TI DSP, MCU 및 Xilinx Zynq FPGA 프로그램 전문가 과정

Innova Lee(이상훈) 강사
gcccompil3r@gmail.com

은태영 학생
zero_bird@naver.com

함수 파일

```
#include "init_sock.h“
// 헤더 파일 init_sock.h 을 참조하여, 평소에 사용하던 err_handler 함수를 다른 파일에서 만든다.
void err_handler(char *msg) {
        fputs(msg, stderr);
        fputc('₩n', stderr);
        exit(1);
}
// sock 을 기본 형식으로 생성한다.
int init_sock(void) {
        int sock;

        sock = socket(PF_INET, SOCK_STREAM, 0);
        if(sock == -1)
                err_handler("socket() error!");
        return sock;
}
```

함수 파일

```
// serv = 0, clnt = 1
// 소캣 주소를 원하는 형식으로 셋팅한다. opt 를 통하여 서버와 클라이언트 모두 사용 가능하다.
void init_sock_addr(si *serv_addr, int size, char **argv, int opt)
{
        memset(serv_addr, 0, size);
        serv_addr->sin_family = AF_INET;
        // 공통 사항으로 memset 과 sin_family 를 처리하고, opt 값을 참조하여 셋팅을 마무리한다.
        if(opt) {
                serv_addr->sin_addr.s_addr = inet_addr(argv[1]);
                serv_addr->sin_port = htons(atoi(argv[2]));
        } else {
                serv_addr->sin_addr.s_addr = htonl(INADDR_ANY);
                serv_addr->sin_port = htons(atoi(argv[1]));
        }
}
```

함수 파일

```
// server 에서만 사용하는 bind 와 listen 을 기본 셋팅한다. 이 부분만 따로 셋팅하는 이유는 hole punching 등 셋팅만 바꾸는 경우가 존재하기 때문이다.
void post_sock(int serv_sock, si *serv_addr, int size) {
        if(bind(serv_sock, (sp)serv_addr, size) == -1)
                err_handler("bind() error!");

        if(listen(serv_sock, 5) == -1)
                err_handler("listen() error!");
}
```

헤더 파일

```
#ifndef __INIT_SOCK_H__                 // ifndef / define / endif 는 헤더 파일의 형식이다.
#define __INIT_SOCK_H__

#include <stdio.h>                      // 공통으로 사용되는 헤더 파일을 관리한다.
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>

typedef struct sockaddr_insi;
typedef struct sockaddr *sp;

void err_handler(char *msg);            // 사용할 함수들의 프로토 타입을 작성한다.
int init_sock(void);
void init_sock_addr(si *, int, char **, int);
void post_sock(int, si *, int);

#endif
```

serv

```
#include "init_sock.h"
// 기존 스크립트 보다 더 빠르고 알아보기 쉽게 구현이 가능하다.
int main(int argc, char **argv) {
        int serv_sock, clnt_sock;
        si serv_addr, clnt_addr;
        socklen_t clnt_addr_size;
        char msg[] = "Hello Network Programming";

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

        serv_sock = init_sock();
        init_sock_addr(&serv_addr, sizeof(serv_addr), argv, 0);
        post_sock(serv_sock, &serv_addr, sizeof(serv_addr));
```

serv

```
        clnt_addr_size = sizeof(clnt_addr);
        clnt_sock = accept(serv_sock, (sp)&clnt_addr, &clnt_addr_size);

        if(clnt_sock == -1)
                err_handler("accept() error");

        write(clnt_sock, msg, sizeof(msg));

        close(clnt_sock);
        close(serv_sock);

        return 0;
}
```

clnt

```
#include "init_sock.h"
// 기존 스크립트 보다 더 빠르고 알아보기 쉽게 구현이 가능하다.
int main(int argc, char **argv) {
        int sock, len;
        si serv_addr;
        char msg[32] = {0};

        if(argc != 3) {
                printf("use: %s <ip> <port>₩n", argv[0]);
        exit(1);
        }

        sock = init_sock();
        init_sock_addr(&serv_addr, sizeof(serv_addr), argv, 1);

        if(connect(sock, (sp)&serv_addr, sizeof(serv_addr)) == -1)
                err_handler("connect() error!");
```

clnt

```
        len = read(sock, msg, sizeof(msg) - 1);

        if(len == -1)
                err_handler("read() error!");

        printf("msg from serv: %s₩n", msg);
        close(sock);

        return 0;
}
```

web_serv

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <pthread.h>

#define BUF_SIZE1024
#define SMALL_BUF100

typedef struct sockaddr_insi;
typedef struct sockaddr *sp;

void error_handling(char *msg) {
                fputs(msg, stderr);
                fputc('₩n', stderr);
                exit(1);
}
```

web_serv

```
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);
}
```

web_serv

```
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";
}
```

web_serv

```
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 ;
        }
```

web_serv

```
		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);
}
```

web_serv

```
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;
        }
```

web_serv

```
        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);
}
```

web_serv

```
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 <prot>₩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]));
```

web_serv

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

        for(;;){
                clnt_addr_size = sizeof(clnt_addr);
                clnt_sock = accept(serv_sock, (sp)&clnt_addr, &clnt_addr_size);
                printf("Connection Request: %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;
}
```