**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**\*\*\*\*\*\*\*\***

A picture containing icon

Description automatically generated

**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 exit! \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:

A screenshot of a computer

Description automatically generated

**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 adrress

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;

}

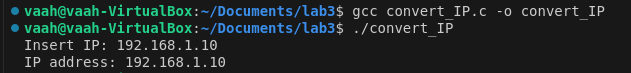
//print IP address

printf ("IP address: %s\n", ip\_buffer);

return 0;

}

* Kết quả:



**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:

A black rectangular object with white text

Description automatically generated