Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정
#28

강사:Innova Lee(이 상훈)
학생: 김시윤

수업내용 복습

---gameserver.c---

```
#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;
//락을 걸때 락의 키값!!

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;

	pthread_mutex_unlock(&mtx);

	alarm(3);
}

void proc_msg(char *msg, int len, int k)
{
	int i;
	int cmp = atoi(msg);
//숫자값으로 변환
	char smsg[64] = {0};

	pthread_mutex_lock(&mtx);

	cnt[k] += 1;
//몇번 입력했는지 카운트

	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]);
```

```
//You win 나오면 클라이언트 소켓 종료 시키면 좋음
		}

		strcat(smsg, "Input Number: \n");
//strcat 스트링에 갖다 붙이는거
		write(clnt_socks[k], smsg, strlen(smsg));

		pthread_mutex_unlock(&mtx);
//작업 끝나서 언락
}

void *clnt_handler(void *arg)
//클라이언트 소켓 넘어감
{
		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);
//다시 락을걸어
		thread_pid[idx++] = getpid();
//clnt_handler 는 스레드에서 동작 그래서 스레드의 피아이디를 넣었음

		i = idx – 1;
//인덱스값 확인하기 위해
		printf("i = %d\n", i);
		write(clnt_socks[i], pattern, strlen(pattern));
//첫번째 클라이언트에 패턴을써주겠다.
		pthread_mutex_unlock(&mtx);
//스레드가 여러개 될수 있으니 스레드간에 접근을 허용시키지 않기위해 크리티컬섹션에대
한 보호

		alarm(3);
		while((str_len = read(clnt_sock, msg, sizeof(msg))) != 0)
//클라이언트가 먼가 숫자를 입력했을때 뭐가 들어왔는지 수신 msg 에 담겨있음
		{
			alarm(0);
//수신되었으니 알람 끈다.
			proc_msg(msg, str_len, i);
			alarm(3);
//프록 엠에스지 끝나서 다시 알람 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);
//초기에 락을한게 없으니 아무것도 없다고 널로 초기화.

	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);
		//클라이언트의 접근을 승인 - 리턴결과 ->클라이언트 소켓 클라이언트의
fd
```

```
		thread_pid[idx++] = getpid();
		//thread pid 얻기?

		pthread_mutex_lock(&mtx);
		//데이터가 꼬이지 말라고 락을 걸어준다. 밑에 데이터에 접근해서 뭔가 작
업을 할수 있으니락을 걸어놓는다 락을 걸어준 순간부터는 다른애들 접근 못함 그래서 데이
터가 다들 다른거임.

		data[clnt_cnt] = rand() % 3333 + 1;
		clnt_socks[clnt_cnt++] = clnt_sock;
		pthread_mutex_unlock(&mtx);
//데이터 작업이 끝난후 언락

		pthread_create(&t_id, NULL, clnt_handler, (void
*)&clnt_sock);
		//thread 가 구동시키는 함수 clnt_handler , clnt_sock thread 에 전달
되는 인자.
		pthread_detach(t_id);
		//떼어내다 atach(붙이다)
		//생성하고 떼어낸다? t_id 생성된 스레드 아이디 식벽자 프로세스랑 스레
드랑 분리시킨다는 얘기(시피유에 할당되는거) 그래서 별도로 동작가능
		printf("Connected Client IP: %s\n",
inet_ntoa(clnt_addr.sin_addr));
	//위로올라감 올라가면 다음 클라이언트가 들어올때까지 블로킹 그담 시엘엔티 핸
들로봐
	}

	close(serv_sock);

	return 0;
}
```

데이터와 cnt 를 공유하려면 데이터와 카운트를 배열로 만들면 안된다.
배열 말고 인트형 변수로 선언하여 관리하도록 해야한다.

그리고 데이터는 for 문안에 있으면 랜덤값이 클라이언트 생길때마다 바뀌기 때문에 클라이언트 밖에 넣어줘야한다.

```
//      if(data[k] > cmp)
        if(dat > cmp)
                sprintf(smsg, "greater than %d\n", cmp);
//      else if(data[k] < cmp)
        else if(dat < cmp)
                sprintf(smsg, "less than %d\n", cmp);
        else
        {
                strcpy(smsg, "You win!\n");
                printf("cnt = %d\n", cnt[k]);//유윈 나오면 클라이언트 소켓 종료 시키면 좋음
        }
```

game_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

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)
//서버에서 연결된 소켓이 넘어옴
//엄청 많은 정보가 들어오면 구조체로 묶어서 보낸다.
{
        int sock = *((int *)arg);
//서버의 소켓이 전달이 됨
        char msg[BUF_SIZE];

        for(;;)
        {
                fgets(msg, BUF_SIZE, stdin);
//입력을 받겠다 msg 에 저장
                write(sock, msg, strlen(msg));
//sock 에다 라이트함 입력한 메시지가 서버로 전송이됨
        //무한루프 돌면서 입력을 받아서 서버로 보내는 역활을 계속함
        }

        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);
//서버로부터 들어온 정보를 수신해서 msg 에 넣어준다.

		msg[str_len] = 0;//뒤에꺼 읽는거 방지하기 위해
		fputs(msg, stdout);
//모니터에 출력해줌 write(0,msg,strlen(msg))해도 무관
	}

	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)
//connect 하는 순간 accept 동작
		err_handler("connect() error");
```

```
//thread 만든이유 송신가 수신 분리.

	pthread_create(&snd_thread, NULL, send_msg, (void *)&sock);

//pthread_create(thread id 저장변수,스레드 특성설정,스레드가 실행할 함수, 호출함수에 전달할 정보) //메시지 보내는 스레드가 하나생성

	pthread_create(&rcv_thread, NULL, recv_msg, (void *)&sock);

//pthread_create(id 저장변수, 스레드 특성설정, 스레드가 실행할 함수, 호출함수에 전달할 정보)  //메시지 수신하는 놈

	pthread_join(snd_thread, &thread_ret);

//snd_thread 함수가 종료되길 기다렸다 종료되면 리턴값 받아옴 //조인하는 순간 본격적인 동작 더이상 센드할게 없거나 리시브할게 없을때 대표적으로 컨트롤 씨일때 종료하게 된다.

	pthread_join(rcv_thread, &thread_ret);

//rcv_thread 함수가 종료되길 기다렸다 종료되면 리턴값 받아옴

	close(sock);

	return 0;
}
```

file_server.c

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

```c
#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("file_server.c", O_RDONLY);//send file name
	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");

	clnt_addr_size = sizeof(clnt_addr);

	clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size);

	for(;;)
	{
		read_cnt = read(fd, buf, BUF_SIZE);
//file_server.c
	//read_cnt = read 한 byte 수  (read 의 리턴은 읽은바이트수)
		if(read_cnt < BUF_SIZE)
		{
			write(clnt_sock, buf, read_cnt);
			break;
//파일 크기가 버프사이즈보다 작으면 여기서 write
		}
		write(clnt_sock, buf, BUF_SIZE);
//file 크기가 buf_size 를 초과하면 여기서 write (파일이 중간에 끊히지 않게 하기 위함)
	}
```

```
        shutdown(clnt_sock, SHUT_WR);
        read(clnt_sock, buf, BUF_SIZE);
//client 의 write(sock, "Thank you", 10); 을 받아온다
        printf("msg from client: %s\n", buf);
//클라이언트에서 받아온 Thank you write

        close(fd);
        close(clnt_sock);
        close(serv_sock);

return 0;
}
```

file_client.c

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.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)
{
        char buf[BUF_SIZE] = {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);
        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 ........");

        while((read_cnt = read(sock, buf, BUF_SIZE)) !=0)
//서버에서 write 하는 값을 읽어서 buf 에 저장
```

```
                write(fd, buf, read_cnt);//recieve.txt:
//buf 에 저장되있는 값을 fd 로 옮김 fd 의 receive.txt 에 write

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

        return 0;
}
```

Gethostbyname.c

```
#include <unistd.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdlib.h>
#include <stdio.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]);
//host 이름받는다.
//ex- ./a.out naver.com

        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");
//AF_INET → Ipv4, AF_NET → Ipv6
//Ipv4 가 참이면 AF_INET 거짓이면 AF_NET

        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]));

//80 번으로 접속한거

return 0;
}
```