

# Linux

- CentOS 8 -

2022.01  
RACOS System

# Contents

1. vi 명령어

01. User

키	커서 이동	키	문자,행 삽입	키	텍스트 삭제	키	행 번호 설정	키	탐색 및 대체
h (←)	왼쪽으로 이동	a	커서 오른쪽에 문자 삽입	x	커서가 있는 문자 삭제	:se number	행 번호 표시	/ {검색할 문자열}	오른쪽 아래 방향으로 검색
j (↓)	아래로 이동	A	커서 오른쪽 행의 끝에 문자 삽입	숫자 x	현재 위치부터 숫자만큼 문자 삭제	:se nonu	행 번호 숨기기	? {검색할 문자열}	왼쪽 위 방향으로 검색
k (↑)	위로 이동	i	커서 왼쪽에 문자 삽입	dw	현재 커서에 있는 한 단어 삭제	키	행 찾기	n	문자열 다음으로 계속 검색
l (→)	오른쪽으로 이동	l	커서 왼쪽 행의 처음에 문자 삽입	dd	커서가 있는 라인 삭제	G	파일의 마지막 행으로 가기	N	문자열 이전으로 계속 검색
w	오른쪽 한 단어의 끝으로 이동	o	커서 아래에 행 삽입	숫자 dd	현재 라인부터 숫자만큼 라인 삭제	21G	21번째 행으로 가지	:g/검색할 문자열/s/	검색 후 각 문자열을 확인하고 대체
e	오른쪽 한 단어의 앞으로 이동	O	커서 위에 행 삽입	db	현재 위치에서 뒤로 한 단어 삭제	Ctrl+G	현재 파일명과 라인 정보	:s/old/new	현재 행의 old를 new로 대체
b	왼쪽 한 단어의 앞으로 이동	Esc	종료	D	커서 오른쪽 행 삭제	키	화면 정리(깨지는 경우)	:1,s/old/new/	1부터 현재 행의 old를 new로 대체
Enter	아래 행 맨 앞으로 이동	키	텍스트 변경	:5,10d	5~10번째 행 삭제	Ctrl+I	화면정리 후 다시 표시	:%s/old/new/g	파일 전체 old를 new로 전부 대체
BS	한 문자 왼쪽으로 이동	cw	단어 변경	키	복사, 이동	키	다른 파일 내용 삽입	:\$/old/new	현 위치로 부터 파일의 끝까지 대체
Space	한 문자 오른쪽으로 이동	cc	행 변경	yy	현재 행 복사	:r {파일명}	커서 다음에 파일 삽입	키	기타
^	행의 맨 왼쪽으로 이동	cw	커서 오른쪽 행 변경	Y	행 Yank 또는 복사	: {행번호} r {파일명}	해당 파일을 행번호 다음에 삽입	.	현재 행
\$	행의 맨 오른쪽으로 이동	s	커서가 위치한 문자열 대체	yh	커서의 왼쪽 문자 복사	키	저장 및 종료	%	전체 행(파일 전체)
H	화면의 맨 위로 이동	S	커서가 위치한 라인의 문자열 대체	yl	커서에 위치한 문자 복사	:w	변경사항 저장	\$	파일 맨끝 행
M	화면의 중간으로 이동	r	커서 위치 문자를 다른 문자로 대체	yi	현재 행과 그 아래줄 복사	:w {파일명}	변경사항을 파일에 저장	1,\$	%와 동일
L	화면의 맨 아래로 이동	r-Enter	행 분리	yk	현재 행과 그 윗줄 복사	:wq	변경사항 저장 후 vi 종료	2,3	2~3 행
숫자 G	"숫자" 만큼 지정한 줄로 이동	J	현재 행과 아래 행 결합	p	yank된 행을 현재 행 아래에 삽입	ZZ	:wq와 동일		
Ctrl+i	커서 이동 없이 한 화면 위로 이동	xp	커서 위치 문자와 오른쪽 문자 교환	P(대문자)	yank된 행을 현재 행 위에 삽입	:q!	변경사항 저장없이 vi 종료		
Ctrl+b	커서 이동 없이 한 화면 아래로 이동	~	문자형(대, 소문자) 변경	:1,2 co 3	1~2행을 3행 다음으로 복사	:q	변경사항 없으면 종료		
Ctrl+d	커서 이동 없이 반 화면 위로 이동	u	이전 명령 취소	:4,5 m 6	4~5행을 6행 위로 이동	:e!	수정한 것을 무시하고 다시 편집모드		
Ctrl+u	커서 이동 없이 반 화면 아래로 이동	U	행 변경 사항 취소, 이전 최종 행 취소	v + 선택 (블록 복사)	1. v를 누른 후 커서 이동하여 블록 설정				
Ctrl+e	커서 이동 없이 한 줄 위로 이동	.	이전 최종 명령 반복		2. y를 눌러 캐시에 복사				
Ctrl+y	커서 이동 없이 한 줄 아래로 이동				3. 원하는 곳으로 이동 후 p를 눌러 복사				

# 기본 명령어 I

man, man hier	pwd	cat	shutdown	rev
history	ls	touch	init 0	yes
whoami	cd	head	init 6	sort
logname	mkdir	tail	ps	cmp
users	rmdir	nl	kill	diff
who	cp	more	cal	diff3
w	mv	less	date	comm
uname	rm	grep	timedatectl	which
id	echo		alias	find
su	clear	> >		locate (updatedb)
sudo		ln		whereis

# 기본 명령어 II

groupadd	useradd	usermod		
groupdel	passwd	usermod -l	grep user /etc/passwd	
	su user01	usermod -d		
	userdel	Usermod -m -d		
		usermod -c		
chmod		usermod -e	chage -l	
chown		usermod -g	id	
chgrp		usermod -G		
tar		usermod -s		

# 리눅스마스터 1급 기출 문제

## [1902회 리눅스마스터 1급 1차 필기]

다음 중 joon이라는 아이디를 lin으로 변경하는 명령으로 알맞은 것은?

1. usermod -l joon lin
2. usermod -l lin joob
3. usermod -n joon lin
4. usermod -n lin joon

답:

## [1901회 리눅스마스터 1급 2차 실기]

다음 조건에 맞게 사용자 정보를 변경하려고 할 때 (괄호) 안에 알맞은 내용을 적으시오.

```
# ( ① ) ( ② ) ( ③ ) ( ④ ) idhuser
```

### ■조건

- idhuser의 사용자명을 kaituser로 변경한다.
- 홈 디렉토리를 /home/idhuser 에서 /home/kaituser로 변경하고, 기존에 소유했던 파일이나 디렉터리도 그대로 이용 가능하도록 한다.
- ①번은 해당 명령어를 기재한다.
- ②~④번은 명령어의 옵션 또는 옵션과 관련된 인자 값을 기재한다.

답:

# 기본 명령어 III

mount	dnf --version	dnf install 패키지	dnf distro-sync	cron
umount	dnf repolist	dnf -y install 패키지	dnf reinstall 패키지	at
file	dnf list all	dnf check-update	dnf --enablerepo=epel install 패키지	
tar (cvf tvf xvf)	dnf list available	dnf update	dnf grouplist	
xz (-d)	dnf list installed	dnf update 패키지	dnf groupinstall 그룹패키지	
bzip2 (-d)	dnf list 패키지이름	dnf remove 패키지	dnf groupupdate 그룹패키지	
gzip (-d)	dnf search 패키지이름	dnf autoremove	dnf groupremove 그룹패키지	
zip 생성할.zip 압축할파일	dnf info 패키지이름	dnf clean all		
unzip 압축파일.zip		dnf help clean		
		dnf history		

# cron & at

```
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
03 4 * * 0 root run-parts /etc/cron.weekly
04 4 1 * * root run-parts /etc/cron.monthly
```

# at 3:00am tomorrow

# at 11:00pm February 21

# at now + 1 hours

Enter 후, ^d

# at -l

# atrm 작업번호



# find 명령어 실습

1. /etc 디렉토리 밑을 확장명이 \*.conf 파일 검색
2. /home 디렉토리 밑으로 소유자가 usr01인 파일 검색
3. 현재 사용자의 홈 디렉토리 하위에 허가원이 644인 파일 검색
4. /usr/bin 디렉토리 밑으로 파일 크기가 10kb ~ 100kb 파일 검색
5. 현재 사용자의 홈 디렉토리 밑으로 파일 크기가 0인 파일의 목록을 상세히 출력
6. /home 디렉토리 밑으로 확장명이 \*.swp인 파일 삭제

# 기본 명령어 IV

echo \$SHELL	XXX=123.txt	date 'test'	tree
ps \$\$	echo \$XXX	date 'test' && printf "RacosWn"	tree -d
echo \$\$	printf \$XXX	date 'test'    printf "RacosWn"	tree -f
env 또는 printenv	printf '%sWn' \$XXX	linux;date	tree -d -L 1 /
env 또는 printenv	unset XXX	history 한 후에 !번호 또는 !문자	cd . cd .. cd - cd ~
env   grep HISTSIZE		date 한 후에 !! 또는 !-1, !-2	cd /dev cd ./dev
ZZZ=123.txt	>	date; !# 또는 date; sleep 2; !#	ls -al   grep ^d
export YYY=123.txt	>>		cat z123.txt
DATE=`date`		find . -type d -print   wc -l	cat z123.txt > /dev/null
DATE=\$(date)	<		cat z123.txt > /dev/null 2>&1
date '+%H:%M'	sort < file		cat z123.txt > 222
dat Tab-Key	sort < file > file.txt		cat z123.txt > 222 2>&1

# Shell : 명령어 활용

--- 명령어 자동 완성

\$ dat            (이 상태에서 Tab 치면?)

--- 다중 명령

\$ linux; date

\$ linux && date

\$ linux || date

\$ date 'racos' || printf "123\n" && printf "456\n"

# Shell : 변수 지정과 출력

```
$ FILENAME = test.txt
```

```
$ FILENAME= test.txt
```

```
$ FILENAME =test.txt
```

```
$ FILENAME=test.txt
```

```
$ printf "%s\n" $FILENAME
```

```
$ echo $FILENAME
```

# Linux 기본 Directories

No	Directory	Description	Remark
01	/	최상위 디렉토리.	Root Directory
02	/bin	실행파일 모음. mv, cat 등의 명령어 프로그램들이 모여 있음	binary의 약자
03	/boot	리눅스커널과 부팅과 관련된 파일들(부트로더)이 모여 있음	vmlinuz-*
04	/dev	물리적인 장치들이 파일화 되어 있음.	device의 약자
05	/etc	각종 환경 설정 파일들이 모여 있음.	
06	/home	개별 사용자들 디렉토리.	
07	/lib	각종 라이브러리 저장 디렉토리.	
08	/mnt	CD-ROM, 네트워크 파일 시스템 등을 마운트 할 때 사용되는 디렉토리.	
09	/proc	현재 실행되고 있는 프로세스들이 파일화 되어서 저장되는 디렉토리.	
10	/root	root계정의 홈 디렉토리.	
11	/sbin	주로 시스템 관리자가 쓰는 시스템 관련 명령어 프로그램들이 모여 있음.	System-binary의 약자
12	/tmp	임시 저장 디렉토리. 일반적으로 모든 사용자들에게 열려 있음.	
13	/usr	주로 새로 설치되는 프로그램들이 저장.	윈도우의 Program Files같은 폴더
14	/var	시스템 로그, Spooling 파일 등이 저장. 메일 서버로 운영될 경우 메일이 여기 저장.	

# 다양한 bin Directories

No	Directory	Description
01	/bin	관리자를 포함한 모든 사용자를 위한 공통 명령어가 이 폴더에 저장되어 있다. 이 디렉토리에 있는 명령어는 다른 파일 시스템이 마운트 되지 않아도 사용 가능하다. 예를 들어, /usr 파티션이 마운트 되기 이전에 사용할 수 있는 바이너리 파일들이 저장되어 있다. 이에 속하는 바이너리 파일로는 흔히 쓰는 cat 명령어나 ls 명령어 등
02	/sbin	/bin 폴더와 같이 멀티 사용자 & 단일 사용자 모드에서 실행할 수 있고, System Programs, Administration Utilities와 같이 루트 권한이 있어야만 실행할 수 있는 명령어가 저장.
03	/usr/bin	/usr 폴더 (파일 시스템의 두번째 메이저 섹션) 가 마운트 돼야만 사용할 수 있고, 모든 사용자가 공통으로 사용할 수 있는 명령어가 저장되어 있다. 이에 속하는 바이너리 파일로는 sudo 명령어, vi 명령어 등
04	/usr/sbin	/usr/bin 폴더와 같지만 루트 권한이 있어야만 실행할 수 있는 명령어가 저장되어 있다.
05	/usr/local/bin	직접 작성한 스크립트 파일 또는 프로그램이나, package manager에 의해 관리되지 않는 소프트웨어 이곳에 위치한 바이너리 파일들과 스크립트들은 일반 사용자를 위한 프로그램이다. 다만, 이 프로그램들은 distribution package manager에 의해 관리되지 않는다. 예를 들어, 사용자 본인이 직접 local directory에서 compile한 프로그램(logically compiled program)이 라면, 절대로 /usr/bin에 위치시켜선 안된다. 추후 설치하는 스크립트 등에 의해 아무런 경고 없이 업그 레이드되거나, 삭제될 수 있기 때문이다. 따라서 이런 프로그램들은 /usr/local/bin 디렉토리에 위치해야 한다.
06	/usr/local/sbin	마찬가지로 루트 권한이 있어야만 실행할 수 있는 명령어가 저장되어 있다.

# i-node

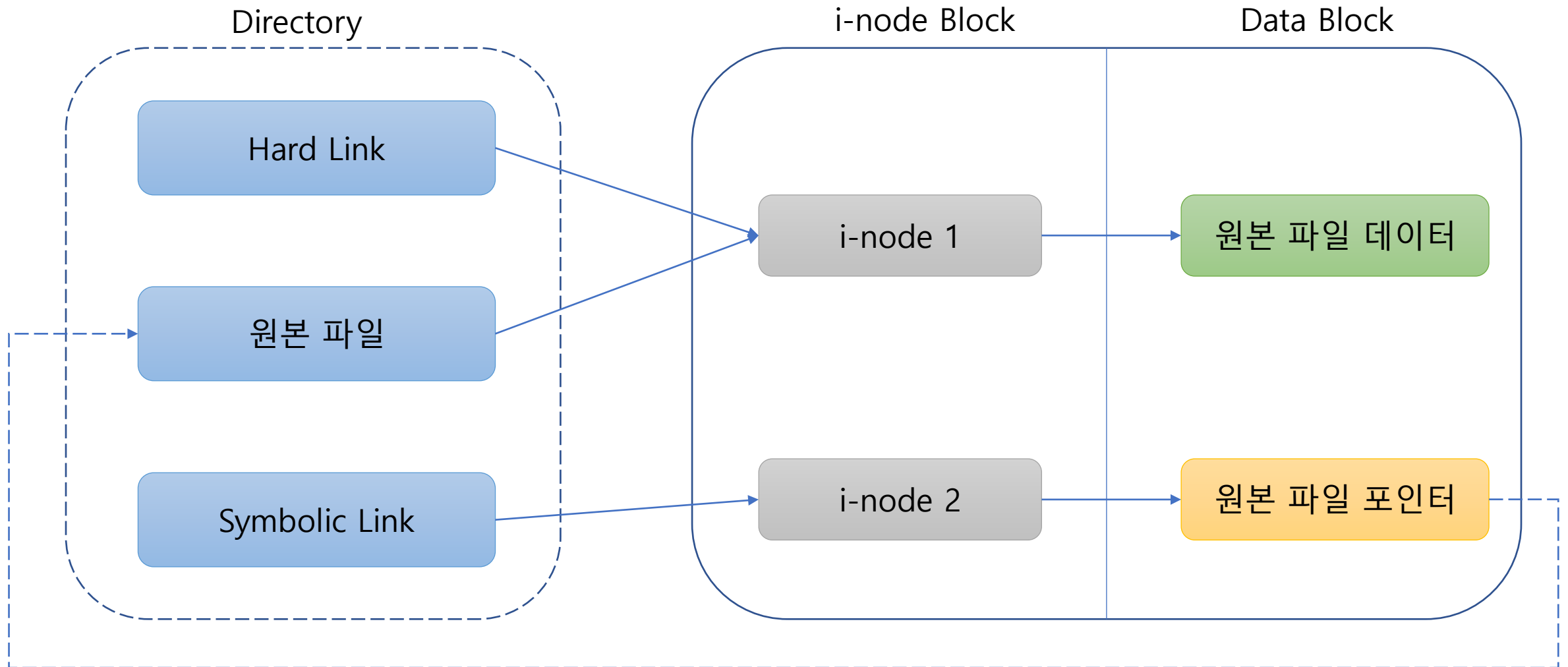
- Index-node 약자
- 리눅스 시스템에서 파일을 빠르게 찾기 위한 수단.
- 파일에 대한 Meta 정보 저장
- 파일이나 디렉토리는 고유의 i-node를 가지고 있음.

# Hard Link & Soft Link

구분	심볼릭 링크	하드 링크
생성 명령어	<code>ln -s [원본 파일명] [링크 파일명]</code>	<code>ln [원본 파일명] [링크 파일명]</code>
생성 종류	파일과 디렉토리 모두 생성	파일만 생성
링크 기능	파일 또는 디렉토리 이름에 대한 링크를 가리킴	원본 파일에 대한 참조 또는 포인터
원본 파일 삭제할 경우	액세스 불가능	액세스 가능
inode 번호	다른 inode 번호	동일한 inode 번호
다른 파티션 링크 여부	다른 파티션에 링크 가능	다른 파티션에 링크 불가능



# Hard Link **vs** Symbolic Link



# Wildcard

*		t*		
?		t*.txt		
[characters]	[:alnum:]	t???		
[!characters]	[:alpha:]	t???.txt		
	[digit:]	[ts]*		
	[:upper:]	[[:lower:]]*		
	[:lower:]	[![:upper:]]*		
		ls 명령어로 디렉토리만		
		ls 명령어로 파일만		

현재 디렉토리에 있는 파일 개수는?

# 기본 명령어 V : filter

sort	표준 입력에 대해 정렬을 수행하여 그 결과를 표준 출력으로 출력.	
uniq	표준 입력으로 부터 정렬된 데이터를 방사서 중복된 항목을 제거하고 출력.	
grep	표준 입력으로 부터 받은 라인 단위의 데이터로 부터 지정한 문자 패턴을 가지고 있는 라인을 찾아서 출력.	
fmt	표준 입력으로 부터 텍스트를 읽고 형식화 된 텍스트를 표준 출력으로 출력.	
pr	표준 입력으로 부터 텍스트를 입력 받아서 페이지 단위로 데이터를 잘라서 출력.	tree /   pr   more
head	입력된 파일에서 앞의 10개의 라인만 출력.	
tail	입력된 파일에서 마지막 10개 라인만 출력.	
tr	입력된 문자를 변경(대/소문자) 하거나, 반복, 삭제하여 출력	ls -al   tr 'd' 'D'
sed	스트림에디터로 tr 명령보다 다양한 문자 변경에 사용	
awk	강력한 필터로서 프로그래밍 언어라고 할 수 있음.	

# 기본 명령어 IV

umask				
umask 022				
lsattr				
chattr				

# umask

- 8진수의 보수(Complement)로서 파일과 디렉토리에 작용하는 마스크
- 리눅스 시스템에서는 기본적으로 파일이 실행 권한(x)을 가지고 못하게 함.
- 디렉토리의 경우에는 실행 권한(x) 그 디렉토리의 접근 권한을 의미하므로 실행 권한을 생성과 동시에 가질 수 있음.
- umask에 의한 파일 생성 퍼미션은 기본값 666에서 umask 값을 빼 주면 됨.
- 디렉토리 생성 퍼미션은 기본값 777에서 umask를 빼 준다.
- root User : umask 022 :  $666-022=644$ (파일),  $777-022=755$ (디렉토리)
- 일반 User : umask 002 :  $666-002=664$ (파일),  $777-002=775$ (디렉토리)

# Linux attribute

- **a속성(append only)**
  - 해당파일을 추가(append mode)만 할 수 있다. 파일보안을 위해 주로 사용하는 속성이다.
  - 디렉토리에 설정하면 해당 디렉토리에 파일을 생성만 할 수 있다.
- **c속성(compressed)**
  - 이 속성이 설정된 파일은 커널에 의해 디스크상에 자동적으로 압축된 상태로 저장되어 있다. 파일을 읽을 경우에는 압축을 해제한 상태로 되돌려주며 쓰기시에는 디스크에 저장하기전에 파일을 압축한다.
- **d속성(no dump)**
  - 이 속성이 설정된 파일은 dump명령어로 백업되지 않는다.
- **i속성(immutable)**
  - 이 속성이 지정되어 있다면 해당파일의 변경, 삭제, 이름 변경 뿐 아니라 파일추가 및 링크파일도 만들 수 없게 된다.
  - 변경추가가 거의 없는 부팅관련 파일들에 설정하면 부팅되지 않는 문제로 인한 시스템장애를 줄일 수 있다.
- **s속성(secure deletion)**
  - 이 속성이 설정된 파일은 파일삭제가 될 경우에 해당 블록이 모두 0으로 되어버리고 디스크에 다시 쓰기가 발생한다. 따라서 삭제파일복구가 불가능하게 된다.
- **S속성(synchronous updates)**
  - 이 속성이 설정된 파일은 변경이 될 경우에 디스크동기화가 일어나는 효과를 그대로 누릴 수 있다.

# Linux attribute

- **u속성(undeletable)**
  - 이 속성을 가진 파일이 삭제가 되었을 경우에는 그 내용이 저장되며 삭제되기 전의 데이터로 복구가 가능해진다.
  - 즉 사용자가 삭제되기 전의 내용을 요구하였을 때에 그 내용을 되살릴 수 있다.
- **j속성(data journalling)**
  - 저널링 파일시스템의 파일을 대상으로 설정하는 속성이다. 이 "j"속성은 오직 ext3타입으로 마운트 된 파일시스템에 매우 유용한 속성이다.
- **A속성(no atime updates)**
  - 이 속성이 설정되어 있는 파일에 대하여 access가 발생하더라도 파일시스템의 파일정보에 저장되는 atime이 갱신되지 않는다.
- **D속성(synchronous directory updates)**
  - 디렉토리에 설정하는 속성으로서 D속성이 설정된 디렉토리는 변경사항을 동기화 시킨다. 그리고 이 D속성은 커널 2.5.19이후 버전에서만 사용될 수 있다.
- **T속성(top of directory updates)**
  - 이 속성이 부여된 디렉토리는 최상위 디렉토리로 인식되어 간주된다. 커널 버전 2.5.46이후 버전에서만 사용될 수 있다.



# Linux attribute

## >> 속성 지정,제거(chattr명령어) <<

chattr명령어를 이용하여 특정파일 및 디렉토리에 위에서 나열된 속성을 지정할수 있다.  
연산자를 이용하여 속성을 변경한다.

연산자	의미
+	지정한 속성 추가
-	지정한 속성 제거
=	기존 속성을 초기화 후 지정한 속성만 부여

## >> chattr의 주요 옵션

- R : 디렉토리내의 서브디렉토리까지 속성을 일괄변경한다.
- f : 대부분의 에러메시지를 출력하지 않는다.
- V : 자세한 출력모드를 제공한다.
- v [version] : 지정된 파일에 버전을 설정할수 있다.

# 환경변수

환경변수	설명	환경변수	설명
HOME	현재 사용자의 홈 디렉토리	PATH	실행 파일을 찾는 디렉토리 경로
LANG	기본 지원되는 언어	PWD	사용자의 현재 작업 디렉토리
TERM	로그인 터미널 타입	SHELL	로그인 해서 사용하는 Shell
USER	현재 사용자의 이름	DISPLAY	X Display 이름
COLUMNS	현재 터미널의 컬럼 수	LINES	현재 터미널 라인 수
PS1	1차 명령 프롬프트 변수	PS2	2차 명령 프롬프트(대개는 '>')
BASH	Bash Shell의 경로	BASH_VERSION	Bash 버전
HISTFILE	History File의 경로	HISTSIZE	History File에 저장되는 개수
HOSTNAME	Host 이름	USERNAME	현재 사용자 이름
LOGNAME	로그인 이름	LS_COLORS	ls 명령어의 확장자 색상 옵션
MAIL	메일을 보관하는 경로	OSTYPE	운영체제 타입

# Shell 작성 및 실행 : name.sh

1. `#!/bin/sh`
2. `echo " User Name : " $USER`
3. `echo " Home Directory : " $HOME`
4. `exit 0`

1행 : 특별한 형태의 주석(#!)으로 bash을 사용하겠다는 의미. 첫 행에 꼭 써주어야 한다.

4행 : 종료 코드를 반환하게 해 준다. 0은 성공을 의미한다.

\*\*\* 실행방법

`$ sh name.sh`

`$ ./name.sh` (먼저 664를 764로 변경한 후)

# Shell : 변수 기본

- Shell 스크립트에서는 변수를 사용하기 전에 미리 선언하지 않는다.
- 처음 변수에 값이 할당되면 자동으로 변수가 생성된다.
- 변수에 넣는 모든 값은 문자열로 취급한다.  
즉, 숫자를 넣어도 문자열로 취급한다.
- 변수 이름은 대소문자를 구분한다.  
\$aa라는 변수이름과 \$AA는 다른 변수다.
- 변수를 대입할 때 "=" 좌우에는 공백이 없어야 한다.

- ① Testval = hello
- ② Testval=hello
- ③ Testval=Yes Sir
- ④ Testval="Yes Sir"
- ⑤ Testval=7+5

var1.sh

```
-----  
1. #!/bin/sh  
2. myvar="Hi Racos"  
3. echo $myvar  
4. echo "$myvar"  
5. echo '$myvar'  
6. echo W$myvar  
7. echo input value :  
8. read myvar  
9. echo '$myvar' = $myvar  
10. exit 0
```

# Shell : 변수(숫자 계산)

- 변수에 넣은 값은 모두 문자열로 취급된다.
- 변수에 들어 있는 값에 사칙 연산을 하려면 `expr` 키워드를 사용한다.
- Expr사용하는 경우, 연산자 좌우는 스페이스로 띄워야 한다.
- 단, 수식과 함께 꼭 키보드 1 왼쪽에 있는 역따옴표로 묶어 주어야 한다.
- 수식에 괄호를 사용하려면 그 앞에 꼭 역슬래시(`\`)를 붙여줘야 한다
- `+`, `-`, `/` 와 달이 곱하기(`*`) 기호도 예외적으로 앞에 `\`를 붙여야 한다.

numcalc.sh

```
-----  
1.  #!/bin/sh  
2.  num1=100  
3.  num2=$num1+200  
4.  echo $num2  
5.  num3=`expr $num1 + 200`  
6.  echo $num3  
7.  num4=`expr \$( $num1 + 200 \) / 10 \* 2`  
8.  echo $num4  
9.  exit 0
```

# Shell : 매개변수(Parameter)

- 매개변수는 \$0, \$1, \$2 등의 형태를 갖는다.
- 실행하는 명령의 부분 하나하나를 변수로 저장한다는 의미
- 전체의 파라미터 변수는 \$\* 로 표현한다.

실행 :

\$ sh paravar.sh 값1 값2 값3

paravar.sh

- ```
-----
```
1. #!/bin/sh
  2. echo "Name of execution file is <\$0>"
  3. echo "1st parameter <\$1>, 2nd parameter <\$2>"
  4. echo "All of parameters <\$\*>"
  5. exit 0

# Shell : 기본 if 문

- if 와 then 같은 라인에 사용하지 말 것
- [ 조건 ] 작성시, "[", "]"와 조건은 띄워 쓸 것.
- "if [ 조건 ]" 과 "if test 조건" 동일한 문장.

```
if [ 조건 ]
```

```
then
```

```
    참일 경우 실행할 명령
```

```
fi
```

```
exit 0
```

if1.sh

```
-----  
1.  #!/bin/sh  
2.  company=RACOS  
3.  if [ $company = "RACOS" ]  
4.  then  
5.      echo "it's true"  
6.  fi  
7.  exit 0
```

# Shell : if - else 문

- if 와 then 같은 라인에 사용하지 말 것
- [ 조건 ] 작성시, "[", "]"와 조건은 띄워 쓸 것.
- "if [ 조건 ]" 과 "if test 조건" 동일한 문장.

```
if [ 조건 ]
```

```
then
```

```
    참일 경우 실행할 명령
```

```
else
```

```
    거짓일 경우 실행할 명령
```

```
fi
```

```
exit 0
```

if2.sh

```
-----  
1.  #!/bin/sh  
2.  company=RACOS  
3.  if [ $company != "RACOS" ]  
4.  then  
5.      echo "it's true"  
6.  else  
7.      echo "it's false"  
8.  fi  
9.  exit 0
```



# Shell : if 조건연산자 (문자열 비교)

| 문자열 비교           | 결과               |
|------------------|------------------|
| "문자열1" = "문자열2"  | 두 문자열이 같으면 참     |
| "문자열1" != "문자열2" | 두 문자열이 다르면 참     |
| -n "문자열"         | 문자열이 NULL이 아니면 참 |
| -z "문자열"         | 문자열이 NULL이면 참    |

if3.sh

---

```
1. #!/bin/sh
2. if [ -n "" ]
3. then
4.     echo "it's true"
5. else
6.     echo "it's false"
7. fi
8. exit 0
```

# Shell : if 조건연산자 (산술 비교)

| 산술 비교       | 결과             |
|-------------|----------------|
| 수식1 -eq 수식2 | 두 수식이 같으면 참    |
| 수식1 -ne 수식2 | 두 수식이 같지 않으면 참 |
| 수식1 -gt 수식2 | 수식1이 크면 참      |
| 수식1 -ge 수식2 | 수식1이 크거나 같으면 참 |
| 수식1 -lt 수식2 | 수식1이 작으면 참     |
| 수식1 -le 수식2 | 수식1이 작거나 같으면 참 |
| !수식         | 수식이 거짓이면 참     |

if4.sh

```
-----  
1. #!/bin/sh  
2. num=100  
3. if [ num -lt 50 ]; then  
4.     echo "num is less than 50"  
5. elif [ num -gt 50 ]; then  
6.     echo "num is greater than 50"  
7. else  
8.     echo "100 != 200"  
9. fi  
10. exit 0
```

# Shell : if 조건연산자 (파일과 관련된 조건)

| 파일 조건       | 결과                       |
|-------------|--------------------------|
| -d 파일이름     | 파일이 디렉토리이면 참             |
| -e 파일이름     | 파일이 존재하면 참               |
| -f 파일이름     | 파일이 일반 파일이면 참            |
| -g 파일이름     | 파일에 set-group-id가 설정되면 참 |
| -r 파일이름     | 파일이 읽기 가능 상태이면 참         |
| -s 파일이름     | 파일 크기가 0이 아니면 참          |
| -u 파일이름     | 파일에 set-user-id가 설정되면 참  |
| -w 파일이름     | 파일이 쓰기 가능 상태이면 참         |
| -x 파일이름     | 파일이 실행 가능상태이면 참          |
| 파일1 -nt 파일2 | 파일1이 파일2보다 최신이면 참.       |

\*\*\* 일반 파일이란 Directory 또는 Device 파일이 아니라는 의미

if5.sh

```
-----  
1. #!/bin/sh  
2. fname=/lib/systemd/system/sshd.service  
3. if [ -f $fname ]  
4. then  
5.     head -5 $fname  
6. else  
7.     echo "sshd server has NOT been installed"  
8. fi  
9. exit 0
```

# Shell : case - esac 문

- if 문은 참/거짓만 평가
- case 문은 다중 값을 평가할 수 있다.
- Default 처리는 "\*"

case1.sh

```
-----  
1.  #!/bin/sh  
2.  case "$1" in  
3.      start)  
4.          echo "Start~~";;  
5.      stop)  
6.          echo "Stop~~";;  
7.      restart)  
8.          echo "Restart~~";;  
9.      *)  
10.         echo "Default";;  
11. esac  
12. exit 0
```

# Shell : case - esac 문

- 여러 개의 값을 평가할 때에는 | 를 사용.
- [nN]\* 는 n 또는 N 이 들어간 모든 단어를 인정해 준다는 의미

case2.sh

```
-----  
1.  #!/bin/sh  
2.  echo "Is your name 'Inho Jang'? (yes/no)"  
3.  read answer  
4.  case $answer in  
5.      yes | Yes | YES | y | Y)  
6.      echo "Hello Mr. Jang";;  
7.      [nN]*)  
8.      echo "Sorry!! What is your name";;  
9.      *)  
10.     echo "Who are you"  
11.     exit 1;;  
12. esac  
13. exit 0
```

# Shell : and or 관계 연산자

- and 는 -a 또는 &&
- or 는 -o 또는 ||
- -a 또는 -o는 [ ] 안에 사용할 수 있으며, 특수문자 앞에는 \ 사용
- 4번 행 동치  
if [ \\$( -f \$fname ) -a \\$( -s \$fname ) ]; then

case2.sh

```
-----  
1. #!/bin/sh  
2. echo "Please type a file name"  
3. read fname  
4. if [ -f $fname ] && [ -s $fname ]; then  
5.     head -5 $fname  
6. else  
7.     echo "No exist or size 0"  
8. fi  
9. exit 0
```

# Shell : for in 문

```
for 변수 in 값1, 값2, 값3 ...
```

```
do
```

```
    반복할 문장
```

```
done
```

- 과제 :

현재 디렉토리에 있는 모든 Shell 파일(\*.sh)의 내용 3줄 씩을 출력.

forin1.sh

```
-----  
1. #!/bin/sh  
2. tot=0  
3. for i in 1 2 3 4 5 6 7 8 9 10  
4. do  
5.     tot=`expr $tot + $i`  
6. done  
7. echo "Total from 1 to 10 : " $tot  
8. exit 0
```

# Shell : while/until 문

```
while [ 조건 ]    // true인 동안 계속 돈다.
```

```
Do
```

```
    반복할 문장
```

```
Done
```

```
until [ 조건 ]    // true 될 때까지 계속 돈다. 즉, false인 동안 계속 돈다.
```

```
do
```

```
    반복할 문장
```

```
done
```

과제1 :

앞의 예제 1 ~ 10을 for 문을 이용하여 합산하는 계산을 while을 이용하여 스크립트를 작성하시어.

과제2 :

비밀번호를 입력 받고, 그 비밀번호가 맞을 때까지 계속 입력하는 스크립트를 작성하시오. (비밀번호는 "1234"로 한다)

```
while1.sh
```

```
-----  
1.  #!/bin/sh  
2.  while [ 1 ]  
3.  do  
4.      echo "I'm LEGEND"  
5.  done  
6.  exit 0
```



# Shell : 사용자 정의 함수 I

```
함수이름 () {  
    내용들...  
}
```

func1.sh

---

```
1. #!/bin/sh  
2. myFunc () {  
3.     echo "You are in the function"  
4.     return;  
5. }  
6. echo "Program is started"  
7. myFunc  
8. echo "Program is ended"  
9. exit 0
```

# Shell : 사용자 정의 함수 II

```
함수이름 () {  
    내용들...  
}
```

func2.sh

---

```
1. #!/bin/sh  
2. myFunc () {  
3.     echo `expr $1 + $2`  
4. }  
5. echo "Excute : 10 + 20"  
6. myFunc 10 20  
7. exit 0
```

# Shell : eval

문자열을 명령문으로 인식시켜 실행하는 것.

eval.sh

---

1. `#!/bin/sh`
2. `str="ls -l eval.sh"`
3. `echo $str`
4. `eval $str`
5. `exit 0`

# Shell : export

외부 변수로 선언한다.

선언한 변수를 다른 프로그램에서도 사용할 수 있게 한다.

exp1.sh

---

1. `#!/bin/sh`
2. `echo $var1`
3. `echo $var2`
4. `exit 0`

exp2.sh

---

1. `#!/bin/sh`
2. `var1="local variable"`
3. `export var2="global variable"`
4. `sh exp1.sh`
5. `exit 0`

# Shell : printf

C 언어의 printf() 함수와 비슷하게 형식을 지정해서 출력한다.

printf.sh

```
-----  
1. #!/bin/sh  
2. var1=100.5  
3. var2="exciting RACOS life"  
4. printf "%5.2f \n\n \t %s \n" $var1 "$var2"  
5. exit 0
```

# Shell : set

- 리눅스 명령을 결과로 사용하려면 '\$(명령)' 형식을 사용해야 한다.
- 결과를 파라미터로 사용하려면 set 명령과 함께 사용해야 한다.

set.sh

---

```
1. #!/bin/sh
2. echo " Today is $(date)"
3. set $(date)
4. echo "This month is $2"
5. exit 0
```

# Shell : shift

- 파라미터 변수를 왼쪽으로 한 단계씩 시프트(이동) 시킨다.

shift.sh

```
-----  
1. #!/bin/sh  
2. myfunc () {  
3.     str=""  
4.     while [ "$1" != "" ];  
5.     do  
6.         str="$str $1"  
7.         shift  
8.     done  
9.     echo $str  
10. }  
11. myfunc AAA BBB CCC DDD EEE FFF GGG HHH  
12. exit 0
```

# Shell 001 : 두 파일을 비교해서 오래된 파일 삭제

```
$ sh olddel.sh 파일1 파일2
```

Hint>

매개변수는 \$0, \$1, \$2 ...으로 참조할 수 있다.

에러는 표준 출력인 >&2로 출력한다.

2개 파일 중 어느 것이 더 최신인지 확인하기 위한 조건연산자 : -nt

```
1.  #!/bin/sh
2.  file1=$1
3.  file2=$2
4.  filecheck()
5.  {
6.      if [ ! -e "$1" ]; then
7.          echo "Error: File $1 does not exist." >&2
8.          exit 1;
9.      fi
10. }
11. filecheck "$file1"
12. filecheck "$file2"
13. If [ "file1" -nt "file2" ]; then
14.     echo "[file1]-> newer, [file2]->older"
15.     rm $file2
16. else
17.     echo "[file2]-> newer, [file1]->older"
18.     rm $file1
19. Fi
20. exit 0
```



# Shell 002 : 선택메뉴 표시 & 처리

```
$ sh select.sh
```

- 1) List files
- 2) Current Directory
- 3) Exit

```
$ 1 -> ls 명령 수행
```

```
$ 2 -> pwd 명령 수행
```

```
$ 3 -> shell 종료
```

```
$ 5 -> 메뉴 외 숫자/문자 선택시 "Unknown Command" 에러 출력
```

```
1. #!/bin/sh
2. while :
3. do
4.     echo "Menu:"
5.     echo "1) List files"
6.     echo "2) Current Directory"
7.     echo "3) Exit"

8.     read num
9.     case $num in
10.         1)
11.             ls
12.             ;;
13.         2)
14.             pwd
15.             ;;
16.         3)
17.             exit
18.             ;;
19.         *)
20.             echo "Error: Unknown Command" >&2
21.             ;;
22.     esac
23.     echo
24. done
```

# Shell 003 : 변수명 확장

- 변수명은 알파벳, 숫자, 언더스코어 세 종류로만 만들어야 한다.
- 3, 4라인의 변수명을 \$today\_log 사용하면 어떻게 될까?

varname.sh

- ```
-----
```
1. #!/bin/sh
  2. today="20220502"
  3. ls -al > \${today}\_log
  4. wc -l \${today}\_log
  5. exit 0

# Shell 004 : 배열 변수

- declare 변수 선언시 사용
- declare -a 는 배열 선언
- \$number[1] 어떻게 해석된 것인가?

array.sh

---

1. #!/bin/sh
2. declare -a number=("zero" "one" "two")
3. echo \${number[1]}
4. echo \$number[1]
5. exit 0

# Shell 005 : 변수선언 declare

- declare -i : 정수 선언
- declare -r : readonly 선언

declare.sh

---

```
1.  #!/bin/sh
2.  declare -i var1
3.  var1=2367
4.  echo "var1 is declared to $var1"
5.  var1=var1+1
6.  echo "var1 increased by 1 is $var1"

7.  echo "Trying to change var1 to 2367.1"
8.  var1=2367.1
9.  echo "var1 is still $var1"

10. declare -r var2=13.36
11. echo "var2 is declared to $var2"

12. var2=13.37
13. echo " var2 is still $var2"

14. exit 0
```

# Shell 006 : 값이 정수인지 확인해서 계산하기

**\$ sh int-check.sh 100a**

- \$ test -f Videos
- \$ echo \$?
- \$ test 100 -eq 0
- \$ echo \$?
- 같으면 0, 다르면 1, 비교불가능하면 2

int-check.sh

---

1. #!/bin/sh
2. test "\$1" -eq 0 2>/dev/null
3. if [ \$? -lt 2 ]; then
4. echo "Argument is Integer."
5. expr 10 + \$1
6. else
7. echo "Argument is not Integer."
8. exit 1
9. fi
10. exit 0

# Shell 007 : Sub Shell

**\$ sh subshell.sh**

- Sub Shell은 현재 Shell 안에서 새롭게 실행되는 Shell.
- Sub Shell을 부른 곳의 환경 설정은 Sub Shell에 이어지지만 Sub Shell 내부에서 일어나는 변경은 호출한 원래 Shell에는 영향을 주지 않는다.
- 즉, Sub Shell은 자식 프로세스가 되고 원래 Shell은 부모 프로세스가 된다.

subshell.sh

---

1. `#!/bin/sh`
2. `(`
3. `echo "List up /var/tmp"`
4. `cd /var/tmp`
5. `ls`
6. `)`
7. `echo "Where are you?"`
8. `pwd`
9. `exit 0`

# Shell 008 : 파일과 디렉토리 수 조사하기

```
$ sh findcount.sh /etc
```

```
findcount.sh
```

- ```
-----
```
1. `#!/bin/sh`
  2. `targetdir="$1"`
  3. `filecount=$(find "$targetdir" -maxdepth 1 -type f -print | wc -l)`
  4. `dircount=$(find "$targetdir" -maxdepth 1 -type d -print | wc -l)`
  5. `dircount=$(expr $dircount - 1)`
  6. `echo "Target : $targetdir"`
  7. `echo "File count : $filecount"`
  8. `echo "Directory Count : $dircount"`
  9. `exit 0`

# Shell 009 : Here Document

**\$ sh here.sh**

- 히어 도큐먼트 사용법

(명령어) << '종료 문자열'

히어 도큐먼트 본체

.....

종료문자열

- 종료문자열에 ' '를 씌우지 않으면
- \$str 변수로 인식된다.
- 변수로 인식되지 않길 원하면 w\$str 하면 된다.

here.sh

- 
1. #!/bin/sh
  2. str="Dummy"
  3. cat << 'EOT'
  4. Here is the body of Here Document.
  5. Even though, \$str is NOT displayed
  6. echo \$str, also not working.
  7. EOT
  8. echo \$str
  9. exit 0



# Shell 010 : Here String

**\$ sh here\_string.sh**

- 히어 도큐먼트와 다르게 << 가 아니라 <<<
- 히어 도큐먼트에서는 EOT 같은 종료 문자열을 사용했지만 히어 스트링은 단순히 " "로 포함하고 싶은 문자열을 묶어 주기만 하면 되므로 더 직관적이다.
- 히어스트링 o의 큰따옴표를 작은따옴표로 바꾸면 어떻게 될까?

here\_string.sh

- ```
-----
```
1. #!/bin/sh
  2. string="Hello"
  3. cat <<< "Greeting sample:
  4. \$string
  5. Annyeonghaseyo
  6. Nihao
  7. Hola
  8. Kon-nichiwa
  9. Salut"
  10. exit 0

# 원격 접속 : 3가지 방법

	telnet	ssh	XRDP
속도	빠르다	빠르다	약간 느리다
그래픽 지원	X	X	O
보안	취약	강	중
사용 가능 명령	CLI 명령만	CLI 명령만	제한없음
클라이언트	대개의 OS에서 사용 가능	리눅스에서 기본 Windows 별도 설치	Windows에 포함

# 원격 접속 : telnet

- 사용 방법 : `$ telnet IP-Address`
- 설치 방법 : `$ dnf -y install telnet-server`
  
- 서버 기동 : `$ systemctl start telnet.socket`
- 서버 기동 : `$ systemctl enable telnet.socket` (시스템 시작시 자동 시작)
- 서버 중지 : `$ systemctl stop telnet.socket`
- 서버 상태 : `$ systemctl status telnet.socket`
  
- 접속할 telnet 서버의 IP 확인 : `$ ifconfig -a` 또는 `$ ifconfig ens160`
  
- 방화벽 설정 확인 : `$ firewall-config`
- `$ firewall-cmd (--permanent) --add-service=telnet`
- `$ firewall-cmd (--permanent) --add-port=23/tcp`
- 재부팅 후에도 방화벽 설정을 유지하려면 `--permanent` 옵션을 붙여 준다.

# 원격 접속 : OpenSSH

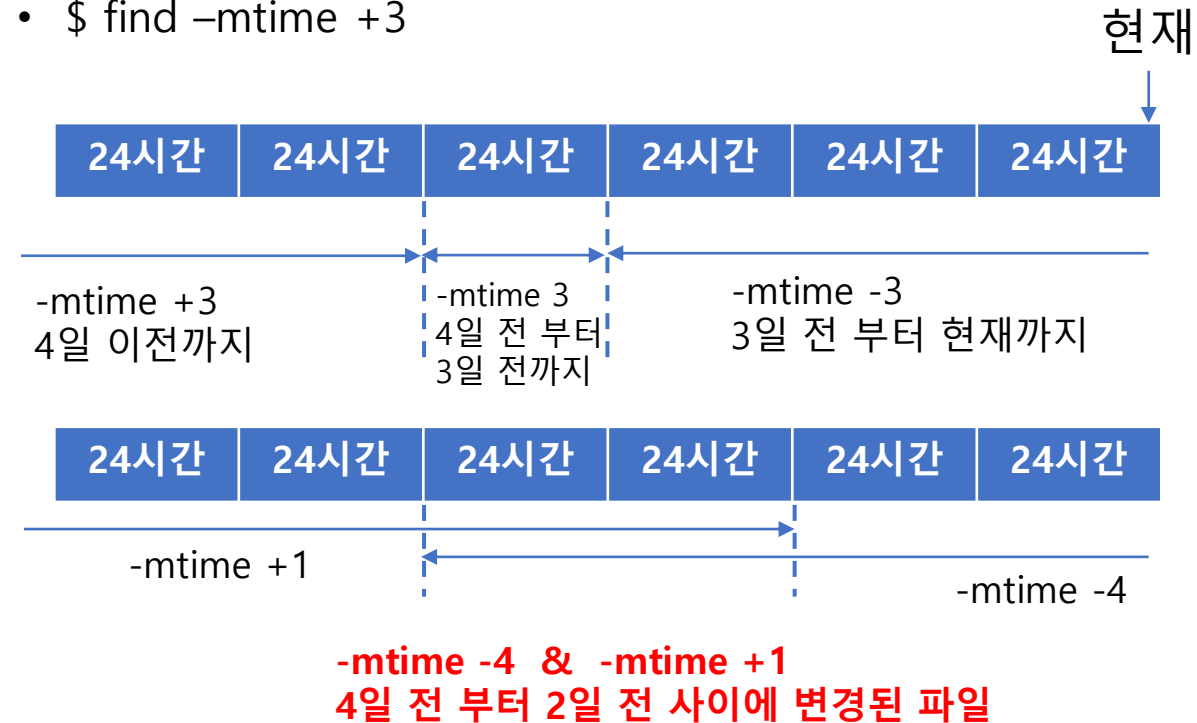
- 사용 방법 : `$ ssh user@IP-Address`
- 설치 방법 : `$ dnf -y install openssh-server` : 리눅스에는 기본 설치/설정 되어 있음.
- 서버 기동 : `$ systemctl start sshd`
- 서버 기동 : `$ systemctl enable sshd` (시스템 시작시 자동 시작)
- 서버 중지 : `$ systemctl stop sshd`
- 서버 상태 : `$ systemctl status sshd`
- 접속할 ssh 서버의 IP 확인 : `$ ifconfig -a` 또는 `$ ifconfig ens160`
- 방화벽 설정 확인 : `$ firewall-config`
- ssh 서버의 Port 기본 번호 : 22
- Windows에서는 ssh 서버에 접속하기 위한 클라이언트가 기본 제공되지 않음 : 한글 PuTTY 설치 필요.

# 원격 접속 : XRDP

- 사용 방법 : Windows에서는 "원격데스크탑" 으로 접속
- Repo설치 : `$ dnf -y install epel-release`
- 설치 방법 : `$ dnf -y install xrdp`
- 서버 기동 : `$ systemctl start xrdp`
- 서버 기동 : `$ systemctl enable xrdp` (시스템 시작시 자동 시작)
- 서버 중지 : `$ systemctl stop xrdp`
- 서버 상태 : `$ systemctl status xrdp`
- 접속할 ssh 서버의 IP 확인 : `$ ifconfig -a` 또는 `$ ifconfig ens160`
- 방화벽 설정 확인 : `$ firewall-config`
- xrdp 서버의 Port 기본 번호 : 3389

# Shell 011 : 특정 디렉토리에 n일 전 ~ m일 전 목록

- \$ find -mtime -3
- \$ find -mtime 3
- \$ find -mtime +3



find-mtime.sh

1. #!/bin/sh
2. logdir="/var/log"
3. find \$logdir -name "\*.log" -mtime -4 -mtime +0  
-print
4. exit 0

# Shell 012 : 1년 이상 갱신되지 않은 파일 삭제

- xargs 명령어는 파일 목록을 인수로 받아서 임의의 명령어를 실행
- rm 명령에서 해당 파일이 하나도 없을 때도 에러가 발생하지 않도록 -f 옵션을 쓰고 동시에 -v 옵션도 써서 삭제한 파일명을 표시한다.

find-rm.sh

```
-----  
1.  #!/bin/sh  
2.  logdir="/var/log"  
3.  find $logdir -name "*.log" -mtime +364 -print | xargs rm -fv  
4.  exit 0
```

# DNS : 개념 잡기

1. DNS : Domain Name Server, 네임서버라고 불리기도 함.
2. URL -> IP Address로 변환을 담당하는 서버, [www.racosys.com](http://www.racosys.com)을 IP 주소로 변환하는 과정을 "Name Resolution" 이라고 함.
3. 전화기 초창기 : 메모나 외워서
4. 전화기 보급기 : 수첩(필요한 전화번호를) 적어서 관리했으나 실시간으로 생성/변경되는 전화번호에 대응하기 어려움.
  - ① Windows에서 hosts 파일의 위치 : C:\Windows\system32\drivers\etc\hosts
  - ② Linux에서 hosts 파일의 위치 : /etc/hosts
5. 전화기 확대기 : 114 -> DNS
  - ① Nameserver 위치 지정 : /etc/resolv.conf
  - ② resolv.conf 파일을 수정하더라도 시스템을 리부팅하거나 네트워크가 재시작되면 다시 /etc/sysconfig/network-scripts/ifcfg-ens160 파일에 설정된 내용으로 초기화 됨.



# DNS : nslookup

- DNS 서버 확인과 대표 URL을 통한 IP 확인

```
$ nslookup
```

```
> server
```

```
> www.naver.com
```

```
> www.google.com
```

```
> www.racosys.com
```

```
> exit
```

- 서강대학교 ( [www.sogang.ac.kr](http://www.sogang.ac.kr) )

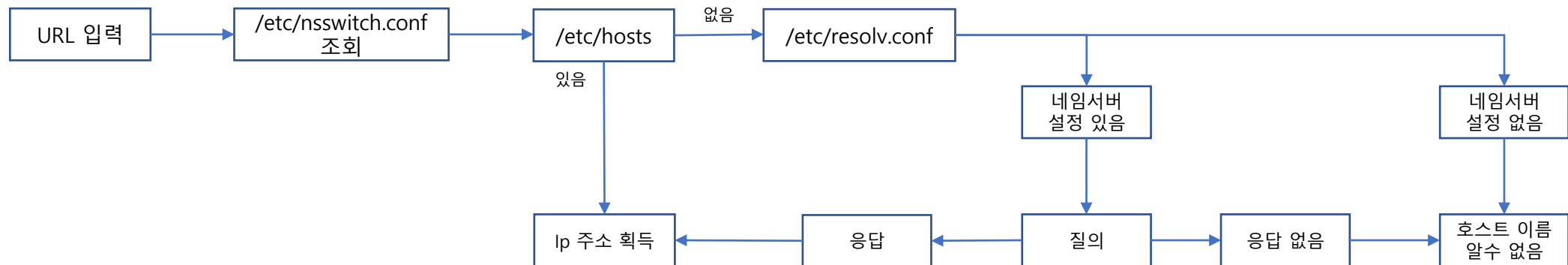
- 1) IP주소를 확인한다.
- 2) URL로 접속해 본다.
- 3) 확인한 IP 주소로 접속해 본다.

- 서울대학교 ( [www.snu.ac.kr](http://www.snu.ac.kr) )

- 1) IP주소를 확인한다.
- 2) URL로 접속해 본다.
- 3) 확인한 IP 주소로 접속해 본다.

# DNS : /etc/host.conf

- resolv.conf 파일의 nameserver 부분을 주석처리 한 후, [www.sogang.ac.kr](http://www.sogang.ac.kr) 로 접속해 본다.
- nslookup으로 확인한 [www.sogang.ac.kr](http://www.sogang.ac.kr)의 IP 주소로 접속해 본다.
- hosts 파일에 서강대 URL과 IP 주소를 등록한 후, [www.sogang.ac.kr](http://www.sogang.ac.kr)로 접속해 본다.
- 위의 실습으로 알 수 있는 사실은?
- hosts 파일에 서울대학교 URL에 서강대학교 IP 주소를 맵핑하면?



/etc/nsswitch.conf : URL 입력시 IP 주소를 얻기 위해 먼저 확인해야 할 것이 결정되어 있다.

hosts: files dns => /etc/hosts 파일 먼저 찾아 보고 없으면 /etc/resolv.conf 에 정의된 nameserver에 질의한다. : Default

hosts: dns files => 위와 반대.

host.conf의 multi on : hosts 파일에 여러 개의 URL을 사용할 수 있다는 의미

# DNS : Root DNS

최상위 DNS으로써, 전 세계적으로 오직 13대의 대형서버로 미국 10대, 일본 1대, 네덜란드, 노르웨이에 각각 1대씩의 대형 서버로 구축되어 있음.

우리나라의 경우 일본의 M-ROOT DNS를 써왔지만, 2003년 8월 ROOT DNS 미러 서버의 국내 운영이 공식 개통.

미러(Mirror) 서버란, 말 그대로 글로벌 오리지널 루트 서버를 복사한 것으로, 거의 같은 기능. 그러나, 설치된 위치만 다를 뿐, 미.일.유럽의 글로벌 오리지널 루트 서버 관리자가 원격으로 통제, 관리하게 있음.

미러 서버가 있는 나라는 굳이 외국 루트 서버에 의존할 필요없이 자체적으로 인터넷 통신을 관리할 수 있고, 현재 전 세계에 67개의 미러 서버가 있으며 미국이 24개, 한국.영국.네덜란드.중국이 3대씩 보유하고 있음.

국내에 설치된 3대의 서버는 F-ROOT 미러 서버와 J-ROOT 미러 서버, M-ROOT 미러 서버이며, KRNIC(한국인터넷진흥원), KT, KINX(한국인터넷연동센터)에서 각각 운영, 관리하고 있으며, 1차 상위 DNS서버인 COM DNS와 NET DNS는 KT 분당 인터넷데이터센터에 운영, 관리 중.

미러 서버가 생긴 이유는, 2002년 ROOT서버에 DDOS(분산 서비스 거부) 공격이 있었고, 이 때, ROOT DNS 13대 중 9대의 대형 서버가 영향을 받게 되어 전세계는 몇시간 동안 도메인을 이용한 인터넷 서비스를 사용 할 수 없는 상황이 생겼고, 2차 DDOS공격의 피해를 막기위해, 애니캐스트(anycast)기술을 적용시켜, 전세계에 67개의 ROOT 미러 DNS를 구축했음.

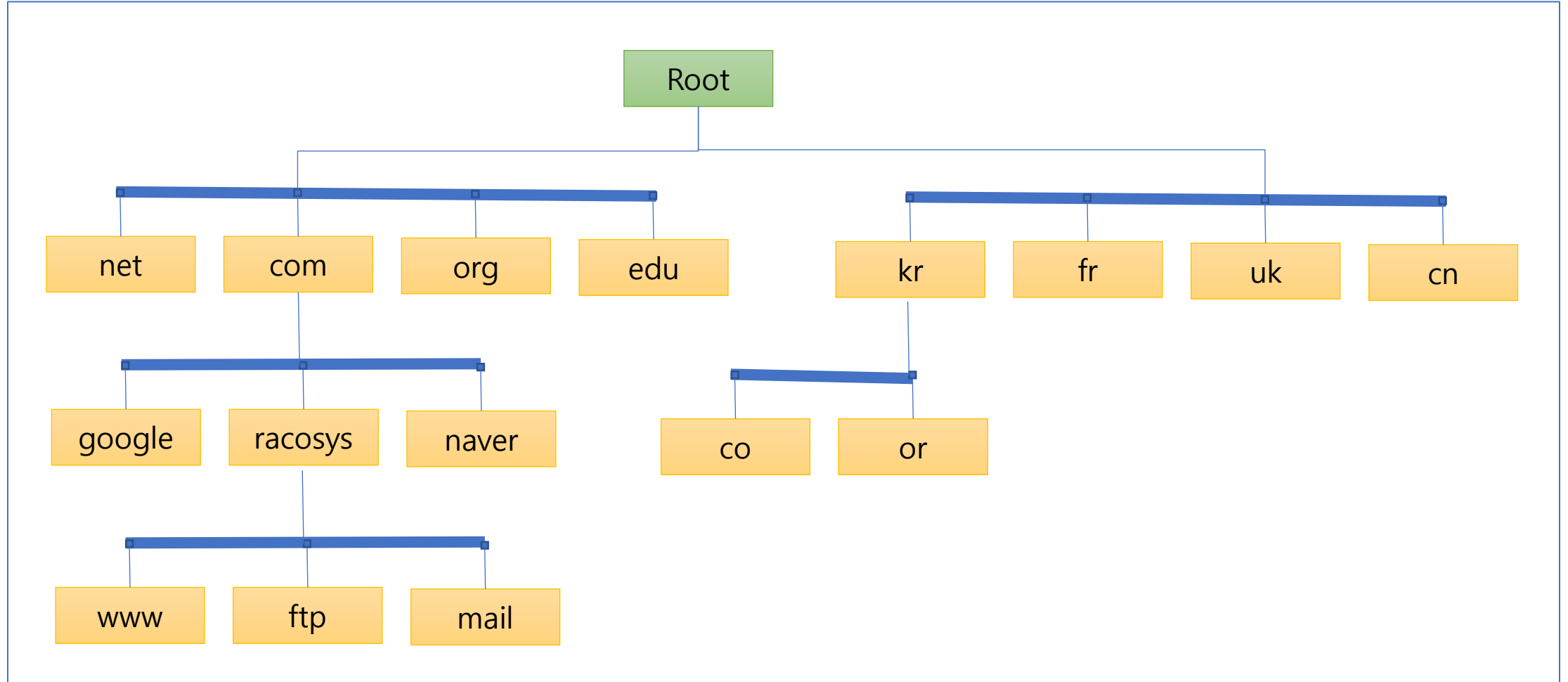
애니캐스트(anycast)란, 라우팅 경로가 가장 가까운곳으로 통신하는 기술로, 도메인에 대한 IP요청시, 가장 가까운 경로에 있는 ROOT 미러 DNS로 통신을 할수 있도록 하는 것.

현재 KT, 하나로통신 등이 F형 미러 서버를 주로 이용하고, 두루넷, 온세통신, 드림라인 등 나머지 업체들은 M형 미러 서버를 이용하고 있음.

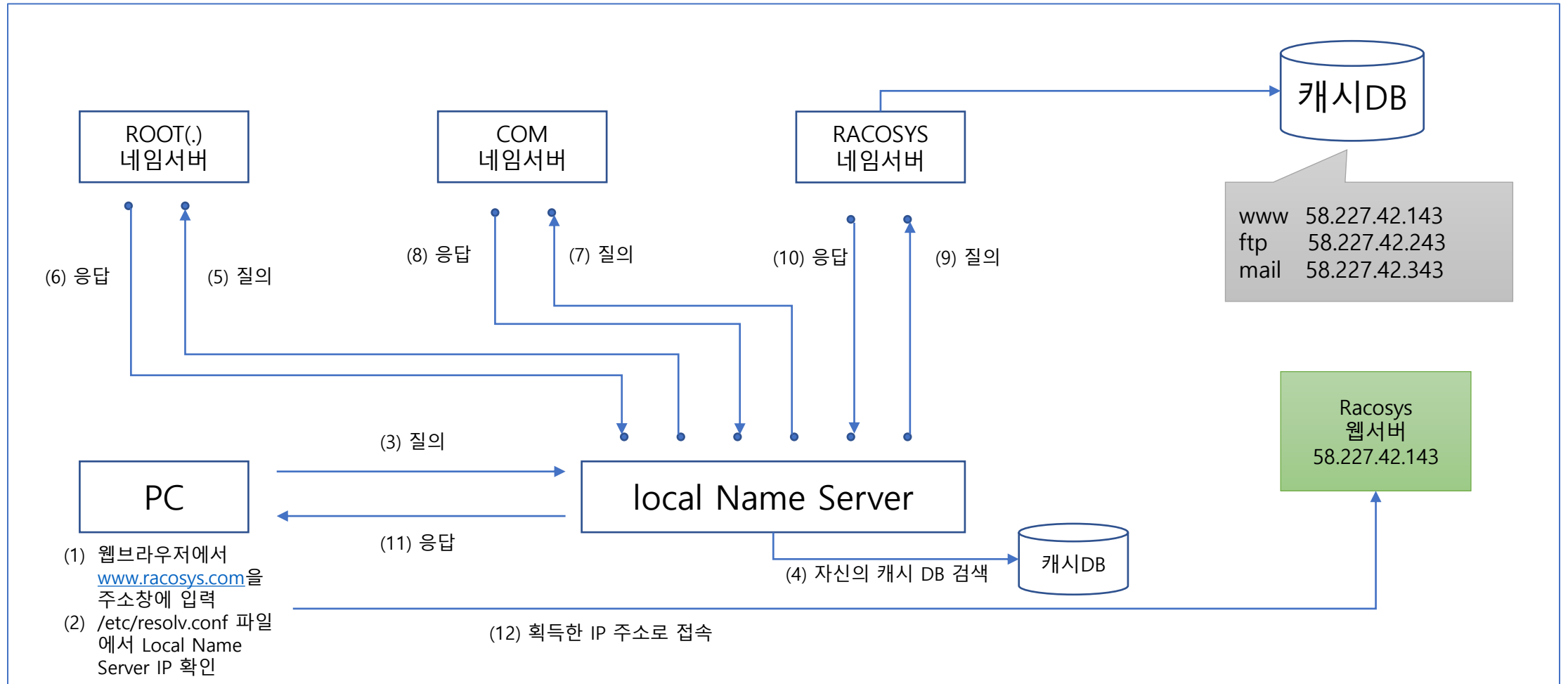
우리나라에서도 ROOT DNS 미러서버와 COM/NET DNS가 국내에 설치 됨에 따라, 그 동안 외국의 DNS에 의존하던 DNS질의를 국내에서 대부분 처리 할 수 있게 되었으며, 국제 트래픽 및 네트워크 회선비용의 감소, DNS질의에 대한 응답 시간 단축 등, 경제적인 효과 뿐만 아니라, 동남아 등 동아시아 지역의 DNS 질의처리도 가능함.

<https://xn--3e0bx5euxnjj69i70af08bea817g.xn--3e0b707e/jsp/statboard/dns/dnsRoot/currentWorld.jsp>

# DNS : 개념 잡기



# DNS : Name Server가 작동하는 순서



# DNS : 캐시전용네임서버

- 설치 : `$ dnf -y install bind bind-chroot`
- 설정 : `/etc/named.conf`

행	before	after
11	<code>listen-on port 53 { 127.0.0.1; };</code>	<code>listen-on port 53 { any; };</code>
12	<code>listen-on-v6 port 53 { ::1; };</code>	<code>listen-on-v6 port 53 { none; };</code>
19	<code>allow-query { localhost; };</code>	<code>allow-query { any; };</code>
34	<code>dnssec-validation yes;</code>	<code>dnssec-validation no;</code>

- `$ systemctl restart named`
- `$ systemctl enable named`
- `$ systemctl status named`
- `$ firewall-config` : 영구적(Permanent) – 공용(public) – 서비스(service) : dns 체크
- `$ dig @192.168.xxx.xxx www.racosys.com`
- `$ nslookup > server 192.168.xxx.xxx > www.racosys.com`

# DNS : xxx.com(사설) 네임서버 설치/운영 (www,ftp)

<p>웹서버 설치 : \$ dnf -y install httpd          웹서버 시작 : \$ systemctl start httpd          웹서버 상태 : \$ systemctl status httpd</p> <p>방화벽 설정 : \$ firewall-config   영구적-공용-서비스 : http, https   체크</p> <p>/var/www/html/index.html 편집 :</p> <p>&lt;h1&gt;It's the CentOS Web Server.&lt;/h1&gt;</p>	<p>FTP 설치 : \$ dnf -y install vsftpd          FTP 시작 : \$ systemctl start vsftpd          FTP 상태 : \$ systemctl status vsftpd</p> <p>방화벽 설정 : \$ firewall-config   영구적-공용-서비스 : ftp   체크</p> <p>/etc/vsftpd/vsftpd.conf 편집(1행에) : banner_file=/var/ftp/welcome.msg          /var/ftp/welcome.msg 편집 :</p> <p>#####          Welcome !!! This is Linux FTP Server.          #####</p>
<p>• /etc/resolv.conf 편집 : nameserver   192.168.xxx.xxx</p>	
<p>/etc/named.conf를 아래와 같이 편집</p> <pre>zone "xxx.com" IN {     type master;     file "xxx.com.db";     allow-update { none; }; };</pre> <p>명령이 문법상 이상 없는지 확인 : \$ named-checkconf</p>	<p>/var/named/로 이동하여 \$ touch xxx.com.db 명령으로 빈파일 생성하여 아래와 같이 편집한 후, 다음과 같이 문법 확인 \$ named-checkzone xxx.com xxx.com.db</p> <pre>\$TTL      3H @ SOA  @   root. ( 2 1D 1H 1W 1H )       IN  NS  @       IN  A   192.168.xxx.xxx  www      IN  A   192.168.xxx.xxx ftp      IN  A   192.168.xxx.xxx</pre>