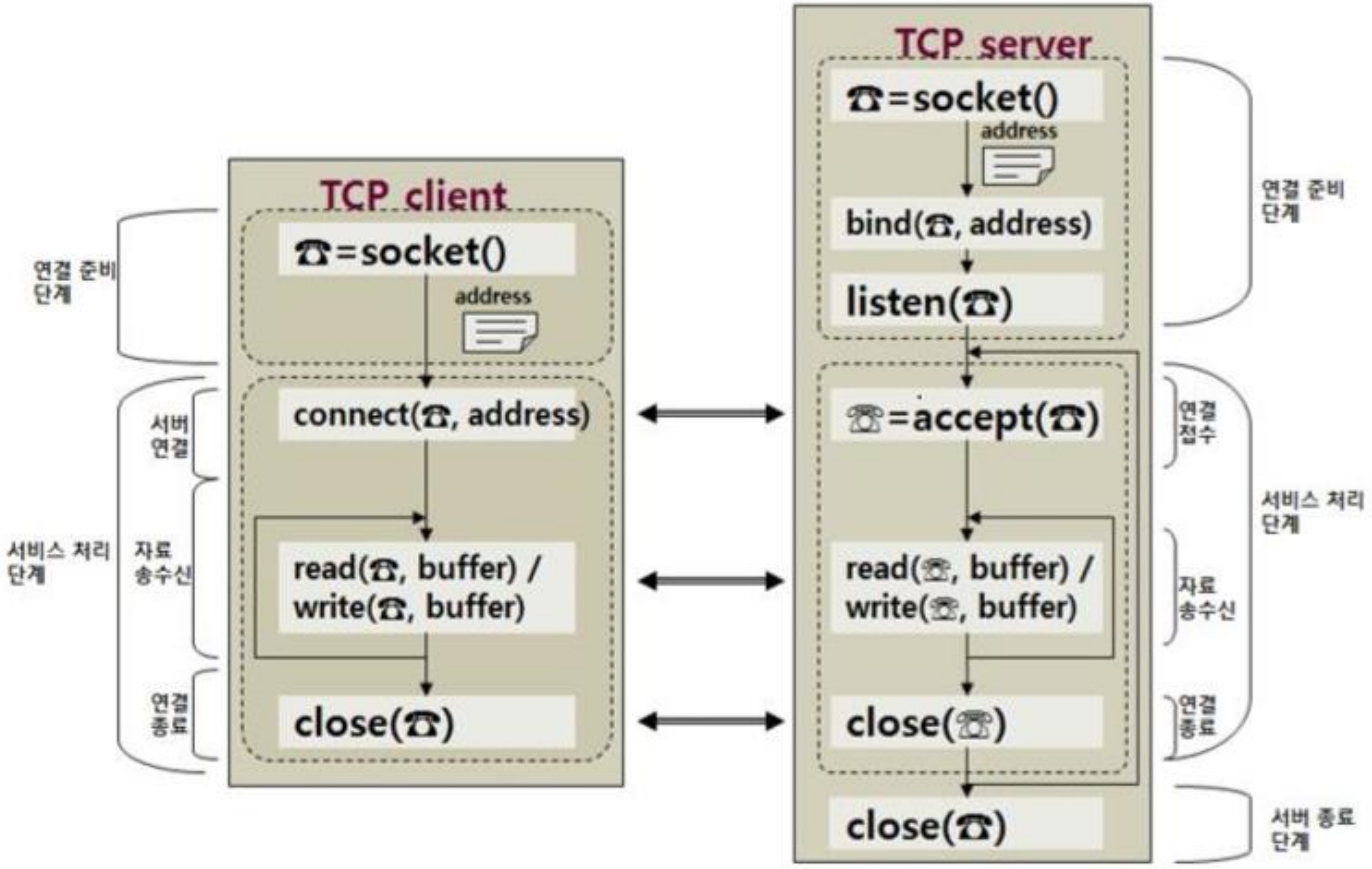
# Chapter3
# 소켓 프로그래밍

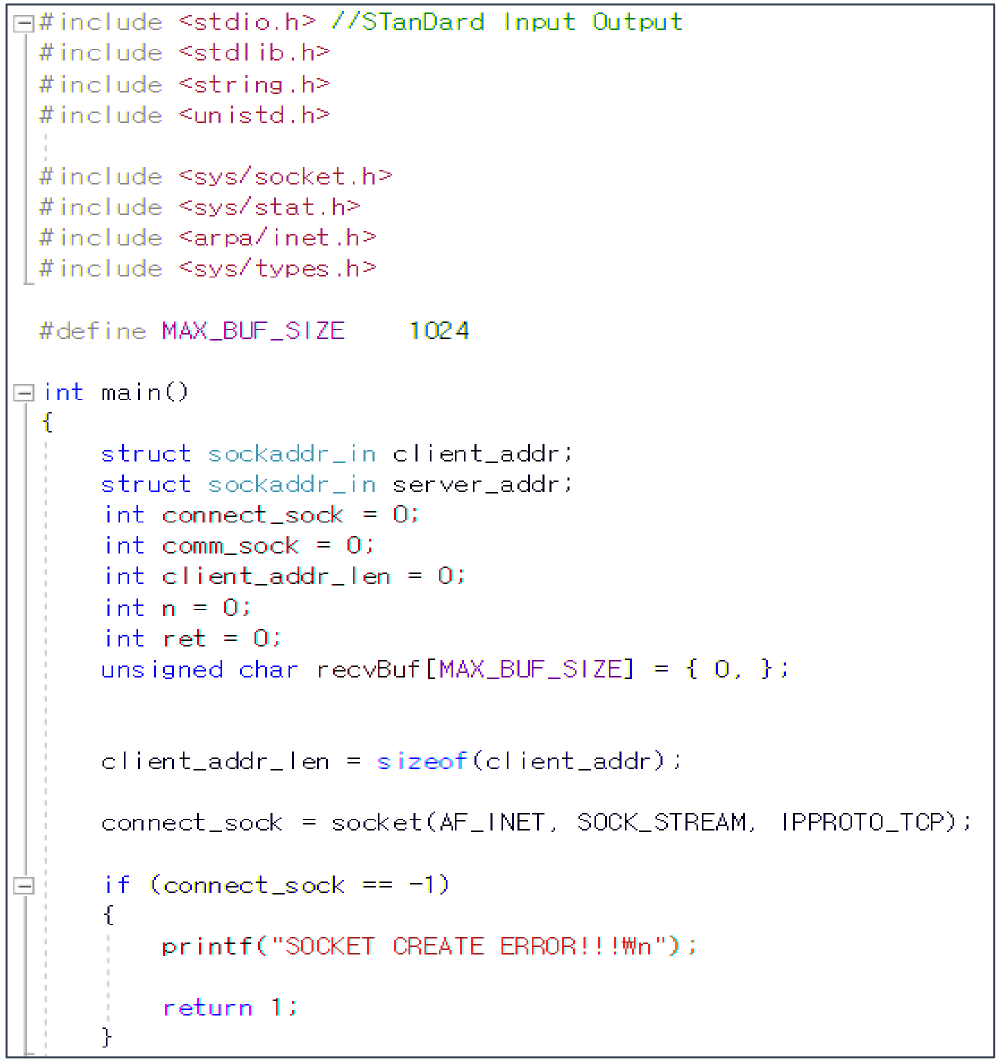
정보보안암호수학과 20182209 김수빈

# 소켓 프로그래밍

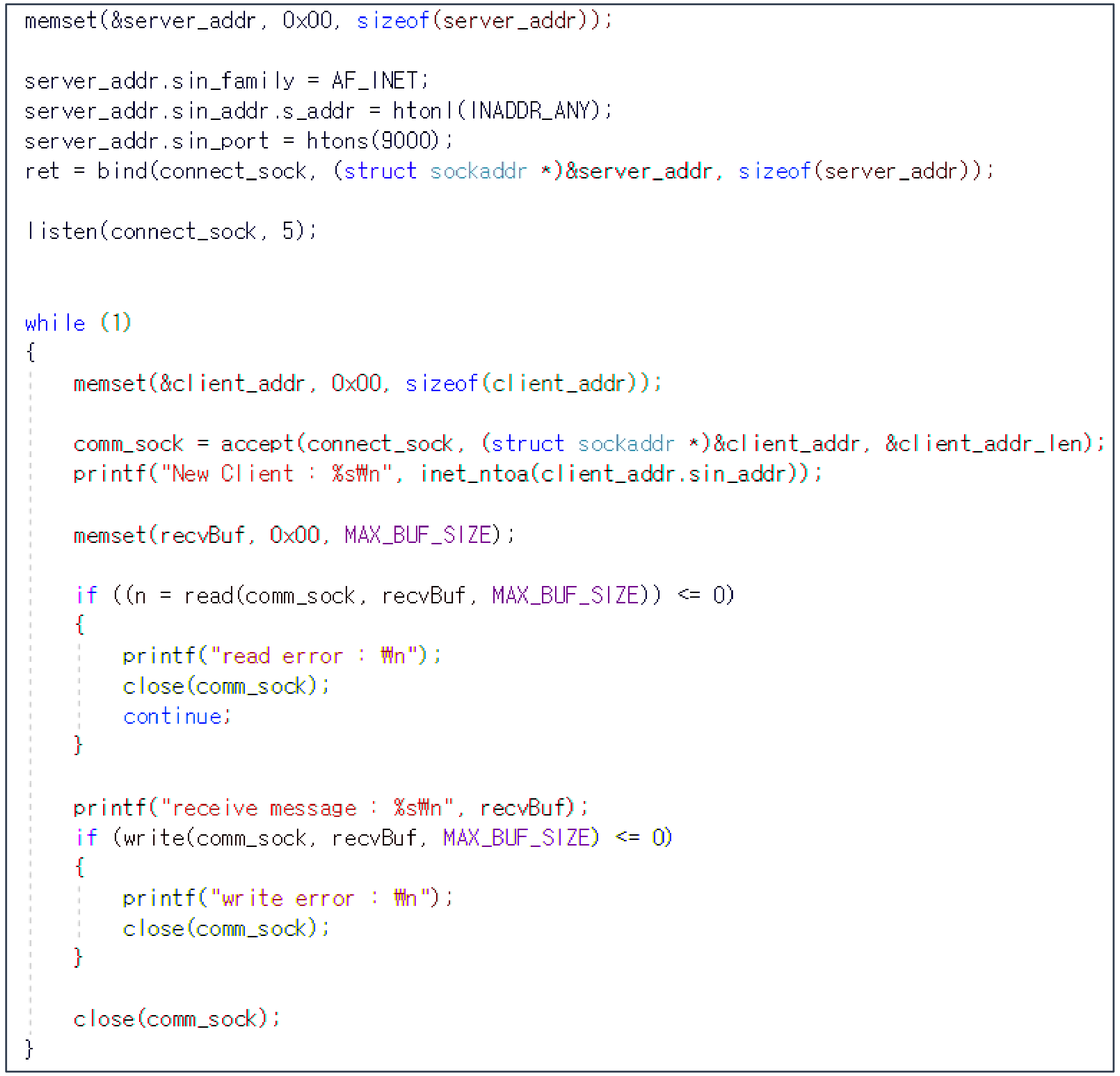

TCP client
연결 준비 단계
☎=socket()
address
서비스 처리 단계
서버 연결
connect(☎, address)
자료 송수신
read(☎, buffer) /
write(☎, buffer)
연결 종료
close(☎)
TCP server
☎=socket()
address
bind(☎, address)
listen(☎)
연결 준비 단계
☏=accept(☎)
연결 접수
서비스 처리 단계
read(☏, buffer) /
write(☏, buffer)
자료 송수신
close(☏)
연결 종료
close(☎)
서버 종료 단계

# 소켓 서버 프로그래밍 | server_echo.c

## server_echo.c 코드



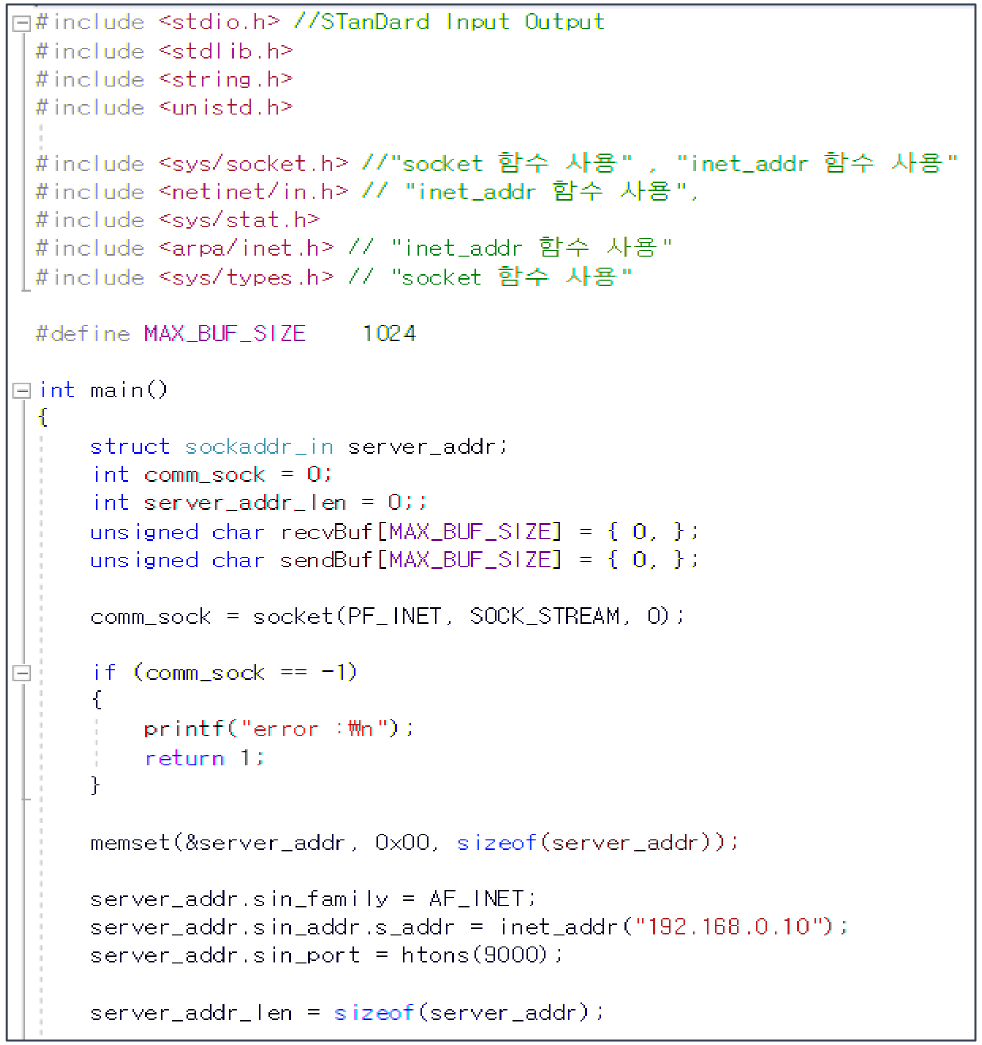
```
#include <stdio.h> //STanDard Input Output
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <sys/socket.h>
#include <sys/stat.h>
#include <arpa/inet.h>
#include <sys/types.h>

#define MAX_BUF_SIZE    1024

int main()
{
    struct sockaddr_in client_addr;
    struct sockaddr_in server_addr;
    int connect_sock = 0;
    int comm_sock = 0;
    int client_addr_len = 0;
    int n = 0;
    int ret = 0;
    unsigned char recvBuf[MAX_BUF_SIZE] = { 0, };


    client_addr_len = sizeof(client_addr);

    connect_sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

    if (connect_sock == -1)
    {
        printf("SOCKET CREATE ERROR!!!₩n");

        return 1;
    }
```



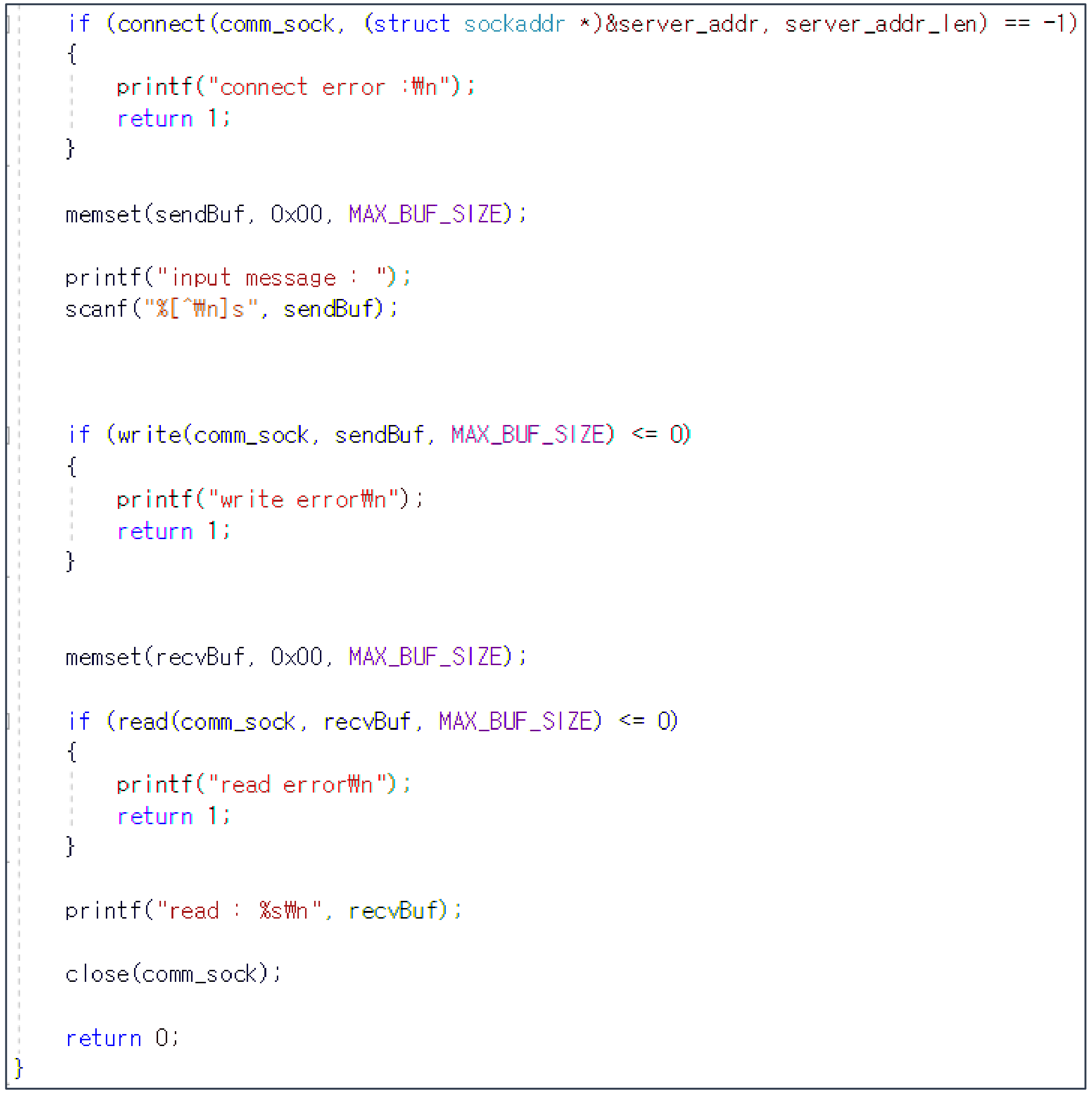
```
memset(&server_addr, 0x00, sizeof(server_addr));

server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
server_addr.sin_port = htons(9000);
ret = bind(connect_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));

listen(connect_sock, 5);


while (1)
{
    memset(&client_addr, 0x00, sizeof(client_addr));

    comm_sock = accept(connect_sock, (struct sockaddr *)&client_addr, &client_addr_len);
    printf("New Client : %s₩n", inet_ntoa(client_addr.sin_addr));

    memset(recvBuf, 0x00, MAX_BUF_SIZE);

    if ((n = read(comm_sock, recvBuf, MAX_BUF_SIZE)) <= 0)
    {
        printf("read error : ₩n");
        close(comm_sock);
        continue;
    }

    printf("receive message : %s₩n", recvBuf);
    if (write(comm_sock, recvBuf, MAX_BUF_SIZE) <= 0)
    {
        printf("write error : ₩n");
        close(comm_sock);
    }

    close(comm_sock);
}
```

# 소켓 클라이언트 프로그래밍 | client_echo.c

## client_echo.c 코드



```c
#include <stdio.h> //STanDard Input Output
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <sys/socket.h> //"socket 함수 사용" , "inet_addr 함수 사용"
#include <netinet/in.h> // "inet_addr 함수 사용",
#include <sys/stat.h>
#include <arpa/inet.h> // "inet_addr 함수 사용"
#include <sys/types.h> // "socket 함수 사용"

#define MAX_BUF_SIZE    1024

int main()
{
    struct sockaddr_in server_addr;
    int comm_sock = 0;
    int server_addr_len = 0;;
    unsigned char recvBuf[MAX_BUF_SIZE] = { 0, };
    unsigned char sendBuf[MAX_BUF_SIZE] = { 0, };

    comm_sock = socket(PF_INET, SOCK_STREAM, 0);

    if (comm_sock == -1)
    {
        printf("error :₩n");
        return 1;
    }

    memset(&server_addr, 0x00, sizeof(server_addr));

    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr("192.168.0.10");
    server_addr.sin_port = htons(9000);

    server_addr_len = sizeof(server_addr);
```



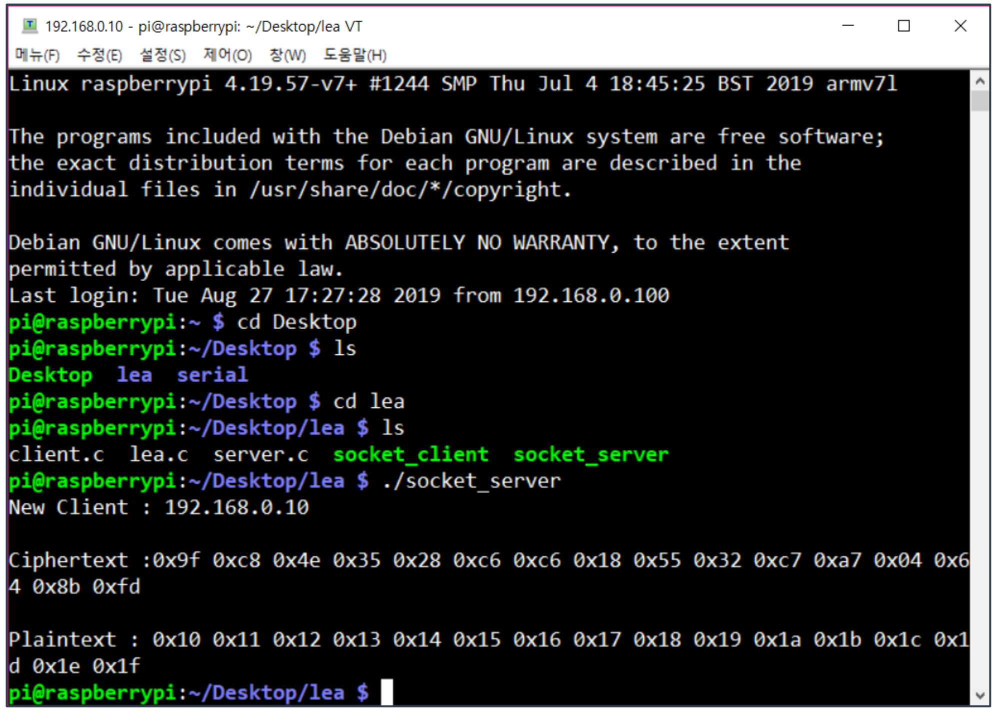
```c
    if (connect(comm_sock, (struct sockaddr *)&server_addr, server_addr_len) == -1)
    {
        printf("connect error :₩n");
        return 1;
    }

    memset(sendBuf, 0x00, MAX_BUF_SIZE);

    printf("input message : ");
    scanf("%[^₩n]s", sendBuf);



    if (write(comm_sock, sendBuf, MAX_BUF_SIZE) <= 0)
    {
        printf("write error₩n");
        return 1;
    }


    memset(recvBuf, 0x00, MAX_BUF_SIZE);

    if (read(comm_sock, recvBuf, MAX_BUF_SIZE) <= 0)
    {
        printf("read error₩n");
        return 1;
    }

    printf("read : %s₩n", recvBuf);

    close(comm_sock);

    return 0;
}
```

# 소켓 프로그래밍 + LEA

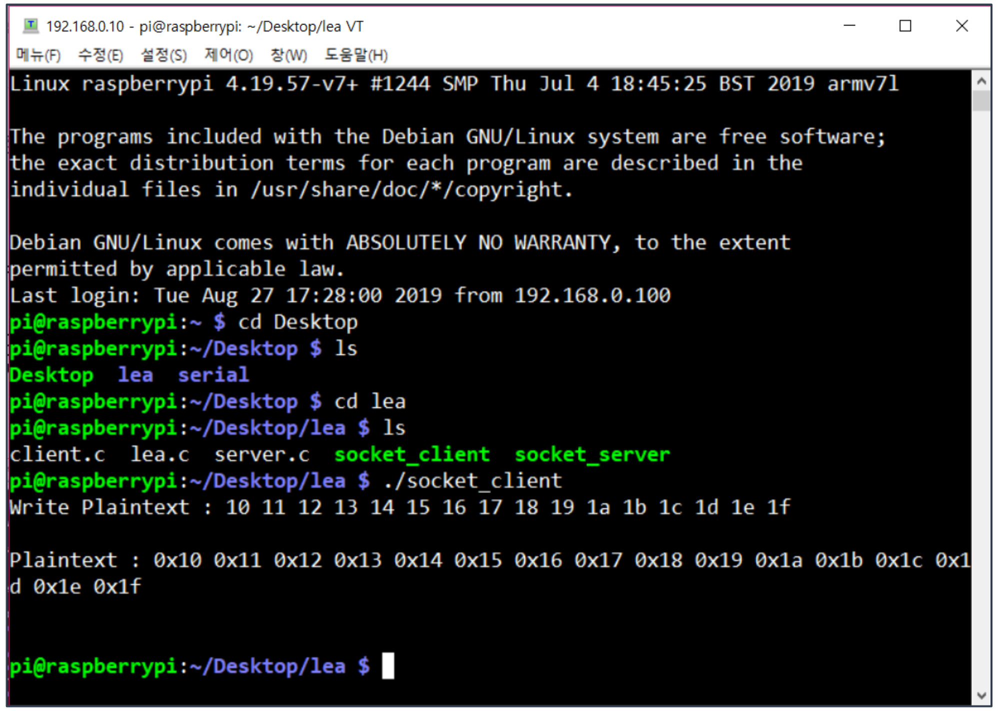
❖ server.c = server_echo.c + LEA복호화.c
❖ client.c = client_echo.c + LEA암호화.c

1. 서버와 클라이언트를 연결
2. 클라이언트에서 평문 입력한 후 암호화하여 서버로 전달
3. 서버에서 암호문을 받아 복호화해 평문 읽음

server.c ↔ client.c


192.168.0.10 - pi@raspberrypi: ~/Desktop/lea VT
메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

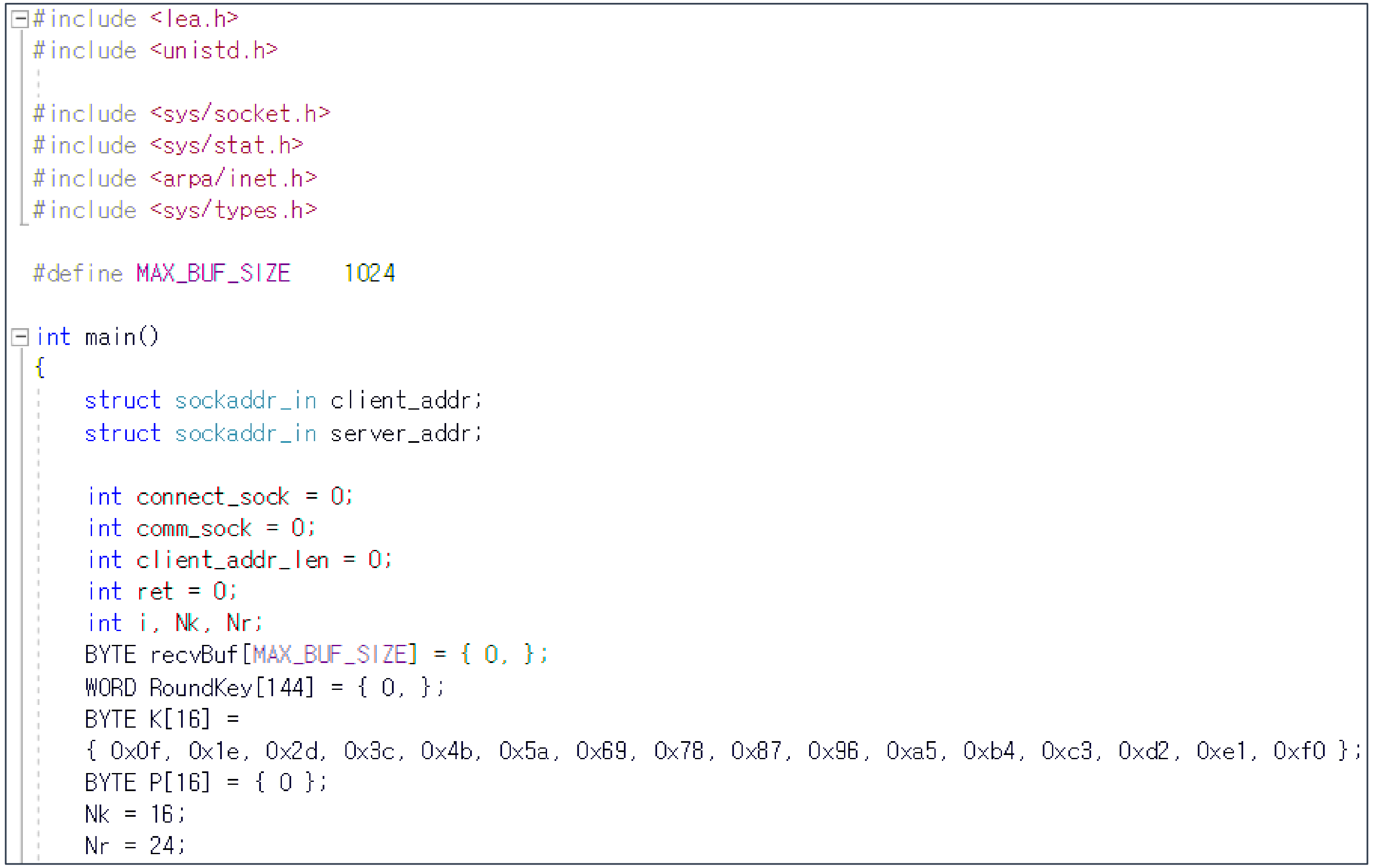
```
Linux raspberrypi 4.19.57-v7+ #1244 SMP Thu Jul 4 18:45:25 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Aug 27 17:27:28 2019 from 192.168.0.100
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ ls
Desktop  lea  serial
pi@raspberrypi:~/Desktop $ cd lea
pi@raspberrypi:~/Desktop/lea $ ls
client.c  lea.c  server.c  socket_client  socket_server
pi@raspberrypi:~/Desktop/lea $ ./socket_server
New Client : 192.168.0.10

Ciphertext :0x9f 0xc8 0x4e 0x35 0x28 0xc6 0xc6 0x18 0x55 0x32 0xc7 0xa7 0x04 0x6
4 0x8b 0xfd

Plaintext : 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1
d 0x1e 0x1f
pi@raspberrypi:~/Desktop/lea $
```


192.168.0.10 - pi@raspberrypi: ~/Desktop/lea VT
메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
Linux raspberrypi 4.19.57-v7+ #1244 SMP Thu Jul 4 18:45:25 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Aug 27 17:28:00 2019 from 192.168.0.100
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ ls
Desktop  lea  serial
pi@raspberrypi:~/Desktop $ cd lea
pi@raspberrypi:~/Desktop/lea $ ls
client.c  lea.c  server.c  socket_client  socket_server
pi@raspberrypi:~/Desktop/lea $ ./socket_client
Write Plaintext : 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

Plaintext : 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1
d 0x1e 0x1f


pi@raspberrypi:~/Desktop/lea $
```

## 소켓 서버 프로그래밍 + LEA 복호화 | server.c

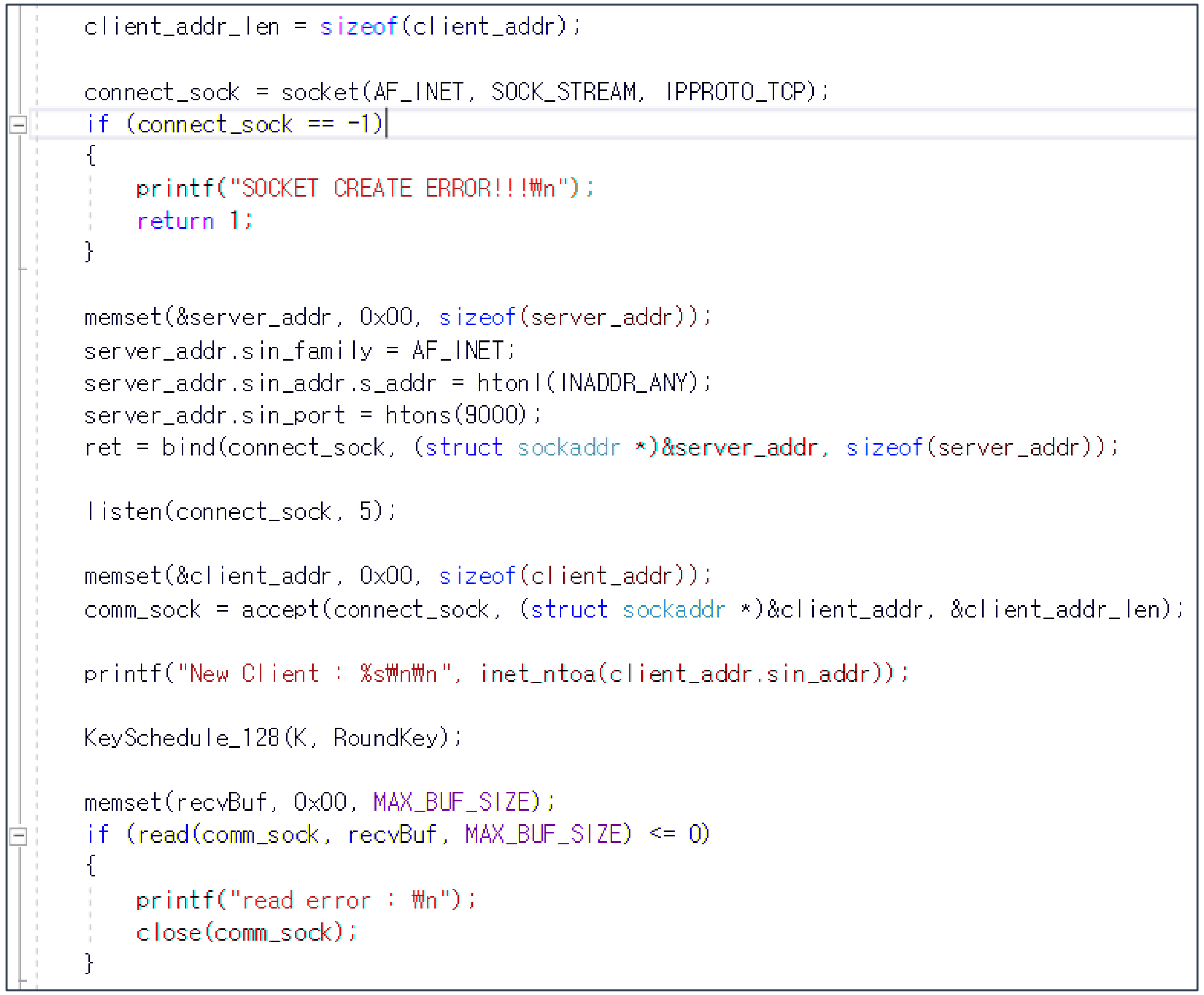
### server.c 코드



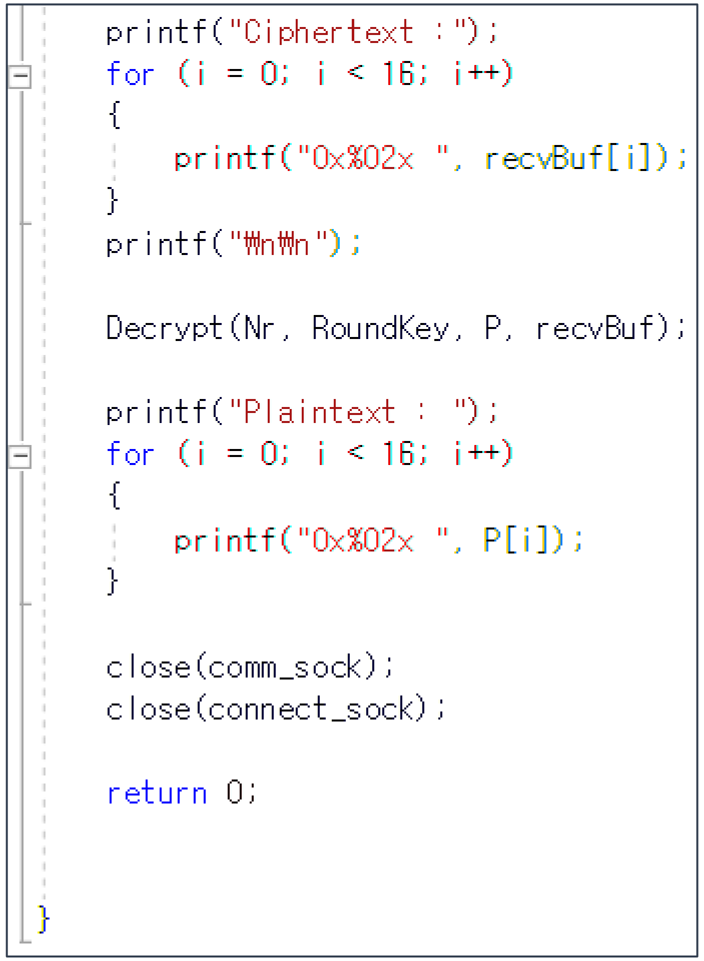
```c
#include <lea.h>
#include <unistd.h>

#include <sys/socket.h>
#include <sys/stat.h>
#include <arpa/inet.h>
#include <sys/types.h>

#define MAX_BUF_SIZE    1024

int main()
{
    struct sockaddr_in client_addr;
    struct sockaddr_in server_addr;

    int connect_sock = 0;
    int comm_sock = 0;
    int client_addr_len = 0;
    int ret = 0;
    int i, Nk, Nr;
    BYTE recvBuf[MAX_BUF_SIZE] = { 0, };
    WORD RoundKey[144] = { 0, };
    BYTE K[16] =
    { 0x0f, 0x1e, 0x2d, 0x3c, 0x4b, 0x5a, 0x69, 0x78, 0x87, 0x96, 0xa5, 0xb4, 0xc3, 0xd2, 0xe1, 0xf0 };
    BYTE P[16] = { 0 };
    Nk = 16;
    Nr = 24;
```

# 소켓 서버 프로그래밍 + LEA 복호화 | server.c

## server.c 코드



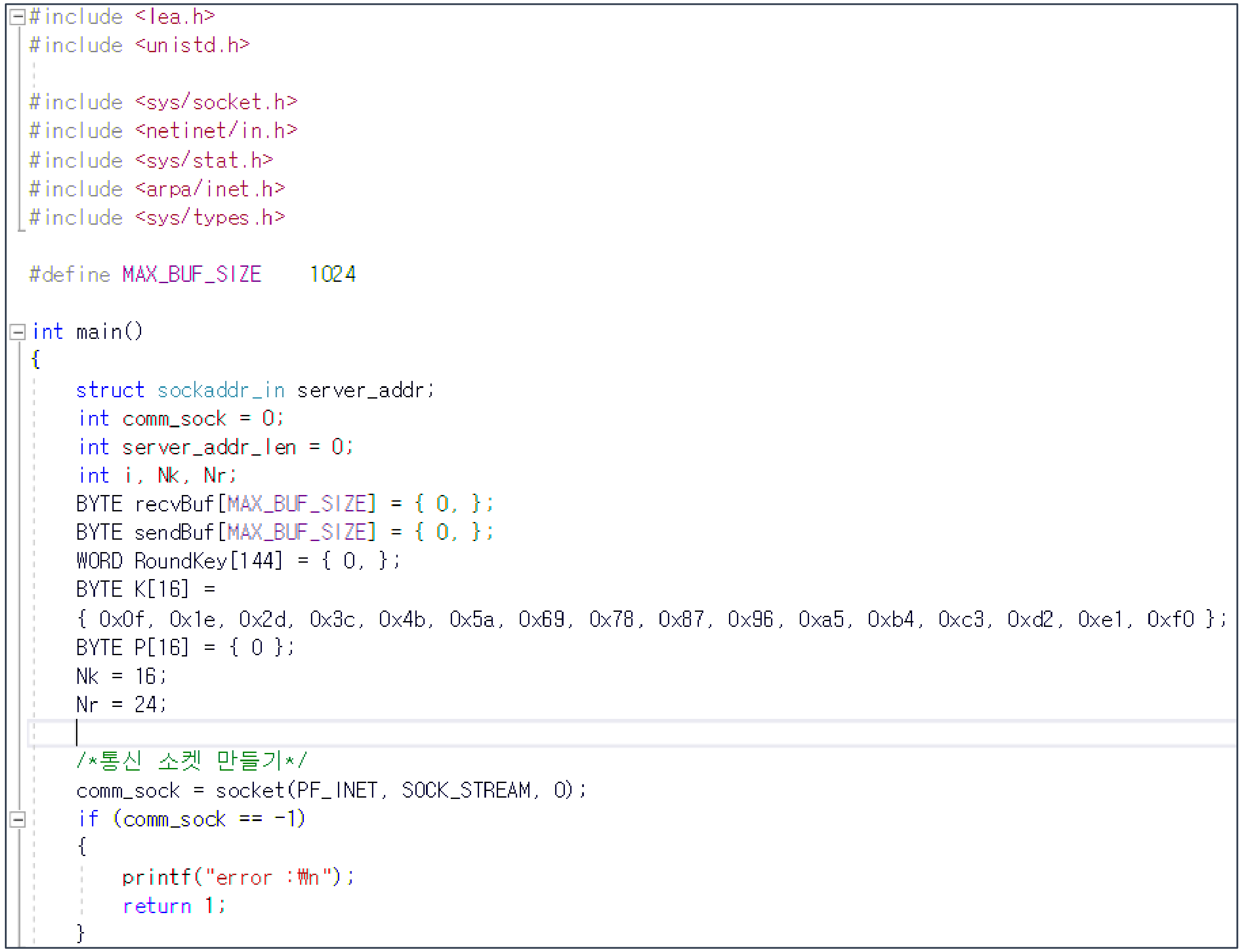
```c
    client_addr_len = sizeof(client_addr);

    connect_sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (connect_sock == -1)
    {
        printf("SOCKET CREATE ERROR!!!₩n");
        return 1;
    }

    memset(&server_addr, 0x00, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(9000);
    ret = bind(connect_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));

    listen(connect_sock, 5);

    memset(&client_addr, 0x00, sizeof(client_addr));
    comm_sock = accept(connect_sock, (struct sockaddr *)&client_addr, &client_addr_len);

    printf("New Client : %s₩n₩n", inet_ntoa(client_addr.sin_addr));

    KeySchedule_128(K, RoundKey);

    memset(recvBuf, 0x00, MAX_BUF_SIZE);
    if (read(comm_sock, recvBuf, MAX_BUF_SIZE) <= 0)
    {
        printf("read error : ₩n");
        close(comm_sock);
    }
```



```c
    printf("Ciphertext : ");
    for (i = 0; i < 16; i++)
    {
        printf("0x%02x ", recvBuf[i]);
    }
    printf("₩n₩n");

    Decrypt(Nr, RoundKey, P, recvBuf);

    printf("Plaintext : ");
    for (i = 0; i < 16; i++)
    {
        printf("0x%02x ", P[i]);
    }

    close(comm_sock);
    close(connect_sock);

    return 0;


}
```

# 소켓 클라이언트 프로그래밍 + LEA 암호화 | client.c

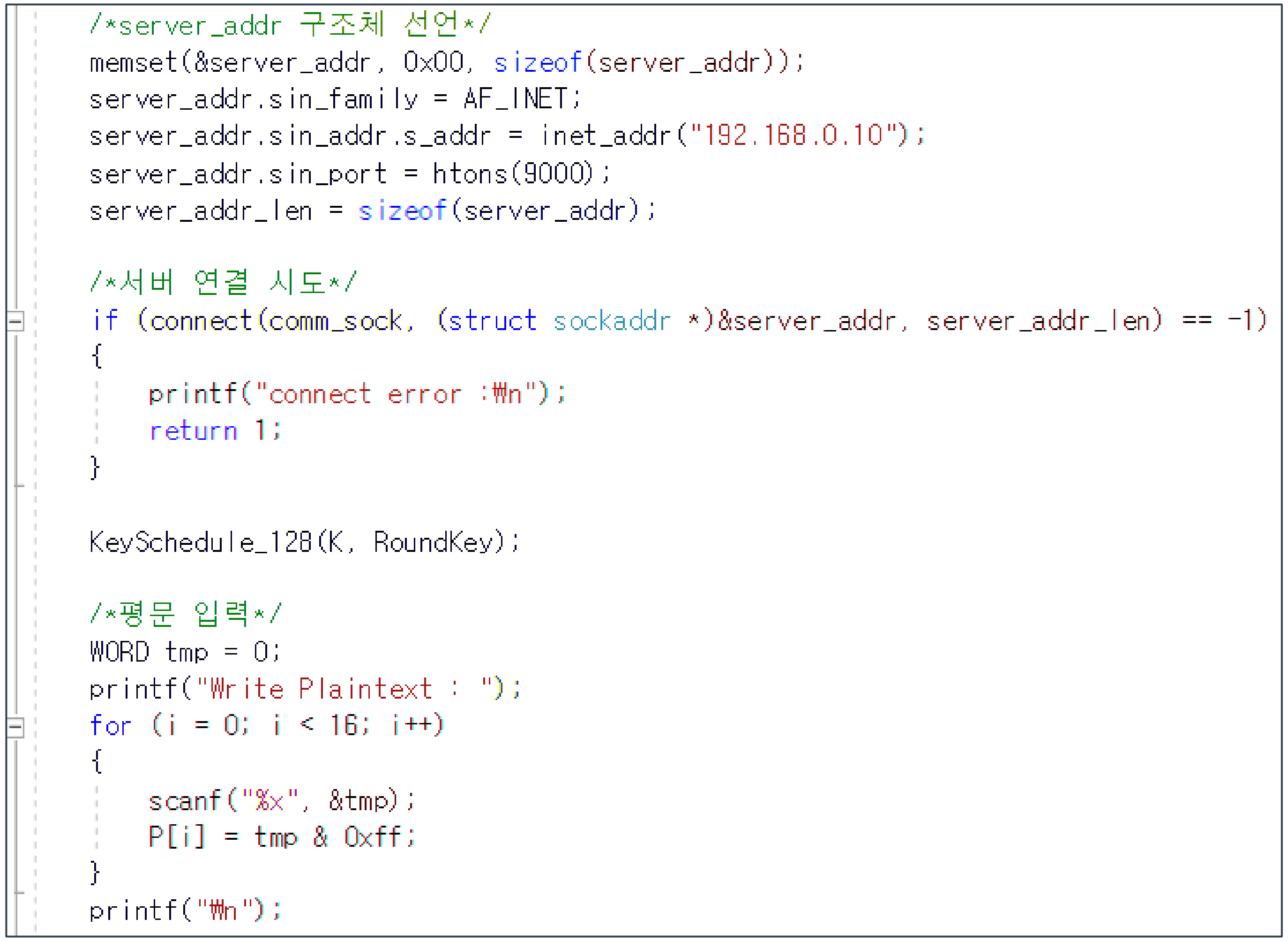
## client.c 코드



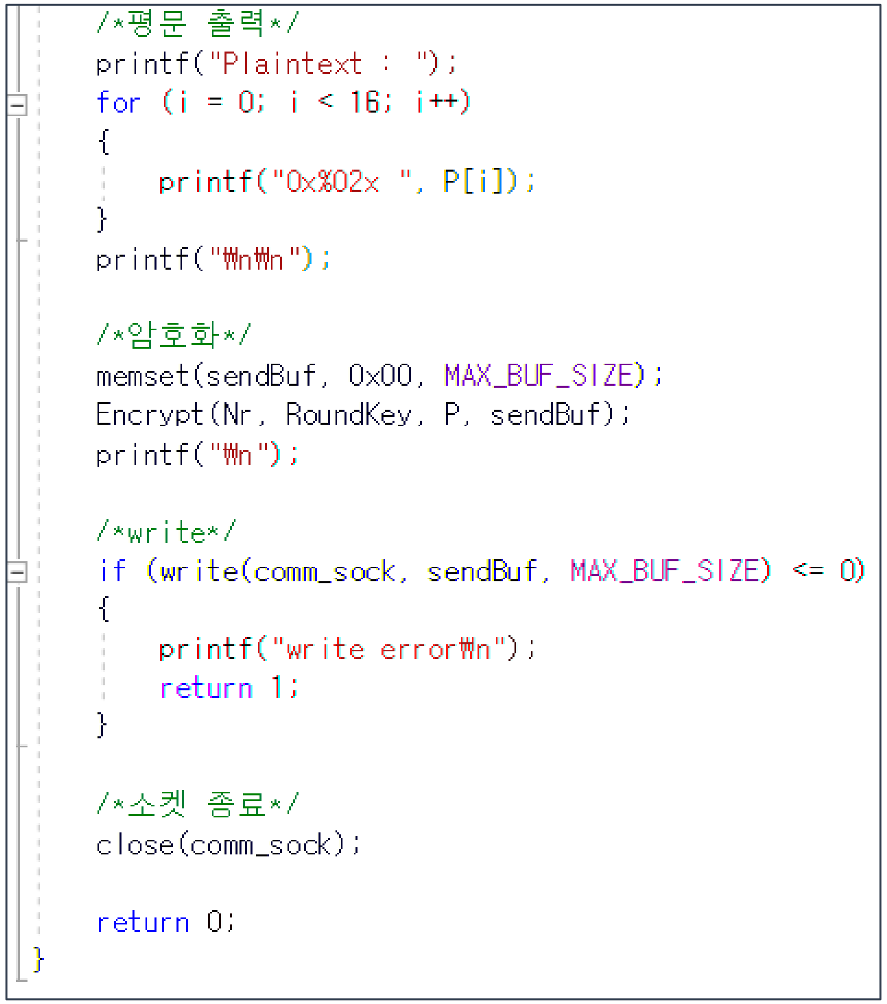
```c
#include <lea.h>
#include <unistd.h>

#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/stat.h>
#include <arpa/inet.h>
#include <sys/types.h>

#define MAX_BUF_SIZE    1024

int main()
{
    struct sockaddr_in server_addr;
    int comm_sock = 0;
    int server_addr_len = 0;
    int i, Nk, Nr;
    BYTE recvBuf[MAX_BUF_SIZE] = { 0, };
    BYTE sendBuf[MAX_BUF_SIZE] = { 0, };
    WORD RoundKey[144] = { 0, };
    BYTE K[16] =
    { 0x0f, 0x1e, 0x2d, 0x3c, 0x4b, 0x5a, 0x69, 0x78, 0x87, 0x96, 0xa5, 0xb4, 0xc3, 0xd2, 0xe1, 0xf0 };
    BYTE P[16] = { 0 };
    Nk = 16;
    Nr = 24;

    /*통신 소켓 만들기*/
    comm_sock = socket(PF_INET, SOCK_STREAM, 0);
    if (comm_sock == -1)
    {
        printf("error :₩n");
        return 1;
    }
```

# 소켓 클라이언트 프로그래밍 + LEA 암호화 | client.c

## client.c 코드



```c
    /*server_addr 구조체 선언*/
    memset(&server_addr, 0x00, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr("192.168.0.10");
    server_addr.sin_port = htons(9000);
    server_addr_len = sizeof(server_addr);

    /*서버 연결 시도*/
    if (connect(comm_sock, (struct sockaddr *)&server_addr, server_addr_len) == -1)
    {
        printf("connect error :₩n");
        return 1;
    }

    KeySchedule_128(K, RoundKey);

    /*평문 입력*/
    WORD tmp = 0;
    printf("Write Plaintext : ");
    for (i = 0; i < 16; i++)
    {
        scanf("%x", &tmp);
        P[i] = tmp & 0xff;
    }
    printf("₩n");
```



```c
    /*평문 출력*/
    printf("Plaintext : ");
    for (i = 0; i < 16; i++)
    {
        printf("0x%02x ", P[i]);
    }
    printf("₩n₩n");

    /*암호화*/
    memset(sendBuf, 0x00, MAX_BUF_SIZE);
    Encrypt(Nr, RoundKey, P, sendBuf);
    printf("₩n");

    /*write*/
    if (write(comm_sock, sendBuf, MAX_BUF_SIZE) <= 0)
    {
        printf("write error₩n");
        return 1;
    }

    /*소켓 종료*/
    close(comm_sock);

    return 0;
}
```