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

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – hoseong Lee(이호성)

hslee00001@naver.com

네트워크

술게임만들기

시그널 , 메모리, 프로세스 관련 함수 재정리.
*Signal ()
void (*signal(int signum, void (*handler)(int)))(int);
int signum : 시그널 번호 , void (*handler)(int) : 시그널을 처리할 핸들러 , void(*)(int) : 이전에 설정된 시그널 핸들러 , 반환

```c
// 아래 예제에서는 while() 문을 이용하여 계속 문자열을 출력하면서
// Ctrl-C 로 SIGINT가 발생하면 바로 종료하는 것이 아니라
// 프로그램에서 작성된 함수를 실행한 후에
// 다시 Ctrl-C 키가 눌리면 기존의 방법에 따라
// 프로세스가 종료되는 것을 보여준다.

#include <stdio.h>
#include <signal.h>
#include <unistd.h>

void (*old_fun)( int);

void sigint_handler( int signo)
{
    printf( "Ctrl-C 키를 누르셨죠!!\n");
    printf( "또 누르시면 종료됩니다.\n");
    signal( SIGINT, old_fun);   // 또는 signal( SIGINT, SIG_DFL);
}

int main( void)
{
    int i;
    old_fun = signal( SIGINT, sigint_handler);
    while(1 )
    {
        printf( "%d\n",i);
        sleep(1);
        i++;
    }
}
```

```
lee@lee-Lenovo-YOGA-720-13IKB:~/my_proj/lhs/lec/c$ ./a.out
0
1
2
3
4
^CCtrl-C 키를 누르셨죠!!
또 누르시면 종료됩니다.
5
6
7
8
^C
```

→ SIGINT :control-c 나 delete 키를 입력했을 때, 발생한다.
369 게임에 적용.

프로세스 함수

*wait - <sys/wait.h>
자식 프로세스 작업이 끝날 때까지 대기하며, 자식 프로세스가 종료한 상태를 구함.
pid_t wait(int *status); → status – wait 가 복귀될 때 자식프로세스의 상태 정보를 나타낸다.

*waitpid - <sys/wait.h>
wait()함수처럼 자식 프로세스가 종료될 때까지 대기함. 차이점은 wait() 함수가 자식 프로세스 중 어느 하나라도 종료되면 복귀되지만, 즉 대기에서 풀리지만 waitpid() 는 특정 자식 프로세스가 종료될 때까지 대기.
pid_t waitpid(pid_t pid, int *status, int option)
Pid- 부모가 기다리고 싶은 자식 프로세스의 프로세스 id
status – wait 가 복귀할때 자식프로세스의 상태 정보를 나타냄.
Options - <sys/wait.h> 에 정의 된 여러 값을 취할 수 있음.

→ 성공시 자식프로세스의 아이디를 반환 실패시 -1 리턴

*execl

*execv

* signal 과 fork 함수

```c
#include<stdio.h>
#include<stdlib.h>
#include<signal.h>
#include<unistd.h>
#include<sys/types.h>

#define MAXBUF 8162

void handler(int sig)
{
    pid_t pid;

    if((pid = waitpid( -1, NULL, 0))< 0)
    {   //waitpid 함수는 블록된 특정 프로세스가 종료할때 까지 현재 프로세스를 대기 시킨다.
        // -1 인자 값으로 임의의 프로세스가 종료할 때까지 부모 프로세스는 대기한다.
        // waitpid 함수를 통해 부모프로세스는 종료되는 자식프로세스의 pid를 확인한다.

        printf("waitpid error\n");
        exit(0);
    }
    printf("Handler reaped child %d\n", (int)pid);
    sleep(2);
}


int main()
{

    int i, n;
    char buf[MAXBUF];

    if(signal(SIGCHLD, handler)==SIG_ERR)
    {
        printf("signal error");
        exit(0);
    }

    for( i=0; i<3; i++){
        if( fork() == 0) {
            printf("Hello from child %d\n",(int)getpid());
            sleep(1);
            exit(0);
        }
    }

    if(( n= read(STDIN_FILENO, buf, sizeof(buf))) <0)
    {
        printf("read error");
        exit(0);
    }

    printf("Parent processing input\n");
    while(1){}

    exit(0);
}
```

gethostbyaddr,name.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)); // 초기화
```

```
   addr.sin_addr.s_addr = inet_addr(argv[1]);//네트워크를 어드레스파일로 바꿈
   host = gethostbyaddr((char *)&addr.sin_addr,4,AF_INET); //실제 ip 값으로 호스트를 찾음.

   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;
}
```

```
lee@lee-Lenovo-YOGA-720-13IKB:~/my_proj/lhs/network/4_2$ ./a.out 168.126.63.1
Official Name: kns.kornet.net
Address Type: AF_INET
IP Addr 1: 168.126.63.1
lee@lee-Lenovo-YOGA-720-13IKB:~/my_proj/lhs/network/4_2$ ./a.out 216.58.220.238
Official Name: nrt13s37-in-f238.1e100.net
Address Type: AF_INET
IP Addr 1: 216.58.220.238
```

mpecho_server,clnt.c

```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

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

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

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

```c
#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);
}

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

        if(str_len == 0)
            return;

        buf[str_len] = 0;
        printf("msg from server: %s",buf);
    }
```

```
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};

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);  //시그널 오면 액티드

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");
```

```
}

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

      if(!strcmp(buf,"q\n") || !strcmp(buf,"0\n"))
      {
         shutdown(sock,SHUT_WR);
         return ;
      }

      write(sock,buf,strlen(buf));
   }
}

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)
```

```
    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(); // 자식생성

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

        if(pid == 0){ //자식 어떠칼랭?
            close(serv_sock);  // 자식은 서버 소켓이 필요없거든..

            while((str_len = read(clnt_sock, buf, BUF_SIZE)) !=0)  //블록킹 새로운 클라이언트 기
달
                write(clnt_sock, buf, str_len);  //에코: 읽고 쏴준다.

            close(clnt_sock);
            puts("Client Disconnected...");
            return 0;
        }
        else
            close(clnt_sock);
    }
    close(serv_sock);

    return 0;
}
```

```
        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)  // 자식은 쓰고
        write_routine(sock,buf);
    else  // 부모는 읽고 있다.
        read_routine(sock,buf);

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

구조체 보내기

구조체 헤더

```
#ifndef __COMMON_H__
#define __COMMON_H__

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

typedef struct sockaddr_in      si;
typedef struct sockaddr *       sp;

typedef struct __d{
        int data;
        float fdata;
} d;

#define BUF_SIZE                    32

#endif
```

.common.h.swp

서버

```
#include "common.h"
#include <signal.h>
#include <sys/wait.h>
```

```
typedef struct sockaddr_in      si;
typedef struct sockaddr *       sp;

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

void read_cproc(int sig)
{
        pid_t pid;
        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, len, state;
        char buf[BUF_SIZE] = {0};
        si serv_addr, clnt_addr;
        struct sigaction act;
        socklen_t addr_size;
        d struct_data;
        pid_t pid;

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

```
act.sa_handler = read_cproc;
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));
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, 5) == -1)
	err_handler("listen() error");

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

	if(clnt_sock == -1)
		continue;
	else
		puts("New Client Connected!\n");

	pid = fork();

	if(pid == -1)
```

```
        {
            close(clnt_sock);
            continue;
        }

        if(!pid)
        {
            close(serv_sock);

            while((len = read(clnt_sock, (d *)&struct_data, BUF_SIZE)) != 0)
            {
                printf("struct.data = %d, struct.fdata = %f\n", struct_data.data, struct_data.fdata);
                write(clnt_sock, (d *)&struct_data, len);
            }

            close(clnt_sock);
            puts("Client Disconnected!\n");
            return 0;
        }
        else
            close(clnt_sock);
    }
    close(serv_sock);

    return 0;
}
```

클라이언트

```
#include "common.h"

void err_handler(char *msg)
```

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

void read_proc(int sock, d *buf)
{
	for(;;)
	{
		int len = read(sock, buf, BUF_SIZE);

		if(!len)
			return;

		printf("msg from serv: %d, %f\n", buf->data, buf->fdata);
	}
}

void write_proc(int sock, d *buf)
{
	char msg[32] = {0};

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

		if(!strcmp(msg, "q\n") || !strcmp(msg, "Q\n"))
		{
			shutdown(sock, SHUT_WR);
			return;
		}

		buf->data = 3;
```

```
            buf->fdata = 7.7;

            write(sock, buf, sizeof(d));
        }
}

int main(int argc, char **argv)
{
        pid_t pid;
        int i, sock;
        si serv_addr;
        d struct_data;
        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, (sp)&serv_addr, sizeof(serv_addr)) == -1)
            err_handler("connect() error");
        else
            puts("Connected!\n");
```

```
	pid = fork();

	if(!pid)
		write_proc(sock, (d *)&struct_data);
	else
		read_proc(sock, (d *)&struct_data);

	close(sock);

	return 0;
}
```

gcc -DDEBUG load_test.s

#if DDEBUG –> 위에 gcc 시 DDEBUGf 를 입력하지않으면 이루프를 제외하고 디버깅함.

#endif

1 채팅방 도배차단기능

2. 자료구조
네트워크 구현

3. 369 게임

4. 데몬화 시켜 절대 죽지 않게 하기.