Xilinx Zynq FPGA, TI DSP MCU 기반의

프로그래밍 및 회로 설계 전문가

강사 이상훈

(Innova Lee)

Gcccompil3r@gmail.com

학생 김민호

minking12@naver.com

```
#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)//네트워크는 기준이 빅 엔디언이다 . 순서대로 저장
{
   char *addr="127.124.73.31";//크로스 매칭(리틀에서 빅으로) -> 1f.49.7c.7f(31.73.124.127)
   si addr_inet;

   if(!inet_aton(addr,&addr_inet.sin_addr))//address to network
         err_handler("conversation error");
   else
         printf("network ordered integer addr : %#x\n",addr_inet.sin_addr.s_addr);

   return 0;
}
```

```
#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);//host to network 네트웤기준이 빅 이니까 순서대로 저장
                                           //16.32.48.64
   addr2.sin_addr.s_addr=htonl(0x12345678);//18.52.86.120

   str=inet_ntoa(addr1.sin_addr);//network to address(host)
   strcpy(str_arr,str);
   printf("not 1: %s\n",str);
   //컴파일러에서 str 을 알아서 띄워준다.
   inet_ntoa(addr2.sin_addr);//network to address(host)
   printf("not2 : %s\n",str);
   printf("not3 : %s\n",str_arr);

   return 0;

}
```

```
#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() 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) // 5 명 받을 수 있다.
	err_handler("listen() error");

	clnt_addr_size = sizeof(clnt_addr);

	for(i=0; i<5;i++) // ARM 은 1byte 연산 지원안함??
	{
		clnt_sock = accept(serv_sock, (struct sockaddr *)&clnt_addr,
&clnt_addr_size);//accept 사용자가 들어오면 실질적으로 승인해준다 . 클라이언트 전용 소켓이 만들어진다.
		if(clnt_sock == -1)
			err_handler("accept() error");
		else
		printf("Connected Client %d\n", i+1);//클라이언트 소켓이 -1 이 아니면 Connected Client
가 나온다. while 로 간다.

		while((str_len = read(clnt_sock, msg, BUF_SIZE)) != 0)//클라이언트 소켓에 있는 msg
를 읽어서 write 해서 에코해주고 있다 ?? read 가 블록이니까 끄게 되면 쭉 나오는데 read write 때문에 블로킹
되어있어 한번에 나온다. 논 블로킹이나 fork 로 한다.
		write(clnt_sock, msg, str_len);//msg 가 출력

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

} /*echo_server*/
```

```
/* 실행: gcc -o serv echo_server.c
gcc -c clnt echo_client.c
./serv 7777
./clnt <ip>
*/

#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 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]);
	}

	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(;;)
```

```
	{
		fputs("Input msg(q to quit): ",stdout);//메세지를 입력 받고 있다. fputs = write
		fgets(msg, BUF_SIZE,stdin);//fgets = read 랑 같다.

		if(!strcmp(msg,"q\n") || !strcmp(msg,"Q\n"))//메세지로 q 나 Q 를 입력했는지 본다 무한루프를 빠져나간다.
		break;

		write(sock, msg, strlen(msg));//fgets 에서 받은 값을 sock 에..
		str_len = read(sock, msg, BUF_SIZE - 1);//sock 에 msg 를 읽어서 서버에 보낸다.

		if(str_len == -1)
			err_handler("read() error!");
		msg[str_len] = 0;
		printf("msg from serv: %s\n", msg);
	}
	close(sock);
	return 0;
}
```

```
#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//operation 의 사이즈 4 개까지 허용하겠다.

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++)//왜 1 부터 ? 이미 result 0 번부터 받아서 opnum 은 처음 시작했을때 몇번
연산할건지 입력한 값.
                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[];

   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);//클라이언트 소켓에서 오퍼레이트 카운트 값을 가져온다. 카운트값은
맨 처음에 입력하는 값 (연산자) 5 개 쓰고 싶으면 5
	recv_len=0;

	while((opnd_cnt * OPSZ +1)>recv_len)//opsz 는 연산자 자체의 사이즈 (정수형이라 4) recv_len
이 0
	{
	  recv_cnt=read(clnt_sock,&opinfo[recv_len],BUF_SIZE-1);//클라이언트 소켓에서
opinfo[recv_len 근데 0] 하고 있다. 읽어서 0 번째에 값을 채워넣는다 ,사이즈는 버프사이즈 -1 , 쓴만큼 읽는다.
	  recv_len+=recv_cnt;//읽은 바이트수=(read 의 리턴값)만큼 recv_len 이 플러스 된다.
	}

	result=calculate(opnd_cnt, (int *)opinfo,opinfo[recv_len-1]);//opinfo 는 우리가 입력한 숫자가
값들이 들어가 있다. cal 함수 opnd_cnt 는 아까 리드한 몇개 작업할건지 ... 클라이언트에서 3 입려갱ㅆ으면 3,
recv-1 이면 마지막에 -1 을 배열에 넣겠다는거
	write(clnt_sock, (char *)&result, sizeof(result));//결과값을 클라이언트 소켓에 &니까 클라이언트
로 날아간다.
```

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

    return 0;
}
```

```
#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);

   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);//3 입력
   scanf("%d", &opnd_cnt);//opnd cnt 에 3 들어감
```

```
    opmsg[0]=(char)opnd_cnt;

    for(i=0;i<opnd_cnt;i++)
    {
        printf("operand %d: ",i+1);//숫자 값 입력
        scanf("%d",(int *)&opmsg[i*OPSZ+1]);//123 입력
    }


    fgetc(stdin);//문자 하나만 1 바이트 읽는다 + - * 인지
    fputs("Operator: ",stdout);
    scanf("%c",&opmsg[opnd_cnt*OPSZ+1]);
    write(sock,opmsg,opnd_cnt*OPSZ+2);//opmsg (숫자 집어넣은거 부호)  를 솤에 보낸다.
    read(sock,&result,RLT_SIZE);

    printf("Operation result: %d\n",result);
    close(sock);
}
```