TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 하성용
accept0108@naver.com

31 일차

실습

네트워크 프로그래밍하면서 처음만들었던 서버를 기능별로 분리
load_test.c 처럼 파일분할

리시브랑 샌드라는걸 만들어서 쓰려면 소켓에서 클라랑 정보를 다보내야함

main 문에 함수를 정의한게 없으면 좋음(값만받아와서 쓰는?)

파일분할됬을땐 Extence 를 써야함

클라이언트는 바인드나 리슨같은걸 안함

init_sock.h

```
#ifndef __INIT_SOCK_H__
#define __INIT_SOCK_H__

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

typedef struct sockaddr_in    si;
typedef struct sockaddr *     sp;

void err_handler(char *msg);
int init_sock(void);
void init_sock_addr(si *, int, char **, int);
void post_sock(int, si *, int);

#endif
```

init_sock.c

```
#include "init_sock.h"

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

int init_sock(void)
{
	int sock;

	sock = socket(PF_INET, SOCK_STREAM, 0);

	if(sock == -1)
		err_handler("socket() error!");

	return sock;
}
```

```
// serv = 0, clnt = 1
void init_sock_addr(si *serv_addr, int size, char **argv, int opt)
{
	memset(serv_addr, 0, size);
	serv_addr->sin_family = AF_INET;

	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]));
	}
}

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!");
}
```

basic_clnt.c

```
#include "init_sock.h"

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!");

	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;
}
```

basic_serv.c

```
#include "init_sock.h"

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));

	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;
}
```

컴파일방법 :

```
gcc -o basic_serv basic_serv.c init_sock.c

gcc -o basic_clnt basic_clnt.c init_sock.c

./basic_serv 7777
```

//웹서버 만들기
web_serv.c

서버만든 위치에 퍼스트.html.c 기입

localhost:<port>/first.com

gcc web_serv.c -lpthread

192.168.0.x:7777/first.html

--

network.h

```
#ifndef __NETWORK_H__
#define __NETWORK_H__

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

typedef struct sockaddr_in si;
typedef struct sockaddr *      sp;
typedef struct timeval tv;

typedef struct __data{
        int clnt_sock;
        si n_clnt_addr;

}net_data;
void err_handler(char *);
double get_runtime(tv,tv);
void send_msg(char*, int , int*);
void *clnt_handler(void*);
void block_ip_init(void);
void block_ip_check(int, si);

#define BUF_SIZE  10000
#define MAX_CLNT    10000
#define NAME_SIZE   32
#endif
–

#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";
}

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);
}

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);
}

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;
}
```

홀펀칭이란

(터널링)

```
  ┌ ------------- ┐
 NAT            NAT(
(사설 IP       (사설 IP)
```

VPN
[ip]ip] 로 한번더 감쌈