

# Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

강사 – Innova Lee( 이상훈 )

[gcccompil3r@gmail.com](mailto:gcccompil3r@gmail.com)

학생 – 장성환

[redmk1025@gmail.com](mailto:redmk1025@gmail.com)

\*gserv.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>
```

```
#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)
```

쓰레드를 이용하여 게임 서버를 구축한 서버이다.

```
{
    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);
}

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; //몇번 입력을 했는지 카운트를 해주는 변수이다.

    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]); //사실 여기서 클라도 종료할 수 있도록 하면
    좋다.
}

strcat(smsg, "Input Number: \n"); //해당 문자열을 스트링에 갖다가 붙인다.
write(clnt_socks[k], smsg, strlen(smsg));
pthread_mutex_unlock(&mtx);
}

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 Number: \n";

    signal(SIGALRM, sig_handler);

    pthread_mutex_lock(&mtx);
    thread_pid[idx++] = getpid(); //스레드의 pid 값을 저장 (이게 진짜 pid 값이
    다.)
    i = idx - 1;
    printf("i = %d\n", i);
    write(clnt_socks[i], pattern, strlen(pattern)); //첫번째 클라에 패턴을 써주겠
    다. idx 는 0 부터 올라감.
    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; //스레드 아이디 값
    int idx = 0;

```

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

srand(time(NULL));

pthread_mutex_init(&mtx, NULL); //초기는 그냥 널로 세팅

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); //클라
의 커넥트를 승인해준다. (다음 클라 올때까지 블록킹)
```

```

thread_pid[idx++] = getpid();
//pthread 에 getpid()를 통하여 현재 프로세스 pid 를 저장(필요없어 보임)

pthread_mutex_lock(&mtx); //mutex 자원을 잠금 (공유 데이터의 꼬임
현상을 방지한다.)
data[clnt_cnt] = rand() % 3333 + 1; //data 에 각각의 랜덤값을 저장
clnt_socks[clnt_cnt++] = clnt_sock; //clnt_socks 에 accept 한 각각의
클라소켓을 저장
pthread_mutex_unlock(&mtx); //mutex 자원 잠금을 해제한다.

pthread_create(&t_id, NULL, clnt_handler, (void *)&clnt_sock);
// pthread_t t_id 전달 clnt_handler 함수 실행, 함수의 인자로 클라소켓
주소값 전달
pthread_detach(t_id);
// 쓰레드를 프로세스에서 독립 시킨다. (별도로 동작하기 시작한다.)
printf("Connected Client IP: %s\n", inet_ntoa(clnt_addr.sin_addr));
//해당 클라이언트의 주소값을 보여줌

/*
for(int i=0; i<idx ; i++){
    printf("%d\n",thread_pid[idx]); // 다 0 으로 저장되어 있음 될까?
thread_pid 가 쓰인 까닭?
}
*/
}

close(serv_sock);

return 0;
}

```

\* gclnt.c

```
#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);
        write(sock, msg, strlen(msg));
    }
}
```

쓰레드를 이용하여 입력과 출력을 분리시켜 작동하는 게임 서버의 클라이언트이다.



```

    }

    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);

        msg[str_len] = 0; //긴게 오고 짧은게 오면 앞에 것만 들어가서 의
        //도치 않는 값이 들어올 수 있다.
        fputs(msg, stdout);
    }

    return NULL;
}

int main(int argc, char **argv)
{
    int sock;
    struct 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");

pthread_create(&snd_thread, NULL, send_msg, (void *)&sock);
pthread_create(&rcv_thread, NULL, recv_msg, (void *)&sock); //
송신과 수신을 분리해주는 것이다.
pthread_join(snd_thread, &thread_ret);
pthread_join(rcv_thread, &thread_ret); //조인하는 순간 쓰레드가 죽어
야 close 한다. (조인 할때 쓰레드가 구동한다.)

close(sock);

return 0;
}
```

```
* 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;
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);
    }
```

파일 송신 서버

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

if(serv_sock == -1)
    err_handler("socket() err");

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);

while(1){
    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;
```

```
}
```

\* file\_client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <fcntl.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];
    int fd, sock, read_cnt;
    si serv_addr;
```

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

파일 수신 클라이언트

```
fd = open("receive.c", O_WRONLY | O_CREAT | O_TRUNC, 0644);
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;
}
```

```
* 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);
    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);
    }
    host = gethostbyname(argv[1]);

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

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

    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) ?
```

```
sunghwan@HWAN:~/Documents/network$ ./a.out naver.com
Official Name: naver.com
Address Type: AF_INET
IP Addr 1: 210.89.160.88
IP Addr 2: 210.89.164.90
IP Addr 3: 125.209.222.141
IP Addr 4: 125.209.222.142
```

IP addr 확인



"AF\_INET" : "AF\_INET6"); //ip4 or ip6 확인

```
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 addr 출력(실제로 웹브라우저에 출력된 ip 주소이다. 포트 번호는 80 번이다 (인터넷 용으로 고정된 포트))
```

```
return 0;  
}
```

시험문제> 2 진 트리에 스레드를 생성하여 트리를 오른쪽 왼쪽 검색을 구현하라.

1. pthread 활용법
2. 네트워크 프로그래밍 기본기
3. 기타정리
4. 만약 시간이 남는다면, 3 6 9 게임을 만든다. (박수는 ctrl +c 을 사용한다. 박수 2 번은 두번 시그널) (+시간제한 2 초) - 목요일 까지

\*th2.c

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
```

```
pthread_t threadA, threadB;
```

```
typedef struct _thval{
    int a;
    pthread_t t_pid;
}thread_data;
```

```
void* task1(void *data){
    int tmp = ((thread_data*)data)->a;
    tmp++;

    ((thread_data*)data)->t_pid = threadA;

    printf("thread 1 start!\n");
    printf("int :%d\n",tmp);
    printf("t_pid :%lu\n",threadA);
    printf("thread 1 end!\n");

    return NULL;
}
```

```
void* task2(void *data){
    int tmp = ((thread_data*)data)->a;
```

pthread threadA 와 thread B 의 값이 각각 다른것을 확인할 수 있다.

```
sunghwan@HWAN:~/Documents/linux/tmp$ ./a.out
0
thread 1 start!
int :1
t_pid :139637420353280
thread 1 end!
thread 2 start!
int :1
t_pid :139637411960576
thread 2 end!
```

각각의 pthreadA 와 pthreadB 의 값이

pthread\_create 이후에 특정 값이 들어간 것을 확인 가능하다.

```
tmp++;

((thread_data*)data)->t_pid = threadB;

printf("thread 2 start!\n");
printf("int :%d\n",tmp);
printf("t_pid :%lu\n",threadB);
printf("thread 2 end!\n");

return NULL;
}

int main (void){

    thread_data data;
    data.a = 0;

    printf("%d\n",data.a);

    pthread_create(&threadA,NULL,task1,(void*)&data);
    pthread_detach(threadA);

    pthread_create(&threadB,NULL,task2,(void*)&data);
    pthread_detach(threadB);

    pause();
    return 0;
}
```

\*th3.c

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
```

```
pthread_t threadA, threadB;
```

```
typedef struct _thval{
    int a;
    pthread_t tid;
}thread_data;
```

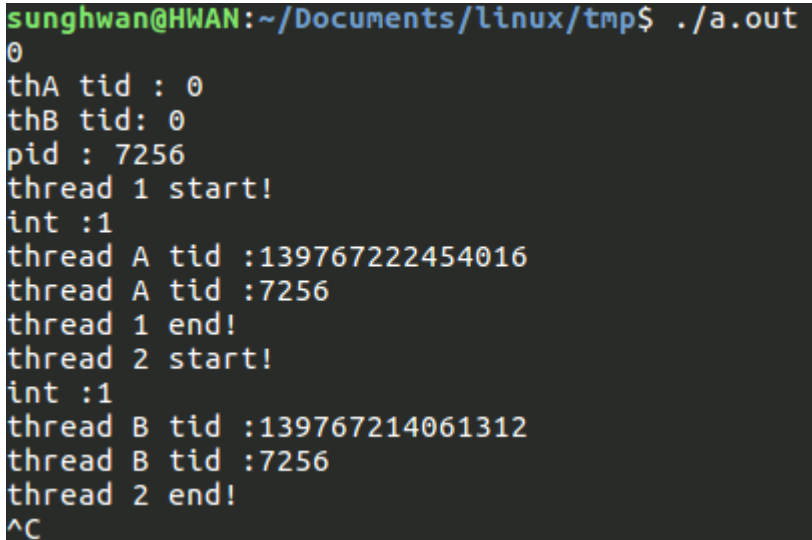
```
void* task1(void *data){
    int tmp = ((thread_data*)data)->a;
    tmp++;
```

```
// ((thread_data*)data)->t_pid = threadA;
```

```
printf("thread 1 start!\n");
printf("int :%d\n",tmp);
printf("thread A tid :%lu\n",threadA);
threadA = getpid();
printf("thread A tid :%lu\n",threadA);
printf("thread 1 end!\n");
```

```
return NULL;
}
```

getpid()로 thread\_t tid 의 값을 변경시켜 보았다.

A terminal window with a dark background and light green text. The prompt is 'sunghwan@HWAN:~/Documents/linux/tmp\$'. The command './a.out' has been executed. The output is as follows:  
0  
thA tid : 0  
thB tid: 0  
pid : 7256  
thread 1 start!  
int :1  
thread A tid :139767222454016  
thread A tid :7256  
thread 1 end!  
thread 2 start!  
int :1  
thread B tid :139767214061312  
thread B tid :7256  
thread 2 end!  
^C

pthread\_create 하기전 쓰레드 식별자는 0

하지만 이후에는 특정 값이 들어있다.

getpid()로 각각의 쓰레드 함수를 돌때, pthread\_t tid 에 해당 프로세스 값을 넣어 보았으나, 값은 동일했다.

??

```
void* task2(void *data){
    int tmp = ((thread_data*)data)->a;
    tmp++;

    //((thread_data*)data)->t_pid = threadB;

    printf("thread 2 start!\n");
    printf("int :%d\n",tmp);
    printf("thread B tid :%lu\n",threadB);
    threadB = getpid();
    printf("thread B tid :%lu\n",threadB);
    printf("thread 2 end!\n");

    return NULL;
}

int main (void){

    //thread_data *data = (thread_data*)malloc(sizeof(thread_data));
    thread_data data;
    data.a = 0;

    printf("%d\n",data.a);
    printf("thA tid : %lu\nthB tid: %lu\n",threadA,threadB);
    printf("pid : %d\n",getpid());

    pthread_create(&threadA,NULL,task1,(void*)&data);
    pthread_detach(threadA);
```

```
pthread_create(&threadB,NULL,task2,(void*)&data);  
pthread_detach(threadB);
```

```
pause();
```

```
return 0;
```

```
}
```