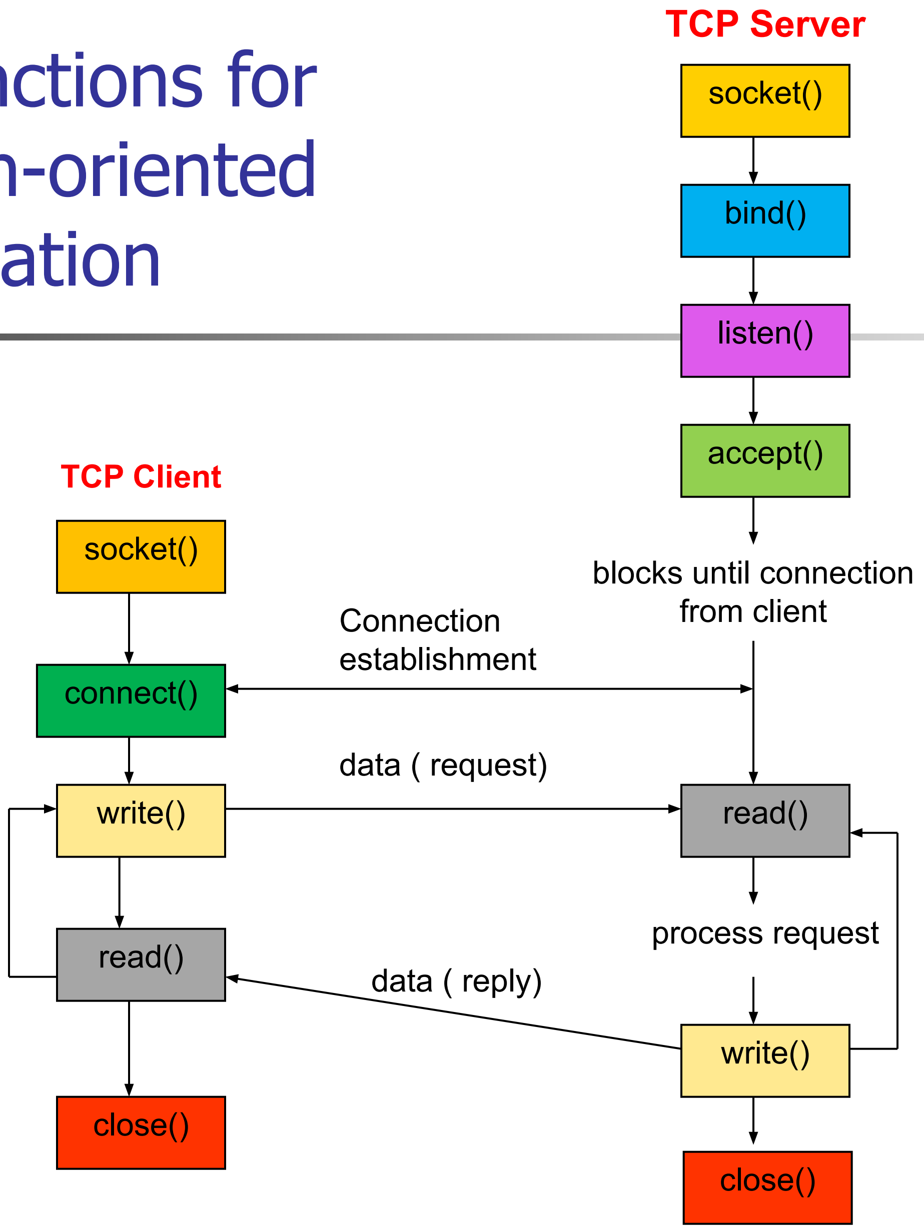
Introduction to Socket Programming

- To build any network application
  - Web browser
  - FTP
- Client – Server model

Socket functions for connection-oriented communication



Data structures

#include <netinet/in.h>

struct sockaddr
{
unsigned short *sa_family*;
// address family, AF_xxx
 	
char *sa_data*[14];
// 14 bytes of protocol address
};
	// IPv4 AF_INET sockets:
	
	struct sockaddr_in
	{
	short *sin_family*;
	// AF_INET
	unsigned short *sin_port*;
	// e.g. htons(3490)
	struct in_addr *sin_addr*;
	char *sin_zero*[8];
	// padding zero to match size of struct sockaddr
	};
		struct in_addr
		{
		unsigned long *s_addr*;
		// load with inet_pton()
		};

Choice of Port number

- Choose a port number from 1024 to 49151
  - ports 1 to 1023 are reserved for use by the Internet Assigned Numbers Authority (IANA)
  - ports 49152 through 65535 are dynamic ports that operating systems use randomly.

Byte ordering

- Host data is ordered either in Little Endian or Big Endian format - Host Byte Order
- Network data - always in Big Endian format - Network Byte Order
- Functions:
  - host to network byte order: htons(), htonl()
  - network to host byte order: ntohs(), ntohl()

Socket()

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

- family: communication domain
  - AF_INET (IPv4 protocol)
  - AF_INET6 (IPv6 protocol)
  - **Note: We'll use AF_INET**
- type: communication type
  - SOCK_STREAM: reliable, 2-way, connection-based service
  - SOCK_DGRAM: unreliable, connectionless
  - **Note: We'll use SOCK_STREAM**
- protocol: transport layer protocol - TCP/UDP
  - IPPROTO_TCP
  - IPPROTO_UDP
  - **Note: We'll set to 0, indicating default based on "type"**
- Returns socket descriptor, an integer (like a file-handle)

Bind()

Binds an IP address for the socket - used in Server

```
#include <sys/socket.h>
int bind(int sockfd, struct sockaddr *address, int addr_len)
```

- sockfd: a socket descriptor returned by the socket()
- *address: a pointer to a protocol-specific address.
- addrlen: the size of the socket address structure

- Returns on success: 0, on error: -1

Listen()

- Prepares the server to accept incoming connections - used by TCP server

  #include <sys/socket.h>

  int listen (int sockfd, int backlog);

- sockfd: a socket descriptor
- backlog: maximum number of connections that the kernel should queue for this socket

- Returns on success: 0, on error: -1

Accept()

- Accepts a connection when a client tries to connect - used by TCP server

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

- sockfd: socket descriptor
- *cliaddr: used to return the protocol address of the connected peer process
- *addrlen: length of the address

- Returns on success: a new (connected)socket descriptor, on error:-1

Connect()

- Establishes a connection with a TCP server - used by TCP client

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

- sockfd: a socket descriptor
- *servaddr: a pointer to a socket address structure
- addrlen: the size of the socket address structure

- Returns on success: 0, on error: -1

Read()

- Receives data from the specified socket

```
#include <unistd.h>
int read(int sockfd, const void * buf, int nbytes);
```

- sockfd: a socket descriptor
- buf: buffer to store the data
- nbytes: size of the buffer

- Returns: number of bytes read if OK, 0 on EOF, -1 on error

Write()

- Sends data through the specified socket

```
#include <unistd.h>
int write(int sockfd, const void * buf, int nbytes);
```

- sockfd: a socket descriptor
- buf: buffer to store the data.
- nbytes: size of the buffer

- Returns: number of bytes written if OK, 0 on EOF, -1 on error

Close()

- Closes a socket and terminates a connection
  ```
  #include <unistd.h>
  int close (int sockfd);
  ```

- sockfd: socket descriptor to be closed

- Returns on success: 0, on error: -1