TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/4/3
수업일수	29 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

- ip 로 host 찾기
- mpecho
- 구조체 전송하기
- 채팅

1. ip 로 host 찾기

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

typedef struct sockaddr_in          si;

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

int main(int argc,char **argv)
{
        int i;
        si addr;
        struct hostent *host;

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

```
        memset(&addr, 0, sizeof(addr));//addr 초기화
        addr.sin_addr.s_addr=inet_addr(argv[1]);//입력받은 인자를 네트워크 주소변환을 하여 addr 에
입력
        host= gethostbyaddr((char *)&addr.sin_addr, 4, AF_INET);//실제 IP 값 가지고 호스트를 찾음
        if(!host)//호스트 값이 없으면 에러
                err_handler("gethost error!");

        printf("Official Name : %s₩n",host→h_name); //official name 출력

        for(i=0; host->h_aliases[i]; i++) //별칭 출력
                printf("Aliases %d : %s ₩n",i+1,host->h_aliases[i]);

        printf("Address type : %s₩n",(host->h_addrtype == AF_INET) ? "AF_INET" : "AF_INET6");
        //address type 출력

        for(i=0; host->h_addr_list[i]; i++) // IP 리스트 출력
                printf("IP Addr %d : %s₩n",i+1,inet_ntoa(*(struct in_addr *)host->h_addr_list[i]));

        return 0;
}
```

```
xeno@xeno-NH:~/proj/0403$ ./a.out 74.125.19.106
gethost error!
xeno@xeno-NH:~/proj/0403$ ./a.out 216.58.197.142
Official Name : nrt12s01-in-f142.1e100.net
Address type : AF_INET
IP Addr 1 : 216.58.197.142
xeno@xeno-NH:~/proj/0403$ ./a.out 125.209.222.141
gethost error!
xeno@xeno-NH:~/proj/0403$ ./a.out 216.58.220.238
Official Name : nrt13s37-in-f238.1e100.net
Address type : AF_INET
```

```
IP Addr 1 : 216.58.220.238
xeno@xeno-NH:~/proj/0403$ ./a.out 168.126.63.1
Official Name : kns.kornet.net
Address type : AF_INET
IP Addr 1 : 168.126.63.1
```

2.mpecho

-mpecho_serv.c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<signal.h>
#include<sys/wait.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);
}

void read_childproc(int sig)//프로세스올때까지 기다렸다가 자식프로세스 죽으면 처리
{
```

```
        pid_t pid;
        int status;
        pid = waitpid(-1, &status, WNOHANG);
            // WNOHANG : 자식프로세스가 종료되지 않아도 즉시 리턴하라는 뜻. block 상태로 만들지
않음.
        printf("Remove proc id : %d₩n", pid);
}


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

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

        act.sa_handler = read_childproc; //SIGCHLD 받으면 read_childproc 함수 실행
        sigemptyset(&act.sa_mask);
        act.sa_flags = 0;
        state = sigaction(SIGCHLD, &act, 0);//자식프로세스가 죽어서돌아오면 act 실행

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

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

for(;;)//클라이언트를 여러명 받기위해
{
        addr_size = sizeof(clnt_addr);
        clnt_sock = accept(serv_sock, (sap)&clnt_addr, &addr_size);
        if(clnt_sock == -1)//accept 에러나면 for 문 다시 돌음
                continue;
        else
                puts("New Client Connected.....");

        pid = fork();//프로세스 생성

        if(pid == -1)//프로세스 오류나면 클라이언트 소켓 닫고 for 문 다시돌음
        {
                close(clnt_sock);
                continue;
        }
```

```
		if(pid==0)//child 프로세스이면
		{
			close(serv_sock);//서버소켓이 필요하지 않으니 닫음

			while((str_len = read(clnt_sock, buf, BUF_SIZE)) != 0)
								//clnt_sock 을 읽어 buf 에 쓴다.
								//blocking 함수여서 clnt_sock 에 값이 들어올 때
까지 기다린다
				write(clnt_sock, buf, str_len);//읽은 값을 쓴다 (에코)
								//while 문에서 클라이언트가 치고 그 값을 읽는걸
계속 반복한다.
			close(clnt_sock); // 클라이언트 소켓 닫음
			puts("Client Disconnected ..."); //q 나 시그널을 받으면 출력됨.
			return 0;//종료되지 않는다. read 가 blocking 이어서 clnt 들어올 때까지
기다린다.
		}
		else
			close(clnt_sock);//clnt 소켓 닫음
	}
	close(serv_sock);

	return 0;
}
```

-mpecho_clnt.c

```
#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) //에러 있을경우  에러 출력
{
	fputs(msg, stderr);
	fputc('₩n',stderr);
	exit(1);
}
void read_routine(int sock, char *buf)
{
	for(;;) //무한반복
	{
		int str_len = read(sock, buf, BUF_SIZE); //소켓을 읽어 buf 에 저장

		if(str_len == 0) //읽을 값이 없으면
			return ; //함수 나간다.
		buf[str_len] = 0;
		printf("msg from server : %s", buf); // 읽은 값을 출력함
	}
}

void write_routine(int sock, char *buf)//입력받은 것을 서버쪽으로 송신
{
	for(;;)
	{
		fgets(buf, BUF_SIZE, stdin);
		if(!strcmp(buf, "q₩n") || !strcmp(buf, "Q₩n"))//q 를 입력하면 shutdown 한다.
```

```
                {
                        shutdown(sock, SHUT_WR);
                        return ;
                }
                write(sock, buf, strlen(buf));//q 나 Q 가 아니면 sock 에 쓴다
        }
}

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

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

        sock = socket(PF_INET, SOCK_STREAM, 0); // 소켓생성

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

```
	if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)//connect 실패
		err_handler("connect() error");
	else // 성공
		puts("Connected.....");

	pid = fork(); //프로세스 생성

	if(pid == 0) //child 프로세스
		write_routine(sock, buf);//자식은 write_routine 실행
	else
		read_routine(sock, buf);//부모는 read_routine 실행

	close(sock);

	return 0;
}
```

3.구조체 전송하기

```
-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; //int 형 변수
	float fdata; //float 형 변수
} d;

#define BUF_SIZE 32

#endif
```

-struct_serv.c

```
#include "common.h"
#include <signal.h>
#include <sys/wait.h>

typedef struct sockaddr_in si;
typedef struct sockaddr * sp;

void err_handler(char *msg) //에러출력
{
	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) //인자 2개 아닐 때
	{
		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); //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)
        err_handler("bind() error");

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

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

        if(clnt_sock == -1) //accept 오류이면 for 문 다시돌음
                continue;
        else //성공하면 출력
                puts("New Client Connected!₩n");

        pid = fork(); // 프로세스 생성

        if(pid == -1) //오류나면 소켓닫고 다시 for 문 돈다
        {
                close(clnt_sock);
                continue;
        }

        if(!pid) child 일 때
        {
                close(serv_sock); // 서버소켓 닫음

                while((len = read(clnt_sock, (d *)&struct_data, BUF_SIZE)) != 0)
                { //클라이언트를 읽는다. blocking 이어서 값이 올 때 까지 기다림
```

```
                                        printf("struct.data = %d, struct.fdata = %f₩n", struct_data.data,
struct_data.fdata);
                                        write(clnt_sock, (d *)&struct_data, len); //클라이언트에서 받은 값
쓴다
                                }

                                close(clnt_sock);
                                puts("Client Disconnected!₩n");
                                return 0; //q 나 시그널 받으면 꺼짐
                        }
                        else //parent 일 때 클라이언트 소켓 닫음
                                close(clnt_sock);
                }
                close(serv_sock);

                return 0;
}
```

-struct_clnt.c

```
#include "common.h"

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

void read_proc(int sock, d *buf)
{
        for(;;) //무한루프
        {
```

```
                int len = read(sock, buf, BUF_SIZE); //소켓을 읽음

                if(!len)//읽은 값이 없을 때 나간다
                        return;

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

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

        for(;;)
        {
                fgets(msg, BUF_SIZE, stdin);

                if(!strcmp(msg, "q₩n") || !strcmp(msg, "Q₩n"))//q 나 Q 를 받으면 shutdown
                {
                        shutdown(sock, SHUT_WR);
                        return;
                }

                buf->data = 3; //data 입력
                buf->fdata = 7.7;

                write(sock, buf, sizeof(d)); //소켓에 데이터 쓴다.
        }
}

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) //인자 3개 아니면 출력
        {
                printf("use: %s <IP> <port>₩n", argv[0]);
                exit(1);
        }

        sock = socket(PF_INET, SOCK_STREAM, 0); //소켓생성

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

        if(connect(sock, (sp)&serv_addr, sizeof(serv_addr)) == -1) //connect 오류
                err_handler("connect() error");
        else //성공적으로 연결되면 출력
                puts("Connected!₩n");

        pid = fork(); //프로세스 생성

        if(!pid) //child 일 때 write_proc 실행
```

```
                write_proc(sock, (d *)&struct_data);
        else //child 일 때  read_proc 실행
                read_proc(sock, (d *)&struct_data);

        close(sock);

        return 0;
}
```

4. 채팅

```
-chat_serv.c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<pthread.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<sys/epoll.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];
pthread_mutex_t mtx;

void err_handler(char *msg) //에러메세지  출력
```

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

void send_msg(char *msg, int len)
{
        int i;
        pthread_mutex_lock(&mtx);
        for(i=0; i<clnt_cnt; i++)
                write(clnt_socks[i], msg, len);

        pthread_mutex_unlock(&mtx);
}

void *clnt_handler(void *arg)
{
        int clnt_sock = *((int *)arg);
        int str_len = 0,i;

        char msg[BUF_SIZE];

        while((str_len = read(clnt_sock, msg, sizeof(msg))) != 0);
                send_msg(msg, str_len);

        pthread_mutex_lock(&mtx);

        for(i=0; i<clnt_cnt; i++)
        {
                if(clnt_sock == clnt_socks[1]);
```

```
                {
                        while(i++< clnt_cnt -1)
                                clnt_socks[i] = clnt_socks[i+1];
                        break;
                }
        }

        clnt_cnt--;
        pthread_mutex_unlock(&mtx);
        close(clnt_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;

        if(argc !=2)
        {
                printf("Usage : %s <port>₩n",argv[9]);
                exit(1);
        }

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

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

```
	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, 10) == -1)
		err_handler("listen() error");

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

		pthread_mutex_lock(&mtx); //lock 설정
		clnt_socks[clnt_cnt++] = clnt_sock; //accept 한 것 clnt_socks 배열에 저장
		pthread_mutex_unlock(&mtx);//lock 해제

		pthread_create(&t_id, NULL, clnt_handler, (void *)&clnt_sock);//쓰레드 생성
		pthread_detach(t_id);//쓰레드 분리
		printf("Conneted Client IP : %s₩n", inet_ntoa(clnt_addr.sin_addr));
			//클라이언트 주소 출력
	}

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

-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] = "[DEFAULT]";
char msg[BUF_SIZE];

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

void *send_msg(void *arg)
{
        int sock = *((int *)arg);
        char name_msg[NAME_SIZE + BUF_SIZE];

        for(;;)
```

```
	{
		fgets(msg, BUF_SIZE, stdin);//msg 입력

		if(!strcmp(msg,"q₩n") || !strcmp(msg, "Q₩n"))//q 나 Q 이면 소켓닫음
		{
			close(sock);
			exit(0);
		}

		sprintf(name_msg, "%s %s", name, msg);//이름과 메시지를 출력함
		write(sock, name_msg, strlen(name_msg));//메시지와 이름 보냄
	}

	return NULL;

}

void *recv_msg(void *arg)
{
	int sock = *((int *)arg);
	char name_msg[NAME_SIZE + BUF_SIZE];
	int str_len;

	for(;;)
	{
		str_len =read(sock, name_msg, NAME_SIZE + BUF_SIZE -1);//소켓읽음

		if(str_len == -1)//read 오류
			return (void*)-1;
		name_msg[str_len] = 0;
		fputs(name_msg, stdout);
	}
```

```
        return NULL;
}

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

        if(argc != 4)//인자 4개 아니면 오류
        {
                printf("Usage : %s <IP> <port> <name>₩n",argv[0]);
                exit(1);
        }

        sprintf(name, "[%s]", argv[3]);
        sock = socket(PF_INET, SOCK_STREAM, 0); //소켓생성

        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);//send_msg 수행쓰레드  생성
```

```
        pthread_create(&rcv_thread, NULL, recv_msg, (void *)&sock);//recv_msg 수행하는 쓰레드 생성
        pthread_join(snd_thread, &thread_ret);
        pthread_join(rcv_thread, &thread_ret);

        close(sock);

        return 0;
}
```

5. 시간 구하는 함수 (채팅 도배 못하게 하기위해)

```
#include "load_test.h"

void get_runtime(tv start, tv end)
{
        end.tv_usec = end.tv_usec - start.tv_usec;
        end.tv_sec = end.tv_sec - start.tv_sec;
        end.tv_usec += end.tv_sec * 1000000;//us

        printf("runtime = %lf sec₩n", end.tv_usec / 1000000.0);
}

#if DEBUG
int main(void)
{
        unsigned int i, cnt = 0;
        tv start, end;

        gettimeofday(&start, NULL);//오늘 현재 시간값이 start 에 셋팅
```

```
        for(i = 0; i < 777777777; i++)//약8억번루프돌음
                cnt++;

        gettimeofday(&end, NULL);//end 에  현재시간  세팅

        get_runtime(start, end);//걸린시간

        return 0;
}
#endif

//걸리는  시간  구하는  함수
```

```
#ifndef __LOAD_TEST_H__

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

typedef struct timeval      tv;

void get_runtime(tv, tv);

#endif
```