

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

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

Server Code:

```
/**
 * 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;
}
```

Output:

FileEditFindViewGoRunToolsWindowSupportPreviewRun

client.c x bash - "ip-172-31-4-202" x

```
1 /**
2  * RA1911030010014 - TCP Day-time Client
3  */
4  #include <sys/socket.h>
5  #include <sys/types.h>
6  #include <netinet/in.h>
7  #include <netdb.h>
8  #include <stdio.h>
9  #include <string.h>
10 #include <stdlib.h>
11 #include <unistd.h>
12 #include <errno.h>
13 #include <arpa/inet.h>
14
15 #define PORT 8014
16 #define EXIT_FAILURE 1
17
18 int main(int argc, char *argv[])
19 {
20     int sockfd = 0, n = 0;
21     char recvBuff[1024];
22     struct sockaddr_in serv_addr;
23
24     if (argc != 2)
25     {
26         printf("\n Usage: %s <ip of server> \n", argv[0]);
27         return EXIT_FAILURE;
28     }
29
30     memset(recvBuff, '0', sizeof(recvBuff));
31     if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
32     {
33         printf("\n Error : Could not create socket \n");
34         return EXIT_FAILURE;
35     }
36
37     memset(&serv_addr, '0', sizeof(serv_addr));
38
39     serv_addr.sin_family = AF_INET;
40     serv_addr.sin_port = htons(PORT);
41
42     if (inet_pton(AF_INET, argv[1], &serv_addr.sin_addr) <= 0)
43     {
44         printf("\n inet_pton error ocured\n");
45         return EXIT_FAILURE;
46     }
47
48     if (connect(sockfd, (struct sockaddr *)&serv_addr,
49                 sizeof(serv_addr)) < 0)
50     {
51         printf("\n Error : Connect Failed \n");
52     }
53 }
```

17:1 C and C++ Spaces: 4

server.c x ./server - "ip-172-31-4-202" x

```
1 /**
2  * RA1911030010014 - TCP Day-time Server
3  */
4  #include <sys/socket.h>
5  #include <netinet/in.h>
6  #include <arpa/inet.h>
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <unistd.h>
10 #include <errno.h>
11 #include <string.h>
12 #include <sys/types.h>
13 #include <time.h>
14
15 #define PORT 8014
16 #define EXIT_FAILURE 1
17
18 int main(int argc, char *argv[])
19 {
20     int listenfd = 0, connfd = 0;
21     struct sockaddr_in serv_addr;
22
23     char sendBuff[1024];
24     time_t ticks;
25
26     listenfd = socket(AF_INET, SOCK_STREAM, 0);
27     memset(&serv_addr, '0', sizeof(serv_addr));
28     memset(sendBuff, '0', sizeof(sendBuff));
29
30     serv_addr.sin_family = AF_INET;
31     serv_addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
32     serv_addr.sin_port = htons(PORT);
33
34     if (bind(listenfd, (const struct sockaddr *)&serv_addr,
35             sizeof(serv_addr)) < 0)
36     {
37         perror("bind failed");
38         exit(EXIT_FAILURE);
39     }
40     else
41     {
42         printf("Server listening on Port %d\n", PORT);
43     }
44
45     listen(listenfd, 10);
46
47     while (1)
48     {
49         connfd = accept(listenfd, (struct sockaddr *)NULL, NULL);
50
51         ticks = time(NULL);
52     }
53 }
```

26:48 C and C++ Spaces: 4

FileEditFindViewGoRunToolsWindowSupportPreviewRun

client.c x bash - "ip-172-31-4-202" x

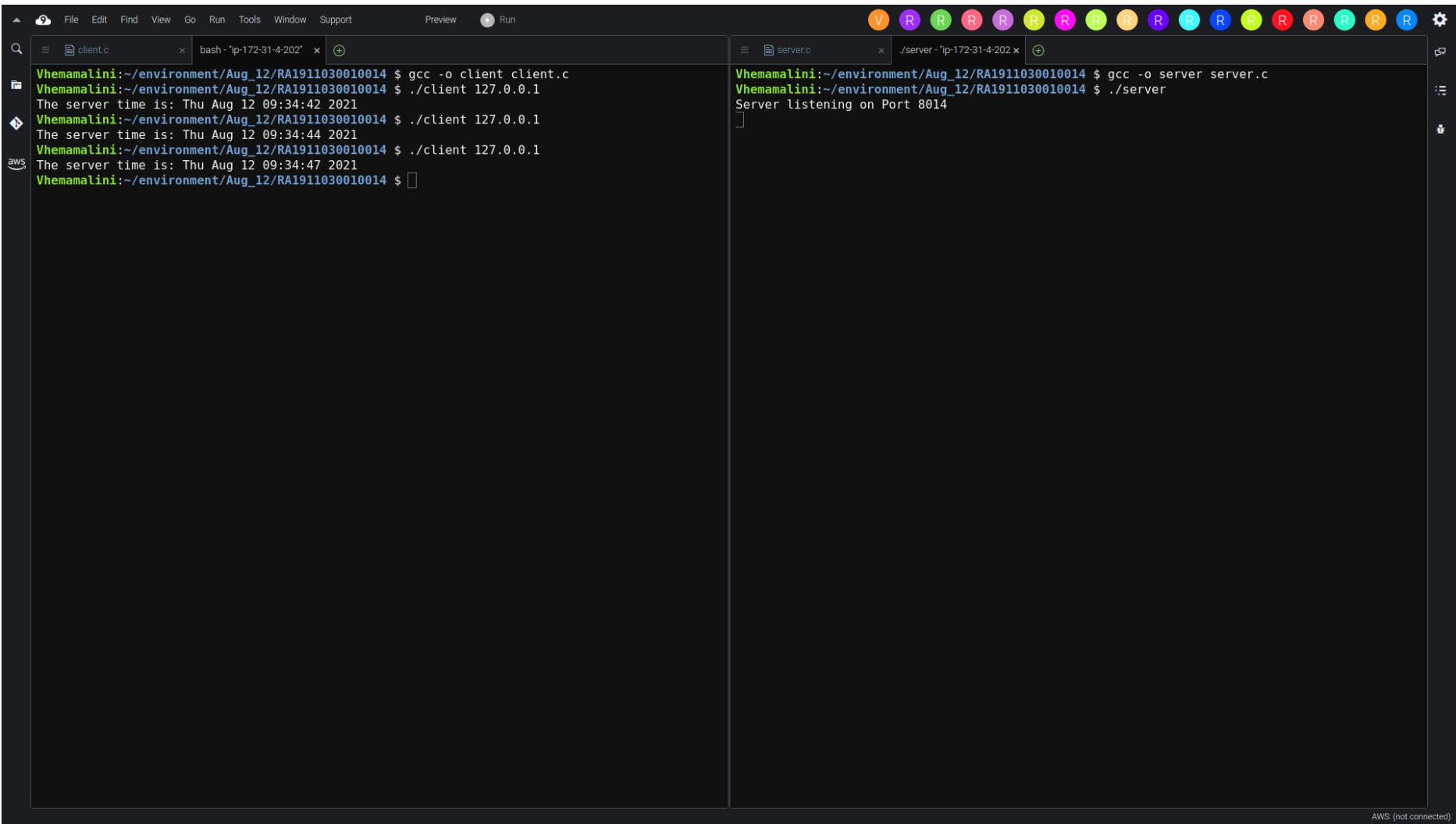
```
20     int sockfd = 0, n = 0;
21     char recvBuff[1024];
22     struct sockaddr_in serv_addr;
23
24     if (argc != 2)
25     {
26         printf("\n Usage: %s <ip of server> \n", argv[0]);
27         return EXIT_FAILURE;
28     }
29
30     memset(recvBuff, '0', sizeof(recvBuff));
31     if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
32     {
33         printf("\n Error : Could not create socket \n");
34         return EXIT_FAILURE;
35     }
36
37     memset(&serv_addr, '0', sizeof(serv_addr));
38
39     serv_addr.sin_family = AF_INET;
40     serv_addr.sin_port = htons(PORT);
41
42     if (inet_pton(AF_INET, argv[1], &serv_addr.sin_addr) <= 0)
43     {
44         printf("\n inet_pton error ocured\n");
45         return EXIT_FAILURE;
46     }
47
48     if (connect(sockfd, (struct sockaddr *)&serv_addr,
49                 sizeof(serv_addr)) < 0)
50     {
51         printf("\n Error : Connect Failed \n");
52         return EXIT_FAILURE;
53     }
54
55     while ((n = read(sockfd, recvBuff, sizeof(recvBuff) - 1)) > 0)
56     {
57         recvBuff[n] = 0;
58         if (fputs(recvBuff, stdout) == EOF)
59         {
60             printf("\n Error : Fputs error\n");
61         }
62     }
63
64     if (n < 0)
65     {
66         printf("\n Read error \n");
67     }
68
69     return 0;
70 }
```

17:1 C and C++ Spaces: 4

server.c x ./server - "ip-172-31-4-202" x

```
10 #include <stdio.h>
11 #include <string.h>
12 #include <sys/types.h>
13 #include <time.h>
14
15 #define PORT 8014
16 #define EXIT_FAILURE 1
17
18 int main(int argc, char *argv[])
19 {
20     int listenfd = 0, connfd = 0;
21     struct sockaddr_in serv_addr;
22
23     char sendBuff[1024];
24     time_t ticks;
25
26     listenfd = socket(AF_INET, SOCK_STREAM, 0);
27     memset(&serv_addr, '0', sizeof(serv_addr));
28     memset(sendBuff, '0', sizeof(sendBuff));
29
30     serv_addr.sin_family = AF_INET;
31     serv_addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
32     serv_addr.sin_port = htons(PORT);
33
34     if (bind(listenfd, (const struct sockaddr *)&serv_addr,
35             sizeof(serv_addr)) < 0)
36     {
37         perror("bind failed");
38         exit(EXIT_FAILURE);
39     }
40     else
41     {
42         printf("Server listening on Port %d\n", PORT);
43     }
44
45     listen(listenfd, 10);
46
47     while (1)
48     {
49         connfd = accept(listenfd, (struct sockaddr *)NULL, NULL);
50
51         ticks = time(NULL);
52         snprintf(sendBuff, sizeof(sendBuff),
53                 "The server time is: %.24s\r\n", ctime(&ticks));
54         write(connfd, sendBuff, strlen(sendBuff));
55
56         close(connfd);
57         sleep(1);
58     }
59 }
```

26:48 C and C++ Spaces: 4



Result:

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