

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

\*\*\*\*\*



# **BÁO CÁO**

**Bài tập thực hành :**

**Chương 3: Mở đầu về Socket**

**Học phần: Thực hành Lập Trình Mạng**

**Giảng viên hướng dẫn** : Trần Hải Anh

**Mã lớp** : 151907

**Sinh viên thực hiện** : Phạm Vân Anh

**Mã số sinh viên** : 20214988

**Hà Nội, tháng 9 năm 2024**

## Bài 1: Giới thiệu về Cấu trúc Địa chỉ Socket

**Yêu cầu 1:** Viết một chương trình khởi tạo cấu trúc `sockaddr_in` với một số cổng và địa chỉ IP (ví dụ: "192.168.1.1").

```
#include <stdio.h>
#include <string.h>
#include <arpa/inet.h>

int main() {
    struct sockaddr_in server_addr;

    memset(&server_addr, 0, sizeof(server_addr)); //set all byte = 0
    server_addr.sin_family = AF_INET; //set IPv4
    server_addr.sin_port = htons(8080); //set port = 8080

    // Convert the IP address from string to binary
    if (inet_pton(AF_INET, "192.168.1.1", &server_addr.sin_addr) <= 0){
        printf ("IP doesn't exist! \n");
        return 1;
    }

    //print
    printf("IP Address: %s\n", inet_ntoa(server_addr.sin_addr));
    printf("Port: %d\n", ntohs(server_addr.sin_port));

    return 0;
}
```

- Kết quả:

```
vaah@vaah-VirtualBox:~/Documents/lab3$ gcc sockaddr_in.c -o sockaddr_in
vaah@vaah-VirtualBox:~/Documents/lab3$ ./sockaddr_in
IP Address: 192.168.1.1
Port: 8080
vaah@vaah-VirtualBox:~/Documents/lab3$
```

## Bài 2: Các hàm Chuyển đổi Địa chỉ

### 2.1 inet\_aton và inet\_ntoa:

- `inet_aton`: Chuyển đổi một chuỗi (ví dụ: "192.168.1.1") thành struct `in_addr`.
- `inet_ntoa`: Chuyển đổi một `in_addr` thành chuỗi địa chỉ IP dễ đọc.

## 2.2 inet\_pton và inet\_ntop:

- inet\_pton: Chuyển đổi địa chỉ IP dạng văn bản (IPv4/IPv6) sang dạng nhị phân.
- inet\_ntop: Chuyển đổi địa chỉ IP dạng nhị phân (IPv4/IPv6) về dạng văn bản dễ đọc.

**Yêu cầu 2:** Viết một chương trình thực hiện:

- Nhập địa chỉ IP dưới dạng chuỗi từ người dùng.
- Sử dụng inet\_pton để chuyển địa chỉ sang dạng nhị phân.
- Chuyển địa chỉ nhị phân về dạng văn bản dễ đọc bằng inet\_ntop và in ra.

```
#include <stdio.h>
#include <string.h>
#include <arpa/inet.h>

int main() {
    char ip_str[INET_ADDRSTRLEN]; //save ip address in string
    struct sockaddr_in sa; //structure to hold the IP address in binary
    form

    //insert IP
    printf("Insert IP: ");
    fgets(ip_str, sizeof(ip_str), stdin); //read IP address
    ip_str[strcspn(ip_str, "\n")] = 0; //delete "\n"

    //convert IP from string to binary by pton
    if(inet_pton(AF_INET, ip_str, &sa.sin_addr) <= 0){
        printf("IP doesn't exist!\n");
        return 1;
    }

    //convert the binary IP address back to string by using inet_ntop
    char ip_buffer[INET_ADDRSTRLEN];
    if(inet_ntop(AF_INET, &sa.sin_addr, ip_buffer, sizeof(ip_buffer))
    == NULL) {
        printf ("Error!\n");
        return 1;
    }

    //print IP address
    printf ("IP address: %s\n", ip_buffer);

    return 0;
}
```

- Kết quả:

```
● vaah@vaah-VirtualBox:~/Documents/lab3$ gcc convert_IP.c -o convert_IP
● vaah@vaah-VirtualBox:~/Documents/lab3$ ./convert_IP
Insert IP: 192.168.1.10
IP address: 192.168.1.10
```

### Bài 3: I/O với Socket Stream

Yêu cầu 3: Tạo một chương trình client-server sử dụng tất cả các hàm đã học ở trên:

- Client đọc địa chỉ IP của server từ người dùng, chuyển đổi bằng `inet_pton`, và gửi thông điệp.
- Server nhận thông điệp, chuyển đổi địa chỉ của client bằng `sock_ntop`, và gửi phản hồi.

#### Server:

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

#define MAXLINE 1024
#define PORT 8080

ssize_t readline(int fd, void *vptr, size_t maxlen);

int main() {
    int listenfd, connfd;
    struct sockaddr_in servaddr;
    char buffer[MAXLINE];
    ssize_t n;

    // Create a listening socket
    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    if (listenfd < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }
}
```

```

// Initialize server address structure
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);

// Bind the socket to the address and port
if (bind(listenfd, (struct sockaddr *)&servaddr, sizeof(servaddr))
< 0) {
    perror("Bind failed");
    exit(EXIT_FAILURE);
}

// Listen for incoming connections
if (listen(listenfd, 10) < 0) {
    perror("Listen failed");
    exit(EXIT_FAILURE);
}

printf("Server listening on port %d...\n", PORT);

// Accept incoming connection
connfd = accept(listenfd, (struct sockaddr *)NULL, NULL);
if (connfd < 0) {
    perror("Accept failed");
    exit(EXIT_FAILURE);
}

// Read the message from the client using readline
while ((n = readline(connfd, buffer, MAXLINE)) > 0) {
    buffer[n] = '\0';
    printf("Received message: %s", buffer);

    // Send the received message back to the client
    if (write(connfd, buffer, n) < 0) {
        perror("Write failed");
        exit(EXIT_FAILURE);
    }
}

if (n < 0) {
    perror("Readline failed");
    exit(EXIT_FAILURE);
}

close(connfd);
close(listenfd);

```

```

        return 0;
    }

    // Function to read a line (up to \n) from a descriptor
    ssize_t readline(int fd, void *vptr, size_t maxlen) {
        ssize_t n, rc;
        char c, *ptr;

        ptr = vptr;
        for (n = 1; n < maxlen; n++) {
            if ((rc = read(fd, &c, 1)) == 1) {
                *ptr++ = c;
                if (c == '\n') {
                    break; // Stop at newline
                }
            } else if (rc == 0) {
                if (n == 1) {
                    return 0; // No data read
                } else {
                    break; // Some data was read
                }
            } else {
                return -1; // Error in read
            }
        }
        *ptr = 0;
        return n;
    }
}

```

## - Client:

```

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

#define MAXLINE 1024
#define PORT 8080

int main() {
    int sockfd;
    struct sockaddr_in servaddr;
    char sendline[MAXLINE], recvline[MAXLINE];
    ssize_t n;
}

```

```

// Create a socket
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}

// Initialize server address structure
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(PORT);

// Convert IPv4 address from text to binary form and set it
if (inet_pton(AF_INET, "127.0.0.1", &servaddr.sin_addr) <= 0) {
    perror("Invalid address or address not supported");
    exit(EXIT_FAILURE);
}

// Connect to the server
if (connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr))
< 0) {
    perror("Connection failed");
    exit(EXIT_FAILURE);
}


// Send a message to the server
printf("Enter message: ");
fgets(sendline, MAXLINE, stdin);
write(sockfd, sendline, strlen(sendline));

// Read the server's response
if ((n = read(sockfd, recvline, MAXLINE)) > 0) {
    recvline[n] = '\0'; // Null-terminate the received string
    printf("Server response: %s", recvline);
} else {
    perror("Read failed");
}

close(sockfd);
return 0;
}

```

## - Kết quả:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● vaah@vaah-VirtualBox:~/Documents/lab3$ gcc server.c -o server
● vaah@vaah-VirtualBox:~/Documents/lab3$ gcc client.c -o client
● vaah@vaah-VirtualBox:~/Documents/lab3$ ./server
Server listening on port 8080...
Received message: Hello guys
○ vaah@vaah-VirtualBox:~/Documents/lab3$

● vaah@vaah-VirtualBox:~/Documents/lab3$ ./client
Enter message: Hello guys
Server response: Hello guys
○ vaah@vaah-VirtualBox:~/Documents/lab3$
```