Functions to create and establish a socket connection

1. Socket creation: ``socket()``

Function:

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

- **Purpose:** creates an endpoint for communication.
- **Arguments:**
  - domain: specifies the communication domain.
    - AF_INET: internet protocol (IPv4).
  - type: specifies the communication type.
    - SOCK_STREAM: a TCP (reliable, connection_oriented) socket.
  - protocol: specifies a particular protocol.
    - IPROTO_TCP: TCP protocol. 0 used to let the system choose.
- Returns:
  - File descriptor for the socket on success.
  - -1 on failure.

2. Address Initialization: struct sockaddr_in

- Used to define the properties of a socket’s address.
- **Fields:**
  - sin_family: address family (AF_INET for IPv4).
  - sin_addr.s_addr: IP address (set to INADDR_ANY for server or use inet_pton() for a specific address).
  - Sin_port(): port number (in network byte order using htons()).

3. IP Conversion: inet_pton()

Function:

```
int inet_pton(int af, const char *src, void *dst);
```

- **Purpose:** converts an IP address from text to binary form.
- **Arguments:**
  - Af: address family (AF_INET for IPv4).
  - Src: string containing the IP address.
  - DST: pointer to a buffer to store the binary form of the address.
- **Returns:**
  - 1 on success
  - 0 if the input string is invalid
  - -1 on error (e.g. invalid address family)

4. Connecting to a Server: connect() (Client-side only)

Function:

```
int connect(int sockfd, const struct sockaddr *addr, sockle
n_t addrlen);
```

- **Purpose:** establishes a connection to a server.
- **Arguments:**
  - Sockfd: socket file descriptor returned by socket().
  - Addr: pointer to struct sockaddr_in defining the server's address and port.
  - Addr len: size of the addr structure (sizeof(struct sockaddr_in)).
- **Returns:**
  - 0 on success
  - -1 on failure (e.g. the server is unreachable)

5. Binding a Socket: bind() (Server-side only)

Function:

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

- **Purpose:** assigns a specific local address and port to the socket.
- **Arguments:**
  - Sockfd: socket file descriptor.
  - Addr: pointer to struct sockaddr_in specifying the local address and port.
  - Addr len: size of the addr structure.
- **Returns:**
  - 0 on success
  - -1 on failure (e.g. the port is already in use)

6. Listening for Connections: listen() (Server-side only)

Function:

```
int listen(int sockfd, int backlog);
```

- **Purpose:** marks the socket as passive, ready to accept incoming connections.
- **Arguments:**
  - Sockfd: socket file descriptor.
  - Backlog: maximum number of queued connections (e.g. 5 for small servers).
- **Returns:**
  - 0 on success
  - -1 on failure

7. Accepting a Connection: accept() (Server-side only)

Function:

```
int accept(int sockfd, struct sockaddr *addr, socklen_t *ad
drlen);
```

- **Purpose:** accepts a new connection from a client.
- **Arguments:**
  - Sockfd: listening socket file descriptor.
  - Addr: pointer to struct sockaddr_in that will hold the client's address.
  - Addr len: pointer to the size of the addr structure.
- **Returns:**
  - New file descriptor for the accepted connection on success
  - -1 on failure

8. Sending Data: send()

Function:

```
ssize_t send(int sockfd, const void *buf, size_t len, int f
lags);
```

- **Purpose:** sends data to a connected socket.
- **Arguments:**
  - Sockfd: connected socket file descriptor.
  - Buf: pointer to the data buffer to send.
  - Len: size of the data buffer.
  - Flags: set to 0 for default behavior.
- **Returns:**
  - Number of bytes sent on success

- -1 on failure

9. Receiving Data: recv()

Function:

```
ssize_t recv(int sockfd, void *buf, size_t len, int flags);
```

- **Purpose:** receives data from a connected socket.
- **Arguments:**
  - Sockfd: connected socket file descriptor.
  - Buf: pointer to the buffer to store received data.
  - Len: size of the buffer.
  - Flags: set to 0 for default behavior.
- **Returns:**
  - Number of bytes received on success
  - -1 on failure

10. Closing a Socket: close()

Function:

```
int close(int fd);
```

- **Purpose:** closes a socket and releases its resources.
- **Arguments:**
  - Fd: socket file descriptor to close.
- **Returns:**
  - 0 on success
  - -1 on failure

Summary

Client Side:

1. Socket(): create a socket.
2. Inet_pton(): converts server IP address to binary form.
3. Connect(): connect to the server.
4. Send() / recv(): exchange data.
5. Close(): close the socket.

Server Side:

1. Socket(): create a socket.
2. Bind(): bind the socket to a local address.
3. Listen(): mark the socket as ready to accept connections.
4. Accept(): accept client connections.
5. Send() / recv(): exchange data with clients.
6. Close(): close the client and server sockets.