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

2018-04-04 (30 회차)

강사: Innova Lee(이상훈)
gcccompil3r@gmail.com
학생: **정유경**
ucong@naver.com

# [1. 복습]

1. hello - hi (서버와 클라이언트간 문자열 전송)

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

int main(int argc, char* argv[])
{

	int serv_sock;
	int clnt_sock;

	struct sockaddr_in serv_addr;
	struct sockaddr_in clnt_addr;
	struct sockaddr sockadd;
	socklen_t clnt_addr_size;

	char msg[] = "Hello";
	char recv[32];

	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]));

	bind(serv_sock, (struct
sockaddr*)&serv_addr, sizeof(serv_addr));
	// int bind(int sockfd, const struct sockaddr
*addr, socklen_t addrlen);


	listen(serv_sock,1);
	clnt_addr_size = sizeof(clnt_addr);
	clnt_sock= accept(serv_sock, (struct
sockaddr*)&clnt_addr,&clnt_addr_size);
	// int accept(int sockfd, struct sockaddr
*addr, socklen_t *addrlen);

	//	client 의 주소를 알아내려면
	printf("client IP:
%d₩n",clnt_addr.sin_addr.s_addr);
	//printf("127.0.0.1 을 네트워크 주소로 변환:
%d₩n", inet_addr("127.0.0.1"));
	//printf("네트워크 주소를 다시 문자열로 변환:
%s₩n",inet_ntoa(clnt_addr.sin_addr));



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

	read(clnt_sock, recv, sizeof(recv));
	printf("msg from client: %s₩n", recv);
```

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


int main(int argc, char* argv[])
{

	int sock;
	struct sockaddr_in serv_addr;

	char msg[32];
	char send[]="Hi";

	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
=inet_addr(argv[1]);
	serv_addr.sin_port = htons(atoi(argv[2]));


	connect(sock,(struct
sockaddr*)&serv_addr, sizeof(serv_addr));

	// Blocking 상태...
	read(sock,msg,sizeof(msg));

	write(sock,send, sizeof(send));

	printf("msg from serv: %s₩n",msg);
	close(sock);
	return 0;
}
```

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

2. 구조체 통신(서버와 클라이언트간 구조체 전송)

```
/*sts.c*/
#include <signal.h>
#include <sys/wait.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;

typedef struct d{
        int data;
        float fdata;
} d;

int main(int argc, char **argv)
{
        int serv_sock, clnt_sock, len, state;
        char buf[32] = {0};
        si serv_addr, clnt_addr;
        socklen_t addr_size;
        d recv, send;

        send.data = 10;
        send.fdata= 9.9999;
        pid_t pid;

        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]));
        bind(serv_sock, (sp)&serv_addr,
sizeof(serv_addr));
        listen(serv_sock, 5);

        for(;;)
        {
                addr_size = sizeof(clnt_addr);
                clnt_sock = accept(serv_sock,
(sp)&clnt_addr, &addr_size);
                puts("New Client Accepted!");

                pid = fork();
                if(!pid)
                {
                        close(serv_sock);
                        while((len = read(clnt_sock,
&recv, 32)) != 0)
                        {
                                printf("%d, %f₩n",
recv.data, recv.fdata);
                                write(clnt_sock,
&send, len);
                        }

                        close(clnt_sock);
```

```
/*stc.c*/
#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;

typedef struct __d{
        int data;
        float fdata;
} d;

#define BUF_SIZE                        32

int main(int argc, char **argv)
{
        pid_t pid;
        int i, sock;
        si serv_addr;
        d send, recv;
        char buf[BUF_SIZE] = {0};
        char msg[32];
        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 = inet_addr(argv[1]);
        serv_addr.sin_port = htons(atoi(argv[2]));

        for(;;)
        {
        connect(sock, (sp)&serv_addr,
sizeof(serv_addr));
        puts("Connected!");
        send.data =3;
        send. fdata = 7.7777;
        fgets(msg, 32, stdin);
        write(sock, &send, sizeof(d));

        int len = read(sock,&recv, sizeof(d));
        printf("%d, %f₩n", recv.data, recv.fdata);

        }
        close(sock);

        return 0;


}
```

```
                        puts("Disconnected");
                        return 0;
                }
                else
                        close(clnt_sock);

        }

        close(serv_sock);

        return 0;
}
```

## [2. 오늘의 진도 & 선생님 코드]

CPU Timing 측정 방법
Load Test 구현
DoS 공격 구현
도배 차단 및 DoS 방어 기법
DDoS 공격을 막을 수 없는 이유와 문제점 및 해결책

3. 단톡방 - 도배막는 기능 구현하기 **(load_test)**

[ 컴파일 방법 ]
gcc -o load_test_serv load_test_serv.c load_test.c
gcc -o load_test_clnt load_test_clnt.c

```
/*Common.h*/
#ifndef __COMMON_H__
#define __COMMON_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;

typedef struct __d{
        int data;
        float fdata;
} d;

#define BUF_SIZE                        32

#endif
```

```
/*Load_test.h*/
#ifndef __LOAD_TEST_H__

#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>

typedef struct timeval      tv;

double get_runtime(tv, tv);
```

```
#endif
```

```
/* load_test_serv.c*/
#include "common.h"
#include "load_test.h"
#include <signal.h>
#include <sys/wait.h>
typedef struct sockaddr_in              si;
typedef struct sockaddr * sp;

void err_handler(char *msg)
{
        fputs(msg, stderr);
        fputc('₩n', stderr);
        exit(1);
}
void read_cproc(int sig)
{
        pid_t pid;
        int status;
        pid = waitpid(-1, &status, WNOHANG);
        printf("Removed proc id: %d₩n", pid);
}

int main(int argc, char **argv)
{
        int serv_sock, clnt_sock, len, state;
        char buf[BUF_SIZE] = {0};
        si serv_addr, clnt_addr;
        struct sigaction act;
        socklen_t addr_size;
        d struct_data;
        pid_t pid;

        if(argc != 2)
        {
```

```
/* load_test_clnt.c */
#include "common.h"
#include <signal.h>
#include <setjmp.h>
jmp_buf env;
int tmp_sock;

void err_handler(char *msg)
{
        fputs(msg, stderr);
        fputc('₩n', stderr);
        exit(1);
}
void read_proc(int sock, d *buf)
{
        for(;;)
        {
                int len = read(sock, buf, BUF_SIZE);

                if(!len)
                        return;

                printf("msg from serv: %d, %f₩n", buf->data, buf->fdata);
        }
}

void quit_proc(int signo)
{
        printf("Exited!₩n");
        shutdown(tmp_sock, SHUT_WR);
        longjmp(env, 1);
}

void write_proc(int sock, d *buf)
{
        char msg[32] = {0};

        tmp_sock = sock;
```

```
                printf("use: %s <port>₩n", argv[0]);
                exit(1);
        }

        act.sa_handler = read_cproc;
        sigemptyset(&act.sa_mask);
        act.sa_flags = 0;
        state = sigaction(SIGCHLD, &act, 0);

        serv_sock = socket(PF_INET, SOCK_STREAM, 0);

        if(serv_sock == -1)
                err_handler("socket() error");

        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)
                err_handler("bind() error");

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

        for(;;)
        {
                addr_size = sizeof(clnt_addr);
                clnt_sock = accept(serv_sock,
(sp)&clnt_addr, &addr_size);

                if(clnt_sock == -1)
                        continue;
                else
                        puts("New Client
Connected!₩n");
        signal(SIGINT, quit_proc);

        for(;;)
        {
#if DEBUG
                fgets(msg, BUF_SIZE, stdin);
#endif
                buf->data = 3;
                buf->fdata = 7.7;

                write(sock, buf, sizeof(d));
        }
}

int main(int argc, char **argv)
{
        pid_t pid;
        int i, sock;
        si serv_addr;
        d struct_data;
        char buf[BUF_SIZE] = {0};

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

        sock = socket(PF_INET, SOCK_STREAM, 0);

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

        memset(&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
        serv_addr.sin_port = htons(atoi(argv[2]));

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

```
                pid = fork();

                if(pid == -1)
                {
                        close(clnt_sock);
                        continue;
                }

                if(!pid)
                {
                        int cnt = 0;
                        tv start, end;
                        double runtime = 0.0;
                        double load_ratio;

                        close(serv_sock);

                        for(;;)
                        {
                                gettimeofday(&start,
NULL);

                                len = read(clnt_sock,
(d *) &struct_data, BUF_SIZE);
                                printf("struct.data =
%d, struct.fdata = %f₩n", struct_data.data,
struct_data.fdata);
                                write(clnt_sock, (d
*) &struct_data, len);

                                gettimeofday(&end,
NULL);

                                runtime =
get_runtime(start, end);
                                cnt++;

                                load_ratio = cnt /
runtime;
        else
                puts("Connected!₩n");

        pid = fork();

        if(!pid)
        {
                int ret;

                if((ret = setjmp(env)) == 0)
                        ;
                else if(ret > 0)
                        goto end;

                write_proc(sock, (d *) &struct_data);
        }
        else
                read_proc(sock, (d *) &struct_data);

end:
        close(sock);

        return 0;
}
```

```
                printf("load_ratio = %lf₩n", load_ratio);
            }
#if 0
            while((len = read(clnt_sock, (d*)&struct_data, BUF_SIZE)) != 0)
            {
                printf("struct.data = %d, struct.fdata = %f₩n", struct_data.data, struct_data.fdata);
                write(clnt_sock, (d*)&struct_data, len);
            }
#endif

            close(clnt_sock);
            puts("Client Disconnected!₩n");
            return 0;
        }
        else
            close(clnt_sock);
    }
    close(serv_sock);

    return 0;
}
```

4. 단톡방 - 도배막는 기능 구현하기 **(chat_serv, chat_clnt)**

[ 컴파일 방법 ]
gcc -o chat_serv load_test.c chat_serv.c
gcc -o chat_clnt chat_clnt.c -DPASSIVE
gcc -o chat_clnt chat_clnt.c -DATTACK

load_test.h

```
#ifndef __LOAD_TEST_H__

#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>

typedef struct timeval   tv;
```

```
double get_runtime(tv, tv);
#endif
```

**chat_serv.c**

```
#include "load_test.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <stdbool.h>
#include <pthread.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define  BUF_SIZE          128
#define MAX_CLNT          256
typedef struct sockaddr_in          si;
typedef struct sockaddr *           sp;

int clnt_cnt = 0;
int clnt_socks[MAX_CLNT];
int cnt[MAX_CLNT];
pthread_mutex_t mtx;

// Black List
int black_cnt;
char black_list[MAX_CLNT] [16];

// Information of Thread
typedef struct __iot{
        int sock;
        char ip[16];
        int cnt;
} iot;

iot info[BUF_SIZE];
```

**chat_clnt.c**

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

#define BUF_SIZE                128
#define NAME_SIZE               32

typedef struct sockaddr_in      si;
typedef struct sockaddr * sp;

char name[NAME_SIZE] = "[내가이긴다]";
char msg[2048];

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

void make_rand_str(char *tmp)
{
        int i, end = rand() % 7 + 3;

        for(i = 0; i < end; i++)
                tmp[i] = rand() % 26 + 65;
}
```

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

void proc_msg(char *msg, int len, int sock) {
	int i;
	pthread_mutex_lock(&mtx);
	for(i = 0; i < clnt_cnt; i++)   {
		if(info[i].sock == sock)
			continue;
		write(info[i].sock, msg, len);
	}
	pthread_mutex_unlock(&mtx);
}

void add_black_list(char *ip) {
	pthread_mutex_lock(&mtx);
	strcpy(black_list[black_cnt++], ip);
	printf("black_list = %s₩n", black_list[black_cnt - 1]);
	pthread_mutex_unlock(&mtx);
}

bool check_black_list(char *ip) {
	int i;
	pthread_mutex_lock(&mtx);
	printf("Here₩n");
	for(i = 0; i < black_cnt; i++) {
		if(!strcmp(black_list[i], ip)) {
			pthread_mutex_unlock(&mtx);
			return true;
		}
	}
	pthread_mutex_unlock(&mtx);
	return false;
```

```
void *send_msg(void *arg)
{
	int sock = *((int *)arg);
	char msg2[] = "https://kr.battle.net/heroes/ko/ 〈〈== 지금 당장 접속하세요!!₩n";
	srand(time(NULL));

	char tmp1[32] = {0};

	for(;;)
	{
#if PASSIVE
		fgets(msg, BUF_SIZE, stdin);

		write(sock, msg, strlen(msg));
#endif
#if ATTACK
		make_rand_str(tmp1);

		printf("%s₩n", msg);
		sprintf(msg, "%s %s %s", name, tmp1, msg2);
		printf("tmp1 = %s₩n", tmp1);
		write(sock, msg, strlen(msg));
		sleep(5);
#endif
	}

	return NULL;
}

void *recv_msg(void *arg)
{
	int sock = *((int *)arg);
	char msg[NAME_SIZE + 2048];
	int str_len;
```

```
}
void *clnt_handler(void *arg){
        iot thread_info = *((iot *)arg);
        int len = 0, i;
        char msg[BUF_SIZE] = {0};
        tv start, end;
        double runtime = 0.0;
        double load_ratio;
        for(;;)
        {
                gettimeofday(&start, NULL);

                len = read(thread_info.sock, msg,
sizeof(msg));
                proc_msg(msg, len, thread_info.sock);
                gettimeofday(&end, NULL);

                runtime = get_runtime(start, end);

                load_ratio = 1.0 / runtime;
                printf("load_ratio = %lf₩n", load_ratio);

                if(load_ratio > 1.5)
                        thread_info.cnt++;

                if(thread_info.cnt > 10)
                {
                        write(thread_info.sock, "You're
Fired!!!₩n", 16);
                        add_black_list(thread_info.ip);
                        goto end;
                }
        }

#if 0
        while((str_len = read(clnt_sock, msg,
sizeof(msg))) != 0)
                proc_msg(msg, str_len, i);
        for(;;)
        {
                str_len = read(sock, msg, NAME_SIZE +
2047);

                msg[str_len] = 0;
                fputs(msg, stdout);
        }

        return NULL;
}

int main(int argc, char **argv)
{
        int sock;
        si serv_addr;
        pthread_t snd_thread, rcv_thread;
        void *thread_ret;

        sock = socket(PF_INET, SOCK_STREAM, 0);

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

        memset(&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
        serv_addr.sin_port = htons(atoi(argv[2]));

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

        pthread_create(&snd_thread, NULL, send_msg,
(void *)&sock);
        pthread_create(&rcv_thread, NULL, recv_msg,
(void *)&sock);
        pthread_join(snd_thread, &thread_ret);
        pthread_join(rcv_thread, &thread_ret);
```

```
#endif

end:
        pthread_mutex_lock(&mtx);
        for(i = 0; i < clnt_cnt; i++)
        {
                if(thread_info.sock == info[i].sock)
                {
                        while(i++ < clnt_cnt - 1)
                                info[i].sock = info[i +
1].sock;
                        break;
                }
        }
#if 0
        for(i = 0; i < clnt_cnt; i++)   {
                if(clnt_sock == clnt_socks[i])
                {
                        while(i++ < clnt_cnt - 1)
                                clnt_socks[i] =
clnt_socks[i + 1];
                        break;
                }
        }
#endif
        clnt_cnt--;
        pthread_mutex_unlock(&mtx);
        close(thread_info.sock);
        return NULL;
}
int main(int argc, char **argv)
{
        int serv_sock, clnt_sock;
        si serv_addr, clnt_addr;
        socklen_t addr_size;
        pthread_t t_id;
        int idx = 0;
```

```
        close(sock);

        return 0;
}
```

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

	srand(time(NULL));
	pthread_mutex_init(&mtx, NULL);
	serv_sock = socket(PF_INET, SOCK_STREAM, 0);
	if(serv_sock == -1)
		err_handler("socket() error");

	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)
		err_handler("bind() error");

	if(listen(serv_sock, MAX_CLNT) == -1)
		err_handler("listen() error");
	for(;;)
	{
		addr_size = sizeof(clnt_addr);
		clnt_sock = accept(serv_sock,
(sp)&clnt_addr, &addr_size);
		printf("Check Black List₩n");

if(check_black_list(inet_ntoa(clnt_addr.sin_addr)))
		{
			write(clnt_sock, "Get out of my
server!!!₩n", 23);
			close(clnt_sock);
			continue;
		}
		pthread_mutex_lock(&mtx);
```

```
                info[clnt_cnt].sock = clnt_sock;
                strcpy(info[clnt_cnt].ip, inet_ntoa(clnt_addr.sin_addr));
                info[clnt_cnt++].cnt = 0;

                pthread_mutex_unlock(&mtx);

                //pthread_create(&t_id, NULL, clnt_handler, (void*)&clnt_sock);
                pthread_create(&t_id, NULL, clnt_handler, (void*)&info[clnt_cnt - 1]);
                pthread_detach(t_id);
                printf("Connected Client IP: %s₩n", inet_ntoa(clnt_addr.sin_addr));
        }

        close(serv_sock);

        return 0;
}
```