TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 : Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 : 황수정
sue100012@naver.com
29 일차 (2018. 04. 02)

※ 리눅스에서 새로 생성된 스레드에서 getpid()를 호출 했을때 메인 프로세스의 pid 를 리턴하는 경우가 있다. 그럴때는 getpid() 대신 syscall(__NR_gettid)를 호출 하면 해당 스레드의 pid 를 얻을 수 있다.

gethostbyaddr

도메인 이름에 대한 인터넷 주소 목록을 가져온다.

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdlib.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)); //소켓 주소 초기화
   addr.sin_addr.s_addr = inet_addr(argv[1]); // 입력받은 인자를 네트워크 주소변환을 하여 addr 에 입력
   host = gethostbyaddr((char *)&addr.sin_addr, 4, AF_INET); //실제 ip 값으로 호스트 찾기

   if(!host)        //host 값이 없으면 에러, 방어벽으로 막아도 이걸로 뜬다.
      err_handler("gethost error!");

   printf("Official Name :%s₩n", host->h_name);          //도메인 이름 출력

   for(i=0; host->h_aliases[i]; i++)        // 별칭
       printf("Aliases %d : %s₩n", i+1, host->h_aliases[i]);

   printf("Address Type : %s₩n", (host->h_addrtype == AF_INET) ? "AF_INEF" : "AF_INET6"); //주소 타입
```

```
   for(i=0; host->h_addr_list[i]; i++)      // IP 리스트, 접속 가능한 다른 IP 주소 출력
      printf("IP Addr %d: %s₩n", i+1, inet_ntoa(*(struct in_addr *)host->h_addr_list[i]));

   return 0;       }
```

결과

```
./a.out 125.209.222.141
gethost error!
./a.out 50.63.202.50
Official Name :ip-50-63-202-50.ip.secureserver.net
Address Type : AF_INEF
IP Addr 1: 50.63.202.50
./a.out 168.126.63.1
Official Name :kns.kornet.net
Address Type : AF_INEF
IP Addr 1: 168.126.63.1
```

어떤 사이트는 host error 가 뜨는데 이는 방어벽으로 막아 놓아서 그렇다.

에코 프로그램

클라이언트 프로세스에서 fork()를 통해 프로세스를 생성하여 에코로 진행한 예제이다. 부모 프로세스에서는 read 를 하고, 자식 프로세스에서 는 write 를 수행한다. write, read 를 병렬적으로 수행한다.

mpecho_serv.c

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <stdlib.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 올 때까지 기다린다. 자식 프로세스가 종료되면 수행한다.
{  pid_t pid;
   int status;
   pid = waitpid(-1, &status, WNOHANG); //자식 프로세스 리턴(종료되지 않아도)한다. 블락 상태가 아님
   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};

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

    act.sa_handler = read_childproc; // read_childproc 를 핸들러에 저장해서 시그널 받으면 구동.
    sigemptyset(&act.sa_mask);  // act.sa_mask 를 비워준다. sigemptyset()이 시그널 셋을 비우는 함수
    act.sa_flags = 0;
    state = sigaction(SIGCHLD, &act, 0); //시그널 오면 act 구동.

    serv_sock = socket(PF_INET, SOCK_STREAM, 0); //소켓 생성

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

    memset(&serv_addr, 0, sizeof(serv_addr)); //0 으로 초기화
    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);//여러 사람을 받기 위해
         if(clnt_sock == -1) //accept 에러 나면 for 문 다시 돌기
           continue;
         else
           puts("New Client Connected ...");
         pid = fork(); // 프로세스 생성 부모와 자식 프로세스
         if(pid == -1) //프로세스 오류 시, 클라이언트 소켓 닫고 for 문 다시 돌기
           {     close(clnt_sock);
                 continue;                       }
          if(pid == 0) //자식
         {       close(serv_sock); //서버 소켓 닫기. 클라이언트에만 쏠꺼니까 필요 없다

                 while((str_len = read(clnt_sock, buf, BUF_SIZE)) != 0) //clnt_sock 읽어서 buf 에 쓴다. 블로킹
                   write(clnt_sock, buf, str_len); //에코. 읽은 값을 쓴다.

                 close(clnt_sock);
                 puts("Client Disconnected ... "); //종료가 안된다. read 가 블로킹이니까 클라이언트 들어올
때까지 계속 기다림
                 return 0;
          }
          else
            close(clnt_sock); //클라이언트 소켓 닫고 새로운 클라이언트가 붙길 기다린다
     }
        close(serv_sock);
        return 0;                   } // 클라이언트가 많으면 빠르게 갱신이 안 된다.

mpecho_clint.c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <stdlib.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) //서버에서 날아온거 읽어서 뿌려줌
{   for(;;) //무한 루프
   {      int str_len = read(sock, buf, BUF_SIZE); //소켓 읽어서 buf 에 저장

    if(str_len == 0) //읽을 값이 없으면 함수 나간다.
          return;
    buf[str_len] = 0;
    printf("msg form server : %s", buf);
   }
}

void write_routine(int sock, char *buf) //키보드에 입력한 것을 서버로 송신
{   for(;;)
   {   fgets(buf, BUF_SIZE, stdin);

       if(!strcmp(buf, "q₩n") || !strcmp(buf, "Q₩n")) //q,Q 입력하면 해당 소켓 닫기
       {     shutdown(sock, SHUT_WR);
             return;                          } //리턴하면서 끝난다.

   write(sock, buf, strlen(buf));    //q, Q 가 아니면 소켓에 쓴다.
   }
}

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(); //프로세스 생성

   if(pid == 0) //자식 프로세스 , write
     write_routine(sock, buf);
   else //부모 프로세스, read
     read_routine(sock, buf);

  close(sock);

   return 0;       }
```

서버와 클라이언트 간의 구조체 전달

```
#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
```

구조체 전달 서버

```
#include "common.h" //헤더파일에 구조체 넣기 위해서
#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; //나머지는 동일. 구조체 전달을 위해
         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); //이 다음부터 신경쓰면 된다.

		if(clnt_sock == -1)  // accept 에러 시, for 문 다시 돌기
			continue;
		else	//성공
			puts("New Client Connected!₩n");

		pid = fork(); //프로세스 생성

		if(pid == -1) //프로세스 에러시, 소켓 닫고 for 문 다시 돌기
		{  close(clnt_sock);
		   continue;		}

		if(!pid)  // 자식 프로세스
		{   close(serv_sock);  //  서버 소켓 닫기
		    while((len = read(clnt_sock, (d *)&struct_data, BUF_SIZE)) != 0) // 블로킹
			{	printf("struct.data = %d, struct.fdata = %f₩n", struct_data.data,
struct_data.fdata);// 들어온 구조체가 맞는지 출력으로 확인
				write(clnt_sock, (d *)&struct_data, len);//클라이언트로 받은 값 쓴다.
			}

			close(clnt_sock);
			puts("Client Disconnected!₩n");
			return 0;
		}
		else  // 부모 프로세스
			close(clnt_sock);
	}
	close(serv_sock);
```

```
        return 0;       }
```

구조체 전달 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) // 읽을 값이 없을 때 끝난다.
          return;

        printf("msg from serv: %d, %f₩n", buf->data, buf->fdata); // 읽은 구조체 출력
    }
}

void write_proc(int sock, d *buf) //d 포인터 = 구조체
{   char msg[32] = {0};
  for(;;)
    {   fgets(msg, BUF_SIZE, stdin);

        if(!strcmp(msg, "q₩n") || !strcmp(msg, "Q₩n")) //q, Q 가 오면 끝
          {  shutdown(sock, SHUT_WR);
             return;                                    }

        buf->data = 3; //구조체
        buf->fdata = 7.7; //구조체

        write(sock, buf, sizeof(d)); // 소켓에 쓴다.
    }
}

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) // 자식 프로세스
                write_proc(sock, (d *)&struct_data);
        else    // 부모 프로세스
                read_proc(sock, (d *)&struct_data);

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

통신 네트워크

서버

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/epoll.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;

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

void send_msg(char *msg, int len)
{  int i;
  pthread_mutex_lock(&mtx); // 임계 영역 잠금으로 다른 쓰레드 접근 막음
  for(i=0; i<clnt_cnt; i++)
     write(clnt_socks[i], msg, len); // 해당 클라이언트 소켓에서 메시지를 적음.
  pthread_mutex_unlock(&mtx);          }

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

while((str_len = read(clnt_sock, msg, sizeof(msg))) != 0)//클라이언트에서 전송된 데이터를 읽고 값 있을 때
      send_msg(msg, str_len);

  pthread_mutex_lock(&mtx);
  for(int i=0; i < clnt_cnt; i++) // 클라이언트 중복 확인
   {   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;

  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, 5) == -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; //accept 한 것 clnt_socks 배열에 저장
    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;        }

#include <stdio.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <string.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")) //Q,q 일 때 종료
      {  close(sock);
         exit(0);                  }

     sprintf(name_msg, "%s %s", name, msg); //name 과 msg 를 배열에 저장
     write(sock, name_msg, strlen(name_msg)); // 소켓에 저장한 배열을 쓴다.
   }
   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);

  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;        }
```

연산하는데 걸리는 시간 / 디버깅시 특정 코드 부분의 수행시간 – 도배 하는 사용자를 퇴출하기 위한 작업

헤더파일

```
#ifndef __LOAD_TEST_H__

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

typedef struct timeval  tv; //ls 관련 모듈 볼 때 나온것

void get_runtime(tv, tv);

#endif

#include "load_test.h"
void get_runtime(tv start, tv end)
{  end.tv_usec = end.tv_usec - start.tv_usec; //수행시간의 마이크로 초 단위를 구한다.
   end.tv_sec = end.tv_sec - start.tv_sec;  // 수행시간의 초 단위를 구한다.
   end.tv_usec += end.tv_sec * 1000000; // 마이크로 단위로 맞추어준다.

   printf("runtime = %lf sec₩n", end.tv_usec / 1000000.0); // 수행시간을 마이크로 단위로 출력
}

#if DEBUG // -디버그 옵션 주는 이유. 주석하고 비슷하다. 디버깅 모드에서만 실행하고 일반모드는 pass
int main(void)
{  unsigned int i, cnt = 0;
   tv start, end;

   gettimeofday(&start, NULL); //오늘 현재 시간이 세팅이 되는 것

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

 gettimeofday(&end, NULL); // 실행 후, 현재 시간 세팅

 get_runtime(start, end);      // 연산 걸린 시간

          return 0;          }
#endif
```