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

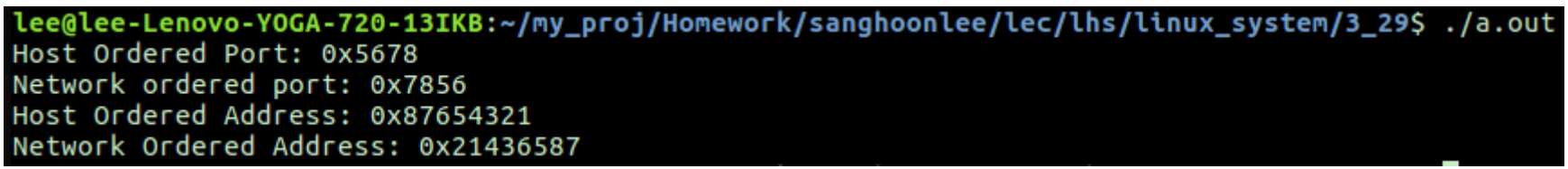
convert_endian.c

```
include <stdio.h>
//  변수는 메모리에 저장하는 공간
int main(void)
{
    unsigned short host_port = 0x5678;
    unsigned short net_port;
    unsigned long host_addr = 0x87654321;
    unsigned long net_addr;


    net_port = htons(host_port);
    net_addr = htonl(host_addr);


    printf("Host Ordered Port: %#x\n",host_port);
    printf("Network ordered port: %#x\n",net_port);
    printf("Host Ordered Address: %#lx\n", host_addr);
    printf("Network Ordered Address: %#lx\n",net_addr);
    return 0;
}
```

// 크로스 매칭을 하는 이유 컴퓨터마다 받아드리는 순거가 다름 그래서 꼬이지말라고 바꿔주는것



3/30 inet_addr.c

```
#include <stdio.h>
#include <arpa/inet.h>

int main(int argc, char **argv)
{
   char *addr1 = "3.7.8.9";  // 빅엔디안, 리틀엔디안 서로다른형식으로 받으면 어떻게 될까연
   //메모리에 09 08 07 03 으로 저장 --> 빅엔디안 : 순서대로 받는다. 03 07 08 09
   //                                --> 리틀엔디안 : 09.05.07.03 으로 받는다
   char *addr2 = "1.3.5.7";

   unsigned long conv_addr = inet_addr(addr1);
   if(conv_addr == INADDR_NONE)
      printf("Error!\n");
   else
      printf("Network Ordered integer Addr: %#lx\n",conv_addr);

   conv_addr = inet_addr(addr2);
   if(conv_addr == INADDR_NONE)
      printf("Error!\n");
   else
      printf("Network Ordered integer Addr: %#lx\n",conv_addr);
   return 0;
}
// 애초에 메모리에 들어가는 형식으로 변환 후 전달하는 개념
```

```
lee@lee-Lenovo-YOGA-720-13IKB:~/my_proj/Homework/sanghoonlee/lec/lhs/linux_system/3_29$ ./a.out
Network Ordered integer Addr: 0x9080703
Network Ordered integer Addr: 0x7050301
```

3/30 inet_aton.c

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

typedef struct sockaddr_in si;
void err_handler(char *msg)
{
    write(2,msg,strlen(msg));
    exit(1);
}

int main(int argc, char **argv)
{
    char *addr = "127.124.73.31";
    si addr_inet;

    if(!inet_aton(addr, &addr_inet.sin_addr))
        err_handler("Conversion Eroor!");
    else
        printf("Network Ordered integer Addr: %#x\n",addr_inet.sin_addr.s_addr);
    return 0;
}
```

메모리에 저장할때, 크로스매칭한다
기본적으로는 네트워크는 빅엔디안방식을 적용한다.

변환과정

호스트	네트워크	호스트
리틀(인텔)	빅	빅
빅	빅	리틀

호스트가 빅엔디안인지 리틀엔디안인지 어떻게 아나?
호스트 정보에 통신하려는 대상이 빅인지 리틀인지 나와있다.

네트워크에서 넘어가는 것은 무조건 빅으로 보내니까 호스트에서 알아서 처리하면된다.
네트워크 주소로 바꾼다는 것은 기준을 준다는것.

```
lee@lee-Lenovo-YOGA-720-13IKB:~/my_proj/Homework/sanghoonlee/lec/lhs/network/3_30$ ./a.out
Network Ordered integer Addr: 0x1f497c7f
```

3/30 inet_ntoa.c

```c
#include <stdio.h>
#include <string.h>
#include <arpa/inet.h>

typedef struct sockaddr_in si;

int main(int argc, char **argv)
{
    si addr1, addr2;
    char *str;
    char str_arr[32] = {0};

    addr1.sin_addr.s_addr = htonl(0x10203040); //호스트를 네트워크로 바꾸겠다
                                               // 크로스 매칭됨
    addr2.sin_addr.s_addr = htonl(0x12345678);

    str = inet_ntoa(addr1.sin_addr); // 네트워크를 호스트로 바꾸겠다.
    strcpy(str_arr,str);
    printf("Not 1: %s\n",str); //잘복사됐나 그냥확인

    inet_ntoa(addr2.sin_addr);
    printf("Not 2: %s\n",str); //10진수로 12. 34. 56.  78 → 18. 52.86.120
    printf("Not 3: %s\n",str_arr);
    return 0;
}
```

```
lee@lee-Lenovo-YOGA-720-13IKB:~/my_proj/Homework/sanghoonlee/lec/lhs/network/3_30$ ./a.out
Not 1: 16.32.48.64
Not 2: 18.52.86.120
Not 3: 16.32.48.64
```

echo_server.c

```
#include <stdio.h>
#include <stdlib.h>
#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         1024

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

int main(int argc,char **argv)
{
    int i, str_len;
    int serv_sock, clnt_sock;
    char msg[BUF_SIZE];

    si serv_addr, clnt_addr;
    socklen_t clnt_addr_size;
```

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

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;                    // TCP 사용
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);   //자기자신을 주소로 받음 로컬호스트
serv_addr.sin_port = htons(atoi(argv[1]));          // 포트번호

if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1) //bind는 서버에 ip 주소 셋팅
   err_handler("bind() error");
 if(listen(serv_sock, 5) == -1)  //5명 받겠다. 그이상은 x
   err_handler("listen() error");
// 리슨함수 호출하면 클라이언트의 접속 요청이 올때까지 대기상태가 된다. 즉 블록된 모습이다.
// 함수가 리턴이 되었을 때는 클라이언트의 접속이 요청됐다던지, 아니면 에러가 발생했을 경우이다.
// 에러없이 함수가 복귀했다면 클라이언트 접속요청이므로 요청을 허락한다.
clnt_addr_size = sizeof(clnt_addr);
for(i=0;i<5;i++)
{
   clnt_sock = accept(serv_sock, (struct sockaddr *)&clnt_addr, &clnt_addr_size);
   //접속요청을하면 자동으로 소켓이 생성된다.
   // 서버 소켓이 클라이언트를 기다린다 ( 실제 클라이언트 기다리는 구간은 listen이다)
   // sockaddr  찍어보면 접근한 ip주소가 나온다.
   // 원격으로 세마포어를 만드는것. 따라서 read write 등 모두 할 수 있다.
```

```
        if(clnt_sock == -1)
            err_handler("accept() error");
        else
            printf("Connected Client %d\n", i+1);

        while((str_len = read(clnt_sock, msg, BUF_SIZE)) != 0)  // read는 blocking 함수
                write(clnt_sock, msg, str_len);
       // 여기서 터미널 한창에서 메세지를 치는동안 나머지는 처리를 하지못했다
      // 그렇다면 여기서 read,write를 논블럭으로 바꿔주면 자연스레 잘 처리하게된다.
        close(clnt_sock);
    }
    close(serv_sock);

    return 0;
}
```

```
lee@lee-Lenovo-YOGA-720-13IKB:~/my_proj/Homework/sanghoonlee/lec/lhs/network/3_30$ ./server 7788
Connected Client 1
Connected Client 2
Connected Client 3
Connected Client 4
Connected Client 5
```

3/30 echo_client.c

```
#include <stdio.h>
#include <stdlib.h>
#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          1024

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

int main(int argc,char **argv)
{
    int sock,  str_len;
    si serv_addr;
    char msg[32];
    char *m = "Input Message(q to quit): ";

    if(argc !=3){ //argc 가 3개 ip주소를 담는다.  사설 ip주소는 무조건 192.168.0.x임
```

```
    printf("use: %s <IP> <port>\n", argv[0]);
    exit(1);
}

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

if(sock == -1)
    err_handler("socket() error");
memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;                    // TCP 사용
serv_addr.sin_addr.s_addr = inet_addr(argv[1]); //아이피 주소
serv_addr.sin_port = htons(atoi(argv[2]));         // 포트번호

if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1) //서버에 접속을 시도
    err_handler("connect() error");
else
    puts("Connected ......");

for(;;)
{
    fputs("Input msg(g to quit): ",stdout);
    fgets(msg,BUF_SIZE,stdin);

    if(!strcmp(msg, "q\n") || !strcmp(msg,"Q\n"))
        break;

    write(sock, msg, strlen(msg));  // 접속 성공하면 msg를 전송
    str_len = read(sock, msg, BUF_SIZE -1); // 서버에서 온것을 다시 읽은것

    if(str_len == -1)
        err_handler("read() error!");
```

```
        msg[str_len]=0;
        printf("msg from serv: %s\n", msg);
    }
    close (sock);
    return 0;
}
```

homework

→ fork를 사용해서 서버에서 read, write 블록킹 된부분 논블록으로 바꿔서 프로세스가 바로바로 처리할 수 있게끔 처리해줘보자.

3/30 op_server.c

```
#include <stdio.h>
#include <stdlib.h>
#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        1024
#define OPSZ            4

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

int calculate(int opnum, int *opnds, char op) //opnum: 인자가 몇개인지.
{
    int result = opnds[0], i;

    switch(op)
```

```
    {
        case '+':
            for(i=1; i <opnum;i++)  //i=1부터인 이유는 전에 opnds[0]을 먼저 더했음.
                result += opnds[i];
            break;
        case '-':
            for(i=1; i <opnum;i++)
                result -= opnds[i];
            break;
        case '*':
            for(i=1; i <opnum;i++)
                result *= opnds[i];
            break;
    }
    return result;
}

int main(int argc, char **argv)
{
    int serv_sock, clnt_sock;
    char opinfo[BUF_SIZE];

    int result, opnd_cnt, i;
    int recv_cnt, recv_len;

    si serv_addr, clnt_addr;
    socklen_t clnt_addr_size;

    if(argc !=2)
```

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

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;                 // TCP 사용
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);   //자기자신을 주소로 받음 로컬호스트
serv_addr.sin_port = htons(atoi(argv[1]));       // 포트번호

if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1) //bind는 서버에 ip 주소 셋팅
   err_handler("bind() error");

if(listen(serv_sock, 5) == -1)  //5명 받겠다. 그이상은 x
   err_handler("listen() error");
// 리슨함수 호출하면 클라이언트의 접속 요청이 올때까지 대기상태가 된다. 즉 블록된 모습이다.
// 함수가 리턴이 되었을 때는 클라이언트의 접속이 요청됐다던지, 아니면 에러가 발생했을 경우이다.
// 에러없이 함수가 복귀했다면 클라이언트 접속요청이므로 요청을 허락한다.

clnt_addr_size = sizeof(clnt_addr);

for(i=0;i<5;i++)
{
   opnd_cnt = 0;
```

```
        clnt_sock = accept(serv_sock, (struct sockaddr *)&clnt_addr, &clnt_addr_size);
        read(clnt_sock, &opnd_cnt,1);

        recv_len = 0;
        while((opnd_cnt * OPSZ + 1) > recv_len) // while문은 읽기 시 끊겼을 때 다시 읽기위해 루프를 돌린것
        {
            recv_cnt = read(clnt_sock, &opinfo[recv_len], BUF_SIZE -1);  // 읽은 바이트수 리턴
            recv_len += recv_cnt;
        }
        result = calculate(opnd_cnt, (int *)opinfo, opinfo[recv_len -1]); //  recv_len  배열의 시작은 0부터기때문에 -1번째
        write(clnt_sock, (char *)&result, sizeof(result)); //&result 결과값이 client 로 날아간다.

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

}
```

→ 클라이언트와는 다르게 서버에서 read를 끊어서 계속 읽었어야했다. 그 이유는 데이터를 끊어서 넣기 때문에

3/30 op_client.c

```
#include <stdio.h>
#include <stdlib.h>
#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        1024
#define RLT_SIZE          4
#define OPSZ              4

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

int main(int argc, char **argv)
{
    int i, sock, result, opnd_cnt;
    char opmsg[BUF_SIZE] = {0};
```

```
si serv_addr;

if(argc !=3)
{
   printf("use: %s <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;                 // TCP 사용
serv_addr.sin_addr.s_addr = inet_addr(argv[1]);   //자기자신을 주소로 받음 로컬호스트
serv_addr.sin_port = htons(atoi(argv[2]));       // 포트번호

if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1) //서버에 접속을 시도
   err_handler("connect() error");
else
   puts("Connected ......");

fputs("Operand Cnt: ", stdout);
scanf("%d" ,&opnd_cnt);

opmsg[0] = (char)opnd_cnt;

for(i=0; i<opnd_cnt;i++)
{
```

```
        printf("Operand %d ",i+1);
        scanf("%d",(int *)&opmsg[i*OPSZ+1]);
    }

    fgetc(stdin);
    fputs("Operator: ",stdout);
    scanf("%c",&opmsg[opnd_cnt * OPSZ +1]);
    write(sock, opmsg,opnd_cnt * OPSZ +2);
    read(sock,&result,RLT_SIZE);

    printf("Operation result: %d\n",result);
    close(sock);

    return 0;

}
```

술게임 만들어보자

함수제한x 시간제한 3초 횟수 카운트 1~3333

서버

```
/* For Network */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>

/* For System */
#include <time.h>
#include <fcntl.h>
#include <signal.h>
#include <stdbool.h>
#include <sys/wait.h>

typedef struct sockaddr_in  si;
typedef struct sockaddr *   sap;

#define BUF_SIZE        1024
#define OPSZ            4

int glob_cnt;

void sig_handler(int signo)
{
   printf("Time Over\n");
   glob_cnt++;
}

void make_game(int *data)
{
   *data = rand() % 3333 + 1;
}
void start_game(int data,int clnt_sock)  // 요기에 clnt_sock 을 추가해야함.
{
```

클라이언트

```
#include <stdio.h>
#include <stdlib.h>
#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        1024
#define RLT_SIZE        4
#define OPSZ            4

void err_handler(char *msg)
{
   fputs(msg, stderr);
   fputc('\n', stderr);
   exit(1);
}
int main(int argc, char **argv)
{
   int i, sock, result, nread;
   int opnd_cnt;
   char buf[BUF_SIZE] = {0};
   char buf2[BUF_SIZE]  = {0};
   si serv_addr;

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

```
    char buf[32] = {0};
    bool fin;
    int i, cmp;

    for(;;)
    {
        signal(SIGALRM, sig_handler);
        alarm(3);
        read(clnt_sock, buf, sizeof(buf));
        alarm(0);
        cmp = atoi(buf);
        printf("cmp = %d",cmp);
        fin = check_correct(data, cmp);

        if(fin)
        {
            break;
        }
        else
        {
            glob_cnt++;
            if(data > cmp){
                //          printf("%d 보다 크다\n", cmp);
                strcpy(buf,"크게써라\n");
                write(clnt_sock, buf, strlen(buf));
            }
            else{
                //      printf("%d 보다 작다\n", cmp);
                strcpy(buf,"작게써라");
                write(clnt_sock, buf, strlen(buf));
            }
        }
    }
}
void err_handler(char *msg)
{
    fputs(msg, stderr);
    fputc('\n', stderr);
    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 .....\n 정답은?: ");
    }
    for(;;)
    {
        scanf("%s",buf);
        write(sock,buf,strlen(buf));
        nread = read(sock, buf2, BUF_SIZE);
        write(1, buf2, nread);
    }

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

```
}

int main(int argc, char **argv)
{
    pid_t pid[5] = {0};
    int status;

    int serv_sock, clnt_sock;
    char opinfo[BUF_SIZE];

    int result, opnd_cnt, i;
    int recv_cnt, recv_len;

    si serv_addr, clnt_addr;
    socklen_t clnt_addr_size;

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

```
   //  --> listen까지 서버 소켓을 통해 클라이언트의 접속 요청을 확인하도록 설정
   for(i = 0; i < 5; i++)
   {
      pid[i] = fork(); //

      if(pid[i] > 0)
         wait(&status);
      else
      {
         int data;
         char buf[32] = "숫자를 맞춰봐!\n";

         srand(time(NULL));
         clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size); // 새소켓
생성
         make_game(&data);
         printf("data = %d\n",data);

         for(;;)
         {
            start_game(data,clnt_sock);
            //write(clnt_sock, buf, strlen(buf));
            glob_cnt++;
            if(glob_cnt > 10)
               break;
         }

         close(clnt_sock);
      }

   }
   close(serv_sock);

   return 0;
}
```

→ 미완성.. fork함수 많이 미흡…