

컴퓨터네트워크 Project #1 Report

2017012251 윤영훈

Server Design

우선, main 함수의 line 85. `printf("Here is the message : \n%s\n", buffer);` 까지의 코드는 ClientServer_Example의 코드를 그대로 적용했다. 단지 좀 더 유동적으로 파일을 받기 위해 line 39. `char buffer[1024];` 로 buffer의 size를 증가시키고, 계속해서 browser의 입력을 받을 수 있도록 `while(1)`을 적용했다.

buffer에서 filename과 filetype을 추출하기 위해 `buffer_cutting` 함수를 구동했고, 이 함수는 main 함수 바깥에 미리 선언해두었다 (line 14 - 25). `buffer_cutting`을 통해 arr 배열 내부에 cutting된 buffer들이 입력되고, 이때 arr[0]의 값이 GET or POST일 경우에만 `filename_and_filetype` 변수에 filename + filetype인 arr[1] + 1을 입력한다. (arr[1] == '/index.html'이므로 '/'을 제거)

다음 작업에서 `filename_and_filetype`을 filename과 filetype으로 나눌 예정이므로 나누기 전에 line 97. `FILE *myfile = fopen(filename_and_filetype, "r");`을 통해 file을 open한다.

filename과 filetype을 선언하고 `strtok` 함수를 통해 'index.html' (예시) 형식의 `filename_and_filetype` 변수를 각각 filename = 'index', filetype = 'html'로 나눠주었다.

file size를 구하기 위해 line 104. `size_t myfile_size;`를 선언하고, file size의 가장 대표적인 측정법인 `fseek -> ftell` 방법을 사용하여 file size를 측정한다.

`fread`를 통해 file을 read하고, Response message를 선언하기 위해 line 112. `char *response = (char *)malloc(myfile_size + 2048);`을 실행한다. 이때 Response message의 업데이트를 위해 동적 할당을 한다.

요구받은 file의 type (확장명)을 확인하기 위해 `content_type`을 역시 동적할당한다. 이후 `strcmp`을 통해 filetype을 확인하여 `content_type`에 다음과 같이 매칭한다.

html	jpeg or jpg	gif	pdf	mp3
text/html	image/jpeg	image/gif	application/pdf	audio/mp3

마지막으로 Response message를 최종적으로 format에 맞게 구현한다. 이때 피드백을 위한 가독성을 위해 실행에 꼭 필요한 `content-length`, `content-type` format만 입력한다. 이후 `memcpy` 함수를 통해 response와 file_buf를 최종적으로 메모리 복사하고, `send` 함수를 통해 socket으로 전달하는 것으로 마무리한다.

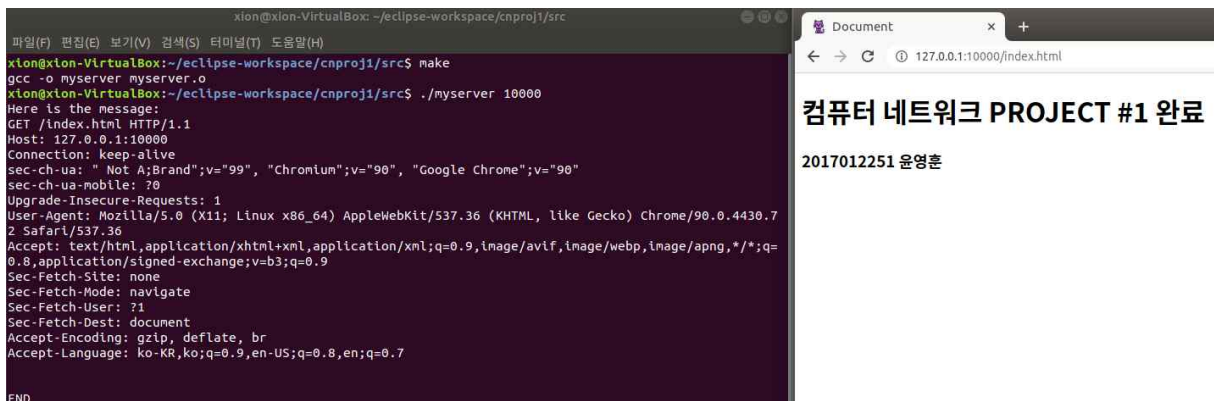
Difficulties

교수님께서 주신 ClientServer_Example 예제의 Skeleton Code와 Socket Programming PDF 및 Internet Searching을 통해 성공적으로 Project를 마무리할 수 있었다. 가장 어려웠던 점은 Socket 관련 작업보다, 의외로 strtok 활용에 대한 부분이었다. Request message에 strtok가 제대로 적용이 되었나 확인해보기 위해 printf 함수를 통해 출력을 지시했는데, 아무리 코드를 바꿔도 Segmentation Fault가 출력되었다. 결국 Youtube 및 strtok 관련 강의와 예제들을 들으며 해결했으나 본인이 얼마나 함수를 이해하지 않은 채 무작정 사용하려고만 했는지, 포인터 관련 지식 및 이해도가 얼마나 부족했는지 느낄 수 있었다.

Socket 관련 작업중에서는, 특별히 어려웠다고는 코드의 전체적인 흐름을 본인이 원하는대로 구현하는 것에 대해 어려움을 느꼈다. 동적할당, buffer의 크기, 변수 memory 시작 주소, file open & read, response message 입력 등 정말 다양한 부분에서 크고 작은 trouble이 발생하여 본인이 design했던 초기 모델과 어쩔 수 없이 변경된 부분, 삭제된 부분, 새로 추가된 부분들이 발생했다. 그 중 하나로 favicon 관련 오류는 원인을 파악하는 것조차 헤메고 시간을 많이 허비했다. favicon 변환기를 통해 favicon.ico를 생성하여 해당 폴더에 insert하는 것으로 해결했다. 또한 segmentation fault 관련 오류에 대해서는 메모리 참조 및 할당, NULL 등 관련 오류가 발생할 수 있는 모든 부분들을 실시간으로 코드를 실행시키는 동시에 결과를 check하면서 해결했다. 이렇게 프로그래머는 자신이 원하지 않는 상황이 발생하면 유동적으로 상황에 맞게 대처할 수 있는 능력이 필요하다는 사실을 다시 한번 깨닫게 되었다.

Sample Output

1. index.html



```
xion@xion-VirtualBox: ~/eclipse-workspace/cnproj1/src
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
xion@xion-VirtualBox:~/eclipse-workspace/cnproj1/src$ make
gcc -o myserver myserver.o
xion@xion-VirtualBox:~/eclipse-workspace/cnproj1/src$ ./myserver 10000
Here is the message:
GET /index.html HTTP/1.1
Host: 127.0.0.1:10000
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7

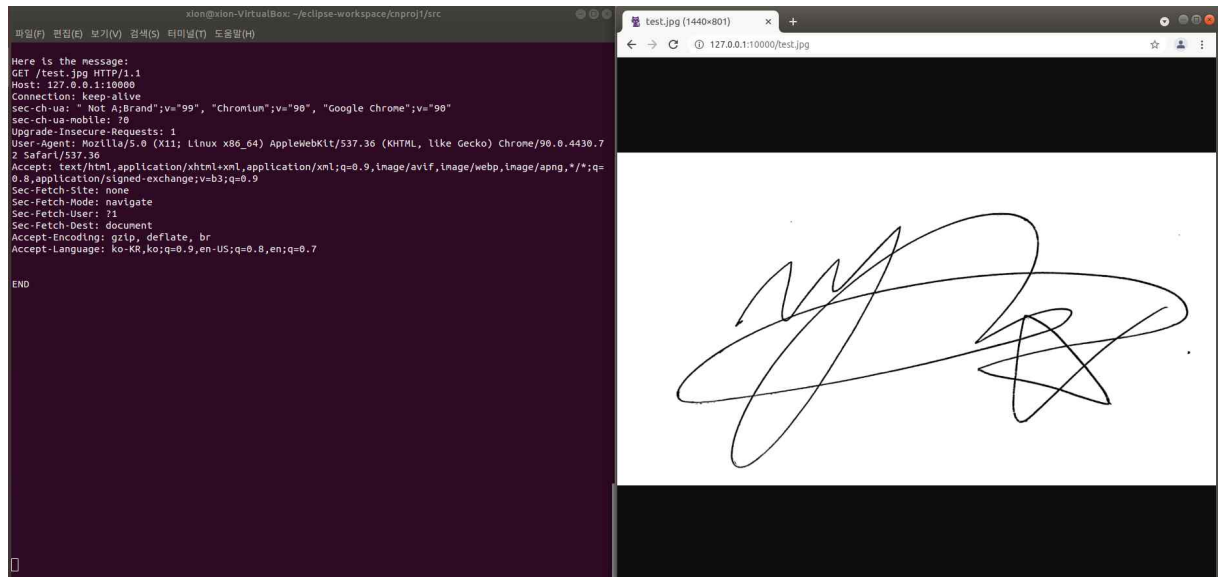
END
```

Document x +
← → ↻ ⓘ 127.0.0.1:10000/index.html

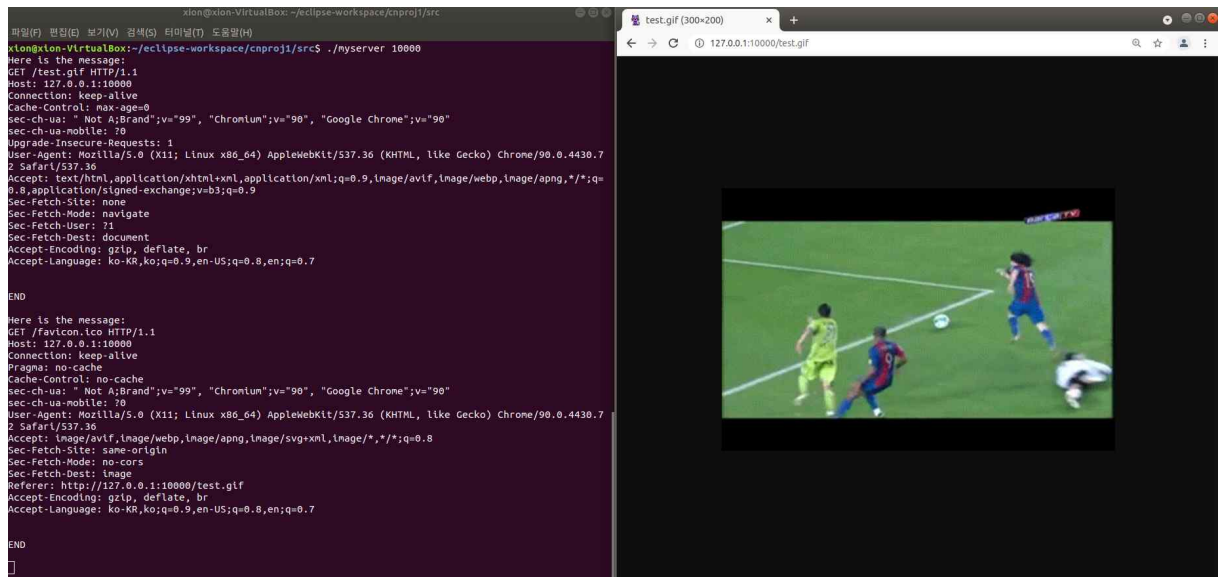
컴퓨터 네트워크 PROJECT #1 완료

2017012251 윤영훈

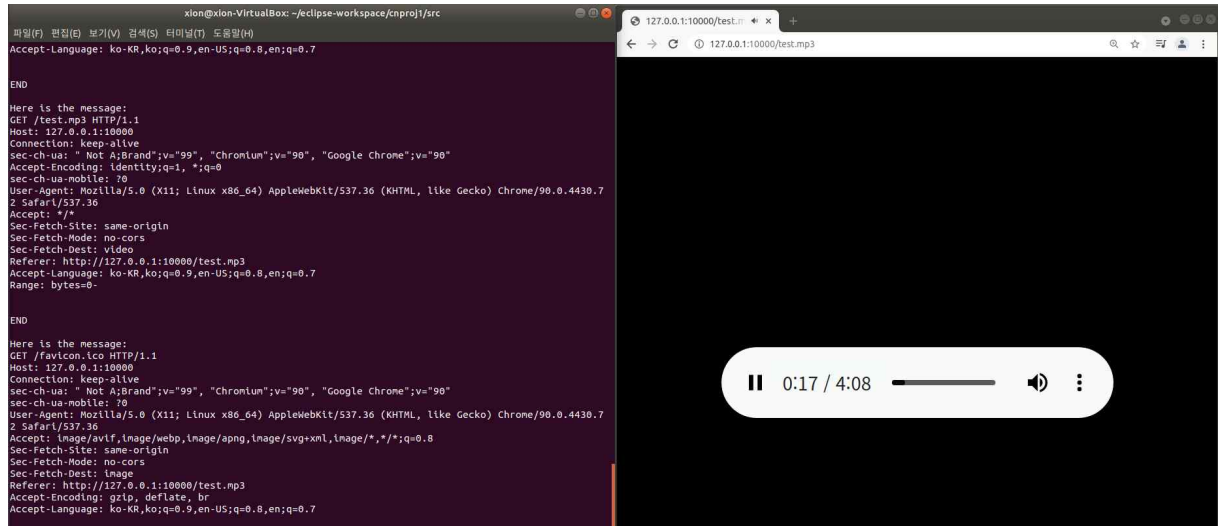
2. test.jpg



3. test.gif



4. test.mp3



5. test.pdf

