computer network submit
2020031267 권소연

1. A high-level description of my server's design.

```
void http_handler(int new_fd); /* http 내용 받아 오고 쓰기 */
void write_content(int new_fd, char *header, int content); /* html 파일을 추가로 만들고 쓰기
*/
void find_mime(char *ct_type, char *uri); /* 파일 형식 지정하기 */
int main(int argc, char **argv)
```

#0> server.c는 http_handler(), write_content(), find_mime(), main()의 4가지 주요 함수로 구성되어 있다.

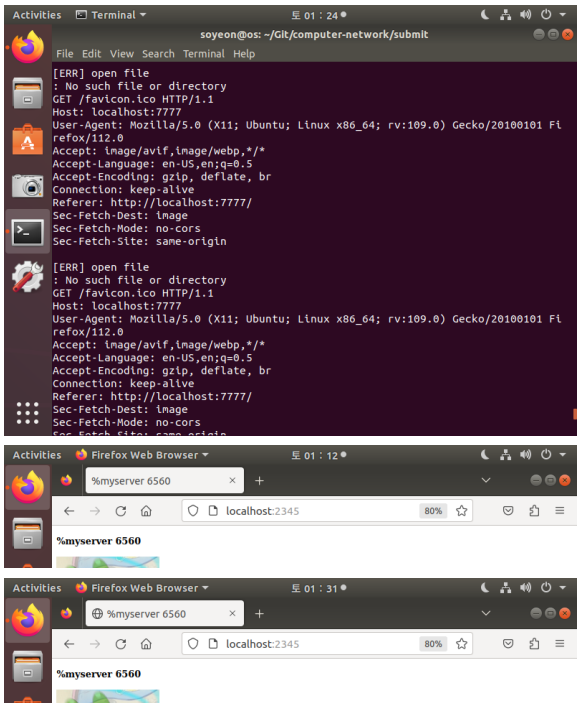
```
   int port = atoi(argv[1]); /* 포트 번호를 main()의 인자로 지정하기 */
   int sock_fd, new_fd; /* sock_fd는 서버의 소켓, new_fd는 클라이언트의 소켓 */
   sock_fd = socket(AF_INET, SOCK_STREAM, 0) /* 소켓 만들기 */
   /* 소켓에 IP 주소와 포트 번호 할당하기 */
   my_addr.sin_family = AF_INET; /* 주소 체계 */
   my_addr.sin_addr.s_addr = htonl(INADDR_ANY); /* 내 장치의 IP 주소 */
   my_addr.sin_port = htons(port); /* 포트 번호 설정 */
   bind(sock_fd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)
   listen(sock_fd, BACKLOG) /* 클라이언트 요청을 대기하기 */
   while(1) { /* 클라이언트 요청을 받아들이는 loop() */
       /* 클라이언트 요청 받아들이기 */
      new_fd = accept(sock_fd, (struct sockaddr *) &their_addr, &sin_size)
       /* 프로세스 생성하기 */
       int pid = fork();
       if (pid == 0) { /* 자식 프로세스 */
           close(sock_fd);
           http_handler(new_fd);
           close(new_fd);
           exit(0);
       }
       if (pid != 0) {
           close(new_fd);
       }
       if (pid < 0) {
           perror("[ERR] fork\n");
       }
   }
}

void http_handler(int new_fd) {
   char header[BUF_SIZE];
   char buf[BUF_SIZE];
   /* buf에 데이터를 읽어오기, http 헤더 내용이 버퍼에 저장 */
   if(read(new_fd, buf, BUF_SIZE) == -1) {
```

```
        perror("[ERR] read request\n");
        write_content(new_fd, header, 5001); /* 500 에러: http 헤더를 읽어올 수 없음
*/
        return;
    }
    printf("%s", buf); /* http 헤더 내용을 터미널에 출력하기 */

    /* http 헤더 내용에서 보낼 데이터 확인하기 */
    char *method = strtok(buf, " "); /* method = "GET" */
    char *uri = strtok(NULL, " "); /* uri = "html.html" */
    if (method == NULL || uri == NULL) {
        perror("[ERR] URI\n");
        write_content(new_fd, header, 5002); /* 500 에러: http 헤더 형식이 맞지 않음
*/
        return;
    }
    /* uri에 맞는 파일 찾기 */
    char safe_uri[BUF_SIZE];
    char *local_uri;
    struct stat st;

    strcpy(safe_uri, uri);
    if (!strcmp(safe_uri, "/")) strcpy(safe_uri, "/html.html");
    local_uri = safe_uri + 1;
    if (stat(local_uri, &st) < 0) {
        perror("[WARN] URI\n");
        write_content(new_fd, header, 404); /* 404 에러: 찾고자 하는 파일이 없음 */
        return;
    }
    /* 보낼 소켓에 파일을 열기 */
    int fd = open(local_uri, O_RDONLY);
    if (fd < 0) {
        perror("[ERR] open file\n");
        write_content(new_fd, header, 5003); /* 500 에러: 파일을 열 수 없음 */
        return;
    }
    /* http 형식에 맞춘 헤더 넣기 */
    long ct_len = st.st_size;
    char ct_type[40];
    find_mime(ct_type, local_uri);
    sprintf(header, HEADER_FMT, 200, "OK", ct_len, ct_type);
    write(new_fd, header, strlen(header));
    /* 파일 쓰기 */
    int cnt;
```

```
    while ((cnt = read(fd, buf, BUF_SIZE)) > 0)
        write(new_fd, buf, cnt);
}
```

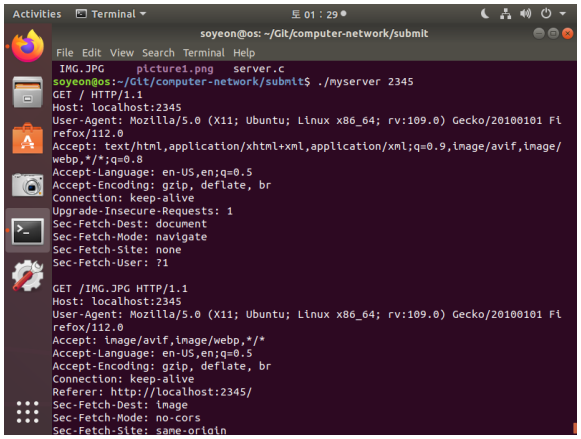
#1> main()함수에서는 port번호를 인자로 받는다. TCP소켓을 생성, 소켓에 주소와 포트 번호 할당, 클라이언트 요청 대기가 이루어진다. 또한, while문을 통해 클라이언트의 요청을 받아 들인다. 프로세스를 사용하여 보다 효율적으로 반복되는 request를 받을 수 있도록한다.
#2> http_handler() 함수에서는 http 내용을 받는다. http 헤더파일을 받아 어떤 파일을 요청하는지 확인한다. 이때, fd라는 소켓을 생성하여 요청 받은 파일을 연다. 이후 파일을 보낸다.
#3> write_content() 함수에서는 파일이 없으면 에러 메시지를 text/html 형식으로 웹브라우저에 띄운다.
#4> find_mime() 함수는 받은 파일 uri의 형식을 배정해 준다.

2. What difficulties did you face and how did you solve them?



#1> 옆에 터미널에서 보다시피, open() 에서 자꾸 오류가 발생했다. 그러나 아무리 봐도 c언어 코드 상에는 오류가 없었기에 찾기가 어려웠다.
그러다 /favicon.ico 파일에서 오류가 난다는 걸 읽었고, c언어 코드를 고치려는 시도에서 벗어나 html에 icon 파일을 추가해야하는 것을 알게 되었다.
/favicon.ico 파일은 html 상에서 header에 작은 아이콘이라는 사실을 알았다.
#2> html 파일에서 아이콘 없음을 처리해 줬다.

3. Include and briefly explain some sample outputs of your client-server (e.g. in Part A you should be able to see an HTTP request)



GET (uri) HTTP/1.1의 형식으로 request 메시지가 터미널에 뜬다.