2020

컴퓨터 네트워크 과제

은하수 교수님

소프트웨어학부 2017012197 여채린

Part 1

1. Server 로 전달된 request message 각 field 에 해당하는 값 설명

```
x220@x220-ThinkPad-X220:~/바탕화면/2017012197$ ./server 8080
GET / HTTP/1.1
Host: 127.0.0.1:8080
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Geck
o) Chrome/75.0.3770.100 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
```

GET / HTTP/1.1

- HTTP 요청 메시지의 첫 줄은 요청 라인(request line)이라 부르고, 이후 줄들은 헤더 라인(header line)이라고 부른다.
- 요청 라인은 3개의 필드, 즉 방식(method)필드, URL필드, HTTP 버전 필드를 갖는다.
  - 방식(method)필드 : GET, POST, HEAD, PUT, DELETE를 포함하는 여러 가지 다른 값을 가질 수 있다. HTTP 메시지의 대부분은 GET 방식을 사용한다. GET 방식은 브라우저가 URL 필드로 식별되는 객체를 요청할 때 사용된다.
  - URL 필드 : GET방식을 통해 브라우저가 URL 필드로 식별되는 객체를 요청하게 된다.
  - HTTP 버전 필드 : HTTP의 버전을 나타낸다. 여기에서 브라우저는 HTTP/1.1 버전을 구현하고 있다.

Host: 127.0.0.1:8080

- 객체가 존재하는 호스트를 명시하고 있다. 여기서는 객체가 존재하는 호스트가 127.0.0.1:8080이다.
- 호스트 헤더라인이 제공하는 정보는 웹 프록시 캐시에서 필요로 한다.

Connection: keep-alive

- 브라우저가 서버에게 지속 연결 사용을 원한다는 것을 의미한다.

Upgrade-Insecure-Requests: 1

- 브라우저가 요청을 가져 오기 전에 안전하지 않은 요청 (예 : http :)을 안전한 대안 (예 : https :)으로 자동 업그레이드하는 데 사용된다.

User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36

- 사용자 에이전트, 즉 서버에게 요청을 하는 브라우저 타입을 명시하고 있다.
- 여기서는 Linux에서 실행되는 Chrome브라우저를 나타낸다.

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3

- 요청에 대한 응답으로 허용되는 미디어 범위 목록을 표시 할 수 있다. 별표 "*"문자는 미디어 유형을 범위로 그룹화하는 데 사용되며 "* / *"는 모든 미디어 유형을 나타내고 "type / *"은 해당 유형의 모든 하위 유형을 나타낸다. 클라이언트가 제공 한 범위 세트는 요청 컨텍스트에서 허용되는 유형을 나타낸다.

Accept-Encoding: gzip, deflate, br

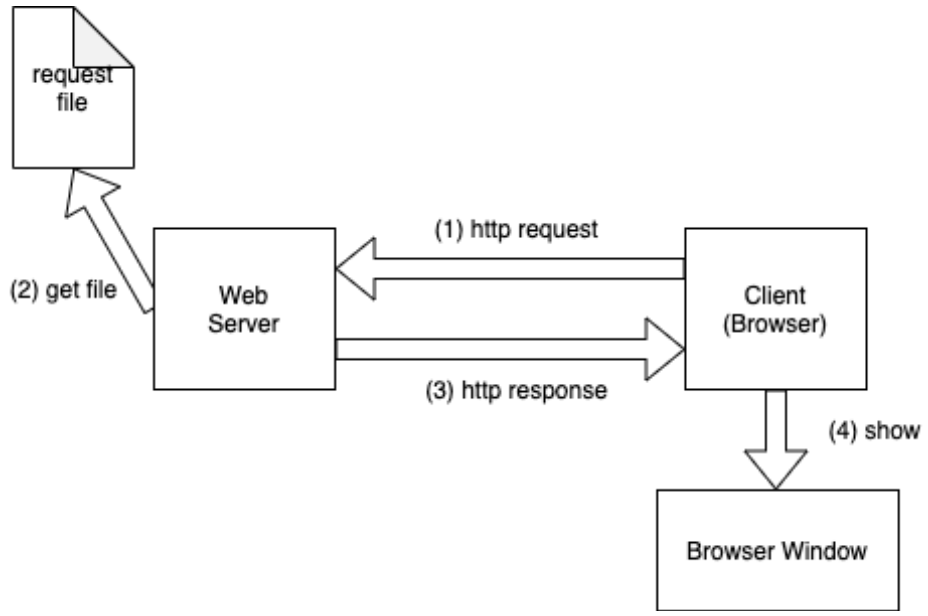
- Accept-Encoding request-header 필드는 Accept와 비슷하지만 응답에서 허용되는 content-coding 값을 제한한다.

Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7

- Accept-Language request-header 필드는 Accept와 비슷하지만 요청에 대한 응답으로 선호되는 자연어 집합을 제한한다.

Part 2

1. 서버 디자인에 대한 대략적인 설명 및 도식



서버를 디자인한 설계 순서

1. 소켓을 연다.
   sockfd = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
2. 클라이언트의 Request를 수락한다.
   newsockfd = accept(sockfd, (struct sockaddr*) & cli_addr, &clilen);
3. Request의 헤더 내용을 가져와서 파싱한다.
   szTokenData = strtok(szCopyData, " "); // GET
   szTokenData = strtok(NULL, " "); // /index.html(파일 형식)
   szTokenData = strtok(NULL, " ₩r₩n"); // HTTP/1.1
   szTokenData = strtok(szCopyData, "."); // /index
   szTokenData = strtok(NULL, "."); // html
4. 파싱하여 얻은 데이터에 알맞는 파일 타입이 존재하는지 체크한다.

```
if (strcmp(szFileFormat, "html") == 0 || strcmp(szFileFormat, "htm") == 0)
        strcat(szContentType, "text/html");
else if (strcmp(szFileFormat, "gif") == 0)
        strcat(szContentType, "image/gif");
else if (strcmp(szFileFormat, "jpeg") == 0 || strcmp(szFileFormat, "jpg") == 0)
        strcat(szContentType, "image/jpeg");
else if (strcmp(szFileFormat, "mp3") == 0)
        strcat(szContentType, "audio/mp3");
else if (strcmp(szFileFormat, "pdf") == 0)
```

```
            strcat(szContentType, "application/pdf");
    else if (strcmp(szFileFormat, "ico") == 0)
            strcat(szContentType, "image/x-icon");
```

5. 알맞는 파일 타입이 존재하면 성공 헤더를 작성한다.

```
char* szResponseHeader = NULL;
int nHeaderSize = strlen(szStatus) + strlen(szDate) + strlen(szContentType) + 1;
szResponseHeader = (char*)malloc(sizeof(char) * nHeaderSize);
bzero(szResponseHeader, sizeof(char) * nHeaderSize);
strcat(szResponseHeader, szStatus);
strcat(szResponseHeader, szDate);
strcat(szResponseHeader, szContentType);
strcat(szResponseHeader, "₩r₩n");
```

6. 성공 헤더와 파일 데이터를 클라이언트에게 전송한다.

```
int n = write(newsockfd, szResponseHeader, nHeaderSize + 1);
```

7. 클라이언트가 요청한 파일을 전송한다.

```
n = write(newsockfd, bFileData, n);
```

8. 소켓을 닫는다.

```
close(sockfd);
```

2. 구현 시 어려웠던 점과 해결 방법

http request 를 한번 이상 받아야 하므로 main 에서 while(1)을 돌렸습니다.

```
while (1)
{
    clilen = sizeof(cli_addr);
    /*accept function:
      1) Block until a new connection is established
      2) the new socket descriptor will be used for subsequent communication with the newly connected client.
    */
    newsockfd = accept(sockfd, (struct sockaddr*) & cli_addr, &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
```

서버에서 클라이언트로 header 를 보낼 때, 크기를 넉넉히 잡고 전송을 했습니다. 처음엔 정상적인 동작인 것처럼 보였습니다. 하지만 이미지파일을 테스트 할 때, 제대로 된 이미지 파일을 불러오지 못했습니다. http 통신을 할 때, header 의 크기가 정확히 일치해야 다음 데이터를 보낼 때 잘못된 데이터가 본문에 포함되지 않도록 할 수 있어서 정확한 크기를 보내야 된다는 점을 알았습니다.

정확한 크기를 보내기 위해 배열 보다는 동적 할당이 나을 것 같아서 전송 데이터를 malloc 으로 할당하여 보내는 부분을 신경 써서 구현하였습니다.

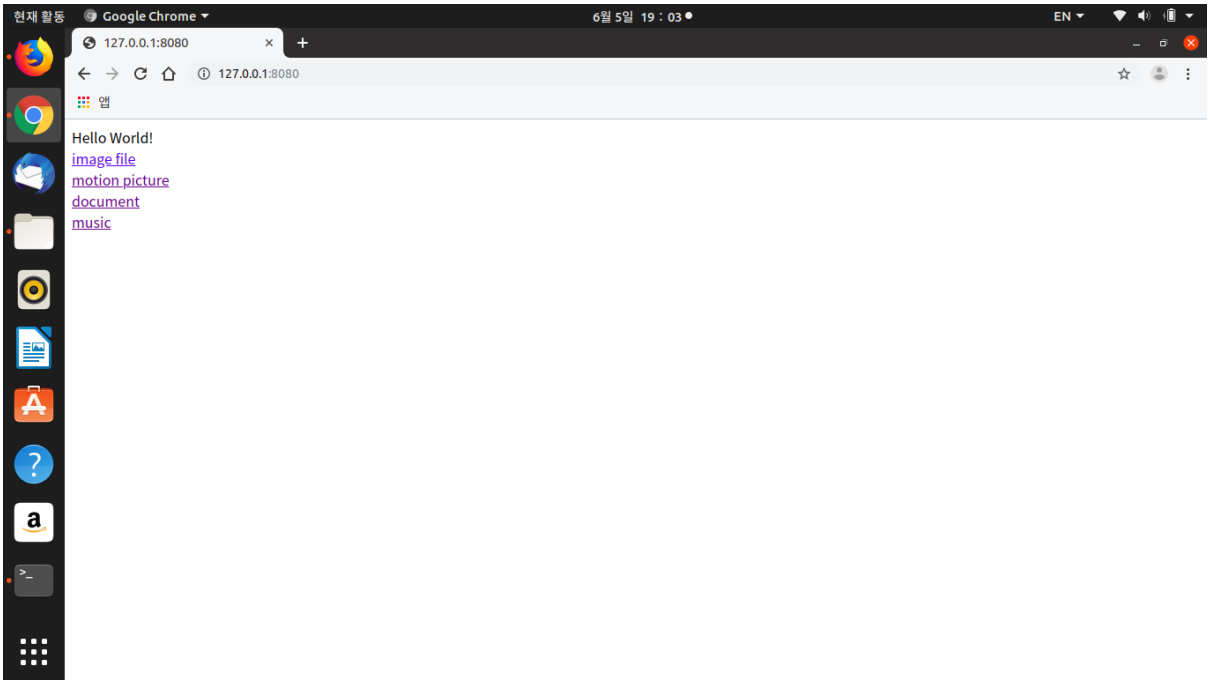
```
// Header 만들기
char* szResponseHeader = NULL;
int nHeaderSize = strlen(szStatus) + strlen(szDate) + strlen(szContentType) + 1;
szResponseHeader = (char*)malloc(sizeof(char) * nHeaderSize);
bzero(szResponseHeader, sizeof(char) * nHeaderSize);
strcat(szResponseHeader, szStatus);
strcat(szResponseHeader, szDate);
strcat(szResponseHeader, szContentType);
strcat(szResponseHeader, "\r\n");

// 요청 파일이 없으면 404 not found 메시지 작성
char szErrorStatus[32] = { 0 };
char* szErrorHeader = NULL;
int nErrorSize = 0;
strcpy(szErrorStatus, szProtocol);
strcat(szErrorStatus, " 404 Not Found\r\n");
nErrorSize = strlen(szErrorStatus) + strlen(szDate) + 1;
szErrorHeader = (char*)malloc(sizeof(char) * nErrorSize);
bzero(szErrorHeader, sizeof(char) * nErrorSize);
strcpy(szErrorHeader, szErrorStatus);
strcat(szErrorHeader, szDate);
```

또한 구현을 하면서 파일 타입은 MIME 형식을 따라야 한다는 걸 알게 되었습니다. 따라서 이에 맞게 파일타입을 결정해주는 함수를 만들었습니다. 헤더 부분을 보고 파일 타입을 결정하는데, 서버를 실행시키면 파일들이 브라우저 화면에 뜨지 않는 오류가 발생했습니다. 이를 해결하기 위해 문자 인코딩 방법을 다음과 같이 추가했습니다.

```
strcat(szContentType,"; charset=utf-8\r\n");  // 문자 인코딩 방법을 추가
```

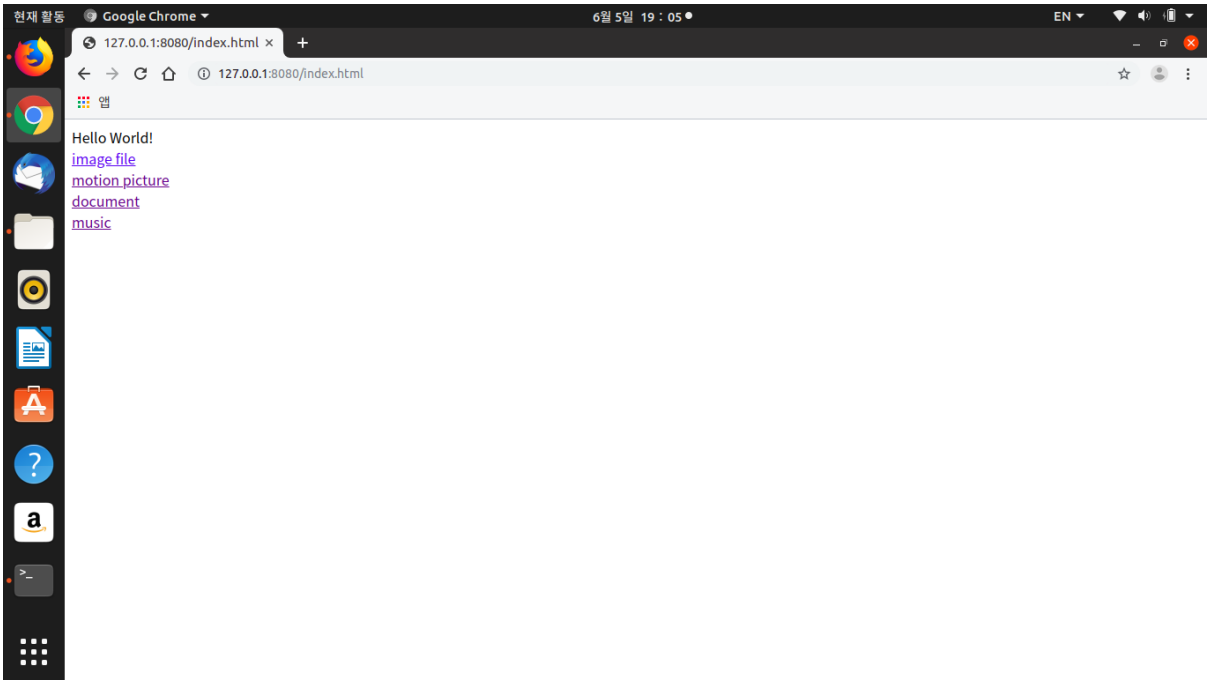
3. 동작 예시

- ./server 8080 으로 서버 실행 및 http://127.0.0.1:8080 입력



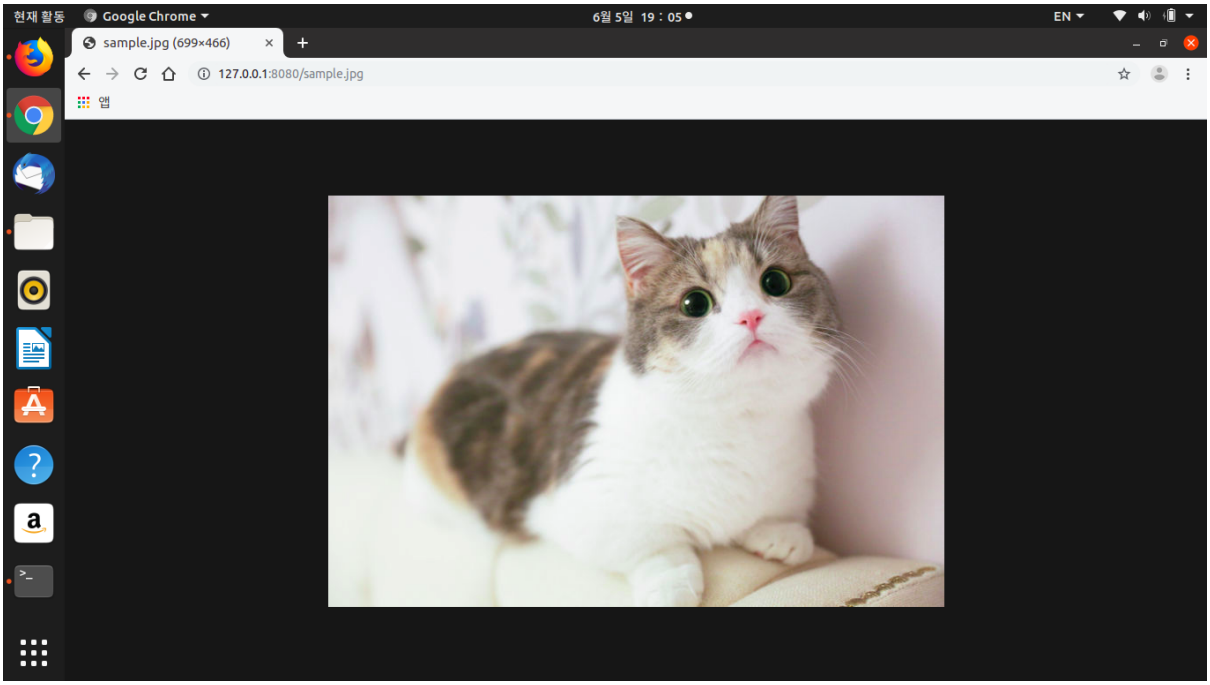
- html file

http://127.0.0.1:8080/index.html 입력



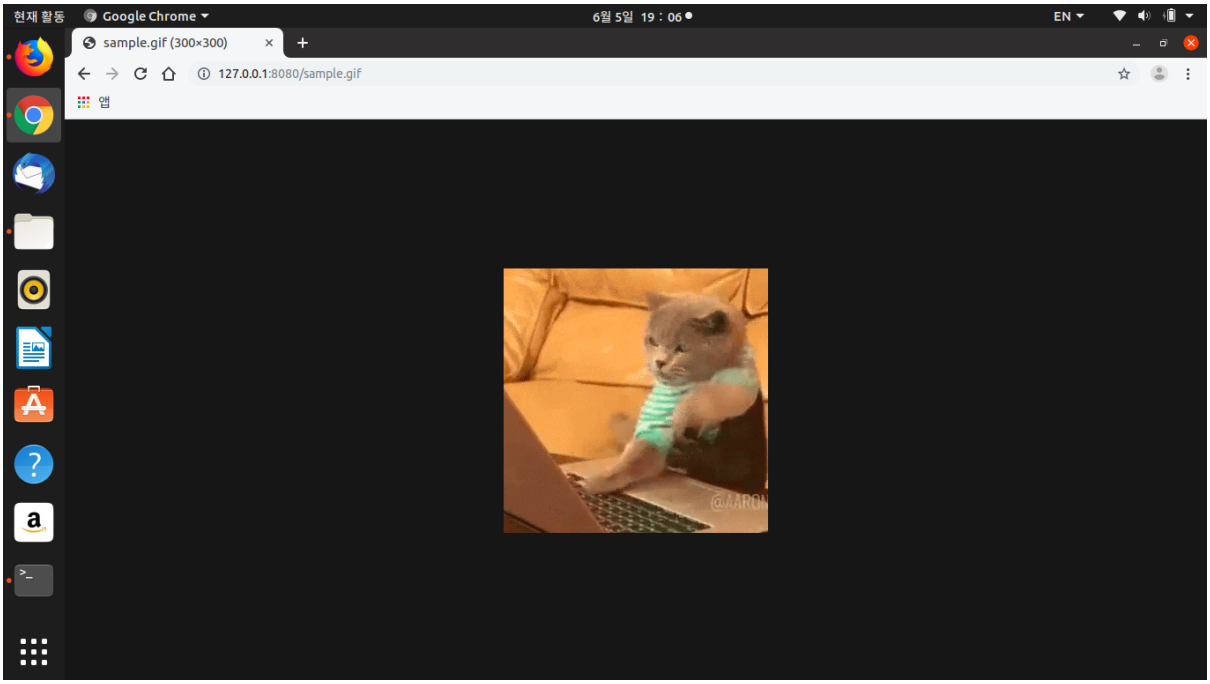
- image file(jpg)

http://127.0.0.1:8080/sample.jpg 입력



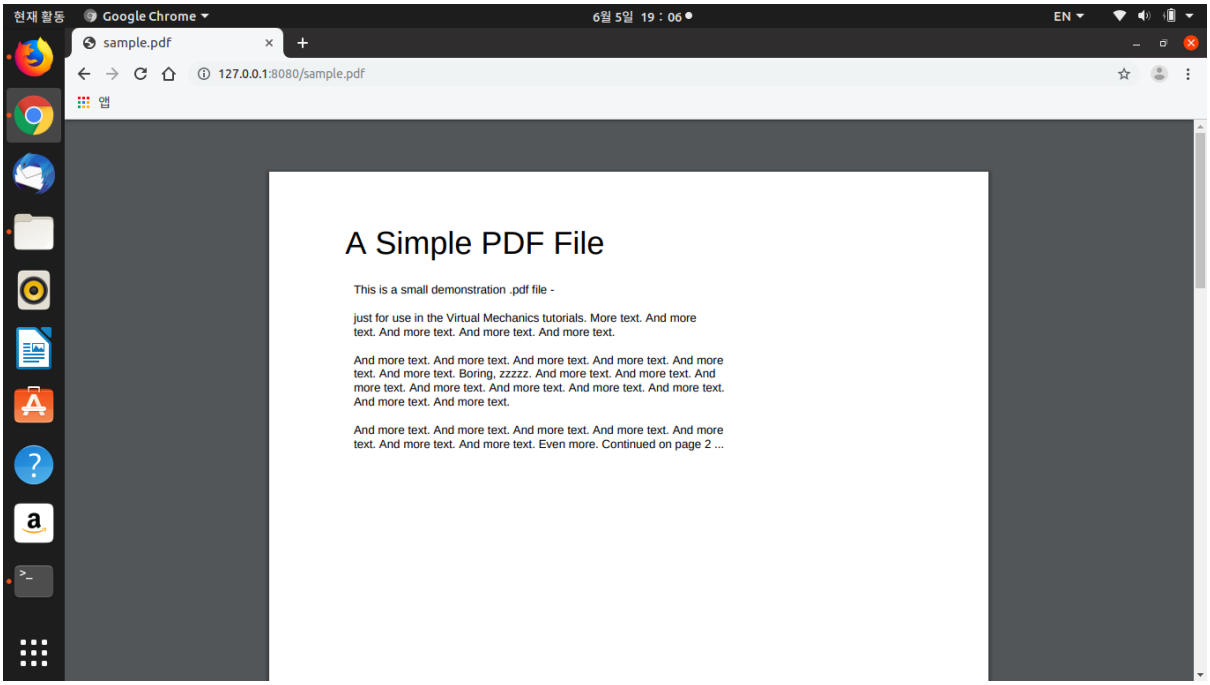
- motion picture(gif)

http://127.0.0.1:8080/sample.gif 입력



- document(pdf)

http://127.0.0.1:8080/sample.pdf 입력



- music(mp3)

http://127.0.0.1:8080/sample.mp3 입력