TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/3/30
수업일수	27 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

1.aton

~소스코드(inet_aton.c)

```
#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) //에러메세지 출력
{
        write(2,msg,strlen(msg));
        exit(1);
}

int main(int argc, char **argv)
{
        char *addr= "127.124.73.31"; //7f. 7c. 49. 1f
        si addr_inet;

        if(!inet_aton(addr,&addr_inet.sin_addr))//값이 없으면
                err_handler("Conversion Error"); //에러 출력
        else //있으면
                printf("network ordered integer
addr : %#x₩n",addr_inet.sin_addr.s_addr);
```

```
                        //네트워크에 저장된 값 출력
        return 0;
}
```

~결과
network ordered integer addr : 0x1f497c7f
big endian 으로 저장됨을 알 수 있다.
네트워크는 big endian 으로만 저장된다.
호스트는 little endian 일 수 도 있고 big enidan 일 수 도 있다.

inet_addr( ) : 빅엔디안 32비트 값으로 변환식켜주는 함수. 성공하면 빅엔디안 형식의 32비트 값을, 실패하면 INADDR_NONE 을 리턴한다. INADDR_NONE 은 -1로 선언되어 있다.

2. ntoa

~소스코드(inet_ntoa.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};
```

```
        addr1.sin_addr.s_addr = htonl(0x10203040); //network 에 big endian 으로 저장
        addr2.sin_addr.s_addr = htonl(0x12345678); //network 에 big endian 으로 저장
        // 네트워크는 big endian 으로 데이터가 저장된다. 호스트는 little 일 수도 big 일 수도 있다.

        str = inet_ntoa(addr1.sin_addr);
        //network 에서 주소를 받는다 호스트에 따라 big/little 형식으로 바꾸어 저장됨
        strcpy(str_arr,str);
        printf("Not 1 :%s₩n",str);

        inet_ntoa(addr2.sin_addr);
        //str 에 안넣어도 컴파일러가 알아서 넣어줌,컴파일러가 딥러닝도 한다.
        //ntoa : 네트워크_network to address
        printf("Not 2 : %s₩n",str);
        printf("Not 3 : %s₩n",str_arr);

        return 0;
}
```

~결과

Not 1 :16.32.48.64

Not 2 : 18.52.86.120

Not 3 : 16.32.48.64

3. echo

- **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) //에러메세지 출력
{
        fputs(msg,stderr);
        fputc('₩n',stderr);
        exit(1);
}

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

-**echo_client2.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) //에러메세지 출력
{
  fputs(msg, stderr);
  //fputs 는 문자열을 출력하고 fputc 는 문자를 출력하는 것 같음
  //stderr 가 정확히 뭔지 모르겠다
  fputc('₩n', stderr);
  exit(1);
}

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

```
        char msg[BUF_SIZE];
        si serv_addr, clnt_addr;
        socklen_t clnt_addr_size;

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

        serv_sock = socket(PF_INET, SOCK_STREAM, 0);
         //소켓생성

        if(serv_sock == -1)//소켓에러
                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 = htonl(atoi(argv[1]));

        if(bind(serv_sock, (sap)&serv_addr,
sizeof(serv_addr))==-1)//소켓에 주소 부여
                err_handler("bind() error");
        if(listen(serv_sock, 5) == -1)//클라이언트를 5명까지 받음
                err_handler("listen() error");
```

```
  int sock;
  int str_len;
  si serv_addr;
  char msg[32];
  char *m = "Input Message(q to quit):";

 if(argc !=3)//인자가 3개가 입력되지 않았을 경우 출력
 {
     printf("use : %s <IP> <port>₩n", argv[0]);
     exit(1);
 }

    sock = socket(PF_INET, SOCK_STREAM, 0);//소켓생성

  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");
//서버와 연결시켜준다
```

```
        clnt_addr_size = sizeof(clnt_addr);

        for(i=0; i<5; i++)
        {
                clnt_sock =accept(serv_sock, (struct sockaddr
*)&clnt_addr, &clnt_addr_size);
                if(clnt_sock == -1)
                        err_handler("accept() error");
                else
                        printf("Conneted Client %d₩n", i+1);

                while((str_len = read(clnt_sock, msg,
BUF_SIZE)) != 0)//클라이언트 소켓을 읽어 msg 배열에 저장
                        write(clnt_sock, msg, str_len);
                        //클라이언트 소켓에 msg 를 읽은 것을 쓴다
                close(clnt_sock);
        }
        close(serv_sock);

        return 0;
}
```

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

  for(;;)//무한루프
  {
      fputs("Input msg(q to quit): ", stdout);//메세지 입력받음
(wirte 랑 같은것)
      fgets(msg, BUF_SIZE, stdin);//read 랑 같음

      if(!strcmp(msg, "q₩n") || !strcmp(msg, "Q₩n"))
          break;//무한루프를 빠져나감

      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;
    //0을 왜 넣는지 모르겠음 ㅠ
    printf("msg form serv : %s₩n", msg);

  }
  close(sock);

return 0;
```

	}

실행했을 때 처음 들어온 클라이언트는 메시지를 입력하면 메시지가 출력한다. 하지만 첫번째로 들어온 클라이언트 이후에 들어온 나머지 클라이언트는 메시지를 입력해도 본인 메시지가 뜨지 않고 첫번째 클라이언트가 나간 이후에 이전에 다른 클라이언트들이 썼던 메시지가 한번에 입력된다. 그 이유는 서버에서 클라이언트가 쓴 메시지를 읽는 read 가 block 함수이기 때문이다. 이런 문제를 해결하기 위해서는 nonblocking 함수로 바꿔주거나 아니면 fork 를 사용하여 프로세스를 여러 개 만들어 사용하는 방법이 있다.

3.operation

- **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

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


int calculate(int opnum, int *opnds, char op)//계산
{
```

- **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 ALT_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;
```

```
        int result = opnds[0],i;

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

                case'-':인자로 -가 들어올 경우
                        for(i=1; i<opnum; i++)
                                result -= opnds[i];
                        break;

                case'*': //인자로 *가 들어올 경우
                        for(i=1; i<opnum; i++)
                                result *= opnds[i];
                        break;

        }
        return result;

}
```

```
        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);
        //sock 에 소켓의 파일 디스크립터를 저장
        if(sock == -1)//소켓에러
                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)//서버와 연결시켜준다
                err_handler("connect() error");
        else
                puts("Conneted .....");
```

```
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) //인자 2개가 입력되지 않으면 오류 메세지 출력
	{
		printf("use: %s <port>₩n", argv[0]);
		exit(1);
	}

	serv_sock = socket(PF_INET, SOCK_STREAM, 0); //소켓 파일디스크립터 출력?

	if(serv_sock == -1) //소켓에러
		err_handler("socket() error");

	memset(&serv_addr,0,sizeof(serv_addr));
	fputs("Operand Cnt: ", stdout);
	scanf("%d", &opnd_cnt);

	opmsg[0] = (char)opnd_cnt;

	for(i=0; i<opnd_cnt ; i++)
	{//인자 3개에 값들을 받는다
		printf("Operand %d:", i+1);
		scanf("%d", (int *)&opmsg[i * OPSZ +1]);
	}//숫자 값 입력

	fgetc(stdin);//1byte 만 읽어 부호값 읽음
	fputs("Operator : ", stdout);
	scanf("%c", &opmsg[opnd_cnt * OPSZ +1]);
	write(sock, opmsg, opnd_cnt * OPSZ +2);
	read(sock, &result, ALT_SIZE);//client 는 끊어서 read 할 필요없음 server 는 필요
	//계산을 서버가 해서 클라이언트가 받는다.
	printf("OPeration result : %d₩n", result);
	//결과 값 출력
	close(sock);

	return 0;
}
```

```
        //serv_addr 배열을 0으로 초기화
        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)//클라이언트를 5명을 받음
                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;
                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;
}
```

~결과

```
xeno@xeno-NH:~/proj/0330$ ./op_client 127.0.0.1 7777
Conneted .....
Operand Cnt: 3
Operand 1:8
Operand 2:13
Operand 3:57
Operator : *
OPeration result : 5928
```

memset( ) : 배열의 초기화를 위한 함수, <string.h>에 있다.
형태 : void *memset(void *s, int c, size_t n);
인자 :
*s : 시작주소
c : 초기화 할 값
n : 배열의 크기

socket( ) : 데이터 전송에 사용되는 소켓을 생성할 때 호출하며, 소켓을 성공적으로 생성하면 소켓 디스크립터를 반환
형태 : int socket(int domain, int type, int protocol);
인자 :
domain : 도메인
type : 서비스 유형
protocol : 프로토콜, 대개 0으로 지정한다.

bind( ) : socket 함수로 소켓을 생성하면 소켓 디스크립터가 반환되는데 이 디스크립터를 이용해 상대 프로세스와 통신하려면 생성된 소켓에 주소를 부여해야한다. 이 bind 함수가 소켓에 주소를 부여하는 기능을 한다.
형태 : int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
인자 :
sockfd : socket( )함수의 반환 값인 소켓 디스크립터 번호
addr : 바인드 할 소켓의 주소
addrlen : 보관된 주소의 크기

accept( ) : 서버 프로그램이 클라이언트의 연결 요청을 받으려면 accept 함수에서 대기해야한다. 성공적으로 연결되면 socket( )함수로 생성한 원래의 소켓과는 별도의 소켓이 새로 만들어진다. 생성된 소켓의 디스크립터는 accept( )함수의 반환값으로 얻을 수 있고, 클라이언트와 통신할 때는 보통 이 소켓을 사용한다.
형태 : int accept(int s, struct sockaddr *addr, socklen_t *addrlen);

인자 :

sockaddr

socklen : 연결을 요청한 클라이언트의 소켓 주소가 저장됨

addrlen

listen( ) : 서버 프로그램에서 실행된다.

형태 : int listen(int sockfd, int backlog);

인자

sockfd :

backlog : 대기할 수 있는 클라이언트의 연결 요청 개수를 지정