```
#!/bin/bash
sudo apt update -y


sudo apt install -y apache2

sudo systemctl start apache2


sudo systemctl enable apache2

echo "<html><h1>Welcome to Apache Web Server on Ubuntu Linux!</h1></html>" >
/var/www/html/index.html
```

….

Evs drop ..

```
mkdir -p ~/udp-eavesdrop && cd ~/udp-eavesdrop

# Server
cat > UDPServer.cpp <<'EOF'
#include <iostream>
#include <cstring>
#include <arpa/inet.h>
#include <unistd.h>

#define BUFFER_SIZE 1024

std::string caesar_decrypt(const std::string &s, int shift) {
    std::string out = s;
    shift = shift % 26;
    for (size_t i = 0; i < out.size(); ++i) {
        char c = out[i];
        if (c >= 'A' && c <= 'Z') {
            out[i] = char((c - 'A' - shift + 26) % 26 + 'A');
        } else if (c >= 'a' && c <= 'z') {
            out[i] = char((c - 'a' - shift + 26) % 26 + 'a');
        } else {
            // leave other characters as-is (spaces, punctuation, digits)
            out[i] = c;
        }
    }
    return out;
}

std::string caesar_encrypt(const std::string &s, int shift) {
    std::string out = s;
    shift = shift % 26;
    for (size_t i = 0; i < out.size(); ++i) {
        char c = out[i];
        if (c >= 'A' && c <= 'Z') {
            out[i] = char((c - 'A' + shift + 26) % 26 + 'A');
        } else if (c >= 'a' && c <= 'z') {
            out[i] = char((c - 'a' + shift + 26) % 26 + 'a');
        } else {
            out[i] = c;
        }
    }
    return out;
}

int main(int argc, char* argv[]) {
    int port = 9876; // default port
    int shift = 3;   // default Caesar shift
    if (argc > 1) port = atoi(argv[1]);
    if (argc > 2) shift = atoi(argv[2]);

    int sockfd;
    struct sockaddr_in serverAddr, clientAddr;
    char buffer[BUFFER_SIZE];
    socklen_t addrLen = sizeof(clientAddr);

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

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    serverAddr.sin_port = htons(port);

    if (bind(sockfd, (const struct sockaddr*)&serverAddr, sizeof(serverAddr)) < 0) {
        perror("Bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }
    std::cout << "UDP server running on port " << port << " with Caesar shift " << shift
              << " (type 'exit' from client to stop)\n";

    while (true) {
        memset(buffer, 0, BUFFER_SIZE);
        int n = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, (struct sockaddr*)&clientAddr,
&addrLen);
        if (n < 0) {
            perror("recvfrom failed");
            continue;
        }

        std::string encrypted_msg(buffer, n);
        std::string decrypted_msg = caesar_decrypt(encrypted_msg, shift);

        std::cout << "Received (ciphertext): '" << encrypted_msg << "' from "
                  << inet_ntoa(clientAddr.sin_addr) << ":" << ntohs(clientAddr.sin_port) << "\n";
        std::cout << "Decrypted (plaintext): '" << decrypted_msg << "'\n";

        if (decrypted_msg == "exit") {
            std::string bye_plain = "Server shutting down as requested.";
            std::string bye_encrypted = caesar_encrypt(bye_plain, shift);
            sendto(sockfd, bye_encrypted.c_str(), bye_encrypted.size(), 0, (struct
sockaddr*)&clientAddr, addrLen);
            std::cout << "Exit command received. Shutting down server.\n";
            break;
        }

        std::string response_plain = "Server ACK: " + decrypted_msg;
        std::string response_encrypted = caesar_encrypt(response_plain, shift);
        sendto(sockfd, response_encrypted.c_str(), response_encrypted.size(), 0, (struct
sockaddr*)&clientAddr, addrLen);
    }

    close(sockfd);
    return 0;
}
EOF

# Client
cat > UDPClient.cpp <<'EOF'
#include <iostream>
#include <cstring>
```

```
#include <arpa/inet.h>
#include <unistd.h>

#define BUFFER_SIZE 1024

std::string caesar_encrypt(const std::string &s, int shift) {
    std::string out = s;
    shift = shift % 26;
    for (size_t i = 0; i < out.size(); ++i) {
        char c = out[i];
        if (c >= 'A' && c <= 'Z') {
            out[i] = char((c - 'A' + shift + 26) % 26 + 'A');
        } else if (c >= 'a' && c <= 'z') {
            out[i] = char((c - 'a' + shift + 26) % 26 + 'a');
        } else {
            out[i] = c;
        }
    }
    return out;
}

std::string caesar_decrypt(const std::string &s, int shift) {
    std::string out = s;
    shift = shift % 26;
    for (size_t i = 0; i < out.size(); ++i) {
        char c = out[i];
        if (c >= 'A' && c <= 'Z') {
            out[i] = char((c - 'A' - shift + 26) % 26 + 'A');
        } else if (c >= 'a' && c <= 'z') {
            out[i] = char((c - 'a' - shift + 26) % 26 + 'a');
        } else {
            out[i] = c;
        }
    }
    return out;
}

int main(int argc, char* argv[]) {
    const char* serverIP = "127.0.0.1";
    int serverPort = 9876;
    int shift = 3; // default Caesar shift

    if (argc >= 2) serverIP = argv[1];
    if (argc >= 3) serverPort = atoi(argv[2]);
    if (argc >= 4) shift = atoi(argv[3]);

    int sockfd;
    struct sockaddr_in serverAddr;
    char buffer[BUFFER_SIZE];

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

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(serverPort);
    inet_pton(AF_INET, serverIP, &serverAddr.sin_addr);

    socklen_t addrLen = sizeof(serverAddr);

    std::cout << "UDP client started. Sending to " << serverIP << ":" << serverPort
              << " with Caesar shift " << shift << "\n";
    std::cout << "Type messages and press Enter. Type 'exit' to stop server and exit.\n";

    while (true) {
        std::cout << "You: ";
        std::string plain;
        std::getline(std::cin, plain);

        std::string encrypted = caesar_encrypt(plain, shift);
        sendto(sockfd, encrypted.c_str(), encrypted.size(), 0, (struct sockaddr*)&serverAddr,
addrLen);

        int n = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, (struct sockaddr*)&serverAddr,
&addrLen);
        if (n < 0) {
            perror("recvfrom error");
            break;
        }
        std::string resp_encrypted(buffer, n);
        std::string resp_plain = caesar_decrypt(resp_encrypted, shift);

        std::cout << "Server (ciphertext): '" << resp_encrypted << "'\n";
        std::cout << "Server (decrypted): '" << resp_plain << "'\n";

        if (plain == "exit") break;
    }
    close(sockfd);
    return 0;
}
EOF
```

..

```
g++ UDPServer.cpp -o UDPServer
g++ UDPClient.cpp -o UDPClient
```

he compile la

..

```
cd ~/udp-eavesdrop
./UDPServer 9876 3
```

he server sathi

…

```
cd ~/udp-eavesdrop
./UDPClient 127.0.0.1 9876 3
```

he client sathi

..

```
sudo tcpdump -i lo -nn -s 0 udp port 9876 -A
```

he attacker sathi (third terminal)

..