

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그램 전문가 과정

강사 - Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 - 은태영

zero_bird@naver.com

Network

```
tewill@tewill-B85M-D3H: ~/Downloads/sanghoonlee/lec/lnp
#include "load_test.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <stdbool.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 cnt[MAX_CLNT];
pthread_mutex_t mtx;

// Black List
int black_cnt;
char black_list[MAX_CLNT][16];
```

구현에 앞서 필요한 헤더 파일을 받아온다.

추가 함수를 받기 위하여 “load_test.h” 를 include 한다.

BUF_SIZE 와 MAX_CLNT 의 크기를 선언한다.

sockaddr * 와 sockaddr_in 의 명칭을 축약한다.

접속한 클라이언트의 숫자를 카운트 한다.

접속한 클라이언트의 IP를 저장한다.

thread 의 mutex key 값을 생성한다.

black_list 정보를 저장할 배열을 선언한다.

배열의 갯수를 관리하기 위해 black_cnt 를 선언한다.

Network

```
tewill@tewill-B85M-D3H: ~/Downloads/sanghoonlee/lec/lnp
#include "load_test.h"

double get_runtime(tv start, tv end)
{
    end.tv_usec = end.tv_usec - start.tv_usec;
    end.tv_sec = end.tv_sec - start.tv_sec;
    end.tv_usec += end.tv_sec * 1000000;

    #if DEBUG
    printf("runtime = %lf sec\n", end.tv_usec / 1000000.0);
    #endif

    return end.tv_usec / 1000000.0;
}
```

1,1 Top

load_test.h 를 통해서 get_runtime 함수를 사용한다.
시작 시간과 끝나는 시간을 인자로 받아온다.
두 인자의 차이를 계산한다.
그 후, 결과값을 리턴 한다.

Network

```
tewill@tewill-B85M-D3H: ~/Downloads/sanghoonlee/lec/lnp
// Information of Thread
typedef struct __iot{
    int sock;
    char ip[16];
    int cnt;
} iot;

iot info[BUF_SIZE];

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

void proc_msg(char *msg, int len, int sock)
{
    int i;

    pthread_mutex_lock(&mtx);

    for(i = 0; i < clnt_cnt; i++)
    {
        if(info[i].sock == sock)
            continue;
        write(info[i].sock, msg, len);
    }

    pthread_mutex_unlock(&mtx);
}
```

28,1 13%

black_list 를 관리하기 위해 iot 구조체를 만든다.

err_handler 로 오류 발생시 msg 를 출력한다.

proc_msg 를 통해 msg 가 보낸 사람에게 중복으로 날아가는 것을 방지하기 위해, sock 을 비교하고 동일할 경우 continue 한다.

동일한 IP 가 아닐 경우, 해당 msg 를 다른 클라이언트의 sock 에 보낸다.

Network

```
tewill@tewill-B85M-D3H: ~/Downloads/sanghoonlee/lec/lnp
void *clnt_handler(void *arg)
{
    iot thread_info = *((iot *)arg);
    int len = 0, i;
    char msg[BUF_SIZE] = {0};

    tv start, end;
    double runtime = 0.0;
    double load_ratio;

    for(;;)
    {
        gettimeofday(&start, NULL);
        //len = read(clnt_sock, msg, sizeof(msg));
        len = read(thread_info.sock, msg, sizeof(msg));
        proc_msg(msg, len, thread_info.sock);
        gettimeofday(&end, NULL);

        runtime = get_runtime(start, end);

        load_ratio = 1.0 / runtime;
        printf("load_ratio = %lf\n", load_ratio);

        if(load_ratio > 1.5)
            thread_info.cnt++;

        if(thread_info.cnt > 10)
        {
            write(thread_info.sock, "You're Fired!!!\n", 16);
            add_black_list(thread_info.ip);
            goto end;
        }
    }
}
```

90,1 46%

클라이언트 접속 시 해당 함수가 호출된다.

sock 을 받아온다.

대기 중인 시간을 start 로 받아온다.

read 를 통해 클라이언트에서 데이터를 입력시킬 때 까지
블럭 시킨다.

데이터가 들어올 경우, msg 에 데이터를 읽어온다.

msg 를 proc_msg 를 통해 보낸다.

메시지를 보낸 후, end 에 시간을 저장한다.

get_runtime 을 통해 두 시간의 차이를 리턴한다.

리턴 값을 횡수로 나누어, load_ratio 에 비율을 저장한다.

비율 값이 1.5 이상일 경우 cnt 를 증가시킨다.

cnt 값이 10 이상일 경우, 해당 ip를 add_black_list 를
통하여 블럭 시킨다.

Network

```
tewill@tewill-B85M-D3H: ~/Downloads/sanghoonlee/lec/lnp
end:
    pthread_mutex_lock(&mtx);
    for(i = 0; i < clnt_cnt; i++)
    {
        if(thread_info.sock == info[i].sock)
        {
            while(i++ < clnt_cnt - 1)
                info[i].sock = info[i + 1].sock;
            break;
        }
    }
#if 0
    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;
        }
    }
#endif

    clnt_cnt--;
    pthread_mutex_unlock(&mtx);
    close(thread_info.sock);

    return NULL;
}
-- INSERT --
```

160,1 65%

만약 black_list 가 될 경우, 반복문에서 빠져나온다.

해당 클라이언트를 찾은 후, 해당 ip 의 sock 을 제거하고 다른 클라이언트들의 값을 내린다.

마지막으로 해당 클라이언트를 close 한다.

Network

```
tewill@tewill-B85M-D3H: ~/Downloads/sanghoonlee/lec/lnp
void add_black_list(char *ip)
{
    pthread_mutex_lock(&mtx);
    strcpy(black_list[black_cnt++], ip);
    printf("black_list = %s\n", black_list[black_cnt - 1]);
    pthread_mutex_unlock(&mtx);
}

bool check_black_list(char *ip)
{
    int i;

    pthread_mutex_lock(&mtx);
    printf("Here\n");

    for(i = 0; i < black_cnt; i++)
    {
        if(!strcmp(black_list[i], ip))
        {
            pthread_mutex_unlock(&mtx);
            return true;
        }
    }

    pthread_mutex_unlock(&mtx);
    return false;
}
```

60,1 29%

add_black_list 는 ip 를 받아온 뒤, 해당 ip를 black_list 에 저장한다.

check_black_list 는 저장된 ip 와 들어온 ip 를 비교한다.
비교를 통해 같을 때 true 를, 아닐 경우 false 를 리턴 한다.

Network

```
tewill@tewill-B85M-D3H: ~/Downloads/sanghoonlee/lec/lnp
int main(int argc, char **argv)
{
    int serv_sock, clnt_sock;
    struct 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, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("bind() error");

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

    -- INSERT --
193,1 82%
```

기존 네트워크 양식과 동일한 형태를 갖고 있다.

Network

```
tewill@tewill-B85M-D3H: ~/Downloads/sanghoonlee/lec/lncp
for(;;){
{
    addr_size = sizeof(clnt_addr);
    clnt_sock = accept(serv_sock, (sp)&clnt_addr, &addr_size);

    printf("Check Black List\n");

    if(check_black_list(inet_ntoa(clnt_addr.sin_addr)))
    {
        write(clnt_sock, "Get out of my server!!!\n", 23);
        close(clnt_sock);
        continue;
    }

    pthread_mutex_lock(&mtx);

    info[clnt_cnt].sock = clnt_sock;
    strcpy(info[clnt_cnt].ip, inet_ntoa(clnt_addr.sin_addr));
    info[clnt_cnt++].cnt = 0;

    pthread_mutex_unlock(&mtx);

    pthread_create(&t_id, NULL, clnt_handler,
                  (void *)&info[clnt_cnt - 1]);
    pthread_detach(t_id);
    printf("Connected Client IP: %s\n",
          inet_ntoa(clnt_addr.sin_addr));

}

close(serv_sock);

return 0;
}
-- INSERT --
```

195,9-16 Bot

accept 를 이용해 sock 을 연결한다.

check_black_list 를 통해 black_list 에 해당 IP 가 있는지 확인한다.

있을 경우, get out 을 출력 후, 해당 sock 을 close 한다.

아닐 경우, 정상적으로 들어와서 동작한다.