Gita Alekhya Paul 12/08/2021

Computer Networks
RA1911030010014
Experiment - 5
Concurrent TCP/IP Day-Time Server

Aim: **To create a Concurrent TCP/IP Day-Time Server.**

Server Code:

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

#define PORT 8014
#define EXIT_FAILURE 1

int main(int argc, char *argv[])
{
   int listenfd = 0, connfd = 0;
   struct sockaddr_in serv_addr;

   char sendBuff[1024];
   time_t ticks;

   listenfd = socket(AF_INET, SOCK_STREAM, 0);
   memset(&serv_addr, '0', sizeof(serv_addr));
   memset(sendBuff, '0', sizeof(sendBuff));

   serv_addr.sin_family = AF_INET;
   serv_addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
   serv_addr.sin_port = htons(PORT);

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

   listen(listenfd, 10);

   while (1)
   {
       connfd = accept(listenfd, (struct sockaddr *)NULL, NULL);

       ticks = time(NULL);
       snprintf(sendBuff, sizeof(sendBuff),
                "The server time is: %.24s\r\n", ctime(&ticks));
       write(connfd, sendBuff, strlen(sendBuff));

       close(connfd);
       sleep(1);
   }
}
```

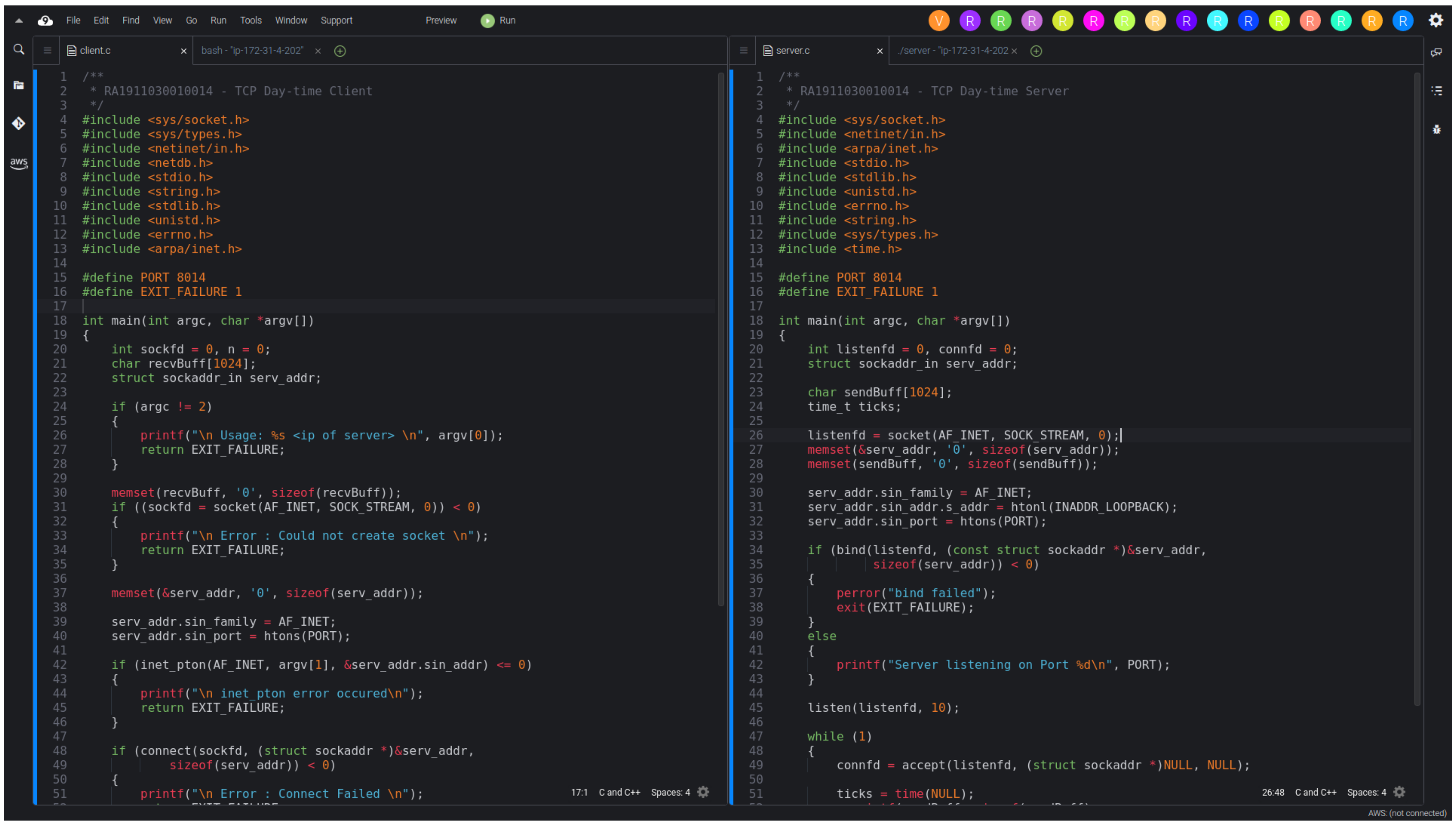
Client Code:

```
/**
* RA1911030010014 - TCP Day-time Client
*/
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

#define PORT 8014
#define EXIT_FAILURE 1

int main(int argc, char *argv[])
{
   int sockfd = 0, n = 0;
   char recvBuff[1024];
   struct sockaddr_in serv_addr;

   if (argc != 2)
   {
       printf("\n Usage: %s <ip of server> \n", argv[0]);
       return EXIT_FAILURE;
   }

   memset(recvBuff, '0', sizeof(recvBuff));
```

```
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Error : Could not create socket \n");
        return EXIT_FAILURE;
    }

    memset(&serv_addr, '0', sizeof(serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, argv[1], &serv_addr.sin_addr) <= 0)
    {
        printf("\n inet_pton error occured\n");
        return EXIT_FAILURE;
    }

    if (connect(sockfd, (struct sockaddr *)&serv_addr,
                sizeof(serv_addr)) < 0)
    {
        printf("\n Error : Connect Failed \n");
        return EXIT_FAILURE;
    }

    while ((n = read(sockfd, recvBuff, sizeof(recvBuff) - 1)) > 0)
    {
        recvBuff[n] = 0;
        if (fputs(recvBuff, stdout) == EOF)
        {
            printf("\n Error : Fputs error\n");
        }
    }

    if (n < 0)
    {
        printf("\n Read error \n");
    }

    return 0;
}
```

Gita Alekhya Paul **12/08/2021**

Output:

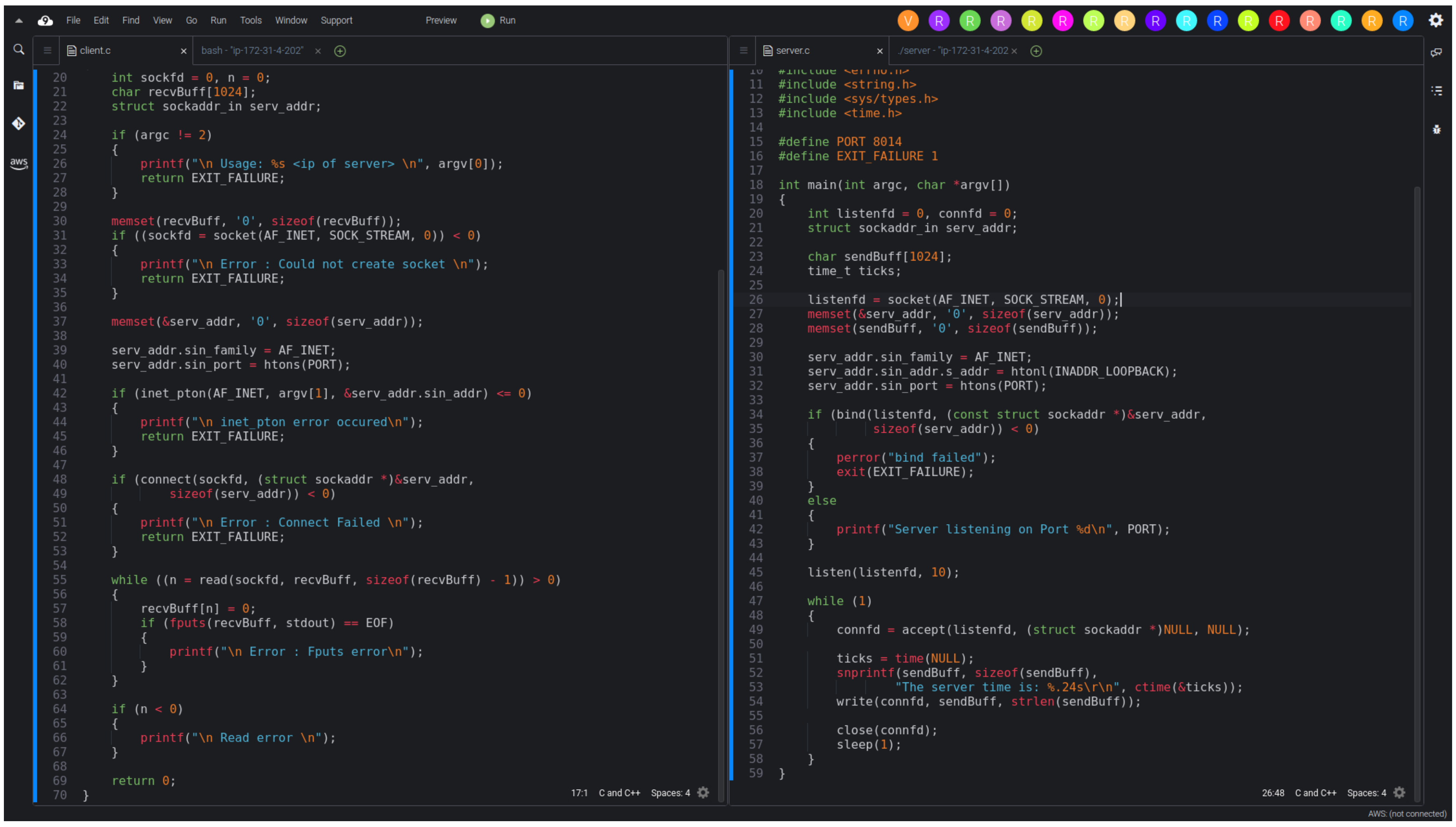


client.c

```c
/**
 * RA1911030010014 - TCP Day-time Client
 */
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

#define PORT 8014
#define EXIT_FAILURE 1

int main(int argc, char *argv[])
{
    int sockfd = 0, n = 0;
    char recvBuff[1024];
    struct sockaddr_in serv_addr;

    if (argc != 2)
    {
        printf("\n Usage: %s <ip of server> \n", argv[0]);
        return EXIT_FAILURE;
    }

    memset(recvBuff, '0', sizeof(recvBuff));
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Error : Could not create socket \n");
        return EXIT_FAILURE;
    }

    memset(&serv_addr, '0', sizeof(serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, argv[1], &serv_addr.sin_addr) <= 0)
    {
        printf("\n inet_pton error occured\n");
        return EXIT_FAILURE;
    }

    if (connect(sockfd, (struct sockaddr *)&serv_addr,
            sizeof(serv_addr)) < 0)
    {
        printf("\n Error : Connect Failed \n");
```

server.c

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

#define PORT 8014
#define EXIT_FAILURE 1

int main(int argc, char *argv[])
{
    int listenfd = 0, connfd = 0;
    struct sockaddr_in serv_addr;

    char sendBuff[1024];
    time_t ticks;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    memset(&serv_addr, '0', sizeof(serv_addr));
    memset(sendBuff, '0', sizeof(sendBuff));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
    serv_addr.sin_port = htons(PORT);

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

    listen(listenfd, 10);

    while (1)
    {
        connfd = accept(listenfd, (struct sockaddr *)NULL, NULL);

        ticks = time(NULL);
```

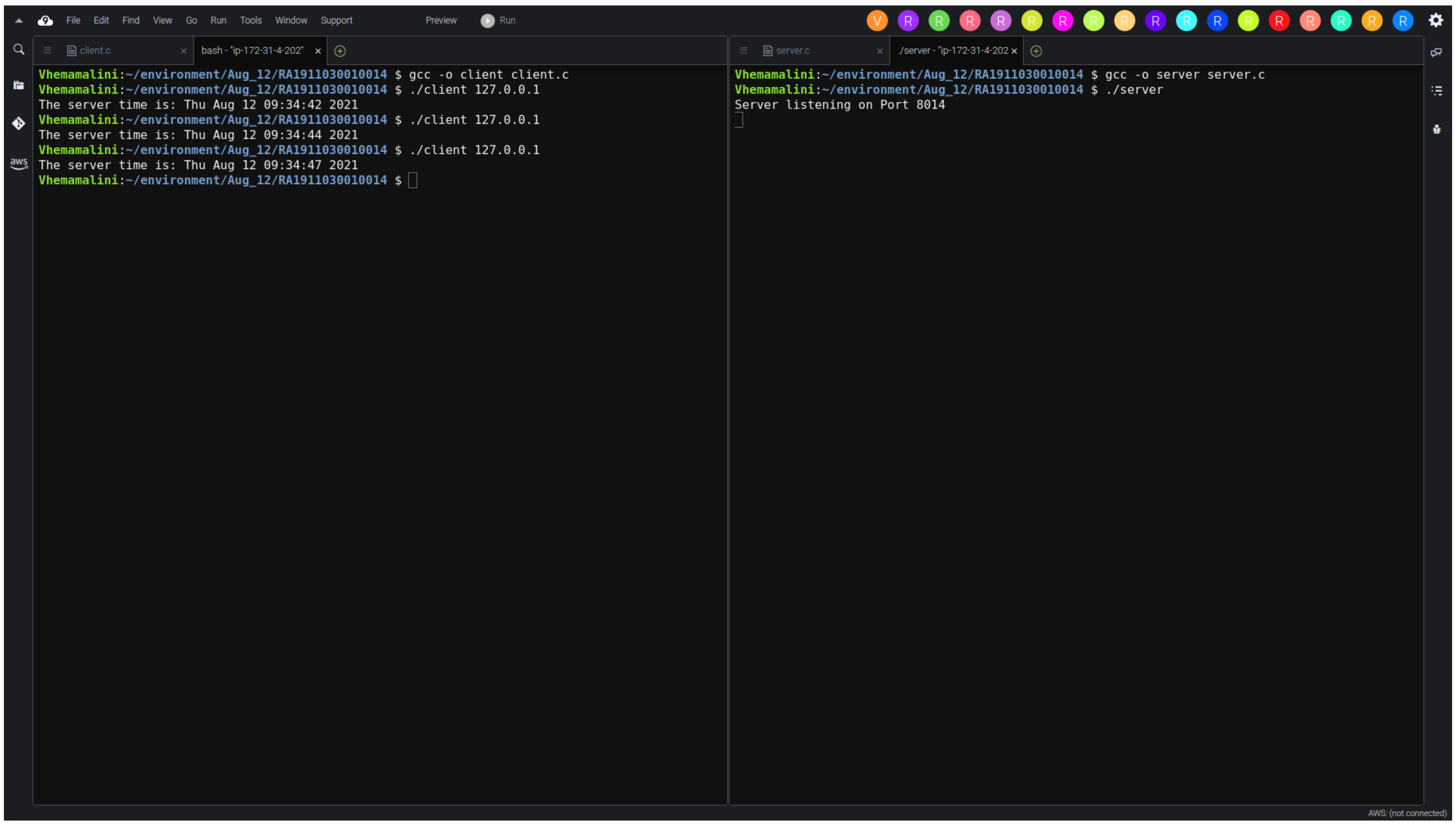


client.c

```c
    int sockfd = 0, n = 0;
    char recvBuff[1024];
    struct sockaddr_in serv_addr;

    if (argc != 2)
    {
        printf("\n Usage: %s <ip of server> \n", argv[0]);
        return EXIT_FAILURE;
    }

    memset(recvBuff, '0', sizeof(recvBuff));
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Error : Could not create socket \n");
        return EXIT_FAILURE;
    }

    memset(&serv_addr, '0', sizeof(serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, argv[1], &serv_addr.sin_addr) <= 0)
    {
        printf("\n inet_pton error occured\n");
        return EXIT_FAILURE;
    }

    if (connect(sockfd, (struct sockaddr *)&serv_addr,
            sizeof(serv_addr)) < 0)
    {
        printf("\n Error : Connect Failed \n");
        return EXIT_FAILURE;
    }

    while ((n = read(sockfd, recvBuff, sizeof(recvBuff) - 1)) > 0)
    {
        recvBuff[n] = 0;
        if (fputs(recvBuff, stdout) == EOF)
        {
            printf("\n Error : Fputs error\n");
        }
    }

    if (n < 0)
    {
        printf("\n Read error \n");
    }

    return 0;
}
```

server.c

```c
#include <string.h>
#include <sys/types.h>
#include <time.h>

#define PORT 8014
#define EXIT_FAILURE 1

int main(int argc, char *argv[])
{
    int listenfd = 0, connfd = 0;
    struct sockaddr_in serv_addr;

    char sendBuff[1024];
    time_t ticks;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    memset(&serv_addr, '0', sizeof(serv_addr));
    memset(sendBuff, '0', sizeof(sendBuff));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
    serv_addr.sin_port = htons(PORT);

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

    listen(listenfd, 10);

    while (1)
    {
        connfd = accept(listenfd, (struct sockaddr *)NULL, NULL);

        ticks = time(NULL);
        snprintf(sendBuff, sizeof(sendBuff),
                "The server time is: %.24s\r\n", ctime(&ticks));
        write(connfd, sendBuff, strlen(sendBuff));

        close(connfd);
        sleep(1);
    }
}
```



```
Vhemamalini:~/environment/Aug_12/RA1911030010014 $ gcc -o client client.c
Vhemamalini:~/environment/Aug_12/RA1911030010014 $ ./client 127.0.0.1
The server time is: Thu Aug 12 09:34:42 2021
Vhemamalini:~/environment/Aug_12/RA1911030010014 $ ./client 127.0.0.1
The server time is: Thu Aug 12 09:34:44 2021
Vhemamalini:~/environment/Aug_12/RA1911030010014 $ ./client 127.0.0.1
The server time is: Thu Aug 12 09:34:47 2021
Vhemamalini:~/environment/Aug_12/RA1911030010014 $
```

```
Vhemamalini:~/environment/Aug_12/RA1911030010014 $ gcc -o server server.c
Vhemamalini:~/environment/Aug_12/RA1911030010014 $ ./server
Server listening on Port 8014
```

Result:

The required code for the Concurrent TCP/IP Day-Time Server was written in the AWS Cloud9 environment and successfully compiled.