2018. 3. 29 목 – 26 회차

과정 : TI, DSP, Xilinx Znq FPGA, MCU 기반의 프로그래밍 전문가 과정

Prof. 이상훈
gcccompil3r@gmail.com
Stu. 정상용
fstopdg@gmail.com

Network Programming

꿀팁 : 코드를 빠르게 해석하기 위하여 &를 먼저 찾아본다.

1. basic server & basic client
→ 패턴을 이해하는 것이 중요.(항상 이 패턴을 사용한다.)
→ 실행방법 : ./serv 7777 → open terminal & ./recv ip 7777
→ 숙지하고 있어야 할 내용.

1. port : 통로, 서비스 번호
(예약되어있는 port : 80(웹, www), 20, 21(파일트랜스프로토콜, 업로드 & 다운로드), 22(ssh) 이외에도 몇 개의 예약되어있는 번호가 존재) : 해당번호가 특정역할을 하고 있음
7777 : 우리가 만든 커스텀번호
2. socket : 네트워크에서 open 과 같은 역할.(socket 도 file 이다. : Unix 의 기본철학생각)
filedescriptor 가 반환된다.
3. 127.0.0.1 : local host(ifconfig 로 확인가능)

→ server code

```
#include <arpa/inet.h>
#include <sys/socket.h>

typedef struct sockaddr_in si;
typedef struct sockaddr *sap;

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

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

```
	int serv_sock;
	int clnt_sock;

	si serv_addr;
	si clnt_addr;
	socklen_t clnt_addr_size; //32bit

	char msg[] = "Hello Network Programming";
	if(argc != 2) //must write ‘port number(7777)’
	{
		printf("use: %s <port>\n", argv[0]);
		exit(1);
	}

	serv_sock = socket(PF_INET, SOCK_STREAM, 0); //socket is a kind of ‘open’.

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

	memset(&serv_addr, 0, sizeof(serv_addr)); //서버어드레스 구조체(si)
	serv_addr.sin_family = AF_INET;  //TCP(우리는 UCP 안 쓰므로 이 패턴 숙지)
	serv_addr.sin_addr.s_addr = htonl(INADDR_ANY); //어떤 IP 주소도 다 받겠다.
	serv_addr.sin_port = htons(atoi(argv[1]));

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

	if(listen(serv_sock, 5) == -1) //5 명까지만 받겠다.
		err_handler("listen() error");

	clnt_addr_size = sizeof(clnt_addr); //32bit
	clnt_sock = accept(serv_sock, (struct sockaddr *)&clnt_addr, &clnt_addr_size); //클라이언
트의 접속을 기다림.
	if(clnt_sock == -1)
		err_handler("accept() error");

	write(clnt_sock, msg, sizeof(msg));
	close(clnt_sock);
	close(clnt_sock);

	return 0;
}
```

→client code

```
#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;

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

int main(int argc, char **argv)
{
	int sock;
	int str_len;
	si serv_addr;
	char msg[32];

	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)); //서버어드레스를 초기화
	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");

	str_len = read(sock, msg, sizeof(msg) - 1);

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

	printf("msg from serv: %s\n", msg);
	close(sock);

	return 0;
}
```

2. read client
→ 예기치 못한 사고로 인하여 라우터가 망가지는 경우가 종종 생김.
그 경우를 대비하여 정보를 &msg 에 미리 저장하기 위한 코드

```
#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;

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

int main(int argc, char **argv)
{
	int sock;
	int str_len = 0;
	si serv_addr;
	char msg[32] = {0};
	int idx = 0, read_len = 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");

	while(read_len = read(sock, &msg[idx++], 1))
	{
		if(read_len == -1)
			err_handler("read() error!");

		str_len += read_len;
	}
```

```
	printf("msg from serv: %s\n", msg);
	printf("read count: %d\n", str_len);
	close(sock);

	return 0;
}
```

3. Socket 이 open 과 같음을 보여주는 예제
→ socket : open & sock : fd
→ socket stream : TCP
→ socket degram : UDP

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/socket.h>

int main(void)
{
	int fd[3];
	int i;

	fd[0] = socket(PF_INET, SOCK_STREAM, 0); //TCP
	fd[1] = socket(PF_INET, SOCK_DGRAM, 0);  //UDP
	fd[2] = open("test.txt", O_CREAT | O_WRONLY | O_TRUNC);

	for(i = 0; i < 3; i++)
		printf("fd[%d] = %d\n", i, fd[i]);

	for(i = 0; i < 3; i++)
		close(fd[i]);

	return 0;
}
```

4. little endian & big endian 이 메모리에 저장될 때
→ 변수에 오는 타입에 따라 litte endian , big endian 으로 나눠진다.
→ 데이터를 받을 시 little 이 big 을 받을 수도있고, big 에서 little 을 받을 수 도 있다. 서로 형태가 다르기때문에 메모리에 저장시 서로 알아볼 수 있는 형태로 저

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

int main(void)
{
	unsigned short host_port = 0x5678;  // 2byte
	unsigned short net_port;
	unsigned long host_addr = 0x87654321;  //4byte
	unsigned long net_addr;

	net_port = htons(host_port); //
```

```
	net_addr = htonl(host_addr); //host to network

	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;
}
```

5. 주소체계 또한 little endian & big endian
→ 맨 앞의 0 은 굳이 출력할 필요 x

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

int main(int argc, char **argv)
{
	char *addr1 = "3.7.5.9";
	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 Inteder 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;
}
```

추가 . Linux system programming
1. sigaction

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

struct sigaction act_new;
struct sigaction act_old;

void sigint_handler(int signo)
{
	printf("Ctrl + C\n");
	printf("If you push it one more time then exit\n");
	sigaction(SIGINT, &act_old, NULL);
}
```

```
int main(void)
{
	act_new.sa_handler = sigint_handler;
	sigemptyset(&act_new.sa_mask); //특정한 시그널이 들어오지 못하게 막
	sigaction(SIGINT, &act_new, &act_old); //액트뉴를 동작시킴음
	while(1)
	{
		printf("sigaction test\n");
		sleep(1);
	}
	return 0;
}
```

2. Thread

→ process 는 독립적

→ thread 는 종속적 -> 메모리를 공유

→ 그러므로 critical section 에서 문제가 생길 가능성 존재. → lock 을 걸어준다.

```
#include <stdio.h>
#include <pthread.h>

void *task1(void *X)
{
	printf("Thread A Complete\n");
}

void *task2(void *X)
{
	printf("Thread B Completet\n");
}

int main(void)
{
	pthread_t ThreadA, ThreadB; //pthread 지정

	pthread_create(&ThreadA, NULL, task1, NULL);
	pthread_create(&ThreadB, NULL, task2, NULL);

	pthread_join(ThreadA, NULL); //조인을 하는 순간 메모리에 올라간다.
	pthread_join(ThreadB, NULL);

	return 0;
}
```

3. kill 을 다른 Terminal 에서 동작

→ ./a.out & : 실행파일을 백그라운드에서 실행(현재 Terminal 에서 SIGINT 불가)

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

```
#include <stdlib.h>

void gogogo(int voidv)
{
	printf("SIGINT Accur!\n");
	exit(0);
}

int main(void)
{
	signal(SIGINT, gogogo);

	for(;;)
	{
		printf("kill Test\n");
		sleep(2);
	}

	return 0;
}
```

→ 다른 Terminal 에서 kill 을 구현

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

int main(int argc, char *argv[])
{
	if(argc < 2)
		printf("Usage : ./exe pid\n");
	else
		kill(atoi(argv[1]), SIGINT);

	return 0;
}
```