Gita Alekhya Paul **08/08/2021**

Computer Networks
RA1911030010014
Experiment - 4
UDP Echo Client-Server Communication

Aim: **To create a UDP Echo Client-Server Communication.**

Server Code:

```c
/**
* RA1911030010014 - UDP Server
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT 4040
#define MAXLINE 1024

int main()
{
   int sockfd;
   char buffer[MAXLINE];
   struct sockaddr_in servaddr, cliaddr;
   if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
   {
       perror("socket creation failed");
       exit(EXIT_FAILURE);
   }
   memset(&servaddr, 0, sizeof(servaddr));
   memset(&cliaddr, 0, sizeof(cliaddr));
   servaddr.sin_family = AF_INET;
   servaddr.sin_addr.s_addr = INADDR_ANY;
   servaddr.sin_port = htons(PORT);

   if (bind(sockfd, (const struct sockaddr *)&servaddr,
            sizeof(servaddr)) < 0)
   {
       perror("bind failed");
       exit(EXIT_FAILURE);
   }
   else
   {
       printf("Server listening on Port %d\n", PORT);
   }
   int len, n;
```

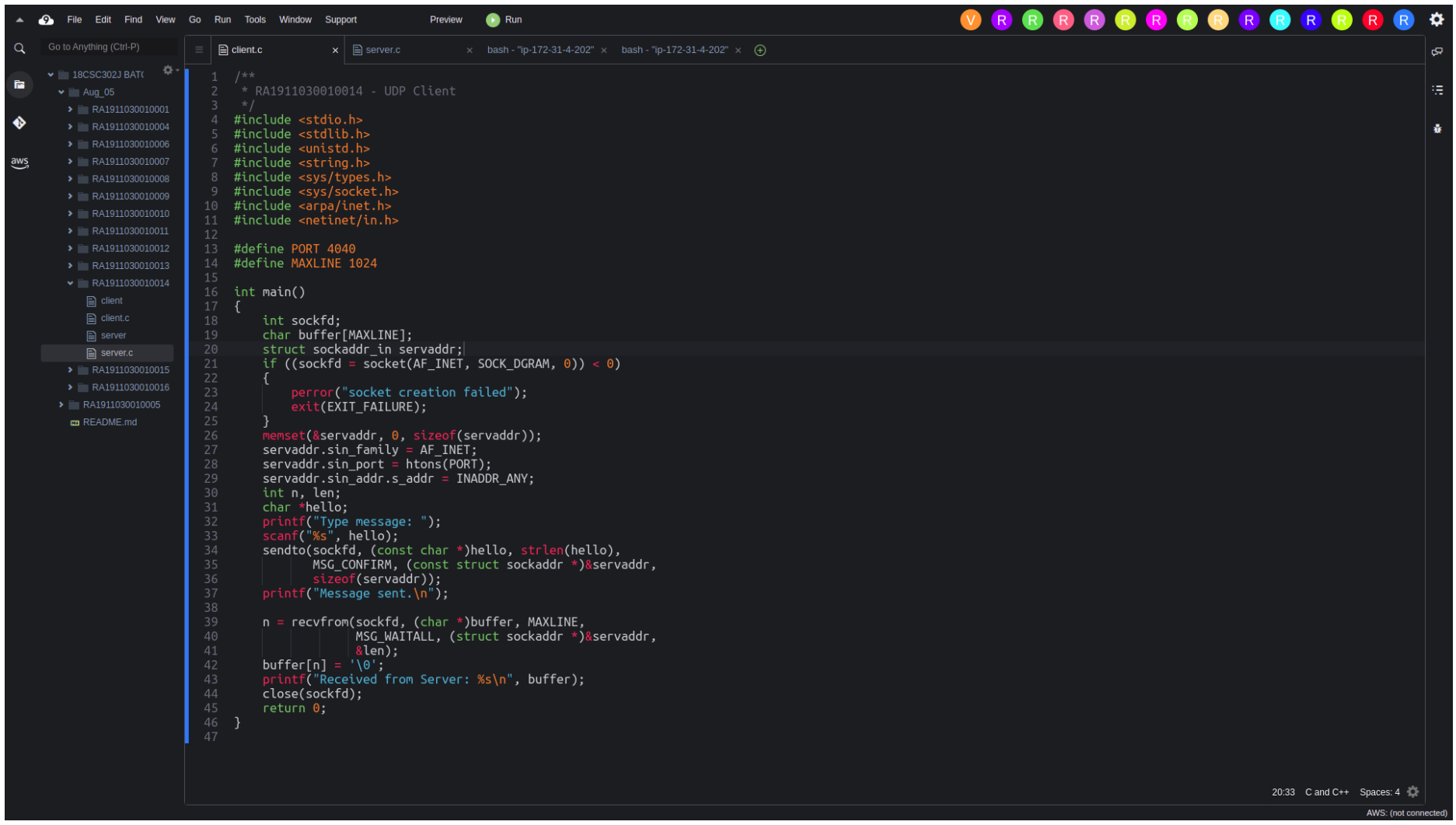
```
    len = sizeof(cliaddr);
    n = recvfrom(sockfd, (char *)buffer, MAXLINE,
                 MSG_WAITALL, (struct sockaddr *)&cliaddr,
                 &len);
    buffer[n] = '\0';
    printf("Client Sent : %s\n", buffer);
    sendto(sockfd, (const char *)buffer, strlen(buffer),
           MSG_CONFIRM, (const struct sockaddr *)&cliaddr,
           len);
    return 0;
}
```

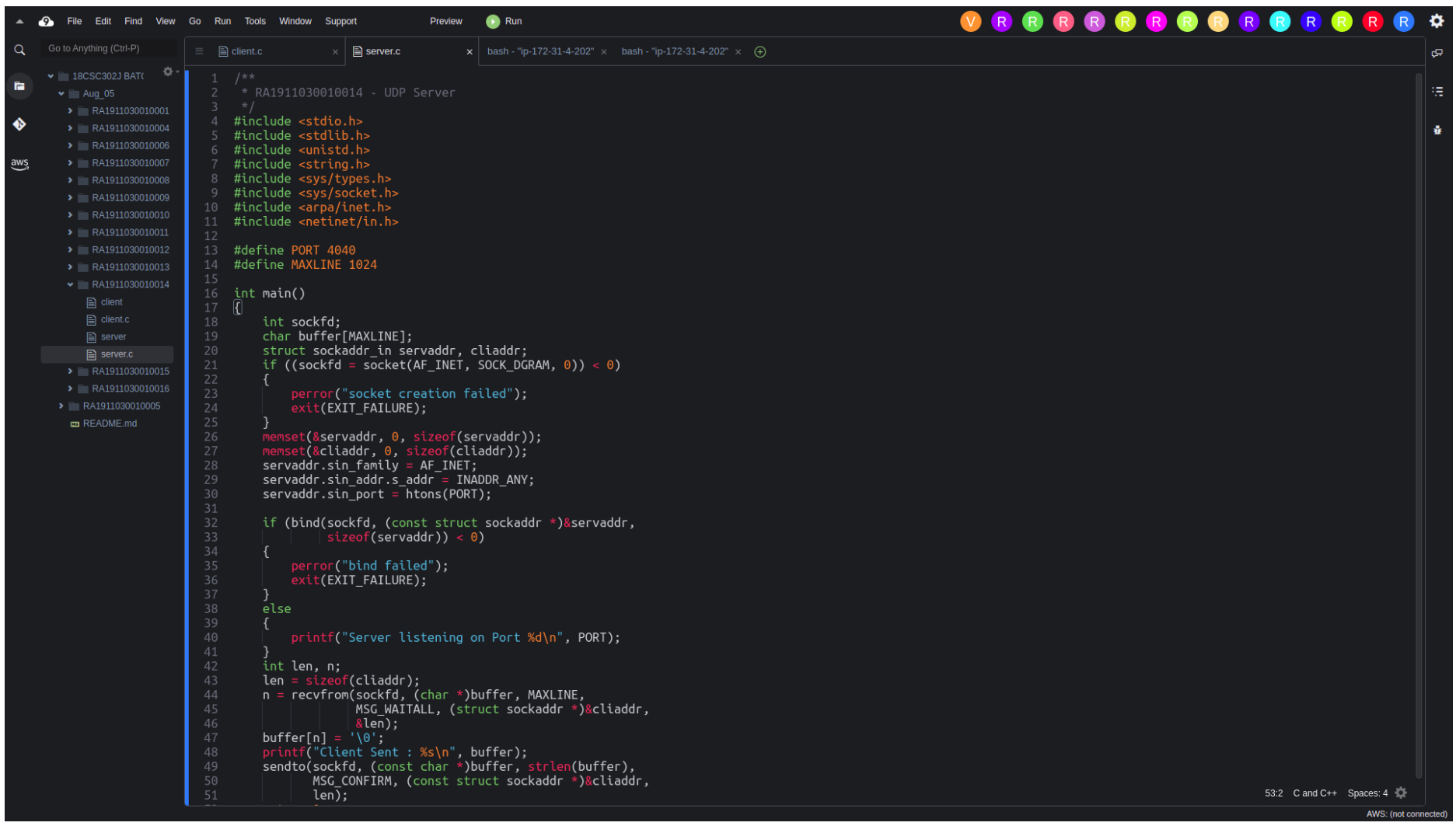
Client Code:

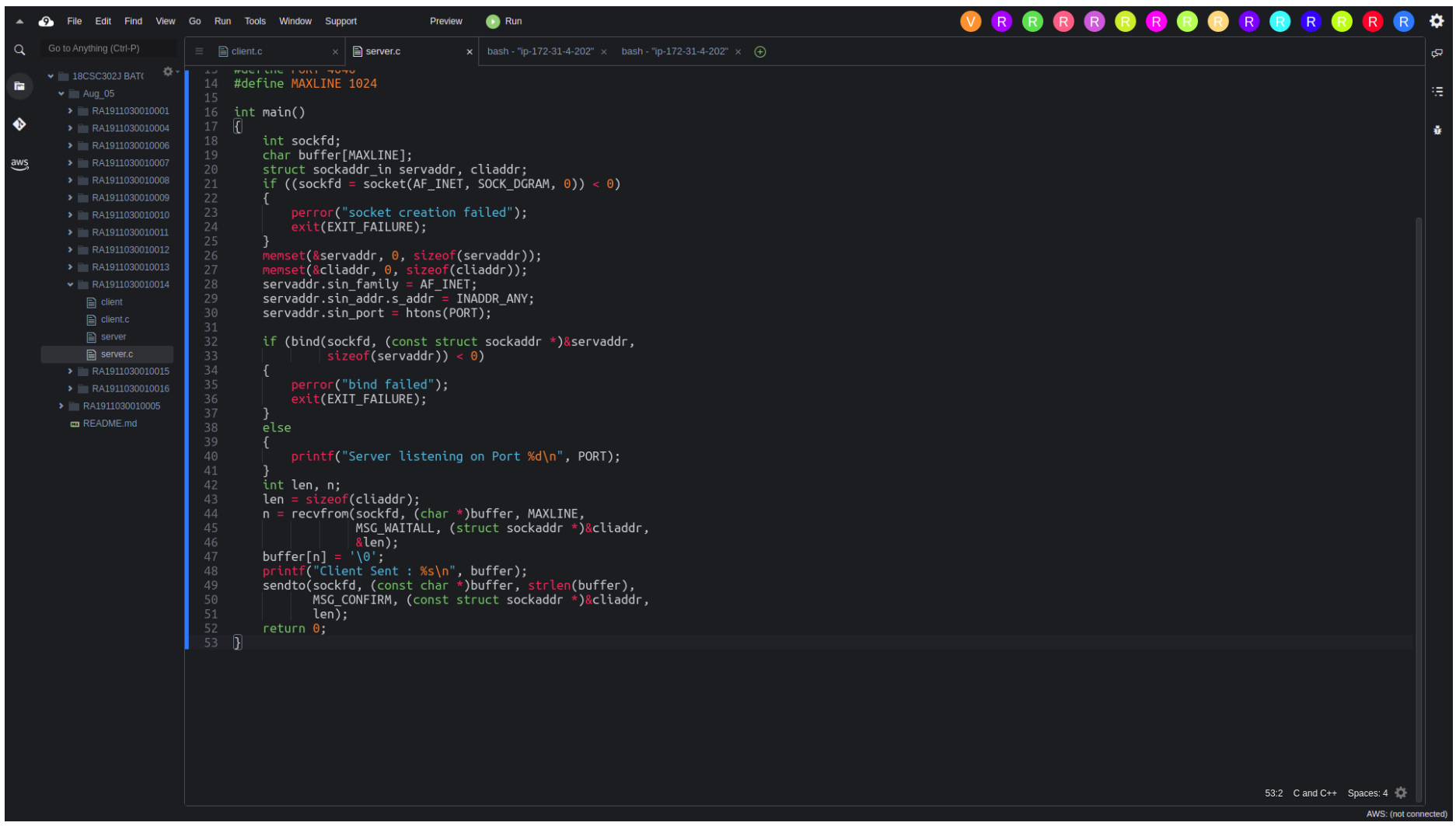
```
/**
* RA1911030010014 - UDP Client
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT 4040
#define MAXLINE 1024

int main()
{
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr;
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
    {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;
    int n, len;
    char *hello;
    printf("Type message: ");
    scanf("%s", hello);
    sendto(sockfd, (const char *)hello, strlen(hello),
           MSG_CONFIRM, (const struct sockaddr *)&servaddr,
```

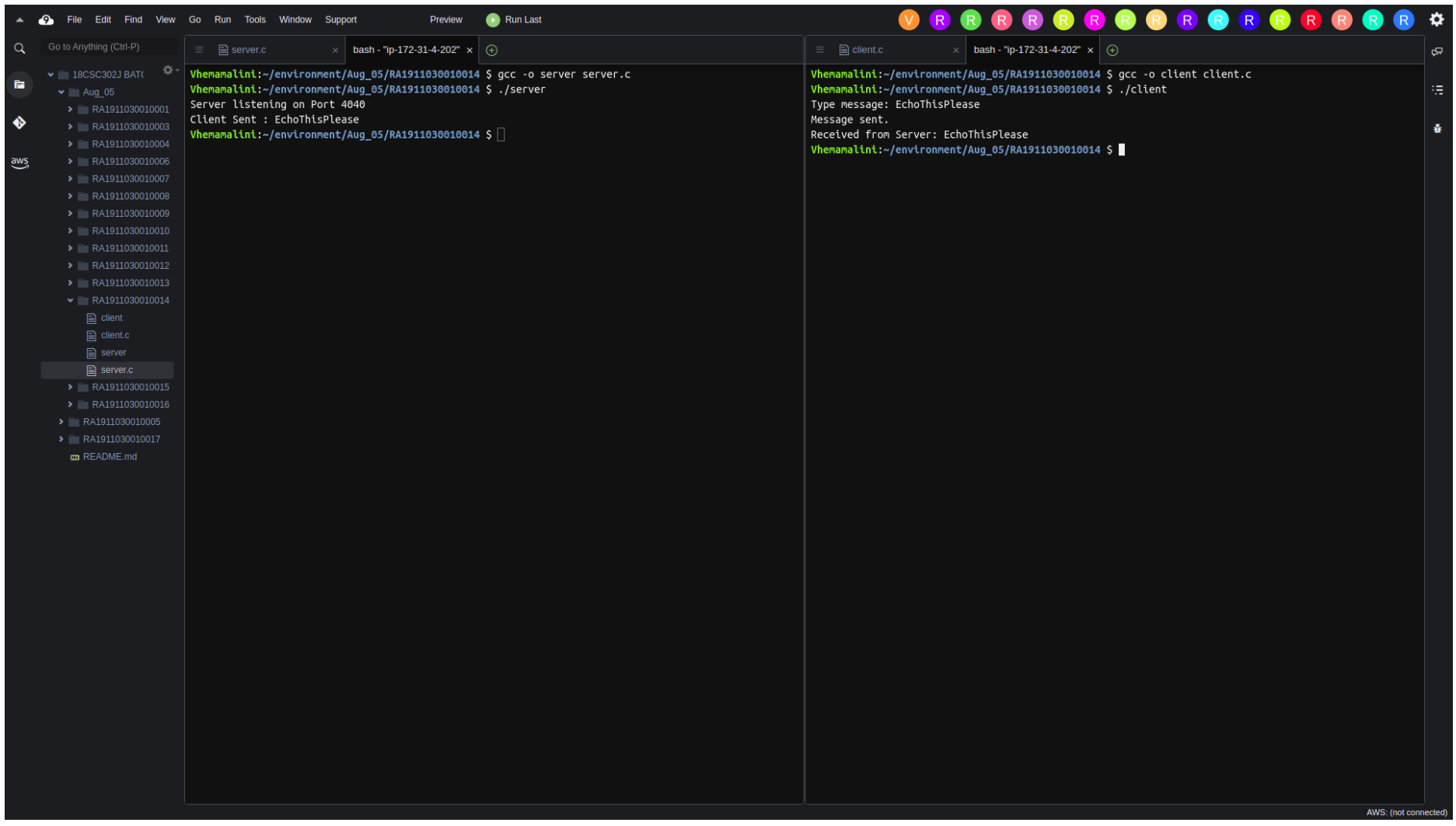
```
            sizeof(servaddr));
    printf("Message sent.\n");

    n = recvfrom(sockfd, (char *)buffer, MAXLINE,
                 MSG_WAITALL, (struct sockaddr *)&servaddr,
                 &len);
    buffer[n] = '\0';
    printf("Received from Server: %s\n", buffer);
    close(sockfd);
    return 0;
}
```

Output:




File Edit Find View Go Run Tools Window Support Preview Run
Go to Anything (Ctrl-P)
client.c
server.c
bash - "ip-172-31-4-202"
bash - "ip-172-31-4-202"
18CSC302J BAT(
Aug_05
RA1911030010001
RA1911030010004
RA1911030010006
RA1911030010007
RA1911030010008
RA1911030010009
RA1911030010010
RA1911030010011
RA1911030010012
RA1911030010013
RA1911030010014
client
client.c
server
server.c
RA1911030010015
RA1911030010016
RA1911030010005
README.md
/**
* RA1911030010014 - UDP Server
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#define PORT 4040
#define MAXLINE 1024
int main()
{
int sockfd;
char buffer[MAXLINE];
struct sockaddr_in servaddr, cliaddr;
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
{
perror("socket creation failed");
exit(EXIT_FAILURE);
}
memset(&servaddr, 0, sizeof(servaddr));
memset(&cliaddr, 0, sizeof(cliaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = INADDR_ANY;
servaddr.sin_port = htons(PORT);
if (bind(sockfd, (const struct sockaddr *)&servaddr,
sizeof(servaddr)) < 0)
{
perror("bind failed");
exit(EXIT_FAILURE);
}
else
{
printf("Server listening on Port %d\n", PORT);
}
int len, n;
len = sizeof(cliaddr);
n = recvfrom(sockfd, (char *)buffer, MAXLINE,
MSG_WAITALL, (struct sockaddr *)&cliaddr,
&len);
buffer[n] = '\0';
printf("Client Sent : %s\n", buffer);
sendto(sockfd, (const char *)buffer, strlen(buffer),
MSG_CONFIRM, (const struct sockaddr *)&cliaddr,
len);
53:2 C and C++ Spaces: 4
AWS: (not connected)


File Edit Find View Go Run Tools Window Support Preview Run
Go to Anything (Ctrl-P)
client.c
server.c
bash - "ip-172-31-4-202"
bash - "ip-172-31-4-202"
18CSC302J BAT(
Aug_05
RA1911030010001
RA1911030010004
RA1911030010006
RA1911030010007
RA1911030010008
RA1911030010009
RA1911030010010
RA1911030010011
RA1911030010012
RA1911030010013
RA1911030010014
client
client.c
server
server.c
RA1911030010015
RA1911030010016
RA1911030010005
README.md
#define MAXLINE 1024
int main()
{
int sockfd;
char buffer[MAXLINE];
struct sockaddr_in servaddr, cliaddr;
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
{
perror("socket creation failed");
exit(EXIT_FAILURE);
}
memset(&servaddr, 0, sizeof(servaddr));
memset(&cliaddr, 0, sizeof(cliaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = INADDR_ANY;
servaddr.sin_port = htons(PORT);
if (bind(sockfd, (const struct sockaddr *)&servaddr,
sizeof(servaddr)) < 0)
{
perror("bind failed");
exit(EXIT_FAILURE);
}
else
{
printf("Server listening on Port %d\n", PORT);
}
int len, n;
len = sizeof(cliaddr);
n = recvfrom(sockfd, (char *)buffer, MAXLINE,
MSG_WAITALL, (struct sockaddr *)&cliaddr,
&len);
buffer[n] = '\0';
printf("Client Sent : %s\n", buffer);
sendto(sockfd, (const char *)buffer, strlen(buffer),
MSG_CONFIRM, (const struct sockaddr *)&cliaddr,
len);
return 0;
}
53:2 C and C++ Spaces: 4
AWS: (not connected)



Result:

The required code for UDP Echo Client-Server Communication was written in the AWS Cloud9 environment and successfully compiled.