Interprocess Communication using Sockets

- Know the telephone number of the person.
- Name of the person

Socket
One end of an interprocess communication channel.

```
struct sockaddr
{
    unsigned short sa_family; // Address family
    char sa_data[14];
};
```
struct sockaddr_in {  
  short sin_family; // eg. AF_INET, AF_INET6.
  unsigned short sin_port; // port # eg. 3000
  struct in_addr sin_addr; // IP address
  char sin_zero[8]; // zero-padding.
};

struct in_addr {  
  unsigned long s_addr; // 4-bytes IP address.
};

SERVER CODE

1. Declare Server and Client address structures.
struct sockaddr_in serv_addr;
struct sockaddr_in cli_addr;

2. Create a new socket.
sockfd = socket(AF_INET, SOCK_STREAM, 0);
  \[AF_UNIX, SOCK_DGRAM, protocol\]  
  0 = OS chooses for TCP/UDP.
3. Fill details in the `serv_addr` structure:

```c
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);
```

```
portno = atoi(argv[1]);
```

4. Bind the server address to the socket.

(II) to assigning a phone number to a telephone.
(a SIM card) to a mobile handset.

```c
bind(sockfd, (struct sockaddr *)&serv_addr,
     sizeof(serv_addr))
```

* Check return value for error.

```
retval < 0 → error in bind.
```

5. Listen on the socket.

```c
IP + port (assigned).
listen(sockfd, 5);
```

(Size of the backlog queue.)
6. Accept a connection from a client and create a new socket to handle that connection.

```c
newsockfd = accept ( sockfd, 
                     (struct sockaddr *) &cli_addr, 
                     &clilen);
```

Check for error.

7. Read data from client and store it in a buffer.

```c
char buffer[256];
bzero(buffer, 256);

n = read (newsockfd, buffer, 255);
```

// Check for error (N < 0 ⇒ error in reading)

// Print message.

```c
// Write some data for client to read.

n = write (newsockfd, "Got it!", 8);
```

// Check for error in writing.

8. END the server.
CLIENT CODE

1. Declare `serv_addr` and server host entry
   
   ```c
   struct sockaddr_in serv_addr;
   struct hostent *server;  // has a h_addr member.
   ```

2. Create a new client socket
   
   ```c
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   ```

3. Get Server's host entry using server's name
   
   ```c
   server = gethostbyname(argv[1]);
   ```
   
   // if `server` == NULL => no such host.

4. Set the fields in the `serv_addr` structure.
   
   ```c
   bzero((char *) &serv_addr, sizeof(serv_addr));
   serv_addr.sin_family = AF_INET;
   ```
bcopy (char *) server → h_addr,
    (char *) &serv_addr.sin_addr.s_addr,
    server → h_length);

* using bcopy since server→h_addr is a char *
    * (string).

void bcopy (char *s1, char *s2, int len);

    source    dest.    no.of.bytes.

    serv_addr.sin_port = htons (portno);

5) Establish a connection to the server.

    connect (sockfd, &serv_addr, sizeof(serv_addr));

    // check for error.

    IP + port no.

6) Write some data to a buffer and send it to
    the server.

    n = write (sockfd, buffer, strlen (buffer));

7) Read some data sent by the server.

    n = read (sockfd, read_buffer, 255);

8) END of client.
Socket Interface

Client

socket

connect

write

read

Server

socket

bind

listen

accept

read

write

CLOSE CONNECTION
**ADDITIONAL NOTES**

```c
struct hostent /* defined in <netdb.h> */
{
    char *h_name; /* official name of the host */
    char **h_aliases; /* alias list */
    int h_addrtype; /* host address type */
    int h_length; /* length of address */
    char **h_addr_list; /* list of addresses */
}

#define h_addr h_addr_list[0]
```
1. Create a socket:
```
sockfd = socket
```

2. Fill `serv_addr`:

<table>
<thead>
<tr>
<th>Family</th>
<th>AF_INET</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address</td>
<td>INADDR_ANY</td>
</tr>
<tr>
<td>Port</td>
<td>3000</td>
</tr>
</tbody>
</table>

3. Bind `serv_addr` to the socket.
4. Listen on the socket

5. Accept a connection from a client
1. Socket

2. Find the server's IP and port.

3. Set the fields in `soc_addr`.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family</td>
<td>AF_INET</td>
</tr>
<tr>
<td>IP address</td>
<td></td>
</tr>
<tr>
<td>Port #</td>
<td></td>
</tr>
</tbody>
</table>
4. Connect to the server.

5. Read/Write Data