main.c

Share Run

Output

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

#define PORT 8080
#define BUFFER_SIZE 256

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[BUFFER_SIZE];

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }

    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt))) {
        perror("Setsockopt failed");
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;  // Listen on all interfaces
    address.sin_port = htons(PORT);          // Convert port to network byte order
```

```
/tmp/BhtAnbCHAD.o
Date and Time Server is listening on port 8080
```

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 5353
#define BUFFER_SIZE 512
#define DOMAIN "example.com"
#define IP_ADDRESS "93.184.216.34"

int main() {
    int sockfd;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);
    char buffer[BUFFER_SIZE];

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

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

```

Output Clear

```
/tmp/g3xGTGs25F.o
DNS Server is listening on port 5353
```

main.c

Share Run

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

#define SERVER_IP "127.0.0.1"
#define PORT 5353
#define DOMAIN "google.com"
#define BUFFER_SIZE 512

int main() {
    int sockfd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];

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

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr);
```

Output Clear

```
/tmp/p6LHscZqSh.o
Sent query for: google.com
```

main.c

Share Run

```
112     qinfo->qclass = htons(1); // Class IN
113
114     // Send the query
115     if (sendto(sock, (char *)buf, sizeof(struct DNS_HEADER) +
            (strlen((const char *)qname) + 1) + sizeof(struct QUESTION
            ), 0, (struct sockaddr *)&dest, sizeof(dest)) < 0) {
116         perror("Query sending failed");
117     } else {
118         // Receive the response
119         int i = sizeof dest;
120         if (recvfrom(sock, (char *)buf, 65536, 0, (struct sockaddr
                *)&dest, (socklen_t *)&i) < 0) {
121             perror("Response reception failed");
122         } else {
123             dns = (struct DNS_HEADER *) buf;
124
125             // Move to the answer section
126             unsigned char *reader = &buf[sizeof(struct DNS_HEADER)
                    + (strlen((const char *)qname) + 1) + sizeof(struct
                    QUESTION)];
127             printf("Resolved IP Addresses:\n");
128
129             // Read each answer record
130             for (i = 0; i < ntohs(dns->ans_count); i++) {
131                 struct RES_RECORD answer;
132                 answer.name = reader;
```

Output Clear

```
/tmp/NVEDkntiaC.o
Usage: /tmp/NVEDkntiaC.o <hostname>


=== Code Exited With Errors ===
```