TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-04-03 (29 회차)

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

1. gethostbyaddr.c - IP 주소로 호스트(도메인)찾기

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

typedef struct sockaddr_in        si;

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

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

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

        memset(&addr,0,sizeof(addr));
        addr.sin_addr.s_addr = inet_addr(argv[1]);
// 문자열로 표현한 IPv4 주소(10 진수, 점표기)를 네트워크 바이트 정렬 방식의 4 바이트 정수로 변환
        host= gethostbyaddr((char*) &addr.sin_addr , 4 , AF_INET);

        if(!host)
        err_handler("gethost error!");
        printf("Official Name: %s₩n", host->h_name);

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

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

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

        return 0;
}
```

2-1 mpecho_serv.c, mpecho_clnt.c

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

        act.sa_handler = read_childproc; // 시그널핸들러 함수 등록
        sigemptyset(&act.sa_mask);
        act.sa_flags =0;
        state = sigaction(SIGCHLD, &act,0); // 쓰레드가 죽어도 SIGCHLD 날아간다// 분리시키지 않았다면
        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, 5)==-1)
                err_handler("listen() error");

        for(;;) // 클라이언트 여러명을 받기 위함
        {
                addr_size = sizeof(clnt_addr);
                clnt_sock = accept(serv_sock, (sap) &clnt_addr, &addr_size);
                if(clnt_sock==-1)
                        continue;
                else
                puts("New Client Connected...");

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

                if(pid==0) // 자식프로세스
                {
                        close(serv_sock);
                        while((str_len = read(clnt_sock, buf, BUF_SIZE)) !=0)
                        write(clnt_sock, buf, str_len);
/*클라이언트 소켓에서 read 한다. 이를 다시 클라이언트 소켓에 쓴다 (echo)
  종료되지 않는다. read 가 블록킹하여 클라이언트의 입력을 계속 기다린다 */
                        close(clnt_sock);
                        puts("Client Disconnected...");
```

```
                return 0;
        }

        else
                close(clnt_sock);
    }
    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)
{
    fputs(msg, stderr);
    fputc('₩n', stderr);
    exit(1);
}

void read_routine(int sock, char *buf)
{

    for(;;)
    {
        int str_len = read(sock, buf, BUF_SIZE);
        if(str_len == 0)
            return;

        buf[str_len] =0; // 널문자
        printf("msg from server: %s", buf);
    }
}

void write_routine(int sock, char* buf)
{
    for(;;)
    {
        fgets(buf, BUF_SIZE, stdin); // read 와 동일 read(0,buf,BUF_SIZE) , 표준입력을 buf 에 저장
        if(!strcmp(buf, "q₩n") || !strcmp(buf, "Q₩n")) // 무한루프 종료조건
        {
            shutdown(sock, SHUT_WR); // 소켓의 쓰기버퍼를 닫는다.
            return;
        }

        write(sock, buf, strlen(buf)); // buf 에 fgets 하고 sock 에 write 무한루프
    }
}

int main(int argc, char **argv)
{
```

```
        pid_t pid;
        int i, sock;
        si serv_addr;
        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, (sap) &serv_addr, sizeof(serv_addr))==-1)
                err_handler("connect() error");
        else
        puts("Connected.....");

        pid = fork(); //클라이언트에서 fork 하여 송수신기능 분리
        if(pid==0) // 1. 자식프로세스는 키보드 입력을 서버에 전송한다.
                write_routine(sock,buf);
        else // 2. 부모프로세스는 서버에서 읽어서 출력한다.
        read_routine(sock,buf);

        close(sock);

        return 0;
}
```

3. Quiz: 서버와 클라이언트간 구조체 통신 → 알고리즘을 네트워크화 가능하다 (네트워크로 다양한 데이터 전송가능)

```
/* "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
```

```
/*struct_serv.c : accept 이후만 신경쓰면 된다*/
#include "common.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; // 헤더의 구조체 d 참고
        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) // pid 에러
		{
			close(clnt_sock);
			continue;
		}

		if(!pid) // 자식
		{
			close(serv_sock);

			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);
			}

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

	return 0;
}
```

```
/*struct clnt.c : fork 이후만 신경쓰면 된다.나머지는 비슷 */
#include "common.h"

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 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"))
            {
                shutdown(sock, SHUT_WR);
                return;
            }

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

            write(sock, buf, sizeof(d)); // fgets→ write 무한반복
        }
}

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) // 자식
            write_proc(sock, (d *)&struct_data);
        else //부모
            read_proc(sock, (d *)&struct_data);
```

```
	close(sock);

	return 0;
}
```

4. 단톡방 구현하기

(+ 추가: 도배차단기능, 데몬프로세스기능)??

```
#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 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)
{
	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++) // 모든 클라이언트에 대해
		write(clnt_socks[i], msg,len); // 각각의 클라이언트 소켓에 msg 를 전송

	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) // 클라이언트 소켓에서 읽어서 msg 에 저장
		send_msg(msg, str_len);

	pthread_mutex_lock(&mtx);
```

```
	for(i=0; i< clnt_cnt; i++) /* 모든 클라이언트들에 대하여.....???  */
	{
		if(clnt_sock == clnt_socks[i]) // 인자로 전달받은 클라이언트 소켓과 기존의 클라이언트 소켓비교
		{
			while(i++ < clnt_cnt-1)
				clnt_socks[i] = clnt_socks[i+1]; // 기존 클라이언트 소켓이랑 같으므로…???
			break;
		}
	}
	clnt_cnt--; // 클라이언트 소켓의 갯수 -1
	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[0]);
		exit(1);
	}

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


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

		pthread_mutex_lock(&mtx);
		clnt_socks[clnt_cnt++] = clnt_sock;
// 클라이언트 소켓의 fd 를 clnt_socks 배열에 저장 // clnt_cnt 증가
		pthread_mutex_unlock(&mtx);
		pthread_create(&t_id, NULL, clnt_handler, (void*) &clnt_sock);
		pthread_detach(t_id); // 메인스레드에서 분리
```

```
                printf("Connected Client IP: %s₩n", inet_ntoa(clnt_addr.sin_addr));
        }
        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] = "[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);
                if(!strcmp(msg, "q₩n") || !strcmp(msg,  "Q₩n"))
                {
                        close(sock);
                        exit(0);
                }
                sprintf(name_msg,"%s %s", name, msg);
// name 과 msg 를 지정한 방식(%s)로 첫번째 인자로 오는 배열(name_msg)에 저장
                write(sock, name_msg, strlen(name_msg)); // 소켓에 전송
        }
        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);
                        if(str_len == -1) // 에러
                                return (void*)-1; //????

                name_msg[str_len]=0;
                fputs(name_msg, stdout); // read → fputs 무한루프
        }
        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)
        {
                printf("Usage: %s <IP> <port> <name>₩n", argv[0]);
                exit(1);
        }

        sprintf(name, "[%s]", argv[3]); // 실행시 네번째 인자를 [%s]의 형식으로 name 에 저장
        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_port = htons(atoi(argv[2]));
        serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
        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;
}
```

5. RUN TIME 측정 프로그램 구현

*. load test.c / load test.h

```
/* load test.h */
#ifndef __LOAD_TEST_H__

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

typedef struct timeval   tv;
```

```
void get_runtime(tv, tv);

#endif
```

```
#include "load.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;

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

#if DEBUG  // -DDBUG 옵션 주어야 실행된다
int main(void)
{
	unsigned int i, cnt = 0;
	tv start, end;

	gettimeofday(&start, NULL);

	for(i = 0; i < 777777777; i++)
		cnt++;

	gettimeofday(&end, NULL);

	get_runtime(start, end);

	return 0;
}
#endif
```

```
yukyoung@yukyoung-Z20NH-AS51B1U:~/Workspace/0403$ gcc -DDEBUG load.c
yukyoung@yukyoung-Z20NH-AS51B1U:~/Workspace/0403$ ./a.out
runtime = 1.604256 sec
yukyoung@yukyoung-Z20NH-AS51B1U:~/Workspace/0403$
```

*. load_test.c 연산하는데 걸리는 시간
gcc -DDEBUG load_test.c

*. 채팅프로그램 블록킹 팁
셧다운 하는 녀석을 억셉트 디나인에 등록하면
클라이언트 억셉트할때 막을 수 있다.
클라이언트 건드릴 필요 없이 서버만 건드리면 된다

6. [과제] 네트워크 상에서 369 게임 구현하기

```
/* 미해결 */
```