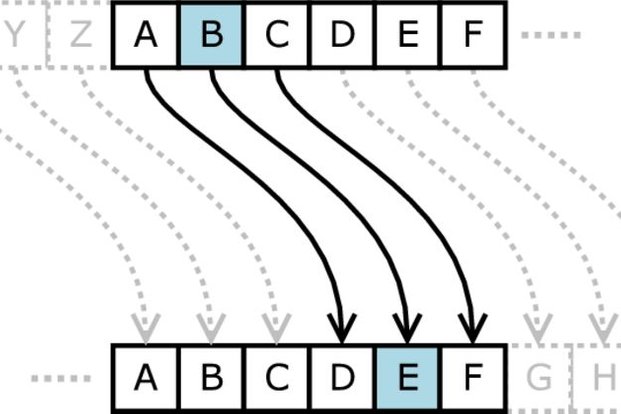


National Institute Of Technology

NIT Warangal

* **Name** : Hala Ali
* **Roll No**:187556
* **Class** : M.tech
* **Brach**:CSIS
* **Subject**: Network Security “LAB”
* **Assignment-1-**
* **Caesar Cipher Algorithm and its steps**



In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of +3 (to right) word "B" will become "E".

**Example**

To pass an encrypted message from one person to another, it is first necessary that both parties have the 'key' for the cipher, so that the sender may encrypt it and the receiver may decrypt it. For the caesar cipher, the key is the number of characters to shift the cipher alphabet.

Here is a quick example of the encryption and decryption steps involved with the caesar cipher. The text we will encrypt is 'hala ali ', with a shift (key) of 1.

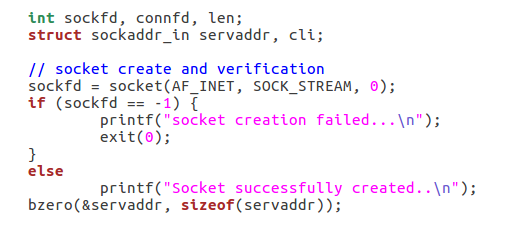
plaintext: hala ali

ciphertext: ibmb bmj

* **Server Code:**

The steps involved in establishing a socket on the server side are as follows:

1. Create a socket with the socket() system call
2. Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.
3. Listen for connections with the listen() system call
4. Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.
5. Send and receive data



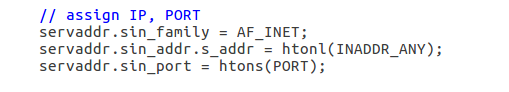
**int sockfd, connfd:** These two variables store the values returned by the socket system call and the accept system call.

**Len:**  stores the size of the address of the client. This is needed for the accept system call.

**struct sockaddr\_in servaddr, cli:** A sockaddr\_in is a structure containing an internet address. This structure is defined in <netinet/in.h>.

**sockfd = socket(AF\_INET, SOCK\_STREAM, 0);:** The socket() system call creates a new socket. It takes three arguments. The first is the address domain of the socket(AF\_INET :Internet domain), the second argument is the type of socket,  The third argument is the protocol. If this argument is zero the operating system will choose the most appropriate protocol. It will choose TCP for stream sockets and UDP for datagram sockets.

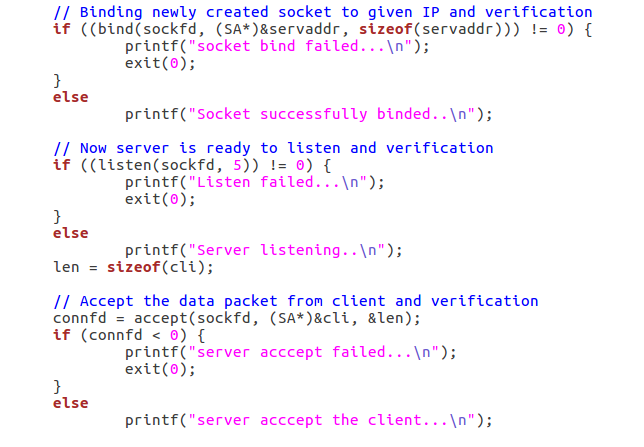
**bzero(&servaddr, sizeof(servaddr));:** he function bzero() sets all values in a buffer to zero. It takes two arguments, the first is a pointer to the buffer and the second is the size of the buffer. Thus, this line initializes serv\_addr to zeros.



**servaddr.sin\_family = AF\_INET; :** The variable serv\_addr is a structure of type struct sockaddr\_in. This structure has four fields. The first field is short sin\_family, which contains a code for the address family. It should always be set to the symbolic constant AF\_INET.

**servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY); :**The third field of sockaddr\_in is a structure of type struct in\_addr which contains only a single field unsigned long s\_addr. This field contains the IP address of the host. For server code, this will always be the IP address of the machine on which the server is running, and there is a symbolic constant INADDR\_ANY which gets this address.

**servaddr.sin\_port = htons(PORT); :**The second field of serv\_addr is unsigned short sin\_port , which contain the port number.

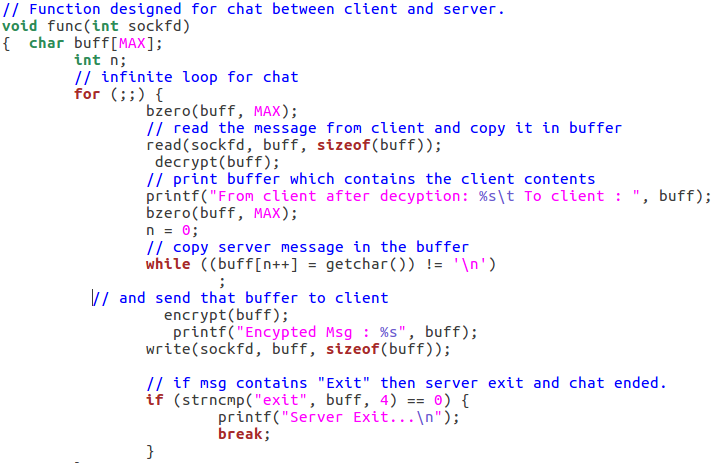


**bind(sockfd, (SA\*)&servaddr, sizeof(servaddr)):** The bind() system call binds a socket to an address.

**listen(sockfd, 5):** The listen system call allows the process to listen on the socket for connections. The first argument is the socket file descriptor, and the second is the size of the backlog queue .

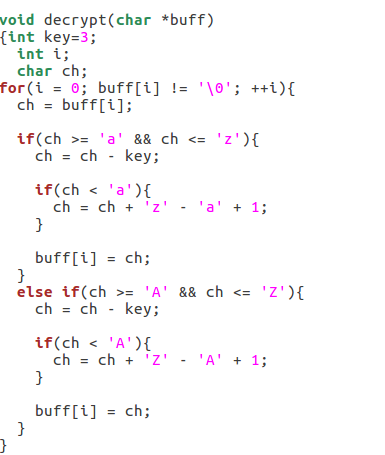
**accept(sockfd, (SA\*)&cli, &len):** The accept() system call causes the process to block until a client connects to the server.

After that we will call a function for chatting called: func



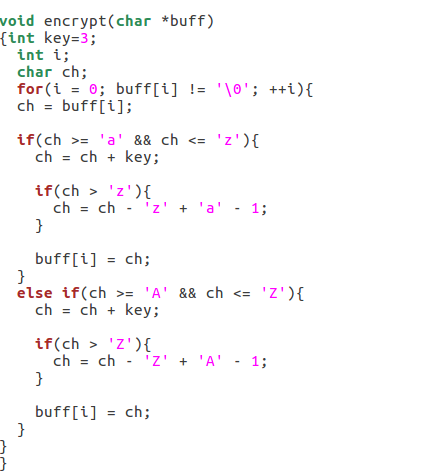
Once a connection has been established, both ends can both read and write to the connection. Naturally, everything written by the client will be read by the server, and everything written by the server will be read by the client.

After reading the received information in buffer we should decrypt it because this information send by client as encrypted information using Caesar Cipher Algorithm .



We also should encrypt information before writing it to the buffer and sent it, using encryption function “Caesar Cipher Algorithm” .

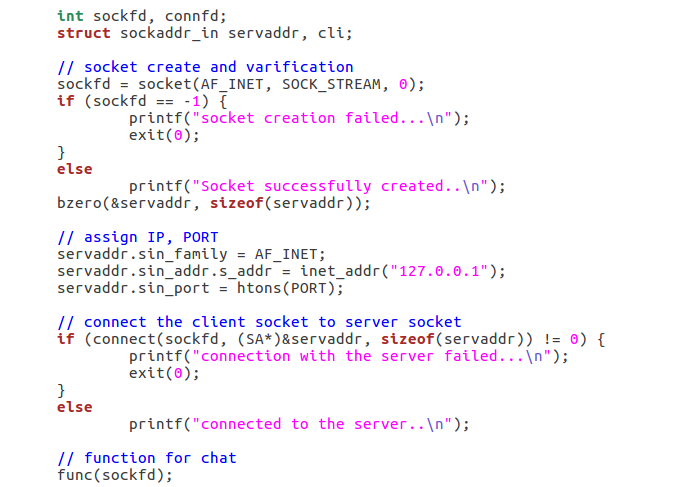
Here in the encryption and decryption functions we have fixed the size of key by 3 (means : shifting each character by 3).



* **Client Code:**

The steps involved in establishing a socket on the *client* side are as follows:

1. Create a socket with the socket() system call
2. Connect the socket to the address of the server using the connect() system call
3. Send and receive data. There are a number of ways to do this, but the simplest is to use the read() and write() system calls



**The connect function** is called by the client to establish a connection to the server. It takes three arguments, the socket file descriptor, the address of the host to which it wants to connect (including the port number), and the size of this address. This function returns 0 on success and -1 if it fails.

The client also use the same encrypt and decrypt functions that are used by the server .

* **OpenSSL** : is the best-known open library for secure communication. It started life in 1998 being derived from the SSLeay library developed by Eric Young and Tim Hudson. Other SSL toolkits include GNU TLS, distributed under the GNU General Public License, and Mozilla Network Security Services (NSS).

## What is SSL?

SSL is an acronym that stands for Secure Sockets Layer. It is the standard behind secure communication on the Internet, integrating data cryptography into the protocol. The data is encrypted before it even leaves your computer, and is decrypted only once it reaches its intended destination.

## What is OpenSSL?

OpenSSL is more than just SSL. It is capable of message digests, encryption and decryption of files, digital certificates, digital signatures, and random numbers.

OpenSSL is a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a general-purpose cryptography library.

### Installing Openssl library

Following command installs all the C libraries needed to use Openssl with your C code.

sudo apt-get install libssl-dev

For example, you will want to include the following header files:

**#include <openssl/evp.h>**

**#include <openssl/ssl.h>**

**#include <openssl/rsa.h>**

**#include <openssl/x509.h>**

### Compiling your C program with the Openssl library

gcc yourprogram.c -lssl -lcrypto

---------------------The End-------------------