

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

**<리눅스 네트워크 프로그래밍>
2018.04.02 - 28 일차**

강사 - 이상훈
gcccompil3r@gmail.com

학생 - 안상재
sangjae2015@naver.com

* 소스코드

1) 서버 프로세스에서 임의의 숫자를 생성하고, 클라이언트 프로세스에서 숫자를 입력해서 맞추는 게임
- 한 프로세스에서 여러 개의 스레드를 만들면, 멀티 스레딩을 하게 된다. 그렇게 되면 여러 스레드들이 데이터를 동시에 공유하게 되므로 데이터의 의도치 않은 결과 값을 얻게 될 수 있다. 그러므로 임계구역(데이터를 공유할 가능성이 있는 부분)에 들어갈 때는 락을 걸고, 임계구역에서 나올 때는 락을 풀어줌으로써 이 문제를 해결할 수 있다.

<서버 프로세스>

```
#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)    // alarm 의 시간이 다 되면 sig_handler 수행
{
    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);    // 임계구역에 들어가기전에 뮤텍스 락을 걸어줌
```

```

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);    // 임계구역에서 나오면 뮤텍스 락을 풀어줌
}

void *clnt_handler(void *arg)    // 스레드 함수 부분
{
    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);    // alarm 시그널 등록

    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) // 소켓 파일에서 데이터를 모두 read 할 때까지 반복
    {
        alarm(0);    // alarm 종료
        proc_msg(msg, str_len, i);
        alarm(3);    // 3 초 alarm 시작
    }

    pthread_mutex_lock(&mtx);    /*blocking(mutex 잠금을 선점한 스레드가 있다면, 선점한 스레드가 mutex 잠
                                금을 되돌려 주기전까지 대기) */
    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;
}

```

```
////////////////////////////////////
<클라이언트 프로세스>
#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)    // 어떤 구조체가 들어올지 몰라서 VOID *로 인자를 받음
{
    int sock = *((int *)arg);
    char msg[BUF_SIZE];

    for(;;)
    {
        fgets(msg, BUF_SIZE, stdin); // 사용자 입력받고 서버소켓에 메시지 전송
        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);

        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");

pthread_create(&snd_thread, NULL, send_msg, (void *)&sock); // 송신과 수신을 분리
pthread_create(&rcv_thread, NULL, rcv_msg, (void *)&sock); /* 4 번째 인자가 thread 가 구동시키는
                                                             함수의 인자가 됨*/

pthread_join(snd_thread, &thread_ret); // snd_thread 가 종료될 때까지 대기함(blocking)
pthread_join(rcv_thread, &thread_ret);

close(sock);

return 0;
}

```

1-1) 결과 분석

```

sangjaehn@sangjaehn-900X5N:~/cod
sangjaehn@sangjaehn-900X5N:~/cod
sangjaehn@sangjaehn-900X5N:~/cod
Connected Client IP: 127.0.0.1
i = 0
Time Over!
cnt = 15
Time Over!
Time Over!
[ ]

less than 3100
Input Number:
3050
greater than 3050
Input Number:
3060
greater than 3060
Input Number:
30370
less than 30370
Input Number:
3070
greater than 3070
Input Number:
3080
less than 3080
Input Number:
3075
greater than 3075
Input Number:
3076
You win!
Input Number:
[ ]

```

2) 파일전송 프로그램

- 서버 프로세스에서는 자신의 파일을 read 한 데이터를 소켓파일에 write 하고, 클라이언트 프로세스에서는 소켓파일의 데이터를 read 해서 자신의 파일에 write 한다.
- 클라이언트 프로세스에서 파일을 open 할 때, 파일 권한을 설정하지 않으면 open 이 안 됨.

<서버 프로세스>

```
#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);
    }

    fd = open("sangjae.txt", O_RDONLY); // "sangjae.txt" 파일을 읽기전용으로 open 함
    serv_sock = socket(PF_INET, SOCK_STREAM, 0); // 소켓 파일을 open 하고 fd 를 반환함(IPv4, TCP)

    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) // 소켓파일에 ip 주소, 포트번호 등록
        err_handler("bind() error");

    if(listen(serv_sock, 5) == -1) // 클라이언트 5 개까지 접속 허용
        err_handler("listen() error");

    clnt_addr_size = sizeof(clnt_addr);

    clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size); // 클라이언트의 connect 요청을 받아들
```



```

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) ? "AF_INET" : "AF_INET6");

    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;
}

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|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)    // 클라이언트 프로세스에 연결을 요청함
    err_handler("connect() error");

else
    puts("Connect .....");

while((read_cnt = read(sock, buf, BUF_SIZE)) != 0) /* 소켓 파일로부터 데이터를 모두 read 할 때까지 fd 에
                                                    write 함*/
    write(fd, buf, read_cnt);

puts("Received File Data");
write(sock, "Thank you", 10);    // 소켓파일에 데이터를 write 함
close(fd);
close(sock);

```

```
    return 0;
}
```

3) struct hostent *gethostbyname(const char *name) 함수 활용

- main 함수의 인자에 도메인 주소를 입력한다.

- hostent 구조체

```
struct hostent{
char *h_name;      // official name of host
char **h_aliases;  // alias list(별명)
int h_addrtype;    // host address type
int h_length;      // length of address
char **h_addr_list; // list of addresses
}
```

```
#include <unistd.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.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) ? "AF_INET" : "AF_INET6");

    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;
}
```

3-1) 결과 분석

```
Josephahn@Josephahn-Z20NH-A551B5U:~/code/linux/network/4.2$ ./a.out naver.com
Official Name : naver.com
Address Type: AF_INET
IP_Addr 1: 210.89.164.90
IP_Addr 2: 125.209.222.141
IP_Addr 3: 210.89.160.88
IP_Addr 4: 125.209.222.142
```