# 리눅스 시스템 관리

# “유/무선 으로 연결되어 있는 Device들의 집합 ”

## 유 / 무선 으로 연결

1. Bluetooth, Wi-Fi, RFID, 적외선 통신(IrDA) 등 근거리 무선 통신

2. WCDMA, LTE 등과 같은 이동 통신 기술

3. Ethernet, Serial 통신 등과 같은 유선통신

4. GPS

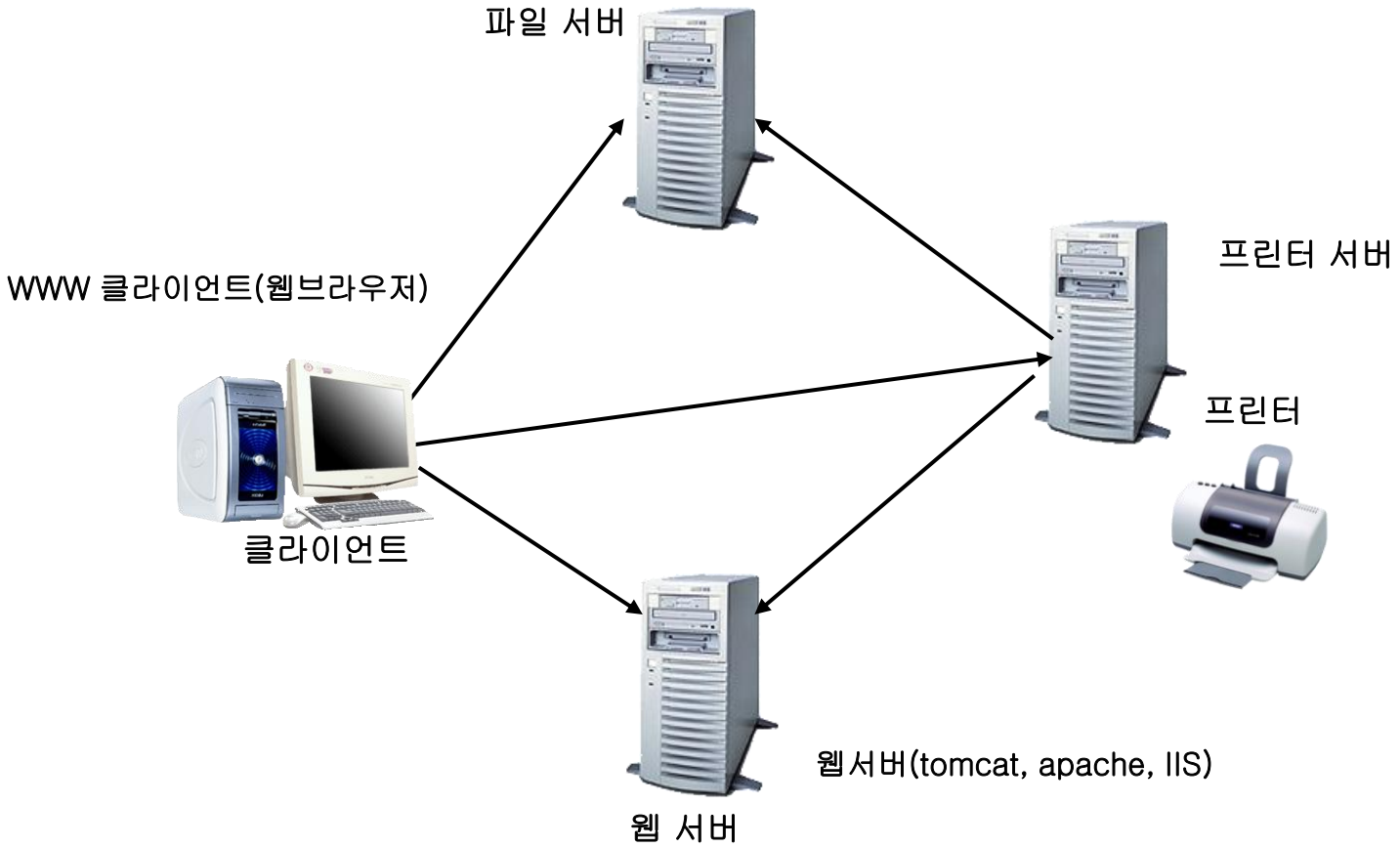
## Device( 디바이스 )

1.보통, 네트워크에 연결된 컴퓨터

2. 컴퓨터가 아닌 다른 Devices
   ex) 프린터, 모바일 디바이스, 가전제품, 웨어러블 컴퓨터 등 다양한 임베디드 제품들

3. 포괄적 개념으로 네트워크에 연결되어 있는 것들을 총칭해서 디바이스라 할 수 있다.

# “ 네트워크에 연결된 디바이스들 간에  미리 약속된 프로토콜 을 사용하여 데이터를  교환 하는 것”



## Protocol( 프로토콜 )

1. 약속
2. 디바이스 상호간 데이터 통신을 위해 필요한 규약
3. **통신 규약**

## 데이터 통신 방법

1. 두 개의 상이한 두 디바이스가 물리적으로 떨어져 있는 경우, 유/무선을 통해 네트워크 에 연결되어야 한다.

2. 연결된 두 디바이스는 전류나 전파, 빛 등의 방식으로 데이터 통신을 한다.

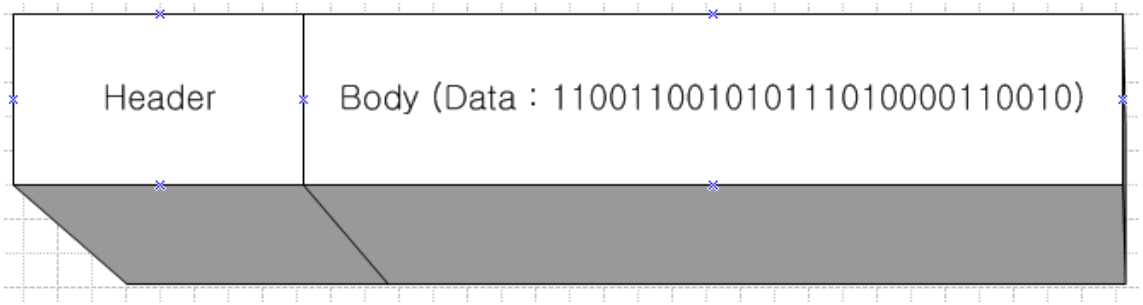
3. 이 데이터는 0/1 또는 on/off의 1bit로 표현되게 된다. (byte 단위 데이터 통신)

4. 주소( address )
   - 두 디바이스가 데이터 통신을 하기 위해서는 서로의 위치를 알아야 한다.
   - 이 위치를 네트워크에서는 node 라고 부른다.
   - 각 node 마다 고유의 주소를 가지고 있어야 한다.

5. 데이터 통신을 할 때는 데이터 외에 어디로 보내야 하는 가 또는 누가 보내는 가 등의 정보를 담고 있어야 한다.

## 데이터 통신 방법

6. 실제, 네트워크를 통한 데이터 통신을 할 때는 **Packet** 을 사용하게 된다.




(1) Header : 송신자/수신자 의 주소, 체크썸(checksum) 그리고 여러 제어 정보
(2) Body : 전송할 데이터를 byte 단위로 포함한다.

## 5.3.1 인터넷의 이해

1. 인터넷 != WWW(World Wide Web)

2. 인터넷 기반의 서비스

| 이름 | 프로토콜 | 포트 | 기능 |
|---|---|---|---|
| WWW | HTTP | 80 | 웹서비스 |
| Email | SMTP/POP3/IMAP | 25/110/114 | 이메일 서비스 |
| FTP | FTP | 21 | 파일 전송 서비스 |
| DNS | DNS | 23 | 네임서비스 |
| NEWS | NNTP | 119 | 인터넷 뉴스 서비스 |

### 인터넷 (Internet)

**TCP/IP** 기반의 네트워크가 전 세계적으로 확대되어 하나로 연결된 네트워크들의 네트워크 ( 네트워크의 결합체 )

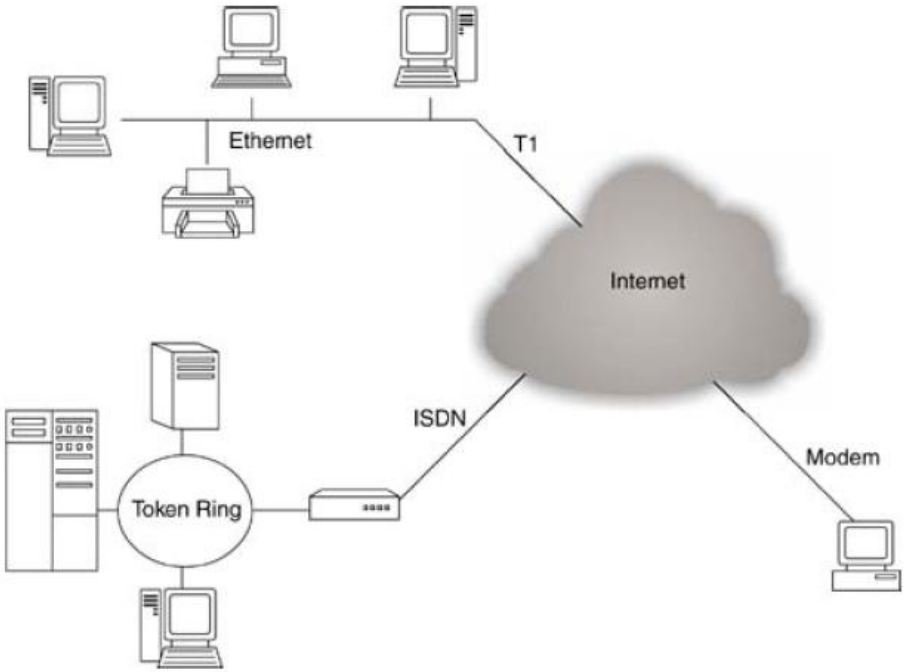
## 5.3.1 인터넷의 이해

### 인터넷의 역사

미국방성의 최초의 연구 목적의 네트워크 ARPANET(1969년) 이 시초

미 과학재단 네트워크인 NSFNET이 연결(1986년)

일반 산업 목적의 네트워크들이 연결(1990년 이후)



## 5.3.1 인터넷의 이해

### OSI 7계층과 TCP / IP 4계층

- 하드웨어, 운영체계, 접속 매체와 관계없이 동작할 수 있는 개방형 구조
- OSI 7 계층에서 4계층으로 단순화.

## 5.3.2 계층과 프로토콜

### Link 계층[ Link Layer ]

1. Host(호스트)간의 네트워크를 통한 데이터 통신을 위한 물리적 연결(유/무선)에 대한 표준
2. LAN, WAN, MAN과 같은 네트워크 구성을 정의
3. 상위 계층인 Intenet 계층에서 형성된 Packet(패킷) 을 전기신호 또는 광 신호로 바꾸어 전달하는 역할을 담당한다.
4. 응용(Application) 개발자가 직접 접근할 수 있는 계층이 아니고 보통 네트워크 장비나 드라이버 개발자 들이 관심을 가지는 계층이다.

### Internet 계층 [ Internet Layer ]

1. Link 계층을 통해 물리적으로 연결된 각 Host간의 Packet의 전달 경로를 결정한다.
2. IP(Internet Protocol)은 인터넷 계층에 존재하는 프로토콜이다.
3. IP 프로토콜에는 IP 주소(Address)를 부여하는 방법과 체계를 정의하고 있다.
4. IP 프로토콜은 IP 주소를 기반으로 경로를 Routing 하는 방법을 정의하고 있다.
5. Transport(전송) 계층과 함께 Internet에서 중요한 계층이다.
6. 데이터가 상대방에게 안전하게 전송되는 것을 보장하지 않는다.
7. Transport(전송) 계층이 데이터 전달에 대한 신뢰성을 책임진다는 가정하에 목적지로 Packet을 어떤 경로로 전송할 것인가에 대한 문제만 해결하는 계층이다.

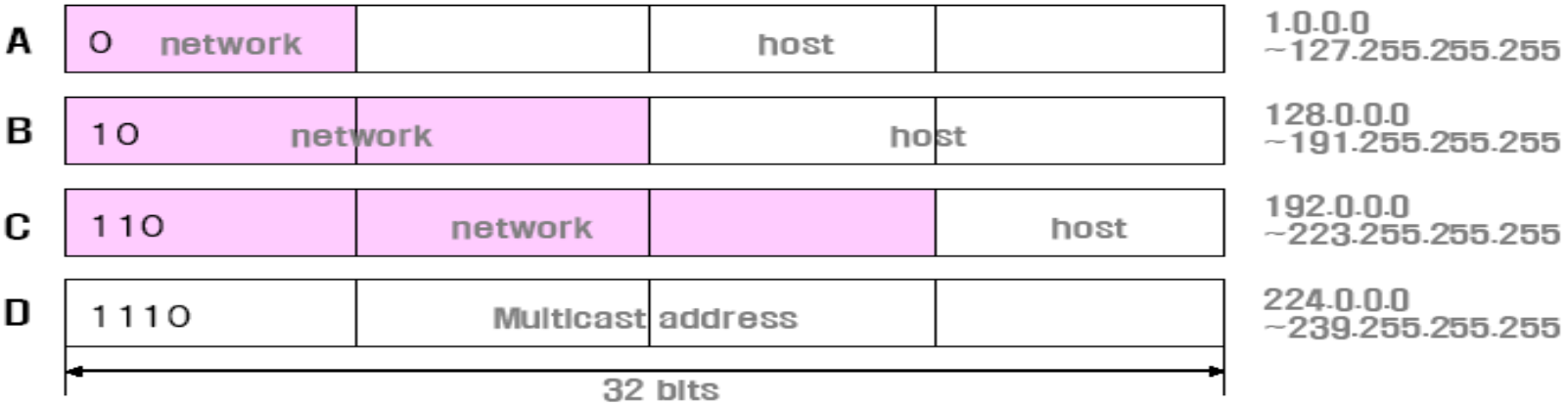
## 5.3.2 계층과 프로토콜

### Internet 계층 [ Internet Layer ]

8. IP 주소는 다음과 같이 2종류로 나뉜다.

IPv4 ( Internet Protocol Version 4) : 4byte 주소체계
IPv6 ( Internet Protocol Version 6) : 16byte 주소체계

9. 4Byte IP주소는 Network주소와 Host(호스트) 주소로  나뉜다.



## 5.3.2 계층과 프로토콜

### 전송 계층( Transport Layer )

1. 하위 계층 Internet 계층의 IP가 해결한 목적지까지의 네트워크상의 경로에서 실제 데이터를 전송하는 역할을 한다.
2. TCP와 UDP 라는 데이터의 전달을 책임지는 프로토콜이 존재한다.
3. IP는 하나의 Packet이 전달되는 과정에만 중심을 두고 설계 되었기 때문에 여러 Packet으로 나눠져 전달되는 데이터의 순서와 전송 자체는 신뢰할 수 없다.
4. 데이터의 순서와 신뢰할 수 있는 데이터 전송을 보장하는 역할을 전송 계층의 프로토콜이 맡는다.

5. TCP( Transmission Control Protocol)

(1) 연결지향 프로토콜이라 데이터를 전송/수신하기 전에 소켓을 통해 양쪽 연결이 성립
(2) 연결이 성립되면 TCP는 데이터의 손실이나 중복 없이 목적지에 확실하게 전달
(3) 만약 이때 보낸 데이터의 일부가 유실되었다면 수신자는 발신자에게 해당 데이터의 재전송을 요청한다.
(4) 흔히, TCP를 전화 통화와 비교된다.
(5) TCP는 UDP에 비해 프로토콜이 더 복잡하고 속도도 느리다.
(6) UDP에 비해 신뢰성 있는 데이터 전송이 가능하다는 장점 때문에 HTTP, FTP, TELNET 등 대 부분 응용계층 프로토콜은 전송 계층으로 TCP를 이용한다.

## 5.3.2 계층과 프로토콜

### 전송 계층[ Transport Layer ]

5. UDP( User Datagram Protocol )

(1) UDP는 비 연결지향 프로토콜이다.
(2) 전송한 데이터가 잘 전달되었는지 확인하지 않고 단지 데이터를 보내는 것으로 자신의 임무를 다한 것으로 생각한다.
(3) UDP는 TCP에 비해 신뢰성이 떨어지는 프로토콜이다.
(4) 흔히, 편지를 보내는 것에 비유
(5) 음악이나 동영상의 Streaming 서비스 같은 것에 적당한 프로토콜이다.

## 5.3.2 계층과 프로토콜

### 응용 계층 (Application Layer)

1. 하위 계층이 목적지가 되는 호스트에 데이터를 안전하게 전달하는 신뢰에서 응용계층에서는 응용 프로그램(프로세스) 들이 데이터 통신을 하게 된다.

2. 응용 계층의 응용 프로그램(프로세스) 들의 데이터 통신에는 매우 다양하다.

   ex) 메일을 보내기(SMTP), 파일을 전송하기(FTP), 웹사이트에 접속하기(HTTP)

   각 각의 목적에 맞는 데이터 통신을 위한 응용 프로토콜이 이미 정의되어 있기도 하고
   사용자 프로토콜을 설계하여 인터넷 기반의 서비스를 개발할 수 있다.

3. Socket(소켓)은 응용 계층에서 개발되는 응용 프로그램에서 하위 계층의 TCP/IP의 역할을 감추어 준다. (투명성)

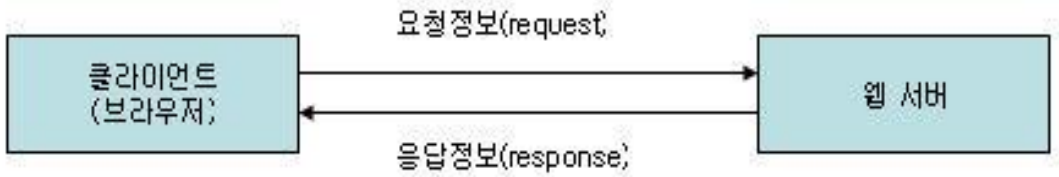
4. Socket(소켓)이라는 도구를 사용하면 응용 프로그램 간의 성격에 따라 기 설계된 응용 프로토콜을 구현한 프로그램을 개발하거나 프로토콜을 설계하고 구현하면 된다.

5. 대부분의 네트워크 프로그래밍은 socket을사용하여 위와 같은 작업을 하는 것이라 할 수 있다.

## 5.3.3 HTTP

1. 웹 서버를 직접 TCP/IP 소켓 프로그래밍을 이용해서 작성

2. HTTP 프로토콜




- 브라우저는 웹 서버에게 요청 정보를 보낸다.
- 웹 서버는 요청정보를 분석하여 응답정보를 브라우저에게 보낸다.

클라이언트

웹서버

1 connect

2 request

**HTTP 요청 메시지**

**GET /gpio/query?a=led&b=90 HTTP/1.1**
**Host: www.kickscar.com**
**User-agent: mozilla/4.0**
**Accept-language: kr**

요청헤더

요청바디

3 response

**HTTP 응답 메시지**

HTTP/1.1 **200** OK
Date : Thu,03 jul 2003 12:00:15 GMT
Server : Apache/1.3.0 (Unix)
Last-Modified : Sun, 5 May 2003 09:23:24 GMT
Content-Length : 6821
Content-Type : text/html

응답헤더

응답바디

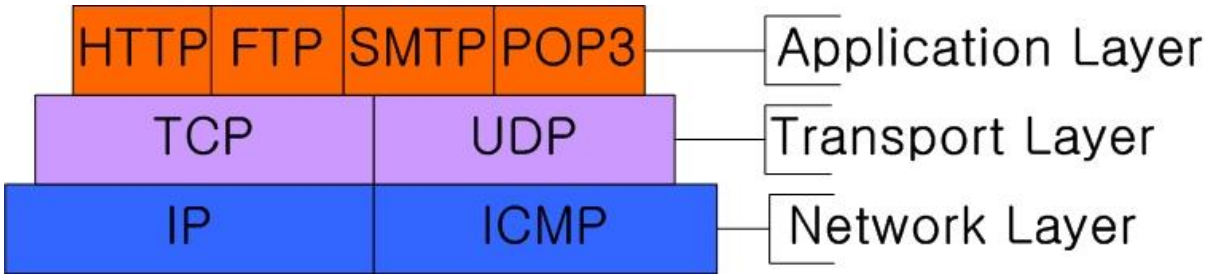
4 close

## 5.3.4 TCP/IP 프로토콜 스택(stack, 계층)

1. TCP/IP 프로토콜 스택은 총 4개 부분으로 나누어 진다.

2. 네트워크상의 데이터 통신 과정을  4개의 영역으로 계층화 한 것

3. 데이터 통신 과정을  하나의 덩치 큰 프로토콜로 해결한 것이 아니고  작게 나눠서 계층화하려는 노력의 결과이다.

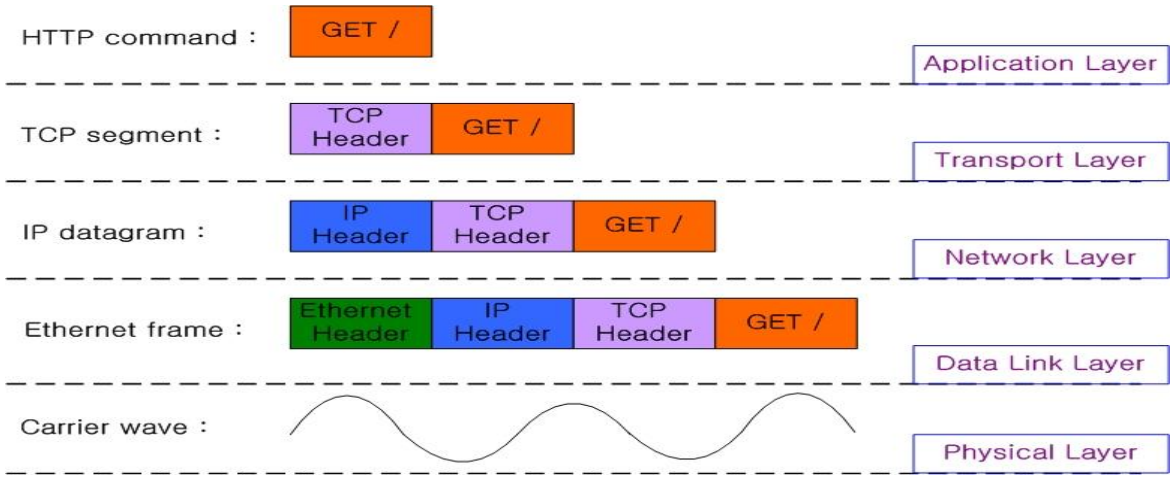


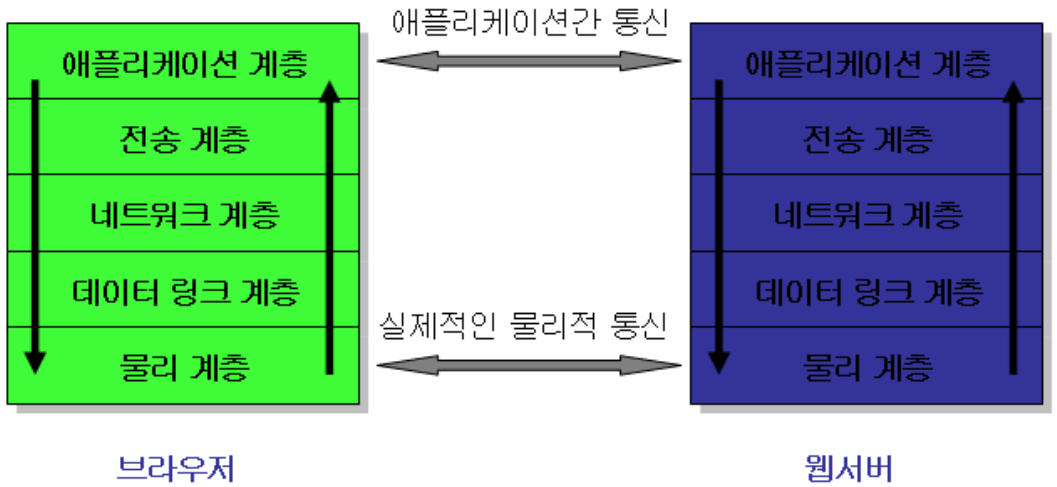

4. 왜 OSI 7계층을 모두 사용하지 않는가?

- OSI 7계층은 이론적인 면과 하드웨어 장비적인 표준을  위해 제정한 것이다.

- 실무에서 네트워크 프로그래밍을 할 경우 90% 이상의 위의 프로토콜 스택을 기반으로 작업 하게 될 것이다.

## 5.3.5 프로토콜 계층간의 데이터 캡슐화



# 5.4 소켓의 이해

1. TCP/IP 프로토콜의 프로그래머 인터페이스

2. 네트워크 프로그래밍에서 개발자에게 네트워크에 접근할 수 있는 인터페이스 제공

3. 1986년 BSD Unix4.3 개정된 소켓 사용

4. 프로세스 간의 통신 방식( 클라이언트 / 서버모델 )

5. 3가지 과정으로 사용

   **(1) 소켓 생성(소켓 열기)**

   **(2) 소켓을 통한 송/수신**

   **(3) 소켓 소멸( 소켓 닫기)**

# 5.4 소켓의 이해

## 5.4.1 소켓 생성

**int socket( int domain, int type, int protocol )**

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

int sockfd = socket( PF_INET, SOCK_STREAM, IPPROTO_TCP );
```

domain : 소켓이 사용할 프로토콜 체계( Protocol Family )
- PF_INET : IPv4 인터넷 프로토콜 체계
- PF_INET6 : IPv6 인터넷 프로토콜 체계
- PF_LOCAL : 로컬 통신을 위한 UNIX 프로토콜 체계
- PF_PACKET : Low Level 소켓을 위한 프로토콜 체계
- PF_IPX : IPX 노벨 프로토콜 체계

# 5.4 소켓의 이해

## 5.4.1 소켓 생성

**int socket( int domain, int type, int protocol )**

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

int sockfd = socket( PF_INET, SOCK_STREAM, IPPROTO_TCP );
```

type : 소켓의 유형

- SOCK_STREAM
  1) TCP 통신 소켓이다.
  2) stream 방식의 연결지향 소켓을 만든다.( 전화와 비슷한 연결 )
  3) 양방향 통신
  4) 바이트를 주고 받을 수 있는 가변 길이 stream
  5) 전달된 모든 데이터는 에러 없이 원격지에 도달
  6) 전달된 순서대로 수신됨

- SOCK_DATAGRAM
  1) UDP 통신 소켓이다.
  2) datagram 방식의 비 연결성 소켓을 만든다. ( 전보와 비슷한 비 연결 )
  3) 양방향 통신
  4) 고정길이의 메시지를 사용
  5) 신뢰성이 없다.
  6) 전달된 순서대로 수신되지 않음

# 5.4 소켓의 이해

## 5.4.1 소켓 생성

**int socket( int domain, int type, int protocol )**

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

int sockfd = socket( PF_INET, SOCK_STREAM, IPPROTO_TCP );
```

protocol : 프로토콜 선택

- 0을 세팅 하여도 충분히 원하는 소켓을 생성할 수 있음
- 하나의 프로토콜 체계에서 데이터 전송방식이 동일한 프로토콜이 2개 이상 존재하기 때문에 마지막 파라미터를 통해 원하는 프로토콜 정보를 조금 더 구체화한다.

1) IPv4 인터넷 프로토콜 체계에서 연결지향형 데이터 송수신 소켓

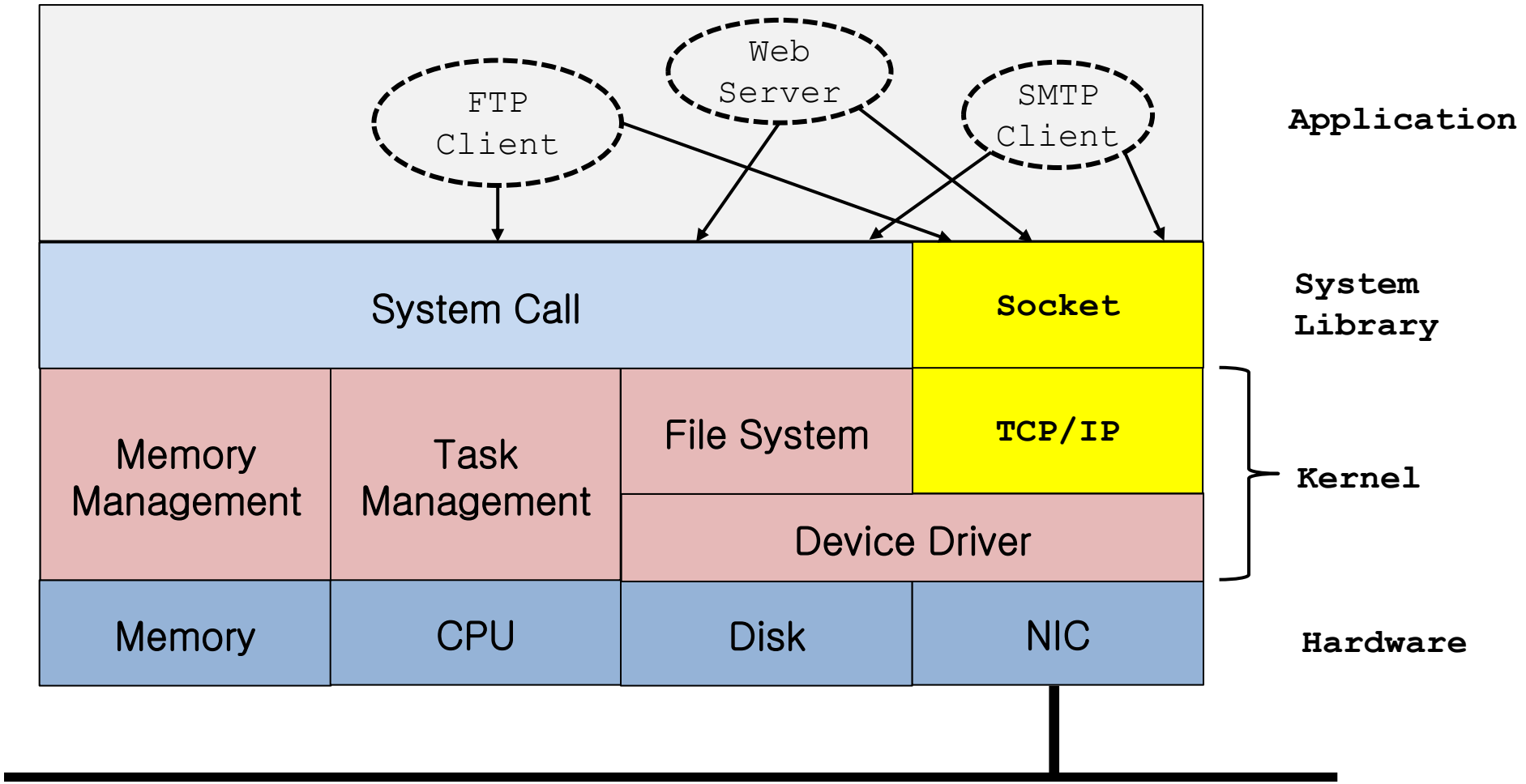
**int sockfd = socket( PF_INET, SOCK_STREAM, IPPROTO_TCP );**

2) IPv4 인터넷 프로토콜 체계에서 비 연결 지향형 데이터 송수신 소켓

**int sockfd = socket( PF_INET, SOCK_DGRAM, IPPROTO_UDP );**

## 5.4.2 포트번호

### OS 구조와 TCP/IP 그리고 socket의 위치



## 5.4.2 포트번호

### 포트의 필요성

- 실제적인 데이터 통신은 연결된 두 Host(컴퓨터)의 Process(프로그램) 사이에서 이루어 진다.

- 여러 계층을 통해 애플리케이션 계층으로 들어온 데이터를 해당 Process에만 정확하게 전달할 필요가 있다.

- 하나의 Host(컴퓨터)에는 여러 개의 Process(프로그램)가 각각의 소켓을 사용하여 데이터 통신을 하고 있기 때문에 TCP에서는 각 소켓을 구분할 필요가 생긴다.

- 이 때 각각의 소켓을 구분 할 때 사용하는 것이 포트 이다.

- 쉬운 예시

**“ 아파트(Host)에 사는 사람(Process)에게 편지(Data)를 보낼 때,**

**동(IP Address)과 호(Port)를 봉투(Packet)에 기입해야 한다.”**

## 5.4.2 포트번호

1. 포트번호는 16비트 정수를 사용한다. ( 0 - 65535 )

2. 포트의 종류
   (1)  1 - 255  : 잘 알려진 인터넷 서비스 포트( Well-Known Port)
   (2)  256 - 1023  : 그 밖의 인터넷 서비스
   (3) 1024 - 4999  :  시스템 예약
   (4) 5000 - 65535  :  사용자 사용

[참고]
http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

3. 포트의 중복은  불가능하다.

4. 하지만,  같은 UDP 포트와  TCP 포트는  중복하여 사용할 수 있다.

## 5.5.1 ping

ping 명령을 이용하면 다름 시스템의 네트워크가 현재 동작 중인지 알 수가 있다.

### 사용법

ping [옵션] 호스트

### 옵션

-s : 패킷 사이즈를 지정한다.
-q : 종합 결과만 보여준다.
-i : 지연시간을 설정한다.
-c : 보낼 패킷 수를 지정해 준다.

### 실습1

```
[root@localhost ~]# ping www.google.com
PING www.google.com (59.18.45.34) 56(84) bytes of data.
64 bytes from cache.google.com (59.18.45.34): icmp_seq=1 ttl=57 time=74.7 ms
64 bytes from cache.google.com (59.18.45.34): icmp_seq=2 ttl=57 time=151 ms
64 bytes from cache.google.com (59.18.45.34): icmp_seq=3 ttl=57 time=187 ms
64 bytes from cache.google.com (59.18.45.34): icmp_seq=4 ttl=57 time=200 ms
64 bytes from cache.google.com (59.18.45.34): icmp_seq=5 ttl=57 time=121 ms
. . .
. . .

--- www.google.com ping statistics ---
89 packets transmitted, 85 received, 4% packet loss, time 105734ms
rtt min/avg/max/mdev = 2.892/173.967/340.652/79.367 ms
```

기본적으로 아무 옵션 없이 사용하면 계속 패킷을 보낸다. Ctrl+C 로 중지.
한다.

### 실습2

패킷 횟수를 지정해서 보낼 수도 있다.

```
[root@localhost ~]# ping -c 5 www.google.com
PING www.google.com (59.18.45.54) 56(84) bytes of data.
64 bytes from cache.google.com (59.18.45.54): icmp_seq=1 ttl=57 time=228 ms
64 bytes from cache.google.com (59.18.45.54): icmp_seq=2 ttl=57 time=186 ms
64 bytes from cache.google.com (59.18.45.54): icmp_seq=3 ttl=57 time=222 ms

--- www.google.com ping statistics ---
5 packets transmitted, 3 received, 40% packet loss, time 4001ms
rtt min/avg/max/mdev = 186.741/212.701/228.387/18.497 ms
```

### 실습3

보낼 패킷의 크기를 지정할 수 있다.

```
[root@localhost ~]# ping -s 1000 -c 5 www.google.com
PING www.google.com (59.18.45.35) 1000(1028) bytes of data.
1008 bytes from cache.google.com (59.18.45.35): icmp_seq=1 ttl=57 time=165 ms
1008 bytes from cache.google.com (59.18.45.35): icmp_seq=2 ttl=57 time=157 ms
1008 bytes from cache.google.com (59.18.45.35): icmp_seq=4 ttl=57 time=230 ms
1008 bytes from 59.18.45.35: icmp_seq=5 ttl=57 time=187 ms

--- www.google.com ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 8396ms
rtt min/avg/max/mdev = 157.476/185.342/230.987/28.545 ms
```

# 5.5 리눅스 네트워크 관리

ping 명령어는 상대 호스트 또는 자신이 정상적으로 네트워크 작동을 하는 지 확인하는 데 아주 유용하게 쓰인다. 하지만 과도하게 사용하면 서버에 부담을 줄 수 있다.

외부의 과도한 ping을 막기 위해  ICMP echo를 ignore 시킨다.

**실습4**

ping 응답 설정 여부 확인

```
[root@localhost ~]# cat /proc/sys/net/ipv4/icmp_echo_ignore_all
0
[root@localhost ~]#
```

**실습5**

ping 응답을 막는 설정하기 위해 #vi /etc/sysctl.conf를 열고 다음을 추가 한다.

```
# For more information, see sysctl.conf(5) and sysctl.d(5).

net.ipv4.icmp_echo_ignore_all=1
```

```
[root@localhost ~]# sysctl -p
net.ipv4.icmp_echo_ignore_all = 1
[root@localhost ~]#
```

외부에서 ping을 시도해서 테스트 한다.

## 5.5.2 nslookup

nslookup은 도메인 네임서버에 질의를 할 수 있는 명령어 이다. 도메인 이름의 호스트의 IP주소를 검색할 수 있고 네임서버가 올바르게 작동하는 지 확인 할 수도 있다.

### 사용법

nslookup [도메인]

도메인을 입력하지 않으면 대화형으로 프로그램이 작동한다.

### 실습 1

한 개의 도메인만 질의할 때 비대화형 방식으로 질의가 가능하다.

```
[root@localhost ~]# nslookup www.naver.com
Server:                 168.126.63.1
Address:        168.126.63.1#53

Non-authoritative answer:
www.naver.com   canonical name = www.naver.com.nheos.com.
Name:           www.naver.com.nheos.com
Address: 125.209.222.142
Name:           www.naver.com.nheos.com
Address: 202.179.177.21

[root@localhost ~]#
```

# 5.5 리눅스 네트워크 관리

## 실습 2

대화형 방식으로 여러 도메인을 질의 할 수 있다.

```
[root@localhost ~]# nslookup
> www.naver.com
Server:                 168.126.63.1
Address:        168.126.63.1#53

Non-authoritative answer:
www.naver.comcanonical name = www.naver.com.nheos.com.
Name:           www.naver.com.nheos.com
Address: 125.209.222.142
Name:           www.naver.com.nheos.com
Address: 202.179.177.21
> www.bitacademy.co.kr
Server:                 168.126.63.1
Address:        168.126.63.1#53

Non-authoritative answer:
Name:           www.bitacademy.co.kr
Address: 218.145.65.233
> www.facebook.com
Server:                 168.126.63.1
Address:        168.126.63.1#53
> exit

[root@localhost ~]#
```

## 5.5.3 hostname

호스트네임을 화면에 출력하고, 호스트네임을 변경할 수 있다.

### 사용법

hostname

### 실습 1

```
[root@localhost etc]# hostname
localhost.localdomain
[root@localhost etc]#
```

### 실습2

호스트 명을 변경해 보자. 비교적 간단하게 바꿀 수 있다.

/etc/hostname 파일을 vi 에디터로 열고 다음과 같이 수정한다.

```
lx.kickscar.com
```

저장하고 반영하기 위해 재부팅 한다.

```
[kickscar@lx ~]$ clear
[kickscar@lx ~]$ hostname
lx.kickscar.com
[kickscar@lx ~]$
```

## 5.5.4 netstat

네트워크 연결, 라우팅 테이블, 네트워크 장치의 통계정보 등 네트워크에 관련된 여러가지 정보를 확인할 수 있다.

**사용법**

netstat [옵션]

**옵션**

-a : 연결된 모든 소켓을 출력한다.
-n : 호스트, 포트등의 정보를 이름대신 숫자로 표시한다.
-p : 소켓을 열고 있는 프로세스의 아이디(PID) 를 출력한다.
-r : 라우팅 테이블을 출력한다.
-t : TCP 연결에 대한 소켓을 출력한다.
-u : UDP 연결에 대한 소켓을 출력한다.

**실습1**

현재 시스템의 라우팅 테이블을 확인해 보자.

```
[root@lx ~]# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         192.168.0.1     0.0.0.0         UG        0 0          0 enp0s3
192.168.0.0     0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
[root@lx ~]#
```

# 5.5 리눅스 네트워크 관리

## 실습2

현재 열려 있는 TCP 포트 정보를 출력해보자

```
[root@lx ~]# netstat -ant | grep LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:25                  :::*                    LISTEN
[root@lx ~]#
```

## 실습3

현재 열려 있는 TCP 포트의 프로세스까지 출력해 보자.

```
[root@lx ~]# netstat -anpt | grep LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      701/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      780/master
tcp6       0      0 :::22                   :::*                    LISTEN      701/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      780/master
[root@lx ~]#
```

PID 780번인 프로세스는 어떤 프로세스인가?

## 5.5.5 ifconfig

네트워크 인터페이스를 설정하고, 현재 네트워크 인터페이스의 정보를 알아보는 명령어이다.
대부분 네트워크 서정을 확인하는 명령어로 많이 사용된다.

### 사용법

ifconfig [옵션]

### 실습1

전체 네트워크 인터페이스의 설정을 확인해 보자

```
[root@lx ~]# ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.3  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe88:f242  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:88:f2:42  txqueuelen 1000  (Ethernet)
        RX packets 1841  bytes 162376 (158.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1297  bytes 250857 (244.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 2  bytes 106 (106.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 106 (106.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@lx ~]#
```

# 5.5 리눅스 네트워크 관리

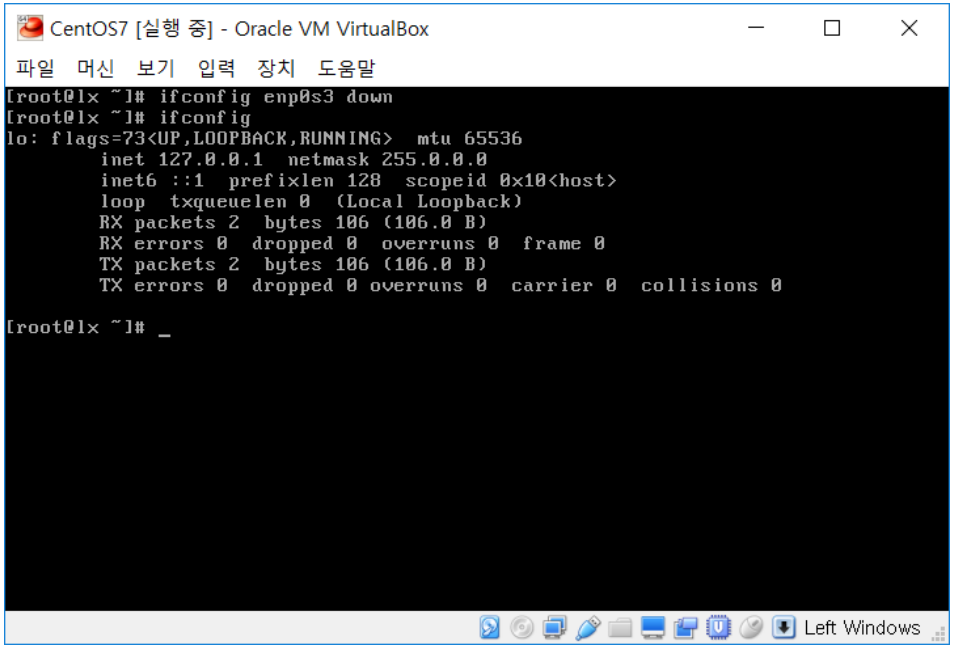
## 실습2

인터페이스(interface)는 NIC(Network Interface Card)를 말하며 보통 랜카드 이더넷 카드라고 부른다.
특정 네트워크 인터페이스에 대한 정보만 출력할 수 있다.

```
[root@lx ~]# ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.3  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe88:f242  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:88:f2:42  txqueuelen 1000  (Ethernet)
        RX packets 1952  bytes 171942 (167.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1365  bytes 259601 (253.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

## 실습3

네트워크를 중지하자.



# 5.5 리눅스 네트워크 관리

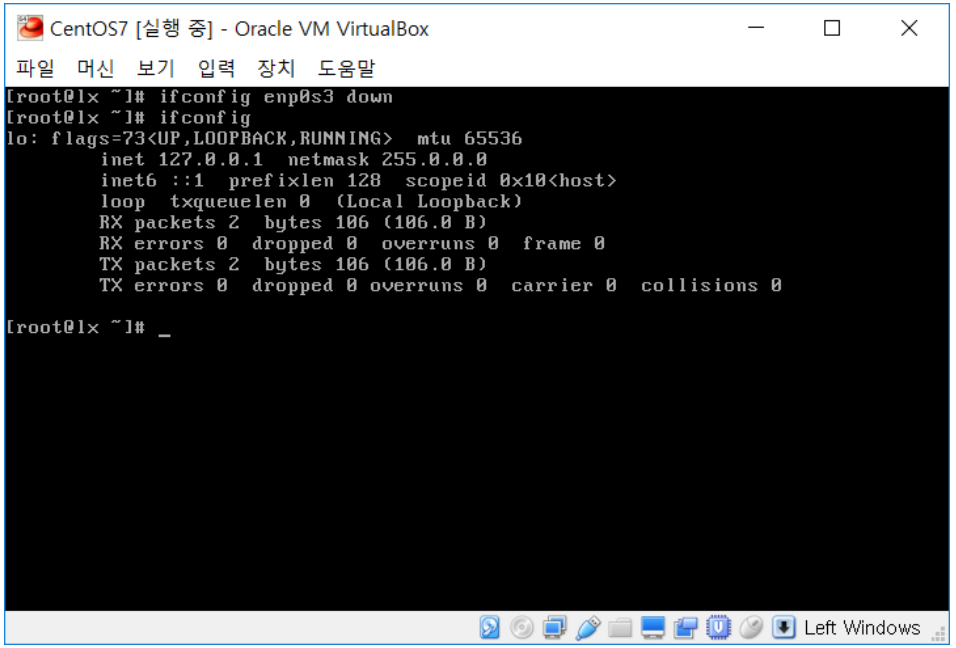
## 실습2

인터페이스(interface)는 NIC(Network Interface Card)를 말하며 보통 랜카드 이더넷 카드라고 부른다.
특정 네트워크 인터페이스에 대한 정보만 출력할 수 있다.

```
[root@lx ~]# ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.3  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe88:f242  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:88:f2:42  txqueuelen 1000  (Ethernet)
        RX packets 1952  bytes 171942 (167.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1365  bytes 259601 (253.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

## 실습3

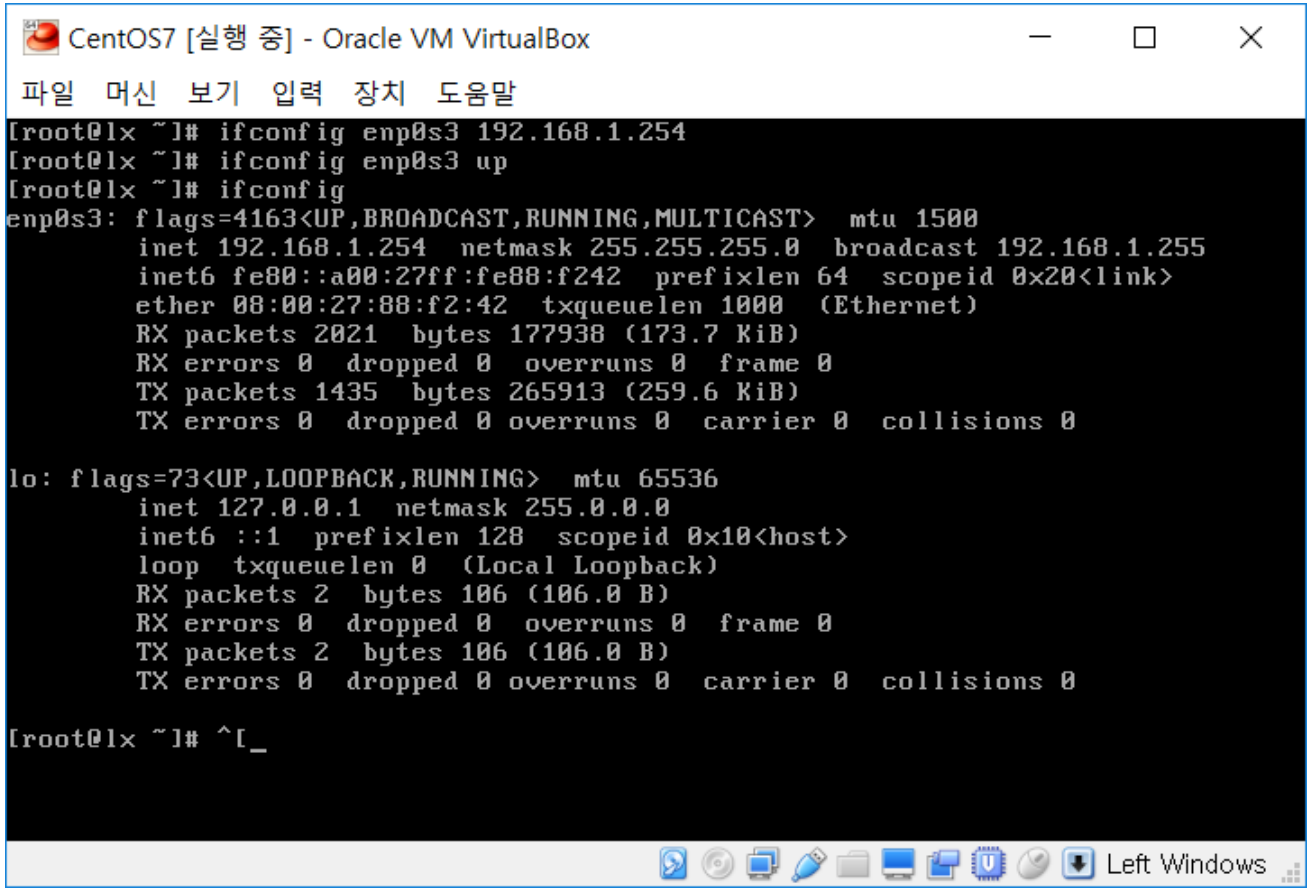
네트워크를 중지하자.



# 5.5 리눅스 네트워크 관리

## 실습2

네트워크 인터페이스 enp0s3에 IP address를 192.168.1.254 바꾸고 다시 시작해 보자.



## 5.5.6 네트워크 설정하기 (고정 아이피 설정)

ifconfig 에서 설정된 IP 주소는 시스템이 재 시작하게 되면 반영이 되지 못한다. 따라서 시스템의 네트워크를 설정에서 영구적으로 반영 되도록 해야 한다.

**설정하기 전에 알아두어 야 할 것**

1. IP Address
2. Subnet Mask
3. Gateway IP Address
4. DNS Server IP Address

**실습1**

네트워크 설정은 /etc/sysconfig/network-scrips/ifcfg-인터페이스 파일에서 한다.
현재 설정 내용을 확인해보자.

```
[kickscar@lx ~]$ cat /etc/sysconfig/network-scripts/ifcfg-enp0s3
TYPE="Ethernet"
BOOTPROTO="dhcp"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
NAME="enp0s3"
UUID="c6755cb5-a27b-4260-a305-aa260b00c275"
DEVICE="enp0s3"
ONBOOT="yes"
PEERDNS="yes"
PEERROUTES="yes"
IPV6_PEERDNS="yes"
IPV6_PEERROUTES="yes"
IPV6_PRIVACY="no"
[kickscar@lx ~]$
```

# 5.5 리눅스 네트워크 관리

설정 내용을 확인해 보면 BOOTPPROTO 가 dhcp 즉, dynamic 하게 dhcp 서버로 부터 IP 주소와 게이웨이 IP 주소 Subnet Mask, DNS 서버 IP 정보를 받아와 세팅하게 된다.

고정 아이피로 바꾸기 위한 설정은 다음과 같다.

```
TYPE="Ethernet"
BOOTPROTO= "static"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
NAME="enp0s3"
UUID="c6755cb5-a27b-4260-a305-aa260b00c275"
DEVICE="enp0s3"
ONBOOT="yes"
PEERDNS="yes"
PEERROUTES="yes"
IPV6_PEERDNS="yes"
IPV6_PEERROUTES="yes"
IPV6_PRIVACY="no"


IPADDR=192.168.1.101
NETMASK=255.555.255.0
GATEWAY=192.168.1.1
DNS1=168.126.63.1
```

수정 후,

# systemctl restart network.service 로 네트워크를 다시 시작한다.