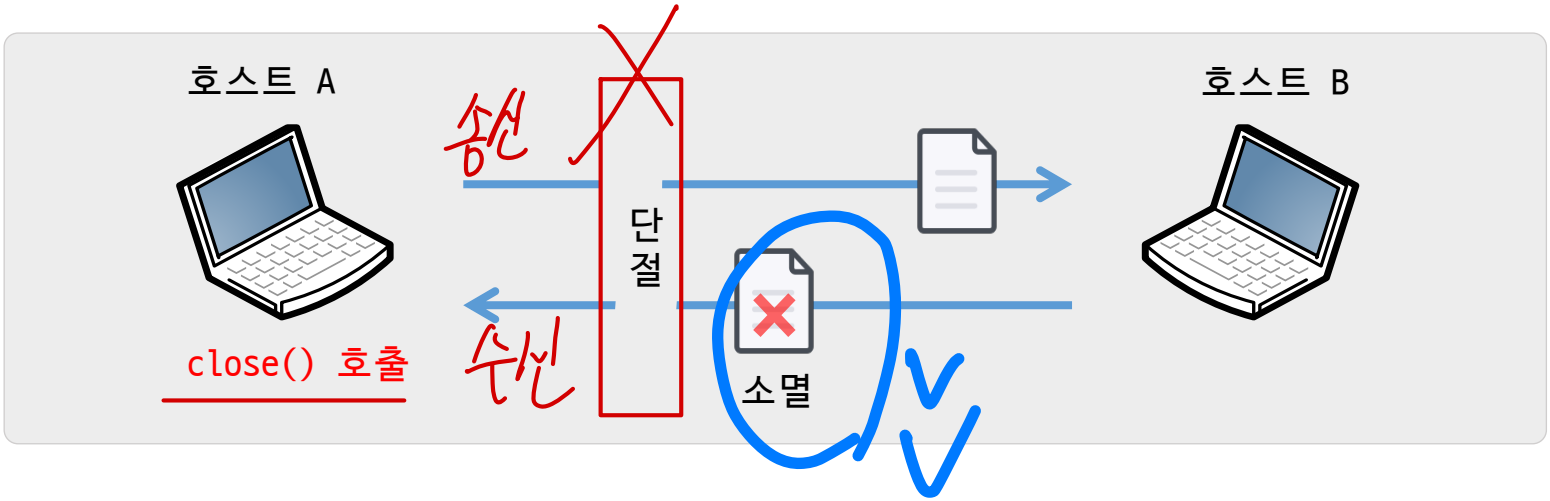
Chapter 07

TCP 기반의 Half-close

일방적인 연결 종료의 문제점

- close 및 closesocket 함수의 기능
  - 소켓의 완전한 소멸을 의미
  - 소켓이 소멸되므로 더 이상의 입출력은 불가능
  - 상대방의 상태에 상관없이 일방적인 종료
  - 상대 호스트의 데이터 송수신이 아직 완료되지 않은 상황이면, 문제가 발생할 수 있음

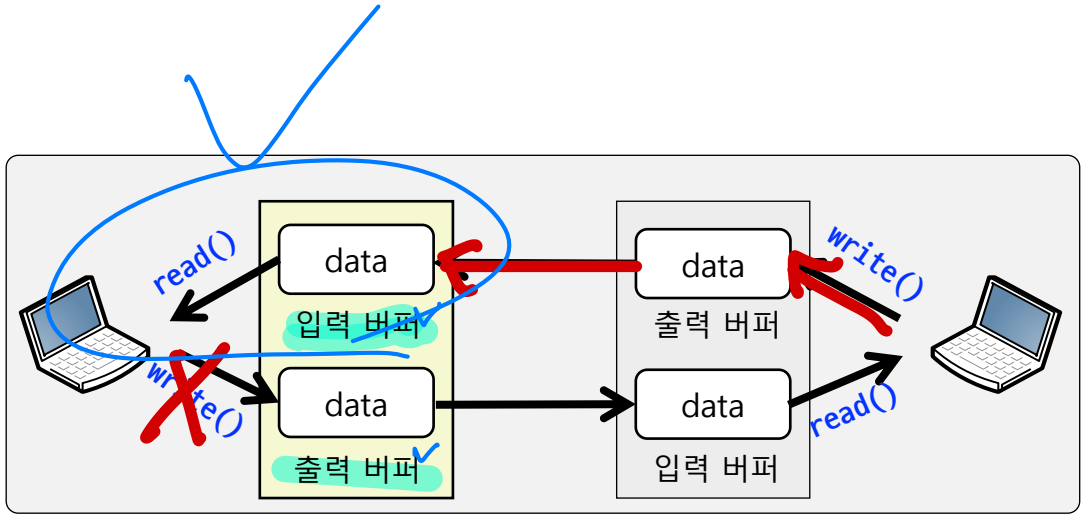
- half-close 기법
  - 일방적인 소켓 종료로 인해 발생되는 문제점 보완



소켓의 half-close

- Half-close
  - 입력 또는 출력 스트림 중 하나만 종료시키는 방법: '우아한 종료'라고 함
  - 종료를 원한다는 것은, 더 이상 전송할 데이터가 존재하지 않는 상황
    - 따라서 출력 스트림은 종료를 시켜도 됨
  - 상대방도 종료를 원하는지 확인되지 않은 상황이므로
    - 입력 스트림은 종료시키지 않음
  - 일반적으로 half-close라고 하면, 출력 스트림만 종료하는 것을 의미

- 스트림
  - 데이터의 흐름 (한쪽 방향으로만 형성)
  - 소켓의 스트림 역시 한쪽 방향으로만 데이터가 이동
  - 양방향 통신을 위해서는 두 개의 스트림이 필요함



shutdown 함수 사용과 필요성

- shutdown() 함수

```
#include <sys/socket.h>

int shutdown(int sock, int howto);
-> 성공 시 0, 실패 시 -1 반환
```

  - sock: 종료할 소켓 파일 디스크립터, howto: 종료 방법

- howto 내용
  - SHUT_RD: 입력 스트림 종료 (데이터 수신 불가능)
  - SHUT_WR: 출력 스트림 종료 (출력 버퍼에 남아 있는 데이터는 목적지로 전송됨)
  - SHUT_RDWR: 입출력 스트림 종료

- close() 함수가 호출되면 상대 호스트로 EOF가 전달됨
  - 모든 데이터 전송이 끝났다는 신호를 의미 (연결 종료 + 데이터 전송 종료)

- 출력 스트림만 종료
  - EOF가 전달됨: close() 함수 호출을 대체하고 상대 호스트의 종료를 기다릴 수 있음

Half-close 기반 파일 전송 프로그램

file_server.c 의 일부

```
while(1)
{
    read_cnt=fread((void*)buf, 1, BUF_SIZE, fp);
    if(read_cnt<BUF_SIZE)
    {
        write(clnt_sd, buf, read_cnt);
        break;
    }
    write(clnt_sd, buf, BUF_SIZE);
}

shutdown(clnt_sd, SHUT_WR);
read(clnt_sd, buf, BUF_SIZE);

printf("Message from client: %s \n", buf);

fclose(fp);
close(clnt_sd); close(serv_sd);
```

파일의 마지막 부분

전송만 종료

수신 가능

close() 대신

file_client.c 의 일부

```
while((read_cnt=read(sd, buf, BUF_SIZE ))!=0)
    fwrite((void*)buf, 1, read_cnt, fp);

puts("Received file data");
write(sd, "Thank you", 10);

fclose(fp);
close(sd);
```

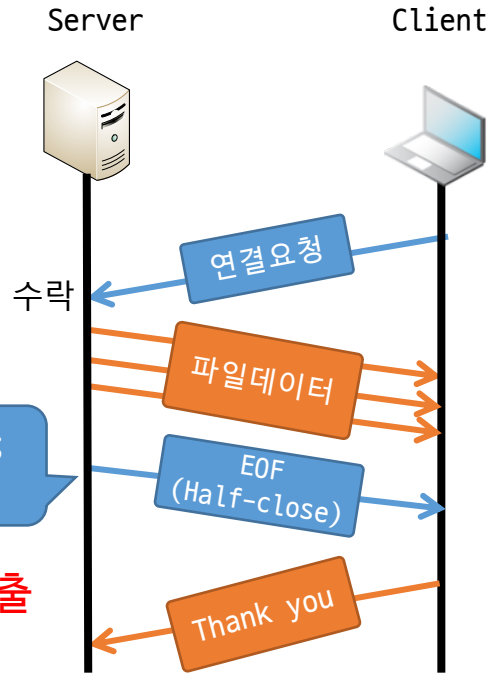
서버의 shutdown() 이후에 실행

close()하면 이제 안됨.

클라이언트가 마지막으로 전송한 데이터를 수신하기 위함.

shutdown(clnt_sd, SHUT_WR); 호출

Half-close가 필요한 상황의 연출

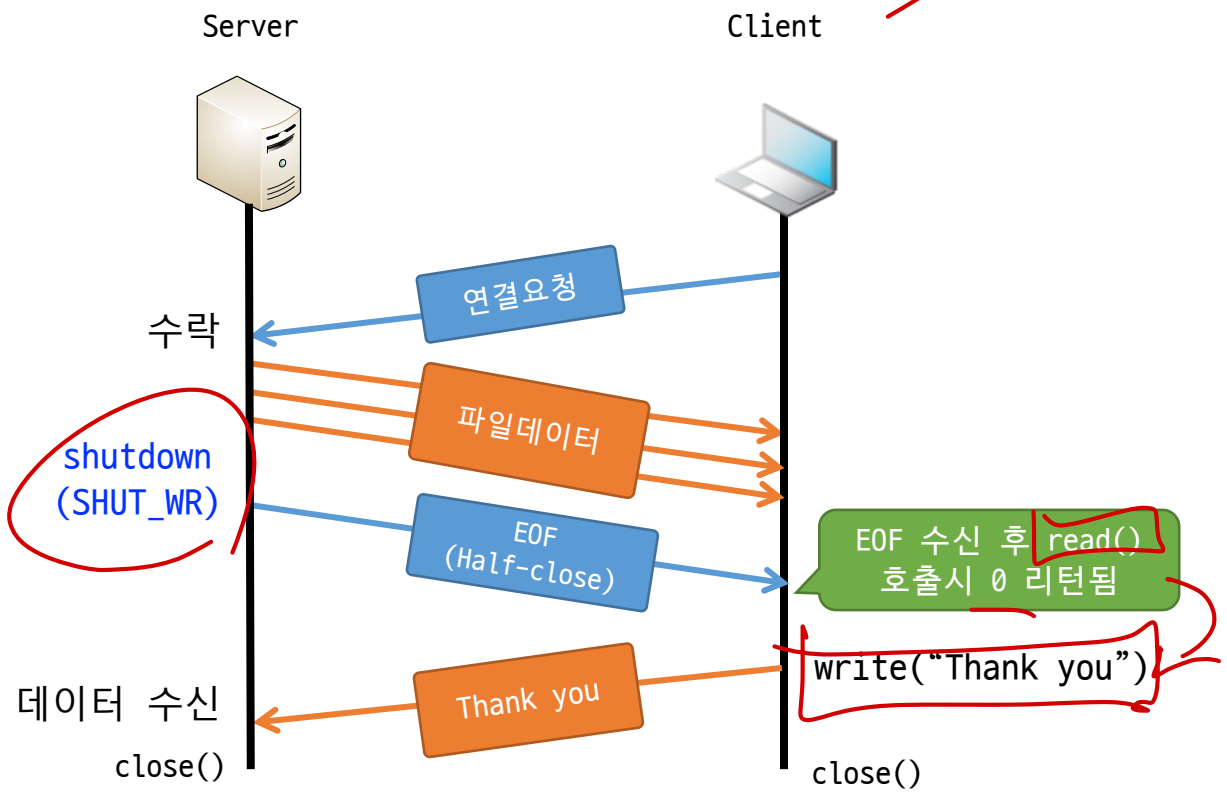


우리 파제랑 비슷하구만!!

출력 스트림 종료

- 출력 스트림을 종료하면
  - 연결되어 있는 호스트로 EOF 메시지 전달
- EOF 전송
  - 데이터 전송의 끝을 알림
  - 상대 호스트의 데이터 수신 함수(read(), recv())는 0을 리턴

shutdown(sock, SHUT_WR);



file_server.c #1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30
void error_handling(char *message);

int main(int argc, char *argv[])
{
    int serv_sd, clnt_sd;
    FILE * fp;
    char buf[BUF_SIZE];
    int read_cnt;
    struct sockaddr_in serv_adr, clnt_adr;
    socklen_t clnt_adr_sz;

    if(argc!=2) {
        printf("Usage: %s <port>\n", argv[0]);
        exit(1);
    }

    fp=fopen("file_server.c", "rb");
    serv_sd=socket(PF_INET, SOCK_STREAM, 0);
```

file_server.c 를 읽기모드로 open

file_server.c #2

```
    memset(&serv_adr, 0, sizeof(serv_adr));
    serv_adr.sin_family=AF_INET;
    serv_adr.sin_addr.s_addr=htonl(INADDR_ANY);
    serv_adr.sin_port=htons(atoi(argv[1]));

    bind(serv_sd, (struct sockaddr*)&serv_adr, sizeof(serv_adr));
    listen(serv_sd, 5);
    clnt_adr_sz=sizeof(clnt_adr);
    clnt_sd=accept(serv_sd, (struct sockaddr*)&clnt_adr, &clnt_adr_sz);
    while(1)
    {
        read_cnt = fread((void*)buf, 1, BUF_SIZE, fp);
        if(read_cnt < BUF_SIZE)
        {
            write(clnt_sd, buf, read_cnt);
            break;
        }
        write(clnt_sd, buf, BUF_SIZE);
    }
    shutdown(clnt_sd, SHUT_WR);
    read(clnt_sd, buf, BUF_SIZE);

    printf("Message from client: %s \n", buf);
    fclose(fp);
    close(clnt_sd); close(serv_sd);
    return 0;
}
```

두 소켓으로 전송

파일을 모두 읽어서
클라이언트로 전송한 경우,
shutdown()함수 호출

```
void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

file_client.c #1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30
void error_handling(char *message);

int main(int argc, char *argv[])
{
    int sd;
    FILE *fp;
    char buf[BUF_SIZE];
    int read_cnt;
    struct sockaddr_in serv_adr;

    if(argc!=3) {
        printf("Usage: %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    fp=fopen("receive.dat", "wb");
    sd=socket(PF_INET, SOCK_STREAM, 0);
```

→ 이게 어떤 함수이고 어) 이름가?

receive.dat 를 쓰기 모드로 open

file_client.c #2

```
    memset(&serv_adr, 0, sizeof(serv_adr));
    serv_adr.sin_family=AF_INET;
    serv_adr.sin_addr.s_addr=inet_addr(argv[1]);
    serv_adr.sin_port=htons(atoi(argv[2]));

    connect(sd, (struct sockaddr*)&serv_adr, sizeof(serv_adr));

    while((read_cnt=read(sd, buf, BUF_SIZE ))!=0)
        fwrite((void*)buf, 1, read_cnt, fp);

    puts("Received file data");
    write(sd, "Thank you", 10);

    fclose(fp);
    close(sd);
    return 0;
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

서버가 EOF를 전송하기 전까지 반복

shutdown

fread(), fwrite() 함수

- fread()
  - 파일에서 (nitems x size) 크기만큼 읽어서 메모리(ptr)에 저장

```
size_t fread(void *ptr, size_t size, size_t nitems, FILE *stream);
```

    - void *ptr: 파일 내용을 읽어들일 메모리 주소
    - size_t size: 데이터 하나의 크기
    - size_t nitems: 읽어올 데이터 개수
    - FILE *stream: 대상 파일 스트림
    - 리턴값: 성공 시 실제 읽은 데이터 개수, 실패 시 -1 리턴

- fwrite() 함수
  - 파일에 버퍼(ptr)의 내용을 저장

```
size_t fwrite(const void *ptr, size_t size, size_t nitems, FILE *stream);
```

    - ptr: 파일에 저장할 메모리 포인터
    - size_t size: 데이터 하나의 크기
    - size_t nitems: 저장할 데이터의 개수
    - FILE *stream: 대상 파일 스트림
    - 리턴값: 성공 시 실제 저장한 데이터 개수, 실패 시 -1 리턴

fread(), fwrite() 함수 예제

```c
#include <stdio.h>

#define MAX_BUF_SIZE 128

int main() {
    FILE* fout;
    FILE* fin;
    size_t ret;
    char rbuf[MAX_BUF_SIZE] = {0,};
    char wbuf[] = "Hello fread, fwrite function test";
    // fwrite()
    fout = fopen("test.txt", "w");
    if (fout != NULL) {
        ret = fwrite(wbuf, sizeof(char), sizeof(wbuf), fout);
        printf("fwrite ret= %d bytes\n", (int)ret);
        fclose(fout);
    }
    // fread()
    fin = fopen("test.txt", "r");
    if (fin != NULL) {
        ret = fread(rbuf, sizeof(char), MAX_BUF_SIZE - 1, fin);
        printf("fread ret = %d bytes\n", (int)ret);
        printf("%s\n", rbuf);
        fclose(fin);
    }
    return 0;
}
```

1x sizeof(wbuf)

```
fwrite ret= 34 bytes
fread ret = 34 bytes
Hello fread, fwrite function test
```

Questions?

LMS Q&A 게시판에 올려주세요.