Implementation of sendto() in c++

```
#include <iostream>

#include <cstring>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <unistd.h>


int main() {

    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    sockaddr_in dest_addr{};

    std::memset(&dest_addr, 0, sizeof(dest_addr));

    dest_addr.sin_family = AF_INET;

    dest_addr.sin_port = htons(12345);

    inet_pton(AF_INET, "192.0.2.1", &dest_addr.sin_addr);


    std::string msg = "hello";

    ssize_t sent = sendto(sockfd, msg.c_str(), msg.size(), 0,

                          reinterpret_cast<sockaddr*>(&dest_addr), sizeof(dest_addr));

    if (sent == -1) perror("sendto failed");

    close(sockfd);

    return 0;

}
```

Overview

This C++ program demonstrates how to create a simple UDP client that sends a message to a specified server IP address and port using the sendto() system call. The example is designed for educational purposes and illustrates basic socket programming concepts in a UNIX-like environment.

Code Description

1. Socket Creation:

    - The program creates a UDP socket using the socket() function with AF_INET (IPv4) and SOCK_DGRAM (UDP) parameters.

2. Destination Address Setup:

    - A sockaddr_in structure is initialized to specify the destination address. The IP address ("192.0.2.1") and port number (12345) are set, with the port number being converted from host byte order to network byte order using htons().

3. Message Preparation:

    - A string message ("hello") is prepared for sending.

4. Sending Data:

    - The sendto() function is called to send the message to the specified destination. If the sending fails, an error message is printed using perror().

5. Cleanup:

    - The socket is closed using close() to release system resources.

Usage

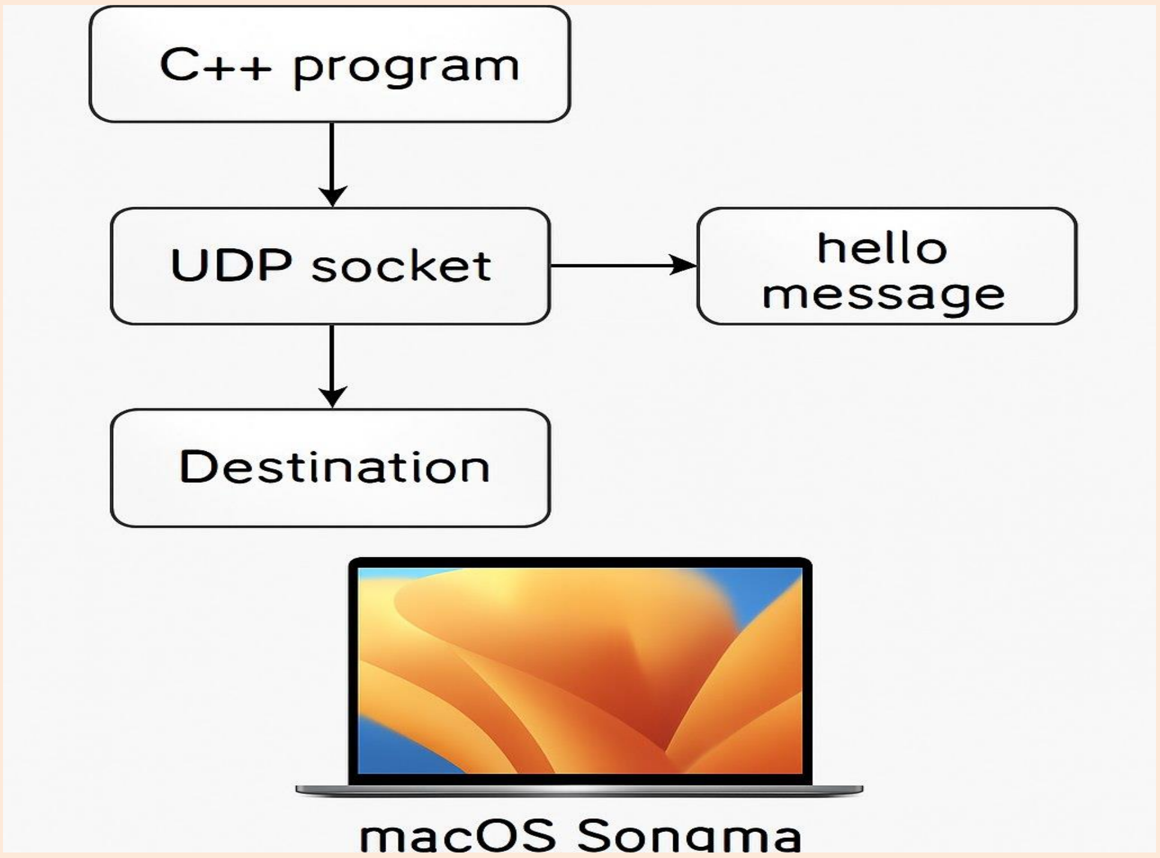
- To compile the code, use a C++ compiler (e.g., g++) in a terminal:

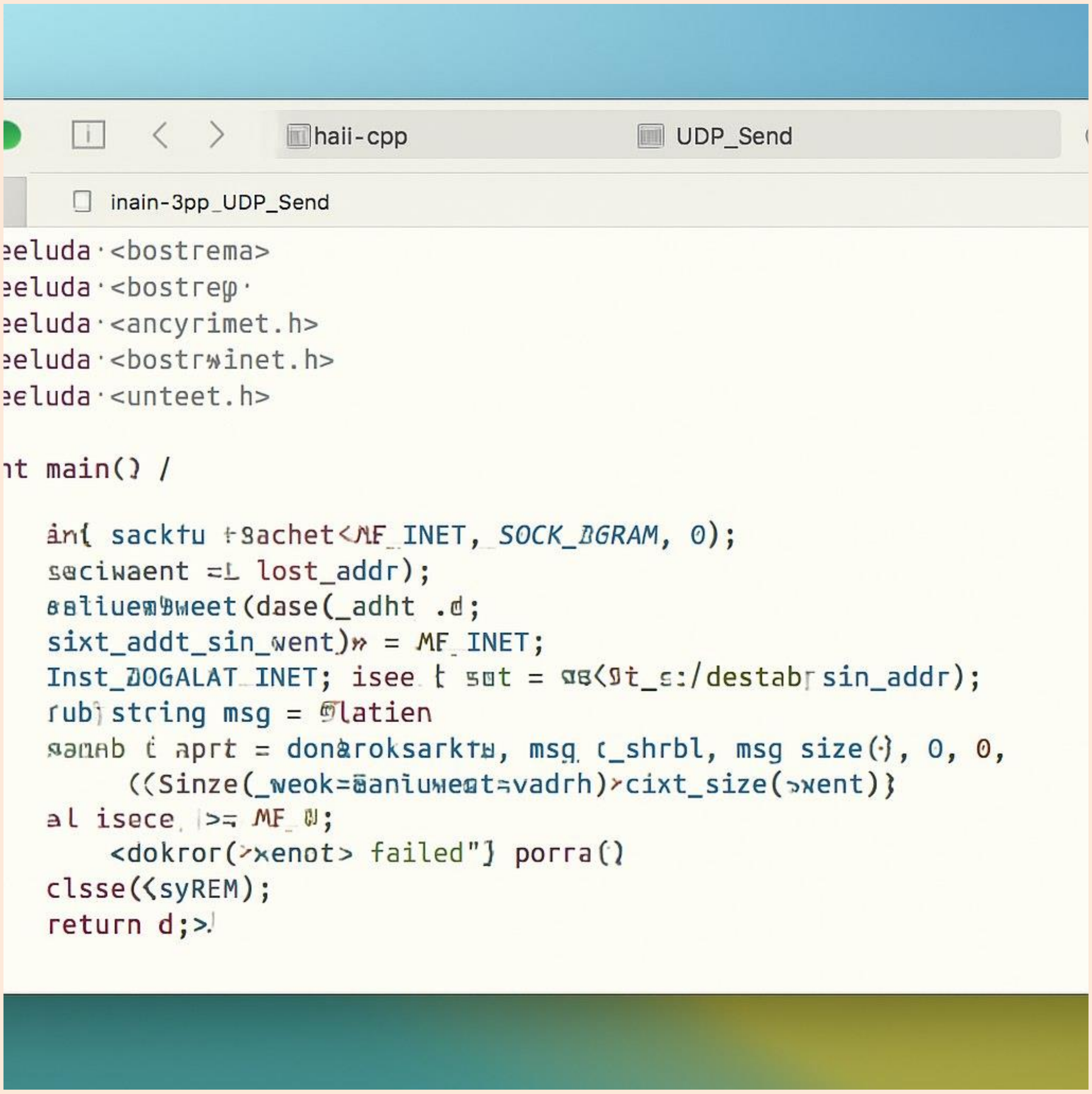
Run the compiled program:

Notes

- ❖ Ensure that there is a UDP server listening on the specified IP address and port (192.0.2.1:12345) to receive the message.
- ❖ Modify the IP address and port as needed for your specific use case.
- ❖ This code does not handle errors related to socket creation or address conversion; additional error checking may be implemented for robustness.

A simple diagram that shows how this message sends to specific address



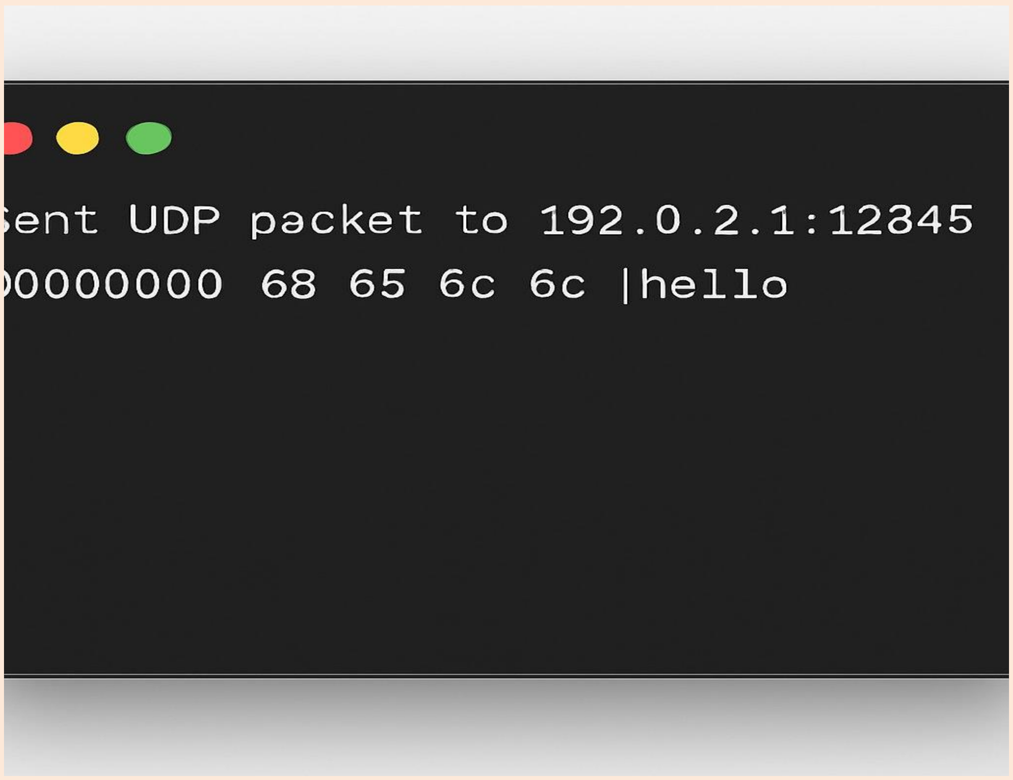


```
eeluda·<bostrema>
eeluda·<bostrep·
eeluda·<ancyrimet.h>
eeluda·<bostrwinet.h>
eeluda·<unteet.h>

nt main() /

   ain( sacktu +8achet<MF_INET, SOCK_DGRAM, 0);
   seciwaent =L lost_addr);
   esliuem9weet(dase(_adht .d;
   sixt_addt_sin_went)» = MF_INET;
   Inst_DOGALAT INET; isee t sut = as(9t_s:/destabr sin_addr);
   rub} string msg = Glatien
   samab t aprt = donaroksarktu, msg c_shrbl, msg size(), 0, 0,
        ((Sinze(_weok=8aniuweat=vadrh)›cixt_size(›went)}
   al isece >= MF_0;
       <dokror(›xenot> failed"] porra()
   clsse(<syREM);
   return d;>
```

This is the implementation of the above c++ code of **sendto()** on the Mac OS Sonoma workstation.



```
ent UDP packet to 192.0.2.1:12345
0000000 68 65 6c 6c |hello
```

This is the output of the above code on **Mac OS Sonoma terminal.**