TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 최대성
c3d4s19@naver.com

2018.04.02. – 028일차 수업

---

* gethostbyname()함수

struct hostent* gethostbyname(const char* [name]);

[name] -> 호스트 이름이거나 표준 점 표기법의 IPv4 주소, 콜론(그리고 점 표기법도 가능)표기법의 IPv6

주어진 호스트 [name] 에 상응하는 hostent 타입의 구조체를 반환한다

```
struct hostent
  {
    char *h_name;    // Official name of host.
    char **h_aliases;     // Alias list.
    int h_addrtype;       // Host address type.
    int h_length;         // Length of address.
    char **h_addr_list;   // List of addresses
from name server.
#define h_addr h_addr_list[0] // Address, for
backward compatibility.
    };
```

gethostbyname()함수 사용 예제

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

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

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

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

    //호스트가 반환
    host = gethostbyname(argv[1]);

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

    printf("Official Name: %s₩n", host->h_name);

//별칭 출력(없을수도 있음)
    for(i = 0; host->h_aliases[i]; i++)
  printf("Aliases %d: %s₩n", i + 1, host->h_aliases[i]);

    printf("Address Type: %s₩n",
            (host->h_addrtype == AF_INET) ?
            "AF_INET" : "AF_INET6");

    //IP주소 프린트
    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]));

    return 0;
}
```

서버에서 클라이언트로 파일 보내기

<file_server.c 파일>

```
#include <stdio.h>
#include <fcntl.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){
    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);
    }
// 첫번째 인자: 보낼 파일
    fd = open("file_server.c", O_RDONLY);
    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);

    clnt_sock = accept(serv_sock, (sap)&clnt_addr,
&clnt_addr_size);

    for(;;){
        read_cnt = read(fd, buf, BUF_SIZE);

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

<file_client.c 파일>

```
#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);
//받을 파일
    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 ...");

    while((read_cnt = read(sock, buf,
BUF_SIZE)) != 0)
        write(fd, buf, read_cnt);

    puts("Received File Data");
    write(sock, "Thank you", 10);
    close(fd);
    close(sock);
    return 0;
}
```

close() 함수와 shutdown() 함수의 차이점

int close(int socket_fd);

-> 소켓을 종료하면서 동시에 recv buffer와
send buffer를 차단하기 때문에
close함수를 통해 소켓이 종료되고 난 시점
이후 부터는 더이상 통신을 주고받을 수가 없다.

int shutdown(int [socket_fd], int [how_to]);

-> shutdown 함수는 close와 마찬가지로 소켓을
종료하지만 두번째 매개변수인 [how_to]에 어떤 값을
넣느냐에 따라 read buffer와 write buffer를 차단할지
선택할 수 있다.

[how_to]

-> SHUT_RD: recv buffer만 차단한다.
SHUT_WR: write buffer만 차단한다.
SHUT_RDWD: recv buffer, write buffer 둘다 차단한다.

참고자료
http://eastroot1590.tistory.com/entry/%EC%86%8C%EC%BC%93-%EC%A2%85%EB%A3%8C-%EB%AA%85%EB%A0%B9-close%EC%99%80-shutdown

---

* <pthread.h>헤더로 Thread 사용

pthread_mutex_t

-> 뮤텍스(Mutex) : lock의 key값 (세마포어와 비슷함)

Mutext 변수는 사용하기 전에 반드시 초기화를 해주어야 한다.
-> 초기화 하는 방법 2가지
1. Mutex변수에 PTHREAD_MUTEX_INITIALIZER를 세팅
2. pthread_mutex_init 함수를 사용

int pthread_mutex_lock(pthread_mutex_t *mutex);
-> 한 스레드가 작업중 다른 모든 스레드가 공유자원(전역변수)에 접근하지 못하도록 lock을 걸어서 critical section을 보호 (뮤텍스 잠금 blocking)

int pthread_mutex_trylock(pthread_mutex_t *mutex);
-> 뮤텍스 잠금 error 보냄

int pthread_mutex_destory(pthread_mutex_t *mutex);
-> 동적 선언된 뮤텍스 free()하기 전에 clear시키는 함수

int pthread_mutex_unlock(pthread_mutex_t *mutex);
-> 뮤텍스 잠금 해제

참고 링크
http://xucxo.blogspot.kr/2011/03/linux-programming-thread-mutex.html

---

int pthread_create(pthread_t* [thread],
const pthread_attr_t* [attr],
void *(*[start_routine])(void *), void* [arg]);
-> (쓰레드 생성만 하고 작동시키지는 않음)
[thread]
-> 쓰레드가 성공적으로 생성되었을때 생성된 쓰레드를 식별하기 위해서 사용되는 쓰레드 식별자(Thread id)이다.
[attr]
-> 쓰레드 특성을 지정하기 위해서 사용하며, 기본 쓰레드 특성을 이용하고자 할경우에 NULL 을 사용한다.
[start_routine]
-> 분기시켜서 실행할 쓰레드 함수
[arg]
-> 위 [start_routine] 쓰레드 함수의 매개변수(쓰레드에 전달되는 인자)로 넘겨짐
(여러 인자를 넘기고 싶으면 구조체 이용)

int pthread_detach(pthread_t [th]);
-> 프로세스와 쓰레드를 분리시킨다
(쓰레드가 종료되면 자동으로 모든 자원을 해제한다)
[th] -> 분리시킬 쓰레드 식별자(id)

int pthread_join(pthread_t [th], void** [thread_return]);
[th] -> 기다릴 쓰레드의 식별자(id)
[thread_return] -> 해당 쓰레드의 리턴값이 NULL이 아닌경우 값을 받아올수 있음
(생성하기만 한 쓰레드는 join 하는 순간 동작 함)

참고 자료: http://bitsoul.tistory.com/157

쓰레드 사용한 숫자 맞추기 게임

<gserv.c 파일>

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <pthread.h>
#include <arpa/inet.h>
#include <sys/socket.h>
```

```c
#define BUF_SIZE 128
#define MAX_CLNT 256

typedef struct sockaddr_in              si;
typedef struct sockaddr *       sp;

int clnt_cnt = 0;
int clnt_socks[MAX_CLNT];
int data[MAX_CLNT];
int thread_pid[MAX_CLNT];
int idx;
int cnt[MAX_CLNT];
pthread_mutex_t mtx;

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

void sig_handler(int signo)
{
        int i;
        printf("Time Over!₩n");
        pthread_mutex_lock(&mtx);
        for(i = 0; i < clnt_cnt; i++)
                if(thread_pid[i] == getpid())
                        cnt[i] += 1;

        pthread_mutex_unlock(&mtx);
        alarm(3);
}


//메세지 보내는 함수
void proc_msg(char *msg, int len, int k)
{
        int i;
        int cmp = atoi(msg);
        char smsg[64] = {0};

        pthread_mutex_lock(&mtx);

        cnt[k] += 1;

        if(data[k] > cmp)
          sprintf(smsg, "greater than %d₩n", cmp);
        else if(data[k] < cmp)
          sprintf(smsg, "less than %d₩n", cmp);
        else
        {
                strcpy(smsg, "You win!₩n");
                printf("cnt = %d₩n", cnt[k]);
        }
        strcat(smsg, "Input Number: ₩n");
        write(clnt_socks[k], smsg, strlen(smsg));

        pthread_mutex_unlock(&mtx);
}
```

```c
//클라이언트 쓰레드 함수
void *clnt_handler(void *arg)
{
        int clnt_sock = *((int *)arg);
        int str_len = 0, i;
        char msg[BUF_SIZE] = {0};
        char pattern[BUF_SIZE] = "Input Num: ₩n";

        signal(SIGALRM, sig_handler);
//lock을 걸어서 critical section을 보호
        pthread_mutex_lock(&mtx);
        thread_pid[idx++] = getpid();
        i = idx - 1;
        printf("i = %d₩n", i);
        write(clnt_socks[i], pattern,
                strlen(pattern));
        pthread_mutex_unlock(&mtx);

        alarm(3);

        while((str_len = read(clnt_sock, msg,
                sizeof(msg))) != 0)
        {
                alarm(0);
                proc_msg(msg, str_len, i);
                alarm(3);
        }

        pthread_mutex_lock(&mtx);

        for(i = 0; i < clnt_cnt; i++)
        {
            if(clnt_sock == clnt_socks[i])
            {
                while(i++ < clnt_cnt - 1)
                 clnt_socks[i]= clnt_socks[i + 1];
                 break;
                }
        }
        clnt_cnt--;
        pthread_mutex_unlock(&mtx);
        close(clnt_sock);

        return NULL;
}

int main(int argc, char **argv)
{
        int serv_sock, clnt_sock;
        si serv_addr, clnt_addr;
        socklen_t addr_size;
        pthread_t t_id;
        int idx = 0;

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

```
        srand(time(NULL));

        pthread_mutex_init(&mtx, NULL);

      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, (sp)&serv_addr,
                    sizeof(serv_addr)) == -1)
                err_handler("bind() error");

        if(listen(serv_sock, 2) == -1)
                err_handler("listen() error");

        for(;;)
        {
            addr_size = sizeof(clnt_addr);
            clnt_sock = accept(serv_sock,
              (sp)&clnt_addr, &addr_size);
            //다음 클라이언트가 올때까지 blocking

            thread_pid[idx++] = getpid();

            pthread_mutex_lock(&mtx);
        //스레드가 작업중 다른 모든 스레드가
스레드 공유자원(전역변수)에 접근하지 못하도록 lock
            data[clnt_cnt] = rand() % 3333 + 1;
            clnt_socks[clnt_cnt++] = clnt_sock;
            pthread_mutex_unlock(&mtx);//lock 해제

            pthread_create(&t_id, NULL,
              clnt_handler, (void *)&clnt_sock);
            //4번째 인자: 쓰레드에 전달되는 인자
            pthread_detach(t_id);
                // t_id -> 쓰레드의 id값
            printf("Connected Client IP: %s₩n",
inet_ntoa(clnt_addr.sin_addr));
        }

        close(serv_sock);

        return 0;
}
```

<gclnt.c 파일>

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/epoll.h>

#define BUF_SIZE          128

typedef struct sockaddr_in        si;
typedef struct sockaddr *sp;

char msg[BUF_SIZE];

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

void *send_msg(void *arg)
{
        int sock = *( (int*)arg );
        char msg[BUF_SIZE];

        for(;;)
        {
                fgets(msg, BUF_SIZE, stdin);
//읽을값 없으면 blocking
                write(sock, msg, strlen(msg));
        }

        return NULL;
}

void *recv_msg(void *arg)
{
        int sock = *((int *)arg);
        char msg[BUF_SIZE];
        int str_len;

        for(;;)
        {
                str_len = read(sock, msg,
BUF_SIZE - 1);   //읽을값 없으면 blocking

                msg[str_len] = 0;
                fputs(msg, stdout);
        }

        return NULL;
}

int main(int argc, char **argv)
{
        int sock;
        si serv_addr;
```

```
        pthread_t snd_thread, rcv_thread;
        void *thread_ret;

        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, (sp)&serv_addr,
sizeof(serv_addr)) == -1)
                err_handler("connect() error");

        //쓰레드 만든 이유 -> 통신과 수신을
분리하기 위해
        pthread_create(&snd_thread, NULL,
send_msg, (void *)&sock);
        pthread_create(&rcv_thread, NULL,
recv_msg, (void *)&sock);
        pthread_join(snd_thread, &thread_ret);

        pthread_join(rcv_thread, &thread_ret);

        close(sock);

        return 0;
}
```