산업기술의 최고 명문대학
한국산업기술대학교

2 윈도우 소켓 시작하기

네트워크 게임 프로그래밍

학습목표

- ❖ 윈속 함수의 오류 처리 방법을 익힌다.
- ❖ 윈속 초기화와 종료 방법을 익힌다.
- ❖ 소켓을 생성하고 닫는 방법을 익힌다.

윈속 함수 오류 처리 (1)

❖ 오류 처리 유형

- 네트워크 프로그램에서 오류 발생시 오류를 체크하여 구체적인 오류 내용을 알려주는 것이 중요.
- 오류 처리 방식은 다음 세가지 유형으로 구분

① 오류를 처리할 필요가 없는 경우

- 리턴값이 없거나 호출 시 항상 성공하는 일부 소켓 함수

② 리턴 값만으로 오류를 처리하는 경우

- WSAStartup() 함수

③ 리턴 값으로 오류 발생을 확인하고, 구체적인 내용은 오류 코드로 확인하는 경우

- 대부분의 소켓 함수
- WSAGetLastError() 함수를 사용해 오류 코드를 얻을 수 있음

윈속 함수 오류 처리 (2)

❖ 오류 코드 얻기

- **WSAGetLastError() 함수 사용**

```
int WSAGetLastError(void);
```

❖ WSAGetLastError() 함수 사용 예

```
if (소켓함수(...) == 실패) {
    int errcode = WSAGetLastError();
    printf(errcode에 해당하는 오류 메시지);
}
```

- WSAGetLastError() 함수의 리턴값을 바로 표시하면 오류코드 의미 해석이 불편하므로 해당 오류 코드를 적절한 문자열 형태로 출력
- FormatMessage() 함수를 사용하면 오류 코드에 대응하는 오류 메시지를 얻을 수 있음
- 세부 리턴값은 MSDN 활용할 것

오류 코드를 문자열로 바꾸기 (1)

❖ FormatMessage() 함수

```
DWORD FormatMessage (
    DWORD dwFlags,                  // 옵션
    LPCVOID lpSource,               // NULL
    DWORD dwMessageId,              // 오류 코드
    DWORD dwLanguageId,             // 언어
    LPTSTR lpBuffer,                // 오류 문자열 시작 주소
    DWORD nSize,                    // 0
    va_list* Arguments              // NULL
);

                                    성공: 오류 메시지의 길이, 실패: 0
```

- dwFlags
  - FORMAT_MESSAGE_ALLOCATE_BUFFER|FORMAT_MESSAGE_FROM_SYSTEM 값을 사용
  - FORMAT_MESSAGE_ALLOCATE_BUFFER: 오류 메시지를 저장할 공간을 FormatMessage() 함수가 알아서 할당
  - FORMAT_MESSAGE_FROM_SYSTEM: 운영체제로부터 오류 메시지를 가져와서 사용
  - dwFlag를 설정하면 lpSource 에는 NULL, nSize는 0, Arguments 는 NULL 값 사용

오류 코드를 문자열로 바꾸기 (1)

❖ FormatMessage() 함수

```
DWORD FormatMessage (
    DWORD dwFlags,                  // 옵션
    LPCVOID lpSource,               // NULL
    DWORD dwMessageId,              // 오류 코드
    DWORD dwLanguageId,             // 언어
    LPTSTR lpBuffer,                // 오류 문자열 시작 주소
    DWORD nSize,                    // 0
    va_list* Arguments              // NULL
);
                                    성공: 오류 메시지의 길이, 실패: 0
```

- **dwMessageId**
  - 오류코드를 나타내며, WSAGetLastError() 함수의 리턴 값을 여기에 넣는다.
- **dwLanguageId**
  - 오류 메시지를 표시할 언어 선택
  - MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT)를 사용하면 제어판에서 설정한 기본언어로 오류 메시지 출력
- **lpBuffer**
  - 오류 메시지의 시작 주소 저장.
  - 오류 메시지를 저장할 공간은 FormatMessage() 가 알아서 할당하므로 주소값을 저장할 변수를 넣어주면 됨

오류 코드를 문자열로 바꾸기 (2)

❖ err_quit() 함수 정의

- FormatMessage() 함수를 사용한 오류 처리 함수 예
- Err_quit() 함수는 msg 인자로 전달된 문자열과 더불어 현재 발생한 오류 메시지를 화면에 메시지 상자로 표시하고 응용프로그램을 종료하는 역할 수행

```
void err_quit(char *msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER
        | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL, WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf, 0, NULL);
    MessageBox(NULL, (LPCTSTR)lpMsgBuf, msg, MB_ICONERROR);
    LocalFree(lpMsgBuf);
    exit(1);
}
```

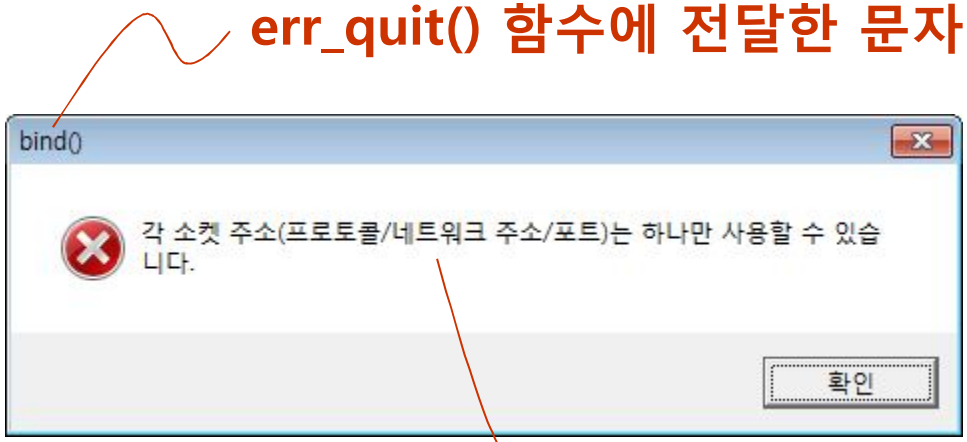
오류 코드를 문자열로 바꾸기 (3)

❖ **err_quit() 함수 사용 예**

```
if(socket(...) == INVALID_SOCKET) err_quit("socket()");
if(bind(...) == SOCKET_ERROR) err_quit("bind()");
```

❖ err_quit() 함수의 오류 메시지

- 예) 위 코드에서 bind() 함수에서 오류가 발생하면 다음과 같이 메시지 상자를 화면에 표시하고 <확인> 버튼을 누르면 응용 프로그램을 종료



오류 코드를 문자열로 바꾸기 (4)

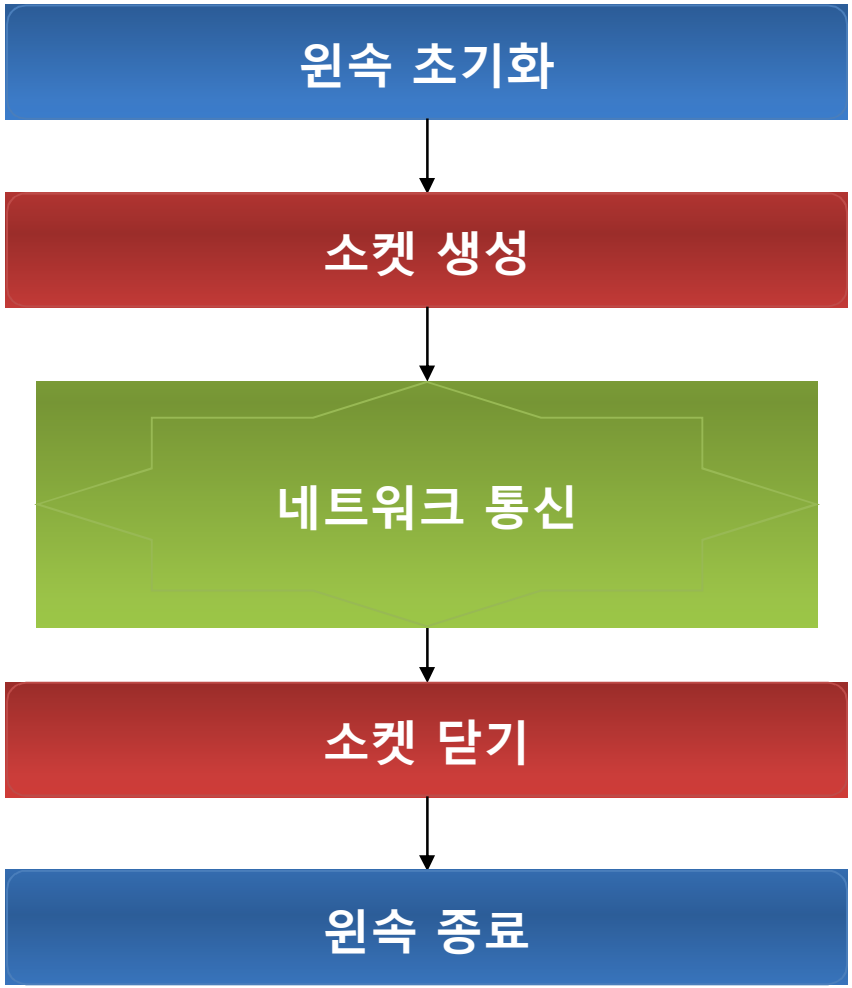
❖ err_display() 함수 정의

- err_display() 함수는 err_quit() 함수에서 마지막 한행을 제거하고 출력 함수로 MessageBox() 대신 printf() 함수를 사용
- 메시지를 출력하되 응용 프로그램은 종료하지 않음. 오류가 발생할때마다 응용프로그램이 종료되는 것을 방지
- 개발자 로그 확인용으로 적함

```
void err_display(char *msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER
        | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL, WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf, 0, NULL);
    printf("[%s] %s", msg, (char *)lpMsgBuf);
    LocalFree(lpMsgBuf);
}
```

윈속 초기화와 종료 (1)

❖ 윈속 응용 프로그램의 공통 구조



윈속 초기화와 종료 (2)

❖ 윈속 초기화

- 모든 윈속 프로그램은 소켓 함수를 호출하기 전에 반드시 윈속 초기화 함수인 WSAStartup() 함수를 호출해야 함.
- WSAStartup() 함수는 프로그램에서 사용할 윈속 버전을 요청함으로써 윈속 라이브러리인 WS2_32.DLL을 초기화하는 역할 수행

```
int WSAStartup (
    WORD wVersionRequested,
    LPWSADATA lpWSAData
);
```

성공: 0, 실패: 오류 코드

- wVersionRequested
  - 프로그램이 요구하는 최상위 윈속 버전. 하위 8비트에 주 버전을, 상위 8비트에 부 버전을 넣어서 전달. 예) MAKEWORD(2,2)
- lpWSAData
  - 윈도우 운영체제가 제공하는 윈속 구현에 관한 정보를 얻을 수 있음(거의 사용 안 함)

윈속 초기화와 종료 (3)

❖ 윈속 종료

- 프로그램을 종료할 때는 윈속 종료 함수인 WSACleanup() 을 호출
- WSACleanup() 함수는 윈속 사용을 중지함을 운영체제에 알리고, 관련 리소스를 반환하는 역할 수행
- WSAGetLastError() 함수를 호출함으로써 구체적인 오류 코드를 획득

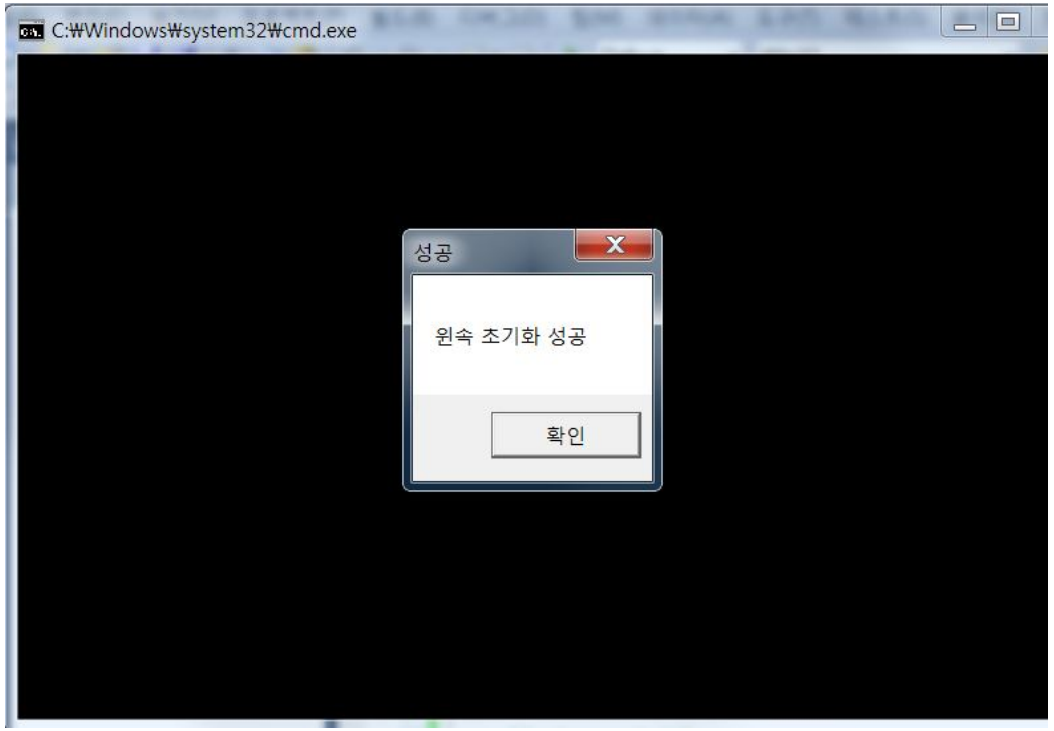
```
int WSACleanup(void);
```

성공: 0, 실패: SOCKET_ERROR

- 참고: WSAStartup() 함수를 두번 이상 호출하는 것도 가능. 단, WSAStartup() 함수를 호출한 횟수만큼 WSACleanup() 함수를 호출해야 함

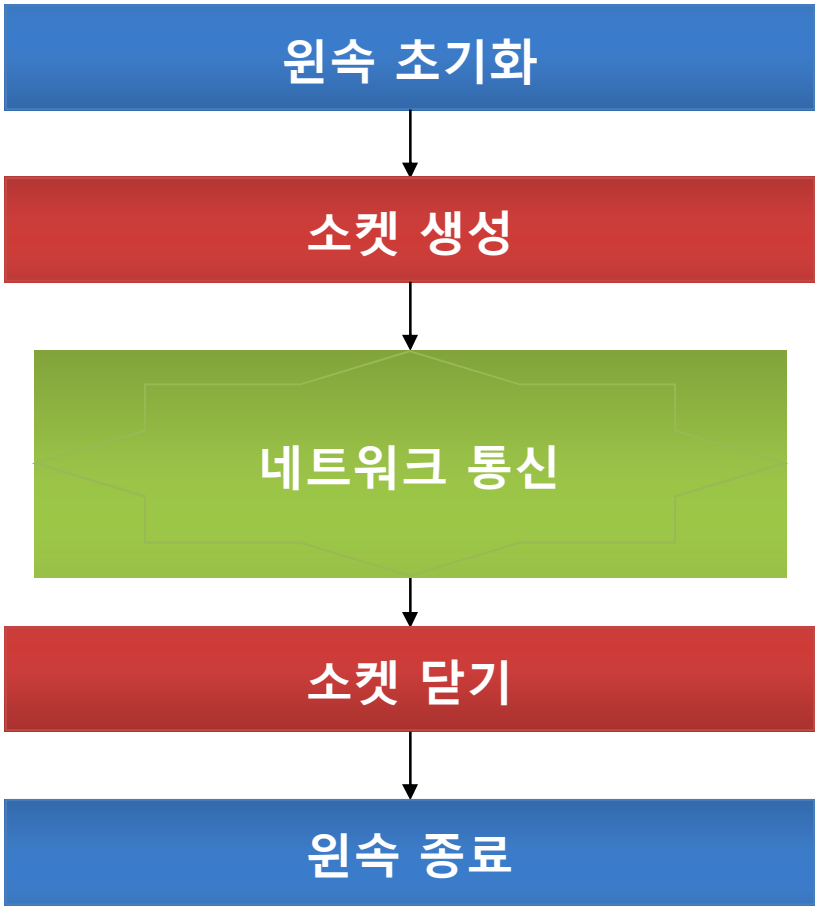
예제 실습

실습 2-1.
윈속 초기화와 종료하기
p. 54



소켓 생성과 닫기 (1)

❖ 윈속 응용 프로그램의 공통 구조



소켓 생성과 닫기 (2)

❖ 소켓 생성

- 예) TCP 혹은 UDP 사용 여부를 결정하고 서로 약속함

```
SOCKET socket (
    int af,                    // 주소 체계 지정
    int type,                  // 소켓 타입 지정
    int protocol               // 사용할 프로토콜 지정
);
                                   성공: 새로운 소켓, 실패: INVALID_SOCKET
```

- 사용자가 요청한 프로토콜을 사용해 통신할 수 있도록 내부적으로 리소스를 할당하고, 이에 접근할 수 있는 일종의 핸들 값(SOCKET 타입, 32비트 정수)인 소켓 디스크립터(socket descriptor)를 리턴

소켓 생성과 닫기 (3)

❖ 주소 체계

- **winsock2.h or ws2def.h 파일 내용**

```
...
#define AF_INET 2               // Internetwork: UDP, TCP, etc.
...
#define AF_INET6 23             // Internetwork Version 6
...
#define AF_IRDA 26              // IrDA
...
#define AF_BTH 32               // Bluetooth RFCOMM/L2CAP protocols
...
```

소켓 생성과 닫기 (4)

❖ 소켓 타입

- 자주 사용하는 소켓 타입

소켓 타입	특성
SOCK_STREAM	신뢰성 있는 데이터 전송 기능 제공, 연결형 프로토콜
SOCK_DGRAM	신뢰성 없는 데이터 전송 기능 제공, 비연결형 프로토콜

- TCP와 UDP 프로토콜 사용을 위한 설정(1)
  - 소켓 타입은 네트워크 프로토콜의 종류에 따라 달라지므로, 소켓 타입 지정은 자신이 사용할 프로토콜을 선택하기 위한 두번째 요소임
  - TCP나 UDP 프로토콜을 사용하려면 하기와 같은 주소 체계와 소켓 타입을 설정해야 함
  - 주소체계가 같아도 소켓 타입을 다르게 설정하면 결과적으로 사용할 프로토콜의 종류가 달라짐

사용할 프로토콜	주소 체계	소켓 타입
TCP	AF_INET 또는 AF_INET6	SOCK_STREAM
UDP		SOCK_DGRAM

소켓 생성과 닫기 (5)

❖ 프로토콜

- 주소 체계와 소켓 타입이 같더라도 해당 프로토콜이 두 개 이상 존재할 경우 프로토콜을 명시적으로 지정
  - TCP와 UDP 프로토콜 사용을 위한 설정(2)

사용할 프로토콜	주소 체계	소켓 타입	프로토콜
TCP	AF_INET 또는 AF_INET6	SOCK_STREAM	IPPROTO_TCP
UDP		SOCK_DGRAM	IPPROTO_UDP

  - TCP와 UDP 프로토콜 사용을 위한 설정(3)
  - TCP와 UDP 프로토콜은 주소 체계와 소켓 타입만으로 프로토콜을 결정할 수 있으므로 프로토콜 부분은 보통은 0을 사용

사용할 프로토콜	주소 체계	소켓 타입	프로토콜
TCP	AF_INET 또는 AF_INET6	SOCK_STREAM	0
UDP		SOCK_DGRAM	0

소켓 생성과 닫기 (6)

❖ 소켓 닫기

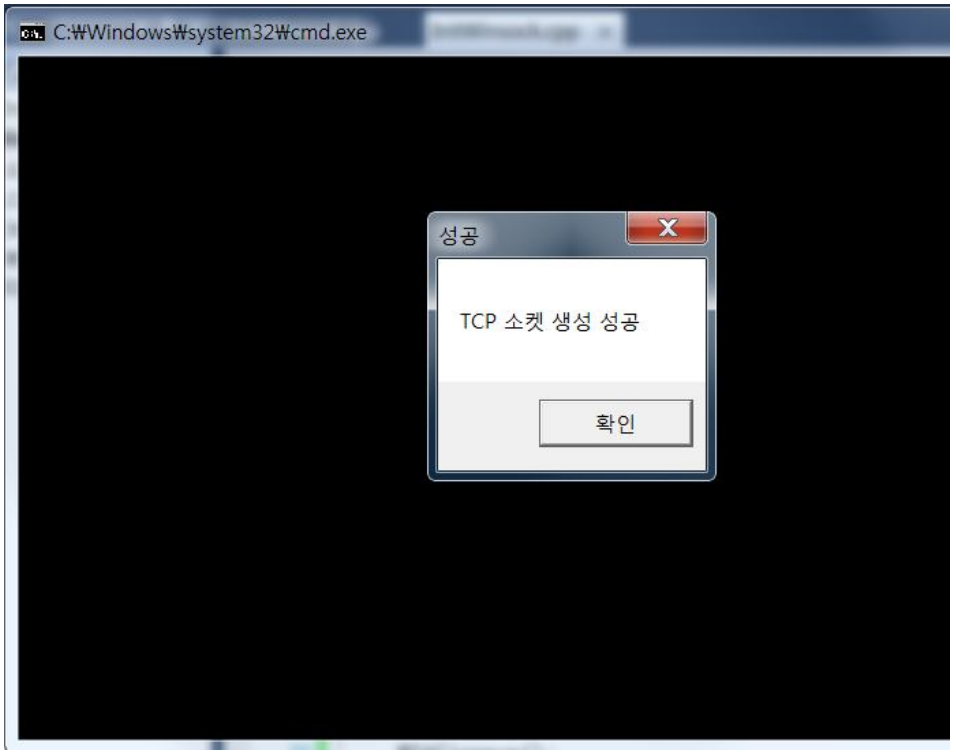
```
int closesocket (
   SOCKET s
);
```

성공: 0, 실패: SOCKET_ERROR

- 소켓을 닫고 관련 리소스를 반환
- closesocket() 함수는 해당 소켓을 닫고 관련 리소스를 반환

예제 실습

실습 2-2.
소켓 생성과 닫기
p. 58



Thank You !

oasis01@gmail.com / rhqudtn75@nate.com