CSE3047
Computer Network
Project#1
Report

2021050300 김재민

Contents

1. Server Design

서버는 클라이언트의 HTTP request를 수신하고, 요청한 파일을 확인한 후 올바른 HTTP 응답을 바로 전송하는 구조로 구현하였다.

- socket() → bind() → listen() → accept() 순서로 서버 소켓 생성
- 무한 루프에서 accept() 호출 시, 각 요청을 fork()를 통해 자식 프로세스에서 처리
- 파일 요청을 확인하고, 적절한 HTTP 응답을 생성 및 전송

주요 기능

- Request message를 파싱하여 파일 이름 추출
- 요청한 파일이 없을 경우 404 Not Found 응답
- 파일 확장자에 따라 Content-Type을 설정하여 응답

2. Difficulties and Solutions

(1) Request Message Parsing

- 참고 : HTTP Request 메시지로부터 파일 이름 추출
- 문제점 : 파일 이름에 확장자 "."이 포함되지 않은 경우, Segmentation Fault 발생
- 해결 방법 : 파싱 전에 확장자 유무를 확인하고, 정상 요청만 파싱하여 예외 상황 방지

(2) HTTP Response Message 구성

- 문제점 : HTTP 응답 메시지 작성 시, 헤더와 본문(body)이 제대로 구분되지 않아 문제 발생
- 해결 방법 : 헤더 끝에 빈 줄(₩n₩n)을 삽입하여 헤더와 본문을 명확히 구분

(3) Binary File Transfer

- 문제점 : HTML과 달리 바이너리 파일은 단순 문자열 결합으로 전송할 수 없음
- 해결 방법 : 파일 크기를 측정한 후 fread(), memcpy()를 사용하여 바이트 단위로 파일을 읽고 전송

3. Test Results

(1) Hello HTML 파일

요청 URL: http://localhost:8081/hello.html

결과: Hello World! 페이지 정상 표시

(2) JPG 이미지

요청 URL: http://localhost:8081/A.jpg

결과: "A"가 표시된 JPG 이미지 정상 출력

(3) MP3 파일

요청 URL: http://localhost:8081/After_You.mp3

결과: MP3 파일 자동 재생 및 다운로드 가능

(4) 존재하지 않는 파일 요청 (404)

요청 URL: http://localhost:8081/error.html

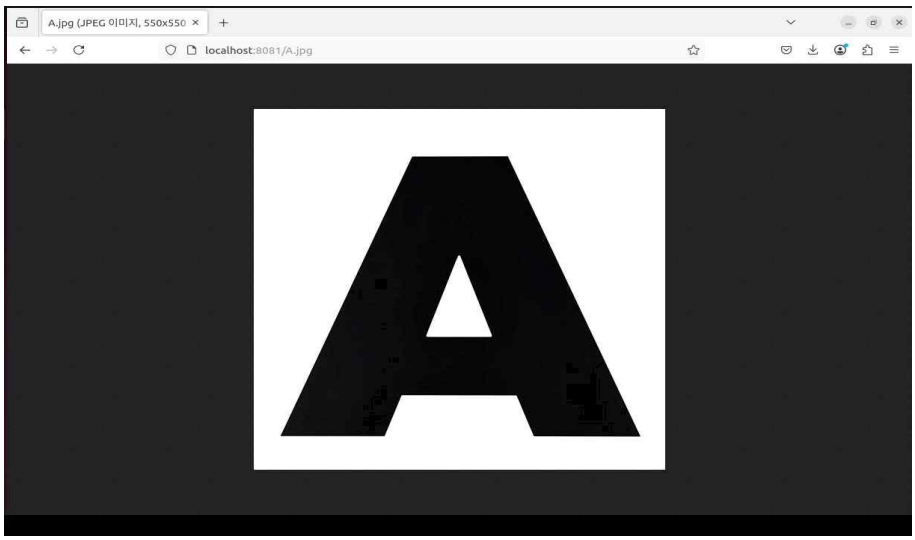
결과: 404 Not Found 에러 페이지 정상 표시

4. Conclusion

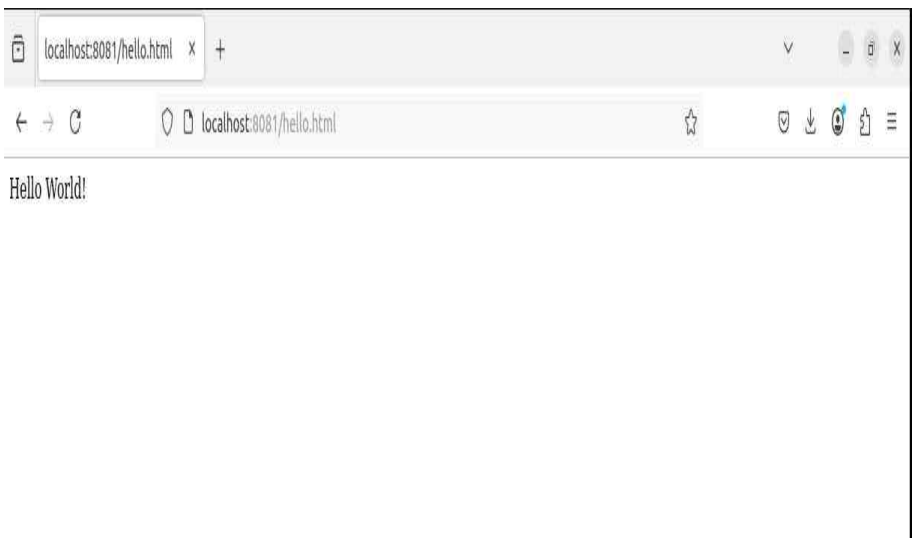
이번 프로젝트를 통해 HTTP 프로토콜의 구조와 BSD 소켓 프로그래밍의 기본을 실습할 수 있었다. fork()를 이용한 다중 클라이언트 프로세스 처리와 바이너리 파일 전송 기술을 직접 구현해 보며 네트워크 서버 구현에 대한 실질적인 경험을 얻을 수 있었다. 프로젝트 결과는 정상적으로 동작하였으며, 서버의 안정성과 기능 면에서도 만족스러운 결과를 얻었다.

5. 참고 사진

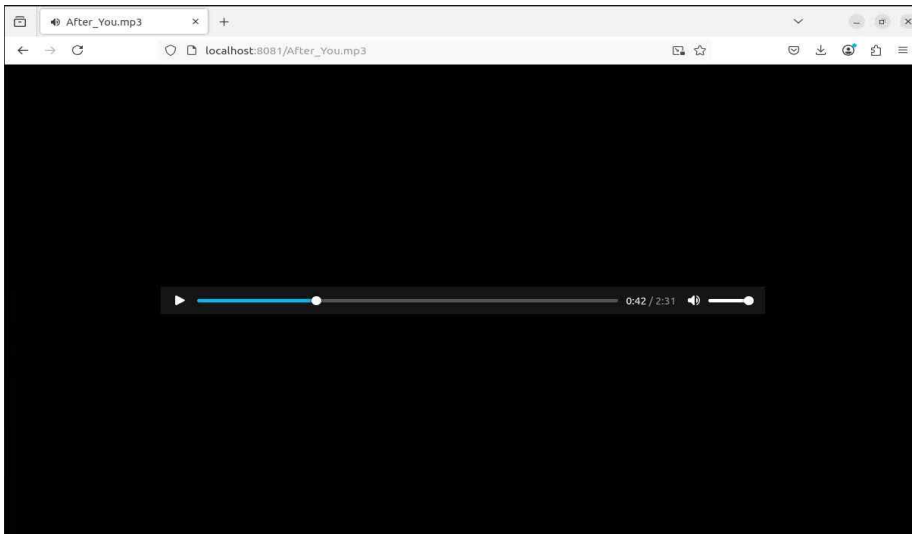
(1) A.jpg 테스트 캡처



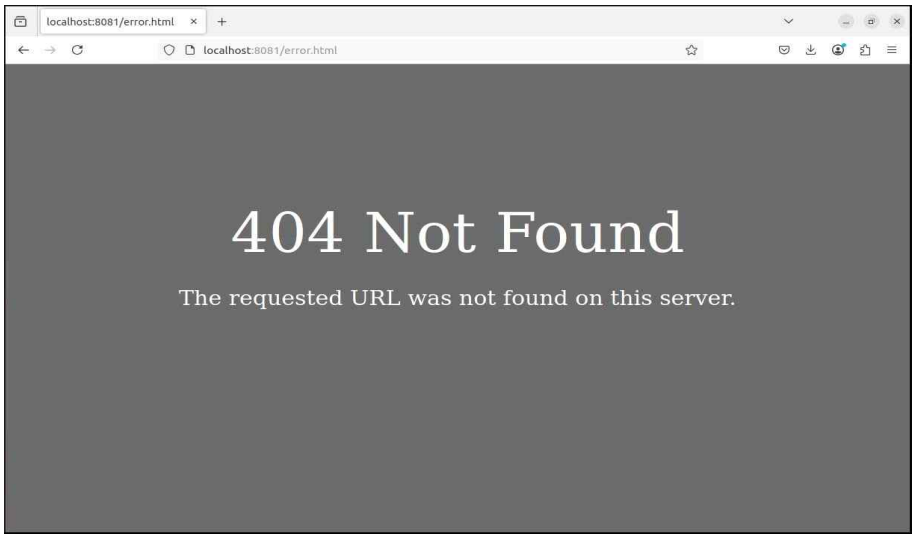
(2) hello.html 테스트 캡처



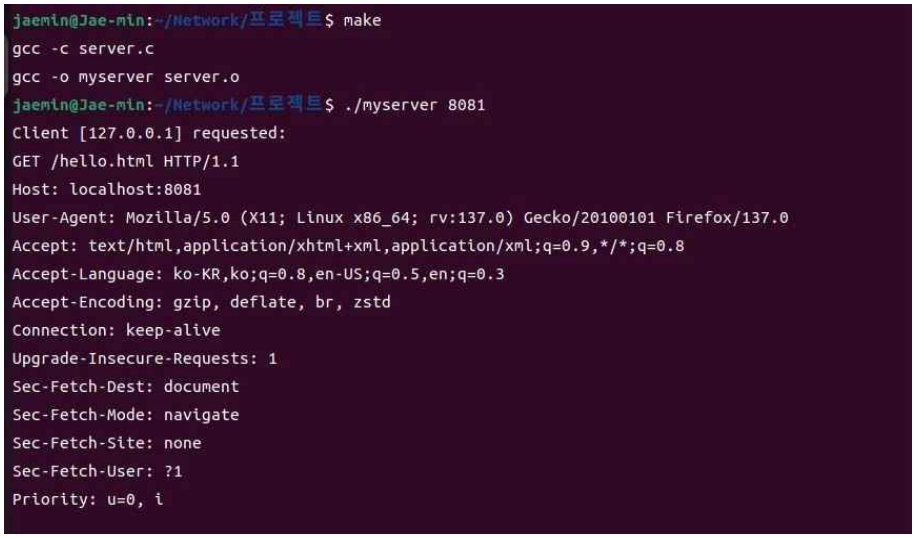
(3) After_You.mp3 테스트 캡처



(4) error (404 Not Found) 테스트 캡처



(5) server 실행 화면 캡처



```
jaemin@Jae-min:~/Network/프로젝트$ make
gcc -c server.c
gcc -o myserver server.o
jaemin@Jae-min:~/Network/프로젝트$ ./myserver 8081
Client [127.0.0.1] requested:
GET /hello.html HTTP/1.1
Host: localhost:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:137.0) Gecko/20100101 Firefox/137.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
```