

오픈 소스 소프트웨어 강의 목표

- 오픈 소스의 세계로 입문하자
- Linux 운영체제의 세계로 입문하자
- Git/ GitHub를 활용하여 오픈 소스의 바다로 나가자

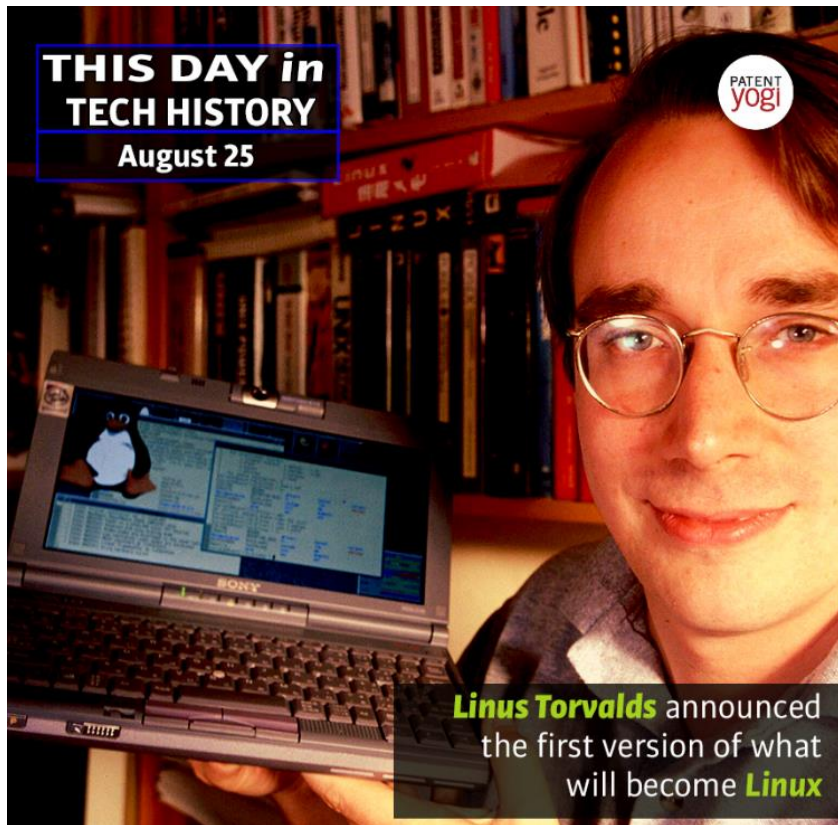


리눅스(Linux) 운영체제

동의과학대학교 컴퓨터정보과

리눅스(Linux)

- 1991년 핀란드 헬싱키대학의 Linus Torvalds(1966년생)가 뉴스그룹에 글을 올리며 개발 시작
 - <https://patentyogi.com/this-day-in-tech-history/day-tech-history-linux-announced-linus-torvalds-august-25-1991/>



리눅스(Linux)

- 리눅스 개발 시작
 - 네덜란드 암스테르담 자유대학교 전산학 교수인 앤디 탄넨바움 교수가 학생들의 학습을 목적으로 개발한 MINIX로 Unix와 호환되는 공개 운영체제의 개발 계획을 MINIX 뉴스그룹에 발표하면서 시작됨
 - 자신의 이름과 기반 시스템인 Unix를 따서 Linux로 명명
- 자유 소프트웨어와 오픈소스 개발의 가장 유명한 표본
- 리눅스로 작동되는 서버
 - 퍼블릭 클라우드의 90%
 - 스마트폰의 82%
 - 임베디드 기기의 62%
 - 슈퍼 컴퓨터의 99%

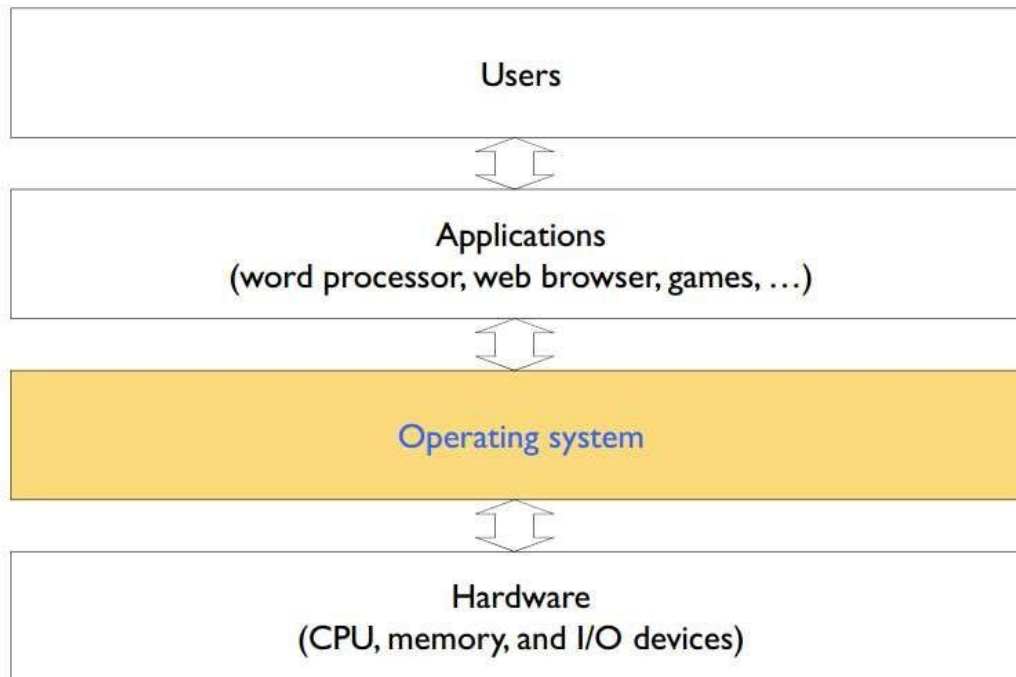
Linus Torvalds

- Linux와 Git(버전관리시스템)을 최초로 개발한 소프트웨어 개발자
- 2016년 TED와 한 인터뷰 영상 (21분)
 - <https://www.youtube.com/watch?v=o8NPllzkFhE&t=18s>

참고 : 운영체제(Operating System)

- 하드웨어를 제어하고 응용 소프트웨어를 위한 기반 환경을 제공
- 하드웨어 바로 위에 설치되는 소프트웨어 계층으로서 모든 컴퓨터 시스템의 필수적인 부분

▶ Four components of a computer system



유닉스(UNIX)

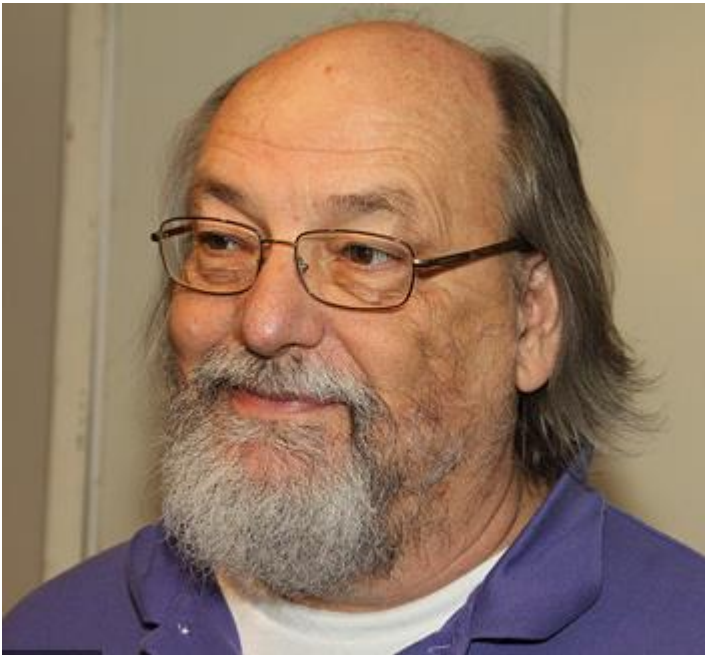
- 등장 배경

- 1964~1969년 AT&T 벨 연구소, MIT, General Electric 에서 **Multics**라는 실험적 운영체제를 공동 개발하는 프로젝트 진행 -> 프로젝트 좌초
- Multics 개발에 참여했던 AT&T 벨 연구소의 Kenneth Thompson, Dennis Ritchie 와 동료들은 1969~1970년 사이에 PDP-7를 이용하여 작은 운영 체제 (하드웨어에 독립적인)를 개발하기 시작
- MULTICS라 불리는 초기의 운영체제 프로젝트를 빗대어 유닉스(UNIX)로 명명



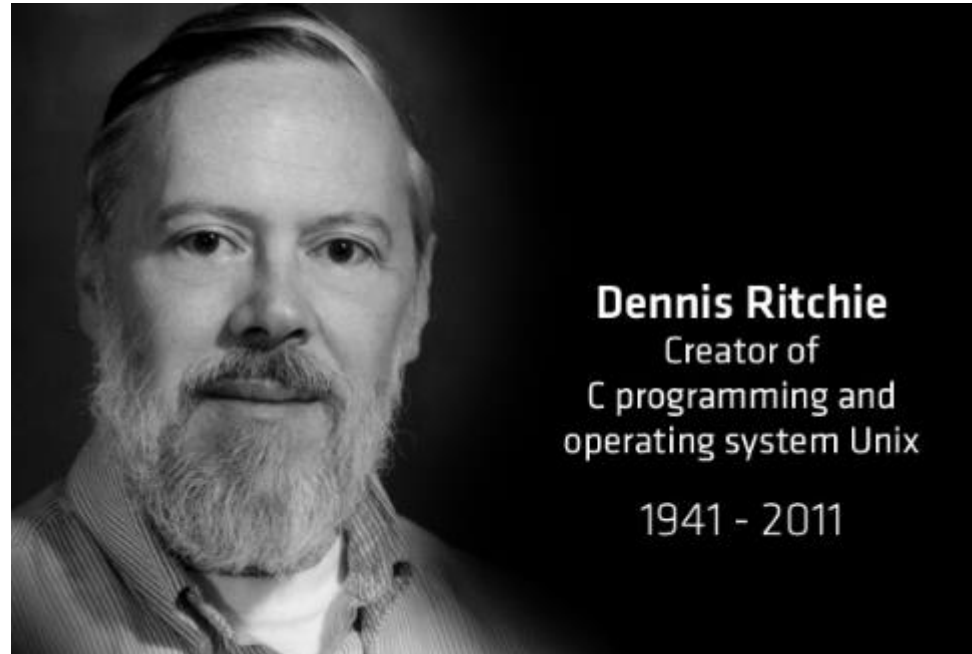
유닉스(UNIX)

- 초기 UNIX는 기계 의존적이며, 기종 간 호환성이 없는 운영체제
- UNIX를 구현하기 위해 Dennis Ritchie가 C언어 개발(이전은 어셈블리 언어)
- UNIX는 **이식성**과 **호환성**있는 운영체제로 사용자들로부터 큰 반향을 일으킴



Kenneth Thompson

C언어와 Unix의 창시자들



Dennis Ritchie

<https://www.youtube.com/watch?v=g3jOJfrOknA>

<https://www.youtube.com/watch?v=JoVQTPbD6UY>

유닉스(UNIX)

- Unix의 유료화
 - AT&T의 거대화로 1984년 7개의 회사로 분리
 - Unix 소스를 유료화하여 판매
 - 다양한 Unix 변형이 일어나 표준화 함(POSIX)
 - FSF(Free Software Foundtion) 설립 : 유닉스를 무료로 배포할 수 있도록 전체 소스코드를 다시 작성하기로 하는 운동 시작(GNU)
- 다양한 유닉스 버전 존재
 - BSD(Berkeley Software Distribution) -> FreeBSD(오픈 소스)
 - System III -> System IV -> System V(정식 표준에 채택, BSD의 혁신을 받아들임)
 - 이 후 다양한 유닉스 버전들이 존재

GNU 프로젝트

- 1984년 MIT의 스톨먼(Richard Stollman)은 소스를 공개하지 못하도록 하는 분위기와 기술을 상업화하려는 조류에 반감 가짐
- C로 작성된, 모두에게 공개된 UNIX 시스템을 위해 GNU(GNU is Not Unix) 프로젝트를 시작
- 1985년 GNU프로젝트 운영을 위해 FSF(Free Software Foundation, 자유 소프트웨어재단)을 설립
- 1990년 GNU 프로젝트는 거의 완성단계에 이르렀으나, 운영체제에서 핵심이 되는 커널이 빠져있는 상태
- 1991년 Linus Torvalds 가 커널 공개
- **GNU 시스템의 커널로 Linux 채택**되어 개발됨



참고 : 오픈 소스 라이선스

- 오픈 소스에 대한 정의
 - <https://opensource.org/osd>
- 주요 오픈 소스 라이선스
 - **GPL**(GNU General Public License)
 - GPL이 적용된 SW로 개량된 SW의 소스코드도 공개해야 함(전체 공개SW의 70~80%)
 - 자유 SW재단의 리처드스톨만이 만들었음
 - **MIT license**
 - MIT의 소프트웨어 공학도들을 돕기 위해 개발한 라이선스
 - 개조한 제품을 반드시 오픈소스로 배포해야 한다는 규정이 없음
 - **Apache License 2.0**
 - 오픈소스를 수정하여 재배포하더라도 소스코드를 공개할 의무가 없음
 - 재배포 경우 아파치 라이선스 2.0 복사본 제공, 아파치에 의해 개발되었음을 표기해야 함
 - Eclipse Public License
 - Mozilla Public License 등

리눅스(Linux)

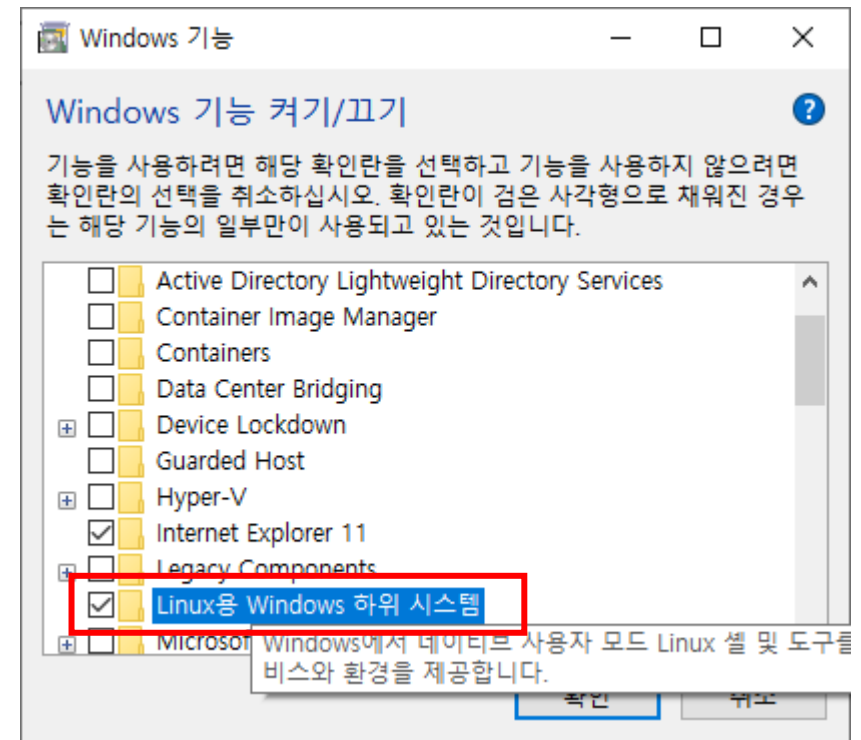
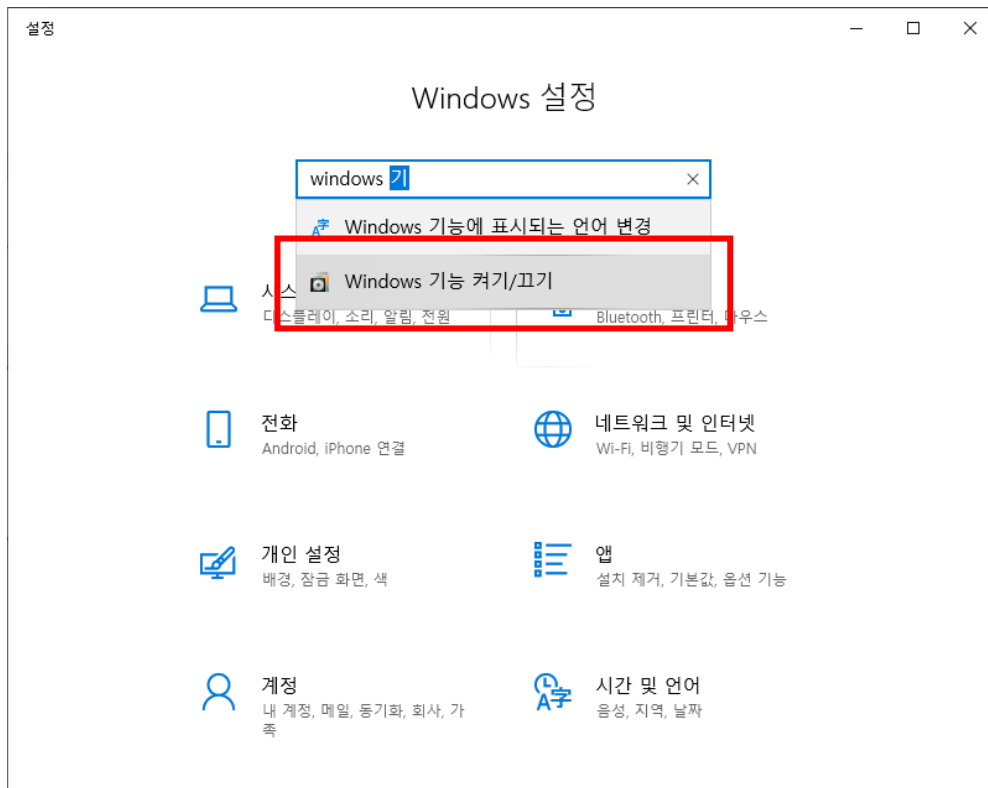
- 다중 사용자(multi-user)와 멀티 태스킹(multi-tasking) 지원 운영체제
- 리눅스 배포판
 - **Debian** 계열 : Debian, **Ubuntu**, KNOPPIX
 - **Red Hat** 계열(상용) : Fedora, RedHat Enterprise, CentOS, Vine Linux
 - **Slackware** 계열 : openSUSE, SUSE Linux Enterprise

multi-user : 하나의 컴퓨터에 여러 사용자가 로그인하여 사용 가능

multi-tasking : 한번에 여러 프로세스를 실행시킬 수 있음

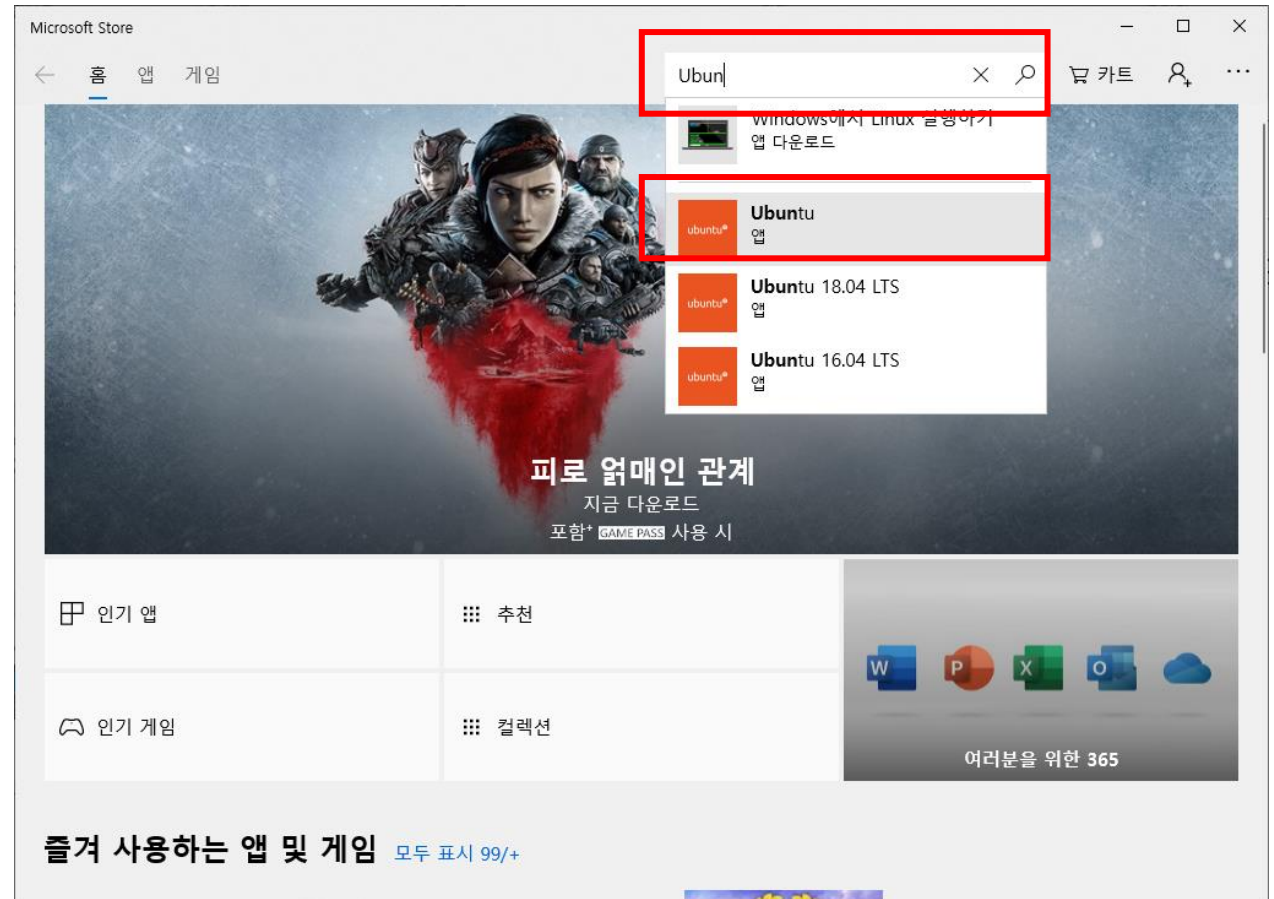
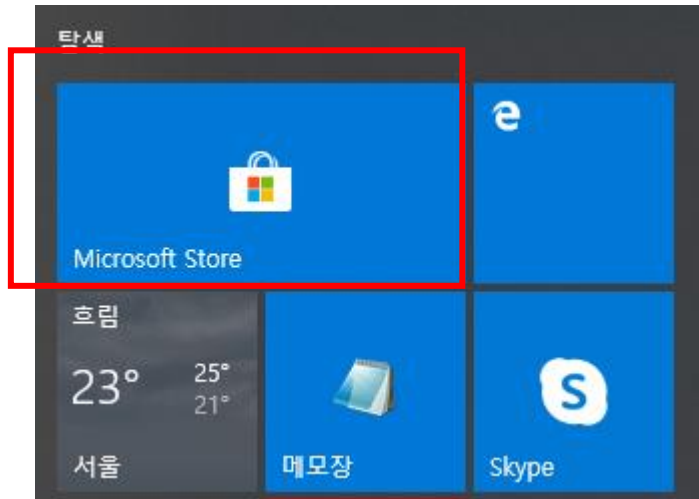
Linux 설치하기

- Windows10에서 우분투 리눅스 앱 설치
 - Step 1 : windows 기능 중 Linux용 windows 하위 시스템 설정 후 재부팅



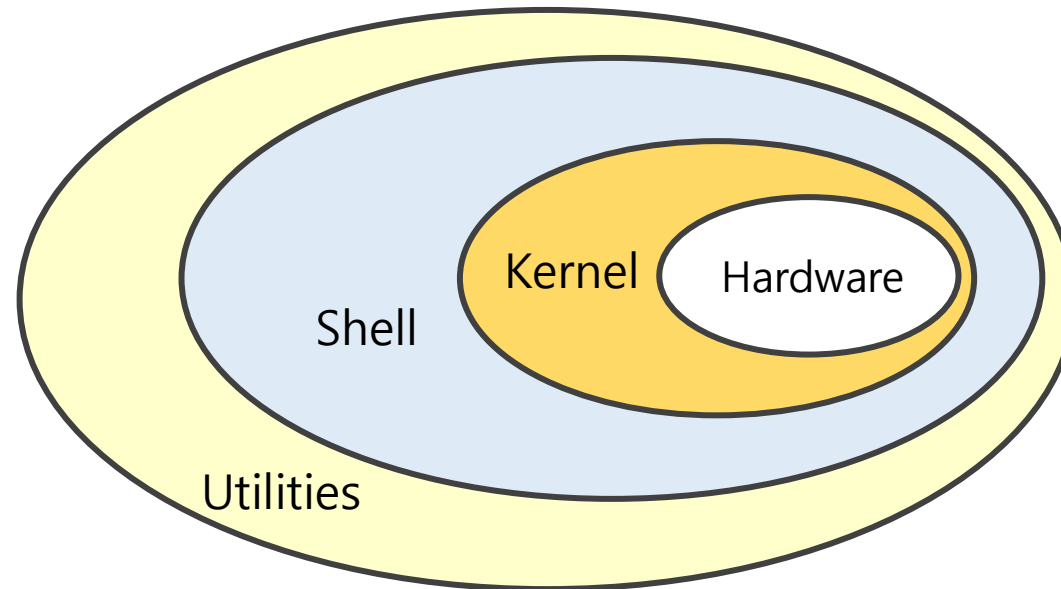
Linux 설치하기

- Step 2 : Microsoft Store에서 Ubuntu 를 검색하여 설치



리눅스 운영체제의 구조

- **Kernel** : 컴퓨터 자원관리(메모리, 파일시스템, 장치 등)
- **Shell**
 - 운영체제 커널과 사용자 사이를 이어주는 역할
 - 사용자의 명령을 해석하고, 커널에 명령을 요청해주는 역할
- **Utilities** : 각종 프로그래밍 개발도구, 문서 편집 도구 등

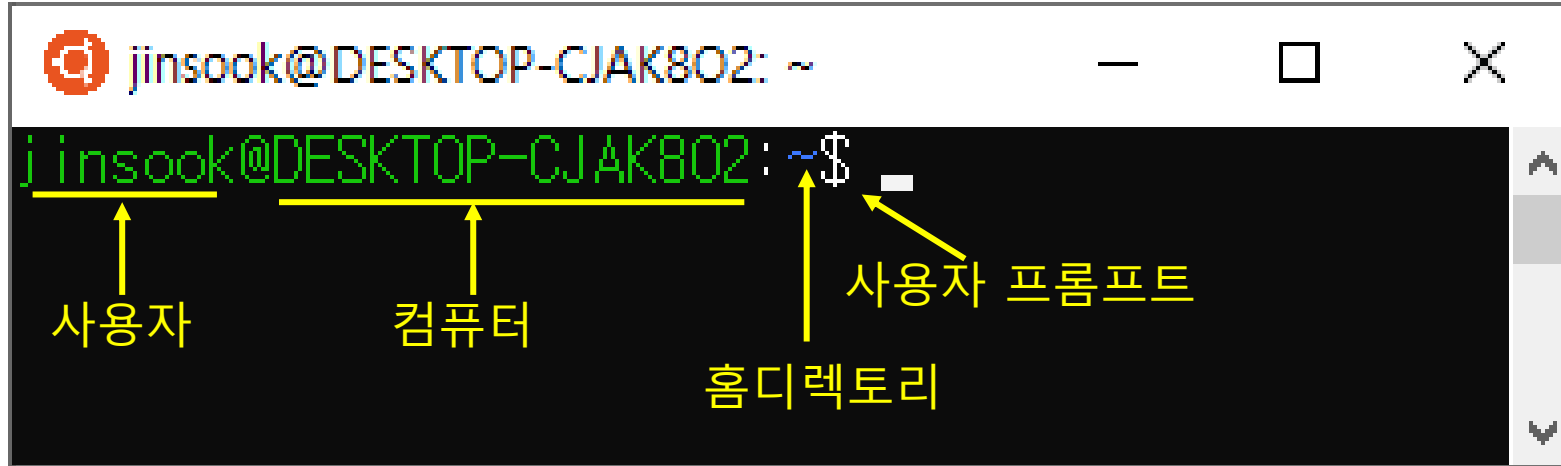


Shell 이란

- Shell 이란?
 - 운영체제는 shell을 통해 사용자와 통신하여 프로그램 실행
 - 윈도우의 탐색기 또는 cmd(라인 명령어) 와 같은 역할
- Shell의 종류
 - **bash** shell
 - Bourne Again Shell(Stephen Bourne이 개발)
 - 최초의 shell인 bourne shell 과 호환되도록 만들어진 shell
 - **csh** : BSD 계열 유닉스 사용자들이 선호
 - **ksh** : Unix System V 계열 유닉스 사용자들이 선호
 - **tcsh** : C shell 과 호환
 - **zsh** : Z shell, 확장형 Bourne shell, 최근 MacOS의 기본으로 채택

Shell 사용하기

- 사용자 계정



A terminal window titled 'jinsook@DESKTOP-CJAK802: ~' with standard window controls. The prompt 'jinsook@DESKTOP-CJAK802: ~\$' is displayed. Yellow arrows point from Korean labels to parts of the prompt: '사용자' (user) points to 'jinsook', '컴퓨터' (computer) points to 'DESKTOP-CJAK802', '홈디렉토리' (home directory) points to '~', and '사용자 프롬프트' (user prompt) points to '\$'.

```
jinsook@DESKTOP-CJAK802: ~$
```

다음 명령문은 무엇을 하는 것일까요?

\$ cal

\$ date

\$ hostname

\$ uname -a

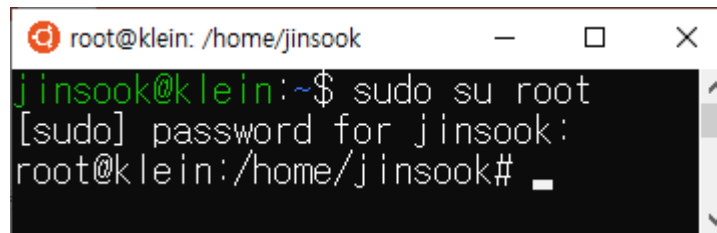
리눅스 명령어

명령어	설명	예시
su <계정>	switch user : 현재 계정을 로그아웃하지 않고 다른 계정으로 전환	\$su user01
whoami	현재 사용자 확인	\$whoami
passwd	현재 사용자 암호 변경	\$passwd
exit	현재 계정 로그아웃	\$exit
sudo <명령어>	superuser do : 현재 계정에서 root 권한을 이용하여 명령어를 실행할 때 사용	\$sudo su root
man <명령어>	명령어 사용법 보기(페이지 별로 볼 수 있음)	\$man su
<명령어> --help	명령어 사용법 보기	\$ls --help

리눅스 명령어

- sudo(Superuser Do)
- 명령어 앞에 sudo를 붙여서 root 권한으로 명령어 실행
 - sudo <명령어>
 - sudo passwd
 - sudo su <계정명>
 - sudo apt-get install <패키지>
- su와 달리 sudo를 사용하는 당사자의 비밀번호를 사용

root 사용자일 경우는
프롬프트가 \$가 아니라
#로 표기된다.



```
root@klein: /home/jinsook
jinsook@klein:~$ sudo su root
[sudo] password for jinsook:
root@klein:/home/jinsook#
```

su, sudo 명령어의 차이점은?

- su (switch user) 명령어
 - 현재 계정을 로그아웃 하지 않고, 다른 계정으로 전환하는 명령어
 - su
 - root 사용자로 변경
 - root 암호를 물어본다
 - su 는 su root와 동일
 - su user01
 - user01 사용자로 변경
 - user01 사용자의 환경변수까지 적용(shell과 홈 디렉토리 변경)
 - % exit(혹은 logout)으로 이전 계정으로 돌아옴
- sudo (superuser do) 명령어
 - 현재 계정에서 root 권한을 이용하여 명령어를 실행할 때 사용
 - \$ apt-get update

* 새로운 사용자 추가 명령어
% **sudo adduser newsuer**

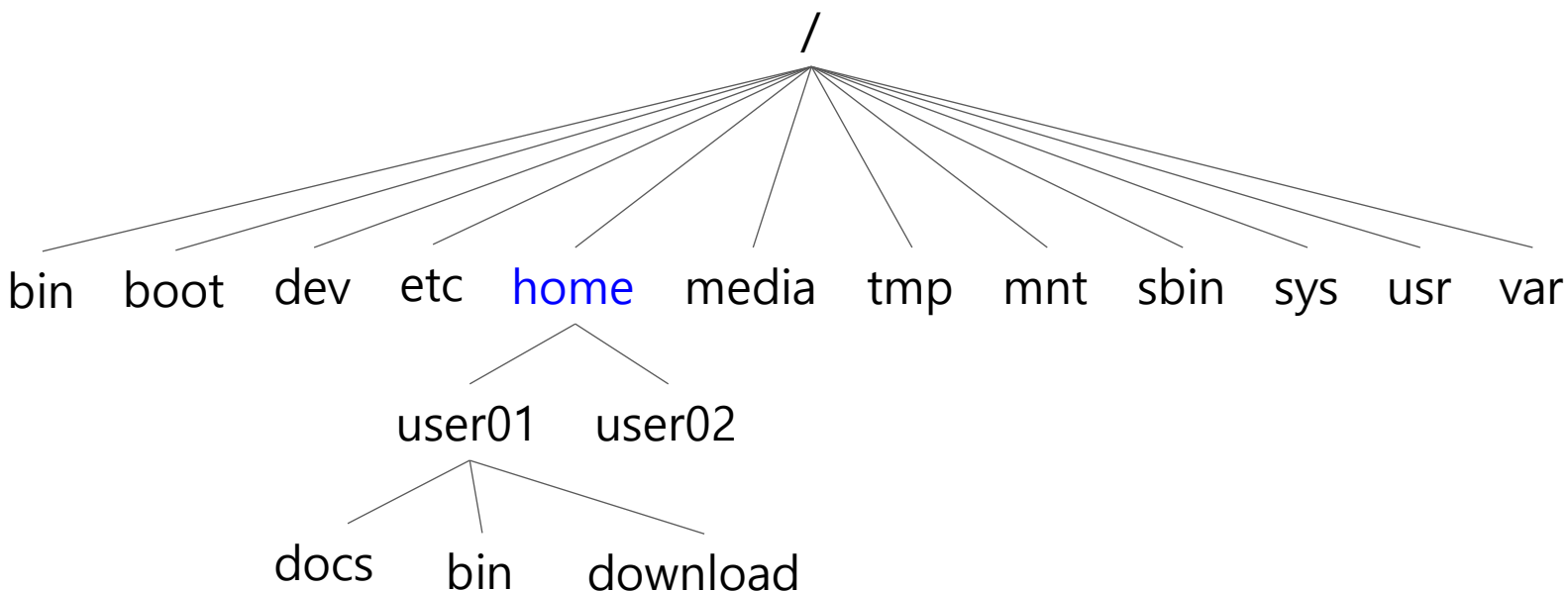
실습

1. root 계정으로 로그인하기
2. 로그인된 계정명 확인하기
3. root 계정에서 로그아웃하기
4. 로그인된 계정명 확인하기
5. 암호 변경하기
6. su 명령어의 사용법(매뉴얼) 확인하기

리눅스 파일 시스템

- 디렉토리 구조

- root(/) 아래에 디렉토리를 갖는 계층구조
- 각 디렉토리들은 각자의 역할을 가짐



directory

```
user01@klein: /
user01@klein:~$ cd /
user01@klein:/$ ls -l
total 112
drwxr-xr-x 1 root root 4096 Sep 14 20:31 bin
drwxr-xr-x 1 root root 4096 May 21 23:42 boot
drwxr-xr-x 1 root root 4096 Sep 15 19:54 dev
drwxr-xr-x 1 root root 4096 Sep 15 19:39 etc
drwxr-xr-x 1 root root 4096 Sep 15 19:39 home
-rwxr-xr-x 1 root root 112600 Jan 1 1970 init
drwxr-xr-x 1 root root 4096 May 21 23:41 lib
drwxr-xr-x 1 root root 4096 May 21 23:39 lib64
drwxr-xr-x 1 root root 4096 May 21 23:39 media
drwxr-xr-x 1 root root 4096 Aug 31 23:19 mnt
drwxr-xr-x 1 root root 4096 May 21 23:39 opt
dr-xr-xr-x 11 root root 0 Sep 15 19:54 proc
drwx----- 1 root root 4096 Sep 3 21:50 root
drwxr-xr-x 1 root root 4096 Sep 15 19:54 run
drwxr-xr-x 1 root root 4096 Sep 14 20:33 sbin
drwxr-xr-x 1 root root 4096 Mar 21 18:55 snap
drwxr-xr-x 1 root root 4096 May 21 23:39 srv
dr-xr-xr-x 12 root root 0 Sep 15 19:54 sys
drwxrwxrwt 1 root root 4096 Sep 15 15:49 tmp
drwxr-xr-x 1 root root 4096 May 21 23:39 usr
drwxr-xr-x 1 root root 4096 May 21 23:42 var
user01@klein:/$
```

각 디렉토리의 구조

- / – Root
 - 모든 파일과 디렉토리의 시작 위치
 - 오직 루트 유저만이 권한을 갖는 디렉토리
 - /가 아닌 /root가 루트 유저의 홈 디렉토리
- /bin – User Binaries
 - 실행 가능한 바이너리 포함
 - 사용자의 **명령어**와 시스템 명령어가 이곳에 위치
 - 예: ps, ls, ping, grep, cp.
- /sbin – System Binaries
 - 실행 가능한 바이너리 포함
 - **시스템 유지보수**를 위한 시스템 관리자 명령어가 이곳에 위치
 - 예: iptables, reboot, fdisk, ifconfig

각 디렉토리의 구조

- /etc – Configuration Files
 - 모든 프로그램의 **설정** 파일 포함
 - 각 프로그램의 시작 및 종료 스크립트 포함
 - 예: /etc/resolv.conf, /etc/logrotate.conf
- /dev – Device Files
 - **장치**(device) 파일 포함
 - 터미널 장치, usb, 시스템에 연결된 모든 장치 포함
 - 예: /dev/tty1, /dev/usbmon0
- /proc – Process Information
 - **시스템 프로세스** 정보 포함
 - 실행 중인 프로세스 정보를 포함하는 pseudo filesystem
 - 예: /proc/{pid} 디렉토리는 해당 pid 프로세스의 정보 포함
 - system resources 정보 포함하는 virtual filesystem

각 디렉토리의 구조

- /var – Variable Files
 - 내용이 증가될 수 있는 파일 포함
 - system log files (/var/log)
 - packages and database files (/var/lib)
 - emails (/var/mail)
 - print queues (/var/spool)
 - lock files (/var/lock)
 - temp files needed across reboots (/var/tmp)
- /tmp – Temporary Files
 - 임시 파일 포함
 - 재부팅 시 삭제됨

각 디렉토리의 구조

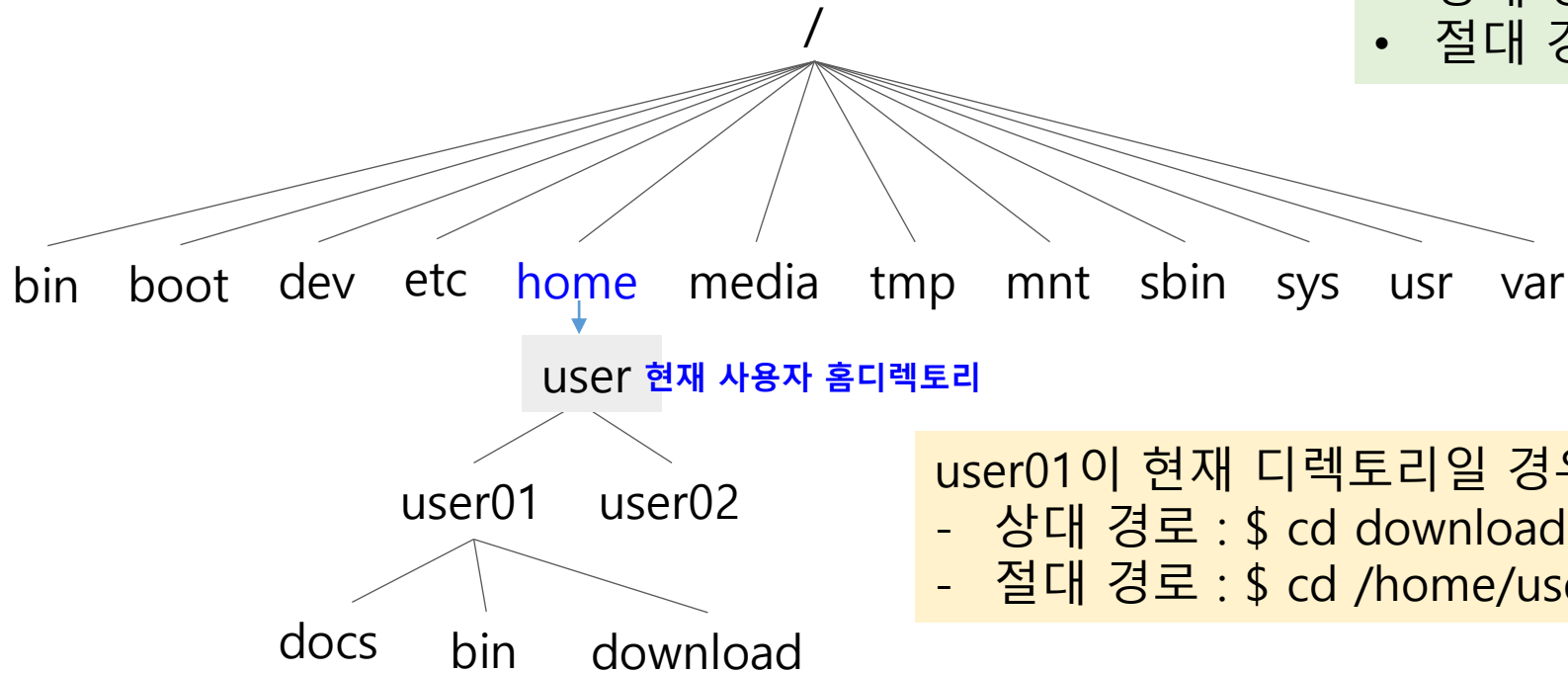
- /home – Home Directories
 - 모든 **사용자**의 개인 파일들이 저장되는 디렉토리
 - 예: /home/john, /home/nikita
- /boot – Boot Loader Files
 - 부트 로더 관련 파일 포함
 - Kernel initrd, vmlinux, grub 파일 포함
 - 예: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic
- /lib – System Libraries
 - /bin and /sbin의 바이너리를 지원하는 **라이브러리** 포함
 - 파일명 : ld* 혹은 lib*.so.*
 - 예: ld-2.11.1.so, libncurses.so.5.7

각 디렉토리의 구조

- /mnt – Mount Directory
 - 마운트된 파일시스템이 연결된 디렉토리
 - 로컬 컴퓨터의 파일 접근 가능
- /media – Removable Media Devices
 - 제거 가능한 장치를 위한 임시 디렉토리
 - 예: /media/cdrom for CD-ROM, /media/floppy for floppy drives; /media/cdrecorder for CD writer

디렉토리 전환

- 부모 디렉토리 : `..`(dot 두 개)
- 현재 디렉토리 : `.` (dot 한 개)
- 홈디렉토리 : `~` (tilde :)
- 상대 경로 : 현재 위치를 기준한 경로
- 절대 경로 : root 부터의 경로



user01이 현재 디렉토리일 경우 download 디렉토리 이동하기

- 상대 경로 : `$ cd download` 또는 `$ cd ./download`
- 절대 경로 : `$ cd /home/user/user01/download`

docs가 현재 디렉토리일 경우 download 디렉토리 이동하기

- 상대 경로 : ?
- 절대 경로 : ?

리눅스 명령어

명령어	설명	예시
cd	change directory : 디렉토리 변경	\$ cd / \$ cd bin \$ cd lib \$ cd ~ \$ cd .. \$ cd ../bin
ls	list : 현재 디렉토리의 파일 목록 보기	\$ ls \$ ls -l \$ ls -a \$ ls -la \$ ls -F
pwd	print working directory : 현재 디렉토리 경로 출력	\$ pwd

실습

- 자신의 홈 디렉토리로 전환하는 방법은?
- ls의 사용법 살펴보기(man ls)
- 현재 디렉토리 확인하기

리눅스 명령어

명령어	설명	예시
mkdir	디렉토리 생성	\$ mkdir document \$ mkdir myspace
rmdir	디렉토리 삭제	\$ rmdir document
touch	빈 파일 생성	\$ touch test.txt
mv	파일 이동 / 파일이름 변경(rename)	\$ mv test.txt mytext.txt \$ mv mytext.txt myspace/
cp	파일 복사	\$ cp mytext.txt mytext1.txt \$ cp mytext.txt mytext2.txt
rm	파일 삭제	\$ rm mytext1.txt \$ rm -i mytext1.txt \$ rm -r gitTest1(폴더)

실습

1. 자신의 홈 디렉토리에서 하위 디렉토리(documents, download) 만들기
2. documents 디렉토리에 빈 파일 first.txt 를 만들기
3. first.txt 를 복사하여 second.txt 를 만들기
4. second.txt 파일을 download 디렉토리에 third.txt로 옮겨 저장하기
5. third.txt 파일 삭제하기
6. documents 디렉토리 삭제하기
7. download 디렉토리 삭제하기(발생되는 문제?)

리눅스 명령어

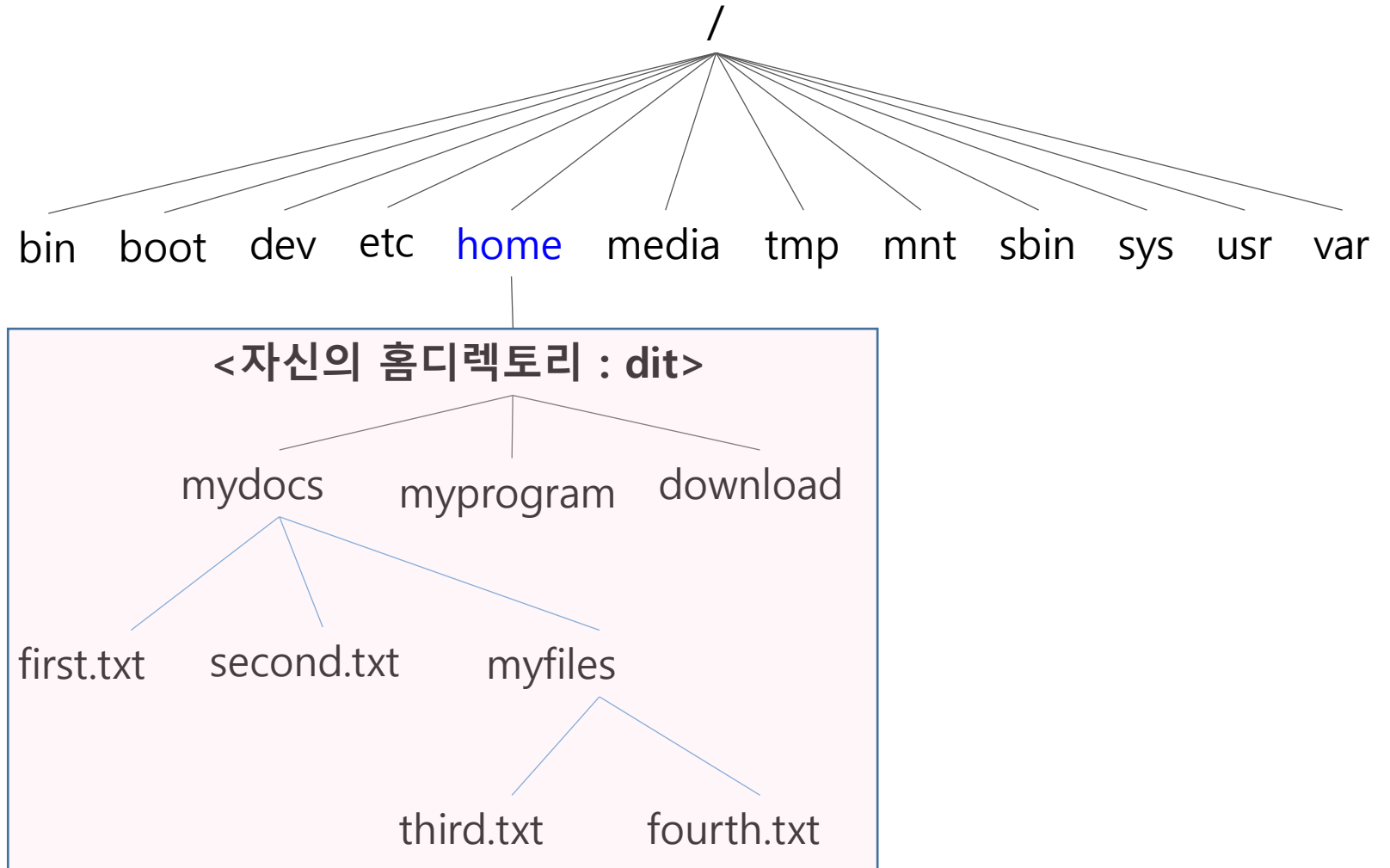
- alias : 별칭 만들기
 - 긴 명령어의 묶음을 별칭을 지정해서 간단하게 사용할 수 있는 기능을 제공
- 등록된 별칭 목록 보기 : `$ alias`
- 별칭 만들기 : `$ alias la='ls -la'`
- 별칭 없애기 : `$ unalias la`

요약

- Unix
- Linux
- Linux 파일 시스템
- Linux 명령어
 - su, whoami, exit, passwd, man, sudo
 - cd, pwd, ls
 - mkdir, rmdir, rm, mv, cp, touch
 - alias, unalias

실습

- 자신의 홈디렉토리에 있는 파일들을 모두 지우시오.
- 다음과 같이 디렉토리와 파일을 작성하십시오.
- 현재 위치에서 다른 디렉토리로 이동하는 것을 연습하십시오.



확인 학습

- <https://gomguard.tistory.com/73>
- <https://itholic.github.io/linux-basic-command/>