Computer Network

Project 2

2015313697
황영준

1. Development environments

<OS version>
Ubuntu 18.04.3 LTS

<Programming language and Compiler version>
C++ / g++ 7.4.0

<Compile option: -pthread>

```
hwang@hwang-VirtualBox:~/3-2/network/proj2$ cat /etc/issue
Ubuntu 18.04.3 LTS \n \l

hwang@hwang-VirtualBox:~/3-2/network/proj2$ g++ --version
g++ (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

hwang@hwang-VirtualBox:~/3-2/network/proj2$ g++ server5.cpp -pthread
```

2. Program Design

<main func>

```
28  int main(){
29      struct sockaddr_in servaddr,clntaddr;
30      int serv_sock;
31      int clnt_sock;
32      int clnt_addr_size;
33      vector<thread> working_thread;
34      // make socket
35      if((serv_sock=socket(PF_INET,SOCK_STREAM,0))<0){
36          perror("socket fail");
37          exit(0);
38      }
39      memset(&servaddr,0,sizeof(servaddr));
40      servaddr.sin_family = AF_INET;
41      servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
42      servaddr.sin_port = htons(port);
43      //bind
44      if(bind(serv_sock,(struct sockaddr *)&servaddr,sizeof(servaddr))<0){
45          perror("bind fail");
46          exit(0);
47      }
48      while(1){
49          listen(serv_sock,5);
50          puts("wait for sig");
51          // accept
52          clnt_addr_size=sizeof(clntaddr);
53          if((clnt_sock=accept(serv_sock,(struct sockaddr *)&(clntaddr),(socklen_
54              puts("accept fail");
55          }
56          puts("making thread");
57          // give to thread
58          working_thread.push_back(thread(connection,clnt_sock));///////////
59      }
60  }
```

29~56: make socket, bind, listen, accept the clint

58: thread를 생성하고 앞으로 client의 요청은 thread에서 concurrent 하게 처리함으로써 여러 client의 요청을 동시에 처리

<connection func>

```
62  void* connection(int clnt_sock){
63      char req_buf[4096];
64      char file_name[100];
65      char file_type[100];
66
67      while(1){
68          // recv request
69          memset(req_buf,0x00,sizeof(req_buf));
70          int req_len=recv(clnt_sock,req_buf,sizeof(req_buf),0);
71          if(req_len==-1){
72              perror("recv fail");
73          }
74          /*puts("the request is====");
75          puts(req_buf);*/
76          //get file name
77          memset(file_name,0x00,sizeof(file_name));
78          char *file_name_pos=strchr(req_buf,'/');
79          int count=0;
80          for((file_name_pos)+=1;*file_name_pos!=32;(file_name_pos)++,count++){
81              file_name[count]=*file_name_pos;
82          }
83          //get file type
84          memset(file_type,0x00,sizeof(file_type));
85          char *file_type_t=strstr(req_buf,"Accept:");
86          count=0;
87          for((file_type_t)+=8;*file_type_t!=',';(file_type_t)++,count++){
88              file_type[count]=*file_type_t;
89          }
90          if(file_name[0]=='\0'){
91              login(clnt_sock);
92              continue;
93          }
94          // open file
95          FILE *fp=fopen(file_name,"rb");
96          if(fp==NULL){
97              //perror("there are no such file");
98              ostringstream temps;
99              temps << header_404 << "\r\n"
100                 << header_con_len << "0" << "\r\n\r\n";
101             string tempss=temps.str();
102             send(clnt_sock,tempss.c_str(),tempss.size(),0);
103             continue;
104         }
105         /*puts("file name is====");
106         puts(file_name);
107         puts("file_type is====");
108         puts(file_type);*/
109         //send response
110         send_response(fp,file_name,file_type,clnt_sock);
111         fclose(fp);
112     }
113 }
114
```

67~73: while문 내에서 client의 request를 받음

77~89: request message에서 file name과 file type을 구함

90~93: 만약 request file이 없이 ip만으로 연결을 요청한 경우 login func에서 login을 처리

94~103: 만약 해당 파일이 존재하지 않을 경우 404 error 메시지를 보냄
110: 해당 파일이 존재할 경우 파일 포인터와 파일 타입, 파일 이름을 send_response function으로 보냄

<send_response>

```
115 void send_response(FILE* fp, char* file_name, char* file_type, int clnt_sock){
116     int len;
117     char buffer[10000];
118     char packet[512];
119     int temp=0;
120     // get file len
121     fseek(fp,0,SEEK_END);
122     len=ftell(fp);
123     fseek(fp,0,SEEK_SET);
124     // make file header
125     /*printf("length====%d\n",len);*/
126     memset(buffer,0x00,sizeof(buffer));
127     int tt;
128     while((tt=fread(buffer,sizeof(char),sizeof(buffer),fp))>0){
129         std::ostringstream body;
130         if(strstr(file_name,"html")==NULL){
131         body << header_200 << "\r\n"
132             << header_con_len << tt << "\r\n"
133             << header_con_con << "\r\n"
134             << header_con_typ << "application/octet-stream" << "\r\n"
135             << "\r\n";
136         }
137         else{
138         body << header_200 << "\r\n"
139             << header_con_len << tt << "\r\n"
140             << header_con_con << "\r\n"
141             << header_con_typ << file_type << "\r\n"
142             << "\r\n";
143         }
144         /*printf("\ntt==%d\n",tt);*/
145         string body1=body.str();
146         send(clnt_sock,body1.c_str(),body1.size(),0);
147         /*printf("header==%d\n",body1.size());
148         puts(body1.c_str());*/
149         send(clnt_sock,buffer,(size_t)tt,0);
150         /*for(int temp=0;temp<tt;temp++){
151             printf("%c",buffer[temp]);
152         }*/
153         memset(buffer,0x00,sizeof(buffer));
154     }
155 }
```

128~143: 해당 파일의 정보를 읽고 buffer에 data를 넣은 후 파일 타입, 읽어 들인 data의 길이에 대한 정보를 바탕으로 HTTP 헤더 생성
145~149: 헤더와 data body를 send로 client에게 전송

<login func>

```
157 void login(int clnt_sock){
158     char buffer[10000];
159     char req_buf[4096];
160     int len;
161     FILE *fp=fopen("index.html","rb");
162     memset(req_buf,0x00,sizeof(req_buf));
163     memset(buffer,0x00,sizeof(buffer));
164     len=fread(buffer,sizeof(char),sizeof(buffer),fp);
165
166     ostringstream body;
167     body << header_200 << "\r\n"
168         << header_con_len << len <<"\r\n"
169         <<header_con_con <<"\r\n"
170         << header_con_typ << "text/html" << "\r\n\r\n";
171     string body1=body.str();
172     send(clnt_sock,body1.c_str(),body1.size(),0);
173     send(clnt_sock,buffer,(size_t)len,0);
174     int req_len=recv(clnt_sock,req_buf,sizeof(req_buf),0);
175     if(req_len==-1){
176         perror("recv fail");
177     }
178     puts("the request is====");
179     puts(req_buf);
180     memset(req_buf,0x00,sizeof(req_buf));
181     req_len=recv(clnt_sock,req_buf,sizeof(req_buf),0);
182     if(req_len==-1){
183         perror("recv fail");
184     }
185     puts("the second request is=====");
186     puts(req_buf);
187 }
```

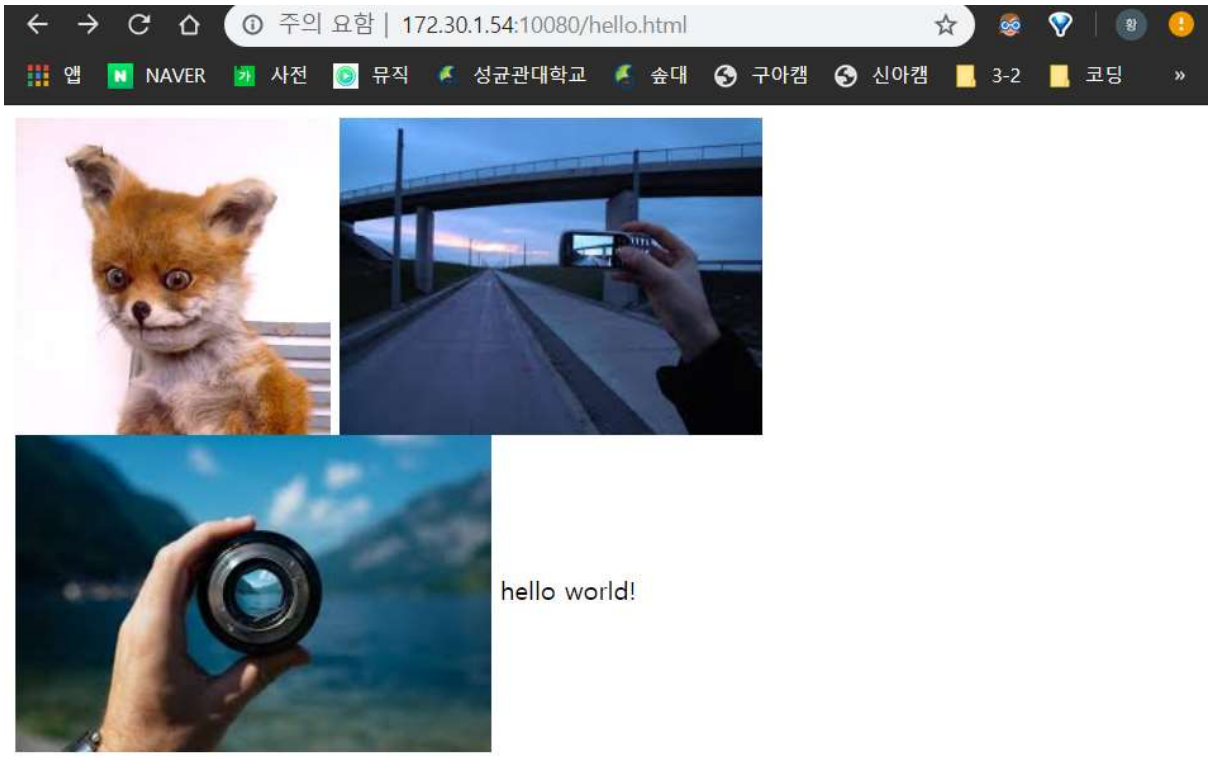
161~173: index.html을 client에게 전송

173~184: client의 index.html에 대한 응답을 받은 후 login session을 진행

3. test scenario

4.1

(1)



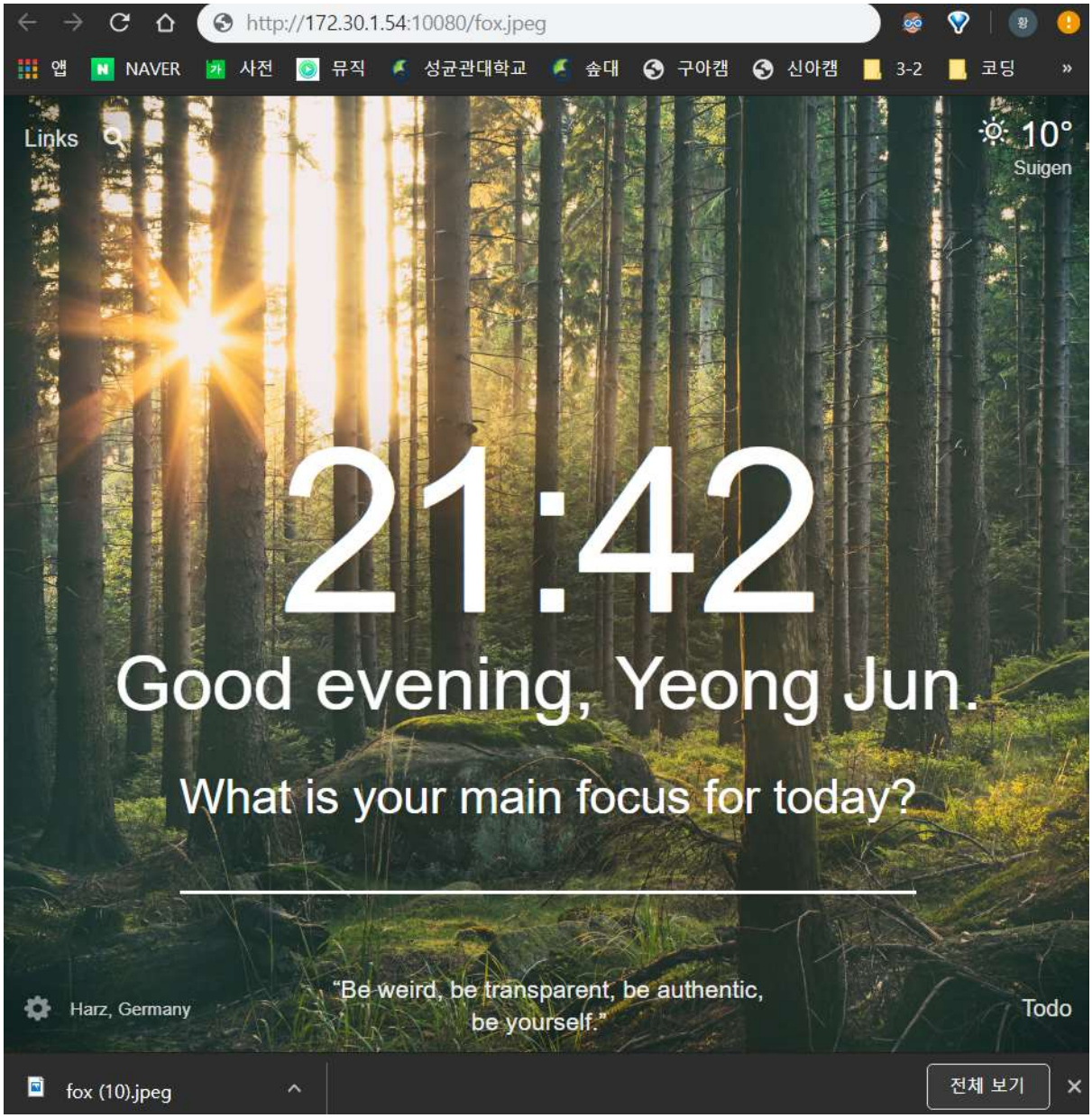
User request: hello.html

(2)

User request: asd.jpeg(존재하지 않는 파일)

➔ Response: 404

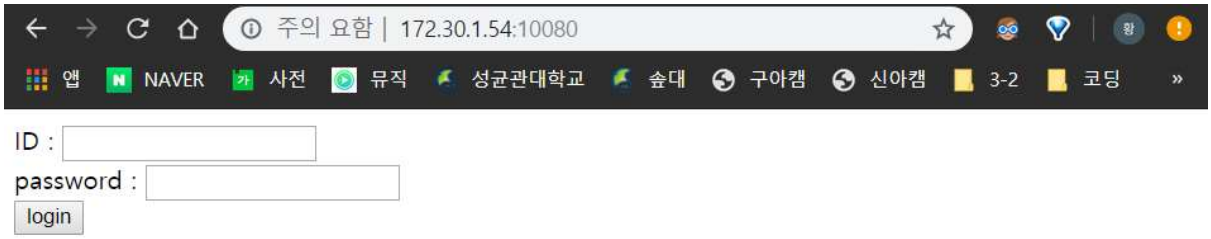
(3)



User request: fox.jpeg

Response: fox.jpeg file

4.2



User request: http://ip addr

Response: login session