TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/4/4
수업일수	30 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

1. Hi ,Hello 주고 받기

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

typedef struct sockaddr_in      si;
typedef struct sockaddr*  sap;

#define BUF_SIZE        32

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

int main(int argc, char **argv)
{
        int serv_sock, clnt_sock;
        int addr_size;
        si serv_addr,clnt_addr;
        char buf[BUF_SIZE]={0};
        char *msg = "Hello";

        if(argc != 2)
        {
```

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

	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, (sap)&serv_addr, sizeof(serv_addr)) == -1)
		err_handler("bind() error");

	if(listen(serv_sock,2) == -1)
		err_handler("listen() error");

	addr_size = sizeof(clnt_addr);
	clnt_sock = accept(serv_sock, (sap)&clnt_addr, &addr_size);
	if(clnt_sock == -1)
		err_handler("accept() error");

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

	read(clnt_sock, buf, BUF_SIZE);
	printf("clnt from msg : %s₩n", buf);

	close(clnt_sock);
```

```
        close(serv_sock);

        return 0;
}
```

-client

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

typedef struct sockaddr_in      si;
typedef struct sockaddr*  sap;

#define BUF_SIZE        32

void err_handler(char *msg)
{

        fputs(msg, stderr);
        fputc('₩n', stderr);
        exit(1);

}

int main(int argc, char **argv)
{
        char buf[32]={0};
```

```
int clnt_sock;
si serv_addr;
char *msg = "Hi";

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

clnt_sock = socket(PF_INET, SOCK_STREAM,0);

if(clnt_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(clnt_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("connect() error");


read(clnt_sock, buf, BUF_SIZE);
printf("serv from msg : %s₩n", buf);

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

```
        close(clnt_sock);

        return 0;
}
```

```
xeno@xeno-NH:~/proj/0404$ ./clnt 127.0.0.1 7778
serv from msg : Hello

xeno@xeno-NH:~/proj/0404$ ./serv 7778
clnt from msg : Hi
```

2.채팅 도배 차단 분석

```
#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];//쓰레드에 대한 정보 구조체

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

void proc_msg(char *msg, int len, int sock)
{
    int i;

    pthread_mutex_lock(&mtx); //쓰레드 잠금

    for(i = 0; i < clnt_cnt; i++)
```

```
    {
        if(info[i].sock == sock)//자기가 쓴 것만 넘긴다, 에코 x
            continue;
        write(info[i].sock, msg, len); //info 에 메시지 입력
    }

    pthread_mutex_unlock(&mtx); //쓰레드 잠금 해제
}

void add_black_list(char *ip) //블랙리스트 추가
{
    pthread_mutex_lock(&mtx); //쓰레드 잠금
    strcpy(black_list[black_cnt++], ip);//ip 값 strcpy,블랙리스트에 한명 더 추가
    printf("black_list = %s₩n", black_list[black_cnt - 1]);
    pthread_mutex_unlock(&mtx); //쓰레드 해제
}

bool check_black_list(char *ip) //ip 를 받음
{
    int i;

    pthread_mutex_lock(&mtx); //쓰레드 잠금

    printf("Here₩n");

    for(i = 0; i < black_cnt; i++) // black_cnt 초기값이 뭐임??
    {
        if(!strcmp(black_list[i], ip))//일치하면
        {
            pthread_mutex_unlock(&mtx); //쓰레드 잠금 해제
            return true; //1 반환
        }
```

```
    }

    pthread_mutex_unlock(&mtx); //쓰레드 잠금 해제

    return false;//일치하지 않으면 0 반환
}

void *clnt_handler(void *arg)
{
    iot thread_info = *((iot *)arg);//info 전달(낱개 하나)
    int len = 0, i;
    char msg[BUF_SIZE] = {0};

    tv start, end;
    double runtime = 0.0;
    double load_ratio;

    for(;;)
    {
        gettimeofday(&start, NULL); //시작시간 구함
        //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); //끝나는 시간 구한다

        runtime = get_runtime(start, end);//걸린시간 구함

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

        if(load_ratio > 1.5)//비율이 1.5 보다 크면 카운트
            thread_info.cnt++;
```

```
        if(thread_info.cnt > 10)//10 번이상하면 도배니까 강퇴
        {
            write(thread_info.sock, "You're Fired!!!₩n", 16);
            add_black_list(thread_info.ip);//블랙리스트 추가
            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) // 인자 2 개 아니면 출력
    {
        printf("Usage: %s <port>₩n", argv[0]);
        exit(1);
    }

    srand(time(NULL)); //랜덤 값 생성을 위함

    pthread_mutex_init(&mtx, NULL);//mutex 초기화
```

```
serv_sock = socket(PF_INET, SOCK_STREAM, 0); // 소켓생성

if(serv_sock == -1) // 소켓에러
    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) //bind 에러
    err_handler("bind() error");

if(listen(serv_sock, MAX_CLNT) == -1) //listen 에러
    err_handler("listen() error");

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

    printf("Check Black List₩n");

    if(check_black_list(inet_ntoa(clnt_addr.sin_addr)))//블랙리스트 체크해서 블랙리스트에 있으면 출력
    {
        write(clnt_sock, "Get out of my server!!!₩n", 23); //clnt 로 보냄
        close(clnt_sock);
        continue; //for 문 다시 돌음
    }

    pthread_mutex_lock(&mtx); //블랙리스트 아니면 쓰레드 잠금
```

```
        info[clnt_cnt].sock = clnt_sock;//accept 된 소켓 info 에 넣음
        strcpy(info[clnt_cnt].ip, inet_ntoa(clnt_addr.sin_addr));
        info[clnt_cnt++].cnt = 0;

        pthread_mutex_unlock(&mtx); //쓰레드 잠금 해제

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

    close(serv_sock);

    return 0;
}
```

```
#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) //에러출력
{
    fputs(msg, stderr);
    fputc('₩n', stderr);
    exit(1);
}

void make_rand_str(char *tmp) //랜덤한 수 만듦
{
    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 접속하세요!!₩n"; //문자 출력
    srand(time(NULL));

    char tmp1[32] = {0};

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

        write(sock, msg, strlen(msg));// 메세지보냄
```

```
#endif
#if ATTACK //공격형 프로그램, BW 보다 많은 데이터 양을 보내면 서버가 죽음
        make_rand_str(tmp1);//랜덤 문자열 만듦

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

    return NULL;
}

void *recv_msg(void *arg)//메세지받기
{
    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);//snd _thread 실행하는 쓰레드 생성
    pthread_create(&rcv_thread, NULL, recv_msg, (void *)&sock);//rcv_thread 실행하는 쓰레드 생성
    pthread_join(snd_thread, &thread_ret);//쓰레드 실행
    pthread_join(rcv_thread, &thread_ret);//쓰레드 실행

    close(sock);

    return 0;
}
```