**Introduction to Computer Networks <Assignment #2>**

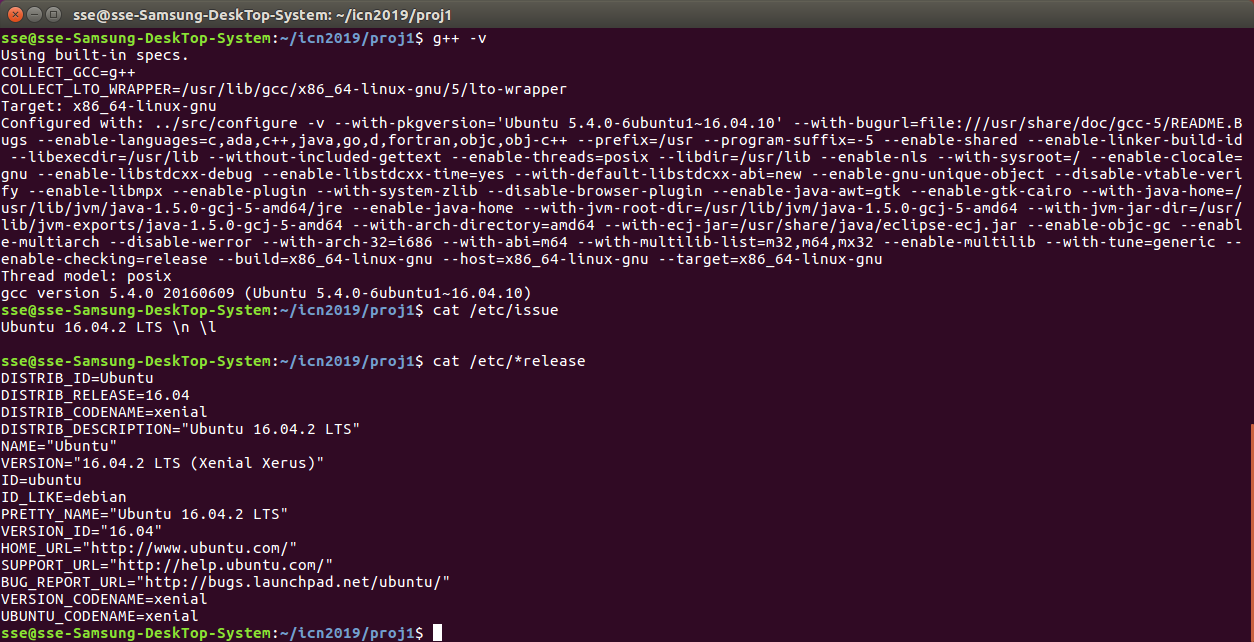
2015313276 김범석

**1. Development environments**

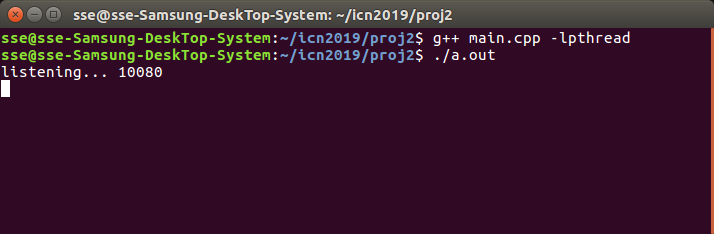
1) Version of OS : Ubuntu 16.04 LTS

2) Programming language : C++

3) Compilers version : GCC 5.4.0



4) Compile & Run



**2. Evaluation Scenarios**

1) Basic HTTP requests & responses – Work

2) Login functionality – Work

3) Persistent HTTP mode – Work

**3. About Code**

일단 여러 client가 동시에 접속을 해서 사용하기 위해서 pthread를 이용한 방법을 사용했습니다.

제일 먼저 코드가 실행이 되면 서버의 binding 작업을 통해 port 10080 소켓을 열고 계속 accept를 하기 위해 while(1)로 무한 루프를 돌면서 값을 받아 줍니다. accept를 이용하여 받은 client정보는 pthread\_create로 스레드를 만들어 변수로 넘겨주는 방식으로 요청을 처리하게 됩니다.

각 스레드에 client가 넘어가게 되면 제일 먼저 요청이 어떻게 들어왔는지에 대해 HTTP message를 분석하게 됩니다. 먼저 buffer에 message 전체를 받아 둡니다. 만약 값을 들어왔는데 첫 글자가 0인 경우(NULL Error)는 잘못해서 값이 들어온 것이므로 바로 close(client)를 하고 스레드를 끝내도록 합니다. 그리고 에러가 없는 경우 첫 줄에 적혀있는 request line을 읽어서 method, url, version을 각각 변수에 나누어 저장을 합니다. 그리고 header line을 읽으면서 login이 되었다는 정보가 cookie에 있을 것이므로 cookie가 왔는지 확인 한 후 cookie가 있다면 적힌 cookie\_number를 저장하도록 합니다. 그리고 login기능을 위해 header line을 다 읽고 나면 혹시 body에 id가 적혀있는지 확인 하고 있다면 해당하는 id로 cookie를 만들기 위해 cook\_flag를 set 해줍니다.

그리고 이제 GET으로 들어가게 되는데 만약 없는 위치의 url에 접근하려고 하면 404 NOT FOUND HTTP message를 보내줍니다. 그리고 받는 파일이 url로 따로 적히지 않았거나 index.html이라면 첫 로그인 화면이 나오도록 합니다. 그 이외에 로그인이 안되어서 cookie가 없는 경우는 403 Forbidden HTTP message를 보내줍니다. 이제 cookie가 있는 경우에는 url에 해당하는 파일을 open해서 HTTP message에 200 OK와 파일의 길이, 타입정보를 Header에 보내고 그 뒤로 body부분에 파일을 전송하게 됩니다. 특이한 케이스로 cookie.html을 두었는데 cookie.html은 따로 파일을 만들어두지 않아서 일단 처음에 확인할 때 예외로 두었고 cookie가 온 상황에서 cookie.html에 대한 요청이 들어오면 코드에서 직접 적은 html 코드를 보내주는데 필요한 것이 받아둔 id와 현재시간과 cookie가 처음 만들어진 시간 차이이므로 cookie와 id의 정보가 저장되어있는 map을 참고하여 저장된 시간과 현재시간의 차이를 계산하여 값을 적절히 넣어주었습니다.

다음으로 로그인 기능으로 로그인을 하게되면 index.html에 따라서 POST로 secret.html을 가져오게 됩니다. 이 때 body에 id가 있다는 것을 확인해준 cook\_flag가 set 되어 있어야 하고, 여기서 cookie를 만들어서 보내주게 되는데 보낼 때 Expires로 30초 후의 시간을 보내서 30초 뒤에는 cookie가 사라지게 합니다. Cookie ID는 cookie.html의 편리를 위해 expired되는 시간과 쿠키번호(global variable이므로 lock을 걸어서 사용)를 같이 붙여서 보내주었습니다. 이렇게 header line을 다 적은 후에는 secret.html 파일을 열어서 body부분에 붙여서 보내주도록 합니다.

위의 과정이 send되는 과정이고 이 과정들은 다른 thread에서 겹칠 수 있으므로 lock을 걸어주었고 close되기 전 마지막으로 map에서 이미 expired된 값을 지워주기 위한 cleaning\_cookie\_table이라는 함수를 만들어서 사용하였습니다.

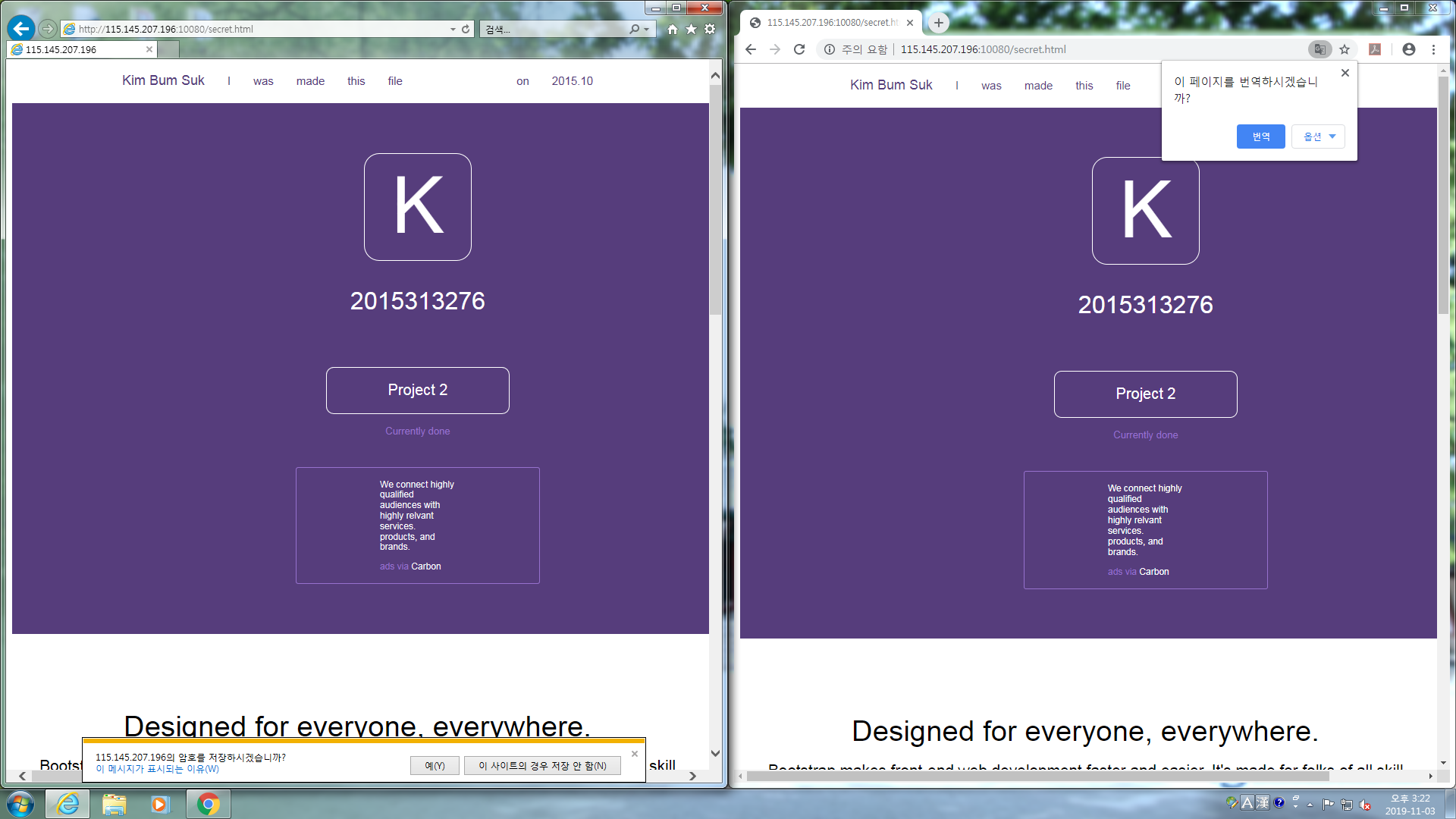
마지막으로 persistent mode를 지원하기 위해 위의 과정을 do-while문으로 묶어서 keep-alive가 왔을 때만 계속 실행하게 하고 main함수에서 10초가 지나면 자동으로 닫기도록 setsocketopt로 하였습니다. packet을 보낼때는 keep-alive가 request로 왔을 때 response도 keep-alive와 timeout을 10으로 설정하여 보냈습니다.

**4. Capture**

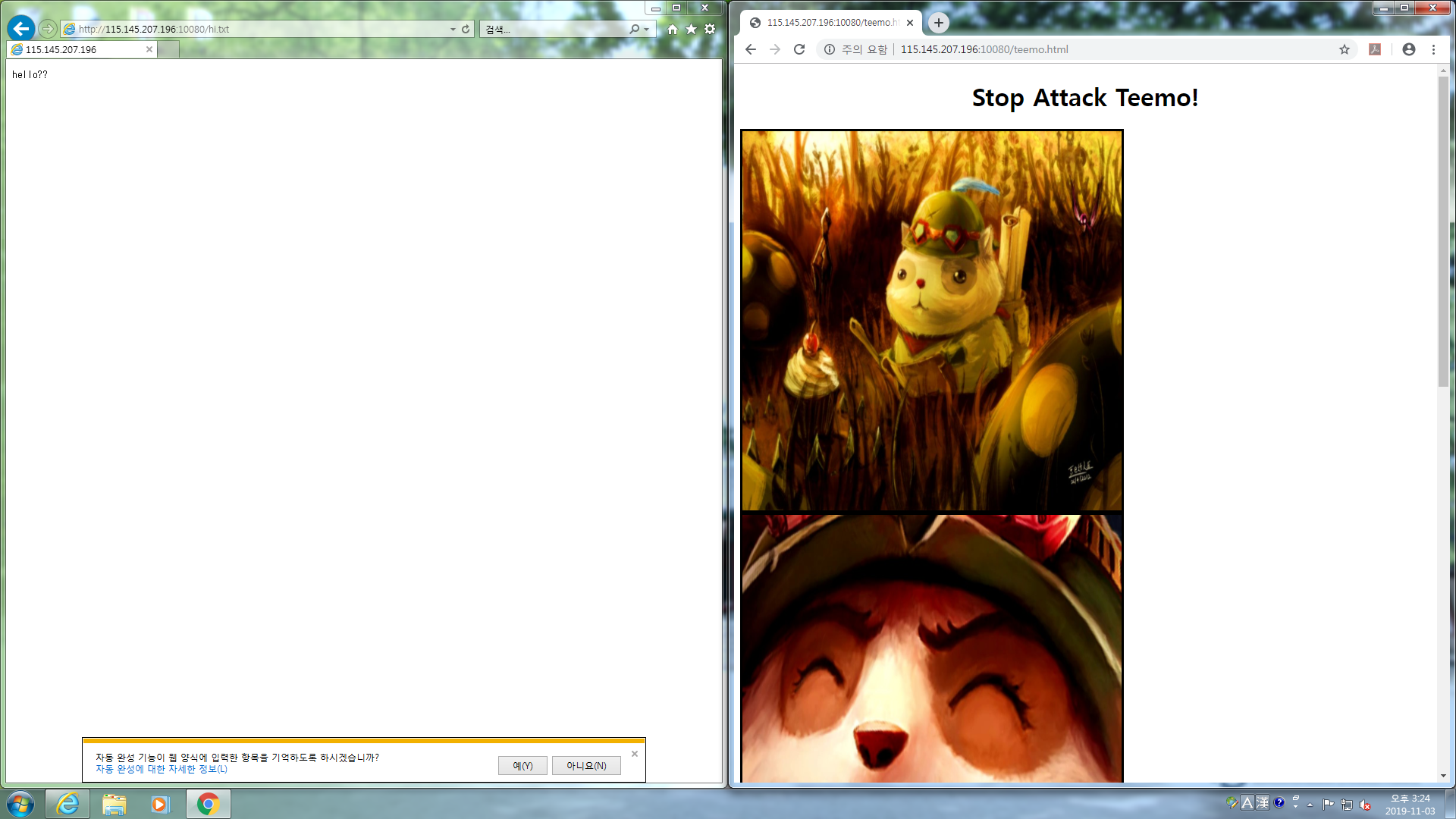
1)Basic HTTP requests & responses



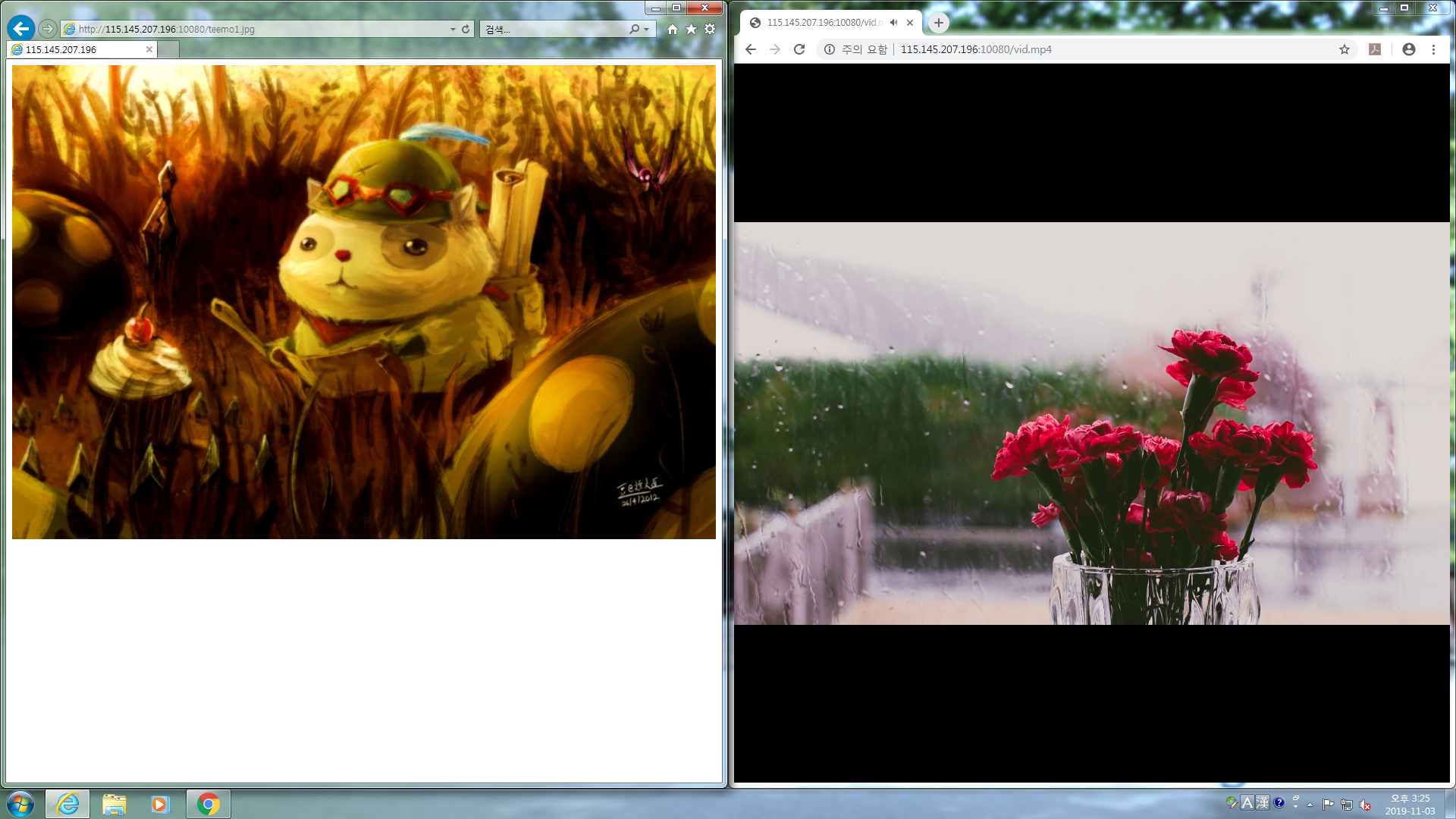
<Fig 1. First (Login)Page>



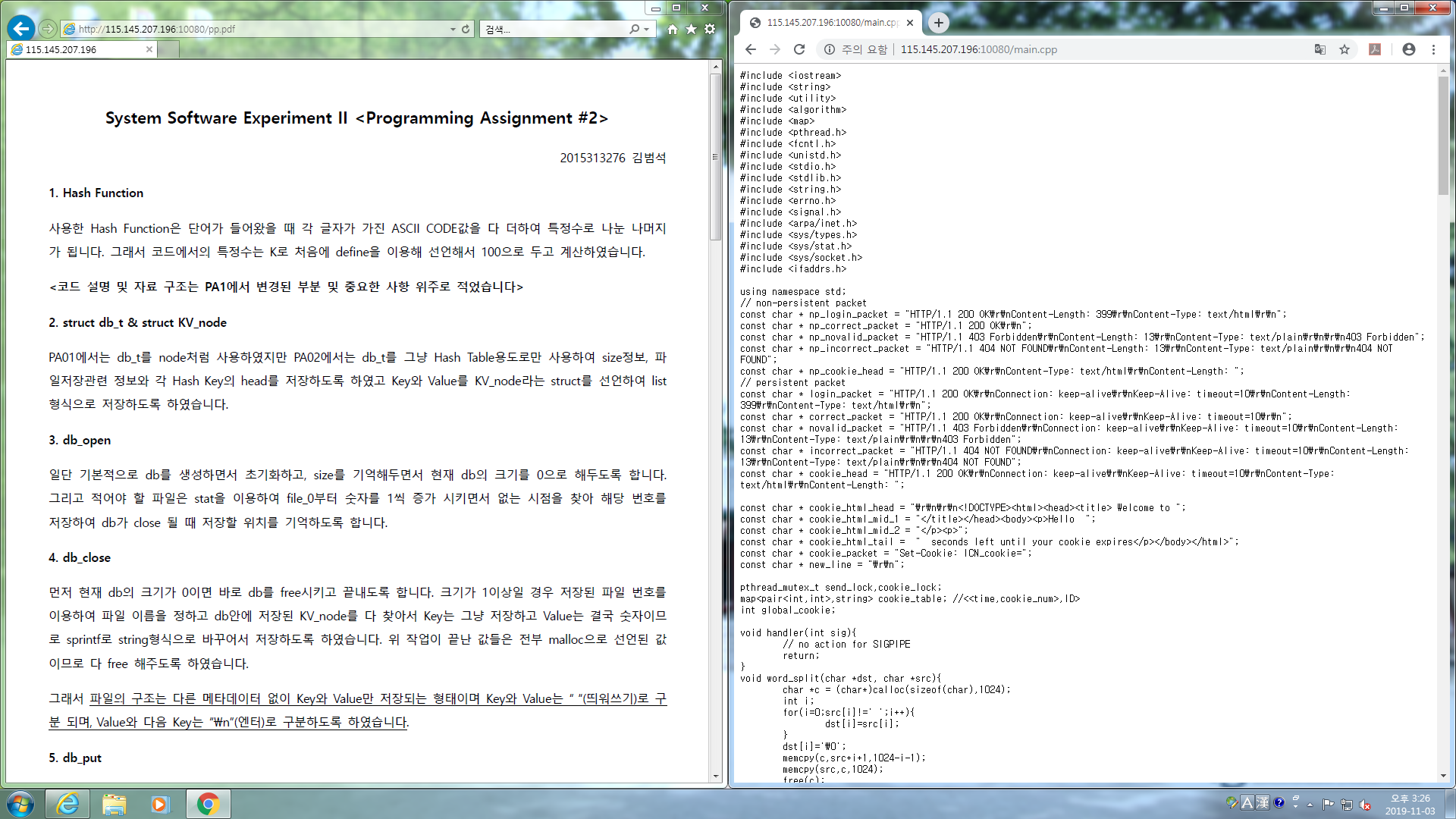
<Fig 2. HTML file>



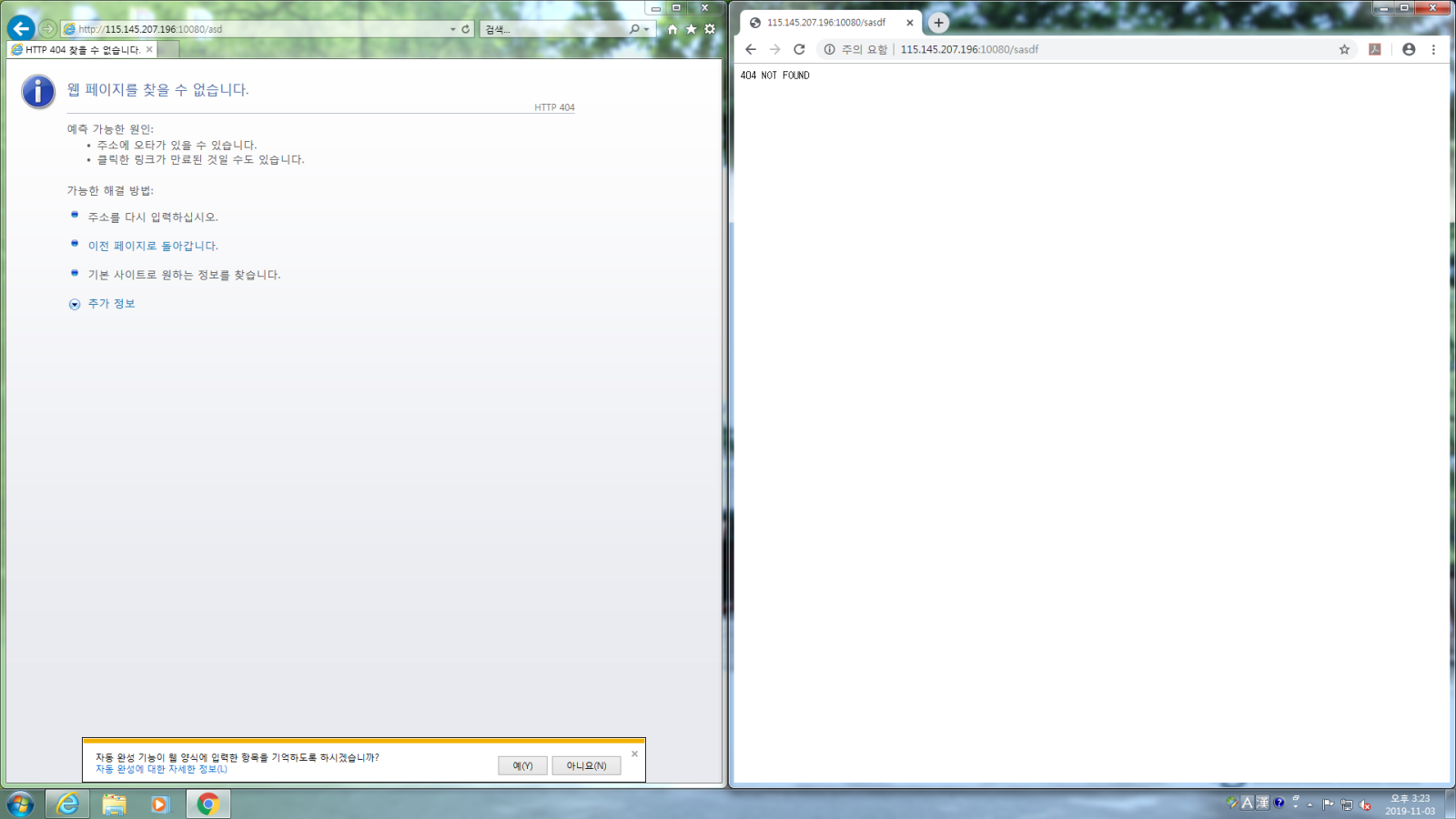
<Fig 3. Text file & HTML file with image>



<Fig 4. Image file & Video file>

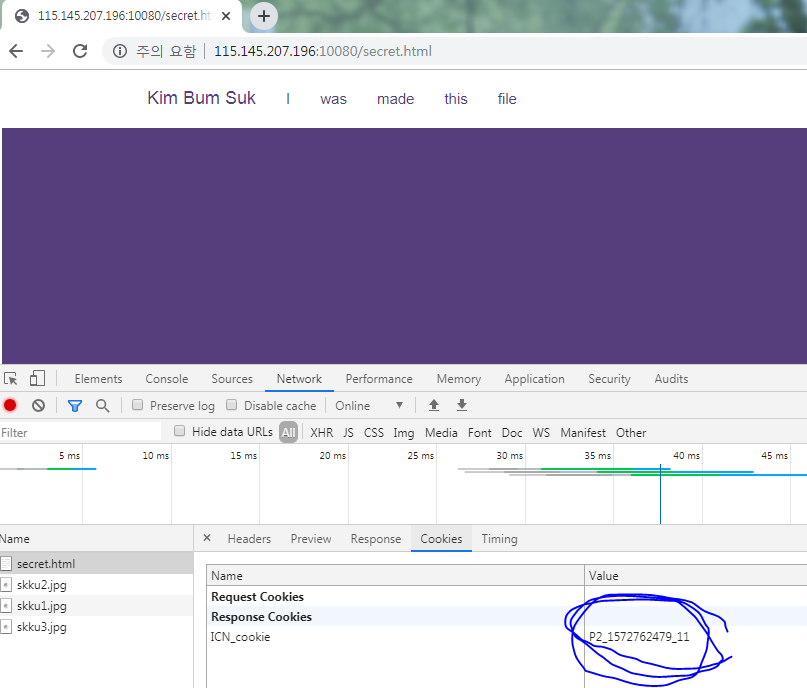


<Fig 5. PDF file & CPP code file>

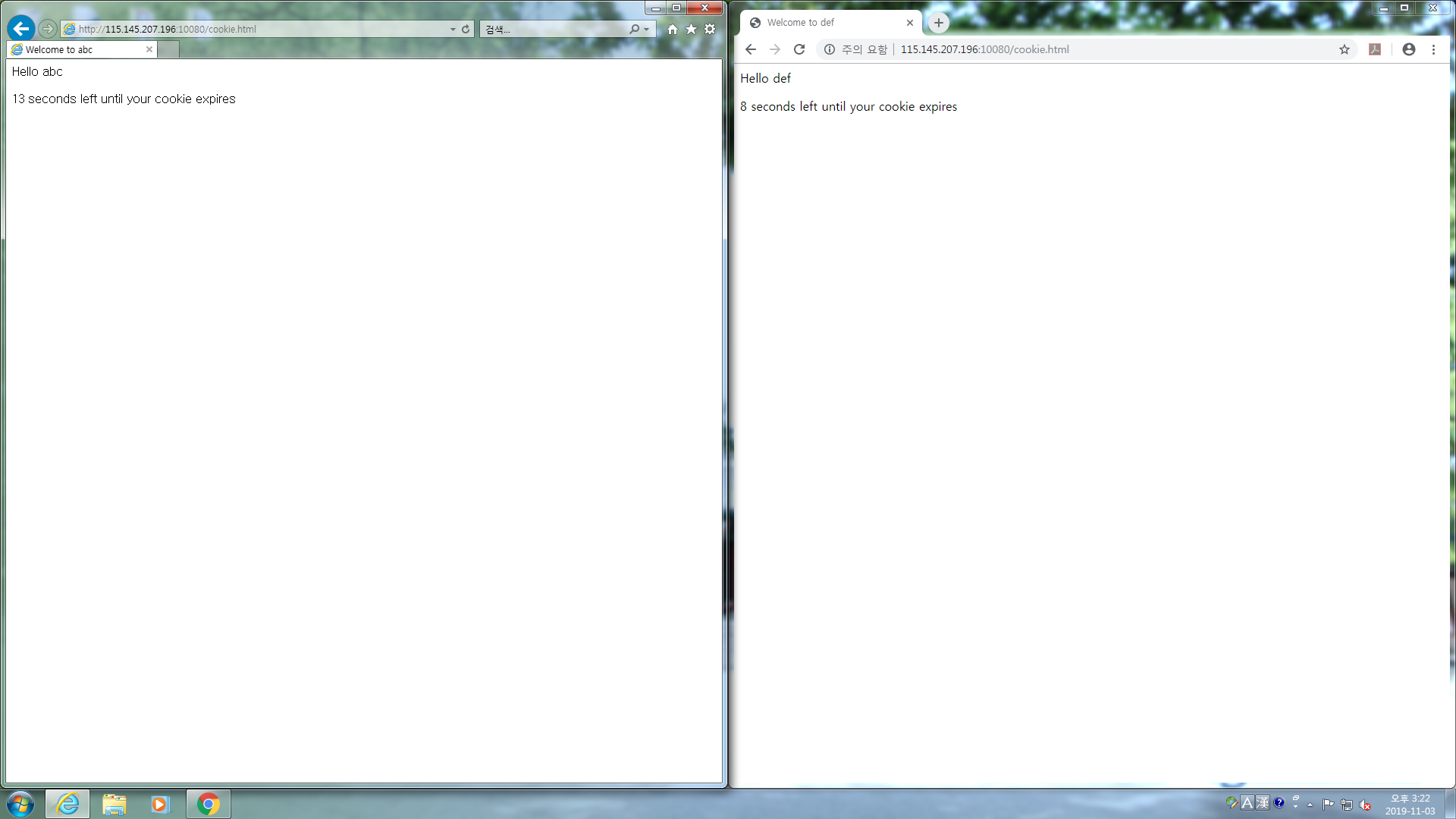


<Fig 6. No file – 404 NOT FOUND>

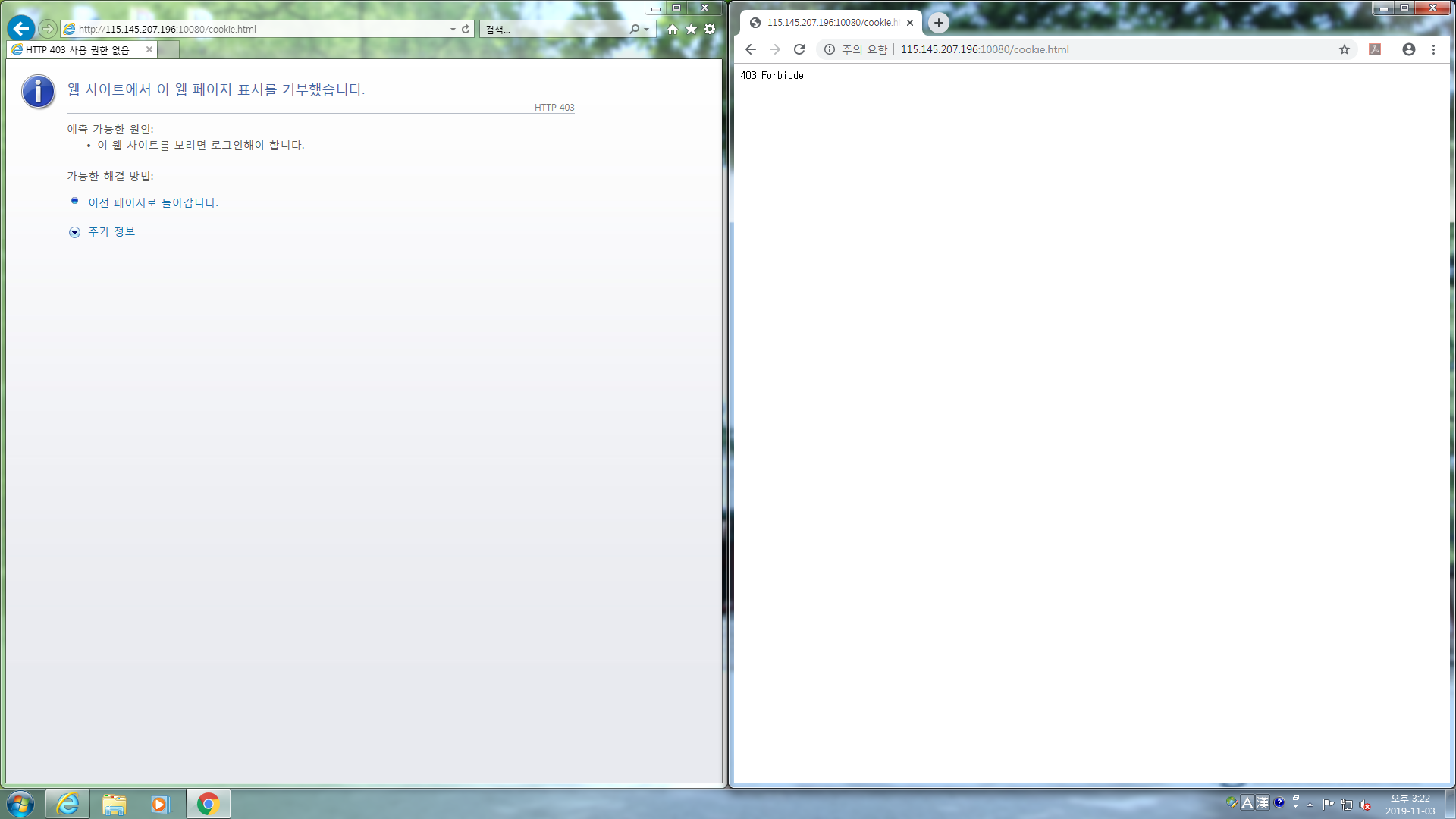
2)Login Function(Cookie)

****

<Fig 7. Cookie transmit>

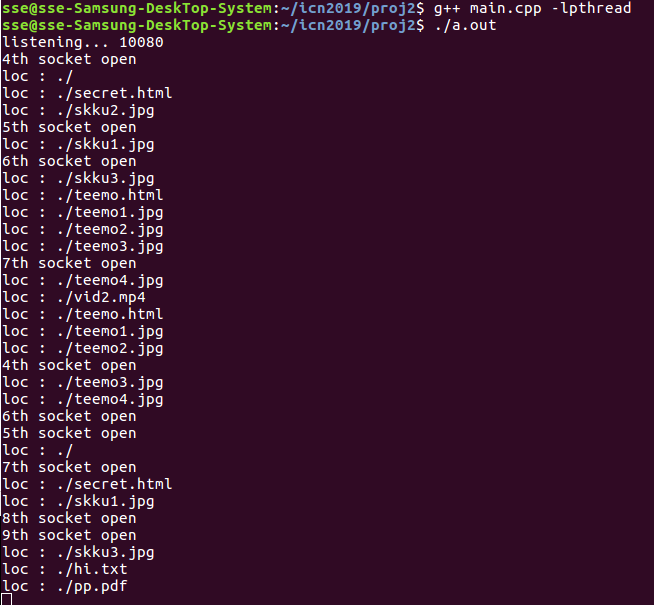


<Fig 8. cookie.html>



<Fig 9. Access without cookie>

3)Persistent Mode(Socket Number Check)



<Fig 10. One socket transmit many file & (Close – reuse) socket – persistent mode success>