

201818716

컴퓨터공학부

김용현

※ 실습 간 Putty · Xshell 대신, VScode상의 SSH 확장프로그램을 이용하여 실습 서버에 접근하였습니다.

[코드파일 스크린샷] : server.c

```
파일(F) 편집(E) 선택 영역(S) 보기(V) 이동(G) 실행(R) 터미널(T) 도움말(H) ← → root [SSH: 106.10.36.113:3240]

검색기
ROOT [SSH: 106.10.36.113...]
> .cache
> .dotnet
> .gnupg
> .local
> .nsgmt
> .ssh
> .vim
> .vscode
> .vscode-server
> test
  P_4
  P_5
  P_6
  P_7
  P_8
  P_9
  client
  client.c
  server
  server.c
  P_10
  thread.c
  thread.out
  P_X
  linux_naver.code-wo...
  .bash_history
  $ .bashrc
  $ .mysql_history
  $ .profile
  $ .selected_editor
  $ .viminfo
  $ .wget-hsts
  $ .xauthority
  개요
  타임라인

C server.c X
test > P_9 > C server.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <sys/socket.h>
6 #include <sys/types.h>
7 #include <arpa/inet.h>
8
9 #define PORT 3240
10 #define IP "127.0.0.1"
11 #define BUFSIZE 4096
12 #define TIME 1
13
14 void p(char* str) {
15     sleep(100);
16 }
17
18 int main(int argc, char* argv[]) {
19     /* Init */
20     /* Init Server */
21     int s_sock;
22     struct sockaddr_in s_addr = {AF_INET, htons(PORT), inet_addr(IP)};
23
24     /* Init Client */
25     int c_sock;
26     struct sockaddr_in c_addr;
27     socklen_t c_addr_size = sizeof(struct sockaddr);
28
29     /* Buf */
30     char buf[BUFSIZE] = {0};
31
32     /* 1_Socket */
33     p("1_Socket Start");
34
35     if((s_sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
36         perror("Error : 1_Socket");
37         exit(1);
38     }
39     int option = 1; setsockopt(s_sock, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option));
40
41     p("1_Socket End");
42
43     /* 2_Bind */
44     p("2_Bind Start");
45
46     if(bind(s_sock, (struct sockaddr*)&s_addr, sizeof(struct sockaddr)) == -1) {
47         perror("Error : 2_Bind");
48         exit(1);
49     }
50
51     p("2_Bind End");
52
53     /* 3_Listen */
54     p("3_Listen Start");
55
56     listen(s_sock, 1);
57     p("3_Listen End");
58
59     /* 4_Accept */
60     for(int i = 0; i++; i++) {
61         p("4_Accept Start(Waiting for a client...) (Exit for Ctrl+C(SIGINT))");
62
63         if((c_sock = accept(s_sock, (struct sockaddr*)&c_addr, &c_addr_size)) == -1) {
64             perror("Error : 4_Accept");
65             exit(1);
66         }
67
68         p("4_Accept End(Connected)");
69
70         /* 5_Send */
71         printf("*****[Connecting Information]*****\n");
72         printf("(SERVER)\n[INFO_SOCKET] s_sock = %d\n[CLIENT] c_sock = %d\n[Client Number] = %d\n[Client IP Address] = %s\n[Port] = %d\n",
73             s_sock, c_sock, i, inet_ntoa(c_addr.s_addr), inet_ntoa(c_addr.sin_addr), c_addr.sin_port);
74         printf("*****\n");
75
76         /* 6_Recv */
77         p("6_Recv Start");
78
79         if(send(s_sock, "Hello, I am server.", sizeof("Hello, I am server.") + 1, 0) == -1) {
80             perror("Error : 6_Send");
81             exit(1);
82         }
83
84         p("6_Send End");
85
86         /* 7_Recv */
87         printf("***** S : I said hello to client. ***\n\n");
88
89         /* Sleep */
90         sleep(3);
91
92         /* 8_Recv */
93         p("8_Recv Start");
94
95         if(recv(c_sock, buf, BUFSIZE, 0) == -1) {
96             perror("Error : 8_Recv");
97             exit(1);
98         }
99
100         p("8_Recv End");
101
102         /* 9_Debug */
103         printf("***** S : Client Says = %s ***\n\n", buf);
104
105         /* 10_CloseClient */
106         close(c_sock);
107
108         /* 11_CloseServer */
109         close(s_sock);
110
111         /* Return */
112         return 0;
113     }
114 }
```

```
파일(F) 편집(E) 선택 영역(S) 보기(V) 이동(G) 실행(R) 터미널(T) 도움말(H) ← → root [SSH: 106.10.36.113:3240]

검색기
ROOT [SSH: 106.10.36.113...]
> .cache
> .dotnet
> .gnupg
> .local
> .nsgmt
> .ssh
> .vim
> .vscode
> .vscode-server
> test
  P_4
  P_5
  P_6
  P_7
  P_8
  P_9
  client
  client.c
  server
  server.c
  P_10
  thread.c
  thread.out
  P_X
  linux_naver.code-wo...
  .bash_history
  $ .bashrc
  $ .mysql_history
  $ .profile
  $ .selected_editor
  $ .viminfo
  $ .wget-hsts
  $ .xauthority
  개요
  타임라인

C server.c X
test > P_9 > C server.c > main(int, char* D)
59 p("3_Listen Start");
60 listen(s_sock, 1);
61 p("3_Listen End");
62
63 /* 4_Accept */
64 for(int i = 0; i++; i++) {
65     p("4_Accept Start(Waiting for a client...) (Exit for Ctrl+C(SIGINT))");
66
67     if((c_sock = accept(s_sock, (struct sockaddr*)&c_addr, &c_addr_size)) == -1) {
68         perror("Error : 4_Accept");
69         exit(1);
70     }
71
72     p("4_Accept End(Connected)");
73
74     /* 5_Send */
75     printf("*****[Connecting Information]*****\n");
76     printf("(SERVER)\n[INFO_SOCKET] s_sock = %d\n[CLIENT] c_sock = %d\n[Client Number] = %d\n[Client IP Address] = %s\n[Port] = %d\n",
77         s_sock, c_sock, i, inet_ntoa(c_addr.s_addr), inet_ntoa(c_addr.sin_addr), c_addr.sin_port);
78     printf("*****\n");
79
80     /* 6_Recv */
81     p("6_Recv Start");
82
83     if(send(s_sock, "Hello, I am server.", sizeof("Hello, I am server.") + 1, 0) == -1) {
84         perror("Error : 6_Send");
85         exit(1);
86     }
87
88     p("6_Send End");
89
90     /* 7_Recv */
91     printf("***** S : I said hello to client. ***\n\n");
92
93     /* Sleep */
94     sleep(3);
95
96     /* 8_Recv */
97     p("8_Recv Start");
98
99     if(recv(c_sock, buf, BUFSIZE, 0) == -1) {
100         perror("Error : 8_Recv");
101         exit(1);
102     }
103
104     p("8_Recv End");
105
106     /* 9_Debug */
107     printf("***** S : Client Says = %s ***\n\n", buf);
108
109     /* 10_CloseClient */
110     close(c_sock);
111
112     /* 11_CloseServer */
113     close(s_sock);
114
115     /* Return */
116     return 0;
117 }
```

[코드파일 스크린샷] : client.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 #include <sys/socket.h>
7 #include <netinet/in.h>
8 #include <arpa/inet.h>
9
10 #define PORT 3240
11 #define IP "127.0.0.1"
12 #define BUFSIZE 4096
13 #define TIME 1
14
15 void p(char* str, char* name) {
16     sleep(1);
17     printf("%s: %s\n", name, str);
18 }
19
20 int main(int argc, char* argv[]) {
21     /* Error */
22     if(argc < 2) {
23         printf("Usage : ./client.out name\n");
24         return 1;
25     }
26
27     /* Init */
28     /* Init Server */
29     struct sockaddr_in s_addr = {AF_INET, htons(PORT), inet_addr(IP)};
30
31     /* Init Client */
32     int c_sock;
33     struct sockaddr_in c_addr;
34     socklen_t c_addr_size = sizeof(struct sockaddr);
35
36     /* Buf */
37     char buf[BUFSIZE] = {0};
38
39     /* 1_Socket */
40     if(1_Socket) {
41         /* 1_Socket */
42         p("1_Socket Start", argv[1]);
43
44         if((c_sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
45             perror("Error : 1_Socket");
46             exit(1);
47         }
48
49         p("1_Socket End", argv[1]);
50
51         /* 2_Connect */
52         p("2_Connect Start(Connecting...)", argv[1]);
53
54         if(connect(c_sock, (struct sockaddr*)&s_addr, sizeof(struct sockaddr)) == -1) {
55             perror("Error : 2_Connect");
56             exit(1);
57         }
58
59         p("2_Connect End(Connected!!!)", argv[1]);
60
61         /* Debug */
62         printf("\n*****[Connecting Information]*****\n(csock=%d)(s_sock=%d)\n", argv[1], c_sock);
63
64         /* 3_Recv */
65         p("3_Recv Start", argv[1]);
66
67         /* 3_Recv */
68         if(recv(c_sock, buf, BUFSIZE, 0) == -1) {
69             perror("Error : 3_Recv");
70             exit(1);
71         }
72
73         p("3_Recv End", argv[1]);
74
75         /* Debug */
76         printf("\n*** c_sock : Server Says = %s ***\n", argv[1], buf);
77
78         /* Sleep */
79         sleep(3);
80
81         /* 4_Send */
82         p("4_Send Start", argv[1]);
83
84         char str1[30] = "Hello, I am client ";
85         char str2 = argv[1];
86         strcat(str1, str2);
87
88         if(send(c_sock, str1, sizeof(str1) + 1, 0) == -1) {
89             perror("Error : 4_Send");
90             exit(1);
91         }
92
93         p("4_Send End", argv[1]);
94
95         /* Debug */
96         printf("\n*** c_sock : I said hello to server. ***\n", argv[1]);
97
98         /* 5_Close */
99         close(c_sock);
100
101         /* Return */
102         return 0;
103     }
```

```
40 p("1_Socket End", argv[1]);
41
42 /* 2_Connect */
43 p("2_Connect Start(Connecting...)", argv[1]);
44
45 if(connect(c_sock, (struct sockaddr*)&s_addr, sizeof(struct sockaddr)) == -1) {
46     perror("Error : 2_Connect");
47     exit(1);
48 }
49
50 p("2_Connect End(Connected!!!)", argv[1]);
51
52 /* Debug */
53 printf("\n*****[Connecting Information]*****\n(csock=%d)(s_sock=%d)\n", argv[1], c_sock);
54
55 /* 3_Recv */
56 p("3_Recv Start", argv[1]);
57
58 if(recv(c_sock, buf, BUFSIZE, 0) == -1) {
59     perror("Error : 3_Recv");
60     exit(1);
61 }
62
63 p("3_Recv End", argv[1]);
64
65 /* Debug */
66 printf("\n*** c_sock : Server Says = %s ***\n", argv[1], buf);
67
68 /* Sleep */
69 sleep(3);
70
71 /* 4_Send */
72 p("4_Send Start", argv[1]);
73
74 char str1[30] = "Hello, I am client ";
75 char str2 = argv[1];
76 strcat(str1, str2);
77
78 if(send(c_sock, str1, sizeof(str1) + 1, 0) == -1) {
79     perror("Error : 4_Send");
80     exit(1);
81 }
82
83 p("4_Send End", argv[1]);
84
85 /* Debug */
86 printf("\n*** c_sock : I said hello to server. ***\n", argv[1]);
87
88 /* 5_Close */
89 close(c_sock);
90
91 /* Return */
92 return 0;
93 }
```

[실행 스크린샷]

```
root@lumin:~/test/P_9# ./server
S : 1_Socket Start
S : 1_Socket End
S : 2_Bind Start
S : 2_Bind End
S : 3_Listen Start
S : 3_Listen End
S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl+C(SIGINT))
S : 4_Accept End(Connected!)

*****[Connecting Information]*****
<SERVER>
[INFO_SOCKET]
s_sock = 3
c_sock = 4
[INFO_CLIENT]
Client Number = 0
Client IP Address = 127.0.0.1
Port = 39118
*****

S : 5_Send Start
S : 5_Send End

*** S : I said hello to client. ***

S : 6_Recv Start
S : 6_Recv End

*** S : Client Says = Hello, I am client Alfa ***

S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl+C(SIGINT))
S : 4_Accept End(Connected!)

*****[Connecting Information]*****
<SERVER>
[INFO_SOCKET]
s_sock = 3
c_sock = 4
[INFO_CLIENT]
Client Number = 1
Client IP Address = 127.0.0.1
Port = 39122
*****

S : 5_Send Start
S : 5_Send End

*** S : I said hello to client. ***

S : 6_Recv Start
S : 6_Recv End

*** S : Client Says = Hello, I am client Bravo ***

root@lumin:~/test/P_9# ./client Alfa
C_Alfa : 1_Socket Start
C_Alfa : 1_Socket End
C_Alfa : 2_Connect Start(Connecting...)
C_Alfa : 2_Connect End(Connected!!!!)

*****[Connecting Information]*****
<CLIENT Alfa>
c_sock = 3
*****

C_Alfa : 3_Recv Start
C_Alfa : 3_Recv End

*** C_Alfa : Server Says = Hello, I am server. ***

C_Alfa : 4_Send Start
C_Alfa : 4_Send End

*** C_Alfa : I said hello to server. ***

root@lumin:~/test/P_9# ./client Alfa & ./client Bravo
[1] 993
C_Bravo : 1_Socket Start
C_Alfa : 1_Socket Start
C_Alfa : 1_Socket End
C_Bravo : 2_Connect Start(Connecting...)
C_Alfa : 2_Connect Start(Connecting...)
C_Bravo : 2_Connect End(Connected!!!!)

*****[Connecting Information]*****
<CLIENT Bravo>
c_sock = 3
*****

C_Bravo : 3_Recv Start
C_Alfa : 3_Recv Start
C_Bravo : 3_Recv End

*** C_Bravo : Server Says = Hello, I am server. ***

C_Bravo : 4_Send Start
C_Bravo : 4_Send End

*** C_Bravo : I said hello to server. ***

root@lumin:~/test/P_9# C_Alfa : 3_Recv End
```

```
*** S : Client Says = Hello, I am client Bravo ***

S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl+C(SIGINT))
S : 4_Accept End(Connected!)

*****[Connecting Information]*****
<SERVER>
[INFO_SOCKET]
s_sock = 3
c_sock = 4
[INFO_CLIENT]
Client Number = 2
Client IP Address = 127.0.0.1
Port = 39124
*****

S : 5_Send Start
S : 5_Send End

*** S : I said hello to client. ***

S : 6_Recv Start
S : 6_Recv End

*** S : Client Says = Hello, I am client Alfa ***

S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl+C(SIGINT))
S : 4_Accept End(Connected!)

*****[Connecting Information]*****
<SERVER>
[INFO_SOCKET]
s_sock = 3
c_sock = 4
[INFO_CLIENT]
Client Number = 3
Client IP Address = 127.0.0.1
Port = 39126
*****

S : 5_Send Start
S : 5_Send End

*** S : I said hello to client. ***

S : 6_Recv Start
S : 6_Recv End

*** S : Client Says = Hello, I am client Bravo ***

S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl+C(SIGINT))
S : 4_Accept End(Connected!)

*****[Connecting Information]*****
<SERVER>

[1]+ Done
./client Alfa
root@lumin:~/test/P_9# ./client Alfa & ./client Bravo & ./client Charlie
[1] 1150
[2] 1151
C_Bravo : 1_Socket Start
C_Charlie : 1_Socket Start
C_Alfa : 1_Socket Start
C_Bravo : 1_Socket End
C_Charlie : 1_Socket End
C_Alfa : 1_Socket End
C_Charlie : 2_Connect Start(Connecting...)
C_Bravo : 2_Connect Start(Connecting...)
C_Alfa : 2_Connect Start(Connecting...)
C_Bravo : 2_Connect End(Connected!!!!)
C_Charlie : 2_Connect End(Connected!!!!)

*****[Connecting Information]*****
<CLIENT Charlie>
c_sock = 3
*****

*****[Connecting Information]*****
<CLIENT Bravo>
c_sock = 3
*****

C_Alfa : 2_Connect End(Connected!!!!)

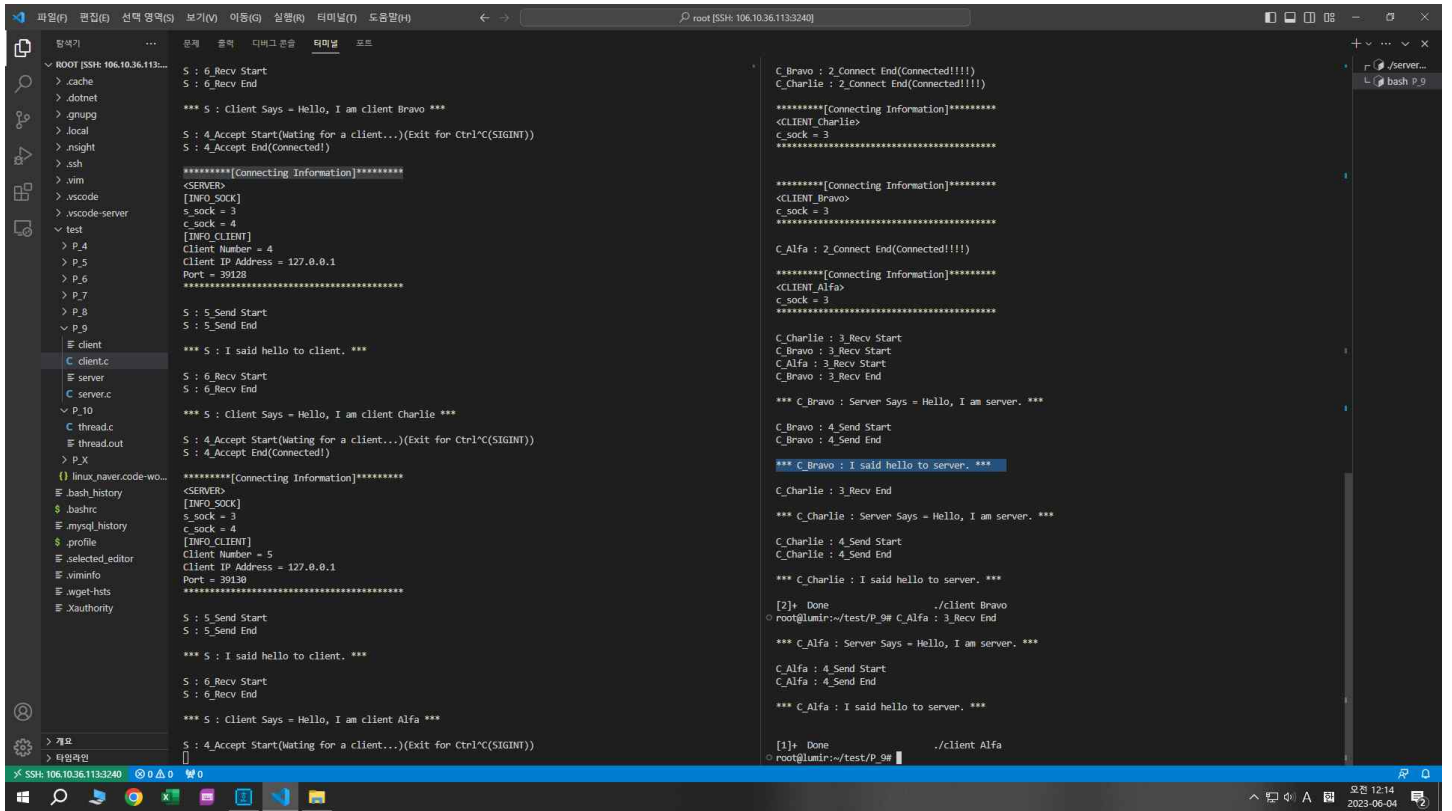
*****[Connecting Information]*****
<CLIENT Alfa>
c_sock = 3
*****

C_Charlie : 3_Recv Start
C_Bravo : 3_Recv Start
C_Alfa : 3_Recv Start
C_Bravo : 3_Recv End

*** C_Bravo : Server Says = Hello, I am server. ***

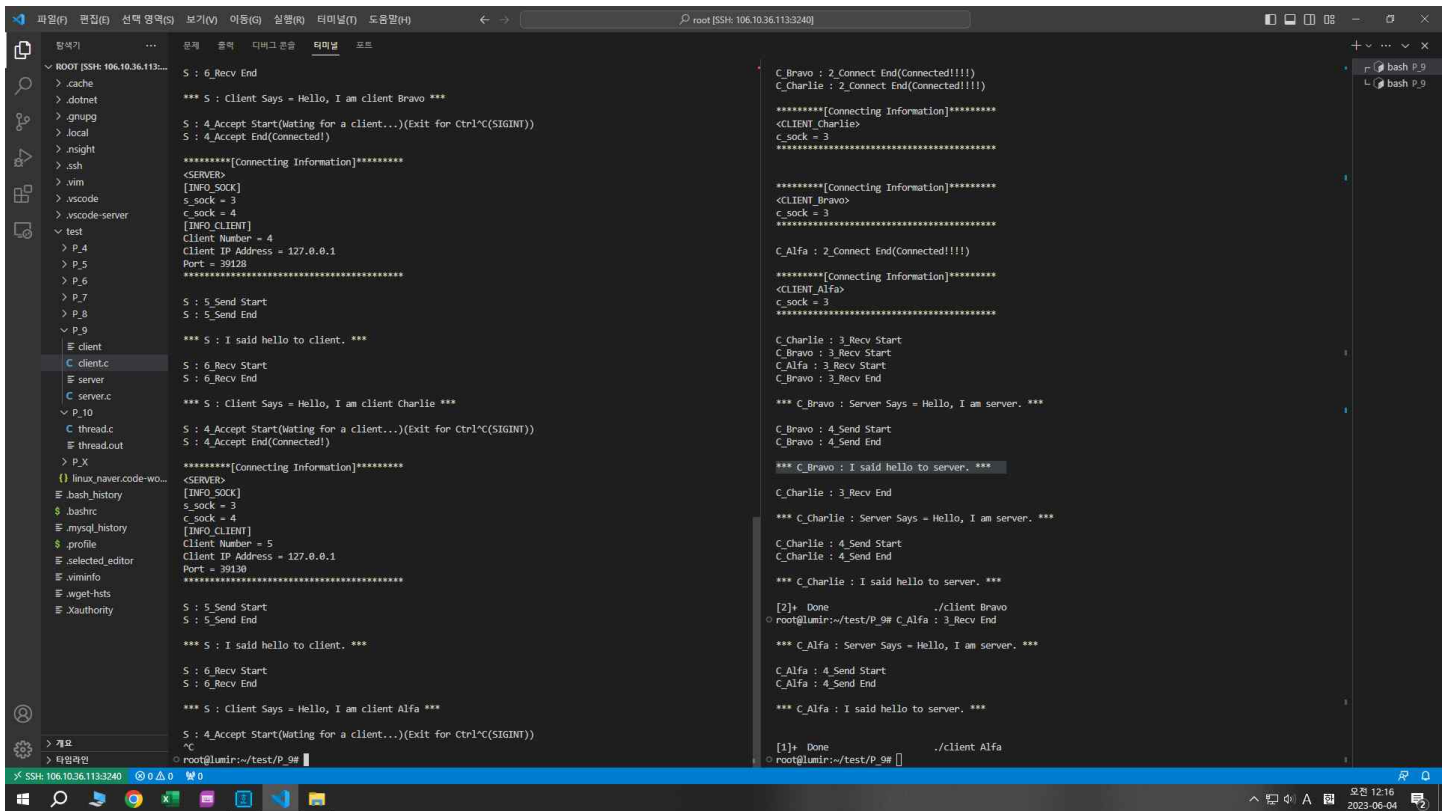
C_Bravo : 4_Send Start
C_Bravo : 4_Send End

*** C_Bravo : I said hello to server. ***
```



```
root@SSH: 106.10.36.113:3240
S : 6_Recv Start
S : 6_Recv End
*** S : Client Says = Hello, I am client Bravo ***
S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl^C(SIGINT))
S : 4_Accept End(Connected!)
*****[Connecting Information]*****
<SERVER>
[INFO_SOCKET]
s_sock = 3
c_sock = 4
[INFO_CLIENT]
Client Number = 4
Client IP Address = 127.0.0.1
Port = 39128
*****
S : 5_Send Start
S : 5_Send End
*** S : I said hello to client. ***
S : 6_Recv Start
S : 6_Recv End
*** S : Client Says = Hello, I am client Charlie ***
S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl^C(SIGINT))
S : 4_Accept End(Connected!)
*****[Connecting Information]*****
<SERVER>
[INFO_SOCKET]
s_sock = 3
c_sock = 4
[INFO_CLIENT]
Client Number = 5
Client IP Address = 127.0.0.1
Port = 39128
*****
S : 5_Send Start
S : 5_Send End
*** S : I said hello to client. ***
S : 6_Recv Start
S : 6_Recv End
*** S : Client Says = Hello, I am client Alfa ***
S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl^C(SIGINT))
^C
root@luminir:~/test/P_9#
```

(server를 SIGINT로 종료시키는 스크린샷) : SIGINT를 통해 종료 가능.



```
root@SSH: 106.10.36.113:3240
S : 6_Recv End
*** S : Client Says = Hello, I am client Bravo ***
S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl^C(SIGINT))
S : 4_Accept End(Connected!)
*****[Connecting Information]*****
<SERVER>
[INFO_SOCKET]
s_sock = 3
c_sock = 4
[INFO_CLIENT]
Client Number = 4
Client IP Address = 127.0.0.1
Port = 39128
*****
S : 5_Send Start
S : 5_Send End
*** S : I said hello to client. ***
S : 6_Recv Start
S : 6_Recv End
*** S : Client Says = Hello, I am client Charlie ***
S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl^C(SIGINT))
S : 4_Accept End(Connected!)
*****[Connecting Information]*****
<SERVER>
[INFO_SOCKET]
s_sock = 3
c_sock = 4
[INFO_CLIENT]
Client Number = 5
Client IP Address = 127.0.0.1
Port = 39128
*****
S : 5_Send Start
S : 5_Send End
*** S : I said hello to client. ***
S : 6_Recv Start
S : 6_Recv End
*** S : Client Says = Hello, I am client Alfa ***
S : 4_Accept Start(Waiting for a client...)(Exit for Ctrl^C(SIGINT))
^C
root@luminir:~/test/P_9#
```

[관찰되는 동작 결과를 바탕으로 설명 작성]

〈서버 실행파일을 먼저 실행했을 때 확인할 수 있는 결과〉

- #. 서버 : Socket · Bind · Listen은 서버의 동작 중 단 1번씩만 수행하면 된다. Socket을 통해 소켓을 생성하며, Bind를 통해 소켓의 이름을 지정해준다. 이후, Listen을 통해 연결을 대기한다. 이때, Server는 Client의 Connect 요청이 있기 전까지 Accept 라인에서 무한 대기한다. (이때, SIGINT등의 시그널이 발생하면 프로그램이 종료되지만, 이외의 경우에는 Client의 Connect 요청이 올 때까지 Accept 함수를 적어둔 해당 라인에서 무한 대기하게 된다.)

〈클라이언트 프로그램 1개를 실행했을 때 확인할 수 있는 결과〉

- #. 서버 : Client로 부터 Connect 요청이 들어왔으므로, 해당 연결을 Accept하고 Send 과정을 수행한다. 이때, Client는 Recv 상태이므로, Server가 보내는 Send를 정상적으로 받을 수 있다. 이후, Server는 Recv 라인에서 Client가 보내는 Send가 올 때까지 무한 대기하게 된다. Client가 Recv 이후 3초간 Sleep을 진행한 뒤 Server에게 Send 신호를 보내면, Server는 해당 신호를 Recv하고, 전달된 값을 출력한 뒤, Close를 통해 Client와 연결된 소켓을 소멸시킨다.
- #. 클라이언트 : Socket을 통해 소켓을 생성한다. 이후, Connect를 통해 Server에 연결 요청을 진행하는데, 이때 Server의 상태가 '4_Accept Start(Waiting for a client...)'로서 현재 Accept 대기중인 상태이다. 따라서, Connect를 통해 Server와 정상 연결이 수립된다. 그 다음, Client는 Recv 라인에서 Server가 보내는 Send가 올 때 까지 무한 대기한다. Server가 Send를 보내면, 해당 값을 Recv하고 출력한 뒤, Send 과정으로 넘어간다. 3초간 Sleep한 후, Server에게 Send를 보내는데, 이때, Server가 Recv 상태이므로 해당값이 정상 전달된다. 이후 Client는 소켓을 Close하고, 프로세스를 정상 종료하게 된다.

〈클라이언트 프로그램 2개를 실행했을 때 확인할 수 있는 결과〉 & 〈클라이언트 프로그램 3개를 실행했을 때 확인할 수 있는 결과〉

- #. 클라이언트 : 클라이언트 각각의 기본 동작은 〈클라이언트 프로그램 1개를 실행했을 때 확인할 수 있는 결과〉와 동일하다. **2개의 클라이언트가 동시 실행되었을 때**를 우선 살펴보자. 2개의 Process가 동시에 실행되므로, Socket의 생성과정은 Alfa와 Bravo 둘 다 거의 동시에 진행된다. (이때, 일반적으로 스택구조로 인하여, Bravo가 먼저 수행되므로, CPU 스케줄링 상 Socket의 생성 과정을 Bravo 먼저 진행하는 경우가 많다. 하지만, 중간에 다른 이유로 인해, CPU 스케줄링이 뒤틀려 Alfa가 먼저 수행되는 경우도 더러 존재한다. 이는 Socket 뿐만 아닌, Connect 과정에도 해당한다.) 이후, Alfa와 Bravo 둘 다 Server에 Connect 요청을 진행한다. 2개의 프로세스 모두 Connect가 수립되는데, 이는 포트가 열려있으므로 커널이 이를 거부하지 않고 수락했기 때문이다. 이때, Connect를 먼저 요청하고 수립된 쪽은 Bravo 이므로, Server는 Bravo에 대한 요청건을 먼저 처리한다. 이에 Server는 Bravo에 대한 Send와 Recv를 처리한다. (Server는 한번에 하나씩의 요청만 처리할 수 있다..) Bravo에 대한 처리 과정 중, Alfa의 경우 Recv에서 대기한다. 이는, Server가 아직 Bravo에 대한 요청건을 처리하고 있어 Alfa가 Send를 받지 못하므로, Alfa는 Send에 해당하는 라인에서 Send가 올 때 까지 무한 대기하게 되는 것이다. 이후, Bravo에 대한 처리가 모두 끝나고 나면 Alfa에 대해 위와 똑같은 과정을 수행해준다. 이후, Alfa까지 종료되게 되면, Server는 다시 Accept 과정에서 무한 대기하게 된다. **3개의 클라이언트가 동시 실행되었을 때**를 살펴보자. Client의 수행과정은 '2개의 클라이언트가 동시 실행되었을 때'와 완전히 동일하다. 이때, 눈여겨 볼만한 부분은 Connect 요청을 누가 먼저 진행하는가 하는 부분이다. 위 실행 예시를 살펴보면, Bravo가 가장먼저 Connect 요청에 성공하고, 그 다음 Charlie, 그 다음 Alfa 순으로 진행된다. (Charlie, Bravo, Alfa순으로 Connect 요청이 진행되지 않은 이유는 위쪽 괄호에서 언급하였다.) 따라서, Server는 Bravo, Charlie, Alpha 순으로 요청건을 처리하고 다시 Accept 과정에서 무한 대기하게 된다.
- #. 서버 : Server에서 한번 Accept가 수행되면, 이후의 동작은 〈클라이언트 프로그램 1개를 실행했을 때 확인할 수 있는 결과〉와 완전히 동일하다. 그렇다면, Accept이 되기까지의 과정만 살펴보자. Server는 앞 페이지 실행 예시에서 살펴볼 수 있듯이, 한번에 하나의 Client의 요청만을 처리할 수 있다. 그렇다면, Server가 Client에서의 Connect 요청을 어떤 순서에 따라 Accept 하는지 살펴보자. 클라이언트에 대한 설명에서 언급했듯이, 먼저 Connect 요청을 한 Client 먼저 선착순과 비슷한 방식으로 처리해준다. (먼저 연결된 Client 순서대로 차례로 처리해준다는 뜻이다.)

〈기타 등등〉

- #. Server 프로그램은 SIGINT가 발생하면 정상 종료된다.