

네트워크프로그래밍-6주 채팅1

정인환교수

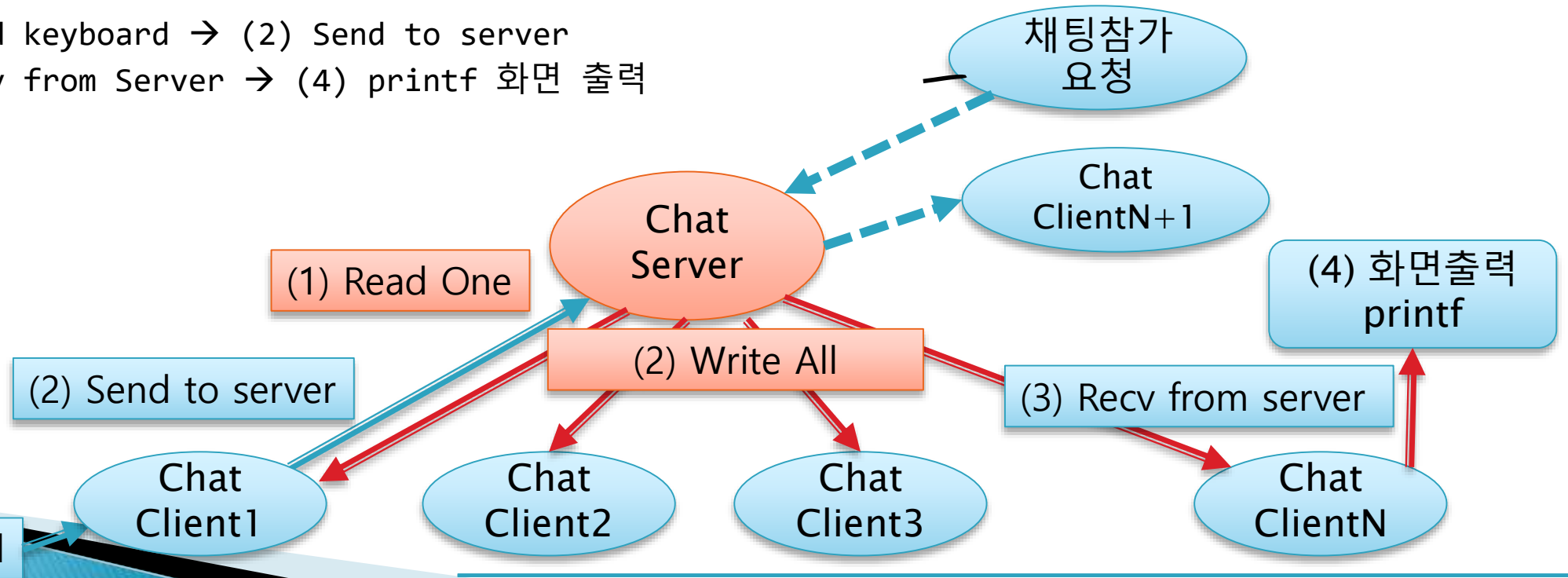
6주 : 채팅 Client/Server

▶ 채팅 Server

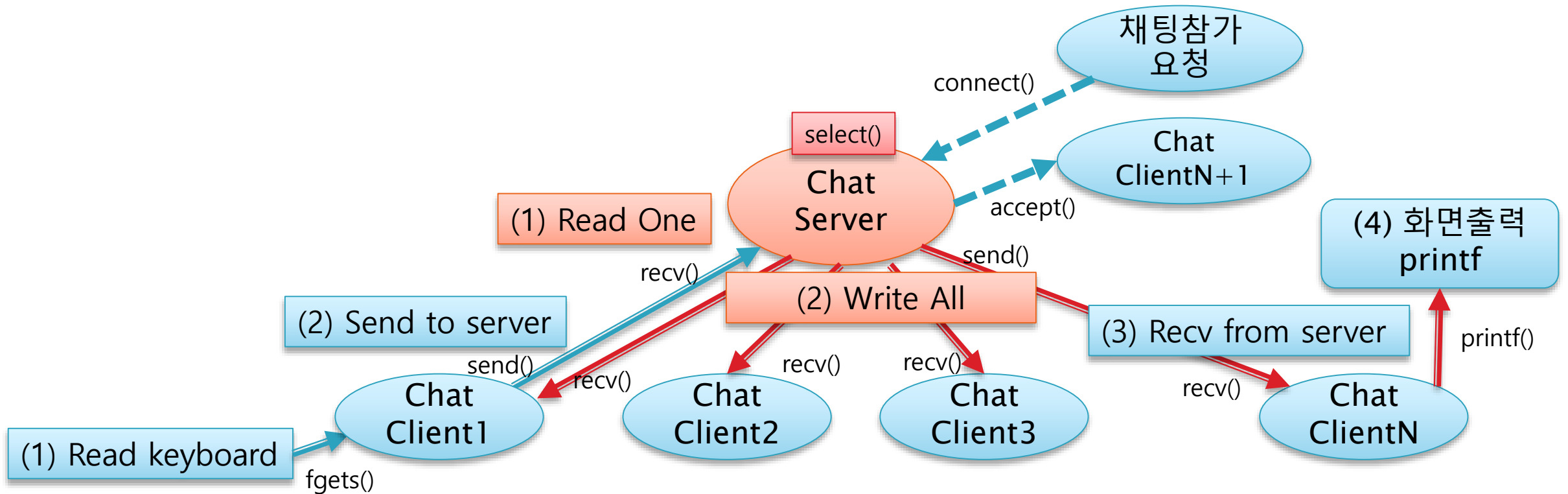
- 채팅 서버는 임의의 클라이언트로부터의 채팅 참가요구를 처리하면서 동시에 어떤 클라이언트가 보내온 채팅 메시지를 모든 클라이언트에게 방송하는 일을 처리한다.
- 채팅 참가 요구 처리
- 채팅 Message 처리
- (1) Read one / (2) Write All

▶ 채팅 Client

- (1) Read keyboard → (2) Send to server
- (3) Recv from Server → (4) printf 화면 출력

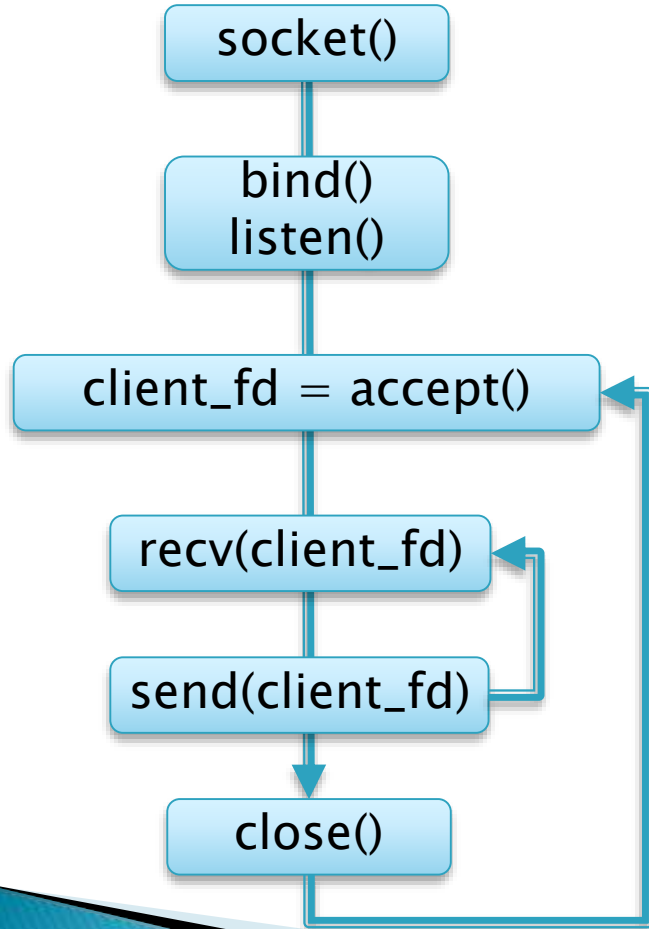


Chat client/server 의 socket 함수 사용

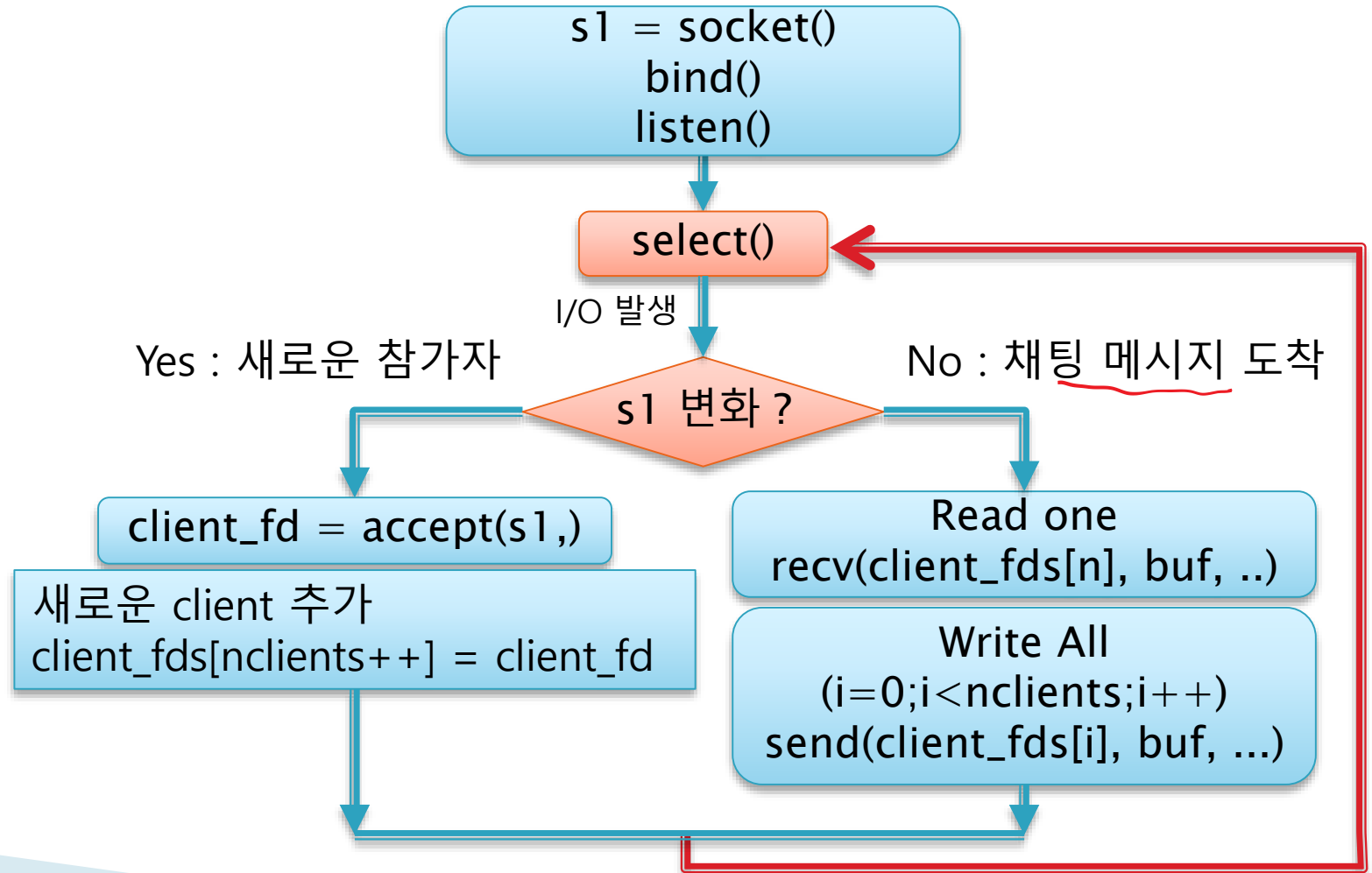


Chat Server 흐름

- ▶ echo_server
- ▶ One by One 방식 : n:n 채팅 불가능

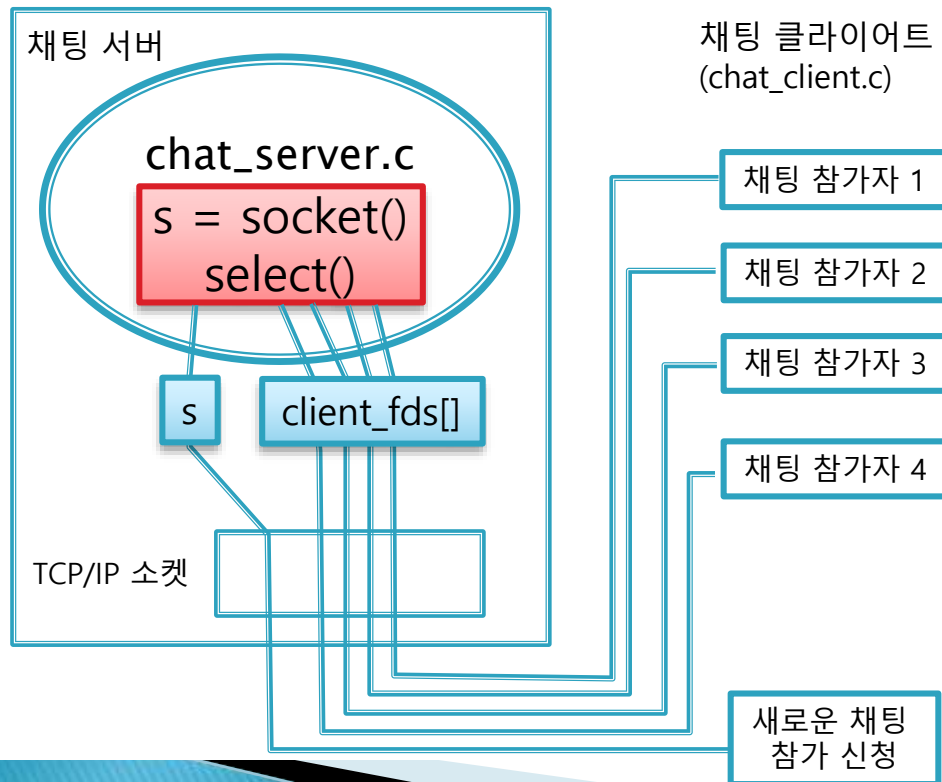


- ▶ Concurrent 방식 필요
- ▶ select() 함수 사용 : socket I/O 검사



Chat server select 함수 동작 구조

- ▶ select() 함수 이용
 - 여러 개의 socket 에 변화가 있는지 감지하는 기능 이용
- ▶ socket 의 구분
 - `s = socket() → accept(s, ..)` 로 새로운 참가자 접속용
 - `client_fd = accept(s, ..);`
 - `client_fds[num_chat++] = client_fd → client 당 1개씩 I/O 용 배열에 보관 → recv(client_fds[i], ...) 사용`



Chat Server 의 핵심 부분

- Read One / Write All
- Select 대상
 - 초기 소켓 : `s (server_fd) - accept() 용`
 - 채팅 Client들 : `client_fds[] - recv() 용`

select 함수

- ▶ `int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout)`
 - `n` : 검사 대상이 되는 file descriptor의 수 (file descriptor 최대값 + 1)
 - **`readfds` : 입력 스트림에 변화가 발생하였는지 확인(connect 요청이 있거나 수신할 데이터가 존재 하는가)**
 - `writefds` : 데이터 전송 시 blocking 되지 않고 바로 전송이 가능한가?
 - `exceptfds` : socket에 예외가 발생하였는지 확인
 - `timeout` : select함수 호출 후 무한 대기상태에 빠지지 않게 시간 설정
- ▶ `fd_set` 구조체
 - `typedef struct fd_set {
 u_int fd_count;
 SOCKET fd_array[FD_SETSIZE]`
 - `} fd_set;`
 - `fd_count` : 설정하는 socket 번호 최대값.
 `fd_array` : 설정된 socket 배열
- ▶ 함수들
 - `FD_ZERO(fd_set* fdset)`
 - `fdset` 포인터가 가리키는 변수의 모든 비트들을 0으로 초기화
 - `FD_SET(int fd, fd_set* fdset)`
 - `fdset` 포인터가 가리키는 변수에 `fd`로 전달되는 file descriptor의 정보 설정
 - `FD_CLR(int fd, fd_set* fdset)`
 - `fdset` 포인터가 가리키는 변수에서 `fd`로 전달되는 file descriptor 정보 삭제
 - `FD_ISSET(int fd, fd_set* fdset)`
 - `fdset` 포인터가 가리키는 변수가 `fd`로 전달되는 file descriptor 정보를 지니고 있는지 확인

Chat server select 예 – select()대기, accept() 처리

```
listen(server_fd, 5);

while (1) {
    FD_ZERO(&read_fds); // 변수 초기화
    FD_SET(server_fd, &read_fds); // accept() 대상 소켓 설정
    for (i = 0; i < num_chat; i++) // 채팅에 참가중인 모든 client 소켓을 read() 대상 추가
        FD_SET(client_fds[i], &read_fds);
    maxfdp = getmax(server_fd) + 1; // 감시대상 소켓의 수를 계산
    if (select(maxfdp, &read_fds, (fd_set*)0, (fd_set*)0, (struct timeval*)0) <= 0) {
        printf("select error <= 0 \n");
        exit(0);
    }
    // 초기 소켓 즉, server_fd 에 변화가 있는지 검사
    if (FD_ISSET(server_fd, &read_fds)) {
        // 변화가 있다 --> client 가 connect로 연결 요청을 한 것
        client_len = sizeof(client_addr);
        client_fd = accept(server_fd, (struct sockaddr*)&client_addr, &client_len);
        if (client_fd == -1) {
            printf("accept error\n");
        }
        else {
            printf("Client connected from %s:%d\n", inet_ntoa(client_addr.sin_addr),
                    ntohs(client_addr.sin_port));

            printf("client_fd = %d\n", client_fd);
            /* 채팅 클라이언트 목록에 추가 */
            client_fds[num_chat++] = client_fd;
            printf("%d번째 client 추가.\n", num_chat);
        }
    }
}
```


Chat server select 예 - 채팅 메시지 처리

```
memset(client_error, 0, sizeof(client_error));
/* 클라이언트가 보낸 메시지를 모든 클라이언트에게 방송 */
for (i = 0; i < num_chat; i++) {
    // 각각의 client들의 I/O 변화가 있는지.
    if (FD_ISSET(client_fds[i], &read_fds)) {
        // Read One 또는 client 비정상 종료 확인
        if ((n = recv(client_fds[i], buf, BUF_LEN, 0)) <= 0) {
            // client 가 비 정상 종료한 경우
            printf("recv error for client[%d]\n", i);
            client_error[i] = 1;
            continue;
        }
        printf("received %d from client[%d] : %s\n", n, i, buf);
        // 종료문자 처리
        if (strncmp(buf, EXIT, strlen(EXIT))==0) {
            RemoveClient(i);
            continue;
        }
        // 모든 채팅 참가자에게 메시지 방송
        //printf("%s", buf);
        // Write All
        for (j = 0; j < num_chat; j++) {
            ret = send(client_fds[j], buf, BUF_LEN, 0);
            if (ret <= 0) {
                printf("send error for client[%d]\n", j);
                client_error[i] = 1;
            }
        }
    }
}
```

```
// 오류가 난 client들을 뒤에서 앞으로 가면서 제거한다.
for (i = num_chat - 1; i >= 0; i--) {
    if (client_error[i])
        RemoveClient(i);
}

/* 채팅 탈퇴 처리 */
void RemoveClient(int i) {
#ifdef WIN32
    closesocket(client_fds[i]);
#else
    close(client_fds[i]);
#endif
    // 마지막 client를 삭제된 자리로 이동 (한칸씩 내릴 필요가 없다)
    if (i != num_chat - 1)
        client_fds[i] = client_fds[num_chat - 1];
    num_chat--;
    printf("client[%d] 퇴장. 현재 참가자 수 = %d\n", i, num_chat);
}

// client_fds[] 내의 최대 소켓번호 확인
// select(maxfds, ..) 에서 maxfds = getmax(server_fd) + 1
int getmax(int k) {
    int max = k;
    int r;
    for (r = 0; r < num_chat; r++) {
        if (client_fds[r] > max) max = client_fds[r];
    }
    return max;
}
```

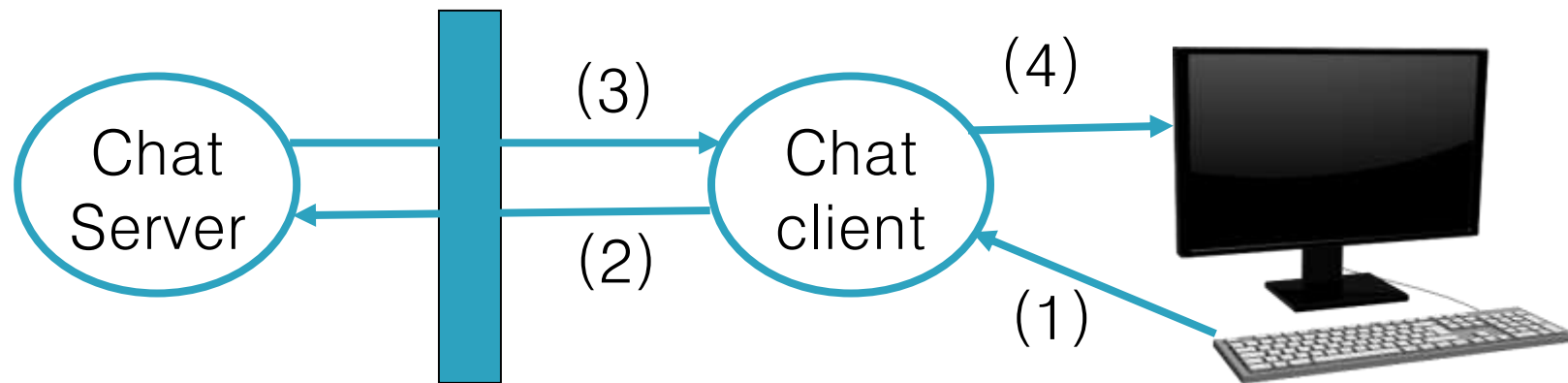

Linux Chat Client – select() 사용

▶ Echo client 와 거의 동일

(1) Read from Keyboard → (2) Write to Network

(3) Read from Network → (4) Write to Screen

- 단, (1)→(2) 와 (3)→(4) 가 동시에 가능해야 한다.
- select()에서 network(socket)과 stdin(keyboard)를 모니터링 한다.
- keyboard 입력이 발생하면 select()에서 감지된다.
- FD_SET(0, &read_fds); // keyboard 번호는 0 이다.



Linux chat_client

```
int maxfdp;
fd_set read_fds;
maxfdp = s + 1; // socket은 항상 0 보다 크게 할당된다.
FD_ZERO(&read_fds);
while (1) {
    FD_SET(0, &read_fds); // stdin:0 표준입력은 file 번호 = 0 이다.
    FD_SET(s, &read_fds); // server 와 연결된 socket 번호

    if (select(maxfdp, &read_fds, (fd_set*)0, (fd_set*)0, (struct timeval*)0) < 0) {
        printf("select error\n");
        exit(0);
    }
    // network I/O 변화 있는 경우
    if (FD_ISSET(s, &read_fds)) {
        if ((n = recv(s, buf2, BUF_LEN, 0)) > 0) {
            printf("%s", buf2);
        }
        else {
            printf("recv error\n");
            break;
        }
    }
    // keyboard 입력이 있는 경우
    if (FD_ISSET(0, &read_fds)) {
        if (fgets(buf1, BUF_LEN, stdin)) {
            //sprintf(buf2, "%s: %s", nickname, buf1);
            if (send(s, buf1, BUF_LEN, 0) < 0) {
                printf("send error\n");
                break;
            }
        }
        if (strncmp(buf1, EXIT, strlen(EXIT))==0) {
            printf("Good bye.\n");
            close(s);
            break;
        }
    }
}
}
```

exit 처리

Windows Chat Client

- ▶ select 를 keyboard(stdin)에 사용할 수 없음
- ▶ socket 을 Non-Blocking Mode 로 설정
 - 수신 Data 가 없을 때 recv()에서 무한대기하지 않는다.
 - recv() return 값이 < 0 이어도 (WSAGetLastError()==WSAEWOLDBLOCK) 이면 Data가 아직 없다는 의미이고 (WSAGetLastError()!=WSAEWOLDBLOCK) 이면 연결이 끊어졌다는 의미

```
u_long iMode = 1;
ioctlsocket(s, FIONBIO, &iMode);
ret = recv(s, buf, BUF_LEN, 0);
if (n > 0) {
    printf(buf);
} else if (WSAGetLastError()!=WSAEWOLDBLOCK) {
    printf("recv error\n"); // server 가 종료되었거나 네트워크 오류
    break;
}
```

- ▶ kbhit() 함수로 keyboard event를 check 해서 Data가 있으면 Read.

```
if (kbhit()) { // key 가 눌러있으면 read key --> write to chat server
    if (fgets(buf1, BUF_LEN, stdin)) { // Enter key 까지 입력 받고 전송
        if (send(s, buf1, BUF_LEN, 0) < 0) {
            printf("send error.\n");
            break;
        }
    }
}
```

Windows chat_client1.c

```
#ifdef WIN32
    u_long iMode = 1;
    ioctlsocket(s, FIONBIO, &iMode); // 소켓을 non-blocking 으로 만든다.
    int maxfdp1;
    while (1) {
        // Non-blocking read이므로 데이터가 없으면 기다리지 않고 0으로 return
        n = recv(s, buf2, BUF_LEN, 0);
        if (n > 0) { // non-blocking read
            // network에서 읽어서
            // 화면에 출력
            printf("%s", buf2);
        }
        else if (WSAGetLastError() != WSAEWOULDBLOCK) {
            printf("recv error\n"); // server 가 종료되었거나 네트워크 오류
            break;
        }
        if (kbhit()) { // key 가 눌러있으면 read key --> write to chat server
            memset(buf1, 0, BUF_LEN);
            if (fgets(buf1, BUF_LEN, stdin)) { // Enter key 까지 입력 받고 전송
                if (send(s, buf1, BUF_LEN, 0) < 0) {
                    printf("send error.\n");
                    break;
                }
                if (strcmp(buf1, EXIT) == 0) {
                    printf("Good bye.\n");
                    break;
                }
            }
            else {
                printf("fgets error\n");
                break;
            }
        }
        Sleep(100); // Non-blocking I/O CPU 부담을 줄인다.
    }
#else
```

exit 처리

Non-blocking의 경우
무한 Loop 상태이므로
CPU 사용이 증가할 수 있다.
Sleep()으로 Delay를 준다

C:\Windows\system32\cmd.exe

Windows : chat_server1 waiting connection..
server_fd = 252
Client connected from 127.0.0.1:8765
client_fd = 244
client[0] 입장. 현재 참가자 수 = 1
received 128 from client[0] : windows hello
Client connected from 192.168.126.130:55138
client_fd = 256
client[1] 입장. 현재 참가자 수 = 2
received 128 from client[1] : linux hello
received 128 from client[0] : hi from windows
received 128 from client[0] : exit
client[0] 퇴장. 현재 참가자 수 = 1

Windows
chat_server

user@user-virtual-machine: ~/netprog/06

user@user-virtual-machine:~/netprog/06\$./chat_client1 192.168.126.1 30000
Linux : Connecting 192.168.126.1 30000
채팅 서버에 접속되었습니다.
linux hello
linux hello
hi from windows

Linux
chat_client

C:\Windows\system32\cmd.exe

Windows : Connecting 127.0.0.1 30000
채팅 서버에 접속되었습니다.
windows hello
windows hello
linux hello
hi from windows
hi from windows
exit
Good bye.
계속하려면 아무 키나 누르십시오 . . .

Windows
chat_client

```
user@user-virtual-machine: ~/netprog/06
user@user-virtual-machine:~/netprog/06$ ./chat_server1
Linux : chat_server1 waiting connection..
server_fd = 3
Client connected from 127.0.0.1:36732
client_fd = 4
client[0] 입장. 현재 참가자 수 = 1
received 128 from client[0] : linux hello
Client connected from 192.168.126.1:9862
client_fd = 5
client[1] 입장. 현재 참가자 수 = 2
received 128 from client[1] : windows hello
received 128 from client[0] : hi
received 128 from client[1] : exit
client[1] 퇴장. 현재 참가자 수 = 1
[]

user@user-virtual-machine: ~/netprog/06
user@user-virtual-machine:~/netprog/06$ ./chat_client1
Linux : Connecting 127.0.0.1 30000
채팅 서버에 접속되었습니다.
linux hello
linux hello
windows hello
hi
hi
[]

C:\Windows\system32\cmd.exe
Windows : Connecting 192.168.126.130 30000
채팅 서버에 접속되었습니다.
windows hello
windows hello
hi
exit
Good bye.
계속하려면 아무 키나 누르십시오 . . .
```

Linux
chat_server

Linux
chat_client

Windows
chat_client

응용1 chat_client2/server2 : Username 사용, exit 처리

- ▶ chat_client1 : username이 보이지 않는다
- ▶ Client가 username 을 입력 받고 메시지와 합쳐서 전송
 - `scanf("%s", username); // atom`
 - `fgets(buf, BUF_LEN, stdin); // Hello`
 - `sprintf(buf2, "[%s] %s", username, buf1); "[atom] Hello"`
 - `send(s, buf2, BUF_LEN, 0); // "[atom] Hello" 전송`
- ▶ exit 문자 처리
 - chat_client1 : buf1을 `strncmp(buf1, EXIT, strlen(EXIT))==0` 검사
 - char_server1 : [username] 제외한 문자열 check
 - `recv(client_fd, buf, 128, 0); sscanf(buf, "%s", username);`
 - `strncmp(buf+strlen(username)+1, EXIT, strlen(EXIT))==0` 검사

echo_client2/server2 실행 화면

```
C:\Windows\system32\cmd.exe
Windows : chat_server2 waiting connection..
server_fd = 240
Client connected from 127.0.0.1:10927
client_fd = 260
client[0] 입장. 현재 참가자 수 = 1
received 128 from client[0] : [atom] hello
Client connected from 127.0.0.1:10928
client_fd = 264
client[1] 입장. 현재 참가자 수 = 2
received 128 from client[1] : [hansung] hi
received 128 from client[0] : [atom] exit
client[0] 퇴장. 현재 참가자 수 = 1

C:\Windows\system32\cmd.exe
chat client2 running.
Enter user name : atom
Windows : chat_client2 connecting 127.0.0.1 30000
채팅 서버에 접속되었습니다.
hello
[atom] hello
[hansung] hi
exit
Good bye.
계속하려면 아무 키나 누르십시오 . . .

선택 C:\Windows\system32\cmd.exe
chat client2 running.
Enter user name : hansung
Windows : chat_client2 connecting 127.0.0.1 30000
채팅 서버에 접속되었습니다.
hi
[hansung] hi
```

chat_client2.c

```
char username[BUF_LEN]; // user name
```

```
int main(int argc, char* argv[]) {
```

```
    if (argc == 3) {
```

```
        ip_addr = argv[1];
```

```
        port_no = argv[2];
```

```
    }
```

```
    printf("chat_client2 running.\n");
```

```
    printf("Enter user name : ");
```

```
    scanf("%s", username); getchar(); // \n제거
```

```
    if (kbhit()) { // key 가 눌러있으면 read key --> write to chat server
        if (fgets(buf1, BUF_LEN, stdin)) { // Enter key 까지 입력 받고 전송
            sprintf(buf2, "[%s] %s", username, buf1);
            if (send(s, buf2, BUF_LEN, 0) < 0) {
                printf("send error.\n");
                break;
            }
            if (strcmp(buf1, EXIT) == 0) {
                printf("Good bye.\n");
                break;
            }
        }
        else {
            printf("fgets error\n");
            break;
        }
    }
    sleep(100); // Non-blocking I/O CPU 부담을 줄인다.
```

chat_server2.c

```
char username[BUF_LEN];  
  
int main(int argc, char* argv[]) {
```

```
memset(client_error, 0, sizeof(client_error));  
/* 클라이언트가 보낸 메시지를 모든 클라이언트에게 방송 */  
for (i = 0; i < num_chat; i++) {  
    // 각각의 client들의 I/O 변화가 있는지.  
    if (FD_ISSET(client_fds[i], &read_fds)) {  
        // Read One 또는 client 비정상 종료 확인  
        if ((n = recv(client_fds[i], buf, BUF_LEN, 0)) <= 0) {  
            // client 가 비 정상 종료한 경우  
            printf("recv error for client[%d]\n", i);  
            client_error[i] = 1;  
            continue;  
        }  
        printf("received %d from client[%d] : %s", n, i, buf);  
        sscanf(buf, "%s", username);  
        // 종료문자 처리  
        // buf = "[hansung] exit" 의 경우 username = [hansung]  
        // "exit" 는 buf + strlen(username) + 1 에서 시작  
        if (strncmp(buf + strlen(username) + 1, EXIT, strlen(EXIT)) == 0) {  
            RemoveClient(i);  
            continue;  
        }  
        // 모든 채팅 참가자에게 메시지 방송
```

exit 문자 처리

```
C:\Windows\system32\cmd.exe
Windows : chat_server waiting connection..
server_fd = 244
Client connected from 127.0.0.1:1142
client_fd = 256
client[0] 입장. 현재 참가자 수 = 1
received 128 from client[0] : [atom] exit

recv error for client[0]
client[0] 퇴장. 현재 참가자 수 = 0
```

exit 문자처리 잘 못 된 경우
recv() 오류로 종료

```
C:\Windows\system32\cmd.exe
Windows : chat_server waiting connection..
server_fd = 236
Client connected from 127.0.0.1:2994
client_fd = 252
client[0] 입장. 현재 참가자 수 = 1
received 128 from client[0] : [atom] hello there

nickname=[atom]
received 128 from client[0] : [atom] exit

nickname=[atom]
client[0] 퇴장. 현재 참가자 수 = 0
```

```
C:\Windows\system32\cmd.exe
Enter name : atom
Windows : Connecting 127.0.0.1 30000
채팅 서버에 접속되었습니다.
hello there
[atom] hello there
exit
Good bye.
계속하려면 아무 키나 누르십시오 . . .
```

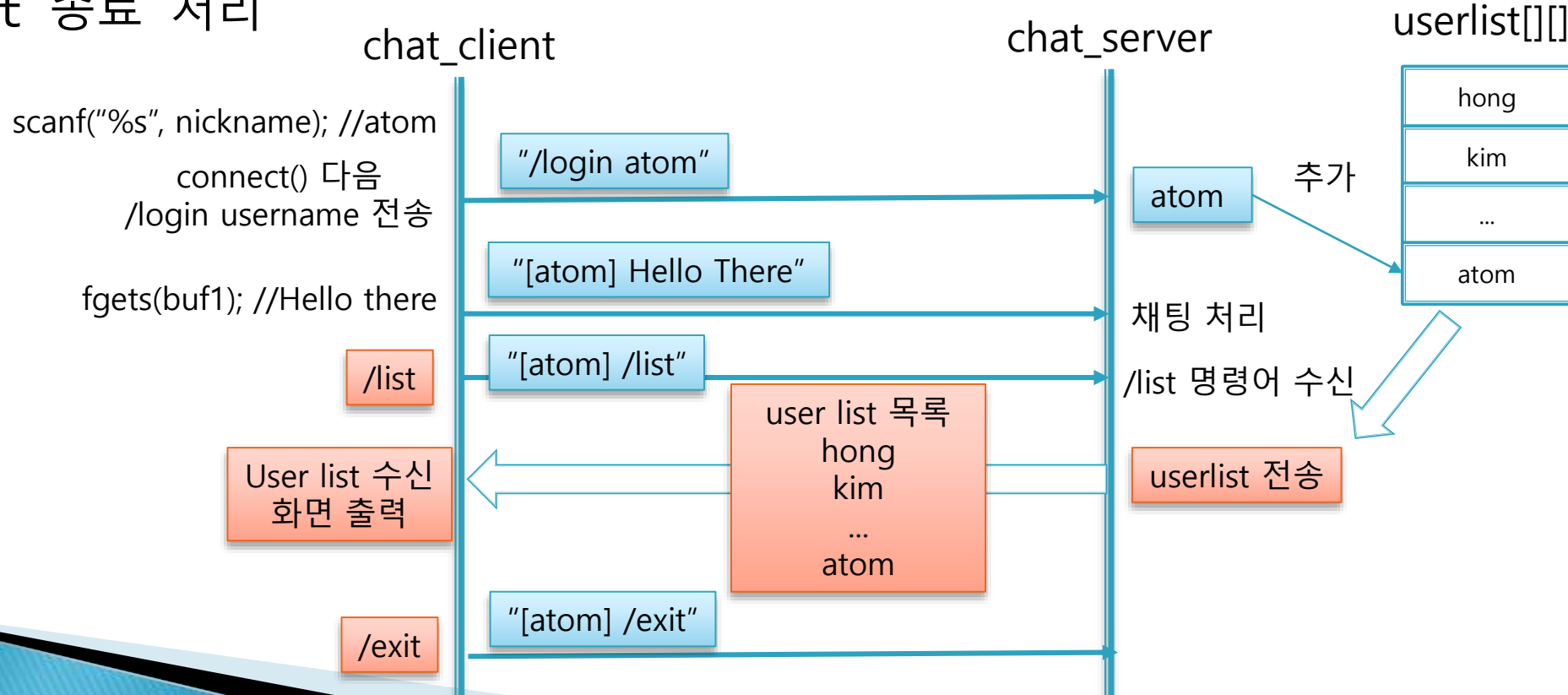
exit 문자 처리 정상

응용2 chat_client3/server3 : 특수 기능 추가하기

- ▶ 채팅 특수 명령어/프로토콜 추가
 - /login, /list, /exit
- ▶ /login : username 전송용
 - client_chat3가 connect 다음 “/login username” 전송
 - Server에서 Client username 보관
- ▶ /list : 사용자 목록보기 - “[atom] /list”
 - /login 으로 저장된 사용자 목록 보기
- ▶ /exit : 채팅 종료 - “[atom] /exit”
 - exit 대신 /exit 처리

chat_client3/server3 : 프로토콜

- ▶ Client는 접속하면 nickname을 우선 Server에게 전송
 - “/login nickname” 형식 (예: “/login atom”)
- ▶ Server가 각각의 Client들의 nickname을 보관
 - char userlist[MAXCLIENTS][BUF_LEN] 사용
- ▶ Client에서 특수문자열(예: /list)을 받으면 Client에게 사용자 목록을 전송
- ▶ /exit 종료 처리



chat_client3/server3 /login, /list, /exit 처리 화면

C:\Windows\system32\cmd.exe

```
Windows : chat_server3 waiting connection..
server_fd = 252
Client connected from 127.0.0.1:13219
client_fd = 248
client[0] 입장. 현재 참가자 수 = 1
received 128 from client[0] : /login atom
userlist[0] = atom
received 128 from client[0] : [atom] hello
received 128 from client[0] : [atom] /list
Sending user list to client[0] atom
Client connected from 127.0.0.1:13221
client_fd = 256
client[1] 입장. 현재 참가자 수 = 2
received 128 from client[1] : /login hansung
userlist[1] = hansung
received 128 from client[1] : [hansung] hi
received 128 from client[1] : [hansung] /list
Sending user list to client[1] hansung
received 128 from client[0] : [atom] hihi
received 128 from client[0] : [atom] /exit
client[0] atom 퇴장. 현재 참가자 수 = 1
received 128 from client[0] : [hansung] /list
Sending user list to client[0] atom
```

C:\Windows\system32\cmd.exe

```
chat_client3 running.
Enter user name : atom
Windows : Connecting 127.0.0.1 30000
채팅 서버에 접속되었습니다.
atom> hello
[atom] hello
atom> /list
User      list
-----
00        atom
-----
[hansung] hi
atom> hihi
[atom] hihi
atom> /exit
Good bye.
계속하려면 아무 키나 누르십시오 . . .
```

C:\Windows\system32\cmd.exe

```
chat_client3 running.
Enter user name : hansung
Windows : Connecting 127.0.0.1 30000
채팅 서버에 접속되었습니다.
hansung> hi
[hansung] hi
hansung> /list
User      list
-----
00        atom
01        hansung
-----
[atom] hihi
hansung> /list
User      list
-----
00        hansung
-----
```


6주 과제1: 강의내용 복습

- ▶ 1단계 : chat_client1/server1
 - 기본적인 chatting
- ▶ 2단계 : chat_client2/server2
 - username 입력, [username] message .. 형식 사용
 - exit 종료 처리
- ▶ 3단계 : chat_client3/server3
 - /list 로 사용자 목록 보기
 - /login, /list, /exit 처리

6주 과제2: chat_client4/server4 채팅 기능 추가하기

- ▶ 새로운 사용자가 입장/퇴장하면 다른 사용자에게 알려주기
 - [user1]님이 입장하였습니다.
 - [user1]님이 퇴장하였습니다.
- ▶ /to 명령어 - 귓속말 기능
 - /list로 사용자 목록 확인후 사용하는 식
 - char userlist[MAXCLIENTS][BUF_LEN] 를 이용
 - atom 화면
 - atom> /to hansung Hello there ..
 - hansung 화면
 - [귓속말] [atom] Hello there ...
- ▶ /sleep 명령어 - 부재중 기능
 - 채팅 메시지 수신 거부
 - atom> /sleep
 - /wakeup 또는 아무 message나 전송하면 깨어난다.
 - userlist[][] 외에 usersleep[] 변수 추가, 서버에서 관리
- ▶ /list 에 user 상태 표시 추가

chat_client4/server4 - 입장/퇴장 알리기

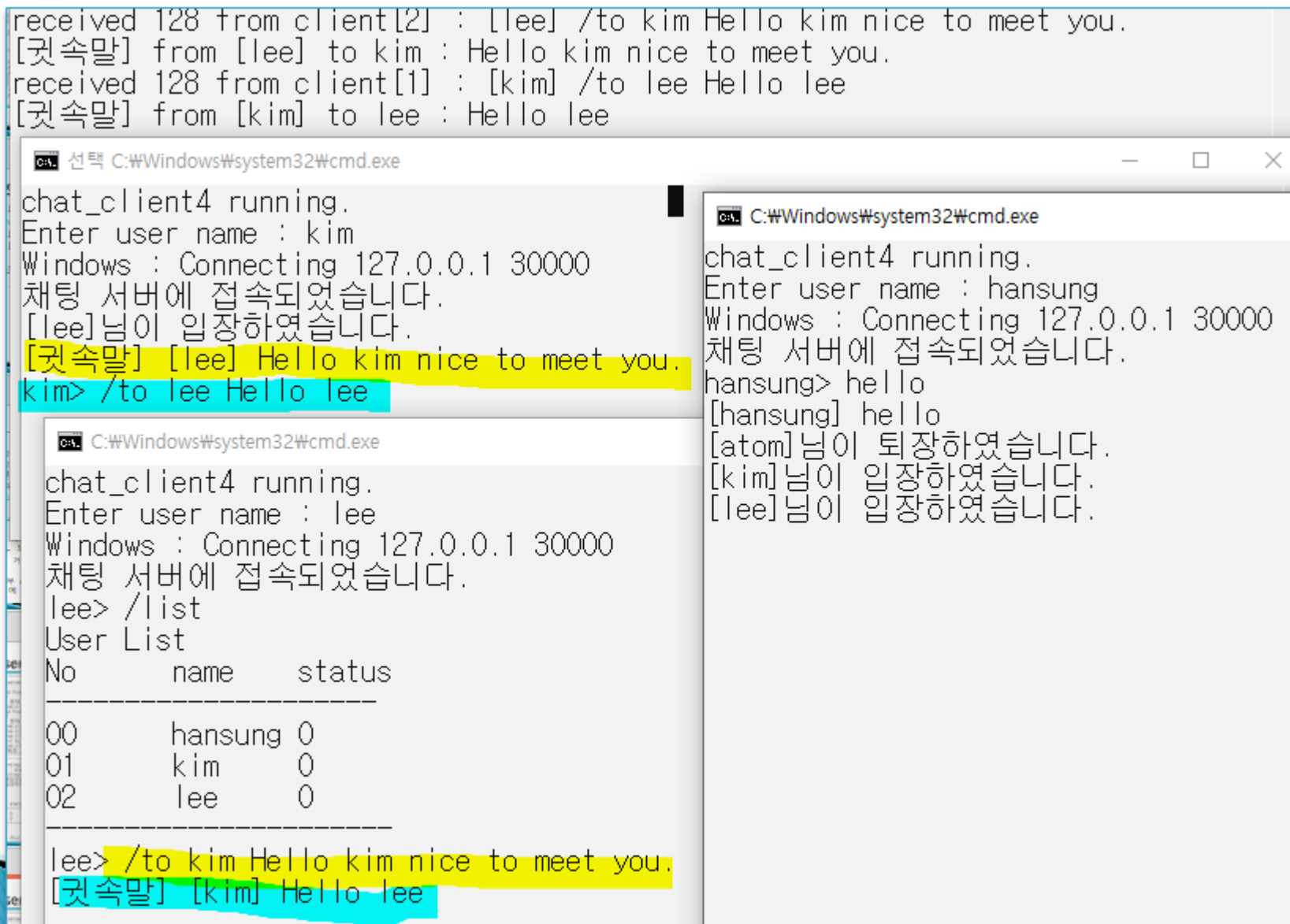
```
C:\Windows\system32\cmd.exe
Windows : chat_server4 waiting connection..
server_fd = 204
Client connected from 127.0.0.1:4631
client_fd = 256
client[0] 입장. 현재 참가자 수 = 1
received 128 from client[0] : /login atom userlist[0] = atom
Client connected from 127.0.0.1:4648
client_fd = 260
client[1] 입장. 현재 참가자 수 = 2
received 128 from client[1] : /login hansung userlist[1] = hansung
received 128 from client[1] : [hansung] hello
received 128 from client[0] : [atom] /list
Sending user list to client[0] atom
received 128 from client[0] : [atom] /exit
client[0] atom 퇴장. 현재 참가자 수 = 1
```

```
C:\Windows\system32\cmd.exe
chat_client4 running.
Enter user name : atom
Windows : Connecting 127.0.0.1 30000
채팅 서버에 접속되었습니다.
[hansung]님이 입장하였습니다.
[hansung] hello
atom> /list
User List
No      name      status
-----
00      atom      0
01      hansung   0
atom> /exit
Good bye.
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
chat_client4 running.
Enter user name : hansung
Windows : Connecting 127.0.0.1 30000
채팅 서버에 접속되었습니다.
hansung> hello
[hansung] hello
[atom]님이 퇴장하였습니다.
```

chat_client4/server4 – 귓속말 보내기

```
received 128 from client[2] : [lee] /to kim Hello kim nice to meet you.  
[귓속말] from [lee] to kim : Hello kim nice to meet you.  
received 128 from client[1] : [kim] /to lee Hello lee  
[귓속말] from [kim] to lee : Hello lee
```



```
C:\Windows\system32\cmd.exe  
chat_client4 running.  
Enter user name : kim  
Windows : Connecting 127.0.0.1 30000  
채팅 서버에 접속되었습니다.  
[lee]님이 입장하였습니다.  
[귓속말] [lee] Hello kim nice to meet you.  
kim> /to lee Hello lee
```

```
C:\Windows\system32\cmd.exe  
chat_client4 running.  
Enter user name : hansung  
Windows : Connecting 127.0.0.1 30000  
채팅 서버에 접속되었습니다.  
hansung> hello  
[hansung] hello  
[atom]님이 퇴장하였습니다.  
[kim]님이 입장하였습니다.  
[lee]님이 입장하였습니다.
```

```
C:\Windows\system32\cmd.exe  
chat_client4 running.  
Enter user name : lee  
Windows : Connecting 127.0.0.1 30000  
채팅 서버에 접속되었습니다.  
lee> /list  
User List  
No      name    status  
-----  
00      hansung  0  
01      kim      0  
02      lee      0  
-----  
lee> /to kim Hello kim nice to meet you.  
[귓속말] [kim] Hello lee
```

chat_client4/server4 – /sleep, /list, /wakeup

```
received 128 from client[1] : [kim] /sleep
received 128 from client[2] : [lee] /list
Sending user list to client[2] lee
received 128 from client[0] : [hansung] /list
Sending user list to client[0] hansung
received 128 from client[0] : [hansung] hello
received 128 from client[0] : [hansung] /to kim hi
[컨속말] from [hansung] to kim : hi
received 128 from client[1] : [kim] /wakeup
received 128 from client[1] : [kim] hi
received 128 from client[0] : [hansung] hello
received 128 from client[0] : [hansung] /list
```

C:\Windows\system32\cmd.exe

No	name	status
00	hansung	0
01	kim	0
02	lee	0

[lee] welcom
hansung> /list
User List

No	name	status
00	hansung	0
01	kim	S
02	lee	0

hansung> hello
[hansung] hello
hansung> /to kim hi
[kim] hi
hansung> hello
[hansung] hello
hansung> /list
User List

No	name	status
00	hansung	0
01	kim	0
02	lee	0

C:\Windows\system32\cmd.exe

[kim] hello
[hansung] hi
kim> /sleep
kim> /wakeup
[lee] welcom
kim> /sleep
kim> /wakeup
kim> hi
[kim] hi
[hansung] hello

C:\Windows\system32\cmd.exe

[hansung] hello
lee> welcom
[lee] welcom
lee> /list
User List

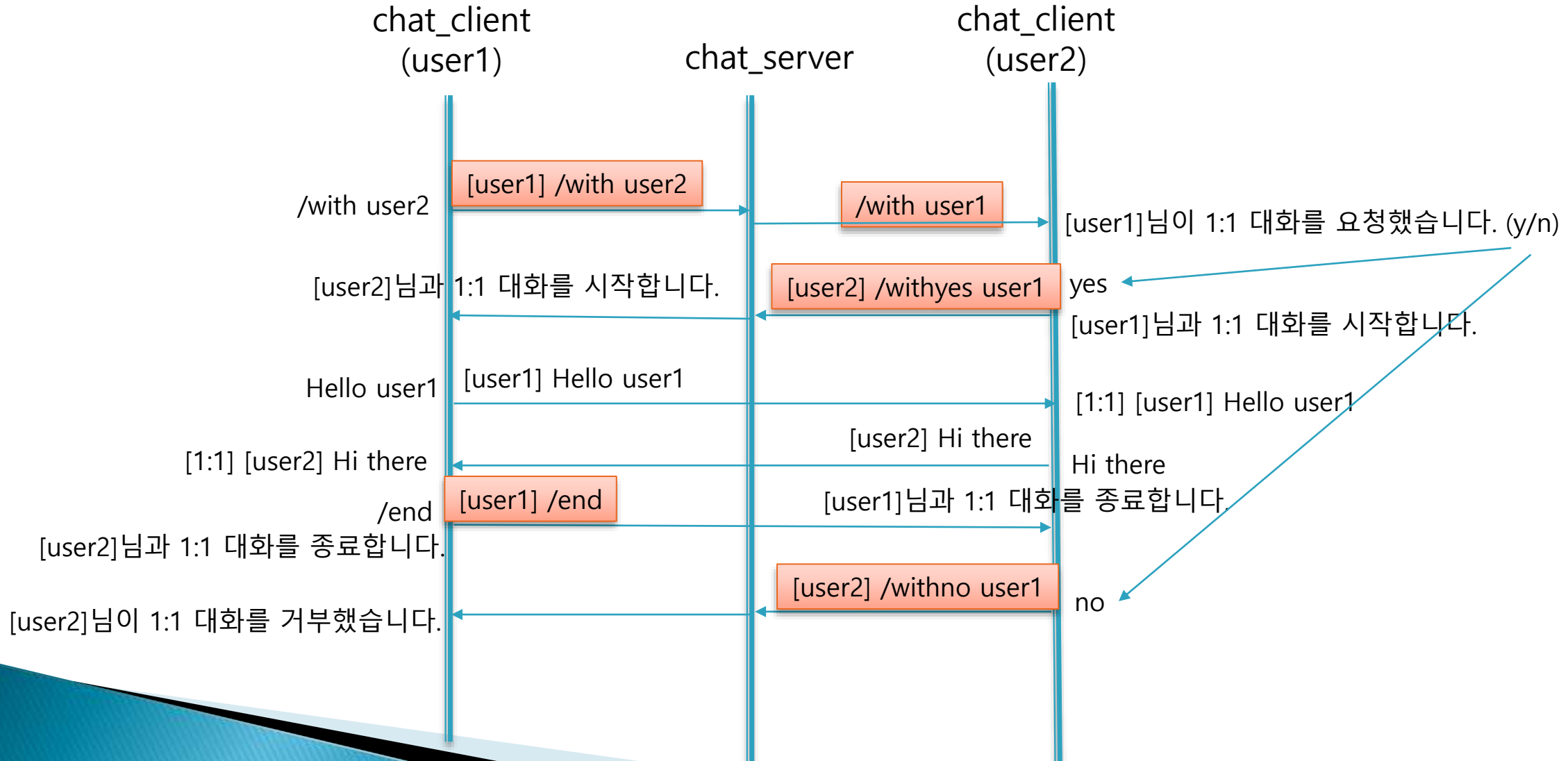
No	name	status
00	hansung	0
01	kim	S
02	lee	0

[hansung] hello
[kim] hi
[hansung] hello

6주 도전과제1: 1:1 채팅모드 전환

- ▶ 명령어 추가
 - /with user2 - user2 와 1:1 채팅 시작
 - /end - 1:1 채팅 종료
- ▶ 1:1 채팅을 user2가 허락해야 가능하도록 프로토콜 구현
예) user1 화면 - user1> /with user2
user2 화면 - [user1]님이 1:1 대화를 요청했습니다. (y/n)? y
user1 화면 - [user2]님과 1:1 대화 시작
user2 화면 - [user1]님과 1:1 대화 시작
user1 화면 - user1> /end (또는 user2 화면 - user2> /end)
user1 화면 - user1> Hllo user2
user2 화면 - [1:1] [user1] Hello user2, user2> Hi user1
user1 화면 - [1:1] [user2] Hi user1
user1 화면 - [user2]님과 1:1 대화 종료
user2 화면 - [user1]님과 1:1 대화 종료
user2 화면 - [user1]님이 1:1 대화를 요청했습니다. (y/n)? n
user1 화면 - [user2]님이 1:1 대화를 거부했습니다.
- ▶ Hint : Sever에서 1:1 제어 int userwith[] 배열에 정보 저장
 - /with 를 요청한 user1 번호 = i, user2 번호 = k 라면
 - userwith[i] = k, userwith[k] = i 로 저장하고
 - user1이 message를 보내면 1:1 대상자에게만 전송

1:1 채팅 프로토콜



6주 도전과제2 : 파일 전송 기능 chat_client6/chat_server6

▶ 명령어 추가

- /filesend user2 filename

▶ 파일 전송을 user2가 허락해야 가능하도록 구현

예) user1 화면 - user1> /filesend user2 data.txt
user2 화면 - [user1]님이 파일을 보내려고 합니다. 수신 (y/n)? y
user1 화면 - [user2]님에게 data.txt 전송중
user1 화면 - data.txt 765 bytes 전송 완료.
user2 화면 - [user1]에서 data.txt 수신중.
user2 화면 - data.txt 765 bytes 수신 완료.
user2 화면 - [user1]님이 파일을 보내려고 합니다. 수신 (y/n)? n
user1 화면 - [user2]님이 파일 수신을 거부했습니다.

▶ 파일전송 프로토콜 고려사항

- Server가 file_client/server 프로토콜을 중계해야 된다.
- 1:1 대화모드와 같은 방식으로 키입력/화면 대신 file을 사용한다.

▶ 주의사항

- 파일전송 테스트할 때 chat_client6를 두 번 실행하면 파일 전송시 같은 폴더에서 같은 폴더로 전송을 시동하기 때문에 오류가 발생할 수 있음.
- 따라서 chat_client6를 chat_client6-2 로 복사해서 폴더를 새로 만들어서 각각 실행하고 테스트해야 됨

파일 전송 프로토콜

