

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 - 이상훈

gcccompil3r@gmail.com

학생 - 이우석

colre99@naver.com

[3/29 (목) - 26 일차]

프로토콜 (Protocol) : 시스템과 애플리케이션 프로그램에서 정보를 교환할 수 있도록 하는 메시지 형식 및 프로시저에 대한 규칙세트.

(=복수의 컴퓨터 사이나 중앙컴퓨터와 단말기 사이에서 데이터 통신을 원활하게 하기위해 필요한 통신규약)

[TCP/IP Protocol]

1. 전송 제어 프로토콜(TCP)과 인터넷 프로토콜(IP)을 의미
2. TCP/IP 프로토콜 조합(Protocol suit)을 의미
3. 통신 소프트웨어의 기반이 되는 모든 프로토콜을 말함

이론용: OSI 7 Layer (7 계층: 응용, 표현, 세션, 전송, 네트워크, 데이터링크, 물리)

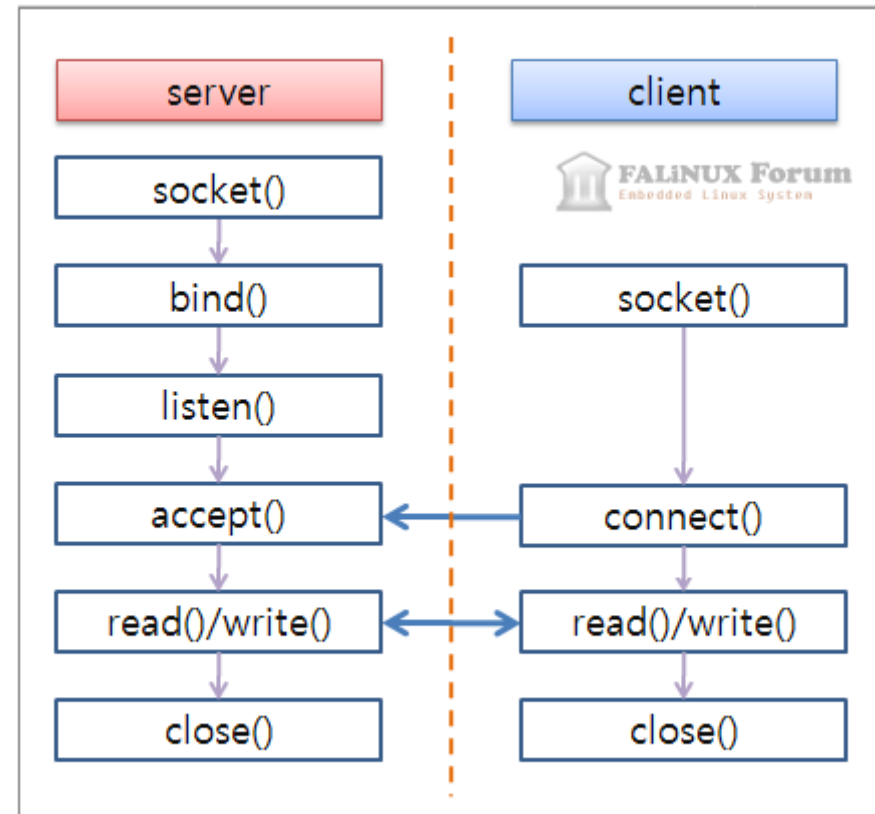
구현용: TCP/IP (4 계층: 응용, 전송, 인터넷, 네트워크)

- TCP 는 거의 SOCK_STREAM 을 사용
- UDP 는 거의 SOCK_DGRAM 을 사용

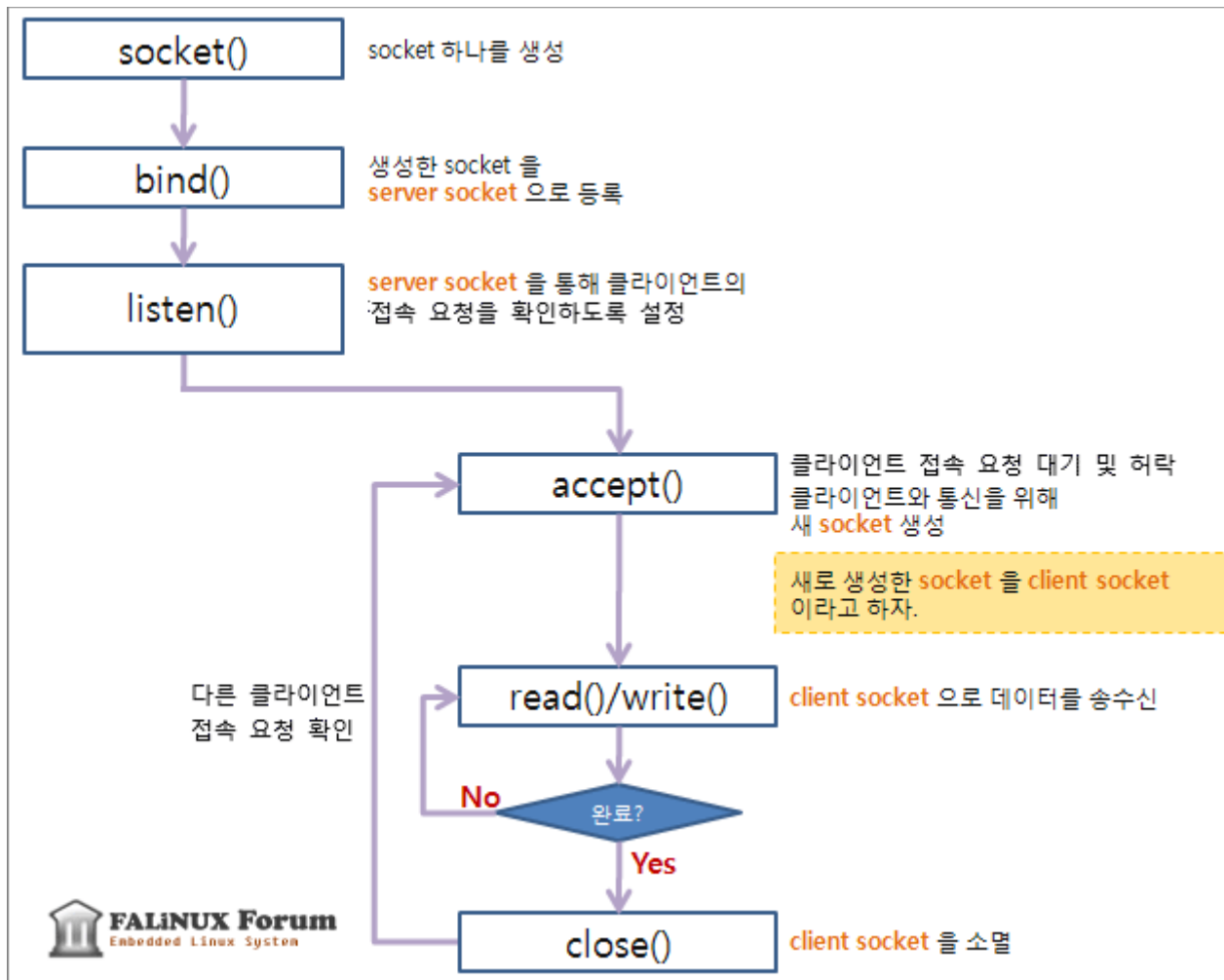
-TCP SOCK_STREAM 은
프로토콜 기반인 커넥션.
이 커넥션은 2 개의 부분을 가지며
다른 한 부분이 제거하거나 에러가
생길때 까지 커넥션.

-UDP SOCK_DGRAM 은 프로토콜
기반인 datagram. 하나의 datagram 을
보내고 하나의 답변만 받는다.
그리고나서 커넥션은 제거됨.

TCP/IP 통신 함수 사용 순서



[서버 프로그램에서 사용해야할 함수와 순서]



[IP 의 종류]

1. 공인 IP (WAN 통신)
2. 사설 IP (공유기)

[NAT] - 4 종류

NAT 의 용도 : 한 아이피에서 여러개를 처리하는것.

NAT 의 필요성 : 내부 망에서는 사설 IP 를 사용하고, 외부 인터넷 망 접속시 사설 IP 를
공인 IP 주소로 변환하여 사용함 으로서 공인 IP 부족문제를 해결.

[MAC 통신]

하드웨어 장치의 고유번호(랜카드의 식별자)

IP 를 보는게 라우터 / MAC 주소를 보고, 보내는게 스위치 / 해킹하면 MAC 에 기록남음

<서버 예제>

[basic_server.c]

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

typedef struct sockaddr_in si;
typedef struct sockaddr * sap;
//si 와 sap 는 사용자 변수.

void err_handler(char *msg)
{
    fputs(msg, stderr);
    fputc('\n', stderr);
    exit(1);
    //stderr: 표준에러
}
```

[illegible]

```
if(serv_sock == -1)    //에러 났으면 아래 출력
    err_handler("socket() error");
```

//memset:메모리를 상수 바이트로 채워주는 함수.

```
memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
```

//서버 어드레스 구조체 si.

```
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY); =>어떤 아이피주소든 받겠다( INADDR)
```

```
serv_addr.sin_port = htons(atoi(argv[1])); =>어떤 서비스번호를 열것인가. 서비스번호=포트번호
```

//htons:(Host to network short)=Host 시스템에서 Network 로 short 형 데이터를 보낼때 바이트오더를 바꿔주는 함수.

//htonl:(Host to network long)=long 형 데이터의 바이트 오더를 바꿔주는 함수.

=>bind()는 생성한 socket 을 server socket 으로 등록.

```
if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1) =>bind , 바인딩 = 서버에 아이피 주소 세팅
    err_handler("bind() error");    //
```

=>listen()은 server socket 을 통해 클라이언트의 접속요청을 확인하도록 설정.

```
if(listen(serv_sock, 5) == -1) =>listen: 서버에서 5 명 받겠다
```

```
err_handler("listen() error"); =>listen 에서 클라이언트 기다림. 클라이언트는 들어온걸 기다리는거고.
```

=>accept()은 클라이언트 접속요청 대기 및 허락. 클라이언트와 통신을 위해 새 socket 생성.

=>새로 생성한 socket 을 client socket 이라고 하자.


```
clnt_addr_size = sizeof(clnt_addr);
clnt_sock = accept(serv_sock, (struct sockaddr *)&clnt_addr, &clnt_addr_size);
    =>서버 소켓이 클라이언트 기다림. 클라이언트의 주소 사이즈도 받음.
if(clnt_sock == -1)          =>클라이언트 받으면 클라 주소도 받음. (= 누가 접속했는지 알수있다)
    err_handler("accept() error");

write(clnt_sock, msg, sizeof(msg)); =>클라이언트 소켓에 쓰고있다.
=>소켓은 원격에 있는파일. 원격으로 동기화(세마포어 만듬)

close(clnt_sock); =>클라이언트 소켓 소멸          // 파일 디스크립터 받음.(읽기쓰기 됨.)
close(serv_sock); =>서버 소켓 소멸. 원격에있는 클라이언트한테 쓰기. 메세지 적으면 클라이언트한테.

return 0;
}
```

[basic_client.c]

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
```

```
typedef struct sockaddr_in si;
typedef struct sockaddr * sap;
```

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

```
int main(int argc, char **argv)
```

```

{
    int sock;
    int str_len;
    si serv_addr;
    char msg[32];

if(argc != 3) => 3 인건 ip 주소 어디 알아야 하니깐.
    {
        printf("use: %s <IP> <port>\n", argv[0]);
        exit(1);
        //SOCK_STREAM : TCP/IP 프로토콜 이용
        //SOCK_DGRAM : UDP/IP 프로토콜 이용
    }

    //1.도메인인자, 2.소켓형태, 3.초기화값
    //1.IPv4 인터넷 프로토콜을 사용 (프로토콜 체계))
    //2.TCP 프로토콜 이용

```

sock = socket(PF_INET, SOCK_STREAM, 0); // sock 에 집어넣음.
 //소켓하면 파일디스크립터 얻음.(네트워크상의) - 내가 송신할수있는.

```

if(sock == -1)
    err_handler("socket() error");

```

=>메모리를 상수바이트로 채워주는 상수함수 =>7777 서버 어드레스 원격 포트
 memset(&serv_addr, 0, sizeof(serv_addr)); =>서버 어드레스를 초기화 한다.

```
serv_addr.sin_family = AF_INET; //주소 체계
serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
serv_addr.sin_port = htons(atoi(argv[2]));
```

=>sock 이 자신의 fd 를 서버 어드레스

=>어디에 접속할지는 포트번호로 안다.

```
if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1) =>커넥트 하면 listen 에서 받는다.
```

```
err_handler("connect() error"); =>서버랑 클라이언트랑 통신된다.
```

=>write 하니깐 메세지가능 .

```
str_len = read(sock, msg, sizeof(msg) -1); =>쓴걸 읽는다. 버퍼해서 fd 로 들어감. sock 으로 전달,
```

=>read 는 블로킹 함수. 파일디스크립터 sock 이 읽는다. msg 로 가는거다. sock 에 메세지 들어있음.

```
if(str_len == -1)
err_handler("read() error!");
```

```
printf("msg from serv: %s\n", msg);
close(sock);
```

```
return 0;
}
```

위 예제들의 실행결과를 보려면 우선 각각 C 코드들을 컴파일 해준다.

```
gcc -o serv basic_server.c
```

```
gcc -o clnt basic_client.c
```

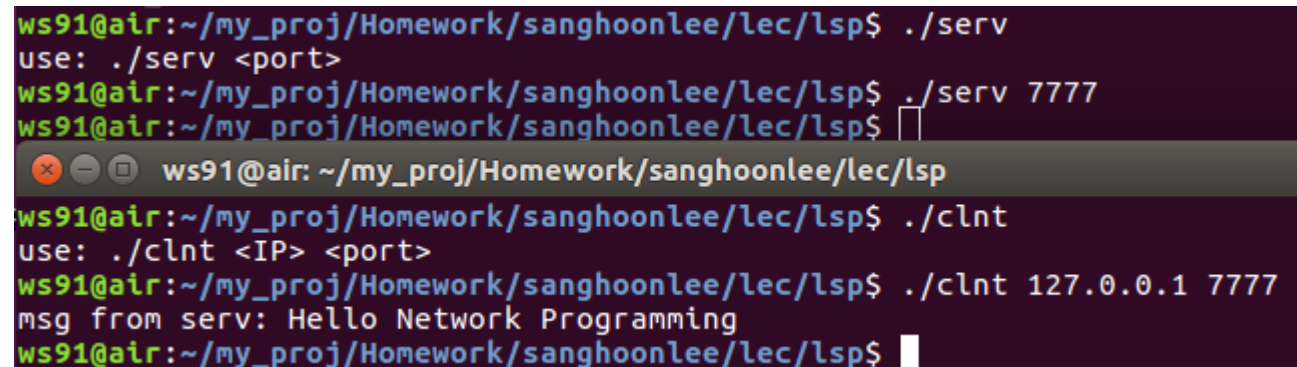
각 각 컴파일 해준후, 터미널 2 개를 추가로 더 열어준다.

A 터미널에선 `./serv` , B 터미널에선 `./clnt` 각각 실행파일을 실행해 준 후,

A 터미널에서 `./serv 7777` 이라고 포트번호와 함께 실행한다.

B 터미널에선 `./clnt 127.0.0.1 7777` 이라고 ip 주소, 포트번호 와 함께 실행해준다.

결과값:



```
ws91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ ./serv
use: ./serv <port>
ws91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ ./serv 7777
ws91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ 
ws91@air: ~/my_proj/Homework/sanghoonlee/lec/lsp
ws91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ ./clnt
use: ./clnt <IP> <port>
ws91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ ./clnt 127.0.0.1 7777
msg from serv: Hello Network Programming
ws91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$
```

위와같이 서버로 부터 메세지가 출력되는걸 확인 할 수 있다.