Client & Server
TCP server & client
---
socket - end point of communication
- ip address port no.

APIs:- socket, bind,
listen, accept connect
read, write
close, shutdown

Step1:- create a socket (empty)
```
int ssd;
ssd = socket(AF_INET, SOCK_STREAM, 0);
if(ssd<0) perror("socket");
```

IP Address families:-
- IPv4
- IPv6
- unix local

Socket Types:-
- Stream sockets
- Datagram sockets
- Raw sockets
- Local sockets

man 7 ip

sockaddr_in
sockaddr_in6
sockaddr_un

sockaddr

Step2:- (bind)

INADDR_LOOPBACK

```
int sport = 1500;
```

inet_addr("0.0.0.0");
INADDR_ANY

```
struct sockaddr_in  saddr;
//bzero(&saddr, sizeof(addr));
saddr.sin_family = AF_INET;
saddr.sin_port = htons(sport);
addr.sin_addr = inet_addr("127.0.0.1"); //TODO: fix
bzero(&saddr.sin_zero, sizeof(saddr.sin_zero));
```

inet_addr("192.168.0.1");

little endian
big endian

```
ret=bind(ssd, (struct sockaddr*) &saddr,
                        sizeof(struct sockaddr_in));
if(ret<0) {
        perror("bind");
}
```

inet_addr:- convert ip addr from a.b.c.d string format
to 32 bit unsigned int, network order

Step3:-

```
        ret = listen(ssd, backlog);             //CLOSED to LISTEN state
        if(ret<0) perror("listen");             //backlog val is 5
```

Step4:-

```
        struct sockaddr_in caddr;
        socklen_t len=sizeof(struct sockaddr_in);
        csd = accept(ssd,  (struct sockaddr*)&caddr, &len);
//block for client conn
        if(csd<0) {
           perror("accept");
        }
        //print ntohs(caddr.sin_port), inet_ntoa(addr.sin_addr)
```

Step5:-

```
        Using csd do read, write operations -- send, recv APIs
```

Step6:-

```
        close(csd);
        close(ssd);
```

---

man ip

man inet_addr

```
struct in_addr {
  uint32_t    s_addr;
}
```

Fix the issue in

```
        addr.sin_addr = inet_addr("127.0.0.1");
```

Retun type of inet_addr ==> in_addr_t

saddr.sin_addr.s_addr = inet_addr("127.0.0.1");

telnet 127.0.0.1 1800
nc 127.0.0.1 1800

Client steps:-

Step1:-

int csd = socket(AF_INET, SOCK_STREAM, 0);

Step2:- (bind to any free port)

struct sockaddr_in caddr;
caddr.sin_family = AF_INET;
caddr.sin_port = htons(0); //choose any free port, dynamic port
caddr.sin_addr.s_addr = INADDR_ANY;
//bzero

ret = bind(csd, (struct sockaddr*)&caddr, sizeof(struct sockaddr_in));

Step 3:-

```
struct sockaddr_in saddr;
saddr.sin_family = AF_INET;
saddr.sin_port = htons(servPort);
saddr.sin_addr.s_addr = inet_addr(servAddress);
//bzero
ret = connect(csd, (struct sockaddr*)&saddr, sizeof(struct sockaddr_in));
if(ret < 0) {
   perror("connect");
}

//do read, write ops using csd ....APIs - send, recv

close(csd);
```

UDP sender & receiver:-

UDP receiver hints:-

Step1:-

```
rsd = socket(AF_INET, SOCK_DGRAM, 0);
```

Step2:-

```
struct sockaddr_in raddr;
raddr.sin_family = AF_INET;
raddr.sin_port = htons(rport);
raddr.sin_addr.s_addr = INADDR_ANY;
```

//no listen, no accept

Step3:-

```
struct sockaddr_in caddr;                                   //sender details
socklen_t len = sizeof(struct sockaddr_in)
nbytes = recvfrom(rsd, buf, maxlen, 0,
                                    &caddr, &len);
//print buf              // write(rsd, buf maxlen);
```

Step4:-

```
close(rsd);
```

sudo apt install wireshark

UDP sender:-

Step1:-

```
csd = socket(AF_INET, SOCK_DGRAM, 0);
```

Step2:-

```
struct sockaddr_in myaddr;
myaddr.sin_family = AF_INET;
myaddr.sin_port = htons(0);                    //any free port, dynamic
myaddr.sin_addr.s_addr = INADDR_ANY;
```

Step3:- //no connect

```
struct sockaddr_in raddr;
raddr.sin_family = AF_INET;
raddr.sin_port = htons(rport);
raddr.sin_addr.s_addr = inet_addr(raddr);

char msg[]="Hello UDP";
int len;
nbyetes = sendto(csd, msg, len, 0,
                    (struct sockaddr*)&caddr, sizeof(caddr);
```

Step4:-

```
close(csd);
```