bind(int sockfd,
         struct sockaddr *addr,
         int addrlen);

addr: sockaddr_un
      - sun_family = AF_UNIX
      - sun_path : string[108]
```

Связывание сокета с адресом

```
int bind(int sockfd,
         struct sockaddr *addr,
         int addrlen);

-> 0 или -1 в случае ошибки
           - порт уже занят
           - нет прав использовать №
```

Прием и передача дейтаграмм

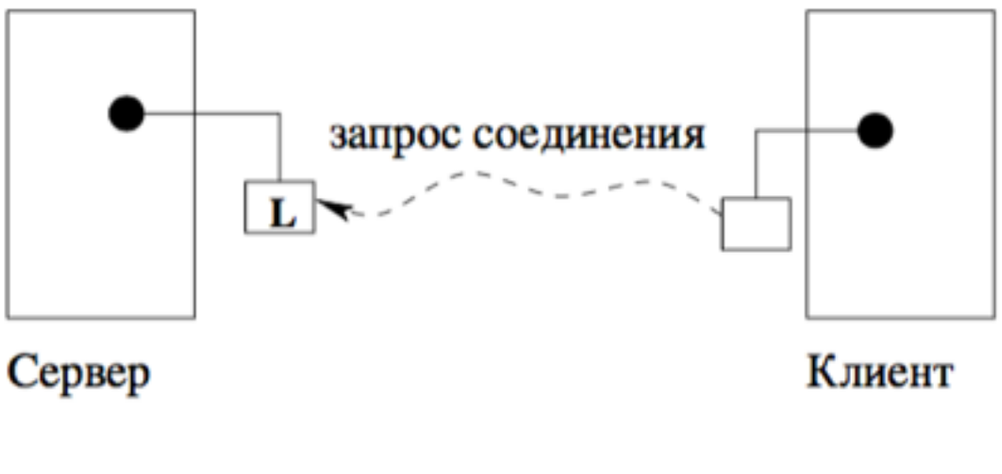
```
socket()
bind()
sendto() / recvfrom()
```

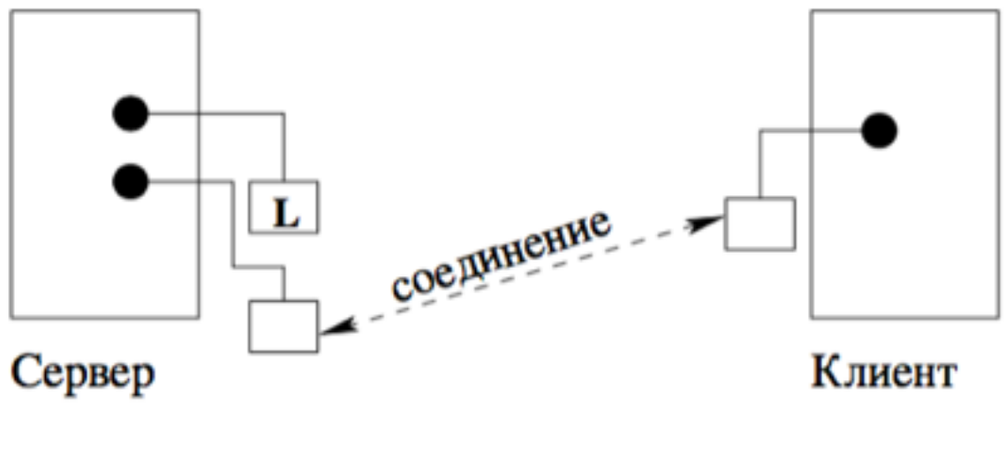
Прием и передача дейтаграмм

```
int sendto(int s, const void *buf,
           int len, int flags,
           const struct sockaddr *to,
           socklen_t tolen);

int recvfrom(int s, void *buf,
             int len, int flags,
             struct sockaddr *from,
             socklen_t *fromlen);
```

Потоковые сокеты





- Клиент: обращается с запросом
- Сервер: ожидает запросы, выполняет действия только в ответ на запрос

Организация сервера

```
socket()
bind()

int listen(int sd, int qlen);  // q=5

int accept(int sd,
           struct sockaddr *addr,
           socklen_t *addrlen);

-> fd нового сокета
```

Организация клиента

```
socket()

int connect(int sd,
            const struct sockaddr *addr,
            socklen_t addrlen);

-> 0 или -1
```

Обмен данными

read()
write()

Если соединение закрыто с той стороны,
read() / recv() вернет 0.

close()
shutdown()

Адреса в AF_INET

Big Endian / Little Endian

```
htonl() / ntohl() - long int
htons() / ntohs() - short int

int inet_aton(const char *cp,
              struct in_addr *inp);
/* 127.0.0.1 -> in_addr */
```

Допустим, нужный нам IP-адрес содержится в строке `char *serv_ip`, а порт — в переменной `port` в виде целого числа. Тогда заполнение структуры `sockaddr_in` может выглядеть так:

```
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = htons(port);
if(!inet_aton(serv_ip, &(addr.sin_addr))) {
   /* Ошибка - невалидный IP-адрес */
}
```

При создании слушающего сокета на сервере задавать конкретный ip-адрес не обязательно, гораздо проще проинструктировать систему принимать соединения на заданный порт на любом из имеющихся в системе ip-адресов. При этом поле `sin_addr` следует заполнить специальным значением `INADDR_ANY`:

```
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = htons(port);
addr.sin_addr.s_addr = INADDR_ANY;
```

Залипание TCP-порта

```
int setsockopt(int sd, int level,
    int optname,
    const void *optval, int optlen);


int opt = 1;
set sockopt(ls, SOL_SOCKET,
    SO_REUSEADDR, *opt, sizeof(opt));
```

Q & A