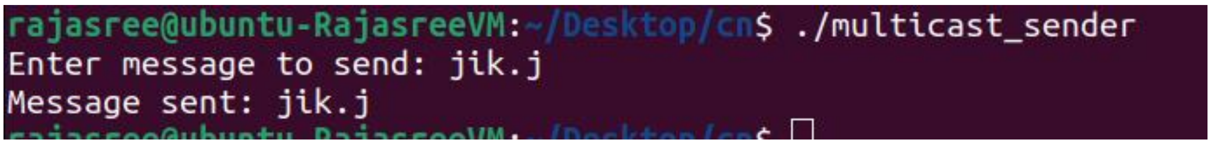
# /*multicast sender receiver*/

## //multicast sender

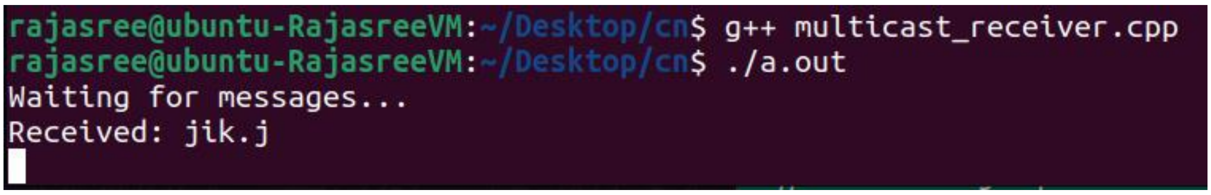
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define MULTICAST_GROUP "239.0.0.1"
#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
  int sockfd;
  struct sockaddr_in multicast_addr;
  char buffer[BUFFER_SIZE];
  int ttl = 1; // Time-To-Live (TTL) for multicast packets

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

  // Set TTL for multicast packets
  if (setsockopt(sockfd, IPPROTO_IP, IP_MULTICAST_TTL, (void *)&ttl, sizeof(ttl)) < 0) {
    perror("setsockopt failed");
```

```
        exit(EXIT_FAILURE);
    }

    // Fill multicast group information
    memset(&multicast_addr, 0, sizeof(multicast_addr));
    multicast_addr.sin_family = AF_INET;
    multicast_addr.sin_addr.s_addr = inet_addr(MULTICAST_GROUP);
    multicast_addr.sin_port = htons(PORT);

    // Input message to send
    printf("Enter message to send: ");
    fgets(buffer, BUFFER_SIZE, stdin);

    // Send multicast message
    if (sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr *)&multicast_addr,
sizeof(multicast_addr)) < 0) {
        perror("sendto failed");
        exit(EXIT_FAILURE);
    }

    printf("Message sent: %s", buffer);

    // Close socket
    close(sockfd);

    return 0;
}
```



```
rajasree@ubuntu-RajasreeVM:~/Desktop/cn$ ./multicast_sender
Enter message to send: jik.j
Message sent: jik.j
```

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

#define MULTICAST_GROUP "239.0.0.1"
#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int sockfd;
    struct sockaddr_in multicast_addr;
    struct ip_mreq multicast_req;
    char buffer[BUFFER_SIZE];

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

    // Fill multicast group information
    memset(&multicast_addr, 0, sizeof(multicast_addr));
    multicast_addr.sin_family = AF_INET;
    multicast_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    multicast_addr.sin_port = htons(PORT);
    // Bind socket to receive address
    if (bind(sockfd, (struct sockaddr *)&multicast_addr, sizeof(multicast_addr)) < 0) {
        perror("bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    // Join the multicast group
    multicast_req.imr_multiaddr.s_addr = inet_addr(MULTICAST_GROUP);
    multicast_req.imr_interface.s_addr = htonl(INADDR_ANY);
    if (setsockopt(sockfd, IPPROTO_IP, IP_ADD_MEMBERSHIP, (void *)&multicast_req,
sizeof(multicast_req)) < 0) {
        perror("setsockopt failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    // Receive multicast messages
    printf("Waiting for messages...\n");
    while (1) {
        ssize_t num_bytes = recv(sockfd, buffer, BUFFER_SIZE - 1, 0);
        if (num_bytes < 0) {
            perror("recv failed");
            close(sockfd);
            exit(EXIT_FAILURE);
        }
        buffer[num_bytes] = '\0';
        printf("Received: %s", buffer);
    }
    // Close socket
    close(sockfd);

    return 0;
}
```



```
rajasree@ubuntu-RajasreeVM:~/Desktop/cn$ g++ multicast_receiver.cpp
rajasree@ubuntu-RajasreeVM:~/Desktop/cn$ ./a.out
Waiting for messages...
Received: jik.j
```