Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정

날 짜 : 2018 . 4. 3

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

<gethostbyaddr.c>

struct hostent *gethostbyaddr(const char *addr, int len, int type);

- ip 주소를 통해서 도메인 주소를 얻어 올 수 있다.

```
struct hostent {
        char    *h_name;        /* 호스트의 공식 이름 */
        char    **h_aliases;    /* 별칭 리스트 */
        int     h_addrtype;     /* 호스트 주소 타입 */
        int     h_length;       /* 주소의 길이 */
        char    **h_addr_list;  /* 주소 리스트 */
}
#define h_addr  h_addr_list[0]  /* 구 버전과의 호환을 위해 */
```

-반환값-

gethostbyname() 그리고 gethostbyaddr() 함수는 hostent 구조체를 반환하거나 만일 에러가 발생한다면 NULL 포인터를 반환한다. 에러시, h_errno 변수는 에러 넘버를 가진다.

```
#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]);
	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("aliases %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;
}
```

// 방화벽이 있고 하면 안될 수 hosterr 라고 뜰수 있다.

<mpecho.c>

fork() 함수를 사용해서 자식 프로세스를 생성하고 echo server 와 client 를 만든다.

<server>

```
#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_childoroc(int sig)
{
        pid_t pid;
        int status;
        pid = waitpid(-1, &status , WNOHANG);
        printf("REMOVED oroc 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_childoroc;
        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 ,(sap)&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 , (sap)&clnt_addr, &addr_size);
		if(clnt_sock ==-1)
			continue;
		else
			puts("New Client Connected ....");

		// fork()  를 통해서 나온 자식 프로세스의 pid 값을 저장한다.
		pid = fork();
		if(pid == -1)
		{
			close(clnt_sock);
			continue;
		}
		if(pid == 0)
		{
			close(serv_sock);

			while((str_len = read(clnt_sock , buf, BUF_SIZE)) != 0)
				write(clnt_sock , buf, str_len);

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

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

<client>

```
#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 ;
                printf("msg from server: %s", buf);
        }
}
void write_routine(int sock , char *buf)
{
        for(;;){
                fgets(buf, BUF_SIZE, stdin);
                if(!strcmp(buf, "q\n") || !strcmp(buf, "Q\n"))
                {
                        shutdown(sock,SHUT_WR);
                        return ;
                }
                write(sock, buf, strlen(buf));
        }
}
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);
        }


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

        // 커넥트 하는 순간 어셉트가 동작한다.
        // 위에서 설정한 (포트와 주소 정보(serv_adr)) 를 가지고 connect 를 시도한다.
        if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
                err_handler("connect() error");

        else
                puts("Connected .....");

        pid = fork();

        if(pid == 0)
                write_routine(sock , buf);
        else
                read_routine(sock, buf);

        close(sock);

        return 0;
}
```

채팅 서버와 클라이언트를 구현하자!!!

<chat_serv.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 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++)
                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[i])
		{
			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[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;
		pthread_mutex_unlock(&mtx);

		pthread_create(&t_id, NULL, clnt_handler, (void *) &clnt_sock);
		pthread_detach(t_id);
		printf("Connected Client IP: %s\n", inet_ntoa(clnt_addr.sin_addr));
	}
	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);

		if(!strcmp(msg, "q\n") || !strcmp(msg, "Q\n"))
		{
			close(sock);
			exit(0);
		}

		sprintf(name_msg, "%s %s", name, msg);
		write(sock , name_msg, strlen(name_msg));
	}
	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)
			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)
	{
		printf("Usage: %s <IP> <port> <name> \n", argv[0]);
		exit(1);
	}
	sprintf(name, "[%s]", argv[3]);
	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]));

	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;
}
```