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

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 장성환
redmk1025@gmail.com

*chat_sercv_final.c

```
#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];

// Information of Thread
typedef struct __iot{
        int sock;
        char ip[16];
        int cnt;
```

* chat_clnt_final.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
#define NAME_SIZE           32

typedef struct sockaddr_in   si;
typedef struct sockaddr *    sp;

char name[NAME_SIZE] = "[내가이긴다]";
char msg[2048];

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

void make_rand_str(char *tmp)
{
        int i, end = rand() % 7 + 3;

        for(i = 0; i < end; i++)
                tmp[i] = rand() % 26 + 65;
}
```

```
} 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);
}

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);
}
```

```

void *send_msg(void *arg)
{
	int sock = *((int *)arg);
	char msg2[] = "https://kr.battle.net/heroes/ko/  <<== 지금 당장 접속
하세요!!\n";
	srand(time(NULL));

	char tmp1[32] = {0};

	for(;;)
	{
#if PASSIVE
		fgets(msg, BUF_SIZE, stdin);

		write(sock, msg, strlen(msg));
#endif
#if ATTACK
		make_rand_str(tmp1);

		printf("%s\n", msg);
		sprintf(msg, "%s %s %s", name, tmp1, msg2);
		printf("tmp1 = %s\n", tmp1);
		write(sock, msg, strlen(msg));
		sleep(5);
#endif
	}

	return NULL;
}

void *recv_msg(void *arg)
{
	int sock = *((int *)arg);
```

```
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;
}

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(;;)
```

```
	char msg[NAME_SIZE + 2048];
	int str_len;

	for(;;)
	{
		str_len = read(sock, msg, NAME_SIZE + 2047);

		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");
	{
		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;
		}
	}

#if 0
	while((str_len = read(clnt_sock, msg, sizeof(msg))) != 0)
		proc_msg(msg, str_len, i);
#endif

end:
	pthread_mutex_lock(&mtx);

	for(i = 0; i < clnt_cnt; i++)
	{
		if(thread_info.sock == info[i].sock)
```

```
	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;
}
```

```
                {
                        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;
}

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

        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
*)&clnt_sock);
                    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;
}
```

```
* chat_serv_van2.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>
#include <signal.h>
#include <sys/time.h>


#define BUF_SIZE 128
#define MAX_CLNT 256

typedef struct sockaddr_in si;
typedef struct sockaddr * sp;
typedef struct timeval tv;

int clnt_cnt = 0;
int clnt_socks[MAX_CLNT];
pthread_mutex_t mtx;
int clnt_attack[MAX_CLNT];
int ssid;

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

long get_runtime(tv start, tv end){
```

```
*chat_clnt_van2.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
#define NAME_SIZE 32

typedef struct sockaddr_in si;
typedef struct sockaddr * sp;

char name[NAME_SIZE] = "[DEFAULT]";
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 name_msg[NAME_SIZE + BUF_SIZE];

   for(;;){
         fgets(msg, BUF_SIZE, stdin);
   long rtime;
   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;

   rtime = end.tv_usec;
   return rtime;
   //printf("runtime = %lf sec\n", end.tv_usec / 1000000.0);
}

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

   pthread_mutex_lock(&mtx);

   for(i=0; i<clnt_cnt;i++)
          write(clnt_socks[i], msg,len);

   pthread_mutex_unlock(&mtx);
} //broad casting 한다.

void stop_talking(int clnt_sock, int i){
   char *msg = "너 말이 너무 많아! 10 초간 채팅금지!\n";
   char *msg2 = "채팅 금지인데 계속 글 쓰지마 ㅡㅡ\n";
   char tmp[BUF_SIZE];
   int len = strlen(msg);

   write(clnt_sock, msg,len);
   sleep(10);
   while(read(clnt_sock,tmp, sizeof(tmp)) == BUF_SIZE){
          write(clnt_sock, msg2, strlen(msg2));
          memset(tmp,0,sizeof(tmp));
```

```
          if(!strcmp(msg, "q\n") || !strcmp(msg, "Q\n")){
             close(sock);
             exit(0);
          }

          sprintf(name_msg, "%s %s", name, msg);
          write(sock, name_msg, strlen(name_msg));
   }
   return NULL;
}

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

   for(;;){
          str_len = read(sock, name_msg, NAME_SIZE + BUF_SIZE -1);

          if(str_len == -1)
             return (void*)-1;

          name_msg[str_len] =0;
          fputs(name_msg, stdout);
   }
   return NULL;
}

int main(int argc, char **argv){

   int sock;
   si serv_addr;
   pthread_t snd_thread, rcv_thread;
```

```
  }
  memset(tmp, 0, sizeof(tmp));
}

void *clnt_handler(void *arg){
  int clnt_sock = *((int*)arg);
  int str_len = 0, i;
  char msg[BUF_SIZE];
  tv start, end;

  while((str_len = read(clnt_sock, msg, sizeof(msg))) != 0){

        gettimeofday(&start, NULL);
        pthread_mutex_lock(&mtx);

        for(i=0; i<clnt_cnt; i++){
           if(clnt_sock == clnt_socks[i]){
                clnt_attack[i] +=1;
                if(clnt_attack[i] >6){
                   pthread_mutex_unlock(&mtx);
                   stop_talking(clnt_sock,i);

                   pthread_mutex_lock(&mtx);
                   clnt_attack[i] = 0;
                   pthread_mutex_unlock(&mtx);
                   break;
                }
                else{
                   break;
                }
           }
        }// 3 초안에 6 번이상 말하면 퇴출
```

```
  void *thread_ret;

  if(argc !=4){
        printf("Usage: %s <IP> <port> <name>\n", argv[0]);
        exit(1);
  }

  sprintf(name, "[%s]", argv[3]);
  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() errpr!");

  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;
}
```

```
        pthread_mutex_unlock(&mtx);

        send_msg(msg, str_len);
        alarm(3);
        gettimeofday(&end, NULL);
        get_runtime(start,end);
  }

  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;
        }
  }//나간놈은 알아서 클라이언트 소켓 저장소에서 퇴출!

  clnt_cnt--;
  pthread_mutex_unlock(&mtx);
  close(clnt_sock);

  return NULL;
}

void sig_handler(int signo){
  int i;
  pthread_mutex_lock(&mtx);
  for(i=0; i<clnt_cnt; i++){
        clnt_attack[i] =0;
  }
```
	기존에 만들고 있던 밴 프로그램을 완성하였다.
	
	선생님과 다른 부분은 처음 커넥트가 되고 거기서 아이피 관리를 하여 못들어오게 막는것이 아니라, 글자 수가 많아지면 10 초간 다른 사람과의 채팅이 불가능하게 하였다.

```
  pthread_mutex_unlock(&mtx);
}

int main (int argc, char **argv){
  int serv_sock, clnt_sock;
  si serv_addr, clnt_addr;
  socklen_t addr_size;
  pthread_t t_id;
  signal(SIGALRM,sig_handler);

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

  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, 10) == -1)
        err_handler("listen() error!");
```

```
    ssid = serv_sock;

    for(;;){
            addr_size = sizeof(clnt_addr);
            clnt_sock = accept(serv_sock, (sp)&clnt_addr, &addr_size);

            pthread_mutex_lock(&mtx);
            clnt_socks[clnt_cnt++] = clnt_sock;
            pthread_mutex_unlock(&mtx);

            pthread_create(&t_id, NULL, clnt_handler, (void *)&clnt_sock);
            pthread_detach(t_id);
            printf("Connected Client IP: %s\n", inet_ntoa(clnt_addr.sin_addr));
    }

    close(serv_sock);
    return 0;
}
```

*** 일정한 패턴이 들어올 경우에 방어하는 방법을 시도해보자. (패턴방지 기법)**