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

날 짜 : 2018 . 3. 31

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

<inet_aton.c>

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

// 호스트(로컬) 네트워크 호스트(로컬)
//      리틀                  빅              빅
//        빅                  빅              리틀

// 네트워크는 무조건 빅으로 들어온다.

//osi7 계층, tcp/ip 에 헤더에 리틀인지 빅인지에 대한 정보가 있다.

typedef struct sockaddr_in si;

void err_handler(char *msg)
{
        write(2, msg, strlen(msg));
        exit(1);
}

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

        char *addr = "127.124.73.31";
        si addr_inet;

        if(!inet_aton(addr, &addr_inet.sin_addr))
        err_handler("Converslon Error!");
        else
        printf("Network Ordered integer Addr:%#x\n", addr_inet.sin_addr.s_addr);

        return 0;
}
```

<inet_aton.c>

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

typedef struct sockaddr_in si;

int main(int argc, char **argv)
{
	si addr1, addr2;

	char *str;
	char str_arr[32] = {0};
	//hton-> l 들어가면 long 타입 s 가 들어가면 short 타입 이다.
	addr1.sin_addr.s_addr = htonl(0x10203040);
	addr2.sin_addr.s_addr = htonl(0x12345678);

	// inet_ntoa 는 Network to Address 이다.
	// str 을 대입안해도 왜 나오냐면 컴파일러가 똑똑해서..
	str = inet_ntoa(addr1.sin_addr);
	strcpy(str_arr,str);
	printf("Not 1: %s\n", str);

	// inet_ntoa...
	inet_ntoa(addr2.sin_addr);
	printf("Not 2: %s\n", str);
	printf("Not 3: %s\n", str_arr);
	return 0;
}
```

<echo_server.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          1024

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

int main(int argc, char **argv)
{
        int i , str_len;
        int serv_sock, clnt_sock;

        char msg[BUF_SIZE];

        si serv_addr, clnt_addr;
        socklen_t clnt_addr_size;

        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(0 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");
        // 받을 사람 수.
        if(listen(serv_sock, 5) == -1)
```

```
		err_handler("lister() error");

	clnt_addr_size = sizeof(clnt_addr);

	for(i = 0 ; i<5; i ++)
	{	// 실질적인 클라이언트  승인
		clnt_sock = accept(serv_sock , (struct sockaddr *)&clnt_addr, &clnt_addr_size);
		if(clnt_sock == -1)
			err_handler("accept() error");
		else
			printf("Connected Clinet %d\n", i +1);
		// read 가 블록킹이라 글자가 전달이 안된다.
		while((str_len = read(clnt_sock, msg, BUF_SIZE)) !=0)
			write(clnt_sock , msg, str_len);

		close(clnt_sock);

	}
	close(serv_sock);

	return 0;
}
```

<echo_server.c>

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


typedef struct sockaddr_in si;
typedef struct sockaddr * sap;

#define BUF_SIZE          1024

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

int main(int argc, char **argv)
{
        int sock , str_len;
        si serv_addr;
        char msg[32];
        char *m = "Input Message(q to quit): ";

        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 ....");
```

```
	for(;;)
	{
		// put 은 read
		fputs("Input msg(q to quit) : ", stdout);
		// get 은 write
		fgets(msg, BUF_SIZE, stdin);

		if(!strcmp(msg, "q\n") || !strcmp(msg, "g\n"))
			break;

		write(sock, msg, strlen(msg));
		str_len = read(sock, msg, BUF_SIZE -1);

		if(str_len == -1)
			err_handler("read() error!");

		msg[str_len] = 0;
		printf("msg from serv: %s\n", msg);

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

<op_server.c>

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

typedef struct sockaddr_in si;
typedef struct sockaddr * sap;

#define BUF_SIZE 1024
#define OPSZ    4
// 사이즈 크기가 4 개라는 뜻으로 씀
void err_handler(char *msg)
{
        fputs(msg, stderr);
        fputc('\n', stderr);
        exit(1);
}
int calculate(int opnum, int *opnds, char op)
{
        int result = opnds[0] ,i;

        switch(op)
        {
                case '+' :
                        for(i = 1; i<opnum; i++)
                                result += opnds[i];
                        break;

                case'-':
                        for(i= 1; i<opnum; i++)
                                result -= opnds[i];
                        break;
                case'*':
                        for(i=1; i<opnum; i++)
                                result *= opnds[i];
                        break;
        }
        return result;
}
int main(int argc, char **argv)
{
        int serv_sock, clnt_sock;
        char opinfo[BUF_SIZE];

        int result, opnd_cnt, i;
        int recv_cnt , recv_len;
```

```
	si serv_addr, clnt_addr;
	socklen_t clnt_addr_size;

	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, 5) == -1)
		err_handler("listen() error");

	clnt_addr_size = sizeof(clnt_addr);

	for(i = 0 ; i <5 ; i++)
	{
		opnd_cnt = 0;
		clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size);
		read(clnt_sock, &opnd_cnt, 1);

		recv_len = 0;
		// OPSZ op 의 크기, 더블을 연산 하면 8 로 해야함.
		// 여러명이 접속할 수 있으니까 반복문
		while((opnd_cnt *OPSZ +1) >recv_len)
		{
			recv_cnt = read(clnt_sock, &opinfo[recv_len], BUF_SIZE -1);
			recv_len += recv_cnt;
		}

		result = calculate(opnd_cnt, (int *)opinfo,opinfo[recv_len -1]);
		write(clnt_sock, (char*)&result, sizeof(result));

		close(clnt_sock);

	}
	close(serv_sock);

	return 0;
}
```

<op_client.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		1024
#define RLT_SIZE		4
#define OPSZ			4


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

int main(int argc, char **argv)
{
	int i , sock, result, opnd_cnt;
	char opmsg[BUF_SIZE]= {0};
	si serv_addr;
	if(argc != 3)
	{
		printf("use:%s <IP> <port>\n", argv[0]);
	}

	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.....");
	fputs("Operand Cnt: ", stdout);
```

```
	scanf("%d", &opnd_cnt);

	opmsg[0] = (char)opnd_cnt;
	for(i = 0; i<opnd_cnt ; i++)
	{
		printf("Operand %d: ", i+1);
		scanf("%d", (int *)&opmsg[i * OPSZ +1]);
	}
	fgetc(stdin);
	fputs("Operator: ", stdout);
	scanf("%c", &opmsg[opnd_cnt *OPSZ +1]);
	write(sock ,opmsg, opnd_cnt *OPSZ +2);
	read(sock , &result, RLT_SIZE);
	printf("Operation result : %d \n", result);
	close(sock);
		return 0;
}
```

<게임만들기...>_ 아직 원활하게 동작 못함..

<game_server...>

```
#include<setjmp.h>
#include<stdio.h>
#include<stdlib.h>
#include<signal.h>
#include<string.h>
#include<arpa/inet.h>
#include<sys/wait.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<unistd.h>
#include<malloc.h>
#include<time.h>
#include<fcntl.h>

#define BUF_SIZE 1024
#define OPSZ		4
#define EMPTY		0

typedef struct sockaddr_in si;
typedef struct sockaddr *sap;

int flag=0;
int cnt=0;
int serv_sock, clnt_sock;
int last_flag=0;
jmp_buf env;
// 사이즈 크기가 4 개라는 뜻으로 씀
void err_handler(char *msg)
{
	fputs(msg, stderr);
	fputc('\n', stderr);
	exit(1);
}

// 카운트를 받아서 카운트를 1 증가한다.
void my_sig(int signo)
{
	//printf("입력 시간이 늦었습니다 카운트 수를 올립니다!\n");
	char inputtime[]="입력 시간이 늦었습니다 카운트 수를 올립니다!\n";
	write(clnt_sock, inputtime, sizeof(inputtime));
	cnt++;
	alarm(3);

	//exit(0);
	//longjmp(env,1);
}

int game(int clnt_sock,int random)
{
	char buf[1024];
	char ret[BUF_SIZE]={0};
	int setnum;
```

```
	//if(flag == 0){
		char up[]="너 높다\n";
		char down[] = "너 낮다\n";
		char correct[]= "맞추셨습니다! 술 드세요!!!\n";
		char start[] = "술게임을 시작합니다.\n 내가 내는 번호를 맞춰 보세요!\n";

		signal(SIGALRM, my_sig);


		write(clnt_sock, start, sizeof(start));

		int i=0;
		flag = 1;
	//}
	// 입력 시간을 3 초로 해준다.

	//setnum = atoi(opnds);
	while(last_flag != 1){

		memset(&buf, 0 , BUF_SIZE);

		read(clnt_sock, buf, sizeof(buf));
		setnum = atoi(buf);


		memset(&ret, 0 , BUF_SIZE);
		sprintf(ret, "random: %d , recv:%s \n", random, buf);
		write(clnt_sock, ret , sizeof(ret));
		//setjmp(env);
		if(setnum == random){
			alarm(0);
			// 여기서 알람을 끄거나 마지막 부분에 끝난다고 해주어야함.
			write(clnt_sock, correct, sizeof(start));
			// 누가 1 등인지 알아맞추는 함수를 만든다.
			last_flag=1;
			// 그리고 꺼준다.
			close(clnt_sock);
		}
		else if(setnum > random){
			alarm(3);
			write(clnt_sock, up, sizeof(up));
		}
		else if(setnum < random){
			alarm(3);
			write(clnt_sock, down, sizeof(down));
		}
		else
			write(clnt_sock, "awwwwwwww" , BUF_SIZE);
		cnt++;
	}
	//카운트를 증가해 준다.
	return cnt;
}

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

```
	char opinfo[BUF_SIZE];

	int opnd[5]={0}, i;
	int client=0;
	char result;
	int recv_cnt , recv_len;
	si serv_addr, clnt_addr;
	socklen_t clnt_addr_size;

	int random;
	int bufnum =0;
	int status;
	char buf[1024];

	srand(time(NULL));
	random = rand()% 3333 +1;

//	game(random);

////////////////////////////////////////////////////////////////////////////////
	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, 5) == -1)
		err_handler("listen() error");

	clnt_addr_size = sizeof(clnt_addr);
////////////////////////////////////////////////////////////////////////////////
	// 여러명이 접속 할 수 있으니까 반복문으로 했다.??



	for(client=0;client<5;client++){
		if((opnd[client]=fork())>0)
		{
			client++;
			waitpid(-1,&status,0);
		}
		else if(opnd[client] == 0){
			clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size);
			//fcntl(clnt_sock, F_SETFL, O_NONBLOCK);
			//fcntl(serv_sock, F_SETFL, O_NONBLOCK);
```

```
                        recv_len = read(clnt_sock, buf, sizeof(buf)); // 읽을 갯수를 받음.
                        bufnum = atoi(buf);
                        // 게임 함수가 return 을 up 인지 down 인지 맞는지 리턴한다.

                        if(bufnum == 1){
                                fcntl(serv_sock, F_SETFL, O_NONBLOCK);
                                result = game(clnt_sock,random);
                        }
                        //write(clnt_sock, &result, sizeof(result));
                        else if(last_flag == 1){
                                printf("끝났습니다!!\n");
                                close(clnt_sock);
                        }

                }
                // 여기는 맞았을 때만 클라이언트를 끝낸다고 해야한다.
                //close(clnt_sock);
                if(last_flag == 1)
                        return 0;
                //client++;
        }
        close(serv_sock);
        return 0;
}
```

<game_client...>

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


typedef struct sockaddr_in si;
typedef struct sockaddr * sap;

#define BUF_SIZE        1024
#define RLT_SIZE        4
#define OPSZ            4


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

int main(int argc, char **argv)
{
        int i , sock, result,idx, nread;
        int str_len = 0;
        char opmsg[BUF_SIZE]= {0};
        char buf[BUF_SIZE] = {0};
        si serv_addr;
        if(argc != 3)
        {
                printf("use:%s <IP> <port>\n", argv[0]);
```

```
	}

	sock = socket(PF_INET , SOCK_STREAM, 0);

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

//	if( fcntl( sock, F_SETFL, O_NONBLOCK) == -1 )
//		return -1;

	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("start button is 1, exit q");

	fcntl(sock, F_SETFL, O_NONBLOCK);

	for(;;)
	{
		// get 은 write
		fgets(buf, BUF_SIZE, stdin);
		//read(1, buf, BUF_SIZE -1);


		if(!strcmp(buf, "q\n") || !strcmp(buf, "g\n"))
			break;

		write(sock, buf, strlen(buf));
		// read 에서 블럭이 걸려버린다... 이걸 풀어주어야한다...
		nread = read(sock, buf, sizeof(buf));

		//if(nread == -1)
		//	err_handler("read() error!");

		//write(1, buf, strlen(buf));
		printf("msg from serv: %s\n", buf);
		// put 은 read
		write(1, buf, strlen(buf));
		fputs("Input msg(q to quit) : \n", stdout);
	}

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