

양햄찌가 만드는 세상

인기글

- [EXCEL] 엑셀 드롭다운 만드는 법, 목록 선택하게 하기
- 하드디스크 인식 해결- HDD 추가 장착 후 D드라이브 생기지 ...
- [리눅스, 유닉스] vi (vim) 편집기 기본 사용법, 명령어, ...
- 자주 사용하는 비주얼스튜디오 단축키 정리 (Visual Stud...
- [SQL] 정렬하기 order by 쿼리 사용법 1, 2 des...
- [ORACLE, MYSQL, SQL] CREATE TABLE 테...
- 자바(JAVA) 다운로드 및 설치하기, 환경설 정 세팅하는 법, ...
- [DB Sql developer 예러] The Network A...
- [VSC 비주얼스튜디오코드] VScode에서 C/C++ 디버깅하...
- [C/C++언어] sprintf 함수와 fprintf 함수 사용법...

별걸다하는 IT/네트워크_소켓_통신

[소켓 프로그래밍 C언어] 기본적인 서버 프로그램 만들기 (리눅스, 유닉스) server 관련 함수 및 소스코드

양햄찌(jhnyang) 2020. 4. 6. 16:57

Node.js와 TypeScript의 꿀조합

백엔드에서 풀스택 개발자가 되고 싶은 여러분을 위해 강의

패스트캠퍼스

안녕하세요~!
오늘은 기본적인 TCP 서버 프로그램을 작성해볼게요.
소켓 프로그래밍으로 간단한 채팅 서버를 만들어볼 생각인데, 찬찬히 진행해보도록 합니다.

소켓 프로그래밍이란?

오늘날 모든 컴퓨터는 소통을 하죠!
보통 내가 먼저 다른 노드(컴퓨터)에 요청을 하면 나는 고객(?)이니까 클라이언트가 됩니다.
그리고 반면에 다른 컴퓨터가 똑똑 두드리면, 신호를 받아서 데이터를 내려주는 역할을 서버라고 간략하게 정의해볼 수 있어요. 네트워크 연결되어 있는 서로 다른 두 컴퓨터가 데이터를 주고받을 수 있도록 하는 것이 네트워크 프로그래밍, 즉 소켓 프로그래밍입니다.

소켓 socket?

소프트웨어적인 데이터 송수신 방법을 이미 운영체제에서 제공해주고 있는데 이게 '소켓(Socket)'입니다. 모든 운영체제에서 지원! 이는 물리적으로 연결된 네트워크상에서의 데이터 송수신에 사용할 수 있는 소프트웨어적인 장치를 의미합니다.

비전공자를 위한 코딩 강의 0원

코딩 공부 해야지. 작심삼일은 이제 그만. 하루 10분 코 공부하고 원하는 목표 달성하기.

AWS 데이터 엔지니어링 정주행

틀은 잘 몰라도 복붙으로 오픈소스만 사용할 수 있으면 수강 가능
패스트캠퍼스

그리고 이미 운영체제에서 제공해주고 있는 함수로 우리는 원하는 통신 프로그램을 소스로 짜서 실현시킬 수 있다는거~

서버를 만들기 위한 절차

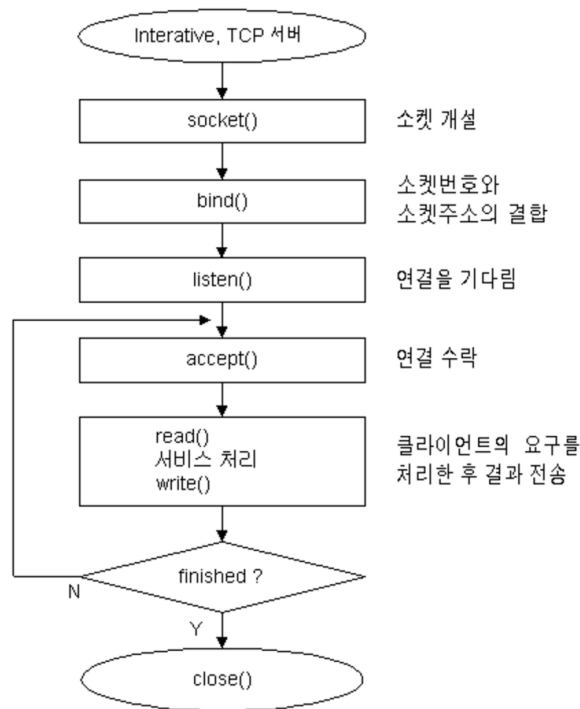


그림 2-11 Iterative 모델의 TCP(연결형) 서버 프로그램 작성 절차

1. socket() 소켓 생성

일단 다른 노드(컴퓨터)들과 통신하려면, 창구가 필요하잖아요. 소켓을 생성해줍니

2. bind()

그 다음 bind라는 걸 해야하는데요. 내가 만드려는 서버의 ip주소와 포트를 담당하는 역할을 수행해요.

3. listen()

그럼 이제 다른 컴퓨터에서 노크를 두드려 해줄 수 있도록 대기상태로 만들기

4. accept()

만약 어떤 클라이언트로부터 연결 요청이 수락해주는 함수

5. read() write()

이제 두 컴퓨터가 연결이 되었으니 필요한 데이터를 전송하고 받고~

6. close()

끝났으면 연결을 끊어줍니다.

서버 만드는데 필요한 함수

이제 해당 순서에 필요한 함수들을 살펴봅시다.

1. socket() 소켓 생성

소켓 생성하는 함수입니다.

```
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```

인자값:

int domain:

어떤 영역에서 통신할 것인지에 대한 영역을 지정합니다 (protocol family지원)

올 수 있는 값 : AF_UNIX, AF_INET, AF_INET6 등

AF_UNIX는 프로세스끼리 통신할 때,

AF_INET은 IPv4, AF_INET6는 IPv6를 의미합니다.

우린 IPv4를 사용하니까 포스팅에선 AF_INET을 사용해주려요.

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.

코드 예시:

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

2. bind()

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

그 다음 소켓이랑 서버의 정보를 묶어주는 함수입니다.

이걸 왜하나, 다른 외부의 컴퓨터가 서버에 연결하려고 요청을 했어요, 물론 IP주소를 기반으로 찾았겠죠. 근데 통신을 위해선 소켓 번호를 알아야하는데 IP주소를 안다고 애를 알 수 있는게 아니잖아요. 그래서 두개를 묶어주는 작업을 하는겁니다.

인자값:

int sockfd:

fd가 파일디스크립터의 약자예요. 즉 1번 함수 리턴값으로 받은 소켓 디스크립터를 여기 넣어주면 됩니다.

struct sockaddr *myaddr:

서버의 IP주소를 넣어줘요.

socklen_t addrlen:

주소 길이를 넣어줍니다.

리턴값:

성공시 0, 실패시 -1.

코드 예시:

```
//소켓과 서버 주소를 바인딩
if(bind(serv_sock, (struct sockaddr*) &serv_addr, sizeof(serv_addr))==-1)
    printf("bind error");
```

3. listen()

이제 어떤 컴퓨터로부터 요청이 와도 수락할 수 있게 대기상태에 들어가는 함수입니다.

```
#include <sys/socket.h>
int listen(int sockfd, int backlog);
```

인자값:

int sockfd:

2번 bind함수의 첫 인자와 같습니다. 소켓 디스크립터를 여기 넣어줍니다.

int backlog:

연결 대기열의 크기를 지정합니다. listen이라는 게 결국 연결요청 소켓이 대기하는 연결 대기열을 생성하는 거거든요. 네트워크 상태와 서비스 종류에 따라서 달라집니다.

리턴값:

성공시 0, 실패시 -1.

4. accept()

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

서버 소켓에다가 이제 클라이언트를 연결하는 함수입니다.

그래서 두 번째와 세 번째 주소 관련 인자에는 클라이언트 정보가 들어가게 돼요.

인자값:

int sockfd:

2번 bind함수의 첫 인자와 같습니다. 서버 소켓 디스크립터를 여기 넣어줍니다.



C언어 서버 프로그램

전체 흐름과 필요한 함수들을 모두 살펴봤으니 아주아주 기본적인 서버 프로그램을 작성해봅시다.
가장 베이직한 거니까, 서버 IP는 임의적으로 사용 가능한걸 할당하게 하고, port만 실행시 인자로 받을 거예요.

이런식으로 명령어를 치게 되면 3550포트에 서버 프로세스를 켜놓는거죠.

```
$server 3550
```

<server.c 소스코드>

```
#include <stdio.h>
#include <stdlib.h> //atoi를 사용하려면 있어야함
#include <string.h> // memset 등
#include <unistd.h> //sockaddr_in, read, write 등
#include <arpa/inet.h> //htnl, htons, INADDR_ANY, sockaddr_in 등
#include <sys/socket.h>
void error_handling(char * message);

int main(int argc, char* argv[])
{
    int serv_sock;
    int clnt_sock;

    //sockaddr_in은 소켓 주소의 틀을 형성해주는 구조체로 AF_INET일 경우 사용
    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr; //accept함수에서 사용됨.
    socklen_t clnt_addr_size;

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

    //주소를 초기화한 후 IP주소와 포트 지정
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family=AF_INET; //타입: ipv4
    serv_addr.sin_addr.s_addr=htonl(INADDR_ANY); //ip주소
    serv_addr.sin_port=htons(atoi(argv[1])); //port

    //소켓과 서버 주소를 바인딩
    if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
        error_handling("bind error");

    //연결 대기열 5개 생성
    if(listen(serv_sock, 5)==-1)
        error_handling("listen error");

    //클라이언트로부터 요청이 오면 연결 수락
    clnt_addr_size = sizeof(clnt_addr);
    clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_addr_size);
    if(clnt_sock==-1)
        error_handling("accept error");

    /*-----데이터 전송-----*/
    char msg[] = "Hello this is server!\n";
    write(clnt_sock, msg, sizeof(msg));

    //소켓들 닫기
    close(clnt_sock);
    close(serv_sock);

    return 0;
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

위 IP주소 설정해주는 부분에서

INADDR_ANY라는게 있는데 '컴퓨터에 존재하는 랜카드 중 사용 가능한 랜카드의 IP주소를 사용하라'라는 의미입니다.

위 소스같은 경우에는 일단 하나의 클라이언트와 연결이 되면, Hello this is server라는 메시지를 클라이언트에 내려주고 통신을 종료요. 정말 기본요소들만 갖고 있는셈이죠.



비전공자를 위한 코딩 강의 0원

코딩 공부 해야지. 작심삼일은 이제 그만. 하루 10분 코 공부하고 원하는 목표 달성하기.

코드잇 codeit

▼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 |

컴파일

```
gcc server.c -o server //gcc컴파일러일 경우
cc server.c -o server //gcc가 아닐 경우 기본
```

다 작성 하였으면 위 명령어로 컴파일을 해줍니다. server.c를 컴파일해서 server라는 실행파일을 만들라는 명령어예요.

```
ubuntu@server:~$ gcc server.c -o server
Command 'gcc' not found, but can be installed with:
sudo apt install gcc

ubuntu@server:~$ cc server.c -o server
Command 'cc' not found, but can be installed with:
sudo apt install gcc
sudo apt install clang
sudo apt install pentium-builder
sudo apt install tcc

ubuntu@server:~$ sudo apt install gcc
```

gcc가 없을 경우 설치

컴파일러가 없을 경우에는 sudo apt install gcc 명령어를 통해 설치해주면 됩니다.

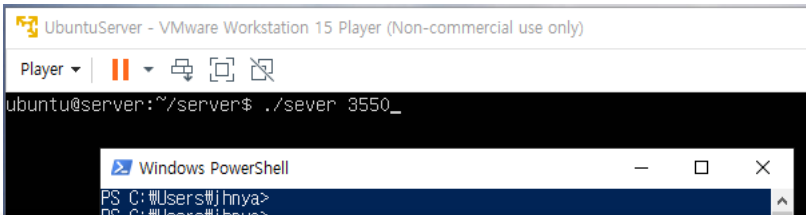
```
ubuntu@server:~/server$ cc server.c -o server
ubuntu@server:~/server$ ls -ltr
total 20
-rw-rw-r-- 1 ubuntu ubuntu 1167 Apr  6 15:55 server.c
-rwxrwxr-x 1 ubuntu ubuntu 13048 Apr  6 15:55 server
```

컴파일 후

그리고 컴파일 하면 .c 소스파일 외에 server 실행파일이 생깁니다.

서버 실행후 클라이언트 접속해보기

서버를 실행한 후에 telnet으로 접속 시도로 결과를 확인해볼게요.



[소켓 프로그래밍 C언어] 기본적인 서버 프로그램 만들기 (리눅스, 유닉스 편) server 관련 함수 및 소스코드

저의 경우 리눅스 가상머신에다가 server를 포트 3550에 실행시켜줬습니다. 3550포트는 임의의 포트입니다. telnet으로 서버에 3550 포트에 접속하면 됩니다.

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

서버 실행하면

그 다음 컴퓨터의 cmd창이나 파워셸에다가 telnet 연결을 시도했어요.

즉 리눅스 가상머신은 서버가! 내가 실행하고 있는 윈도우 데스크탑은 클라이언트가되는거죠!

```
PS C:\Users\jhnya> telnet 192.168.30.128 3550
```

서버 접속 시도

참고로 서버의 ip주소는 ifconfig라는 명령어로 확인해볼 수 있습니다.

```
root@server:~# hostname -I
192.168.30.128
```

hostname

또는 l 옵션과 함께 hostname 명령어로 IP주소를 확인할 수 있어요.

실행 결과,

```
Windows PowerShell
Hello this is server!

호스트에 대한 연결을 잃었습니다.
PS C:\Users\jhnya>
```

'Hello this is server!'라는 응답을 정상적으로 수신한 후 통신이 종료되는 것을 확인할 수 있어요.

오늘은 간단한 서버 프로그램을 짜봤어요. ㅎㅎ 다음 포스팅에는 기본적인 클라이언트 프로그램을 작성해보도록 할게요 ㅎㅎ 지금 클라이언트 프로그램이 없어서 telnet을 이용해 테스트 해봤지만, 다음 시간엔 가상머신 두 대를 켜놓고 서버와 클라이언트 간의 통신을 해보도록 합시다.

▼다음 포스팅 바로가기!

[소켓 프로그래밍 C언어] 기본적인 클라이언트 프로그램 만들기 (리눅스...
jhnyang.tistory.com

도움이 되셨다면 좋아요, 댓글, 광고보답은 어떤가요?! 정보 공유에 큰 활력이 됩니다. 감사합니다.


Node.js를 2가지 언어로 학습

백엔드에서 풀스택 개발자가 되고 싶은 여러분을 위해 강의

패스트캠퍼스

- '별걸다하는 IT > 네트워크 소켓 통신' 카테고리의 다른 글
- [네트워크/소켓 C언어 프로그래밍] 주소 체계 저장 방법 sockaddr, sockaddr_in, sockaddr_in6 구조체 알아보기, 설명과 사용예제. (1)
 - [소켓 프로그래밍 C언어] 기본적인 클라이언트 프로그램 만들기 (리눅스, 유닉스 편 client) 관련 함수 및 소스코드 (7)
 - [네트워크] IP주소란? IPv4와 IPv6. IP주소는 왜 필요한가? 내 컴퓨터 IP 확인하는 법 (2)
 - [endian 2탄]리틀엔디언 vs 빅엔디언, 각 엔디언방식의 장단점, NBO(network byte order), CPU별 엔디언 차이. (7)
 - [네트워크 network] OSI 7계층 모델 완벽 이해- 이유를 알고 개념을 이해하고 전체 보기(7 layer) (19)

오류(활성) E0020 식별자 "listen"이(가) 정의되어 있지 않습니다. SocketChatting 35
오류(활성) E0167 const char * 형식의 인수가 "char *" 형식의 매개 변수와 호환되지 않습니다. SocketChatting 36
오류(활성) E0070 불완전한 형식은 사용할 수 없습니다. SocketChatting 39
오류(활성) E0020 식별자 "accept"이(가) 정의되어 있지 않습니다. SocketChatting 40
오류(활성) E0167 const char * 형식의 인수가 "char *" 형식의 매개 변수와 호환되지 않습니다. SocketChatting 42
오류(활성) E0020 식별자 "write"이(가) 정의되어 있지 않습니다. SocketChatting 46
오류(활성) E0020 식별자 "close"이(가) 정의되어 있지 않습니다. SocketChatting 49
오류 C1083 포함 파일을 열 수 없습니다. 'unistd.h': No such file or directory SocketChatting 4



양행찌(jhnyang)

2021.02.21 17:37

신고

댓글주소

넵 해당 소스코드는 리눅스 운영체제에서 지원하는 라이브러리로 작성한 코드로, 윈도우 소켓 프로그래밍을 할 경우에는 winsocket과 같은 다른 라이브러리를 인클루드해주셔야 합니다 π_π 하지만 라이브러리명만 조금 차이가 있을뿐 윈!눅스나 유사합니다 ㅎㅎ



CdspaceNoob

2021.09.29 08:00

댓글주소 수정/삭제

굉장히 자세하고 이해하기가 쉽네요 감사합니다
잘 따라해보겠습니다!




치킨

2022.05.12 15:36

댓글주소 수정/삭제

혹시 mac os에서 하는 방법은 없나요?




ㅇㅇ

2022.05.12 21:30

댓글주소

대신 답변드립니다. 아마존 AWS EC2에서 리눅스 서버 키시고 vscode로 원격 접속 하시면됩니다



익명

2022.08.14 16:58

댓글주소 수정/삭제

비밀댓글입니다

이름

암호

☐ Secret

여러분의 소중한 댓글을 입력해주세요.

댓글달기