TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – 문한나

mhn97@naver.com

예제 1)

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

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("Conversion Error!");
        else
                printf("Network Ordered integer Addr: %#x\n",addr_inet.sin_addr.s_addr);

        return 0;

}

//address to network
//기본적으로 빅엔디안(순서대로 저장)
//네트워크 주소로 바꾼다는 것은 기준을 둔다는 것(빅 엔디안)
//인텔은 무조건 리틀엔디안
//이 예제는 리틀을 빅으로 바꾼것임
```

```
mhn@mhn-Z20NH-AS51B5U:~/linux/28$ ./a.out
Network Ordered integer Addr: 0x1f497c7f
mhn@mhn-Z20NH-AS51B5U:~/linux/28$
```

(빅으로 고정임)

로컬		네트워크		로컬	읽을 때
리틀	→	빅	→	빅	그대로
빅	→	빅	→	빅	크로스매칭

예제 2)

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

typedef struct sockaddr_in si;

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

      si addr1,addr2;

      char *str;
      char str_arr[32]={0};

      addr1.sin_addr.s_addr = htonl(0x10203040);
      addr2.sin_addr.s_addr = htonl(0x12345678);

      str = inet_ntoa(addr1.sin_addr); //network to address
      strcpy(str_arr,str);
      printf("Not 1: %s\n",str);

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

            return 0;

}

//network to address
//다시 크로스매칭 시키는 것임
//컴파일러 똑똑함(딥러닝함)
```

```
mhn@mhn-Z20NH-AS51B5U:~/linux/28$ ./a.out
Not 1: 16.32.48.64
Not 2: 18.52.86.120
Not 3: 16.32.48.64
mhn@mhn-Z20NH-AS51B5U:~/linux/28$
```

예제 3)

<op_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
#define OPSZ 4 //사이즈 크기

void err_handler(char *msg){

	fputs(msg,stderr);
	fputc('\n',stderr);
	exit(1);

}

int calculate(int opnum,int *opnds,char op){ //연산 수행

	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); //파일 디스크립터 넘어옴

	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) //서버의 ip 주소 세팅됨
		err_handler("bind() error");

	if(listen(serv_sock,5)==-1)  //클라이언트를 5 명까지 받겠다
		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); //클라이언트의 접속을 허용해
준다
		read(clnt_sock, &opnd_cnt,1); //클라이언트에서 넘어온 정보를 읽음

		recv_len = 0;

		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]); //클라이언트에서 날라온 데이터
로 연산 수행
		write(clnt_sock, (char *)&result, sizeof(result)); //결과를 클라이언트에게 전송

		close(clnt_sock);



	}
	close(serv_sock);
	return result;

}
```

<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]);
		exit(1);

	}

	sock = socket(PF_INET, SOCK_STREAM, 0); //파일 디스크립터 넘어옴

	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) //서버 연결
		err_handler("connect() error");
	else
		puts("Connected .....");

	fputs("Operation Cnt: ", stdout);
	scanf("%d",&opnd_cnt);  //몇번 연산할 것인지 적음
```

```
        opmsg[0] = (char)opnd_cnt; //버퍼에 넣음

        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); //서버에서 날라온 결과 값 받음
        close(sock);


        return 0;

}
```

```
mhn@mhn-900X3L:~/my_proj/linux/28$ gcc -o ser op_server.c
mhn@mhn-900X3L:~/my_proj/linux/28$ ./ser 7777
```

```
mhn@mhn-900X3L:~/my_proj/linux/28$ ./cl 127.0.0.1 7777
Connected .........
Operation Cnt: 2
Operand 1: 3
Operand 2: 5
Operator: *
Operation result: 15
mhn@mhn-900X3L:~/my_proj/linux/28$
```

이 예제는 계산기를 만든 것이다.
기본적으로 틀은 똑같다 서버는 소켓을 생성하여 bind()로 서버 소켓으로 등록한다. 그 후 listen()으로 클라이언트의 접속 요청을 확인하고 accept()로 접속 요청을 받는다. 여기까지 진행되면 클라이언트 소켓이 생긴다. 클라이언트는 소켓을 만들고 connect()로 서버와 연결한다. 이제 통신이 가능하다. 이 예제에서는 우선 연산 횟수를 먼저 받고, 그리고 연산할 숫자들을 받는다. 이 정보들은 버퍼에 한번에 넣어서 포인터로 옮겨가며 읽는다. 서버에서 연산 수행 후 결과를 클라이언트쪽에 넘기고, 클라이언트는 그 데이터를 읽어서 프린트로 찍어준다.