

Chapter 05 데이터 전송하기

응용 프로그램끼리 어떻게 주고받는지 설계

- 응용 프로그램 프로토콜의 필요성과 메시지 설계 방식을 이해한다.
- 데이터 전송 시 고려 사항을 파악한다.
- 다양한 데이터 전송 방식을 이해하고 활용한다.

목차

01 응용 프로그램 프로토콜과 데이터 전송

02 다양한 데이터 전송 방식

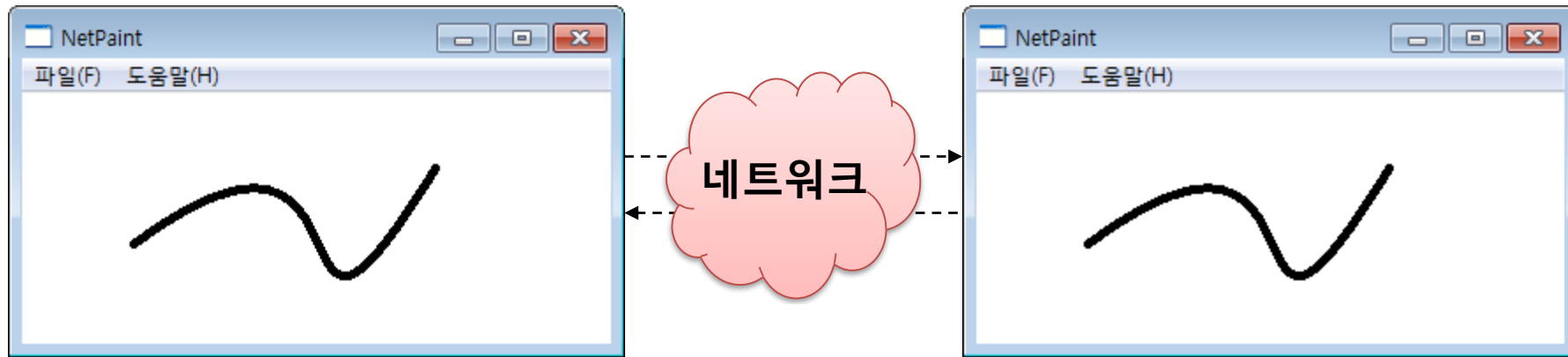
01 응용 프로그램 프로토콜과 데이터 전송



응용 프로그램 프로토콜과 데이터 전송 (1)

■ 응용 프로그램 프로토콜

- 응용 프로그램 수준에서 주고받는 데이터의 형식과 의미, 처리 방식을 정의한 프로토콜
- 예 : 네트워크 그림판 프로그램



응용 프로그램 프로토콜과 데이터 전송 (2)

- 주고받아야 할 정보
 - 직선의 시작과 끝점
 - 선의 두께와 색상
- C 언어의 구조체로 표현

```
struct DrawingMessage1
{
    int x1, y1;    // 직선의 시작점
    int x2, y2;    // 직선의 끝점
    int width;     // 선의 두께
    int color;     // 선의 색상
};
```

응용 프로그램 프로토콜과 데이터 전송 (3)

- 원 그리기 기능 추가
 - 원의 중심 좌표
 - 원의 반지름
 - 내부 채우기 색상
 - 테두리 두께와 색상
- C 언어의 구조체로 표현

```
struct DrawingMessage2
{
    int x, y;           // 원의 중심 좌표
    int r;              // 원의 반지름
    int fillcolor;      // 내부 채우기 색상
    int width;          // 테두리 두께
    int color;          // 테두리 색상
};
```

응용 프로그램 프로토콜과 데이터 전송 (4)

- 메시지 타입을 구분할 수 있도록 필드 추가!

```
struct DrawingMessage1
{
    int type;          // = LINE
    int x1, y1;        // 직선의 시작점
    int x2, y2;        // 직선의 끝점
    int width;         // 선의 두께
    int color;         // 선의 색상
};

struct DrawingMessage2
{
    int type;          // = CIRCLE
    int x, y;          // 원의 중심 좌표
    int r;             // 원의 반지름
    int fillcolor;     // 내부 채우기 색상
    int width;         // 테두리 두께
    int color;         // 테두리 색상
};
```


응용 프로그램 프로토콜과 데이터 전송 (5)

■ 메시지 경계 구분

- ① 송신자는 항상 고정 길이 데이터를 보내고, 수신자는 항상 고정 길이 데이터를 읽음
→ 예) password, 1b
- ② 송신자는 가변 길이 데이터를 보내고 끝부분에 특별한 표시(EOR, End Of Record)를 붙임.
수신자는 EOR이 나올 때까지 데이터를 읽음
NULL → 보내고자 할 데이터가 없거나 될 리는 없는 경우
- ③ 송신자는 보낼 데이터 크기를 고정 길이 데이터로 보내고, 이어서 가변 길이 데이터를 보냄.
수신자는 고정 길이 데이터를 읽어서 뒤따라올 가변 데이터의 길이를 알아내고, 이 길이만큼 데이터를 읽음
→ 크기-이-크리 but 바르다 (데이터 사이즈가 변할 수 있음)
- ④ 송신자는 가변 길이 데이터 전송한 후 연결을 정상 종료. 수신자는 recv() 함수의 리턴값이 0(정상 종료)이 될 때까지 데이터를 읽음
→ 밀방 선택으로 받기 시작할 때

응용 프로그램 프로토콜과 데이터 전송 (6)

■ 메시지 경계 구분

■ 방법 ①

- 주고받을 데이터의 길이 변동폭이 크지 않을 때 적합

■ 방법 ②

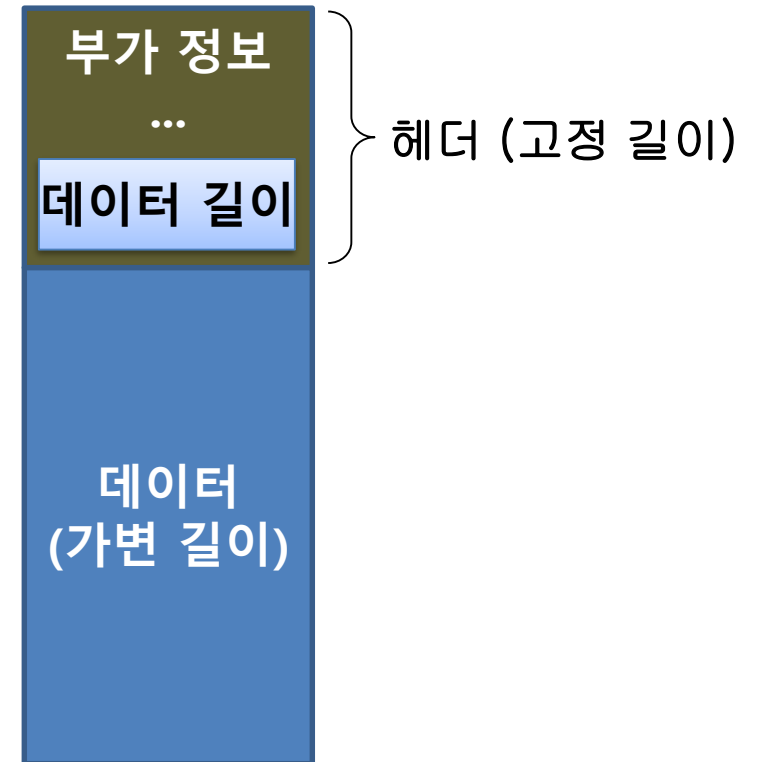
- 생성 데이터의 길이를 미리 알 수 없을 때 적합

■ 방법 ③

- 생성 데이터의 길이를 미리 알고 있는 상황에서 구현이 쉽고 처리 효율성이 높음

■ 방법 ④

- 한쪽에서 일방적으로 가변 길이 데이터를 보낼 때 적합

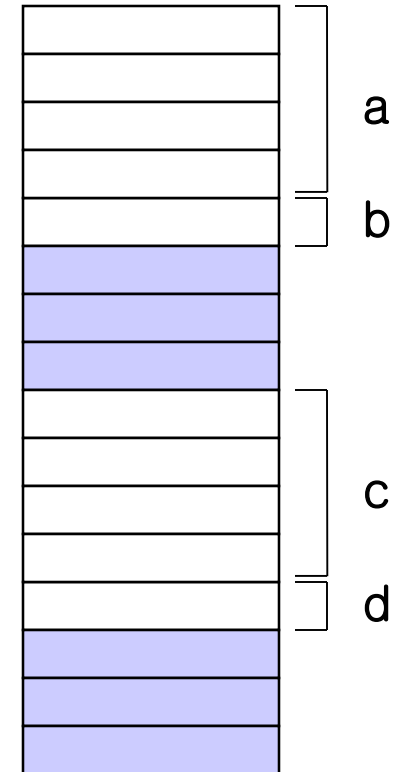


↑ 방법 ③을 사용할 때의 메시지 구조

응용 프로그램 프로토콜과 데이터 전송 (7)

■ 구조체 멤버 맞춤의 예 - 초기 상태

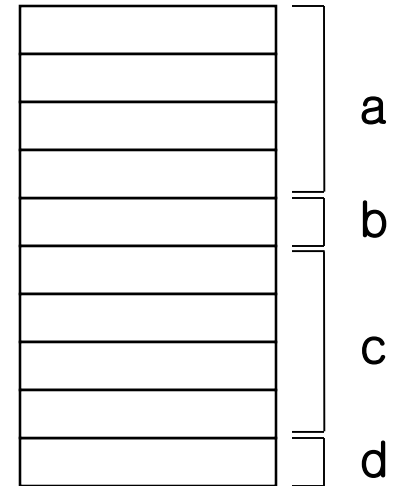
```
struct MyMessage
{
    int a;    // 4바이트
    char b;   // 1바이트
    int c;    // 4바이트
    char d;   // 1바이트
};
MyMessage msg;
...
send(sock, (char *)&msg, sizeof(msg), 0);
```



응용 프로그램 프로토콜과 데이터 전송 (8)

■ 구조체 멤버 맞춤의 예 - #pragma pack 적용 후 [윈도우]

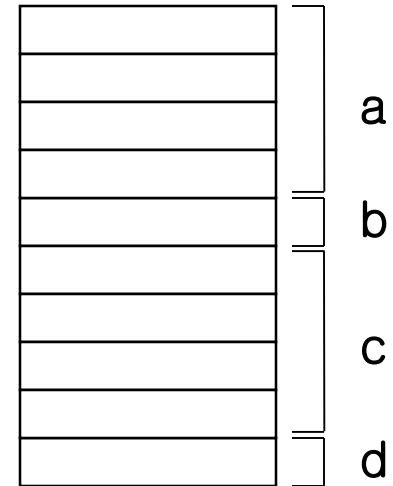
```
#pragma pack(1)
struct MyMessage
{
    int a;    // 4바이트
    char b;   // 1바이트
    int c;    // 4바이트
    char d;   // 1바이트
};
#pragma pack()
MyMessage msg;
...
send(sock, (char *)&msg, sizeof(msg), 0);
```



응용 프로그램 프로토콜과 데이터 전송 (9)

■ 구조체 멤버 맞춤의 예 - `__attribute__((packed))` 적용 후 [리눅스]

```
struct __attribute__((packed)) MyMessage
{
    int  a;  // 4바이트
    char b;  // 1바이트
    int  c;  // 4바이트
    char d;  // 1바이트
};
struct MyMessage msg;
...
send(sock, (char *)&msg, sizeof(msg), 0);
```



02 다양한 데이터 전송 방식



다양한 데이터 전송 방식 (1)

■ 고정 길이 데이터 전송

- 서버와 클라이언트 모두 크기가 같은 버퍼를 정의해두고 데이터를 주고받음

■ 실습 5-1 고정 길이 데이터 전송 연습



The image shows two overlapping Windows command prompt windows. The top window, titled 'C:\Source\Windows\Chapter05\TCPServer_Fixed\x64\Debug\TCPServer_Fixed.exe', displays the server's output. It shows a client connection from IP 127.0.0.1 on port 13574, followed by four lines of received data: '안녕하세요', '반가워요', '오늘따라 할 이야기가 많을 것 같네요', and '저도 그렇네요'. The connection then ends. The bottom window, titled 'Microsoft Visual Studio 디버그 콘솔', shows the client's output. It displays four lines of sent data, each '50바이트를 보냈습니다.', followed by a message indicating the client process (9580) has terminated due to code 0, with a prompt to press any key to close the window.

```
C:\Source\Windows\Chapter05\TCPServer_Fixed\x64\Debug\TCPServer_Fixed.exe
[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=13574
[TCP/127.0.0.1:13574] 안녕하세요#####
[TCP/127.0.0.1:13574] 반가워요#####
[TCP/127.0.0.1:13574] 오늘따라 할 이야기가 많을 것 같네요#####
[TCP/127.0.0.1:13574] 저도 그렇네요#####
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=13574

Microsoft Visual Studio 디버그 콘솔
[TCP 클라이언트] 50바이트를 보냈습니다.
[TCP 클라이언트] 50바이트를 보냈습니다.
[TCP 클라이언트] 50바이트를 보냈습니다.
[TCP 클라이언트] 50바이트를 보냈습니다.

C:\Source\Windows\Chapter05\TCPClient_Fixed\x64\Debug\TCPClient_Fixed.exe(9580
프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

다양한 데이터 전송 방식 (2)

■ 실습 5-1 고정 길이 데이터 전송 연습

- TCPServer_Fixed.cpp
 - [윈도우] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter05/TCPServer_Fixed/TCPServer_Fixed.cpp
 - [리눅스] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter05/TCPServer_Fixed.cpp
- TCPClient_Fixed.cpp
 - [윈도우] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter05/TCPClient_Fixed/TCPClient_Fixed.cpp
 - [리눅스] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter05/TCPClient_Fixed.cpp

다양한 데이터 전송 방식 (3)

■ 가변 길이 데이터 전송

- 가변 길이 데이터 경계를 구분하기 위해 EOR로 사용할 데이터 패턴을 정해야 함
 - 흔히 '\n'이나 '\r\n'을 사용
- 예) '\n'을 검출하는 가상 코드

성능 저하 요인!

```
while(1){
```

→ 소켓 수신 버퍼에서 1바이트 데이터를 읽는다.

읽은 데이터가 '\n'이 아니면 응용 프로그램 버퍼에 저장한다.

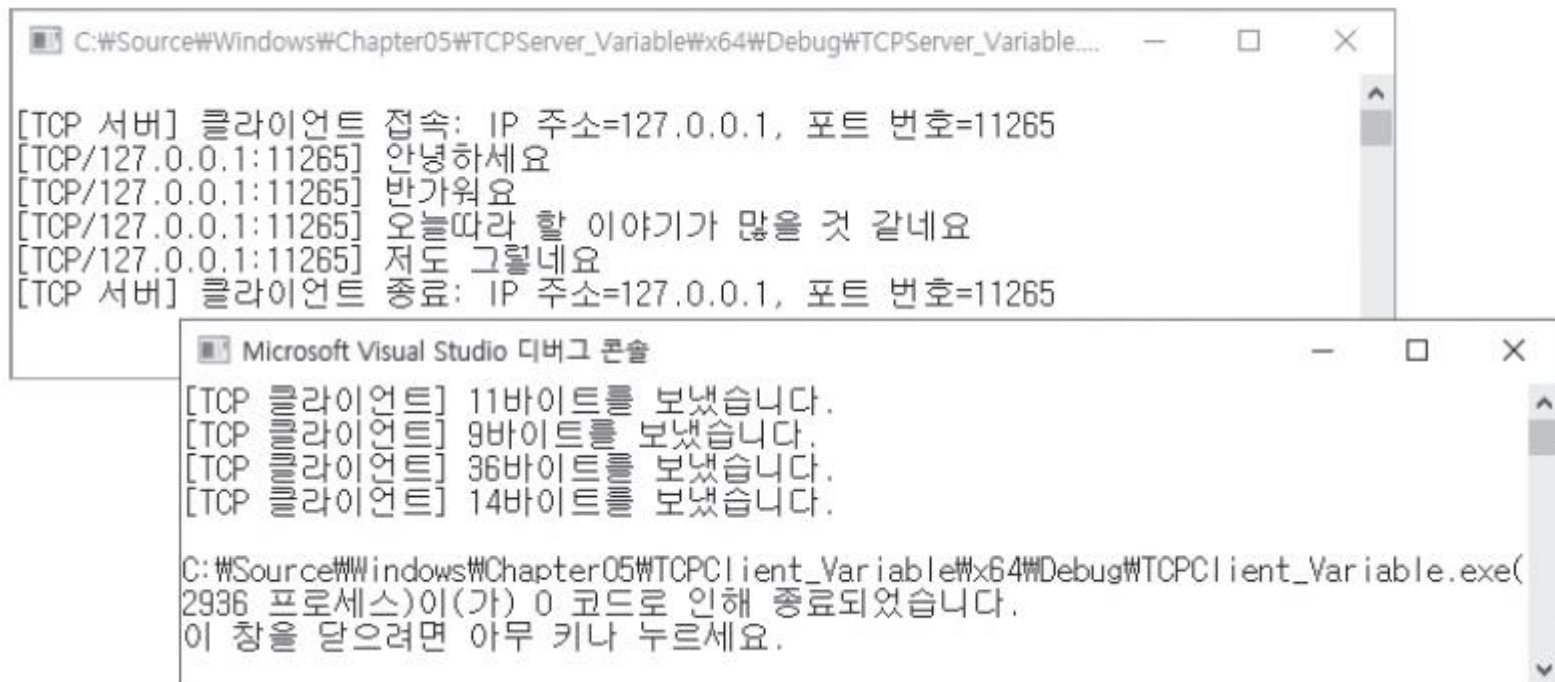
읽은 데이터가 '\n'이면 루프를 빠져나온다.

```
}
```

응용 프로그램 버퍼에 저장된 데이터를 사용한다.

다양한 데이터 전송 방식 (4)

■ 실습 5-2 가변 길이 데이터 전송 연습



The image shows two overlapping Windows command prompt windows. The top window, titled 'C:\Source\Windows\Chapter05\TCPServer_Variable\Debug\TCPServer_Variable...', displays the output of a TCP server application. It shows a client connection from IP 127.0.0.1 on port 11265, followed by four lines of received data: '안녕하세요', '반가워요', '오늘따라 할 이야기가 많을 것 같네요', and '저도 그렇네요'. The connection then ends. The bottom window, titled 'Microsoft Visual Studio 디버그 콘솔', shows the output of a TCP client application. It displays four lines of sent data: '11바이트를 보냈습니다.', '9바이트를 보냈습니다.', '36바이트를 보냈습니다.', and '14바이트를 보냈습니다.'. Below this, it shows the client application (TCPCClient_Variable.exe) ending with a return code of 0.

```
C:\Source\Windows\Chapter05\TCPServer_Variable\Debug\TCPServer_Variable...  
[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=11265  
[TCP/127.0.0.1:11265] 안녕하세요  
[TCP/127.0.0.1:11265] 반가워요  
[TCP/127.0.0.1:11265] 오늘따라 할 이야기가 많을 것 같네요  
[TCP/127.0.0.1:11265] 저도 그렇네요  
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=11265  
  
Microsoft Visual Studio 디버그 콘솔  
[TCP 클라이언트] 11바이트를 보냈습니다.  
[TCP 클라이언트] 9바이트를 보냈습니다.  
[TCP 클라이언트] 36바이트를 보냈습니다.  
[TCP 클라이언트] 14바이트를 보냈습니다.  
  
C:\Source\Windows\Chapter05\TCPCClient_Variable\Debug\TCPCClient_Variable.exe(  
2936 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.
```

다양한 데이터 전송 방식 (5)

■ 실습 5-2 가변 길이 데이터 전송 연습

- TCPServer_Variable.cpp
 - [윈도우] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter05/TCPServer_Variable/TCPServer_Variable.cpp
 - [리눅스] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter05/TCPServer_Variable.cpp
- TCPClient_Variable.cpp
 - [윈도우] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter05/TCPClient_Variable/TCPClient_Variable.cpp
 - [리눅스] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter05/TCPClient_Variable.cpp

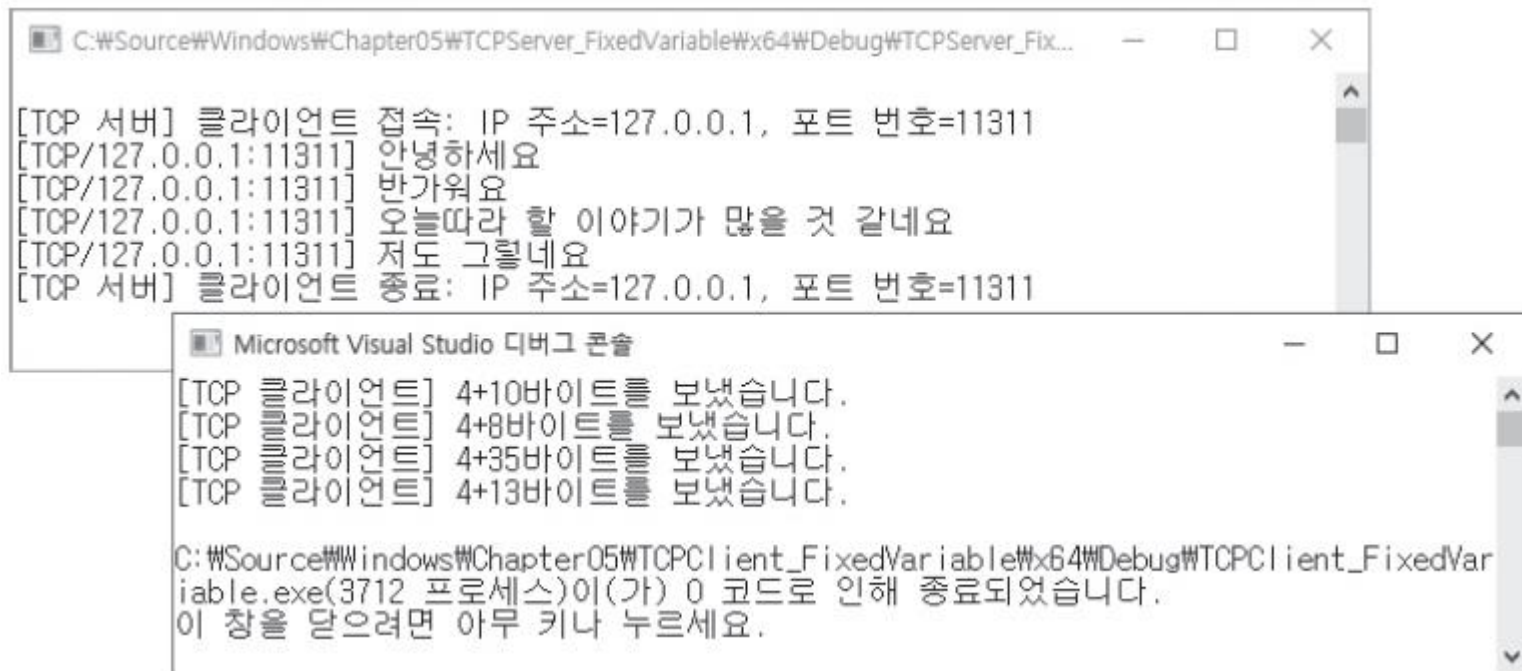
다양한 데이터 전송 방식 (6)

■ 고정 길이+가변 길이 데이터 전송

- 송신 측에서 가변 길이 데이터의 크기를 미리 계산할 수 있다면 '고정 길이 + 가변 길이 데이터' 전송이 효과적
- 수신 측에서는 ① 고정 길이 데이터 수신 ② 가변 길이 데이터 수신, 두 번의 데이터 읽기 작업으로 가변 길이 데이터의 경계를 구분해 읽을 수 있음

다양한 데이터 전송 방식 (7)

■ 실습 5-3 고정 길이 + 가변 길이 데이터 전송 연습



```
C:\Source\Windows\Chapter05\TCPServer_FixedVariable\x64\Debug\TCPServer_Fix...  
[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=11311  
[TCP/127.0.0.1:11311] 안녕하세요  
[TCP/127.0.0.1:11311] 반가워요  
[TCP/127.0.0.1:11311] 오늘따라 할 이야기가 많을 것 같네요  
[TCP/127.0.0.1:11311] 저도 그럴네요  
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=11311  
  
Microsoft Visual Studio 디버그 콘솔  
[TCP 클라이언트] 4+10바이트를 보냈습니다.  
[TCP 클라이언트] 4+8바이트를 보냈습니다.  
[TCP 클라이언트] 4+35바이트를 보냈습니다.  
[TCP 클라이언트] 4+13바이트를 보냈습니다.  
  
C:\Source\Windows\Chapter05\TCPClient_FixedVariable\x64\Debug\TCPClient_FixedVar  
iable.exe(3712 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.
```

다양한 데이터 전송 방식 (8)

■ 실습 5-3 고정 길이 + 가변 길이 데이터 전송 연습

- TCPServer_FixedVariable.cpp
- [윈도우] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter05/TCPServer_FixedVariable/TCPServer_FixedVariable.cpp
- [리눅스] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter05/TCPServer_FixedVariable.cpp
- TCPClient_FixedVariable.cpp
- [윈도우] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter05/TCPClient_FixedVariable/TCPClient_FixedVariable.cpp
- [리눅스] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter05/TCPClient_FixedVariable.cpp

다양한 데이터 전송 방식 (9)

■ 실습 5-4 데이터 전송 후 종료 연습

- 일종의 가변 길이 데이터 전송 방식
 - EOR로 특별한 데이터 패턴 대신 연결 종료를 사용



The image shows two overlapping Windows command prompt windows. The top window, titled 'C:\Source\Windows\Chapter05\TCPServer_CloseOnTransfer\x64\Debug\TCPServer_...', displays logs for a TCP server. It shows four successful connections from IP 127.0.0.1 on ports 11408, 11409, 11410, and 11411. Each connection includes a greeting from the client and a corresponding response from the server. The bottom window, titled 'Microsoft Visual Studio 디버그 콘솔', shows logs for a TCP client. It reports sending 10, 8, 35, and 13 bytes of data to the server. At the bottom of this window, a message states that the application (TCPCClient_CloseOnTransfer.exe) has terminated with code 0.

```
C:\Source\Windows\Chapter05\TCPServer_CloseOnTransfer\x64\Debug\TCPServer_...
[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=11408
[TCP/127.0.0.1:11408] 안녕하세요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=11408

[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=11409
[TCP/127.0.0.1:11409] 반가워요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=11409

[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=11410
[TCP/127.0.0.1:11410] 오늘따라 할 이야기가 많을 것 같네요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=11410

[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=11411
[TCP/127.0.0.1:11411] 저도 그럴네요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=11411

Microsoft Visual Studio 디버그 콘솔
[TCP 클라이언트] 10바이트를 보냈습니다.
[TCP 클라이언트] 8바이트를 보냈습니다.
[TCP 클라이언트] 35바이트를 보냈습니다.
[TCP 클라이언트] 13바이트를 보냈습니다.

C:\Source\Windows\Chapter05\TCPCClient_CloseOnTransfer\x64\Debug\TCPCClient_CloseOnTransfer.exe(9316 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

다양한 데이터 전송 방식 (10)

■ 실습 5-4 데이터 전송 후 종료 연습

- TCPServer_CloseOnTransfer.cpp
- [윈도우] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter05/TCPServer_CloseOnTransfer/TCPServer_CloseOnTransfer.cpp
- [리눅스] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter05/TCPServer_CloseOnTransfer.cpp

- TCPClient_CloseOnTransfer.cpp
- [윈도우] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter05/TCPClient_CloseOnTransfer/TCPClient_CloseOnTransfer.cpp
- [리눅스] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter05/TCPClient_CloseOnTransfer.cpp