

Computer Network #02

Programming Assignment #01

21800409 신지영

21800510 이민규

Network Environment

```
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
en3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> ntu 1500
inet 10.1.0.1 netmask 255.255.255.0 broadcast 10.1.0.255
ether b0:26:28:4f:ea:a8 txqueuelen 1000 (Ethernet)
RX packets 505888 bytes 426770471 (407.0 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 125206 bytes 43430879 (41.4 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 16

en4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> ntu 1500
inet 203.252.112.25 netmask 255.255.255.192 broadcast 203.252.112.63
ether b0:26:28:4f:ea:a9 txqueuelen 1000 (Ethernet)
RX packets 90587870 bytes 14886963174 (13.8 GiB)
RX errors 1 dropped 44 overruns 0 frame 1
TX packets 15507231 bytes 11425411046 (10.6 GiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 17

lo: flags=73<UP,LOOPBACK,RUNNING> ntu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 33

en3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> ntu 1500
inet 10.1.0.2 netmask 255.255.255.0 broadcast 10.1.0.255
ether 54:9f:35:0d:52:a2 txqueuelen 1000 (Ethernet)
RX packets 94457 bytes 27130898 (25.8 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 2694253 bytes 693069266 (660.9 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 47

en4: flags=4099<UP,BROADCAST,MULTICAST> ntu 1500
ether 54:9f:35:0d:52:a3 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 49


lo: flags=73<UP,LOOPBACK,RUNNING> ntu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
```

먼저 ifconfig를 통해 10.1.0.1, 10.1.0.2인지 확인을 하였습니다.

Test Data

File Name	text.txt
File Contents	Hello world! I am Olaf! I like warm hugs!
Wc Result	2 9 41

File Name	img.jpg
File Contents	
Wc Result	57 109 12018

File Name	f.png (UDP에서 사용, 사용이유는 UDP 전송시간이 너무 오래걸림)
File Contents	
Wc Result	0 3 20

TCP program

1. Result

```
[s21800409@localhost TCP]$ ./TCPserver.out 1234
filename: ../data/text.txt
받은 데이터:Hello world!
I am Olaf!
I like
받기 성공
받은 데이터: warm hugs!
받기 성공
전송 완료
filename:../data/text.txt
===WC result===
 2  9 41 ../data/text.txt
```

〈figure1 transfer text.txt by TCPserver〉

```
[s21800409@localhost TCP]$ ./TCPclient.out 10.1.0.1 1234 ../data/text.txt
filename: ../data/text.txt
보내는 데이터:Hello world!
I am Olaf!
I like
보내기 성공
보내는 데이터: warm hugs!
보내기 성공
전송 완료
filename:../data/text.txt
===WC result===
 2  9 41 ../data/text.txt
```

〈figure2 transfer text.txt of TCPclient〉

```

받기 성공
받은 데이터: Ei+M
받기 성공
받은 데이터: ~y~+
받기 성공
받은 데이터: <ay 10
받기 성공
받은 데이터:
받기 성공
받은 데이터:
받기 성공
전송 완료
filename:../data/img.jpg
===WC result===
57 109 12018 ../data/img.jpg

```

<figure3 transfer img.jpg by TCPserver>

```

보내기 성공
보내는 데이터:
보내기 성공
보내는 데이터: 0
보내기 성공
보내는 데이터: {SV00E00E
보내기 성공
전송 완료
filename:../data/img.jpg

===WC result===
57 109 12018 ../data/img.jpg

```

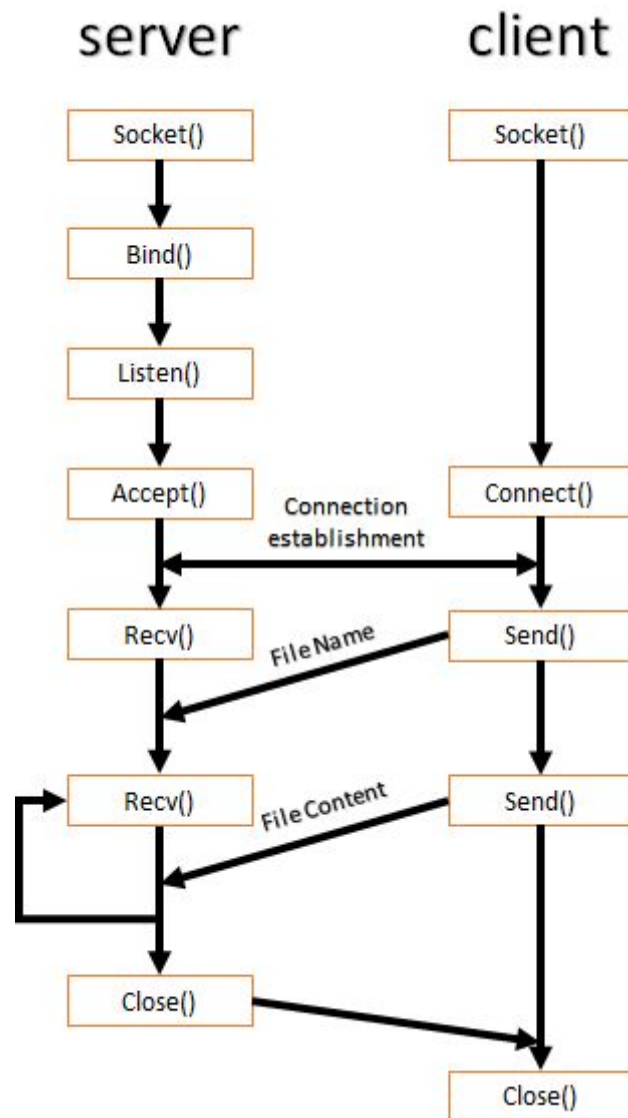
<figure4 transfer img.jpg by TCPclient>

2. Issue

- a. TCP에서는 Stream을 이용하다보니 File Name에 File Content의 일부가 포함되는 문제가 있었다. 이것을 지연없이 해결할 수 있는 방법은 무엇인가?
 - 구분자(Wn)를 이용하는 방법을 사용하였다.
 - Client: strcat을 이용하여 File Name에 Wn을 추가해서 전송한다.
 - Server: recv()에서 MSG_PEEK flag를 이용하여 Stream에서 제거하지 않고 File Name에 BUFSIZE만큼 미리 받아온 다음에 strtok를 이용하여 받아온 File Name의 Wn를 W0로 교체하여 File Name의 문자열을 자르고 문자열의 길이를 알아낸다. 문자열 길이만큼 recv()를 한 번 더 실행한다.
- b. 전송을 제때에 종료할 수 있는 방법은 무엇인가?
 - Client: FEOF를 이용하여 File의 End of File을 만났을 때 전송을 종료한다.
 - Server: Client에서 전송된 데이터가 없을 때, 즉 File Buf의 길이가 0이 될 경우 전송을 종료한다.

3. Code Explanation

a. Overview



b. TCPserver.c

```
// Receive the file name
recv(new_fd, filename, BUFSIZE, MSG_PEEK);
char * ptr = strtok(filename, "\n");
filename_len = strlen(ptr);

recv(new_fd, filename, filename_len+1, 0);
filename[filename_len]=0;
printf("filename: %s\n", filename);

// Save the file
file = fopen(filename, "wb");

//Store the chunked data to the file
while(1){
    filebuf_len = recv(new_fd, filebuf, BUFSIZE, 0);
    if(filebuf_len==0) break;
    filebuf[filebuf_len]=0;
    printf("받은 데이터:%s\n", filebuf);
    fwrite(filebuf, sizeof(char), filebuf_len, file);
    printf("받기 성공\n");
}
printf("전송 완료\n");
printf("filename:%s\n", filename);
fclose(file);
close(new_fd);
```

- fopen을 통해 파일을 열고 client쪽에서 fread를 통해 client 쪽에 저장된 바이너리 혹은 ASCII 파일을 불러서 읽어들이니다. 그리고 읽어들이 내용을 버퍼 크기에 맞게 send 함수를 통해 보냅니다.

c. TCPclient.c

```
// Send the file data
file = fopen(filename, "rb");

// Send the file name
strcat(filename, "\n");
send(sock, filename, strlen(filename), 0);

while(1){
    if(feof(file)) break;
    filebuf_len = fread(filebuf, sizeof(char), BUFSIZE, file);
    filebuf[filebuf_len]=0;
    printf("보내는 데이터:%s\n", filebuf);
    send(sock, filebuf, filebuf_len, 0);
    printf("보내기 성공\n");
}
printf("전송 완료\n");
printf("filename:%s\n", filename);
fclose(file);
```

```
close(sock);
```

- client 쪽에서 보낸 파일이름을 recv를 통해 받습니다. 이 파일이름을 기반으로 하여 fopen을 통해 파일을 열고 파일에 있는 값들을 recv를 이용하여 버퍼 크기만큼 계속 읽어들이입니다.

UDP program

1. Result

```
[s21800510@localhost UDP]$ ./UDPclient 10.1.0.1 2020 test.txt
filename: test.txt
보내는 데이터:hello world! byebye!
보내기 성공
전송 완료
===WC result===
0 3 20 test.txt
```

〈figure1 transfer text.txt by UDPclient〉

```
[s21800510@localhost UDP]$ ./UDPserver 2020
filename: test.txt
받은 데이터:hello world! byebye!
받기 성공
===WC result===
0 3 20 test.txt
```

〈figure2 transfer text.txt of UDPserver〉

```
[s21800510@localhost UDP]$ ./UDPserver 2020
filename: f.png
받은 데이터:PNG
받기 성공
받은 데이터:
받기 성공
받은 데이터:a
받기 성공
받은 데이터:T(Sc
받기 성공
===WC result===
2 3 22 f.png
[s21800510@localhost UDP]$
```

〈figure3 transfer f.png by UDPserver〉

```
[s21800510@localhost UDP]$ ./UDPclient 10.1.0.1 2020 f.png
filename: f.png
보내는 데이터:PNG
보내기 성공
보내는 데이터:
보내기 성공
보내는 데이터:a
보내기 성공
보내는 데이터:T(Sc
보내기 성공
보내는 데이터:
보내기 성공
전송 완료
===WC result===
3 8 145 f.png
[s21800510@localhost UDP]$
```

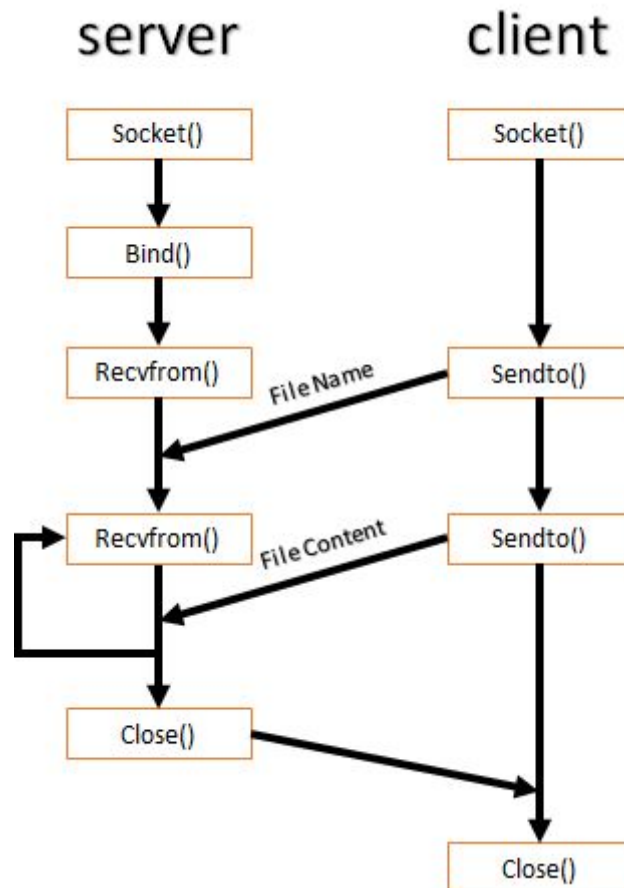
〈figure4 transfer f.png by UDPclient〉

2. Issue

- a. File Name에 File Content의 일부가 포함되는 문제가 있었다. 이것을 자연스럽게 해결할 수 있는 방법은 무엇인가?
 - Client에서 File Name 전송과 File Content 전송 사이에 sleep 함수를 배치하여 파일 이름과 파일 내용 사이에 시간차를 두어 패킷의 순서를 정렬하였습니다.
- b. 정확히 종료하려면 어떻게 해야하는가? 종료 Condition?
 - while문 안에 버퍼 크기가 0이하일 경우(에러코드) 종료를 하도록 조건문을 삽입했고 전송하는 client 쪽에서 전송 마지막에 에러코드를 전송하도록 했습니다.
- c. 패킷의 순서가 바뀌는 경우가 생기는데 어떻게 해결할 수 있는가?
 - 패킷의 순서가 다른 것에 대해 Error Recovery를 해야하는데 우리 프로그램에서는 따로 복구하지 않았습니다.

3. Code Explanation

- a. Overview



b. UDPserver.c

```
filename_len = recvfrom(serv_sock, filename, BUFSIZE, MSG_PEEK, (struct
sockaddr*)&clnt_addr, &clnt_addr_size);
recvfrom(serv_sock, filename, filename_len+1, 0, (struct sockaddr*)&clnt_addr,
&clnt_addr_size);
filename[filename_len]=0;
printf("filename: %s\n",filename);
// Save the file
file = fopen(filename, "wb");
// Store the chunked data to the file
while(1){
    clnt_addr_size = sizeof(clnt_addr);
    filebuf_len = recvfrom(serv_sock, filebuf, BUFSIZE, 0, (struct
sockaddr*)&clnt_addr, &clnt_addr_size);
    if(filebuf_len==0||filebuf_len<0) break;
    filebuf[filebuf_len]=0;
    printf("받은 데이터:%s\n",filebuf);
    fwrite(filebuf, sizeof(char), filebuf_len, file);
    printf("받기 성공\n");
}
fclose(file);
close(serv_sock);
```

- recvfrom을 통해 파일 이름을 가져오고 가져온 파일 이름을 통해 fopen으로 파일을 엽니다. 그리고 recvfrom을 통해 파일의 내용을 버퍼크기만큼씩 가져오고 만약 버퍼에 들어있는 파일의 크기가 0 혹은 그 이하의 수일 경우 while문을 빠져나옵니다. 버퍼에 들어있는 값의 마지막 자리에 0을 넣고 fwrite를 통해 버퍼 값을 파일 이름에 맞는 파일에 저장합니다. 그리고 fclose를 이용해 파일을 닫습니다.

c. UDPclient.c

```
sendto(sock, filename, strlen(filename),0,(struct
sockaddr*)&serv_addr,sizeof(serv_addr));

// Send the file data
file = fopen(filename, "rb");
while(1){
    if(feof(file)) break;
    sleep(3);
    filebuf_len = fread(filebuf,sizeof(char),BUFSIZE, file);
    printf("보내는 데이터:%s\n",filebuf);
    sendto(sock, filebuf, strlen(filebuf),0,(struct sockaddr*)&serv_addr,
sizeof(serv_addr));
    printf("보내기 성공\n");
}
sendto(sock, NULL, 0,0,(struct sockaddr*)&serv_addr, sizeof(serv_addr));
printf("전송 완료\n");
fclose(file);
```

- sendto를 통해 파일명을 먼저 보냅니다. 그리고 fopen으로 파일을 열고 feof로 파일이 끝났는지 체크를 합니다. 그리고 sleep을 통해 파일 이름과 파일내용의 전송 시간 차를 만듭니다. 그리고 sendto를 통해 파일 내용을 전송합니다. while문 끝에 NULL값을 보내는 sendto 코드를 통해 서버를 종료시킵니다. 그리고 fclose를 통해 파일을 닫습니다.