

Global KPU!

글로벌 경쟁력을 갖춘 산업기술 명문대학
세계를 향해 더 큰 미래를 펼쳐갑니다

5 데이터 전송하기

네트워크 게임 프로그래밍

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

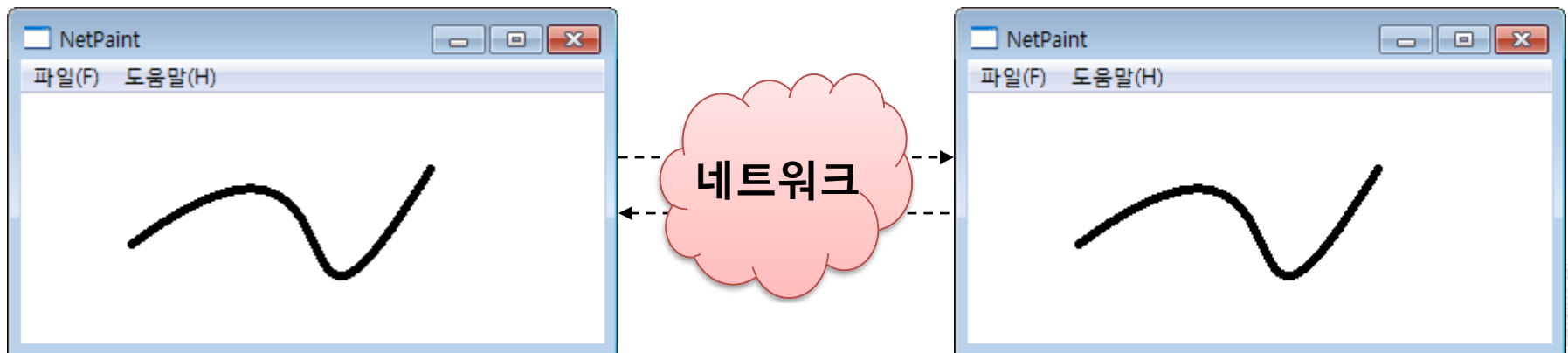


❖ 응용 프로그램 프로토콜

- 응용 프로그램 수준에서 주고받는 데이터의 형식과 의미 그리고 처리 방식을 정의한 프로토콜 (개발자 정의 프로토콜)
- 개발자가 가장 많이 사용할 프로토콜 (서버와 클라이언트간의 약속된 규약/규칙)
- 소켓 프로토콜처럼 규격화(표준화)되어 있지 않음
- 소켓 함수를 이용하여 구현하면 됨

❖ 응용 프로그램 프로토콜의 예

- 네트워크 그림판 프로그램
 - 직선의 시작과 끝점
 - 선의 두께와 색상



❖ 메시지 정의 ①

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

❖ 메시지 정의 ②

```
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
};
```



❖ 메시지 정의 ③

- 원인지 직선인지 구분할 수 없으므로 타입을 나타내는 필드를 추가
- int 를 읽으면 직선인지 원 데이터인지 알 수 가 있음

```
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
};
```



❖ 메시지 경계 구분 방법

- ① 송신자는 항상 고정 길이 데이터를 보냄. 수신자는 항상 고정 길이 데이터를 읽음
- ② 송신자는 가변 길이 데이터를 보내고 끝 부분에 특별한 표시(EOR, End Of Record)를 붙임. 수신자는 EOR이 나올 때까지 데이터를 읽음
- 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
- ④ 송신자는 가변 길이 데이터 전송 후 접속을 정상 종료. 수신자는 `recv()` 함수의 리턴 값이 0(=정상 종료)이 될 때까지 데이터를 읽음
- ⑤ 기타: 파일 데이터등으로 전송



❖ 메시지 경계 구분 방법

■ 방법 ①

- 주고받을 데이터의 길이 변동폭이 크지 않을 경우에 적합
- 가장 긴 데이터를 기준으로 고정길이를 정해야 하므로 확장성이나 효율성에서 떨어짐

■ 방법 ②

- 생성될 데이터의 길이를 미리 알 수 없을 때 적합
- 데이터 중간에 EOR(End Of Record)과 똑같은 패턴이 들어있으면 데이터가 완전히 전송되지 않음
- 데이터를 효율적으로 수신하는 방식의 알고리즘이 없으면 성능이 떨어짐(한바이트씩 읽어드리므로)

■ 방법 ③

- 일반적으로 권장. 구현의 편의성과 처리 효율면에서 유리

■ 방법 ④

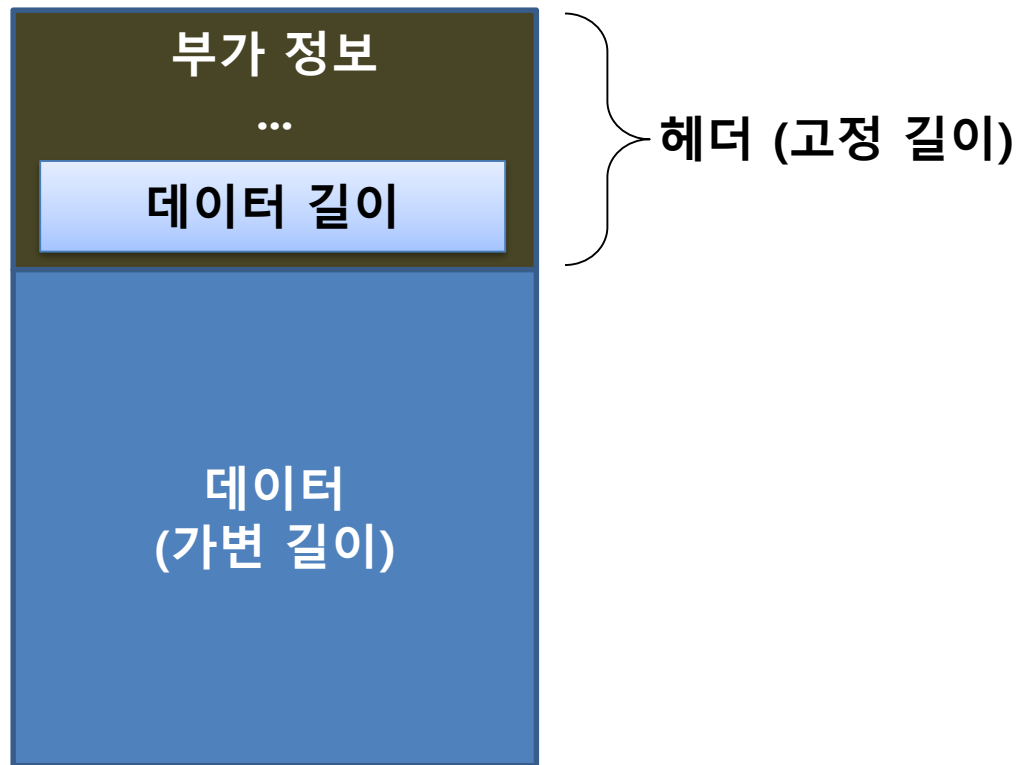
- 한쪽에서 일방적으로 데이터를 보내는 경우에 적합
- 소켓 연결 설정/종료를 반복해야 하므로 비효율적임

■ 방법 ⑤

- 파일 데이터를 읽어 전송하고 저장하는 경우에 적합



❖ 메시지 구조의 예 - 방법 ③을 사용할 경우



❖ 바이트 정렬

- 빅 엔디안 방식으로 통일

❖ 구조체 멤버 맞춤

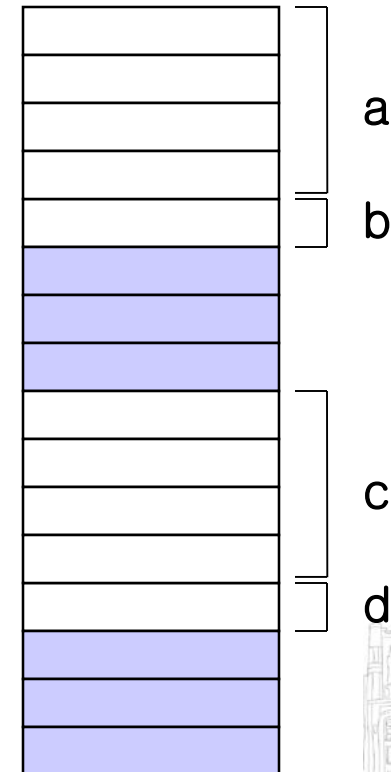
- 구조체(C++의 클래스도 포함) 멤버의 메모리 시작 주소를 결정하는 컴파일러의 규칙
- #pragma pack 지시자 사용



❖ 구조체 멤버 맞춤의 예 - #pragma pack 적용 전

- 일반적으로는 10(4+1+4+1)바이트가 아닌 16바이트가 전송됨
- 구조체 멤버 맞춤은 이러한 메모리의 낭비를 줄여주기 위해 사용

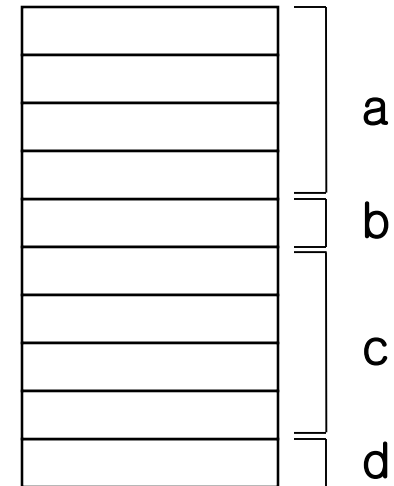
```
struct MyMessage
{
    int a;    // 4바이트
    char b;   // 1바이트
    int c;    // 4바이트
    char d;   // 1바이트
};
MyMessage msg;
...
send(sock, (char *)&msg, sizeof(msg), 0);
```



❖ 구조체 멤버 맞춤의 예 - #pragma pack 적용 후

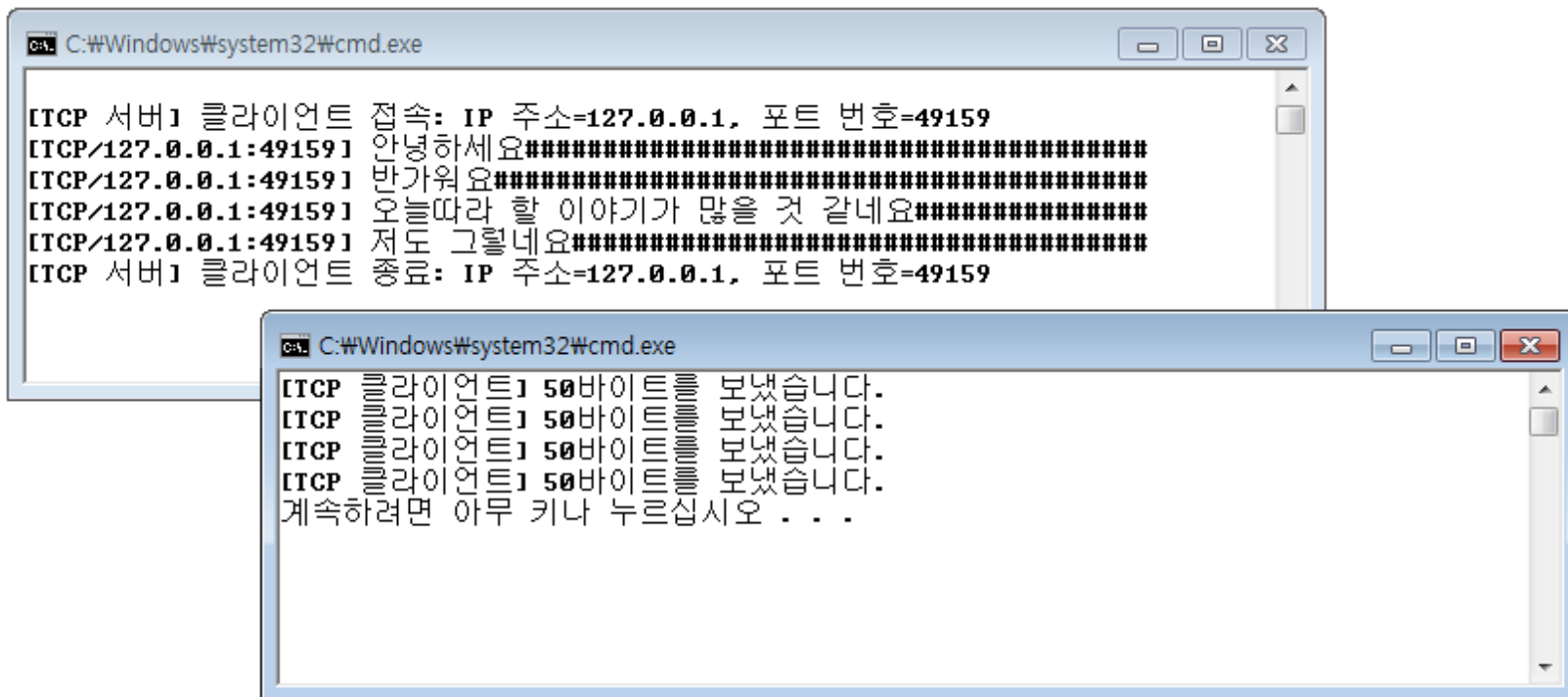
- 근래엔 많이 사용하지 않음

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



❖ 고정 길이 데이터 전송

- 서버와 클라이언트 모두 크기가 같은 버퍼를 정의해두고 데이터를 주고 받도록 구성
- 기본 TCP Server/Client 코드를 기본으로 하되, 간단한 구현을 위해 클라이언트에서 서버로 데이터를 보내는 방식으로 구현
- 루프를 돌면서 길이가 다른 문자열 데이터 4개를 보냄



```
C:\Windows\system32\cmd.exe

[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=49159
[TCP/127.0.0.1:49159] 안녕하세요#####
[TCP/127.0.0.1:49159] 반가워요#####
[TCP/127.0.0.1:49159] 오늘따라 할 이야기가 많을 것 같네요#####
[TCP/127.0.0.1:49159] 저도 그럴네요#####
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=49159

C:\Windows\system32\cmd.exe

[TCP 클라이언트] 50바이트를 보냈습니다.
[TCP 클라이언트] 50바이트를 보냈습니다.
[TCP 클라이언트] 50바이트를 보냈습니다.
[TCP 클라이언트] 50바이트를 보냈습니다.
계속하려면 아무 키나 누르십시오 . . .
```

실습 5-1 고정 길이 데이터 전송 p138~

예) TCPServer_Fixed.cpp / TCPClient_Fixed.cpp

```
C:\Windows\system32\cmd.exe

[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=3658
[TCP/127.0.0.1:3658] 안녕하세요#####
[TCP/127.0.0.1:3658] 반가워요#####
[TCP/127.0.0.1:3658] 오늘따라 할 이야기가 많을 것 같네요#####
[TCP/127.0.0.1:3658] 저도 그렇네요#####
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=3658
```

```
C:\Windows\system32\cmd.exe

[TCP 클라이언트] 50바이트를 보냈습니다.
[TCP 클라이언트] 50바이트를 보냈습니다.
[TCP 클라이언트] 50바이트를 보냈습니다.
[TCP 클라이언트] 50바이트를 보냈습니다.
계속하려면 아무 키나 누르십시오 . . .
```

❖ 가변 길이 데이터 전송

- 가변 길이 데이터 경계를 구분하기 위해 EOR로 사용할 데이터 패턴을 정해야 함
 - 흔히 '\n'(linux)이나 '\r\n'(windows)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.

- 소켓 수신 버퍼에서 데이터를 한 번에 많이 읽어 1바이트씩 리턴해주는 사용자 정의 함수가 필요!



다양한 데이터 전송 방식 (3)

❖ 가변 길이 데이터 전송

```
C:\Windows\system32\cmd.exe

[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=49161
[TCP/127.0.0.1:49161] 안녕하세요
[TCP/127.0.0.1:49161] 반가워요
[TCP/127.0.0.1:49161] 오늘따라 할 이야기가 많을 것 같네요
[TCP/127.0.0.1:49161] 저도 그럴네요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=49161

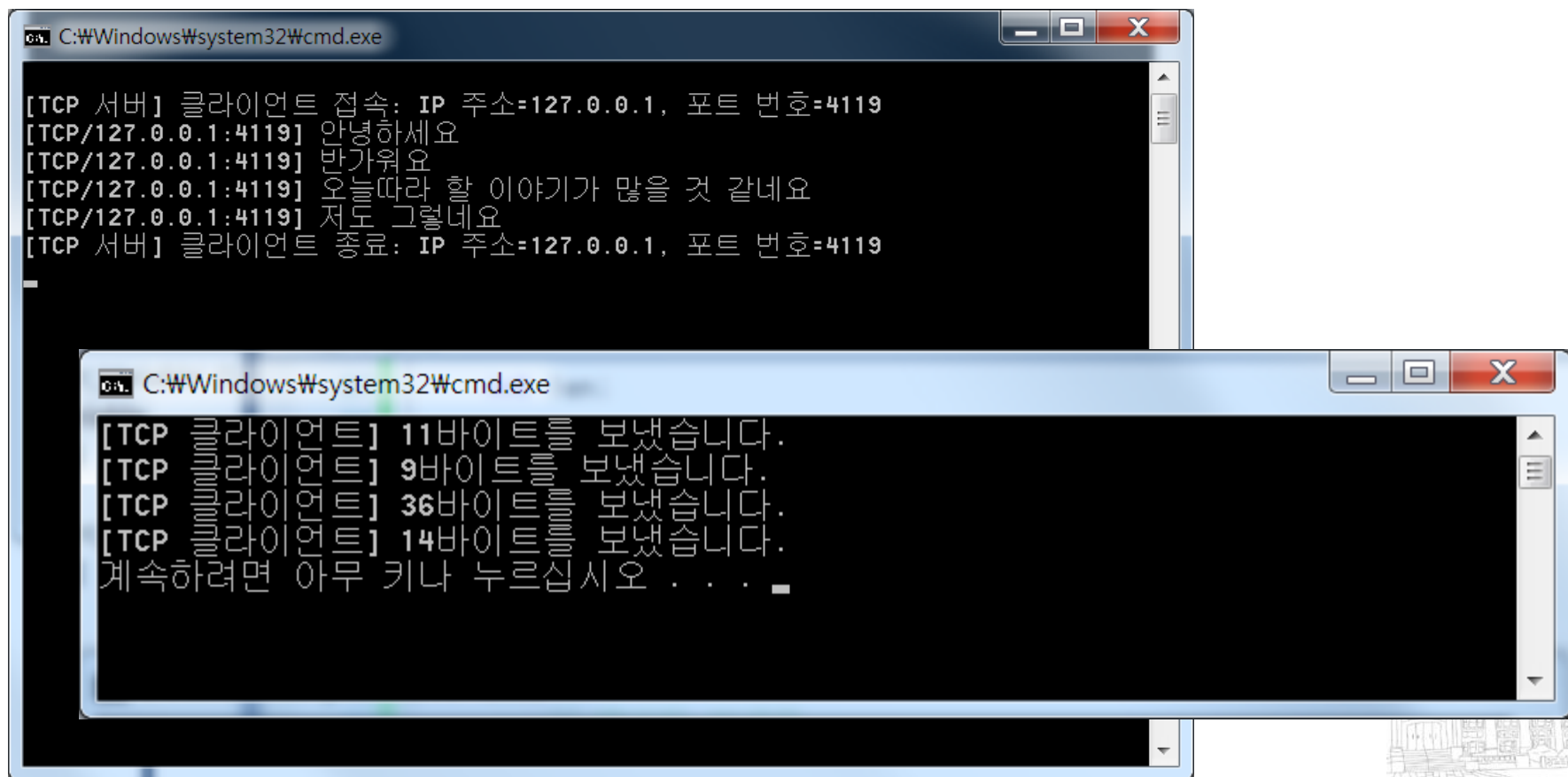
C:\Windows\system32\cmd.exe

[TCP 클라이언트] 11바이트를 보냈습니다.
[TCP 클라이언트] 9바이트를 보냈습니다.
[TCP 클라이언트] 36바이트를 보냈습니다.
[TCP 클라이언트] 14바이트를 보냈습니다.
계속하려면 아무 키나 누르십시오 . . .
```



실습 5-2 가변 길이 데이터 전송 p146~

예) TCPServer_Variable.cpp / TCPClient_Variable.cpp



The image shows two overlapping Windows command prompt windows. The top window, titled 'C:\Windows\system32\cmd.exe', displays the server's log. The bottom window, also titled 'C:\Windows\system32\cmd.exe', displays the client's log.

Top Window (Server Log):

```
[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=4119  
[TCP/127.0.0.1:4119] 안녕하세요  
[TCP/127.0.0.1:4119] 반가워요  
[TCP/127.0.0.1:4119] 오늘따라 할 이야기가 많을 것 같네요  
[TCP/127.0.0.1:4119] 저도 그렇네요  
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=4119
```

Bottom Window (Client Log):

```
[TCP 클라이언트] 11바이트를 보냈습니다.  
[TCP 클라이언트] 9바이트를 보냈습니다.  
[TCP 클라이언트] 36바이트를 보냈습니다.  
[TCP 클라이언트] 14바이트를 보냈습니다.  
계속하려면 아무 키나 누르십시오 . . .
```


❖ 고정 길이+가변 길이 데이터 전송

- 송신 측에서 가변 길이 데이터의 크기를 곧바로 계산할 수 있다면 고정 길이 + 가변 길이 전송이 효과적
- 수신 측에서는 [① 고정 길이 데이터 수신 ② 가변 길이 데이터 수신] 두 번의 데이터 수신으로 가변 길이 데이터의 경계를 구분해 읽을 수 있음



다양한 데이터 전송 방식 (5)

❖ 고정 길이+가변 길이 데이터 전송

```
C:\Windows\system32\cmd.exe

[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=49167
[TCP/127.0.0.1:49167] 안녕하세요
[TCP/127.0.0.1:49167] 반가워요
[TCP/127.0.0.1:49167] 오늘따라 할 이야기가 많을 것 같네요
[TCP/127.0.0.1:49167] 저도 그럴네요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=49167

C:\Windows\system32\cmd.exe

[TCP 클라이언트] 4+10바이트를 보냈습니다.
[TCP 클라이언트] 4+8바이트를 보냈습니다.
[TCP 클라이언트] 4+35바이트를 보냈습니다.
[TCP 클라이언트] 4+13바이트를 보냈습니다.
계속하려면 아무 키나 누르십시오 . . .
```



실습 5-3

고정 길이 + 가변 길이 데이터 전송 p154~

예) TCPServer_FixedVariable.cpp / TCPClient_FixedVariable.cpp

```
C:\Windows\system32\cmd.exe

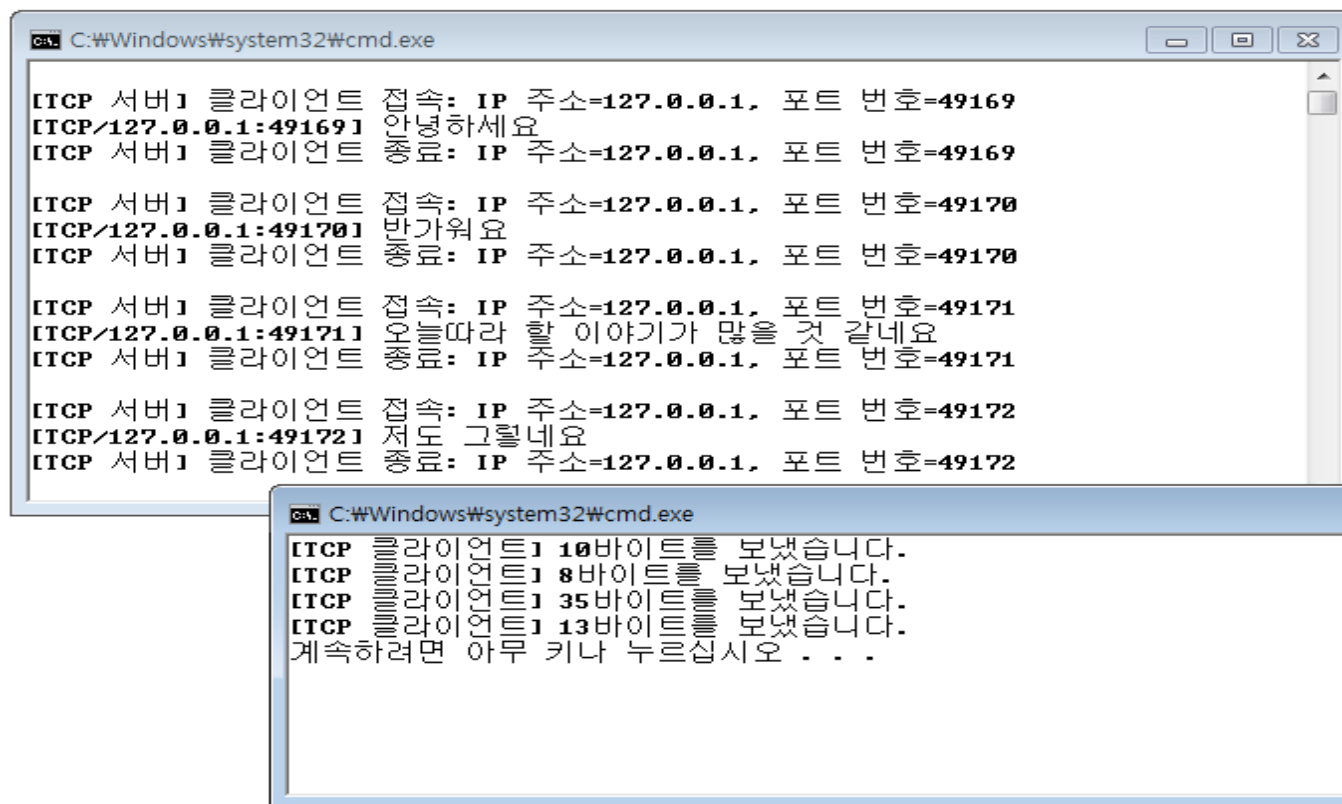
[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=5169
[TCP/127.0.0.1:5169] 안녕하세요
[TCP/127.0.0.1:5169] 반가워요
[TCP/127.0.0.1:5169] 오늘따라 할 이야기가 많을 것 같네요
[TCP/127.0.0.1:5169] 저도 그렇네요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=5169
```

```
C:\Windows\system32\cmd.exe

[TCP 클라이언트] 4+10바이트를 보냈습니다.
[TCP 클라이언트] 4+8바이트를 보냈습니다.
[TCP 클라이언트] 4+35바이트를 보냈습니다.
[TCP 클라이언트] 4+13바이트를 보냈습니다.
계속하려면 아무 키나 누르십시오 . . .
```

❖ 데이터 전송 후 종료

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



```
C:\Windows\system32\cmd.exe

[ITCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=49169
[ITCP/127.0.0.1:49169] 안녕하세요
[ITCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=49169

[ITCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=49170
[ITCP/127.0.0.1:49170] 반가워요
[ITCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=49170

[ITCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=49171
[ITCP/127.0.0.1:49171] 오늘따라 할 이야기가 많을 것 같네요
[ITCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=49171

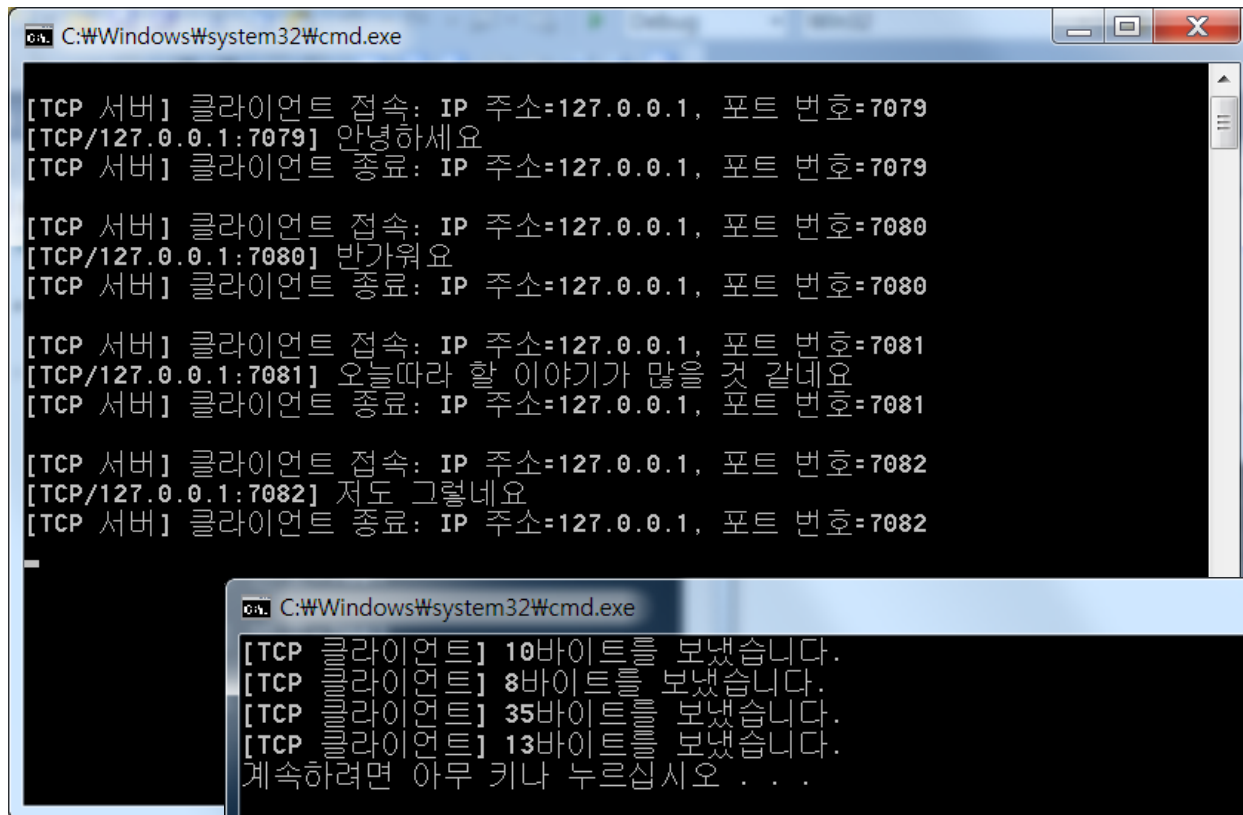
[ITCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=49172
[ITCP/127.0.0.1:49172] 저도 그럴네요
[ITCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=49172

C:\Windows\system32\cmd.exe

[ITCP 클라이언트] 10바이트를 보냈습니다.
[ITCP 클라이언트] 8바이트를 보냈습니다.
[ITCP 클라이언트] 35바이트를 보냈습니다.
[ITCP 클라이언트] 13바이트를 보냈습니다.
계속하려면 아무 키나 누르십시오 . . .
```

실습 5-4 데이터 전송 후 종료 p138~

예) TCPServer_CloseOnTransfer.cpp / TCPClient_CloseOnTransfer.cpp



The image shows two overlapping Windows command prompt windows. The top window displays the logs of a TCP server, and the bottom window displays the logs of a TCP client.

Top Window (C:\Windows\system32\cmd.exe):

```
[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=7079
[TCP/127.0.0.1:7079] 안녕하세요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=7079

[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=7080
[TCP/127.0.0.1:7080] 반가워요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=7080

[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=7081
[TCP/127.0.0.1:7081] 오늘따라 할 이야기가 많을 것 같네요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=7081

[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=7082
[TCP/127.0.0.1:7082] 저도 그럴네요
[TCP 서버] 클라이언트 종료: IP 주소=127.0.0.1, 포트 번호=7082
```

Bottom Window (C:\Windows\system32\cmd.exe):

```
[TCP 클라이언트] 10바이트를 보냈습니다.
[TCP 클라이언트] 8바이트를 보냈습니다.
[TCP 클라이언트] 35바이트를 보냈습니다.
[TCP 클라이언트] 13바이트를 보냈습니다.
계속하려면 아무 키나 누르십시오 . . .
```

연습 문제(1)

1. 고정/가변 데이터 전송 소스코드를 촬영하여 간단한 네트워크 기반 채팅 프로그램 만들기.
2. 서버, 클라이언트 모두 명령행 인자로 입력 받는다.
3. 단순한 코딩을 위해 클라이언트->서버->클라이언트->서버 방식으로만 동작하도록 구성한다.
4. 채팅 id로 서버 실행 및 클라이언트 접속이 가능하도록 구성한다.



연습 문제(2)

1. 고정/가변 데이터 전송 소스코드를 촬영하여 간단한 네트워크 기반 파일 데이터 전송 프로그램 만들기.
2. 클라이언트에서 파일 내 데이터를 읽어서 서버에 전송하는 프로그램으로 구성한다
3. 서버도 같은 파일로 저장하도록 한다.
4. 서버에서 같은 파일이 있어도 overwrite 하도록 한다.





Thank You !

oasis01@gmail.com / rhqudtn75@nate.com