**http\_server 작성**

**20160400 조홍철**

**1. 접속**

접속의 전체적인 순서는 socket 생성 -> bind -> listen -> accept 가 된다. socket 생성에서는 listening socket 역할을 할 소켓 s를 생성한다. s를 생성한 후에 sockaddr\_in 구조체 server\_addr 변수를 사용하여 s에 ip주소와 포트를 bind한다. 이 때 포트번호의 경우 프로세스를 실행할 때 1번째 변수로 넘겨주는 argv[1]를 사용한다.

bind의 과정에서 인자로 전달받은 argv를 확인하는 과정이 필요하다. argc(인자의 수)가 2개가 맞는지 확인하는 과정을 거치게 되고 argc가 2개라면 인자가 잘 전달된 것이기 때문에 bind를 진행하게 된다. argv[1]은 문자열로 넘어오기 때문에 atoi(argv[1])을 통해 숫자로 변환하고, htons(atoi(argv[1])을 함으로써 Big endian으로 변환해준다.

listen과정에서 문제가 없다면 accept을 통해서 큐에 대기하고 있던 client의 요청을 가져와 data 통신을 위한 소켓(client\_s)을 생성한다. accept를 while문 안에 넣어 client가 접속을 끊어도 계속 다른 client와 접속할 수 있게 된다.

**2. 데이터 전송**

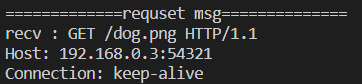
여러 client의 접속을 동시에 해결하기 위해서 다중 접속 서버 형식으로 만든다. 여기서는 fork를 사용하여 다중 접속 서버를 구현한다. while 문 안에서 accept가 되면 data 송수신을 위한 소켓 clinet\_s가 만들어지고, fork를 실행하여 자식과 부모 프로세스로 나눈다.

-부모 프로세스: 부모 프로세스는 fork == 0을 만족하지 못하기 때문에 바로 else문으로 들어가게 되어 data 송수신 소켓(client\_s)을 닫고, 다시 while문의 처음으로 돌아와 accept에서 blocking 상태가 된다. 자식 프로세스가 죽으면 SIGCHILD에 따른 zombie\_handler함수를 실행함으로써 좀비 프로세스가 생성되지 않도록 한다.

-자식 프로세스: 자식 프로세스는 실질적으로 client와 데이터를 송수신한다. 먼저 필요 없는 listening socket(s)을 close한다. http 서버는 client의 request 메시지를 토대로 response 메시지를 보내야하기 때문에 먼저 client에서 보낸 메시지를 받는다.

데이터 전송은 recv -> file이름 추출, file 타입 추출 -> file 타입에 따른 데이터 전송을 while문으로 반복한다. while문 안에 recv ~ send를 반복함으로써 접속을 끊지 않고 통신을 할 수 있게 된다.

recv를 통해서 client의 request 메시지를 버퍼(buf)에 담는다. strtok 함수를 사용하여 buf에 있는 내용을 \r\n 자르고 포인터를 반환한다. 이렇게 반환된 포인터는 clinet가 보낸 request 메시지의 첫 줄이 된다. 이 첫 줄에는 파일의 이름과, 파일 이름을 통해서 type을 알 수 있다.

다음과 같이 GET /dog.png HTTP/1.1인 메시지는 파일의 이름이 dog.png이고 type이 png임을 알 수 있다.

이렇게 잘라낸 첫 줄의 정보에 따라 request 메시지가 달라지게 된다.

-파일 이름과 파일 형식

파일 이름을 알아내기 위해서 find\_file\_name 함수를 만들었다. 함수의 인자로는 request 메시지의 첫 줄을 넘기게 되고, 이 문자열을 다시 “ “(공백)을 기준으로 나누고 각각의 주소를 char \*b[5]에 저장한다. “ “을 기준으로 GET /dog.png HTTP/1.1 을 나누면 파일의 이름이 있는 /dog.png는 b[1]에 그 주소가 저장되고 /를 제외하기 위해서 dog.png부터의 주소(&b[1][1])를 반환하여 파일의 이름을 알 수 있다. 이 때 파일 이름이 /만 있는 경우에는 index.html이기 때문에 반환 전에 strcmp를 통해서 “/”문자열과 동일하면 b[1]의 값을 “/index.html”로 바꿔준다.

파일의 형식을 알아내기 위해서는 check\_content\_type 함수를 만들었다. 함수의 인자로는 request 메시지의 첫 줄을 넘기고 strstr 함수를 통해 문자열 비교를 한다. 이를 통해서 알 수 있는 파일의 형식은 html, png, jpg가 된다. str 함수를 통해서 request 메시지 첫 줄에 .html, .jpg, .png 중에 어떤 문자열이 있는지 알 수 있고 각각의 형식에 따라 1~4(1 : html, 2 : jpg, 3:png ,4: 나머지 알 수 없는 파일형식) 값을 반환하였다.

-파일의 형식

- response 메시지 전송

메시지를 전송하기 위해서 send\_request 함수를 만들었고, 그 인자로는 데이터 송수신을 위한 소켓의 descriptor번호, 파일 이름, 파일 형식을 넘기게 된다. 파일 이름과 파일 형식은 앞에 find\_file\_name 함수와 check\_content\_type을 통해서 알 수 있다. send\_request함수 안에서는 file type에 따른 switch 문을 통해서 보내는 header를 달리한다. switch 문을 통해서 header를 작성 후 먼저 보내고, 뒤이어 while문을 통해서 data를 보낸다. header를 보낼 때는 html, jpg, png, 그 외 형식에 대해서 나누어 보내는데, 알 수 없는 형식에 대해서는 error.html을 보낼 수 있도록 file을 error.html로 바꿔준다. 데이터 전송을 할 때는 인자로 넘겨받은 파일 이름(알 수 없는 경우에는 error.html)을 통해서 read함수로 파일 내용을 data\_buf에 담고 read의 return값이 0이 될 때 까지 while문을 반복하여 데이터를 전송한다.

header나 데이터를 보낼 때에 data 크기가 중요한대, header를 전송할 때 header의 크기를 알아내기 위해서 header\_buf에 담기 내용의 길이를 strlen(header\_buf)를 통해 알아낸다. data의 경우 stat구조체를 이용하여 open한 파일의 크기를 알 수 있다.

**3.실행결과**

