Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정

날 짜 : 2018 . 4. 3

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

< 블록하기 > _(아직 세번 벙어리 시키고 말 못하게 하는거 구현중)

<blocking_server.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/time.h>
#include<sys/epoll.h>
#include<malloc.h>

#define BUF_SIZE                128
#define MAX_CLNT                256
#define SET_TIMES               6

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

int cnt[MAX_CLNT];
int clnt_cnt = 0;
int idx;
int clnt_socks[MAX_CLNT];
int thread_pid[MAX_CLNT];
double runtime=0.0;
double load_ratio;
int flag;

pthread_mutex_t mtx;
tv start, end;


// 카운트를 위한 구조체 // 안씀
typedef struct __count
{
        int send_client[0];
}count;


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

```
count *ct = 0;
// 시간을 구해주는 함수.
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((end.tv_usec / 1000000.0) > 10)
	{
		gettimeofday(&start, NULL);
		//cnt[]
	}

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

void proc_msg(char *msg, int len, int k)
{
	int i;
	//int cmp = atoi(msg);
	char smsg[256] = {0};
	char clnt_count[BUF_SIZE];

	pthread_mutex_lock(&mtx);

	cnt[k] += 1;

	sprintf(smsg,"[<말한 횟수 : %d>]\n",cnt[k]);
	write(clnt_socks[k], smsg, strlen(smsg));

	printf("cnt = %d\n", cnt[k]);

/*
	if(data[k] > cmp)
		sprintf(smsg, "greater than %d\n", cmp);
```

```
		else if(data[k] < cmp)
			sprintf(smsg, "less than %d\n", cmp);
		else
		{
			sprintf(clnt_count,"[<말한 횟수(1.6):%d>]\n",cnt[k]);
			write(clnt_socks[k], clnt_count, 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];
		char clnt_count[BUF_SIZE];
		i = clnt_cnt - 1;

		flag = 0;
		//tv start, end;


		pthread_mutex_lock(&mtx);
		thread_pid[idx++] = getpid();
		pthread_mutex_unlock(&mtx);

		gettimeofday(&start, NULL);
		while((str_len = read(clnt_sock , msg, sizeof(msg))) != 0){


			//ct->send_client[clnt_cnt-1] += 1;
			proc_msg(msg, str_len, i);
			send_msg(msg, str_len);

			// 끝나는 시간을 구함.
			gettimeofday(&end, NULL);

			//pthread_mutex_unlock(&mtx);

			runtime = get_runtime(start, end);
			if(runtime > 3)
			{
				gettimeofday(&start, NULL);
				cnt[i]=0;
			}
			load_ratio = 1.0/runtime;
			pthread_mutex_lock(&mtx);
```

```
                if(load_ratio > 5 || cnt[i]>40)
                {
                        flag++;
                        // 여기서 나갈때 같이 나가버림.
                        if(flag == 3){
                                write(clnt_socks[i], "너 이제 진짜 말못함\n", 128);
                                shutdown(clnt_socks[i], SHUT_WR);
                        }
                        write(clnt_socks[i], "당신은 잠시 벙어리\n", 128);
                        sleep(6);
                        while(read(clnt_sock, msg, sizeof(msg)) > BUF_SIZE-1){
                                memset(msg, 0, sizeof(msg));
                        }
                        memset(msg, 0, sizeof(msg));
                        gettimeofday(&start, NULL);
                        write(clnt_socks[i], "당신은 기적으로 말할 수 있게 되었습니다. \n", 128);

                }
                pthread_mutex_unlock(&mtx);

                printf("runtime = %lf sec\n", runtime);
                printf("{load_ratio} = %lf sec\n", load_ratio);

        }


        pthread_mutex_lock(&mtx);
        // 여기가 뭐하기 위한 건지 정확히 모르겠다.
        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;

unsigned int i, cnt = 0;


ct = (count*)malloc(1024);

gettimeofday(&start, NULL);

for(i = 0; i < 777777777; i++)
	cnt++;

gettimeofday(&end, NULL);

// 여기서 시간을 구한다.
runtime = get_runtime(start, end);


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 , 25)== -1)
	err_handler("listen() error!");

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


	pthread_mutex_lock(&mtx);
	ct->send_client[clnt_cnt] = 0;
```

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

                printf("clnt_cnt (사람수) : %d\n", clnt_cnt);

                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;
}
```

<blocking_client.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);

		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;
	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);

        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;
}
```