Socket Programming

컴퓨터네트워크(12510, 조인휘 교수님) 기말 과제

컴퓨터소프트웨어학부 2017029425 김선웅

개발환경 및 개요

개발환경

ubuntu 18.04.1 LTS

vim version 8.0.1453

개요

client-server model에서 TCP/IP protocol을 이용한 소켓 프로그래밍.

server에서 client로 파일(*.txt*)을 전송할 수 있도록 구현하였다.
편의상 server에서 client로 `sending.txt` 파일을 전송하고, client에서 해당 파일을 `received.txt` 로 저장한다.
파일 전송이 완료될 경우 server는 client로 부터 ***Thank you*** 라는 메세지를 받고 종료가 되고, client는 그냥 종료가 된다.

client의 ip adress는 *127.0.0.1*이고, port number는 50000으로 설정하였다.

[Server/Client 공통] Error handling

server.c와 client.c에서 모두 error handle을 위하여 `error_handling()` 함수를 구현하였다.
만약 에러가 발생할 경우 `error_handling()` 의 인자로 error message를 넣고 호출하여, 해당 message와 개행문자(\n)를 **STDERR**로 출력을 보내고 프로그램을 종료하게 구현하였다.

Server

실행

server에서 client로 파일을 전송하기 위해서 server을 실행할 때 client의 port number를 main의 인자로 보내야한다. 때문에 실행 format인 `./server [client_port_number]` 을 준수하지 않았을 경우,

```
if(argc != 2)
  error_handling("Format : server [port]");
```

을 실행시켜 프로그램을 종료한다.

개요

server에서 client로 파일을 전송하기 위해서는 크게 아래의 함수 호출의 과정을 거친다.

순서: *socket -> bind -> listen -> accept -> send -> close*

socket

TCP protocol에서 사용하는 socket을 생성하기 위해서 3번째 argument로 6을 설정하였다. socket은 close하더라도 커널은 socket을 바로 kill하지 않고 일정 시간동안 alive한 상태(TIME_WAIT)로 유지한다. 이는 client로 아직 전송하지 않은 파일을 처리할 수 있도록 하기 위함이다. 이 때문에 프로그램을 종료한 뒤 일정시간 동안은 프로그램을 다시 실행시킬 경우 bind error가 발생한다.

이를 해결하기 위해서 `server_socket` 을 생성할 때 option을 부여해야한다. socket에 `SO_REUSEADDR` 이라는 option을 부여할 경우, 같은 port에 대해 다른 socket이 bind되는 것을 가능하게 해준다. 따라서 아래와 같은 코드를 추가하였다.

```
option = 1;
setsockopt( server_socket, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option) );
```

`option = 1` 을 통해서 `SO_REUSEADDR` 의 option을 1(TRUE)하게 설정하고, `setsockopt()` 을 통해서 socket의 option을 설정해주었다.

bind, listen, accept

함수 호출 형식에 맞춰서 호출을 하고, 호출한 결과가 error을 나타낼 경우 `error_handling()` 을 통해서 error handle을 하였다.

bind, listen, accept의 선언부를 살펴보면 모두 *sys/socket.h*에 `extern` 으로 선언이 되어있는데, 각 함수에 관해 설명이 되어있는 주석을 살펴보면, 함수 호출 결과가 success인 경우 0을, error가 발생할 경우 -1을 return한다고 되어있다. 따라서 각 함수의 return value가 -1일 경우 error로 간주하고 `error_handling()` 을 수행하도록 하였다.

send

이번 socket program에서 server에서 client로 파일 전송이 이루어지기 때문에 `server.c` 에서는 buffer size(BUFSIZE, 2048)만큼 file에서 buffer로 읽고 client socket에 write하는 과정을 반복하도록 하였다. `read()` 의 경우 읽은 file의 size만큼을 return value로 돌려준다. 따라서 return value가 0일 경우에는 파일에 읽을 것이 없다는 의미이므로

```
while((len = read(file, buffer, BUFSIZE)) != 0)
  write(client_socket, buffer, len);
```

을 통해 파일에 읽을 것이 있는 동안 read와 write를 반복하도록 하였다.

client

실행

server와 연결하기 위해서, client를 실행할 때는 server와 matching이 되는 ip address와 port number을 main의 인자로 넣어야 한다. 따라서
`./client [ip_address] [port_number]` 의 형식으로 실행하지 않으면 error을 발생시켰다.

```
if(argc != 3)
  error_handling("Format : client [IP address] [port number]");
```

개요

server에서 전송한 파일을 받기 위해서 아래와 같은 함수 호출 순서를 거쳐야한다.

순서: *socket -> connect -> recv -> close*

파일 열기

server에서 받은 파일을 disk에 저장하기 위해서 file을 write mode로 open해야한다. 그런데 write 전용 모드로 열 경우, 같은 이름의 파일이 disk에 있을 경우 error을 발생시키기 때문에 아래와 같은 과정을 거쳤다.

> 1. 파일을 write 전용 모드로 열고, 기존에 같은 이름의 파일이 있을 경우 아래의 과정을 거친다.
> 2. 기존에 있던 파일을 삭제하고 wirte 전용 모드로 새로 파일을 open한다.
> 3. 만약 새 파일이 open되지 않을 경우 error을 발생시킨다.

또한 파일을 open할 때 모든 권한을 모든 user에게 부여하기 위하여 `open()` 의 세번째 인자로 **0777**을 주었다.

```
if((file = open("received.txt", O_WRONLY | O_CREAT | O_EXCL, 0777)) == -1) {
  remove("received.txt");
  file = open("received.txt", O_WRONLY | O_CREAT, 0777);

  if(file == -1)
      error_handling("RECIEVE FILE OPEN ERROR");
}
```

socket

client에서는 server에서 설정한 socket의 종류에 알맞게 socket을 생성할 수 있도록 socket의 세번째 인자로 0을 주었다.

```
client_socket = socket(PF_INET, SOCK_STREAM, 0);
```

connect

`connect()` 가 `extern` 으로 선언되어있는 `sys/socket.h` 파일을 살펴보면 `connect()` 에 대한 설명으로, 함수 호출이 성공적으로 마무리 되었으면 0을, error가 발생했을 경우에는 -1을 return한다고 되었다. 따라서 `connect()` 의 return value가 -1일 경우 `error_handling()` 을 호출해서 error handle을 하였다.

read

`recv()` 의 경우 client socket으로 받은 내용을 buffer에 저장을 한다. 이 때 buffer에 저장한 크기, 즉 client socket으로부터 읽은 파일의 크기만큼을 return하게 되고, 만약 읽은 파일이

빈 파일일 경우에는 0을 return한다. 따라서 읽은 파일이 빈파일이 아닌 동안 계속해서 client socket으로부터 읽은 내용을 buffer에 저장하고, buffer에 저장된 내용을 다시 file에 wirte하도록 하였다.

```
while((len = recv(client_socket, buffer, BUFSIZE, 0)) != 0)
  write(file, buffer, len);
```

파일 수신이 완료된 경우

파일 수신이 완료되었을 때, server로 **Thank you** message를 보내어 띄우도록 하였다.

실행 결과