Programming Assignment #5:

HTTP/1.0 Web server

Due: 24rd Dec. (Sun), 11:59 PM, **HARD DEADLINE**

1. Introduction

이번 과제는 본 과목의 마지막 과제로, HTTP/1.0을 따르는 멀티 쓰레드 웹 서버를 제작하는 것이다. 이번 과제를 수행함으로써 소켓 프로그래밍과 쓰레드 프로그래밍을 연습한다. 또한, 웹 서버의 동작 원리에 대해서 배울 수 있다.

이번 과제는 본 과목의 기말고사로 간주되며, 제출하지 않으면 큰 불이익을 받는다. 또한, 성적 처리를 하기 위해 이번 과제의 지연 제출은 허용되지 않는다.

PA #4에서 작성했던 서버-클라이언트 프로그램의 서버를 확장해 Google Chrome과 같은 상용 웹 브라우저에서 접속시킬 수 있는 웹 서버를 제작한다. 해당 웹 서버는 HTTP 프로토콜을 준수하여 클라이언트로부터 들어오는 요청에 대한 적절한 응답을 하고, 멀티 쓰레드를 이용해 동시에 여러 요청을 처리한다.

2. Background

2.1. HTTP (Hypertext transfer protocol)

HTTP는 인터넷에서 가장 널리 쓰이는 프로토콜의 하나로, WWW(world wide web)에서 HTML(hypertext markup language) 문서를 전달하기 위해 이용한다. HTTP는 TCP나 UDP의 80번 포트로 데이터를 주고 받는다.

웹 브라우저로 도메인 주소 "**http://nyx.skku.ac.kr/**" 에 접속한 상태를 소켓 연결의 측면으로 간단히 보면 다음과 같다.

1. 웹 브라우저 프로세스(이하 클라이언트)가 새로운 TCP 소켓을 만든다.
2. "**nyx.skku.ac.kr**" 서버(이하 서버)의 **TCP 80번 포트**에 접속해 연결을 맺는다.
3. 클라이언트는 서버로 "**/**"에 위치한 웹 문서를 달라는 HTTP 요청 메시지를 보낸다.
4. 서버는 클라이언트의 요청에 대응하는 "**/**"에 해당하는 HTML 문서를 찾거나 동적으로 만들고, 적절한 HTTP 응답 헤더와 HTML 문서를 클라이언트에게 전송한다.
5. 소켓 연결을 종료하고, 새로운 사용자 요청이 발생하면 1로 되돌아간다.

HTTP에 관한 더 구체적인 정보는, 다음 링크와 교재 11.5장을 참고한다.

http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

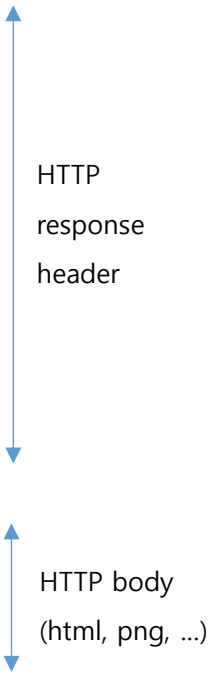
2.2. Example of a HTTP Request/Response

요청 (여기서 빨간색 **/**은 서버의 상대 주소 "**/**"에 위치한 문서를 요청하는 것이다)

```
GET / HTTP/1.0
Host: nyx.skku.ac.kr
(NULL LINE)
```

응답 (**굵게 처리된 부분**을 잘 확인할 것)

```
HTTP/1.1 200 OK
Date: Mon, 02 Dec 2014 05:27:18 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.3.3-7+squeeze19
Expires: Tue, 01 Jan 2002 00:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Vary: Accept-Encoding
Content-Length: 7212
Connection: close
Content-Type: text/html; charset=UTF-8
(NULL LINE)
<!DOCTYPE html PUBLIC "……">
<html>
[[이하 생략]]
```



HTTP 정의에 따르면, HTTP 헤더(응답과 요청 모두)는 **개행 문자로 <CR><LF>를 사용**한다. NULL LINE은 HTTP의 개행 문자(<CR><LF>)로만 이루어진 라인이다. HTTP 요청 메시지의 경우 **요청의 끝**을 알리며, HTTP 응답에선 HTTP 응답 헤더와 HTML 문서 본문의 **경계**를 나눠주는 역할을 한다.

이를 확인하는 간단한 테스트 코드는 다음과 같다. (sockfd는 **nyx.skku.ac.kr:80** 에 연결된 소켓)

```
char request[] = "GET / HTTP/1.0\r\nHost: nyx.skku.ac.kr\r\n\r\n";
char response[5000];
write(sockfd, request, strlen(request));
int read_bytes = read(sockefd, response, 5000);
write(1, response, read_bytes);
read_bytes = read(sockefd, response, 5000);
write(1, response, read_bytes);
```

2.3. HTTP version

HTTP는 1991년에 V0.9, 1996년에 V1.0, 1999년에 V1.1이 발표되었고, 현재 거의 모든 상용 프로그램은 V1.1을 따르고 있다(통신 예제 응답의 "HTTP/1.1" 이란 문구). V1.0에서 V1.1으로 발전하면서 개선된 사항엔, 한 TCP 연결에서 하나 이상의 요청/응답을 처리하도록 하는 **Persistent connections**, 여러 요청/응답을 하나로 묶어서 한꺼번에 전송하는 **Pipelining** 등이 있다. 하지만 이런 기능들을 구현하는 것은 복잡하니, 본 과제에선 HTTP/1.0 을 따르는 서버를 구현한다.

위의 통신 예시를 보면, 임의로 구현한 **HTTP/1.0 요청**에 대해, NYX의 웹 서버(아파치)는 **HTTP/1.1로 응답**을 한 것이다. 본 과제에선 서버를 구현하는 것이니, 모든 요청에 대해 **HTTP/1.0으로 응답**을 해야 한다. 참고로, 상용 웹 브라우저는 HTTP/1.1로 요청을 전송할 것이다.

2.4. Request methods

다음 HTTP 요청의 시작을 보면, GET이란 문자열로 요청이 시작함을 알 수 있다.

```
GET / HTTP/1.0
```

이 **GET**이란 문자열은 HTTP 요청의 한 method로써, 주로 데이터를 받아올 때 사용하는 것이다. 다른 방식은 **POST, HEAD, PUT** 등의 다른 methods가 있는데, 이번 과제의 입력으로는 GET 요청만 입력된다고 가정한다. GET을 받으면, 특별한 처리 없이 요청 받은 해당 파일만 전송하도록 한다. 상용 웹 브라우저 상에서 서버상으로 웹 페이지를 호출하면, GET으로 요청이 전송될 것이다.
http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods

2.5. Response headers

2.2절의 예제를 보면 10줄의 응답 헤더가 존재한다. 제각각 의미가 있는 헤더이지만, 이번 절에선 본 과제에서 사용할 **세 가지 중요한 헤더**만 설명한다. 다른 헤더에 관심이 있다면, 다음 페이지를 참조할 수 있다: http://en.wikipedia.org/wiki/List_of_HTTP_header_fields#Response_fields

2.5.1. Status code

응답 헤더의 첫 줄을 보면 다음과 같이 시작한다.

```
HTTP/1.1 200 OK
```

HTTP/1.1은 HTTP V1.1 방식으로 응답을 한다는 선언이고, **200 OK**는 HTTP 요청을 정상적으로 처리해 응답한다는 코드이다. 다른 코드의 종류와 의미는 다음 웹 페이지를 참조한다.
http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

본 과제에서 사용하는 다른 응답 코드로는 **403 Forbidden**, **404 Not Found**이 있다. 제각각 문자

그대로의 의미를 뜻하며, 4절에서 상태 코드를 전송하는 규칙을 확인하도록 한다. 참고로, 본 과제에선 **HTTP/1.1**이 아닌 **HTTP/1.0** 으로 응답을 시작할 것이다.

2.5.2. Content-Length

이 헤더는 HTTP 응답 헤더를 제외한, HTTP 응답 본문, 즉 **해당 파일의 크기**를 뜻한다. 예제에서 보면, **Content-Length**로 7212란 길이가 명시되었으니, **HTTP body**는 정확히 7212바이트이다. 참고로, 헤더의 종료는 NULL LINE이 등장하는 것으로 확인할 수 있다.

2.5.3. Content-Type

이 헤더는 해당 HTTP body, 즉 파일의 종류를 뜻하고 MIME type으로 기술된다. 예시의 **`text/html`**는 text 형태의 html 파일임을 뜻한다. 이 필드를 참조함으로써 브라우저는 해당 파일을 적절한 방식으로 그려낼 수 있다. 왜냐하면 파일의 형식에 따라 그리는 방식도 다르기 때문이다. 본 과제에서 쓰이는 것과 달리 실제 웹 페이지 주소엔 확장자란 개념이 존재하지 않기 때문에 이렇게 MIME type을 지정하면 브라우저가 파일의 형식을 잘 파악할 수 있다.
http://en.wikipedia.org/wiki/Internet_media_type

2.6. Handling sockets

소켓 read/write 시엔, 내가 요청한 크기만큼 실제로 송신된다는 보장이 없다. 예를 들어, 받는 쪽과 보내는 쪽의 read/write 처리 속도가 다를 경우, 소켓 버퍼에 처리되지 못한 데이터가 쌓이다 어느 순간 가득 차게 된다. 이 때 write를 수행하면 요청한 크기보다 작은 양만큼 수행될 것이다.

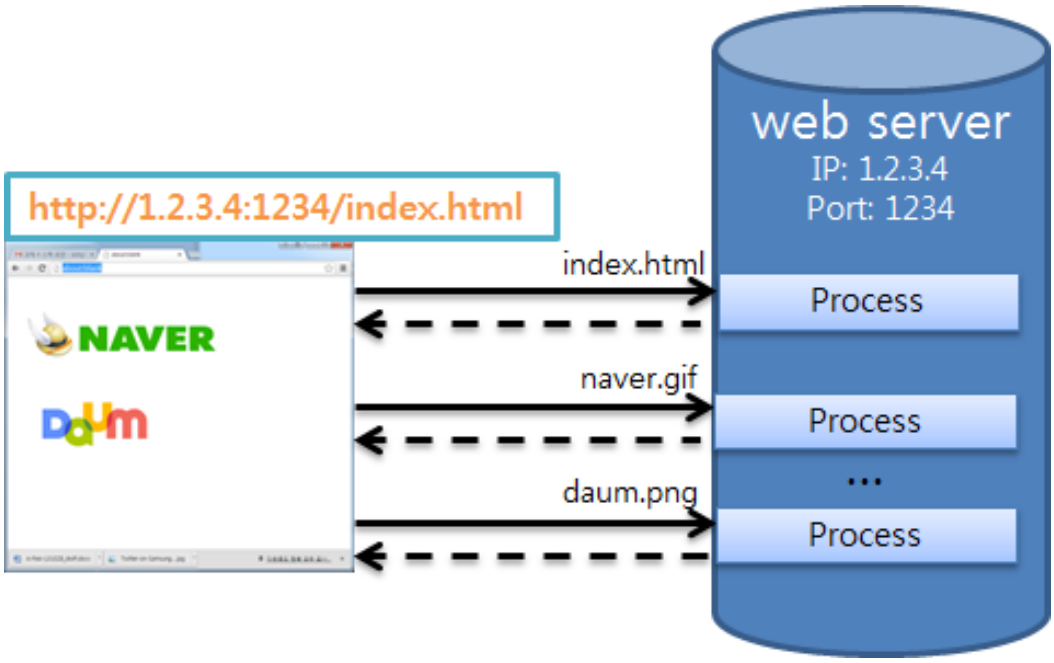
다음은 어떤 파일을 읽어서 소켓에 쓰는 예시이다.

```
long length = st.st_size; // file length
while (length > 0)
    long sent = 0;
    long bytes_in_buffer = read(fd, buffer, min(length, sizeof buffer));
    if (bytes_in_buffer == 0) // EOF
        break;
    do {
        sent += write(sk, buffer + sent, bytes_in_buffer - sent);
    } while (sent < bytes_in_buffer);
    length -= sent;
}
```

내가 100 바이트를 전송하고 싶다면, loop를 돌며 write를 시도하여 정확히 100 바이트가 전송됨을 확인하여야 한다.

3. Overview

웹 서버를 그림으로 그린 예시는 다음과 같다.

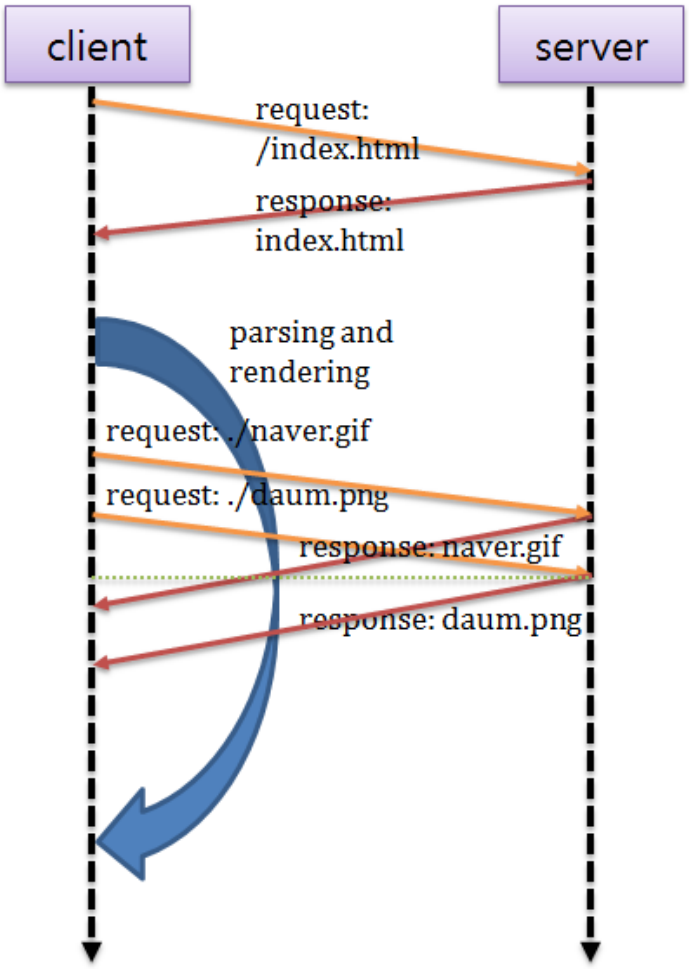


위 그림의 브라우저처럼 출력되는 html 문서의 코드는 다음 상자에 있다.

```
<html>
<body>
<img src="./naver.gif"><br>
<img src="./daum.png">
</body>
</html>
```

서버로부터 위와 같은 html 문서를 받으면, 웹 브라우저는 해당 파일의 위에서 아래로 해석하며 우리가 보는 화면을 그린다. 거의 모든 html 문서는 독자적으로 완성되지 않고 다른 외부 파일을 같이 이용한다. 예를 들면, javascript, css, 그림 등이 있고 위 코드에선 붉은 색으로 강조된 그림 파일이 이에 해당한다. 이런 외부 자원을 만나면 브라우저는 해당 서버로 요청을 보내 파일을 다운로드 받는다. 모든 외부 자원을 받은 다음, 이 모든 파일을 조합하여 하나의 페이지를 그려내어 사용자에게 보여준다.

아래 그림은 해당 html 코드를 그리기 위하여 브라우저와 서버 사이에 일어나는 통신의 순서를 나타낸 그림이다. 그림과 같이, 어떤 요청에 대한 응답 도중에 새로운 요청에 대한 응답을 시작할 수 있어야 할 것이다. (**Multi-processing**)



본 과제에서는 이런 멀티-프로세싱을 구현하기 위해 쓰레드란 개념을 이용하도록 한다. 또한, 교재 11.5장과 11.6장에 HTTP에 대한 설명과 예제 웹 서버가 있으니 참고할 수 있다.

4. Problem specification

과제를 단순화하기 위해, 웹 브라우저로 접속하면 화면에 DEFAULT란 문자열을 찍도록 하는 웹 서버가 스켈레톤 코드로 제공된다.

<서버 생성>

- 서버의 기본 포트는 1234이고, 프로그램 인자가 1개 존재하면 그 숫자를 포트로 쓴다.
- 소켓에 접속 요청이 들어오면, 연결을 맺고 **새로운 쓰레드를 생성**하여 해당 연결을 전담하여 처리하게 하고, 메인 쓰레드는 바로 다음 요청을 처리하기 위해 대기한다.
  - 쓰레드를 생성할 때, **쓰레드간 원치 않는 공유**가 발생하지 않도록 한다. (e.g., socket descriptors, etc.)

<요청의 처리>

- 소켓에서부터 클라이언트의 요청을 전부 읽어낸다.
  - (참고) 채점시 서버로 들어오는 한 요청의 크기는 1024바이트를 넘지 않는다.
- 요청의 첫째 줄을 분석해 적절한 문서를 찾는다.
  - 요청의 시작은 **`GET /test.html HTTP/1.0`** 과 같은 형식이며, **`/test.html`** 이란 상대 주소를 구하고 이를 서버의 working directory에서부터 찾는다.
  - 만약 상대 주소가 **`/`** 였다면, **`/index.html`** 이란 상대 주소를 받았다고 본다.
- 파일 확장자는 html, jpg, png, gif, js, css만 허용한다. (다른 확장자는 이용할 권한이 없다)
  - 아래 **수행 기록**은 예외로 처리한다.

<응답의 전송>

- 클라이언트 요청에 대응하는 응답을 해당 소켓으로 전송한다.
- HTTP 응답은 <HTTP 헤더> + <NULL LINE> + <파일의 내용> 으로 구성된다.
- HTTP 헤더는 다음 정보를 이용해 만든다.
  - <Status code, Content-type, Content-length> 세 가지 필드만 만들어 전송한다.
  - 스켈레톤 코드 `http.c` 파일의 `make_http_header()` 함수를 이용하면 HTTP 헤더로 사용할 수 있는 문자열을 동적으로 생성한다. 이 헤더를 사용한 후 반드시 **메모리를 해제**한다.
  - Status code는 아래 세 가지 경우만 고려한다.
    - 권한과 파일 존재는 서버에서 `open` 시도 시 발생하는 에러로 판별한다.

정상일 때	OK
해당 파일에 접근할 권한이 없을 때	E_NOT_ALLOWED
파일이 존재하지 않을 때	E_NO_FILE

  - Content-type은 해당 파일의 확장자에 따라 다음 값을 이용한다.

HTML 파일	TYPE_HTML
JPG 파일	TYPE_JPG
PNG 파일	TYPE_PNG
GIF 파일	TYPE_GIF
JS 파일	TYPE_JS
CSS 파일	TYPE_CSS
에러 발생	TYPE_ERROR

  - Content-length는 해당 파일의 크기이다. <파일의 내용>의 크기와 Content-length는 일치하여야 한다.
- 전송을 마친 뒤 소켓을 닫는다.

<연결 종료 이후>

- 메인 쓰레드는 스스로 종료하지 않고 무한히 연결을 기다린다.
- 다른 쓰레드는 **수행 기록**을 남기고 종료한다.
  - 다른 쓰레드들은 종료 직전엔 반드시 **detached** 상태이어야 한다.

<수행 기록>

- 전송 이력을 메모리에 기록하고, 사용자의 요청에 따라 이 정보를 전송한다.
- 서버 쓰레드가 종료되기 전, 다음과 같이 수행 기록을 남긴다.
  - (전송을 완료한 다음, 다른 말로 소켓을 닫고 종료되기 직전에 업데이트할 것)
  - 발생한 모든 연결의 개수
  - 헤더를 제외한 보낸 데이터 크기의 합(byte), Content-length의 합이라 생각해도 좋음.
  - 각 파일 확장자마다, 에러가 발생하지 않고 정상적으로 전송한 횟수의 합.
- 여러 쓰레드가 동시에 수행 기록을 남길 때 동기화시키기 위해 lock 등을 이용한다.

- `/stat` 이란 주소를 요청 받으면, `make_report()` 함수를 호출해 수행 기록에 대한 리포트를 만든다. (HTML 파일로 취급할 것)

```
char *make_report(int connections, long long int sent_bytes,
          int html, int jpg, int png, int gif, int js, int css, int error)
```

  - `http.c`에 구현된 `make_report()` 함수에 인자를 넣으면 수행 기록에 대한 리포트를 동적으로 만든다.
  - `/stat` 이 요청된 시점에 커넥션이 3번 발생했고, 총 2400 바이트를 전송했으며, 각각 html, jpg, hwp 파일을 요청 받았다면 다음과 같이 `make_report()` 함수를 호출한다.
  - `make_report(3, 2400, 1, 1, 0, 0, 0, 0, 1);`
  - `connections = html + jpg + png + gif + js + css + error` 식은 항상 성립한다.
- `make_report()` 로 만들어진 문자열을 <파일의 내용>으로 클라이언트에게 전송한다.
  - 이 상황은 HTML 요청을 정상적으로 전송했다고 간주한다.
  - 파일의 크기는 `make_report()` 함수가 생성한 문자열의 길이이다.
  - 문자열을 전송하기 전 적절한 헤더를 생성하여 보내줘야 할 것이다.
- 전송 이후 자기 자신에 대한 수행 기록을 업데이트한 뒤, `free()`로 할당 받은 메모리를 해제하고 쓰레드를 종료한다.

- `/reset` 주소를 요청 받으면, 다음과 같이 처리한다. (HTML 파일로 취급)
  - 새로운 서버 쓰레드를 생성하지 않고, (메인 쓰레드는 `accept` 이후 그냥 대기)
  - 다른 서버 쓰레드가 수행 기록을 남기고 전부 종료될 때까지 기다린 다음,
  - 현재까지 전송한 데이터 크기의 합을 0으로 초기화한 뒤, (`sent_bytes`)

- `/stat` 을 요청 받은 것과 같이 현재까지 수행 기록을 클라이언트에게 전송한다. (전송 이후 자기 자신의 기록을 업데이트하는 것까지 포함)
- 수행 기록을 남긴 뒤, 새로운 서버 쓰레드를 생성할 수 있도록 서버를 정상화한 뒤 쓰레드를 종료한다.

5. Skeleton codes

이번 과제 수행을 위해 다음 9개의 파일이 주어진다:

`Makefile:`	GNU `make` 도구를 위해 필요한 파일
`webserver.c:`	기초적인 웹 서버가 구현된 파일
`http.c:`	제공 함수들이 구현된 파일
`http.h:`	프로그램에서 이용하는 헤더
`string_sw.c:`	문자열 처리 함수를 구현한 C 파일
`string_sw.h:`	문자열 처리 함수가 선언된 헤더 파일
`index.html:`	서버를 테스트 할 수 있는 웹 페이지 파일
`daum.png:`	`index.html`을 그리기 위해 포함된 그림 (다음 로고)
`naver.gif:`	`index.html`을 그리기 위해 포함된 그림 (네이버 로고)

프로그램을 컴파일하기 위해 `make` 명령을 사용하면 `webserver` 프로그램이 생성된다.
http.c에 담긴 함수들은 전부 thread-safe하다.

6. Restrictions

- 과제 구현을 위해 여태까지 배운 리눅스 시스템 콜/라이브러리 함수를 이용한다.
- 동적 메모리 할당 함수를 제외한 다른 라이브러리 함수는 **사용할 수 없다**. 필요하다면 직접 구현하여 사용한다.
  - 쉬운 디버깅을 하기 위해 라이브러리 함수를 사용할 수 있다. 다만, 제출한 파일엔 해당 함수들이 직접 사용되지 않아야 한다. (주석 등으로 처리해도 괜찮음)
- 어떤 자원을 동적으로 할당 받았다면, 프로그램 종료 전에 반드시 해제해야 한다.
  - 여기서 자원이란 파일, 메모리, 소켓 등을 뜻한다.
  - 내가 할당 받지 않은 자원을 해제하지 않도록 조심한다.
- **프로그램 컴파일 시 발생하는 모든 warning을 잡는다.**

7. Hand in instruction

- 작성한 코드 상단의 주석에 이름과 학번을 작성한다.
- **`make` 명령을 통해 프로그램이 제대로 만들어지는지 확인한다.**

    - **제출한 코드가 컴파일되지 않을 경우, 경우에 따라서 해당 제출은 채점되지 않거나 심한 페널티를 받게 될 것이다.**
- `webserver.c` 파일의 이름을 학번.c 로 명명한다.
- 본 과제 수행 시 구현 방법과 디자인을 설명하는 보고서를 PDF 포맷으로 작성하여 "`학번.pdf`" 이란 이름을 붙인다. (가능하면 PDF가 가장 좋지만, 대중적인 문서 포맷은 다른 포맷도 괜찮음)

8. Logistics

- 본 과제는 혼자 수행한다.
- **본 과제의 지연 제출은 허용되지 않는다.**
- **본 과제에선 문서의 평가 비중이 높으니, 문서를 잘 작성하도록 한다.**
    - 동기화 등의 이슈를 해결하기 위해 어떤 디자인을 이용했는지 설명한다.
- 다른 사람의 과제를 copy할 경우, 개입한 사람 전부 해당 과제에 대해 0점 처리되고, 교수님께 보고되며, **성적 산정에 불이익이 있다**. 또한, copy가 두 차례 이상 적발될 경우 **F 학점이 부여될 수 있다**.

Have fun!

이하윤, 담당 조교
임베디드소프트웨어연구실
성균관대학교