별걸다하는 IT/네트워크_소켓_통신

# [소켓 프로그래밍 C언어] 기본적인 클라이언트 프로그램 만들기 (리눅스 닉스 편 client) 관련 함수 및 소스코드

양햄찌(jhnyang) 2020. 4. 8. 00:01



안녕하세요!
저번 시간에는 기초적인 서버를 만들어서 제대로 동작하는지 테스트 하는 시간을 가졌었어요.ㅎㅎ
이제 서버가 있으니 오늘은 클라이언트를 만들어서 상호간 통신을 시켜보도록 합시다.

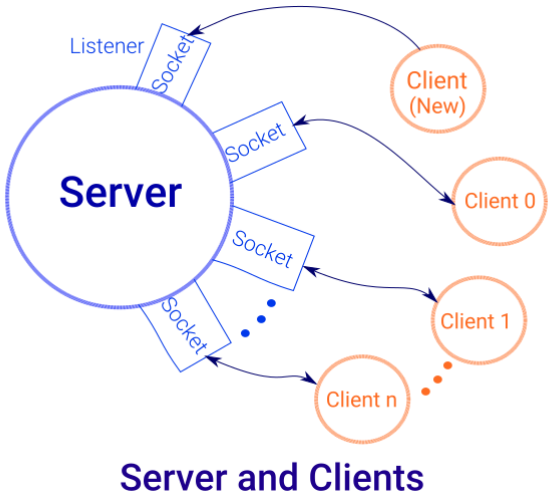
지난 포스팅이 궁금하신 분은 아래 링크를 참조해주세요!

[소켓 프로그래밍 C언어] 기본적인 서버 프로그램 만들기 (리눅스, 유…

jhnyang.tistory.com

저번 포스팅에서 서버가 가지고 있는 최소한의 구조를 살펴보았죠?
이번에는 클라이언트를 만드는데 있어서 필요한 구조를 서버 것과 같이 연결지어서 생각해보도록 합시다.

한 컴퓨터에 여러 개의 포트들이 있는데, 소켓은 결국 하나의 포트와 연결된다고 생각하면 됩니다.




Server and Clients

서버가 상황에 따라 프로세스 여러 개를 포트별로 띄울 수 있겠죠? 그럼 위 그림처럼 하나의 서버에 소켓이 여러개인 형태가 될거예
예를 들어 채팅 프로그램이 채팅 방 한개만 만들 수 있는건 아니잖아요. 나는 남자친구와 대화창을 하나 띄어놓을 수도 있고, 친구들
대화창을 띄어놓을 수도 있죠.



채팅 서버가 한개라 쳤을 때 15500포트에 연결된 소켓으로 남친과 대화를 하는 프로세스를 띄우고, 3700 포트에서 단체 대화를 나
있는거죠! 클라이언트도 마찬가지로 다수의 소켓을 가질 수 있습니다. 내가 채팅을 하면서, 웹서핑을 못하는건 아니잖아요? 다대다
다.

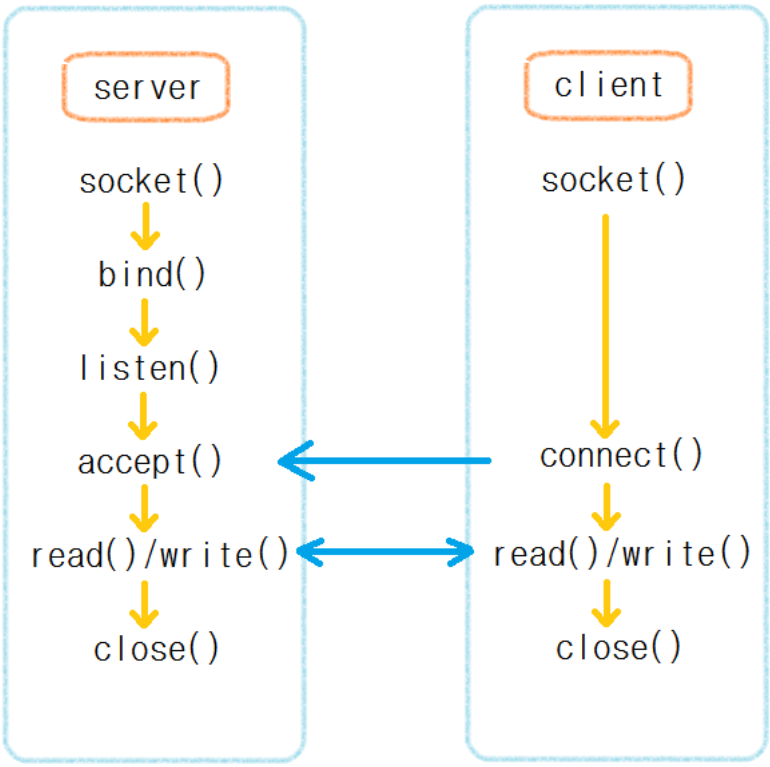
## 클라이언트를 만들기 위한 절차

**1. socket() 소켓 생성**
클라이언트도 마찬가지로 통신을 위한 소켓을 생성해줍시다.

**2. bind는 없다.**
클라이언트에서는 bind처리가 필요하지 않습니다.

왜 클라이언트는 바인드를 해주지 않을까요?
TCP연결 방식을 알면 이해하기 편한대, 클라이언트는 우리가 요청하는 것이기 때문에 어디다가 요청하는지 즉 주소 정보가 중요합

내가 게임을 다운로드 받으려고 하는데, 게임 사이트를 모르면 다운을 못받겠죠? 근데 게임 사이트(서버)는 나에게 접속해오는 수많
터들(클라이언트)의 정보를 알필요가 없어요. 이렇게 이해하시면 돼요.






즉 클라이언트는 서버의 주소를 찾아가야하는데 서버의 주소는 'IP+포트번호'입니다. 포트가 달라지면 전혀 다른 서비스가 되버리기 서버는 포트를 고정해줄 필요가 있는거예요.

그렇다고 클라이언트가 포트가없는건 아니고, 커넥트시 내부적으로 커널이 바인드처리를 합니다. 서버는 고정할필요가 있어서 우리 처리를 하는것!

**3. connect()**
커넥트! 말그대로 서버에 연결을 요청하는거예요. 그럼 대기중인 서버가 바쁘면 대기열에 넣었다가 때가 되면 accept! 수락을 해줍 렇게 서로 연결이 되면 본격적인 데이터 전송이 되는거죠.

**4. read() write()**
서버에서 데이터를 받기도 하고, 데이터를 전송하기도 하고, --> 다운로드 업로드 이죠 뭐.

**5. close()**
다 끝났으면 연결을 끊어줘야 합니다.

## 클라이언트 만드는데 필요한 함수

서버 때 배웠던 함수와 겹치는게 있는데 복습한다 생각하고 읽고 넘어갑시다.

**인자값:**

**int domain:**

어떤 영역에서 통신할 것인지에 대한 영역을 지정합니다 (protocol family지원)
올 수 있는 값 : AF_UNIX, IF_INET, IF_INET6 등
AF_UNXI는 프로세스끼리 통신할 때,
IF_INET은 IPv4, IF_INET6는 IPv6를 의미합니다.
우린 IPv4를 사용하니까 포스팅에선 IF_INET을 사용해줄거예요.

**int type:**

어떤 서비스 타입의 소켓을 생성할건지 적는건데요, 값 내용은 아래와 같습니다.
SOCK_STREAM(TCP), SOCK_DGRAM(UDP), SOCK_RAW(Raw 방식 TCP나 UDP를 거치지 않고 바로 IP계층 사용시)
우리는 TCP연결 지향형 통신을 생성하고자 하니 SOCK_STREAM을 적어주겠네요.

**int protocol:**

소켓에서 사용할 프로토콜.
IPPROTO_TCP: TCP방식
IPPROTO_UDP: UDP방식
0: type에서 미리 정해진 경우.

**리턴값:**

소켓을 가리키는 소켓 디스크립터를 반환합니다.
-1 소켓 생성 실패
0 이상의 값 : 소켓 디스크립터

**코드 예시:**

```
//TCP연결지향형이고 ipv4 도메인을 위한 소켓을 생성
serv_sock=socket(PF_INET, SOCK_STREAM, 0);
if(serv_sock == -1)
     printf("socket errorWn");
```

## 2. bind()

클라이언트에는 바인드 함수 필요 없음. 서버 포스팅에서 확인바람!

## 3. connect() 연결요청

서버로 연결요청하는 함수

```
#include <sys/socket.h>
int connect(int sockfd, struct sockaddr* serv_addr, socklen_t addrlen);
```

**인자값:**

**int sockfd:**

fd가 파일디스크립터의 약자예요. 즉 1번 함수 리턴값으로 받은 소켓 디스크립터를 여기 넣어주면 됩니다. 이제 이 소켓과 2번인자(
서버 주소 정보에 해당하는 소켓이 연결되어 데이터를 주고 받을 수 있는 기반을 마련해주는거죠.

**sockaddr* serv_addr**

서버의 주소정보를 넣어줍시다.

**socklent_t addrlen**

2번 인자의 크기를 넣어주면 돼요.

**리턴값:**

성공시 0, 실패시 -1.

## 5. close()

소켓을 닫고 통신을 종료하는 함수입니다.

```
#include <unistd.h>
int close(int sockfd);
```

**인자값:**

**int sockfd:**

## C언어 클라이언트(Client) 프로그램 작성하기

클라이언트에 필요한 요소들을 모두 알아봤으니 이제 기본 프로그램을 작성해줍시다.
server.c 만들었던 리눅스컴에다가 client.c를 작성해줘도 되지만 (그럼 서버 IP주소 = 클라이언트 IP주소 이므로 자기자신이라는 고 있는 127.0.0.1 루프백IP로 연결을 요청해도 됩니다.),
저는 가상머신을 하나 더 만들어서 클라이언트 소스는 따로 작성해줬어요. 실제 다른 컴퓨터에서 통신이 되나 확인해야하는거니까!

연결할 서버의 정보는 인자로 전달해줄거예요.

```
$client <IP주소> <port번호>

//같은 컴에서 작성하였을 경우
$client 127.0.0.1 <port번호>
```

<client.c 소스코드>

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
void error_handling(char* message);

int main(int argc, char* argv[])
{
        int clnt_sock;
        struct sockaddr_in serv_addr;
        char message[1024] = {0x00, };

        //TCP연결지향형이고 ipv4 도메인을 위한 소켓을 생성
        clnt_sock = socket(PF_INET, SOCK_STREAM, 0);
        if(clnt_sock == -1)
                error_handling("socket error");

        //인자로 받은 서버 주소 정보를 저장
        memset(&serv_addr, 0, sizeof(serv_addr));
        //서버주소체계는 IPv4이다
        serv_addr.sin_family = AF_INET;
        //서버주소 IP저장해주기(인자로 받은거 넘겨주기)
        serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
        //서버주소 포트번호 인자로 받은거 저장해주기
        serv_addr.sin_port = htons(atoi(argv[2]));

        //클라이언트 소켓부분에 서버를 연결!
        if(connect(clnt_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1)
                error_handling("connect error");

        //연결이 성공적으로 되었으면 데이터 받기
        if(read(clnt_sock, message, sizeof(message)-1) == -1)
                error_handling("read error");
        printf("Message from server :%sWn", message);

        //통신 후 소켓 클로우즈
        close(clnt_sock);
        return 0;
}
void error_handling(char* message)
{
        fputs(message, stderr);
        fputc('Wn', stderr);
        exit(1);
}
```

주석을 최대한 자세하게 작성했어요. 실제 업무에서 사용하려면 손봐야할 곳 투성이지만, 이렇게 가장 기초적인 클라이언트 프로그 어봤습니다.

▼혹시! 해당 소스코드 중 sockaddr 등, 주소체계를 저장하는 구조체에 대한 내용이 궁금하신 분은 아래 포스팅을 참조해주세요~!

[네트워크/소켓 C언어 프로그래밍] 주소 체계 저장 방법 sockaddr, s…

jhnyang.tistory.com

## 컴파일하기

```
ubuntu@server:~/client$ ls
client.c
ubuntu@server:~/client$ gcc client.c -o client
ubuntu@server:~/client$ ls
client  client.c
ubuntu@server:~/client$ s
```

짠! 컴파일 하고나니 기존에 없던 client 파일(초록색)이 생긴 것을 확인할 수 있어요.

## 서버와 클라이언트 접속 테스트 해보기

서버는 이전 포스팅에서 사용했던 정보를 사용하도록 할게요.
주인장 서버 주소정보
IP: 192.168.30.128
PORT: 3550
(서버 주소가 기억이 안난다면 서버로 간 후 hostname -I 명령어로 IP주소를 확인해보세요)

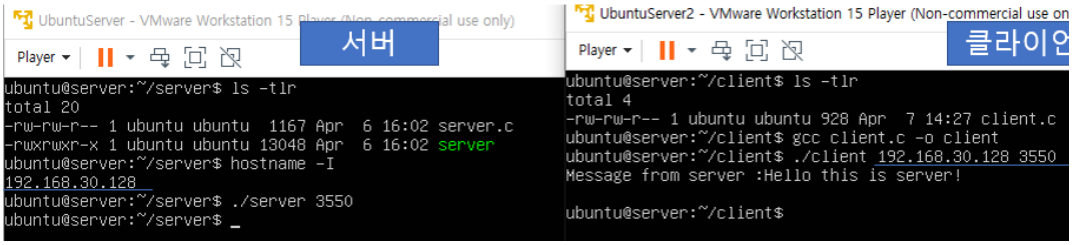
서버를 먼저 실행시켜둡시다. 서버가 켜져있어야 클라이언트가 접속 요청을 할테니까 말예요!

```
$./server 3550
```

```
ubuntu@server:~/server$ ./server 3550
```

그 다음 해당 서버에 연결요청하는 클라이언트를 실행해봅시다.

```
$./client 192.168.30.128 3550
```




결과

왼쪽 노드에는 서버 파일을, 오른쪽 노드에는 클라이언트 파일을 뒀고 두 노드는 IP가 다릅니다.
하지만! 클라이언트에 연결 명령어를 치는 순간, 서로 연결이 되어서 클라이언트가 서버로부터 메시지를 받아오는 것을 확인할 수 있

이로써~ 여러분들은 가장 간단한 서버와 클라이언트를 작성할 수 있습니다. (짝짝) 다음 시간에는 좀 더 심화된 서버/클라이언트에 가보도록 해요. 오늘은 여기까지~ 모두 고생하셨습니다.



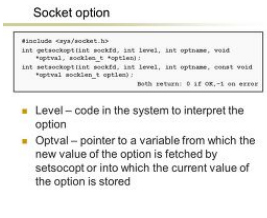
도움이 되셨다면 공감/댓글/광고보답 중 하나는 어떤가요?! 정보공유에 큰 활력이 됩니다 :)

'별걸다하는 IT > 네트워크_소켓_통신' 카테고리의 다른 글

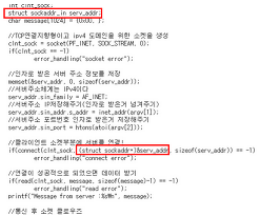
[네트워크/소켓 프로그래밍] setsockopt, getsockopt 소켓옵션 상세설명, 세부 옵션 설정하기, 소켓 버퍼 SO_SNDBUF & SO_KEEPALIVE (0)
[네트워크/소켓 C언어 프로그래밍] 주소 체계 저장 방법 sockaddr, sockaddr_in, sockaddr_in6 구조체 알아보기, 설명과 사용예제. (1)
[소켓 프로그래밍 C언어] 기본적인 서버 프로그램 만들기 (리눅스, 유닉스 편) server 관련 함수 및 소스코드 (12)
[네트워크] IP주소란? IPv4와 IPv6. IP주소는 왜 필요한가? 내 컴퓨터 IP 확인하는 법 (2)
[endian 2탄]리틀엔디안 vs 빅엔디안, 각 엔디안방식의 장단점, NBO(network byte order), CPU별 엔디안 차이. (7)

태그 #connect 연결 요청, #C언어 client.c, #네트워크 프로그래밍, #서버와 클라이언트 통신 테스트, #소켓 종료, #소켓프로그래
#클라이언트 구성도, #클라이언트 컴파일, #클라이언트 프로그램 소스코드, #클라이언트에 bind가 없는 이유

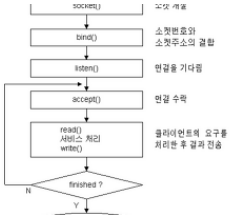
## '별걸다하는 IT/네트워크_소켓_통신' Related Articles



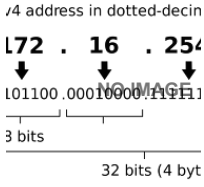

**[네트워크/소켓 프로그래밍] setsockopt, getsockopt ...**



**[네트워크/소켓 C언어 프로그래밍] 주소 체계 저장 방법...**




**[소켓 프로그래밍 C언어] 기본적인 서버 프로그램 만들기 (...**




**[네트워크] IP주소란? IPv6. IP주소는 왜 필**

오류(활성) E1696 파일 소스을(를) 열 수 없습니다. "arpa/inet.h" SocketChatting \SocketChatting.cpp 5
오류(활성) E1696 파일 소스을(를) 열 수 없습니다. "sys/socket.h" SocketChatting \SocketChatting.cpp 6
오류(활성) E0070 불완전한 형식은 사용할 수 없습니다. SocketChatting \SocketChatting.cpp 12
오류(활성) E0020 식별자 "socket"이(가) 정의되어 있지 않습니다. SocketChatting \SocketChatting.cpp 16
오류(활성) E0020 식별자 "PF_INET"이(가) 정의되어 있지 않습니다. SocketChatting \SocketChatting.cpp 16
오류(활성) E0020 식별자 "SOCK_STREAM"이(가) 정의되어 있지 않습니다.SocketChatting \SocketChatting.cpp 16
오류(활성) E0167 "const char *" 형식의 인수가 "char *" 형식의 매개 변수와 호환되지 않습니다.SocketChatting SocketChatting.cpp 18
오류(활성) E0070 불완전한 형식은 사용할 수 없습니다. SocketChatting \SocketChatting.cpp 21
오류(활성) E0070 불완전한 형식은 사용할 수 없습니다. SocketChatting \SocketChatting.cpp 23
오류(활성) E0020 식별자 "AF_INET"이(가) 정의되어 있지 않습니다. SocketChatting \SocketChatting.cpp 23
오류(활성) E0070 불완전한 형식은 사용할 수 없습니다. SocketChatting \SocketChatting.cpp 25
오류(활성) E0020 식별자 "inet_addr"이(가) 정의되어 있지 않습니다. SocketChatting \SocketChatting.cpp 25
오류(활성) E0070 불완전한 형식은 사용할 수 없습니다. SocketChatting \SocketChatting.cpp 27
오류(활성) E0020 식별자 "htons"이(가) 정의되어 있지 않습니다. SocketChatting \SocketChatting.cpp 27
오류(활성) E0020 식별자 "connect"이(가) 정의되어 있지 않습니다. SocketChatting \SocketChatting.cpp 30
오류(활성) E0070 불완전한 형식은 사용할 수 없습니다. SocketChatting \SocketChatting.cpp 30
오류(활성) E0167 "const char *" 형식의 인수가 "char *" 형식의 매개 변수와 호환되지 않습니다. SocketChatting \SocketChatting.cpp 31
오류(활성) E0020 식별자 "read"이(가) 정의되어 있지 않습니다. SocketChatting \SocketChatting.cpp 34
오류(활성) E0167 "const char *" 형식의 인수가 "char *" 형식의 매개 변수와 호환되지 않습니다. SocketChatting\SocketChatting.cpp 35
오류(활성) E0020 식별자 "close"이(가) 정의되어 있지 않습니다. SocketChatting \SocketChatting.cpp 39
오류 C1083 포함 파일을 열 수 없습니다. 'unistd.h': No such file or directory SocketChatting \SocketChatting.cpp 4

김미주 2021.04.06 04:11 댓글주소 수정/삭제

질문이 있습니다
client 코드에서는 왜 client에 대한 sockaddr_in 구조체에 대한 정보(port번호,ip주소)를 작성하지 않아도 되나요?
그러면 서버코드에서 sockaddr_in clntAdr을 읽어올때 클라이언트에 대한 소켓주소를 모르지 않나요?

양햄찌(jhnyang) 2021.04.06 16:00 신고 댓글주소

클라이언트와 서버의 관계를 생각해보시면 됩니다. 클라이언트에서 서버에 대한 아이피포트정보가 필요한 이유는, 클라
서버에 요청을 하는 관계이기 때문에 어디로 가야할지 목적지 정보를 지정해줘야하기 때문이예요. 애초에 네트워크단 패
발지의 IP정보등이 같이 포함되어 오기 때문에 서버가 요청을 받으면 클라이언트의 IP주소를 모르지는 않습니다. 클라이
로그램 입장에서 굳이 IP정보를 정의해야할 이유가 없을 뿐이죠.

CdspaceNoob 2021.09.29 08:23 댓글주소 수정/삭제

다음 포스팅이 없어서 아쉬워요ㅠㅠ
하지만 이번 포스팅도 짱이었습니다! 감사합니다

이름 암호 Secret

여러분의 소중한 댓글을 입력해주세요.

댓글달기