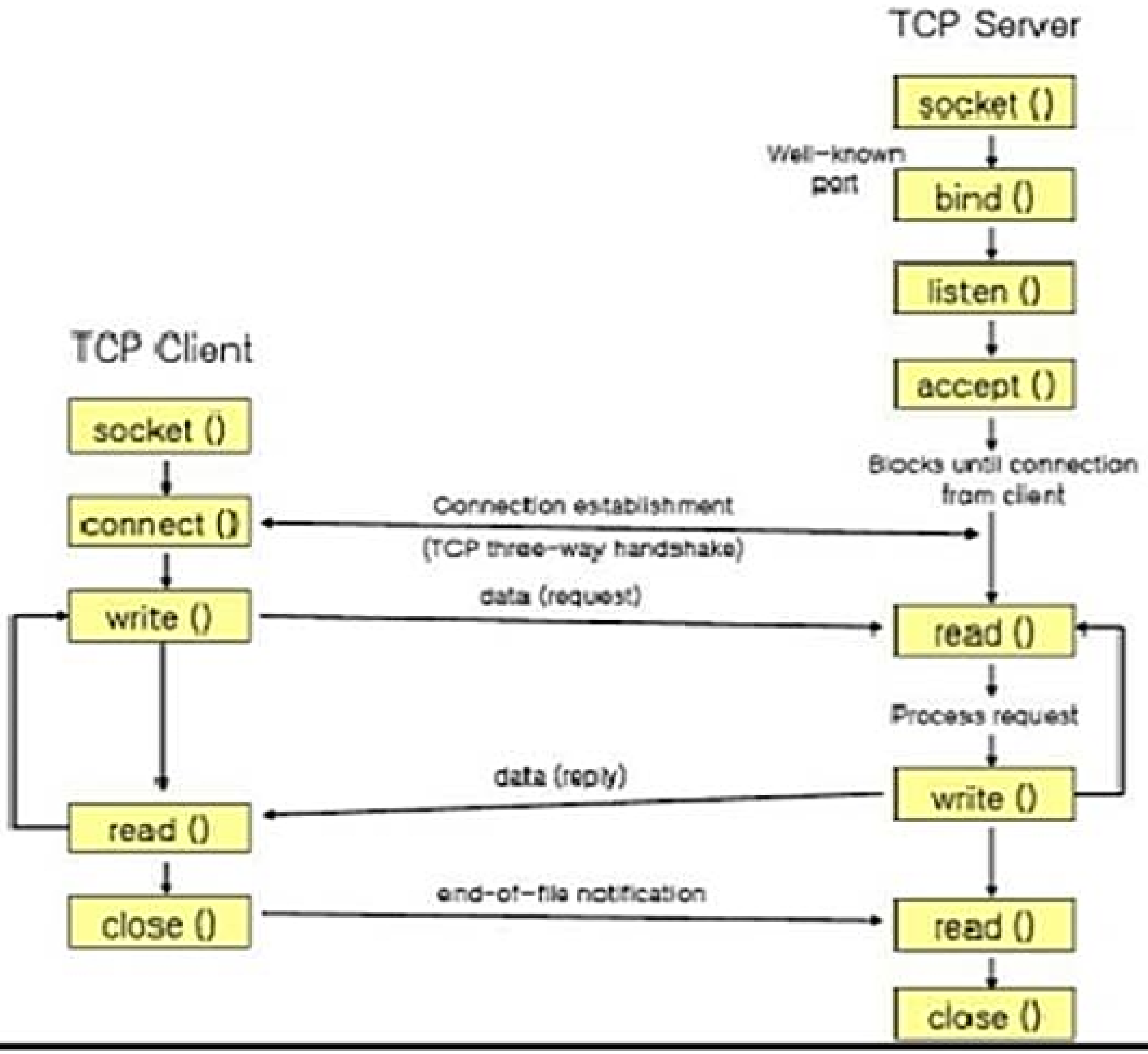
# Elementary TCP socket functions

Prof. Anand D Dave
Department of Information Technology,
Dharmsinh Desai University, Nadiad.

# TCP Client-Server Program

- List of functions
  - socket function
  - connect function
  - bind function
  - listen function
  - accept function
  - close function
  - getsockname and getpeername function

# Structure of TCP client-server program



## TCP Client-Server Program socket function

- To perform network I/O, the process first creates a socket.

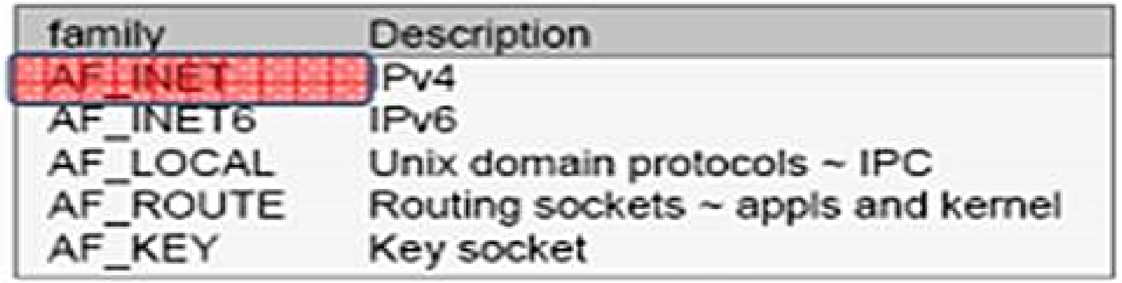
```
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

returns:nonnegative descriptor if OK, -1 on error

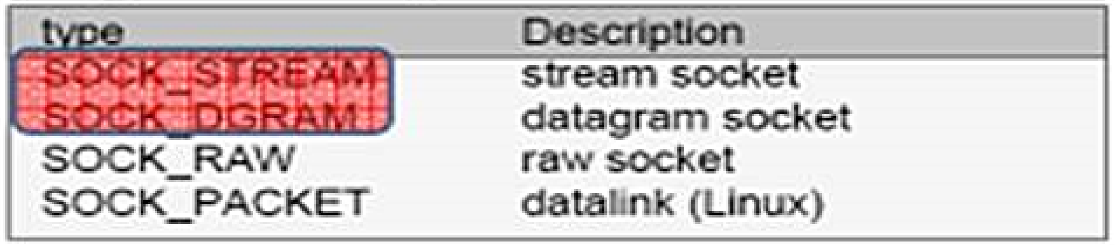
Note: Normally the protocol argument to the socket function is set to 0 except for raw socket.

# TCP Client-Server Program socket function

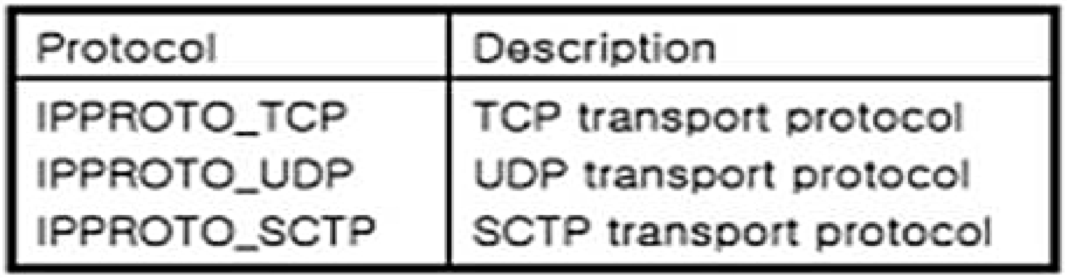
- socket function: protocol family and type



| family | Description |
|---|---|
| AF_INET | IPv4 |
| AF_INET6 | IPv6 |
| AF_LOCAL | Unix domain protocols ~ IPC |
| AF_ROUTE | Routing sockets ~ appls and kernel |
| AF_KEY | Key socket |



| type | Description |
|---|---|
| SOCK_STREAM | stream socket |
| SOCK_DGRAM | datagram socket |
| SOCK_RAW | raw socket |
| SOCK_PACKET | datalink (Linux) |



| Protocol | Description |
|---|---|
| IPPROTO_TCP | TCP transport protocol |
| IPPROTO_UDP | UDP transport protocol |
| IPPROTO_SCTP | SCTP transport protocol |

## TCP Client-Server Program socket function

- AF_xxx vs PF_xxx
  - AF_ : address family
    - used in socket address structures
  - PF_ : protocol family
    - used to create the socket
    - We use only the AF_ constants.

# TCP Client-Server Program connect function

**Connect**: This function is used by a **client** to establish a connection with a server via a **3-way handshake. (Recall TCP socket state transition diagram!** i.e., transition from CLOSED to SYN_SENT to ESTABLISHED.)

```
#include <sys/socket.h>
int connect(int sockfd, const struct sockaddr *servaddr,
socklen_t addrlen);
```

Returns:0 if OK, -1 on error

– sockfd is a socket descriptor returned by the socket() function
– servaddr contains the IP address and port number of the server
– addrlen has the length (in bytes) of the server socket address structure

## TCP Client-Server Program connect function

- The function returns only when the connection is established or an error occurs.

  Return error
  – ETIMEOUT : no response from server
  – RST : server process is not running
  – EHOSTUNREACH : client's SYN unreachable from some intermediate router.

  The client does not have to call bind

  The **kernel** chooses both an **ephemeral port** and the **source IP addres**

## TCP Client-Server Program connect function

Possible responses to the Connect

- Response 1:

If the client TCP receives no response to its SYN segment, ETIMEDOUT is returned. (server is busy, queue is full, will see later)

4.4BSD sends one SYN when connect is called, another 6 seconds later, and another 24 seconds later.

If no response is received after a total of 75 seconds, the error is returned.

## TCP Client-Server Program connect function

Response 2: (error due to transport layer)

- If the server's response to the client's SYN is a reset(RST) (i.e,to client's TCP),this indicates that no process is waiting for connections on the server host at the port specified. This is a hard error and the error ECONNREFUSED is returned to the client (i.e, application) as soon as the RST is received.
- RST is a type of TCP segment that is sent by TCP when something is wrong.
- Three conditions that generate an RST

– When a SYN arrives for a port that has no listening server.

– When TCP wants to abort an existing connection.

– When TCP receives a segment for a connection that does not exist.

# TCP Client-Server Program connect function

- Response 3: (error due to network layer)
- If the client's SYN elicits (Eng. Meaning Derive by reason) an ICMP "destination unreachable" from some intermediate router.

– This is considered a soft error.

– The client kernel saves the message but keeps sending SYNs with the same time between each SYN.

- If no response is received after some fixed amount of time, the saved ICMP error is returned to the process as either EHOSTUNREACH or ENETUNREACH

## TCP Client-Server Program bind function

The bind function assigns a local protocol address to a socket.

```
#include <sys/socket.h>

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

Returns: 0 if OK, -1 on error

- sockfd is a socket descriptor returned by the socket() function
- myaddr is a pointer to a protocol-specific address. With TCP, it has the IP address and port number of the server.
- addrlen has the length (in bytes) of the server socket address structure.

## TCP Client-Server Program bind function

- Binding of port number:

  Servers bind to their well-known port when they start.

  – If TCP client or server skips this call, the kernel chooses an ephemeral port for the socket when either connect (by client) or listen (by server) is called.
- Normally, TCP client let the kernel choose an ephemeral port. (Unless the client requires a reserved port).
- Server runs on a well known port. TCP server does not let the kernel choose an ephemeral port.

## TCP Client-Server Program bind function

**Binding of IP address:**

- A process can bind a specific IP address to its socket. This address must belong to an interface on the host.

  – For a TCP client: this assigns the source IP address that will be used for IP datagrams sent on the socket.
  – For a TCP server: this restricts the socket from receiving incoming client connection requests destined to some other IP address of the server host (applicable for multi-homed server host).
- Normally, a client does not bind an IP address to its socket.
  – The kernel chooses when connect is called based on outgoing interface that is used, which in turn is based on the route required to reach the server.

## TCP Client-Server Program listen function

- The listen function is used by a server to convert an unconnected socket to a passive socket.
- This function moves the socket from CLOSED state to the LISTEN state. (Recall TCP State transition diagram).

```
#include <sys/socket.h>
      int listen(int sockfd, int backlog);
```

Returns:0 if OK, -1 on error

– sockfd is a socket descriptor returned by the socket() function

– Backlog: Its specifies the maximum number of connections that the kernel should queue for this socket.

- This function is called only by a TCP server

# TCP Client-Server Program listen function

- For a given listening socket, the kernel maintains 2 queues

– An incomplete connection queue:

- It contains an entry for each SYN received from a client, for which the server is awaiting completion of the TCP 3-way handshake.
- If entry times out (before TCP handshake is over), it is removed from the queue.

– A completed connection queue

- It contains an entry for each client with whom the TCP 3-way handshake process has completed.
- backlog is the sum of these two queues.

– One may select any number other than 0. Different implementations interpret it differently.

## TCP Client-Server Program accept function

- Accept is called by a TCP server to return the next completed connection from the front of the completed connection queue. A new descriptor is created automatically by the kernel for the connected socket.

- If completed queue is empty, the process is put to sleep.

```
#include <sys/socket.h>
int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);
Returns: non-negative descriptor if OK, -1 on error
```

- sockfd: a socket descriptor returned by the socket() function
- cliaddr: contains the IP address and port number of the connected client addrlen: has the length (in bytes) of the returned client socket address structure (a value-result argument)
- If the completed connection queue is empty, the process is put to sleep

## TCP Client-Server Program accept function

- The new socket descriptor returned by accept() is called a connected socket, whereas the one returned by socket() is called a listening socket.

– A given server usually creates only one listening socket. It exists for the lifetime of the server.

– A connected socket is created for each client connection that is accepted. It exists only for the duration of the connection.

- Both cliaddr and addrlen may be set to the NULL pointer, if the server is not interested in knowing the identity of the client

## TCP Client-Server Program
## accept function

The new socket descriptor returned by accept() is called a connected socket, whereas the one returned by socket() is called a listening socket.

- A given server usually creates only one listening socket. It exists for the lifetime of the server.

- A connected socket is created for each client connection that is accepted. It exists only for the duration of the connection.
- Both cliaddr and addrlen may be set to the NULL pointer, if the server is not interested in knowing the identity of the client.

## TCP Client-Server Program
## UNIX close() Function

- It is used to close a socket and **terminate a TCP connection**.

  #include <unistd.h>

  int close(int sockfd);

  – sockfd is a socket descriptor returned by the socket() function

  - Returns: 0 if ok, -1 on error

# TCP Client-Server Program UNIX close() Function

- close() marks socket as closed and returns immediately.
  - sockfd is no longer usable
  - TCP continues to try sending unsent data
- close() simply decrements the reference count
  - Socket goes away when the reference count becomes 0
- In concurrent server, what if the parent does not close the connected socket for the client ?
  - May run out of descriptors eventually
  - No client connection will be terminated
  - Reference count remains at 1

## TCP Client-Server Program getsockname Function

- It returns the local protocol address associated with the socket.

```
#include<sys/socket.h>
int getsockname(int sockfd, struct sockaddr *localaddr,
    socklen_t *addrlen);
```

- sockfd: a socket descriptor returned by the socket() call
- localaddr and addrlen are value-result arguments

## TCP Client-Server Program getpeername Function

- It returns the foreign protocol address associated with the socket.

```
int getpeername(int sockfd, struct sockaddr*peeraddr,
    socklen_t *addrlen);
```

– sockfd: a socket descriptor returned by the socket() call.

– peeraddr/addrlen are value-result arguments.

## TCP Client-Server Program getsockname & getpeername Function

Reasons why these two functions :

- A TCP client that does not call bind(). How does it get the local IP address and port number assigned after connect returns successfully?
  - Ans: using getsockname().

- A TCP server that called bind() with a port number 0 uses getsockname() to get the local port number assigned.
- A TCP server that called bind() with the wildcard IP address uses getsockname() to get the local IP address assigned.
- To obtain the address family of a socket, use getsockname()
- A TCP server that called accept() uses getpeername() to get

the identity of the client (its IP address and port number)

## References

- Sources :
- https://www.javatpoint.com/osi-model
- https://www.tutorialspoint.com/software_architecture_design/distributed_architecture.htm
- https://sites.google.com/site/prajapatiharshadb/class-notes-for-students
- UNIX Network Programming by W. Richard Stevens, Prentice Hall Publication
- Distributed Computing: Concepts &Applications: by M. L. Liu Addison Wiselly