Socket: 데이터를 전달하는 종단점 – 응용계층

윈속의 단점: 프로토콜을 프로그래머가 직접 설계해야 함, 서로 다른 바이트 정렬 방식

오류 코드 함수: int WSAGetLastError(void);

윈속 초기화와 종료방법: 윈속 초기화->소켓 생성->네트워크 통신->소켓 닫기->윈속 종료

윈속 초기화

```
int WSAStartup(
 WORD wVersionRequested(프로그램이 요구하는 최상위 윈속 버전),
 LPWSADATA lpWSAData(윈속 구현에 관한 정보)
);
```

윈속 종료: int WSACleanup(void);

소캣 생성

```
SOCKET socket(
 int af "주소 체계",
 int type, ->SOCK_STREAM(TCP), SOCK_DGRAM(UDP)
 int protocol
);
```

소켓 닫기: int closesocket(SOCKET s);

소켓 데이터 구조체: 지역IP주소, 지역 포트 번호, 원격 IP 주소, 원격 포트 번호, rec버퍼, send버퍼

TCP 서버: socket()->bind()->listen()->accept()->recv()->send()->closesocket()
TCP 클라이언트: socket()->connect()->send()->recv()->closesocket()

bind함수: 소켓을 인터넷 주소에 묶어준다.

```
SOCKADDR_IN serveraddr;
ZeroMemory(&serveraddr, sizeof(serveraddr));
serveraddr.sin_family = AF_INET;
serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
serveraddr.sin_port = htons(SERVERPORT);
retval = bind(listen_sock, (SOCKADDR*)&serveraddr, sizeof(serveraddr));
if(retval == SOCKET_ERROR) err_quit("bind");
```

listen함수: 수신 대기열 생성함수

```
retval = listen(listen_sock, SOMAXCONN);
if(retval == SOCKET_ERROR) err_quit("listen()");
```

accept함수: 접속할 클라이언트를 수용하는 함수

```
SOCKET client_sock;
SOCKETADDR_IN clientaddr;
int addrlen;
```

while(1){
addrlen = sizeof(clientaddr);
client_sock = accept(listen_sock, (SOCKADDR *)&clientaddr, &addrlen);
if(client_sock == INVALD_SOCKET){err_display("accept()"); break;};

connect함수: 서버와 연결하는 함수
SOCKADDR_IN serveraddr;
ZeroMemory(&serveraddr, sizeof(severaddr));
serveraddr.sin_family = AF_INET;
serveraddr.sin_addr.s_addr = inet_addr(SERVERIP);
serveraddr.sin_port = htons(SERVERPORT);
retval = connect(sock, (SOCKADDR*)&serveraddr, sizeof(servaddr));
if(retval == SOCKET_ERROR) err_quit("connect()");

send()함수
int send(SOCKET s(식별자), const char *buf(버퍼), int len(크기), int flags(옵션));

recv()함수
int recv(SOCKET s, char *buf, int len, int flags);

쓰레드는 code영역, data영역, heap영역 공유하고 stack영역 따로 사용, 독립적으로 실행하고 작업처리의 주체

HANDLE CreateThread(LPSECURITY_ATTRIBUTES 핸들의 상속여부 결정, SIZE_T 스택 크기 설정, LPTHREAD_START_ROUTINE 쓰레드 함수, LPVOID 쓰레드 함수의 전달 인자, DWORD 생성 속성, LPDWORD 쓰레드 id);

스레드 대기 함수
DWORD WaitForSingleObject(DWORD thread, DWORD 유지시간);
DWORD WaitForMultipleObject(DWORD thread개수, thread, BOOL bWaitAll(true: 모든 쓰레드 종료시, false: 한 스레드 종료시), DWORD 유지시간)

스레드 중지 함수
DWORD SuspendThread(HANDLE hThread);
void Sleep(DWORD 유지시간);

스레드 재실행 함수
DWORD ResumeThread(HANDLE hThread);

스레드 종료 함수
CloseHandle()

크리티컬 섹션
자료형: CRITICAL_SECTION
InitializeCriticalSection: 초기화, EnterCriticalSection: 획득, LeaveCriticalSection: 반환, DeleteCriticalSection: 종료

뮤텍스: 소유라는 개념을 가지며 임계영역 요구권을 얻기 위해 키 사용
생성: HANDLE CreateMutex (LPSECURITY_ATTRIBUTES 보안 설정, BOOL "소유자 지정 true: main, false: 없음", LPCTSTR 이름 지정)
획득: WaitForSingleObject, 반납: ReleaseMutex(HANDLE 뮤텍스);

세마포어: 여러 개의 뮤텍스, 소유라는 개념이 없지만 counting 개념은 존재
생성 함수: HANDLE CreateSemphore(LPSECURITY_ATTRIBUTES 보안 설정, LONG 초기 개수, LONG 최대 개수, LPCTSTR 이름 지정);
획득: WaitForSingleObject
반환: BOOL ReleaseSemphore(HANDLE 세마포어, LONG 반환 개수, LPLONG IpPreviousCount);

이벤트 생성
HANDLE CreateEvent(LPSECURITY_ATTRIBUTES 보안 설정, BOOL "수동(true) 자동(false)", BOOL "sign(true) non-sign(false)", LPCTSTR 이름 지정);
non-signaled: 임계영역, signaled: 임계영역 아님
비신호 상태->신호 상태: BOOL SetEvent(HANDLE hEvent);
수동 리셋 모드 신호 상태->비신호 상태: BOOL ResetEvent(HANDLE hEvent);
자동 리셋 모드 신호 상태->비신호 상태: WaitForSingleObject

주소 체계
#define AF_INET 2: Internetwork: UDP, TCP, 등등
#define AF_INET6 23: Internetwork Version 6
#define AF_IRDA 26: IrDA
#define AF_BTH 32: Bluetooth RFCOMM/L2CAP protocols