TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 하성용
accept0108@naver.com

27 일차

inet_aton.c

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

//si 가 소켓
typedef struct sockaddr_in    si;

void err_handler(char *msg)
{
        while(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 intager Addr: %#x\n",
                                addr_inet.sin_addr.s_addr);

        return 0;
}
```

```
Network Ordered intager Addr: 0x1f497c7f
```

네트워크 주소를 바꾸는것

빅엔디안은 순서대로 저장됨
실제 변환과정은

호스트(로컬) 네트워크(=빅) 호스트(로컬)
리틀 → 빅 → 빅

빅 → 빅 → 리틀
주소를 어떻게 변환해줄지 결정

네트워크에서 변환할걸 모른다해도 호스트에서 자신이 무엇인지 알아서 변환함
들어오는건 빅으로 확정이라 빅을 자신에 맞게 변환함

그림
127.124.73.31 //리틀
4 3 2 1
1f 49 7c 7f //네트워크 주소로 바꾼것

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); //l 은 롱타입
        addr2.sin_addr.s_addr = htonl(0x12345678); //호스트엘로 크로스매칭

        str = inet_ntoa(addr1.sin_addr); //네트워크를 호스트로 변경하는것
        strcpy(str_arr, str);
        printf("Not 1: %s=n", str); //16 33 64 4? 가나옴

        //inet=네트워크 명시,_ntoa=network to adrress
        inet_ntoa(addr2.sin_addr); //addr 2 에 sub_addr
        printf("Not 2: %s\n", str);
        // addr2 에 값을 원상복구작업 (16+2=18. 34=52. 16x5=80 112+8=120)
        printf("Not 3: %s\n", str_arr);

        return 0;
}
```

```
Not 1: 16.32.48.64=nNot 2: 18.52.86.120
Not 3: 16.32.48.64
```

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() 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) //받을수있는 사람숫자 5
		err_handler("listen() error");

	clnt_addr_size = sizeof(clnt_addr);

	for(i=0; i<5; i++)
	{
		clnt_sock = accept(serv_sock, (struct sockaddr *)&clnt_addr,
					&clnt_addr_size); /*엑셉은 승인해주는거, 전용소켓
만들어지고 클라이언트소켓이 0 이아니라면 while 로가서 read 하고 read 하면서 clinet 에 들어온걸
읽으면서 write 에 echo 해주고있음
즉 끝날일없음*/

/* 클라이언트가 끝나면 메시지가 주루룩나오는데 그이유는 처리가안되고 블로킹되었기때문에 동시에
처리되는거
이부분에 문제가 생기지않으려면 논블로킹으로 만들어줘야함
1:1 통신은 read-write-read-write 로 해도 상관없음
2 개 만들어두면 (ex.fork) 상대가 읽는거하나 본인이 읽는거하나 쓰는것도 하나씩만들면됨
이걸해결하려면 포크를 많이만들거나 논블로킹으로만들면됨 */


		if(clnt_sock == -1)
			err_handler("accept() error");
		else
			printf("Connected Client %d\n", i+1);

		while((str_len = read(clnt_sock, msg, BUF_SIZE)) !=0)
			write(clnt_sock, msg, str_len);

		close(clnt_sock);
	}
	close(serv_sock);

	return 0;
}
```

서버만든후 클라이언트 만들기

echo_client.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, stdin 은 fd 오는 자리
		fgets(msg, BUF_SIZE, stdin);
```

```
                        if(!strcmp(msg, "q\n") || !strcmp(msg, "Q\n"))
                        //메세지 q 나 Q 를 입력했는지, 입력했으면 무한루프 빠져나옴
                                break;

                        write(sock, msg, strlen(msg)); //read 한걸 write 해서 보내줌
                        str_len = read(sock, msg, BUF_SIZE -1); // 다시 읽어옴

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

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

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

컴파일 하는방법
gcc -o serv echo_server.c
gcc -o clnt echo_client.c
./serv 7777
./clnt [ip] 7777
ip // 127.0.0.1

서로채팅은 안되고 서로 에코만되는 상태

네트워크로 계산기를 구현하는것

오포레이션의 사이즈
사이크즈기가 4 개라는소리

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)
{
        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;

                        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]);
                        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);
        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;
}

//네트워크로 게임을 구동시킬수있음


//서버쪽도 동일함
//for 문안에서 read
//read 로 클라이언트 소켓에서 오퍼레이트 카운트값을 가져오고있음
//연산자 3 개를 쓰려면 3
//opnd_cnt 가 가져오게될것

//while 문
// recv_len 0 보다 크다
//클라이언트 소켓에서 읽어서 0 번째에 값을 채워넣는거, 버프사이즈(1024) -1
//쓴만큼만 읽는다. 는 3 개

//읽은 바이트수만큼 리시브렌이 +됨

//중간에 노드가 끊기면 데이터가 끊어지는걸 대비해서 와일루프를 걸어둠

//리시브렌이 오퍼랜드 cnt 보다 크다면 무언가 문제가있다는것으로 인지하고 조치

//캐릭터형이 오프인즈사이즈만큼 되어있음


//계산기
//opnd_cnt 3 을 입력해서 3 이 들어가있고
```

//배열의 시작은 0 이기에 마지막문자에는 -1 이들어감

//1+2+3 = 6 리턴해서 결과값나옴 리턴한 결과는 result 에 저장됨
//클라이언트 소켓에 해주는데 =리줄트의 결과값이 날아감,

//opnd_cnt 에는 3 들어가잇음
//for 문돌면서 (숫자입력했다고 가정)
//read 해서 1byte 만큼 받음

//write 를 했으니까 서버에 날아가고
//read 하면 클라이언트한테 서버가 write 했던결과가 돌아옴
//이번엔 소켓을 통해서 result 로 들어오고
//result 에서 출력됨

* 술게임의 네트워크화!
횟수 제한 x
1~3333
시간제한 3 초
횟수 카운트

game1_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              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, nread;
        char buf[BUF_SIZE] = {0};
        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 ......");

	for(;;)
	{
		nread = read(sock, buf, BUF_SIZE);
		write(1, buf, nread);
	}

	close(sock);

#if 0
	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);
#endif

	return 0;
}
```

game1_serv.c

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

/* For System */
```

```
#include <time.h>
#include <fcntl.h>
#include <signal.h>
#include <stdbool.h>
#include <sys/wait.h>

typedef struct sockaddr_in	si;
typedef struct sockaddr *	sap;

#define BUF_SIZE		1024
#define OPSZ				4

int glob_cnt;

void sig_handler(int signo)
{
	printf("Time Over\n");
	glob_cnt++;
}

void make_game(int *data)
{
	*data = rand() % 3333 + 1;
}

bool check_correct(int data, int cmp)
{
	if(data == cmp)
		return true;
	else
		return false;
}

void start_game(int data)
{
	char buf[32] = {0};
	bool fin;
	int i, cmp;

	for(;;)
	{
		signal(SIGALRM, sig_handler);
		alarm(1);
		read(0, buf, sizeof(buf));
		alarm(0);
		cmp = atoi(buf);

		fin = check_correct(data, cmp);

		if(fin)
		{
			break;
		}
		else
		{
			glob_cnt++;
			if(data > cmp)
				printf("%d 보다 크다\n", cmp);
			else
				printf("%d 보다 작다\n", cmp);
```

```
                }
        }
}

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)
{
        pid_t pid[5] = {0}; //5 개받기
        int status;

        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);

	//signal();

	for(i = 0; i < 5; i++)
	{
		pid[i] = fork(); //pid 배열에 포크받기, 자식프로세스입장에선 자식 pid 값은 전부 0

		if(pid[i] > 0) //기다려.부모
			wait(&status);
		else
		{
			int data;
			char buf[32] = "숫자를 맞춰봐!\n";

			srand(time(NULL));
			clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size); //엑센트해서 클라이언트 연결됨, 여기다 쓰면 클라이언트한테 감
			make_game(&data); //데이터 난수만듬

			for(;;) //무한루프
			{
				write(clnt_sock, buf, strlen(buf)); //clnt_sock 에 쓰고있음, 클라이언트에 쏘고있고 클라이언트는 리드해야 읽을수있음
				glob_cnt++;
				if(glob_cnt > 10)
					break;
			}

			close(clnt_sock);
#if 0
			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);
#endif
                        }

        }
#if 0
        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);
        }
#endif

        close(serv_sock);

        return 0;
}


// sig handler(int signo) 횟수카운트
//sig handler 가 끝날때 알람을 넣어줌
```