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

#30

2018.04.04

강사:Innova Lee(이 상훈)

학생: 김시윤

수업내용 복습

Hi_serv.c

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

typedef struct sockaddr_in      si;
typedef struct sockaddr *       sap;

#define BUF_SIZE        32

//char h[BUF_SIZE]="Hi clnt";

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

void read_childproc(int sig)
{
        pid_t pid;
        int status;
        pid = waitpid(-1, &status, WNOHANG);
//프로세스 올때까지 기다렸다가 리무브드
        printf("Removed proc id: %d\n",pid);
}
```

```
int main(int argc, char **argv)
{
        int serv_sock, clnt_sock;
        si serv_addr, clnt_addr;
        pid_t pid;
        struct sigaction act;
        socklen_t addr_size;
        int str_len, state;
        char buf[BUF_SIZE]={0};
        char h[BUF_SIZE]="Hi clnt";

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

        act.sa_handler = read_childproc;
        sigemptyset(&act.sa_mask);
        act.sa_flags = 0;
        state = sigaction(SIGCHLD, &act, 0);//시그차일드 오면 read_childproc

        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");

	for(;;)
	{
		addr_size = sizeof(clnt_addr);
		clnt_sock = accept(serv_sock, (sap)&clnt_addr, &addr_size);
		if(clnt_sock == -1)
			continue;
		else
			puts("New Client Connected...");
		pid = fork();//프로세스 생성
		if(pid == -1)//오류
		{
			close(clnt_sock);
			continue;
		}
		if(pid == 0)//자식
		{
			close(serv_sock);//서버소켓 닫는다.

			while((str_len = read(clnt_sock, buf, BUF_SIZE)) !=0)
			{

				write(1,"msg from clnt : ",16);
//모니터에 메시지 프롬 클라이언트 출력
				write(1,buf,str_len);
//모니터에 클라이언트에서 읽은 데이터 출력
				write(clnt_sock, h, strlen(h));
//클라이언트에 헬로 전달
			}

			close(clnt_sock);//read 가 block 니까 종료될일 없음.
			puts("Client Disconnected ...");
			return 0;
		}
		else
```

```
			close(clnt_sock);//클라이언트 소켓 닫고 새로운 클라이언트 올
때까지 어셉트에서 기다림
		}
		close(serv_sock);

		return 0;
}
```

```
#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
char h[BUF_SIZE]="Hello\n";

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

void read_routine(int sock, char *buf)
{
	for(;;)
	{
		int str_len = read(sock,buf,BUF_SIZE);

		if(str_len == 0)
```

```
                        return;

                write(1,"msg from server: ",17);
                write(1,buf,strlen(buf));//서버에서 보낸 하이 출력
        //      printf("msg from server: %s",buf);
        }
}

void write_routine(int sock, char *buf)
{
        for(;;)
        {
                fgets(buf, BUF_SIZE, stdin);

                if(!strcmp(buf, "q\n") || !strcmp(buf, "Q\n"))
                {
                        shutdown(sock, SHUT_WR);
                        return;
                }
//      write(sock,buf,strlen(buf));
        write(sock,h,strlen(h));//서버에 헬로 출력
        }
}

int main(int argc, char **argv)
{
        pid_t pid;
        int i, sock;
        si serv_addr;
        char buf[BUF_SIZE] ={0};

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

```
        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 .....");

        pid = fork();//자식은라이트 부모는 리드

        if(pid ==0)
                write_routine(sock, buf);
        else
                read_routine(sock,buf);

        close(sock);

        return 0;
}
```

---

write 와 printf 를 같이 쓰면 printf 가 씹히는 현상이 있다.
이유를 많이 생각했는데 이 이유가 write 가 시스템 콜이라 시스템콜 요청을 먼저 수행한 후 사이즈가 버프사이즈를 넘어가게 되면 printf 가 출력되는거같다.

chat_serv.c (도배감지시 클로우즈)
소스코드를 못가져왔습니다..

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

                printf("cnt %d \n",cnt);

        }
```

clnt_handler 부분을 바꾸었습니다

```
for(;;)
{
                gettimeofday(&start,NULL);
                str_len = read(clnt_sock,msg,sizeof(msg));
                cnt +=1;
                send_msg(msg,str_len);
                gettimeofday(&end,NULL);
                runtime = get_runtime(start, end);
                load_ratio = cnt / runtime;

                if(load_ratio > 6)
                {
                        close(clnt_sock);
                }
                 cnt = 0;
}
```

Chat_serv.c (teacher ver)

```
#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];//ip 를 저장할 버퍼
        int cnt;//도배횟수를 감지하기 위해
} iot;
//thread 정보를 갖고있기 위해
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)//에코를 볼필요 없으니 자기 해당하면 재껴라 .
			continue;
		write(info[i].sock, msg, len);
	}

	pthread_mutex_unlock(&mtx);
}

void add_black_list(char *ip)//블랙리스트 추가
{
	pthread_mutex_lock(&mtx);
	strcpy(black_list[black_cnt++], ip);//해당아이피값 스트링 카피 해서 넣어줌
	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++)//블랙리스트 카운트만큼돌면서 블랙리스트에 있는값과 비교
	{
		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);//구조체를 전달
	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));//스레드 인포에 있는 클라이언트 소켓을 리드
		proc_msg(msg, len, thread_info.sock);//읽은 메시지 엠에스지에 그담 프록메시지
		gettimeofday(&end, NULL);

		runtime = get_runtime(start, end);//런타임 구해옴

		load_ratio = 1.0 / runtime;//1 번 걸리는 시간을 봄
```

```
                printf("load_ratio = %lf\n", load_ratio);

                if(load_ratio > 1.5)
                        thread_info.cnt++;//1 번씩 증가

                if(thread_info.cnt > 10)//10 번 넘으면 강퇴
                {
                        write(thread_info.sock, "You're Fired!!!\n", 16);
                        add_black_list(thread_info.ip);
                        goto end;//엔드로 감
                }
        }

#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)
                {
                        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)))//블랙리스트 체크
		{
			write(clnt_sock, "Get out of my server!!!\n", 23);
			close(clnt_sock);
			continue;
		}

		pthread_mutex_lock(&mtx);

		info[clnt_cnt].sock = clnt_sock;//스레드 정보의 구조체 가 인포 0 번 현재소켓 에셉트된 소켓 셋팅
		strcpy(info[clnt_cnt].ip, inet_ntoa(clnt_addr.sin_addr));//벤할라고 아이피 집어넣음 해당아이피가 몇번반복했는지
		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]);//배열 주소가 전달됨 (낱개 하나만 전달함)
		pthread_detach(t_id);
		printf("Connected Client IP: %s\n", inet_ntoa(clnt_addr.sin_addr));
	}

	close(serv_sock);

	return 0;
}
```

오늘은 구현 하지 않고 구현하기전 알고리즘을 생각해보았습니다

len = read(thread_info.sock, msg, sizeof(msg));
위에 cmpmsg 라는 배열을 선언합니다.
If(!strcmp(cmpmsg,msg))
{
	cmpcnt ++;

	if(cmpcnt > 5)
	{
		close(thread_info.sock);
	}
}

이런식으로 하면 다섯번일때 클라이언트를 닫아줄거 같습니다.
그렇다면 선생님께서 만든 함수를 이용하여

len = read(thread_info.sock, msg, sizeof(msg));

If(!strcmp(cmpmsg,msg))
{
	cmpcnt ++;

```
            if(cmpcnt > 5)
            {
               write(thread_info.sock, "You're Fired!!!\n", 16);
               add_black_list(thread_info.ip);
              goto end;//엔드로 감
            }
}
```

이렇게하면 될거같습니다.

chat_clnt (teacher ver)

```
#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;
//3~9 까지
//각 요소에 골고루 넣어줌
//길이와 횟수를 랜덤으로 선언
}

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

        char tmp1[32] = {0};

        for(;;)
        {
#if PASSIVE
                fgets(msg, BUF_SIZE, stdin);
//수동으로 입력이 패시브
                write(sock, msg, strlen(msg));
#endif
#if ATTACK
                make_rand_str(tmp1);
//옵션디로 어택 -DATTACK
                printf("%s\n", msg);
                sprintf(msg, "%s %s %s", name, tmp1, msg2);
                printf("tmp1 = %s\n", tmp1);
```

```
                write(sock, msg, strlen(msg));
                sleep(5);
//랜덤으로 들어감.
#endif
        }

        return NULL;
}

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

        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;
}
```

3 6 9 game

Game.h

```
#include <stdio.h>
#include <stdlib.h>

struct data
{
	char signal[128];//시그날 신호처리 횟수 카운트
	char ms[128];//전달될 메시지 저장 버퍼
};
typedef struct data Data;//데이터 구조체 선언
```

369game_serv.c

```
#include "game.h"
#include <malloc.h>
#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 dat=0;
int thread_pid[MAX_CLNT];
int idx;
int cnt[MAX_CLNT];
int lum;
int totalcnt=0;
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;
			totalcnt +=1;
			}

	pthread_mutex_unlock(&mtx);

	alarm(3);
}
```

```
void proc_msg(Data *data, int k)
{
	int i;
	int cmp = atoi(data->ms);//클라이언트에서 받은 메시지를 cmp 에 저장
	char smsg[64] = {0};
	char numc[1024];//3,6,9 의 갯수를 세기위한 배열
	int ncnt=0;//실질적으로 3,6,9 의 수를 알려주는 녀석

	dat += 1;
	sprintf(numc,"%d",dat); //현재 데이터 값을 배열에 저장

	pthread_mutex_lock(&mtx);
	totalcnt +=1;
	cnt[k] += 1;
	sprintf(smsg,"You Player[%d]\n",k);
	write(clnt_socks[k],smsg,strlen(smsg));

	for(i=0;numc[i];i++)
	{
		if(numc[i] =='3' || numc[i] == '6' || numc[i] == '9')
		{
			ncnt+=1;
		}
	} //3,6,9 의 개수를 구해주는 로직

	if(ncnt != 0) //3 , 6, 9 가 한개라도 존재한다면
	{
		printf("cnt3 =%d\n",ncnt);
		printf("dat= %d\n",dat); //디버깅용
		lum=atoi(data->signal); // 클라이언트에서 발생한 시그날 횟수를 저장
		printf("lum = %d\n",lum);//디버깅용
		if(lum == ncnt) //시그날 횟수와 3,6,9 의 갯수가 같다면
		{
			sprintf(smsg,"Next turn %d\n",cmp);//다음턴
		}
		else if(lum != ncnt) //다르면
		{
			sprintf(smsg,"youloose %d\n",cmp);
			//넌 졌어
		}

	}
	else//3,6,9 의 갯수가 없다면
	{

	if(dat > cmp){
		sprintf(smsg, "youloose %d\n", cmp);}
	else if(dat < cmp){
		sprintf(smsg, "you loose %d\n", cmp);}//데이터 값이 다르면 넌졌어
	else//데이터값이 같으면 다음턴
	{
		strcpy(smsg, "Next turn\n");
		printf("player[%d]cnt = %d\n",k, cnt[k]);
		printf("totalcnt = %d\n",totalcnt);
	}
	}
	strcat(smsg, "Input Number: \n");
	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();

        Data *data;
        data =(Data *)malloc(sizeof(Data));
        //입력받는게 숫자 몇글자라 malloc 은 필요없지만 그래도 배열을 사용하니
          혹시 모를 공간부족을 대비

        i = idx - 1;
        printf("i = %d\n", i);
        write(clnt_socks[i], pattern, strlen(pattern));
        pthread_mutex_unlock(&mtx);
        alarm(3);

        while(read(clnt_sock,data,sizeof(Data)) !=0)
//클라이언트에서 넘어온 데이터 구조체 자체를 읽는다
        {
                alarm(0);
                proc_msg(data,i);//구조체 변수 데이터와 인덱스를 넘긴다.
                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);
		thread_pid[idx++] = getpid();
		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;
}
```

369game_clnt.c

```
#include "game.h"
#include <signal.h>
#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 <malloc.h>

#define BUF_SIZE		128

typedef struct sockaddr_in	si;
typedef struct sockaddr *	sp;
int sign;
char msg[BUF_SIZE];
int sock;

void my_sig(int signo)
{

	char buf[BUF_SIZE];
	int len;
	Data *data;
	data =(Data *)malloc(sizeof(Data));
	sign +=1;
	len = strlen(buf);
}
//시그널 발생시 시그널 횟수를 세어 주는 함수

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

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

```
	char msg[BUF_SIZE];
	Data *data;
	data = (Data *)malloc(sizeof(Data));
//마찬가지로 데이터 구조체 선언. 굳이 malloc 을 사용할 필요는 없지만
혹시모를 경우를 위해 ..
	for(;;)
	{
		signal(SIGINT,my_sig);  //ctrl + c 발생시 my_sig 함수 호출
		read(0,data->ms,BUF_SIZE);    //입력되는 숫자값 메시지에 저장
		sprintf(data->signal,"%d",sign);//my_sig 에서 올린 숫자값 signal
저장
		write(sock,data,sizeof(Data));
		//구조체 자체를 서버에 전달 (정말 편리 데이터 여러개 한번에 전달가능)
	sign=0; //다음 시그널값 받기위해 시그널값 초기화
	}

	return NULL;
}

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

	Data *data;
	data = (Data *)malloc(sizeof(Data));

	write(1,"369game start\n",15);
	for(;;)
	{
		str_len = read(sock,data->ms, BUF_SIZE - 1);

		data->ms[str_len] = 0;
		fputs(data->ms,stdout);
	}
	return NULL;
}

int main(int argc, char **argv)
{
	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);
	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;
}
```