



# 네트워크 2

MM4220 게임서버 프로그래밍

정내훈

# 내용

---

- 네트워크 복습
- 기본 프로그래밍
- 속제

# 기본 정리

- 네트워크란?
  - 컴퓨터를 연결하여 **Data**주고 받기
- 계층이란?
  - 호환성을 위해 **Data**전송 작업을 단계별로 나누어 놓은것
- Internet이란?
  - 가장 널리 쓰이는 네트워크 규약
- IP 주소와 Port란?
  - IP주소 : **Network**상에서 특정 컴퓨터를 찾는데 사용되는 주소
  - Port : 컴퓨터 안에서 특정 프로그램을 찾는데 사용되는 주소
- 패킷이란?
  - 네트워크상에서 전송되는 **Data**의 묶음
- 소켓이란?
  - 프로그램상에서 네트워크를 지정할 때 쓰이는 자료구조

# 기본 프로그래밍

- 네트워크 프로그래밍 단계
  - 클라이언트
    - 소켓 생성 (Socket)
    - 서버 소켓 연결 (Connect)
    - Data 송/수신 (Recv/Send)
    - 소켓 끊기
  - 서버
    - 소켓 생성 (Socket)
    - 소켓 묶기 (Bind)
    - 소켓 접속 대기 (Listen)
    - 연결 소켓 생성 (Accept)
    - Data 송/수신 (Recv/Send)
    - 소켓 끊기 (Close)

# 기본 프로그래밍

- 소켓

- 네트워크 프로그램의 기본 자료 구조
- C의 **Binary File I/O**에 쓰이는 것처럼 **Network**에서 데이터를 읽고 쓸때 사용
- int type

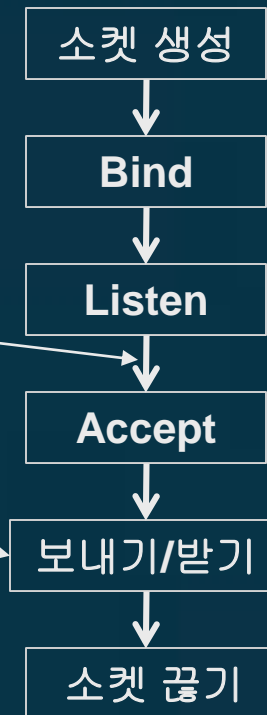
```
int fh1;  
int bytesread;  
  
fh1 = _open( "CRT_OPEN.C", _O_RDONLY ); // C4996  
  
bytesread = _read( fh, buffer, nbytes );
```

# 기본 프로그래밍

클라이언트



서버



# Unix와 Windows 차이

- 유닉스가 오리지널
  - BSD 4.2
- Windows에서는 Winsock을 사용
  - Winsock 1.0은 오리지널 Socket과 동일
  - Winsock 2.0은 WSA를 붙여 독자 API사용
    - Windows Socket API
    - 추가 함수
  - WSASocket(), WSACleanup() 필요

# Windows Network

- Socket 만들기

- SOCKET WSA Socket(int af, int type, int protocol, LPWSAProtocolInfo lpProtocolInfo, GROUP g, DWORD dwFlags)
  - af : address family
    - AF\_INET만 사용 (AF\_NETBIOS, AF\_IRDA, AF\_INET6)
  - type : 소켓의 타입
    - tcp를 위해 SOCK\_STREAM 사용 (SOCK\_DGRAM)
  - protocol : 사용할 프로토콜 종류
    - IPPROTO\_TCP (IPPROTO\_UDP)
  - lpProtocolInfo : 프로토콜 정보
    - 보통 NULL
  - g : 예약
  - dwFlags : 소켓의 속성
    - 보통 0 (또는 WSA\_FLAG\_OVERLAPPED)



# Windows Network

- TCP와 UDP
  - TCP(Transmission Control Protocol)
    - 데이터의 무손실 순서지킴을 보장
    - bind, accept, listen을 통한 연결관리 필요
  - UDP(User Datagram Protocol)
    - 패킷의 크기가 작다
    - 빠르다
    - 데이터의 전송 순서와 무손실을 보장하지 않는다.

# Windows Network

- Socket 연결

- 대기하고 있는 상대방소켓에 자신의 소켓을 연결
- `int WSAConnect(SOCKET s, const struct sockaddr* name, int namelen, LPWSABUF lpCallerData, LPWSABUF lpCalleeData, LPQOS lpSQOS, LPQOS lpGQOS)`
  - `s` : 소켓
  - `name, namelen` : 상대 소켓 주소, 주소 길이
  - 나머지 : 생략

# Windows Network

- Sockaddr 지정 방법

```
#define DEST_IP  "132.241.5.10"  
#define DEST_PORT 23
```

```
main()  
{
```

```
    int sockfd;  
    struct sockaddr_in dest_addr; /* will hold the destination addr */
```

```
    sockfd = WSASocket(AF_INET, SOCK_STREAM, 0, NULL, 0);
```

```
    ZeroMemory(dest_addr, sizeof(dest_addr));  
    dest_addr.sin_family = AF_INET; /* host byte order */  
    dest_addr.sin_port = htons(DEST_PORT); /* short, network byte order */  
    dest_addr.sin_addr.s_addr = inet_addr(DEST_IP);  
    bzero(&(dest_addr.sin_zero), 8); /* zero the rest of the struct */
```

```
    WSAConnect(sockfd, (struct sockaddr *)&dest_addr, sizeof(struct sockaddr), NULL,  
               NULL, NULL, NULL);
```

# Windows Network

- htons()?
  - short로 된 숫자를 네트워크 바이트 순서로 변환
  - 바이트 순서?
    - 빅엔디안(big endian)
      - SPARC, MIPS, IBM 370
    - 리틀엔디안(little endian)
      - Intel x86, DEC Alpha
    - network에서는 빅엔디안
  - htonl, ntohl, ntohs가 있음

# Windows Network

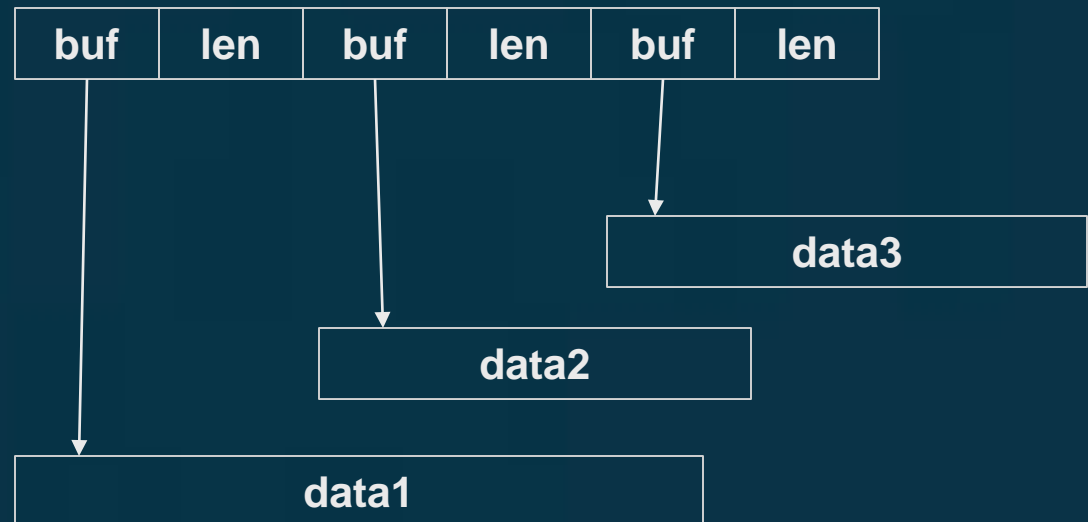
- Socket에서 데이터 받기
  - int WSARecv(SOCKET s, LPWSABUF lpBuffers, DWORD dwBufferCount, LPDWORD lpNumberOfBytesRecv, LPDWORD lpFlags, LPWSAOVERLAPPED lpOverlapped, LPWSAOVERLAPPED\_COMPLETION\_ROUTINE lpCompletionRoutine)
    - s : 소켓
    - lpBuffers : 받은 데이터를 저장할 버퍼
    - dwBufferCount : 버퍼의 개수
    - lpFlags : 동작 옵션(MSG\_PEEK, MSG\_OOB)
    - lpNumberOfBytesRecv : 받은 데이터의 크기
    - lpOverlapped, lpCompletionRoutine : 뒤에 설명

# Windows Network

- WSABUF?

- 흠어진 데이터를 관리하는 구조
- 성능 저하의 주범인 데이터 복사를 줄여줌

```
typedef struct __WSABUF
{
    u_long len;
    char FAR *buf;
} WSABUF, *LPWSABUF;
```



# Windows Network

- Socket에서 데이터 보내기
  - `int WSA_send(SOCKET s, LPWSABUF lpBuffers, DWORD dwBufferCount, LPDWORD lpNumberOfBytesSent, LPDWORD lpFlags, LPWSAOVERLAPPED lpOverlapped, LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine)`
    - `s` : 소켓
    - `lpBuffers` : 보내는 데이터가 저장된 버퍼
    - `dwBufferCount` : 버퍼의 개수
    - `lpFlags` : 동작 옵션 (`MSG_OOB`, ...)
    - `lpNumberOfBytesSent` : 보낸 데이터의 크기
    - `lpOverlapped`, `lpCompletionRoutine` : 뒤에 설명

# Windows Network

---

- Socket **끝**기

- int closesocket(SOCKET s)

- s : 소켓

- return value : 0 on success



# 기본 프로그래밍

- 네트워크 프로그래밍 단계
  - 클라이언트
    - 소켓 생성 (Socket)
    - 서버 소켓 연결 (Connect)
    - Data 송/수신 (Recv/Send)
    - 소켓 끊기
  - 서버
    - 소켓 생성 (Socket)
    - 소켓 묶기 (Bind)
    - 소켓 접속 대기 (Listen)
    - 연결 소켓 생성 (Accept)
    - Data 송/수신 (Recv/Send)
    - 소켓 끊기 (Close)

# 기본 프로그래밍

- 서버가 클라이언트와 다른 점
  - 누가 연결 할 지 모른다.
  - 여러 개의 연결을 관리해야 한다.
- 해결
  - 연결을 관리하는 소켓을 따로 둔다.
  - 외부에서 오는 연결 신청을 받는 **API**를 사용한다.
    - 연결이 성공될 때 마다 새로운 소켓을 만든다.

# Windows Network

- Socket 묶기

- 포트를 선점해서 다른 프로그램이 사용하지 못하도록 한다.
- `int bind(SOCKET s, const struct sockaddr* name, int namelen)`
  - `s` : 소켓
  - `name, namelen` : 상대 소켓 주소, 주소 길이
    - `INADDR_ANY`로 전체 주소로부터의 접속을 허용
    - `info.sin_addr.S_in, S_addr = htonl(INADDR_ANY)`

# Windows Network

- Socket 접속 대기
  - 소켓이 접속을 받을 수 있도록 만든다.
  - `int listen(SOCKET s, int backlog)`
    - `s` : 소켓
    - `backlog` : 접속 대기 큐의 최대 연결 가능 숫자
      - 서버에 도착한 후 `accept`될 때 까지 기다리고 있는 소켓연결의 최대 개수
      - 싱글 스레드 프로그램에서는 5정도가 적당
      - 대용량 서버는 20-200, 또는 **SOMAXCONN**

# Windows Network

- 연결 Socket 생성
  - 연결된 소켓을 만든다.
  - SOCKET WSAAccept(SOCKET s, struct sockaddr\* name, LPINT addrlen, LPCONDITIONPROC lpfnCondition, DWORD dwCallbackData)
    - s : listen을 하고 있는 소켓
    - name, addrlen : 연결된 상대 소켓의 주소
    - lpfnCondition : 연결 거절을 판단하는 함수
      - 보통 NULL
    - dwCallbackData : lpfnCondition에 들어갈 값
    - Return Value : 데이터 전송용 소켓

# Windows Network

- 소켓의 옵션을 변경
  - `int setsockopt(SOCKET s, int level, int optname, const char* optval, int option)`
    - `s` : 옵션을 조절할 소켓
    - `level` : 옵션이 정의된 레벨
    - `optname` : 변경할 소켓 옵션
    - `optval` : 변경할 소켓의 값
    - `option` : 변경할 값의 크기

```
BOOL NoDelay = TRUE;  
setsockopt(mSocket, IPPROTO_TCP, TCP_NODELAY, (const char FAR *) &NoDelay, sizeof(NoDelay));
```

# Windows Network

- 예제



Accept.txt

# Windows Network

- 참고
  - Beej's Guide to Network Programming
    - <http://beej.us/guide/bgnet/>
- 다음 주 : Network 3
  - Overlapped I/O, Non Blocking I/O
  - Multithread programming
  - IOCP



# 숙제 (#2) (2019-화목)

- 게임 서버/클라이언트 프로그램 작성
  - 내용
    - 숙제 (#1)의 프로그램을 Client/Server 모델로 분리
    - 프로그램 2개 작성
    - Client :
      - 키입력을 받아서 서버 프로그램에 보낸다
      - 서버에서 보내온 좌표로 말을 이동시킨다.
      - 시작할 때 서버의 IP주소를 입력 받는다.
    - Server
      - 클라이언트에서 보내온 키입력을 보고 말의 위치 변경
      - 변경된 위치를 클라이언트에 전송
  - 목적
    - Windows 네트워크 프로그래밍 숙달
  - 제약
    - Windows에서 Visual Studio로 작성 할 것
    - 그래픽의 우수성을 보는 것이 아님
  - 제출
    - 3월 12일 화요일 오후 1시 30분 까지
    - Zip으로 소스를 묶어서 e-mail로 제출
    - 제목에 “2019 게임서버[화목] 학번 이름 숙제 2번” 또는 “2019 게임서버[수목] 학번 이름 숙제 2번”