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

* Question
서버-클라 간에 구조체를 주고 받을 수 있도록 한다.

* mpecho_serv(구조).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

typedef struct _structype{
   int x;
   char buf[BUF_SIZE];
}SDATA;

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

void read_childproc(int sig){
   pid_t pid;
```

* mpecho_clnt(구조체).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

typedef struct _structype{
   int x;
   char buf[BUF_SIZE];
}SDATA;

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

void read_routine(int sock, SDATA data){

   for(;;){
```

```
  int status;
  pid = waitpid(-1, &status, WNOHANG);
  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};
  SDATA data;

  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);

  serv_sock = socket(PF_INET, SOCK_STREAM, 0);

  if(serv_sock == -1)
        err_handler("socket() error");

  memset(&serv_addr, 0, sizeof(serv_addr));
```

```
        int str_len = read(sock, &data, sizeof(SDATA));
        if(str_len == 0)
           return;

        printf("msg from server: %s\n",data.buf);
        printf("msg frem server id:%d\n",data.x);
  }
}



void write_routine(int sock, SDATA data){

  for(;;){
        char tmp[10];
        fgets(tmp,sizeof(tmp),stdin);
        data.x = atoi(tmp);
        fgets(data.buf,BUF_SIZE,stdin);
        write(sock, &data, sizeof(data));
  }
}

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

  pid_t pid;
  int i, sock;
  si serv_addr;
  SDATA data;
  //char buf[BUF_SIZE] = {0};

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

```
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();

        if(pid == -1){
           close(clnt_sock);
           continue;
        }

        if(pid == 0){
           close(serv_sock);

           while((str_len = read(clnt_sock, &data, sizeof(SDATA))) !=0)
                 write(clnt_sock, &data, sizeof(SDATA));

           close(clnt_sock);
           puts("Client Disconnected...");
        }

        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();

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

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

구조체 전달도 보내고 받는 쪽의 데이터가 같은 형식이면
일반적으로 주고 보내는 문자열 처럼 처리가 가능하다.

이전 프로세스 fork 를 이용하여 구성한 코드와는 다른점이 있다.

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

    return 0;
}
```

이전에는 자식 프로세스 2 개로 read() write()를 각각 구현하였다면,
이번에는 부모가 한 파트를 맡는 형식이다.

* 채팅 프로그램 (말많은 사용자 10 초간 VAN 구현)

* chat_serv_van.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>

#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];
pthread_mutex_t mtx;
int clnt_attack[MAX_CLNT];

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

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

*chat_clnt_van.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(;;){
```

```
  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){
  char *msg = "너 말이 너무 많아! 10 초간 채팅금지!\n";
  int len = strlen(msg);
  write(clnt_sock, msg,len);
  sleep(10);
}

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

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

        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);
```

```
        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;
```

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

        pthread_mutex_unlock(&mtx);

        send_msg(msg, str_len);
        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;
        }
    }//나간놈은 알아서 클라이언트 소켓 저장소에서 퇴출!

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

    return NULL;
}
```

```
    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);

    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;
}
```

```
void sig_handler(int signo){
   int i;
   pthread_mutex_lock(&mtx);
   for(i=0; i<clnt_cnt; i++){
         clnt_attack[i] =0;
   }

   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]));
```

*출력결과

```
sunghwan@HWAN:~/Documents/network/4-3$ ./chclnt 127.0.0.1 7777 sunghwan
hello?
[sunghwan] hello?
[hosung] hi?
nice to meet you~
[sunghwan] nice to meet you~
[hosung] fuck i'm destoyer
[hosung] now iam start !
[hosung] f
[hosung] f
[hosung] f
[hosung] f
[hosung] f
[hosung] f
what?
[sunghwan] what?
[hosung] f
[hosung] f
[hosung] sorry...
sunghwan@HWAN:~/Documents/network/4-3$ ./chclnt 192.168.0.17 7777 hosung
[sunghwan] hello?
hi?
[hosung] hi?
[sunghwan] nice to meet you~
fuck i'm destoyer
[hosung] fuck i'm destoyer
now iam start !
[hosung] now iam start !
f
[hosung] f
f
[hosung] f
f
[hosung] f
f
[hosung] f
f
[hosung] f
f
[hosung] f
f
너 말이 너무 많아! 10초간 채팅금지!
f
[sunghwan] what?
[hosung] f
[hosung] f
sorry...
[hosung] sorry...
```

```
  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!");

  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;
}
```

클라이언트에도 채금 당하면 시그널을 이용하든 뭘하든 sleep 으로 잠재우는 코드를 추가하는 것이 더 안정적으로 동작이 가능 할 것 같다. (버퍼에 쓴 값이 저장되어 채금이 풀리면 다시 나옴)

*load_test.h

```
#ifndef __LOAD_TEST_H__

#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>

typedef struct timeval tv;

void get_runtime(tv, tv);

#endif
```

*load_test.c

```
#include "load_test.h"

void 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;

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

#if DEBUG
int main(void)
{
	unsigned int i, cnt = 0;
	tv start, end;

	gettimeofday(&start, NULL);
```

gcc -DDEBUG load_test.c 하면 런타임을 알 수 있다.

스타트 시각과 어떤 처리를 하고나서의 엔드 시각의 차이를
구하는 코드이다.

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

	gettimeofday(&end, NULL);

	get_runtime(start, end);

	return 0;
}
#endif
```

*gethostbyaddr.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>

typedef struct sockaddr_in si;

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

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

   int i;
   si addr;
   struct hostent *host;

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

   memset(&addr, 0, sizeof(addr));
   addr.sin_addr.s_addr = inet_addr(argv[1]);
   host = gethostbyaddr((char*)&addr.sin_addr, 4, AF_INET);//실제 ip 값을
통하여 호스트 얻음
```

```
  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");

  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]));

  return 0;
}
```