**Network System Calls**

This section contains libraries and functions essential for network programming in C.

## Libraries

- **#include <stdio.h>**
  - **Purpose:** Part of the Standard Input/Output library in C, used for basic input and output operations.
  - **Common Functions:**
    - **printf()**: Used to print formatted output to the console.
    - **scanf()**: Used to read formatted input from the console.
    - **fopen(), fclose()**: Used for opening and closing files.
- **#include <stdlib.h>**
  - **Purpose:** Provides functions for performing general utility operations, like memory management and process control.
  - **Common Functions:**
    - **malloc()**, **free()**: For dynamic memory allocation and deallocation.
    - **exit()**: To terminate a program.
    - **atoi(), atof()**: To convert strings to integers or floating-point numbers.
- **#include <unistd.h>**
  - **Purpose:** Contains definitions for various constants and types, and declares functions for low-level I/O operations and system calls.
  - **Common Functions:**
    - **read(), write()**: To read from and write to file descriptors (like sockets).
    - **fork()**: To create a new process.
    - **getpid()**: To get the process ID of the current process.
- **#include <string.h>**
  - **Purpose:** Provides functions for manipulating C strings.
  - **Common Functions:**
    - **strlen()**: To find the length of a string.
    - **strcpy(), strcat()**: To copy and concatenate strings.
    - **strcmp()**: To compare two strings.
- **#include <sys/types.h>**
  - **Purpose:** Contains definitions for data types used in system calls and low-level operations.
  - **Common Types:**
    - **pid_t**: Type for process IDs.
    - **size_t**: Type for sizes of objects (like arrays).
    - **off_t**: Type for file sizes.

- **#include <sys/socket.h>**
  - **Purpose:** Contains definitions necessary for the socket programming interface.
  - **Common Functions:**
    - **socket()**: To create a new socket.
    - **bind()**: To associate a socket with a specific local IP address and port.
    - **connect()**: To connect to a remote socket.
- **#include <arpa/inet.h>**
  - **Purpose:** Provides functions for manipulating Internet addresses.
  - **Common Functions:**
    - **inet_pton()**: To convert an IPv4 or IPv6 address from text to binary form.
    - **inet_ntop()**: To convert an address from binary to text form.
- **#include <netinet/in.h>**
  - **Purpose:** Contains constants and structures needed for Internet domain addresses, especially for defining socket options and protocols.
  - **Common Functions/Structures:**
    - **sockaddr_in**: A structure for handling Internet addresses (specifically for IPv4).
    - **IPPROTO_TCP, IPPROTO_UDP**: Constants that specify TCP or UDP protocols for sockets.

## Socket Creation

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

This function creates a new socket.

- **Parameters:**
  1. **domain (e.g., AF_INET)**:
     - **Purpose:** Specifies the communication domain or address family (how addresses are represented).
     - **Pre-defined Constant:**
       - **AF_INET**: For IPv4 addresses.
       - **AF_INET6**: For IPv6 addresses.
       - **AF_UNIX**: For communication between processes on the same machine.
  2. **type (e.g., SOCK_DGRAM or SOCK_STREAM)**:
     - **Purpose:** Specifies the type of socket to create.
     - **Pre-defined Constants:**

- **SOCK_DGRAM**: For connectionless communication (UDP).
- **SOCK_STREAM**: For connection-oriented communication (TCP).

3. **protocol (usually 0)**:
   - **Purpose:** Specifies the protocol to use; usually set to 0 for default (UDP for datagrams, TCP for streams).

**sockfd = socket(AF_INET, SOCK_DGRAM, 0); // Creates a UDP socket**

**sockfd = socket(AF_INET, SOCK_STREAM, 0); // Creates a TCP socket**

## Server Information

**struct sockaddr_in servaddr, cliaddr;**

This defines the server and client socket addresses.

- **sockaddr_in**:
  - **Purpose:** Structure for IPv4 addresses, defined in <netinet/in.h>.

**Fields:**

- **sin_family**: Specifies the address family (e.g., AF_INET for IPv4).
- **sin_addr.s_addr**: Holds the IP address of the server.
- **sin_port**: Holds the port number the server listens on.

**servaddr.sin_family = AF_INET; // Specifies IPv4**

**servaddr.sin_addr.s_addr = INADDR_ANY; // Accepts connections from any IP address**

**servaddr.sin_port = htons(PORT); // Converts port number to network byte order**

## Memory Initialization

- **Purpose:** Initializes the memory of buffer to zero before use.
- **Parameters:**

- **&buffer**: Pointer to the buffer to initialize.
- **0**: Value to set the memory to (zero).
- **sizeof(buffer)**: Size of the buffer in bytes.

**memset(&buffer, 0, sizeof(buffer));**

## Binding the Socket

- **Purpose:** Associates the socket with a specific IP address and port.
- **Parameters:**

- **sockfd**: File descriptor for the socket.
- **(const struct sockaddr *)&servaddr**: Pointer to the address structure (cast to sockaddr type).
- **sizeof(servaddr)**: Size of the address structure.

**bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr));**

### Listening for Connections

 **Purpose:** Marks the socket as a passive socket that will accept incoming connection requests.
 **Parameters:**

- **sockfd**: File descriptor for the socket.
- **10**: Maximum length of the queue of pending connections.

**listen(sockfd, 10);**

### Receiving Data on Server Side

- **Purpose:** Receives data sent to the server.
- **Parameters:**
  - **len**: Size of the client address structure.
  - **buffer**: Buffer to store received data.
  - **MAXLINE**: Maximum number of bytes to receive.
  - **MSG_WAITALL**: Waits for all bytes to be received.
  - **&cliaddr**: Pointer to store the client's address.
- **buffer[n] = '\0';**: Null-terminates the received string for proper string handling.

**int len = sizeof(cliaddr);**

**int n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *) &cliaddr, &len);**

**buffer[n] = '\0';**

### Sending Data on Server Side

 **Purpose:** Sends data back to the client.
 **Parameters:**

- **hello**: Pointer to the message to send.
- **strlen(hello)**: Length of the message.
- **MSG_CONFIRM**: Confirm the data has been sent.
- **&cliaddr**: Pointer to the client's address.
- **sizeof(cliaddr)**: Size of the client address structure.

**sendto(sockfd, (const char *)hello, strlen(hello), MSG_CONFIRM, (const struct sockaddr *) &cliaddr, sizeof(cliaddr));**