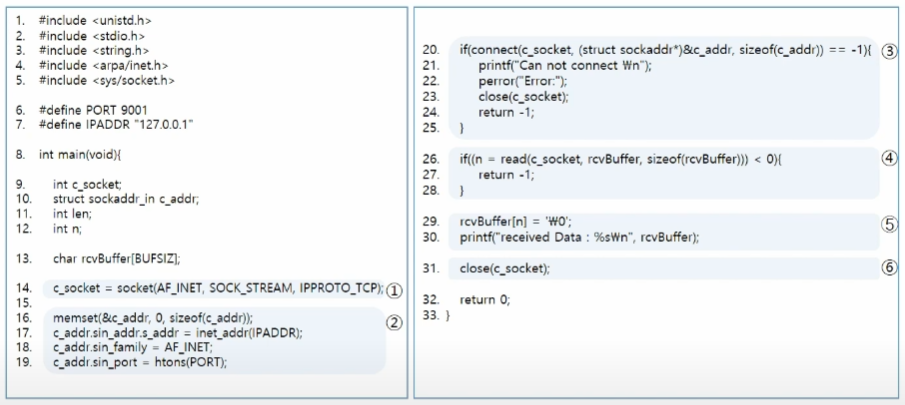
**hello, world 출력 소켓 프로그램 / 클라이언트**



1. 소켓을 생성
2. 연결할 서버의 주소 설정
3. 소켓을 서버에 연결
4. 서비스 요청과 처리
5. 서비스 요청과 처리
6. 소켓 연결을 종료

**socket()을 이용한 소켓 생성**

int socket(int domain, int type, int protocol);

・인터넷과 연결하기 위한 접점인 소켓(endpoint socket) 생성

* domain: 소켓의 사용 영역을 정의
* type: 소켓 유형을 정의
* protocol: 소켓이 사용할 프로토콜을 정의

・socket 함수를 이용한 소켓 생성의 예

1. TCP 소켓:  
   ・socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP)
2. UDP 소켓  
   ・socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP)

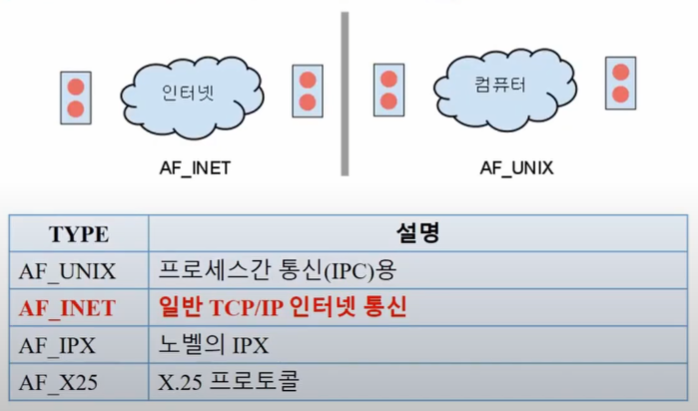
・socket 함수 반환 값

* 성공적으로 소켓을 만들면 0보다 큰 int 값을 반환
* 소켓지정번호, socket descriptor라고 부름
* 소켓을 지시하며, 이를 사용해서 소켓을 제어

・소켓지정번호(socket description)

* descriptor  
  ・유닉스에서 파일을 새로 열면 int형 타입의 파일 디스크립터를 리턴  
  ・프로그램에서 이 파일을 액세스할 때 해당 파일 디스크립터를 사용  
  ・유닉스에서는 각종 하드웨어 장치 파이프 소켓 등을 파일로 취급
* 소켓지정번호는 응용 프로그램 내에서 순서대로 배정됨  
  ・프로그램 내에서만 유일하게 구분

int socket(int domain, int type, int protocol);

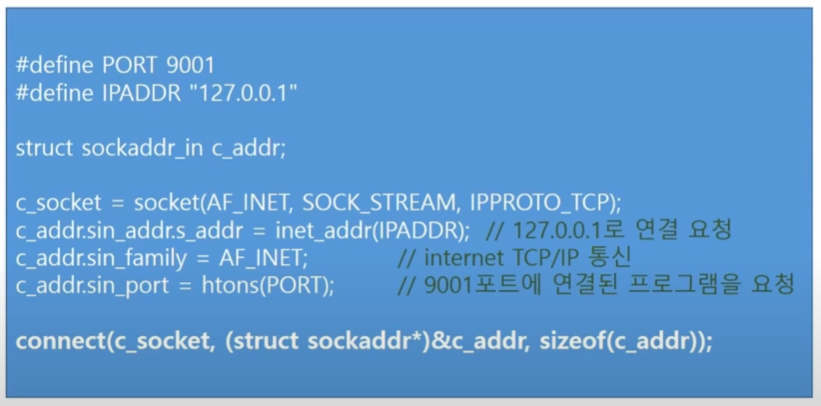
* domain: 소켓이 사용되는 네트워크의 영역을 정의  
  
* Type: 통신에 사용할 패킷의 타입을 지정
* Protocol: 통신에 사용할 프로토콜 지정
* Type에 따라서 Protocol이 정해짐  
    
  ・SOCK\_STREAM & IPPROTO\_TCP: TCP 기반의 통신에 사용  
  ・SOCK\_DGRAM & IPPROTO\_UDP: UDP 기반의 통신에 사용  
  ・SOCK\_RAW & (원하는 프로토콜): RAW Socket으로 저 수준에서 프로토콜을 직접 다룰 때 사용

**connect()을 이용하여 연결하기**

int connect(int sockfd, struct sockaddr \*serv\_addr, socklen\_t addrlen);

・연결하고자 하는 상대 Node의 IP 주소와 Port 번호를 이용하여 소켓 연결

* sockfd: 소켓파일 지정 번호 -> 클라이언트
* serv\_addr: 연결할 서버의 IP 주소와 Port 번호를 포함한 구조체
* addrlen: 구조체 \*serv\_addr의 크기

・connect 함수를 이용한 연결 요청의 예  


**데이터 통신하기**

・데이터 쓰기

int write(int sockfd, const void \*buf, size\_t count);

・데이터 읽기

int read(int sockfd, void \*buf, size\_t count);

* sockfd: 소켓 지정 번호
* buf: 통신에 사용할 데이터를 가리키는 포인터
* count: 통신에 사용할 데이터의 크기

**연결 종료**

int close(int sockfd);

・소켓을 닫고 연결 종료

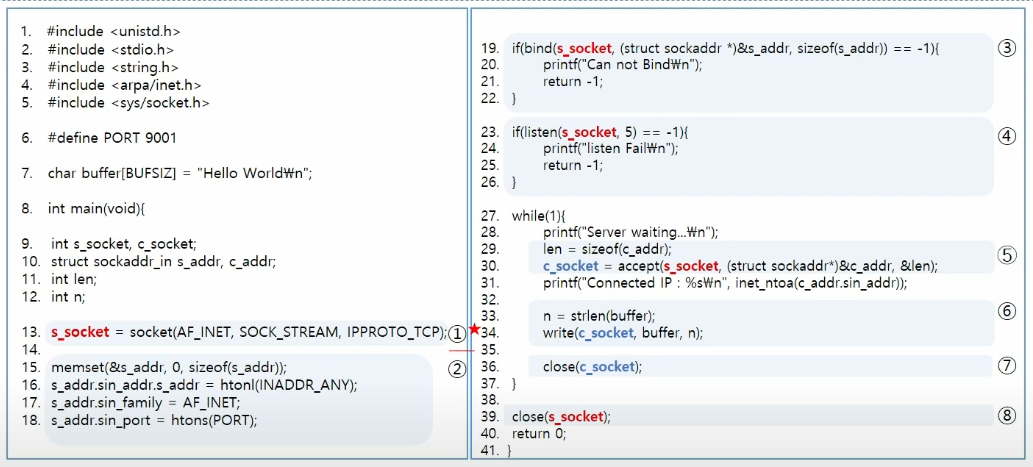
* 데이터 통신이 끝났다면, close 함수를 이용해서 소켓을 닫음
* 소켓을 닫지 않을 경우 자원 누수 발생

**클라이언트 네트워크 프로그램의 흐름**

・클라이언트 프로그램의 흐름

* socket(): 소켓 생성
* connect(): 연결 요청
* read()/write(): 데이터 통신
* close(): 소켓 닫기

**hello, world 출력 소켓 프로그램 / 서버**



1. 소켓을 생성
2. 연결 요청을 수시한 주소 설정
3. 소켓을 주소와 포트에 연결
4. 수신대기열 생성
5. 클라이언트 연결 요청 수신
6. 클라이언트 요청 서비스 제공
7. 클라이언트와 연결 종료
8. 서버 종료

・s\_socket: 클라이언트의 연결 요청을 처리하는 듣기 소켓

・c\_socket: 연결된 클라이언트의 소켓과 직접 통신하는 연결 소켓

**서버 네트워크 프로그램의 흐름**

・서버 프로그램의 흐름

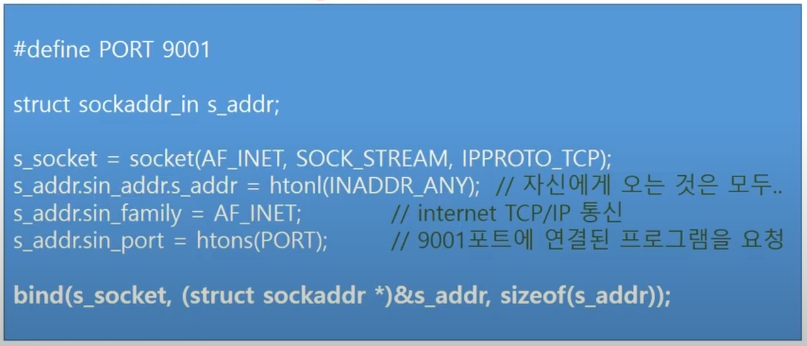
* socket(): 소켓 생성
* bind(): 소켓을 인터넷 주소와 포트 번호에 묶음
* listen(): 수신 대기열 생성 (listen queue)
* accept(): 연결 대기
* read()/write(): 데이터 통신
* close(): 소켓 닫기

**서버 프로그램 만들기 – bind()**

・bind 함수

* 소켓을 인터넷 주소(IP 주소 & Port 번호)에 묶어줌

int bind(int sockfd, struct sockaddr \*my\_addr, socklen\_t addrlen);



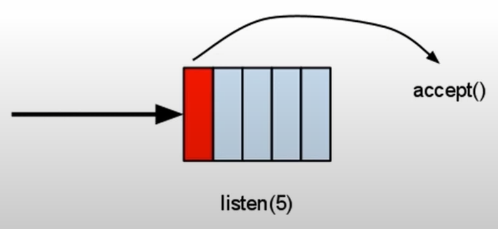
**서버 프로그램 만들기 – listen()**

・listen 함수: 수신 대기열 생성

* 클라이언트의 요청은 먼저 수신 대기열에 들어감

int listen(int sockfd, int backlog);

* sockfd: 소켓 지정 번호
* backlog: 연결 대기열의 크기



**서버 프로그램 만들기 – accept()**

・accept 함수

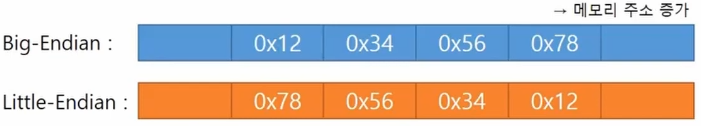
* 수신 대기열의 맨 앞에 있는 클라이언트 요청을 읽음
* 클라이언트와의 통신을 담당할 소켓 지정 번호를 반환

int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen);

* sockfd: 소켓 지정 번호
* \*addr: 연결 요청을 한 클라이언트의 소켓 주소 구조체
* \*addrlen: \*addr 구조체 크기의 포인터

**기타 함수**

・컴퓨터마다 메모리에 데이터를 저장하는 방식이 다름

* 빅엔디안(Big-Endian), 리틀엔디안(Little-Endian)
* 예: 0x12345678 저장 시  
  

・호스트 바이트 순서

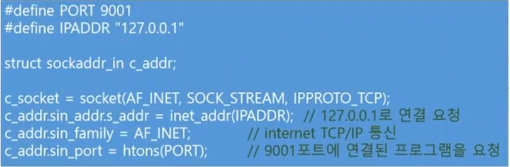
* 컴퓨터가 내부 메모리에 숫자를 저장하는 순서
* CPU종류에 따라 빅엔디안 또는 리틀엔디안 방식 사용

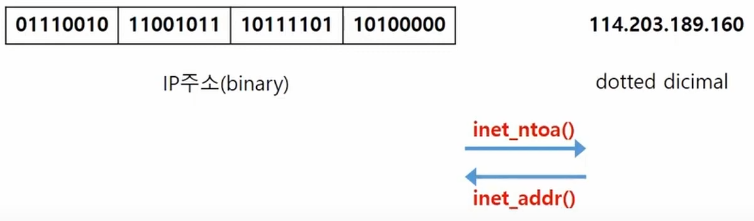
・네트워크 바이트 순서

* 포트번호나 주소와 같은 정보를 바이트 단위로 네트워크로 전송하는 순서
* 빅엔디안 사용

・바이트 순서가 바뀌는 문제의 해결 방법

* 네트워크로 전송하기 전에 htons()함수를 사용하여 네트워크 바이트 순서로 바꿈
* 네트워크로부터 수신한 숫자는 ntohs()함수를 사용하여 자신의 호스트 바이트 순서로 바꿈  
  cf) htonl() / ntohl()

・IP 주소변환  


* 4바이트 IP주소를 dotted decimal 방식으로 상호 변환 가능  
  
* inet\_ntoa(): 바이너리 IP주소를 dotted decimal 형태의 IP주소로
* inet\_addr(): dotted decimal 형태의 IP주소를 바이너리 IP주소로