Homework #3

✓ Please upload your answer sheet in LMS. The uploading file must be PDF.

✓ You must show the screen capture for your answer.

✓ Also upload your source codes in the Linux servers as shown below.

✓ If you do not follow the file name rule and uploading policy, you will receive penalty points.

- **Make a new directory, hw3, in your home directory.**

Server Name	Problems	Source Codes
Server #1 (R640) IP: 203.252.112.25	1	hw3/tcp_ft_server.c
	2	hw3/udp_ft_server.c
	3	hw3/hostname.c
Server #2 (New Dell) IP: 203.252.112.26	1	hw3/tcp_ft_client.c
	2	hw3/udp_ft_client.c

✓ Due date: 11pm, 11/15(Mon)

1. Implement the TCP-based file transfer program as shown and answer the questions. (50points)

   1) Implement the TCP-based file transfer program as shown below. (30points)

   - If you do not follow the instructions below, than you will receive penalty points.

   - You should implement the both server and client program using C language. And upload your source codes in the server as shown in above table.
     Source code filename for server program: tcp_ft_server.c (Server #1)
     Source code filename for client program: tcp_ft_client.c (Server #2)

   - Client sends the file name and contents of the file to the server.

   - Client sends the file name and contents of the file to the server.

   - Server is iterative type.

   - Server and client program should support both ASCII-based file and binary file.

   - Do not use sleep function in the programs.

   - After uploading the file, the client program should display the elapsed time and throughput.

   - Your programs should follow the usage below.
     Server: # ./tcp_ft_server
     Client: # ./tcp_ft_client < port>

2) Use the small buffer size at 'send()', for example, 10. And then observe the data size arrived at the server side (you have to set the large buffer size at 'recv()'.) What happens? (Compare the data size of send() and the size of the received packets.) You should answer with your evidence (i.e., screenshot).   (10points)

- Result of arrived packet, arrived data size in server side: buffer size = 30

```
[s21600635@localhost hw3]$ ./tcp_server 9191
* wait to connect to client
* success to connect to client

filename: asdf.c
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3
0 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 25
77 packets arrived
completed file transmit!
_____
```

- Result of sent packet, sent data size in client side: buffer size = 10

```
[s21600635@localhost hw3]$ ./tcp_client 203.252.112.25 9191 asdf.c
filename: asdf.c
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 1
0 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 1
0 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 5
success to send file

231 packets arrived
total bytes: 2305 bytes
elapsed time: 0.000468 sec
throughput: 4925213.675214 byte/sec
```

The tcp/ip has no message boundary. Therefore, regardless of the buffer size defined server, the server can read data as much as the buffer size with no data loss.

3) What happens in case that the data size of 'send()' is larger than the buffer size of 'recv()'? (Compare the data size of send() and the size of the received packets.). You should answer with your evidence (i.e., screenshot) (10points)

- Result of arrived packet, arrived data size in server side: buffer size = 30

```
[s21600635@localhost hw3]$ ./tcp_server 9191
* wait to connect to client
* success to connect to client

filename: asdf.c
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3
0 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 25
77 packets arrived
completed file transmit!
```

- Result of sent packet, sent data size in client side: buffer size = 50

```
[s21600635@localhost hw3]$ ./tcp_client 203.252.112.25 9191 asdf.c
filename: asdf.c
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 5
success to send file

47 packets arrived
total bytes: 2305 bytes
elapsed time: 0.000158 sec
throughput: 14588607.594937 byte/sec
```

The tcp/ip has no message boundary. Therefore, regardless of the buffer size defined server, the server can read data as much as the buffer size with no data loss.

※ **Hint! Problem 1-2) and 1-3) are intended to help you understand the TCP which has no**
message boundaries.

2. **Implement UDP-based file transfer program. (40points)**

    1) Convert the programs in Problem 1 to the UDP-based file transfer programs. When you
        implement the programs, please follow the instructions described in Problem 1. (30points)

    -   You should implement the both server and client program using C language.

    -   Source code filename for server program: udp_ft_server.c (Server #1)
        Source code filename for client program: udp_ft_client.c (Server #2)

    2) What happens in case that the data size of 'sendto()' is larger than the buffer size of 'recvfrom()'.
        You should answer with your evidence (i.e., screenshot)? (10points)

        - Result of arrived packet, arrived data size in server side: buffer size = 30

```
[s21600635@localhost hw3]$ ./udp_server 9191
filename: asdf.c
start downloading

30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3
0 30 30 30 30 30 5
47 packets arrived
completed file transmit!
```

        - Result of sent packet, sent data size in client side: buffer size = 50

```
[s21600635@localhost hw3]$ ./udp_client 203.252.112.25 9191 asdf.c
len of filename 6
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 5
success to send file

47 packets arrived
total bytes: 2305 bytes
elapsed time: 0.000316 sec
throughput: 7294303.797468 byte/sec
```

        The client sends a packet to server with 50 buffer sizes. However, the udp has a message
        boundary. Therefore, when server receive the packet came from client, it can read only buffer size
        which defined from server. As a result, server can read only as amount of buffer size defined on
        server and the rest of data is discarded like figure in below.

```c
C asdf.c
 1   #include <stdio.h>
 2   #include <string.h>
 3   #include <unistd.h>
 4   #>
 5   #include <sys/socket.h>
 6   #incdefine BUF_SIZE 30
 7
 8   int
 9   main gv[] ) {
10      if ( argc != 4 ) age: %s <IP> <PORT> <FILENAME>t(1);
11      }
12
13      // open sendi ;
14      if ( ( fp = fopen( argvL ) {
15         printf( "fail to ) ;
16         exit( 1 ) ;
17      }    int sd ;
18      if ( ( sd = s_STREAM, 0 ) ) == -1 ) {
19       generate socket\n" ) ;
20           // set info for tcp trans
```

3. Implement the program using gethostbyname() as shown below. (10points)

- You should implement the program using gethostbyname().
  Source code filename: hostname.c (Server #1)
- Your program should display all of aliases, address type, and IP addresses.
- Your programs should follow the usage below.
  # ./hostname www.naver.com
  Official name: e60630.a.akamiedge.net
  Aliases 1: www.naver.com
  Aliases 2: www.naver.com.nheos.com
  Aliases 3: www.naver.com.edgekey.net
  Address type: AF_INET
  IP addr 1: 23.46.23.18

```
[s21600635@localhost hw3]$ ./hostname www.naver.com
Official name: www.naver.com.nheos.com
Aliases 1: www.naver.com
Address type: AF_INET
IP addr 1: 223.130.195.95
IP addr 2: 223.130.200.104
```