TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

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

30 일차

서버와 클라이언트 이해가 부족하여 복습입니다

socket 함수 : 소켓 지정자 socket(인자 도메인, 소켓의 형태, 프로토콜 종류)

inet_addr 함수 : 이진 바이너리 형식의 IP 주소 inet_addr(문자형식의 IP 주소)

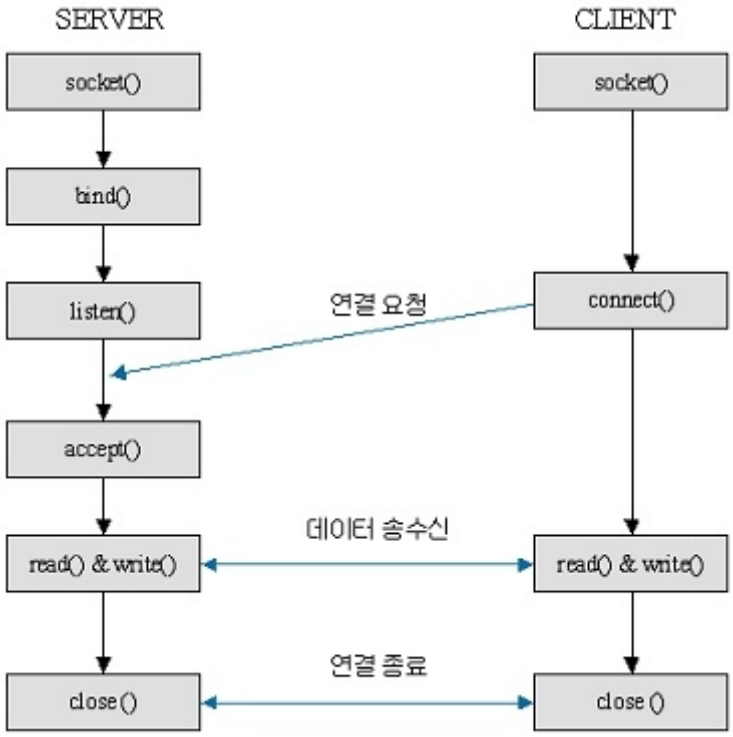
htons 함수 : 네트워크 바이트 오더에 적용된 이진 바이너리 16 비트값 htons(16 비트 변수값)

connect 함수 : 성공 여부 반환 connect(소켓 디스크립션, 서버의 IP 주소와 포트번호, 길이)

read 함수 : 읽은 값의 길이 read(파일 디스크립터, 읽은데이터를 저장할 버퍼, 읽을 데이터 최대길이)
→ 오류 발생시 -1 리턴

close 함수 : 성공 여부 반환 close(파일 디스크립터)
→ 리턴값이 0 이면 성공 -1 이면 실패

INADDR_ANY 는 서버의 IP 주소를 자동으로 찾아서 대입해주는 함수



-DATTACK → 옵션주는법
-DPASSIVE → 옵션주는법

gcc -o load_test_serv load_test_serv.c load_test.c
gcc -o load_test_clnt load_test_clnt.c

gcc -o chat_serv load_test.c chat_serv.c
gcc -o chat_clnt chat_clnt.c -DAPSSIVE
gcc -o chat_clnt chat_clnt.c -DATTACK

//fork 기반 채팅 도배 차단

load_test.h // 사용자 지정 헤더파일

```
#ifndef __LOAD_TEST_H__

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

typedef struct timeval          tv;

double get_runtime(tv, tv);

#endif
```

common.h // 사용자 지정 헤더파일

```
#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_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;
	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));
#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	}
}

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");
	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;
}
```

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)
        {
                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");

		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); //구하는 함
수=get_runtime
				cnt++; //++시키고

				load_ratio = cnt / runtime; //횟수/시간
				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;
}
```

//thread 기반 채팅 도배 차단

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 *send_msg(void *arg)
{
        int sock = *((int *)arg);
        char msg2[] = "https://kr.battle.net/heroes/ko/  <<== 지금 당장 접속하세
요!!\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); //랜덤으로 만든 문자열이 tmp1 에 저장됨

			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;

		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);

	close(sock);

	return 0;
}
```

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];

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; //자기에 해당하는건 제낄것(에코)
                        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]); //블랙리스트 값추가
                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 *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(clnt_sock, msg, sizeof(msg));
                        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); //load_ratio 값 출력
```

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

                if(thread_info.cnt > 10) //10 회이상의 ratio 값이 나올만큼의 작업을 했다면
                {
                        write(thread_info.sock, "You're Fired!!!\n", 16);
                        add_black_list(thread_info.ip); //블랙리스트
                        goto end;
                }
        }

#if 0
        while((str_len = read(clnt_sock, msg, sizeof(msg))) != 0)
                proc_msg(msg, str_len, i);
#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;

        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;
		//구조체 안에있는 ip 값에 현재있는 ip 값을 넣는다 = 밴하려고
		strcpy(info[clnt_cnt].ip, inet_ntoa(clnt_addr.sin_addr));
		//해당 Ip 가 몇번이나 했는지
		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]); //배열 낱개 한개만 전달
		pthread_detach(t_id);
		printf("Connected Client IP: %s\n", inet_ntoa(clnt_addr.sin_addr));
	}

	close(serv_sock);

	return 0;
}
```