요즘 통신을 배우고 있는데
기본적인 용어의 정의조차 몰라서 모르는거 위주로 조사,공부 했습니다.

**********문자열 공부를 해보자**********

```
#include <stdio.h>
int main()
{
    char sentence_1[4] = {'P', 's', 'i', '\0'};
    char sentence_2[4] = {'P', 's', 'i', 0};
    char sentence_3[4] = {'P', 's', 'i', NULL};
    char sentence_4[4] = {"Psi"};

    printf("sentence_1 : %s \n", sentence_1); // %s 를 통해서 문자열을 출력한다.
    printf("sentence_2 : %s \n", sentence_2);
    printf("sentence_3 : %s \n", sentence_3);
    printf("sentence_4 : %s \n", sentence_4);

    return 0;
}
```

1.문자열을 사용할때 마지막에 NULL 값을 넣어야한다. 이유는 NULL 이 나오기 전까지 출력하라는 뜻.
NULL 을 사용안하면 '이 s 문자열은 3 개의 문자인데,출력해줘'라고 말해야되는데 그럴필요없이
'널이 나올때까지 걍 출력해' 라고 할 수가 있어서 사용.

2.%c 는 한 문자만을 출력. %s 는 문자열 출력.

3.""는 문자열(한 개 이상의 문자)를 지정
  "는 한개의 문자를 지정

!!문자의 개수를 세자.!!

```
#include <stdio.h>
int str_length(char *str);
int main()
{
    char str[]={"What is your name?"};

    printf("이 문자열의 길이 : %d \n", str_length(str));

    return 0;
}
int str_length(char *str)
{
    int i = 0;
    while(str[i])
```

```
    {
        i++;
    }

    return i;
}
```

******fputs********
int fputs (const char * str, FILE * stream);

/str 이 가리키는 문자열을 stream 에 쓴다./

*str

스트림에 쓰여진 널 문자로 끝나는 문자들의 배열

*stream

문자열을 쓸 스트림의 FILE 객체를 가리키는 포인터

?stream 은 무엇일까?

->기본적으로 제공되는 '표준 스트림'.(에러 스트림 포함)

stdin 표준입력 스트림 - 키보드 대상으로 입력
stdout 표준출력 스트림 - 모니터 대상으로 출력
stderr 표준에러 스트림 - 모니터 대상으로 출력

*리턴값

쓰기를 성공한다면 음이 아닌 수가 리턴된다.
오류가 발생한다면 EOF 를 리턴한다.

```
int main ()
{
    char str[] = "string to print \n";

    fputs(str, stdout);

    return 0;
}
```

**********sockaddr**********
sockaddr_in 이라는 구조체가 정의되어 있다. 보통 타입데프해서 si 라고함.
-> bind 함수에 주소정보를 전달 할 때 사용.

3 개의 멤버를 가지고있다.
1.sin_family
2.sin_addr

3.sin_port

!!멤버에 대해 분석해보자!!

sin_family

->주소체계 정보를 저장.

AF_INET : ipv4 인터넷 프로토콜에 적용하는 주소체계
AF_INET6: ipv6 인터넷 프로토콜에 적용하는 주소체계
AF_LOCAL: 로컬 통신을 위한 유닉스 프로토콜의 주소체계

sin_port

->16 비트(0~65535) PORT 번호를 저장한다.

단,'네트워크 바이트 순서'로 저장.
PORT 번호를 저장한다는 것보다, 네트워크 바이트 순서로 저장해야 한다는것이 더 중요.

sin_addr

->32 비트 IP 주소정보를 저장.
역시 '네트워크 바이트 순서'로 저장.

*******AF_INET 과 PF_INET 의 차이점 **********

PF_INET 은 프로토콜(프로토콜 패밀리)이고
AF_INET 은 주소 체계(주소 패밀리)중 하나이다.

프로토콜 패밀리는 소켓을 생성할 때 이 소켓이 어떤 프로토콜을 사용해 통신을 할지 정함.
네트워크 통신뿐 아니라 유닉스 계열의 시스템에서 시스템 내부의 프로세스들끼리 통신을 하기 위해서도 사용됨.

AF_INET 은 주소 구조체 안에 주소 패밀리를 정의할때 사용.

결과적으로, 프로토콜 체계를 설정하는 부분은 PF 로 시작하는 상수를 사용하고,
주소 체계를 설정하는 부분은 AF 로 시작하는 상수를 사용하는 것이 좋습니다.

실제 코딩 부분에서 socket()함수에 프로토콜 패밀리에 AF_INET 를 넣어도 되지만 PF_INET 를 넣는게 바람직하고
struct sockaddr_in 구조체에 주소 체계를 넣을 때에도 PF_INET 를 넣어도 되지만 AF_INET 를 넣는게 바람직하다.

*****기본적으로 프로토콜의 정의를 알아보자******

프로토콜이란 컴퓨터간의 통신을 위해 미리 정해진 특별한 규칙이자 규약이다.

**TCP/IP 란?

TCP/IP 는 컴퓨터의 기종에 관계없이 정보교환이 가능케 해주는 것으로
인터넷에서 정보전송을 위해 사용하는 프로토콜(통신규약)이다.
TCP = Transmission Control Protocol
IP = Internet Protocol

**TCP 와 IP 프로토콜의 역할

1.TCP 프로토콜
ㄱ.IP 프로토콜에 의해서 만들어진 패킷들의 전송 흐름 부분을 담당.
ㄴ.패킷이 목적지 주소에 올바른 순서로 도달하도록 제어.
ㄷ.패킷이 중간에 유실되지 않도록 함.

2.IP 프로토콜
ㄱ.보내고자 하는 자료를 패킷으로 쪼갠다.
ㄴ.목적지 주소를 지정.
ㄷ.쪼개진 패킷을 목적지 주소로 운반.

**********socket 에 대해서 알아보자**********8

int socket (int domain, int type , int protocol)
4.반환값 1.프로토콜패밀리 2.소켓형식 3.프로토콜

1.프로토콜 패밀리 ->PF_INET: IPv4 인터넷
PF_INET6: IPv6 인터넷 프로토콜 체계
PF_UNIX: 유닉스

2.소켓형식 ->SOCK_STREAM: TCP 형 소켓을 만들겠다.
SOCK_DGRAM: UDP 형 소켓을 만들겠다.

3.프로토콜 ->IPPROTO_TCP : TCP
IPPROTO_UDP : UDP

4.반환값 -> 성공: 음수가 아닌 값
실패: -1

예제 int socket(PF_INET,SOCK_STREAM,0)

소켓형식과 프로토콜은 TCP 형이면 TCP, UDP 형이면 UDP 로 같게 되므로
소켓형식에 SOCK_STREAM 을 넣은 뒤 프로토콜에 0 을 넣으면 소켓형식에 맞춰
TCP protocol 로 맞춰진다.

*****memset 이란 ?!

void *memset(void *ptr, int value, size_t num)

메모리 블록을 채운다.

ptr 로 시작되는 메모리 주소부터 num 개의 바이트를 value 값으로 채운다.
이때 value 는 unsigned char 로 형변환된다.

ptr
값을 채울 메모리의 시작주소

value
채울 값.
인자로는 int 로 전달되지만 함수 내부적으로는 unsigned char 로 형변환.

num
value 로 채울 바이트 수.

리턴값

ptr 이 리턴된다.

예제

```
#include <stdio.h>
#include <string.h>

int main ()
{
   char str[] = "almost every programmer should know memset!";
   memset (str,'-',6);
   puts (str);
   return 0;
}
```

결과: 앞에 almost 가 ------으로 바뀐다.

여기서, puts 와 fputs 의 차이점을 모르니 puts 를 알아보자

*****puts(gets 도)********

gets() 함수는 키보드로부터 문자열을 입력받고,
puts() 함수는 화면에 문자열을 출력하는데 사용되는 함수입니다.

1.gets()

-gets() 함수는 Enter 키를 누르기 전까지 공백을 포함한 모든 문자열을 입력 받습니다.

2.puts()

-puts() 함수는 문자열을 화면에 출력하고 printf()함수와는 다르게 자동으로 줄이 바뀝니다.

********Socket shutdown 과 Close 차이점**********

int close(int sockfd)

소켓을 닫고 통신을 종료.
성공하면 0, 실패하면 -1 을 반환.
닫힌 소켓은 더 이상 사용 x.
내부적으로 TCP 는 send buffer 와 recv buffer 가 있다.
만약에 close 를 호출 하였는데 send buffer 에 보낼 데이터가 남아 있으면
그것을 모두 보낸 후에 TCP 연결 종료 절차를 따름.

int shutdown(int sockfd, int howto)

네트워크 연결을 종료시키는데 사용.
close()는 참조 카운터를 1 감소시키고 참조 카운터가 0 이되면 종료.
shutdown()은 참조 카운터와 상관없이 TCP 의 연결 종료 절차를 시작.
그런데 close()는 양방향(send recv) 둘다 종료시키는데 반해,
shutdown 함수는 howto 인자에 따라 동작이 달라진다.

close()호출 후에 받을 데이터가 있다면 ? 받을 수 없다.
shutdown 은 howto 인자를 설정하면 가능.

SHUT_RD

연결의 recv 한쪽만 닫음. 이 소켓으로는 데이터를 받지 못함.
그리고 recv buffer 도 폐기.

SHUT_WR

연결의 send 한쪽만 닫음.
이 소켓으로는 어떤 데이터도 보낼 수 없음.
send buffer 에 남아 있는 데이터 모두 보낸 뒤에 TCP 연결 종료 절차가 뒤따름.

SHUT_RDWR

연결의 양쪽 다 받음. 만약에 자신은 데이터를 다 보냈다 하면,
SHUT_WR 인자를 설정하여 shutdown()을 호출하면
다른 쪽이 보내는 데이터를 받을 수 있게 됨.

*********리틀엔디언 빅엔디언??*********

우리가 사용하는 PC 들은 리틀엔디언이라고 해도 과언이 아님.
네트워크 통신에서 사용하는 방법은 빅엔디언.
이 엔디언들은 주소저장 방법.

만약에 \x41424344 를 저장한다고 해보면

만약 int 형 의 4byte 주소가 아래 와 같다고 하면

0xbffffea8,0xbffffea9,0xbffffeaa,0xbffffeab

그렇다면 순서대로

0xbffffea8 = 41, 0xbffffea9 = 42, 0xbffffeaa = 43, 0xbffffeab = 44

라고 저장이 안됨.

0xbffffea8 = 44, 0xbffffea9 = 43, 0xbffffeaa = 42, 0xbffffeab = 41

이렇게 생각한 것과 반대로 저장이 됨
이게 바로 리틀엔디언이기 때문.

그래서 리틀을 빅으로 빅을 리틀으로 바꿔주는게 인디언.

h 는 리틀엔디언을 빅엔디언으로
n 은 빅엔디언을 리틀엔디언으로.

htons 는 short 형 (2 바이트)
htonl 은 long 형 (4 바이트)

ntohs,ntohl 도 마찬가지.

**********atoi**********

10 진 정수 문자열을 정수로 변환합니다.

1. 문자열에서 10 진 정수 숫자 문자 뒤의 일반 문자는 취소되며, 10 진 정수 숫자 문자까지만 숫자로 변환됩니다.
2. 10 진 정수 숫자 문자 앞의 공백문자는 자동 제거되어 10 진 정수 숫자 문자까지만 숫자로 변환됩니다.
3. 공백 및 10 진 정수 문자가 아닌 문자로 시작하면 0 을 반환합니다.

형태 Int atoi(const char *str);
인수 char *str 정수로 변환할 10 진수 정수 문자열
반환 int 10 진 정수 값

**********Thread**********

프로세스란 프로그램이 실행되는 단위인데
메신져를 켜놓고 음악듣고 이런 작업을 동시에 할 수 있는 것은 , 윈도우즈가 멀티 프로세스를 지원하기 때문.
스레드는 프로세스 내의 실행 단위를 말함. (스레드는 메모리 그 자체를 공유한다)

멀티스레드 실행경우 교차할때 우리가 원하는 것 처럼 안되기때문에
lock 을 걸어줘야한다.
(여러 task 들이 동시에 접근해서 정보가 꼬일 수 있음)