<RO:BIT Linux & git 보고서>

18기 지능팀 인턴 선현빈

▶ Linux

1. Linux의 정의 및 특징

1) Linux

Linux는 한 마디로 리눅스 커널 기반의 '오픈소스 OS(운영 체제)'이다. 1991년 9월 경에 최초 버전이 출시되어, 계속해서 발전해오면서 현재에는 다양한 언어의 사용이 가능하다. 보통 Linux 기반의 OS는 Linux 커널과 GNU 툴과 같은 공통적인 구성 요소가 있다.

[Linux의 로고와 마스코트]

Linux 배포판으로 칭할 수 있는 운영 체제로 우분투, 레드햇, 데비안, 페도라 등이 존재하며, 안드로이드도 리눅스 기반 운영체제이다. 이중에서도 우분투는 개인 데스크탑용도로, 데비안은 안정성, 보안성이 높아 서버 및 네트워크 용도로 사용되며, 페도라는 커뮤니티 기반으로 최신 기술 적용을 위해 사용된다. 이외에 다른 배포판들도 각각의 용도 차이가 존재한다.

[Linux 배포판 운영체제]

2) Linux : Windows 비교

	Linux	Windows
소스 형태	오픈 소스	Closed 소스
기본 UI	다양함	윈도우 셸
안정성/보안성	비교적 高	비교적 低
운영 및 관리	콘솔에서 명령 입력	GUI 기반
Application	오픈소스 프로그램 多	호환 프로그램 多

+) OS는 응용 프로그램(어플리케이션)과 하드웨어 사이에서 소프트웨어 작업을 수행하고 물리적인 리소스를 연결하는 역할을 수행한다.

+) GNU는 운영 체제의 하나이면서, 컴퓨터 '자유' 소프트웨어의 모음이라고 할 수 있다.

2. Linux(ubuntu) 설정

설치 이후, 설정은 다음과 같다.

1. 패키지 업데이트 및 업그레이드
   - $ sudo apt-get update
   - $ sudo apt-get upgrade
2. 터미네이터 설치
   - $ sudo apt-get install terminator
3. 터미널 편집기 neo-vim 설치
   - $ sudo apt-get install neovim
4. Windows와 듀얼 부팅할 경우, 시간대를 재설정한다.
5. TLP(노트북용 전원 관리 패키지) 설치(선택)
   - $ sudo add-apt-repository ppa:linrunner/tlp
   - $ sudo apt-get update
   - $ sudo apt-get install tlp

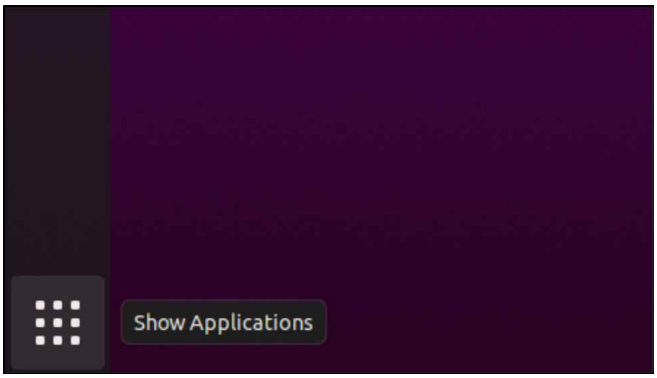
- $ sudo tlp start(tlp 실행, 재부팅 시에도 유지)

- $ tlp-stat -s(현재 전원 상태 확인)

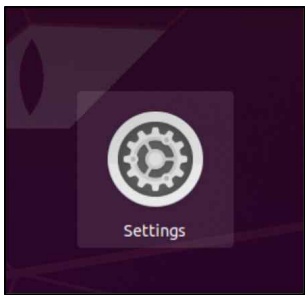
6. 한글 설정(application -> Settings에서 설정 가능)

▶ https://shanepark.tistory.com/231(참고)

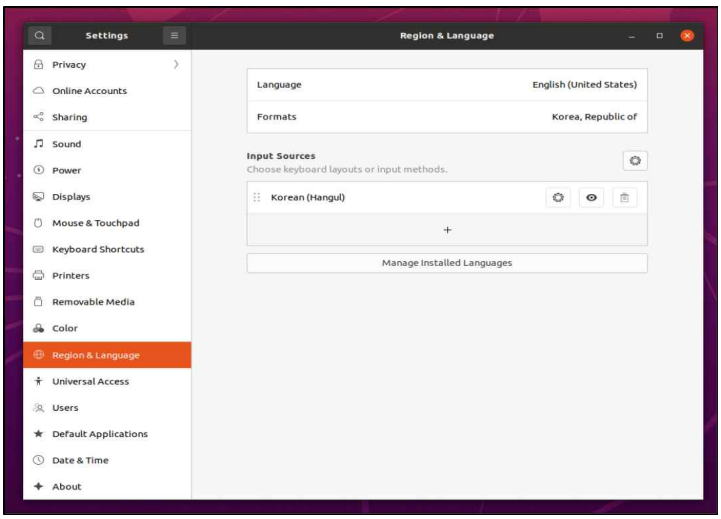
a. 바탕화면의 좌측 하단 Show Applications 클릭



b, Settings 실행

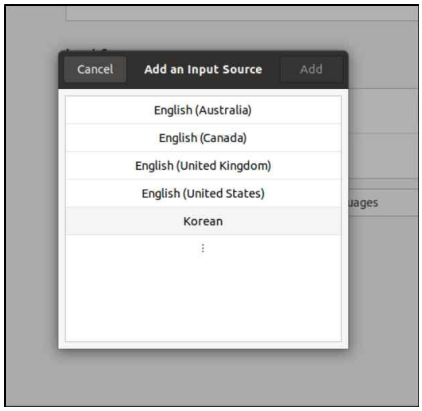


c. Region & Language 카테고리 클릭

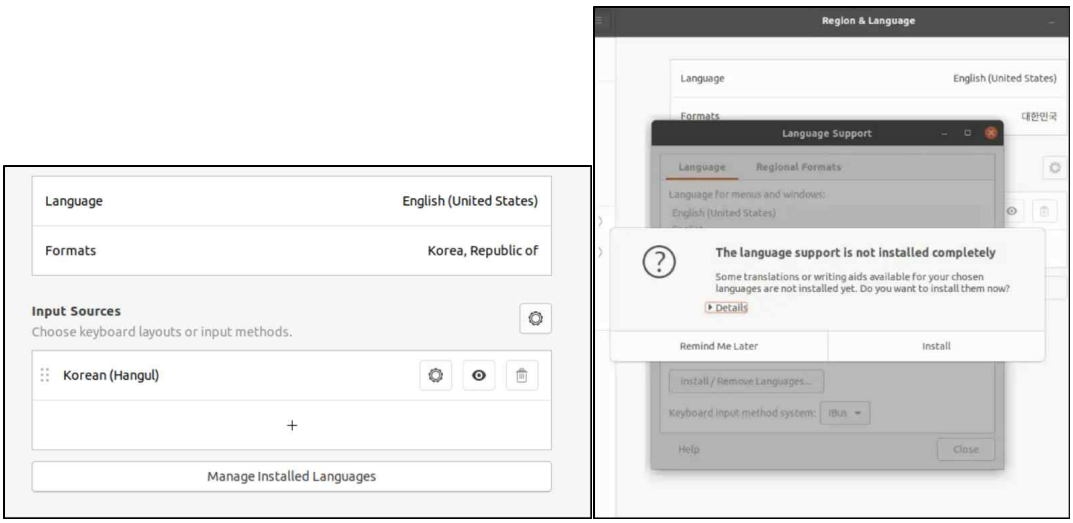


[Input Sources의 '+'를 클릭한다]

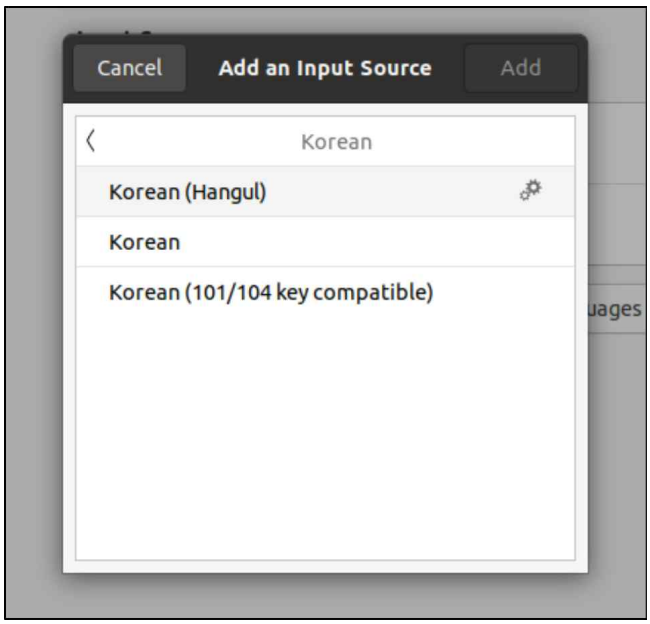
d. Korean - Korean 선택 후, 우측 상단의 Add 클릭



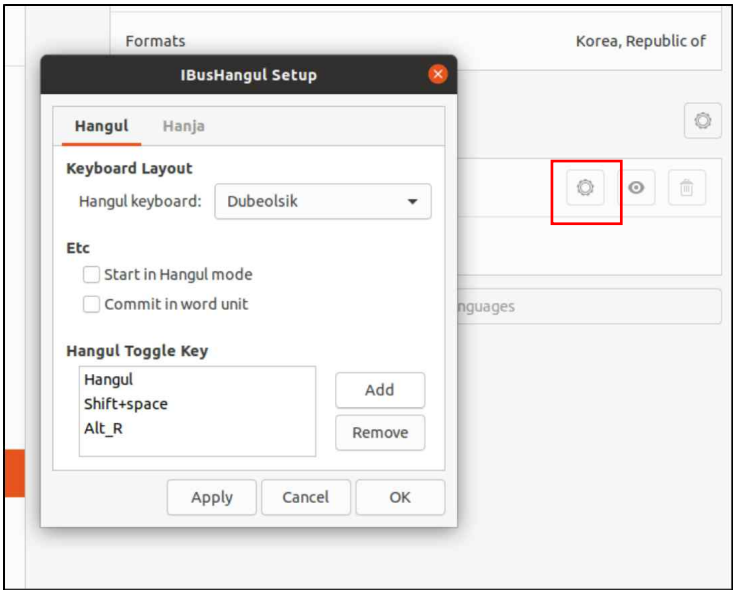
e. Manage Installed Languages 클릭 후, Install 선택



f. 재시작 후, a~c 반복 후, d단계에서 Korean-Korean(Hangul) 선택 후, Add



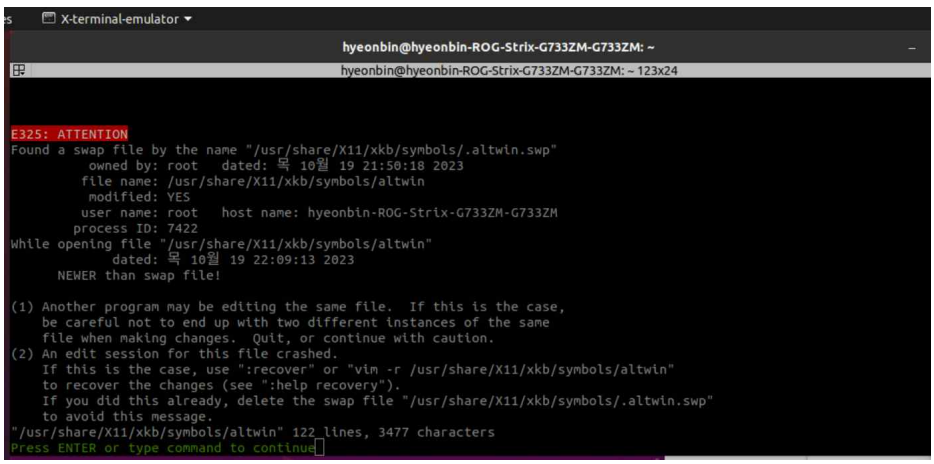
g. Red block 부분 클릭 후, 하단 화면에서 Add 클릭 - 원하는 한/영 단축키 입력



h. 단축키 바인딩 지정을 위한 설정(선택)

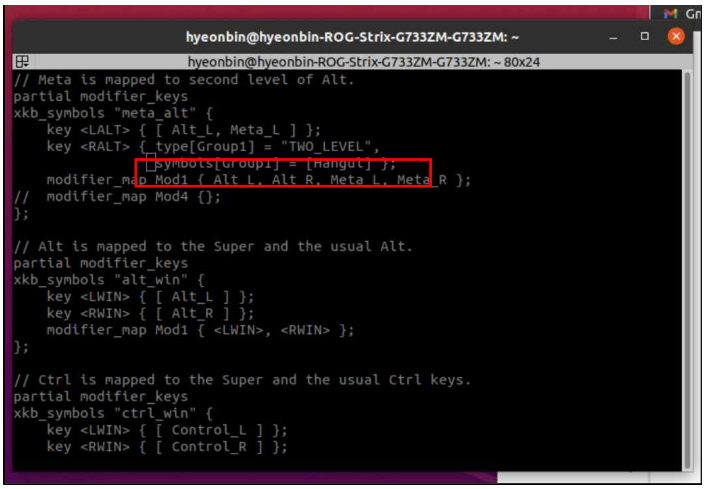
[altwin 파일 오픈]

▼



[press the Enter]

▼



[Red block 안과 같이 내용 수정 후. 저장 - 재시작]

3. Linux 명령어 정리

1) 기본 명령어(유닉스 계열)

sudo	유닉스 및 유닉스 계열 OS에서 다른 사용자 권한으로 프로그램을 구동할 수 있도록 하는 프로그램. substitute user do의 약자이며, 터미널 각 명령줄의 대부분에 기본적으로 사용한다.
alias	명령어 축소를 위해 사용. ex) alias copy = 'cp'
cat	파일들을 연결하고 표시하기 위해 사용. 'cat'enate(연결하다)에서 유래
cd(chdir)	change directory의 약어. 현재 작업 디렉터리의 위치를 변경한다.
ls	list segments의 약어. 기본적으로 현재 작업 중인 디렉터리의 전체 폴더 및 파일 리스트를 출력한다.
mkdir	make directory의 약어. 디렉터리를 생성한다.
pwd	Print work directory의 약어. 현재 작업 중인 디렉터리를 출력한다.
그외	https://en.wikipedia.org/wiki/List_of_Unix_commands

2) 패키지 관련

apt와 apt-get은 큰 차이는 없으며, apt가 apt-get을 조금 더 사용자 친화적으로 개선한 명령어이다. apt는 진행 상황을 보여주는 기능이 기본적으로 활성화되어 있는 명령어이며, 여러 작업을 하나의 명령어로 결합할 수 있다.

apt	기능적 측면에서 더 뛰어나며, 추가 정보를 출력
apt-get	더 안정적이고 호환성이 높기에, 스크립트 작성 시에 사용

dpkg	.deb로 구성된 패키지 파일 설치 및 삭제에 사용

[apt 관련 명령어(형태 : apt command)]

install	지정한 패키지를 설치 ex) apt-get install 패키지명
remove	지정한 패키지를 삭제(파일은 제거되지 않는다) ex) apt-get remove 패키지명
purge	지정한 패키지와 패키지 파일까지 삭제 ex) apt-get purge 패키지명
update	저장소 정보를 갱신 ex) apt-get update
upgrade	업그레이드 가능한 모든 패키지를 업그레이드 ex) apt-get upgrade
search	지정한 패키지를 검색 ex) apt search 검색어
show	지정한 패키지 상세 정보 확인 ex) apt show 패키지명
list --installed	설치된 패키지 명단을 출력 (list만 쓰면, 설치 가능한 모든 패키지가 검색됨)
install --only-upgrade	특정 패키지만 업그레이드

[dpkg 관련 명령어(형태 : dpkg 옵션)]

-I	해당 .deb 파일 정보 확인 ex) dpkg -I .deb파일
-i	해당 파일 설치/업그레이드 ex) sudo dpkg -i .deb파일
-L	해당 패키지로부터 설치된 모든 파일 목록 확인 ex) dpkg -L 패키지명
-l	설치된 패키지 목록 확인 ex) dpkg -l
-S	해당 파일명/경로가 포함된 패키지 검색 ex) dpkg -S 파일명
-s	해당 패키지에 대한 정보 확인 ex) dpkg -s 패키지명

-C	해당 .deb 파일이 설치한 파일 목록 확인 ex) dpkg -C .deb파일
-r	해당 패키지 삭제 ex) sudo dpkg -r 패키지명
-P	해당 패키지 및 패키지의 설정파일까지 모두 삭제 ex) sudo dpkg -P 패키지명
-x	파일에 포함된 파일들을 지정한 디렉터리에 압축 해제(해당 디렉터리는 초기화된다) ex) sudo dpkg -x .deb파일 디렉터리

3) 네트워킹 관련

참고로 ifconfig과 같은 명령어의 'if'는 interface를 의미한다.

ifconfig	네트워크 인터페이스에 관한 여러 정보 제공(구성 데이터, 패킷 등)
ifup/ifdown	네트워크 인터페이스 연결을 불러오거나 종료
ifquery	네트워크 인터페이스 리스트를 요약해 출력
netstat	라우팅 및 네트워크 연결 정보 제공
netstat -a	모든 네트워크 연결 정보 제공
host	원격 시스템의 IP 주소 조회
dig	통신 중인 네임 서버, 쿼리 응답 소요 시간 등 원격 시스템 연결 정보 제공
ping	연결 상태 확인
route	라우팅 테이블 정보 제공
ip	인터페이스, IP 주소, 라우팅 정보 설정 및 출력

+) 쿼리 : 데이터베이스에 정보를 요청하는 일

4) 편집 관련

입력모드에서 명령모드로 바꿀 때에는 ESC 키를 누르면 된다.

입력모드		명령모드	
a	커서 우측에 입력	:q	종료
A	행의 마지막에 입력	:q!	강제 종료(저장X)
i	커서 좌측에 입력	:w	저장

I	행의 처음에 입력	:wq	저장 + 종료
o	커서 아래, 빈 행을 추가하여 입력	:ZZ	위와 동일
O	커서 위, 빈 행을 추가하여 입력	:wq 파일명	저장할 때, 파일이름 지정 가능
s	커서에 위치한 글자를 지우면서 입력		

커서 이동 및 붙여넣기		삭제 및 복사	
h,l	좌우 이동	x	커서에 위치한 글자를 삭제
k,j	상하 이동	X	커서 앞 글자를 삭제
w	다음 단어 첫 글자로 이동	dw	커서 뒤 단어 삭제
b	이전 단어 첫 글자로 이동	db	커서 앞 단어 삭제
G	마지막 행으로 이동	dd	커서가 있는 행 삭제
:Num	지정한 Num 행으로 이동	yw	커서 기준 뒤 단어 복사
p	커서 다음에 붙임	yb	커서 기준 앞 단어 복사
P	커서 이전에 붙임	yy	커서가 있는 행 복사

5) 그외

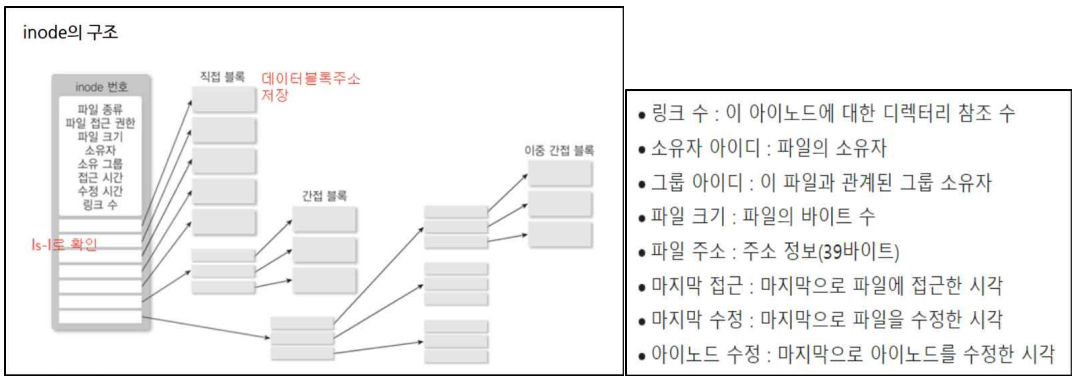
wget	파일 다운로드 관련 명령어
clear	터미널 창에 있는 모든 내용 삭제
/문자열	앞에서 시작하여 문자열 탐색
?문자열	뒤에서 시작하여 문자열 탐색
n	뒤 방향으로 탐색
N	앞 방향으로 탐색
u	Undo
:set number	행번호를 출력
:set nonuber	행번호를 숨긴다.
:%s/A/B	각 행에서 처음 등장하는 A를 B로 변경
:%s/A/B/g	모든 A를 B로 변경
:%s/A/B/gc	모든 A를 B로 변경, 변경 전에 체크 메세지 출력

4. 파일 시스템

파일 시스템은 컴퓨터에서 파일 및 자료에 대한 탐색과 접근을 용이하게 하도록 조직된 체계이자, 파일이 디스크 상에서 구성되는 방식을 말한다.

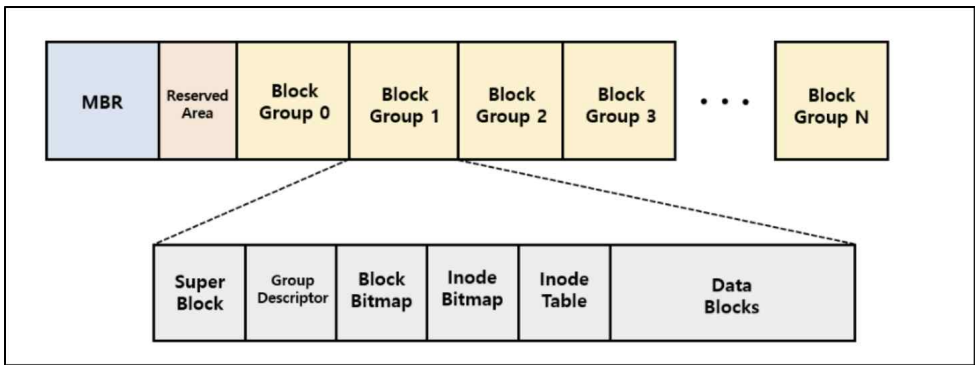
1) 파일 시스템의 구조

파일 시스템은 운영 체제마다 제각기 다른 방식을 사용하고 있으며, 대표적으로 Windows는 FAT, NTFS 등을, MacOS는 APFS, Linux는 ext를 사용한다. 아래는 ext4의 시스템 구조이다.



[inode의 구조]

파일 시스템의 구조를 알아보기 앞서, inode가 무엇인지 알아야 한다. inode는 유닉스 계열의 파일 시스템에서 사용하는 자료구조인데, 파일이나 디렉터리 등 파일 시스템에 관한 정보를 갖고 있다. 각각의 파일마다 1개의 inode를 갖고 있으며, 이를 통해 파일을 식별할 수 있다. inode는 파일 정보 저장 부분과 데이터 블록의 주소를 저장하는 부분으로 나뉘어진다. 이중에서도 전자는 터미널에 ls -l 명령어를 입력하면 출력되는 정보와 동일하다. 직접-간접-이중 간접 블록은 각각 선행한 데이터 블록에 대한 주소를 저장한다.



[ext4 기본 구조]

- MBR : OS의 위치 및 정보, 파티션의 위치 정보 등을 가지며, 부팅 과정을 진행하는 BootLoader를 저장한다.

- Block Group : Super Block부터 Data Block까지의 Block들로 구성된다. 모든 그룹은 Super Block과 Group Descriptor의 사본을 가짐으로써 파일 시스템의 데이터 정보를 보호한다.

- Super Block : 파일 시스템에 사용되는 주 설정 정보를 기록(블록 크기, 그룹 개수, inode 개수, 총 블록 개수 등). 여러 개의 블록 그룹 모두에 동일한 값으로 존재한다.

- Group descriptor : 파일 시스템 내부의 모든 블록 그룹들에 대한 정보 기록.

- Bitmap : 현재의 inode와 data에 할당된 정보의 존재 유무에 대한 정보를 담는다. 할당된 정보가 있으면 1, 없으면 0이다.

- inode table : 각 파일의 inode 정보가 표처럼 배열되어, 각 inode에는 접근, 수정, 삭제한 시각, extents 트리 데이터, 데이터 위치 등의 정보가 담겨있다.

- Data Blocks : 실제 데이터가 저장되어, Block 단위로 계산된다. inode 정보로 파일의 실제 데이터 위치를 파악한다.

2) 파일 시스템의 역할

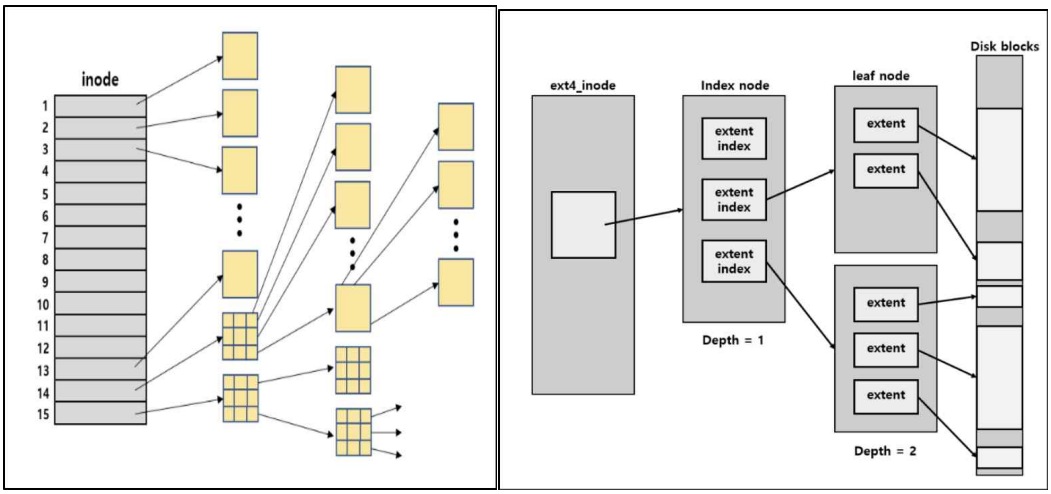
파일 시스템은 BIOS(Basic I/O System)에 의해 논리적으로 디스크 공간에 주소를 할당, 주소 정보의 저장 위치, 사용 방식을 관리한다.

파일 시스템으로 인해 디스크 저장 공간을 효율적으로 사용할 수 있으며, 데이터를 더 신속하게 처리하고 저장할 수 있는 단위 블록을 소프트웨어적으로 계산한다. 또한, 분산 저장되어 있는 연관 데이터를 빠르게 탐색한다. 이러한 세 가지의 주요 역할을 어떤 방식으로 하느냐에 따라 파일 시스템의 종류가 구분되게 된다.

3) Linux 파일 시스템의 종류

리눅스의 파일 시스템에는 확장 파일 시스템(Extended File System)인 ext ~ ext4와 ZFS, XFS 등이 존재한다. 리눅스의 모든 파일 시스템에서 파일은 inode로 관리되며, 특수 파일을 통해 장치에 접근할 수 있다. 파일 시스템 각각의 특징은 아래와 같다.

ext	Linux OS를 위한 가장 초기의 파일 시스템으로, 현재는 사용되지 않는다. Free block와 inode의 트랙을 다루기 위해 연결 리스트를 사용했기에, 사용할수록 리스트가 복잡해지는 단점이 있다.
ext2	압축 및 암호화는 지원하지 않으며, ext에 비해 파일, 파일명, 파일 시스템의 최대 크기가 증가. Block Mapping를 적용하여 파일 저장이 보다 효율적이게 되었으며, 현재도 사용되고 있기는 하다.
ext3	저널링 기능이 추가되었다. 저널의 기록을 통해 파일 시스템을 복구하는 기능을 지원하며, 보안 기능이 대폭 향상되었다. 파일, 파일 시스템 등의 크기 범위는 ext2와 큰 차이가 없다.
ext4	1EB 이상의 용량을 지원하며, 16TB까지의 파일 크기가 허용된다. 저널 checksum의 추가로 삭제 파일 복구 기능 및 파일 시스템 점검 속도가 향상되었다. 파일 시스템의 축소와 확장이 자유로우며, extents tree 구조로 파일이 관리된다. [참고 : https://ext4.wiki.kernel.org/index.php/Main_Page]
XFS	대용량 파일 시스템으로 저널링 기능도 존재하며, 읽고 쓰는 속도가 빠르고 확장성이 높다. ext3에 비해 8배 빠르게 inode 생성이 가능하며, 안정적이다.

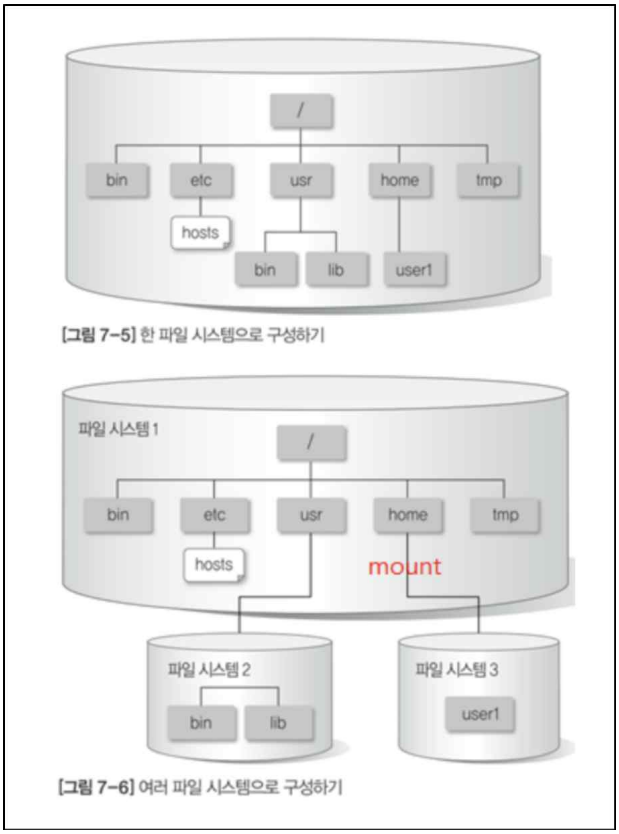


[좌측의 14, 15 inode가 Block Mapping 방식, 우측이 extents tree 구조]

+) 저널링 파일 시스템? : 메인 파일 시스템에 변경 사항을 반영하기 전에, 즉 데이터를 디스크에 기록하기 이전에 '저널'에 변경 사항을 기록 및 추적하는 시스템이다. 이를 통해 시스템 충돌이나 전원 off 등의 문제 상황에서 파일의 손상을 최소화하는 것이 가능하다.

4) 디렉터리 계층 구조

디렉터리 계층 구조는 전체 파일 및 디렉터리를 어떤 구조로 정리하고 관리할 것인지를 정의한 것이다. 파일 시스템은 이러한 계층 구조에 연결되어야 유저가 접근할 수 있다.



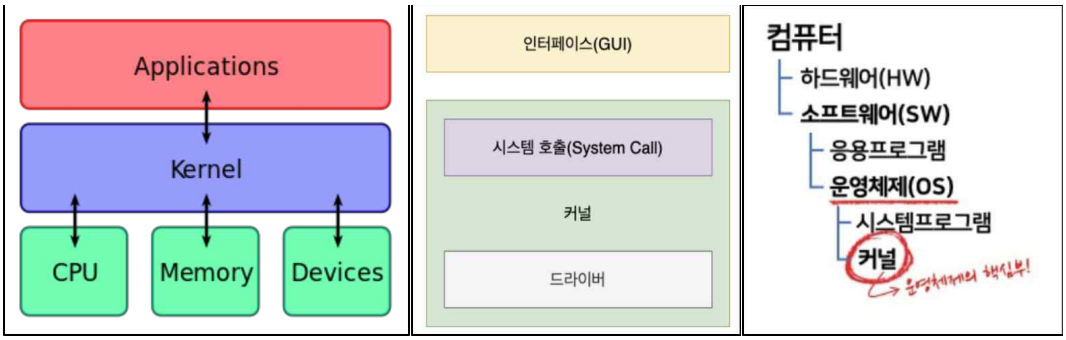
[파일 시스템에서의 디렉터리 계층 구조]

여러 파일 시스템으로 나누어 구성한다면 예를 들어, 위의 그림에서 파일 시스템2에 문제가 생기더라도 시스템1이나 3의 파일은 안전하다는 장점이 있다.

5. 커널

1) 커널이란?

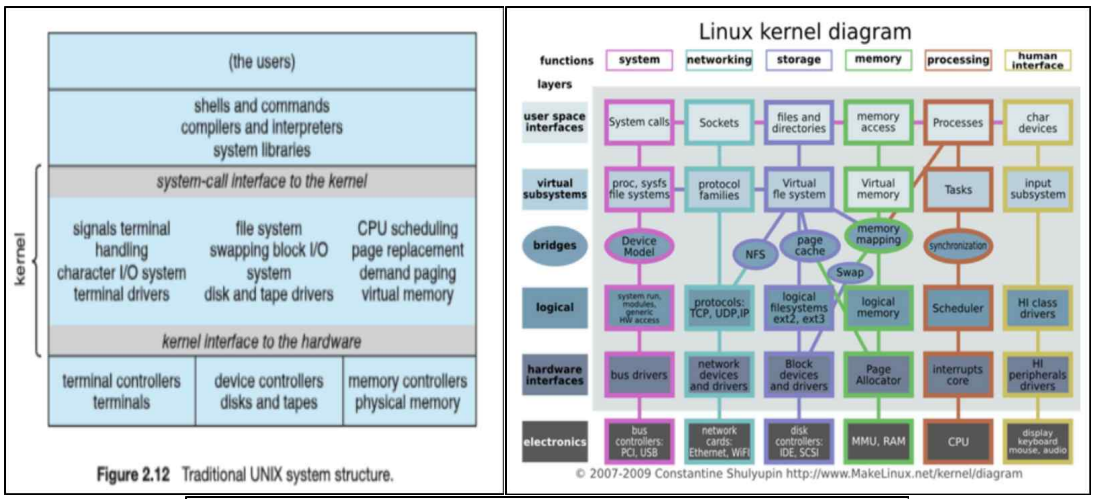
운영 체제를 구성하는 요소 중에서 가장 중요한 것이라 말할 수 있는 것이 커널과 인터페이스이다. 커널은 OS의 정체성이자 기반이라고 할 수 있으며, 프로세스나 메모리, 저장장치와 같은 컴퓨터 자원들을 관리하고 제어한다. 예를 들어, 메모리를 얼마나 할당하고, 어떤 프로세스를 실행할 것이며, 외부 디바이스와 어느 정보를 주고받고, 그 정보를 어떻게 해석할 것인지 등을 제어하는 부분이다.

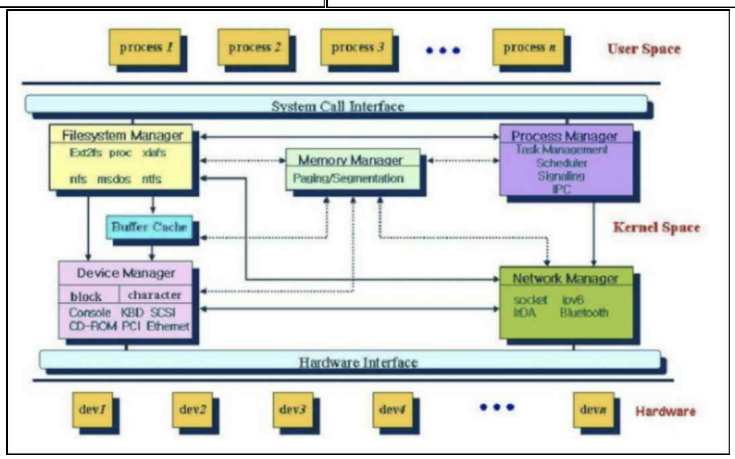


[▲커널의 상호작용, OS의 기초 구조▲ , 컴퓨터에서 커널의 위치▲]

2) 커널의 구조

커널은 크게 자원 관리를 위한 Manager가 존재하는 Kernel Space와 사용자 및 프로그램과의 안전한 상호작용을 위한 System Call, 하드웨어와의 상호작용을 위한 인터페이스로써 하드웨어를 추상화하는 Driver가 존재한다.





[모놀리식 커널(단일 구조), 리눅스 커널 구조, 커널의 기초 구조(시계방향 순서)]

+) HAL(Hardware Abstraction Layer) ?: 하드웨어 추상화 계층은 하드웨어와 소프트웨어 사이에서, 무수히 다양한 하드웨어 간의 차이를 숨겨서 음용 프로그램(소프트웨어)이 작동할 수 있게 하는 계층이다. HA는 특정 플랫폼의 구체적 부분과 하드웨어 자원의 직접적 접근을 묘사하는 소프트웨어의 집합이라고 할 수 있다. 보다 쉽게 말하자면, 하드웨어에 따라 다르게 수행해야 하는 복잡한 과정을 일관적이고 간단한 API로 응용프로그램이 호스트 시스템의 하드웨어를 발견하고 사용 가능하게 한다.

ex) Word로 작성한 글을 프린트할 때, 프린터기를 바꾸어야 한다면 HAL이 없는 경우 하드웨어에 맞추어 프로그램을 수정해야 하지만, HAL이 있기에 복잡한 과정없이 프린터기의 변경을 쉽게 할 수 있다.

3) 커널의 역할

- 사용자와 프로그램의 직접적 접근을 방지하여 하드웨어와 프로세스의 보안과 손상 방지를 수행한다.

- 앞서 언급했듯이, 시스템 자원을 관리하여 프로그램 실행을 원활하게 한다.

- 몇몇 하드웨어 추상화 기능을 구현하며, 이러한 HAL은 소프트웨어 드라이버에 의지한다.

- 자원 관리자를 거쳐, 물리적 자원(CPU, 메모리, 디스크, etc..)을 추상적 자원(Task, page/segment, File, etc..)으로 제공한다. 이를 통해 유저가 하드웨어 접근 및 사용하는 것이 가능하다.

- 구체적으로 프로세스, 메모리, 파일 시스템, 입출력(Device Driver), IPC(Inter-Process Communication)을 관리하여, 유저에게 물리적 자원에 대한 환경 및 서비스를 제공한다.

4) 커널의 종류

단일형 커널 (모놀리식 커널) 및 계층형 커널	커널의 핵심 기능 모듈이 하나로 구성되어 있기에, 빠르고 효율적이다. 그러나 수정이 어렵고 한 부분에서의 문제가 전체에 영향을 미친다. 비슷한 기능의 모듈을 묶어 계층을 만들어 여러 계층 간 통신으로 OS를 구현하는 개선된 구조가 존재한다(계층형). ex) 유닉스, 리눅스, 윈도우NT

마이크로 커널	단일형과 달리, 매우 간결한 HAL을 제공한다. 그리고 프로세스 및 메모리, IPC 관리 등 가장 기본적 기능만 제공하여 커널 크기를 축소시켰다. 단일형 커널의 단점을 해결하였으나, IPC 발생으로 속도가 느리다. ex) IOS
하이브리드 커널	마이크로 커널과 단일형 커널을 혼합한 커널. 사용자 단계에서 수행이 느린 일부 코드를 커널 단계에서 수행하도록 수정한 커널이다. ex) React OS 등 윈도우 NT 계열 일부
엑소 커널	Library OS라는 개념을 도입하여 개발자에게 하드웨어 추상화에 대한 선택 폭의 넓혀주고, 맞춤형 추상화 구현 등 프로그래머에게 추상화에 대한 유동적 방법을 제공한다. 대규모 상용 OS는 아직 없다.

6. Shell, Bash, Script 기본 개념

1) Shell

리눅스에서 셸은 OS에서 다양한 기능과 서비스를 구현하는 인터페이스를 제공하는 프로그램이다. 유저와 커널 사이의 인터페이스를 감싸는 층이기에 Shell이라는 이름이 붙게 되었다. 예를 들어, 셸은 리눅스에서 '터미널'처럼 명령어를 입력할 수 있는 환경을 의미한다.

따라서 사용자의 명령을 해석해 커널에 전달하는 명령어 해석, 셸 자체만으로 프로그램을 만들 수 있는 프로그래밍(Shell Script), 리눅스 환경을 설정할 수 있는 사용자 환경 설정 기능이 있다.

셸은 크게 그래픽 셸과 명령 줄 셸로 분류할 수 있으며, 그중에서도 명령 줄 셸에는 본 셸(bourne shell, /bin/sh)을 기반으로하는 bash(bourne again shell, /bin/bash)가 존재한다.

2) Bash

bash는 리눅스에서 가장 많이 사용하는 표준 셸로 채택되었다. 본 셸을 기반으로 GNU 프로젝트를 위해 작성되었다. 지원하는 대표적인 기능은 아래와 같다.

- Alias : 명령어 단축

- History 기능

- 연산 기능

- 자동 이름 완성 기능(tab)

- 명령 편집 기능

- 프롬프트 제어 기능(일반적인 셸 프롬프트 = "사용자 @ 호스트명 : 현재 디렉토리 전체 경로 명 $")

3) Script

기존에 존재하는 응용 소프트웨어를 제어하기 위해 사용되는 프로그래밍 언어를 말한다. 프로그래밍 언어의 소스 코드를 바로 실행하기 위한 '인터프리터'의 용도로 사용된다.(인터프리터는 원시코드를 기계어로 번역하는 컴파일러와 대비되는 개념이다) 그렇기에, 단순하고 쉬운 문법을 사용하며 실행 속도가 느리다.

일반적으로 응용 프로그램의 언어와 다른 언어로 사용되며, 응용 프로그램의 동작을 유저의 요구에 맞게 수행할 수 있도록 한다. 자바 스크립트, 유닉스 셸 스크립트(sh, bash, etc..) 등이 스크립트 언어에 속한다. 아래는 스크립트 언어를 사용하면 안되는 경우이다.

1. 리소스에 민감한 작업들, 특히 속도가 중요한 요소일 때(정렬, 해쉬 등)
2. 강력한 산술 연산 작업들, 특히 임의의 정밀도 연산
3. 플랫폼 간 이식성이 필요할 때
4. 구조적 프로그래밍이 필요한 복잡한 애플리케이션(변수의 타입 체크나 함수 프로토 타입 등이 필요할 때)
5. 업무에 아주 중요하거나 회사의 미래가 걸렸다는 확신이 드는 애플리케이션
6. 보안 상 중요해서 시스템 무결성을 보장하기 위해 외부의 침입이나 크래킹, 파괴 등을 막아야 할 필요가 있을 때
7. 서로 의존적인 관계에 있는 여러 컴포넌트로 이루어진 프로젝트
8. 과도한 파일 연산이 필요할 때
9. 다차원 배열이 필요할 때
10. 링크드 리스트나 트리 같은 데이터 구조가 필요할 때
11. 그래픽이나 GUI를 만들고 변경하는 등의 작업이 필요할 때
12. 시스템 하드웨어에 직접 접근해야 할 때
13. 포트나 소켓 I/O가 필요할 때
14. 예전에 쓰던 코드를 사용하는 라이브러리나 인터페이스를 써야 할 필요가 있을 때
15. 독점적이고 소스공개를 안 하는 애플리케이션을 만들어야 할 때(오픈소스 스크립트 언어의 경우)

7. 단축키

Shift + Ctrl + C	복사
Shift + Ctrl + V	붙여넣기
Ctrl + L	터미널 화면 Clear
Ctrl + Alt + T	터미널 실행
Shift + Ctrl + N	터미널 내에서 새 창으로 터미널 실행
Shift + Ctrl + Q	현재 터미널 종료
Shift + Ctrl + F	터미널 내에서 문자열 검색
%	일시정지 상태의 프로그램 재실행
Ctrl + C	실행 중 프로그램 중단
Ctrl + Z	실행 중 프로그램 일시정지
Tab	자동 완성
Ctrl + M	Enter
Ctrl + P	이전 명령어

Ctrl + N	다음 명령어
Ctrl + B	커서를 좌측으로 이동
Ctrl + F	커서를 우측으로 이동
Alt + Fnum	콘솔 이동

8. 추천 프로그램

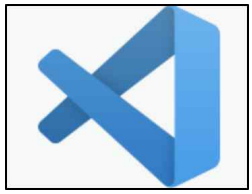
1) Wine(필수)

Wine은 유닉스 계열의 운영 체제(리눅스 등)에서 Windows에서 제작된 프로그램을 실행시킬 수 있게 해주는 프로그램이다. 에뮬레이터라고 오인할 수 있으나, 개발자와 공식 사이트에 따르면 Windows에서 제작된 바이너리 프로그램을 해독하는 '호환성 계층'임을 명시하고 있다. 자유 소프트웨어로, 오픈 소스이므로 접근 및 수정이 용이하다.

[Wine Software Logo]

+) 호환성 계층? : Host 시스템이 아닌, 다른 외부 시스템의 이진 프로그램 및 파일을 Host 시스템에서 실행시킬 수 있도록 하는 구성 요소를 말한다. 예를 들어, Wine은 프로그램 로더와 윈도우 API를 통해 Windows 이진 파일을 유닉스 계열(ubuntu 등) 시스템에서 실행할 수 있도록 한다.

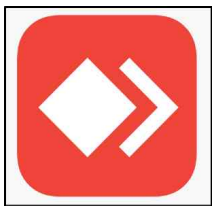
2) VSCode(필수)



[VSCode logo]

Visual Studio Code는 마이크로소프트가 제공하는 개발 환경이자, 소스 코드 편집기이다. Windows, macOS, Linux에서 사용 가능하다. C, C++, Java, Python 등 다양한 언어를 지원하며, 디버깅, 구문 강조, Git 제어 등의 기능을 포함한다.

3) Anydesk



[Anydesk's app icon]

원격 데스크톱, 원격 지원 및 관리 용도의 소프트웨어 프로그램으로, 파일 전송, VPN, 가상 회의, 데스크톱 공유 등의 기능을 수행할 수 있다.

4) WPS

[WPS's app icon]

WPS Office는 MS Office 문서 형식 및 HTML, PDF 등의 문서를 읽고, 편집 및 저장할 수 있는 응용 프로그램이다. Windows, macOS, Linux, iOS, Android 등의 플랫폼에서 사용 가능한 오피스 제품군이다.

▶ git

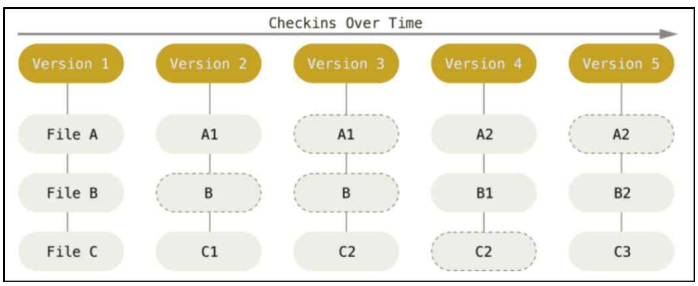
1. git의 정의 및 용도

1) git이란?

git은 리눅스 커널을 개발한 '리누스 토르발스'가 개발한 소프트웨어로, 스냅샷 스트림 기반의 분산 버전 관리 시스템으로, 컴퓨터 파일의 변경 사항을 추적하고 다수의 사용자들이 서로 파일에 대한 작업 조율이 필요할 때 사용된다.

[git software system logo]

+) 스냅샷 스트림 : '스냅샷'은 파일 시스템에서 과거에 어느 시점에 존재했고, 유지시킨 파일 및 디렉토리의 집합이자 작업 공간의 상태이다. git은 저장소의 파일 시스템 전체를 스냅샷으로 다뤄, 커밋 시점의 저장소 상태가 하나의 버전이 된다. 다만, 마지막 커밋의 스냅샷만 전체를 저장하고, 그 이전의 시점들에 대해서는 각각의 스냅샷들의 차이만 기록한 '델타'를 저장한다. 따라서 버전 관리 시스템 중 크기가 작고 속도가 빠른 편이다.



[git 스냅샷 스트림]

2) git의 용도 및 필요성

적게는 10명 이하에서, 많게는 수십 수백명까지의 개발자가 하나의 소프트웨어 개발 프로젝트에 공동으로 참여할 경우, 각 파트별로 나뉘어 있는 파일을 업데이트하게 된다. 이 상황에서 개인마다 파일의 버전을 업데이트하며, 변경하는 파일명의 기준이 다르기 때문에 최종 프로젝트 버전에 합치는 과정은 매우 복잡해지게 된다. 특히나 파일명의 언어가 다를 경우에 더 혼잡한 상황에 놓이게 된다. 따라서 이런 프로젝트에서 버전 관리를 돕기 위해 필요한 것이 git과 같은 버전 관리 시스템이라고 할 수 있다.

3) git의 특징

- commit : 현재 변경된 작업 상태에 대해 점검을 마치고, 확정되면 저장소에 저장하는 작업을 말한다.

- branch : 각각의 개발자가 bug fix, update, 새 아이디어 적용, 이전 버전으로 복귀 등 다양한 시도 및 과정을 거치는 단계이다. 이런 각각의 분기이자 가지(branch)에서 각자 동시에 작업할 수 있다. 특히 git은 거의 모든 작업이 Local이므로 오프라인에서도 작동한다.(참고로 현재 작업 중인 branch를 head라고 한다)

- merge : 각각의 branch에서 개발된 내용을 본 프로그램에 합치는 것을 merge 방식이라 한다. 이를 병렬 개발이라고 하며, 전체 저장소를 복사해서 각자 local 저장소로 가져와 개발을 진행하고 다시 합치는 것을 반복하므로, 수많은 백업이 분산되어 저장되는 것과 동일하다. 이는, 메인 프로그램 및 저장소의 복구가 용이하다는 장점을 갖게 된다.

- Staging Area : 커밋을 준비하는 곳으로 커밋이 발생하기 이전에 검증하는 장소이다. 커밋은 Staging Area에서 레포지토리로 파일을 전송하는 것을 뜻한다. 또한, 변화가 있는 파일을 구분할 수 있기 때문에 커밋의 속도가 빠르다.

- 대부분의 동작이 로컬에서 진행되므로 서버와의 통신 비용이 적고, git은 C언어로 작성되어 있기 때문에 속도가 빠르다.

- 오픈 소스로 누구든지 사용이 가능하다.

2. 버전 관리 시스템

1) 버전 관리 시스템이란?

버전 관리 시스템, VCS는 파일의 변화를 시간 흐름에 따라 기록해두었다가, 필요한 상황에 해당 버전의 파일을 불러올 수 있는 시스템을 의미한다. 주로 소프트웨어 및 프로그램에서 소스 코드의 변경 사항에 따라 각각의 버전을 저장하여, 복구를 쉽게 하거나 대규모 공동 작업을 안전하게 진행하기 위해 사용된다.

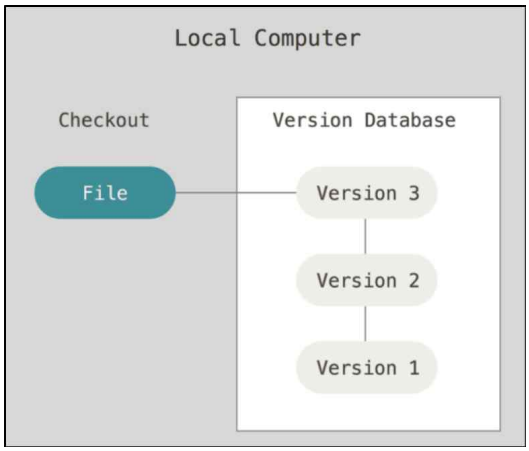
2) 버전 관리 기본 개념

Repository	저장소	Merge	합치기, 접붙이기
Trunk/Main	주류, 본류	Conflict	충돌 사항
Revision	개정판	Check out	저장소에서 파일 불러오기

Head	최신	Commit	버전 갱신(check in)
History	수정 기록	Import	로컬 디렉토리의 파일을 처음으로 저장소에 복사
Update/Sync	동기화		
Branch	가지 생성	push	변경사항을 서버로 전달
Diff	차이 보기	pull	최신 버전을 컴퓨터로 가져옴

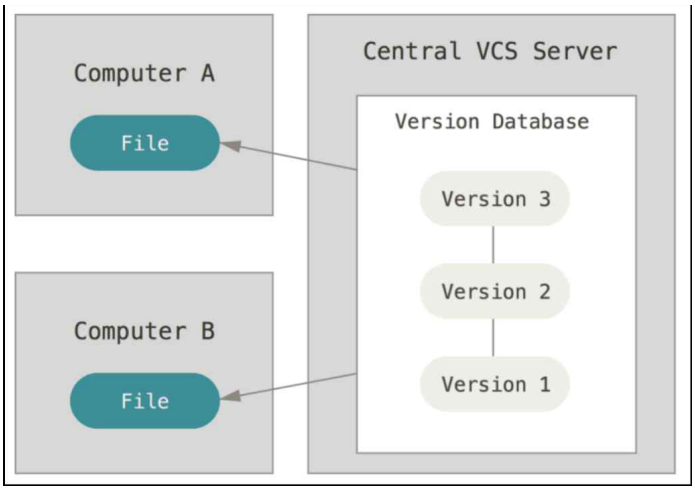
3) 버전 관리 시스템의 종류

- Local VCS : RCS(Revision Control System)을 활용하는데, RCS는 기본적으로 Patch Set, 즉 파일에서 변경되는 부분을 관리하여 모든 파일을 특정 시점으로 되돌릴 수 있다. DB를 사용하여 Patch Set을 관리하기 때문에 매우 간단하지만 실수하기 쉬운 것이 단점이고, 팀 프로젝트에서 사용하기 힘들다.



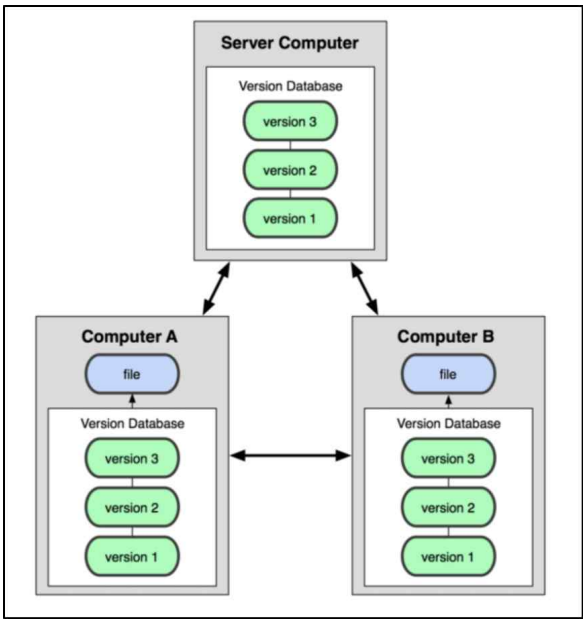
[Local VCS]

- Centralized VCS : 중앙 집중 방식의 버전 관리 시스템으로, 팀 프로젝트를 위해 사용한다. 파일 관리를 수행하는 서버가 별도로 존재하며, 클라이언트가 중앙 서버로 부터 파일을 받아 사용한다. 누가 어떤 작업을 하는지 쉽게 확인 가능하지만, 프로젝트 규모가 커지면 변경 사항 충돌 문제 등이 발생할 수 있고, 중앙 서버가 다운될 경우, 그동안 작업이 불가능하다. 또한, 하드 디스크에 문제가 발생하면 모든 히스토리가 사라질 수도 있다.



[CVCS]

- Distributed VCS : 분산 버전 관리 시스템으로 git이 이에 해당한다. 분산에 집중하여 개발자들이 각자 독립적으로 작업하기 때문에 충돌 문제가 거의 발생하지 않고, 클라이언트가 메인 저장소의 파일을 복제할 때, 단순히 마지막의 스냅샷만 복제하는 것이 아니라, 히스토리와 함께 전부 복제하기 때문에 중앙 서버에 문제가 생기더라도 작업을 재개할 수 있으며, 서버 복원도 클라이언트 각각이 복제한 것 중 아무거나 골라서 복원하면 되므로 매우 쉽다. 따라서 DVCS는 Merge 시에만 조금 신경써서 업로드하면 된다.



[DVCS]

3. 주요 명령어 정리

분류	명령어	기능
저장소 관련	git init	저장소를 초기화. 이 명령을 입력하고 난 뒤에만, 추가적인 깃 명령어를 입력할 수 있다.
	git status	깃 저장소의 상태를 확인한다.
	git clone <저장소 URL>	깃 저장소에 있는 소스 코드를 복제한다.
	git remote add < 원격 저장소> <저장소 URL>	새로운 원격 저장소를 추가한다.
	git clone /로컬/저장소/경로	로컬 저장소를 복제한다.
	git clone 사용자명@호스트:/원격/저장소/경로	원격 서버의 저장소를 복제한다.
tag 작업	git log	전체 로그를 확인한다.(현 위치의 브랜치 커밋 내용을 확인하고 식별자를 부여한다)
추가	git add [업로드 할 파일 or 디렉토리 경로]	해당 파일 혹은 디렉토리의 변경 사항을 staging area에 등록한다.
	git add -A	작업 디렉토리의 모든 변경 내용을 staging area에 등록한다.
	git add -p	터미널에서 staging area로 넘길 파일을 선택할 수 있다.
	git add *	인덱스에 추가된 상태
commit	git commit -m “커밋 메세지”	staging area에 있는 변경 내용을 묶고 정의한다. 다시 말해, 실제 변경 사항을 확정한다.
branch	git branch	브랜치 목록을 보여준다.
	git branch <브랜치명>	새 브랜치를 로컬로 생성한다.
	git checkout -b <브랜치명>	브랜치를 생성하고, 생성한 브랜치로 이동한다.
	git checkout master	master 브랜치로 복귀한다.
	git branch -d < 브랜치명>	해당 브랜치를 제거한다.
	git push origin < 브랜치명>	생성한 브랜치를 원격 서버로 전송한다.
	git push -u < remote > <브랜치명>	새 브랜치를 원격 저장소로 push
	git pull < remote	원격에 저장되어 있는 git 프로젝트의 현재 상태를 다운로드

	> <브랜치명>	하고, 현재 위치한 브랜치로 병합시킨다.
	git branch -a	모든 브랜치를 확인한다.
	git remote add origin [깃허브 주소]	깃허브 주소를 연결한다.(깃에 새 원격 서버 주소를 알리는 것)
config	git config --list	전체 config 리스트 확인
	설정1	git config --global user.name “이름”
	설정2	git config --global user.email “이메일”
	삭제1	git config --unset --global user.name
	삭제2	git config --unset --global user.email
push	git push origin master	변경 사항을 원격 서버에 업로드한다.
	git push < remote > <브랜치명>	커밋을 원격 서버에 업로드 한다. 동일 : git push -u < remote > <브랜치명>
	git remote remove <등록된 주소>	클라우드 주소를 삭제한다.
merge	git pull	원격 저장소의 변경 내용을 현재 디렉토리에 가져오고, 병합한다.
	git merge <다른 브랜치명>	현재 브랜치에 다른 브랜치의 수정 사항을 병합한다.
	git add <파일명>	각 파일을 병합한다.
	git diff <브랜치명> <다른 브랜치명>	변경 사항 merge 하기 이전에 바뀐 내용을 비교할 수 있다.
return	git checkout --< 파일명>	로컬 변경 사항을 이전으로 되돌린다.
	git fetch origin	원격에 저장되어 있는 깃 프로젝트의 현 상태를 다운로드한다.

4. 관련 플랫폼 및 서비스

1) Github

깃허브는 깃을 지원하는 웹 호스팅 서비스 시스템이다. 달리 말해, 깃이 저장소로 사용하는 클라우드라고 보면 되며, 로컬에 있는 데이터를 깃허브에 업로드할 수도 있고, 다운로드 받을 수도 있다.

[github logo]

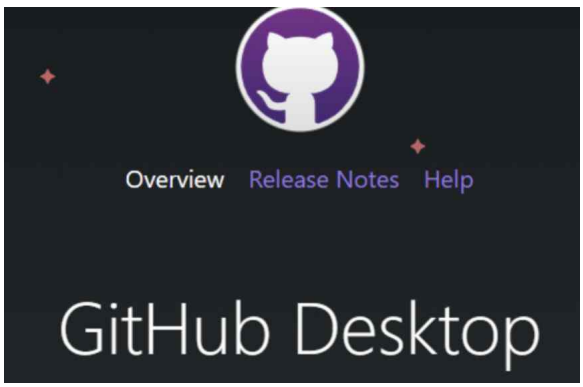
2) GitLab

깃랩은 깃 저장소 및 보안성 테스트 등의 기능을 갖춘 소프트웨어 플랫폼으로 깃허브와 마찬가지로 협업형 버전 관리 기능을 수행할 수 있다.

[gitlab logo]

3) Github Desktop

깃허브 데스크탑은 깃허브에서 소스 코드의 add, commit, push 등의 작업을 편하게 하기 위한 기능을 제공하는 통합 tool이다.



[github desktop]