

# EECS302002 計算機網路概論

## Linux socket Tutorial

1. Set up Linux from <https://blog.xuite.net/yh96301/blog/432341564-VirtualBox+5.1%E5%AE%89%E8%A3%9DUbuntu+16.04>

Download ubuntu for 16.04 LTS

2. Compiler

You can use terminal for programming(A) or you can download Visual Studio Code(B) for programming.

### A. terminal

- a. Use terminal to install gcc for compile: `sudo apt install gcc`

```
lighthan@lighthan-VirtualBox:~$ sudo apt install gcc
[sudo] password for lighthan:
正在讀取套件清單... 完成
正在重建相依關係
正在讀取狀態資料... 完成
下列的額外套件將被安裝：
gcc-7 libasan4 libatomic1 libc-dev-bin libc6-dev libcilkrts5 libgcc-7-dev
libitm1 liblsan0 libmpx2 libquadmath0 libtsan0 libubsan0 linux-libc-dev
manpages-dev
建議套件：
gcc-multilib make autoconf automake libtool flex bison gcc-doc
gcc-7-multilib gcc-7-doc gcc-7-locales libgcc1-dbg libgomp1-dbg libitm1-dbg
libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg
libcilkrts5-dbg libmpx2-dbg libquadmath0-dbg glibc-doc
下列【新】套件將會被安裝：
gcc gcc-7 libasan4 libatomic1 libc-dev-bin libc6-dev libcilkrts5
libgcc-7-dev libitm1 liblsan0 libmpx2 libquadmath0 libtsan0 libubsan0
linux-libc-dev manpages-dev
升級 0 個，新安裝 16 個，移除 0 個，有 11 個未被升級。
需要下載 18.8 MB 的套件檔。
此操作完成之後，會多佔用 75.2 MB 的磁碟空間。
是否繼續進行 [Y/n]? [Y/n] Y
```

- b. Use vim to start programming: `sudo apt install vim`

```
lighthan@lighthan-VirtualBox:~$ sudo apt install vim
[sudo] password for lighthan:
正在讀取套件清單... 完成
正在重建相依關係
正在讀取狀態資料... 完成
下列的額外套件將被安裝：
vim-common vim-runtime vim-tiny
建議套件：
ctags vim-doc vim-scripts indent
下列【新】套件將會被安裝：
vim vim-runtime
下列套件將會被升級：
vim-common vim-tiny
升級 2 個，新安裝 2 個，移除 0 個，有 9 個未被升級。
需要下載 6,589 kB/7,136 kB 的套件檔。
此操作完成之後，會多佔用 32.0 MB 的磁碟空間。
是否繼續進行 [Y/n]? [Y/n] Y
```

- c. Learn how to use vim, start from title "9.2 vi 的使用"

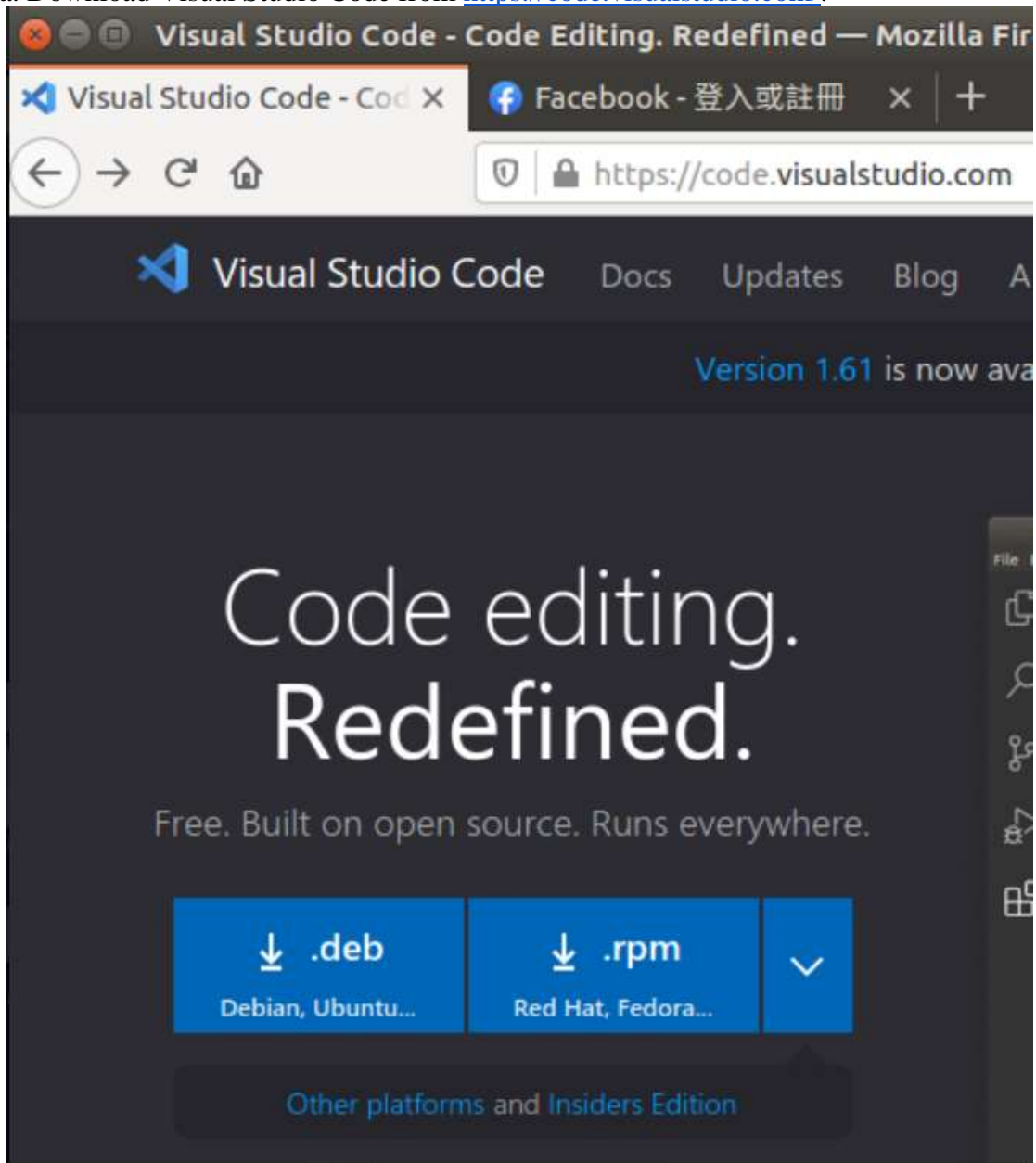
[http://linux.vbird.org/linux\\_basic/0310vi.php](http://linux.vbird.org/linux_basic/0310vi.php)

- d. How to compile and run your code

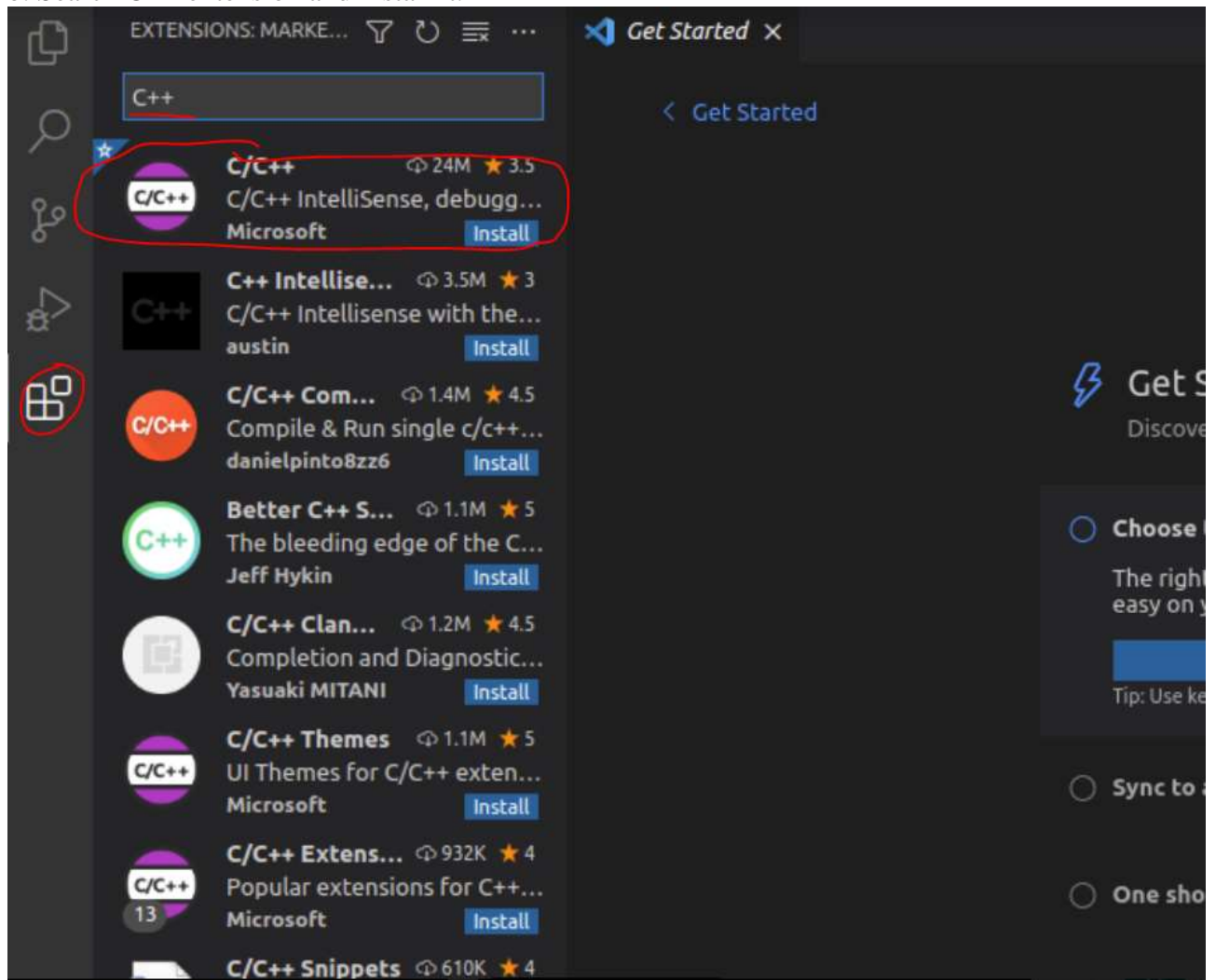
<https://blog.gtwang.org/programming/gcc-comipler-basic-tutorial-examples/>

B. Visual Studio Code

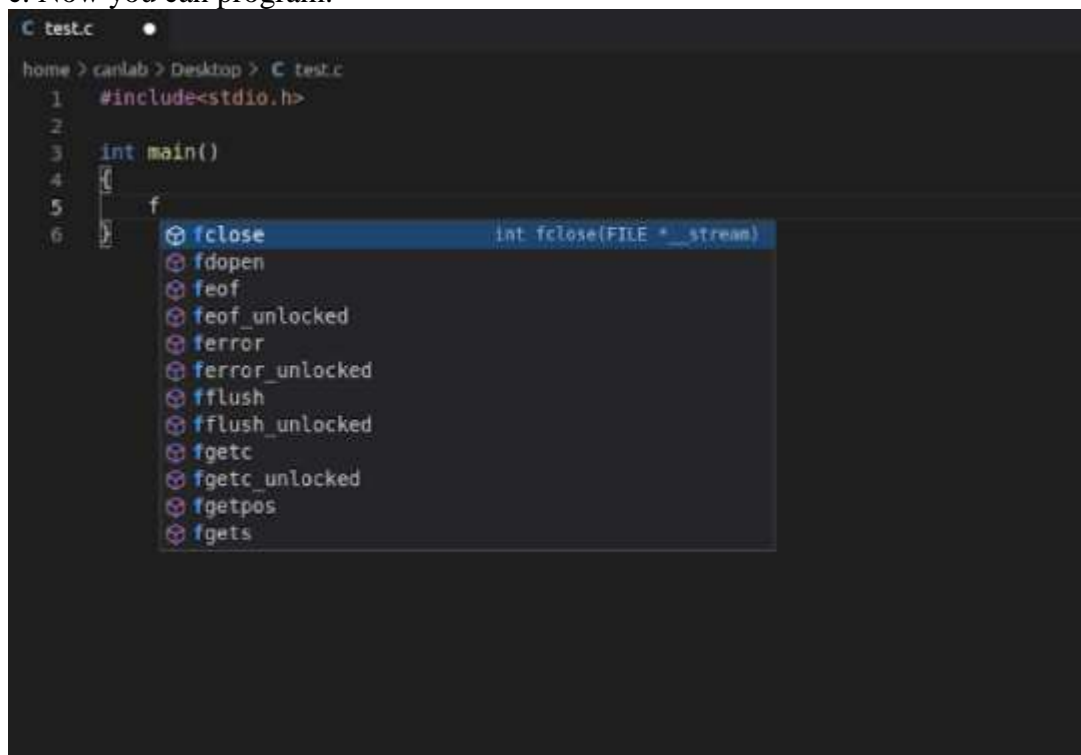
- a. Download Visual Studio Code from <https://code.visualstudio.com/>.



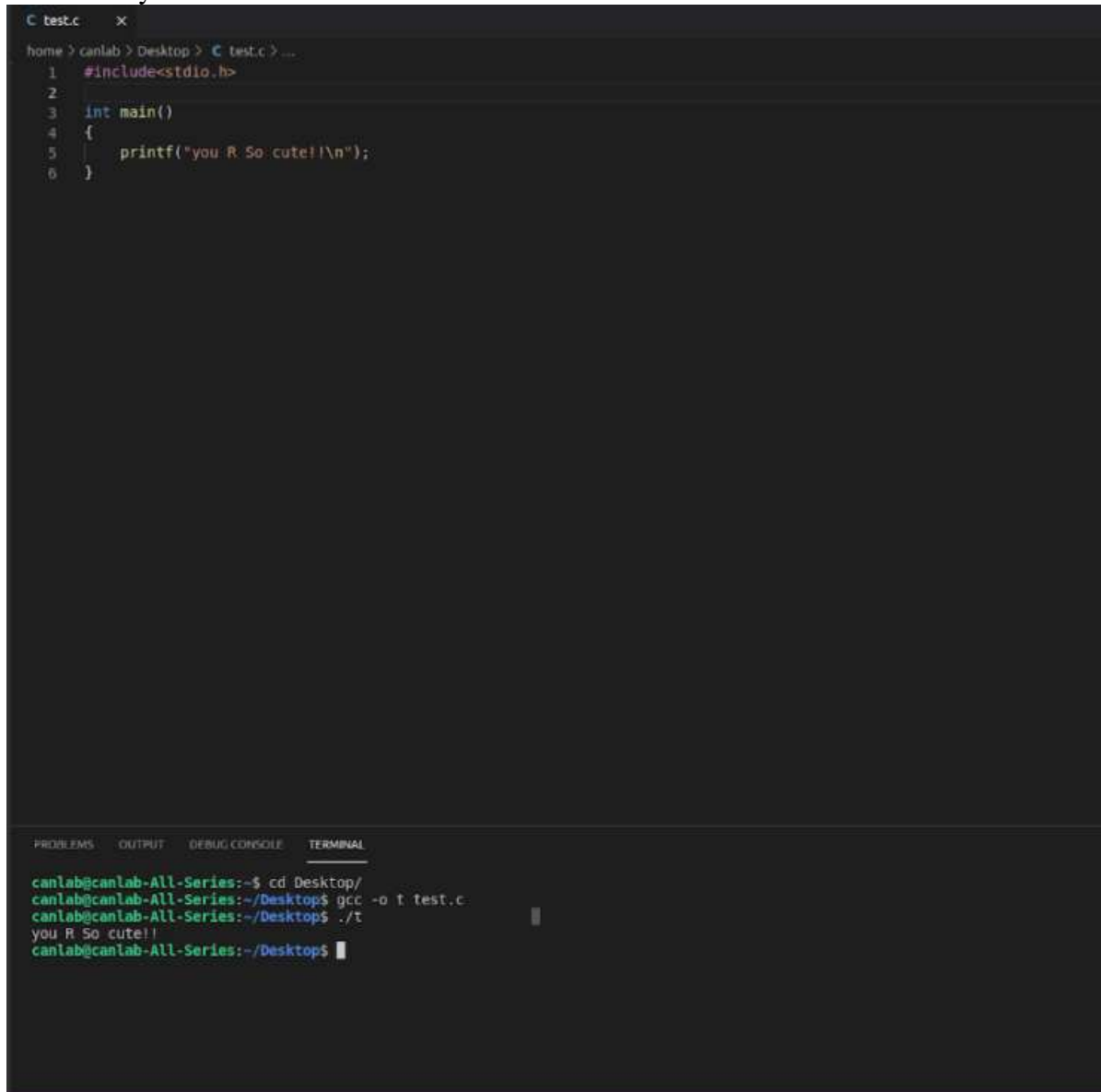
b. Search C++ extension and install it.



c. Now you can program.



d. Execute your code like that.



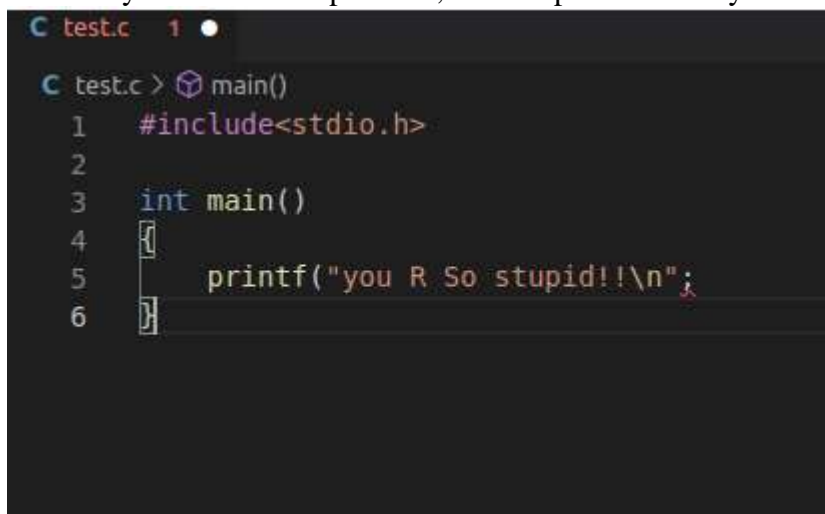
The screenshot shows a code editor window titled 'C test.c' with the following code:

```
1 #include<stdio.h>
2
3 int main()
4 {
5     printf("you R So cute!!\n");
6 }
```

Below the code editor is a terminal window. The terminal shows the following commands and output:

```
canlab@canlab-All-Series:~$ cd Desktop/
canlab@canlab-All-Series:~/Desktop$ gcc -o t test.c
canlab@canlab-All-Series:~/Desktop$ ./t
you R So cute!!
canlab@canlab-All-Series:~/Desktop$
```

e. When your code has a problem, the compiler will tell you where you might be wrong.



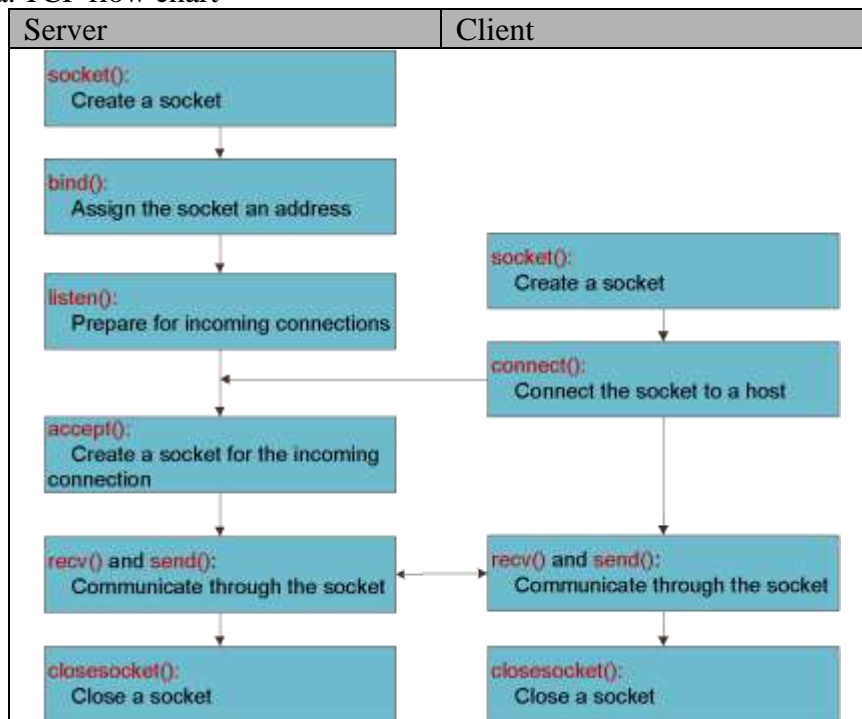
The screenshot shows a code editor window titled 'C test.c' with the following code:

```
1 #include<stdio.h>
2
3 int main()
4 {
5     printf("you R So stupid!!\n");
6 }
```

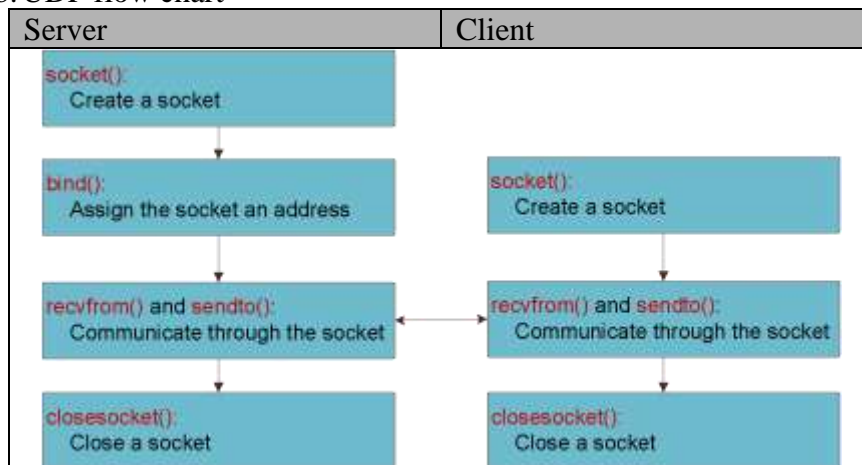
The code is syntactically correct, but the terminal output (not fully visible in the screenshot) would show an error if there was a problem. The error message would typically indicate the line number and the nature of the error.

### 3. Socket programming

#### a. TCP flow chart



#### b. UDP flow chart



### c. Data structure of address

Structure	Usage
<pre>struct sockaddr_in {     sa_family_t    sin_family;     in_port_t      sin_port;     struct in_addr sin_addr; };</pre>	<pre>sin_family = AF_INET; sin_port = htons(8080); sin_addr.sin_addr = INADDR_ANY; (for server)</pre>

\*htons(unsigned short port): host to network byte order for unsigned short (16 bits)

\*Use domain name instead of IP as the address

- struct sockaddr\_in serverAddress
- struct hostent \*h = gethostbyname(hostname)
- memcpy(&serverAddress.sin\_addr, h->h\_addr\_list[0], h->h\_length)

### d. Functions

<pre>socket(AF_INET, SOCK_STREAM, 0)</pre> <p>Create a TCP socket (<b>SOCK_STREAM</b>) or UDP socket (<b>SOCK_DGRAM</b>)</p>
<pre>bind(server_fd, (struct sockaddr *)&amp;address, sizeof(address))</pre> <p>Assign serverSocket serverAddress</p>
<pre>listen(server_fd, 3)</pre> <p>Prepare for incoming connections (maximum 3 connections)</p>
<pre>accept(server_fd, (struct sockaddr *)&amp;address, (socklen_t *)&amp;addrlen)</pre> <p>Create a socket for the incoming connection, and the address of the target host is stored in address</p>
<pre>send(new_socket, hello, strlen(hello), 0)</pre> <p>Send buf(hello) of size len (TCP socket)</p>
<pre>read(new_socket, buffer, 1024)</pre> <p>read data of maximum size 1024, and store the data in buffer (TCP socket)</p>
<pre>sendto(sockfd, (const char *)hello, strlen(hello), MSG_CONFIRM, (const struct sockaddr *)&amp;cliaddr, len);</pre> <p>Send buf of size len to client (UDP socket)</p>
<pre>recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *)&amp;cliaddr, &amp;len);</pre> <p>Receive data of maximum size MAXLINE, and store the data in buffer (UDP socket)</p>

#### 4. Examples

##### a. TCP echo server and client

###### TCP echo server

```
// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8080
int main(int argc, char const *argv[])
{
    int server_fd, new_socket, valread;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char *hello = "Hello from server";

    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
    // Forcefully attaching socket to the port 8080
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
        sizeof(opt)))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons( PORT );
    // Forcefully attaching socket to the port 8080
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address))<0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }
    valread = read( new_socket , buffer, 1024);
    printf("%s\n",buffer );
    send(new_socket , hello , strlen(hello) , 0 );
    printf("Hello message sent\n");
    return 0;
}
```

###### TCP echo client

```

// Client side C/C++ program to demonstrate Socket programming
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080

int main(int argc, char const *argv[])
{
    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    char *hello = "Hello from client";
    char buffer[1024] = {0};
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Socket creation error \n");
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to binary form
    if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0)
    {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
    send(sock , hello , strlen(hello) , 0 );
    printf("Hello message sent\n");
    valread = read( sock , buffer, 1024);
    printf("%s\n",buffer );
    return 0;
}

```



## b. UDP echo server and client

### UDP echo server

```
// Server side implementation of UDP client-server model
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT 8080
#define MAXLINE 1024

// Driver code
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from server";
    struct sockaddr_in servaddr, cliaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));

    // Filling server information
    servaddr.sin_family = AF_INET; // IPv4
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    // Bind the socket with the server address
    if ( bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0 )
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    int len, n;

    len = sizeof(cliaddr); //len is value/result

    n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, ( struct sockaddr *) &cliaddr, &len);
    buffer[n] = '\0';
    printf("Client : %s\n", buffer);
    sendto(sockfd, (const char *)hello, strlen(hello), MSG_CONFIRM, (const struct sockaddr *)
    &cliaddr, len);
    printf("Hello message sent.\n");

    return 0;
}
```

### UDP echo client

```
// Client side implementation of UDP client-server model
```

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT 8080
#define MAXLINE 1024

// Driver code
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));

    // Filling server information
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

    int n, len;

    sendto(sockfd, (const char *)hello, strlen(hello), MSG_CONFIRM, (const struct sockaddr *)
&servaddr, sizeof(servaddr));
    printf("Hello message sent.\n");

    n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *) &servaddr,
&len);
    buffer[n] = '\0';
    printf("Server : %s\n", buffer);

    close(sockfd);
    return 0;
}

```