TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 - 이상훈
gcccompil3r@gmail.com
학생 – 이우석
colre99@naver.com

[4/2(월) - 28일차]

[file_server.c]

```
#include <fcntl.h>
#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;       → sockaddr_in 의 구조체, 변수명: si
typedef struct sockaddr * sap;       → sockaddr *의 구조체, 변수명: sap

#define BUF_SIZE  32

void err_handler(char *msg)
{
        fputs(msg, stderr);    → fputs()함수는 string 을 현재위치의 출력 stream 으로 복사한다.
```

```
        fputc('\n', stderr);
        exit(1);
}
```

String 끝의 null 문자(\0)를복사하지 않는다.

→ fputc()함수는 c 를 unsigned char 로 변환한 다음 c 를 현재위치의 출력 stream 으로 작성하며 그에 맞게 파일 위치를 변경한다. stream 이 추가 모드중 하나로 열리면, 문자는 stream 의 끝에추가.

```
int main(int argc, char **argv)
{
        int serv_sock, clnt_sock, fd;
        char buf[BUF_SIZE] = {0};
        int read_cnt;

        si serv_addr, clnt_addr;
        socklen_t clnt_addr_size;

        if(argc != 2)
        {
                printf("use: %s <port>\n", argv[0]);
                exit(1);
        }

        fd = open("file_server.c", O_RDONLY);
        serv_sock = socket(PF_INET, SOCK_STREAM, 0);

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

→ 구조체 si 에 serv_addr 와 clnt_addr 을 넣는다

→file_server 를 열면서 읽기전용으로.

```
memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;      → 주소 체계로.
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);  → 어떤 아이피 주소든 받겠다
serv_addr.sin_port = htons(atoi(argv[1]));    → 어떤 서비스 번호를 열것인가.

if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1) → socket 에 IP 주소와 port 번호를 지정.
        err_handler("bind() error");

if(listen(serv_sock, 5) == -1)  → 몇명까지 받을것인가 지정.
        err_handler("listen() error");

clnt_addr_size = sizeof(clnt_addr);  → client 의 주소 사이즈 지정

clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size);
                → client 접속대기 및 허락. 동시에 socket 생성 . 이로서 server  와 client 사이에 통신이 가능하다
for(;;)
{
        read_cnt = read(fd, buf, BUF_SIZE);

        if(read_cnt < BUF_SIZE)
        {
                write(clnt_sock, buf, read_cnt);
                break;
        }
```

```
        write(clnt_sock, buf, BUF_SIZE);
    }

    shutdown(clnt_sock, SHUT_WR);
    read(clnt_sock, buf, BUF_SIZE);
    printf("msg from client: %s\n", buf);

    close(fd);
    close(clnt_sock);
    close(serv_sock);

    return 0;

}
```

→ shutdown 함수는 close 와 마찬가지로 소켓을 종료하지만 두번째 매개변수인 how_to 에 어떤값을 넣느냐에 따라 read buffer 와 write buffer 를 차단할지 선택할 수 있다. 옆의경우, SHUT_WR 의 경우는 더이상 해당 socket 에게 송신을 할 수 없다.

[file_client.c]

```
#include <fcntl.h>
#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   32

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

int main(int argc, char **argv)
{
        char buf[BUF_SIZE] = {0};
```

```
int fd, sock, read_cnt;
si serv_addr;

if(argc != 3)
{
        printf("use: %s <IP> <port>\n", argv[0]);
        exit(1);
}

fd = open("receive.txt", O_CREAT | O_WRONLY);
sock = socket(PF_INET, SOCK_STREAM, 0);
```

→ read 만 볼수 있고, write 면 볼수없다.
그럼 chmod 0644 receive.txt 해준다.
권한변경으로 vi receive.txt 다시열면 볼수있다.

```
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.....");
```

```
        while((read_cnt = read(sock, buf, BUF_SIZE)) != 0)          →  read – write 반복. Blocking 되어있다.
                write(fd, buf, read_cnt);

        puts("Received File Data");      → 표준 출력 파일 스트림에 문자열을 옆과 같이 출력
        write(sock, "Thank you", 10);    → socket 에 10 크기로 옆과같이 write 해준다
        close(fd);다
        close(sock);

        return 0;

}
```

[실행방법]:

file_server.c , file_client 두 코딩을 컴파일 해주고 (gcc -o file_serv / clnt), 터미널 하나에서는 서버를 열어주기 위해 ./file_serv 7777 명령어를 입력해주고, 다른 터미널 창를 하나 더 띄어서 ./file_clnt 127.0.0.1 7777 을 입력해준다.
자신의 아이피를 이용함으로써 두 개가 정상작동 하는지 알아볼 수 있다.

결과값으로 server 를 연 터미널에선 ‘msg from client: Thank you’가 출력됨을 볼 수 있다. 다른 터미널창인 client 를 연결해준 곳에서는 ‘Connected…….. Received File Data’ 가 출력됨 또한 볼 수 있다.

결과값:

[gethostbyname.c]

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

void err_handler(char *msg)
{
        fputs(msg, stderr); → string 을 현재위치에 출력 stream 으로 복사
```

```
        fputc('\n', stderr);   → unsigned char 로 변환한 다음 c 를 현재위치의 출력 stream 으로 작성. 파일위치 변경
        exit(1);
}


int main(int argc, char **argv)
{
        int i;
        struct hostent *host;   →   pointer host

        if(argc != 2)
        {
                printf("use: %s <port>\n", argv[0]);
                exit(1);
        }

        host = gethostbyname(argv[1]);       → hostent 구조체 구하기

        if(!host)
                err_handler("gethost ... error!");

        printf("Official Name: %s\n", host->h_name);      → 호스트 이름출력

        for(i = 0; host->h_aliases[i]; i++)   → 별칭 or 가명
                printf("Aliases %d: %s\n", i + 1, host→h_aliases[i]);   →호스트 별명 출력
```

```
    printf("Address Type: %s\n", (host->h_addrtype == AF_INET) ? "AF_INET" : "AF_INET6");
        → IP 타입 알아보는것.                호스트 주소타입 출력

    for(i = 0; host->h_addr_list[i]; i++)
        printf("IP Addr %d: %s\n", i + 1, inet_ntoa(*(struct in_addr *)host->h_addr_list[i]));
          → IP web port number 80   호스트주소를 dotted-dec

    return 0;

}
```

위의 코딩을 컴파일을 한 후, 실행파일 a.out 과 함께 어느 사이트를 함께 입력해주면 출력값은 아래와 같이 나오게 된다.

결과값:

```
ws91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ ./a.out naver.com
Official Name: naver.com
Address Type: AF_INET
IP Addr 1: 125.209.222.141
IP Addr 2: 210.89.160.88
IP Addr 3: 125.209.222.142
IP Addr 4: 210.89.164.90
ws91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ ./a.out google.com
Official Name: google.com
Address Type: AF_INET
IP Addr 1: 216.58.220.206
```

[gethostbyname() 함수}

:

gethostbyname()함수는 주어진 호스트 name 에 상응하는 hostent 타입의 구조체를 반환한다. name 은 호스트이름이거나 표준 점 표기법의 IPv4 주소, 혹은 IPv6 주소이다. 만일 name 이 IPv4 나 IPv6 주소라면 어떠한 찾기도 행해지지 않고 단지 name 을 h_name 필드에 복사하고 해당 in_addr 구조체를 반환하는 hostent 구조체의 h_addr_list[0]에 복사한다. 만일 name 이 점으로 끝나지 않고 환경변수 HOSTALIASES 가 설정되어 있다면 HOSTALIASES 가 가리키는 별칭 파일이 name 을 위해 우선 탐색된다. 현재 도메인과 상위 도메인은 name 이 이 점으로 끝나지 않는다면 탐색된다.

gethostbyname(), 그리고 gethostbyaddr() 함수는 hostent 구조체를 반환하거나 만일 에러가 발생한다면 NULL 포인터를 반환한다. 에러시, h_errno 변수는 에러 넘버를 가진다.