Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정
#29

강사:Innova Lee(이 상훈)
학생: 김시윤

수업내용 복습

mp_echo_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);
//프로세스 올때까지 기다렸다가 리무브드
        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;
        sigemptyset(&act.sa_mask);
        act.sa_flags = 0;
        state = sigaction(SIGCHLD, &act, 0);//시그차일드 오면

        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...");
		pid = fork();//프로세스 생성
		if(pid == -1)//오류
		{
			close(clnt_sock);
			continue;
		}
		if(pid == 0)//자식
		{
			close(serv_sock);//서버소켓 닫는다.

			while((str_len = read(clnt_sock, buf, BUF_SIZE)) !=0)
//들어오는거 읽어다 다시쏴줌 에코
				write(clnt_sock, buf, str_len);

			close(clnt_sock);//read 가 block 니까 종료될일 없음.
			puts("Client Disconnected ...");
			return 0;
		}
		else
			close(clnt_sock);
//클라이언트 소켓 닫고 새로운 클라이언트 올때까지 어셉트에서 기다림
	}
	close(serv_sock);

	return 0;
}
//여러사람이 접속했을때 갱신이 빨리 안됨.
```

mp_echo_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;
		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);
	}

	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();
//자식은라이트 부모는 리드

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

	close(sock);

	return 0;
}
```

get_host_by_addr.c

```
#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]);//argv[1]에 IP 입력
	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");
//3 항 연산자로 ip 타입 알아냄

	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;
}
```

Serv.h

```
#include <stdio.h>
#include <stdlib.h>

struct data
{
	char data1[32];
	char data2[32];
};
typedef struct data Data;
```

---

struct_serv.c

```
#include "serv.h"
#include <malloc.h>
#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;
	sigemptyset(&act.sa_mask);
	act.sa_flags = 0;
	state = sigaction(SIGCHLD, &act, 0);//시그차일드 오면

	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");

	Data *data;
	data=(Data *)malloc(sizeof(Data));

	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...");
		pid = fork();
		if(pid == -1)
		{
			close(clnt_sock);
			continue;
		}
		if(pid == 0)//자식
		{
			close(serv_sock);

		//	while((str_len = read(clnt_sock, buf, BUF_SIZE)) !=0)
			while((str_len = read(clnt_sock,data->data1,BUF_SIZE)) !=0)

				write(clnt_sock,data->data1, str_len);

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

```
                }
                else
                        close(clnt_sock);
        }
        close(serv_sock);

        return 0;
}
```

struct_clnt.c

```
#include "serv.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <malloc.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)
void read_routine(int sock)
{
        Data *data;
        data = (Data *)malloc(sizeof(Data));
        for(;;)
        {
                int str_len = read(sock,data->data1,BUF_SIZE);

                if(str_len == 0)
                        return;

                //buf[str_len] = 0;
                data->data1[str_len]=0;
                printf("msg from server: %s",(data->data1));
        }
}

//void write_routine(int sock, char *buf)
void write_routine(int sock)
{
        Data *data;
        data = (Data *)malloc(sizeof(Data));
        for(;;)
        {
                //fgets(buf, BUF_SIZE, stdin);
                fgets(data->data1,BUF_SIZE,stdin);
        //      if(!strcmp(buf, "q\n") || !strcmp(buf, "Q\n"))
                if(!strcmp(data->data1,"q\n") || !strcmp(data->data1, "Q\n"))
                {
                        shutdown(sock, SHUT_WR);
                        return;
                }
        //write(sock,buf,strlen(buf));
        write(sock,data->data1,strlen(data->data1));
        }
}

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();

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

	close(sock);

	return 0;
}
```

*구조체 자체를 보내려면!!!

```
struct data{
	char d1;
	char d2;
};
typedef struct data D;
```

라고할때

메인에 D *data; 라고 선언 되어있을때

```
read(clnt_sock,data,sizeof(D));
```

라고 하면된다.
D 라는 구조체 사이즈만큼 데이터에 클라이언트 값을 적는다.
클라이언트가

```
write(sock,data,sizeof(D));
```

를 했으면 구조체 자체가 전달된다.

서버에 전달된 클라이언트 구조체를 화면에 출력해줄때

```
read(0,data→d1,strlen(d1));
```

하면 확인이 가능하였다.
놀랍고 신기한 방법이다.

malloc 은 문자열을 전달했기 때문에 구조체 사이즈가 모자를 수 있음을 대비해 사용하였다.

chat_serv.c

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

```c
#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;

time_t start =0;
int clnt_cnt = 0;
int clnt_socks[MAX_CLNT];
pthread_mutex_t mtx;


printf("start = %d ",time);
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);
//클라이언트 전체에 메시지가 뿌려짐

	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)
//클라이언트가 쓰는 메시지를 읽음
	{	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]";
//DEFAULT 가 NULL 과 같은건지??
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));
//fgets 에서 얻은 메시지를 서버에 넘겨줌
        }
        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);
        //서버에서 보내줌 메시지를 받음 (for 문에서 뿌리는 전체 clnt 에 보내는 메시지
                if(str_len == -1)
                        return (void *)-1;

                name_msg[str_len] = 0;
                fputs(name_msg, stdout);
//받은 메시지와 이름을 출력
        }
        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);

        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;
}
```

3,6,9 게임 진행중

3,6,9 게임을 학원에 놓고와버렸습니다.

현재 진행상황

숫자 1 씩 증가 완료.
같은 데이터 여러클라이언트 동시 공유 완료.(스레드)
3,6,9 일때 3,6,9 의 갯수 카운트 완료
클라이언트에서 시그날 받아서 서버로 보내는거 시그널 한번은 구현 가능
시그널이 두번 들어왔을때 서버로 넘기는게 안됨..
실패..

(현재 3,6,9 게임 3,6,9 갯수 관계없이 한번 시그널 발생으로 넘기는것까지는 완료 했습니다.)