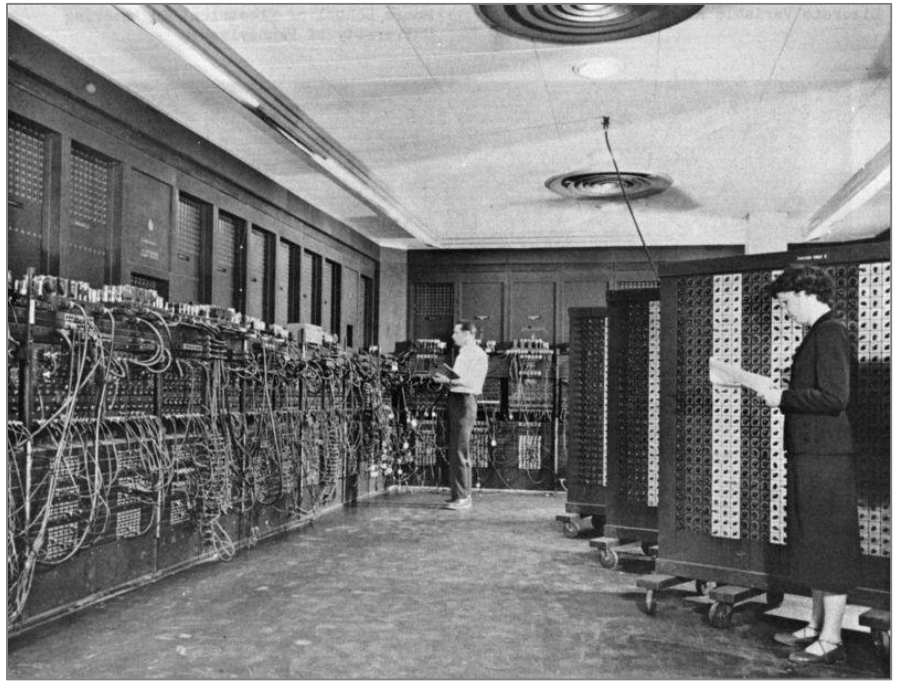
Linux™

알쓰신잡

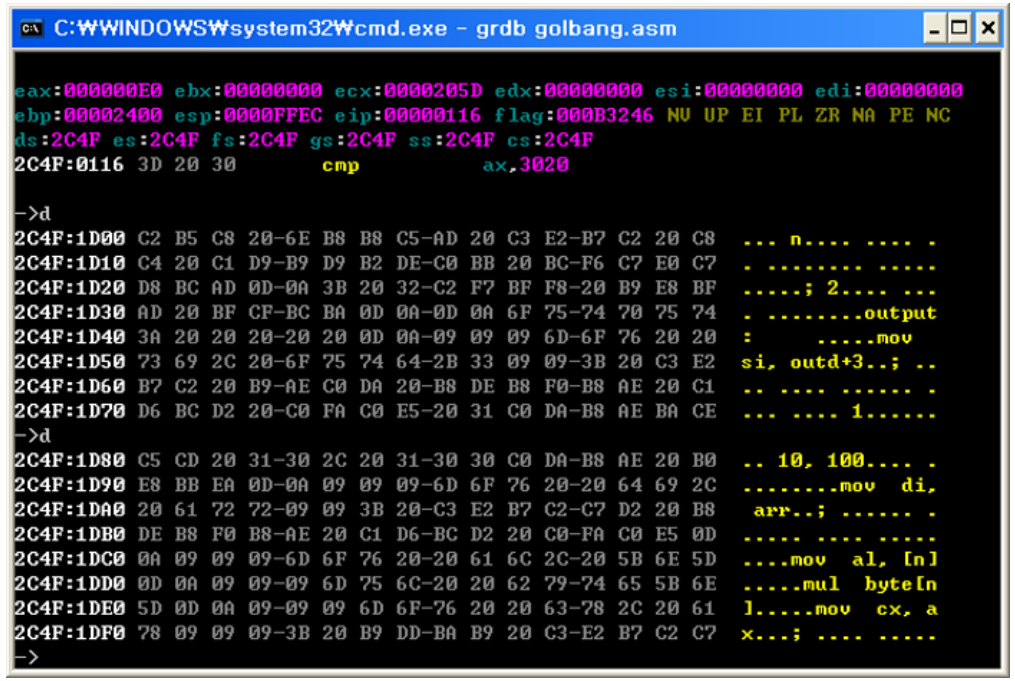
Made by GyuWon Hong

OS (Operating System)

- 컴퓨터 하드웨어를 관리하는 프로그램



△ 애니악 (0과 1의 기계어로만 명령실행)



△ 어셈블리어 (기계어와 일대일 대응이 되는 컴퓨터 프로그래밍의 저급 언어

컴퓨터는 0과 1로만 모든 것을 이해한다.

사람들은 0과 1로 코딩을 했고, 조금 더 프로그래밍을 발전시키며,

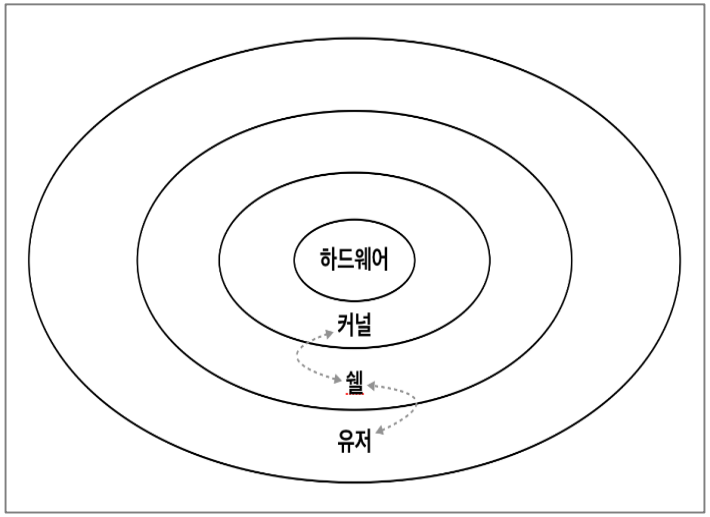
어셈블리어(assembly language, 기계어와 일대일 대응이 되는 컴퓨터 프로그래밍의 저급 언어)부터
지금의 고급 프로그래밍 언어(python, go, JavaScript ..) 까지 발전시켰다.

OS (Operating System)

- 컴퓨터 하드웨어를 관리하는 프로그램

△ C언어

△ 다양한 운영체제의 등장
(하드웨어적인 요소를 몰라도 프로그래밍이 가능)



△ 운영체제 제어

대부분 OS는 초기의 언어 중
하나인 C로 생성이 되었다

사람들은 하드웨어과 소프트웨어를
분리하길 원했고, OS가 등장하게 되었다

OS는 우리의 언어를 쉘에서 번역하여
커널로 넘기고 하드웨어 기능을 실행한다

사용자(명령) -〉 쉘(해석) -〉 커널(명령 수행 후 결과 전송) -〉 쉘(해석) -〉 사용자(결과 확인)

UNIX

1. 대화형 시스템 : 명령어 기반 사용자 인터페이스(CLI, Command Line Interface)를 통해 사용자와 유닉스가 대화한다. 즉, 사용자가 명령어를 입력하면 유닉스는 명령의 결과를 화면에 출력한다.

2. 다중 사용자 시스템 : 네트워크를 통해 여러 사람이 같은 컴퓨터에 동시에 접속해서 작업을 할 수 있다.

3. 멀티태스킹 시스템 : 하나의 컴퓨터에서 여러 작업을 동시에 수행할 수 있다.

4. 높은 이식성, 확장성: 유닉스는 어셈블리어가 아닌 C로 작성돼 있기 때문에 이식성과 확장성이 높다.

5. 계층적 트리 파일 시스템 (디렉토리 구조)

6. 다양한 부가 기능 제공 : 개발, 디버깅 도구, 문서 편집 도구, 출력 도구 등을 제공한다.

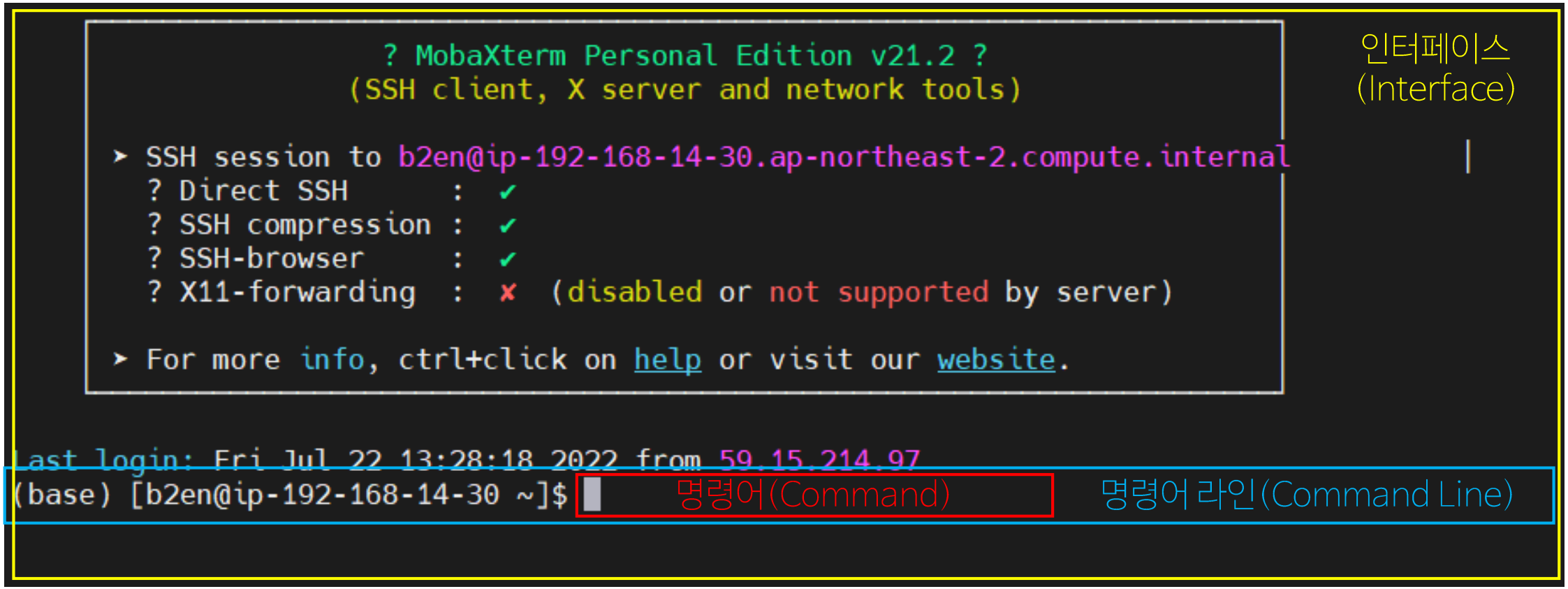
리눅스는 UNIX를 보완하여, 만들어진 오픈소스(Open Source)이다.

따라서 UNIX의 기본적인 특징을 물려받기 때문에, UNIX의 특징에 대한 이해가 필요하다.

왜 대부분 개발 프로젝트 환경에서 OS로 리눅스를 쓰는 지 생각해 본적이 있는가?

프로그램은 외우는 것이 아니라, 이해하는 것이다! 인간의 효율성의 산물인 컴퓨터를 이해해보자!

1. 대화형 시스템 : 명령어 기반 사용자 인터페이스(CLI, Command Line Interface)를 통해 사용자와 대화한다.
   즉, 사용자가 명령어를 입력하면 명령의 결과를 화면에 출력한다.



2. 다중 사용자 시스템 : 네트워크를 통해 여러 사람이 같은 컴퓨터에 동시에 접속해서 작업을 할 수 있다.

```
(base) [b2en@ip-192-168-14-30 etc]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
```

△ 리눅스 OS의 계정들

- 상황 :

Q1. 같은 시간에 계정 A와 계정 B가 동시 접속 할 수 있을 까? 가능

Q2. 계정 A접속하고 5분 뒤 계정 A로 또 접속 할 수 있을 까? 가능

다중 사용자 시스템은 동시 작업을 할 수 있다는 점에서
개발 환경 OS로 리눅스가 채택이 되는 강력한 이유 중 하나이다

3. 멀티태스킹 시스템 : 하나의 컴퓨터에서 여러 작업을 동시에 수행 할 수 있다.

```
(base) [b2en@ip-192-168-14-30 ~]$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Mar30 ?        00:23:30 /usr/lib/systemd/systemd --switched-root --system --deserialize 21
root           2       0  0 Mar30 ?        00:00:01 [kthreadd]
root           6       2  0 Mar30 ?        00:05:13 [ksoftirqd/0]
root           7       2  0 Mar30 ?        00:01:20 [migration/0]
root           8       2  0 Mar30 ?        00:00:00 [rcu_bh]
root           9       2  0 Mar30 ?        04:50:44 [rcu_sched]
root          10       2  0 Mar30 ?        00:00:00 [lru-add-drain]
root          11       2  0 Mar30 ?        00:00:33 [watchdog/0]
root          12       2  0 Mar30 ?        00:00:28 [watchdog/1]
root          13       2  0 Mar30 ?        00:01:22 [migration/1]
root          14       2  0 Mar30 ?        00:03:27 [ksoftirqd/1]
root          17       2  0 Mar30 ?        00:00:27 [watchdog/2]
root          18       2  0 Mar30 ?        00:01:21 [migration/2]
root          19       2  0 Mar30 ?        00:03:23 [ksoftirqd/2]
root          21       2  0 Mar30 ?        00:00:00 [kworker/2:0H]
root          22       2  0 Mar30 ?        00:00:29 [watchdog/3]
root          23       2  0 Mar30 ?        00:01:20 [migration/3]
root          24       2  0 Mar30 ?        00:03:20 [ksoftirqd/3]
root          26       2  0 Mar30 ?        00:00:00 [kworker/3:0H]
root          27       2  0 Mar30 ?        00:00:29 [watchdog/4]
root          28       2  0 Mar30 ?        00:01:19 [migration/4]
```

데몬

컴퓨팅

멀티태스킹 운영 체제에서 데몬은 사용자가 직접적으로 제어하지 않고, 백그라운드에서 돌면서 여러 작업을 하는 프로그램을 말한다. 시스템 로그를 남기는 syslogd처럼 보통 데몬을 뜻하는 'd'를 이름 끝에 달고 있으며, 일반적으로 프로세스로 실행된다. 위키백과

컴퓨터는 동시에 여러 프로그램을 돌릴 수 있다.

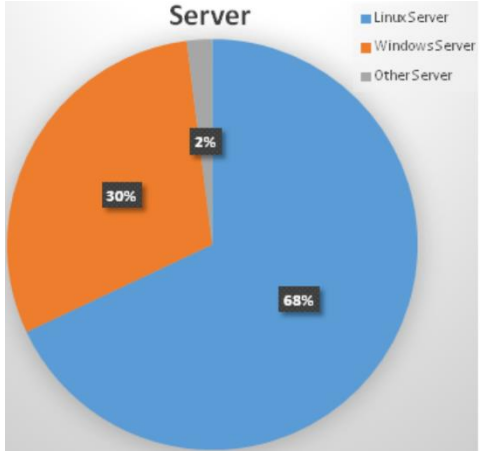
백그라운드 프로세스에서 여러 프로그램들이 돌고 있고, 프로그램을 백그라운드로 돌릴 수도 있다.

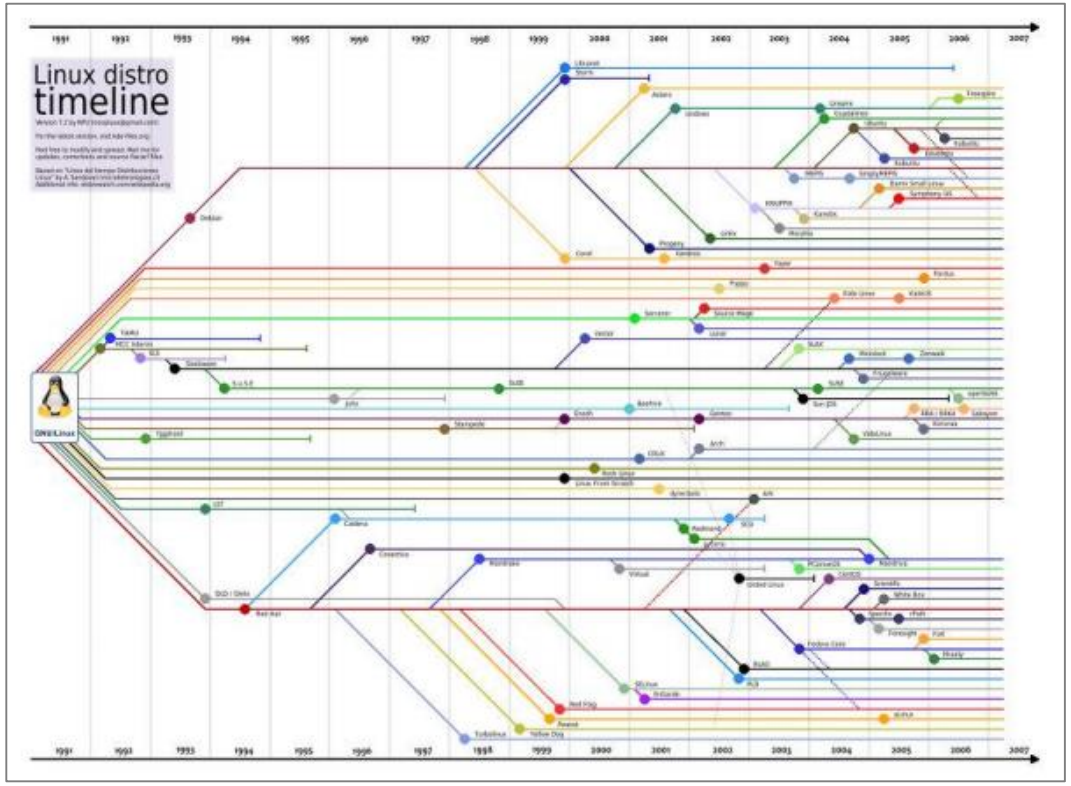
사용자 제어 없이 백그라운드에서 도는 프로그램들을 데몬(daemon)이라고 한다.

LINUX

Linus Torvalds가 만든 운영 체제입니다. UNIX를 모델로 한 오픈 소스 운영 체제이다.
(Tip. 유명 버전 관리 시스템인 git 개발에도 Linus Torvalds가 기여하였다.)

오픈소스이기 때문에, 초기 모델을 변형하여 만든 수 많은 배포 버전이 있다.
그 중 유명한 건 Ubuntu, CentOS .. 가 있다.





△ 리눅스 배포 Timeline

△ 리눅스의 종류와 인기있는 리눅스 종류

4. 높은 이식성, 확장성: 유닉스는 어셈블리어가 아닌 C 언어로 작성돼 있기 때문에 이식성과 확장성이 높다.

이식성과 확장성의 경우 상대적으로 판단 할 필요가 있다.

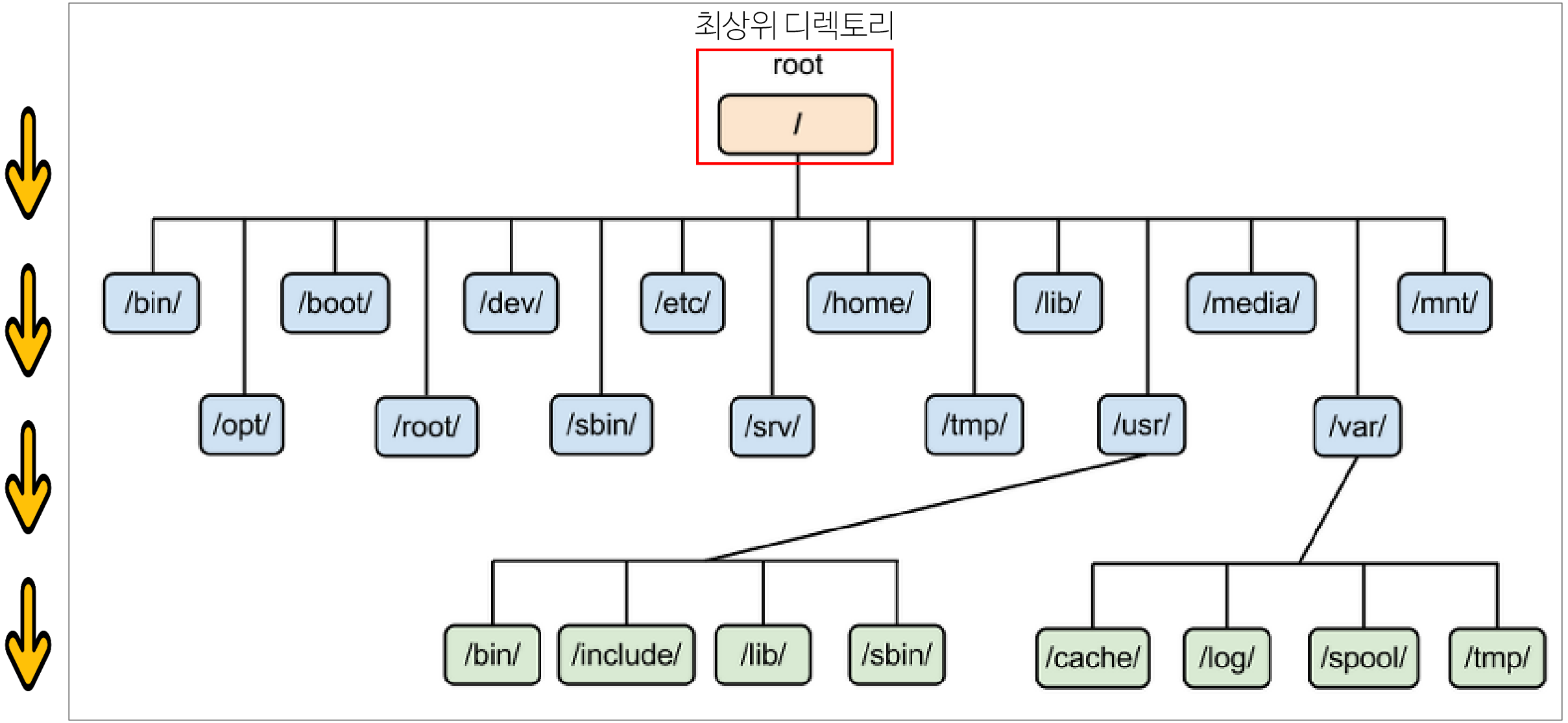
대부분 환경에서 C언어를 기본적으로 지원하기 때문에, 높은 확장성을 가지고 있다고 볼 수 있다.

하지만, Java 또한 개별 Host OS 위에 맞는 JVM(Java Virtual Machine, 자바가상환경)을 설치하여
JVM 위에서 모든 프로그램을 똑같이 실행 할 수 있는 구조를 갖는다.

따라서 어떤 언어가 더 좋다, 나쁘다고 정의하기는 어렵다는 것이 저자의 견해이다.
C언어처럼 저수준 언어는 하드웨어를 조금 더 정밀하게 프로그래밍 가능하여 빠르지만,
일반인이 프로그래밍을 할 때 객체지향 언어보다는 시간이 많이 걸리고, 메모리에 대한 이해 등 개발 시간이 더 걸릴 수 있다.

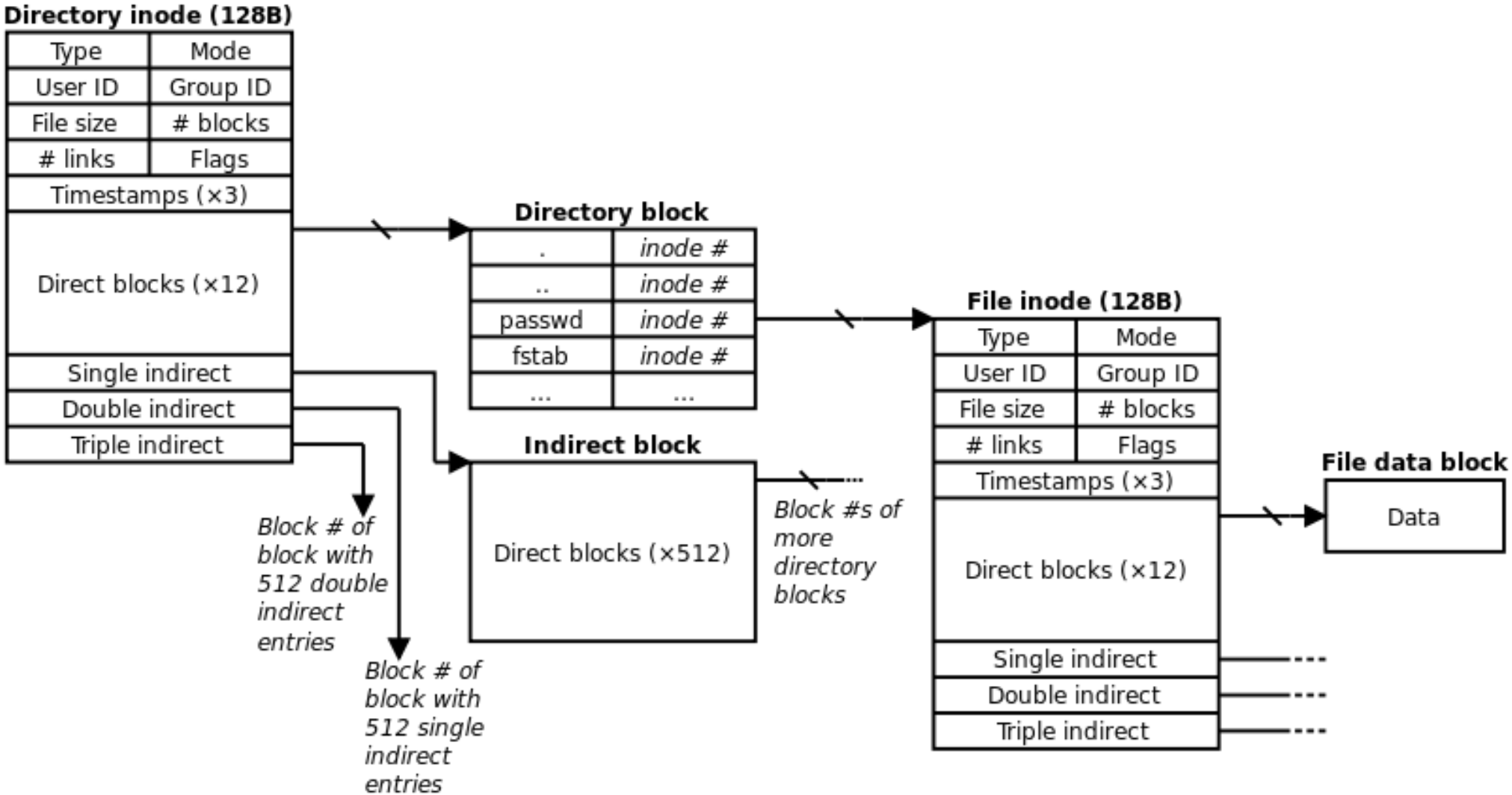
프로그램에서 더 나은 개발 환경을 고려할 때는 언어 뿐 아니라 개발자와 운영, 유지보수 방안 등
복합적 요소를 고려하여 최적의 환경을 선택해야만 한다.

5. 계층적 트리 파일 시스템 (디렉토리 구조)



리눅스의 파일 시스템은 최상위 디렉토리부터 아래로 진행되는 트리 형식의 구조를 가지고 있다

리눅스 : 파일 스토리지 (데이터 저장 방법)



리눅스의 파일 시스템은 inode를 통해 디스크에 파일과 디렉토리의 정보를 저장하고 있다

6. 다양한 부가 기능 제공 : 개발, 디버깅 도구, 문서 편집 도구, 출력 도구 등을 제공한다.

리눅스 OS를 설치하면, 기본적으로 설치가 되어있는 프로그램들이 있다.

대표적으로는 VIM (VI 에디터)가 있다.
VI 명령어를 이용하면 편집 에디터로 접속할 수 있다.

← 리눅스 개발 에디터 VIM

▼ Top 명령어

System time, Uptime, User

Load Average

Task →

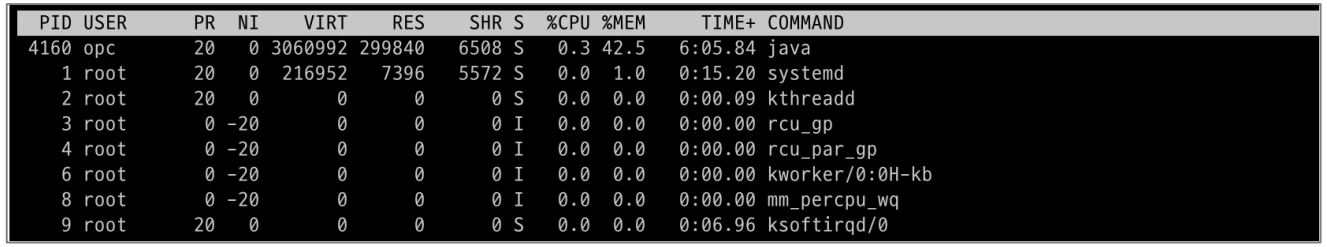
CPU →

Memory

```
top - 10:10:03 up 141 days, 18:01,  3 users,  load average: 5.55, 6.55, 6.88
Tasks: 294 total,   4 running, 290 sleeping,   0 stopped,   0 zombie
%Cpu(s): 18.2 us, 19.0 sy,  0.0 ni, 62.4 id,  0.1 wa,  0.0 hi,  0.3 si,  0.0 st
KiB Mem : 64299256 total, 15911892 free, 35912108 used, 12475256 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 24501976 avail Mem
```

시스템 시간, OS가 살아있는 시간, 접속 유저수
로드에버리지 : 1분, 5분, 15분의 cpu load 이동 평균
타스크 : total 294개, 실행(running) 4개, 290개 대기상태(sleeping)
CPU : 유저(us) 18.2%, 커널(sy) 19%, 우선순위설정(ni) 0%, 사용하지않음(id) 62.4%,
IO완료대기(wa) 0.1%, 하드웨어인터럽트(hi) 0%, 소프트웨어인터럽트(si) 0.3%, VM사용(st) 0%
메모리 : total 64G, free 16G, used 36G, buff/cash 12G (IO와 관련되어 사용하는 버퍼 메모리)

6. 다양한 부가 기능 제공 : top



```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 4160 opc       20   0 3060992 299840   6508 S   0.3 42.5   6:05.84 java
    1 root      20   0  216952   7396   5572 S   0.0  1.0   0:15.20 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.09 kthreadd
    3 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 rcu_par_gp
    6 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/0:0H-kb
    8 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S   0.0  0.0   0:06.96 ksoftirqd/0
```

PID : 프로세스 ID. 프로세스를 구분하기 위한 겹치지 않는 고유한 값

USER : 프로세스를 실행한 USER 이름 또는 효과를 받는 USER의 이름

PR : 커널에 의해서 실행되는 우선순위

NI : PR에 영향을 주는 nice라는 값

VIRT : 프로세스가 소비하고 있는 총 메모리(virtual memory)

RES : RAM에서 사용중인 메모리(실제 사용중인 물리 메모리)

SHR : 다른 프로세스와 공유 메모리

%CPU : 프로세스가 사용하는 CPU 사용률

%MEM : 프로세스가 사용하는 메모리 사용률

S : 프로세스의 현재 상태, R(running), S(sleeping), I(I/O 대기)

TIME+ : 프로세스가 사용한 토탈 CPU 시간

COMMAND : 해당 프로세스를 실행한 커맨드

Top 실행 후 명령어

[shift + m] : 메모리 사용량이 큰 순서로 정렬

[shift + p] : CPU 사용량이 큰 순서로 정렬

[q] : 종료

6. 다양한 부가 기능 제공 : free

```
              total        used        free      shared  buff/cache   available
Mem:       64299256    36798924    19108744     3493904     8391588    23598988
Swap:             0           0           0
```

total : 설치된 총 메모리 크기 / 설정된 스왑 총 크기

used : total에서 free, buff/cache를 뺀 사용중인 메모리. / 사용중인 스왑 크기

free : total에서 used와 buff/cahce를 뺀 실제 사용 가능한 여유 있는 메모리량 / 사용되지 않은 스왑 크기

Shared : tmpfs(메모리 파일 시스템), ramfs 등으로 사용되는 메모리. 여러 프로세스에서 사용할 수 있는 공유 메모리

Buff/cache : 버퍼와 캐시를 더한 사용중인 메모리

Available : swapping 없이 새로운 프로세스에서 할당 가능한 메모리의 예상 크기. (예전의 -/+ buffers/cache이 사라지고 새로 생긴 컬럼)

free 명령어 옵션

[-h] : 사람이 읽기 쉬운 단위로 출력한다.

[-b | -k | -m | -g] : 바이트, 킬로바이트, 메가바이트, 기가바이트 단위로 출력한다.

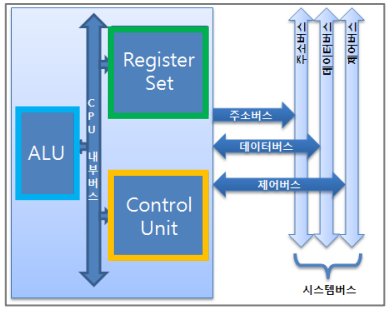
[-s '초'] : 지정한 초 만큼 딜레이를 두고 지속적으로 실행한다.

[-s '반복횟수'] : 지정한 반복횟수 만큼 free를 연속적으로 실행한다.

[-w] : 와이드 모드로 cache와 buffers를 따로 출력한다.

CPU - Memory(SRAM/DRAM) - Disk(HDD/SDD)

** RAM : Random Access Memory
** ROM : Read Only Memory

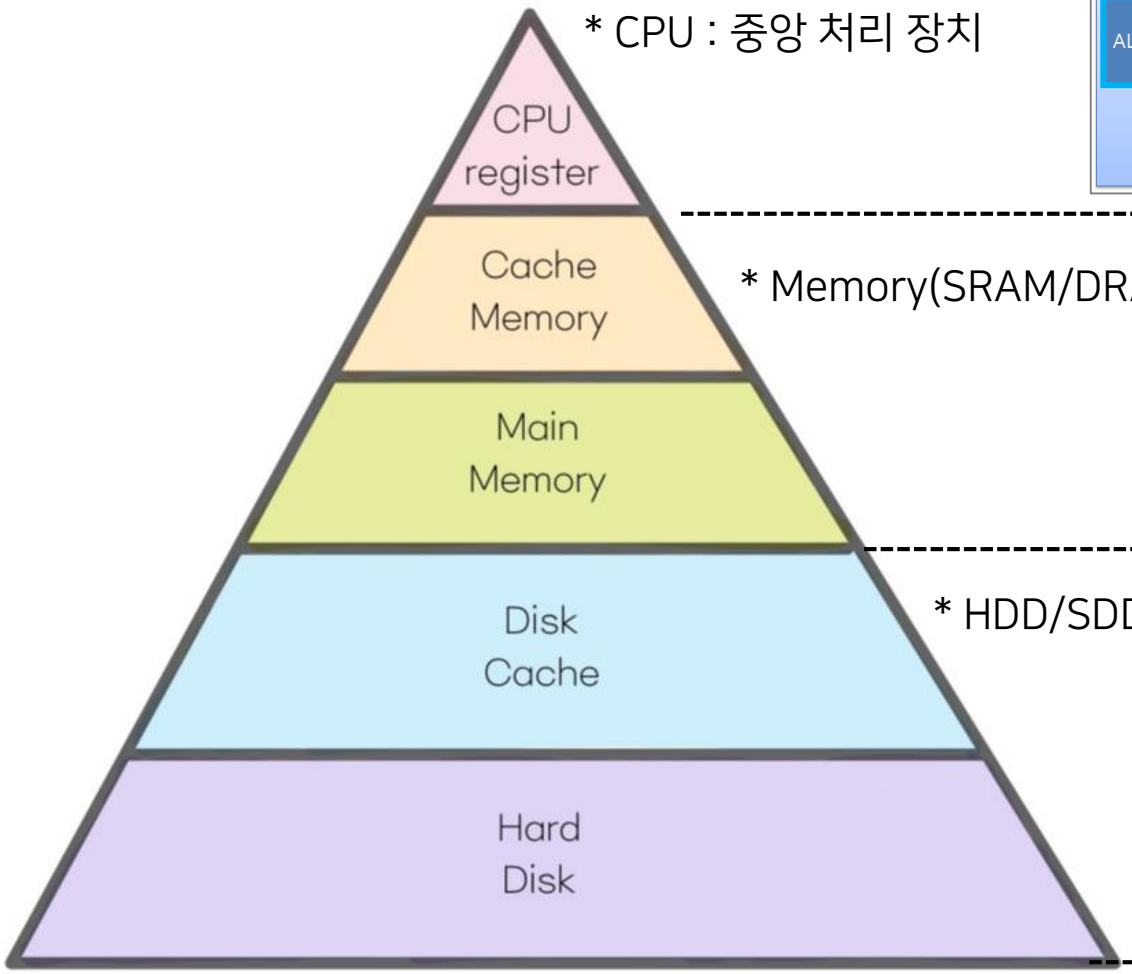


* 산술/논리 연산 장치(ALU, Arithmetic and logical unit)
  - 제어 장치의 명령에 따라 실제 연산을 수행하는 장치

* 제어 장치
  - 컴퓨터에 있는 모든 장치들의 동작을 지시하고 제어

* 레지스터
  - CPU 내부에서 처리할 명령이나 연산의 결과, 주소 등을 일시적으로 저장

* CPU : 중앙 처리 장치

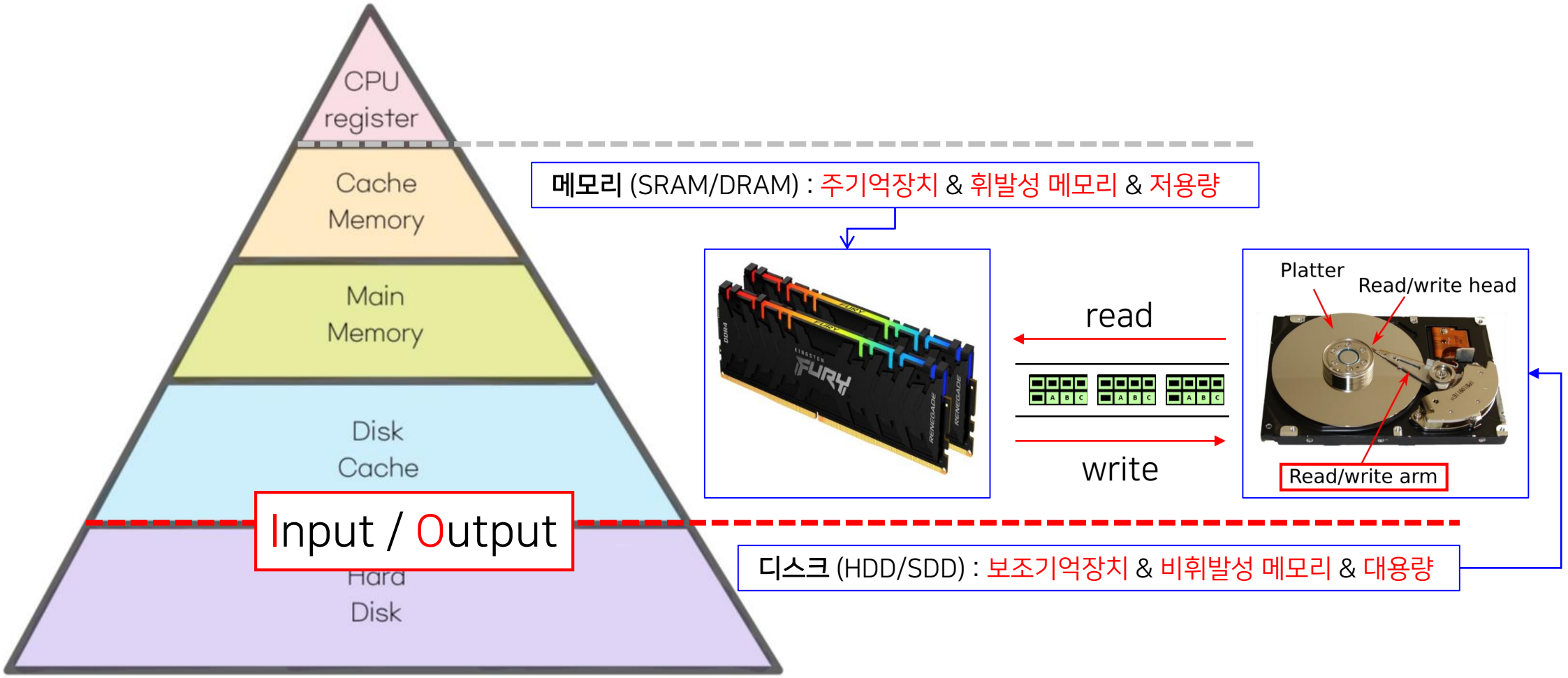


* Memory(SRAM/DRAM) : 주기억장치 & 휘발성 메모리

* HDD/SDD : 보조기억장치 & 비휘발성 메모리

△ 메모리의 계층 구조

CPU - Memory(SRAM/DRAM) - Disk(HDD/SDD)



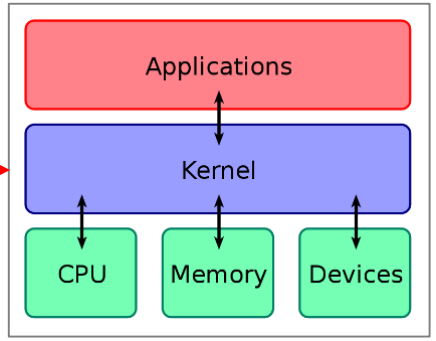
△ 메모리의 계층 구조

Buffer와 Cache

커널은 블록디바이스(디스크)로부터 데이터를 읽거나 사용자의 데이터를 디스크에 저장한다.

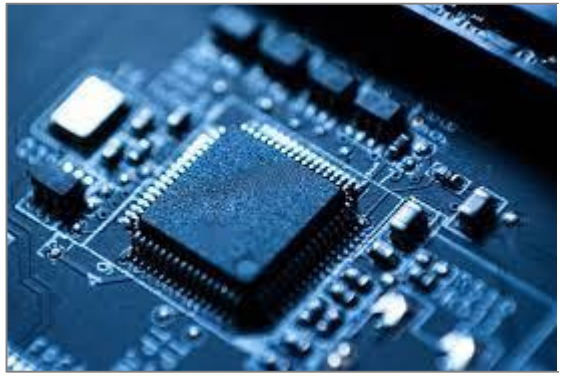
사용자(명령) -> 쉘(해석) -> 커널(명령 수행 후 결과 전송) -> 쉘(해석) -> 사용자(결과 확인)

I/O, Memory, CPU 제어

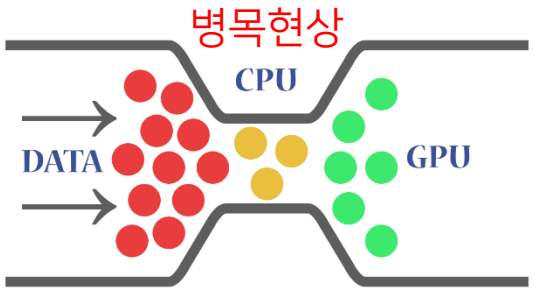


하지만 디스크(영구저장장치)는 다른 장치에 비해 매우 느리기 때문에 디스크에 요청을 기다리는 시간이 상당히 많이 소요되고, 이로 인해 시스템에 부하가 발생하게 된다.

때문에 리눅스는 항상 여유 메모리 공간을 Buffer와 Cache로 사용하려고 시도함.
메모리에 데이터를 저장해서 느린 디스크로의 접근(I/O, Input/Output) 을 최대한 줄여 성능을 향상.

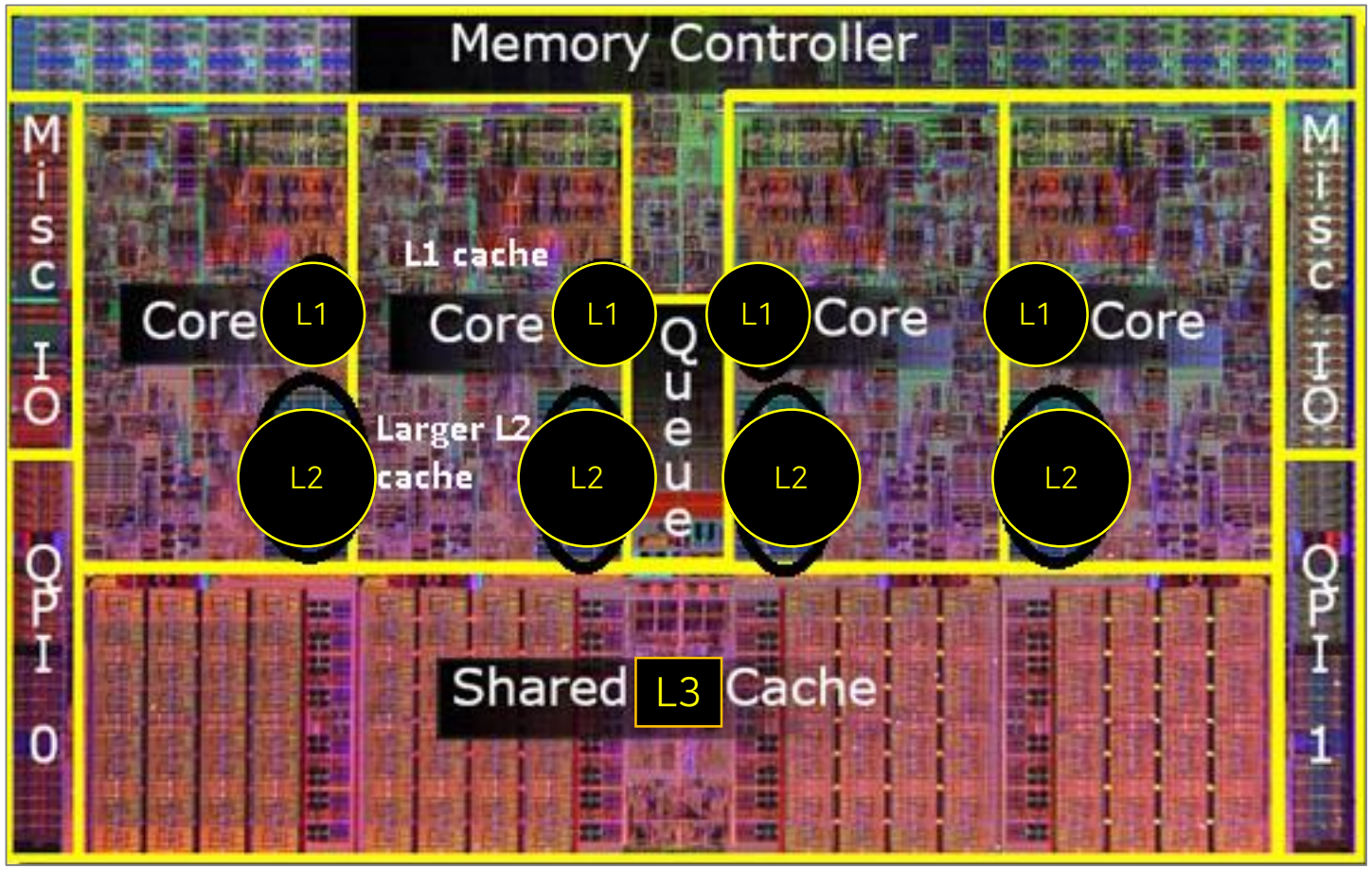


△ 메모리 (고속)

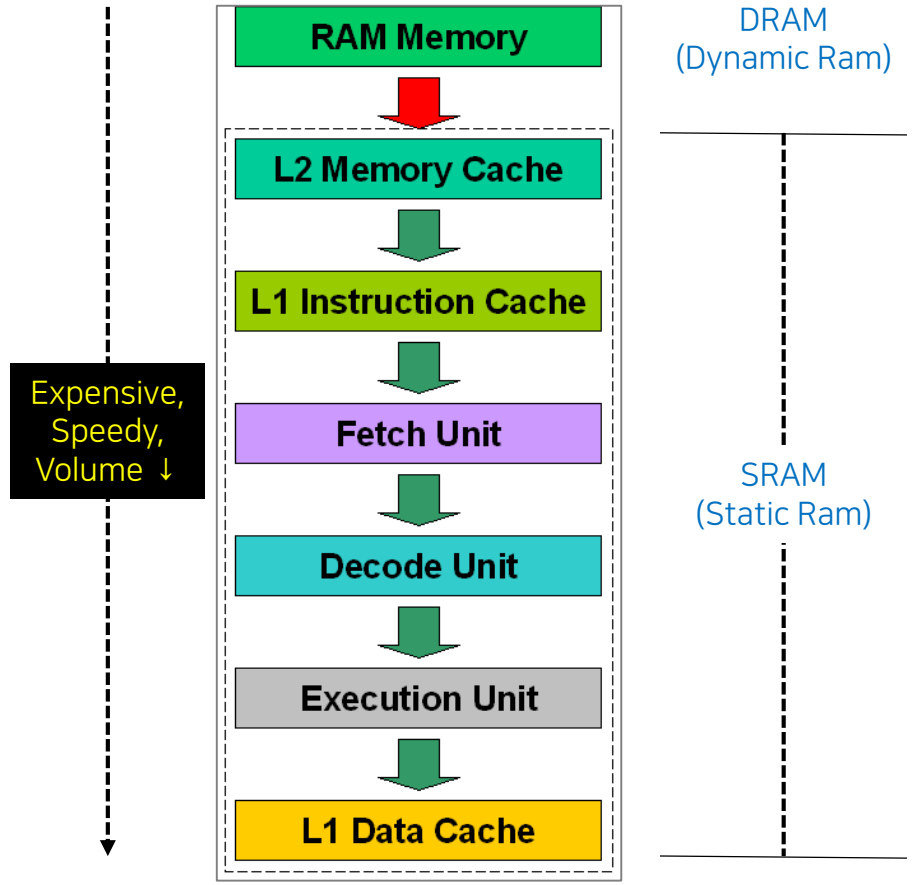


△ 디스크 (저속)

RAM (SRAM / DRAM)



△ 메모리 기판 (RAM 구성)



△ SRAM(L1, L2 Cache) & DRAM

Buffer

Buffer는 버퍼 캐시로 디바이스 블록에 대한 메타데이터들을 메모리에 캐싱한 크기.
블록 디바이스로부터 데이터를 읽어오기 위해 필요한 정보들을 메모리에 저장

데이터를 한 곳에서 다른 곳으로 전송하는 동안 일시적으로 그 데이터를 보관하는 메모리 영역
일시적으로 데이터를 보관하기 때문에 버퍼는 사용 후에 데이터를 폐기

Cache

Cache는 페이지 캐시와 slab으로 사용중인 메모리 크기.

- 페이지 캐시는 한 번 디스크로 읽어온 데이터를 메모리에 저장하여 같은 데이터를 다시 읽을 때 디스크로 요청하지 않고 메모리에서 바로 읽어올 수 있게 함.
- Slab은 커널에서 관리하는 커널 오브젝트를 저장하는 단위. 커널은 애플리케이션 할당 단위인 페이지보다 작은 slab 단위로 메모를 사용함. 하나의 메모리 페이지에 여러 slab이 존재 가능
- slab에 파일의 inode나 dentry 정보들을 캐싱할 수 있음.

Swap
디스크의 일정 부분을 메모리 공간 부족시 메모리처럼 사용하기 위해 설정해둔 공간
커널은 메모리가 부족한 상황에서 Buffer와 Cache로 할당된 것 중에 자주 사용되지 않은 데이터를 SWAP 공간으로 이동시킴 (SWAP-OUT). 그리고 SWAP으로 옮겨진 데이터를 프로세스가 읽기 위해 메모리로 데이터를 다시 가져옴. (SWAP-IN). 이는 디스크로부터 옮겨 졌던 데이터를 다시 메모리로 가져오기 때문에 latency가 길어져 성능 저하로 이어짐.

Cache와 Buffer

일시적인 메모리 사용 증가로 일정 SWAP을 사용하는 상황과 지속적인 메모리 부족으로 인해 SWAP이 커지는 경우를 지속적으로 모니터링 하여 메모리 증설 시점을 고려해야 함.

로드밸런싱이나 워크로드를 분산 배치 시키는 프로그램들은 /proc/meminfo를 참고하여 캐싱 영역을 비워 얼마만큼의 메모리 할당이 가능한지에 대한 정보도 확인. 수동으로 명령어를 이용해 캐싱을 비울 수도 있음. 하지만 캐싱을 적절히 사용하는 것이 시스템 성능 향상을 가져올 수 있고, 너무 자주 캐싱을 비우면 오히려 성능 저하로 이어질 수 있음.

그러니 시스템 특성에 맞춰서 캐싱에 대한 커널 파라미터를 적절히 설정 하는 것에 대한 고민이 필요.

Inactive (anon)는 익명 메모리 중 참조가 적은 메모리 크기.
Inactive (file)은

환경변수

환경 변수는 무엇에 쓰이는 걸까?
커맨드 라인에서 python 명령어를 실행하는 데, 이건 어떻게 실행되는 건지?

```
(base) [b2en@ip-192-168-14-30 etl]$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

△ 리눅스 CLI에서 python 실행

```
(base) [b2en@ip-192-168-14-30 etl]$ env
MAPR_HBASE_SERVER_OPTS=-Dhadoop.login=simple -Djava.security.auth.login.config=/opt/mapr/conf/mapr.login.conf  -Dzookeeper.sasl.clientconfig=Client_s
imple -Dzookeeper.saslprovider=com.mapr.security.simplesasl.SimpleSaslProvider  -Dmapr.library.flatclass
XDG_SESSION_ID=323062
HOSTNAME=ip-192-168-14-30.ap-northeast-2.compute.internal
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
MAPR_JMXDISABLE=false
MAPR_ECOSYSTEM_SERVER_LOGIN_OPTS=-Dhadoop.login=simple -Djava.security.auth.login.config=/opt/mapr/conf/mapr.login.conf  -Dzookeeper.sasl.clientconfi
g=Client_simple -Dzookeeper.saslprovider=com.mapr.security.simplesasl.SimpleSaslProvider  -Dmapr.library.flatclass
SSH_CLIENT=175.197.30.20 57880 22
CONDA_SHLVL=1
```

모든 환경변수 출력 (현재 화면보다 더 많습니다)

△ 리눅스 환경변수 보는 커맨드 (env / printenv)

```
(base) [b2en@ip-192-168-14-30 etl]$ env | grep python
CONDA_PYTHON_EXE=/home/user/b2en/.anaconda3/bin/python
```

△ 리눅스 환경변수 (grep 명령어를 통해 문자열 추출)

환경변수

기초 용어 먼저 숙지

* bash = bourne again shell (bourne이 UNIX에서 만든 shell 프로그램을 향상시킨 버전. 명령줄 편집 같은 기능 추구)

* rc = run commands

* ~ = /home/user/사용자계정 =〉 자신이 접속하고 있는 계정의 로컬 디렉토리 | (Example) "/home/user/b2en"

* /etc =〉 모든 계정의 전역(공용) 디렉토리

```
# /etc/profile
# 시스템 전역에 걸친 환경을 정의. 시작시 /etc/profile.d 하위에 존재하는 shell들이 함께 실행

# /etc/profile.d/*.sh 또는 *.csh
# 시스템 부팅과 동시에 로딩되는 shell들. 사용자 전체가 공통으로 할당받는 환경

# /etc/bashrc
# 시스템 전반에 걸친 함수와 alias. /etc/profile을 그대로 참조하여 사용하도록 되어 있는 하위 개념)

# ~/.bash_profile
# 사용자 환경에 맞게 로그인 이후에 로딩되어 할당받는 환경

# ~/.bashrc
# 사용자 홈 디렉토리에 위치한 .bashrc는 로그인하거나, 현재 터미널에서 다시 .bashrc를 로딩하면 즉각 환경이 반영
```

Linux 알쓰신잡

1. 리눅스의 파일 또는 디렉토리 이름에 '.' 으로 시작하는 파일들은 무엇인가요?
- Answer : 숨겨진 파일 (히든 파일) 입니다. 리눅스 또는 유닉스에서는 점 하나를 찍음으로써 파일을 숨길 수 있다.

2. 해당 숨겨진 해당 폴더들은 ll 또는 ls 명령어로는 보이지 않는 데, 어떻게 볼 수 있나요? 〉
- Answer : ll 또는 ls 명령어 뒤에 옵션을 '-a' 추가적으로 붙여 주셔야 합니다. 'ls - al' 을 기억하시면 유용합니다.

3. bin 폴더는 무엇인가요?
- Answer : bin 폴더는 binary (실행파일이 담겨져 있는 폴더) 입니다. 항상 그렇지는 않지만,
  일반적으로 프로그램들의 실행파일은 bin에 있다고 보셔도 될 것 같습니다.

4. 만약, bin 폴더에서 순서대로 파일을 읽는 데, 파일마다 버전이 다른 경우 특정 파일을 지정해서 읽는 방법은 무엇인가요?
- /etc/alternatives/* 에 가면 많은 파일들이 있습니다. 해당 폴더에서 심볼릭 링크를 생성하여, 특정 파일을 읽도록 명시적으로 지정할 수 있습니다.

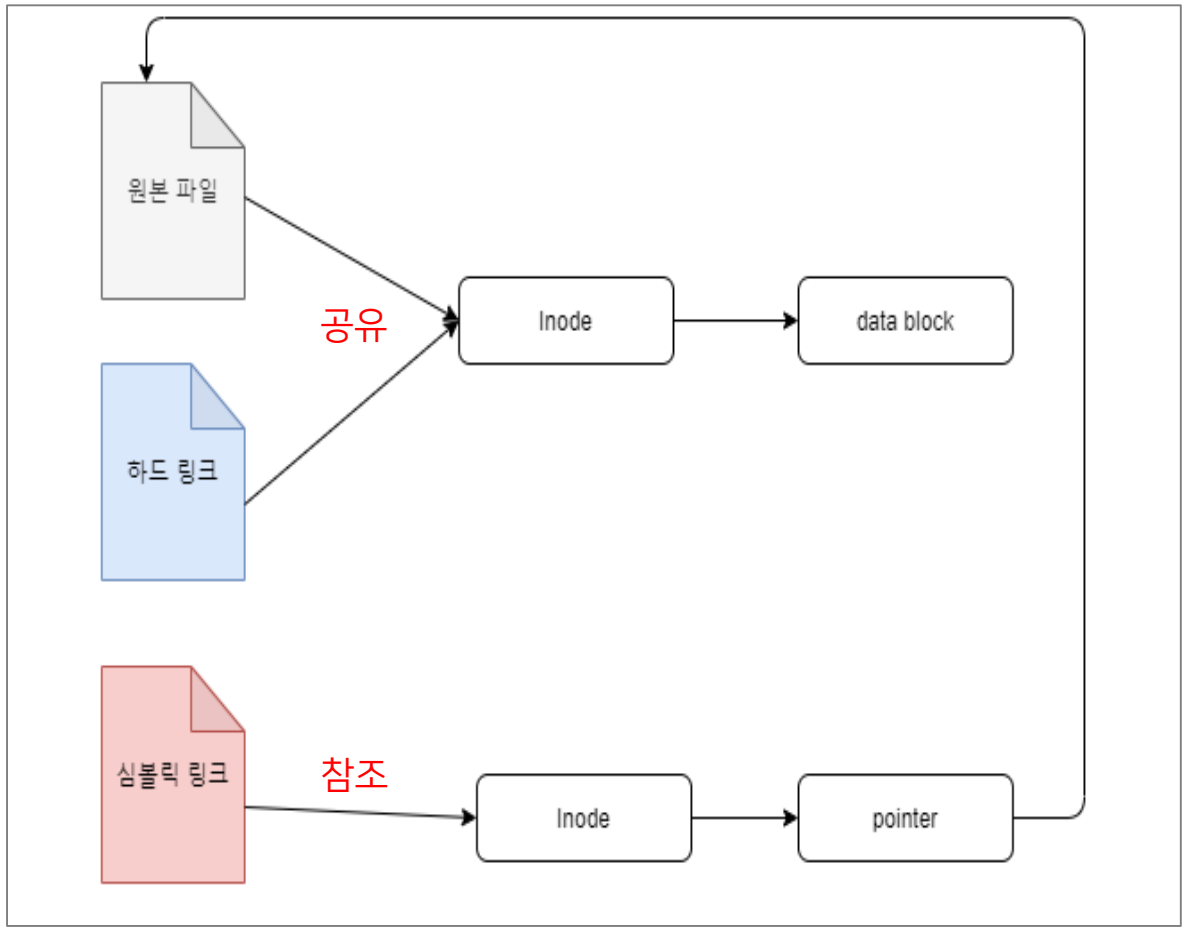
Linux 알쓰신잡

1. 리눅스의 파일 또는 디렉토리 이름에 '.' 으로 시작하는 파일들은 무엇인가요?
- Answer : 숨겨진 파일 (히든 파일) 입니다. 리눅스 또는 유닉스에서는 점 하나를 찍음으로써 파일을 숨길 수 있다.

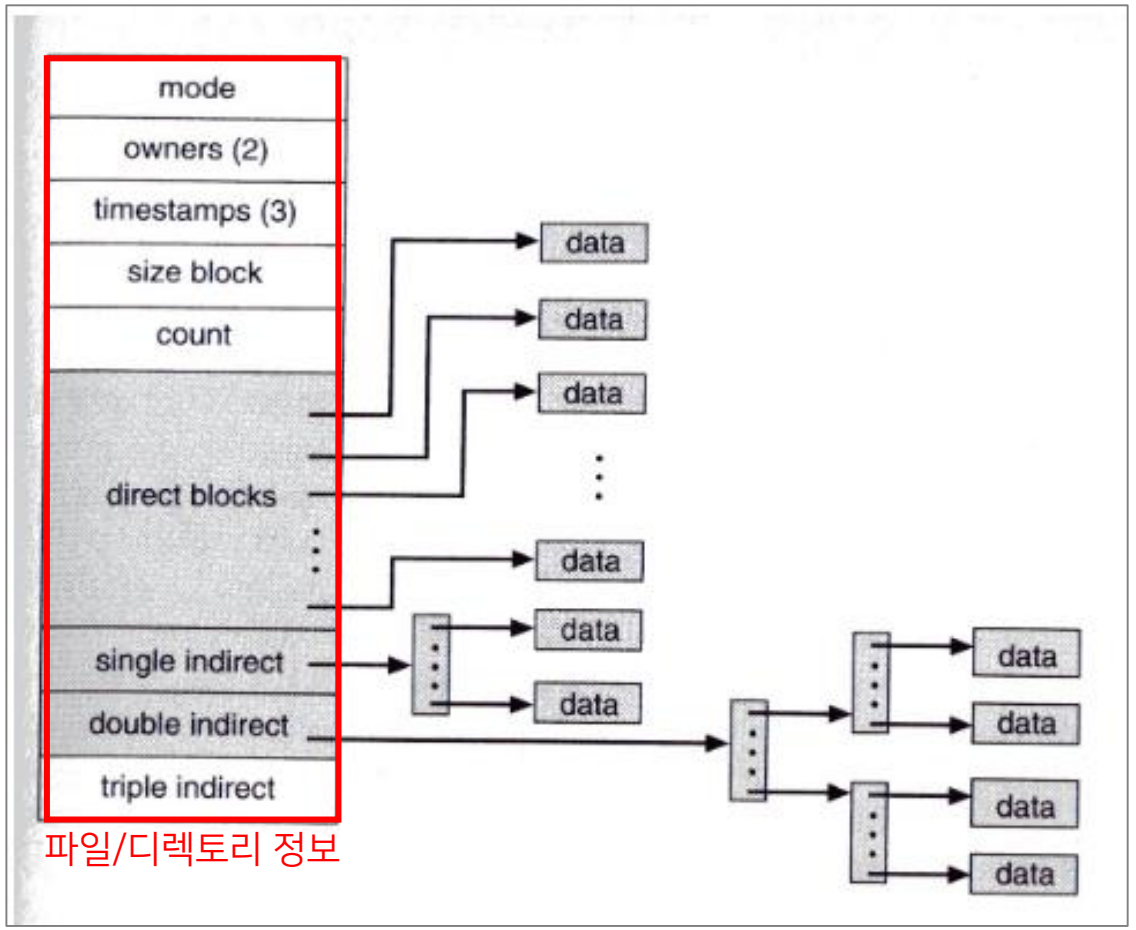
2. 해당 숨겨진 해당 폴더들은 ll 또는 ls 명령어로는 보이지 않는 데, 어떻게 볼 수 있나요? 〉
- Answer : ll 또는 ls 명령어 뒤에 옵션을 '-a' 추가적으로 붙여 주셔야 합니다. 'ls - al' 을 기억하시면 유용합니다.

3. bin 폴더는 무엇인가요?
- Answer : bin 폴더는 binary (실행파일이 담겨져 있는 폴더) 입니다. 항상 그렇지는 않지만,
  일반적으로 프로그램들의 실행파일은 bin에 있다고 보셔도 될 것 같습니다.

4. 만약, bin 폴더에서 순서대로 파일을 읽는 데, 파일마다 버전이 다른 경우 특정 파일을 지정해서 읽는 방법은 무엇인가요?
- /etc/alternatives/* 에 가면 많은 파일들이 있습니다. 해당 폴더에서 심볼릭 링크를 생성하여, 특정 파일을 읽도록 명시적으로 지정할 수 있습니다.



△ 심볼릭 링크와 하드 링크 연결 방식



△ inode의 구조

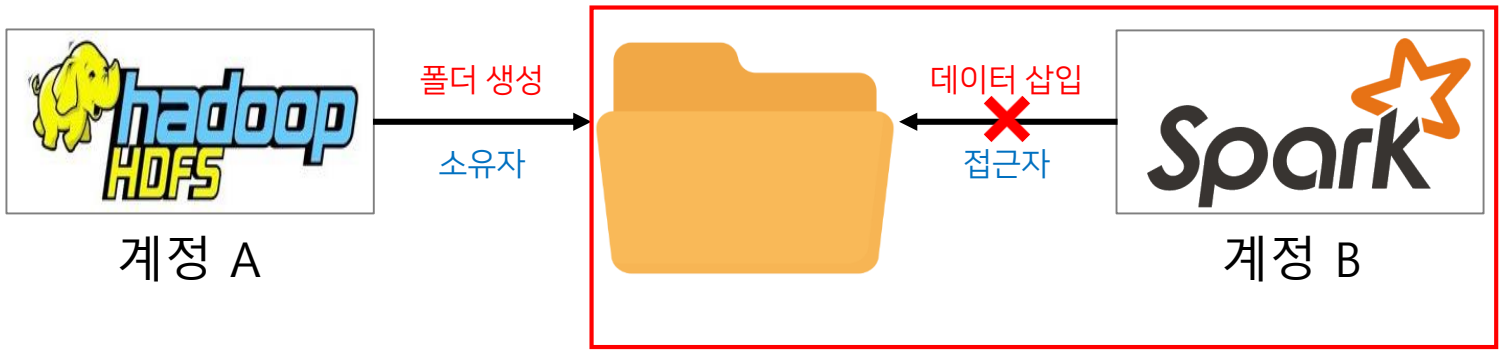
허가권(Permission)과 소유권(Ownership)

모든 파일 또는 디렉토리는 권한이 있다. 파일 권한은 왜 중요할까?

```
(base) [b2en@ip-192-168-14-30 /]$ ll
허가권(permission)    소유권(ownership)
lrwxrwxrwx.    1 root root       7 Feb 29  2020 bin -> usr/bin
dr-xr-xr-x.    5 root root    4096 Jun 21  2021 boot
lrwxrwxrwx     1 root root      31 Jun 21  2021 data -> /mapr/mapr.daegu.go.kr/ETL/data
drwxr-xr-x    16 root root    2720 Mar 30 16:08 dev
drwxr-xr-x.   90 root root    8192 Mar 30 13:20 etc
drwxr-xr-x.    5 root root      45 Jul 13  2021 home
```

△ 'll' 커맨드로 리스트를 출력한 화면

* 스마트시티 구축사업에서 실제 문제가 되던 상황



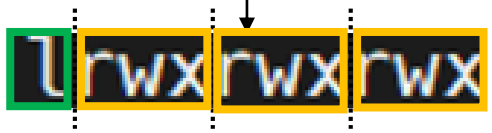
△ 폴더에 대한 접근 권한이 없어 진입이 안되어, 데이터를 넣기 못함

허가권(Permission)과 소유권(Ownership)

① 퍼미션 정보 (rwx의 의미)

허가권(permission)

```
lrwxrwxrwx    1 root root     31 Jun 21  2021 data -> /mapr/mapr.daegu.go.kr/ETL/data
```



[_ | (rwx) | (rwx) | (rwx)] 첫 글자를 제외하고, 뒤에 9자리는 3자리 씩 반복

* 첫 글자

약자	철자	의미
_		일반 파일
d	directory	디렉토리
l	symbolic link	심볼릭 링크
더 있지만, 기본적으로 이 정도만 이해하고, 다음에 모르는 게 나왔을 때 스스로 학습 필요!		

* 뒤 9 글자 (rwx)

약자	철자	의미
_		권한 없음
r	read	읽기
w	wrtie	쓰기
x	excute	실행

△ 중요한 건 이쪽!

허가권(Permission)

디렉토리와 파일의 권한에 따라, rwx가 무엇을 의미하는지

* 디렉토리

약자	철자	의미	개념 설명
r	read	읽기	디렉토리 내부 파일 무엇이 있는지 보기
w	wrtie	쓰기	디렉토리 내부 파일 생성, 삭제, 이동
x	excute	실행	디렉토리 실행기능 내부 이동 수행

Q. 디렉토리에 x권한이 없는데, r권한이 있다면 내부 파일이 무엇이 있는 지 알 수 있을까?
A. 없다. 파일 리스트를 보는 것 또한, 디렉토리 내부로 들어가서 리스트를 확인하는 작업이라,
디렉토리 안으로 들어가는 실행권한(x)가 없으면 불가.

* 파일

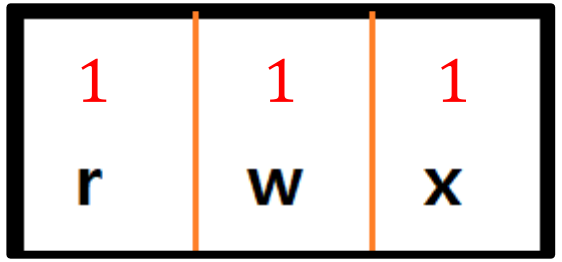
약자	철자	의미	개념 설명
r	read	읽기	파일 내용(code) 보기
w	wrtie	쓰기	파일 내용(code) 수정
x	excute	실행	파일 실행기능(프로그램 기능) 수행

허가권(Permission)

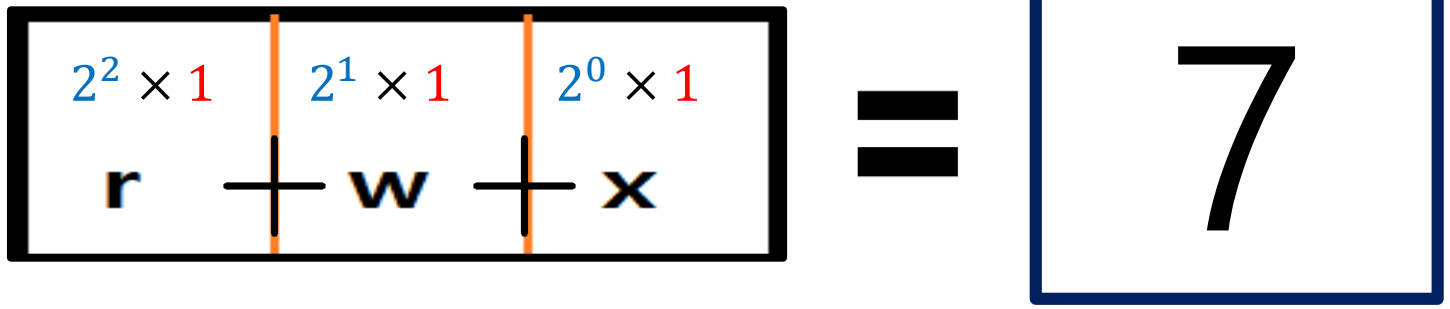
① 퍼미션 정보 (rwx의 의미)

rwx 를 숫자로 부르는 또 다른 방식 => 2진수를 이용해서 숫자로 부르기
해당 (읽기 | 쓰기 | 실행) 권한의 각 자리를 비트의 有(1) / 無(0)로 보자.

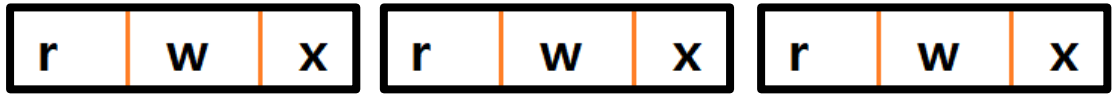
해당 권한이 모두 있다면, 아래와 같이 표시



만약, 이진법으로 자리 수 까지 고려해서 표시한다면?



즉, 퍼미션 권한 rwx는 각 자리의 권한 유무에 따라 0 ~ 7(모든 권한 없음 ~ 있음)으로 표시 가능

그렇다면 

는, 뭐라고 부르면 될까?

그렇다. 777 (‘칠칠칠’로 호칭하면 된다. 맨 앞 한 글자인 파일 타입은 일단 제외하는 것으로 함) 이다.

허가권(Permission)

② 퍼미션 위치 (rwx가 나눠지는 각 자리 위치 의미)

유저 소유자가 가질 수 있는 권한. 근데 소유자라는 말은 무엇일까?
소유자란, 파일에 대한 주체 (특정한 설정이 없다면, 보통 파일을 만들 때 접속한 계정으로 소유권이 부여)

소유권(Ownership)

그러면 만들어진 파일 또는 디렉토리의 소유권자가 누구인지 어떻게 볼 수 있을까?

```
lrwxrwxrwx    1 root root     31 Jun 21  2021 data -> /mapr/mapr.daegu.go.kr/ETL/data
```

유저 소유권

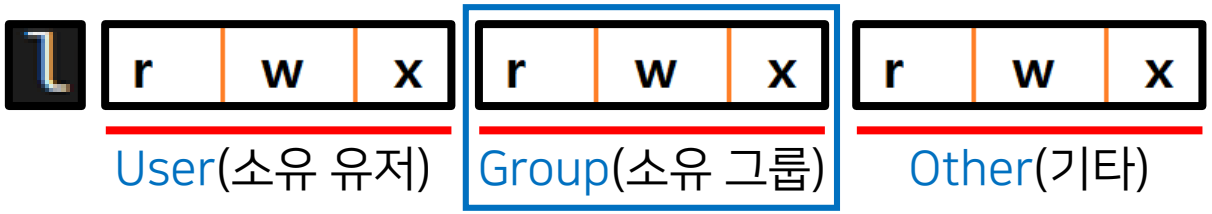
△ 파일 또는 디렉토리의 소유권자 확인

리눅스 알쓰신잡 [미리보기]

Q. 기본적으로 root라는 계정이 있던 데 이 계정은 무엇인가요?
A : 리눅스나 macOS와 같은 유닉스 계열의 운영체제에서 모든 권한을 가지고 있는 최고 관리자가 사용하는 ID
즉, super user 또는 관리자 라고 볼 수 있다.

허가권(Permission)

② 퍼미션 위치 (rwx가 나눠지는 각 자리 위치 의미)



그룹이 가질 수 있는 권한. 근데 그룹은 어떻게 알 수 있을 까?
소유자란, 파일에 대한 주체 (특정한 설정이 없다면, 보통 파일을 만들 때 접속한 계정으로 소유권이 부여)

소유권 (ownership)

```
lrwxrwxrwx    1 root root     31 Jun 21  2021 data -> /mapr/mapr.daegu.go.kr/ETL/data
```

그룹 소유권

그러면 만들어진 파일 또는 디렉토리의 소유권자가 누구인지 어떻게 볼 수 있을까?

/etc/group 를 확인한다.

사용자 정보 관리(/etc/passwd)

0 1 2 3 4 5 6

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
centos:x:1000:1000:Cloud User:/home/centos:/bin/bash
ntp:x:38:38::/etc/ntp:/sbin/nologin
hpedf:x:5000:5000::/home/hpedf:/bin/bash
b2en:x:6000:5000::/home/user/b2en:/bin/bash
```

Index	Field	Description		예시
0	사용자 계정			root
1	패스워드	사용자 계정의 패스워드. X로 암호화되어 표시. 패스워드는 /etc/shadow에 저장		x
2	UID (사용자 식별자)	사용자의 유저 ID.	[UID 0 : 관리자 계정(root)] : 로그인 가능 [UID 1 ~ 1000 / 65534 : 시스템 유저] : 로그인 불가능 [UID 1000 ~ 65535 : 일반 유저] : 로그인 가능	0
3	GID (그룹 식별자)	사용자의 그룹 ID.		0
4	comment	사용자와 관련한 기타 정보. 일반적으로 사용자 이름을 나타냄		root
5	홈 디렉토리	사용자의 홈 디렉토리.	[관리자 계정(root)] : /root [일반 사용자] : /home/하위 계정	/root
6	로그인 쉘	사용자가 로그인 시 사용할 쉘. 보통 사용자의 쉘은 성능이 우수한 bash쉘을 사용		

그룹 정보 관리(/etc/group)

0 1 2 3

```
root:x:0:hpedf
bin:x:1:
daemon:x:2:
sys:x:3:
```

Index	Field	Description	예시
0	그룹 이름		root
1	패스워드	사용자 계정의 패스워드. X로 암호화되어 표시. 패스워드는 /etc/shadow에 저장	x
2	GID (그룹 식별자)	사용자의 그룹 ID.	0
4	그룹에 소속된 사용자 계정	그룹에 소속된 사용자 계정 리스트.	hpedf

/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:998:995::/var/lib/chrony:/sbin/nologin
centos:x:1000:1000:Cloud User:/home/centos:/bin/bash
ntp:x:38:38::/etc/ntp:/sbin/nologin
hpedf:x:5000:5000::/home/hpedf:/bin/bash
b2en:x:6000:5000::/home/user/b2en:/bin/bash
ccmedia:x:7000:7000::/home/user/ccmedia:/bin/bash
datahub:x:8000:8000::/home/user/datahub:/bin/bash
clamupdate:x:997:994:Clamav database update user:/var/lib/clamav:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
clamscan:x:996:992:Clamav scanner user:/:/sbin/nologin
sflow:x:1001:1001::/home/user/sflow:/bin/bash
postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
```

```
(base) [b2en@ip-192-168-14-30 ~]$ id
uid=6000(b2en) gid=5000(hpedf) groups=5000(hpedf),6000(b2en)
```

/etc/group

```
root:x:0:hpedf
bin:x:1:
daemon:x:2:
sys:x:3:
adm:x:4:centos
tty:x:5:
disk:x:6:
lp:x:7:
mem:x:8:
kmem:x:9:
wheel:x:10:centos
cdrom:x:11:
mail:x:12:postfix
man:x:15:
dialout:x:18:
floppy:x:19:
games:x:20:
tape:x:33:
video:x:39:
ftp:x:50:
lock:x:54:
audio:x:63:
nobody:x:99:
users:x:100:
utmp:x:22:
utempter:x:35:
input:x:999:
systemd-journal:x:190:centos
systemd-network:x:192:
dbus:x:81:
polkitd:x:998:
rpc:x:32:
ssh_keys:x:997:
cgred:x:996:
rpcuser:x:29:
nfsnobody:x:65534:
sshd:x:74:
postdrop:x:90:
postfix:x:89:
chrony:x:995:
centos:x:1000:
ntp:x:38:
shadow:x:5001:hpedf
hpedf:x:5000:hpedf,b2en,sflow,ccmedia
b2en:x:6000:b2en
ccmedia:x:7000:
datahub:x:8000:
clamupdate:x:994:
virusgroup:x:993:clamupdate,clamscan
tss:x:59:
clamscan:x:992:
```

네트워크 기초

① IP주소, subnet mask, broadcast 주소, MAC 주소

```
(base) [b2en@ip-192-168-14-30 ~]$ ifconfig
```

```
ens5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 192.168.14.30  netmask 255.255.255.0  broadcast 192.168.14.255
        inet6 fe80::8b6:16ff:fe72:70a4  prefixlen 64  scopeid 0x20<link>
        ether 0a:b6:16:72:70:a4  txqueuelen 1000  (Ethernet)
        RX packets 12203498377  bytes 20035568267690 (18.2 TiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13445481943  bytes 32219501181639 (29.3 TiB)
        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 1000  (Local Loopback)
        RX packets 201193738929  bytes 37690320743165 (34.2 TiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 201193738929  bytes 37690320743165 (34.2 TiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

IP주소 (IPv4)

네트워크 기초

① IP주소, subnet mask, broadcast 주소, MAC 주소

- IP (Internet Protocol address) :

** IP 주소는 OSI 7계층 중 3계층(네트워크 계층)에서 생성 및 사용

OSI 7 Layer Model

Layer 7	응용 계층 (Application Layer)
Layer 6	표현 계층 (Presentation Layer)
Layer 5	세션 계층 (Session Layer)
Layer 4	전송 계층 (Transport Layer)
Layer 3	네트워크 계층 (Network Layer)
Layer 2	데이터 링크 계층 (Data Link Layer)
Layer 1	물리 계층 (Physical Layer)

TCP/IP 4 Layer Model

Layer 4	응용 계층 (Application Layer)
Layer 3	전송 계층 (Transport Layer)
Layer 2	인터넷 계층 (Internet Layer)
Layer 1	네트워크 엑세스 (Network Access Layer)