📝 Socket in Linux System Programming

🧠 What is a Socket?

A **socket** is an endpoint for communication between two machines (or processes) over a **network** (TCP/IP).
Used in **client-server** communication: web servers, chat apps, file transfers, etc.

📌 Socket Types

Socket Type	Description	Protocol
SOCK_STREAM	Reliable, connection-based	TCP
SOCK_DGRAM	Connectionless, fast but unreliable	UDP
SOCK_RAW	Raw socket, used for custom protocols	IP layer

🔧 Key System Calls

Function	Purpose
socket()	Create a socket
bind()	Bind socket to IP and port
listen()	Wait for connections (server side)
accept()	Accept incoming connection
connect()	Connect to server (client side)
send() / write()	Send data
recv() / read()	Receive data

Function	Purpose
close()	Close the socket

👷 Example: TCP Server and Client in C

🔷 TCP Server Example

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

int main() {
    int sockfd, new_sock;
    struct sockaddr_in server, client;
    char buffer[1024] = {0};
    socklen_t addr_size;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    server.sin_family = AF_INET;
    server.sin_port = htons(8080);
```

```
    server.sin_addr.s_addr = INADDR_ANY;

    bind(sockfd, (struct sockaddr*)&server, sizeof(server));
    listen(sockfd, 5);

    printf("Waiting for connection...\n");
    addr_size = sizeof(client);
    new_sock = accept(sockfd, (struct sockaddr*)&client, &addr_size);

    read(new_sock, buffer, 1024);
    printf("Client says: %s\n", buffer);

    send(new_sock, "Hello Client", strlen("Hello Client"), 0);

    close(new_sock);
    close(sockfd);
    return 0;
}
```

🔶 **TCP Client Example**

```
#include <stdio.h>
#include <string.h>
```

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>

int main() {
    int sockfd;
    struct sockaddr_in server;
    char buffer[1024] = "Hello Server";

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    server.sin_family = AF_INET;
    server.sin_port = htons(8080);
    server.sin_addr.s_addr = INADDR_ANY;

    connect(sockfd, (struct sockaddr*)&server, sizeof(server));

    send(sockfd, buffer, strlen(buffer), 0);
    read(sockfd, buffer, 1024);
    printf("Server replied: %s\n", buffer);
    close(sockfd);
    return 0;
}
```

Summary

Term	Meaning
socket()	Create socket
bind()	Assign IP/port to socket
listen()	Prepare to accept connections
accept()	Accept incoming connection
connect()	Client connects to server
send()/recv()	Data transfer

TCP Server Code Breakdown

```
#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>
```

✅ These are **header files** for:

- Input/output (stdio.h)
- Strings (string.h)
- Socket system calls (sys/socket.h)
- Address structures (netinet/in.h)
- Closing socket (unistd.h)

🔹 **Step 1: Create a socket**

```
int sockfd;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

- AF_INET → IPv4
- SOCK_STREAM → TCP protocol
- Returns a **socket file descriptor (sockfd)**

🔹 **Step 2: Define server address**

```
struct sockaddr_in server;
server.sin_family = AF_INET;
server.sin_port = htons(8080);
server.sin_addr.s_addr = INADDR_ANY;
```

- Set address family to IPv4
- Port is **8080** (converted to network byte order using htons())
- INADDR_ANY allows the server to accept connections from any IP

- **Step 3: Bind socket to address**

bind(sockfd, (struct sockaddr*)&server, sizeof(server));

- Binds the socket to the specified IP address and port
- Now the server will listen on port 8080

- **Step 4: Listen for connections**

listen(sockfd, 5);

- Puts the socket into listening mode
- 5 is the max number of pending connections

- **Step 5: Accept connection**

int new_sock;

socklen_t addr_size = sizeof(client);

new_sock = accept(sockfd, (struct sockaddr*)&client, &addr_size);

- Blocks until a client connect
- Creates a **new socket (new_sock)** for the connection

🔹 **Step 6: Read and write data**

```
read(new_sock, buffer, 1024);
printf("Client says: %s\n", buffer);

send(new_sock, "Hello Client", strlen("Hello Client"), 0);
```

- Reads data sent by client
- Sends reply "Hello Client" to client

🔹 **Step 7: Close sockets**

```
close(new_sock);
close(sockfd);
```

- Gracefully closes the client socket and server socket

🟧 **TCP Client Code Breakdown**

🔹 **Step 1: Create socket and define server address**

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
server.sin_family = AF_INET;
```

server.sin_port = htons(8080);

server.sin_addr.s_addr = INADDR_ANY;

- Client creates a socket and prepares server address

🔹 **Step 2: Connect to the server**

connect(sockfd, (struct sockaddr*)&server, sizeof(server));

- Attempts to connect to the server's IP and port

🔹 **Step 3: Send and receive data**

send(sockfd, buffer, strlen(buffer), 0);

read(sockfd, buffer, 1024);

- Sends a message to the server
- Receives response from server

🔹 **Step 4: Close connection**

close(sockfd);

- Terminates the connection

✅ **Output When Run:**

Server:

Waiting for connection...

Client says: Hello Server

Client:

Server replied: Hello Client