

Chapter 10 GUI 소켓 응용 프로그래밍: 윈도우

- GUI 응용 프로그램의 구조와 동작 원리를 이해한다.
- GUI 소켓 응용 프로그램 작성 기법을 익힌다.
- 대화상자 기반 응용 프로그램의 구조와 동작 원리를 이해한다.
- 대화상자 기반 소켓 응용 프로그램 작성 기법을 익힌다.

목차

- 01 GUI 응용 프로그램
- 02 GUI 소켓 응용 프로그램
- 03 대화상자 기반 응용 프로그램
- 04 대화상자 기반 소켓 응용 프로그램

01 GUI 응용 프로그램



GUI 응용 프로그램 (1)

■ GUI 응용 프로그램 특징

- 다양한 API로 구현된 편리하고 화려한 사용자 인터페이스 제공
- 메시지 구동 구조로 동작

■ 용어

- API(*Application Programming Interface*)
 - 윈도우 운영체제가 응용 프로그램에 제공하는 함수 집합
- 메시지(*Message*)
 - 윈도우 운영체제가 응용 프로그램의 외부나 내부에 변화가 생겼음을 해당 응용 프로그램에 알리는 데 사용하는 개념

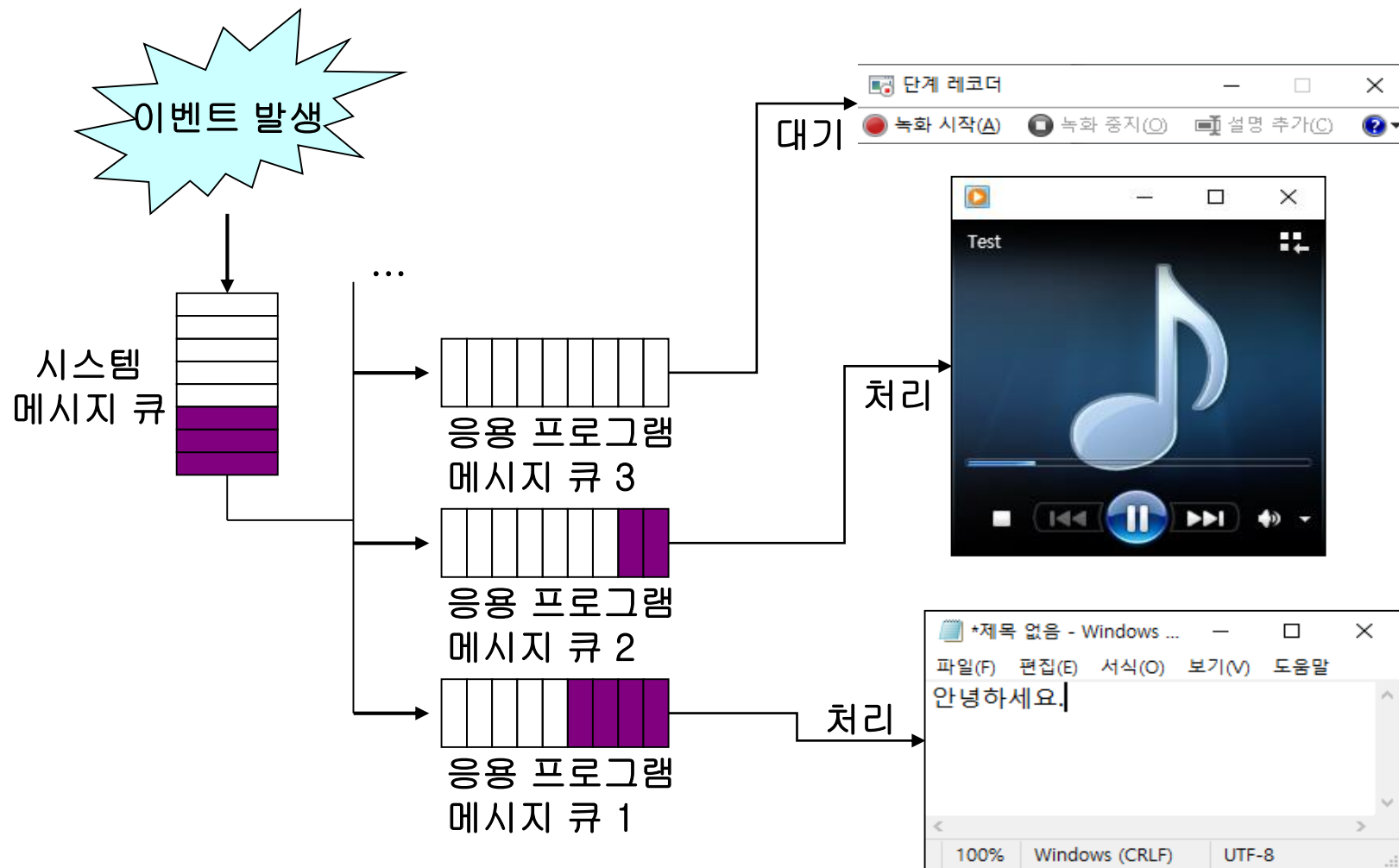
GUI 응용 프로그램 (2)

■ 메시지 구동 구조 동작(간단)

- 이벤트가 발생하면 운영체제가 관리하는 시스템 메시지 큐에 저장됨
- 운영체제는 시스템 메시지 큐에 저장된 메시지를 적절한 응용 프로그램의 메시지 큐에 보냄
- 응용 프로그램은 자신의 메시지 큐를 감시하다가 메시지가 발생해서 큐에 들어오면 하나씩 꺼내 처리하고, 메시지가 없을 때는 대기함

GUI 응용 프로그램 (3)

■ 메시지 구동 구조(간단)



GUI 응용 프로그램 (4)

■ 메시지 구동 구조 동작(상세)

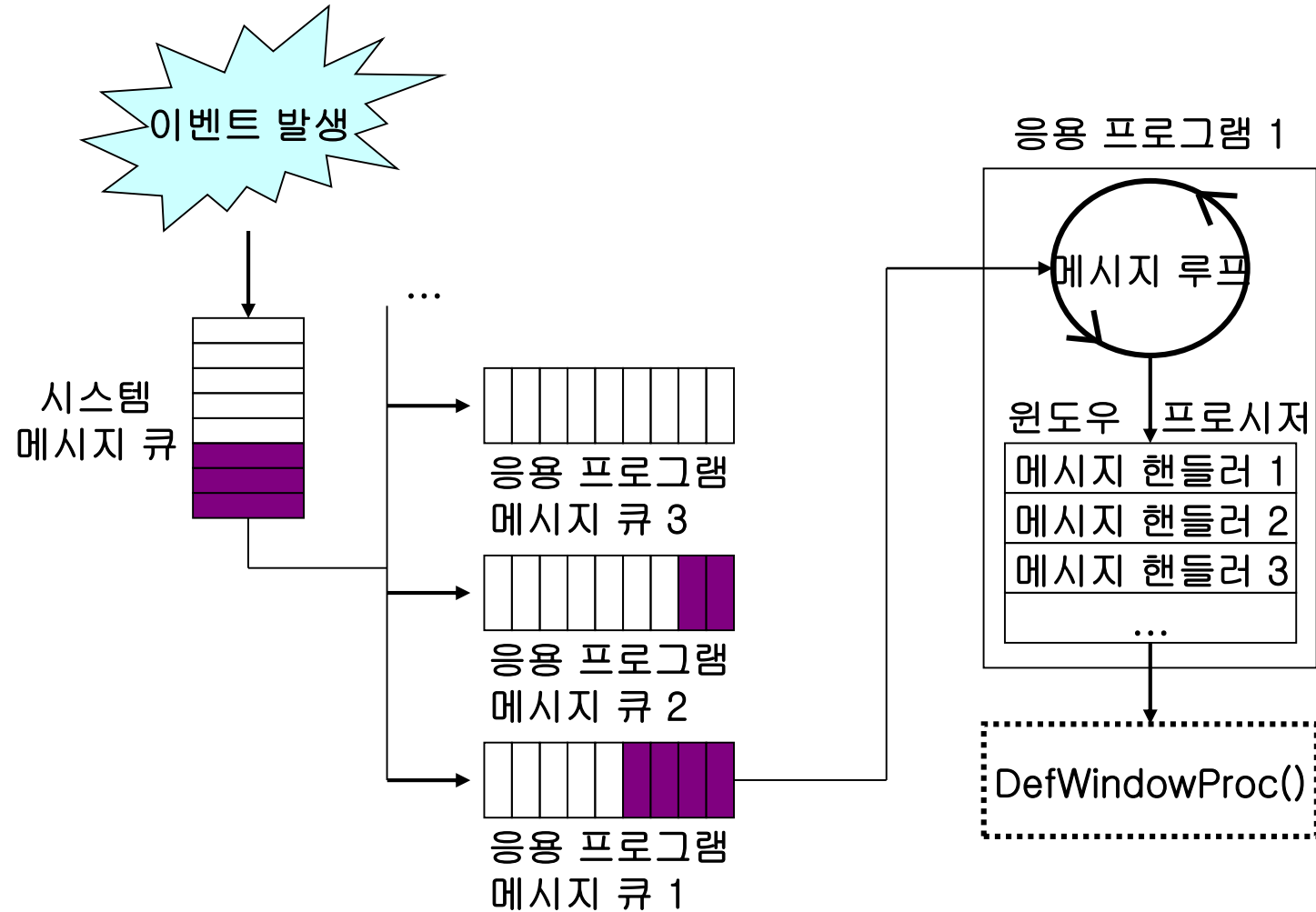
- GUI 응용 프로그램은 윈도우 프로시저에 전달된 메시지를 (메시지 핸들러에서) 자신만의 방식으로 처리
- 처리하지 않은 메시지는 윈도우 운영체제에 맡겨서 자동으로 처리

■ 용어

- 메시지 핸들러(*Message Handler*)
 - 메시지를 받았을 때 동작을 결정하는 코드
- 윈도우 프로시저(*Window Procedure*)
 - 메시지 핸들러의 집합

GUI 응용 프로그램 (5)

■ 메시지 구동 구조(상세)



GUI 응용 프로그램 (6)

■ 500×220 크기의 윈도우를 표시하는 간단한 형태의 GUI 응용 프로그램 코드



그림 10-3 간단한 GUI 응용 프로그램

```
1 #include <windows.h>
   #include <tchar.h>

2 LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

3 int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
   LPSTR lpCmdLine, int nCmdShow)
{
4     // 윈도우 클래스 등록
   WNDCLASS wndclass;
   wndclass.style = CS_HREDRAW | CS_VREDRAW;
   wndclass.lpfnWndProc = WndProc;
   wndclass.cbClsExtra = 0;
   wndclass.cbWndExtra = 0;
```

GUI 응용 프로그램 (7)

```
wndclass.hInstance = hInstance;  
wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);  
wndclass.hCursor = LoadCursor(NULL, IDC_ARROW);  
wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);  
wndclass.lpszMenuName = NULL;  
wndclass.lpszClassName = _T("MyWndClass");  
if (!RegisterClass(&wndclass)) return 1;
```

5

// 윈도우 생성

```
HWND hWnd = CreateWindow(_T("MyWndClass"), _T("Simple Window"),  
    WS_OVERLAPPEDWINDOW, 0, 0, 500, 220, NULL, NULL, hInstance, NULL);  
if (hWnd == NULL) return 1;  
ShowWindow(hWnd, nCmdShow);  
UpdateWindow(hWnd);
```

GUI 응용 프로그램 (8)

```
6 // 메시지 루프
MSG msg;
while (GetMessage(&msg, NULL, 0, 0) > 0) {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
return (int)msg.wParam;
}

7 // 윈도우 프로시저
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch (uMsg) {
        case WM_CREATE:
            return 0;
        case WM_SIZE:
            return 0;
        case WM_DESTROY:
            PostQuitMessage(0);
            return 0;
    }
    return DefWindowProc(hWnd, uMsg, wParam, lParam);
}
```

GUI 응용 프로그램 (9)

■ WinMain() 함수

- 콘솔 응용 프로그램의 main() 함수와 같은 기능을 하는 GUI 응용 프로그램의 실행 시작점
- 각 인수의 의미

표 10-1 WinMain() 함수 인수

인수	의미
hInstance	인스턴스 핸들 ^{Instance Handle} 이라고 부르는 값으로, 실행 파일에 포함된 각종 리소스에 접근할 때 사용한다. 여러 함수에서 필요하므로 전역 변수에 저장해두고 사용하면 편리하다.
lpCmdLine	프로그램 실행 시 전달된 명령행 인수 ^{Command-line Argument} 를 담고 있는 문자열이다.
nCmdShow	프로그램이 시작할 때 메인 윈도우를 보일 방법(최대화, 최소화 등)을 제어하는 값이다.

GUI 응용 프로그램 (10)

■ 윈도우 프로시저

- 윈도우 메시지를 처리하는 핵심 함수
- 윈도우 프로시저에 전달되는 인수의 의미

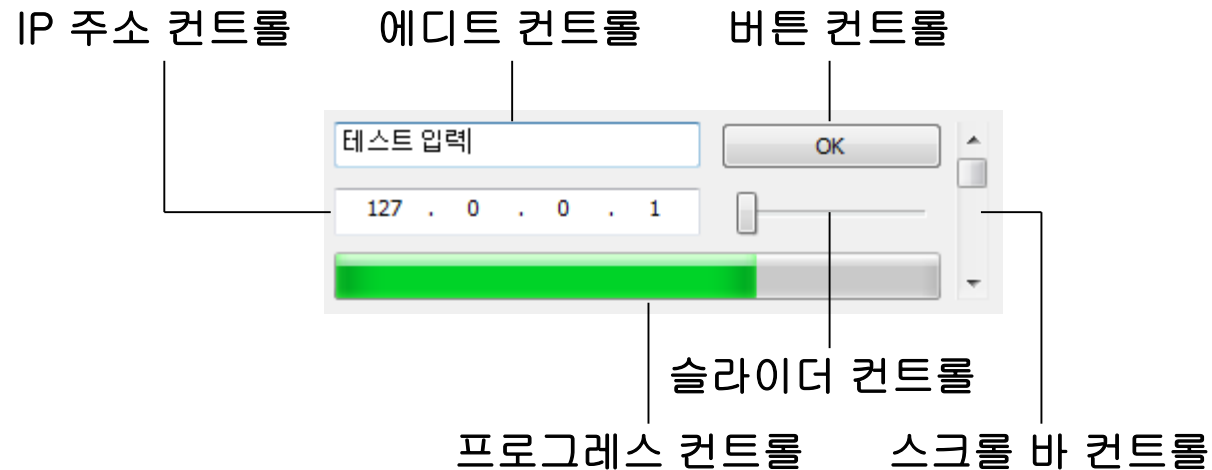
표 10-2 윈도우 프로시저 인수

인수	의미
hWnd	메시지가 발생한 윈도우를 나타내는 핸들값이다. 윈도우 프로시저에서 윈도우를 조작할 때는 항상 이 값을 사용해야 한다.
uMsg	발생한 메시지를 나타내는 값이다. WM_* 형태의 상수로 정의되어 있다.
wParam lParam	메시지와 더불어 전달되는 두 개의 부가 정보다(윈도우 운영체제에 따라 32비트 또는 64비트 크기). 메시지마다 의미가 다르므로 도움말을 참조하여 해석하고 처리해야 한다.

GUI 응용 프로그램 (11)

■ 컨트롤

- 표준화된 형태와 특성을 제공하는 일종의 윈도우
 - 사용자의 입력을 받거나 출력할 수 있음



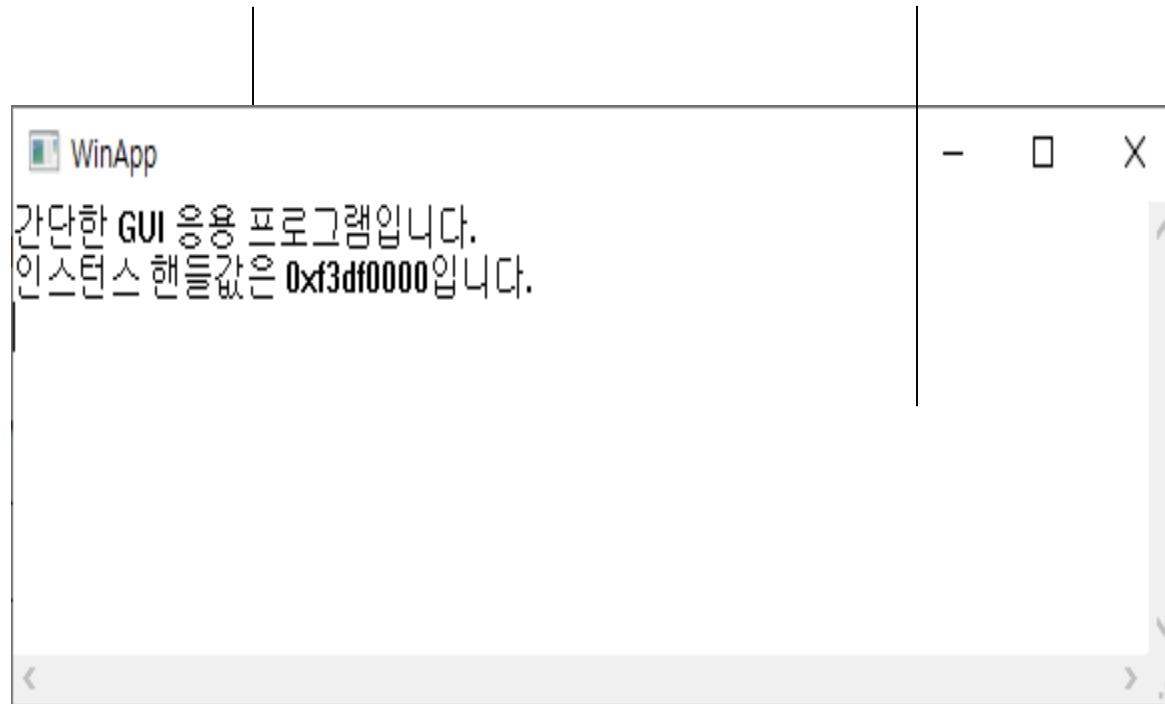
- 독립적인 윈도우가 아닌 자식 윈도우로 존재
 - 부모 윈도우는 SendMessage() 함수를 사용해 컨트롤에 메시지를 보냄으로써 컨트롤을 기정의된 방식으로 제어함

GUI 응용 프로그램 (12)

■ 예제 프로그램의 구조

메인 윈도우(부모 윈도우)

에디트 컨트롤(자식 윈도우)



GUI 응용 프로그램 작성과 테스트 (1)

- 실습 10-1 GUI 응용 프로그램 작성과 테스트
 - <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter10/WinApp/WinApp.cpp>

02 GUI 소켓 응용 프로그램



GUI 소켓 응용 프로그램 (1)

■ 윈도우 메시지 처리 지연 상황

```
case WM_SIZE:
    MoveWindow(hEdit, 0, 0, LOWORD(lParam), HIWORD(lParam), TRUE);
    Sleep(3000);
    return 0;
```

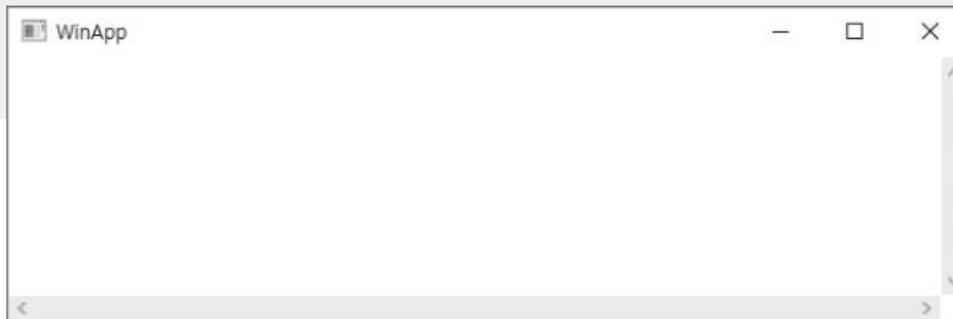
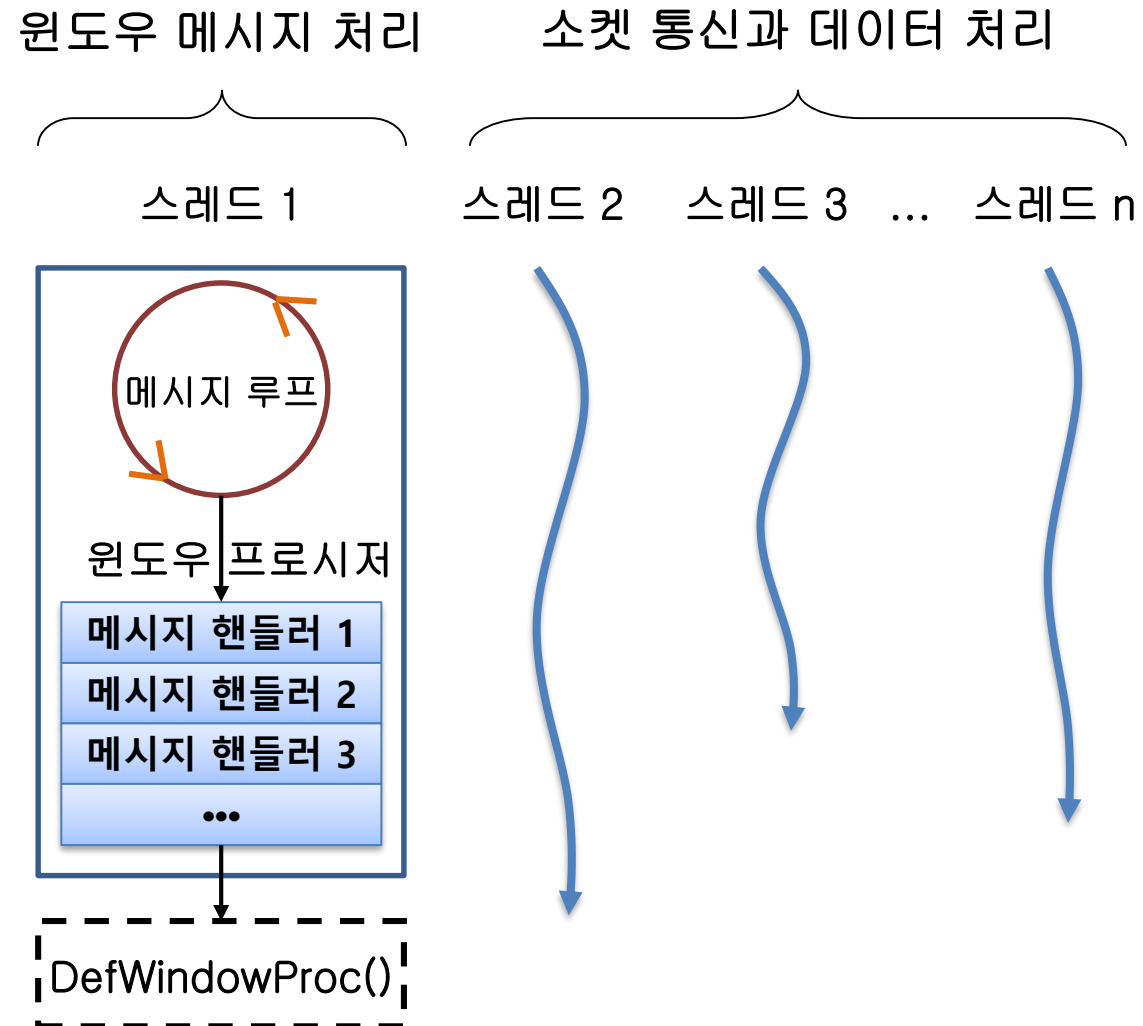


그림 10-6 윈도우 메시지 처리 지연 상황

- 약 3초 동안 화면이 제대로 표시되지 않음
- 함수를 호출할 때 조건이 만족하지 않으면 스레드는 대기 상태가 되므로 윈도우의 메시지 루프 처리가 지나치게 지연될 수 있음
- 이런 문제를 해결하려면 소켓 코드를 별도의 스레드로 분리해야 하며, 윈도우 프로시저와 소켓 코드가 공유하는 데이터가 있을 때는 스레드 동기화 기법을 사용하여 보호해야 함

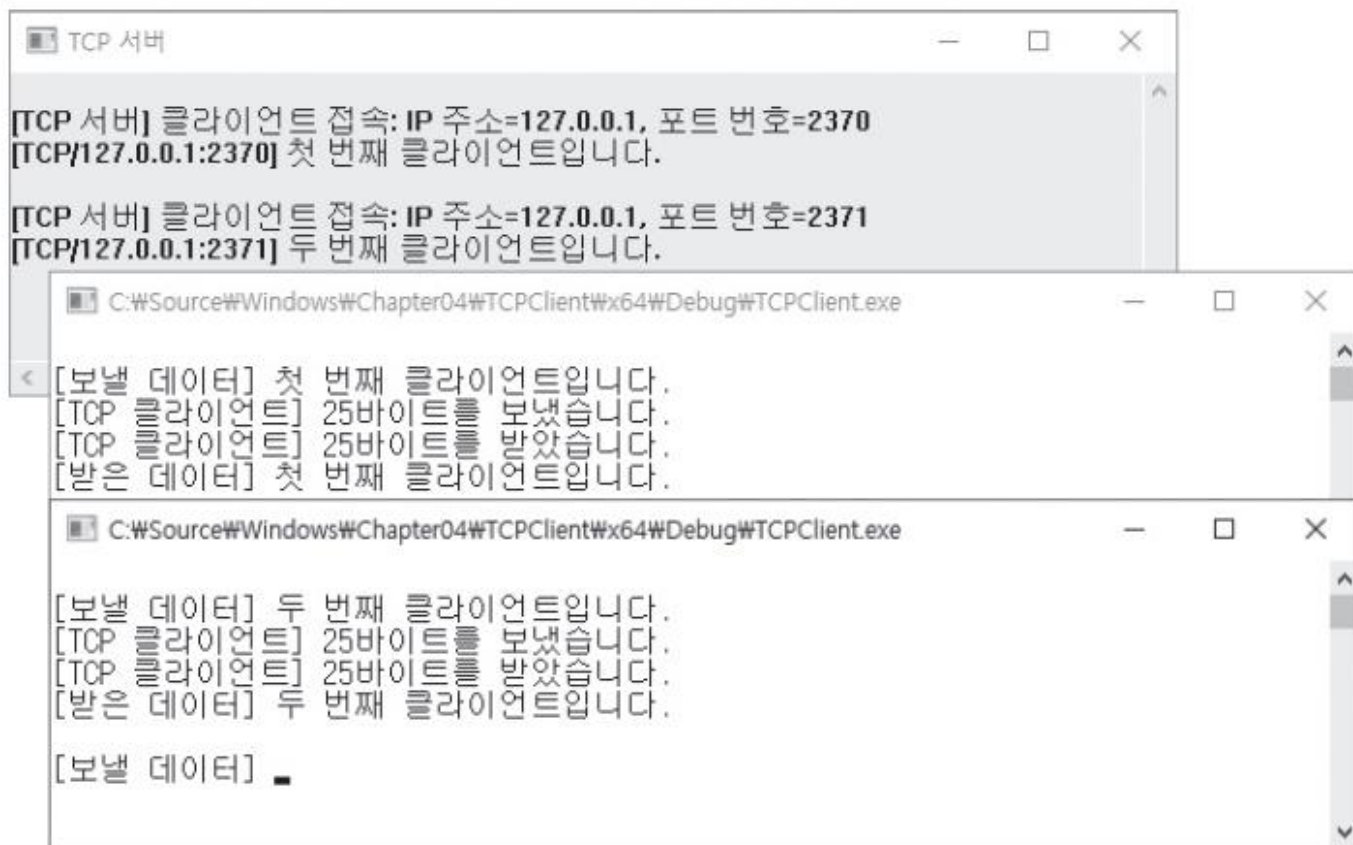
GUI 소켓 응용 프로그램 (2)

■ GUI 소켓 응용 프로그램 구조



GUI 소켓 응용 프로그램 작성 (1)

■ 실습 10-2 GUI 소켓 응용 프로그램 작성과 테스트



The screenshot displays three overlapping windows from a Windows operating system. The top window, titled 'TCP 서버', shows two log entries: '[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=2370 [TCP/127.0.0.1:2370] 첫 번째 클라이언트입니다.' and '[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=2371 [TCP/127.0.0.1:2371] 두 번째 클라이언트입니다.'. Below it, a window titled 'C:\Source\Windows\Chapter04\TCPClient\wx64\Debug\TCPClient.exe' shows the first client's activity: '[보낼 데이터] 첫 번째 클라이언트입니다.', '[TCP 클라이언트] 25바이트를 보냈습니다.', '[TCP 클라이언트] 25바이트를 받았습니다.', and '[받은 데이터] 첫 번째 클라이언트입니다.'. The bottom window, also titled 'C:\Source\Windows\Chapter04\TCPClient\wx64\Debug\TCPClient.exe', shows the second client's activity: '[보낼 데이터] 두 번째 클라이언트입니다.', '[TCP 클라이언트] 25바이트를 보냈습니다.', '[TCP 클라이언트] 25바이트를 받았습니다.', '[받은 데이터] 두 번째 클라이언트입니다.', and '[보낼 데이터] _'.

```
TCP 서버
[ TCP 서버 ] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=2370
[ TCP/127.0.0.1:2370 ] 첫 번째 클라이언트입니다.

[ TCP 서버 ] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=2371
[ TCP/127.0.0.1:2371 ] 두 번째 클라이언트입니다.

C:\Source\Windows\Chapter04\TCPClient\wx64\Debug\TCPClient.exe
[ 보낼 데이터 ] 첫 번째 클라이언트입니다.
[ TCP 클라이언트 ] 25바이트를 보냈습니다.
[ TCP 클라이언트 ] 25바이트를 받았습니다.
[ 받은 데이터 ] 첫 번째 클라이언트입니다.

C:\Source\Windows\Chapter04\TCPClient\wx64\Debug\TCPClient.exe
[ 보낼 데이터 ] 두 번째 클라이언트입니다.
[ TCP 클라이언트 ] 25바이트를 보냈습니다.
[ TCP 클라이언트 ] 25바이트를 받았습니다.
[ 받은 데이터 ] 두 번째 클라이언트입니다.

[ 보낼 데이터 ] _
```

GUI 소켓 응용 프로그램 작성 (2)

■ 실습 10-2 GUI 소켓 응용 프로그램 작성과 테스트

- GUITCPServer.cpp

- <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter10/GUITCPServer/GUITCPServer.cpp>

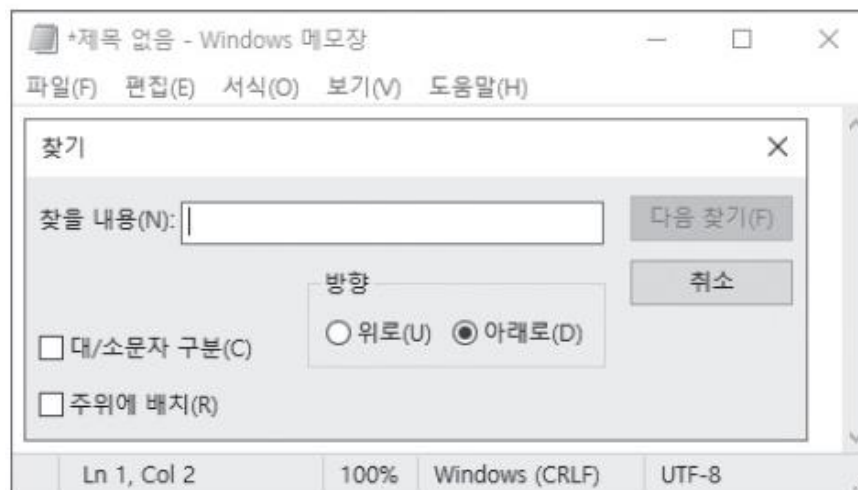
03 대화상자 기반 응용 프로그램



대화상자 기반 응용 프로그램 (1)

■ 대화상자

- 다양한 컨트롤을 포함하는 일종의 윈도우
 - 사용자의 입력을 받거나 정보를 출력



(a) 대화상자



(b) 대화상자 기반 응용 프로그램

그림 10-8 대화상자와 대화상자 기반 응용 프로그램

대화상자 기반 응용 프로그램 (2)

■ 대화상자 디자인부터 최종적으로 화면에 보이기까지 절차

- ❶ 프로젝트에 리소스 파일(*.rc) 추가
- ❷ 리소스 파일에 대화상자 리소스를 추가하고 비주얼 스튜디오의 리소스 편집기를 이용해 시각적으로 디자인
- ❸ 프로젝트를 빌드하면 컴파일 단계에서 리소스 컴파일러가 실행되어 *.rc 파일을 이진 형식인 *.res로 변환
 - *.res 파일은 링크 단계에서 실행 파일 내부에 리소스로 포함됨
- ❹ 프로그램 실행 중 대화상자 생성을 요청하는 윈도우 API 함수를 호출
 - 윈도우 운영체제는 실행 파일 내부의 대화상자 리소스(대화상자 템플릿)를 토대로 대화상자를 생성

대화상자 기반 응용 프로그램 (3)

■ 간단한 대화상자 기반 응용 프로그램

- 대화상자를 표시하고 [확인] 또는 [취소] 버튼을 누르면 종료하는 기능

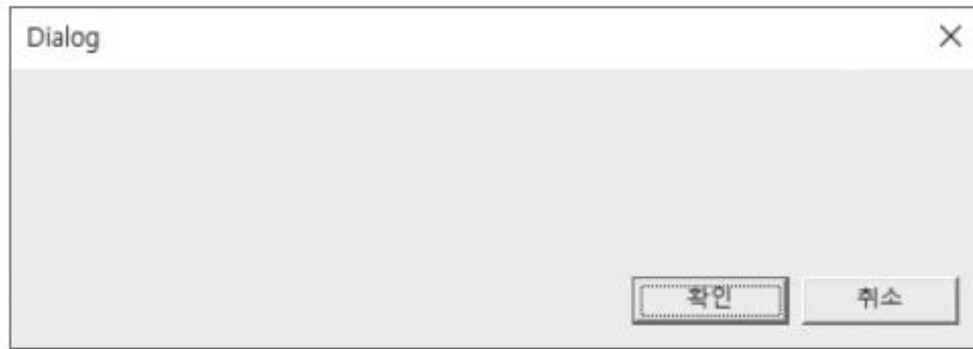


그림 10-9 간단한 대화상자 기반 응용 프로그램

대화상자 기반 응용 프로그램 (4)

■ 예제 코드

```
#include <windows.h>
❶ #include "resource.h"

❷ INT_PTR CALLBACK DlgProc(HWND, UINT, WPARAM, LPARAM);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    LPSTR lpCmdLine, int nCmdShow)
{
    ❸ DialogBox(hInstance, MAKEINTRESOURCE(IDD_DIALOG1), NULL, DlgProc);
    return 0;
}

❹ INT_PTR CALLBACK DlgProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch (uMsg) {
        ❺ case WM_INITDIALOG:
            return TRUE;
        ❻ case WM_COMMAND:
            switch (LOWORD(wParam)) {
                case IDOK:
```

대화상자 기반 응용 프로그램 (5)

■ 예제 코드(계속)

```
7 EndDialog(hDlg, IDOK);  
    return TRUE;  
case IDCANCEL:  
7 EndDialog(hDlg, IDCANCEL);  
    return TRUE;  
}  
return FALSE;  
}  
return FALSE;  
}
```

대화상자 기반 응용 프로그램 작성

■ 간단한 대화상자 기반 응용 프로그램

- 에디트 컨트롤과 버튼 컨트롤을 포함



대화상자 기반 응용 프로그램 작성과 테스트

■ 실습 10-3 대화상자 기반 응용 프로그램 작성과 테스트

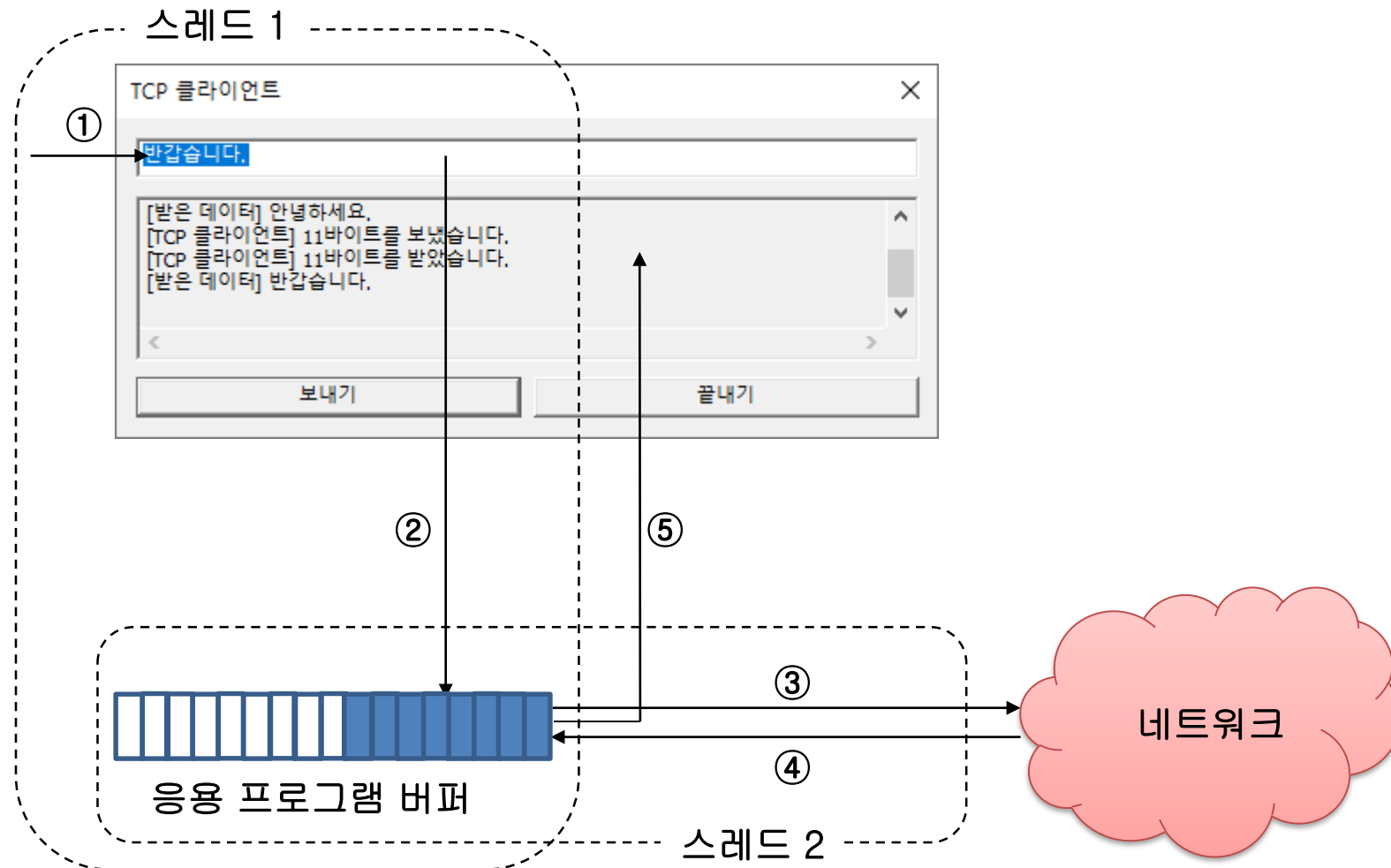
- DlgApp.cpp
- <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter10/DlgApp/DlgApp.cpp>

04 대화상자 기반 소켓 응용 프로그램



대화상자 기반 소켓 응용 프로그램

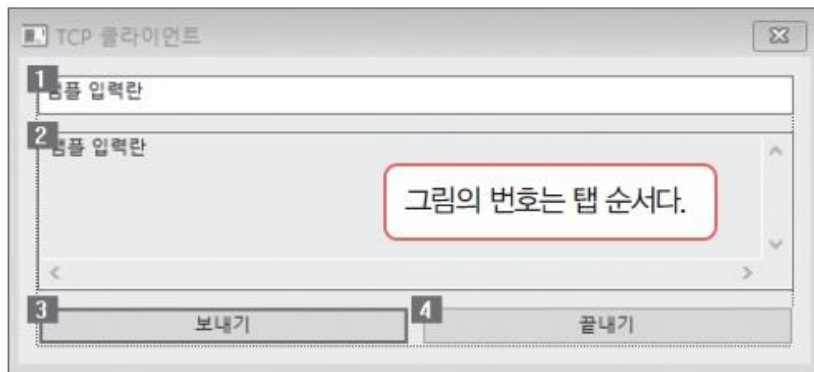
■ 스레드 동기화가 필요한 상황



대화상자 기반 소켓 응용 프로그램 작성 (1)

■ 실습 10-4 대화상자 기반 소켓 응용 프로그램 작성과 테스트

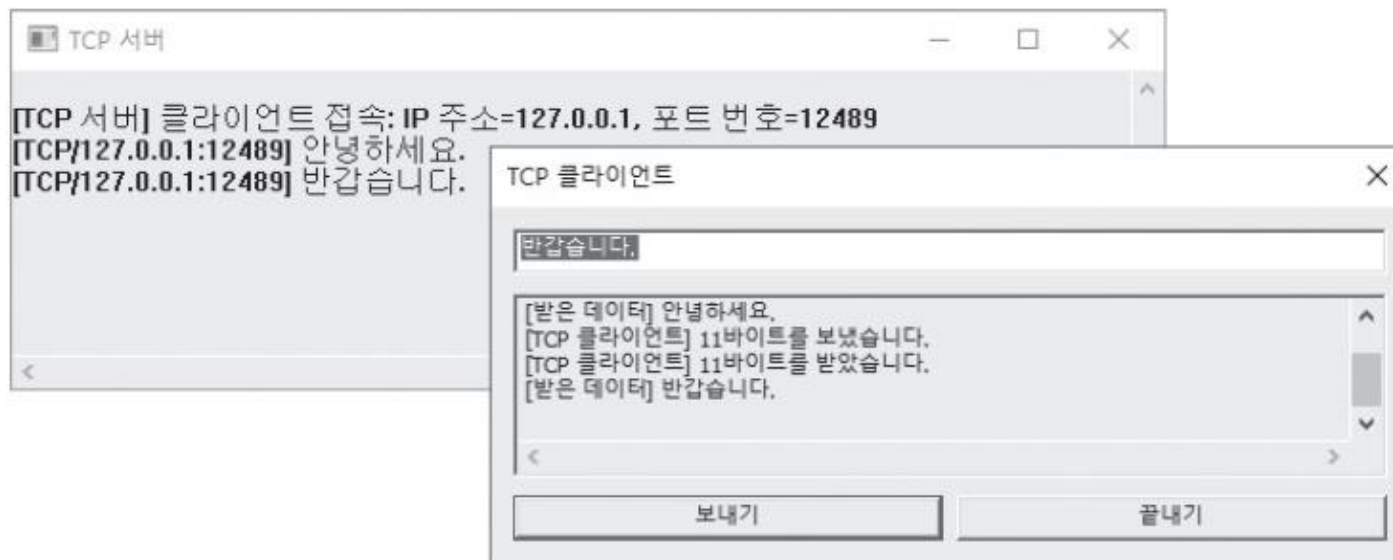
- ① GUITCPCClient.cpp 작성
- ② 대화상자 디자인



변경 대상	변경 내용
대화상자 자체	Caption을 'Dialog'에서 'TCP 클라이언트'로 바꾼다.
3번 컨트롤	Caption을 '확인'에서 '보내기'로 바꾼다.
4번 컨트롤	Caption을 '취소'에서 '끝내기'로 바꾼다.

대화상자 기반 소켓 응용 프로그램 작성 (2)

③ 실행



대화상자 기반 소켓 응용 프로그램 작성 (3)

- 실습 10-4 대화상자 기반 소켓 응용 프로그램 작성과 테스트
 - GUITCPClient.cpp
 - <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter10/GUITCPClient/GUITCPClient.cpp>