# TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – GJ (박현우)
uc820@naver.com

# 2. 네트워크 프로그래밍 – web_server code – 1

```c
#include "network.h"

#define SMALL_BUF       128

void error_handler(char *msg){
        fputs(msg, stderr);
        fputc('\n',stderr);
        exit(1);
}

void send_error(FILE *fp){

        char protocol[] = "HTTP/1.0 400 Bad Request\r\n";
        char server[] = "Server:Linux Web Server\r\n";
        char cnt_len[] = "Content-length:2048\r\n";
        char cnt_type[] = "Content-type:text/html\r\n\r\n";
        char content[] = "<html><head><title>Network</title></head>"
                "<body><font size=+5><br> 오류 발생! 요청 파일명 및 방식 확인!"
                "</font></body></html>";

        fputs(protocol, fp);
        fputs(server, fp);
        fputs(cnt_len, fp);
        fputs(cnt_type,fp);
        fflush(fp);
}

char *content_type(char *file){

        char extension[SMALL_BUF];
        char file_name[SMALL_BUF];
        strcpy(file_name, file);
        strtok(file_name,".");
        strcpy(extension, strtok(NULL,"."));

        if(!strcmp(extension, "html") || !strcmp(extension, "htm"))
                return "text/html";
        else
                return "text/plain";
}
```

```c
void send_data(FILE *fp, char *ct, char *file_name){
        char protocol[] = "HTTP/1.0 200 OK\r\n";
        char server[] = "Server:Linux Web Server\r\n";
        char cnt_len[] = "Content-length:2048\r\n";
        char cnt_type[SMALL_BUF];
        char buf[BUF_SIZE];
        FILE *send_file;

        sprintf(cnt_type, "Content-type:%s\r\n\r\n", ct);
        send_file = fopen(file_name, "r");

        if(send_file == NULL){
                send_error(fp);
                return;
        }

        fputs(protocol, fp);
        fputs(server, fp);
        fputs(cnt_len, fp);
        fputs(cnt_type, fp);

        while(fgets(buf, BUF_SIZE, send_file) != NULL){
                fputs(buf, fp);
                fflush(fp);
        }
        fflush(fp);
        fclose(fp);
}
```

```c
void *request_handler(void *arg){
        int clnt_sock = *((int *)arg);
        char req_line[SMALL_BUF];
        FILE *clnt_read;
        FILE *clnt_write;

        char method[10];
        char ct[15];
        char file_name[30];

        clnt_read = fdopen(clnt_sock, "r");
        clnt_write = fdopen(dup(clnt_sock), "w");
        fgets(req_line, SMALL_BUF, clnt_read);

        if(strstr(req_line, "HTTP/") == NULL){
                send_error(clnt_write);
                fclose(clnt_read);
                fclose(clnt_write);
                return;
        }

        strcpy(method, strtok(req_line, " /"));
        strcpy(file_name, strtok(NULL, " /"));
        strcpy(ct, content_type(file_name));

        if(strcmp(method, "GET") != 0){
                send_error(clnt_write);
                fclose(clnt_read);
                fclose(clnt_write);
                return;
        }

        fclose(clnt_read);
        send_data(clnt_write, ct, file_name);
}
```

# 2. 네트워크 프로그래밍 – web_server code – 2 ( + html code)

```c
int main(int argc, char **argv){

        int serv_sock, clnt_sock;
        si serv_addr, clnt_addr;
        int clnt_addr_size;
        char buf[BUF_SIZE];
        pthread_t t_id;

        if(argc != 2){
                printf("Use: %s <port>\n", argv[0]);
                exit(1);
        }

        serv_sock = socket(PF_INET, SOCK_STREAM, 0);
        memset(&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        serv_addr.sin_port = htons(atoi(argv[1]));

        if(bind(serv_sock, (sp)&serv_addr, sizeof(serv_addr)) == -1)
                error_handler("bind() error");
        if(listen(serv_sock, 20) == -1)
                error_handler("listen() error");

        for(;;){
                clnt_addr_size = sizeof(clnt_addr);
                clnt_sock = accept(serv_sock, (sp)&clnt_addr, &clnt_addr_size);
                printf("Connection Reques: %s:%d\n)"
                                ,inet_ntoa(clnt_addr.sin_addr), ntohs(clnt_addr.sin_port));

                pthread_create(&t_id, NULL, request_handler, &clnt_sock);
                pthread_detach(t_id);
        }

        close(serv_sock);
        return 0;
}
```

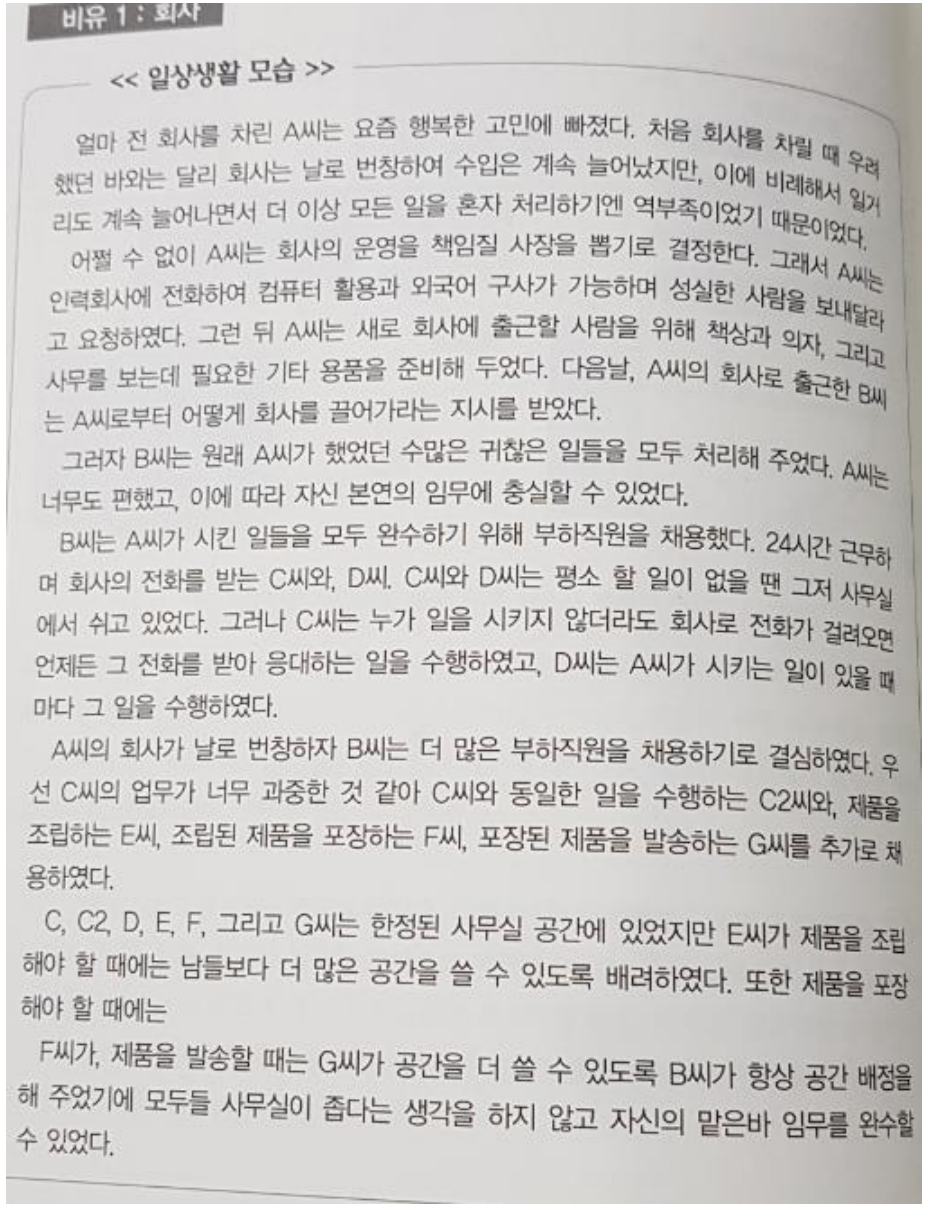
```html
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Heos</title>
</head>
<body>

<h1> 히오스 사랑 = 대성</h1>
<p> 바로 접속하세요~!</p>
<iframe width="854" height="480" src="https://www.youtube.com/embed/oOjELtf5bYc?list=PLC8HEVSBDS71QipzjtQ_z5LsbYBqDse26&amp;ecver=1" frameb


</body>
</html>
```

- 일반적인 server와 거의 방식은 동일하며 html파일을 만들어 둔다.

- IP: port number /(file name).html

Ex) 192.168.0.24: 7777/ firtst.hmtl 을 웹브라우저에 입력하면 웹 서버에 접속할 수 있게 된다.

# 3. 일상생활에 지금까지 배운 내용을 비추어 보기 1



비유 1 : 회사

<< 일상생활 모습 >>

얼마 전 회사를 차린 A씨는 요즘 행복한 고민에 빠졌다. 처음 회사를 차릴 때 우려했던 바와는 달리 회사는 날로 번창하여 수입은 계속 늘어났지만, 이에 비례해서 일거리도 계속 늘어나면서 더 이상 모든 일을 혼자 처리하기엔 역부족이었기 때문이었다.

어쩔 수 없이 A씨는 회사의 운영을 책임질 사장을 뽑기로 결정한다. 그래서 A씨는 인력회사에 전화하여 컴퓨터 활용과 외국어 구사가 가능하며 성실한 사람을 보내달라고 요청하였다. 그런 뒤 A씨는 새로 회사에 출근할 사람을 위해 책상과 의자, 그리고 사무를 보는데 필요한 기타 용품을 준비해 두었다. 다음날, A씨의 회사로 출근한 B씨는 A씨로부터 어떻게 회사를 끌어가라는 지시를 받았다.

그러자 B씨는 원래 A씨가 했었던 수많은 귀찮은 일들을 모두 처리해 주었다. A씨는 너무도 편했고, 이에 따라 자신 본연의 임무에 충실할 수 있었다.

B씨는 A씨가 시킨 일들을 모두 완수하기 위해 부하직원을 채용했다. 24시간 근무하며 회사의 전화를 받는 C씨와, D씨. C씨와 D씨는 평소 할 일이 없을 땐 그저 사무실에서 쉬고 있었다. 그러나 C씨는 누가 일을 시키지 않더라도 회사로 전화가 걸려오면 언제든 그 전화를 받아 응대하는 일을 수행하였고, D씨는 A씨가 시키는 일이 있을 때마다 그 일을 수행하였다.

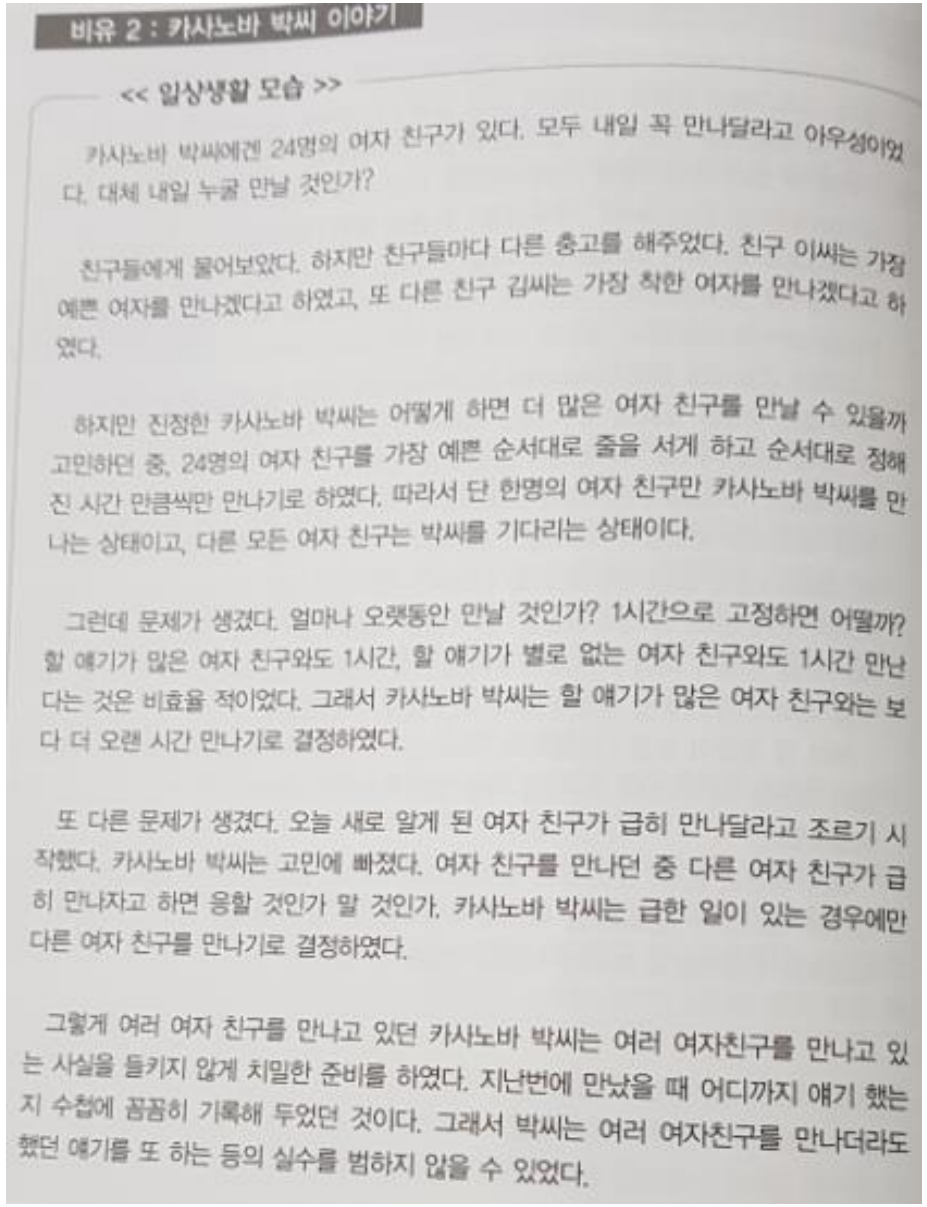
A씨의 회사가 날로 번창하자 B씨는 더 많은 부하직원을 채용하기로 결심하였다. 우선 C씨의 업무가 너무 과중한 것 같아 C씨와 동일한 일을 수행하는 C2씨와, 제품을 조립하는 E씨, 조립된 제품을 포장하는 F씨, 포장된 제품을 발송하는 G씨를 추가로 채용하였다.

C, C2, D, E, F, 그리고 G씨는 한정된 사무실 공간에 있었지만 E씨가 제품을 조립해야 할 때에는 남들보다 더 많은 공간을 쓸 수 있도록 배려하였다. 또한 제품을 포장해야 할 때에는

F씨가, 제품을 발송할 때는 G씨가 공간을 더 쓸 수 있도록 B씨가 항상 공간 배정을 해 주었기에 모두들 사무실이 좁다는 생각을 하지 않고 자신의 맡은바 임무를 완수할 수 있었다.

- A씨 = process
- B씨 = A의 자식 process (fork)
- B씨 -> C씨와 D씨 고용 ( Thread 2개 생성 )
- C씨와 D씨의 역할 분담 ( 각각 handler 함수 구현 )
- C씨는 daemon으로 만들어 백그라운드에서 계속 수행
- D씨는 A가 signal을 보낼 때마다 해당 manual을 수행
- C씨는 fork로 같은 일하는 자식 프로세서 C2씨 생성
- B씨 -> F씨와 G씨 추가 고용 ( Thread 2개 추가 생성)
- G씨 -> E씨에게 더 많은 메모리 할당
- B씨는 F씨 Thread가 돌아 갈 때마다 G씨에게 더 많은 메모리 할당

# 3. 일상생활에 지금까지 배운 내용을 비추어 보기 2



비유 2 : 카사노바 박씨 이야기

<< 일상생활 모습 >>

카사노바 박씨에겐 24명의 여자 친구가 있다. 모두 내일 꼭 만나달라고 아우성이었다. 대체 내일 누굴 만날 것인가?

친구들에게 물어보았다. 하지만 친구들마다 다른 충고를 해주었다. 친구 이씨는 가장 예쁜 여자를 만나겠다고 하였고, 또 다른 친구 김씨는 가장 착한 여자를 만나겠다고 하였다.

하지만 진정한 카사노바 박씨는 어떻게 하면 더 많은 여자 친구를 만날 수 있을까 고민하던 중, 24명의 여자 친구를 가장 예쁜 순서대로 줄을 서게 하고 순서대로 정해진 시간 만큼씩만 만나기로 하였다. 따라서 단 한명의 여자 친구만 카사노바 박씨를 만나는 상태이고, 다른 모든 여자 친구는 박씨를 기다리는 상태이다.

그런데 문제가 생겼다. 얼마나 오랫동안 만날 것인가? 1시간으로 고정하면 어떨까? 할 얘기가 많은 여자 친구와도 1시간, 할 얘기가 별로 없는 여자 친구와도 1시간 만난다는 것은 비효율 적이었다. 그래서 카사노바 박씨는 할 얘기가 많은 여자 친구와는 보다 더 오랜 시간 만나기로 결정하였다.

또 다른 문제가 생겼다. 오늘 새로 알게 된 여자 친구가 급히 만나달라고 조르기 시작했다. 카사노바 박씨는 고민에 빠졌다. 여자 친구를 만나던 중 다른 여자 친구가 급히 만나자고 하면 응할 것인가 말 것인가. 카사노바 박씨는 급한 일이 있는 경우에만 다른 여자 친구를 만나기로 결정하였다.

그렇게 여러 여자 친구를 만나고 있던 카사노바 박씨는 여러 여자친구를 만나고 있는 사실을 들키지 않게 치밀한 준비를 하였다. 지난번에 만났을 때 어디까지 얘기 했는지 수첩에 꼼꼼히 기록해 두었던 것이다. 그래서 박씨는 여러 여자친구를 만나더라도 했던 얘기를 또 하는 등의 실수를 범하지 않을 수 있었다.

- 카사노바의 줄 세워서 여자 만나기
-> RR 방식 알고리즘

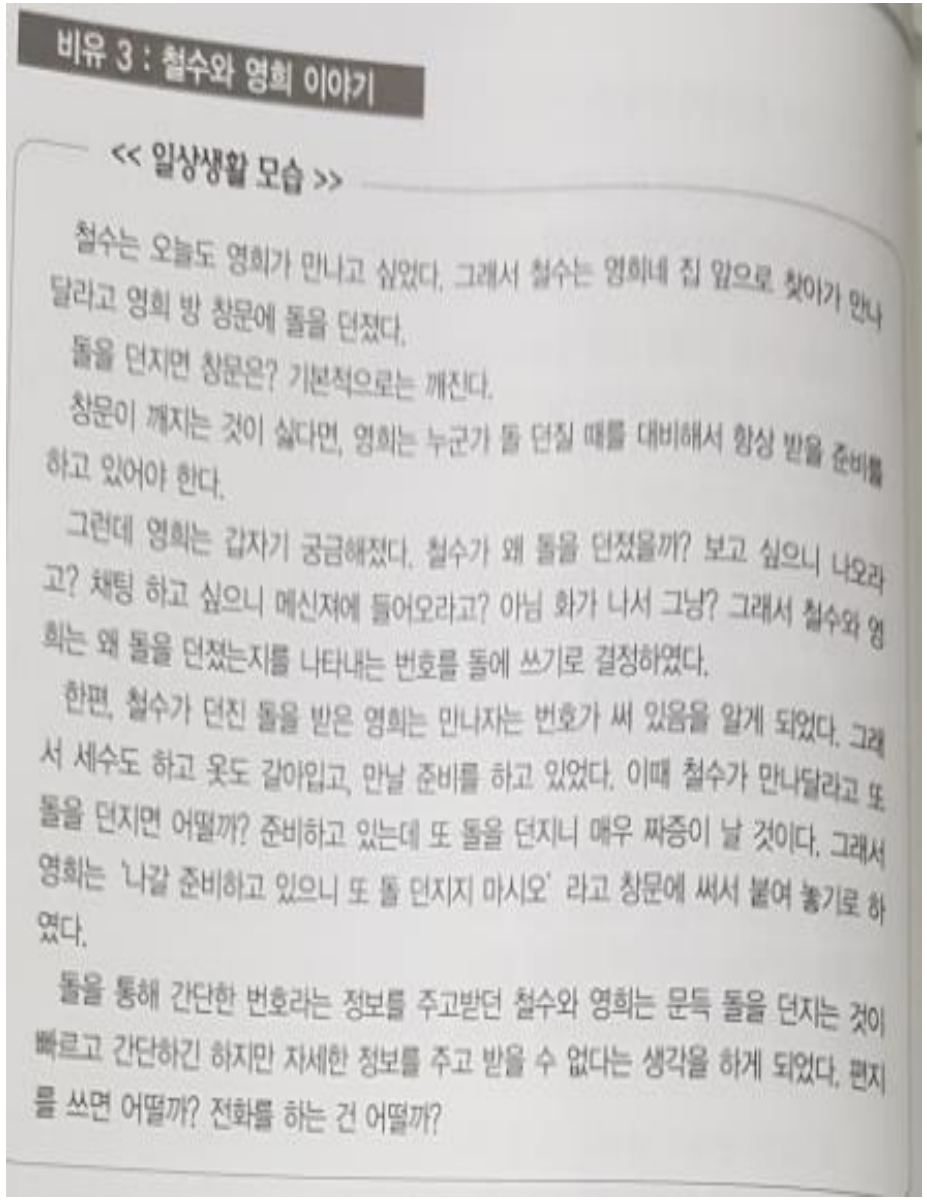
- 카사노바 한 명만 만나고 있으면 23명은 대기
-> run queue에서 하나 실행 & wait queue에서 23명 대기

- 카사노바 박씨 할 애기가 많은 여자와 더 많은 시간을 보내기로 결정
-> 중요한 process일수록 우선순위를 도입

- 급하게 만나 달라는 여자
-> 우선 순위 높은 프로세스의 요청을 받아 급하게 run queue로 처리

- 했던 얘기를 또 하는 실수
-> context switching을 하면서 process마다 데이터가 꼬이지 않게 해야 함. lock을 도입해서 critical section을 보호해야 함.

# 3. 일상생활에 지금까지 배운 내용을 비추어 보기 3



비유 3 : 철수와 영희 이야기

<< 일상생활 모습 >>

철수는 오늘도 영희가 만나고 싶었다. 그래서 철수는 영희네 집 앞으로 찾아가 만나달라고 영희 방 창문에 돌을 던졌다.

돌을 던지면 창문은? 기본적으로는 깨진다.

창문이 깨지는 것이 싫다면, 영희는 누군가 돌 던질 때를 대비해서 항상 밭을 준비를 하고 있어야 한다.

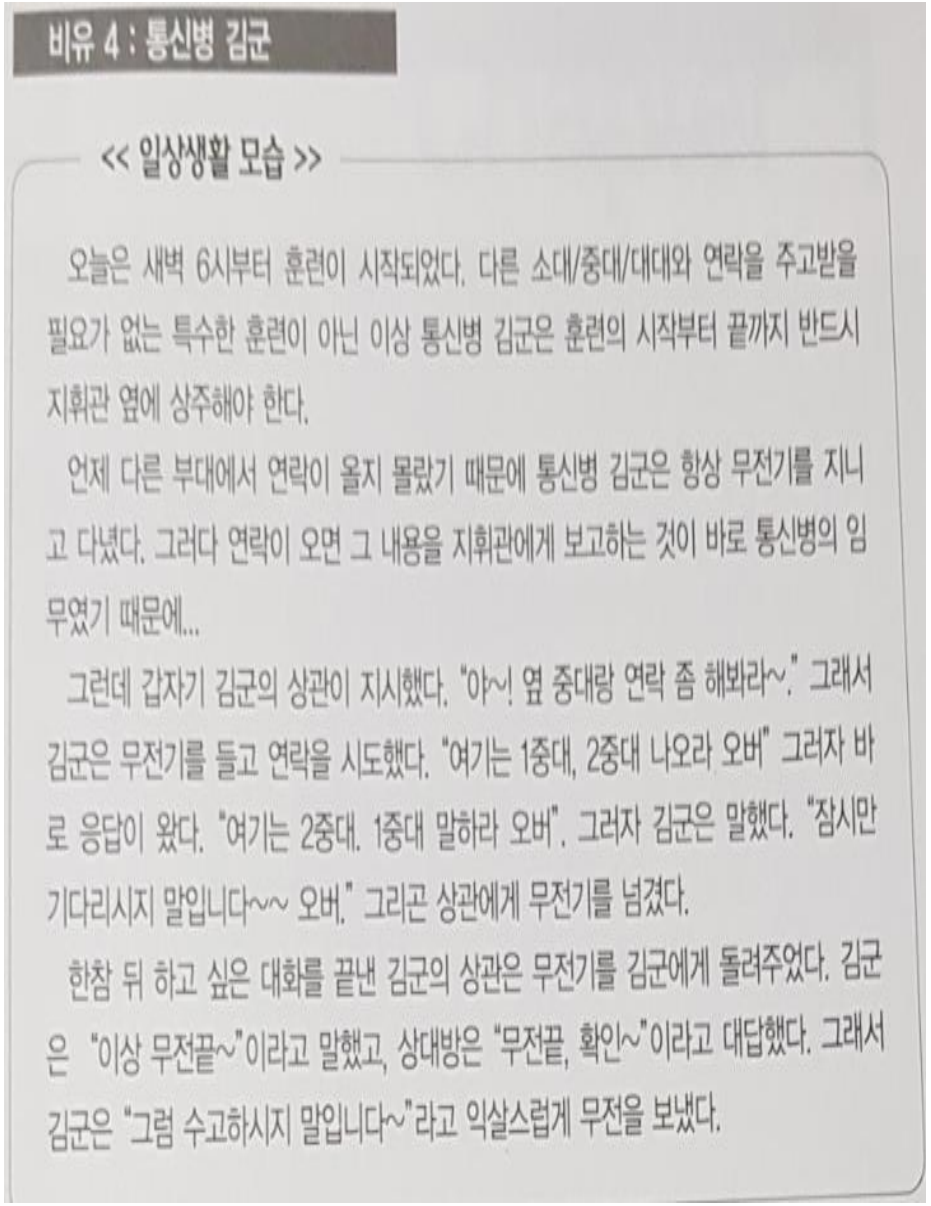
그런데 영희는 갑자기 궁금해졌다. 철수가 왜 돌을 던졌을까? 보고 싶으니 나오라고? 채팅 하고 싶으니 메신저에 들어오라고? 아님 화가 나서 그냥? 그래서 철수와 영희는 왜 돌을 던졌는지를 나타내는 번호를 돌에 쓰기로 결정하였다.

한편, 철수가 던진 돌을 받은 영희는 만나자는 번호가 써 있음을 알게 되었다. 그래서 세수도 하고 옷도 갈아입고, 만날 준비를 하고 있었다. 이때 철수가 만나달라고 또 돌을 던지면 어떨까? 준비하고 있는데 또 돌을 던지니 매우 짜증이 날 것이다. 그래서 영희는 '나갈 준비하고 있으니 또 돌 던지지 마시오' 라고 창문에 써서 붙여 놓기로 하였다.

돌을 통해 간단한 번호라는 정보를 주고받던 철수와 영희는 문득 돌을 던지는 것이 빠르고 간단하긴 하지만 자세한 정보를 주고 받을 수 없다는 생각을 하게 되었다. 편지를 쓰면 어떨까? 전화를 하는 건 어떨까?

- 누군가 돌을 던지면 창문이 깨지니 항상 대비를 해야 한다.
-> 예외 상황 (Signal)이 발생하여 미리 manual을 준비해 놓고 실행 시켜야 한다.

# 3. 일상생활에 지금까지 배운 내용을 비추어 보기 4



**비유 4 : 통신병 김군**

<< 일상생활 모습 >>

오늘은 새벽 6시부터 훈련이 시작되었다. 다른 소대/중대/대대와 연락을 주고받을 필요가 없는 특수한 훈련이 아닌 이상 통신병 김군은 훈련의 시작부터 끝까지 반드시 지휘관 옆에 상주해야 한다.

언제 다른 부대에서 연락이 올지 몰랐기 때문에 통신병 김군은 항상 무전기를 지니고 다녔다. 그러다 연락이 오면 그 내용을 지휘관에게 보고하는 것이 바로 통신병의 임무였기 때문에...

그런데 갑자기 김군의 상관이 지시했다. "야~! 옆 중대랑 연락 좀 해봐라~." 그래서 김군은 무전기를 들고 연락을 시도했다. "여기는 1중대, 2중대 나오라 오버" 그러자 바로 응답이 왔다. "여기는 2중대. 1중대 말하라 오버". 그러자 김군은 말했다. "잠시만 기다리시지 말입니다~~ 오버." 그리곤 상관에게 무전기를 넘겼다.

한참 뒤 하고 싶은 대화를 끝낸 김군의 상관은 무전기를 김군에게 돌려주었다. 김군은 "이상 무전끝~"이라고 말했고, 상대방은 "무전끝, 확인~"이라고 대답했다. 그래서 김군은 "그럼 수고하시지 말입니다~"라고 익살스럽게 무전을 보냈다.

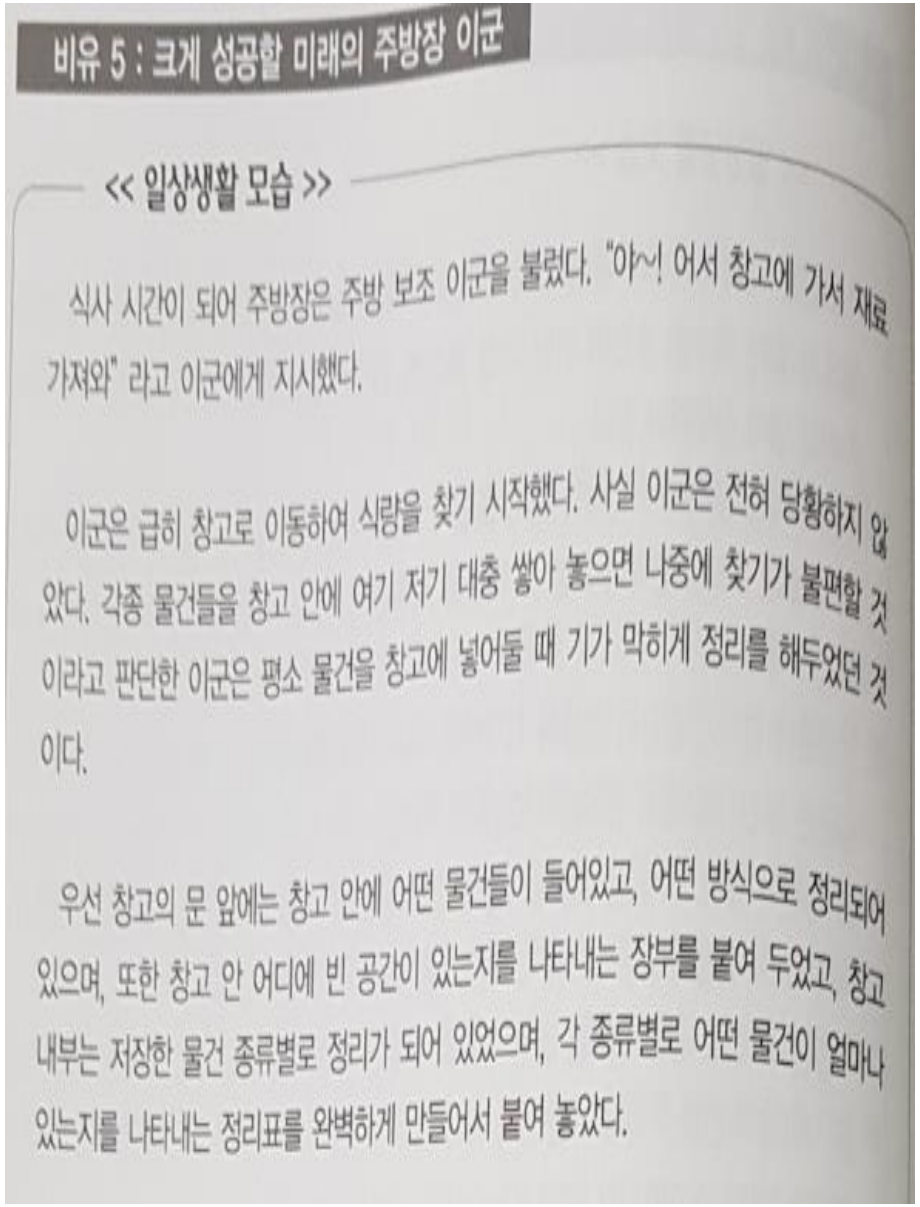
- 통신병 김군은 훈련의 시작부터 끝까지 지휘관 옆에 상주
-> server에서 listen으로 client로부터 정보가 전달되는지 계속 기다려야 한다.

- 연락이 오면 내용을 지휘관에게 보고하는 것
-> accept으로 client에서 요청이 오면 read를 한다.

- 상관의 지시 '연락해봐라'
->write로 메시지를 client들에게 전달한다.

- 무전이 끝남
->close로 client와 연결을 끊는다.

# 3. 일상생활에 지금까지 배운 내용을 비추어 보기 5



비유 5 : 크게 성공할 미래의 주방장 이군

<< 일상생활 모습 >>

식사 시간이 되어 주방장은 주방 보조 이군을 불렀다. "아~! 어서 창고에 가서 재료 가져와" 라고 이군에게 지시했다.

이군은 급히 창고로 이동하여 식량을 찾기 시작했다. 사실 이군은 전혀 당황하지 않았다. 각종 물건들을 창고 안에 여기 저기 대충 쌓아 놓으면 나중에 찾기가 불편할 것이라고 판단한 이군은 평소 물건을 창고에 넣어둘 때 기가 막히게 정리를 해두었던 것이다.

우선 창고의 문 앞에는 창고 안에 어떤 물건들이 들어있고, 어떤 방식으로 정리되어 있으며, 또한 창고 안 어디에 빈 공간이 있는지를 나타내는 장부를 붙여 두었고, 창고 내부는 저장한 물건 종류별로 정리가 되어 있었으며, 각 종류별로 어떤 물건이 얼마나 있는지를 나타내는 정리표를 완벽하게 만들어서 붙여 놓았다.

- 식사 시간이 되어 주방장이 주방보조에게 지시를 함.
-> user가 a.out을 실행하라고 명령한다.

- 주방보조는 정리 해둔 대로 차분하게 찾는다.
-> file_struct, file, path, I-node, super-block 순으로 차분하게 내려간다.

- 종류별로 장부를 만들어 완벽하게 정리표를 만들어 붙여둠.
Super-block에 관련된 정보가 잘 들어가 있다.