CSE3027: Computer Networks

Project 1:
Concurrent Web Server using BSD Sockets Spring 2022
Instructor: Prof. Suk-Bok Lee

2016010555 문화콘텐츠학과 김우재

A. high-level description of design

(Part A)

1. 주요 변수 및 구조체 선언

```
int servsockfd, clisockfd, fd;
int portno;
struct sockaddr_in serv_addr, cli_addr;
```

첫번째로, 변수 선언입니다. server socket과 client socket을 Int타입으로 선언합니다. Port number 또한 변수로 선언하며 실제 value는 인자로 입력 받아 할당합니다. 소켓 bind시 필요한 ip주소값, 포트번호 등의 할당을 위한 구조체 또한 선언합니다.

2. 소켓생성

```
servsockfd = socket(AF_INET, SOCK_STREAM,0);
```

socket(af_inet = ipv4사용 , SOCK_STREAM = tcp타입의 전송방식 , 0 = tcp 프로토콜사용) 함수를 사용하여 tcp소켓을 생성하게 됩니다. 이는 int타입의 번호를 리턴하는데, 이를 server socket에 할당하여 server socket을 생성하게 됩니다.

3. bind작업을 위한 구조체 값 할당(ip번호, 포트넘버 등)

```
portno = atoi(argv[1]);
```

입력받은 인자를 포트번호로 할당합니다.

```
serv_addr.sin_family = AF_INET;
```

이는 bind시 소켓이 ipv4의 사용 선언을 위한 할당입니다.

```
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
```

Bind시 서버의 ip를 할당하기 위한 line입니다. INADDR_ANY는 현재의 장치의 ip주소값을 자동으로 반환해주는 명령어입니다.

```
serv_addr.sin_port = htons(portno);
```

인자로 전달받은 포트번호를 할당합니다.

4. 소켓 bind작업(소켓의 구체적 정보 bind)

```
bind(servsockfd,(struct sockaddr *) &serv_addr, sizeof(serv_addr))
```

서버소켓에 서버 ip와 포트넘버 등의 내용이 담겨있는 구조체를 파라미터로 넘겨주어 소켓을 bind합니다.

5. 소켓 listen(client접속시도 파악)

```
listen(servsockfd , 5)
```

서버소켓에 client의 접속시도를 파악하는 함수인 listen()을 사용했습니다. Client는 최대 5명까지 listen()상태에서 대기 가능하다는 의미입니다.

6. accept작업

```
clisockfd = accept(servsockfd, (struct sockaddr *) &cli_addr , &clilen);
```

server소켓과 client ip주소를 함수에 전달하여 accept 서로 accept하고 반환된 값을 client소켓 변수에 할당합니다.

7. HTTP 리퀘스트 메시지 read, print

```
n = read(clisockfd, head_buf , 9999);
printf("Here is the msg :  %s\n", head_buf);
```

클라이언트로 부터의 메시지를 read하여 버퍼공간에 저장한 뒤 버퍼의 내용(리퀘스트메시지)을 출력합니다.

(Part B)

8. HTTP 리퀘스트 메시지에서 요청파일이름 추출

```
int x=5, i=0;
bzero(req_file_name ,255);
while(head_buf[x] != 32)
{
  req_file_name[i] = head_buf[x];
  i++; x++;
}
```

리퀘스트 메시지는 포맷이 정해져있으므로, 요청파일이름이 시작되는 인덱스넘버는 항상 동일합니다. 따라서 공백문자가 나타나기 전까지의 문자를 버퍼공간에 저장하여 요청파일이름을 추출합니다.

9. HTTP 리스폰스 메시지(헤더)와 파일(실제 데이터)전송 과정

9-1 . 응답메시지(헤더) 전송 "HTTP/1.1 --- --\r\nContent-Type: ---/--- ”

1. 요청파일이 서버 디렉토리 내에 위치하지 않는다면 `"HTTP/1.1 404 NOT FOUND\r\nContetn-Type: text/html\r\n\r\n"`
2. 추출한 요청파일이름에서 파일형식을 다시 추출합니다. 추출된 파일형식이 지원하는 파일 형식이라면 `"HTTP/1.1 200 OK\r\nContent-Type: ";` Content-Type에 전송파일 형식을 명시하여 줍니다.
3. 지원하지 않은 파일형식이라면 `"HTTP/1.1 404 NOT FOUND\r\nContetn-Type: text/html\r\n\r\n"` 를 응답메시지로 전송합니다.

9-2 . 파일(데이터) 전송

파일데이터를 read > read한 데이터를 write 의 과정을 더이상 read할 파일의 데이터가 0일때 까지 반복합니다. 반복 사이클 중에 Write(출력)버퍼가 차게되면 자동으로 tcp send버퍼로 데이터가 내려가게 됩니다.

10. 소켓 close()

작업을 마무리하면 소켓을 close하여줍니다.

B. Difficulties of design server socket

소켓생성 – bind – listen – accept를 통해 소켓을 맺고 client의 요청메시지를 출력하는 과정까지는 이해와 프로그래밍이 어렵진 않았습니다. 또한 서버의 응답 메시지 전송까지도 문제는 없었습니다.
하지만 헤더필드보다 상대적으로 큰 용량인 파일데이터를 전송하는 과정, 리눅스의 read-write 과정을 파악하는 것이 엄청나게 큰 어려움이었습니다. 프로그래밍을 하면서 리눅스의 파일시스템과 read, write를 이해하는데 오랜 시간을 할애하였습니다. 결국 출력(write) 버퍼가 가득차게 되면 tcp의 전송버퍼에 자동으로 데이터가 내려간다는 것을 이해하고 난 후 읽어들일 파일데이터가 없을때까지 read-write를 반복하여 파일전송을 할 수 있겠다는 생각을 하게 되었고, 성공적으로 작동하게 되었습니다.

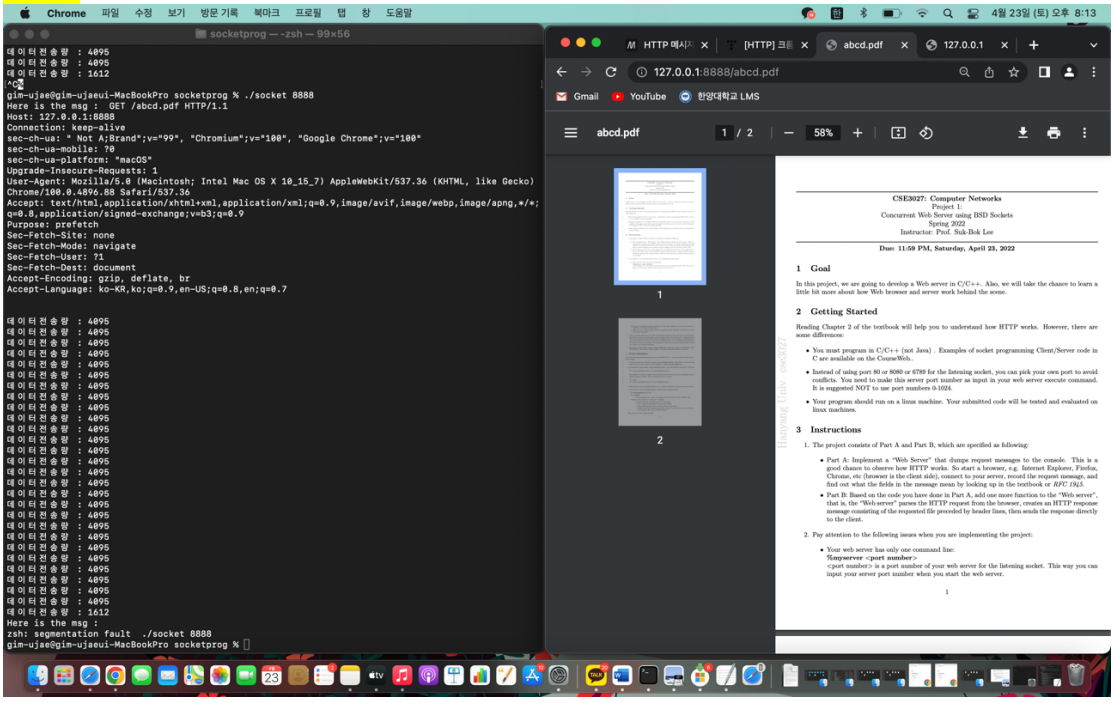
C. Explain Sample output

partA)

```
gim-ujae@gim-ujaeui-MacBookPro socketprog % ./socket 8888
Here is the msg :  GET /abcd.pdf HTTP/1.1
Host: 127.0.0.1:8888
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="100", "Google Chrome";v="100"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/100.0.4896.88 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
Purpose: prefetch
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
```

Line 1 : 8888번으로 포트번호 인자 전달.
Line 2 : (http 요청메시지) - GET 메소드(리소스를 클라이언트로 get 요청) , 요청파일 디렉토리 , HTTP버전
Line 3 ~ : http 헤더.(호스트명, 브라우저 종류 및 버전 등)

partB)



127.0.0.1 로컬주소와 디렉토리 주소 접속시 브라우저 화면